Flat Files and FTP

Recap – Day6

Learnt So Far (Recap & assessment)

- Integration Server Roles
- Integration Server Administrator
- Starting Integration Server
- Integration server folder structure
- Integration Server Package Management
- Integration Server ACL
- Integration Server Trigger Management.

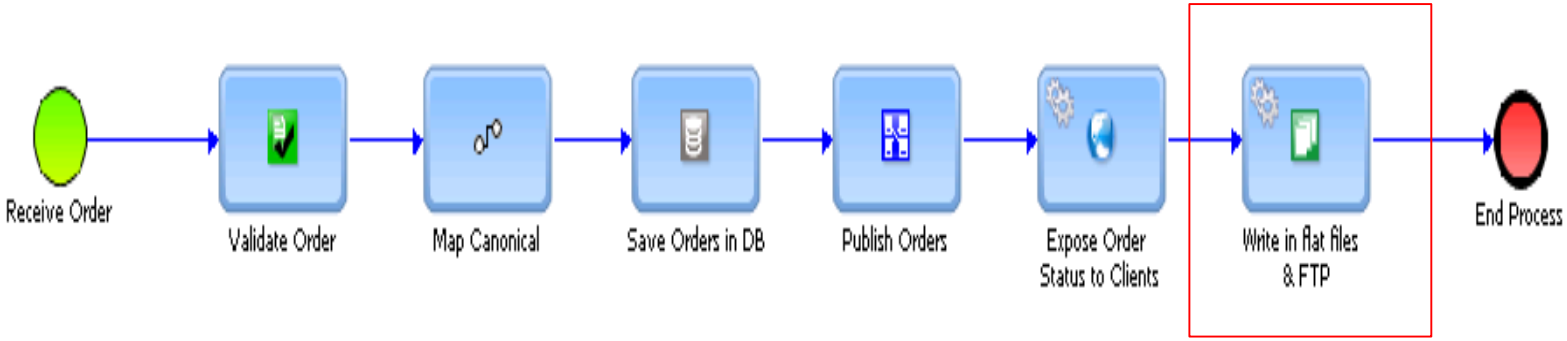
Introduction

- This course will help participants to understand the basics about Flat Files & FTP.
- Trainees would be able to handle flat files, dictionaries and schemas and FTP them to share folders

Objectives

- Get hands on experience in Flat Files
- Theoretical clarity on concepts Flat Files & FTP.

webMethods Pilot Project Progress



<u>Outcome of this course :</u>

Trainees should be able to code a flow service which captures the OrderCustomer information, writes it into a flat file and FTP into a shared path as part of the Practical session

Software versions

- This class focuses on the version 8 of the webMethods suite

webMethods Integration Server 8

webMethodsBroker 8

Software AG Designer 8

Software AG Developer 8

Chapters

Day 7

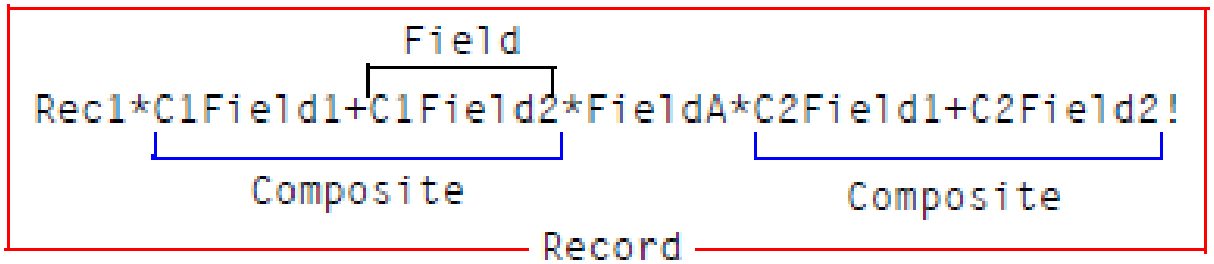
- Flat Files
- Record Parsers
- Flat File Schemas
- Flat File Dictionary
- Processing Flat Files
- FTP

Flat Files

What is a 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.

- 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.
  - Records (also known as segments) are sequences of fields and/or composites



record delimiter = !
field delimiter = *
subfield delimiter = +

webMethods supports

WmFlatFile

- The WmFlatFile package can exchange all types of flat files but can process only certain types of flat files.

- The records in the flat file are defined using one of the following methods:
  - Delimiters. Each record in the flat file is separated by a delimiter.
  - Fixed length. Each record is a fixed number of bytes (for example, mainframe punch or print records).
  - Variable length. Each record is preceded by two bytes that indicate the length of the record. Records in the flat file can have different lengths

- 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.

webMethods supports

WmFlatFile

- The WmFlatFile package can exchange all types of flat files but can process only certain types of flat files.

- The records in the flat file are defined using one of the following methods:
  - Delimiters. Each record in the flat file is separated by a delimiter.
  - Fixed length. Each record is a fixed number of bytes (for example, mainframe punch or print records).
  - Variable length. Each record is preceded by two bytes that indicate the length of the record. Records in the flat file can have different lengths

- 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.

Record Parsers

- A record parser breaks a flat file into individual records. In the WmFlatFile package, you can choose from one of its four record parsers:.
- Delimited Record Parser : This parser expects a specific delimiter to indicate the end of a record.
- Fixed Length Record Parser : This parser splits a file into records of the same pre-specified length.
- Variable Length Record Parser : This parser expects each record to be preceded by two bytes that indicate the length of the record.
- EDI Document Type Record Parser : This parser is used only for EDI flat files and provides additional functionality needed to properly parse EDI documents.

Flat File Schema

CheckOrders.ffSchemas:OrderDetailsSchema

Flat File Definition | Flat File Structure

Description:

Record Parser

- (●) Delimiter
- ( ) Fixed Length
- ( ) Variable Length
- ( ) EDI Document Type

Record

- (●) Character:
- ( ) Character position:

Field or Composite

- (●) Character:
- ( ) Character position:

Subfield

- (●) Character:
- ( ) Character position:

Quoted Release Character

- (●) Character:
- ( ) Character position:

Record Identifier

- (●) Starts at position: 0
- ( ) Nth Field: 0

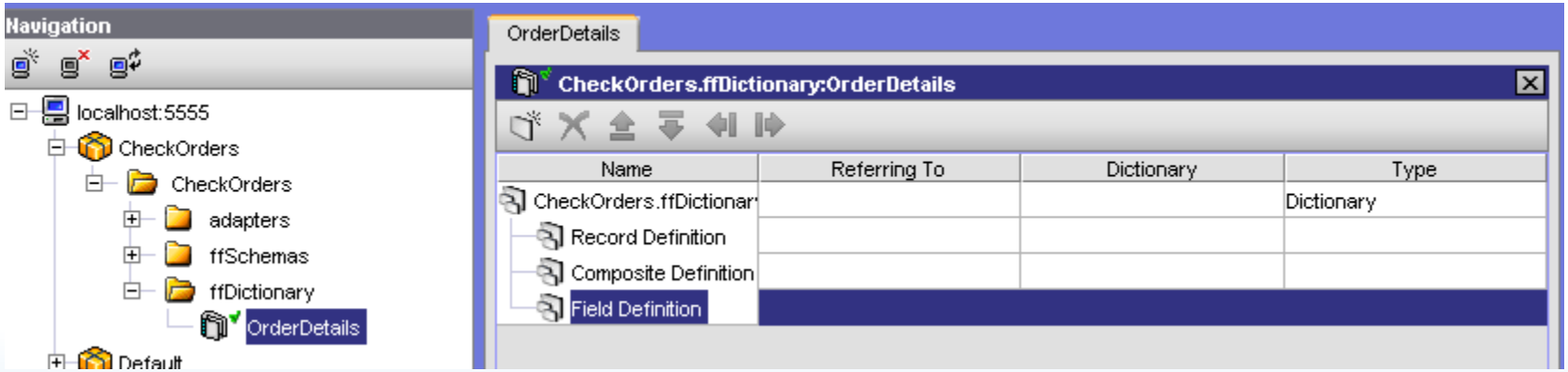
Flat File Dictionary

- A flat file schema can contain either record definitions or references to record definitions that are stored elsewhere in the namespace in a *flat file dictionary*.

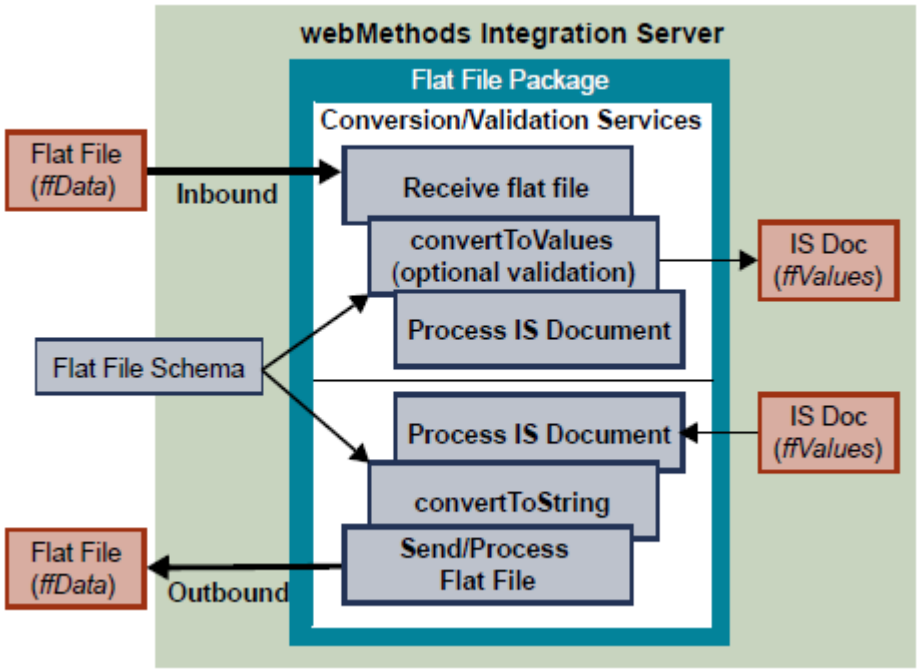
- A flat file dictionary is simply a repository for elements that you reference from flat file schemas.

- This allows you to create record definitions in a dictionary that can be used across multiple flat file schemas.

- Reusing record definitions reduces the amount of memory consumed by a flat file schema. When you change a definition in a flat file dictionary that is referenced in multiple flat file schemas, the element definition is updated automatically in all of the flat file schemas.



Processing Flat Files



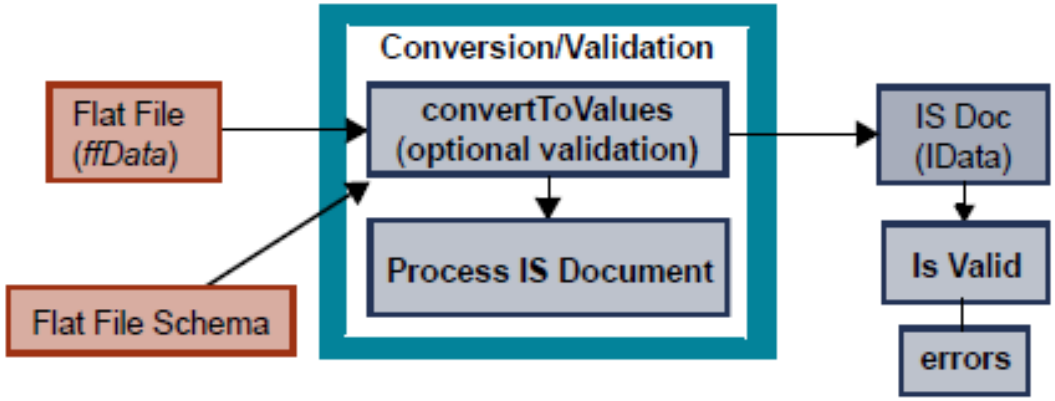
Item	Description
Flat File (*ffData*)	Flat file input with type of String, InputStream, or ByteArray (received by the Integration Server via File Polling, HTTP, FTP, etc. For more information, see Chapter 4, "Sending and Receiving Flat Files" in this guide.)
Conversion/ Validation Services	WmFlatFile package provides services (pub.flatFile:convertToValues and pub.flatFile:convertToString) that parse, validate, and convert inbound flat files to IS documents (IData objects), and construct outbound documents from IS documents. For more information about these services, see *webMethods Integration Server Built-In Services Reference*.
Flat File Schema	Flat File schema containing the structure of the flat file.
IS Doc (ffValues)	An IS document (IData object) represents the structure and values of your flat file for eventual mapping or other processing. All inbound documents must be converted to IS documents before being processed further.

Inbound Flat Files

- The WmFlatFile package provides the pub.flatFile:convertToValues service, which uses a flat file schema to parse flat files inbound to a webMethods Integration Server.

- The input of this service is a flat file and the name of a flat file schema, and the output of this service is an IS document (IData object).

Inbound Flat File Process

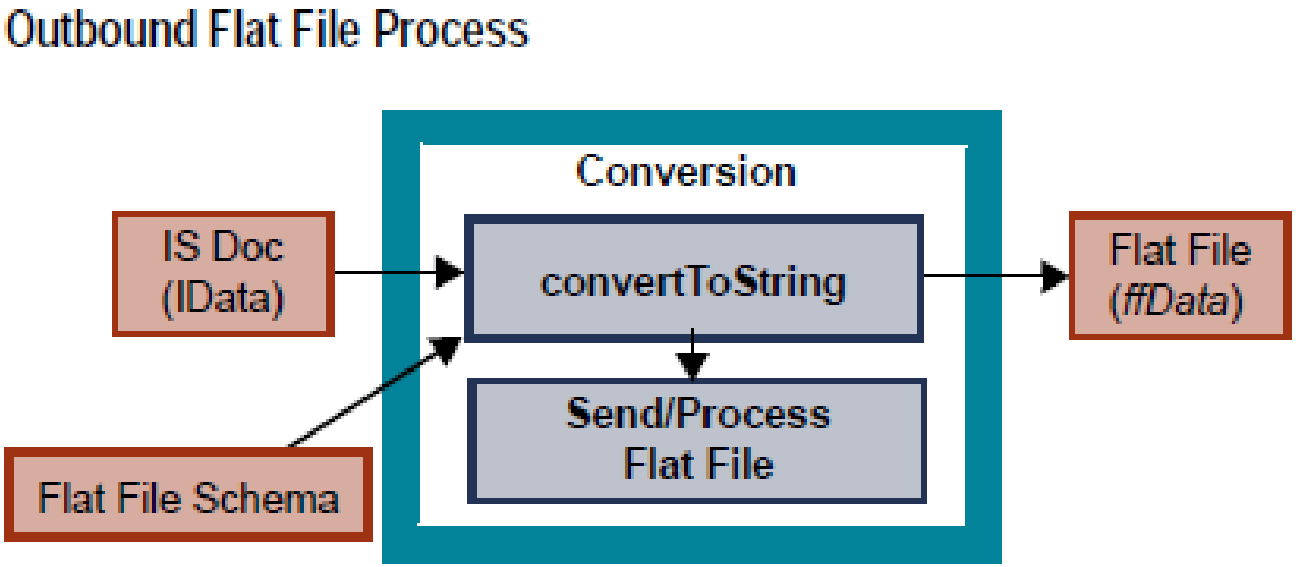


The flat file schema provides the convertToValues service with the information necessary to detect the boundaries between records and fields. After a record boundary is detected, the service matches the record in the flat file with a record defined in the flat file schema using the record identifier.

Outbound Flat Files

- The WmFlatFile package provides the pub.flatFile:convertToString service, which uses a flat file schema to create a flat file outbound from a webMethods Integration Server.

- The input of this service is an IS document (IData object) and a flat file schema, and the output of this service is a flat file.

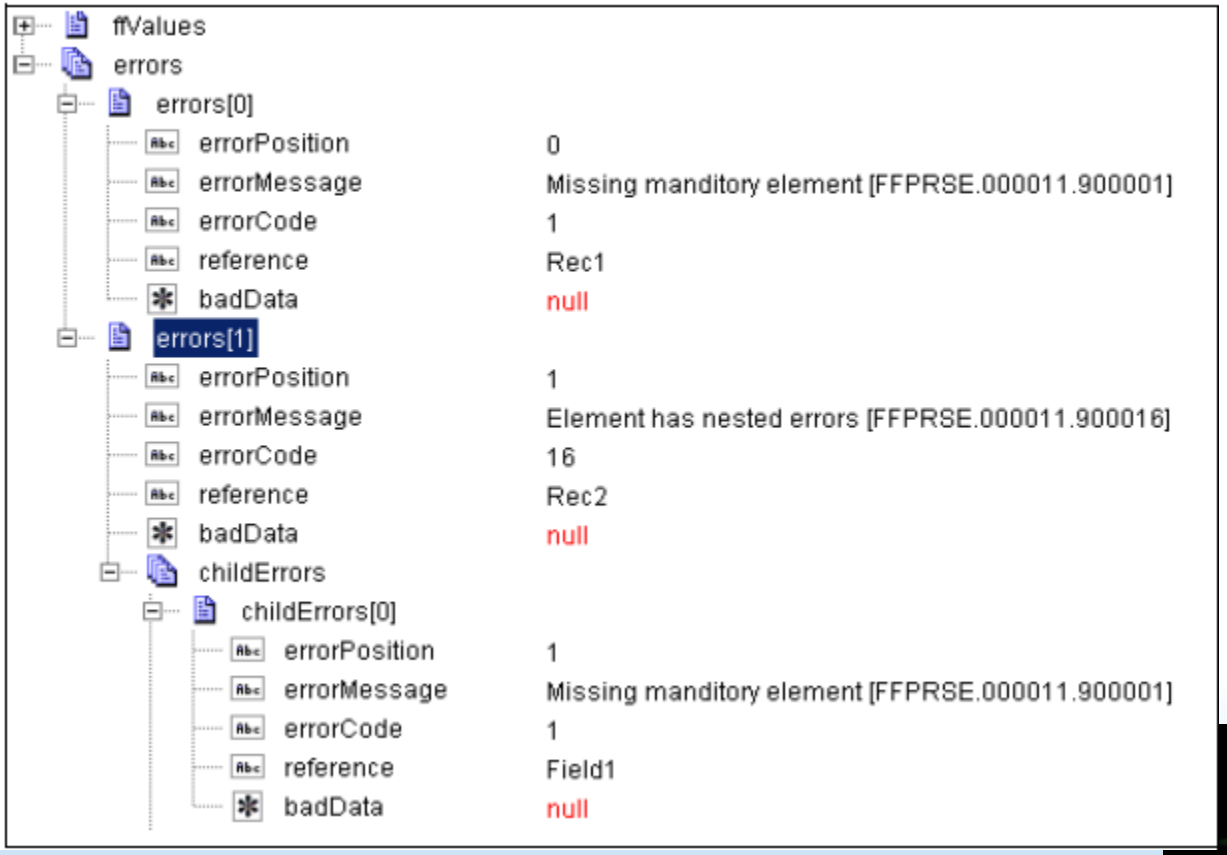


Validation errors

- When the *validate* variable of the pub.flatFile:convertToValues service is set to true and an object within the flat file does not conform to the flat file schema, the service generates errors when validating the flat file.

- If the service finds that an object is invalid, it returns validation errors in the *errors* output of the convertToValues service.

Example of Validation Results



Example – How to process flat files

- For an input Flat file sample like below.

```
PurchaseOrder*PO1+080502*Buyer+ABC Corp.+123 Main St.+Fairfax+VA+22030!
OrderDetail*100*1*1.00!
OrderDetail*200*2*2.00!
```

- Create a new Flat file dictionary define all your fields like this

Tutorial:OrderData

Name	Referring To	Dictionary	Type
Tutorial:OrderData			Dictionary
Record Definition			
Composite Definition			
Field Definition			
Address			Field Definition
City			Field Definition
CompanyDUNS			Field Definition
CompanyName			Field Definition
Date			Field Definition
DocumentID			Field Definition
Price			Field Definition
ProductID			Field Definition
Quantity			Field Definition
State			Field Definition
Zip			Field Definition

Example – How to process flat files

10. In Flat file dictionary define all your Composites like this

```
PurchaseOrder*PO1+080502*Buyer+ABC Corp.+123 Main St.+Fairfax+VA+22030!
OrderDetail*100*1*1.00!
OrderDetail*200*2*2.00!
```

Composite

Name	Referring To	Dictionary	Type
Tutorial:OrderData			Dictionary
Record Definition			
Composite Definition			
Buyer			Composite Definition
CompanyDUNS	CompanyDUNS	Tutorial:OrderData	Field Reference
CompanyName	CompanyName	Tutorial:OrderData	Field Reference
Address	Address	Tutorial:OrderData	Field Reference
City	City	Tutorial:OrderData	Field Reference
State	State	Tutorial:OrderData	Field Reference
Zip	Zip	Tutorial:OrderData	Field Reference
Header			Composite Definition
DocumentID	DocumentID	Tutorial:OrderData	Field Reference
Date	Date	Tutorial:OrderData	Field Reference
Field Definition			

Example – How to process flat files

10. In Flat file dictionary define all your Records like this

```
PurchaseOrder*PO1+080502*Buyer+ABC Corp.+123 Main St.+Fairfax+VA+22030!
OrderDetail*100*1*1.00!
OrderDetail*200*2*2.00!
```

Name	Referring To	Dictionary	Type
Tutorial:OrderData			Dictionary
Record Definition			
Order			Record Definition
Header	Header	Tutorial:OrderData	Composite Reference
DocumentID	DocumentID	Tutorial:OrderData	Field Reference
Date	Date	Tutorial:OrderData	Field Reference
Buyer	Buyer	Tutorial:OrderData	Composite Reference
CompanyDUNS	CompanyDUNS	Tutorial:OrderData	Field Reference
CompanyName	CompanyName	Tutorial:OrderData	Field Reference
Address	Address	Tutorial:OrderData	Field Reference
City	City	Tutorial:OrderData	Field Reference
State	State	Tutorial:OrderData	Field Reference
Zip	Zip	Tutorial:OrderData	Field Reference
OrderDetail			Record Definition
ProductID	ProductID	Tutorial:OrderData	Field Reference
Quantity	Quantity	Tutorial:OrderData	Field Reference
Price	Price	Tutorial:OrderData	Field Reference
OrderItems			Record Definition
ProductID	ProductID	Tutorial:OrderData	Field Reference
Quantity	Quantity	Tutorial:OrderData	Field Reference
Price	Price	Tutorial:OrderData	Field Reference

Example – How to process flat files

10. In Flat file dictionary define all your Records like this

```
PurchaseOrder*PO1+080502*Buyer+ABC Corp.+123 Main St.+Fairfax+VA+22030!
OrderDetail*100*1*1.00!
OrderDetail*200*2*2.00!
```

Name	Referring To	Dictionary	Type
Tutorial:OrderData			Dictionary
Record Definition			
Order			Record Definition
Header	Header	Tutorial:OrderData	Composite Reference
DocumentID	DocumentID	Tutorial:OrderData	Field Reference
Date	Date	Tutorial:OrderData	Field Reference
Buyer	Buyer	Tutorial:OrderData	Composite Reference
CompanyDUNS	CompanyDUNS	Tutorial:OrderData	Field Reference
CompanyName	CompanyName	Tutorial:OrderData	Field Reference
Address	Address	Tutorial:OrderData	Field Reference
City	City	Tutorial:OrderData	Field Reference
State	State	Tutorial:OrderData	Field Reference
Zip	Zip	Tutorial:OrderData	Field Reference
OrderDetail			Record Definition
ProductID	ProductID	Tutorial:OrderData	Field Reference
Quantity	Quantity	Tutorial:OrderData	Field Reference
Price	Price	Tutorial:OrderData	Field Reference
OrderItems			Record Definition
ProductID	ProductID	Tutorial:OrderData	Field Reference
Quantity	Quantity	Tutorial:OrderData	Field Reference
Price	Price	Tutorial:OrderData	Field Reference

Example – How to process flat files

10. Create a new Flat File Schema can be created using the **References**

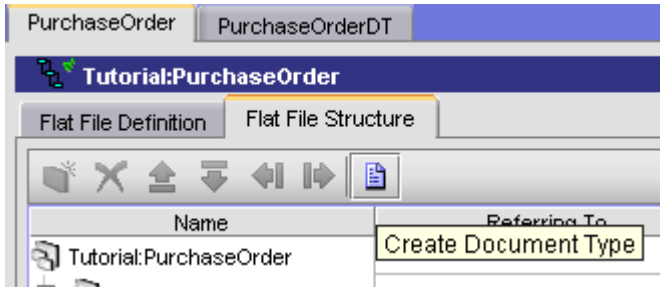
```
PurchaseOrder*PO1+080502*Buyer+ABC Corp.+123 Main St.+Fairfax+VA+22030!
OrderDetail*100*1*1.00!
OrderDetail*200*2*2.00!
```

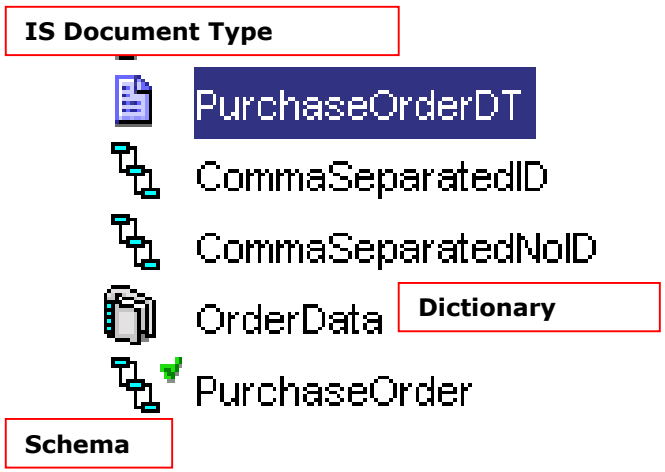
Tutorial:PurchaseOrder

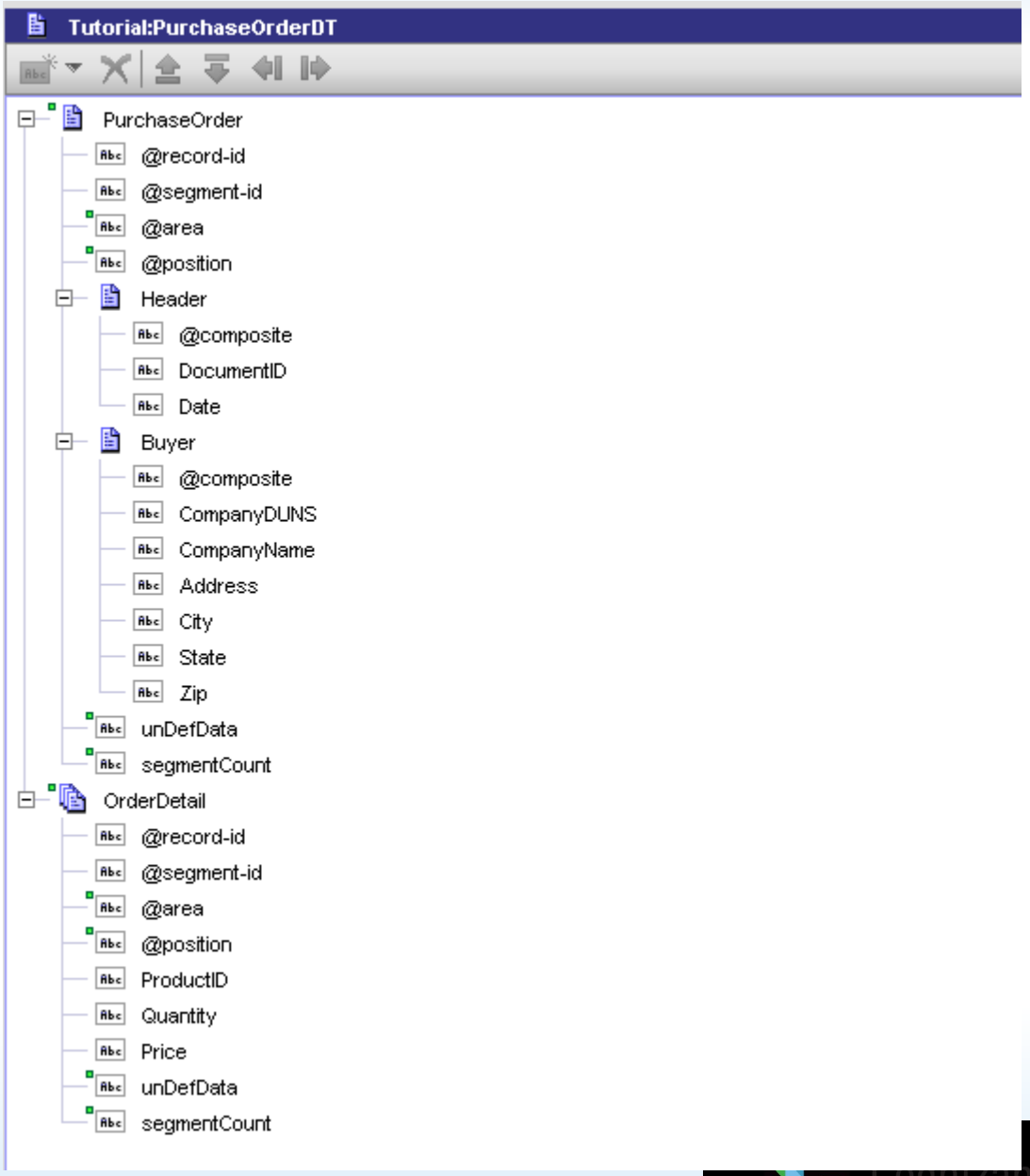
Flat File Definition | Flat File Structure

Name	Referring To	Dictionary	Type
Tutorial:PurchaseOrder			Schema Definition
PurchaseOrder	Order	Tutorial:OrderData	Record Reference
Header	Header	Tutorial:OrderData	Composite Reference
DocumentID	DocumentID	Tutorial:OrderData	Field Reference
Date	Date	Tutorial:OrderData	Field Reference
Buyer	Buyer	Tutorial:OrderData	Composite Reference
CompanyDUNS	CompanyDUNS	Tutorial:OrderData	Field Reference
CompanyName	CompanyName	Tutorial:OrderData	Field Reference
Address	Address	Tutorial:OrderData	Field Reference
City	City	Tutorial:OrderData	Field Reference
State	State	Tutorial:OrderData	Field Reference
Zip	Zip	Tutorial:OrderData	Field Reference
OrderDetail	OrderDetail	Tutorial:OrderData	Record Reference
ProductID	ProductID	Tutorial:OrderData	Field Reference
Quantity	Quantity	Tutorial:OrderData	Field Reference
Price	Price	Tutorial:OrderData	Field Reference

IS Document type

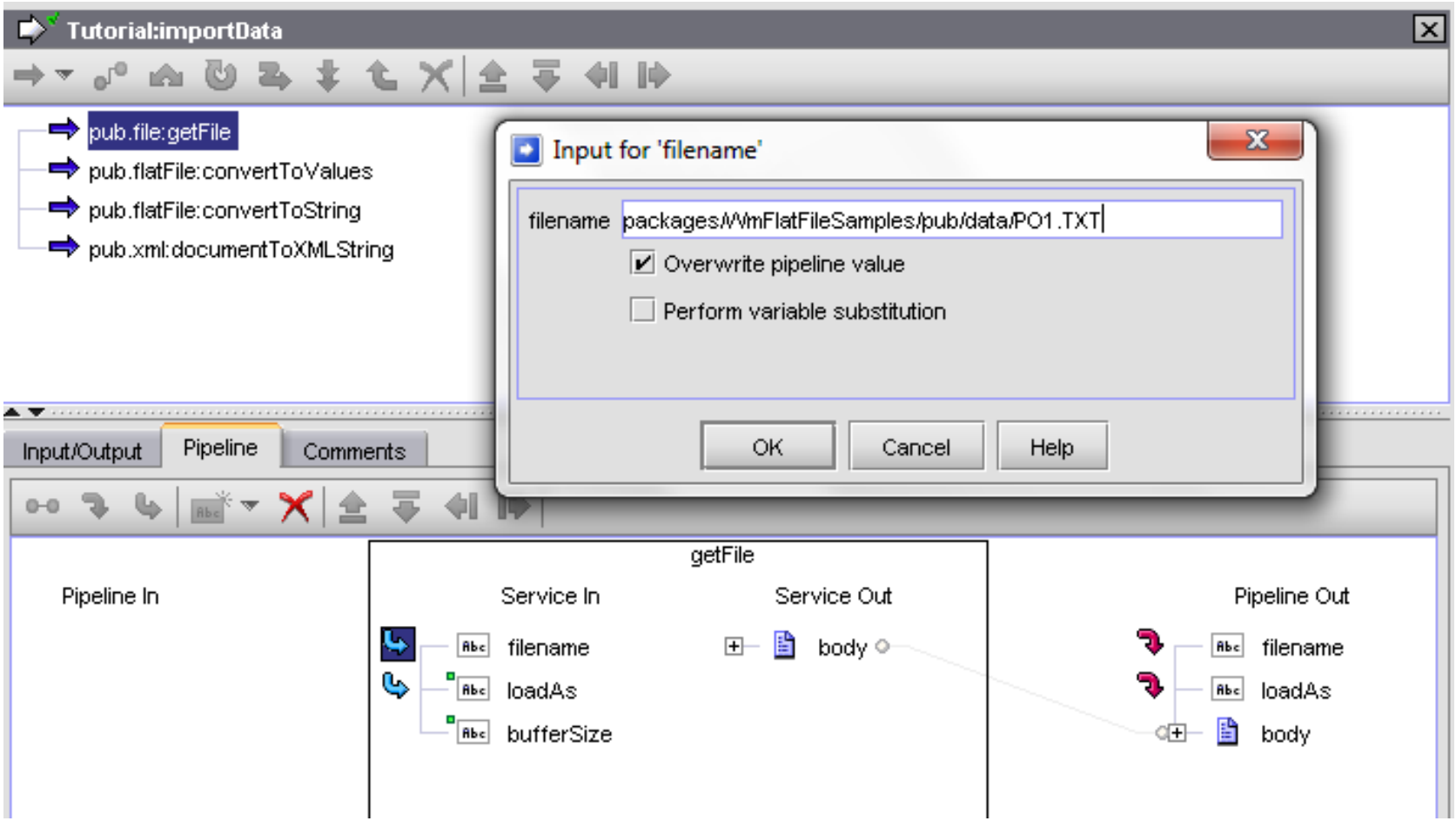




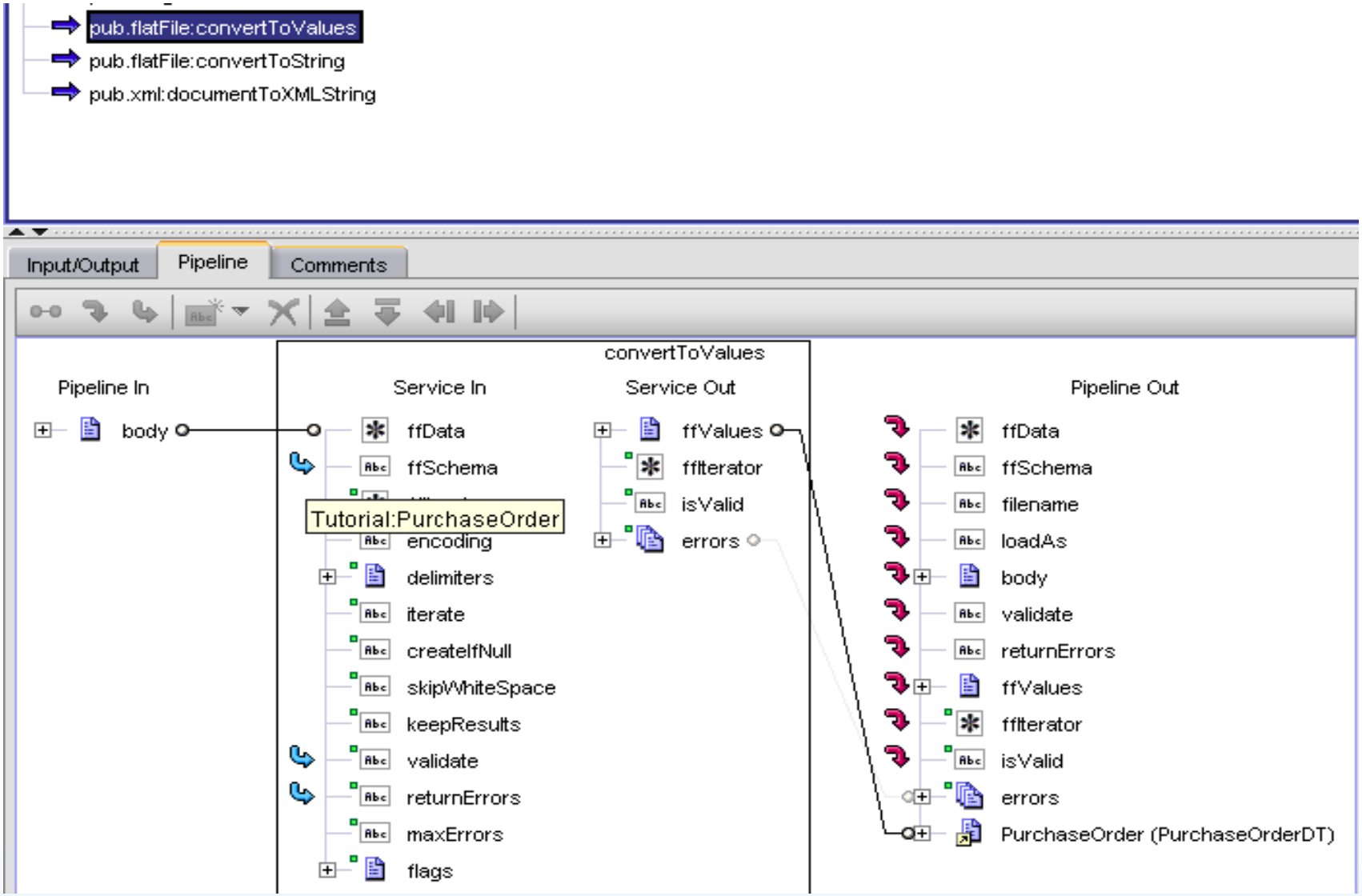


Example – Service Implementation

10. Create a new flat file processing service. Sample implementation shown below



Example – Service Implementation



Test Results

```
PurchaseOrder*PO1+080502*Buyer+ABC Corp.+123 Main St.+Fairfax+VA+22030!
OrderDetail*100*1*1.00!
OrderDetail*200*2*2.00!
```

Results

Name	Value
PurchaseOrder	
PurchaseOrder	
@record-id	PurchaseOrder
@delimiters	

Name	Value
PurchaseOrder	
@record-id	PurchaseOrder
@delimiters	
@segment-id	PurchaseOrder
Header	
@composite	yes
DocumentID	PO1
Date	080502
Buyer	
@composite	yes
CompanyDUNS	Buyer
CompanyName	ABC Corp.
Address	123 Main St.
City	Fairfax
State	VA
Zip	22030
OrderDetail	
OrderDetail[0]	
OrderDetail[1]	

FTP – webMethods Implementation

getRemoteFiles

CheckOrders.services:getRemoteFiles

- SEQUENCE (-- Main --)
  - SEQUENCE (-- Try --)
    - pub.client.ftp:login — **Login to FTP**
    - pub.client.ftp:cd — **Change the directory**
    - pub.client.ftp:dir
    - pub.client.ftp:get — **Get or Multiple Get files at a time**
    - pub.client.ftp:mget
    - pub.client.ftp:put — **Put or Multiple Put files**
    - pub.client.ftp:mput
    - pub.client.ftp:logout — **Logout of FTP**
  - SEQUENCE
    - pub.flow:getLastError
    - pub.client.ftp:logout — **Do not forget to Logout in Catch block if in case of errors**

Input/Output | Pipeline | Comments

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

Summary

What have we learnt today ?

- Flat File
- Flat File Schemas
- Flat File Dictionaries
- Record parsers
- Processing Flat files inbound / outbound
- FTP

Q & A

- What is a flat file ?
- What are records? Define a composite field ?
- How to process flat files inbound / outbound ? Mention the inbuilt services which will be used
- What are Record parsers.
- How to convert flat files to webMethods IS Documents ?
- How to achieve FTP in webMethods
- How to read / write data from or into flat files ?

Thank you