Cognizant Technology Solution

Exercises – Flat File And FTP

WebMethods Integration Workshop

WebMethods CoE

Table of Contents

Introduction 3

Flat File 3

Flat File Schema 3

FTP 4

Theory 4

Practical 5

Flat File Dictionary Creation 5

Steps to Create Flat File Dictionary: 5

Flat File Schema Creation 7

Steps to create Flat File Schema: 7

Document Creation 11

Steps to create document type: 11

Flow Service Creation 12

Steps to create flow service: 12

FTP Session Creation 16

Flat File And FTP

Introduction

Flat File

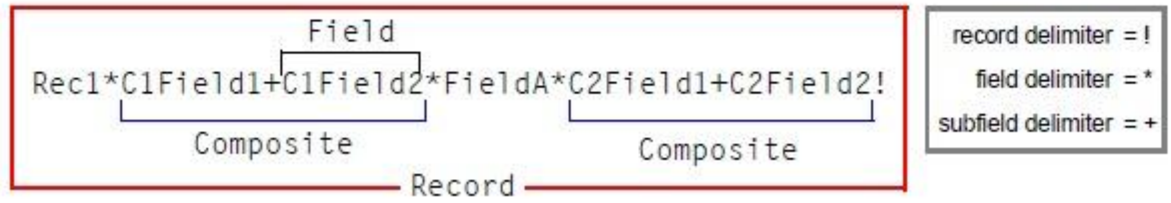
When different applications attempt to communicate with one another, they may not speak the same "language." **Flat files** enable you to send data to any application in a mutually agreed upon format so that the data in the files can be read and processed.

Flat files present complex hierarchical data in a record–based storage format. In other words, the metadata of a flat file is separated from the data and contained in a flat file schema. A single logical record of application data is externalized as a set of records without any structural information. Therefore, the application receiving the flat file must have knowledge of the structure of the flat file, through the flat file schema, to read the flat file.

All flat files consist of a list of records containing fields and composites:

- **Fields:** are atomic pieces of data (for example, postal code).
- **Composites:** contain multiple fields (for example, ID and ID qualifier, Date and time).
  The fields within a composite are referred to as subfields. A composite definition contains the same information as a record definition, except that composites are not allowed inside composites.
- **Records:** (also known as segments) are sequences of fields and/or composites.

For example, the following flat file data and its list of delimiters enables you to see how *elements* (records, composites, and fields) within a flat file can be identified.



Flat File Schema

A **flat file schema** is the blueprint that contains the instructions for parsing or creating a flat file and is created as a namespace element in the webMethods Integration Server. This blueprint details the structure of the document, including delimiters, records, and repeated record structures. A flat file schema also acts as the model against which you can validate an inbound flat file.

A flat file schema consists of hierarchical *elements* that represent each record, field, and subfield in a flat file. Each element is a *record*, *composite*, or *field*, and either a *definition* or *reference*. You then configure each element with the necessary constraints.

FTP

FTP or **file transfer protocol** is a commonly used protocol for exchanging files over any network that supports the TCP/IP protocol (such as the Internet or an intranet). There are two computers involved in an FTP transfer: a server and a client. The **FTP server**, running FTP server software, listens on the network for connection requests from other computers. The client computer, running **FTP client** software, initiates a connection to the server. Once connected, the client can do a number of file manipulation operations such as uploading files to the server, download files from the server, rename or delete files on the server and so on.

Theory

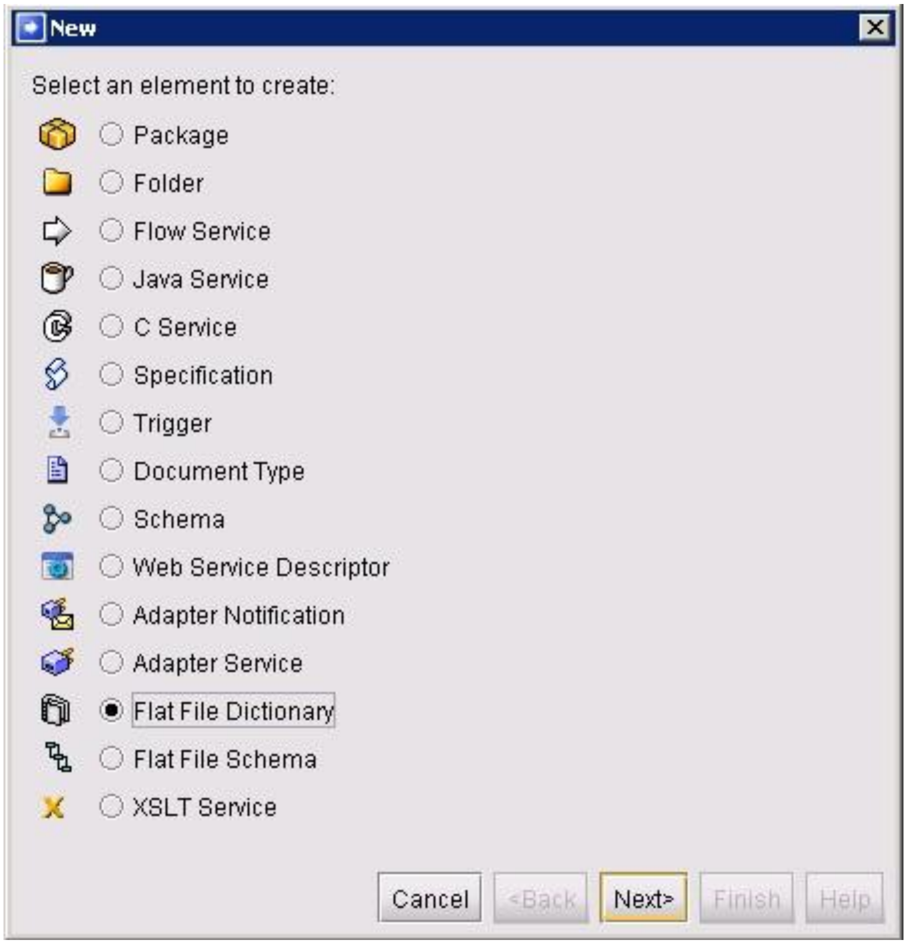
- 8-2-SP1_Flat_File_Schema_Developers_Guide - Chapters 1,2 and 3

Practical

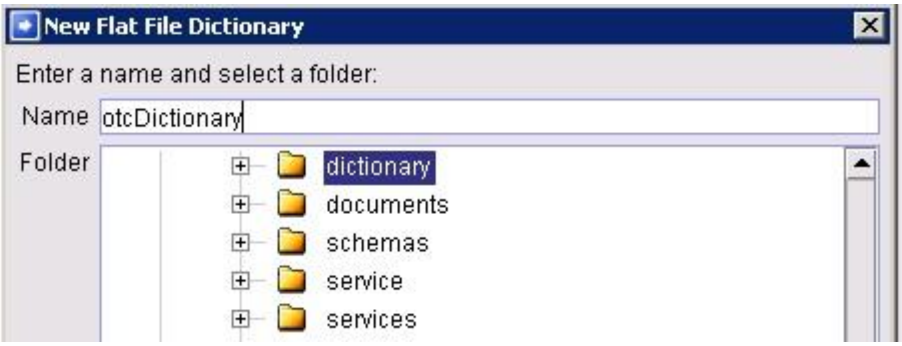
Flat File Dictionary Creation

Steps to Create Flat File Dictionary:

1) Right click on dictionary folder select **New** -> **All Choices**. Developer opens the **New** wizard.
2) Select **Flat File Dictionary** from the list of elements and click **Next**.

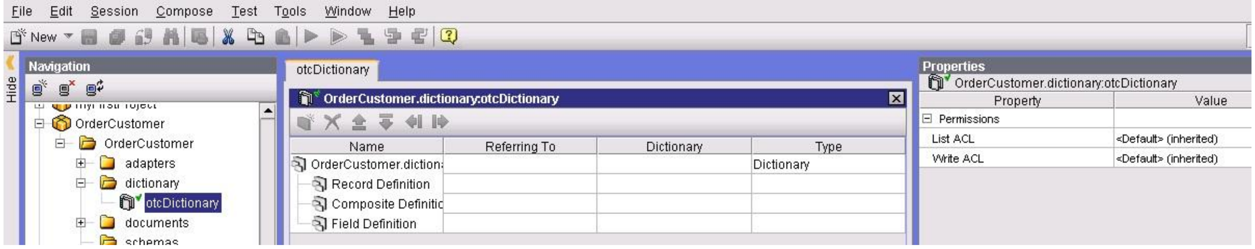


3) Type a unique name for the dictionary and select the appropriate folder. Click **Finish**.



Name otcDictionary

4) A flat file dictionary is created with specified name.

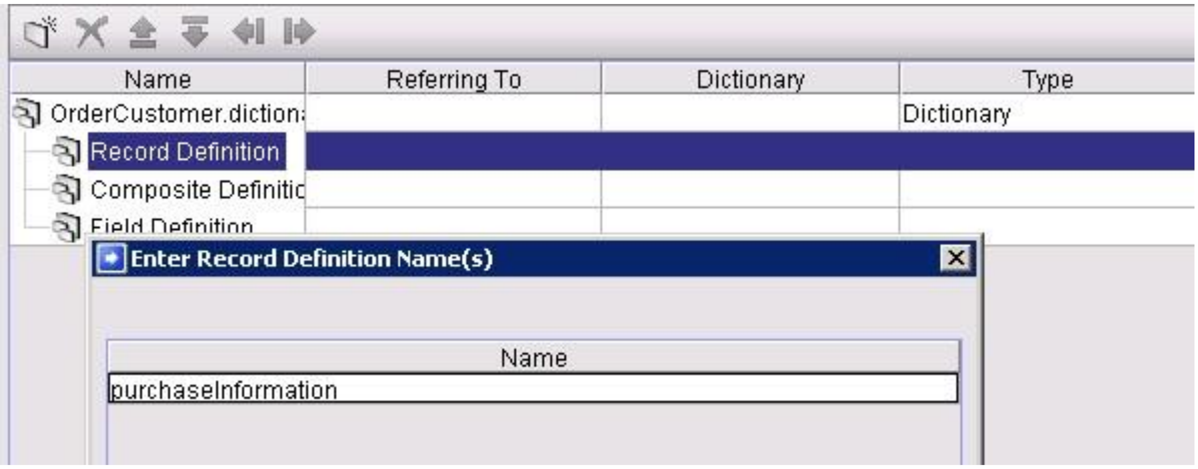


Name	Referring To	Dictionary	Type
OrderCustomer.diction:			Dictionary
Record Definition			
Composite Definitic			
Field Definition			

OrderCustomer.dictionary:otcDictionary

Property	Value
Permissions	
List ACL	<Default> (inherited)
Write ACL	<Default> (inherited)

5) To add record definition in flat file dictionary, right click on **Record Definition** and select **New**. Type the appropriate name for record and click Finish.



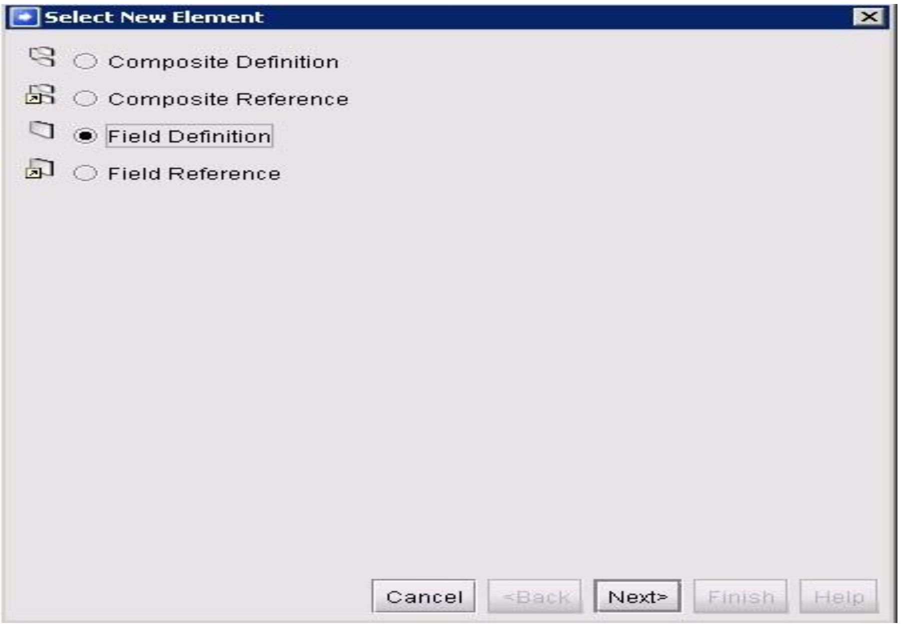
Name	Referring To	Dictionary	Type
OrderCustomer.diction:			Dictionary
Record Definition			
Composite Definitic			
Field Definition			

Enter Record Definition Name(s)

Name
purchaseInformation

6) To add field for record, do the following steps:

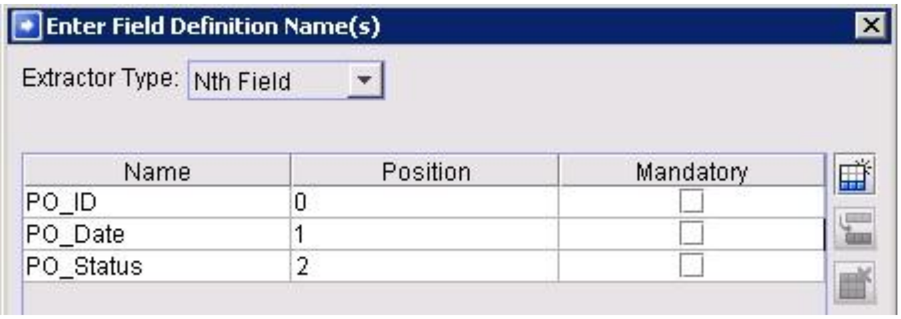
- Right click on the record name and select **New**, developer opens the **Select New Element** wizard.
- Select **Field Definition** and click **Next**. Developer opens **Enter Field Definition Names(s)** wizard.



Select New Element

○ Composite Definition
○ Composite Reference
● Field Definition
○ Field Reference

- Choose Extractor type as **Nth Field**. And enter the field names, position and mandatory in **Name**, **Position** and **Mandatory** columns respectively. Click **Finish**.

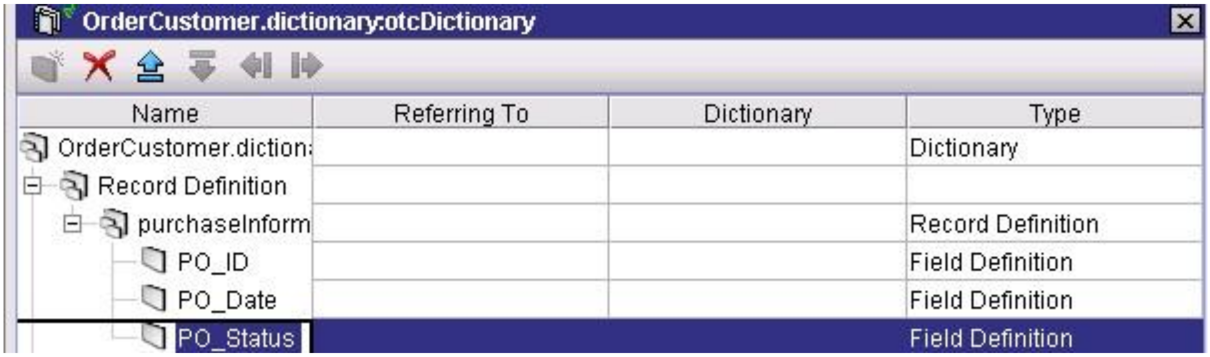


Enter Field Definition Name(s)

Extractor Type: Nth Field

Name	Position	Mandatory
PO_ID	0	□
PO_Date	1	□
PO_Status	2	□

7) The flat file dictionary should look like this.



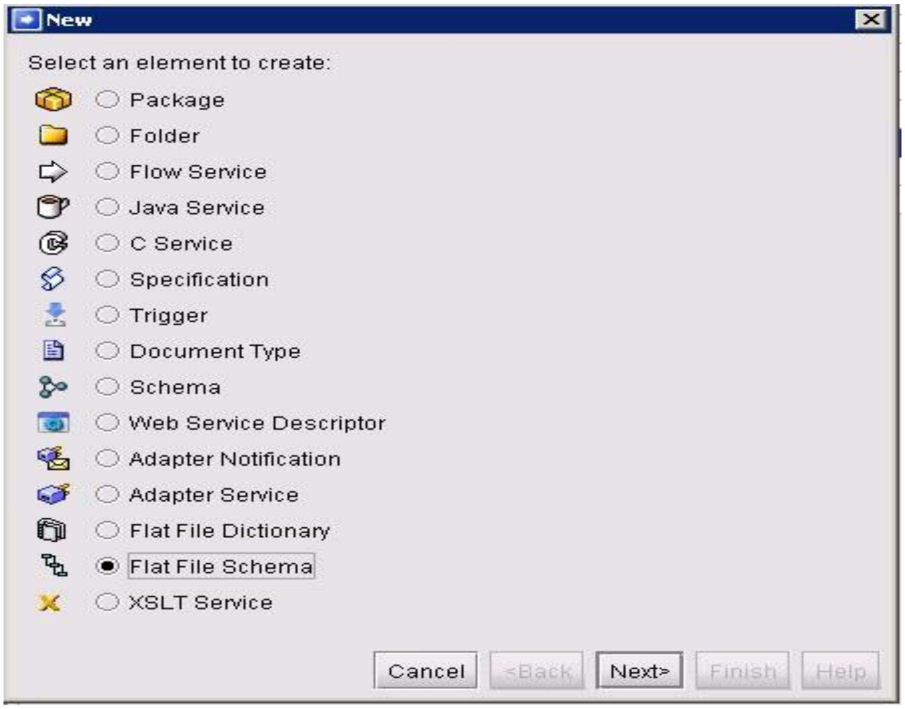
OrderCustomer.dictionary:otcDictionary

Name	Referring To	Dictionary	Type
OrderCustomer.diction:			Dictionary
Record Definition			
purchaseInform			Record Definition
PO_ID			Field Definition
PO_Date			Field Definition
PO_Status			Field Definition

Flat File Schema Creation

Steps to create Flat File Schema:

1) Right click on schema folder select **New** -> **All Choices**. Developer opens the **New** wizard.
2) Select **Flat File Schema** from the list of elements and click **Next**.

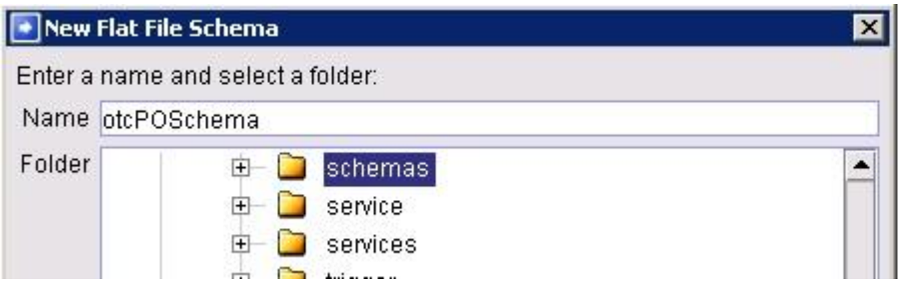


New

Select an element to create:

○ Package
○ Folder
○ Flow Service
○ Java Service
○ C Service
○ Specification
○ Trigger
○ Document Type
○ Schema
○ Web Service Descriptor
○ Adapter Notification
○ Adapter Service
○ Flat File Dictionary
● Flat File Schema
○ XSLT Service

3) Type a unique name for the schema and select the appropriate folder. Click **Finish**.



New Flat File Schema

Enter a name and select a folder:

Name otcPOSchema

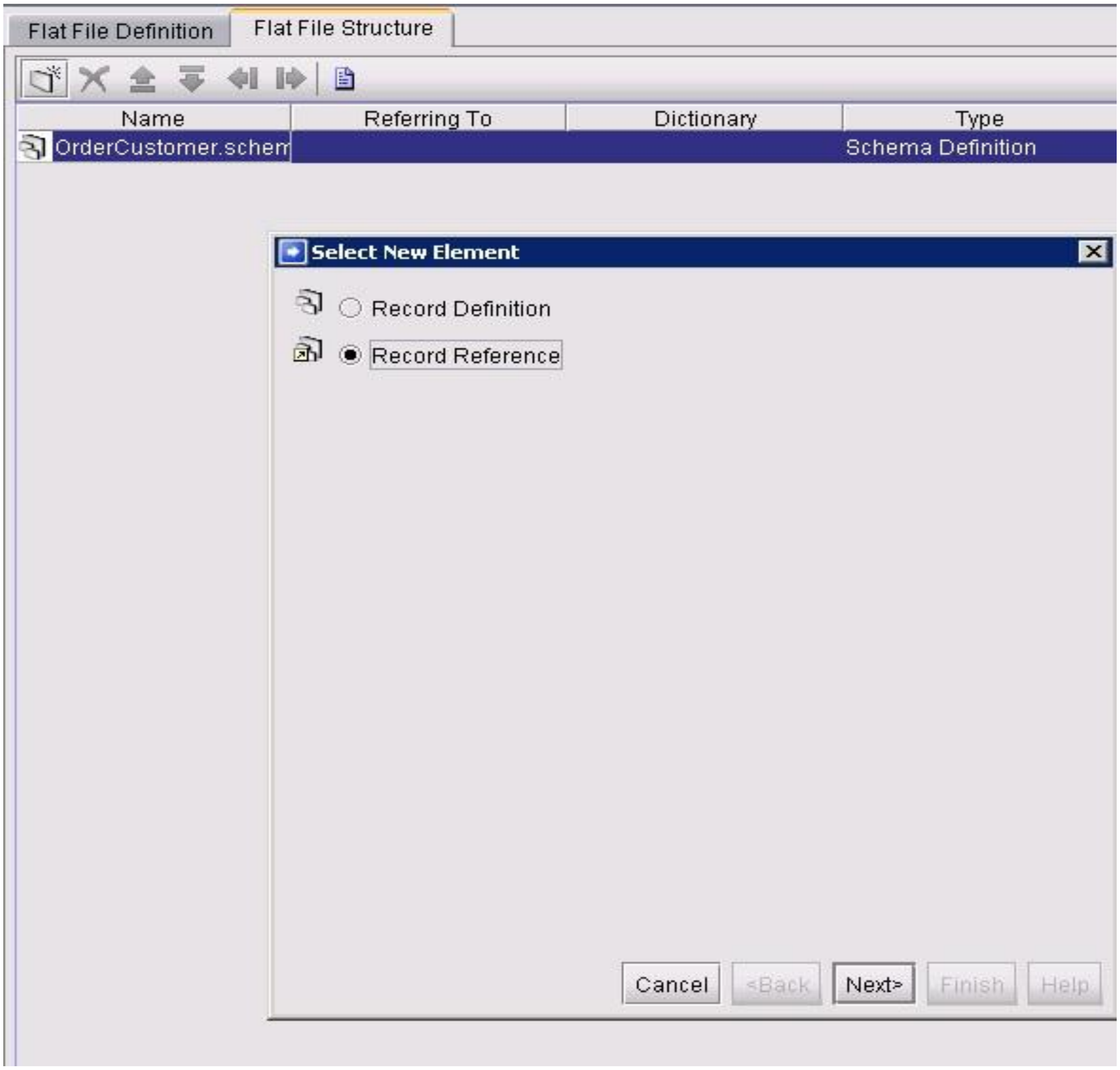
Folder

- schemas
- service
- services

4) A flat file schema is created with specified name.

5) To add records and delimiter types, do the following steps:

- Select the **Flat File Structure** tab, click on new( ), developer opens the **Select New Element** wizard.
- Select **Record Reference** and click **Next**.



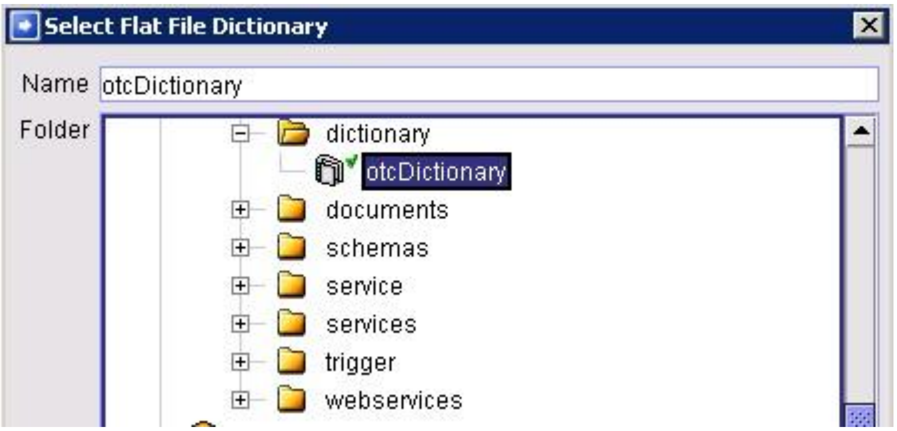
Name	Referring To	Dictionary	Type
OrderCustomer.schem			Schema Definition

Select New Element

○ Record Definition

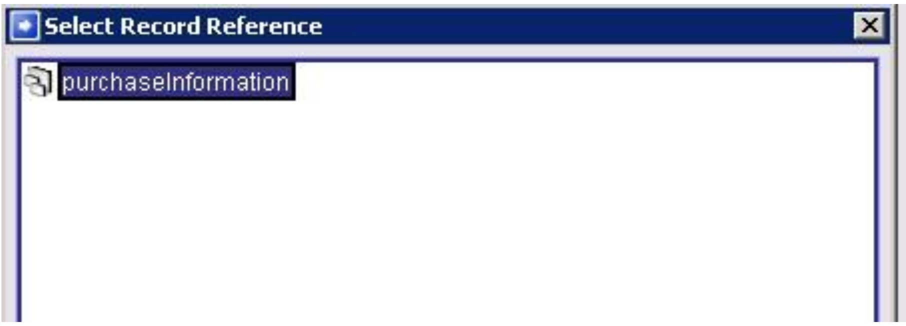
● Record Reference

- Browse and select the dictionary that has record definition which is previously created. Click **Next**.

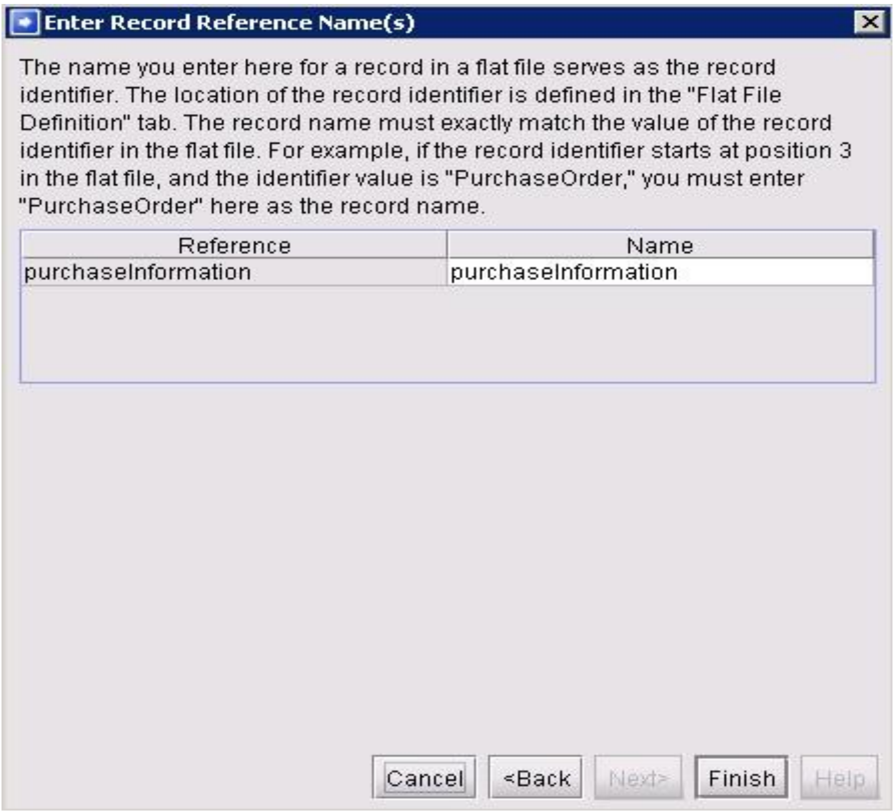


Name otcDictionary

- Select the **Record Reference** and click **Next**.



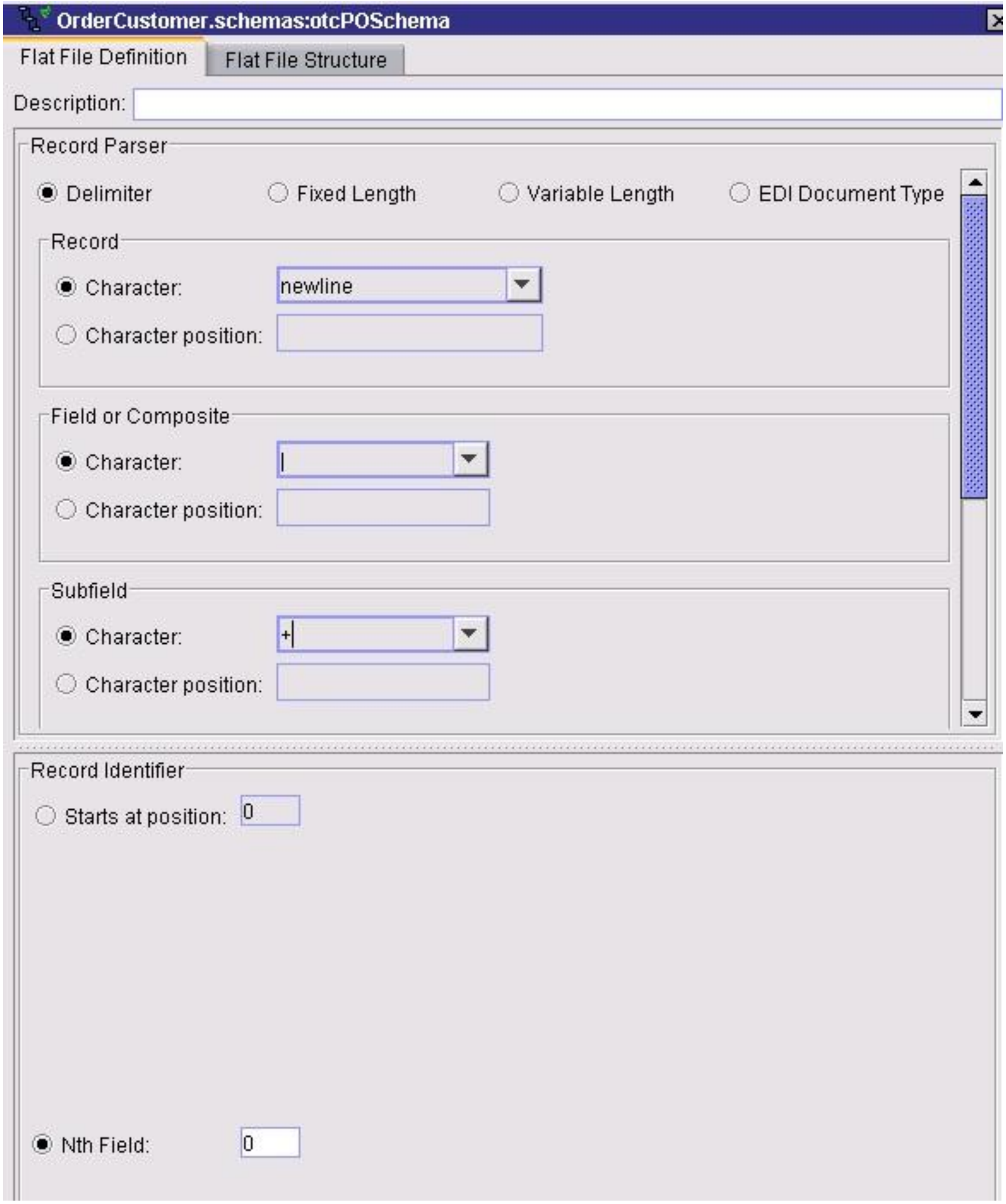
- Retain the same name or give name to record and click **Finish**.



The name you enter here for a record in a flat file serves as the record identifier. The location of the record identifier is defined in the "Flat File Definition" tab. The record name must exactly match the value of the record identifier in the flat file. For example, if the record identifier starts at position 3 in the flat file, and the identifier value is "PurchaseOrder," you must enter "PurchaseOrder" here as the record name.

Reference	Name
purchaseInformation	purchaseInformation

- Select **Flat File Definition** tab, select **Delimiter** in the **Record Parser** and set the character field for **Record**, **Field or Composite** and **Subfield**.



OrderCustomer.schemas:otcPOSchema

Flat File Definition | Flat File Structure

Description:

Record Parser

● Delimiter ○ Fixed Length ○ Variable Length ○ EDI Document Type

Record

● Character: newline

○ Character position:

Field or Composite

● Character: |

○ Character position:

Subfield

● Character: +

○ Character position:

Record Identifier

○ Starts at position: 0

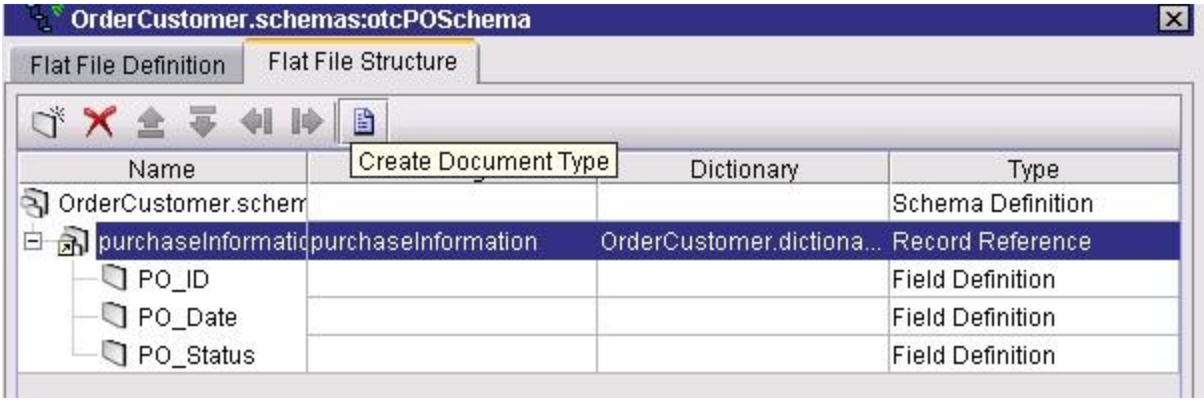
● Nth Field: 0

6) **Save** the flat file schema.

Document Creation

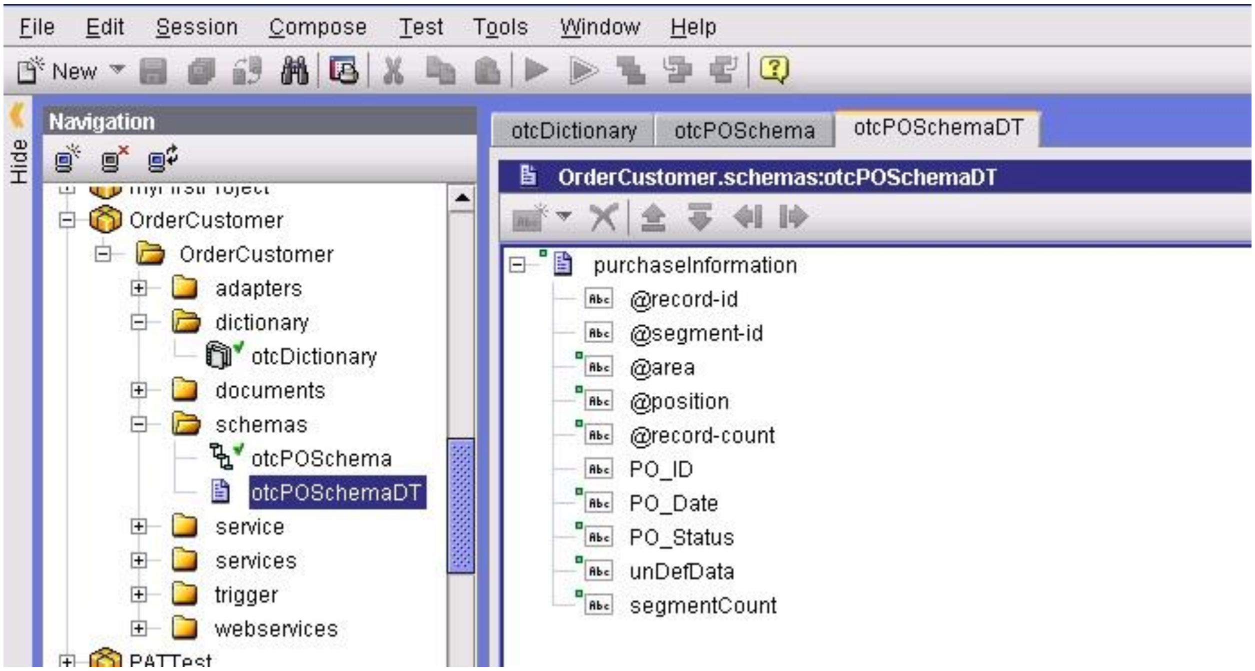
Steps to create document type:

1) Select the Flat File Schema and select **Flat File Structure** tab, click on(🗎) to create the document.



Name		Dictionary	Type
OrderCustomer.schem			Schema Definition
purchaseInformatic	purchaseInformation	OrderCustomer.dictiona...	Record Reference
PO_ID			Field Definition
PO_Date			Field Definition
PO_Status			Field Definition

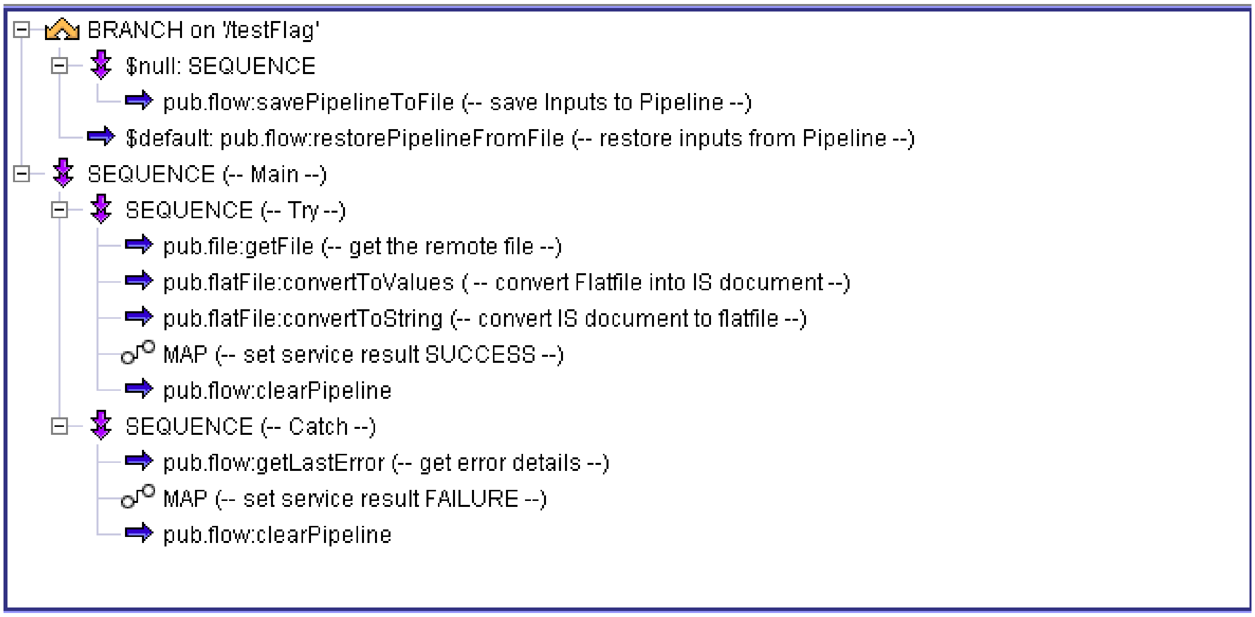
2) The document will be generated automatically and has record and fields.



Flow Service Creation

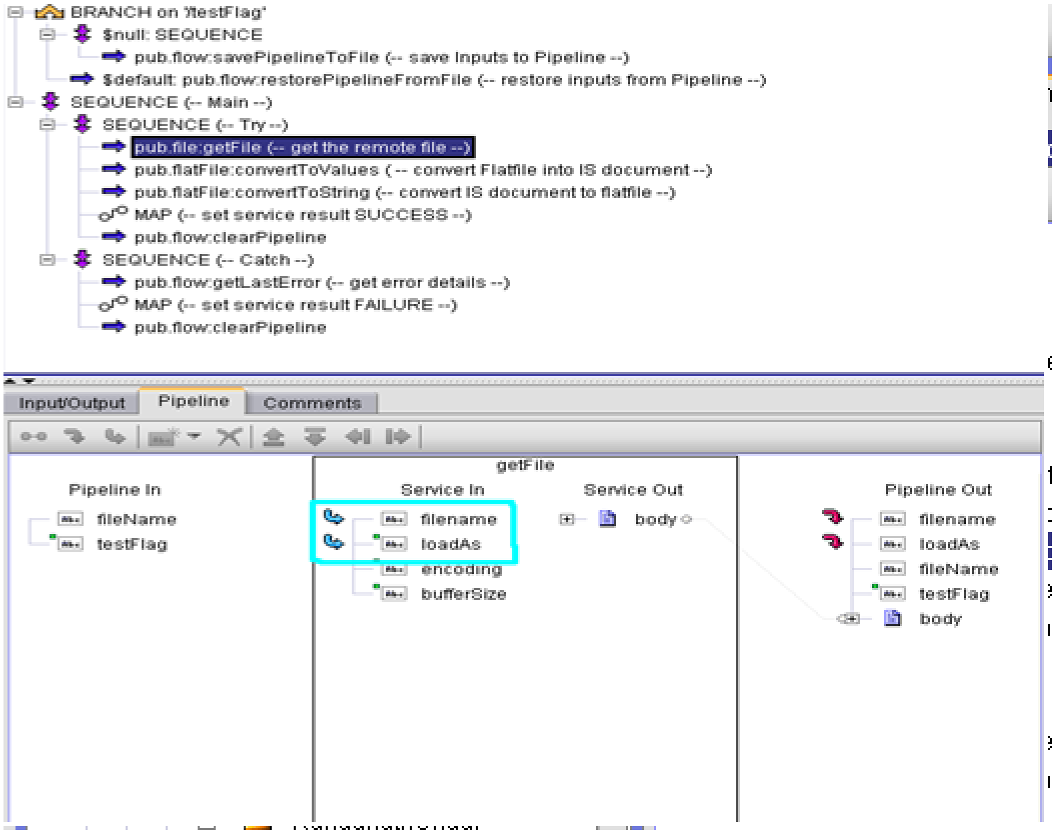
Steps to create flow service:

1) Right click on services folder and select **Flow Service**.
2) Type the unique name for service and press enter.
3) Have the following steps in the flow service.



```
BRANCH on '/testFlag'
    $null: SEQUENCE
        pub.flow:savePipelineToFile (-- save Inputs to Pipeline --)
    $default: pub.flow:restorePipelineFromFile (-- restore inputs from Pipeline --)
SEQUENCE (-- Main --)
    SEQUENCE (-- Try --)
        pub.file:getFile (-- get the remote file --)
        pub.flatFile:convertToValues ( -- convert Flatfile into IS document --)
        pub.flatFile:convertToString (-- convert IS document to flatfile --)
        MAP (-- set service result SUCCESS --)
        pub.flow:clearPipeline
    SEQUENCE (-- Catch --)
        pub.flow:getLastError (-- get error details --)
        MAP (-- set service result FAILURE --)
        pub.flow:clearPipeline
```

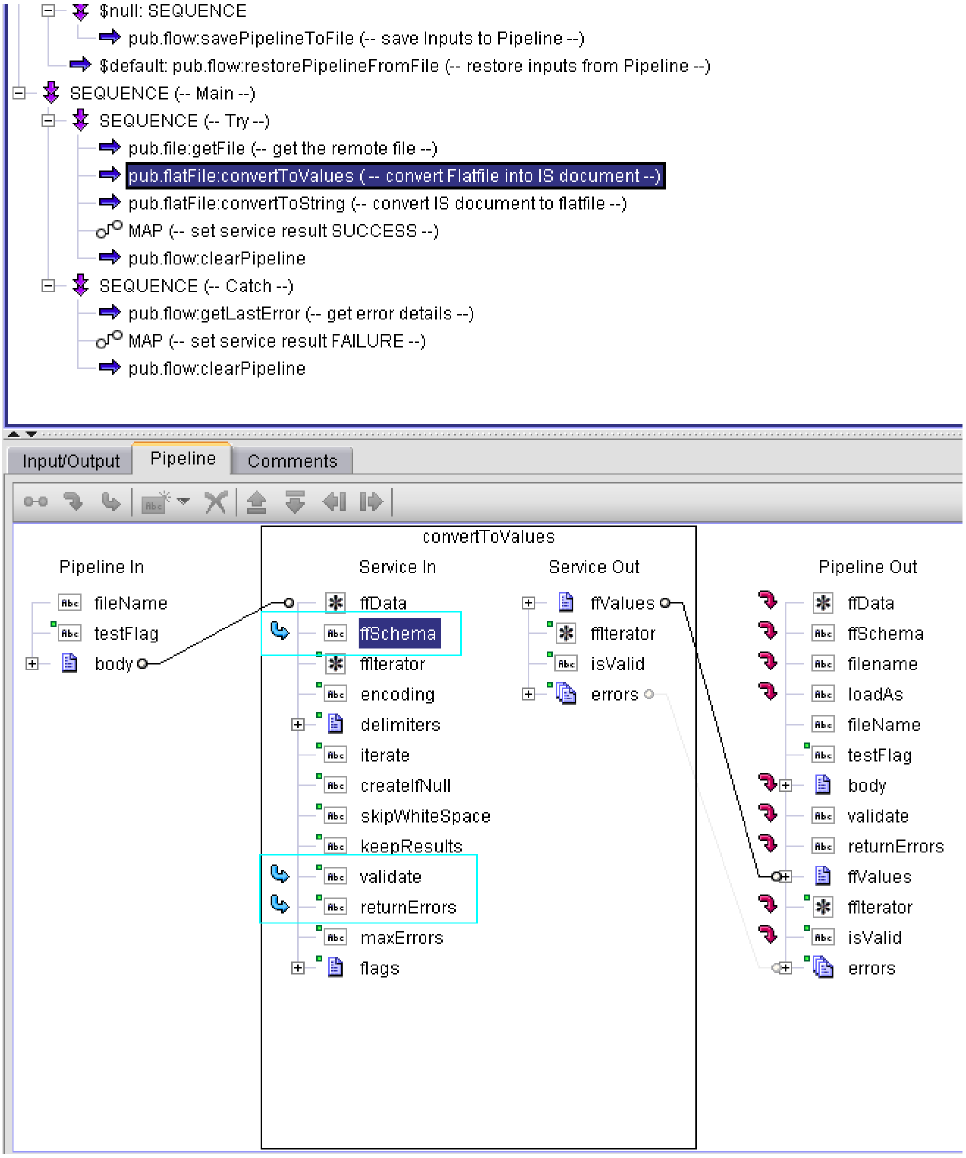
4) Click on **pub.file:getFile**. Select the **Pipeline** tab and do the following:
    - Double click on the **filename** and paste the path of where the text file is stored.
    - Double click on **laodAs** and enter bytes.



```
BRANCH on '/testFlag'
    $null: SEQUENCE
        pub.flow:savePipelineToFile (-- save Inputs to Pipeline --)
    $default: pub.flow:restorePipelineFromFile (-- restore inputs from Pipeline --)
SEQUENCE (-- Main --)
    SEQUENCE (-- Try --)
        pub.file:getFile (-- get the remote file --)
        pub.flatFile:convertToValues ( -- convert Flatfile into IS document --)
        pub.flatFile:convertToString (-- convert IS document to flatfile --)
        MAP (-- set service result SUCCESS --)
        pub.flow:clearPipeline
    SEQUENCE (-- Catch --)
        pub.flow:getLastError (-- get error details --)
        MAP (-- set service result FAILURE --)
        pub.flow:clearPipeline
```

5) Click on **pub.flatFile:convertToValues**. Select **Pipeline** tab, do the following:

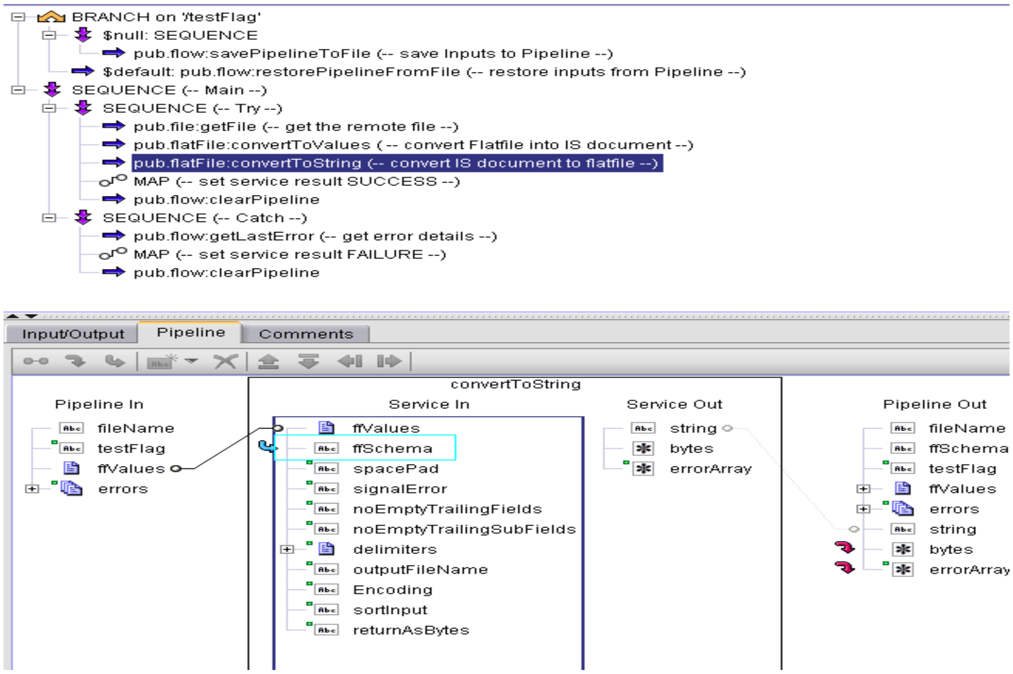
- Double click on **ffSchema** and enter the fully qualified path of the schema created previously.
- Double click on **validate** and enter true.



```
$null: SEQUENCE
    pub.flow:savePipelineToFile (-- save Inputs to Pipeline --)
  $default: pub.flow:restorePipelineFromFile (-- restore inputs from Pipeline --)
SEQUENCE (-- Main --)
  SEQUENCE (-- Try --)
    pub.file:getFile (-- get the remote file --)
    pub.flatFile:convertToValues (-- convert Flatfile into IS document --)
    pub.flatFile:convertToString (-- convert IS document to flatfile --)
    MAP (-- set service result SUCCESS --)
    pub.flow:clearPipeline
  SEQUENCE (-- Catch --)
    pub.flow:getLastError (-- get error details --)
    MAP (-- set service result FAILURE --)
    pub.flow:clearPipeline
```

6) Click on **pub.flatFile:convertToString**. Select the Pipeline tab and do the following:

- Double click on **ffSchema** and enter the fully qualified path of the schema created previously..

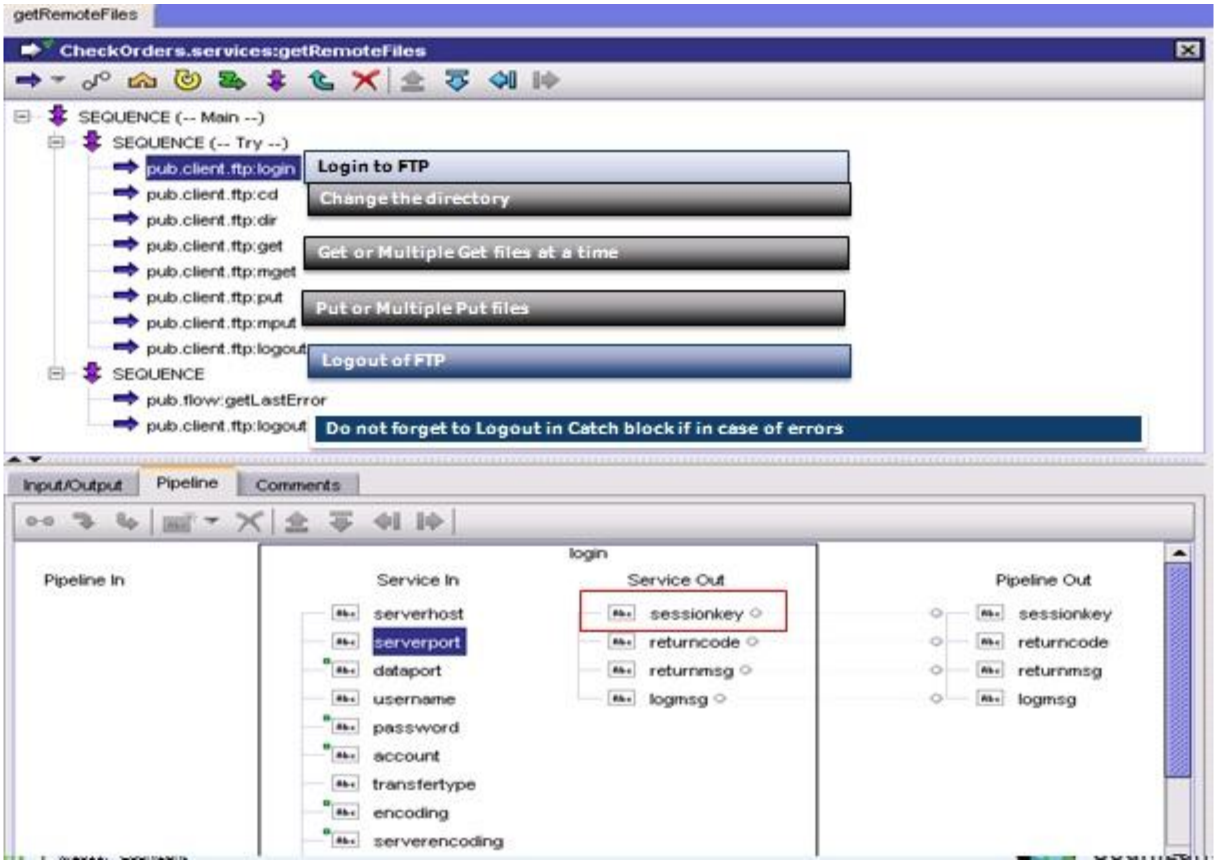


```
BRANCH on '/testFlag'
    $null: SEQUENCE
        pub.flow:savePipelineToFile (-- save Inputs to Pipeline --)
    $default: pub.flow:restorePipelineFromFile (-- restore inputs from Pipeline --)
SEQUENCE (-- Main --)
    SEQUENCE (-- Try --)
        pub.file:getFile (-- get the remote file --)
        pub.flatFile:convertToValues ( -- convert Flatfile into IS document --)
        pub.flatFile:convertToString (-- convert IS document to flatfile --)
        MAP (-- set service result SUCCESS --)
        pub.flow:clearPipeline
    SEQUENCE (-- Catch --)
        pub.flow:getLastError (-- get error details --)
        MAP (-- set service result FAILURE --)
        pub.flow:clearPipeline
```

7) Save the flow service and Run service and check for valid output.

FTP Session Creation

Create a flow service that illustrate FTP session:



```
SEQUENCE (-- Main --)
    SEQUENCE (-- Try --)
        pub.client.ftp:login
        pub.client.ftp:cd
        pub.client.ftp:dir
        pub.client.ftp:get
        pub.client.ftp:mget
        pub.client.ftp:put
        pub.client.ftp:mput
        pub.client.ftp:logout
    SEQUENCE
        pub.flow:getLastError
        pub.client.ftp:logout
```

Login to FTP

Change the directory

Get or Multiple Get files at a time

Put or Multiple Put files

Logout of FTP

Do not forget to Logout in Catch block if in case of errors

login

Pipeline In	Service In	Service Out	Pipeline Out
	serverhost	sessionkey	sessionkey
	serverport	returncode	returncode
	dataport	returnmsg	returnmsg
	username	logmsg	logmsg
	password		
	account		
	transfertype		
	encoding		
	serverencoding		