WebSphere software

IBM

e business software

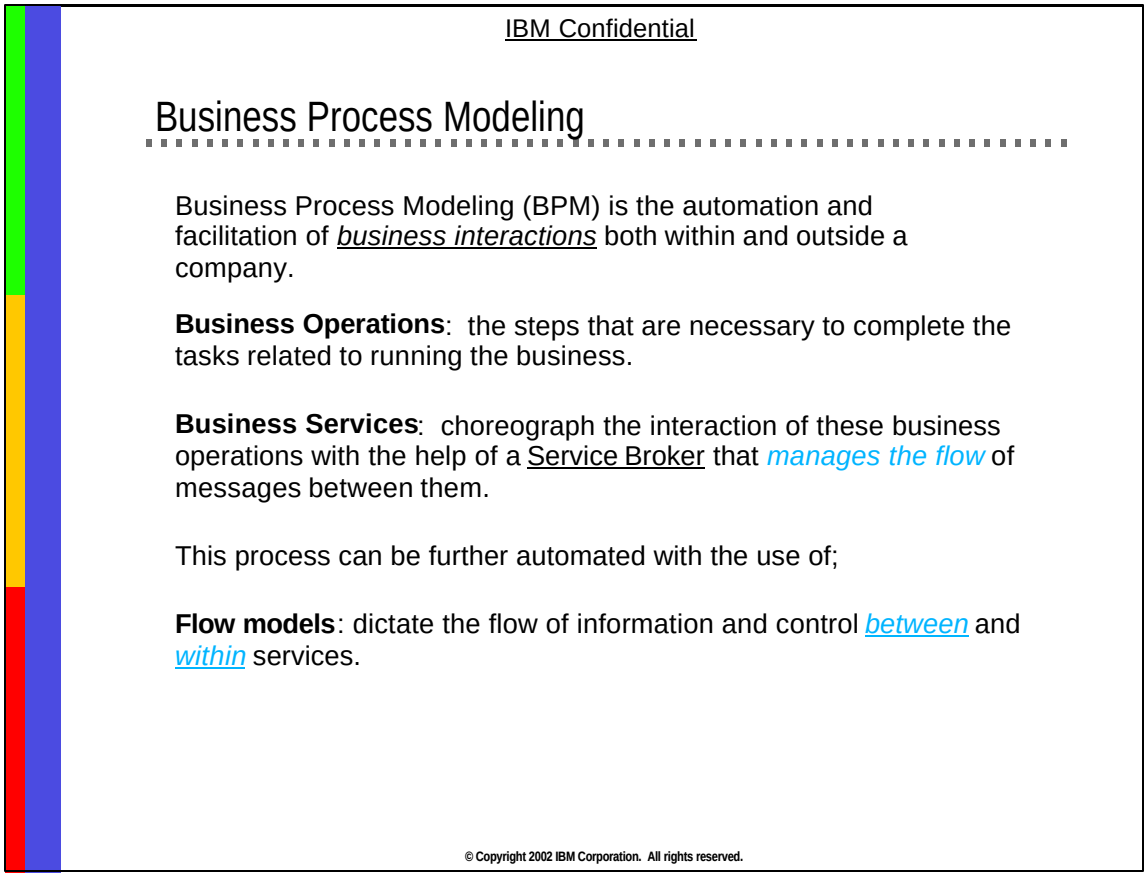
# WSAD IE 5.0 and WebSphere 5.0 Beta

Workflow Concepts

IBM Software Group

## Overview

- Business Process Modeling
- Programming Model
- Runtime Model ( overview )
- External Interfaces
- Elements of a Flow
- Control and Data
- Operational semantics
- Human Interactions
- Kinds of Flows
- Exceptions and Compensation

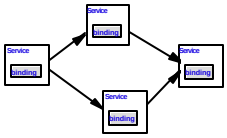



- Emphasis on the Process in "BPM" e.g. interactions.
- This really starts with the corporate information model for the corporation
- The Service Broker is the underlying workflow engine which is part of the WS EE 4.1 runtime.

IBM Confidential

## Flow Model

A flow model is a directed graph with activity nodes and links showing the flow of control and information between them. WebSphere defines two major types of flows:




- **Micro-flows** are short-lived operations composed of business operations that occur *within* a business service.
- **Macro-flows** are long-running process flows that are composed of business services.



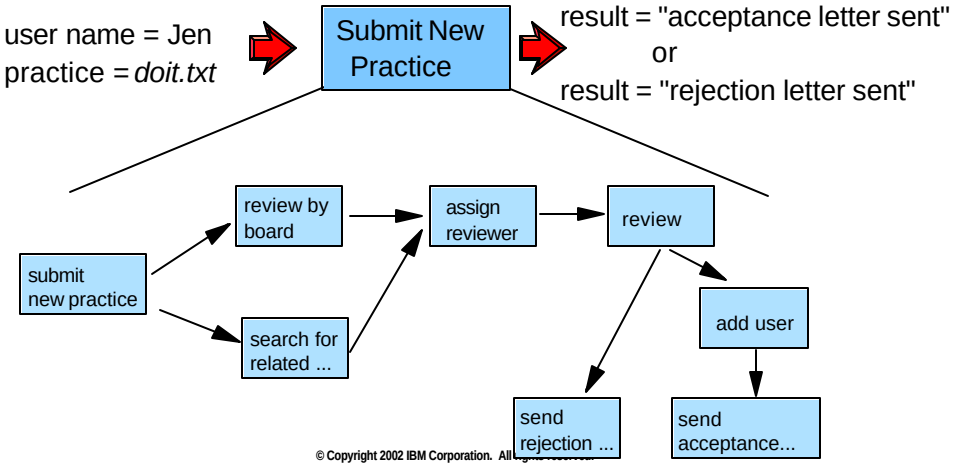
- A directed graph is one with arrows that indicate a specific direction
- Interactions come in both large and small units of work. Some are asynchronous and take a long time and others are synchronous and must be completed within the scope of a single transaction. In the long run we will be able to use WSAD to do both but for this release we will be focusing on the shorter, synchronous kinds and we'll call them micro-flows. The longer running asynchronous kind we'll call workflows. You will not be able to create and work with work flows in this release of the product.
- Work Flows will not be available until WS 5.0. They are mentioned here for the purpose of discussing flows in general and providing a glimpse of the direction we are taking with regard to this functionality.

IBM Confidential

## Example of a Workflow

Business Operation: Review a "Best Practice" for publication on the "Best Practices" website (portal)

Description: A person submits a **WebSphere Best Practice** idea which is reviewed by a member of the "Best Practices" review board and accepted or rejected.





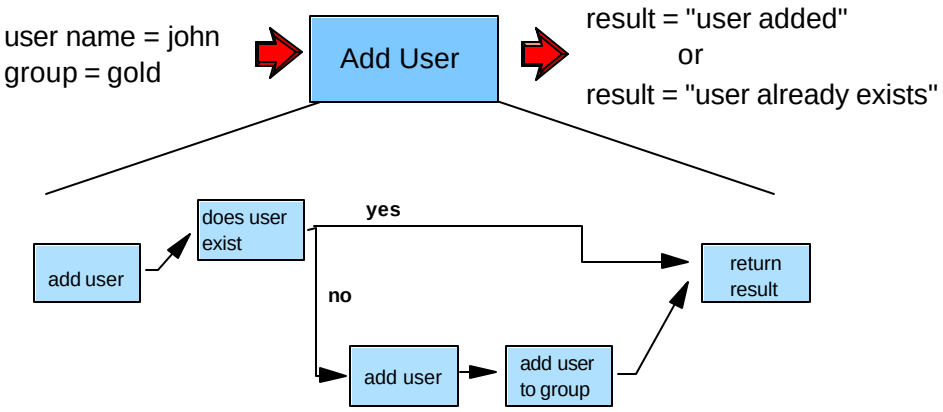

- This is a long running flow that chains several business processes together, also known as macroflow.
- You will see shortly that the 'add user' activity is what we call a microflow.

IBM Confidential

## Example of a Micro-flow

Business Operation: Add a user to the database of authorized users for the on-line research facility.

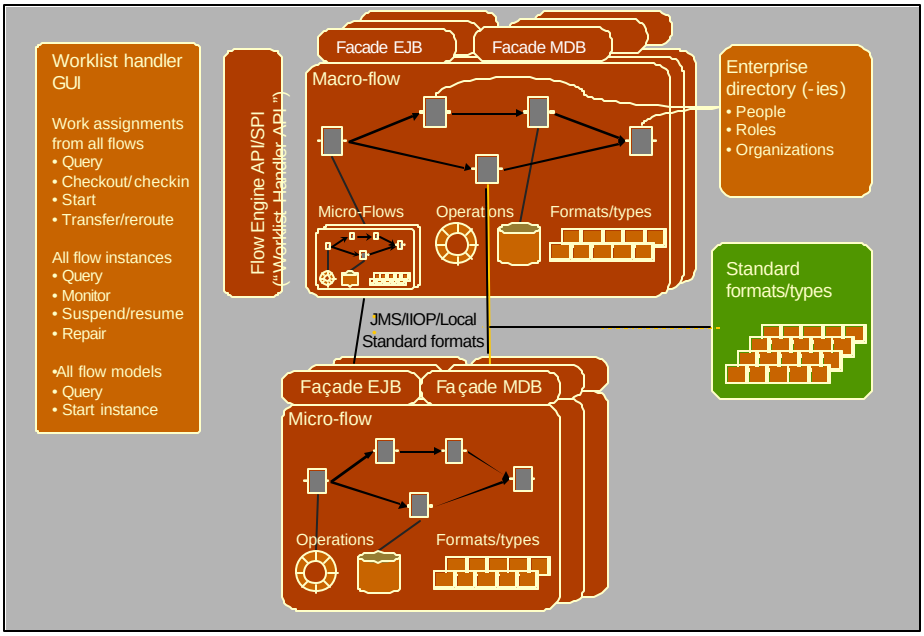
Description: Given the user name and the name of the group to which they belong add them to the database. Check to make sure they have not already been added.






- Other examples of micro-flows would be access to CICS or IMS
- This illustrate the control flow for a simple business operation.

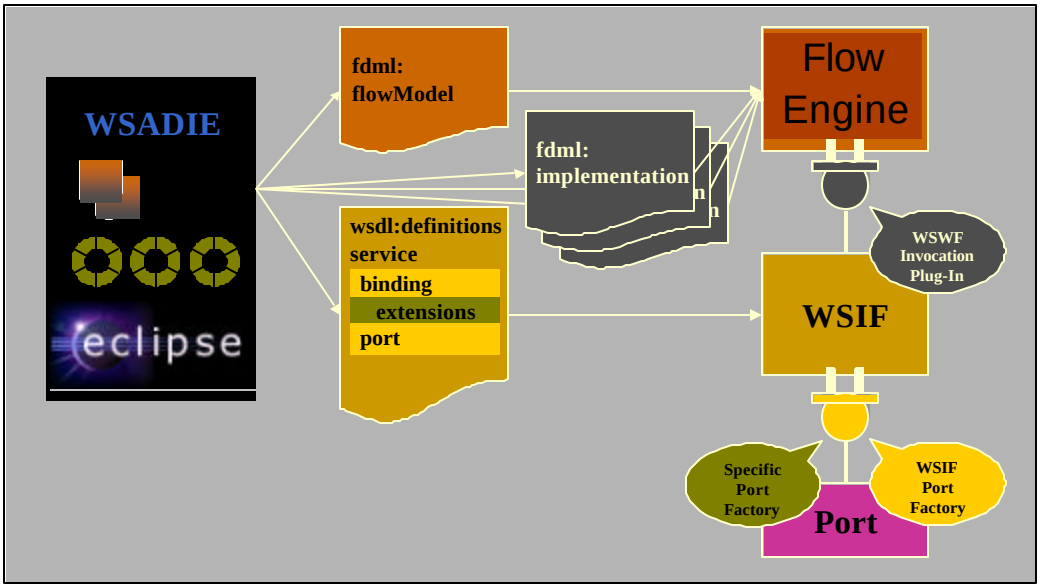
## Programming Model





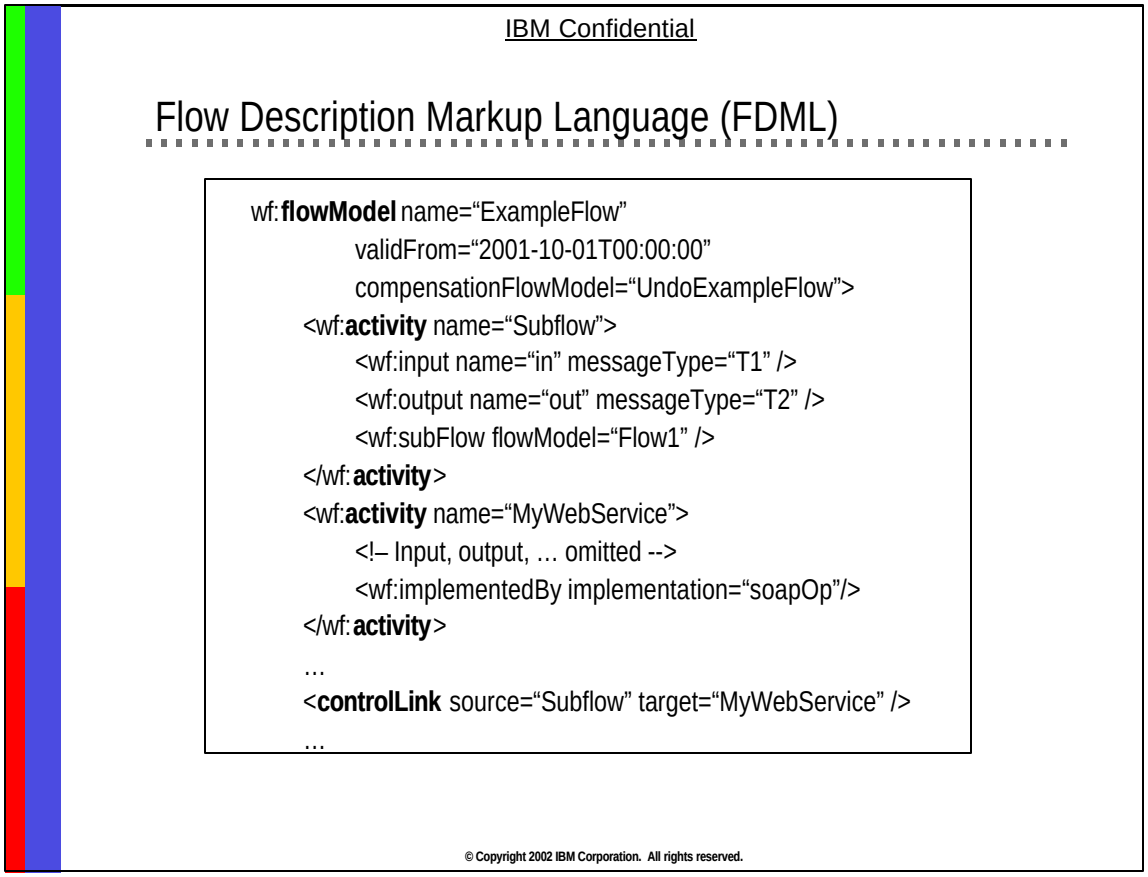

- Asynchronous programming model is supported with the Facade MDB
- Synchronous programming model is supported with the EJB facade
- Interruptible flows are supported with Events and Staff (human interactions)

## WSADIE, Workflow and WSIF





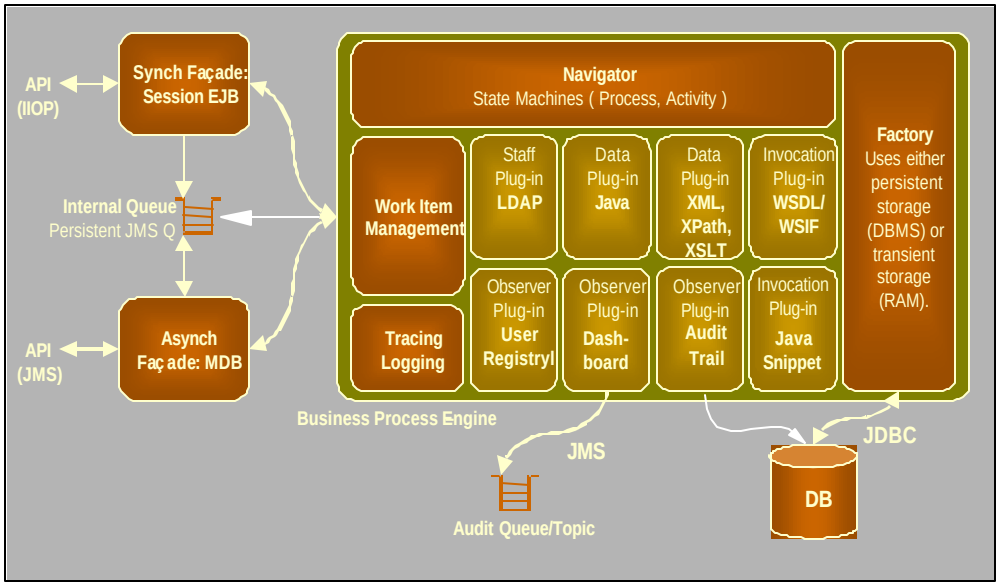

- Wherever you see FDML think of a generalized markup language used to describe the semantic of a flow.
- FDML is an internal implementation detail that is changing.
- We expose it to you hear in the beta because you will see it as you are developing and debugging your applications.




- **Note:** this is an internal detail that will be changing.
- XML language, based on
  - FDL (Flow Definition Language)
  - Flow model-related aspects of WSFL (Web-services flow language)
- Extensible
  - Architected extension points
  - Associated plugin structure for deployment and runtime
- Described by XML schema
  - Sub-schemas for extensions

IBM Confidential

## Flow Container: Internal Structure





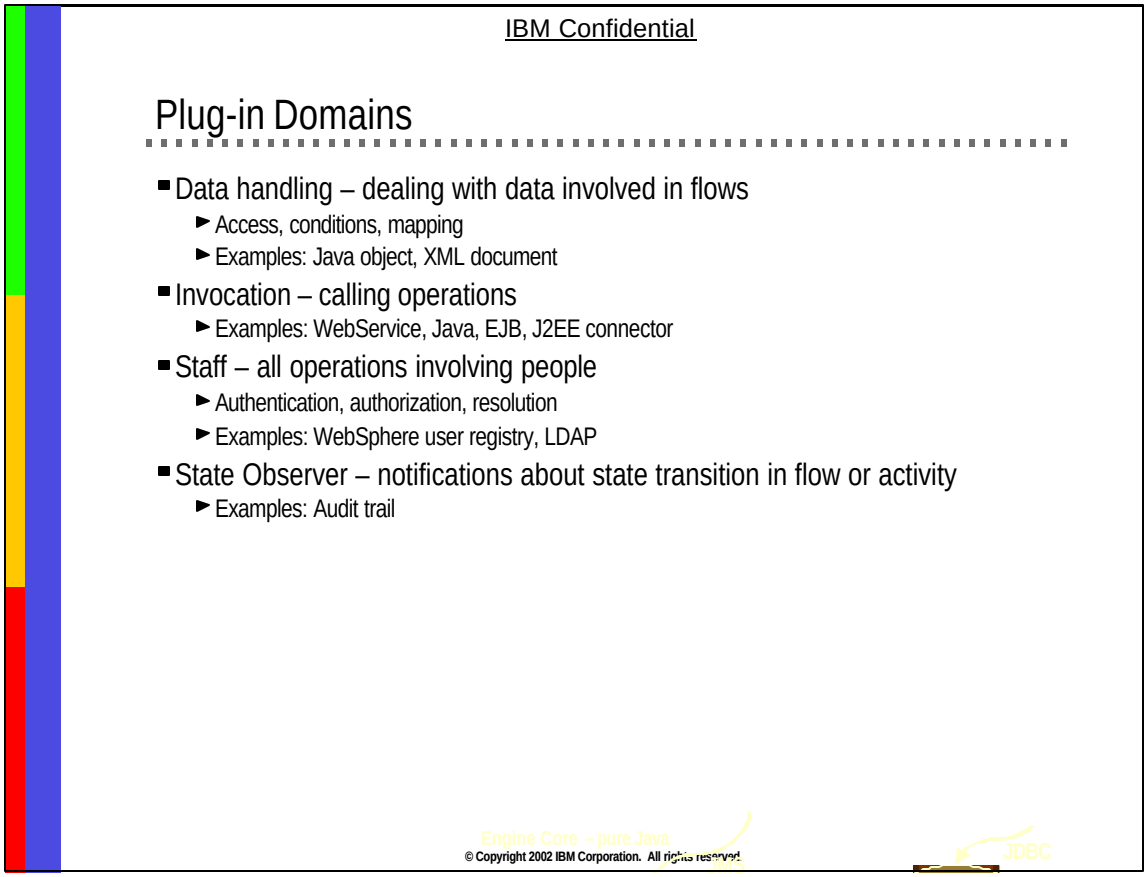

- Remember we are still in the Beta. Everything is subject to change.
- The name Flow Container may change to something like Business Process Container
- The Dashboard plugin will not be available for a while.
- The intent of this picture is to show you that we have an architecture that is designed with extensibility in mind.

# External Interfaces (1)

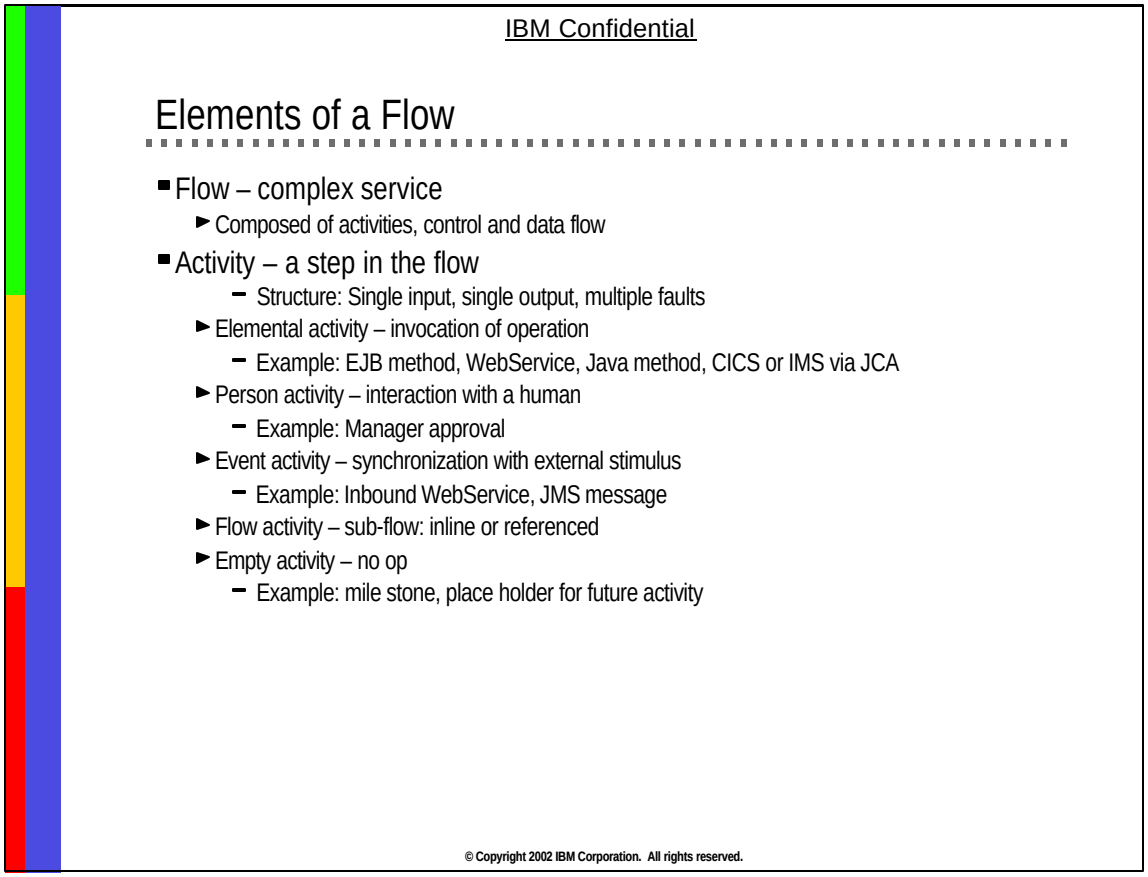
- Flow Level
  - **call** - invoke flow, pass inputs and receive results
    - used for synchronous activation of a flow
    - mico-flows
  - **initiate** - invoke flow, pass inputs, no resutl expected
    - used for asynchronous activation of a flow
    - macro-flow
  - **sendEvent** - pass asynchronous message to a flow instance
  - **terminate**, **suspend**, **resume** - adminstration actions
  - **monitor** - get the current state of execution
- Activity Level
  - **claim** - get exclusive access to a work activity
  - **complete** - let the flow continue from the claimed activity
  - **cancelClaim** - return activity to other potential owners

# External Interfaces (2)

- Work Item level
  - Query work items
  - Retrieve flow instance or activity associated with a work item
  - Transfer work item
- Rendering
  - Methods of stateless session EJBs
    - to be used by clients
  - Pure Java methods
    - To be use by the plug-ins
  - JMS messages
    - To be used by message-based applications
    - Subset only: call, initiate, send event



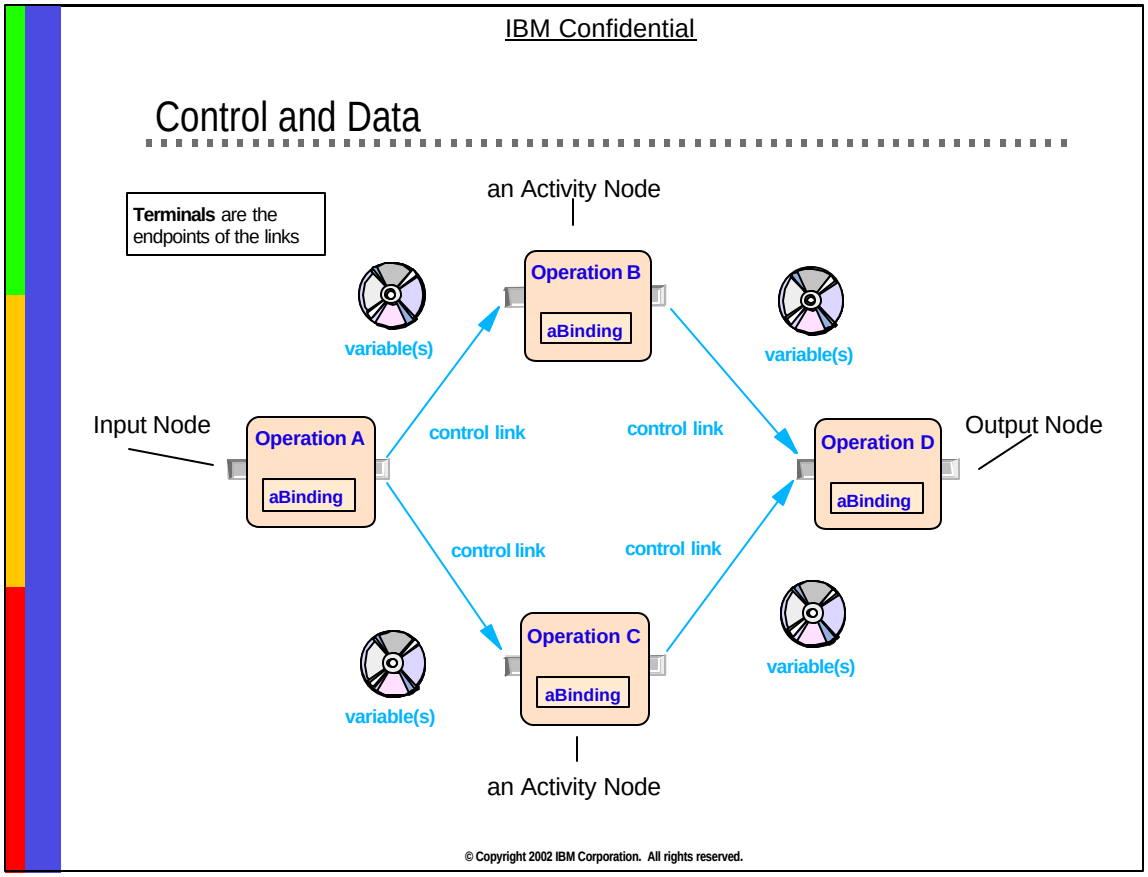

- dashboard is a future featue



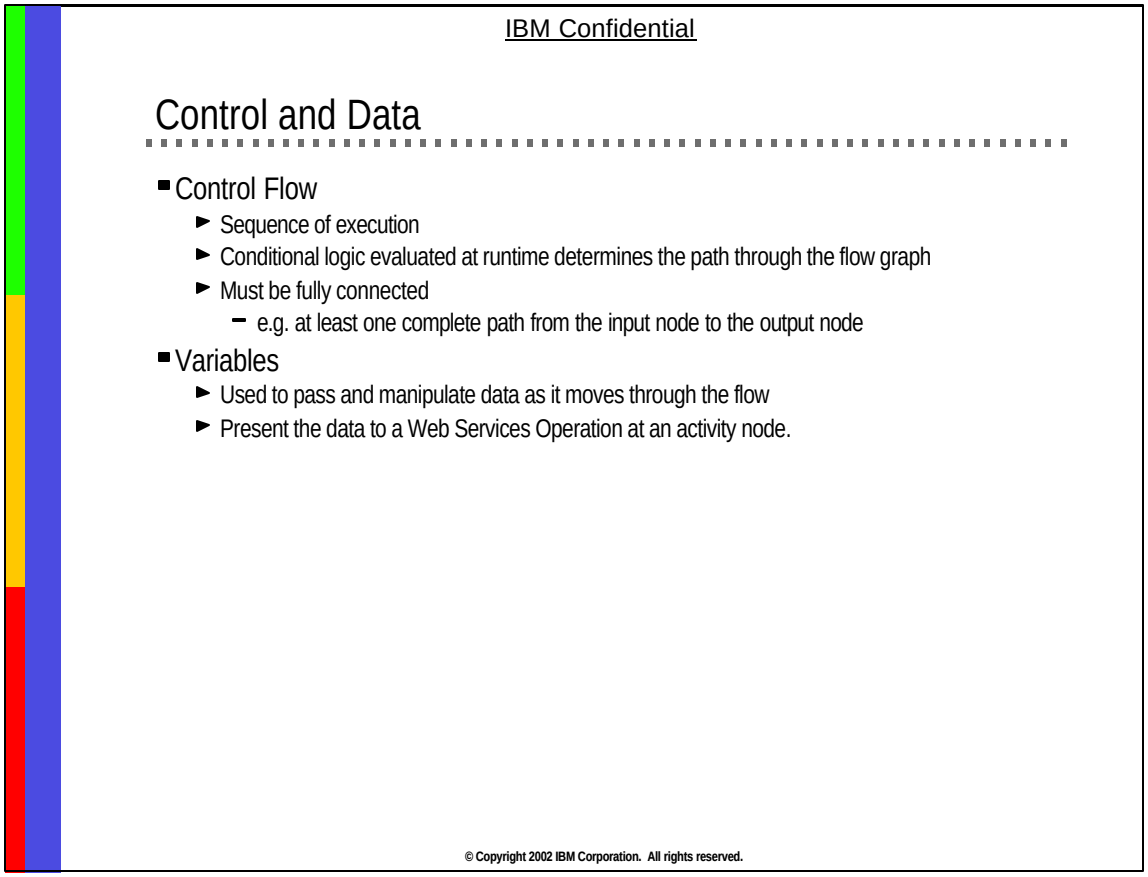

- Elemental Activity - this is where the work really gets done.
- Person Activity - Finding the right person, with the proper qualifications to do the item of work in the Staff Directory - staff resolution.
- Event Activity - waits for some event to be triggered. Events are triggered using the Flow Choreographer APIs.
- Flow Activity - flows can be used in the composition of other flows. Flows can be nested.
- Empty Activity - a noop, a place holder, to be changed to some other kind of activity in the future.

# Elements of a Flow

- Global data context
  - "Global variable" at flow level
- Control Link
  - Prescribes execution order between two activities
  - Carries optional transition condition
- Data Variable
  - Data storage for manipulation and passing.
- Loops
  - Activities can be looped
  - Styles: Repeat-until, While-do
- Message types
  - WSDL message, Java bean
- Conditions
  - Java code
  - Built-ins




- Operations are terms from Web Services
- Activities are what connect the workflow world with the Services World.
- Variables are used in conjunction with inline Java code known as Java snippets, to present input data to the Operations.




- When we model a business process flow there are two dimensions to be considered.
  - Control and Data
  - These are distinct and separate.

- First there is the flow of control. How do you decide which activity to do next?
  - we call this the control flow.

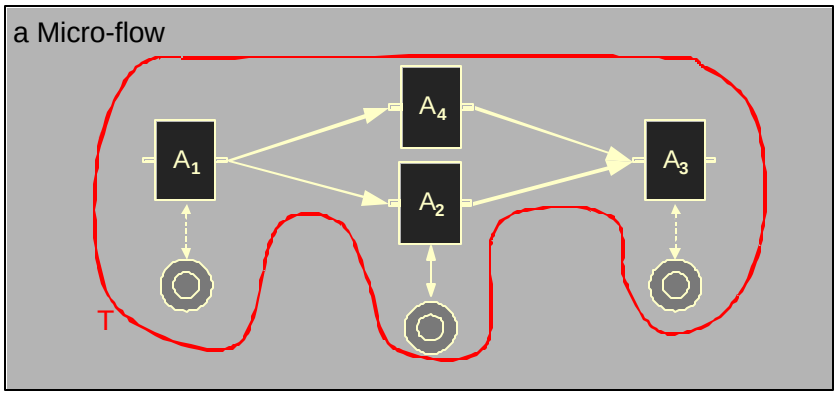
- Second we have to consider what data are required for each activity.
  - we call this the data flow.
  - as we flow from one node to the next, the data types may need to be converted or data structures may need to be merged (joined). In other words, the data may need to be transformed as it moves from node N1 to node N2. To accommodate this we have Data Mapping nodes

- Business Operations:
  - the steps that are necessary to complete the tasks related to running the business.
  - Business operations are defined within the context of Services. A Service may have many operations.

IBM Confidential

## Operational Semantics: Transactional Behavior

- Uninterrupted flow (“atomic sphere”)
  - Single transaction from start to end
  - Only synchronous invocations
  - Short lived
  - Invocations tied into single transaction (security context, transaction context are passed)



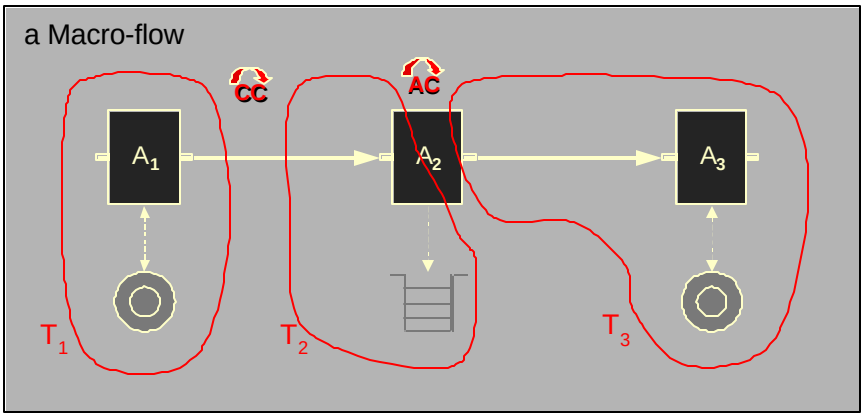



- The WebSphere flow engine does not establish it's own transactional semantics, but participate in the standard WebSphere transactions.
- For microflows, this is the transaction established by the client.

IBM Confidential

## Operational Semantics: Transactional Behavior

- Transactions connected via messages
  - Establishes new “MDB transaction”
- Types of messages:
  - “Continue control connector” (CC) for synchronous invocations
  - “Activity complete” (AC) in asynchronous case



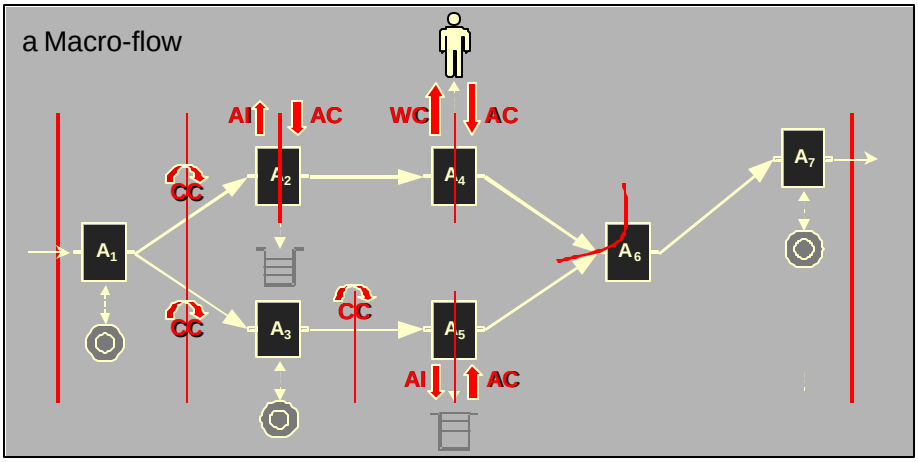



- For macroflows there is the initial transaction established by the client where triggering the flow
- plus the a number of chained transactions triggered by JMS messages, either internal or external to the flow engine.
- Transactions are chained together by
  - 1.The "Continue Control" message is an internal message generated by the flow engine to commit T1 and guarantee continuation of the flow with T2.
  - 2. JMS messages
  - 3. Human interaction with a workitem.
- When the AI message is placed on the outgoing queue the the flow engine must commit T2 for the message to be delivered.

## Operational Semantics: Transactional Behavior

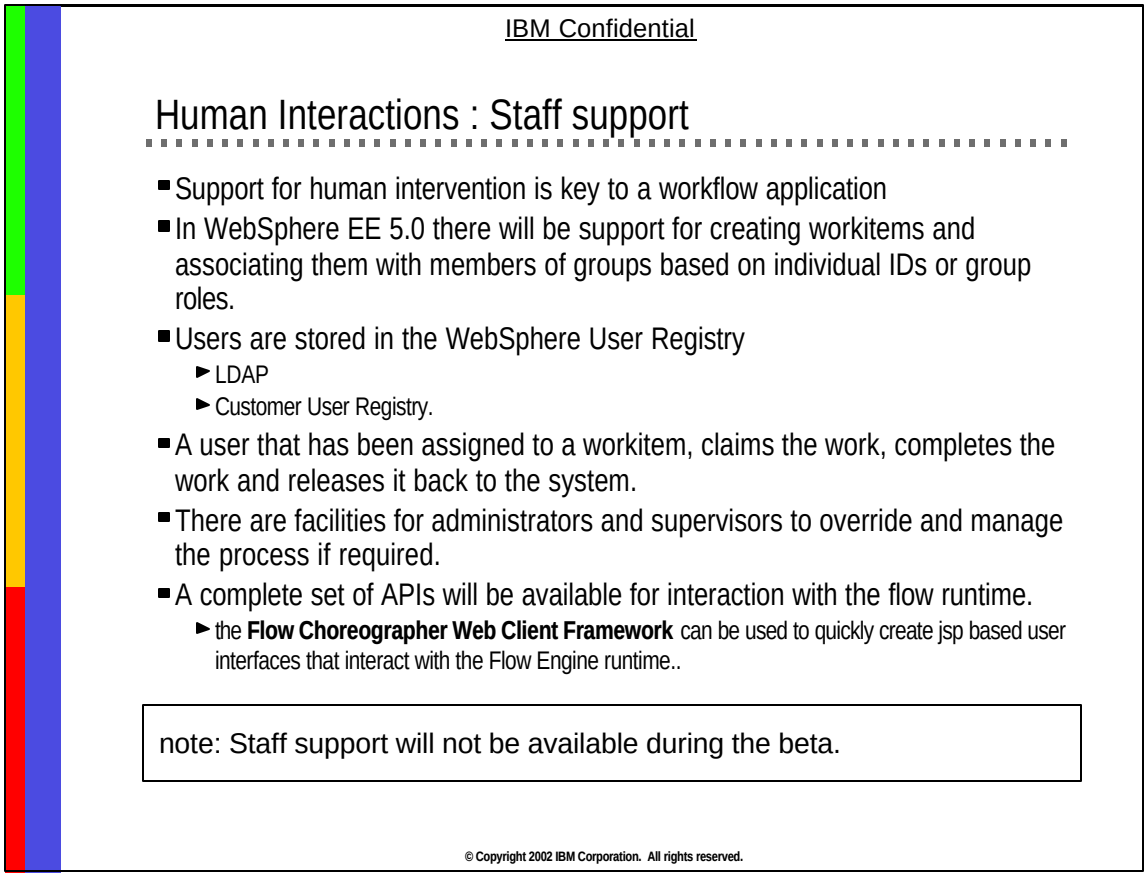
- More complex example

CC – continue control connector
AI – activity invoke
WC – workitem created
AC – activity complete





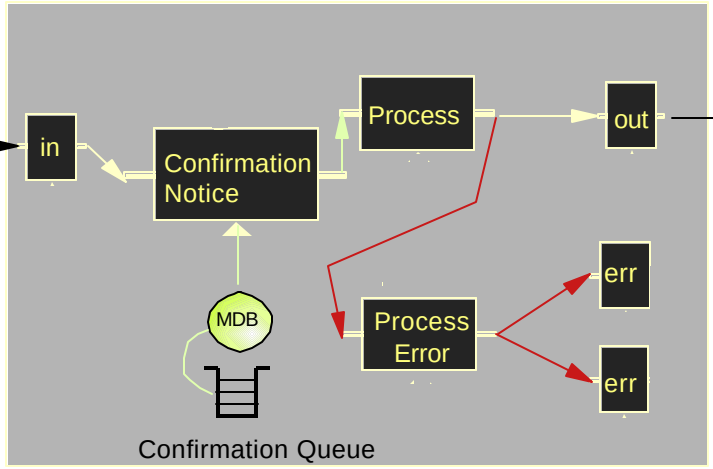

- The Work Item create (WC) message is placed on the queue before committing the transaction.
- When the person has completed the workitem, the Activity Complete (AC) message is sent and the message is picked up and the flow continues.

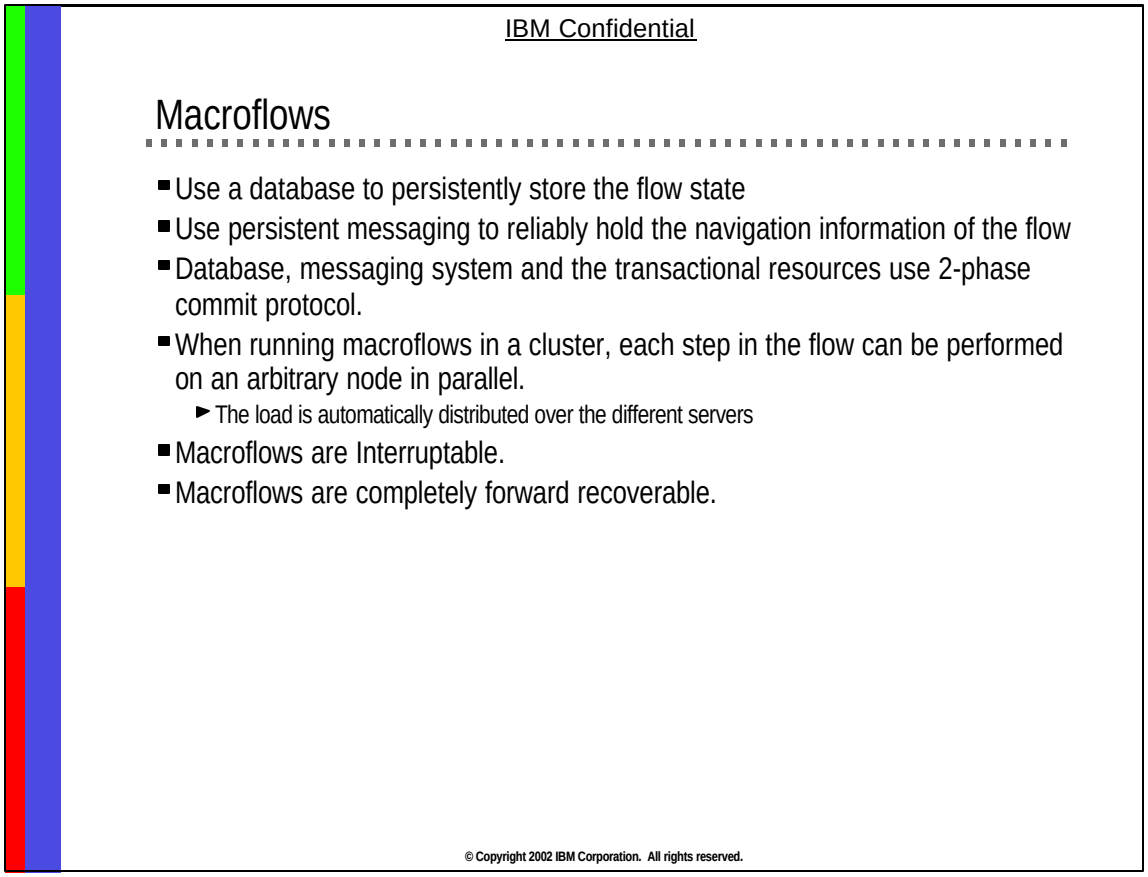



- More on the the Web Client Framework in the runtime section.
- The **Web client framework** is based on the established J2EE pattern, "Model-View-Controller" pattern. Its main components are:
  - WorkflowController servlet and its configuration component
  - WorkflowView JSPs - a set of default JSPs for displaying typical flow UIs
  - Utility objects and methods for accessing actual content and for referencing and executing flow actions
- 
- **Note**: this is a runtime feature and not part of WSAD IE; for more information see the WAS Infocenter.

# Events

- Events represent another form of Interruptable flows.
- With an event the flow will wait until a message is received via a Message Driven Bean (MDB).
- When the message is received the flow is resumed and the message routed appropriately.

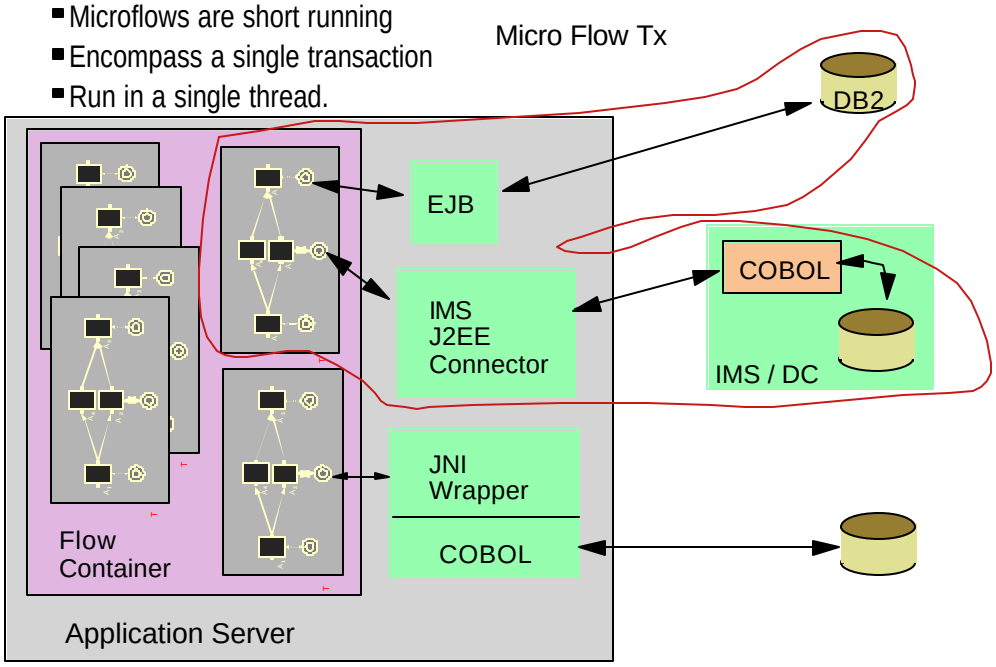


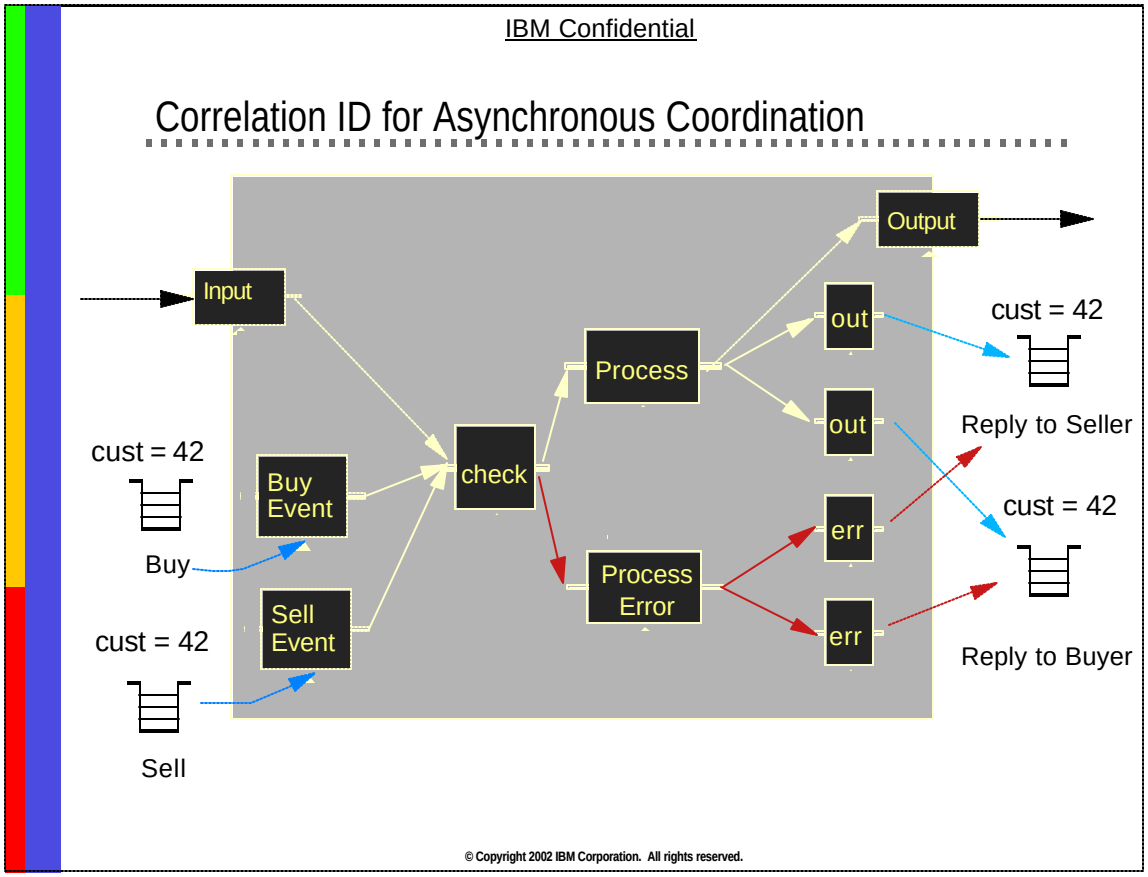



- If the app server currently processing a macroflow unexpectedly terminates, no information is lost.
- If in a clustered environment, another server picks up the work.
- If in a non-clustered environment, then the work will be resumed when the server comes back on line.

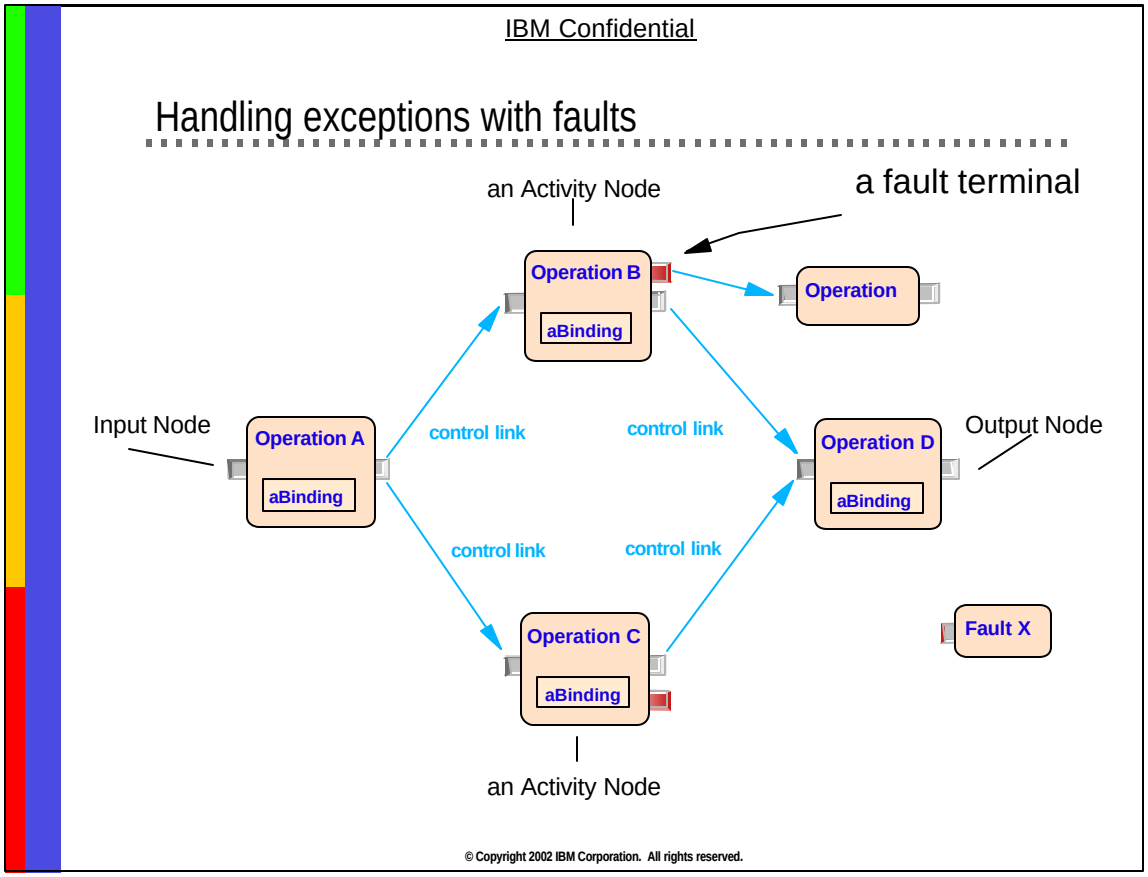
# Microflows

- Microflows are short running
- Encompass a single transaction
- Run in a single thread.






- Asynchronous input via MDB facade
- Settle trade –
  - incoming sell and buy messages are correlated by trade ID.
  - First one to arrive creates new process instance, second is correlated to this instance.
  - Correlation requires statefulness.
- The Correlation ID is must be composed of data unique to the transaction, such as the contract number or purchase order.

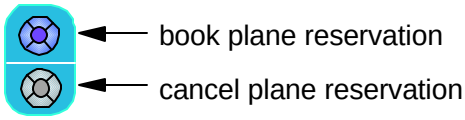



- Faults are used as you would use your Java exceptions
- Primarily to manage exceptions with your business logic
- The fault terminal can be wired directly to a fault node or another Business Operation or a Java Snippet.
- The fault terminal does not have to be wired at all.
  - The Business Operation can associate with a fault node via it'c configuration
    - this saves on wiring clutter.

IBM Confidential

## Compensation

- What do you do when you need to undo your process, because the order was canceled or the loan was rejected or you manager didn't approve the travel request?
  - the order has been on the books (in the database) for a week
  - the account was setup in anticipation
  - the reservations were made yesterday because there was a 7 day window for a deal.
- All of these share the characteristic of spanning multiple operations over time and the operations that need to be undone are outside the scope of the traditional transaction boundary. ( e.g. they were already committed )
- The mechanism for handling this kind of situation is called **compensation**
- With compensation, for a given business service, you define an 'undo' business service and these two services are associated into a pair called a **Compensation Pair**.



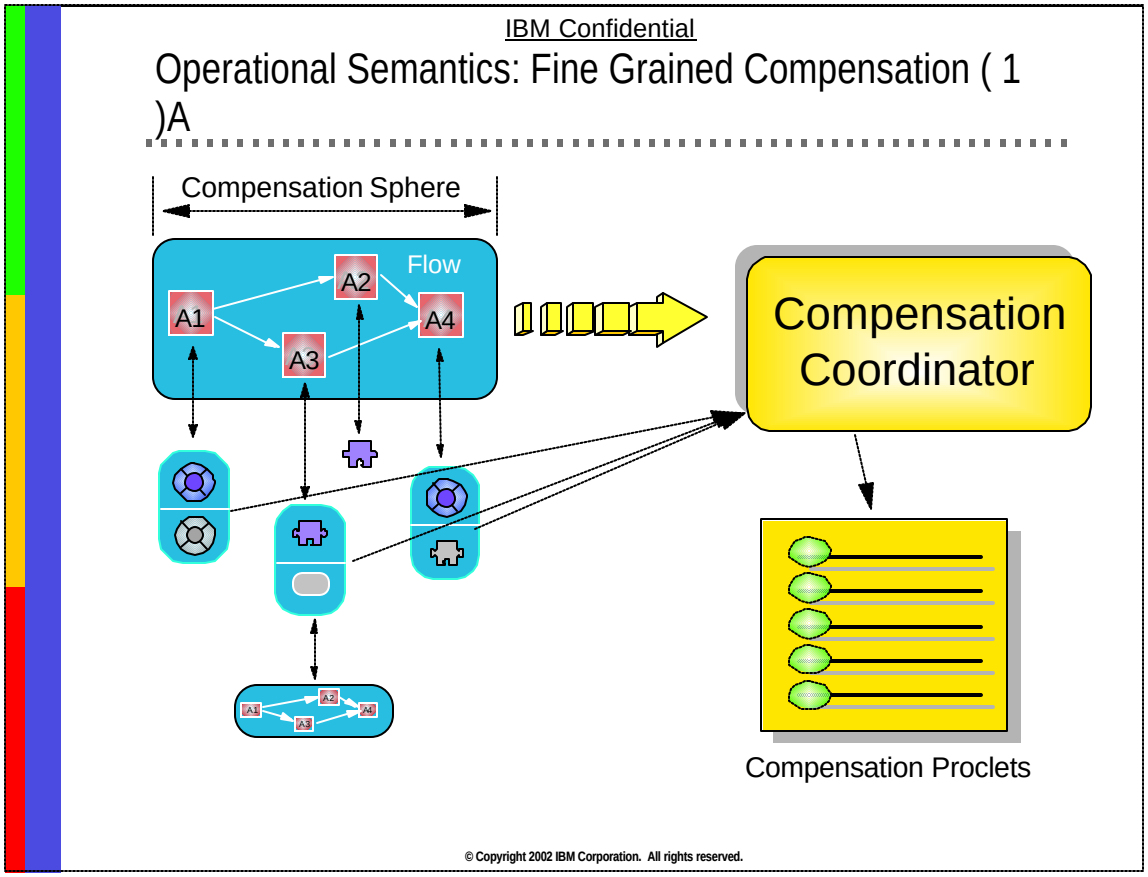



- Your business process says you have to go back and start from a different point.
- This is for macro-flows only

# Compensation Flow

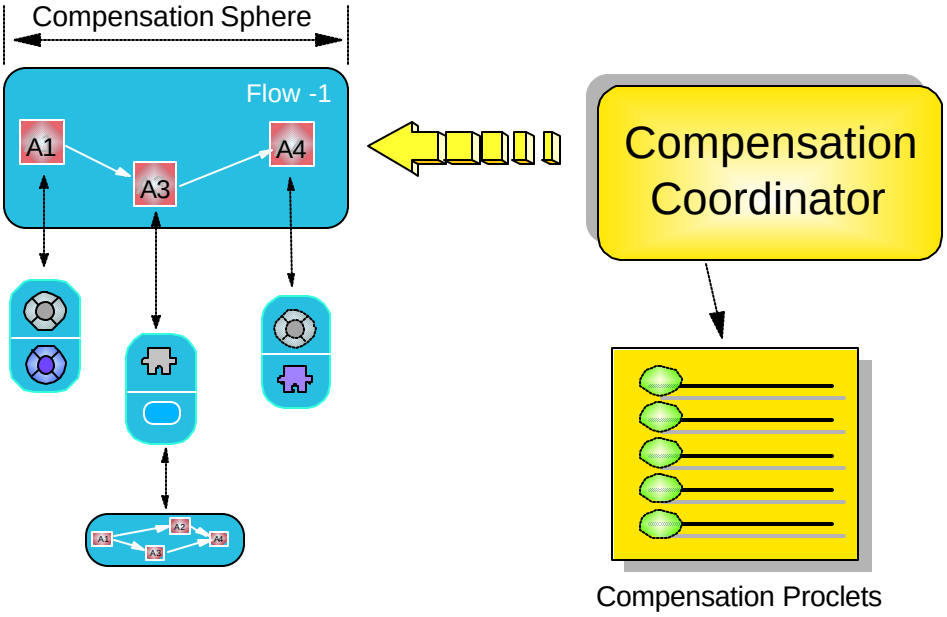
- A Flow may have associated compensating activities
  - Compensation is triggered by
    - Uncaught top-level exception
    - Explicitly via request - I want an example of this
  - Compensation results in the execution of the compensating activities in reverse order
    - Context data passed as input
- Compensation using compensation pairs as activity implementations
  - Perform logging during forward navigation
  - Associated compensation sphere, can be “rolled-back”

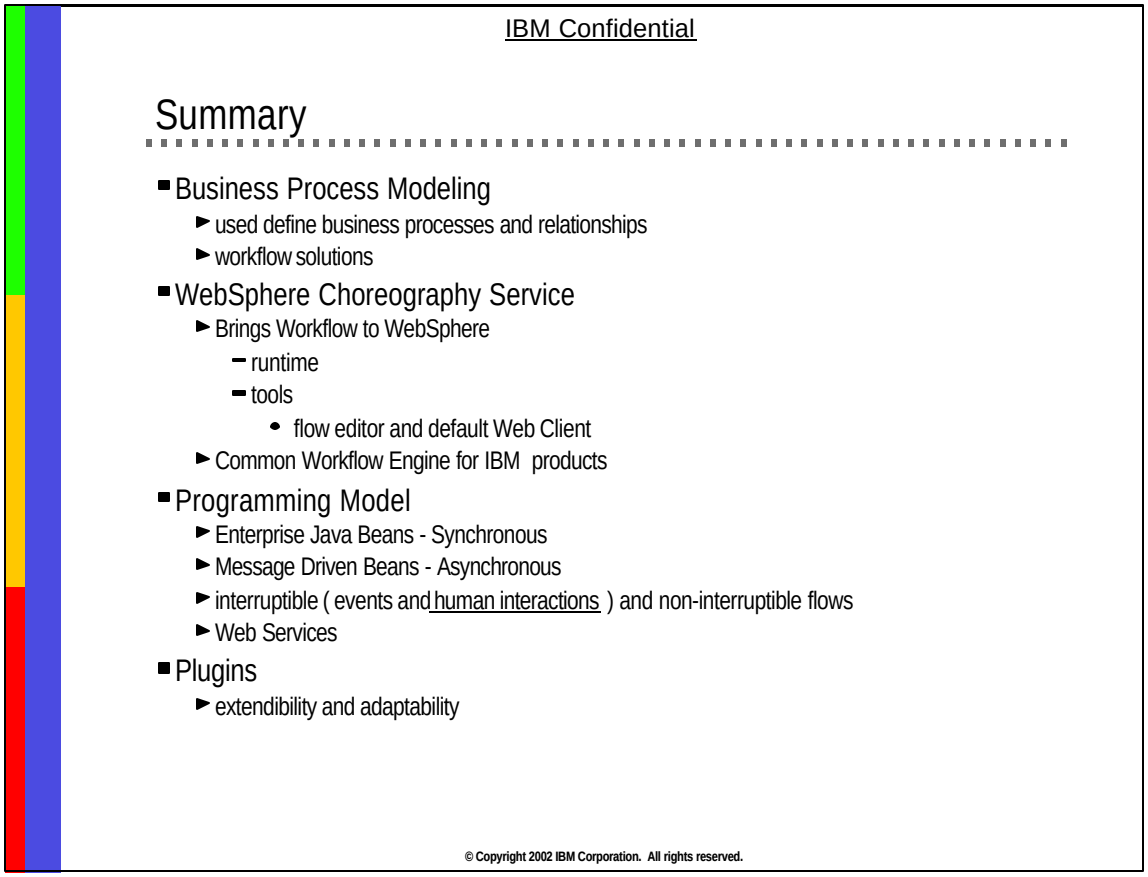
note: Compensation will not be available during the beta.




- Each compensation pair runs its primary operation then registers with the compensation coordinator, creating a proclet to be used if compensation is needed.
- When a flow is started (as a compensation scope provider), it creates a new compensation coordinator to start a compensation sphere.
- Each compensation pair called as a node of the flow registers with the compensation coordinator, runs its primary operation, and stores the information needed to run its compensation operation if the flow needs to be compensated.
- When the flow ends it closes the compensation scope, and the compensation coordinator checks to see if any proclets have been registered and are interested in the need for compensation. The flow decides whether to accept all of the updates made by primary operations or to run the compensation operation of each compensation pair:

# Operational Semantics: Fine Grained Compensation ( 1 )B






- Key Concepts:
  - Compensation
  - Human interaction ( Staff )
  - Interruptible flows
  - Asynch/Synch Programming models
  - Separation of control and function