

Laurentian IBM Business Process Manager Advanced 8.0.x Application Development Workshop



Dave Spriet (spriet@ca.ibm.com)
Senior BPM Consultant, Expert Access
Services, ISSW

Agenda

- **Introduction to IBM Business Process Manager Advanced V8**
- **IBM Business Process Manager Standard V8 Capability**
 - IBM Process Center Overview
 - IBM Business Process Portal Overview
 - IBM Process Designer Overview
 - Process Designer
 - Process Inspector
 - Business Process Definition Model
 - Pools and Swimlanes
 - Flow objects
 - Start Events
 - Intermediate and boundary events
 - Stop Events
 - Activity Task

Agenda

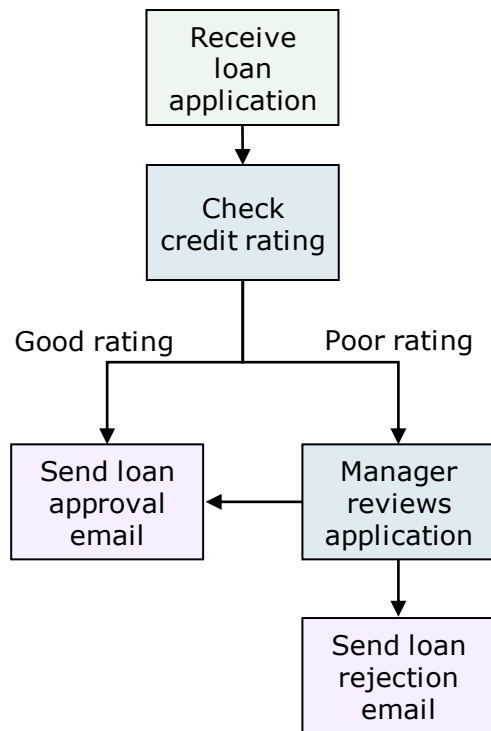
- **IBM Business Process Manager Advanced V8 Capability**
 - IBM Integration Designer Overview
 - Module Packaging and Deployment
 - Service Component Architecture
 - Web Services
 - Interfaces, Business Objects and Data Objects
 - Mediation Modules and Primitives
 - Business Process Choreography (BPEL)
 - Business Rules
 - Human Tasks
 - Testing

IBM Business Process Manager Advanced V8 Application Development

- Introduction to IBM Business Process Manager Advanced V8

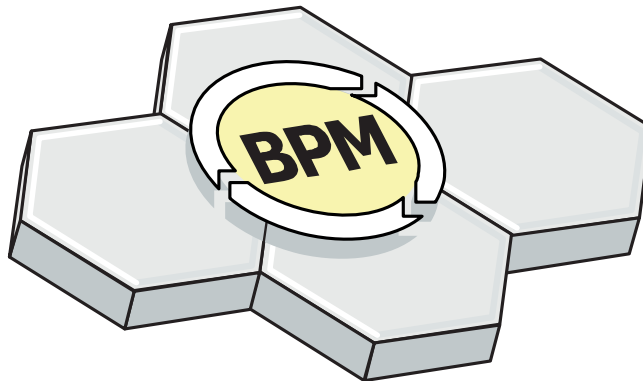
Introduction to IBM Business Process Manager

- A *business process* is a collection of service interactions and activities that are run to fulfill a business need
- A business process defines the potential execution order of services:
 - Defines how to coordinate interactions between a process instance and its partners
 - Specifies how to handle errors (faults)
 - Specifies other required technology patterns like compensation



IBM Business Process Management

- *Business process management (BPM)* is a systematic approach to improving business processes for an organization
 - BPM makes business processes more effective and efficient through a cycle of continuous improvement
- BPM often includes the steps:
 - Model, test, deploy, run, and monitor



IBM Business Process Manager

- IBM Business Process Manager gives you visibility into your business processes
 - Enables the development and management of business processes
 - Can be configured to support various levels of complexity and integration between IBM BPM components
 - An integrated runtime for all business processes, services, and enterprise applications
 - Tools for developers, administrators, and users
- Components of IBM Business Process Manager
 - IBM Process Server: The runtime operating system
 - IBM Process Center: A unified BPM asset repository
 - IBM Integration Designer (available in Advanced edition only): An authoring environment for developing services and self-contained enterprise applications
 - IBM Process Designer: An authoring environment for developing process models

IBM Business Process Manager editions

Advanced

Transformation

Complete set of advanced BPM capabilities

- Extended support for high-volume process automation, with high quality-of-service
- Built-in SOA components for extensive enterprise-wide service integration, orchestration

Standard

Program

Configured for typical BPM projects

- For multi-project improvement programs, with high business involvement
- Basic system integration support
- Rapid time-to-value: improved user productivity

Express

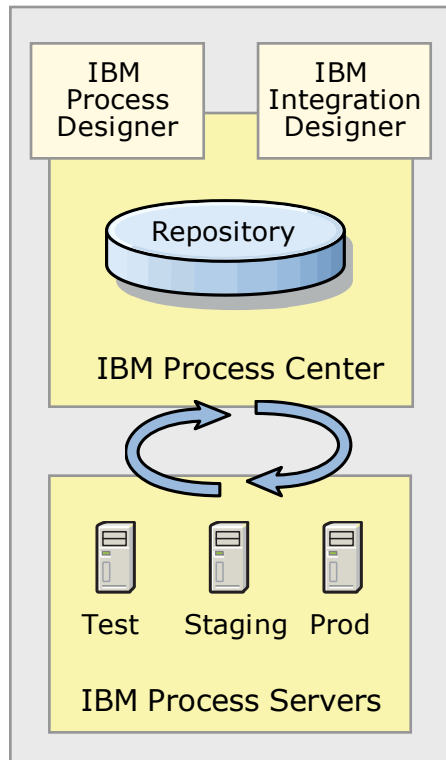
Project

Configured for first BPM project

- For small number of users – single server, no clustering
- Low entry price
- Install in a few clicks

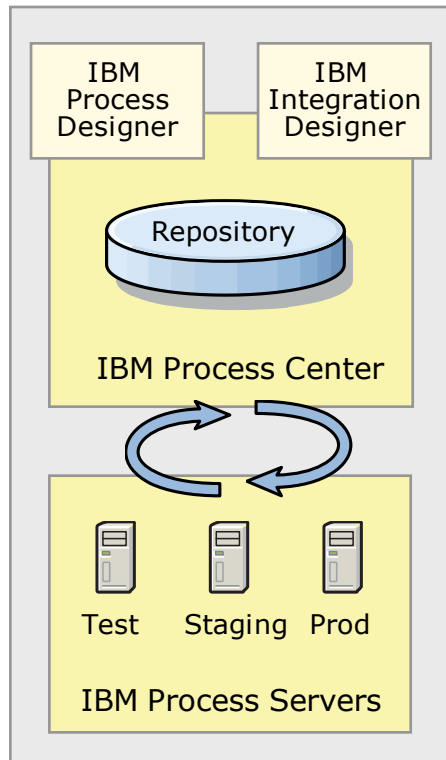
IBM Business Process Manager V8 (1 of 3)

- Tools for modeling, designing, implementing, and deploying business processes
- Includes:
 - **IBM Process Designer:** An authoring environment that is used for creating process models that contain automated and human tasks
 - **IBM Integration Designer:** An authoring environment that is used for creating process models and advanced implementations, including mediations, business rules, and human tasks
 - **IBM Process Center (Design+Governance+Runtime):** Includes a repository for all processes, services, and other assets that are created in the authoring environments
 - **IBM Process Server (Runtime only):** Provides a single runtime environment for supporting process models, service orchestration, and integration capabilities

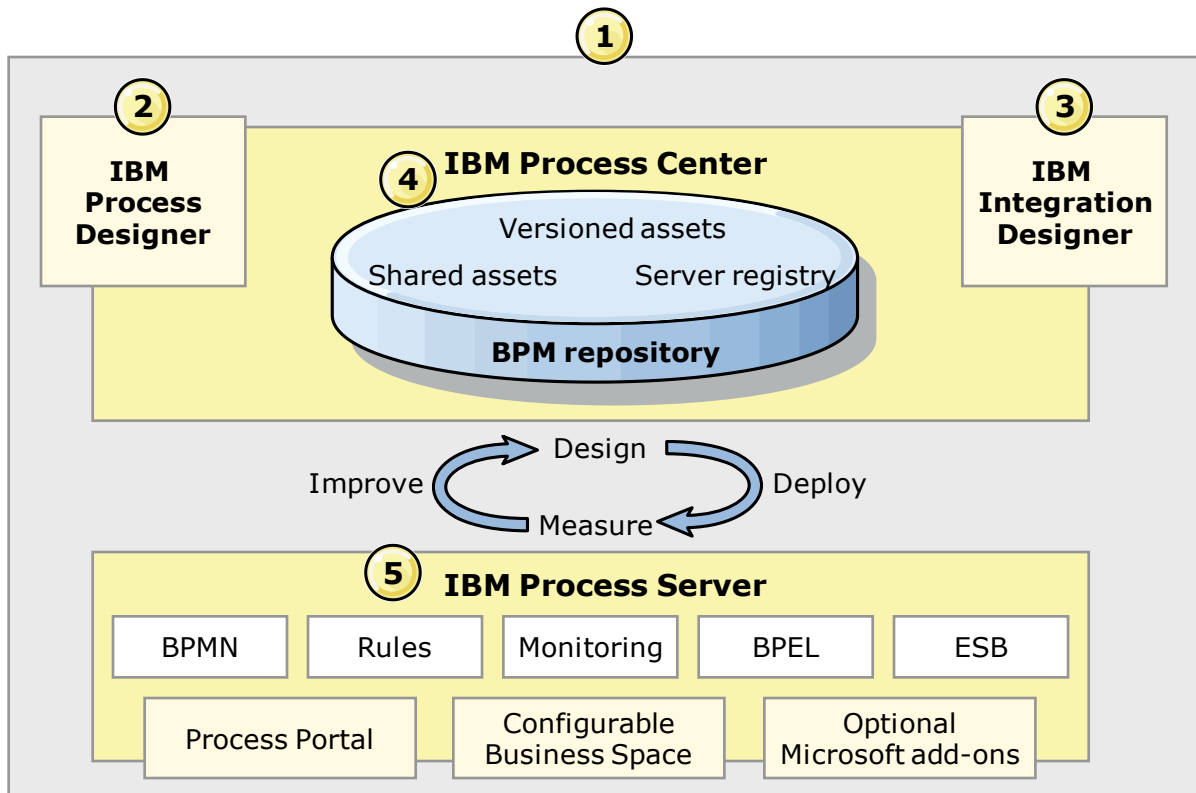


IBM Business Process Manager V8 (2 of 3)

- **Concept & Vocabulary:**
 - **Process Application:** A process application is the minimum BPM deployment unit, it is developed using the Process Designer, it mainly contains:
 - Business Process Definitions (BPDs)
 - Services
 - Data Model
 - Process Application configuratoin settings (exposure, variables etc ...)
 - Etc ...
 - **Toolkit:** A toolkit contains same items as a process app. It is used as a library of reusable assets for different process apps
 - **Process Application Snapshot:** A snapshot is a version of a Process Application, it is a deployment unit. Snapshots can be deployed to runtime environments (process servers).
 - **Business Process Definition (BPD) :** A BPD is a BPMN (Business Process Model Notation) diagram that represents the business process resulting from the business analysis.
 - **Business Process Instance :** An Business Process Instance (also called "process instance" or just "instance") is a running execution of a BPD. Each instance has its own history and data and it is related to only one BPD from a specific snapshot of a Process Application.



IBM Business Process Manager V8 (3 of 3)



IBM products for the SOA development lifecycle

Stage 1: Blueworks Live, IBM Process Designer

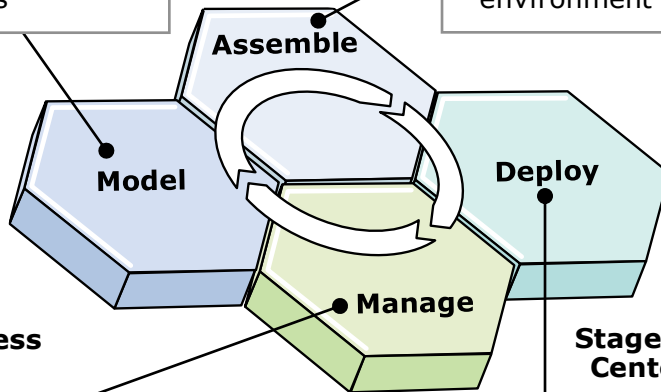
Design, model, simulate, and optimize business processes

- Gather requirements

Stage 2: IBM Process Designer and Integration Designer

Discover, assemble, and test

- Standards-based development environment



Stage 4: IBM Business Monitor

Business monitoring

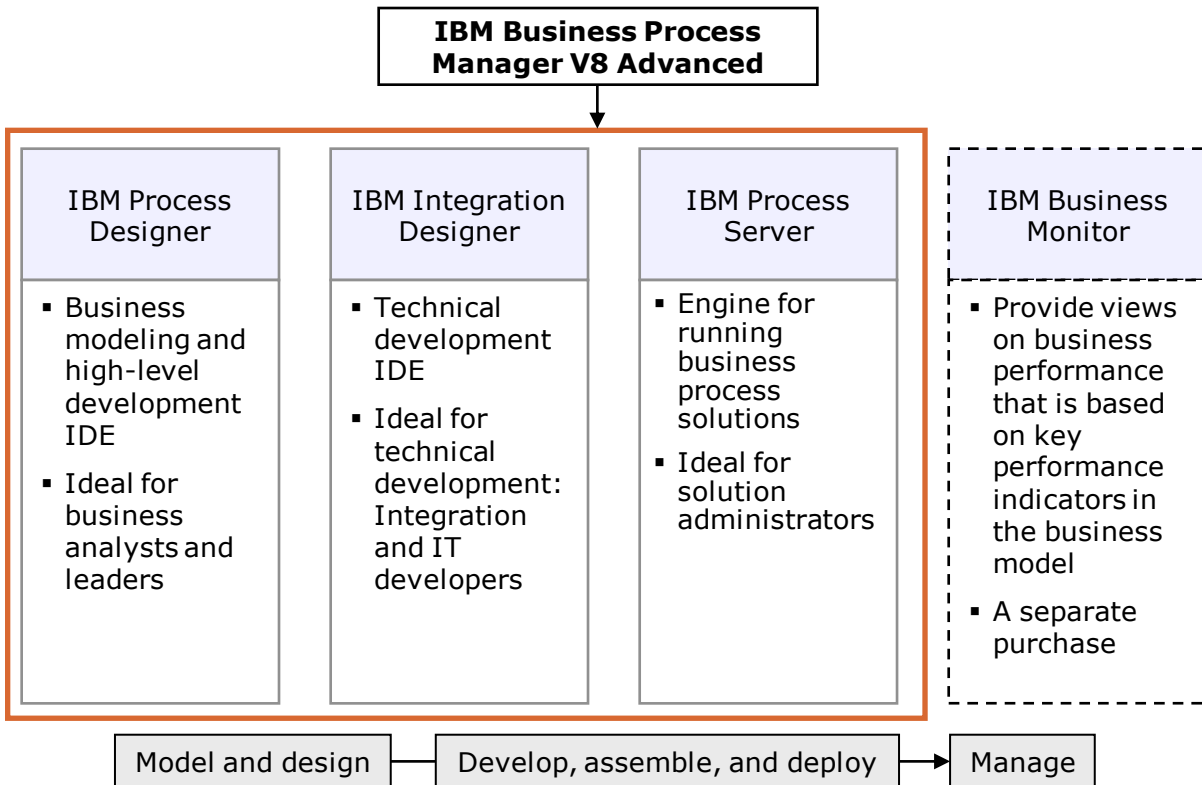
- Analytics and optimization of processes
- Real-time visibility into business processes, enabling process intervention, and continuous improvement

Stage 3: IBM Process Center and Process Server

Deploy applications

- Support business processes and human-intensive process steps with system automation

Business integration roles in IBM Business Process Manager



Standard versus Advanced edition key capabilities (1 of 2)

Capability	Advanced edition	Standard edition
WebSphere Lombardi Edition compatible execution	X	X
Process Designer - BPMN	X	X
Coach user interface	X	X
Process Portal	X	X
Reporting	X	X
Performance Data Warehouse	X	X
Process Center shared asset repository	X	X
Unlimited authors and users	X	X

Standard versus Advanced edition key capabilities (2 of 2)

Capability	Advanced edition	Standard edition
WebSphere Process Server compatible execution	X	
Integration Designer – BPEL / SOA	X	
Transaction	X	
Adapters	X	
Business Space	X	
ESB	X	
Cluster	X	X
Advanced operating system support - Linux on System z, IBM AIX, Solaris	X	X

Comparison: IBM Integration Designer and IBM Process Designer

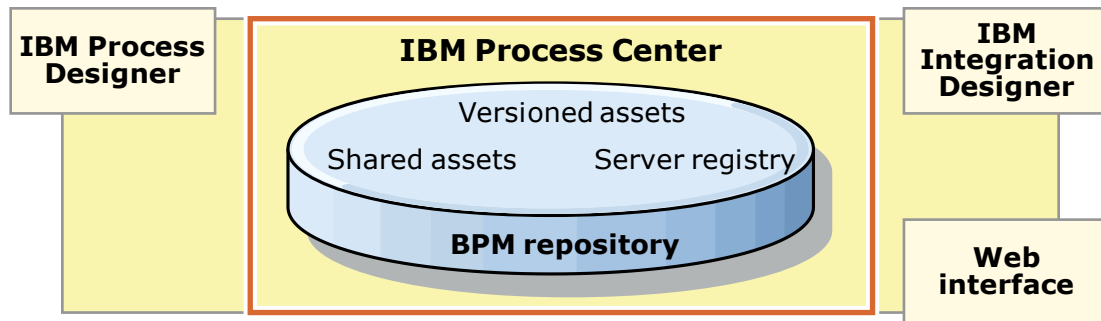
	IBM Integration Designer	IBM Process Designer
Container for integration artifacts	Module , which includes: <ul style="list-style-type: none"> • Integration logic (BPEL processes, human tasks, and business rules) • Data and interfaces • Transformations 	Process App , which includes: <ul style="list-style-type: none"> • Processes (BPD, human tasks, and rules) • Data and services
Container for shareable artifacts	Library , which includes: <ul style="list-style-type: none"> • Integration logic • Data and interfaces • Transformations • Web Service Ports 	Toolkit , which includes: <ul style="list-style-type: none"> • Processes • Data and services
Container for mediation services	Mediation module , which includes: <ul style="list-style-type: none"> • Mediation flows 	N/A

IBM Business Process Manager Advanced V8 Application Development

- **IBM Business Process Manager Standard V8 Capability**
 - IBM Process Center Overview
 - IBM Business Process Portal Overview
 - IBM Process Designer Overview
 - Process Designer
 - Process Inspector
 - Business Process Definition Model
 - Pools and Swimlanes
 - Flow objects

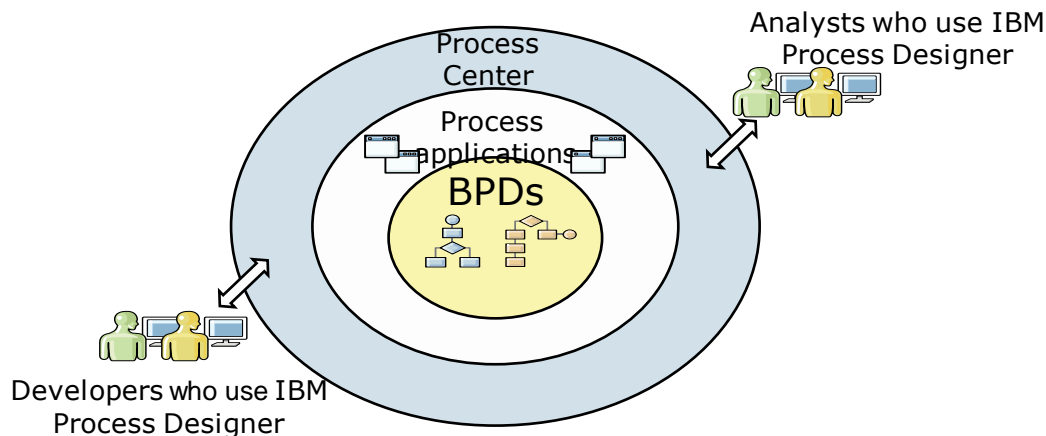
IBM Process Center Overview

- Repository for all Business Process Manager assets
- Lifecycle management and deployment of all applications
- Includes execution environment for development and testing
- Accessible from IBM Process Designer and from IBM Integration Designer
- Web interface by using IBM Process Center Console
- Includes Process Center server and the Performance Data Warehouse server



The Process Center: Process applications

- In Business Process Manager, process applications are the containers for the process models (BPDs) and supporting implementations that BPM analysts and developers create in IBM Process Designer. Supporting the implementation artifacts are services and toolkits
- Process applications can be either created in the Process Center or exported and imported.



IBM Process Center: Managing a process application

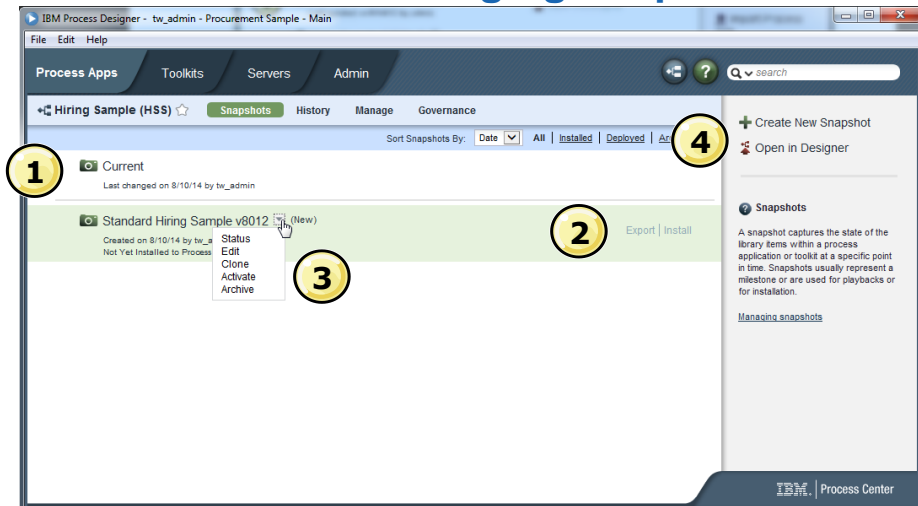


1 The repository list currently contains five process applications

2 Links used to create or import process applications

After a process application is created or imported, it becomes part of the repository list in the Process Center. This action allows authors to view the different process applications available and click to manage each one.

IBM Process Center: Managing snapshots



- 1 This area shows a list of snapshots for the process application
- 2 Links used to export a snapshot or move a snapshot to a different server
- 3 Menu to manage snapshots
- 4 Links to take a snapshot or open it in the Designer perspective of IBM Process Designer

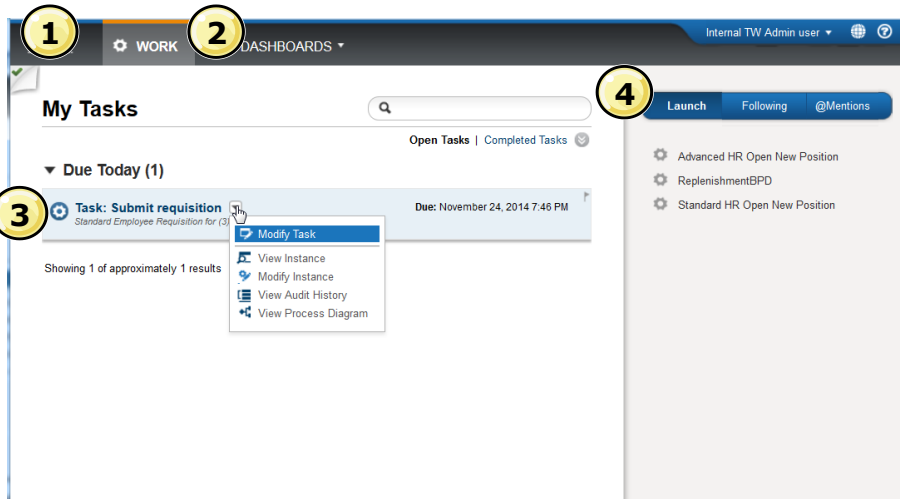
IBM Process Center: Managing snapshots

- Management of process applications in the Process Center revolves around process application snapshots
- A snapshot captures the state of the library items within a process application or toolkit at a specific point in time
- Each process app can have multiple versions or snapshots that designate development fidelity, such as version 1 or version 2
- Management of process applications is by snapshot and includes these possibilities:
 - Export a snapshot version for other authors to import into a different Process Center
 - Install snapshots to different environments
 - Examine the status of a snapshot
 - Edit snapshot profile information
 - Clone a snapshot, and activate or archive snapshot versions of the process application

IBM Business Process Portal Overview

- The Process Portal is the main tool that process participants or business users interact with to complete tasks and processes.
- Other tools, such as the IBM Business Process Manager mobile application can also be used to complete tasks and processes.
- The Process Portal also has use for project development, especially in terms of validation.
- BPM teams and business stakeholders want to reach consensus in the playback session to end a stage of development.
- When consensus is the goal, the Process Portal allows the team to view the process performance as it would function in a user environment.

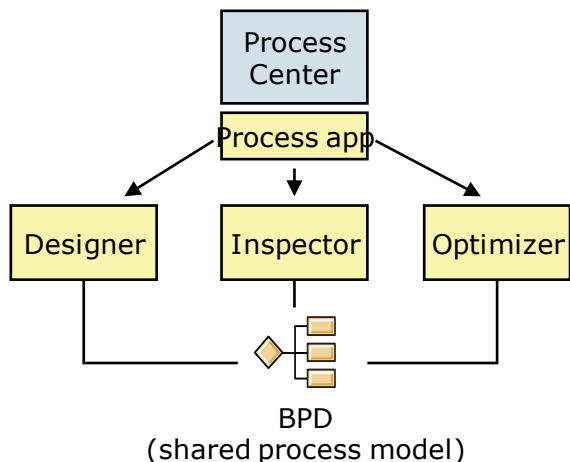
IBM Business Process Portal Overview



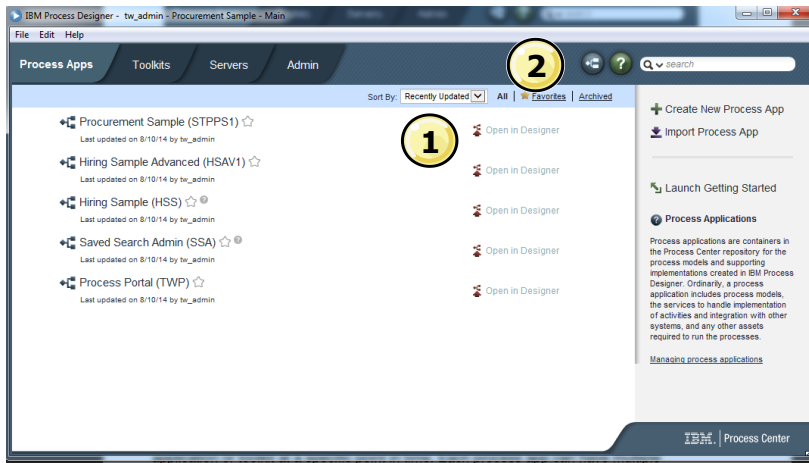
- 1 Tab to access My Tasks page
- 2 Menu to access dashboards (My Performance)
- 3 List of tasks; click to claim and complete task; use menu at right to manage tasks
- 4 Tabs for launching a process instance, following processes, and seeing when other users mentioned you

IBM Process Designer Overview

- Process modeling in IBM Business Process Manager is accomplished through the IBM Process Designer views or interfaces
- These interfaces allow developers or authors to create, manage, test, and optimize process models
- When modeling a process in IBM Process Designer, BPM teams are creating a business process definition (BPD)
- A BPD is the reusable shared model of a process, defining what is common to all runtime instances of that process model
- IBM Process Designer is composed of three key interfaces:
 - Designer (model and service)
 - Inspector
 - Optimizer



IBM Process Designer: Opening a process application

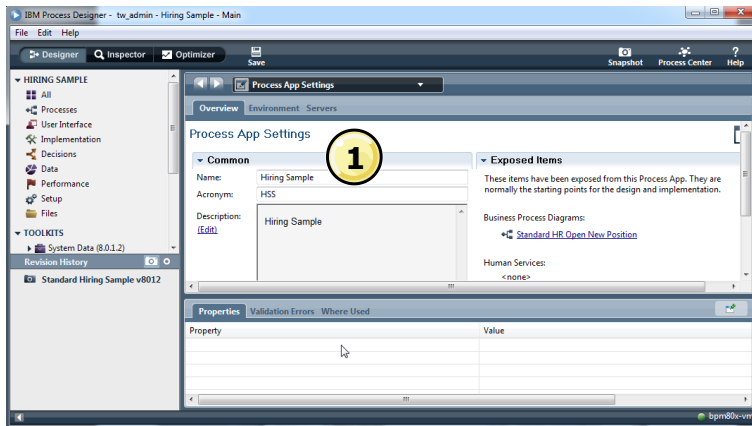


1 Link used to open application in Designer

2 Button that is used to open selected process application in Designer

Clicking either the Designer icon with the process application highlighted or the link Open in Designer to the right of the process application opens the process application in IBM Process Designer.

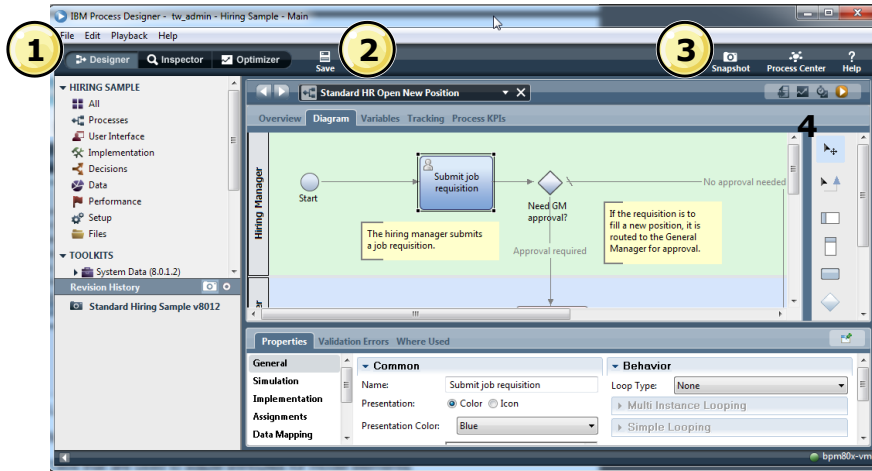
IBM Process Designer: Default view



1 Default application view of process application settings

- During typical process authoring, clicking Open in Designer or clicking the designer icon returns you to the last asset or place you were working.
- IBM Process Designer remembers where you were and opens to that screen
- The first time that you open a new process application, you see the default interface for the process application, the designer, which displays the process app settings.
- Process application settings are typically set in later playbacks during process implementation

IBM Process Designer: Application perspective

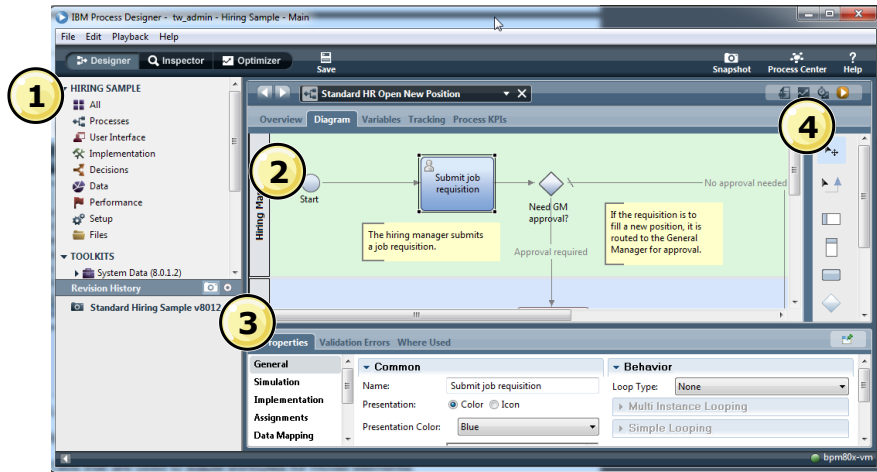


1 Three different views: Designer, Inspector, and Optimizer

2 Button for saving processes

3 Buttons for taking a snapshot, returning to the Process Center, and help

IBM Process Designer: Designer perspective



- 1 Library that is used to create and access artifacts
- 2 Modeling canvas that is used to model processes and services
- 3 Tabs that are used to adjust attributes for model elements
- 4 Element palette that is used for creating models

IBM Process Designer: Inspector perspective

The screenshot displays the IBM Process Designer Inspector perspective. The top menu bar includes File, Edit, Playback, and Help. Below the menu is a toolbar with buttons for Designer, Inspector (selected), and Optimizer, along with Save, Process Center Ser..., and All versions. The main area is divided into two panes. The left pane, titled 'Process Instances', shows a table of active process instances. The right pane, titled 'Execution State', shows the execution tree for the selected process instance. The bottom pane displays the process diagram for 'Standard HR Open New Position', which includes a start node, a task node 'Submit job request', a decision node 'Need GM approval?', and a flow node 'Approval required'. The diagram is annotated with five numbered circles: 1 points to the Refresh button in the top toolbar; 2 points to the Run task and debug buttons in the top toolbar; 3 points to the Process instance list in the left pane; 4 points to the Task list in the left pane; and 5 points to the Active process diagram with tokens in the bottom pane.

Instance Name	Snapshot	Process	Status	Due Date	Inst.	Status	Owner	Subject
Standard Employee Req...	Tip	Standard HR Open...	Active	Dec 9, 20...	4	Received	tw_admin	Task: Submit request
Standard Employee Req...	Tip	Standard HR Open...	Active	Dec 9, 20...	3			

1

Refresh button

2

Run task and debug buttons

2

Process instance list

4

Task list

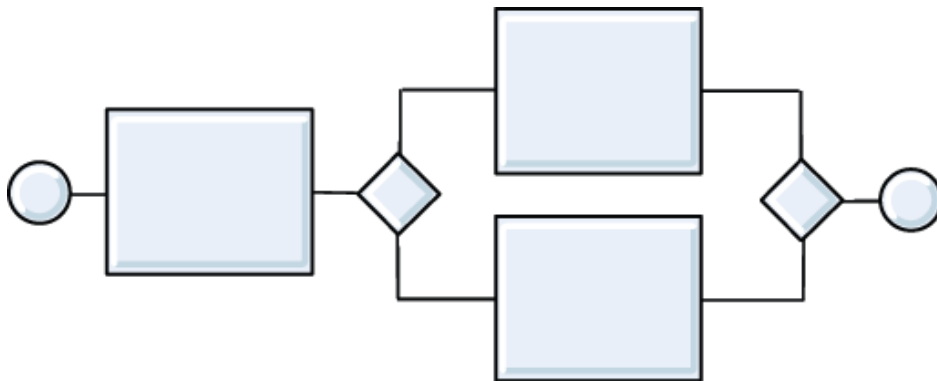
3

Active process diagram with tokens to represent active task

IBM Process Designer: Inspector perspective

- The inspector is a vital component of the IBM Process Designer and is key to an iterative approach to process development
- Using the inspector, individual authors and developers can run processes and services at any time on the Process Center server or remote runtime Process Servers
- This function allows BPM teams to run and debug process models consistently throughout the development cycle to ensure that the process model is meeting all the requirements of development.
- This action is important throughout the modeling phases.
- Likewise, the BPM development team can use the inspector to demonstrate current process model implementation to other modelers and developers that complement the user.
- It also shows portal runtime playback of the application.
- Playback sessions help capture important process model adjustment information from different stakeholders in a process, such as management, users, and business analysts.

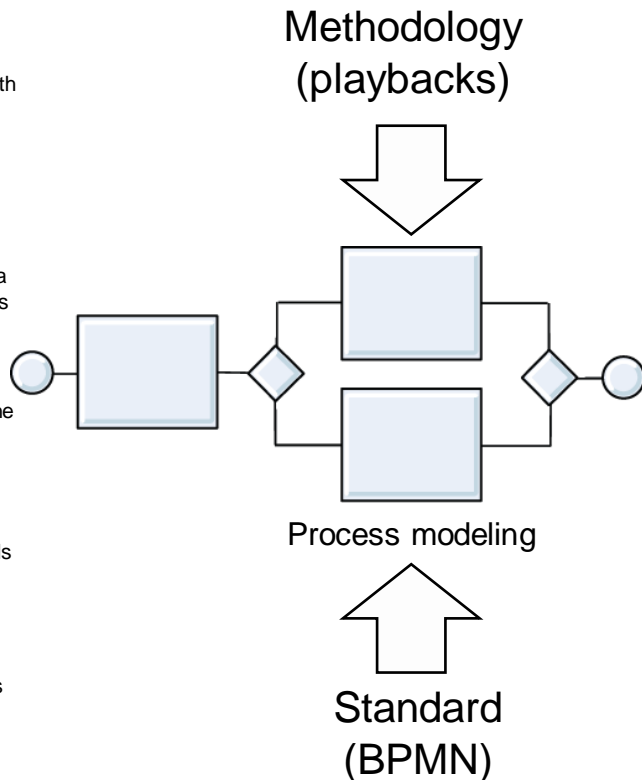
IBM Process Designer: Business Process Definition Model



- If creating a process model during discovery of the business process, the process model should reflect only the captured data
- The process model should not concern itself with solving process pain points (problems) until analytical modeling
- The process model is agile enough for continued adjustments, so the focus is to have the expected order of process tasks reflected in the model first

IBM Process Designer: Where to start

- As described earlier, process modeling captures the ordered sequence of activities within a process along with supporting information from start to end
- In modeling, the business process is framed with a workflow model to reflect component activities, the roles that are conducting those activities, conditional branching, and the sequencing of the flow of work between activities.
- In IBM Process Designer, this workflow model is called a business process definition (BPD), but is also sometimes called a process diagram.
- To translate process requirements that are documented in the discovery sessions into a process model, it is important for the BPM team to understand how to use the best methods and standards available
- Not only is it necessary to translate requirements, they must be translated correctly so that everyone clearly understands the process model
- Where a BPM team starts is by adhering to the standards used in process modeling, Business Process Model and Notation (BPMN)
- Concurrent to using BPMN to model the business process, a BPM team also uses a development method that works best to collaborate on modeling with business and IT. This development method is called playbacks which will be discussed later in this presentation



IBM Process Designer: About BPMN

- The standard flow chart-based notation for defining business processes
- Creates a standardized bridge for the gap between business process design and process implementation
- IBM Business Process Manager's Process Designer uses several core elements from BPMN
 - Pool
 - Lane
 - Event
 - Activity
 - Flow
 - Gateway

Process Designer element palette



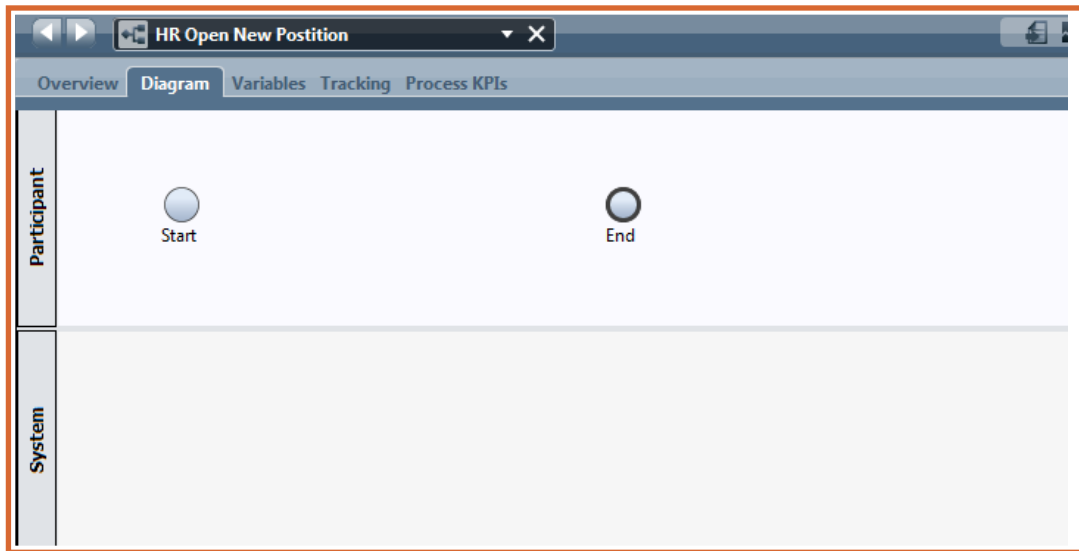
IBM Process Designer: Process modeling guidelines

- A process diagram or model is called a business process definition (BPD) in IBM Process Designer
- In general, a BPD should be as simple an abstraction as you can make it
 - A highly conceptual BPD is resilient to change
- Make sure that you use the Documentation area in the Properties tab for each element in IBM Process Designer to include important requirement notes

IBM Process Designer: Pools and Swimlanes

- The discovery and analysis session provide details about the business process that can be converted into BPMN process model elements
- These elements can be used in conjunction as a diagram that describes the business process and later runs the process application
- This section deals with two specific elements: pools and lanes.
- In IBM Process Designer, the default setup for newly created process models, or BPDs, is one pool and two lanes
- One lane represents a team and the other a system lane

IBM Process Designer: Pool

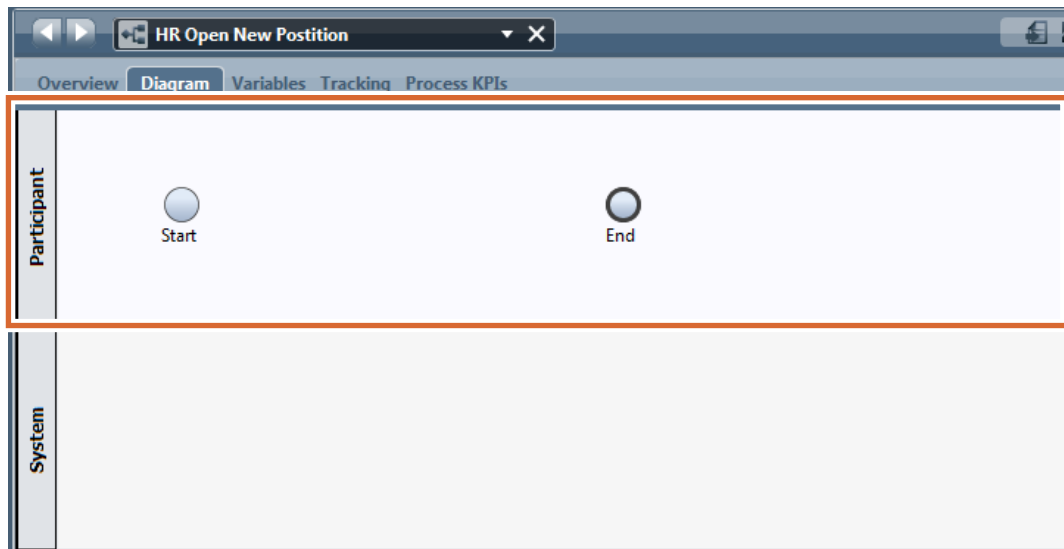


- A graphical element that is called a pool contains each business process definition
- The name of the pool is the same as the name of the BPD

IBM Process Designer: Pool

- A process that you model in IBM Process Designer includes the default IBM Process Designer pool, which consists of two default lanes
- In essence, the pool is the BPMN element that represents the entire business process
- The pool is the only element that is not found in the element palette and does not have properties, but it is the default setup for all models that are created in IBM Process Designer.

IBM Process Designer: Lanes

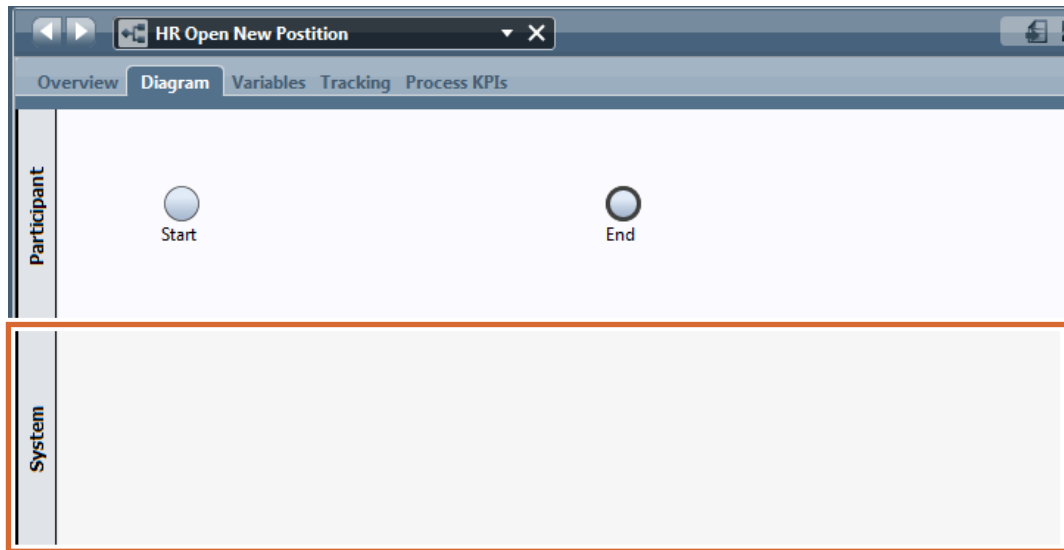


- Contained in each pool are lanes
- The top lane is a participant/team lane

IBM Process Designer: Lanes

- Each lane represents a team, and the process task responsible role is detailed in the discovery and analysis session.
- Lanes provide context for a process model as each lane contains a series of activities that are assigned to a specific team member or events that transpire in the process.
- Activities and events are covered in more detail in the other sections
- To obtain the details for the team during discovery and analysis, user stories help determine which teams are responsible to conduct specific process tasks
- Each of these teams is assigned to a lane when you model the process.
- It is important to remember that a team is a role, and not a person, in a process model.

IBM Process Designer: System lane

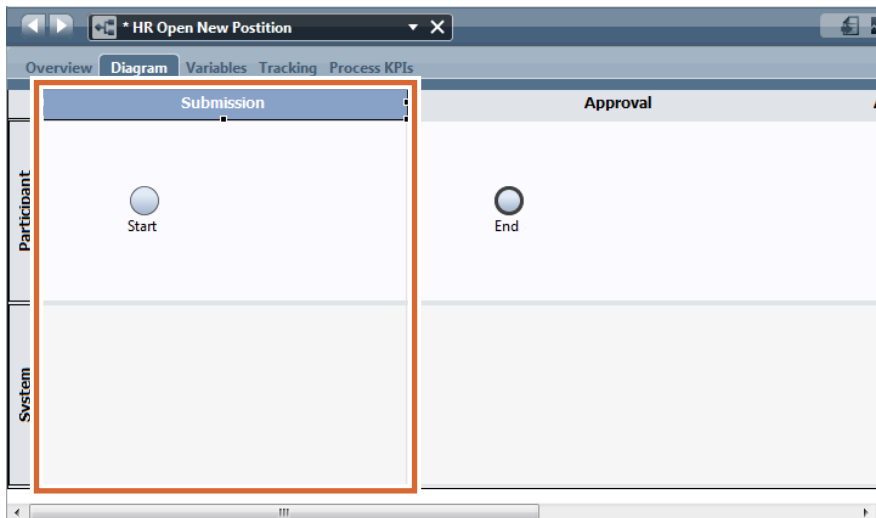


- Lanes can also be assigned to systems, and automated tasks are often in the designated system lane

IBM Process Designer: System lane

- When it comes time to define process tasks that are automated, the process model needs a way to communicate the automated tasks
- A process participant who is assigned to a lane is not always a responsible human role
- Process participants who are assigned to lanes can also be systems
- For example, the discovery and analysis session might find that a system, rather than a human role, completes a certain set of process tasks such as conducting a background check on a loan.
- IBM Process Designer has a specific default lane to contain these sorts of automated tasks: the system lane.
- During the initial process model build, tasks that are automated are represented as part of the system lane.
- Further automation of process tasks is designed as the process is improved and validated through the iterative playback project development.
- This iterative development can mean system lane movement and rearrangement to indicate where efficiency is found for the entire business process.

IBM Process Designer: Phases

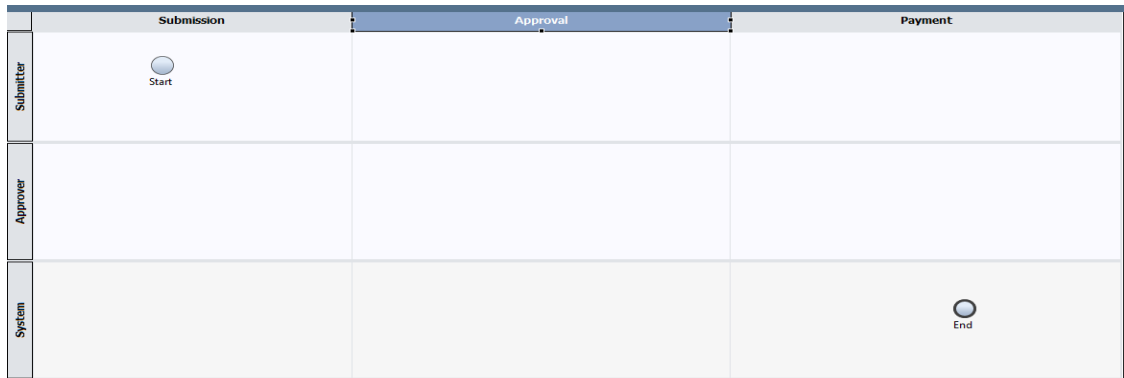


- The phase, Submission, is named with a noun that follows the naming convention

IBM Process Designer: Phases

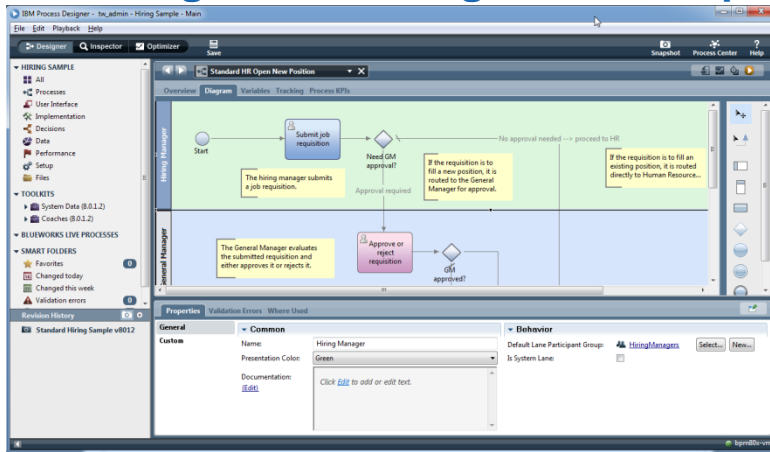
- When using IBM Process Designer, your company can optionally choose to use phases
- They do not provide any function for process implementation in IBM Process Designer, but they are a good organizational framework for descriptive and analytical process modeling.
- Phases are common in many process discovery and documenting products such as IBM Blueworks Live.
- If a process diagram is imported into IBM Process Designer from tools like IBM Blueworks Live, it is likely to automatically carry phases into your BPD.
- Vertical boxes represent phases in IBM Process Designer and contain various tasks that are correlated to the particular phase.
- If you choose to use phases, the good practice is to name each with a noun. Here are some examples for the naming conventions of a phase:
 - Approval
 - Orientation
 - Application Processing

IBM Process Designer: Example process Expense reimbursement



- The BPD has the lanes “Submitter”, “Approver”, and “System”, phases “Submission”, “Approval”, and “Payment”, plus a default start event and a default end event
- Here is an example business process that went through a process discovery and analysis stage
- The resulting documentation for the process contained the following elements:
 - Teams: Submitter, Approver
 - Phases: Submission, Approval, Payment
- As previously covered, translating the process information yields a start to the process model (BPD) in IBM Process Designer similar to the one pictured

IBM Process Designer: Modeling teams and phases



Modeling teams

1. Drag the lane icon from the element palette to add the necessary lanes to the two default lanes (participant and system) provided.
2. Click to select the lane and change the lane name on the properties tab to appropriately model the teams.

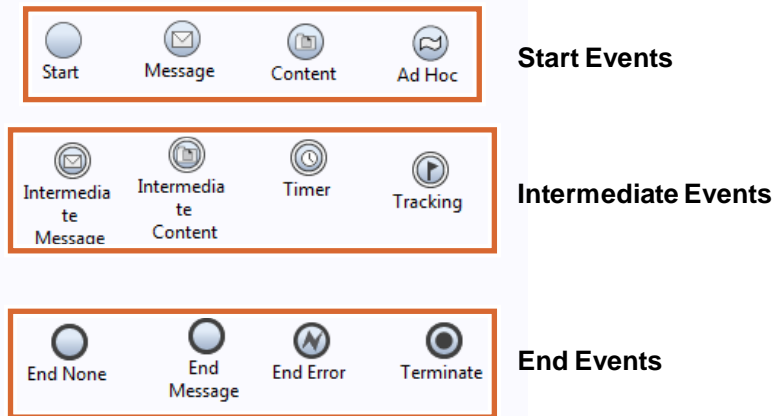
Modeling phases

1. Drag the phase icon from the element palette to any lane to add a milestone.
2. Click to select the lane and change the lane name on the properties tab to appropriately name the phases.

IBM Process Designer: Flow objects

- At this stage of diagramming a business process, an author considers flow objects for the model
- Flow objects in a process model are in the lane for teams because they represent either process task assignments or process controls
- This workshop will mainly focus on the most commonly used types of flow objects and not focus on every type of event, activity, or gateway available in IBM Process Designer

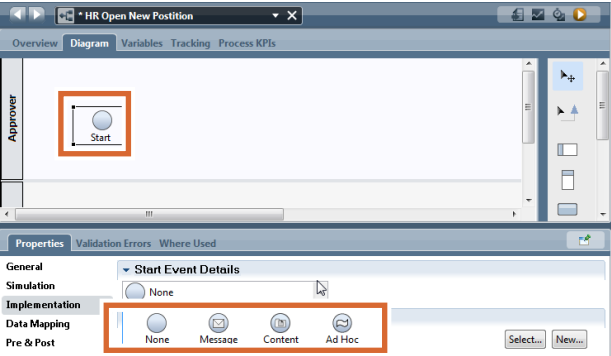
IBM Process Designer: Events



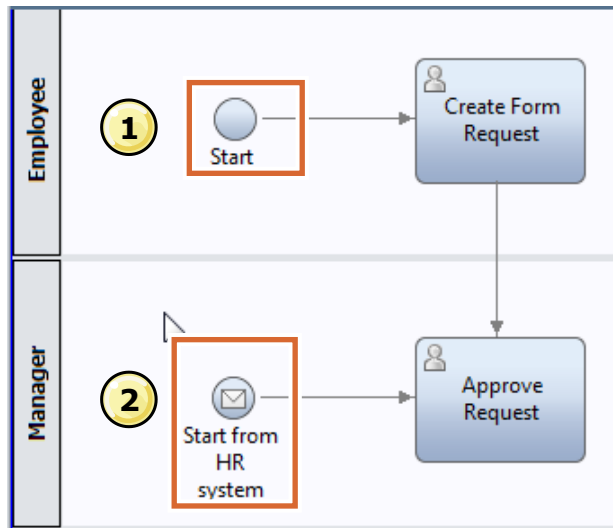
- Events are control flow objects for a process model
- There are three categories of events:
 - start events
 - intermediate events
 - end events

IBM Process Designer: Start events

Start events

- A circle encompassed by a single line represents a start event.
 - Start events trigger the initiation of the process through a manual or automatic input.
 - Authors describe the input in the properties tab documentation box that is provided for the element.
- 
- The screenshot shows the IBM Process Designer interface. The top bar has tabs for Overview, Diagram, Variables, Tracking, and Process KPIs. The main diagram area shows a process flow with a start event (a circle with a line) labeled 'Start'. The Properties panel on the right shows the 'Start Event Details' section, which includes a dropdown menu for 'Implementation' with options: None, Message, Content, and Ad Hoc. The 'None' option is selected and highlighted with a red box.
- There are four types of start events in IBM Process Designer
 - **None** – Use to define the starting point for a sub-process or a process that is called by another process
 - **Message** - Use the message implementation option if you want an incoming message to kick off a process or an event subprocess.
 - **Content** - Use the content implementation option if you want an event to kick off a process.
 - **Ad hoc** - Use this implementation option when you include ad hoc actions that are run at any time during process execution

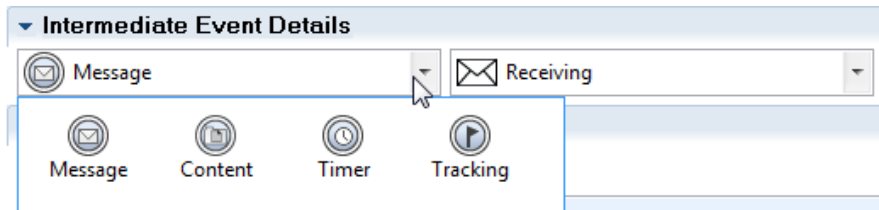
IBM Process Designer: Multiple start events



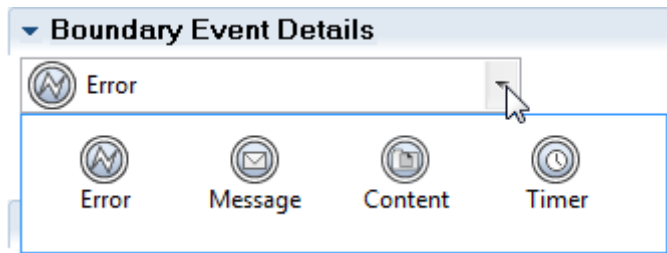
- 1 There can be only one “none” start event per BPD
- 2 To add a second start event to a BPD, you must use a message or ad hoc start event
 - This example shows a message start event

IBM Process Designer: Intermediate events

- Four types of sequence flow intermediate events: message, content, timer, and tracking



- Four types of boundary (attached) intermediate events: error, message, content, and timer



IBM Process Designer: Intermediate events: Timer

- Used for modeling escalation paths or delays in your BPDs

The screenshot displays the IBM Process Designer interface for a process titled "HR Open New Position". The main diagram area shows a flow starting from a "Start" event, followed by a "Submt Claim" task, and then a timer intermediate event. The timer event is represented by a circle with a clock icon. The "Properties" panel on the right is expanded, showing the "Intermediate Event Details" section where the event type is set to "Timer". Below this, the "Timer Properties" section is visible, with "Trigger On" set to "After Start of Parent Process or Subprocess", "Custom Date" set to an empty field, and "Before/After Difference" set to "0" hours.

Process Diagram:

```

graph LR
    Start((Start)) --> SubmtClaim[Submt Claim]
    SubmtClaim --> Timer((Timer))
  
```

Properties Panel:

- General**
- Simulation**
- Implementation**
- Data Mapping**
- Pre & Post**

Intermediate Event Details

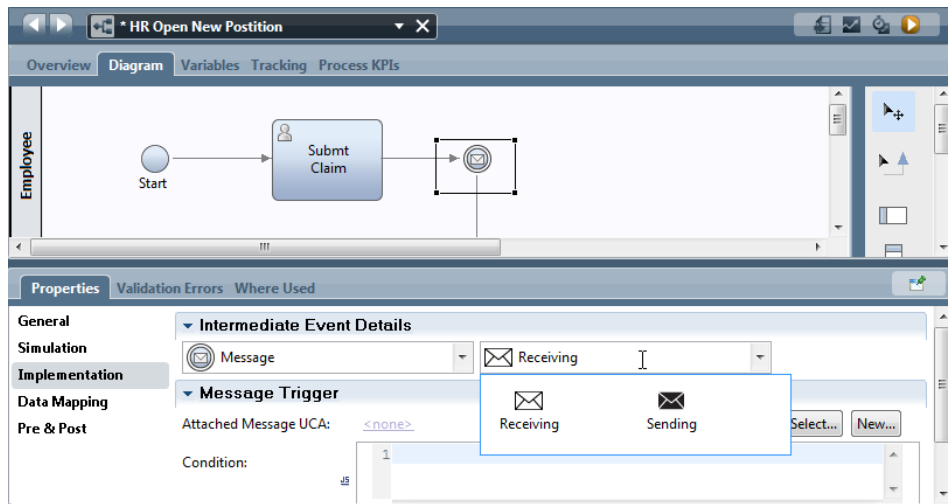
- Timer

Timer Properties

- Trigger On: After Start of Parent Process or Subprocess
- Custom Date:
- Before/After Difference: 0 Hours

IBM Process Designer: Intermediate event: Message

- Used for modeling a message event that is received or sent while a process is running



IBM Process Designer: Intermediate events: Error

- Used for catching errors and handle errors with login in the process flow

The screenshot displays the IBM Process Designer interface for a process titled "HR Open New Postition". The "Diagram" tab is active, showing a process flow starting with a "Start" event (blue circle) leading to a "Submt Claim" task (blue rectangle with a person icon). An "Error" event (blue circle with a lightning bolt icon) is attached to the task. The "Properties" panel is open, showing the "Implementation" tab. Under "Boundary Event Details", the event type is set to "Error". The "Interrupt activity" checkbox is checked, and the "Repeatable" checkbox is unchecked. Under "Error Properties", the "Catch All Errors" radio button is selected, and the "Catch Specific Errors" radio button is unselected. The "Error code" field is empty.

Properties Validation Errors Where Used

General

Simulation

Implementation

Data Mapping

Pre & Post

Boundary Event Details

Error

Interrupt activity: ☒

Repeatable: ☐

Error Properties

☒ Catch All Errors ☐ Catch Specific Errors

Error code:

IBM Process Designer: Intermediate events: Content

- Used for modeling an Enterprise Content Manager event that the BPD receives

The screenshot displays the IBM Process Designer interface. The top window shows a process diagram titled "HR Open New Postition" (sic). The diagram includes a "Start" event, a "Submit Claim" task, and an intermediate event (represented by a circle with a document icon). The "Properties" panel on the right is open, showing the "Intermediate Event Details" section. The event type is set to "Content". The "Content Trigger" section shows the "Attached Content UCA" as "<none>" and the "Condition" as "1".

Process Diagram:

```

graph LR
    Start((Start)) --> SubmitClaim[Submit Claim]
    SubmitClaim --> Event(( ))
    style Event fill:#fff,stroke:#000,stroke-width:2px
  
```

Properties Panel:

- General**
- Simulation**
- Implementation**
- Data Mapping**
- Pre & Post**

Intermediate Event Details

- Event Type: Content

Content Trigger

- Attached Content UCA: <none> [Select... New...]
- Condition: 1

IBM Process Designer: Intermediate events: Tracking

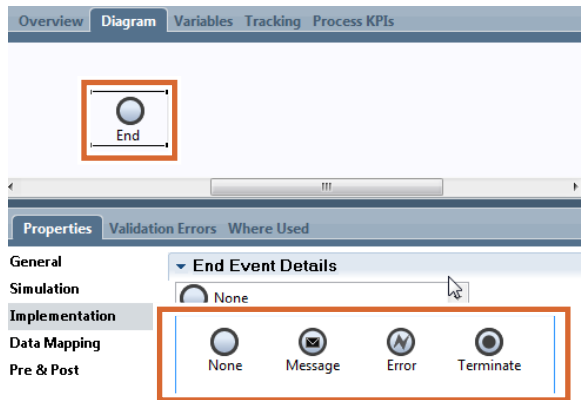
- Used for indicating a point in a service at which you want Process Designer to capture the runtime data for reporting purposes
- This event is an IBM specific intermediate event

The screenshot displays the IBM Process Designer interface. The top window, titled '* HR Open New Postition', shows a process diagram with a 'Start' event, a 'Submt Claim' task, and an intermediate event. The 'Properties' panel is open, showing the 'Implementation' tab. Under 'Intermediate Event Details', the event type is set to 'Tracking'. Under 'Tracking Properties', the 'Tracking Group' is set to '<none>' and the 'Performance Data Warehouse ID' is 'K3883a64d8e941'. There are 'Select...' and 'New...' buttons next to the Tracking Group field.

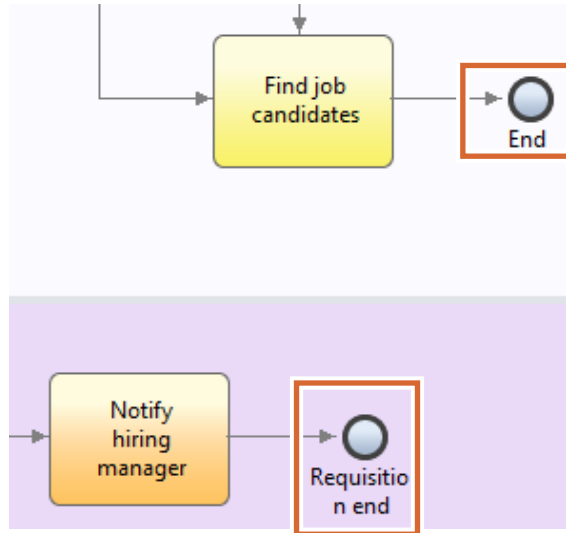
IBM Process Designer: End events

End Events

- An end event is represented as a circle encompassed by a dark thick single line.
- End events are reached in a process when a final decision from all activities or a partial set of activities is reached.
- There are four types of end events
 - **None** - Use to show where a path ends
 - **Message** - Sends a message at the conclusion of a process instance, which is typically received by a start message event or intermediate message event in another process or processes
 - **Error** - Use to catch process exceptions errors. It can trigger further process flow to take action.
 - **Terminate** - Terminates the process instance and all activities, even if parallel paths are still run



IBM Process Designer: Multiple end events



- This process has two none end events: End and Requisition ended
- When you have more than one none end event, each should have a unique name

IBM Process Designer: Activity Task



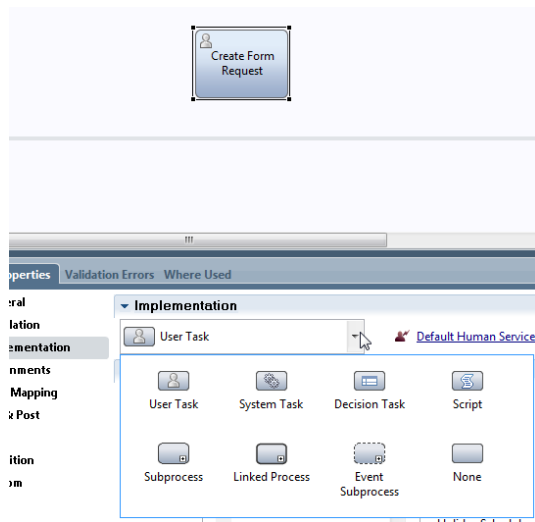
The BPMN element representation of an activity is a rectangle with rounded corners

- An activity represents a single task that a process participant accomplishes from beginning to end

IBM Process Designer: Activity Task types

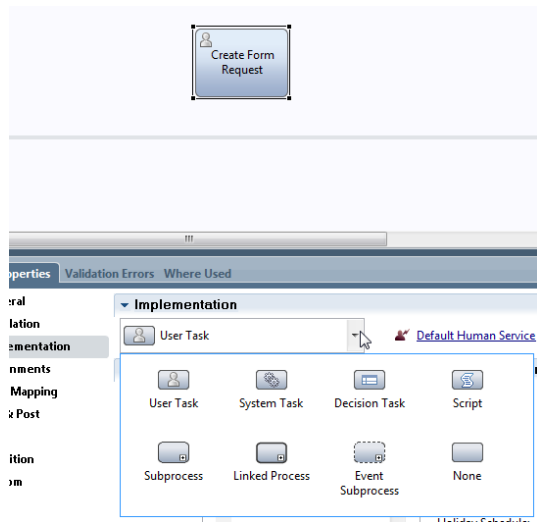
There are eight types of Activities

- **User Task** - A task that is performed by a process end user (a human, performer). Can be implemented by a human service, integration service, advanced integration service, or external implementation
- **System Task** - An automated activity that is performed by the system. Can be implemented by an integration services, advanced integration services, general service, or external implementation
- **Decision Task** - A task that returns the result of a decision, as implemented by a rule. In BPMN 2.0, it is also referred to as a business rule task.
- **Script** - The activity will execute a Java script when performed. In BPMN 2.0, this activity is also called a Script Task



IBM Process Designer: Activity Task types

- **Subprocess** - A non-reusable process embedded within another process. A subprocess shares the data used by its parent process.
- **Linked Process** - The activity will call another process within the process application. The process that is called has its own data, requiring data mapping from this activity.
- **Event Subprocess** - A specialized subprocess used for event handling. It is not connected to other activities through sequence flow and occurs only if its start event is triggered.
- **None** - A task that has no specified implementation or performer. At runtime, the task will complete immediately after it starts.



IBM Business Process Manager Advanced V8 Application Development

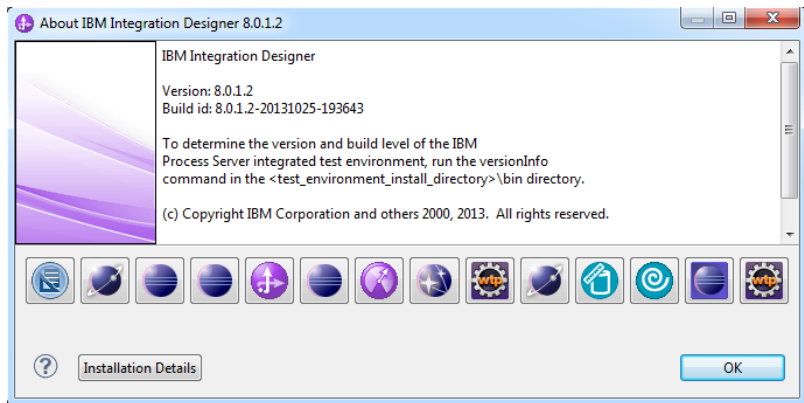
- **IBM Business Process Manager Advanced V8 Capability**
 - IBM Integration Designer Overview
 - Module Packaging and Deployment
 - Service Component Architecture
 - Web Services
 - Interfaces, Business Objects and Data Objects
 - Mediation Modules and Primitives
 - Business Process Choreography (Bpel)
 - Business Rules
 - Human Tasks
 - Testing

IBM Integration Designer Overview



IBM Integration Designer: Overview

- IBM Integration Designer is the unified development tool for building SOA-based integration applications for IBM Process Server, IBM Process Center, and WebSphere Enterprise Service Bus
 - Visual development environment that requires minimal programming skill
 - Provides prebuilt mediation functions and BPEL activities
 - A comprehensive environment for developing, assembling, testing, deploying, and managing integration modules and mediation modules for run time

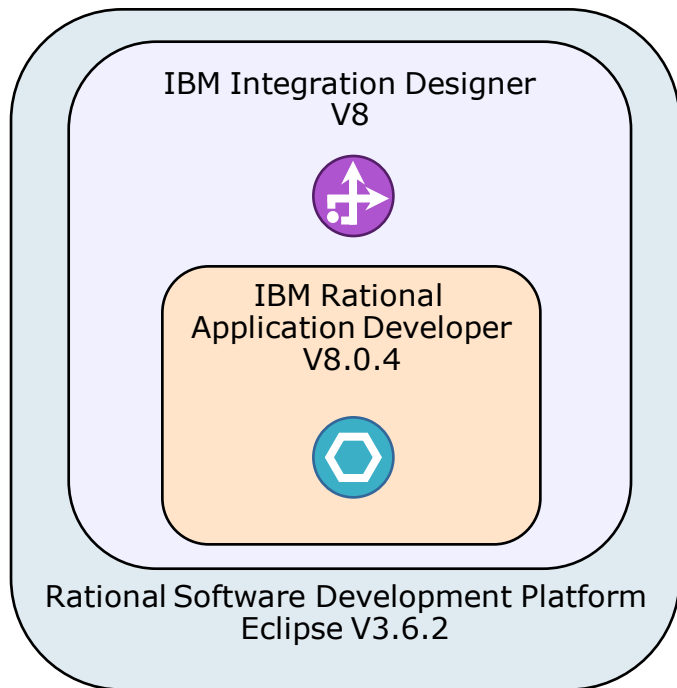


IBM Integration Designer: Roles

Role	Responsibilities
Integration developer	<ul style="list-style-type: none"> • Focuses on building SOA and EAI solutions <ul style="list-style-type: none"> – Top-down, bottom-up, or meet-in-the-middle • Creates composite applications from integrated components • Has a basic understanding of business modeling • Expects authoring tools to simplify and abstract advanced implementation details • Is familiar with basic programming concepts <ul style="list-style-type: none"> – Loops, conditions, string manipulation, and other programming concepts • Understands business process choreography, workflow (including human interaction), WSDL, and BPEL • Creates mediation modules to implement connectivity logic • Works with the IBM Process Center repository • Manages and deploys snapshots in IBM Process Center
Application (IT) developer	<ul style="list-style-type: none"> • Is knowledgeable in one or more application development platforms (Java EE) • Understands SOA, process choreography, workflow, WSDL, and BPEL • Implements application-specific business logic for integrated solutions such as EJBs • Exposes application logic as a service

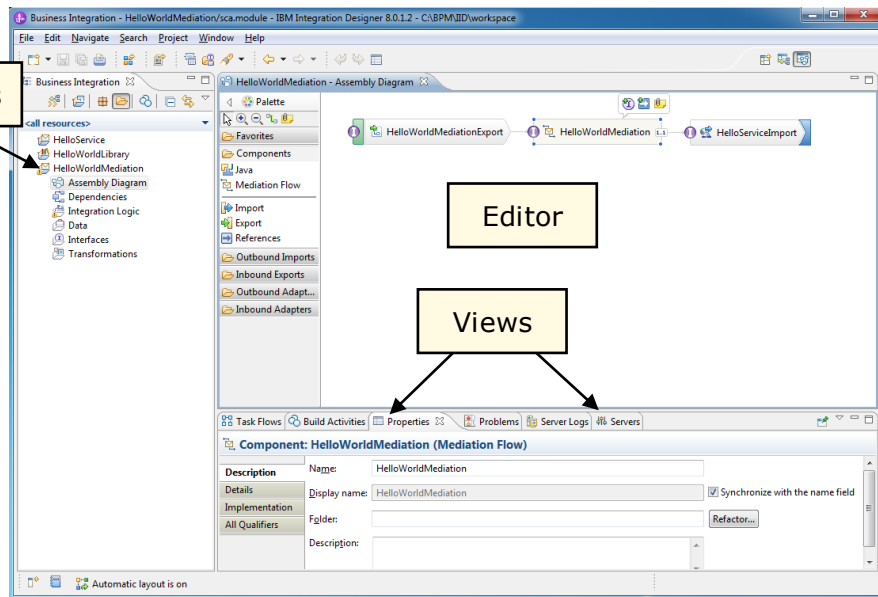
IBM Integration Designer: Platform architecture

- IBM Integration Designer is based on Rational Application Developer
 - Not all Rational Application Developer features are included
- Rational Software Development Platform provides the foundation for IBM Integration Designer and Rational Application Developer
 - Based on Eclipse V3
 - Contains the common components for Eclipse-based products
 - Installed once per system with the first product



IBM Integration Designer: Application composition and deployment

- Use the Business Integration perspective to develop business integration projects
- Default views: Business Integration, Editor, Task Flows, Build Activities, Properties, Problems, Server Logs, Servers



IBM Integration Designer: Creating modules and libraries

- In the Business Integration perspective, you create an SCA library, a module (called a business integration module in this course), or a mediation module
- Modules and libraries contain multiple SCA artifacts that are grouped according to type
- Libraries are projects that are used to store shared resources
 - To access libraries, add them to module dependencies
- Integration modules provide the business services and mediation modules provide connectivity logic
 - Mediation flows and business services are modeled as SCA components
 - SCA components are wired together in the assembly diagram to form applications

IBM Integration Designer: Module components

- Business integration modules include:
 - **Assembly diagrams:** wire SCA components together to form applications
 - **Dependencies:** include other modules, libraries, Java EE projects, and predefined resources
 - **Integration logic:** artifacts that perform specific tasks (business processes, state machines, human tasks, business rules and rule groups, or mediation flows)
 - **Data:** business objects
 - **Interfaces:** service interfaces and their operations
 - **Transformations:** data (XML) maps and relationships
- Mediation modules include:
 - **Integration logic:** *only* mediation flows and subflows for processing messages that are passed between services
 - Artifacts present in integration modules: assembly diagram, dependencies, data types, interfaces, and transformations

Integration module

- ▲ HelloWorldProcess
 - ▷ Assembly Diagram
 - ▷ Dependencies
 - ▲ Integration Logic
 - ▷ BPEL Processes
 - ▷ Human Tasks
 - ▷ Java Usages
 - ▷ Data
 - ▷ Interfaces
 - ▷ Transformations

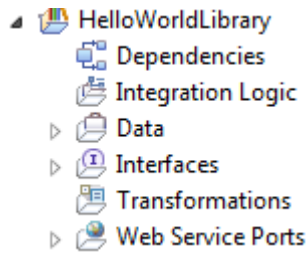
Mediation module

- ▲ HelloWorldMediation
 - ▷ Assembly Diagram
 - ▷ Dependencies
 - ▲ Integration Logic
 - ▷ Mediation Flows
 - ▷ Mediation Subflows
 - ▷ Data
 - ▷ Interfaces
 - ▲ Transformations
 - ▷ Data Maps
 - ▷ Data Map Test Data

IBM Integration Designer: Library components

- Libraries are project types for storing artifacts that are shared between several modules
- Libraries contain the following artifacts:
 - **Dependencies:** are used to include other libraries, predefined resources, and so on
 - **Integration logic:** contains artifacts that do specific tasks (business processes, state machines, human tasks, business rules and rule groups, mediation flows or subflows)
 - **Data:** business objects
 - **Interfaces:** service interfaces and operations
 - **Transformations:** contains data (XML) maps and relationships
- Libraries are not runnable applications
 - No assembly diagram

Library



Comparing IBM Integration Designer and Process Designer

	IBM Integration Designer	IBM Process Designer
<i>Container for integration artifacts</i>	Module, which includes: <ul style="list-style-type: none"> • Integration logic (BPEL processes, human tasks, business rules) • Data and interfaces • Transformations 	Process app, which includes: <ul style="list-style-type: none"> • Processes (BPD, human tasks, rules) • Data and services
<i>Container for shareable artifacts</i>	Library, which includes: <ul style="list-style-type: none"> • Integration logic • Data and interfaces • Transformations • Web service ports 	Toolkit, which includes: <ul style="list-style-type: none"> • Processes • Data and services
<i>Container for mediation services</i>	Mediation module, which includes: <ul style="list-style-type: none"> • Mediation flows 	n/a

Module Packaging and Deployment



Building modules

- When artifacts are created and assembled, you build the projects for testing and deployment
- IBM Integration Designer projects are automatically built (compiled) by default when they are saved in the workspace
 - Choose **Project > Build Automatically** to toggle automatic builds
- The Build Activities view is the central point for controlling build behavior in IBM Integration Designer
 - The Build Activities view has a simple user interface with which you can easily select build activities for both automatic and manual builds
 - You can also start immediate manual builds that are independent of your build activity selections

Build Activities view

- The Build Activities view is divided into three sections:
 - Select workspace activities to run during a build: specify build activities to run when an automatic or manual build occurs
 - Manual triggering of workspace build activities: immediate manual build that temporarily overrides build activity selections
 - Project status: view the build or deployment status of projects and operational state of servers

1 Select workspace activities to run during a build

☐ Validate
☒ Validate and update deployment code
☐ Validate, update deployment code, and update running servers

2 Manual triggering of workspace build activities

3 Project status

Name	Validated	Deployment Code Updated	Status on the started 'IBM Process Server v8.0 at localhost' Server
HelloService	Yes	Yes	① Not deployed to this server
HelloWorldLibrary	Yes	Not applicable	Not applicable
HelloWorldMediation	Yes	Yes	① Not deployed to this server
HelloWorldProcess	Yes	Yes	① Not deployed to this server

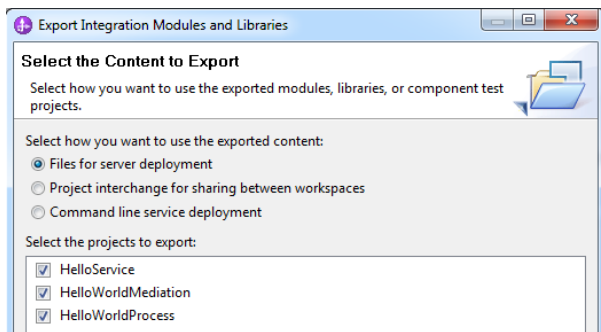
Packaging modules

- In IBM Integration Designer, publishing modules to the test runtime environment automatically packages them in enterprise archive (EAR) files and installs them as enterprise applications
- Modules that are created in IBM Integration Designer can be packaged for use outside the test environment:
 - You can export a module as an EAR file for remote deployment
 - You can use the **serviceDeploy** command-line tool to create an installable EAR file that includes components outside your workspace (**serviceDeploy** is covered in a later unit)
 - You can export project interchange files for exchange between workspaces

Exports as EAR file

Exports as PI file

Exports as compressed file



Library deployment

- Deploying libraries in modules (default setting)
 - A copy of the library JAR file is included in each module that uses it
 - After deployment, if library resources change, modules by using the library must be updated
- Deploying libraries globally
 - The library is exported as a deployable JAR file
 - The resource references for a module are configured to use the global library by using the module deployment editor
 - This option saves memory but you must deploy each library independently

▼ Sharing Across Runtime Environments

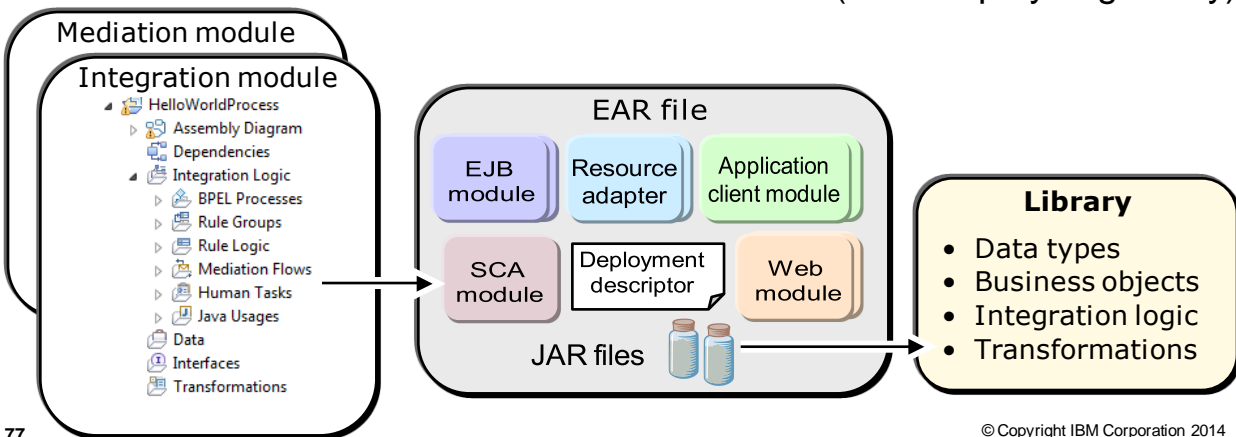
Specify how this library will be shared when it is running on the server. [More...](#)

- ☒ **Module** A copy of this library will exist on the server for each module that uses it.
- ☐ **Global** The library will be shared among all modules that are running. This option will be more memory efficient when many modules need to use this library.

[Instructions for runtime installation.](#)

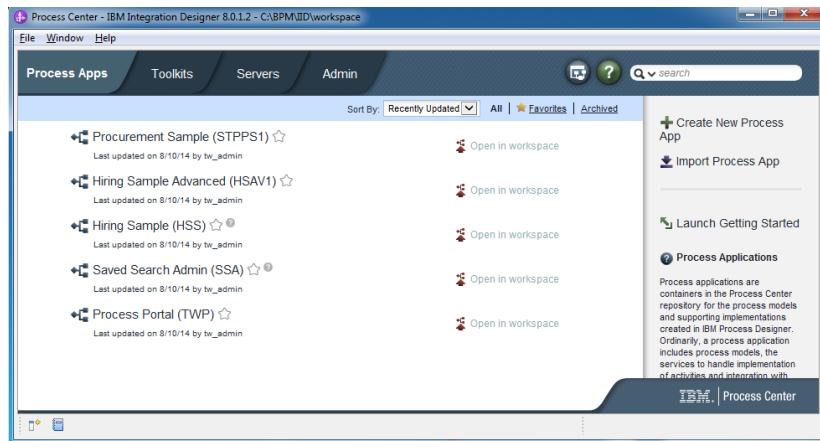
Module deployment

- When a module is built and packaged, it is the basic unit of deployment
 - Integration modules are packaged in EAR files as SCA modules
 - The EAR is deployed to the runtime environment
- The EAR file can contain other components:
 - Java EE projects (EJB modules, and web modules)
 - Java projects
 - Dependent libraries
- A library might be included as a JAR file (if not deployed globally)



IBM Process Center repository

- Use the repository in IBM Process Center to share artifacts with other developers
- When an application is built, it can be placed on the repository
- The Process Center perspective in IBM Integration Designer might be used to:
 - Access the repository
 - Import process applications and toolkits
 - Get updates from the repository and send updates to the repository

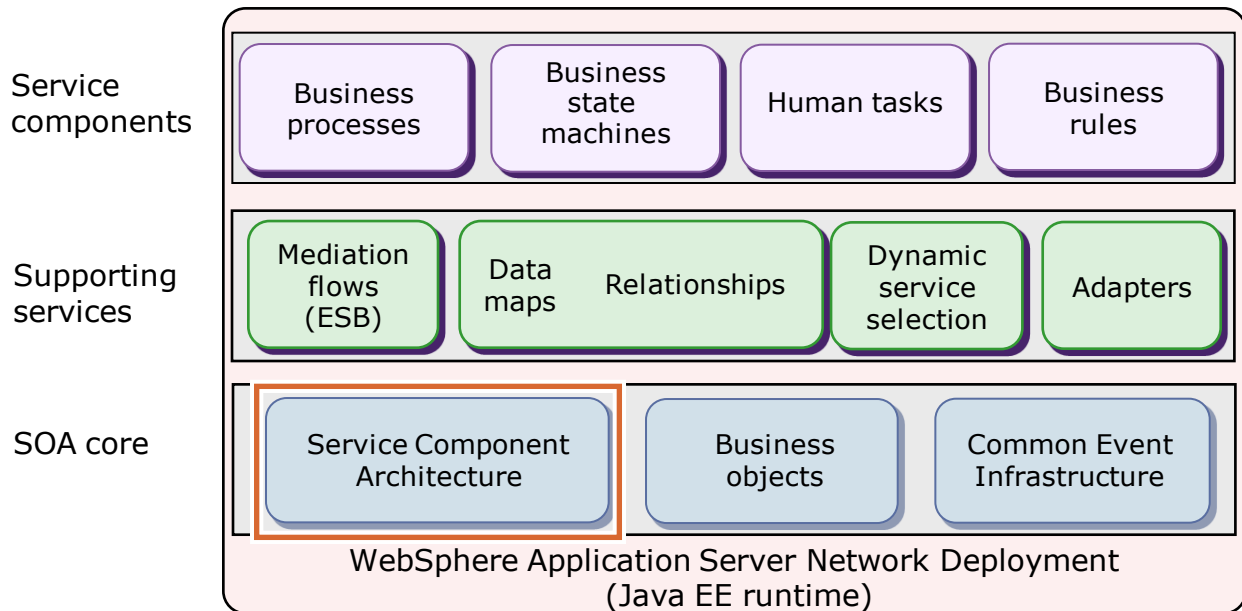


Service Component Architecture



Service Component Architecture

- Service Component Architecture is an SOA core component
- SCA provides the programming model for business integration modules and mediation modules

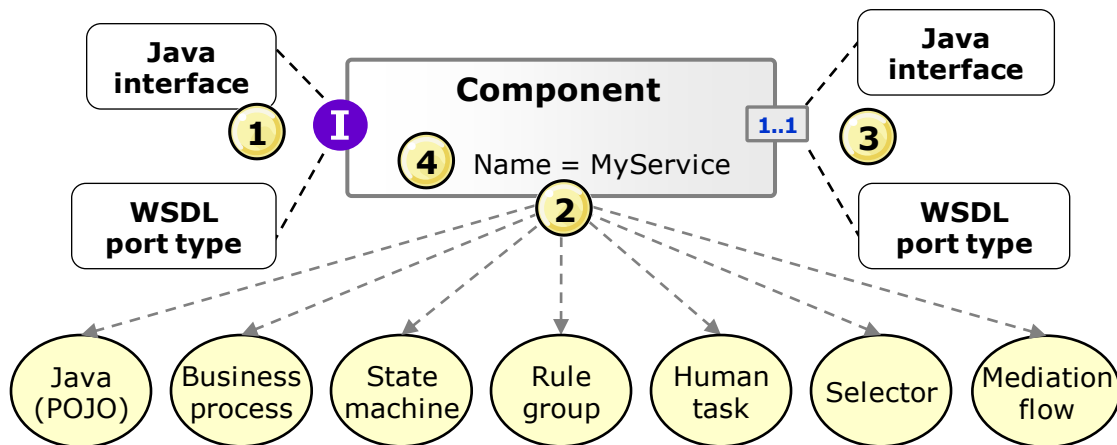


Overview and business value of SCA

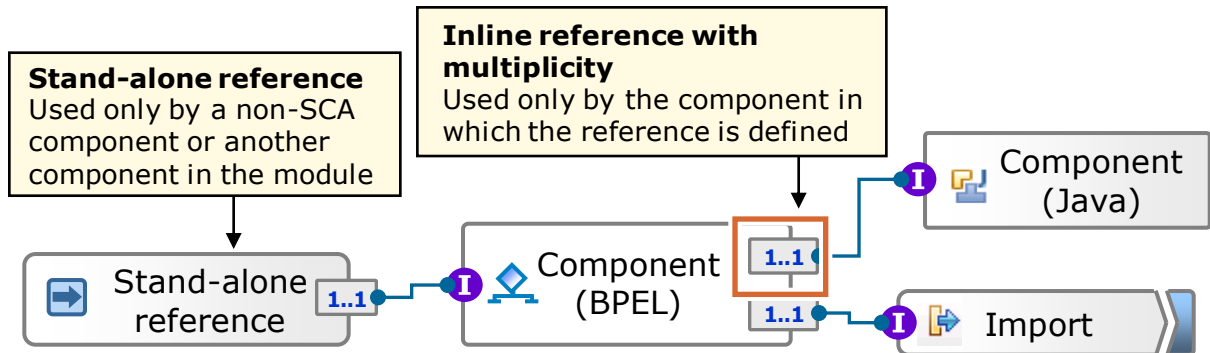
- SCA is a service-oriented component model that provides a declarative, high-level method of creating relationships between services
 - Only the service contract is visible
 - Service implementation details are not considered
 - When services change, only the declaration is changed, not the mechanical steps in the application code
- SCA provides a single service component abstraction for services that might already be implemented as business processes, Java classes, and mediation flows
 - Abstraction separates “business logic” from “infrastructure logic,” allowing developers to focus on business problems, not infrastructure code
- Without SCA, you must write code to communicate with the services in your SOA-based application
 - This application is not loosely coupled, and is not easy to change
 - You must change application code to respond to infrastructure changes

SCA components

- Components are discrete units of business logic that contain:
 - 1..N interfaces: are used to call the component and provide the service contract
 - Implementation: a representation of the service type (the physical implementation is separate from the SCA component)
 - 0..N references: are used to call other components
 - Name: unique in the SCA module

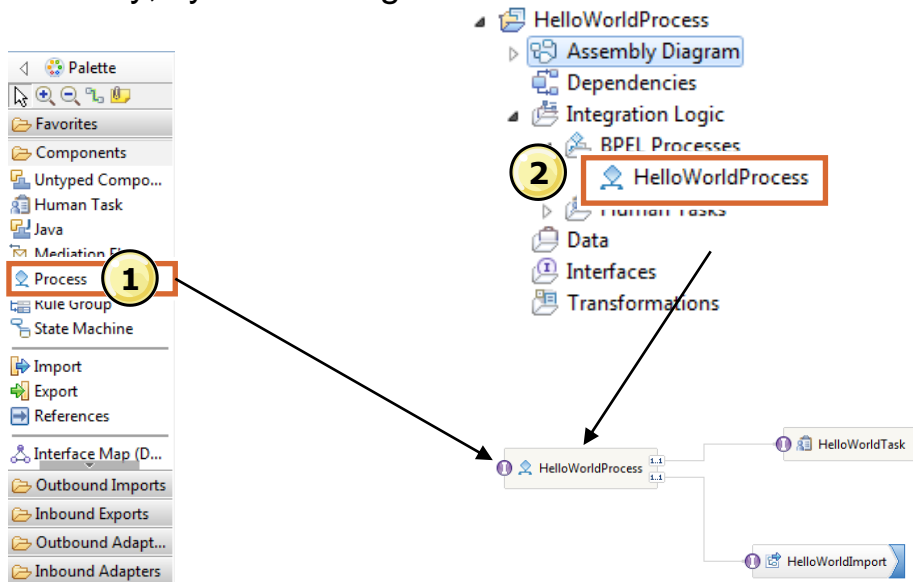


- A reference is used to specify the target of a service invocation
 - Defined on the calling component or in a stand-alone reference
- Reference definitions include:
 - Name: is used to look up the appropriate service
 - Multiplicity: number of services that can be wired to the reference
 - Interface: interface used to invoke the target component
 - Wire: used in a module assembly to identify the target service component that resolves the reference (a component or import)

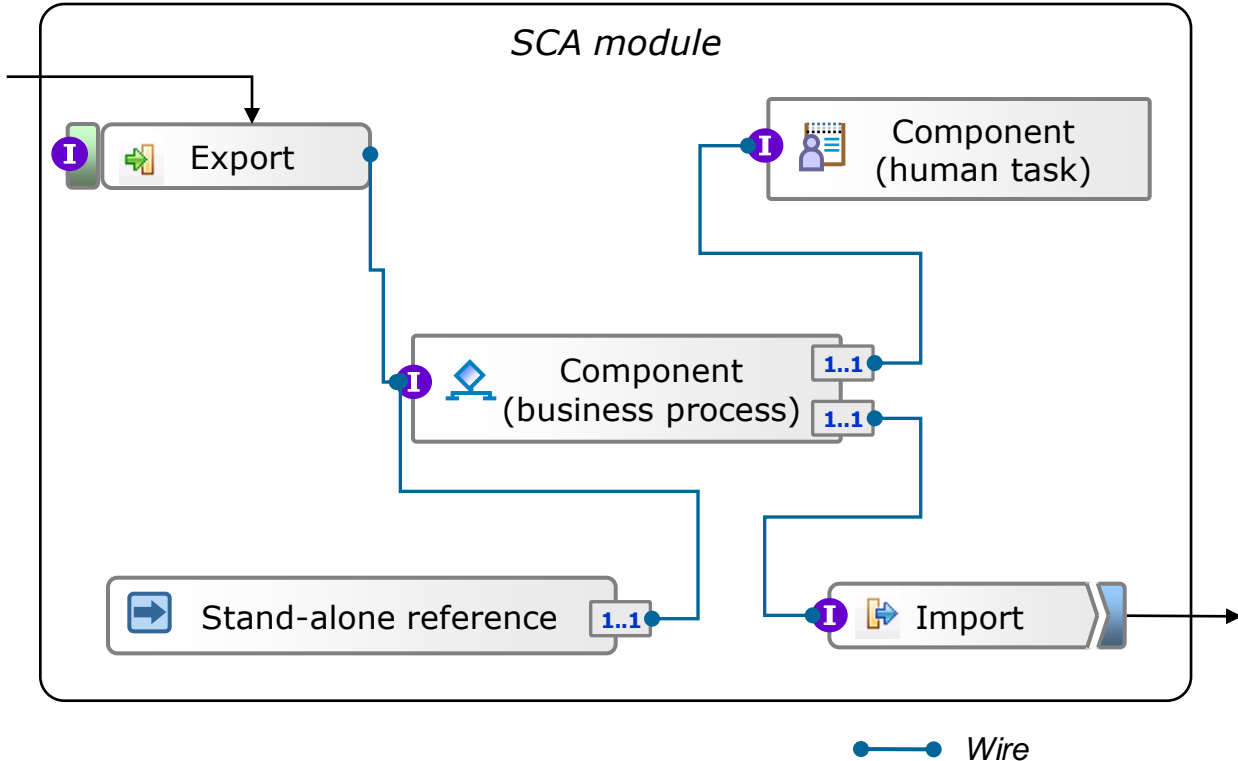


Composing SCA modules: Assembly Diagram editor

- You assemble SCA modules by first adding SCA components to the assembly diagram
- You then create relationships between the components in the module visually, by connecting them with wires

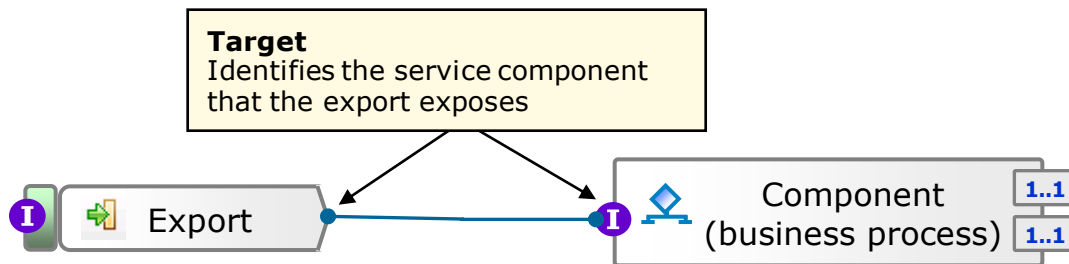


SCA module components: Overview



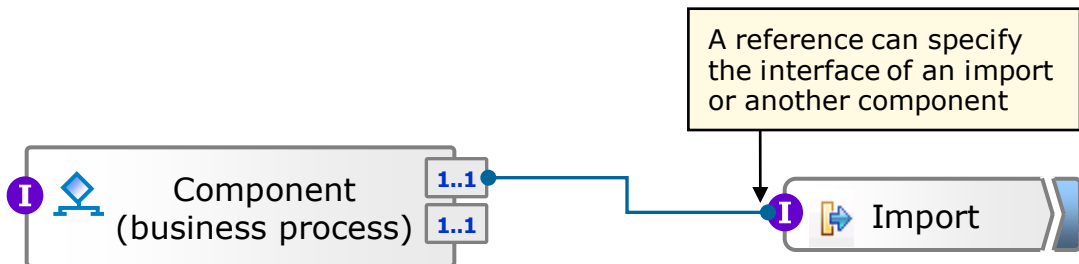
SCA module components: Exports

- Exports provide access to callers outside the module
- Exports separate the infrastructure details required to communicate with the service from the service implementation
- Export definitions include:
 - Name
 - Target
 - 1...N interfaces
 - Binding (describes the communication method that callers use)
 - Wire (an export can have one wire to the exposed service only)



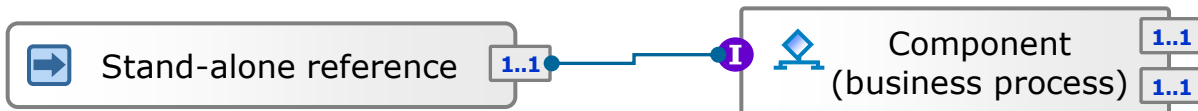
SCA module components: Imports

- Imports allow components to call services outside the module
 - Component references can be wired to imports or other components
- Imports separate the infrastructure details required to communicate with the external service from the service implementation
- Import definitions include:
 - Name
 - 1...N interfaces
 - Binding (describes the communication method that is used to call the external service)



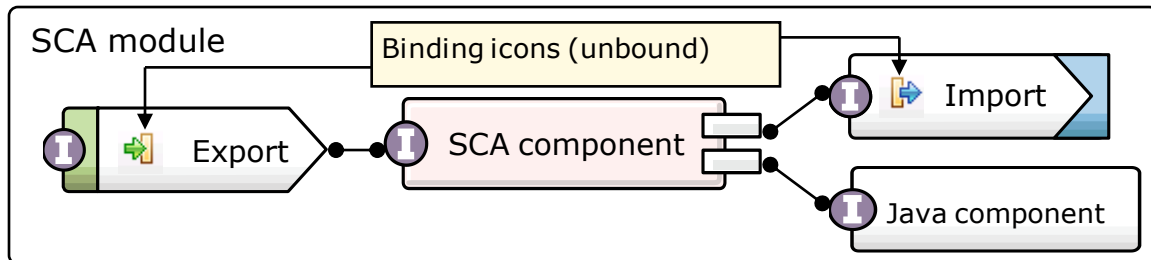
SCA module components: Stand-alone references

- A stand-alone reference:
 - Is defined in an **sca.references** file
 - A non-SCA component within the module uses it to invoke an SCA component interface; for example, a JavaServer Page (JSP)
- Stand-alone references are defined like inline references by using:
 - Name
 - Multiplicity
 - Interface
 - Wire



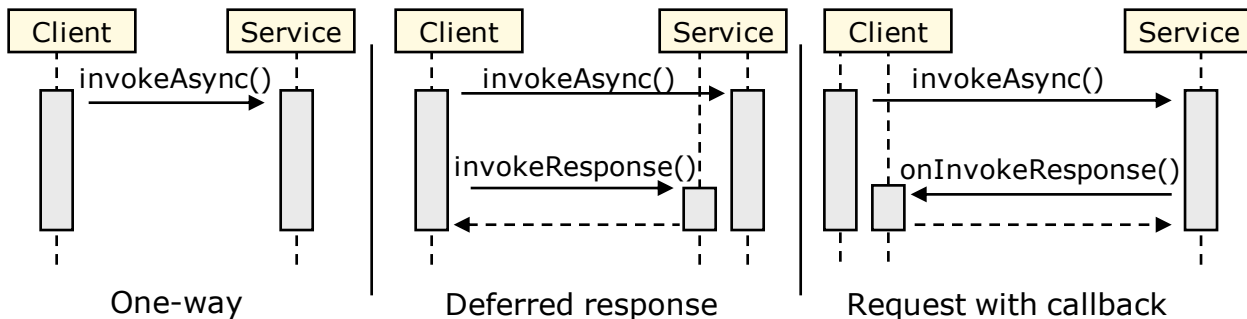
Incoming and outgoing interactions with external components

- Exports process **incoming** requests from outside SCA modules
 - The reference of the export is associated with a specific interface type
 - The export is connected to an SCA component through a wire
- Imports process **outgoing** requests to components outside SCA modules
 - The import contains a specific interface type
 - The interface of the import is the target of a reference through a wire
- Bindings determine how imports and exports interact with components outside a module
 - Bindings specify the means of transporting the data (protocol)



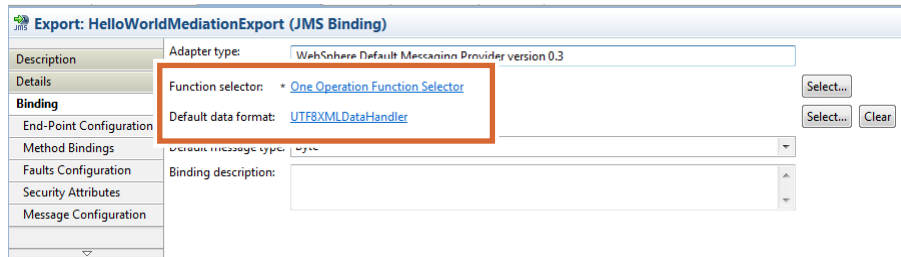
SCA invocation styles

- SCA components might call services synchronously or asynchronously, depending upon the preferred interaction style
 - The **synchronous** method is `invoke`
 - The **asynchronous** method is `invokeAsync`
- A synchronous call waits for the return value before proceeding
- There are three kinds of asynchronous invocations:
 - One-way**: no response is expected (“fire and forget”)
 - Deferred response**: caller fetches the response later by using a ticket
 - Request with callback**: callee sends the response back to the caller when the result becomes available



Import and export resources

- Most imports and exports use the following components:
 - **Data binding** is a map between a native data format and a business object
 - The `DataBinding` Java class takes a stream of data and builds a business object or takes a business object and builds a stream of data
 - **Function selectors** assign incoming messages or requests to the correct service operation
 - **Data handlers** are used by data bindings or function selectors to transform data from one type to another
- The generated resources are normally adequate
 - For the remaining cases, import and export binding resources can be customized, depending on the binding type



Export binding types

- When an export is created in a mediation module or integration module, it must be bound to a transport type
- Transport types include:
 - Service Component Architecture (default)
 - Web service
 - Hypertext Transfer Protocol (HTTP)
 - EJB (stateless session bean)
 - EIS (Java Connector Architecture adapter)
 - Java Message Service (WebSphere Application Server or WebSphere Enterprise Service Bus)
 - Generic Java Message Service (for independent vendor JMS providers)
 - WebSphere MQ
 - WebSphere MQ JMS
- Several export components with different binding types can be exposed and used by many callers

Import binding types

- When an import is created in a mediation module or integration module, it must be bound to a particular transport
- Transport types include:
 - EIS (Java Connector Architecture adapter)
 - SCA
 - WebSphere MQ
 - EJB (stateless session bean)
 - Java Message Service (WebSphere Application Server or WebSphere Enterprise Service Bus)
 - Generic Java Message Service (for independent vendor JMS providers)
 - Web service
 - Hypertext Transfer Protocol (HTTP)
 - WebSphere MQ JMS
- Modules can also contain several import components with different binding types so they can call various services

Web Services



Web Services Overview

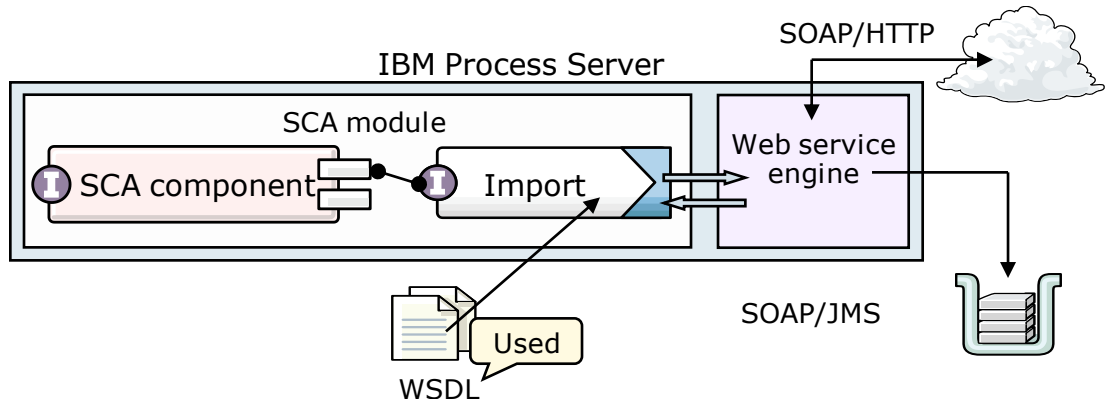
- Web services connect businesses to each other and invoke services with appropriate security, reliability, and confidentiality
- If XML defines a platform-independent standard way to represent data, then web services define a platform-independent exchange for data
 - Application integration becomes easier
 - Web services use core technologies: XML, WSDL, and SOAP
- XML (Extensible Markup Language)
 - XML solves the problem of data independence
 - Use XML to describe data and to map that data into and out of any application
- WSDL (Web Services Description Language)
 - XML-based language to create a description of an underlying application
 - The description turns an application into a web service by acting as the interface between the underlying application and other web-enabled applications
- SOAP
 - SOAP is the core communications protocol for the web
 - Most web services use this protocol to communicate with each other
 - SOAP can be used over HTTP or JMS

Discovering web services

- You typically discover service descriptions through a Universal Description, Discovery, and Integration (UDDI) registry or through WebSphere Service Registry and Repository
- WebSphere Service Registry and Repository allows you to store, access, and manage information about services
 - You can use this information to select, invoke, and reuse services
- You can use WebSphere Service Registry and Repository to store information about services in your systems or in other systems that you already use, that you plan to use, or of which you want to be aware
 - An application can check WebSphere Service Registry and Repository before it invokes a service to locate the most appropriate service that satisfies its functional and performance needs
 - This capability helps make your deployment more dynamic and more adaptable to changing business conditions
 - You can access WebSphere Service Registry and Repository through mediation flows
 - You can use the external service wizard with WebSphere Service Registry and Repository

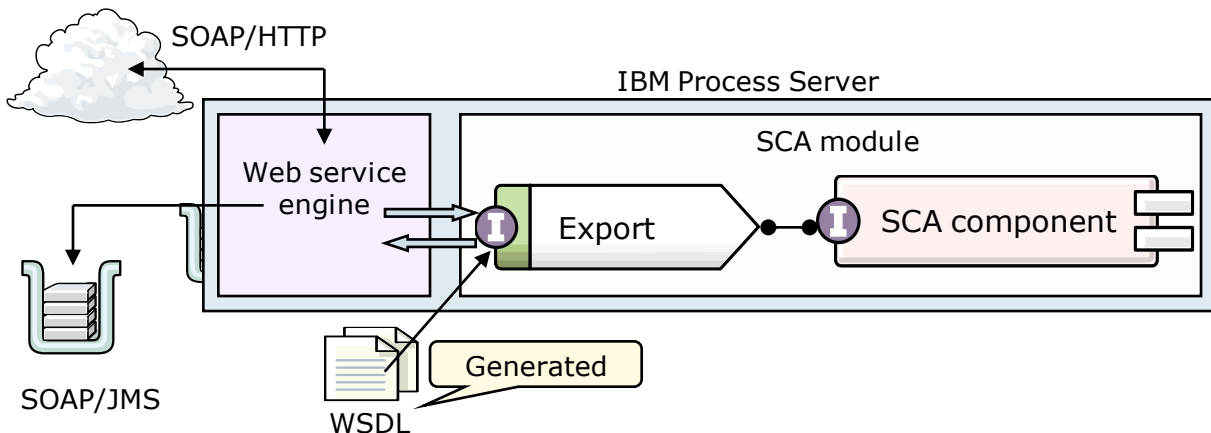
Importing a web service WSDL interface

- When you discover and import a web service WSDL interface and drag it onto an assembly diagram in IBM Integration Designer, a web service bound import is created to call the service
- The component then refers to a web service endpoint
 - This endpoint is the location at which the web service is listening for incoming requests
- An import with a web service binding results in a web service call to an external partner that uses SOAP over HTTP or SOAP over JMS



Web service export binding

- Exports with web service binding declare that the module exposes an interface that can be invoked remotely as a web service
- The export uses SOAP over HTTP or SOAP over JMS as the transport protocol
- When a web service export is declared, an associated WSDL file is generated



Interfaces, Business Objects and Data Objects

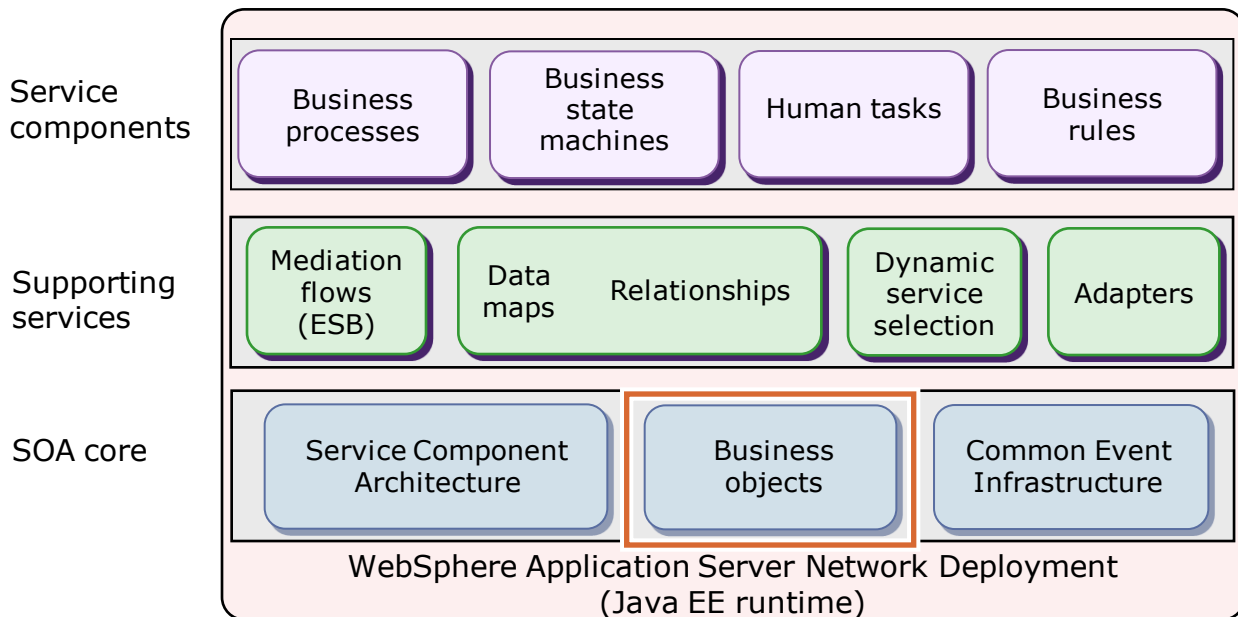


Web Services Description Language Interfaces

- Web Services Description Language (WSDL) provides an industry standard way of describing services in a service-oriented architecture
- WSDL is a way for service providers to describe the basic format of requests to their systems, regardless of the underlying implementation
- WSDL allows a provider to specify the characteristics of a service:
 - The name of the service and addressing information
 - The protocol and encoding style that is used to access the public operations
 - The operations, parameters, and data types in the service interface
 - The preferred interaction style: synchronous or asynchronous
- WSDL documents are defined in XML
 - An industry standard language
 - Platform and technology independent
 - Capable of describing a wide range of services; service definitions are flexible and extensible
- Interface elements can be structured in one or more WSDL files

Business objects

- Business objects are an SOA core component
- Business objects provide an abstraction layer for data objects



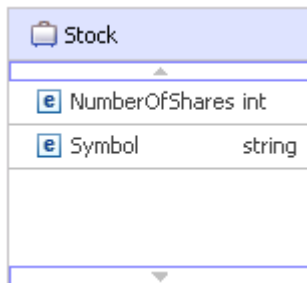
Business objects (1 of 3)

- Business objects are the primary data structure for business data and data types that are defined in WSDL (interface) definitions
- Business objects are modeled by using XML schemas (XSD)
 - Can import business object schema definitions from, or export to, other systems
 - Support for the full XML schema data type system
- At run time, `commonj.sdo.DataObject` is used to represent business objects in memory as an SDO instance
 - Created from XSD files by using the business object factory
 - Accessible using the SDO API and XPath
- Support is provided for data object schemas from industry standards organizations
 - HL7, ACORD, OAGIS
 - IBM Business Process Manager Industry Packs provide prebuilt data objects

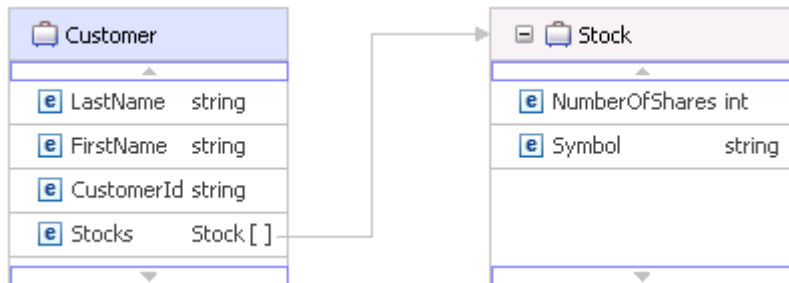
Business objects (2 of 3)

- Business objects are collections of elements with names and data types
- There are two types of business objects:

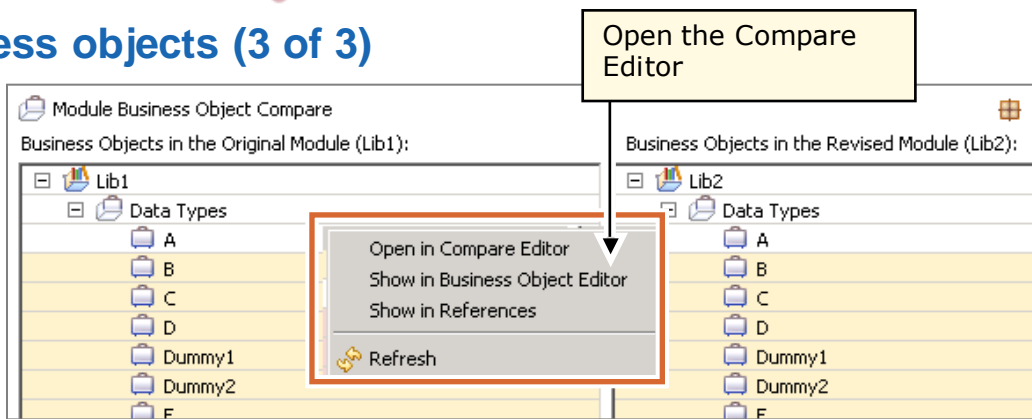
- Simple business objects that are composed of scalar (single-value) elements



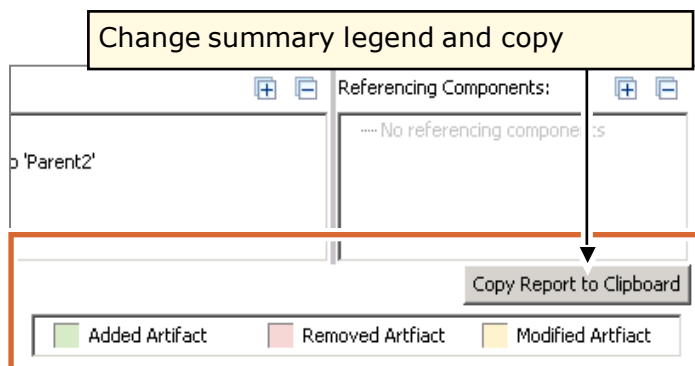
- Hierarchical business objects with elements that contain other (child) business objects



Business objects (3 of 3)

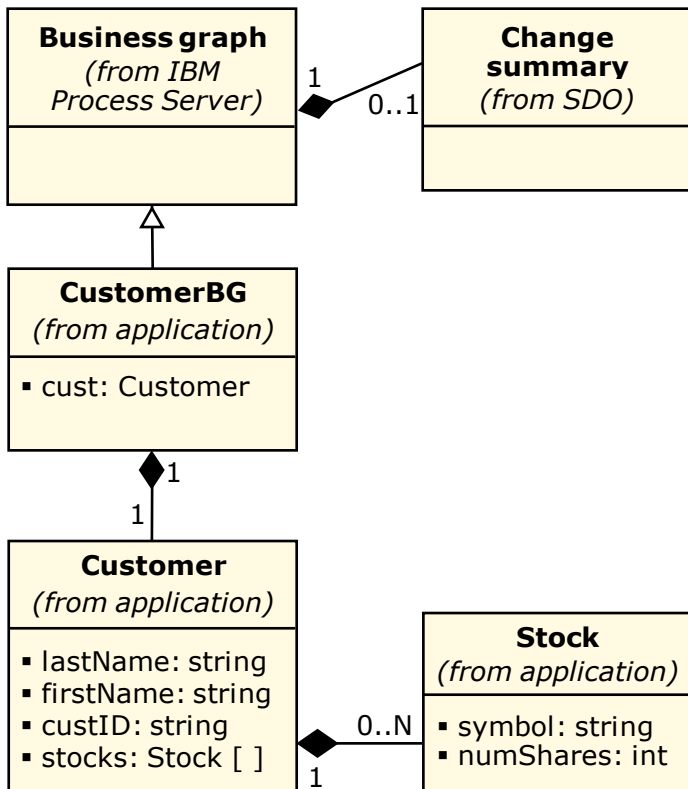


- Facilitates working with large business objects
- Test environment allows you to import from XML data file, create reusable data pools
- Business object compare feature allows you to compare business objects in different modules or libraries



Business graphs

- A business graph is an optional container around a business object or hierarchy of business objects
- Business graphs are SDO data graphs with extensions
- Business graphs include:
 - Root and associated business objects (in a tree)
 - A change summary header
- Graphs are seldom used outside the adapters (some adapters require graphs for the change summary)



Business graph: Change summary

- The change summary is used to record changes to contained business objects
- Change summaries support disconnected data patterns and command-event models
 - Disconnected data pattern:** records changes to business objects while disconnected from the data source
 - Command event model:** allows an EIS to capture and publish data changes for use by other systems that need to be aware of these events
- There are two types of change summary usage:
 - Implicit:** change logging is turned on, and any changes to contained BOs are logged
 - Explicit:** modify the change summary explicitly (IBM extension)

[-] [icon]	CustomerBG
[e]	verb string
[-] [icon]	Customer
[e]	changeSummary
[e]	eventSummary
[e]	LastName string
[e]	FirstName string
[e]	CustomerId string
[+] [e]	Stocks Stock []

Introduction to Service Data Objects (SDO)

- SDO provides a framework for data manipulation
 - SDO API unifies representation of data from multiple sources
 - No need to know multiple technology-specific APIs
- SDO framework supports a disconnected programming model (manipulate data without connection to source)
- SDO is integrated with XML
- The key components of the SDO framework are: data objects, data graphs, and data object metadata

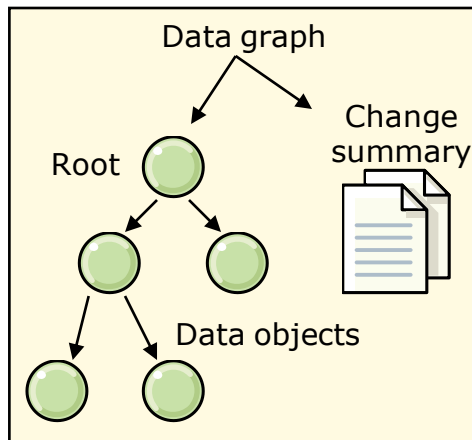
Component	Description
Data object	Fundamental data structure for representing business data
Data graph	A container for a hierarchical set (tree) of data objects
Data object metadata	Metadata is the schema definition of the object; it contains information about the data in the data object (property types, relationships, constraints)

Data objects

- Data objects are the fundamental structures for representing business data
- A data object holds data as a set of properties
- Each data object provides read and write access to properties through:
 - Getters and setters
 - XPath (XML Path Language)
- Properties can be:
 - Primitive data types (such as strings)
 - Commonly used data types (such as dates)
 - Multivalued fields (such as arrays)
 - Other data objects
- In memory, data objects are represented as instances of `commonj.sdo.DataObject`
 - Objects are serialized to XML

Data graphs

- Data graphs are an optional wrapper around a root data object and associated data objects (in a tree structure)
 - Can include data objects from different data sources
- Data graphs include a change summary that records modifications to the data tree
 - Object changes: reference to the object whose properties were changed, the property that changed, the new value, and the old value
 - Object creations: data objects that are added to the data graph
 - Object deletions: data objects that are removed from the data graph

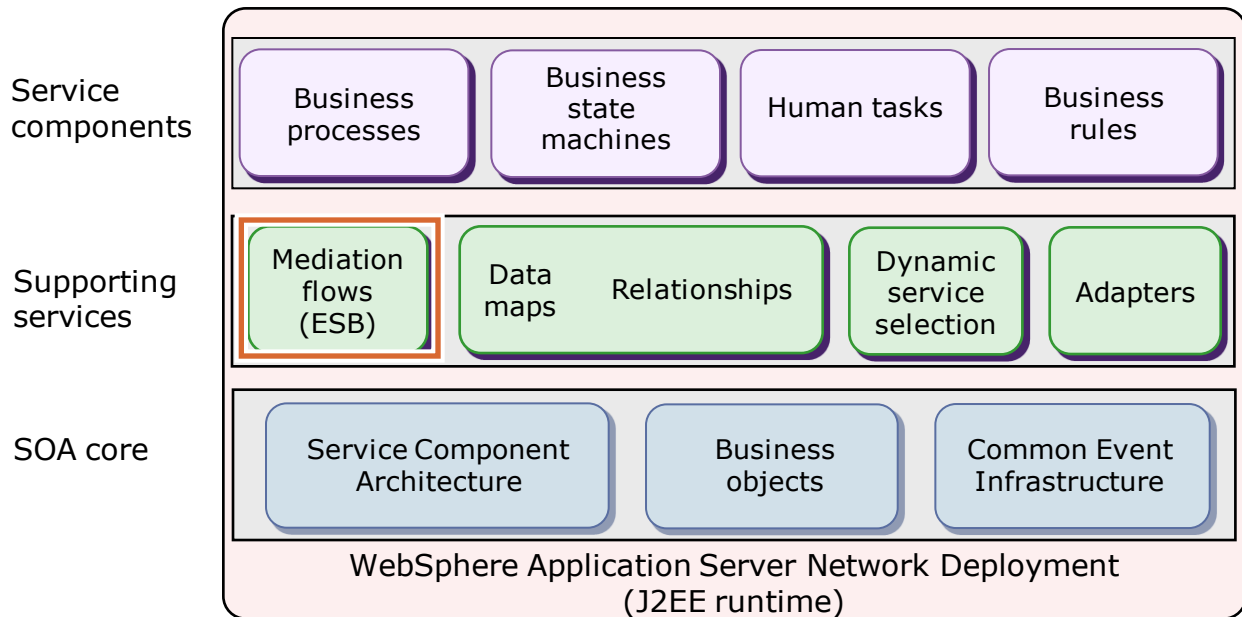


Mediation Modules and Primitives



Mediation flows are supporting services

- Mediation modules can be deployed to IBM Process Server or WebSphere Enterprise Service Bus
- There is no equivalent support in IBM Process Designer



Mediations: Key concepts

- Mediation module
 - Special type of SCA module
 - Mediate messages that flow between service requesters and providers
- Mediation flow component
 - Contains the mediation flow logic
 - Unique flow logic for every interface operation
 - Modules can contain zero or multiple mediation flow components
- Mediation primitives
 - Used to construct the logic of a mediation flow
 - Each primitive performs a specific part of the flow logic
 - Encapsulated unit of logic that manipulates the message as it passes through the enterprise service bus
- Service message object (SMO)
 - Internal representation of message body and headers
 - Mediation primitives act upon the SMO within the mediation flow

IBM Integration Designer: Typical task flow

1. Identify the service endpoints that need to be integrated

- Service requesters and service providers



2. Assert the connectivity between these endpoints

- Link service requester to service provider



3. Build the **mediation flow necessary to allow the endpoints to communicate effectively**

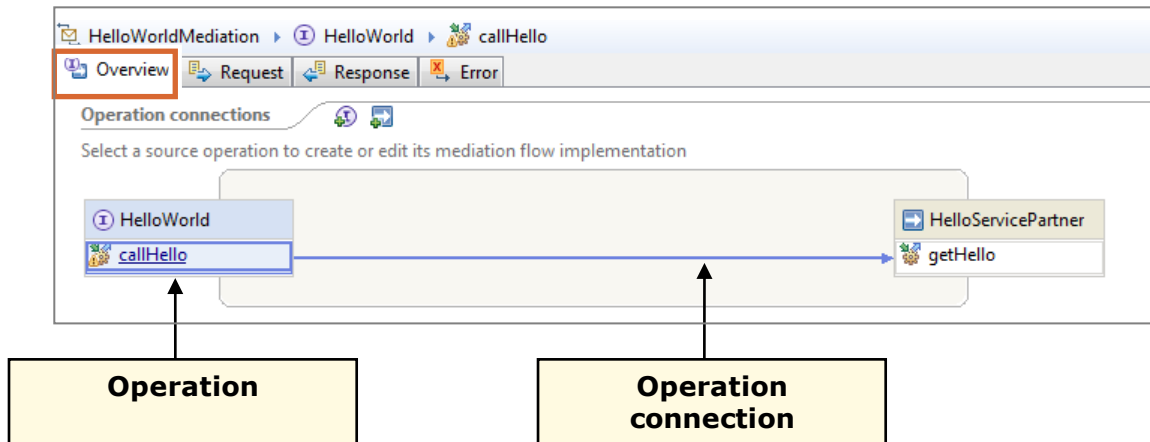
- Build a mediation flow from supplied mediation primitives (and custom-written mediations, if needed)
- Configure the selected primitives



4. Test and debug the mediation flow

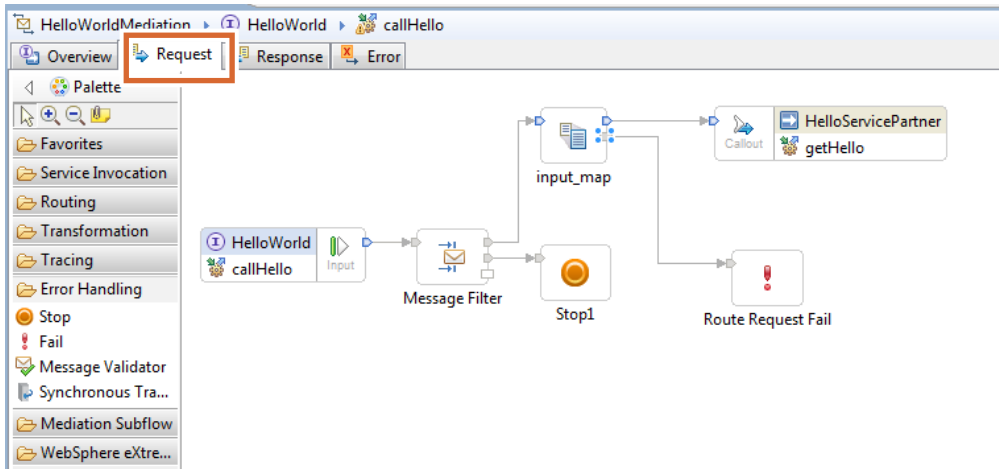
- Examine the message contents
- Determine the message flow that was taken

Mediation flow editor: Operation connections



- Mediation flow editor has two main sections:
- Operation connections section (the “Overview” tab)
 - Mappings between the corresponding operations of interfaces and references are displayed
- Request mediation flow
 - If mediation is a two-way operation, request and response flows are shown

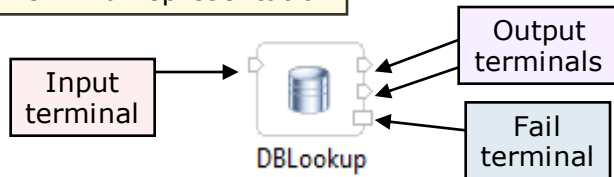
Mediation flow editor: Flow view



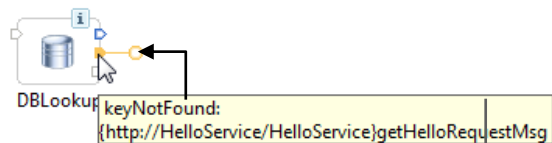
Mediation primitives from palette are wired together in the drawing canvas (request and response views)

Terminals in the mediation flow editor

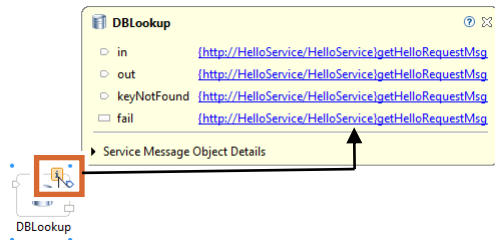
Terminal representation



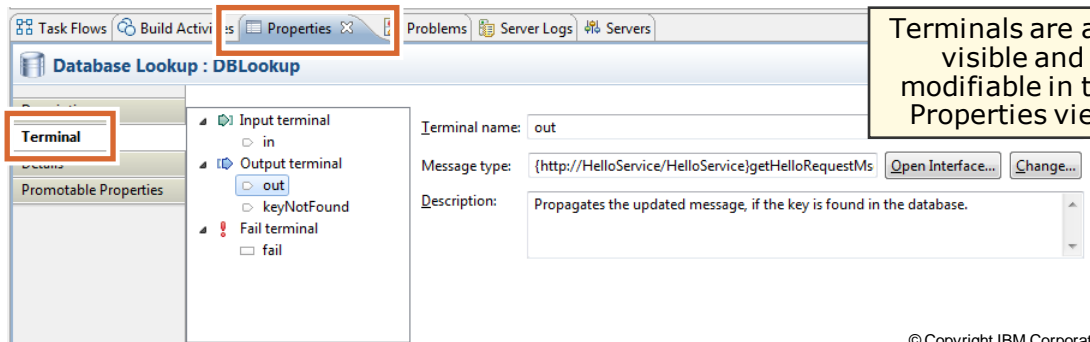
Hovering over terminal display name, type



Click information icon to display all terminals and their types; click individual link for detail

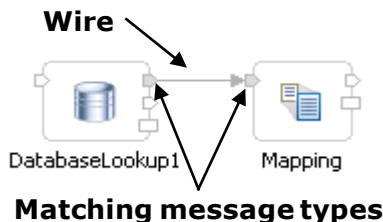


Terminals are also visible and modifiable in the Properties view

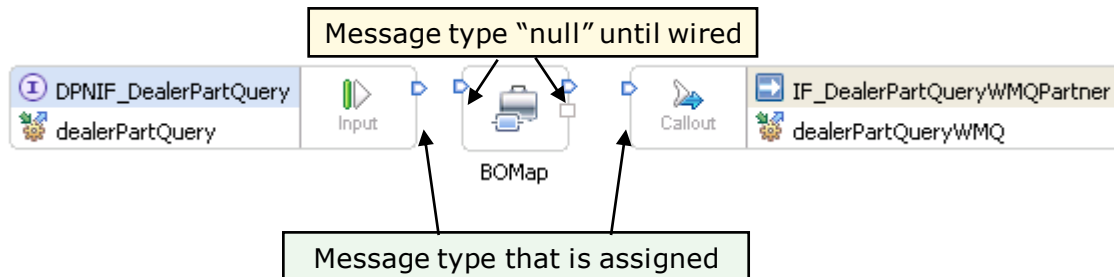


Wiring of mediation primitive terminals

- Connections between terminals are represented with wires
- A connection must have matching terminal message types



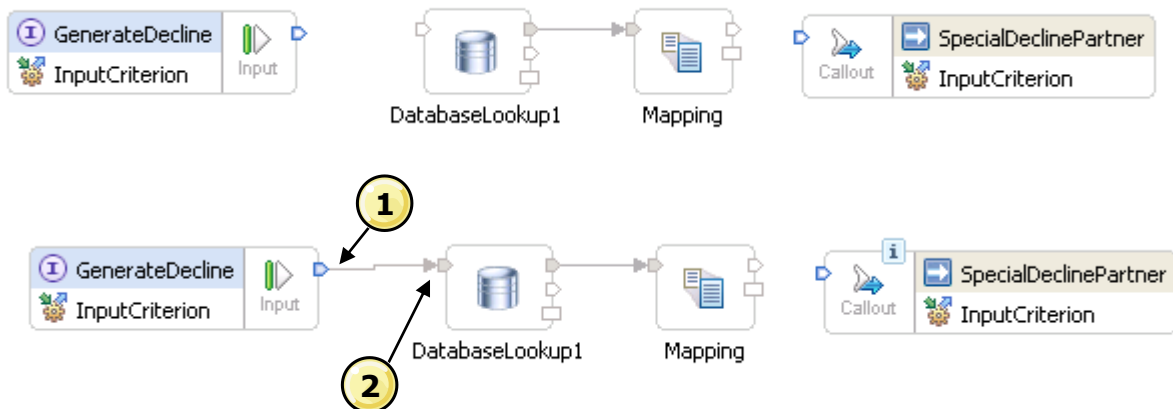
- Editor dynamically manages terminal message types



Example of wiring of mediation primitive terminals

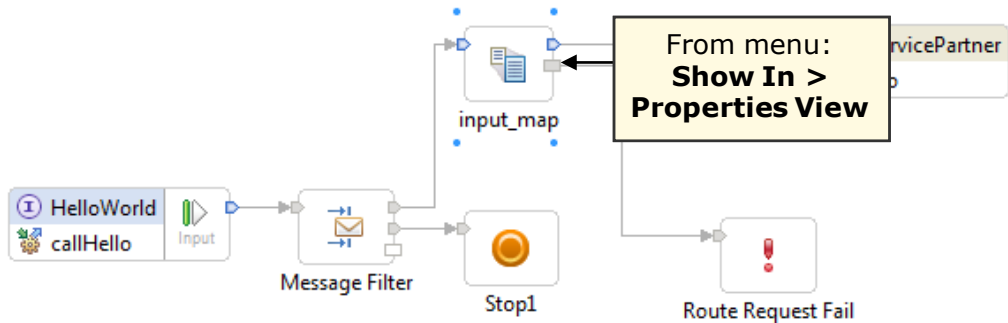
- Example of wiring and message type handling

Wiring "null" to "null" type keeps type as "null"

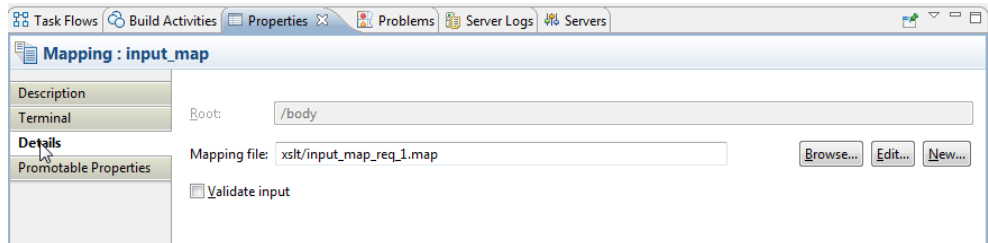


Wiring from (1) to (2) assigns the type of terminal (1) to terminal (2)

Mediation flow editor: Properties view



Properties view



Mediation subflow

- Facilitates reuse of mediation logic without rewriting
- Composed of:
 - Mediation primitives that are wired together
 - Start with an **in** primitive and end with an **out** primitive; act like input and callout nodes of a mediation flow
- Used by a parent as if it were a primitive:
 - After a subflow is developed, it is dropped onto the canvas from the primitive as if it were a built-in primitive
 - Wired into the calling flow like any other primitive
- Promoted properties of a subflow are settable by the caller
- Subflows can be nested
- Can be contained in a library or module (library increases reusability)



Promoted properties: Overview

- “Promoting” the properties of a mediation primitive allows them to be changed at run time

The screenshot shows the IBM Expert Access Services console with the 'Properties' tab selected. The main window displays the configuration for the 'XSL Transformation : DealerETransform' primitive. On the left, a sidebar contains links for 'Description', 'Terminal', 'Details', and 'Promotable Properties'. The 'Promotable Properties' section is active, showing a table of properties that can be promoted for runtime changes.

Filter:

Property	Promoted	Group	Alias	Alias value	Description
Root	<input type="checkbox"/>				
Mapping file	<input type="checkbox"/>				
Validate input	<input type="checkbox"/>				

- Properties are changed at run time by using the administrative console; no server restart or redeployment of the mediation is required
- Not all primitive properties can be promoted

Promoted properties: Administrative management

Cell=xpbaseNode01 Cell, Profile=ProcSrv01 [Close page](#)

SCA modules

[SCA modules](#) > [RouterMediationService](#) > **Properties**

The properties that are set for this module.

Configuration

General Properties

RouterMediationService.RouteRequest

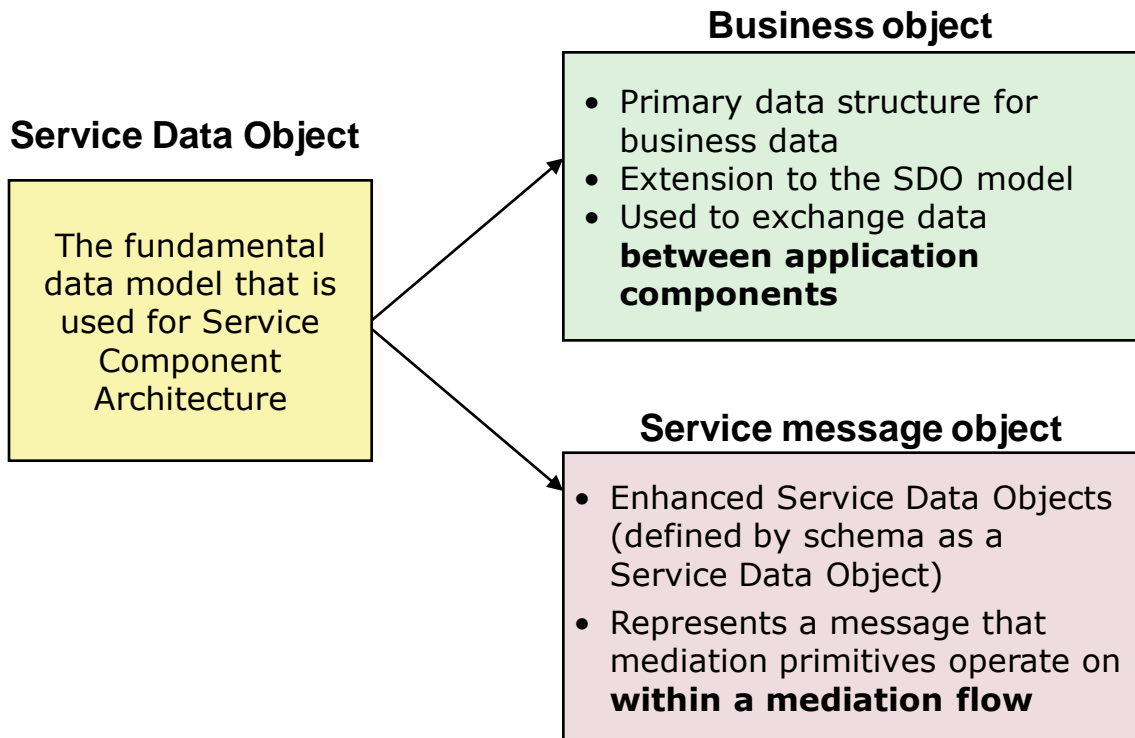
property group

property alias names

Name	Type	Value
Callout1.retryDelay1	INTEGER	0
Callout1.retryOn	STRING	0
Callout1.retryCount1	INTEGER	0
Callout1.retryCount	INTEGER	0

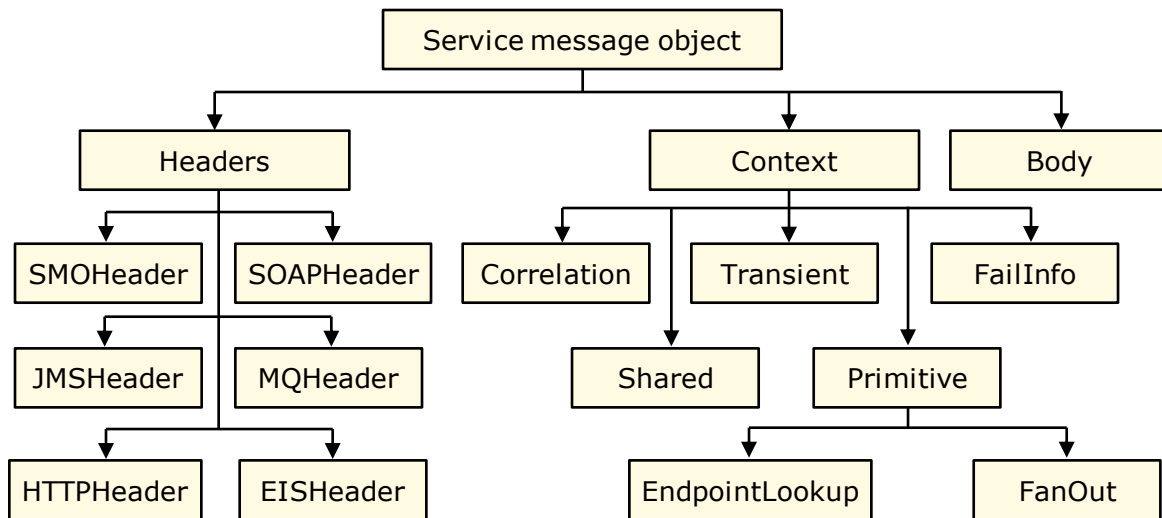
Administrator manages promoted properties from the Integrated Solutions Console

Service message object (SMO) Overview



Service message object (SMO)

- At top level, SMO is composed of headers, context, and body
 - Header is information about transport protocol that is used to send message
 - Context is other data specific to logic of flow as data passed between primitives (such as failure information)
 - Body (optional) is application data (payload) of the message that contains input or output values of the operation



SMO structure: Headers

The headers might include:

- **SMOHeader**: information about the message (for example, message ID, SMO version)
 - SMO header **always** present
- **JMSHeader**: contains a JMS header
 - Used when there is a JMS import or export binding
- **SOAPHeader**: contains SOAP header information
 - Used when there is a web services import or export binding
- **SOAPFaultInfo**: contains information about SOAP faults
- **properties[]**: arbitrary list of name-value pairs (for example, JMS user properties)
- **MQHeader**: contains md (MQMD), control (format information of the message body), and other headers (like RFH and RFH2)
- **HTTPHeader**: contains HTTP headers, when there is an HTTP import or export binding
- **EISHeader**: present when the email, flat file, or FTP adapters are used
 - Different headers exist for each adapter
- **WSAHeader**: used when export is configured with JAX-WS binding to provide WS-Addressing support

[-]	[e] headers
[+]	[e] SMOHeader
[+]	[e] JMSHeader
[+]	[e] SOAPHeader
[+]	[e] SOAPFaultInfo
[+]	[e] properties
[+]	[e] MQHeader
[+]	[e] HTTPHeader
[+]	[e] EISHeader
[+]	[e] WSAHeader

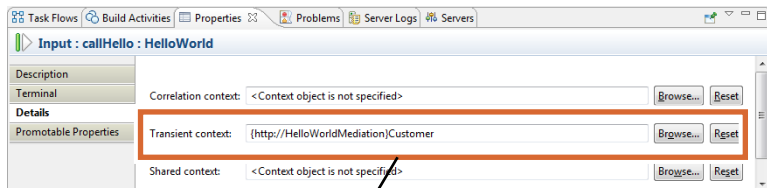
SMO structure: Correlation and transient contexts

- The context includes the **correlation** and **transient** contexts

- Both of them are:

- Used to pass application data between mediation primitives
- An XSD defined data object
- Specified on the input node properties of the mediation flow

input node properties

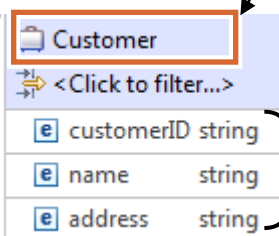


- Correlation** context maintains data across a request/response flow

- Transient** context:

- Exists while the flow is running only
- Maintains data only during one direction (request or response), but same data object definition that is used for both the request and response

business object



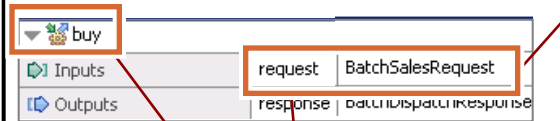
context in SMO

Name	Type
context	ContextType
shared	Shared
transient	Customer
customerID	String
name	String
address	String

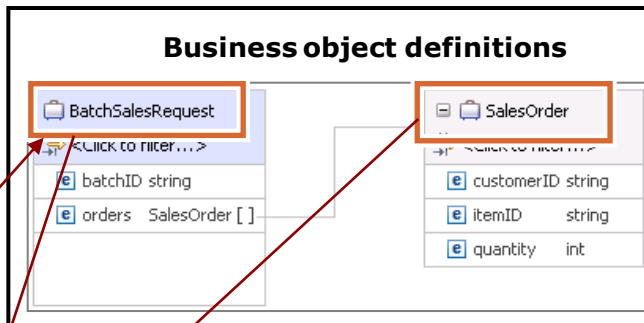
SMO structure: Body

- The body contains the payload of the message
 - Payload is the application data that flows in the message
 - It identifies the operation and either its inputs, outputs, or faults
- Operation is defined in WSDL by using the interface editor

Operation definition



Business object definitions

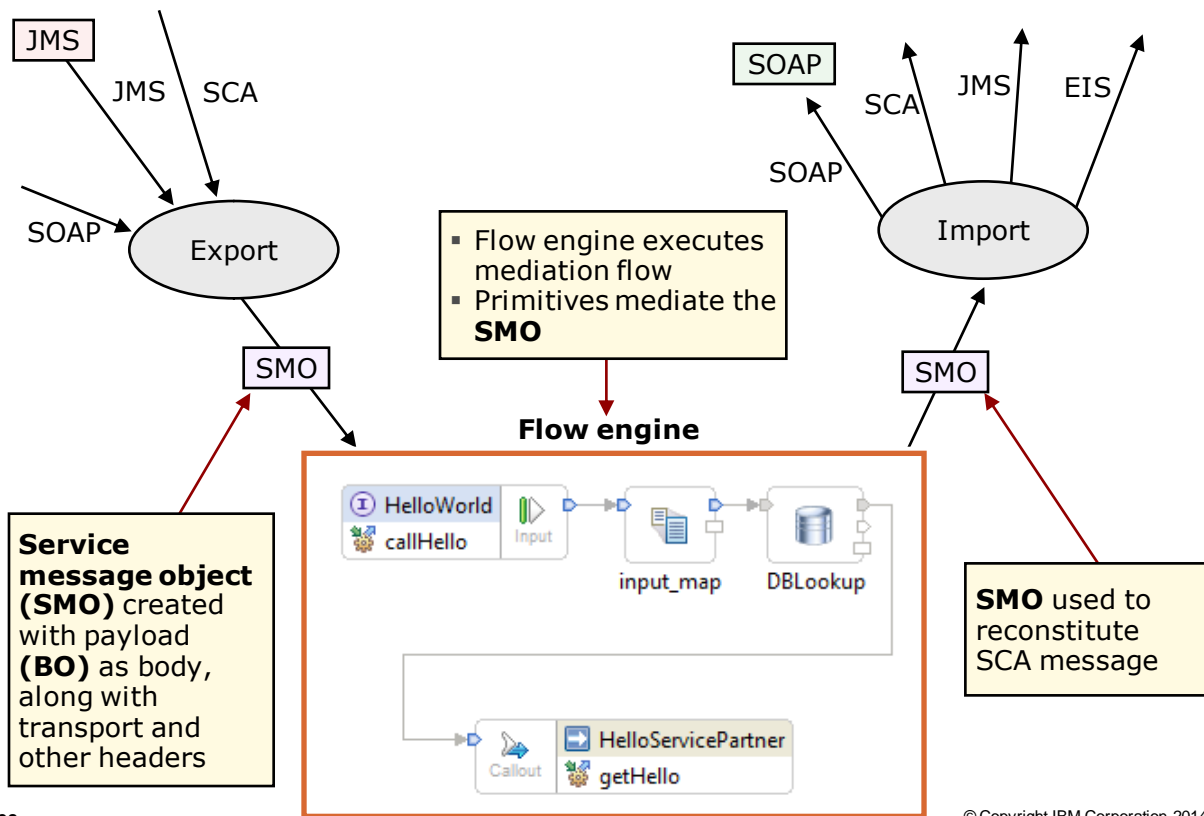


Body in SMO

Name	Type	Value
body	buyRequestMsg	[ab]
buy	buy_type	[ab]
request	BatchSalesRequest	[ab]
batchID	String	[ab]
orders	SalesOrder[] <SalesOrder>	[ab]
orders[0]	SalesOrder	[ab]
itemID	String	[ab] 1234

- Inputs, outputs, and faults can be simple types or XSD defined types
 - XSD defined types are created by using the business object editor

SMO in a mediation flow



Manipulating SMOs

- Three ways to access and manipulate SMOs: XPath, XSL, Java
- **XPath expressions**
 - Primary mechanism for accessing the SMO
 - Used in some form by all of the mediation primitives
 - Identify elements to read or update conditional expressions
- **XSL style sheets**
 - Used by the mapping mediation primitive
 - Normally used to modify SMO type within a flow
 - Can also be used to manipulate SMO content without changing type
- **Java code**
 - Used by the custom mediation primitive
 - SMO accessed through Generic Service Data Object APIs (loosely typed), or the SMO APIs (“strongly typed”)
 - Can access and update content; can also modify SMO type

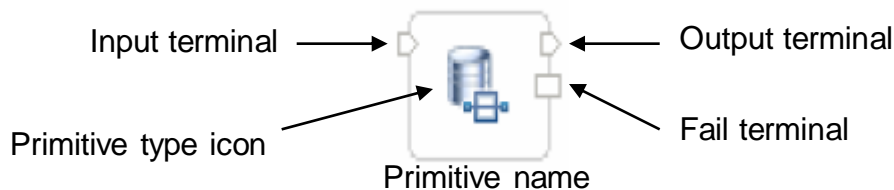
Manipulating SMOs

- A set of Java services is available to allow business objects to be created and manipulated
- These services are part of the `com.ibm.websphere.bo` package
- Classes include:
 - **BOFactory**: create instances of business objects
 - **BOXMLSerializer**: converts to a business object from a stream, or writes the content of a business object, in XML format, to a stream
 - **BOCopy**: methods that make copies of business objects (“deep” and “shallow” semantics)
 - **BODataObject**: allow access to the data object aspects of a business object, such as the change summary, the business graph, and the event summary
 - **BOXMLDocument**: allows the manipulation of the business object as an XML document
 - **BOChangeSummary** and **BOEventSummary**: to access and manipulate the change summary and event summary portion of a business object
 - **BOEquality**: enables you to determine whether two business objects contain the same information, both “deep” and “shallow” equality
 - **BOType** and **BOTypeMetaData**: materialize instances of the `commonj.sdo` type; allows you to manipulate the associated metadata
 - **BOInstanceValidator**: validates the data in a business object to view if it conforms to the XSD

What is a mediation primitive?

- An encapsulated unit of logic that manipulates the message as it passes through the enterprise service bus
 - Mediation primitives accept and process messages to allow you to change format, content, or target service provider
 - Contained within a mediation flow
- WebSphere Enterprise Service Bus provides variety of predefined mediation primitives
 - Allows you to implement mediation logic
 - In most cases without the need to write code
- Custom primitives can be constructed
 - Also available from third-party sources

Common features of mediation primitives



- Most mediation primitives share common attributes:
 - A default name, which is assigned when the primitive is moved to the drawing canvas
 - An input terminal, where the message is received
 - An output terminal, where the message is sent after the primitive successfully performs its actions
 - A fail terminal, where the original message and error information are sent if an exception occurs while the primitive is processing
- Some primitives have fewer or more terminals
- Some primitives have associated configurable properties

Message Transformation and enrichment primitives



Mediation primitives: Transformation (1 of 2)

Mediation primitive	Description
Business object map	<ul style="list-style-type: none"> ▪ Allows graphical creation of message transformations by reusable business object map ▪ Use to change message content or message type
Custom mediation	Allows you to use Java code to implement custom mediation logic
Data handler	Converts an element of a message between a physical format and a logical structure
Database lookup	Modifies the content of a message by reading data from a user-supplied database

Mediation primitives: Transformation (2 of 2)

Mediation primitive	Description
Message element setter	Modifies the content (but not the type) of a message by adding, deleting, or changing message elements
Message header setters (four types)	<ul style="list-style-type: none"> Creates, modifies, copies, or deletes message transport headers Specific header setter primitives exist for HTTP, JMS, WebSphere MQ, and SOAP messages
Set message type	Overlays message fields with more detailed structures to allow easier manipulation of message content; allows you to treat weakly typed fields as strongly typed
Mapping	Allows you to modify the content of a message by using Extensible Style sheet Language (XSL) transformations or business object map transformations

Mapping mediation primitive

- Allows you to manipulate messages by using an XSLT transformation or business object maps
- Override support for mapping complex structures
- XSLT V2.0 and XPath V2.0 support for XSL transformation or a business object map transformation
- Can use a graphical editor to change the headers, context, or body of the SMO by mapping between the input and output message
- The XSL transformations operate on an XML serialization of the message, whereas the Business Object Map transformation operates on the Service Data Objects (SDO)

Message element setter primitive

- Modifies the body of an SMO by specifying:
- The element that needs to be modified (specified as XPath expression)
- The operation that you want to perform on the target:
 - **Set**: assigns a constant value to the target element
 - **Copy**: copies a source element to a target element
 - **Append**: copies from a source element to a new element instance, by appending to a repeating element in the output (can only append to a repeating element in the output)
 - **Delete**: deletes an element instance
- The value that is going to be used, if a **copy** or **append** operation is selected (ensure that types are compatible, or runtime exception occurs)
- **Validate input** property: If set to “true,” and the input message is invalid (does not match its schema), a runtime exception occurs



Message Element Setter

Action:	Set	
Target:	<Type an XPath expression here. Content assist available (Ctrl+space)>	Browse...
Type:		Browse...
Value:		

You cannot change the **type** of a message by using the message element setter, only the **body** of the message

Message header setter primitives

- Four unique message header setter primitives:
 - HTTP header setter
 - JMS header setter
 - WebSphere MQ header setter
 - SOAP header setter
- Modifies the transport protocol header of an SMO message by specifying:
 - Type (“mode”) of operation you want to perform (create, copy, modify, or delete)
 - The name of the header to be modified
 - The value that is used if a create, copy, or modify operation is requested
 - A flag that indicates whether the value is an XPath expression or a literal
- Validate input** property: If set to “true,” and the input message is invalid (does not match its schema), a runtime exception occurs



Mode:	Create
Header Name:*	<Select the name of a standard Header or enter your own>
Type:	String
<input type="checkbox"/> Set Value using XPath	
Value:	

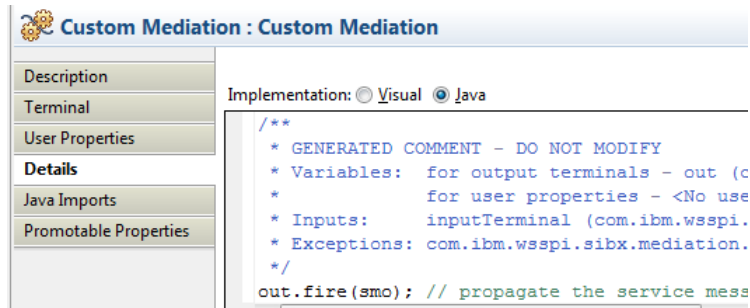
You cannot change the **body** of a message that uses the message header setter primitives, only the specified transport protocol headers

Custom mediation primitive

- Allows you to implement custom mediation logic in mediation flows by using Java snippets or visual snippets
- Useful when no built-in primitive provides the needed function
- Can define more input and output terminals and properties on the primitive at development time for increased flexibility
- Must explicitly send the message through the output terminal by using the `out.<terminal name>` notation
- Need to explicitly define the message types on the terminals
- When custom mediation primitive is created on the canvas, initial “skeleton” code is created for you



Custom Mediation



Flow Control Primitives



Mediation primitives: Routing (1 of 2)

- These primitives control service invocations inside the mediation flow and message routing outside the mediation flow

Mediation primitive	Description
Message filter	Routes messages within a mediation flow that is based on the message content
Service invoke	Calls a service from within a mediation flow (instead of waiting until the end of the mediation flow and by using the callout mechanism)
Type filter	Routes messages within a mediation flow that is based on the message type

Mediation primitives: Routing (2 of 2)

- These primitives control routing within the mediation flow

Mediation primitive	Description
Fan in	Aggregates multiple messages (created by a fan out primitive) based on a decision point, then outputs a single message
Fan out	Splits a message that contains repeating elements into multiple messages, or sends the same message more than once to implement message aggregation
Flow order	Specifies the order in which branches of a flow run

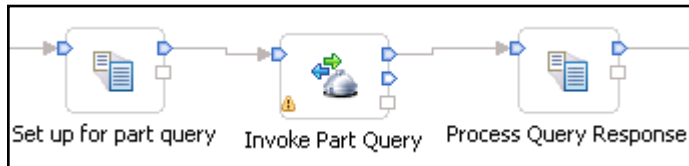
Message filter primitive



- Routes a message within a mediation flow that is based on the message content
- Acts like an “if . . . goto” statement to send incoming message through one or more paths if filter criteria met
- Any number of output terminals can be defined
- Filter criteria are XPath expressions in a table
- If no match, then message is propagated through **default** terminal
- Usage:
 - Verify that the contents of a message meets conditions, such as required field present; If conditions not met, invoke error handler subflow or fail primitive to stop processing
 - Route the message based on its contents (value in field indicates service provider A invoked, other values indicate service provider B invoked)
 - Conditionally bypass steps in a mediation based on message content

Service invoke primitive

- Like callout node, but can be used anywhere in request flow
- Sends message to (invokes) requested service and operation
- Has **in** and **out** terminals for inbound and outbound message
- Has **timeout** terminal that is fired if timeout threshold is exceeded, like callout node
- Has a **<fault message name>** terminal for each modeled fault; Propagate the modeled message that is returned from the invocation when a modeled fault occurs



Sample flow that contains service invoke

Type filter primitive

- Routes a message within a mediation flow based on the message type
- Acts like an “if . . . goto” statement to send message through down a specific path if filter criterion is met
- Any number of output terminals can be defined at development time
- Filter criteria are XPath expressions that you enter in a table at development time
- If no match occurs, message is propagated through *default* terminal



Faults, tracing and error handling primitives



Mediation primitives: Error handling, debugging, and event recording

Mediation primitive	Description
Event emitter	Sends a monitoring event (in Common Base Event format) from within a mediation flow to a Common Event Infrastructure server
Fail	Stops processing in a mediation flow and generates an exception
In	Used as an entry point in a subflow
Out	Used as an exit point in a subflow
Message logger	Stores messages in a relational database or other medium
Message validator	Validates all or part of a message against its schema
Stop	Stops processing in a mediation flow, without generating an exception
Trace	Writes trace messages to server log or log files

Event emitter primitive

- Emits event to the CEI server
- Event is a business event that can cause mediation flow failure or unusual branch
- Events are in Common Base Event format
- Can include or exclude message in event
- Usage:
 - Use the event emitter primitive to emit a business event to CEI
 - Typically used for “unusual” business events
 - Use to record a failure in a node but continue processing; wire to fail terminal of another primitive, and then wire event emitter to next primitive
 - Can run in its own transaction or within the transaction of the mediation flow



Event Emitter

Message logger primitive

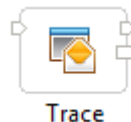
- Stores message in relational database table
- Can also write message to another medium by using custom log facility
- Saves selectable amount of message content plus identifying information (timestamp, message identifier, primitive, and module names, and other information)
- Messages can be logged for later review, post-processing, auditing, or whatever purposes you require
- Original inbound message passes through without alteration



Message Logger

Trace primitive

- Writes a message to the server log, user trace log file, or other file
- Can be used for debugging or tracing a mediation flow
- Saves selectable amount of message content plus identifying information (timestamp, message identifier, primitive and module names, and other information)
- Original inbound message passes through without alteration



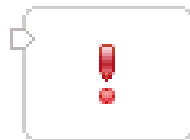
Stop primitive

- Terminates execution of mediation flow
- Consumes the incoming message silently and then terminates the flow
- Contains only an **in** terminal, no **out** or **fail** terminals
- Has no configurable properties
- If wired to a normal output terminal, behavior is the same as if the output terminal were unwired (the message is lost, silently)
- If wired to fail terminal, exception from the fail terminal is consumed silently, rather than propagated



Fail primitive

- Terminates execution of mediation flow
- Consumes the incoming message, raises a `FailFlowException`, then terminates the flow
- You can specify an error message, and part or all of the SMO to include with the exception (default SMO content that is listed is `/context/failInfo`)
- Uses the same message format (with substitution variables) as the message logger primitive



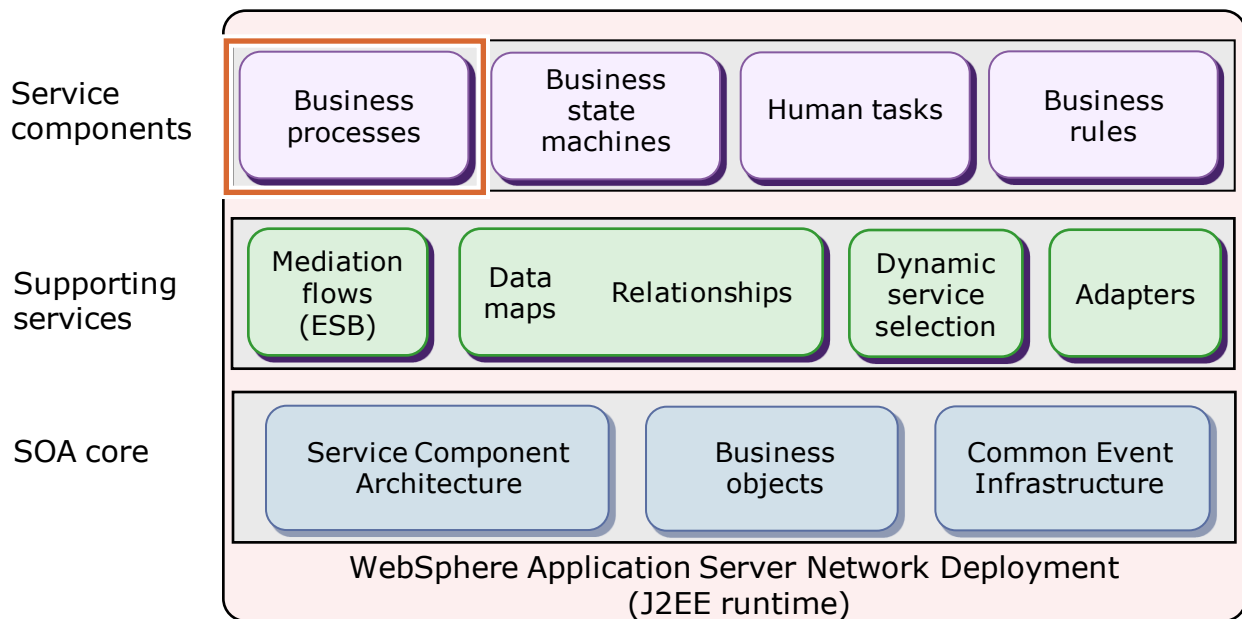
Fail

Business Process Choreography



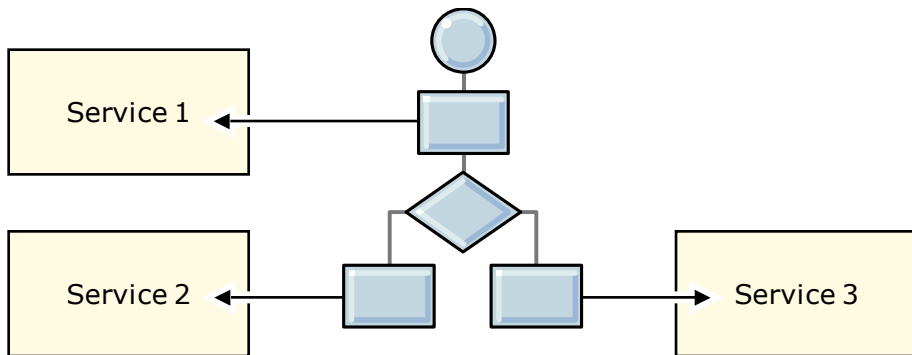
Business processes are service components

- Business processes are part of the service component layer



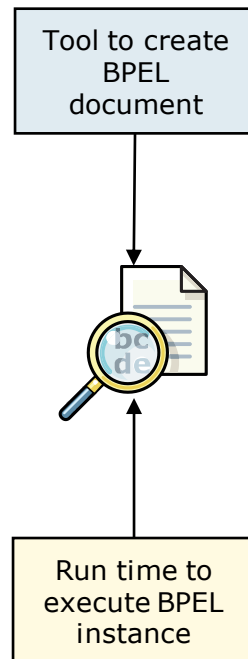
Business Process Execution Language Overview (1 of 2)

- A business process is a flow of execution paths that are described in WS-BPEL (Web Services Business Process Execution Language), including:
 - Which services are invoked
 - In what order services are invoked
 - The movement of data between services
- BPEL facilitates the building of composite integration applications by allowing reuse of existing IT assets that are exposed as services



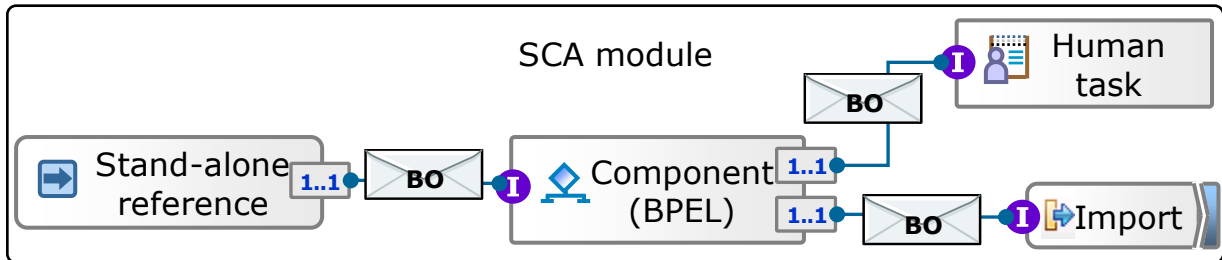
Business Process Execution Language Overview (2 of 2)

- Implementation in BPEL decouples the process from the runtime engine, making processes vendor and technology neutral
 - Other companies provide products with BPEL runtimes
- BPEL supports the required technology patterns for business
 - For example: error handling, compensation, and asynchronous processing
- The BPEL-compliant business process container included with IBM Process Server manages and runs business processes
 - Runs complex business processes securely, consistently, and with transactional integrity
 - Provides high performance and quality of service
 - Provides fault tolerance and error-detection capability



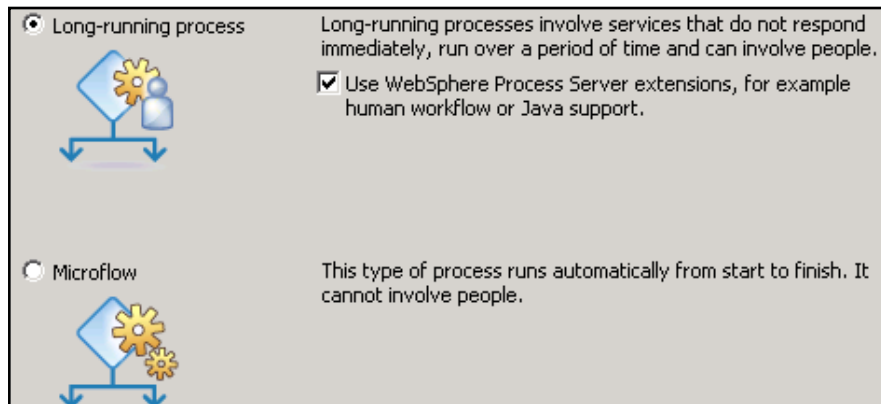
BPEL interoperates with SCA and SDO

- BPEL processes:
 - Can invoke other SCA components
 - Can be invoked as SCA components
- BPEL implements reference partners and interface partners (called partner links) that are translated into SCA component interfaces and references
- SDOs (business objects) are the standard data format for messages
 - SDO messages sent and received
 - Business process variables use SDOs



Microflows versus long-running processes

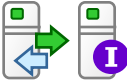





- Business processes run on IBM Process Server as either microflows (short-running) or long-running processes
- A microflow process is used for running short business processes or small units of work within a larger business process (subprocesses)
- A long-running process might run for hours, days, or weeks
 - Frequently involve components with lengthy response times such as human tasks
 - State of the process must be persisted



Business processes in IBM Business Process Manager

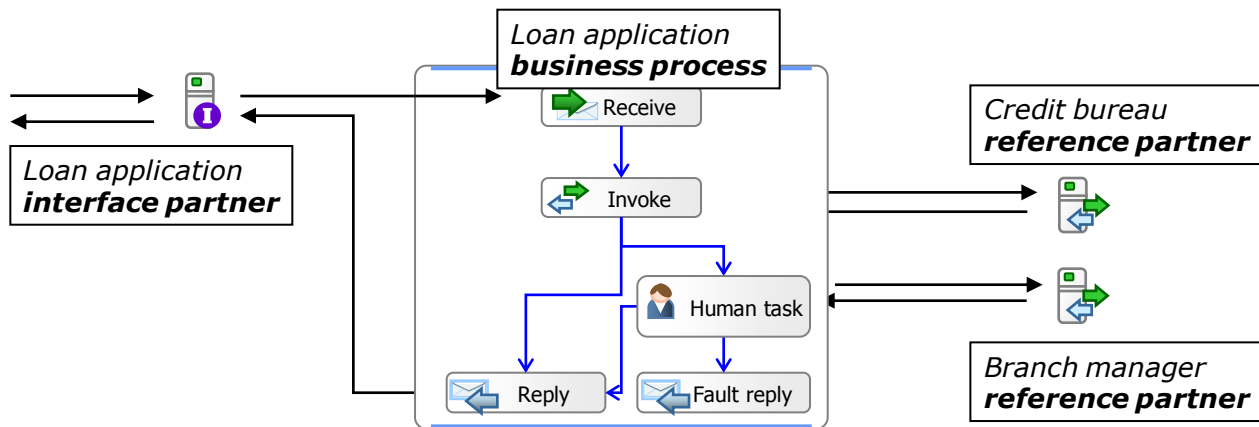
- Business processes might be captured in IBM Process Designer or in IBM Integration Designer
- IBM Process Designer
 - Captured as business process
 - Represented as Business Process Diagram (BPD)
 - Implementations that are captured in one process application or toolkit only
 - Limited implementation options
 - One human task client
- IBM Integration Designer
 - Captured as BPEL process
 - Represented with Business Process Execution Language (BPEL)
 - Implementation might span several modules
 - Might be shared in libraries
 - Several implementations options
 - Several human task clients

Seven main elements of a business process

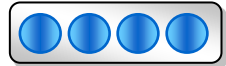
Icon	Element description
	1. Partners define parties that interact with the process
	2. Variables specify information that is used while running the business process
	3. Correlation sets match messages to the correct process instance
	4. *Fault handlers recover from partial and unsuccessful work that is done in the current scope of the business process
	5. *Compensation handlers contain actions that do reverse operations for a particular scope or activity
	6. *Event handlers do work that is based on an event or an asynchronous message
Varied	7. *Activities are used to define the process logic

Process elements: Partners

- **Partners** represent the external entities with which your business process interacts
 - An interface file describes the operations that the partner advertises
- The business process engine recognizes two categories of partners:
 - **Interface partners** define the services that the business process provides, which clients can invoke
 - **Reference partners** define the services that the business process can invoke



Process elements: Variables



- **Variables** hold data that represents the state of a process
- Variables:
 - Can be internal, received from partners, or sent to partners
 - Can be inputs or outputs for Invoke, Receive, and Reply activities
 - Can be manipulated by Assign and Snippet activities
 - Can be global (process-level) or local
- Variables are typically associated with WSDL message types
 - XSD data types are also available
- Variables are tied to scopes (containers of one or more activities)
 - Must be initialized by the runtime per scope before they can be used
 - You can control initialization order for multiple variables
- Global variables:
 - Are available to the process and embedded scopes
 - Can be queried at run time
- The process tray shows both global variables and variables that are defined in the selected scope (local variables)
 - Can drag variables from tray onto activities to assign inputs and outputs

Process elements: Correlation sets



- **Correlation sets** route an incoming message to a long-running business process instance
 - Messages must contain a set of variables and values that uniquely identify the contents
- A correlation set has a name and is composed of correlation properties
 - Each property consists of a name and a data type
 - Message contents are mapped to correlation properties
- At run time, values in the message determine the business process instance to which that message is routed
 - The unique values must be initialized the first time when the correlation set is used
- Correlation sets can be specified for processes or for individual activities

Process elements: Activities

- **Activities** are the individual business tasks that implement the larger business goal that the process represents
- An activity can be: basic, structured, or associated with error processing
 - Basic activities do not contain other activities: human task, snippet, reply
 - Structured activities are activities that contain other activities: sequence, while loop
 - Activities for fault handling and compensation are activities that process expected and unexpected error conditions
- BPEL also uses handlers with certain activities
 - Handlers contain activities that are run as a result of events or during error processing

Overview of WS-BPEL activities

- Activities are the individual business tasks that operate with each other to implement the larger business goal, which is represented as the process that contains them
- An activity can be one of several different types: basic activities, structured activities, or activities that are associated with error processing
 - Basic actions are activities that have no structure and do not contain other activities: human task, snippet, and reply
 - Structured activities are activities that contain other activities: sequence and while loop
 - Activities for fault handling and compensation are activities that process expected and unexpected error conditions in processes
- BPEL also uses handlers with certain activities
 - Handlers contain other activities that run based on events or during error processing

Basic activities: Receive, reply, and invoke

- The **receive** activity receives a message sent to a business process
 - Can start a new business process
 - Can restart an existing process
 - Request can be synchronous or asynchronous
- The **reply** activity responds to a message received
 - Typically used as a response to a synchronous request
 - Can return either response message or fault message
- The **invoke** activity calls a one-way or a request/response operation that a partner offers
 - Calls another service or business process



Basic activities: Assign

Assign From

Assign To

Risk was LOW. Application automatically approved.

MessageVariable message

Variables

Value Composer

Assign

- An **assign** activity is used to update the values of variables with new data
 - Values can be copied from source variables to destination variables
 - Full XPath support is provided
 - A Value Composer is provided to specify initial values for complex variables

Basic activities: Receive choice

Select an Interface

You can generate a new interface with one operation, or use an existing interface with one, or more operations.



- ☐ Generate a new Interface
- ☒ Select an existing Interface

Interface:

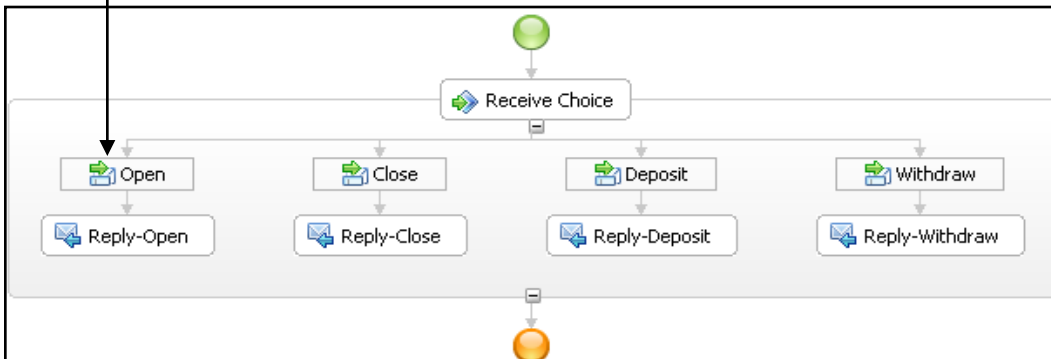
Select operations to start the process.

- Operations:
- ☒ open
 - ☒ close
 - ☒ deposit
 - ☒ withdraw



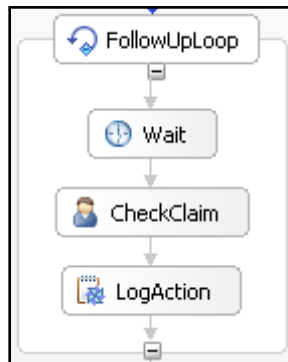
Receive choice

- **Receive choice** selects one branch of activities to execute based on a receive case
- Receive cases are interface operations



Basic activities: Wait

- A **wait** activity stops the business process for a specific amount of time that an expression evaluates
 - Java and XPath expressions allow date (visual, Java, literal date-time) or duration (visual or Java)
 - Timeout expressions allow simple calendar, WebSphere CRON, or user-defined values
 - Business calendars also supported
- Wait is only available for long-running processes



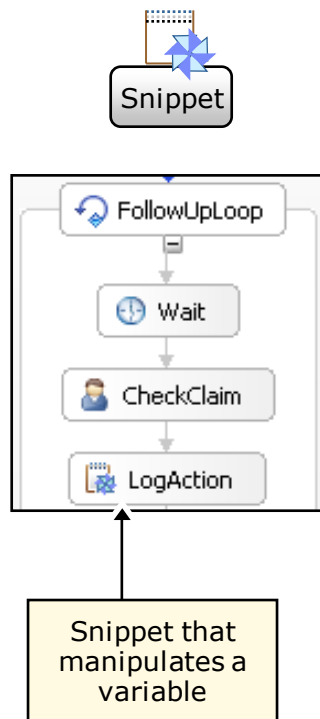
Basic activities: Human task

- **Human task** activities are an IBM extension to BPEL
 - Human tasks are defined in the BPEL4People specification
- Human task activities in a business process are called inline human tasks
 - Inline tasks send process-related messages to humans for completion
 - Humans can send messages to processes by being given authorization to a receive or a case in a receive choice
- Assigning a task to a human involves the interaction between two editors in IBM Integration Designer
 - You use the process editor to compose a process that requires human interaction
 - You use the human task editor to configure the task
- More details on human tasks are provided later in this course



Basic activities: Snippets

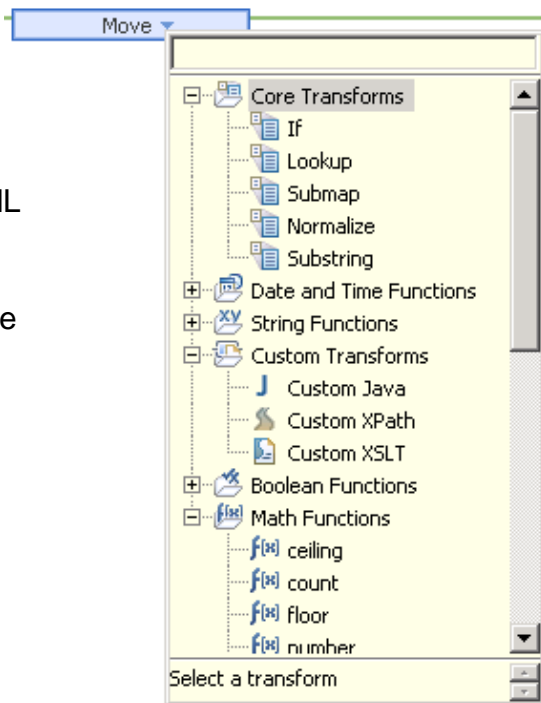
- The **snippet** activity allows developers to embed inline Java code in a business process
- Java snippets are an IBM extension to BPEL
 - Java snippets are detailed in the BPELJ specification
- The code runs locally in the BPEL process
 - Snippets in a BPEL process are compiled into a single class
- Snippets are typically used for working with the contents of variables
- Snippets are also used for:
 - Loop counters
 - Data validation
 - Data normalization
- Java snippets can be edited graphically (visual) or textually (Java)
 - A visual snippet can be converted to Java and edited textually
 - Moving back to a visual snippet after converting to Java is not supported



Basic activities: Data map



- Data maps provide structural and semantic transformation of business objects
- Maps can be business object maps or XML maps
 - Business object maps are needed only for relationships
- XSL style sheets that are generated from XML maps are used in XSL Transformation primitives in mediation flow components
- XML maps can be stored in catalogs for reuse
- Rich set of predefined transformations, including:
 - Core transforms: if, submap, lookup, normalize, substring
 - Date and time functions
 - String functions
 - Boolean functions
 - Math functions
 - Custom transforms



Activity exit conditions

- Most basic activities and human tasks support exit conditions
 - Receive choice activity does not
- Exit conditions can be set:
 - **On entry:** Determine whether an activity should be performed
 - **On exit:** Determine whether navigation should continue as expected upon completion
 - Both entry and exit
- Expressions can be Java, simple (boolean), or XPath

Invoke - CreditReportService

Specify a condition that ensures that the activity has produced all the data it is expected to produce, and the point in time when this should be checked.

Evaluate condition: -- None --

Expression language: No Expression

Condition:

Create a New Condition

- var1 is equal to var2
- var1 is greater than var2
- All of the following are true
- Any of the following is true
- input1 : string
- String
- Number
- true
- false

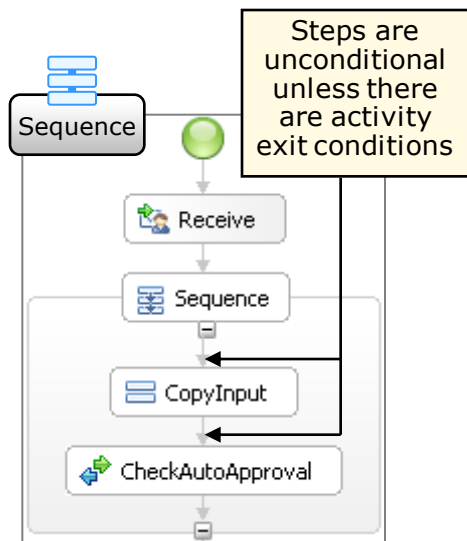
Evaluate condition: -- None --

Expression language: on both entry and exit on entry on exit

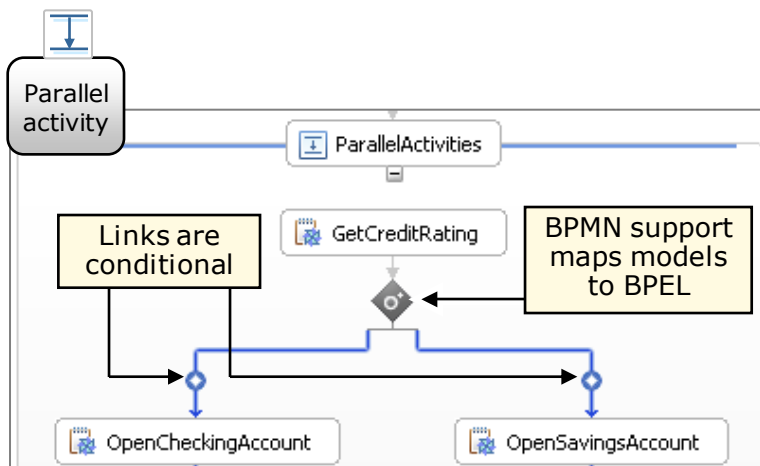
Condition:

Structured activities: Sequences and parallel activities

- A **sequence** serializes the execution of nested activities
 - Activities run one at a time, in order

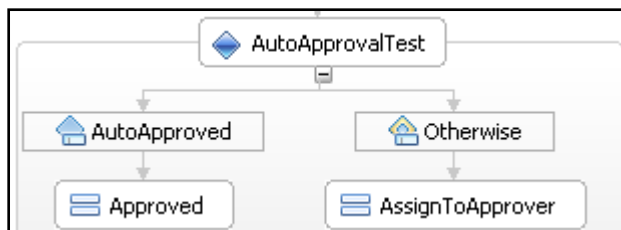
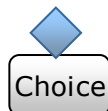


- A **parallel activity** depicts potentially parallel paths
 - Conditional logic is specified on interactivity links
 - Parallel activities in microflows run sequentially



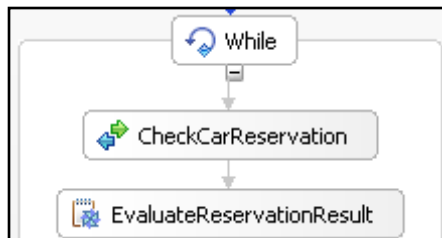
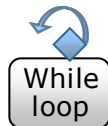
Structured activities: Choice

- A **choice** activity selects one activity branch from a set of case elements that are based on a runtime condition
- The condition is created by using an expression language:
 - Java expressions (visual or text)
 - XPath expressions
 - Simple expressions (boolean true or false)
- At run time:
 - The first case element to evaluate to true is run
 - If no case element evaluates to true, the otherwise element runs
 - If otherwise is omitted, control passes to the activity after the choice



Structured activities: While loop

- The **while loop** repeatedly executes the activities in its scope while the condition is true
- The condition is checked before the first iteration so it is possible that **none** of the nested activities run
- The condition can be created by using:
 - Java expressions (text or visual)
 - XPath expressions
 - Simple expressions (boolean true or false)
- A fault (handled or unhandled) terminates the loop
 - It is more efficient to end the loop by meeting the condition than by throwing an exception



Structured activities: Repeat until loop

- The **repeat until loop** executes the activities in its scope at least once and then continues to loop until the condition evaluates to true
- The condition is checked at the end of each iteration so the activities run **at least** once
- The condition can be created by using:
 - Java expressions (visual or text)
 - XPath expressions
 - Simple expressions (boolean true or false)
- A fault (handled or unhandled) terminates the loop
 - It is more efficient to end the loop by meeting the condition than by throwing an exception

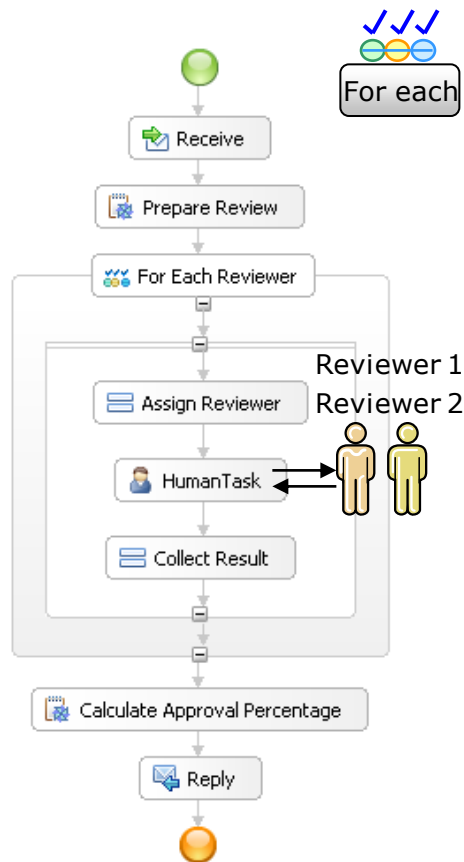


Repeat until
loop



Structured activities: For each

- The **for each** activity enables the bundling of work
 - Requests requiring a dynamic number of reviewers
 - Continue after some of the quotes are received
- The for each supports the WebSphere Business Modeler concept of arbitrary looping (repeating single activities)
 - Not necessary to place all repeated activities in a while loop
- Within the bundle, you control:
 - Whether dynamic number of branches can be run serially or in parallel
 - Whether all branches are required for completion
 - Early exit criterion specifies termination after certain subset of branches



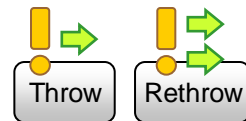
Structured activities: Scope

- A **scope** is a behavioral container for one or more activities in your process
 - Encapsulates variables and correlation sets (state)
- By definition, your entire process is contained within a single global scope
- You can nest other scopes within the global scope, forming a hierarchy
- Each scope can have its own:
 - Fault handlers (for expected error processing in the scope)
 - Event handlers (for event processing in the scope)
 - Compensation handlers (for unexpected error processing in the scope)
 - Local variables
- Scopes support dynamic runtime behavior



Handlers and error processing activities: Throw and rethrow

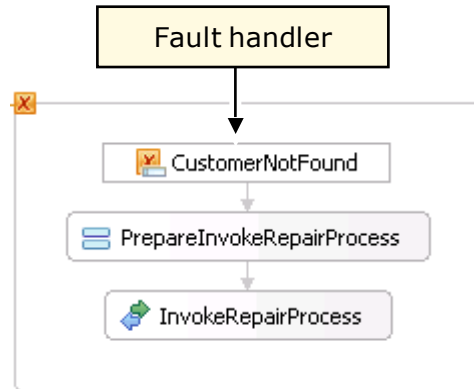
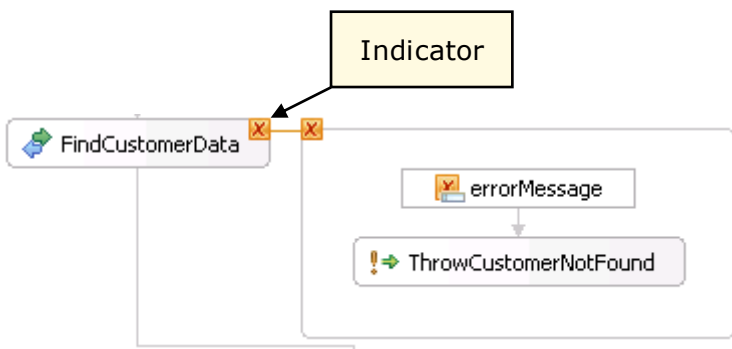
- Throws can raise a built-in or user-defined fault
 - The engine throws built-in faults to signal low-level failures in the system
 - BPEL-defined standard faults for common failures (such as joinFailure)
 - User-defined faults that a business partner or process throws
- The rethrow activity takes the current fault in a fault handler and raises it to the next enclosing scope
 - Rethrow raises a new fault with the same name and parameters as the fault that the containing handler caught
 - Indicates that error state still holds
 - Rethrow only available in fault handlers



Handlers and error processing activities: Fault handler

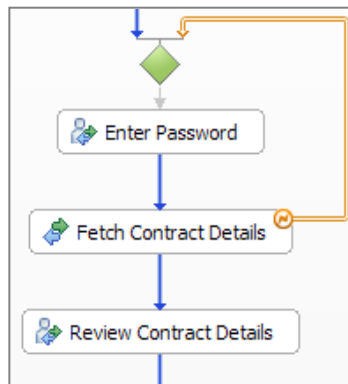


- A fault handler is a collection of specific activities that run when a fault (expected error) is thrown on the activity or scope
- Fault handlers use “catch” and “catch all” elements to handle error conditions in a process
 - “Catch” elements process specific faults
 - “Catch all” elements process faults that a “catch” element does not intercept
- A scope can have multiple fault handlers
 - Different kinds of faults can have different fault-handling activities



Error processing: Fault links

- Fault links are run after a fault occurs in the source activity of the link
 - When a fault occurs and is caught, only the fault link is navigated
 - The fault path can merge back into the regular flow
 - Evaluation order of multiple fault links can be specified
- Fault catching rules are the same as for fault handlers
 - A fault name and fault variable to catch can be specified (Catch), or all faults can be caught (Catch All)
- Fault links are available in generalized flows

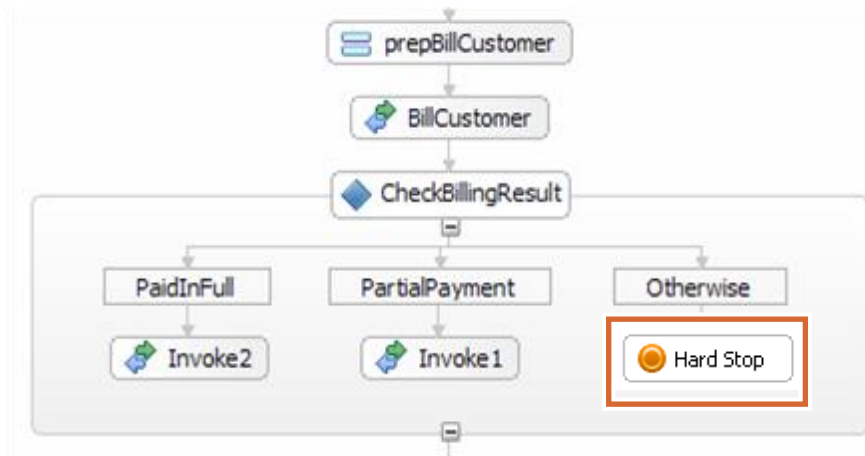


Fault Link - Link6

<div>Description</div> <div>Details</div>	<input checked="" type="radio"/> Catch <input type="radio"/> Catch All
	Fault Type: <input type="radio"/> Built-in <input checked="" type="radio"/> User-defined
	Fault Name: <input type="text" value="{http://BetterFinancials/Processes/ProvideLoanInformation/ProvideLoanInformationInterface}badPasswordFault"/>
	Fault Variable: <input type="text" value="(none)"/> <input type="button" value="Browse..."/>

Handlers and error processing activities: Terminate

- A **terminate** activity ends processing of the business process instance (hard stop) or structured activity
 - Can be used to allow administrator to make repairs
- All execution ends, including parallel execution paths
- Compensation of terminated activities does not occur



Business Process Choreographer Explorer client

Business Process Choreographer Explorer

[Welcome admin](#) | [Logout](#) | [Manage Views](#) | [Customize](#) | [Help](#) | [About](#)

Views

Process Templates

Currently Valid

All Versions

Process Instances

Started By Me

Administered By Me

Critical Processes

Terminated Processes

Failed Compensations

Activity Instances

Stopped Activities

Task Templates

My Task Templates

Task Instances

My To-dos

All Tasks

Initiated By Me

Administered By Me

All Tasks

Use this page to work with task instances for which you have access rights. ⓘ

Work on

Release

Transfer

Start

Change Business Category

Refresh

<input type="checkbox"/>	Priority ▾	Task Name ▾	State ▾	Kind ▾	Owner ▾
<input type="checkbox"/>	5	Request More Documentation	Ready	To-do Task	

Items found: 1
Items selected: 0

<< Page 1 of 1 >>

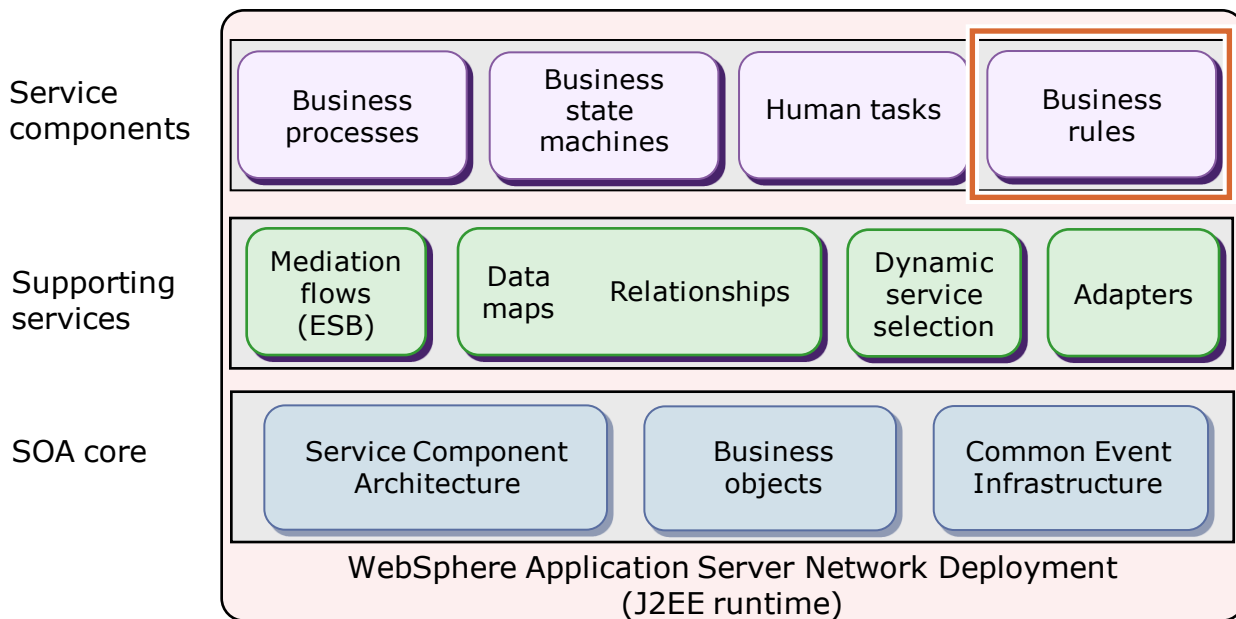
- Business Process Choreographer Explorer is included for basic process administration
- Business Explorer is built with reusable JavaServer Faces (JSF) components

Business Rules



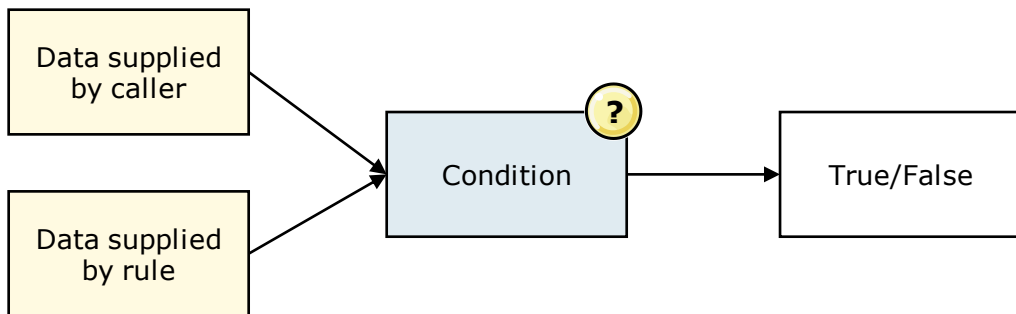
Business rules are a service component

- Business rules are part of the service component layer



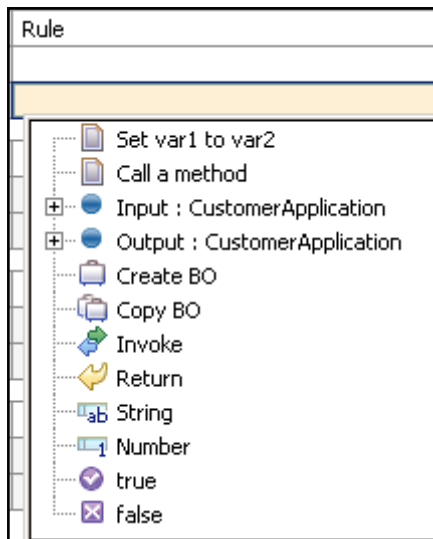
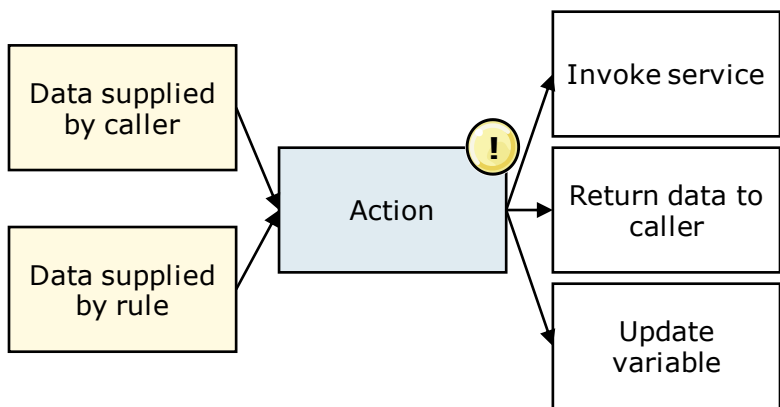
What is a business rule?

- A business rule captures and implements business policies and practices by using one or more if-then statements
 - For example: If orderTotal \geq 1000, then discount = 0.10
- A business rule consists of a condition (an expression that uses data that the caller and the rule supply) and one or more actions
 - The condition is the “if” portion of the statement
 - Evaluation of the condition is either true or false
 - The action is the “then” portion of the statement



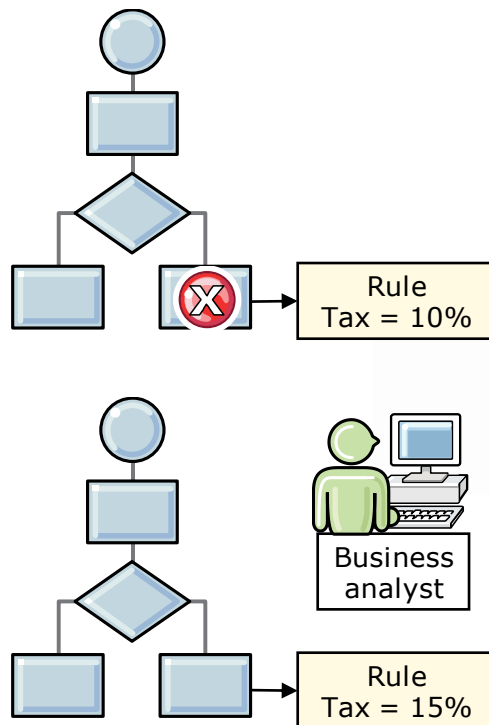
Business rule actions

- Possible actions that result from evaluation of the condition:
 - Invoking a service
 - Updating a local variable
 - Modifying data that is going to be returned to the caller
 - Creating or copying a business object
 - Returning (stop execution of the rules early)



Business value of rules

- By using rule groups to expose rules as services, rules are separated from processes that use them
 - Rule sets are reusable
 - Multiple business processes can use the same sets of business rules
 - Rules are no longer in application code
- The business analyst can quickly change exposed rules at run time, providing business agility and responsiveness
 - You are no longer bound to IT development cycles if rules are not in application code
 - The developer is needed only for more complex changes
- Rule groups are SCA components
 - As an SCA implementation type, it abstracts and decouples the rule implementation



Rule sets defined

- Two types of rules can be used in a rule set: if-then and action
 - An if-then rule evaluates a condition, and then performs one or more actions
 - An action rule always performs actions, regardless of the input, without using a condition
- All rules in a rule set are evaluated sequentially: first to last
- Each **condition** that evaluates to true causes an **action** to be done
 - More than one rule can be fired
- If-then and action rules can be exposed as template rules
 - Templates are exposed at run time in a natural language format
- The Business Rules Manager web client or Business Space client can be used to edit template rule parameters at run time
 - The developer can constrain parameter values

Rule sets in IBM Integration Designer (1 of 2)

- Rule sets in IBM Integration Designer contain:
 1. Name (and Display Name)
 2. Interface: provides the operation, inputs, and outputs
 3. Variables: rule actions can update local variables that you define

1

▼Rule Set

Name	checkCustomerTypeRS	Display Name
------	---------------------	--------------

2

▼Interface

Interface	CheckCustomerType	
Operation	checkCustomerType	
Input	inputCustomerType	string
Output	outputValidType	boolean

3

▼Variables

Name	Type

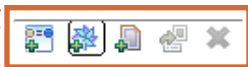
(Rule set contents continue on the following page)

Rule sets in IBM Integration Designer (2 of 2)

- Rule sets in IBM Integration Designer also contain:
 - Rules: if-then rules and action rules
 - Templates: rules with editable parameters that are exposed in the runtime environment

4

Rules



The **rules** icon palette includes:

- Add If-Then Rule
- Add Action Rule
- Add Template Rule
- Convert Rule to a Template

Name	Rule1
Template	Template_Rule1
Presentation	If inputCustomerType.equalsIgnoreCase(Competitor) == false then outputValidType = true

5

Templates



The **templates** icon palette includes:

- Add If-Then Template
- Add Action Template

Name	Template_Rule1
Presentation	If inputCustomerType.equalsIgnoreCase(stringParam0) == booleanParam1 then outputValidType = booleanParam2
Description	

Parameters	Name	Type	Constraint
	stringParam0	string	None
	booleanParam1	boolean	None
	booleanParam2	boolean	None

Exposed parameters can be constrained to a predefined set of values

If	inputCustomerType.equalsIgnoreCase(stringParam0) == booleanParam1
Then	outputValidType = booleanParam2

Rule groups

- A rule group is an SCA component that is used to dynamically invoke rule sets and decision tables, based on a set criteria
 - The criteria is a date and time range
 - Only one target runs based on a date selection criteria
 - Start and end date/time criteria, and target rule sets or decision tables, can be modified at run time
 - Criteria can also be based on the content of the business object input, but this type of choice must be created programmatically
 - New rule sets and decision tables cannot be created at run time, but new rules can be created from exposed templates
- Groups organize rules that share a common business purpose
 - Groups are searchable at run time
 - A rule group might contain any number of rule sets or decision tables
- Rule groups are presented like any other SCA component

Rule group components

- Rule groups are composed of:
 1. An interface
 2. A default destination (guarantees a target)
 3. One or more target destinations corresponding to start and end date/time ranges (dates cannot overlap)
 4. A date selection criteria (value is compared to the target start-end dates)
 5. A set of available destinations (rule sets and decision tables)

The screenshot displays the configuration interface for a rule group, divided into several sections:

- General:** A tab at the top left.
- Interfaces:** A section containing a list of interfaces. Item 1 points to the **CreditRiskAssessment** interface, which includes an **InputCriteria** sub-item.
- Scheduled Rule Logic:** A section containing a table of scheduled rule logic. Item 2 points to the **Default Rule Logic** tab, which is set to **CreditRiskAssessmentRS**. Item 3 points to the table rows showing date ranges and rule logic.

Start Date	End Date	Rule Logic
Jan 13, 2008 12:00 AM	Mar 8, 2008 12:00 AM	CreditRiskAssessmentRS
Mar 9, 2008 12:00 AM	Jul 1, 2008 12:00 AM	CreditRiskAssessmentDT
- Selection Criteria:** A section containing a table of selection criteria. Item 4 points to the **Selection Criteria** tab, which is set to **Current date**.

Selection Criteria	Current date
- Available Rule Logic:** A section containing a list of available rule logic. Item 5 points to the list, which includes **CreditRiskAssessmentRS** and **CreditRiskAssessmentDT**.

Business rule group properties

- There are two types of rule group properties: system and user-defined
- System properties are read-only, and IBM Integration Designer and IBM Process Server use them internally
- User-defined (custom) properties are read and write
- Custom properties are used for management of rule groups
 - Number of user properties that you can define is unlimited
 - Added and deleted in IBM Integration Designer, modifiable in Business Rules Manager
 - Can be queried through Business Rules Manager and the public business rules management API
 - Can hold customer-specific information

Rule Group - CreditRiskAssessment

▼Custom Properties

+

×

Name	Value
Contact	john.doe@us.ibm.com

Business rules in IBM Process Designer

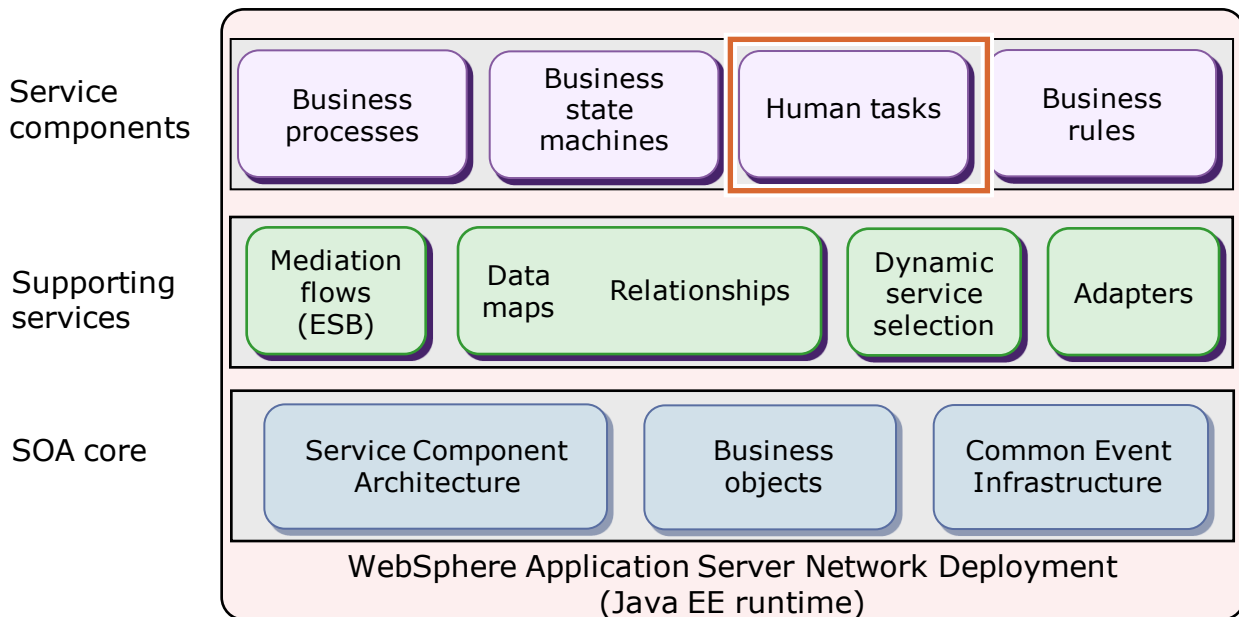
- An activity implementation which contains business rules is possible in IBM Process Designer
 - Called a “decision service”
 - Written in Business Action Language (BAL)
- There are differences between rules in IBM Process Designer and IBM Integration Designer:
 - Decision services implement activities or actions only; they do not assign values or run self-contained code
 - BAL is a declarative language, business rules in IBM Integration Designer support declarative presentations and programmatic rules
 - Rule sets are not programmatically selected
 - Decision services can integrate with ILOG JRules

Human Tasks



Human tasks are service components

- The Human Task Manager runs in a separate container on IBM Process Server
 - Accessible through SCA or specific APIs



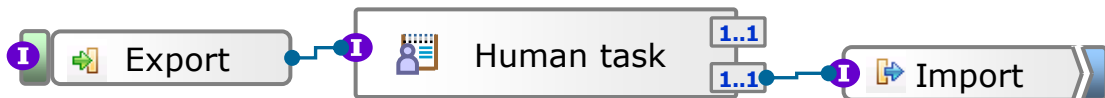
Overview of human task support

- Human interaction is key to many business integration applications
 - Human input or review is often required
 - Automation is not possible for some tasks
 - Error or exception situations might require handling by person

- Human tasks in IBM Integration Designer are robust and follow a service-oriented approach
 - Generate, assign, and store tasks for individuals and groups that are listed in organizational directories
 - Specify different levels of authority
 - Transfer and suspend tasks
 - Enable expiration, escalation, notification, subtasks, follow-on tasks, and participant substitution
 - Generate custom web clients for tasks

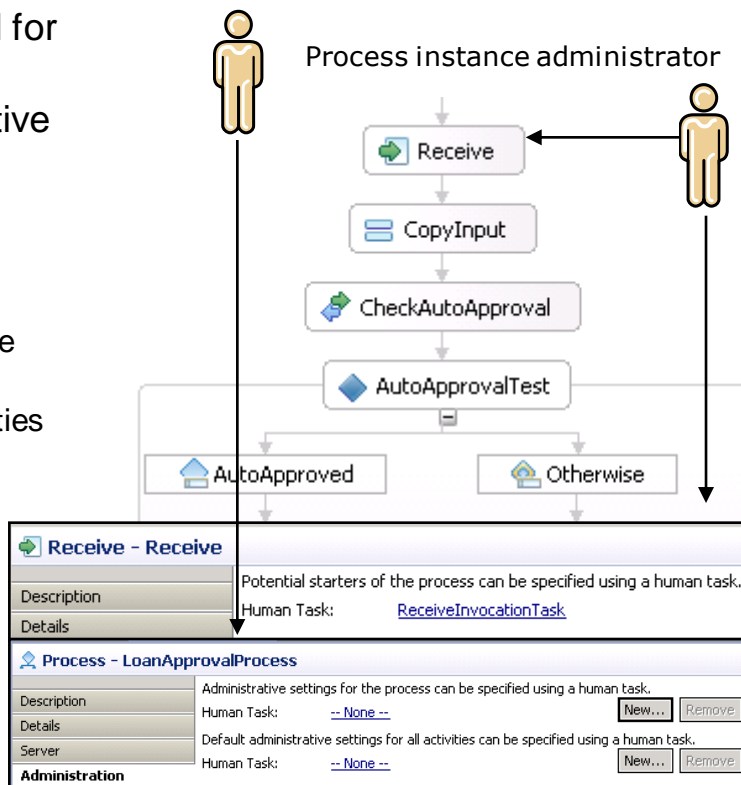
Inline, stand-alone tasks and SCA

- Tasks that are implemented in a business process are called **inline** tasks
 - Inline tasks are activities in a business process, not SCA components
 - Inline tasks have access to process related information
 - Inline tasks can be used to apply permissions to BPEL activities like invoke, receive, and receive choice and to apply permissions to event handlers
 - Inline tasks can be used to create human tasks for process administration
- Stand-alone tasks are independent SCA components (not part of a business process)
 - Stand-alone tasks are invoked as services and can invoke other services
 - Stand-alone tasks can be invoked by other SCA components like business processes and other non-SCA components like JSP pages
 - Stand-alone tasks do not have direct access to process-related information
 - Business objects provide a standard data format for sending and receiving messages in human tasks






Inline tasks in business processes

- Human tasks can be used for process administration
- Grant humans administrative authorization to:
 - Suspend a process
 - Terminate a process
 - Restart processes
 - Force-retry or force-complete processes
 - Administer all process activities
 - And others
- Inline tasks can also be used for authorization:
 - On incoming operations: receive, receive choice, event handlers
 - On other individual activities: invoke, snippet, and others



Types of human tasks

- There are different types of human tasks available for different integration situations

<p>To-do task</p> <ul style="list-style-type: none"> A service (WS-BPEL) creates work items for humans to perform A to-do task can be either stand-alone or inline 	 <p>To-do Task</p>
<p>Invocation task</p> <ul style="list-style-type: none"> A human invokes a service An invocation task can be either stand-alone or inline 	 <p>Invocation Task</p>
<p>Collaboration task</p> <ul style="list-style-type: none"> A human creates a task for another human Collaboration tasks are stand-alone There is no interaction between them and a business process 	 <p>Collaboration Task</p>

Bind to-do task to invoking process

- Processes can manage the lifecycle of stand-alone, “child” to-do tasks
 - If SCA by BPEL invoke activity calls the task, it behaves like a child task if bound to the process lifecycle; otherwise, it behaves like a stand-alone task
 - When parent process is terminated, any bound stand-alone tasks are also terminated
 - “Bind the lifecycle to the invoking business process” is available only for to-do tasks

People directory:	<u>Everyone</u>		
Task priority:	5		
Work basket identifier:			Insert Vari
Business category:			
Default language: *	English - United States		
Event handler name:			
Substitution policy:	No substitution		
Date when task becomes valid:	None	Select Date	Time zone: America/New_York Time: 0 : 0 : 0
<input type="checkbox"/> Business-relevant	<input checked="" type="checkbox"/> Enable subtask creation		
<input type="checkbox"/> Task can be claimed when it is suspended	<input checked="" type="checkbox"/> Task can be delegated		
<input checked="" type="checkbox"/> Enable follow-on task creation	<input type="checkbox"/> Bind the life cycle to the invoking component		
<input type="checkbox"/> Give owner read access to surrounding process context data			

Clients for human tasks

- Several clients are available for working with human tasks that are specified in the task template user interface settings
- Most can be generated by using the graphical client wizard
 - Business Process Choreographer Explorer: JSF-based application for basic and administrative actions on tasks
 - IBM WebSphere Portal clients: use existing portlet or generate a portlet by using the portlet generator
 - IBM Forms clients: generate human task client, based on electronic forms
 - Business Space client: Web 2.0 client for creating human workflow workspaces
 - HTML-Dojo pages can be created and rendered in Business Space
 - JSF and JSP clients
 - Custom clients: generate client by using calls to task container APIs
 - Coaches: web-based interfaces where process participants do the work that is required to complete each task

JSF and JSP clients

- Use a graphical wizard to generate JSP or JSF-based clients for human tasks
 - Stand-alone and inline human tasks for individual module or multiple modules
 - Specify which tasks to display in the generated client
 - Web or Java EE perspective is used to further customize JSF-based web clients
 - Provides easy way to demonstrate human task capabilities that include ad hoc tasks (subtasks only)
- The wizard also supports JSF clients for tasks that are associated with business processes

Generator type:

Description: This generator creates a client application that is based on JSF technology.

Human Task(s)

- ☒ **LoanApplicationModule**
 - ☒ Human Tasks
 - ☒ To-do
 - ☒ ManualApprover
 - ☒ Processes
 - ☒ LoanApprovalProcess
 - ☒ Invocation
 - ☒ Receive

Business Process Choreographer Explorer client

Business Process Choreographer Explorer

[Welcome admin](#) | [Logout](#) | [Manage Views](#) | [Customize](#) | [Help](#) | [About](#)

Views

- Process Templates
 - Currently Valid
 - All Versions
- Process Instances
 - Started By Me
 - Administered By Me
 - Critical Processes
 - Terminated Processes
 - Failed Compensations
- Activity Instances
 - Stopped Activities
- Task Templates
 - My Task Templates
- Task Instances
 - My To-dos
 - All Tasks
 - Initiated By Me
 - Administered By Me

All Tasks

Use this page to work with task instances for which you have access rights. ⓘ

[Work on](#) | [Release](#) | [Transfer](#) | [Start](#) | [Change Business Category](#) | [Refresh](#)

<input type="checkbox"/>	Priority ▾	Task Name ▾	State ▾	Kind ▾	Owner ▾
<input type="checkbox"/>	5	Request More Documentation	Ready	To-do Task	

Items found: 1 Items selected: 0 << Page 1 of 1 >>

- The prebuilt Business Explorer web client is included for basic actions and administrative tasks
 - Provides task administration: transfer, suspend, resume
 - Provides basic task actions: view, claim, complete
 - Supports dynamic tasks (called related tasks in the interface)

IBM WebSphere Portal clients

- Client generation wizard supports the portlet generator
 - Generates portlets for tasks
 - Used with MyTasks portlet supplied with IBM WebSphere Portal
 - Only supported for to-do tasks
 - Portlet creation requires Portal Toolkit

User Interface Wizard for Human Tasks

Client Generator Selection

Select a generator type and human tasks to generate a user interface.

Generator type:

Welcome
 Work with Pages
 Workflow

Tasks

Flight Book Portlet

Travel Request:

Employee	<input checked="" type="checkbox"/> <u>Juergen Employee</u>
Reason	Conference
Departure Date	5.10.2002, 9:00 am
Origin Airport	Frankfurt
Destination Airport	New York
Airline	Lufthansa
Class	Economy

Origin airport (IATA code):

Destination airport (IATA code):

Airport selection list:

Aalesund, Norway (AES)
 Aberdeen, Scotland, United Kingdom (ABZ)
 Aberdeen, SD, USA (ABR)
 Abu Dhabi, United Arab Emirates (AUH)

Departure date (yyyy/mm/dd):

Return date (yyyy/mm/dd):

Departure time (hh:mm):

Return time (hh:mm):

Seat Class:

IBM Forms clients

Business Cases > New > HealthCheck

Enter the values for the input data and/or provide additional information to create your task.

Input Data

The screenshot shows the IBM Workplace Forms client interface. At the top is a toolbar with icons for file operations (open, save, print, etc.) and a zoom level of 75%. Below the toolbar is a header area with the IBM logo and the text "IBM Workplace Forms". The main title of the form is "Health Insurance ~ Claim Form". On the left side, there is a navigation pane with three sections: "Client Info", "Dependent Info", and "Claim Details". The "Client Info" section is currently selected, and it displays a form titled "Client Information". The form contains the following fields: "Client Number" and "ID" (both with input boxes), "Last Name", "First Name", and "Initial" (each with an input box), "Street Address" and "City" (each with an input box), and "State" (a dropdown menu showing "Select State") and "Zip Code" (an input box).

- Client generation wizard supports IBM Forms
 - “Print equivalent” rendering
 - Requires IBM Forms Viewer (or Designer) to display forms
 - Supports digital signing of forms by multiple users
 - Can create tasks and processes from existing forms

Business Space for human workflow

- Business Space is Web 2.0 BPM client for business users
- Spaces, pages that are built by using mashup technology that integrates widgets
 - Predefined space templates that contain task-related widgets support human-centric workspaces
 - Customized layouts can be defined by using available widgets
 - Includes graphical process widget, escalations, task lists
- Widgets:
 - Use Representational State Transfer (REST) APIs to access runtimes
 - Dojo Toolkit and JavaScript used to create iWidget compliant widgets
 - iWidget editor is provided as part of the web development tools
 - Widget content can be used in IBM WebSphere Portal
- Uses Business Space Manager
 - Common framework for user interfaces in IBM products

Testing



Testing SCA components in the integrated test client

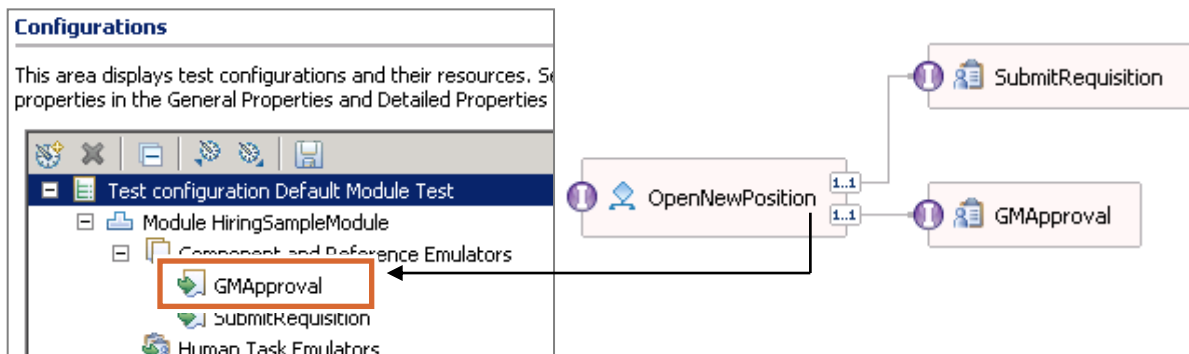
- Using the integrated test client in IBM Integration Designer, you can test:
 - An individual module
 - A set of interacting modules
 - An individual component
 - A set of interacting components
 - A test suite
- When testing components, tests are done on interface operations
 - Enables you to determine whether the components are correctly implemented and the references are correctly wired
- In component tests, unimplemented components or unwired references can be emulated
 - Modules do not need to be complete before testing
 - Emulation can be programmatic

Test configurations

- Test configurations control your tests
- A test configuration specifies one or more modules to test, each of which might include:
 - Zero or more emulators for components or references in the module
 - Zero or more monitors for the wires in the module
- When you open the integrated test client, a default test configuration is automatically created that you can immediately use for testing
- The default test configuration is often all that you need for testing your modules and components
 - You can choose to edit and customize the default test configuration, or you can create and edit one or more new test configurations
 - Customized test configurations can be saved and reused

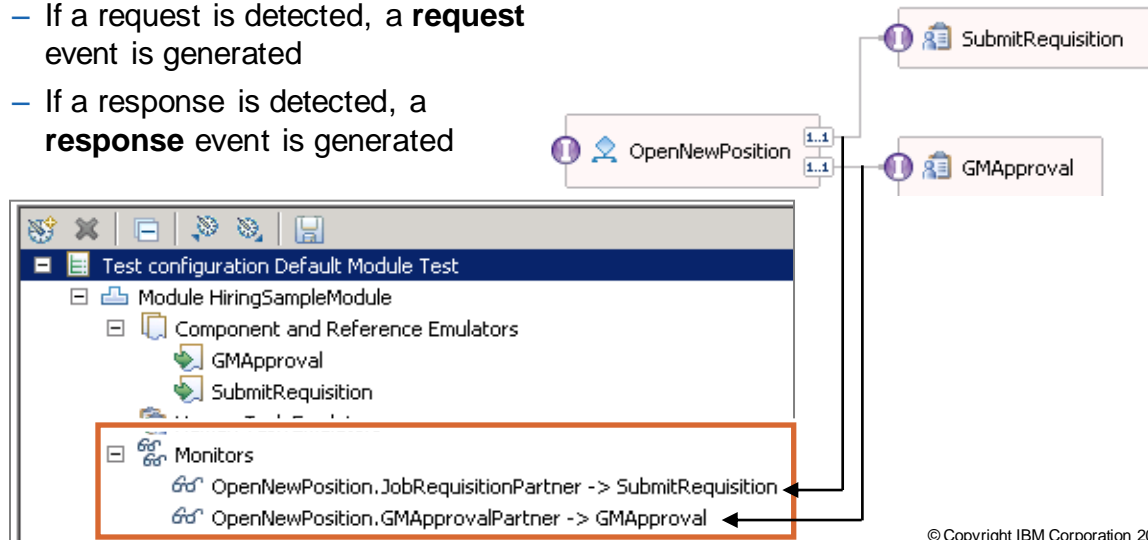
Emulators

- The integrated test client enables you to use emulators to emulate components and references in your modules
- During a test, when control flows to an emulated component or reference, the integration test client intercepts the invocation and routes it to the associated emulator
- There are two types of emulators:
 - Manual: test pauses for you to manually specify output parameter values
 - Programmatic: uses a Java snippet to provide response values automatically



Monitors (1 of 2)

- When you use the default test configuration or when you add a test configuration, monitors are automatically added for any component wires and exports in the module
- When you invoke an operation and run a test, monitors listen for requests and responses that flow over the wires and exports
 - If a request is detected, a **request** event is generated
 - If a response is detected, a **response** event is generated



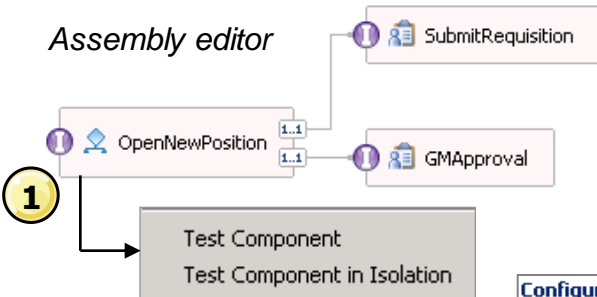
Monitors (2 of 2)

- Events show parameter data that flows across the wires
 - Added to the events that are displayed in the events area of the integrated test client
- You can edit the monitors and change whether they monitor requests, responses, or both
- You can also remove the monitors or add more monitors as required

▶ General Properties
▼ Detailed Properties
Source component: OpenNewPosition
Source reference: JobRequisitionPartner
Target component: SubmitRequisition
Target interface: JobRequisition
<div> <div>Monitor behavior</div> <div> <input checked="" type="checkbox"/> Monitor requests <input checked="" type="checkbox"/> Monitor responses </div> </div>

Testing SCA components (1 of 2)

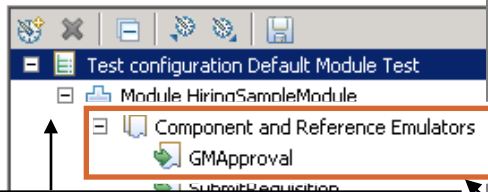
Assembly editor



1. Right-click the component and choose **Test Component** or **Test Component in Isolation** from the menu
2. Switch to the **Configurations** tab to view emulators and monitors

Configurations

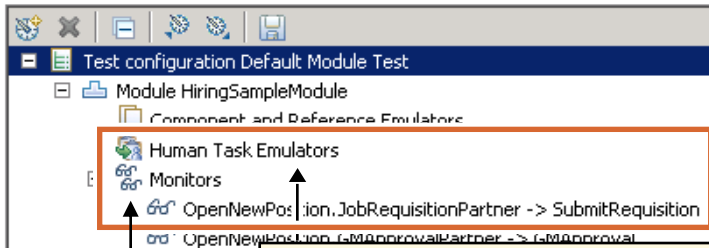
This area displays test configurations and their resources. Select a test configuration and view its properties in the General Properties and Detailed Properties sections. [More...](#)



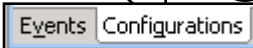
Test Component in Isolation tests the component and automatically configures monitors and emulators for all other components

Configurations

This area displays test configurations and their resources. Select a test configuration and view its properties in the General Properties and Detailed Properties sections. [More...](#)



Test Component automatically configures monitors but not emulators; tests the module that starts with the selected component



Testing SCA components (2 of 2)

Displays detailed properties for events such as initial request and return parameters

► **General Properties**

▼ **Detailed Properties**

Module: [HiringSampleModule](#)

Source component: [OpenNewPosition](#)

Source reference: [JobRequisitionPartner](#)

Target component: [SubmitRequisition](#)

Target interface: [JobRequisition](#)

Target operation: [createRequisition](#)

Request parameters:

Value Editor XML Source

Events

This area displays the events in a test trace. Select an event to display its properties in the Detailed Properties sections. [More...](#)

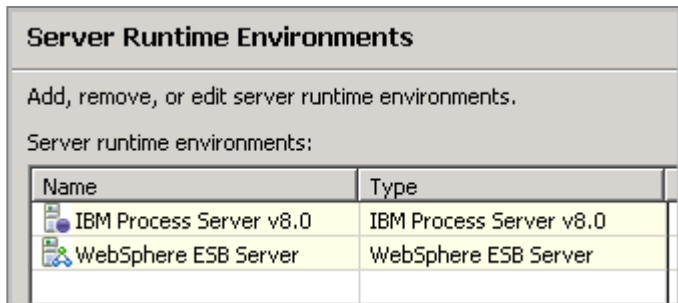
Invoke (OpenNewPosition:createPosition)

- Invoke started
 - Invoke (OpenNewPosition:createPosition)
- Fine-Grained Trace (OpenNewPosition:OpenNewPosition)
 - Receive (createPosition)
- GeneralizedFlow
 - Submit Job Requisition (createRequisition) [Running]

Execution trace (events)

Testing SCA modules in the integrated test client

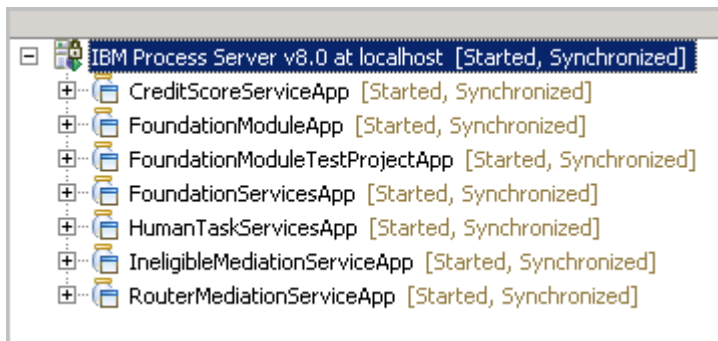
- Entire modules and groups of interacting modules are tested by using a configured server runtime
 - Installation of the runtime environment is optional
 - The workspace preferences display installed runtimes
 - Runtimes can be local or remote
- By default, testing a module or group of modules automatically builds and packages the projects, publishes them to the server, starts the server, and starts the applications
 - You can disable automatic publishing in the preferences



Note: Test environment configuration is done through the server administration console

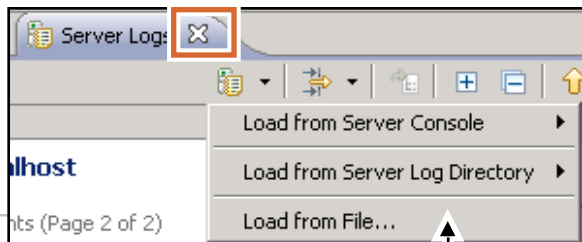
Using the Servers view when testing modules

- After deployment, the Servers view displays information about the state of the server and the projects that are published to it
- Projects and their state are listed under the server in a navigation tree
 - Projects are synchronized or need to be republished
 - Right-click modules, and choose **Remove** to remove them individually
- Server state is also visible:
 - Server is stopped, started, or in debug mode



Server Logs view (1 of 2)


- Server Logs view is used to display server console and SystemOut.log messages
 - Server console view also available
 - Filter messages by type and load invocation records
 - Refresh every 5 seconds (can be changed)




Server Logs

The Server Logs view is used to display the contents of the server console and server log files. It automatically displays console output for each server that is started, but you can also manually load and display the contents of the console and log files for any server. The Server Logs view provides several advantages over the traditional Console view, such as the ability to filter records and display invocation records in hierarchical format. If you want to open the Console view, select **Window > Show View > Console**.

Getting started with the Server Logs view

To load server console or log records into the Server Logs view, click the **Load Server Console or Log** icon .

To filter records in the Server Logs view, click the **Select Records to Display** icon .

To enable or disable cross-component tracing, click the **View Menu** icon .

To load invocation records into the integration test client, click the **Load into Test Client** icon .

- Load SystemOut.log from directory or other computer
 - Able to load multiple logs
- Message color:
 - Green: message
 - Purple: warning
 - Red: error

Server Logs view (1 of 2)

- Console view returns a sizable volume of messages
- Server Logs view is filterable view of server logs in one view
 - SystemOut.log, SystemErr.log
- Able to view SCA errors along with server errors
- Highlight events with different colors
- Displays only server logs
- Potential performance effect
 - Full logs are loaded in memory

(Filter: Show server state and errors)

Server

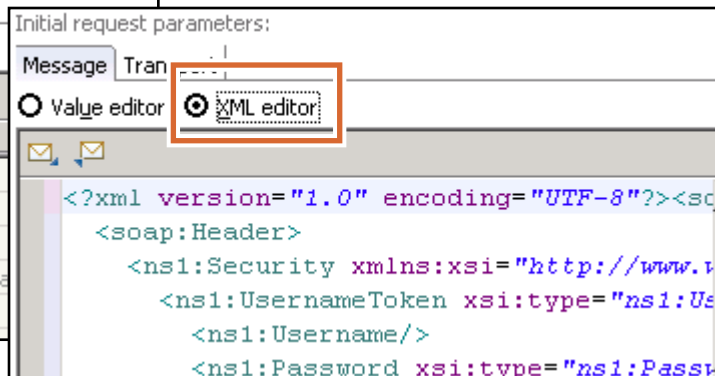
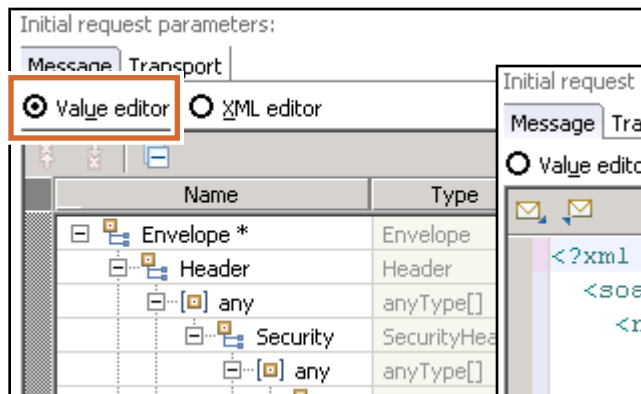
Log Fi

Log Records (Page 1 of 4 Filter matched 308 of 4115 records)

Creation time	Severity	Message	Repeat count	Source Thread
Jun 13, 2008 12:12:09.765000 AM	10	WSVR0200I: Starting application: IBMUTC	0	00000033
Jun 13, 2008 12:12:07.703000 AM	10	WSVR0220I: Application stopped: IBMUTC	0	00000033
Jun 13, 2008 12:12:07.656000 AM	10	WSVR0217I: Stopping application: IBMUTC	0	00000033
Jun 13, 2008 12:12:00.453000 AM	10	WSVR0001I: Server server1 open for e-business	0	0000000a
Jun 13, 2008 12:11:59.796000 AM	50	WWLM0069E: No CFEndPoint associated with the chai...	0	0000000a
Jun 13, 2008 12:11:59.796000 AM	50	WWLM0069E: No CFEndPoint associated with the chai...	0	0000000a
Jun 13, 2008 12:11:59.796000 AM	50	WWLM0069E: No CFEndPoint associated with the chai...	0	0000000a
Jun 13, 2008 12:11:59.796000 AM	50	WWLM0069E: No CFEndPoint associated with the chai...	0	0000000a

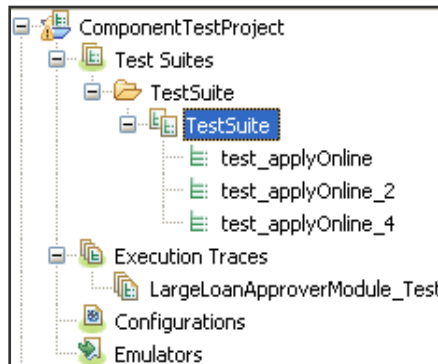
Testing web services

- When testing web service exports, you can import data from XML or HTTP files
 - SOAP messages are stripped from the file
 - Input SOAP messages are editable
 - HTTP file is captured from TCP/IP Monitor or attached
- The integrated test client has a full XML editor (value editor and XML editor) with syntax validation



Component test projects (1 of 2)

- Component test projects provide a way to automate running test cases
 - Component tests automate and test operations in the integrated test client
 - Test projects provide testing for components in integration (SCA) modules
 - Test projects are packaged as SCA modules (can add Java libraries by using dependencies)
- Component test projects can be deployed and run on the server
 - Results are displayed in the integration test client
 - Component test projects include suites, cases, and configurations
- **Test suites:** containers for test cases
 - You create test cases to test individual operations or groups of operations
- **Test cases:** containers for operations
 - Operation testing is automated by using predefined input and output variables
 - Test cases can be authored manually or by using the integration test client execution trace
- **Test configurations:** used to control tests
 - Test configurations specify required emulators and monitors



Component test projects (2 of 2)

- Component test projects are created in the test suite editor
- Associated wizards are used to create and define test cases that comprise multiple operations

Events

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)

General Properties

Detailed Properties

Select a test configuration for the Run Test Cases event, then click the Continue icon in the Events area to run the test. [More...](#)

To set the environment variables for this test, go to the selected [test configuration](#).

Configuration: AccountVerification_MED

Verdict : Passed

Total: 1/1

1/1

Passed: 1

1

Failed:

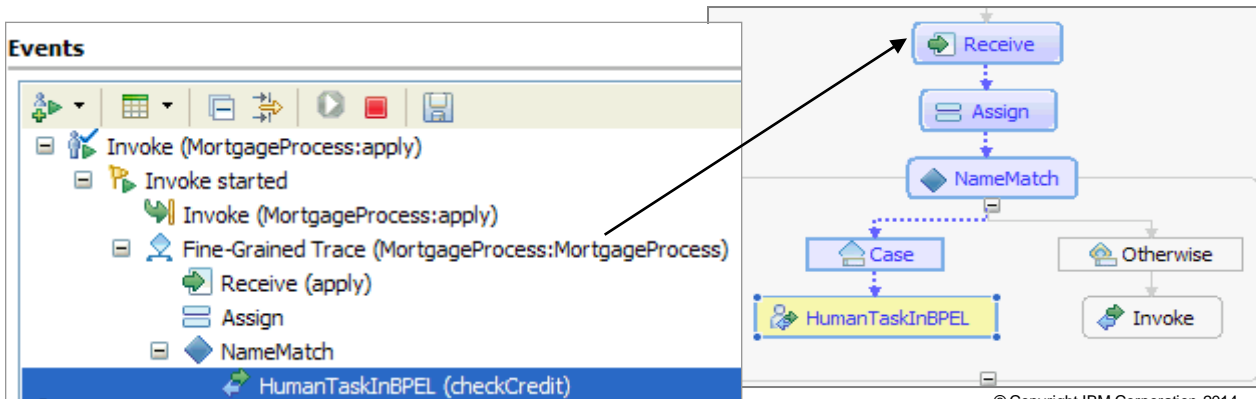
0

Error:

0

Fine-grained trace

- Fine-grained trace combines the test client and debugger to overlay the editors with a visualization to track steps in the path of events
 - Quickly visualize the path that is taken for a single execution for a component
 - Split screen feature while testing highlights activities as you pass through
- Fine-grained trace can be used in BPEL, state machines, and mediations and is enabled by default
 - Appropriate editors are auto-opened by clicking the event
 - The path of events is highlighted in the business process editor, mediation flow editor, and state machine editor



Command-line test invocation

- Integrated test client includes command-line invocation
- Automates tests by using Ant scripts
 - Schedule the tests during low usage time
 - Result is XML file that describes results
 - Batch files (for Windows platforms) and shell scripts (for Linux platforms) provided to run the Ant scripts
- Servlet that the test client calls to run test cases remotely in test project
 - Generated in a side Java EE project along with modules
 - URL for the servlet is in the form:
`http://<host>:<port>/<testproject>Web/TestServlet`
- Ant has core support for CVS
 - Test projects and modules can be extracted from CVS, built, deployed, tested, and removed
 - User needs to know CVS commands

Component Test Explorer (1 of 2)

- The Component Test Explorer web application allows you to:
 - Displays component test projects on the server
 - Query, run, and schedule tests
 - Globally emulate SCA components or human tasks
 - View component test projects, test suites, and test cases that run on the server
 - View test case results: pass, fail, exception

The screenshot shows the 'Component Test Explorer' web application. The interface includes a top navigation bar with tabs: 'Test Cases', 'Human Task Emulators', 'Component Emulators', 'Scheduler', 'Help', and 'Log'. The 'Test Cases' tab is active, displaying a tree view of test projects and suites. A red box highlights the tree view, and an arrow points to it from a yellow box labeled 'All test cases on server'. The right side of the interface contains a 'Details area' with fields for 'Description', 'Environment Variables', and 'Run Log'. A yellow box labeled 'Details area' points to the 'Environment Variables' section. Below this, there is an 'Execution history' section with a table showing test results. A red box highlights a row in the table, and an arrow points to it from a yellow box labeled 'Execution history'.

Component Test Explorer

Test Cases | Human Task Emulators | Component Emulators | Scheduler | Help | Log

FoundationModuleTestProject
 AccountVerificationTestSuite
 AccountVerification_MED
 RiskTestProject
 RiskTestSuite
 Risk_InputCriterion

Description

Environment Variables

Name

Add

Remove

Run Log

2012-09-05 05:14 RiskTestProject->RiskTestSuite

All test cases on server

Details area

Execution history

Component Test Explorer (2 of 2)

- Component Test Explorer emulators:
 - Use if component not available at test time or if execution can produce errors
 - Define a global emulator for any component in any module on server
 - Groovy assistance to emulate interface information for request and response parameters
 - Define and manage global emulators for specified human tasks

The screenshot displays the 'Component Test Explorer' application window. The top navigation bar includes tabs for 'Test Cases', 'Human Task Emulators', 'Component Emulators' (which is selected), 'Scheduler', 'Help', and 'Logout'. On the left, a tree view shows a hierarchy of components: 'BSM_VendingMachine', 'PurchaseOrderModule', and 'OrderReceiver'. The 'OrderReceiver' component is selected and highlighted with a dashed border. A yellow callout box points to the 'OrderReceiver' icon, stating: 'Check mark indicates that emulator is active'. The main right pane shows the configuration for the selected component, titled 'GlobalEmulator > Component'. It contains a 'Component' section with an 'ID' field set to 'PurchaseOrderModule::OrderReceiver'. Below this is a section titled 'Emulate this Component?' with a 'YES' button. A yellow callout box points to the 'YES' button, stating: '"Yes" enables the emulator'.

References

Other Reference Materials which will help give an introduction of IBM BPM

IBM Business Process Manager V8.0.1 documentation

- http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.0.1/com.ibm.wbpm.main.doc/ic-homepage-bpm.html

What's new in IBM Business Process Manager V8

- http://www.ibm.com/developerworks/bpm/bpmjournal/1206_pacholski/1206_pacholski.html

Creating a BPM Center of Excellence (CoE)

- <http://www.redbooks.ibm.com/abstracts/redp4898.html>

Scaling BPM Adoption: From Project to Program with IBM Business Process Manager

- <http://www.redbooks.ibm.com/abstracts/sg247973.html>

References

Business Process Management with IBM Business Process Manager

- <http://www.redbooks.ibm.com/abstracts/tips0938.html>

•

The Process Architect: The Smart Role in Business Process Management

- <http://www.redbooks.ibm.com/abstracts/redp4567.html>

Combining Business Process Management and Enterprise Architecture for Better Business Outcomes

- <http://www.redbooks.ibm.com/abstracts/sg247947.html>

BPM for Dummies

- https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=sw-app&S_PKG=bpmebk&S_TACT=109KA6HW&S_CMP=web_ibm_ws_bpm_herolt_bpmbasic

References

Kolban's Book on IBM BPM

- <http://www.neilkolban.com/IBM/>

Leveraging the IBM BPM Coach Framework in Your Organization

- <http://www.redbooks.ibm.com/redbooks.nsf/RedpieceAbstracts/sg248210.html>

IBM Business Process Manager V8.0 Performance Tuning and Best Practices

- <http://www.redbooks.ibm.com/abstracts/redp4935.html>

IBM Business Process Manager Version 8.0 Production Topologies

- <http://www.redbooks.ibm.com/abstracts/sg248135.html>

IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide

- <http://www.redbooks.ibm.com/abstracts/sg247957.html>

Question & Answers



