webMethods
Integration Workshop – Day 2

Recap – Day1

Learnt So Far (Recap & assessment)

- webMethods Overview
- Integration Challenges
- Using SoftwareAG Designer
- Perspectives, Server, Packages, Locks & ACL

Introduction

- This course introduces to basic webMethods DataTypes, FlowServices and flow steps.

- Participants will understand the list of webMethods DataTypes and their usage.

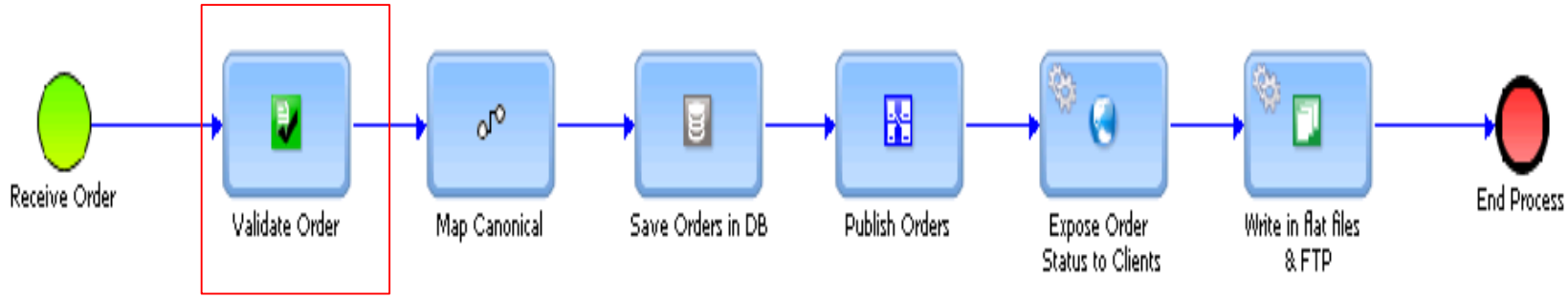
- Participants also learn about creation of flow services

- Participants will be able to simulate the basic FlowSteps (Map, Invoke and Branch) after this course.

Objectives

- Describe the different DataTypes and services in webMethods
- Get hands on experience in creation of flow service
- Basic coding standards, i/o signature.
- To know about basic flow steps – MAP, INVOKE and BRANCH.

webMethods Pilot Project Progress



Outcome of this course :

Trainees should be able to create a flow service which does the validation of the OrderCustomer information as part of Practical session

Software versions

- This class focuses on the webMethods suite

webMethods Integration Server

Broker /UM

Software AG Designer

Chapters

Day 2

- webMethods Datatypes
- Introduction to Services
- FlowServices
- Documents
- Flow Step - MAP
- Flow Step - INVOKE
- FlowStep - BRANCH

Chapters

Day 2

- Flow Step - SEQUENCE
- Flow Step - LOOP
- FlowStep - REPEAT
- FlowStep - EXIT

Basic Coding Standards

- Package names should always be capitalized. If a name is composed of many words, the first letter of each word must be capitalized.

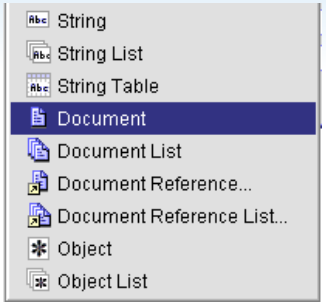
- webMethods node objects cannot contain:
  - Reserved words and characters that are used in Java or C/C++ (such as for, while, and if)
  - Digits as their first character & Spaces
  - Control characters and special characters like periods (.), including:
    ' - # =) (. / \ ; & @ ^ ! | } { ` > < % * : $] [" + , ~

- Create only one top-level folder in each package.

- No folder should be created outside the package base folder

- Service names should start with a verb, in lowercase, and any subsequent objects in a service name should be capitalized (Examples: postNewHireToPeopleSoft, writeFile)

webMethods DataTypes



String	java.lang.String
String List	java.lang.String [] – a string array
String Table	Two-dimensional String array
Document	Structure containing various data types IData
Document List	Same as Document, but an array IData []
Object	Doesn't fit one of the already listed types Any subclass of java.lang.Object (java.util.InputStream)
Object List	Same as object, but array type. e.g. java.util.InputStream [ ]

Introduction to Services

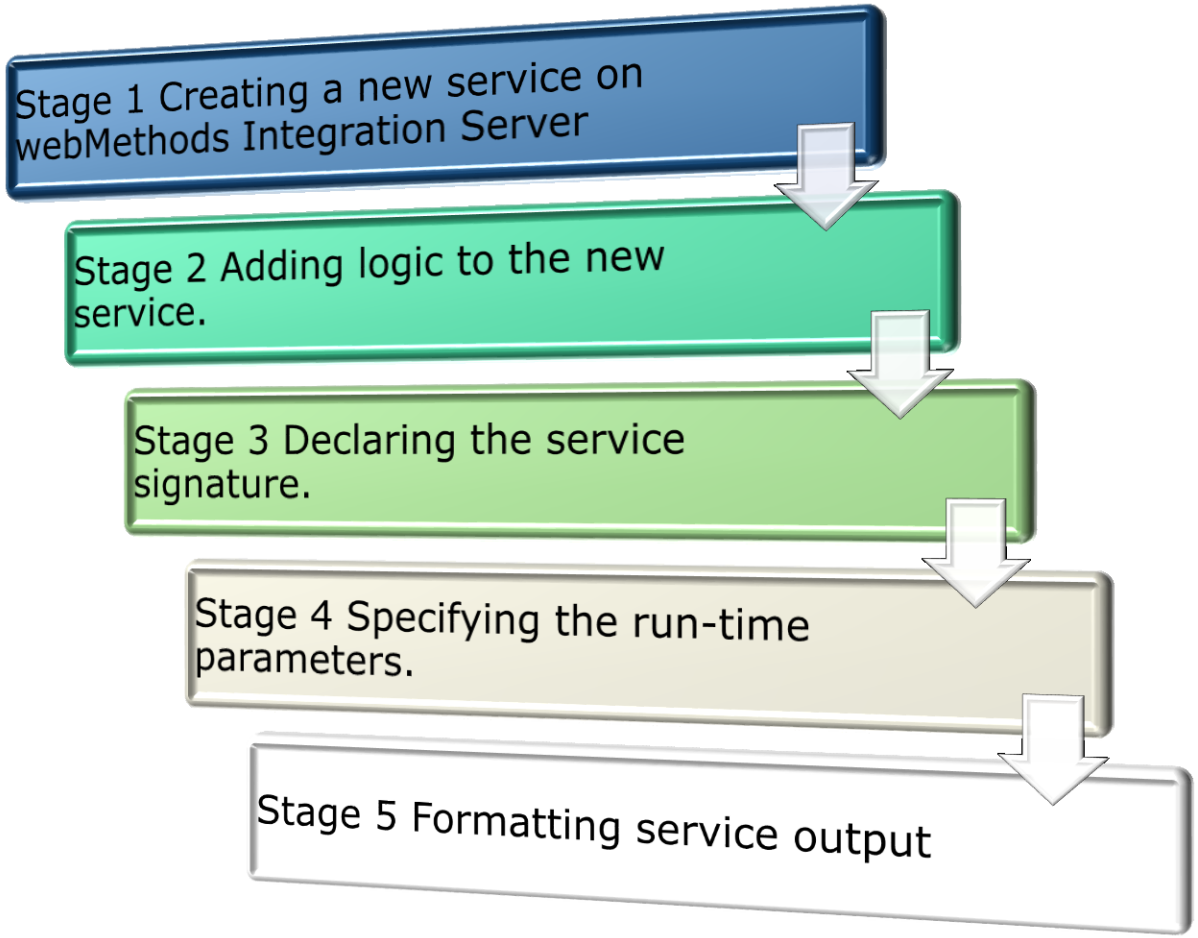
- Services are method-like units of logic that operate on documents. They are executed by Integration Server.

- You build services to carry out work such as extracting data from documents, interacting with back-end resources (for example, submitting a query to a database or executing a transaction on a mainframe computer), and publishing documents to the Broker.

- Integration Server is installed with an extensive library of built-in services for performing common integration tasks.

- webMethods graphical implementation language, flow, enables you to quickly aggregate services into powerful sequences called flow services.

Service Development - Process



Flow Services

- A flow service is a service that is written in the webMethods flow language.

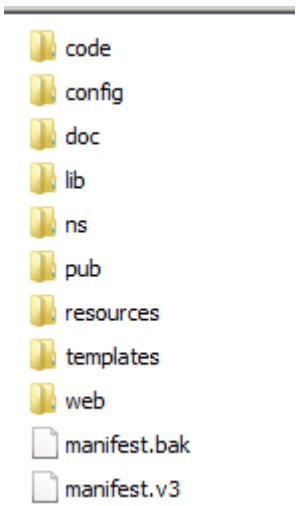
- This simple yet powerful language lets you encapsulate a sequence of services within a single service and manage the flow of data among them..

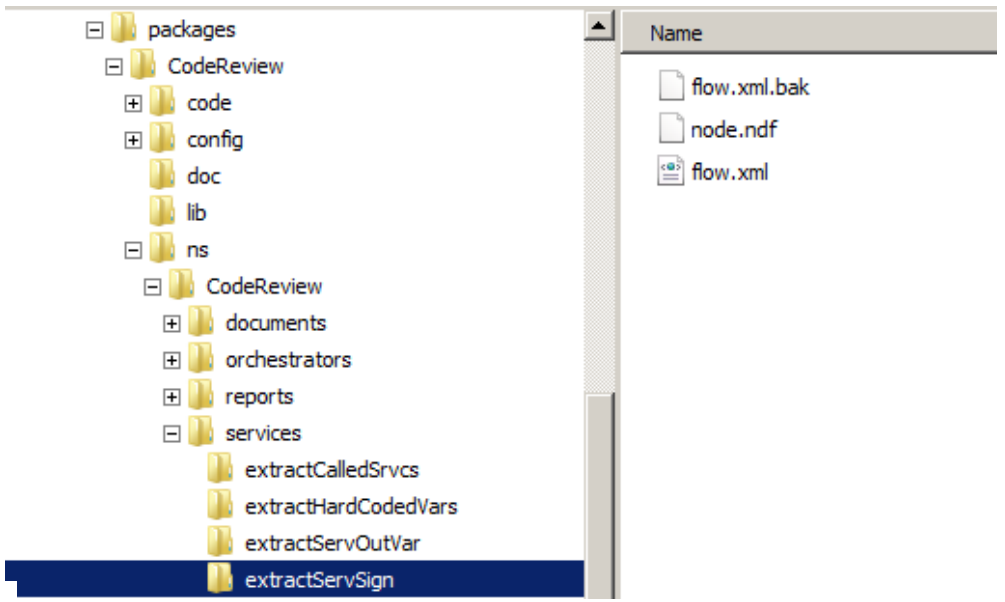
- Any service can be invoked within a flow (including other flow services). For instance, a flow might invoke a service that you create, any of the built-in services provided with the Integration Server, and/or services from a webMethods add-on product such as the webMethods JDBC Adapter.

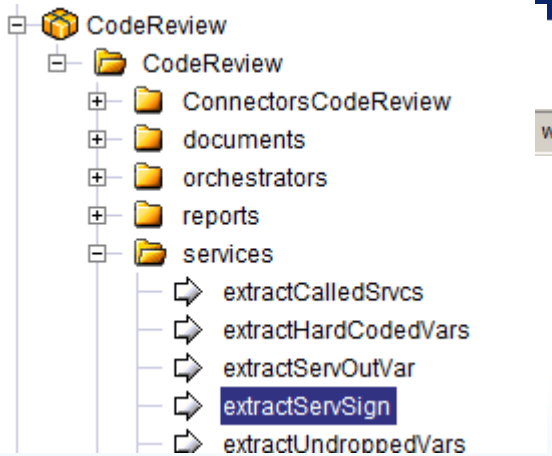
- You create flow services using Designer. They are saved in XML files on Integration Server.

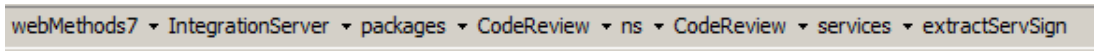
- Flow services are written as XML files in a format that is understood by Designer. You cannot create or edit a flow service with a text editor.

Flow Services Storage





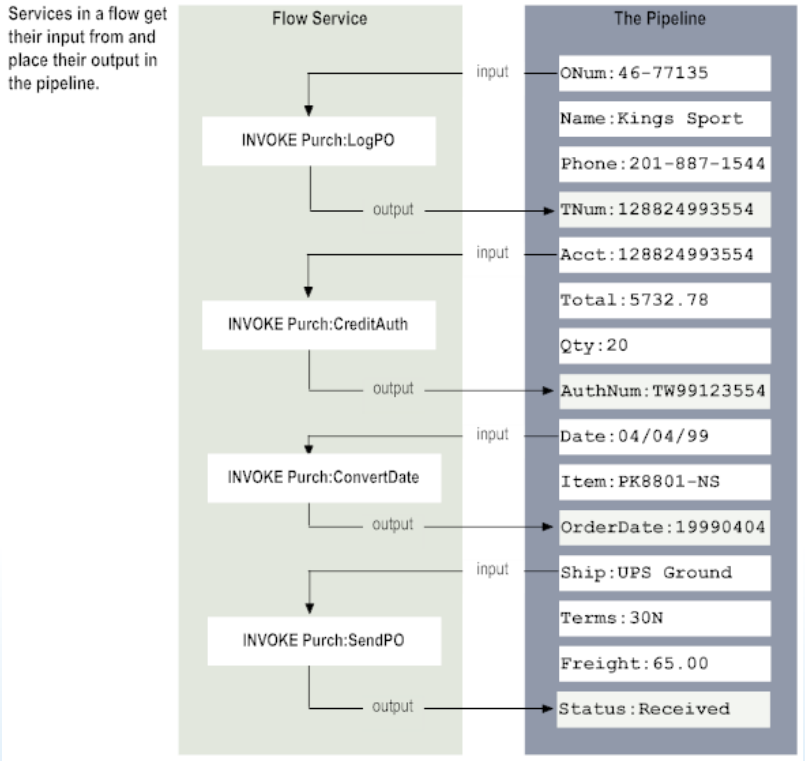




What is Pipeline

- The pipeline is the general term used to refer to the data structure in which input and output values are maintained for a flow service. It allows services in the flow to share data.

- The pipeline starts with the input to the flow service and collects inputs and outputs from subsequent services in the flow. When a service in the flow executes, it has access to all data in the pipeline at that point.



FlowStep

- A flow service contains flow steps. A flow step is a basic unit of work (expressed in the webMethods flow language) that webMethods Integration Server interprets and executes at run time.

- The webMethods flow language provides flow steps that invoke services and flow steps that let you edit data in the pipeline.

- webMethods flow language also provides a set of control steps that allow you to direct the execution of a flow service at run time.

- Control Steps are used to direct the execution of the flow. The flow steps are categorized into Invocation steps & Data – Handling Steps.

- Invocation step – INVOKE

- Data–Handling step - MAP

- Control steps - BRANCH, LOOP, REPEAT, SEQUENCE, EXIT

FlowSteps – Continued..

- **Invocation Steps** INVOKE - Executes a specified service. For more information about this step.

- **Data-Handling Steps** MAP - Performs specified editing operations on the pipeline (such as mapping variables in the pipeline, adding variables to the pipeline, and dropping variables from the pipeline).

- **Control Steps** BRANCH - Executes a specified flow step based on the value of a specified variable in the pipeline. For more information

- LOOP - Executes a set of flow steps once for each element in a specified array.

- REPEAT - Re-executes a set of flow steps up to a specified number of times based on the successful or non-successful completion of the set.

- SEQUENCE Groups a set of flow steps into a series. The SEQUENCE step is implicit in most flow services

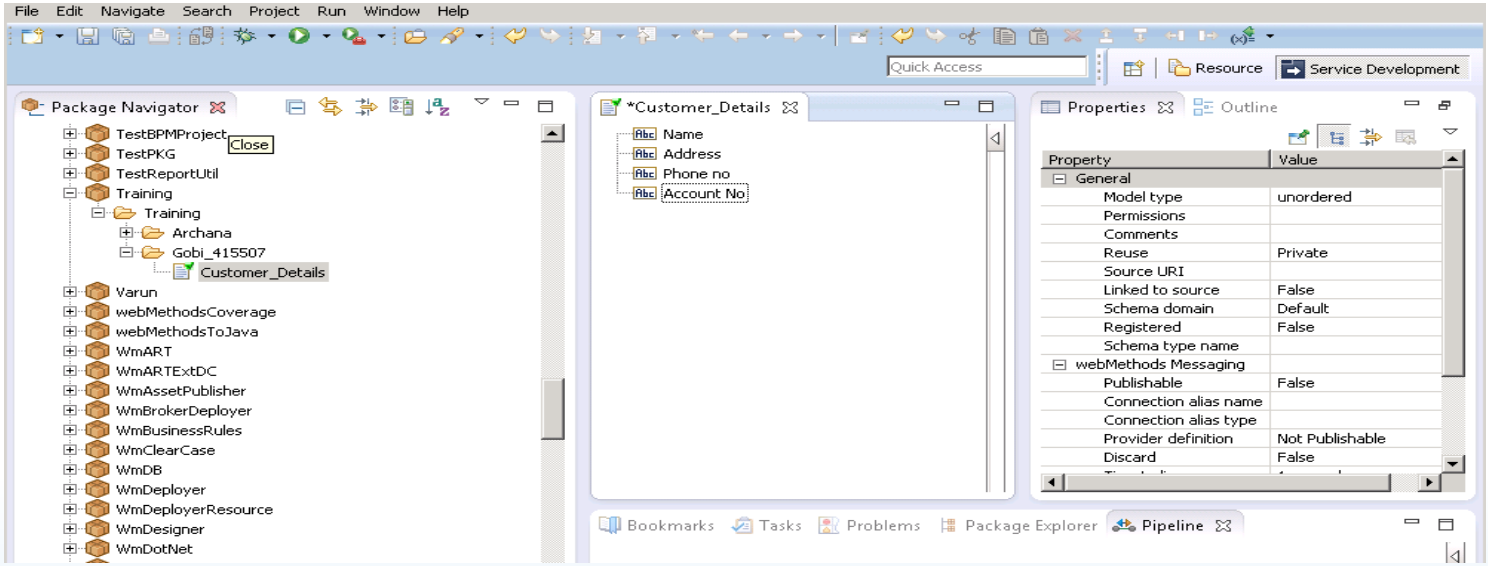
- EXIT Controls the execution of a flow step

Documents

- Documents are objects that webMethods components use to encapsulate and exchange data.

- A document represents the body of data that a resource passes to webMethods components.

- In an integration solution built on the publish-and-subscribe model, applications publish and subscribe to documents.



Canonical Documents

- A canonical document is a standardized representation that a document might assume while it is passing through your webMethods system.
- A canonical document acts as the intermediary data format between resources.
- By converting a document to a neutral intermediate format, subscribers (such as adapter services) only need to know how to convert the canonical document to the required application format.
- If canonical documents were not used, every subscriber would have to be able to decode the native document format of every publisher.
- In flow services, you can map documents from the native format of an application to the canonical format.

Flow Step - MAP

- The MAP step lets you adjust the contents of the pipeline at any point in a flow service. When you build a MAP step, you can:

- Prepare the pipeline for use by a subsequent step in the flow service by linking, adding, and dropping variables in the pipeline.

- Clean up the pipeline after a preceding step by removing fields that the step added but are not needed by subsequent steps.

- Move variables or assign values to variables in the pipeline.

- Initialize the input values for a flow service.

- Invoke several services (transformers) in a single step.

- Map documents form one format to another. For example, you can map a document in an XML format to an another format or a proprietary format.

Mapping Data in Flow Services

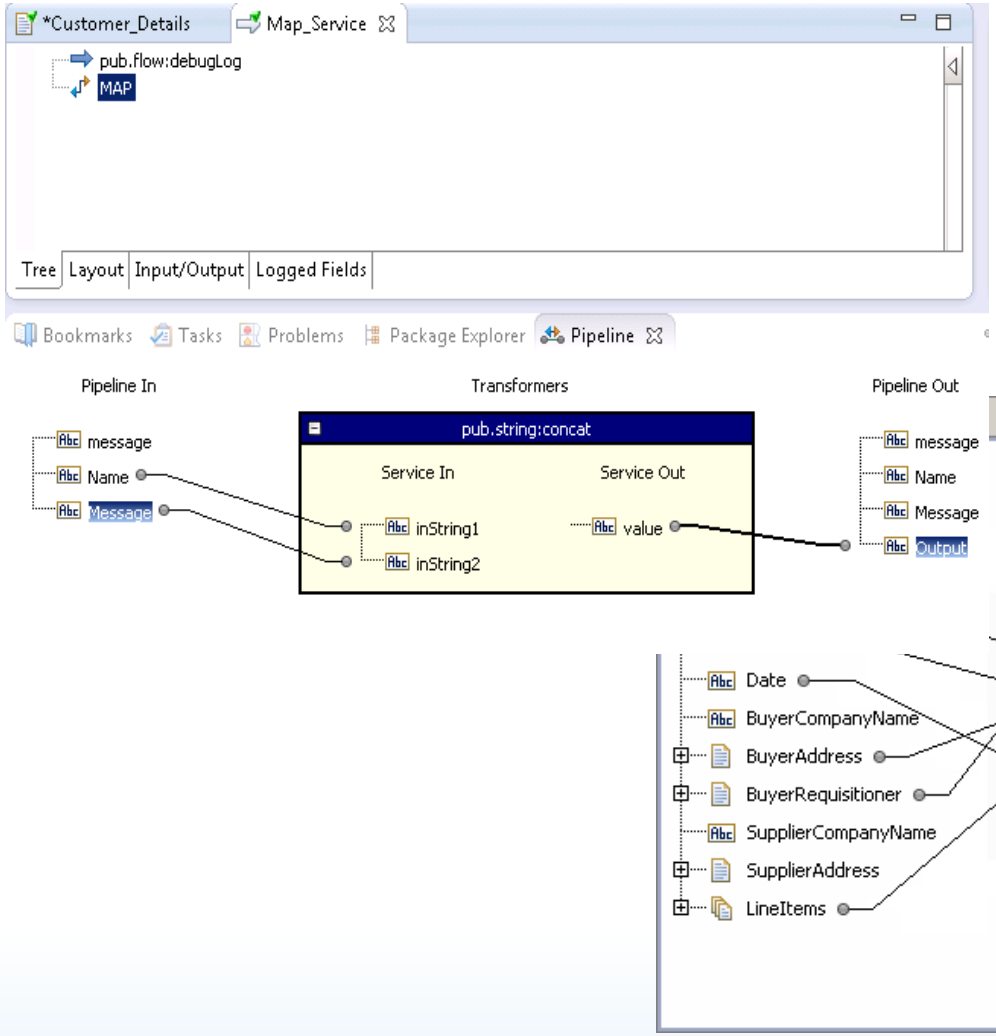
- By mapping you can accomplish the following transformations.

- Name transformations : This type of transformation resolves differences in the way data is named. For example, one service or document format might use telephone as the name of the variable for telephone number information and another might use phone number.

- Structural transformations : This type of transformation resolves differences in the data type or structure used to represent a data item. When you perform structural transformations, the value of the variable remains the same, but the data type or position of the variable in the Document structure changes.

- Value transformations. This type of transformation resolves differences in the way values are expressed. For example, you can change the format of a date, concatenate two Strings, or add the values of two variables together.

Pipeline view of a MAP Step



Flow Step - INVOKE

- Use the INVOKE step to request a service within a flow. You can use the INVOKE step to:

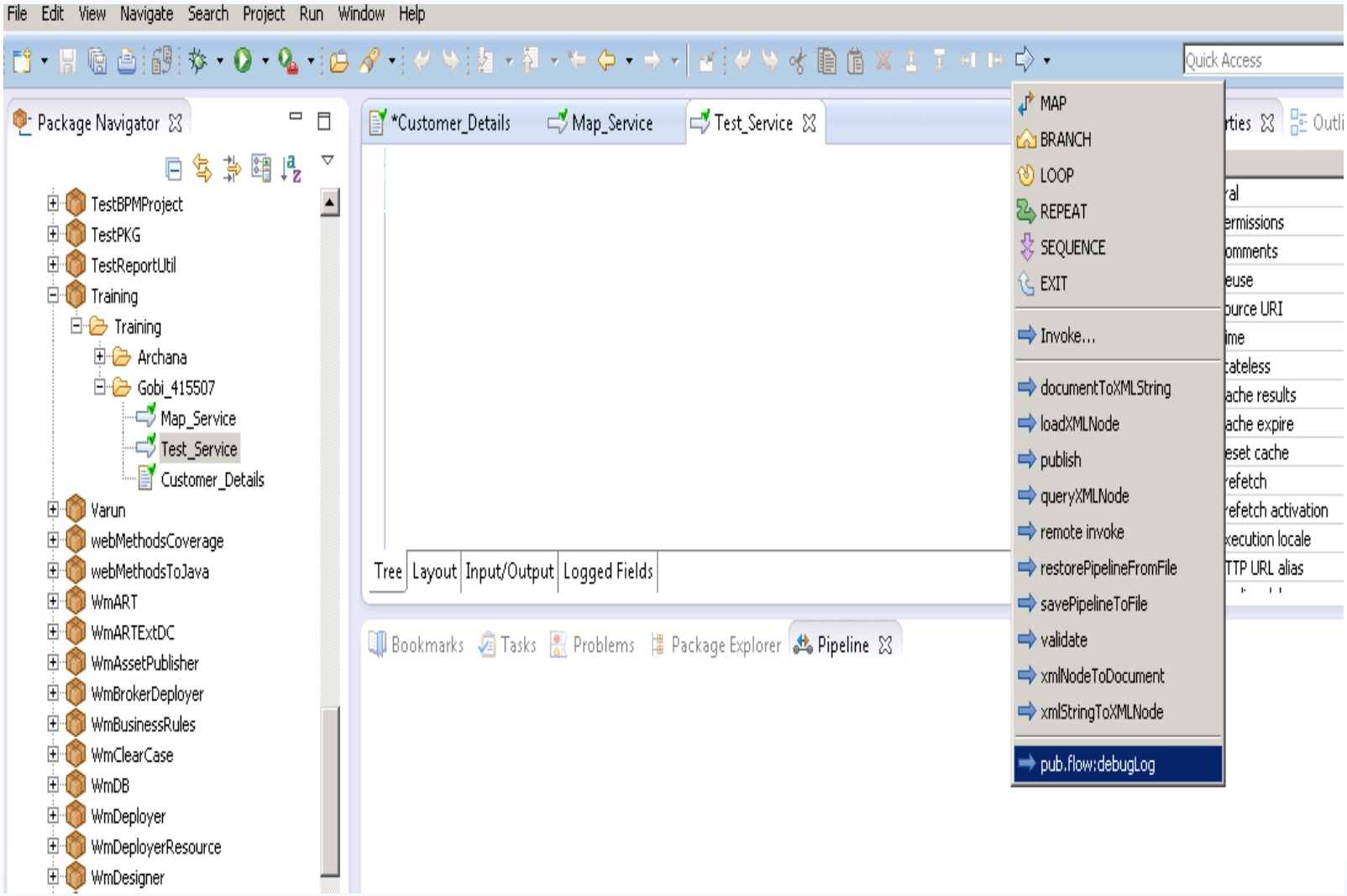
- Invoke any type of service, including other flow services and Web service connectors.

- Invoke any service for which the caller of the current flow has access rights on the local webMethods Integration Server.

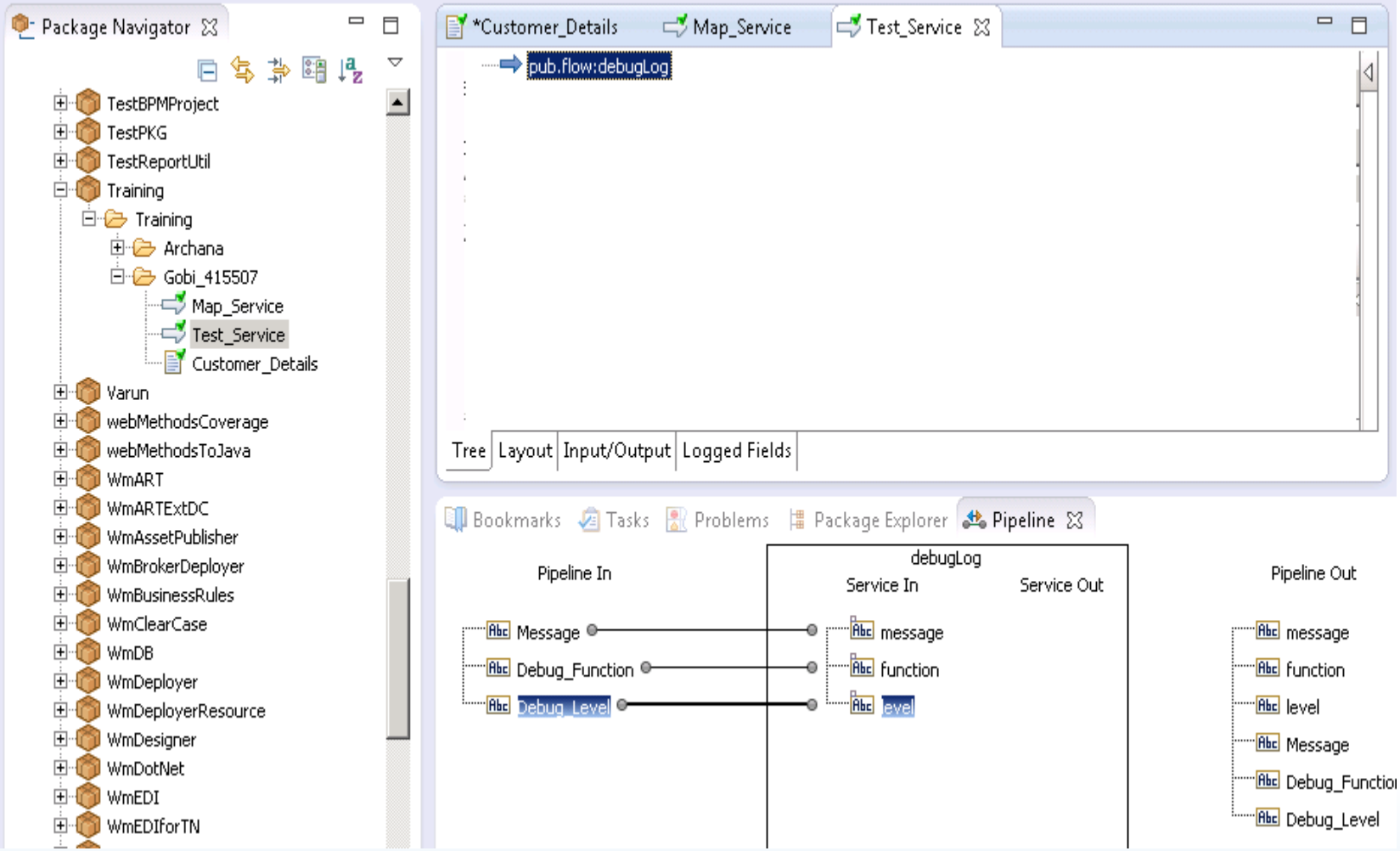
- Invoke built-in services and services on other webMethods Integration Servers.

- Invoke flow services recursively (that is, a flow service that calls itself). If you use a flow service recursively, bear in mind that you must provide a means to end the recursion.

- Invoke any service, validating its input and/or output.



Pipeline view of a Invoke Step



Flow Step - BRANCH

- The BRANCH step allows you to conditionally execute a step based on the value of a variable at run time. For example, you might use a BRANCH step to process a purchase order one way if the *PaymentType value is "CREDIT CARD" and another way if it is* "CORP ACCT".

When you build a BRANCH step, you can:

- Branch on a switch value. Use a variable to determine which child step executes.

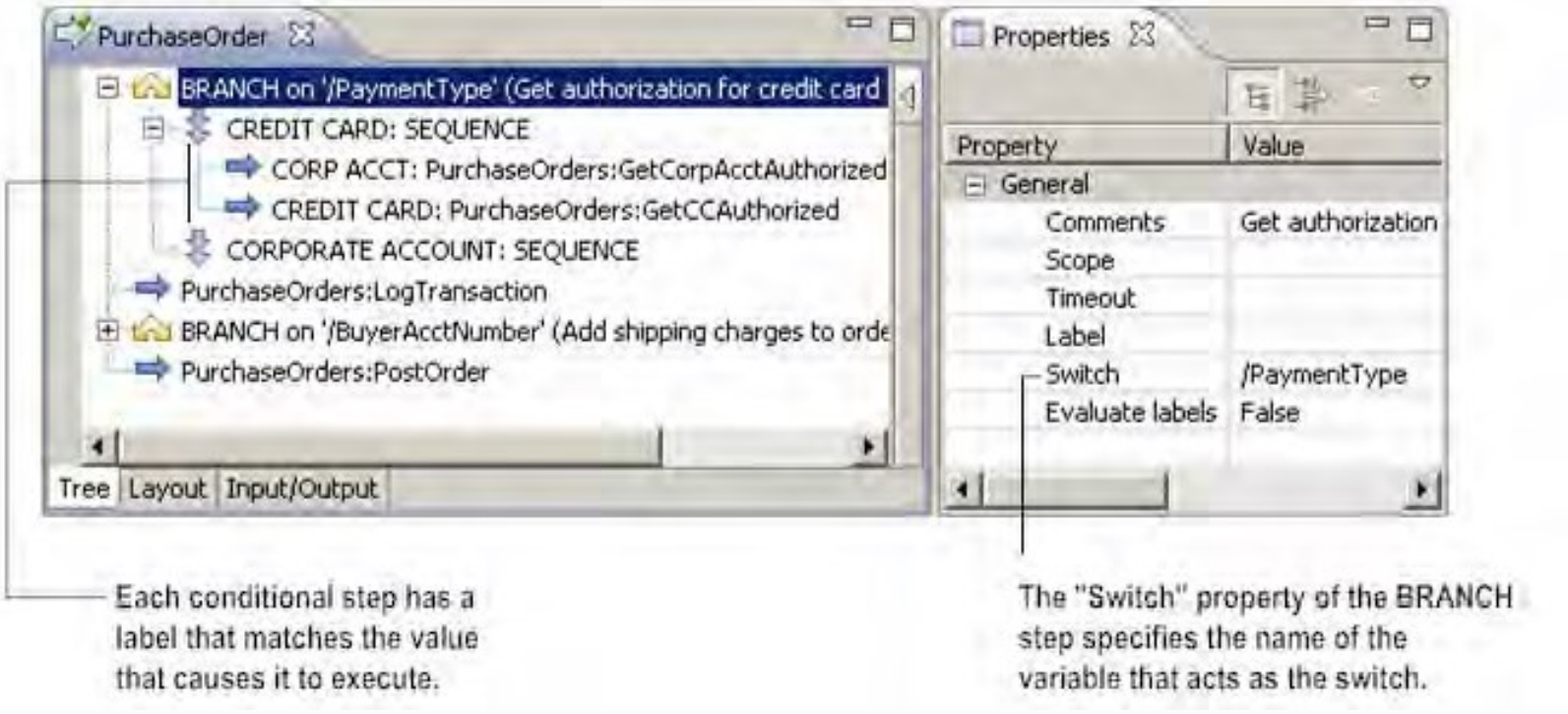
- At run time, the BRANCH step matches the value of the switch variable to the Label property of each of its targets. It executes the child step whose label matches the value of the switch.

- Branch on an expression. Use an expression to determine which child step executes.

- At run time, the BRANCH step evaluates the expression in the Label property of each child step. It executes the first child step whose expression evaluates to "true."

BRANCH on Switch value

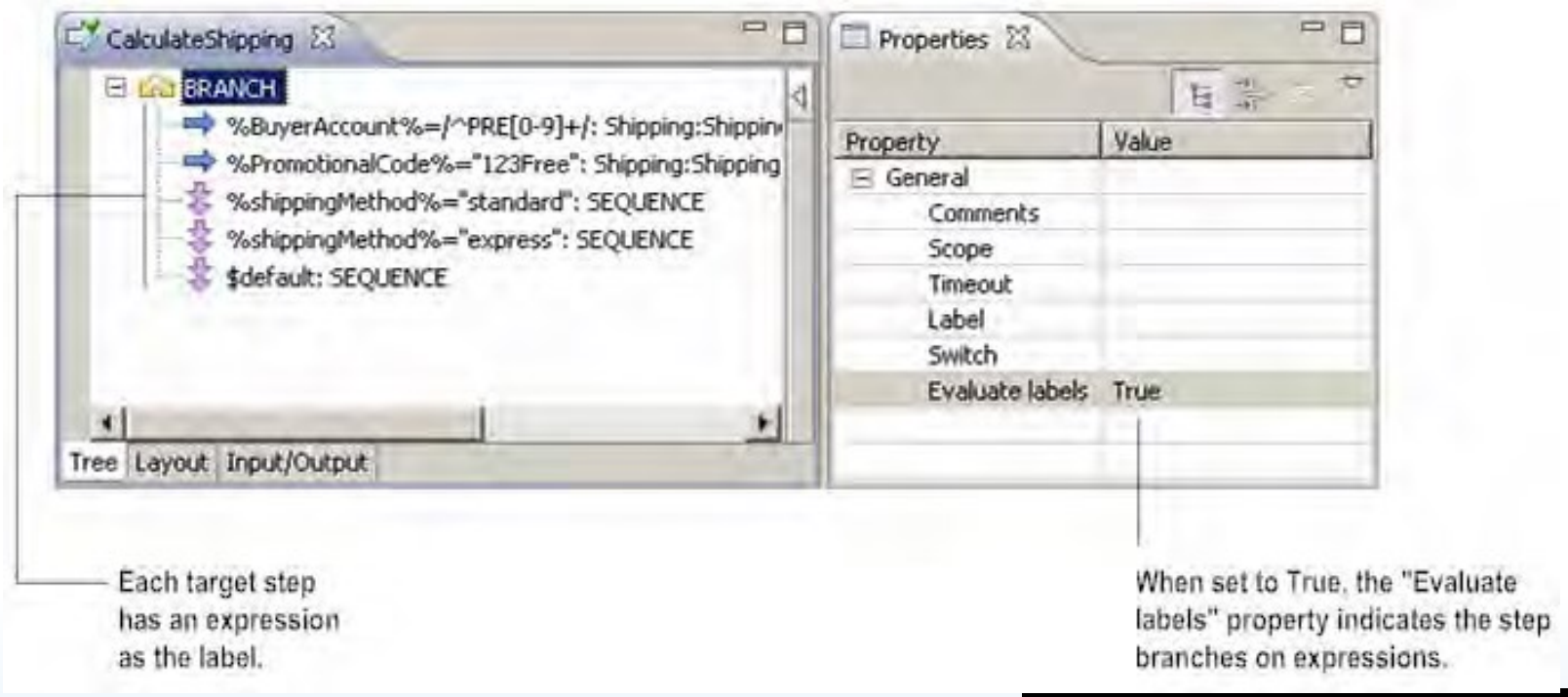
- When you branch on a switch value, you branch on the value of a single variable in the pipeline.



BRANCH on an expression

- When you branch on an expression, you assign an expression to each child of a branch step.

- At run time, the BRANCH step evaluates the expressions assigned to the child steps. It executes the first child step with an expression that evaluates to true.



Flow Step - SEQUENCE

- You use the SEQUENCE step to build a set of steps that you want to treat as a group.

- Steps in a group are executed in order, one after another. By default, all steps in a flow service, except for children of a BRANCH step, are executed as though they were members of an implicit SEQUENCE step (that is, they execute in order, one after another).

The most common reasons to do this are:

- To group a set of steps as a single alternative beneath a BRANCH step.

- To specify the conditions under which the server will exit a sequence of steps without executing the entire set.

- In an implicit sequence, when a step fails, the server automatically exits the sequence (that is, the **Exit on property is set to FAILURE).**

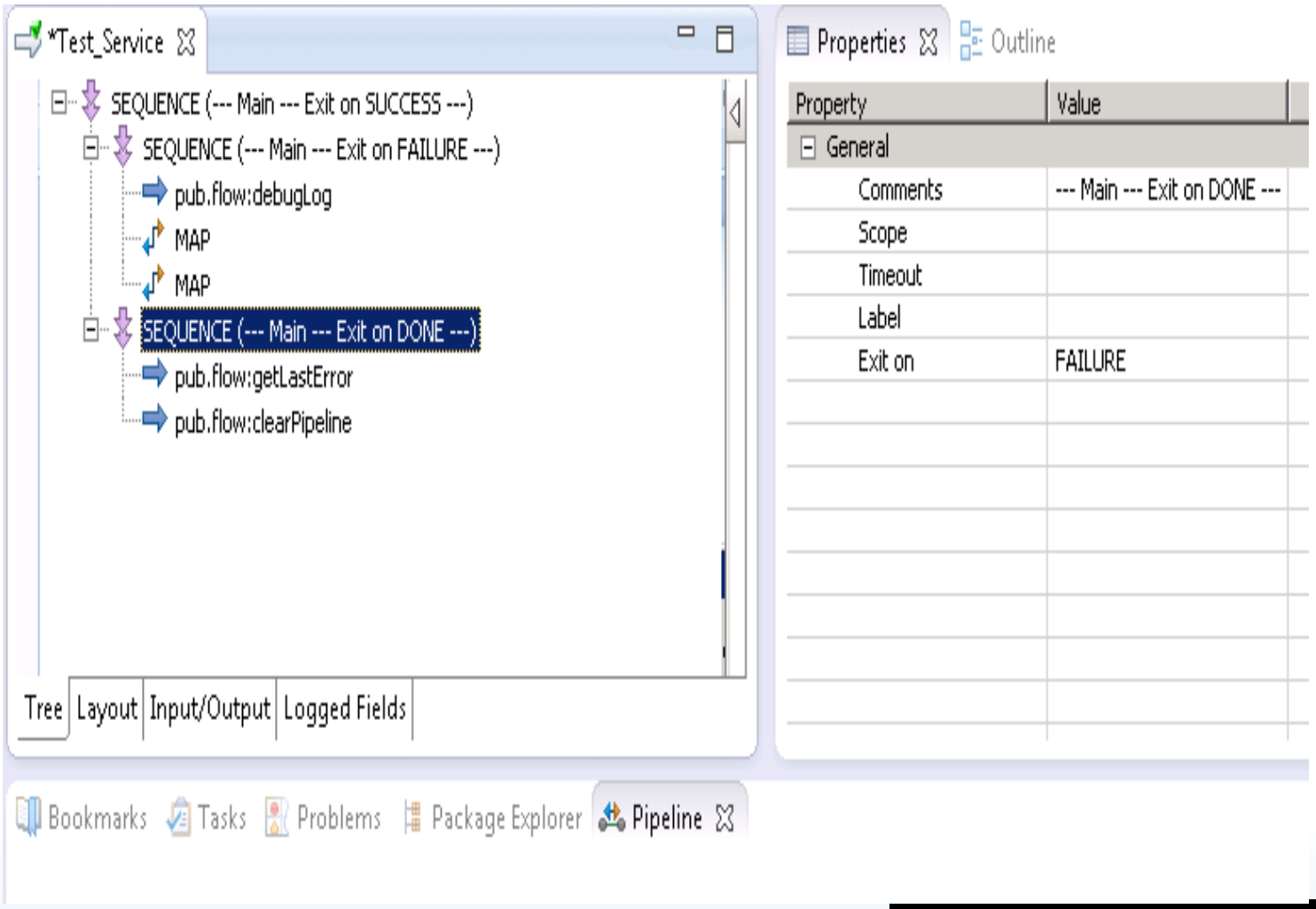
SEQUENCE – Exit on Condition

- FAILURE -- Exit the sequence when a step in the sequence fails. This is the default behavior of a sequence of steps. When a SEQUENCE exits under this condition, the SEQUENCE step fails.

- SUCCESS -- Exit the sequence when any step in the sequence succeeds. When a SEQUENCE exits under this condition, the server considers the SEQUENCE step successful, even if all its children fail.

- DONE -- Execute every step in the sequence even if one of the steps in the sequence fails. The server considers a SEQUENCE step successful as long as it executes all of its children within the specified time-out limit.

SEQUENCE – Main Try Catch Block

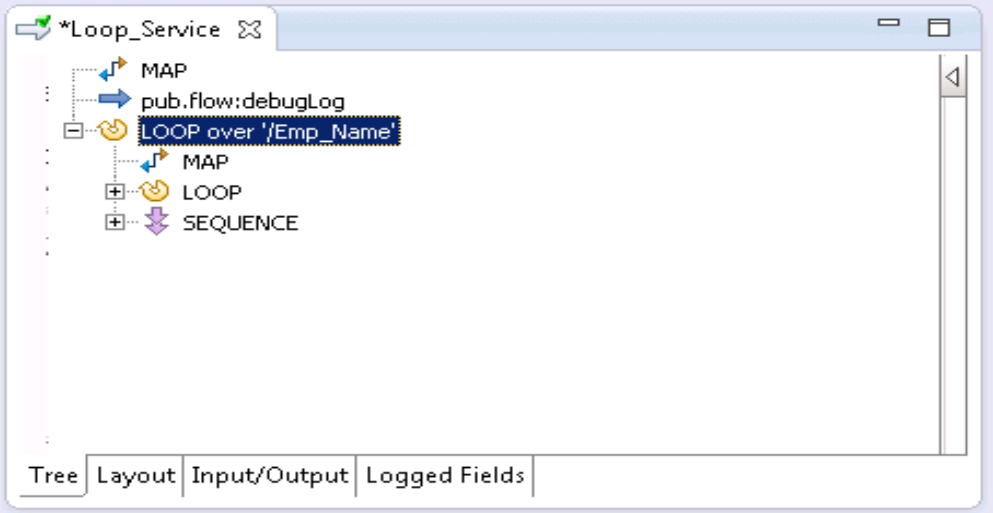


Flow Step - LOOP

- The LOOP step repeats a sequence of child steps once for each element in an array that you specify. For example, if your pipeline contains an array of purchase-order line items, you could use a LOOP step to process each line item in the array.

- To specify the sequence of steps that make up the body of the loop (that is, the set of steps you want the LOOP to repeat), you indent those steps beneath the LOOP as shown in the following example.

- Simple Loop step



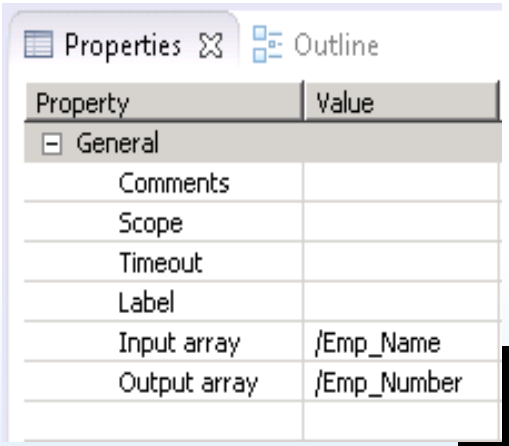
LOOP – Specifying Input Array

- Property – Input Array - The LOOP step requires you to specify an input array that contains the individual elements that will be used as input to one or more steps in the LOOP.

- At run time, the LOOP step executes one pass of the loop for each member in the specified array.

- For example, if you want to execute a LOOP for each line item stored in a purchase order, you would use the document list in which the order's line items are stored as the LOOP's input array.

- The array you specify can be any of the following data types:
  - String list
  - String table
  - Document list
  - Object list



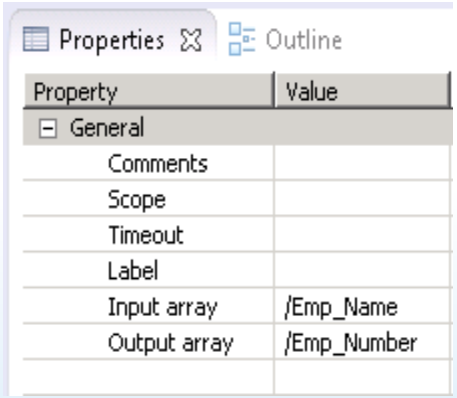
LOOP – Collecting output - OutputArray

- Property – Output Array - If your LOOP step produces an output variable, the server can collect that output into an array in the pipeline.

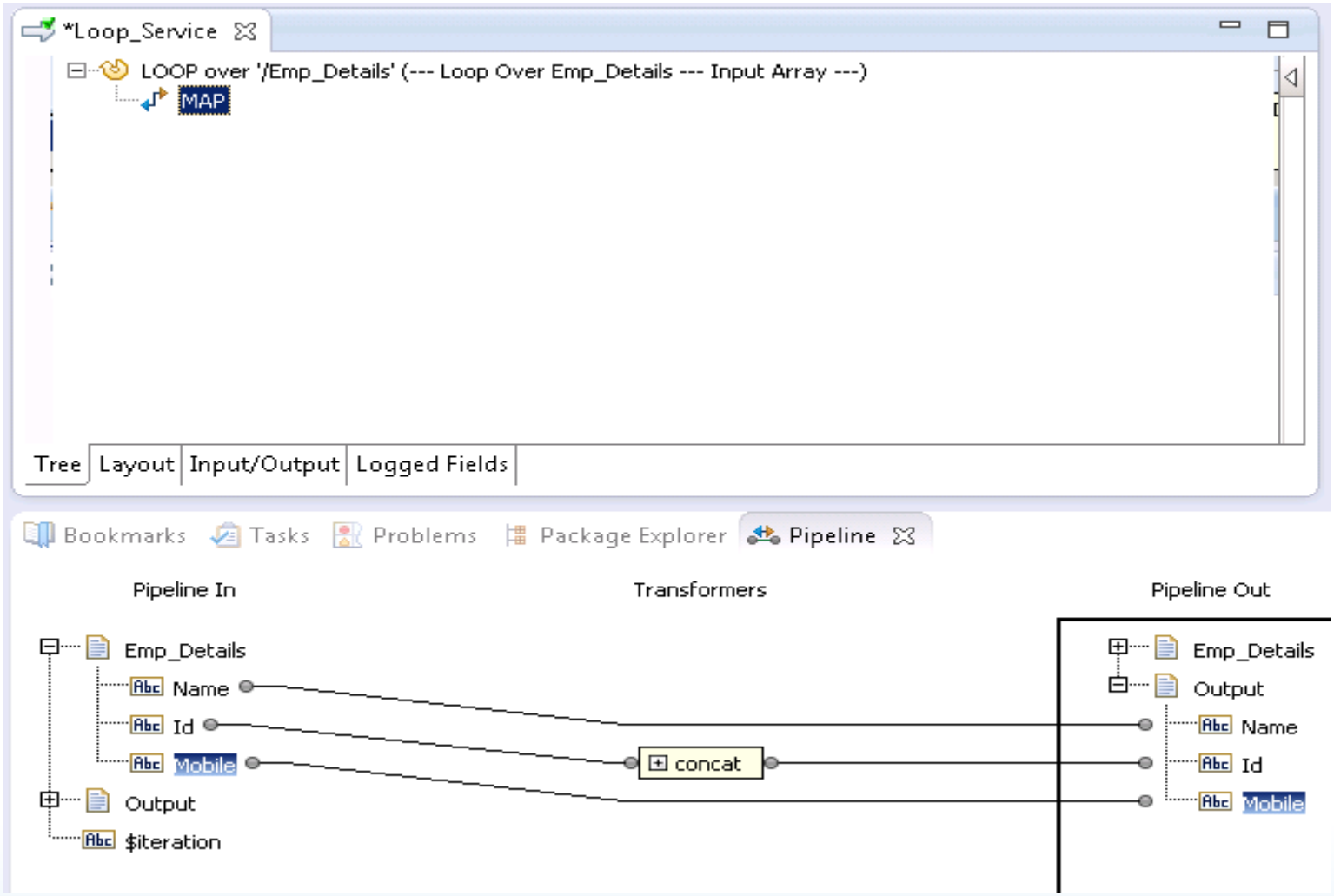
- To do this, you use the Output array parameter to specify the name of the array variable into which you want the server to collect output for each iteration of the loop.

- For example, if your loop checks inventory status of each line item in a purchase order and produces a String called InventoryStatus each time it executes, you would specify InventoryStatus as the value of Output array.

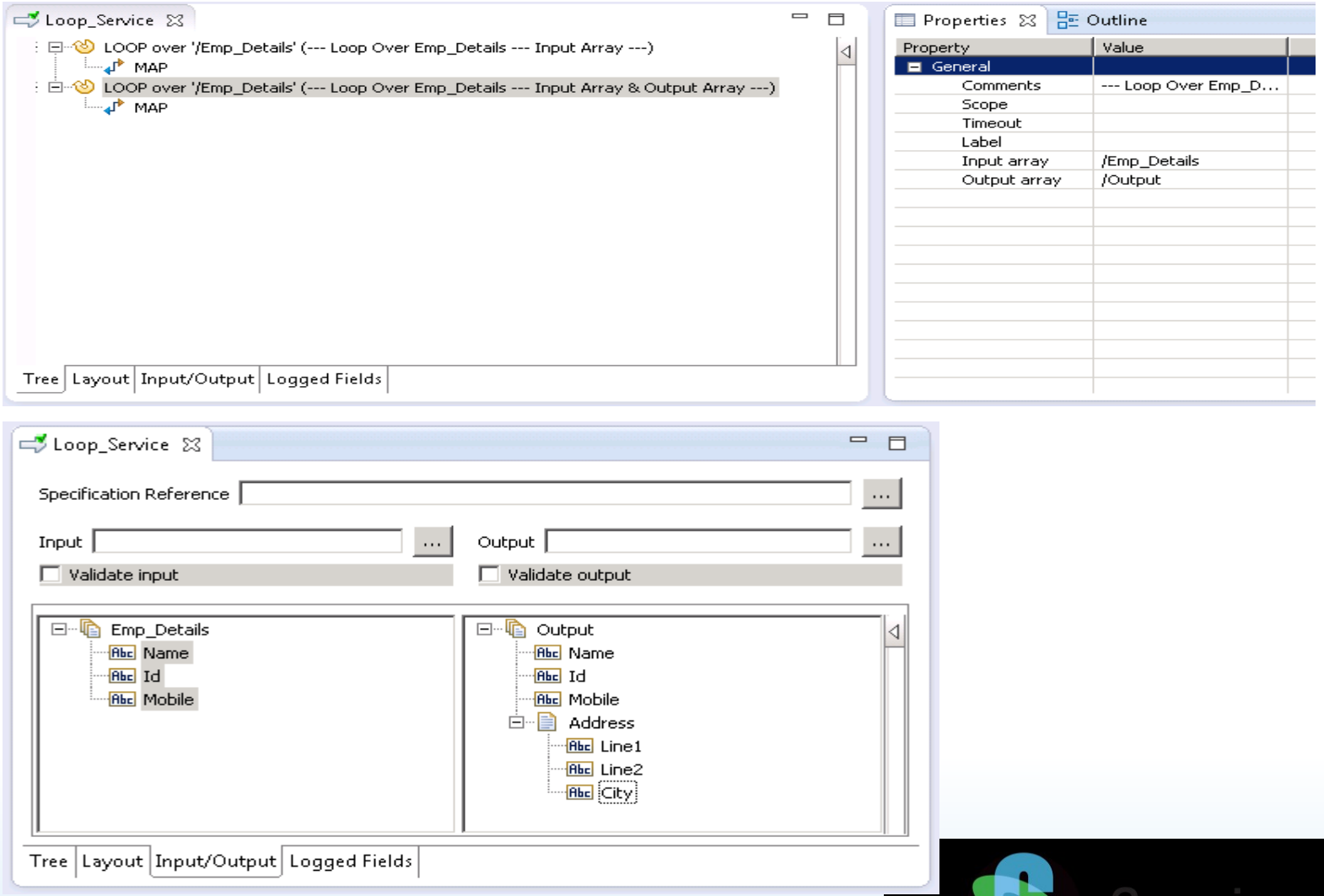
- To collect output from each pass of the loop, specify the name of the output variable that you want the server to collect for each iteration.



LOOP – Pipeline view (Input Array)



LOOP – Pipeline view – Output Array



Flow Step - REPEAT

- The REPEAT step allows you to conditionally repeat a sequence of child steps based on the success or failure of those steps.

- *Re-execute (retry) a set of steps if any step within the set fails*. This option is useful to accommodate transient failures that might occur when accessing an external system.

- *Re-execute a set of steps until one of the steps within the set fails*. This option is useful for repeating a process as long as a particular set of circumstances exists.

REPEAT – Properties

- "Repeat On" When you build a REPEAT step, you set the Repeat on property to specify the condition (success or failure) that will cause its children to re-execute at run time.

- *FAILURE* -- Re-executes the set of child steps if any step in the set fails.

- *SUCCESS* -- Re-executes the set of child steps if all steps in the set complete successfully.

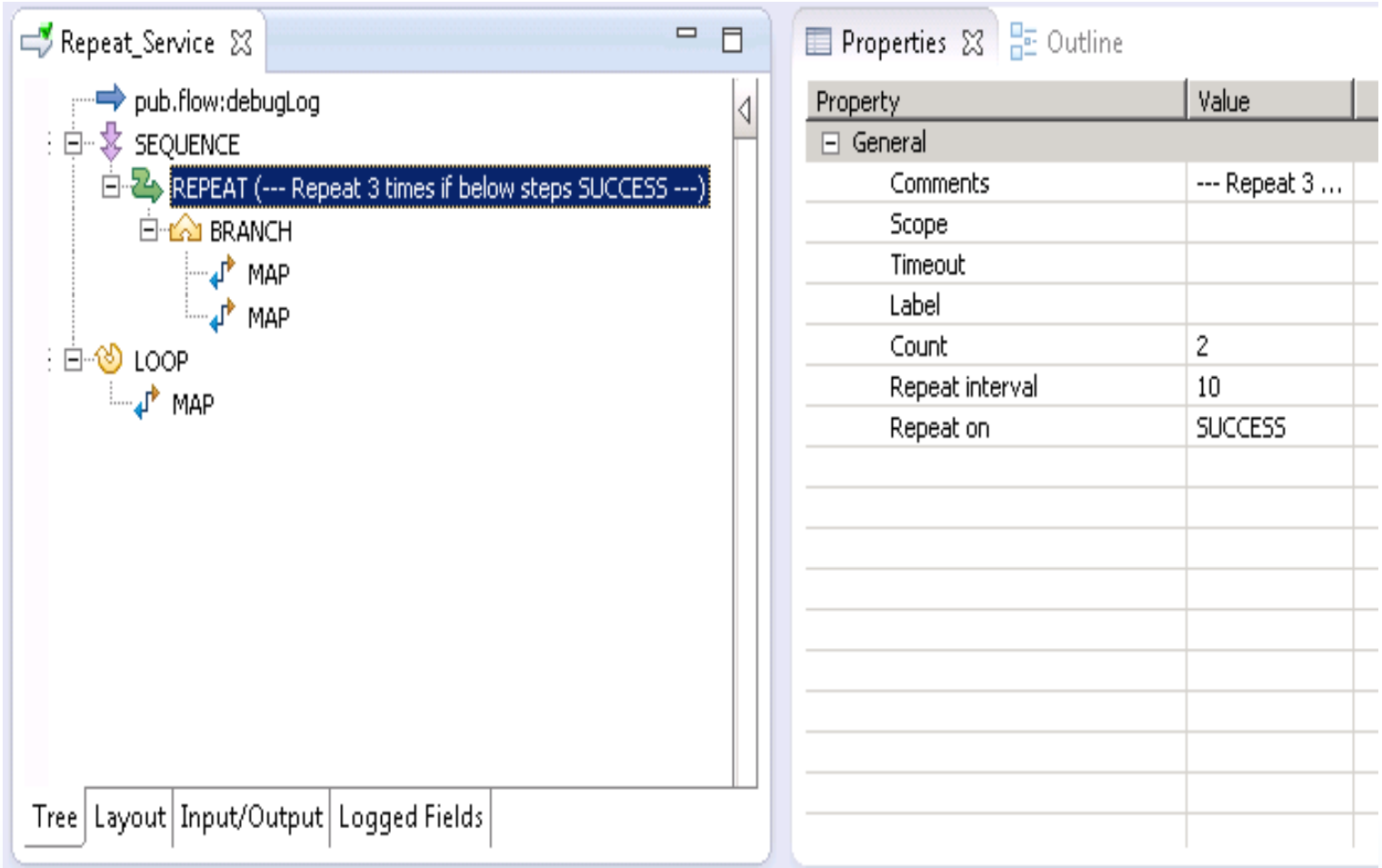
- "Repeat Count" The REPEAT step's Count property specifies the maximum number of times the server re-executes the child steps in the REPEAT step..

- *0* -- Does not re-execute children.

- *Any value > 0* – Re-executes children up to this number of times.

- *-1 or blank* - Re-executes children as long as the specified Repeat on condition is true.

REPEAT– Pipeline view



Flow Step - EXIT

- The EXIT flow step allows you to exit the entire flow service or a single flow step. You specify whether you want to exit from.

- When you use the EXIT step, you indicate whether exiting should return a successful condition or a failure condition.

Examples of when to use the EXIT step include to:

- Exit an entire flow service from within a series of deeply nested steps.

- Throw an exception when you exit a flow or a flow step without having to write a Java service to call Service.throwError( ).

- Exit a LOOP or REPEAT flow step without throwing an exception.

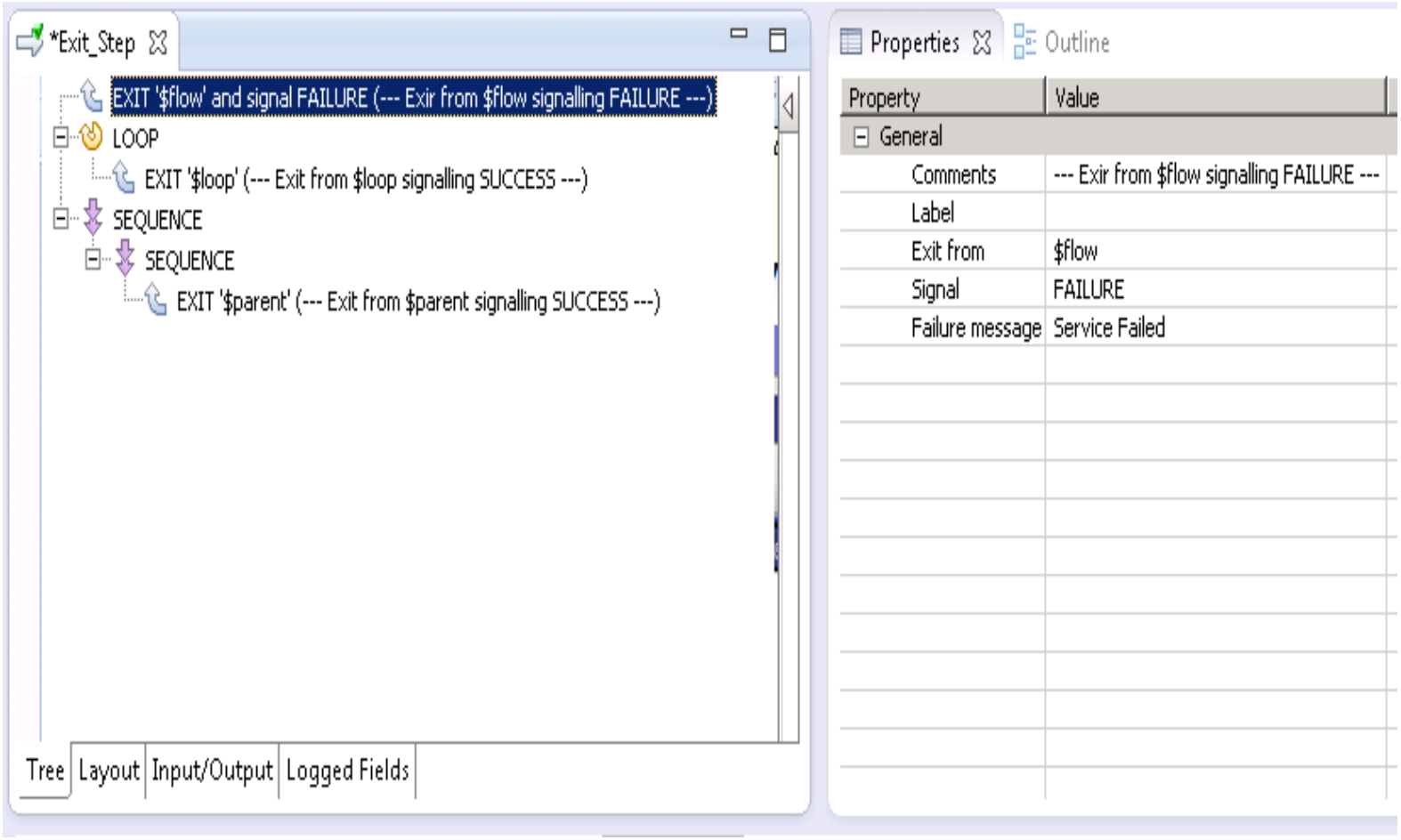
EXIT – Properties

- "Exit from " -- The flow step from which you want to exit. Specify one of the following:
  - $loop -- Nearest ancestor LOOP or REPEAT flow step
  - $parent -- Parent flow step, regardless of the type of step
  - $flow -- Entire flow service.

- "Label" -- Nearest ancestor flow step that has a label that matches this value.

- "Signal" --Whether the exit is to be considered a success or a failure. Specify one of the following:
  - SUCCESS -- Exit the flow service or flow step with a success condition.
  - FAILURE -- Exit the flow service or flow step with a failure condition. An exception is thrown after the exit. You specify the error message with the Failure message property.

EXIT– Pipeline view



Summary

What have we learnt today ?

- webMethods Datatypes
- Introduction to Services
- Documents
- Flow Services
- Flow Steps - MAP
- Flow Steps – INVOKE
- Flow Steps - BRANCH

Q & A

- List out the different DataTypes used in webMethods
- What is a service ?
- How flow services are getting stored in Integration Server?
- What is a pipeline?
- What are different CONTROL steps available in Flow language?
- What is a Canonical Document ?
- List out the different transformations available in webMethods?
- What are the two ways you can build a BRANCH?

Thank you