ChainBuilder ESB

Visual Enterprise Integration™

Version 1.2 – 2008/03/18

Getting Started Guide

Bostech Corporation
2800 Corporate Exchange Drive
Suite 260
Columbus, OH 43231

Acknowledgements

This document contains proprietary information that is the property of Bostech Corporation. Any reproduction, disclosure, or transfer of this document or the information contained herein without the express written consent of Bostech Corporation is strictly prohibited.

The use of the information contained in this document and the implementation of any of its techniques are the sole responsibility of the client and depend on the client's ability to evaluate the information and implement it into the client's operational environment.

Except for any express written warranties made by it, Bostech Corporation makes no warranties or representations with respect to any information contained herein, whether express, implied, statutory, or otherwise, in fact or in law, including without limitation, any implied warranties of merchantability or fitness for a particular purpose; and in no event shall Bostech Corporation be liable for any special, consequential, indirect, punitive, or exemplary damages in connection with the use of the information contained herein. The information contained in this document is subject to change at any time without notice.

Trademarks

The following trademarks and acknowledgments apply to the information presented in this manual:

- ChainBuilder is a registered trademark of Bostech Corporation.
- Adobe and Acrobat Reader are registered trademarks of Adobe, Inc.
- Java is a registered trademark of Sun Microsystems, Inc.
- Windows (NT, 2000, XP, and Server 2003), .NET Framework, Internet Information Services (IIS) are registered trademarks of Microsoft Corporation.

Credits

The following third-party products are used within the ChainBuilder product, and acknowledgments apply to the information presented in this manual:

- Acrobat Reader is created and licensed by Adobe, Inc.
- This product includes software developed by the Apache Software Foundation (http://www.apache.org/)
- This product includes software developed by Eclipse (http://www.eclipse.org/)

Table of Contents

1. Introduction

1.1. Overview

This guide is designed to help you quickly get ChainBuilder ESB installed and run the sample implementations. Before starting with ChainBuilder ESB, it may be helpful to present a high level overview of the product.

- ChainBuilder ESB is a set of tools and components designed to make Java Business Integration (JBI) implementation faster, easier and more reliable. It also provides functionality not commonly found in SOA systems like support for legacy data formats in addition to XML.
- ChainBuilder ESB is built from the JBI specification and is container neutral. One of the great features of JBI is the ability to mix and match components. ChainBuilder ESB can be made to interoperate with any JBI compatible containers or components. The current release is bundled with ServiceMix v3.1 but other containers can be used. Future releases of ChainBuilder ESB will be certified with additional environments.
- There are two main features of ChainBuilder ESB
    1. A set of graphical user interfaces for creating and manipulating JBI artifacts. The main interfaces are the flow editor, format editor and map editor. There are also specialized format editors, such as X12 and HL7, with additional standard formats to be added based on customer requirements.
    2. A set of JBI components and libraries that provide valuable features for building implementations. These features include support for legacy formats (X12, HL7, CSV and others), a mapping component, user scripting attached to any component, error handling and others.

1.1.1. Prerequisites

The user should have a basic understanding of JBI and Service Oriented Architecture. However, the system is designed to be self contained and easy to install. Also, the sample implementations provide a good basis for learning how JBI works.

1.2 Sample Use Case

The Use Case in this Gettting Started manual covers a simple case of reading an XML file and mapping the contents into an XML format. It will give you a good opportunity to open and use each tool. You can build on this knowledge to create other translations and applications that leverage ChainBuilder ESB's unique abilitys to integrate older applications with fixed, comma delimited and other formats into an SOA environment. Check out the documentation available on the ChainForge.net in the ChainBuilder Resources area to see if additional Use Case studies are available for the application you are working on.

2. Installation

2.1. Obtaining the Software

An open source version of ChainBuilder ESB can be downloaded from:
http://download.chainforge.net

This download is licensed under the common open source General Public License (GPL). The formal terms of the GPL license can be found in the license text file included with the software or on the GNU GPL site at: http://www.gnu.org/copyleft/gpl.html.

A flexible and affordable ChainBuilder ESB commercial license is also available. For more information about alternative licensing for ChainBuilder ESB, contact Bostech Corporation at info@bostechcorp.com.

2.2. Quick Install on Windows

Unzip and run the install executable. The executable has the following naming convention which includes the product release number:

cbesb-n.n_xxxxxx_install.exe

(where n.n is the release number and xxxxxx indicates a build date)

If running on Microsoft Windows Vista, right-click on the executable and select to run it in compatibility mode.

Prior to installing, install Sun's Java 5 JDK (if not already installed), and set the JAVA_HOME environment variable to the location of the JDK.

During the first portion of the installation process you may experience a delay as the executable expands to its full file size. Follow the prompts provided within the installer. If you choose to modify the default install location, select a directory without any spaces in the path. Installation should only take a few minutes.

After installation, you must **reboot your system** to enable the new environment settings on Windows.

2.3. Quick Installation on Linux

The download package is a bin file which is a self-extracted executable file which can run directly on Linux. It is named as **cbesb_install.bin.**

Copy the cbersb_install.bin file into the directory where you want to install ChainBuilder ESB and run the program from a Linux Shell. When you finish the installer, it will create the directory structure of the following naming convention:

cbesb-n.n_xxxxxx
(where n.n is the release number and xxxxxx indicates a build date.)

After you finish, you can source the set_cbesb.sh script to setup ChainBuilder ESB environment variables.

3. Using the Integration Development Environment (IDE)

This section provides a variety of examples and screen walk-throughs, although it is not intended to be a detailed start-to-finish use case. For a detailed use case, see chapter 4.

3.1. Starting the Eclipse IDE

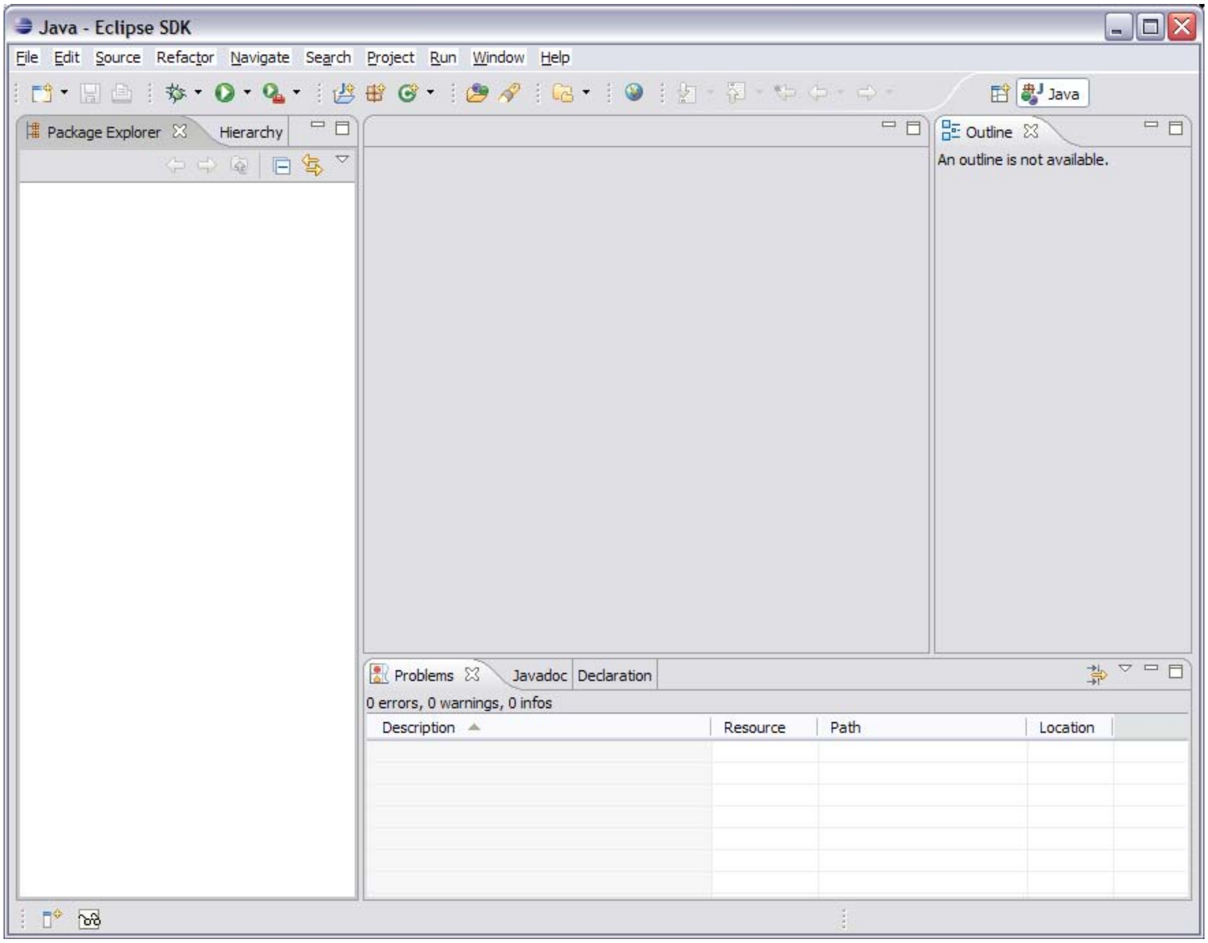
The open source Eclipse product with all the required plugins to develop a ChainBuilder ESB application was bundled within your download.

Launch ChainBuilder ESB from the Start Menu.

To launch directly, the executable is in %CBESB_HOME% directory. By default that location is %CBESB_HOME%\eclipse\eclipse.exe.

On Linux, you can lauch the ChainBuilder ESB by typing "$CBESB_HOME/eclipse/eclipse " from a Linux Shell.

At the Eclipse Welcome screen, click the X in the upper left area to close the Welcome window, and the following appears:



An outline is not available.

0 errors, 0 warnings, 0 infos

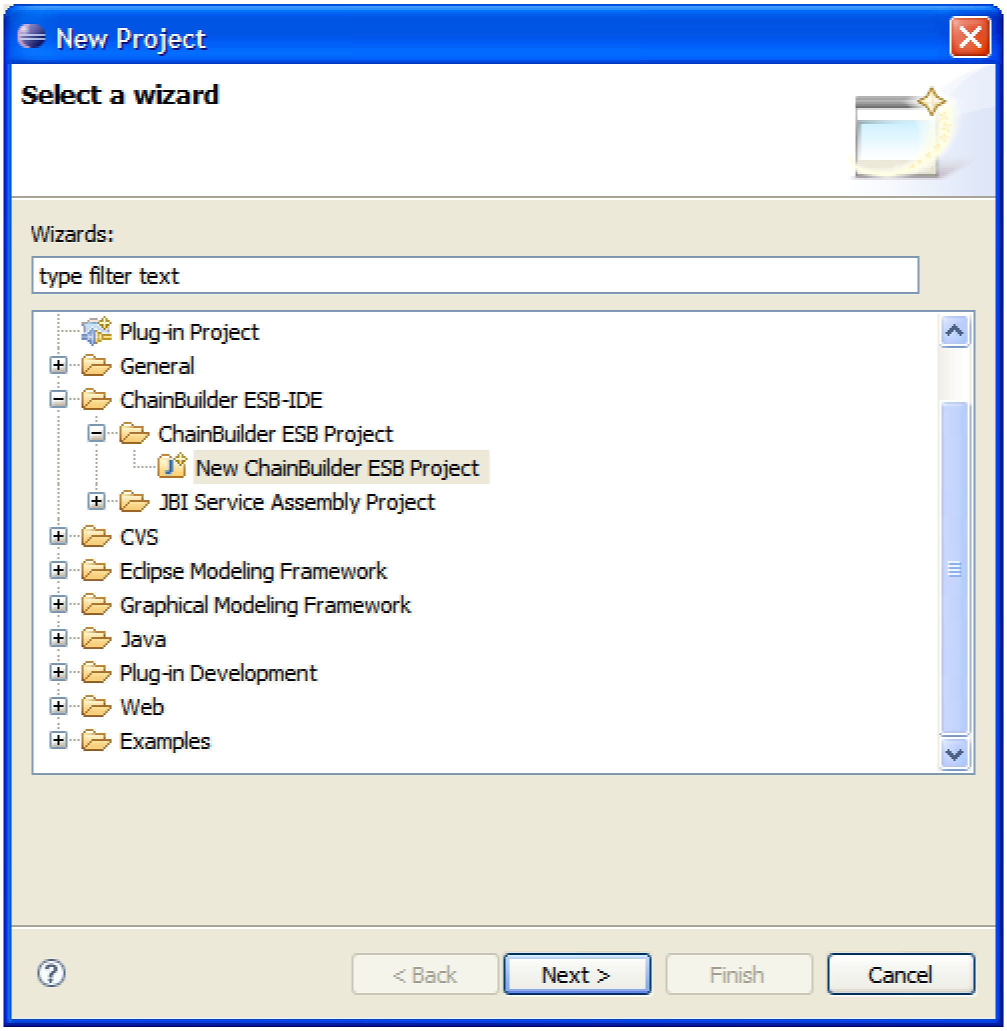
3.2. Creating the Project

A ChainBuilder ESB project can have multiple JBI Service Assembly projects.

If you are familiar with J2EE applications, a ChainBuilder ESB Project is the equivalent to a J2EE project within a J2EE application. That is, one J2EE project can contain multiple EJB projects, Web projects, or J2EE client projects. Similarly, one ChainBuilder ESB project can have multiple JBI Service Assembly projects.

Futher, the definition files created in a ChainBuilder ESB project (definition files like message format, X12 format, Schema and transformation) can be referenced by the child ChainBuilder JBI Service Assembly projects.

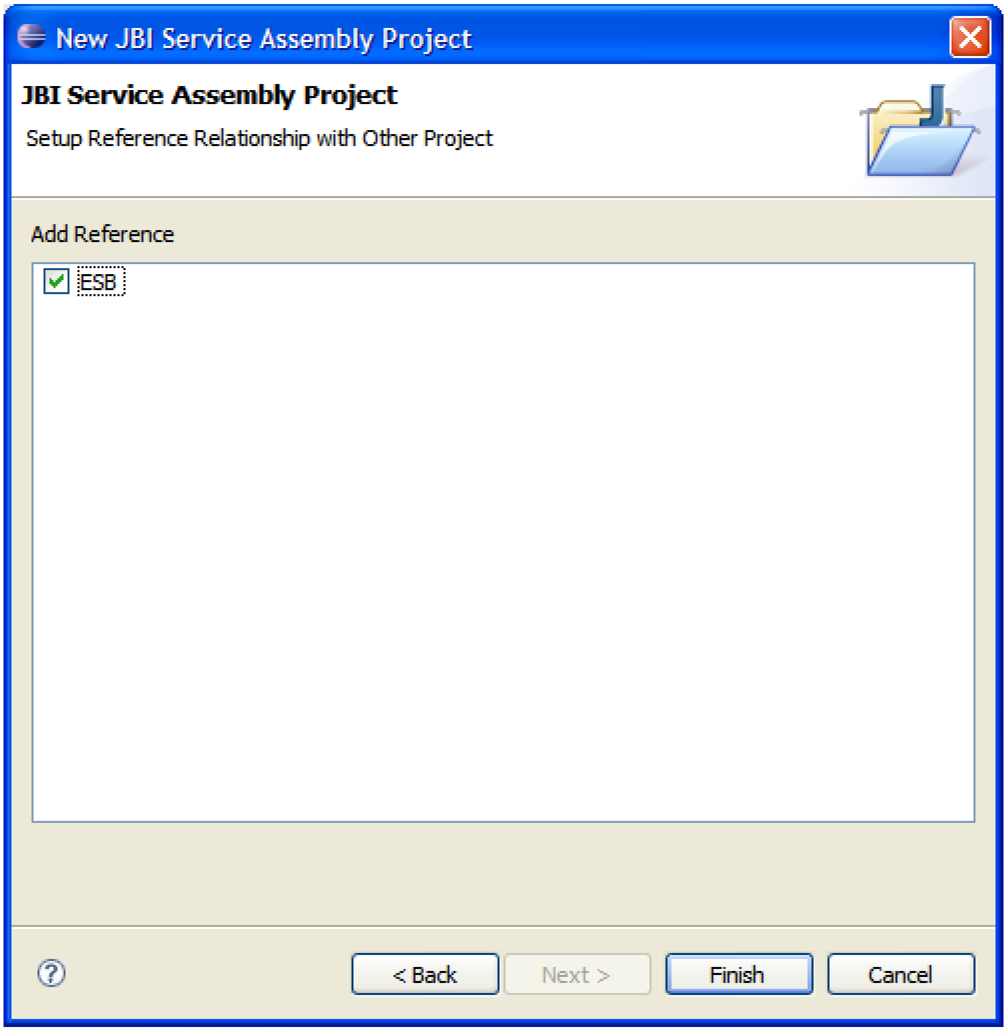
To create a new ChainBuilder ESB Project, select File → New → Project. Then choose ChainBuilder ESB-IDE → ChainBuilder ESB Project → Open New ChainBuilder ESB Project. Click through "Next>" and "Finish" buttons to create the Project.



To create a new JBI Service Assembly Project, select File → New → Project. Then choose ChainBuilder ESB IDE → JBI Service Assembly Project. Click the "Next>" button.

Name the JBI Service Assembly Project. In our example, our name is "SA". Click the "Next>" button.

The next screen allows you to associate the JBI Service Assembly with a parent ChainBuilder ESB project.



JBI Service Assembly Project

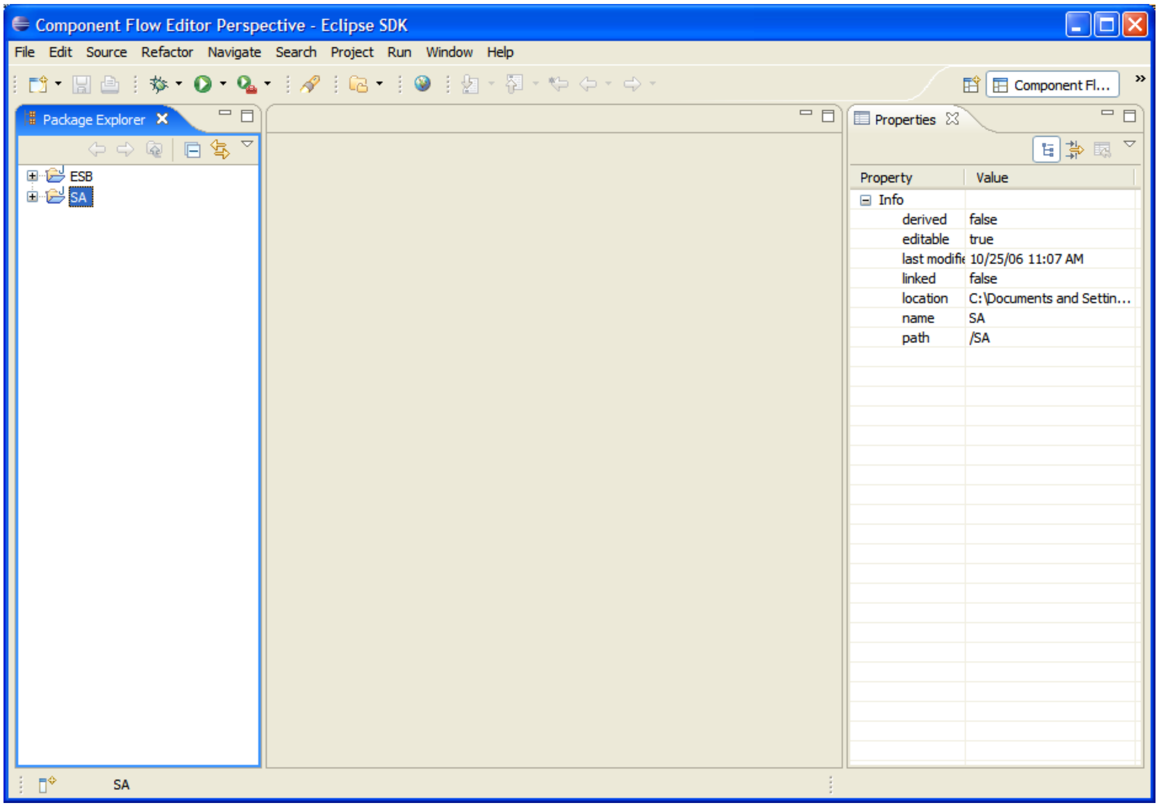
Setup Reference Relationship with Other Project

Add Reference

☑ ESB

Click the “Finish” button, and you have a new JBI Service Assembly project.

The next screen shows the result of the two newly created projects: ESB and SA.

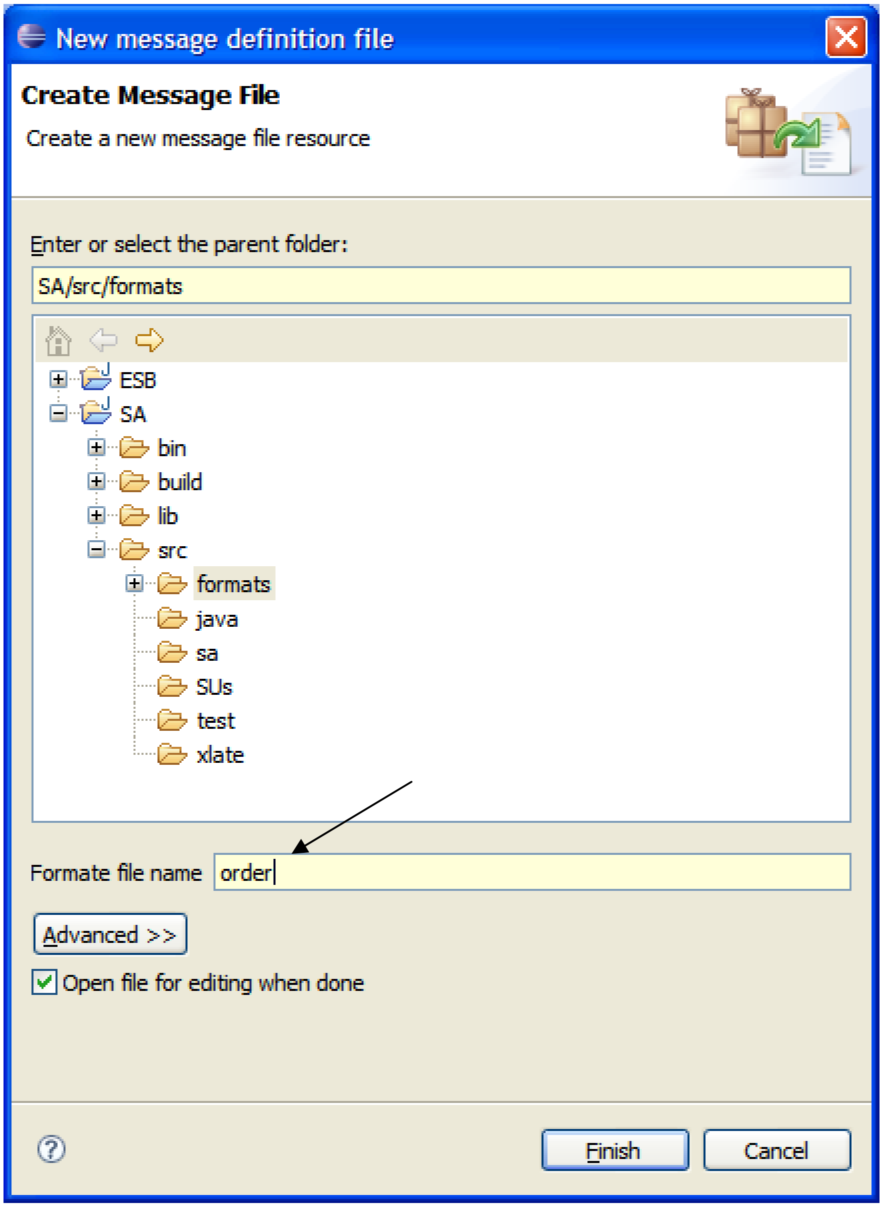

Component Flow Editor Perspective - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Component Fl...
Package Explorer
ESB
SA
Properties
SA

Property	Value
Info	
derived	false
editable	true
last modifie	10/25/06 11:07 AM
linked	false
location	C:\Documents and Settin...
name	SA
path	/SA

Please refer to *ChainBuilder ESB Reference Guide* for additional details on how to use ChainBuilder ESB IDE.

3.3. Creating the Message Definitions

To create a new fixed, delimited, or hierarchical message format definition, right-click the **src/formats** folder from the package explorer and then select **`New`** → **`Message Format File`**. Next you are prompted to name the message format.



New message definition file

Create Message File

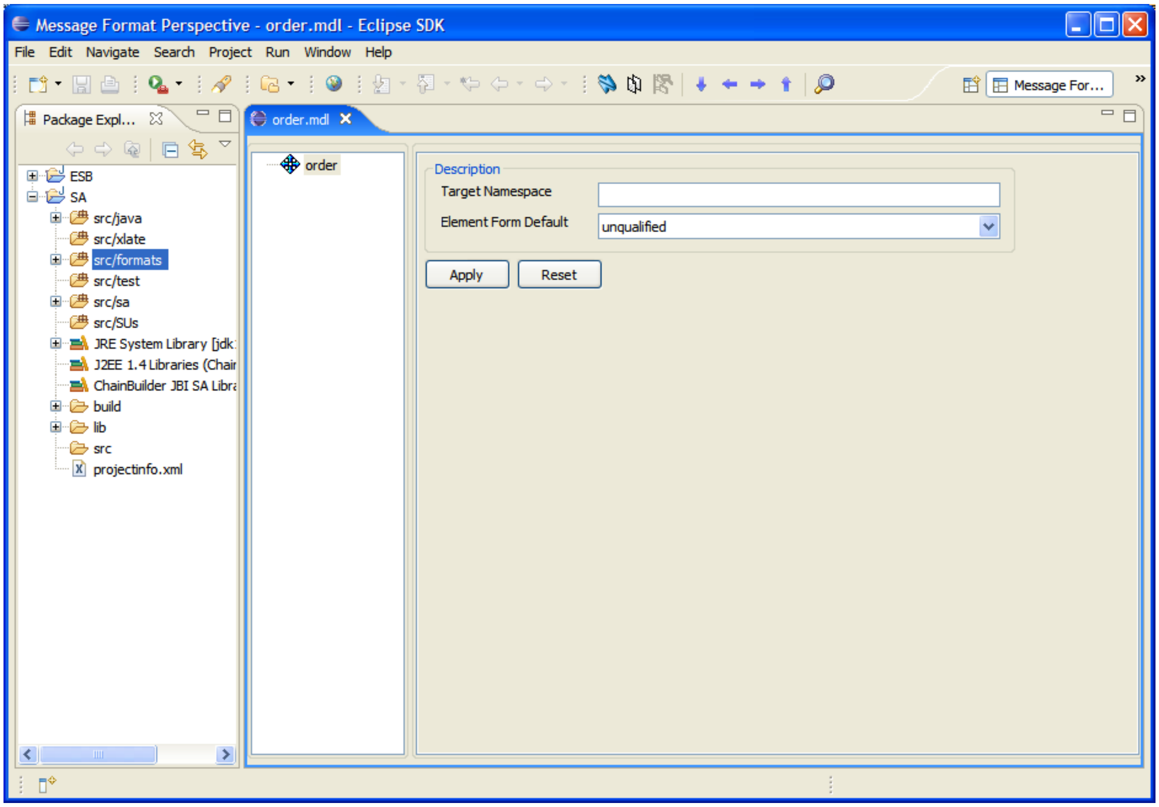
Create a new message file resource

Enter or select the parent folder: SA/src/formats

Formate file name order

☑ Open file for editing when done

Click the "Finish" button to create a message definition file. Your area is similar to this:

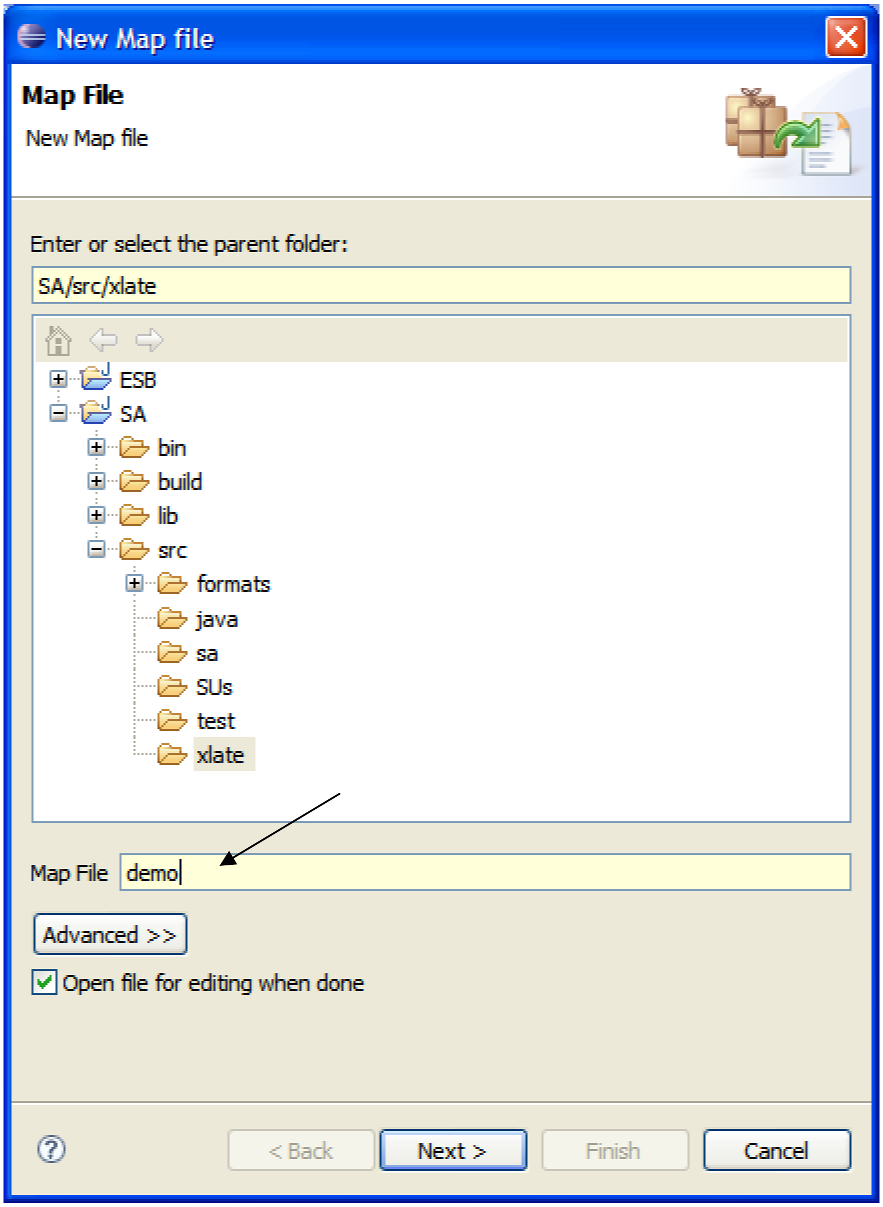


Description
Target Namespace
Element Form Default: unqualified

Please refer to *ChainBuilder ESB Message Format Editor Guide* for details on defining Message Formats.

3.4. Creating the Transformation

To create a transformation file, you right click on the **src/xlate** folder from the package explorer and select **`New`** → **`Map File`** which will prompt you to name the transformer file.



New Map file

Map File

New Map file

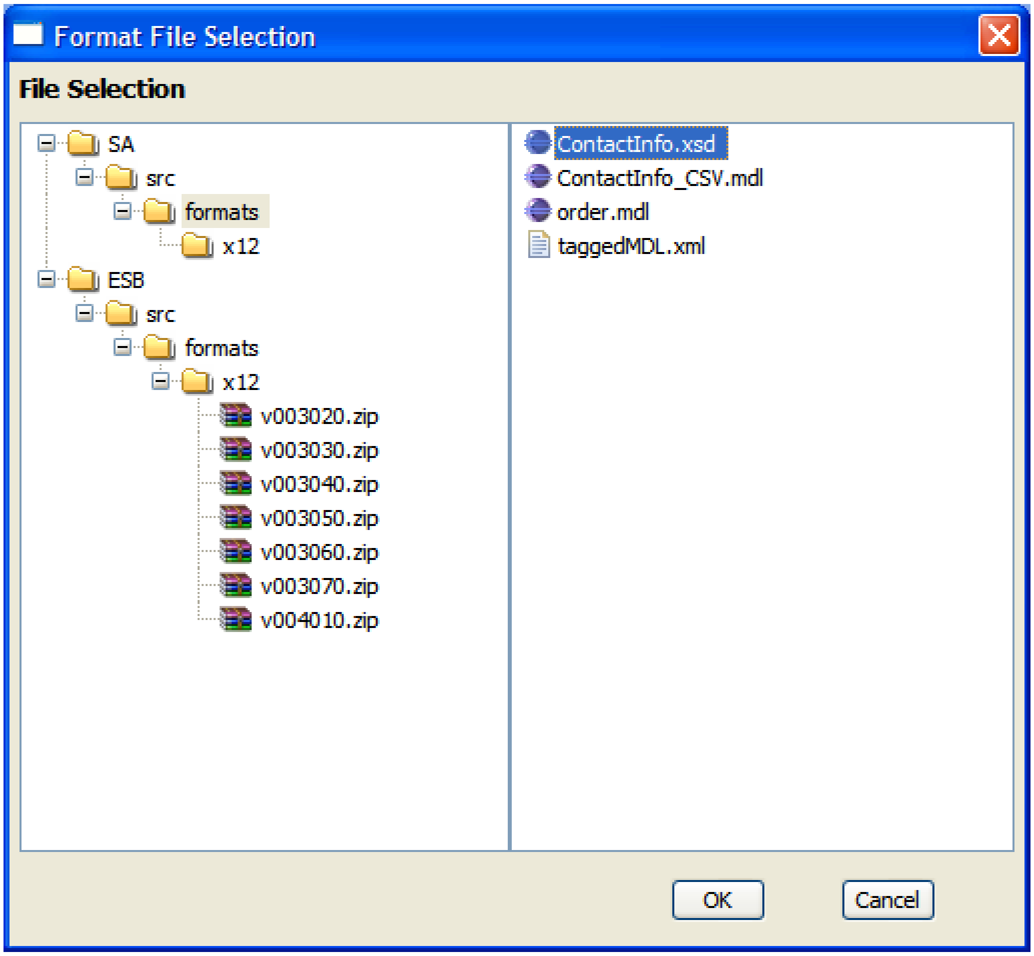
Enter or select the parent folder:

SA/src/xlate

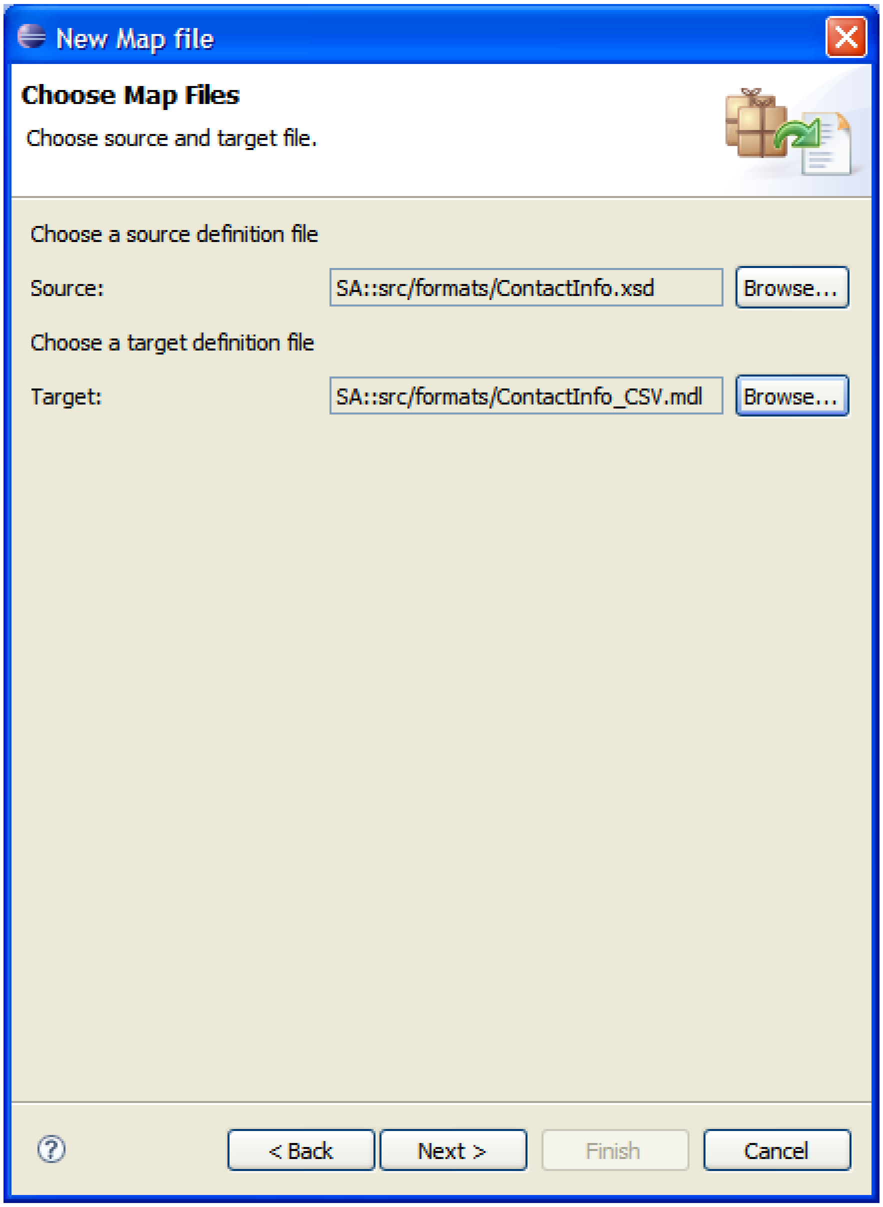
Map File demo

☑ Open file for editing when done

Select the source and target message format definitions on the next screen. They can be MDL (created in the Message Format Editor), XSD (XML Schema), HL7 or X12 message definitions.



The next screen shows the result with both source and target message definitions selected.



Choose Map Files

Choose source and target file.

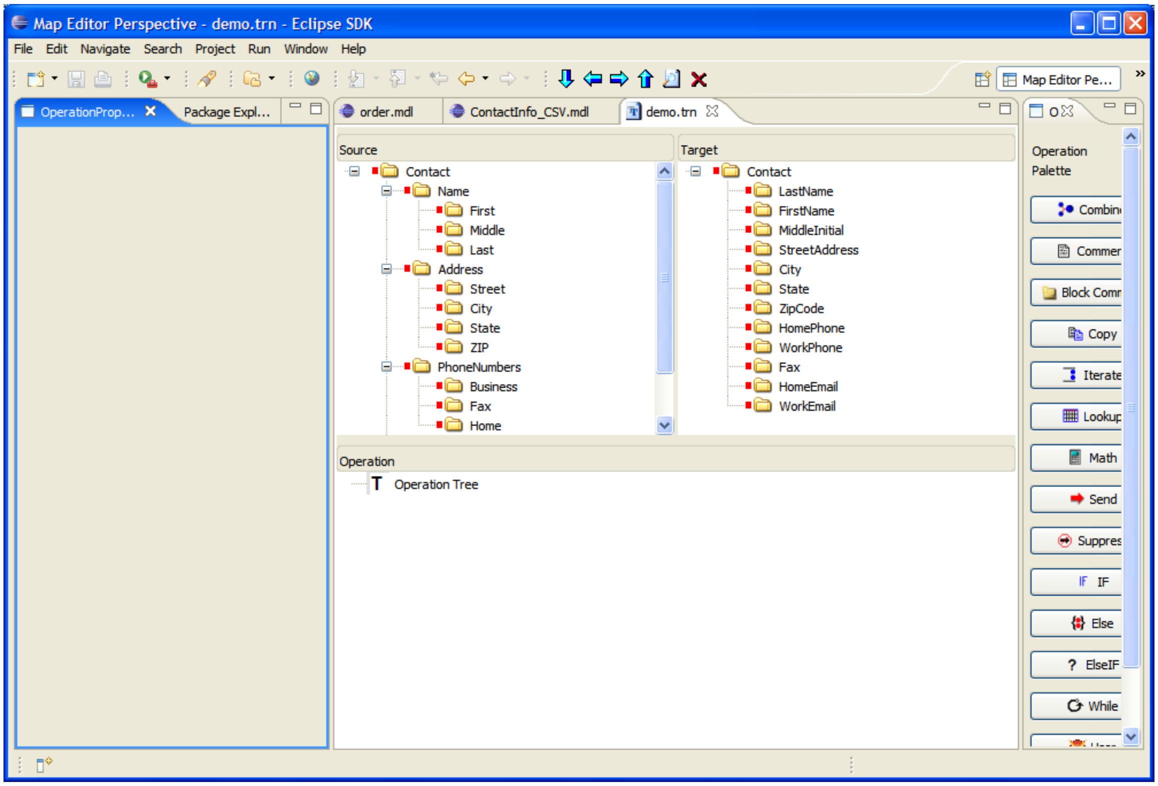
Choose a source definition file

Source: SA::src/formats/ContactInfo.xsd Browse...

Choose a target definition file

Target: SA::src/formats/ContactInfo_CSV.mdl Browse...

Click the “Next>” button and select the root message for both source and target. The following main Map Editor screen appears:

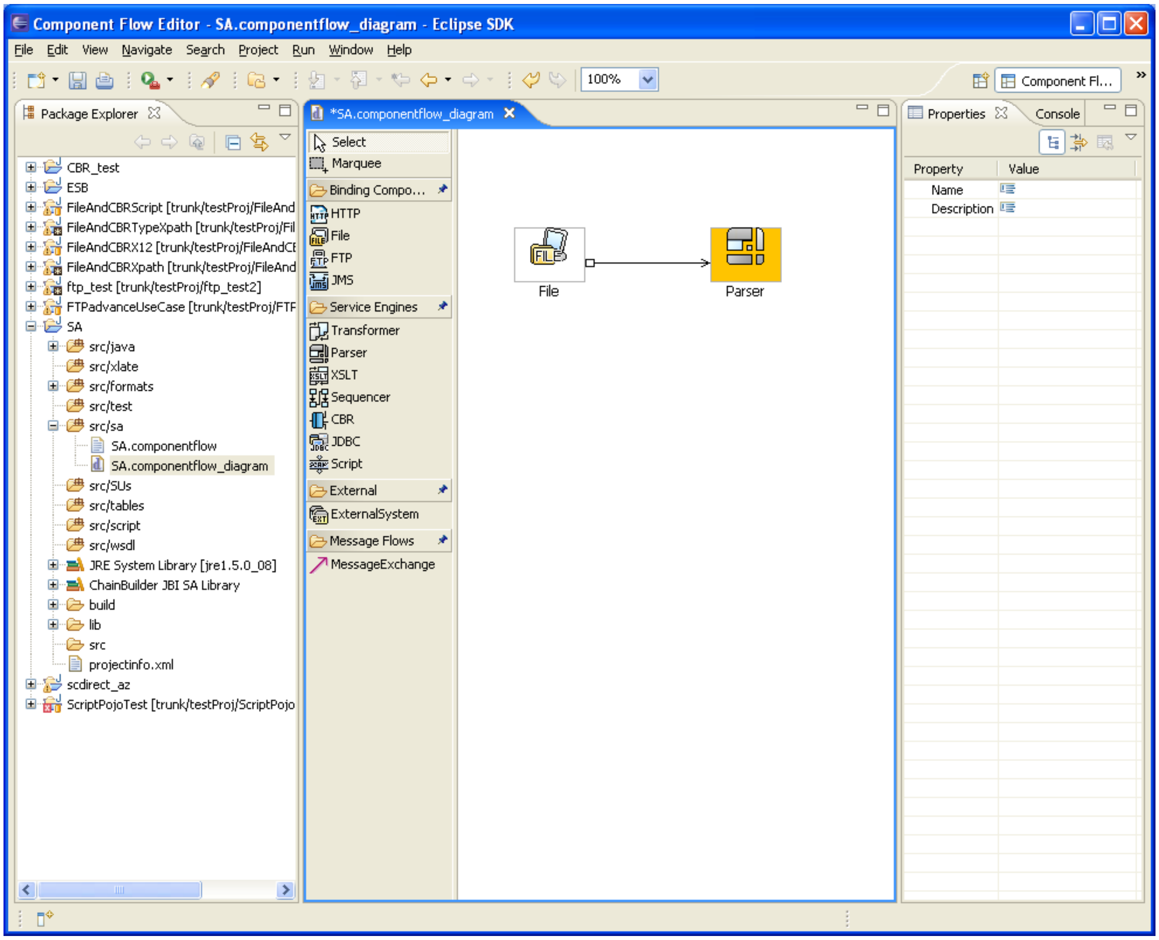


Please refer to *ChainBuilder ESB Map Editor Guide* for details how to use the Map Editor

3.5. Creating the Flow

To start the Component Flow Editor, create a new JBI Service Assembly project or open an existing JBI Service Assembly project. The **src/sa** folder of the project contains two files. Double-click on the **SA.componentflow_diagram** file, to start the Component Flow Editor.

The following screen shows the result after adding a File and Parser component and connecting them with an in-out message exchange. The technique is to **click the palette item and then the canvas**, not to drag from the palette into the layout canvas.


Component Flow Editor - SA.componentflow_diagram - Eclipse SDK
File Edit View Navigate Search Project Run Window Help
100%
Component Fl...
Package Explorer
CBR_test
ESB
FileAndCBRScript [trunk/testProj/FileAnd
FileAndCBRTypeXpath [trunk/testProj/Fil
FileAndCBRX12 [trunk/testProj/FileAndC
FileAndCBRXpath [trunk/testProj/FileAnd
ftp_test [trunk/testProj/ftp_test2]
FTPadvanceUseCase [trunk/testProj/FTF
SA
src/java
src/xlate
src/formats
src/test
src/sa
SA.componentflow
SA.componentflow_diagram
src/SUs
src/tables
src/script
src/wsdl
JRE System Library [jre1.5.0_08]
ChainBuilder JBI SA Library
build
lib
src
projectinfo.xml
scdirect_az
ScriptPojoTest [trunk/testProj/ScriptPojo
*SA.componentflow_diagram
Select
Marquee
Binding Compo...
HTTP
File
FTP
JMS
Service Engines
Transformer
Parser
XSLT
Sequencer
CBR
JDBC
Script
External
ExternalSystem
Message Flows
MessageExchange
File
Parser
Properties
Console
Property
Value
Name
Description

3.6 Deploying the JBI Service Assembly Project

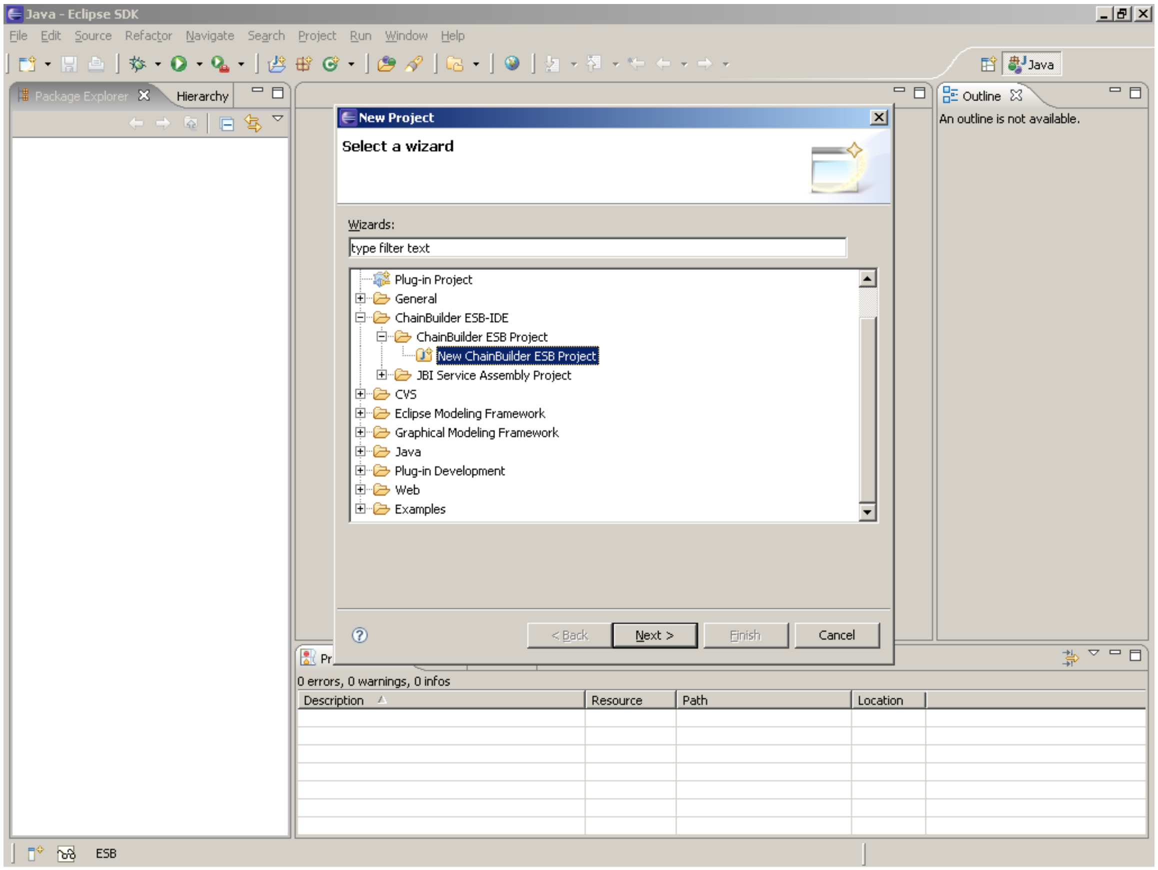
Please refer to *ChainBuilder ESB Reference Guide* for instructions on how to deploy the Service Assembly archive.

4. Step by Step Guide to Configure and Run a Simple XML to XML mapping using ChainBuilder ESB IDE

This Use Case walks the user through the process of creating a ChainBuilder ESB instance which is capable of reading an XML file containing order information (from an order-processing system) and mapping that to an XML file containing customer information for use by a CRM system.

Step 1. Create Project

Use ChainBuilder ESB IDE to create a ChainBuilder ESB project called "ESB". If you already have an ESB project titled "ESB", skip this step. The following shows the start screen:

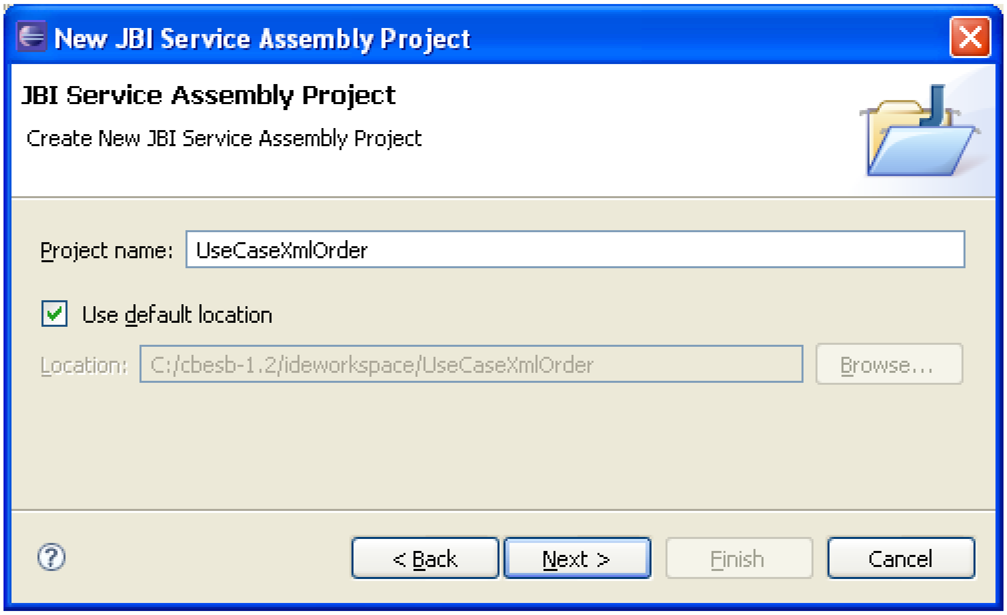


An outline is not available.

0 errors, 0 warnings, 0 infos

Step 2. Create JBI Service Assembly

Create a ChainBuilder JBI Service Assembly project named "UseCaseXmlOrder" and give it a reference to the ESB Project named "ESB". First, name the project.



New JBI Service Assembly Project

JBI Service Assembly Project

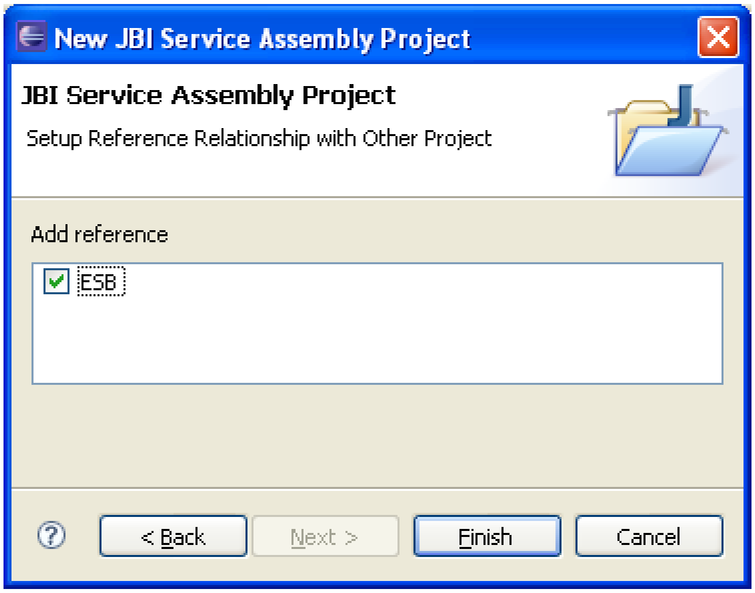
Create New JBI Service Assembly Project

Project name: UseCaseXmlOrder

☑ Use default location

Location: C:/cbesb-1.2/ideworkspace/UseCaseXmlOrder

Click Next after naming the project, then Next past the Build Path screen, then select the ESB project



New JBI Service Assembly Project

JBI Service Assembly Project

Setup Reference Relationship with Other Project

Add reference

☑ ESB

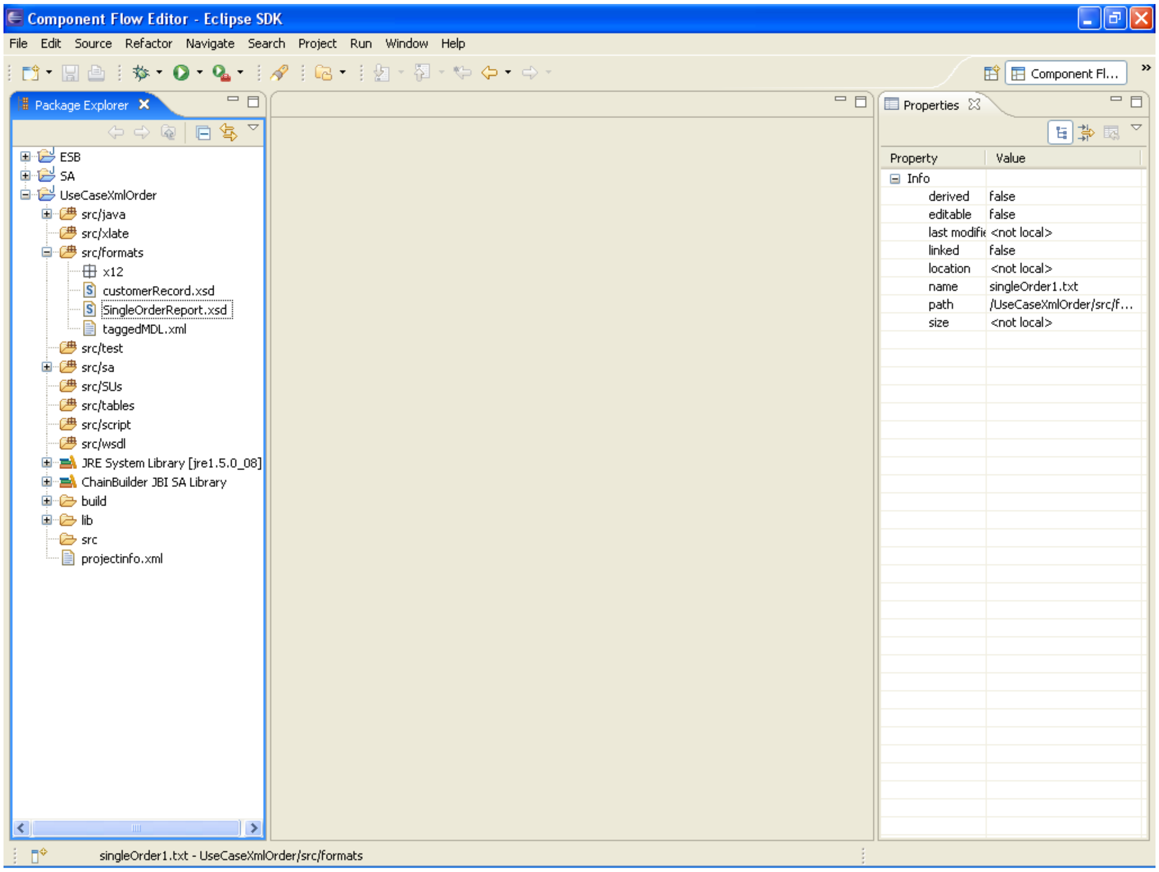
then click Finish.

Step 3. Move Source Files.

Copy in XSD files **customerRecord.xsd** and **SingleOrderReport.xsd** from file cbesb_usecase1.zip, from: http://www.chainforge.net/chainbuilder/usecase.html into the **UseCaseXMLOrder** project in its **<cbesb_home>\ideworkspace\UseCaseXmlOrder\src\formats** folder.

Alternately, you can right-click on **src/formats** and use the Eclipse Import to copy the file.

The resulting screen looks like this:



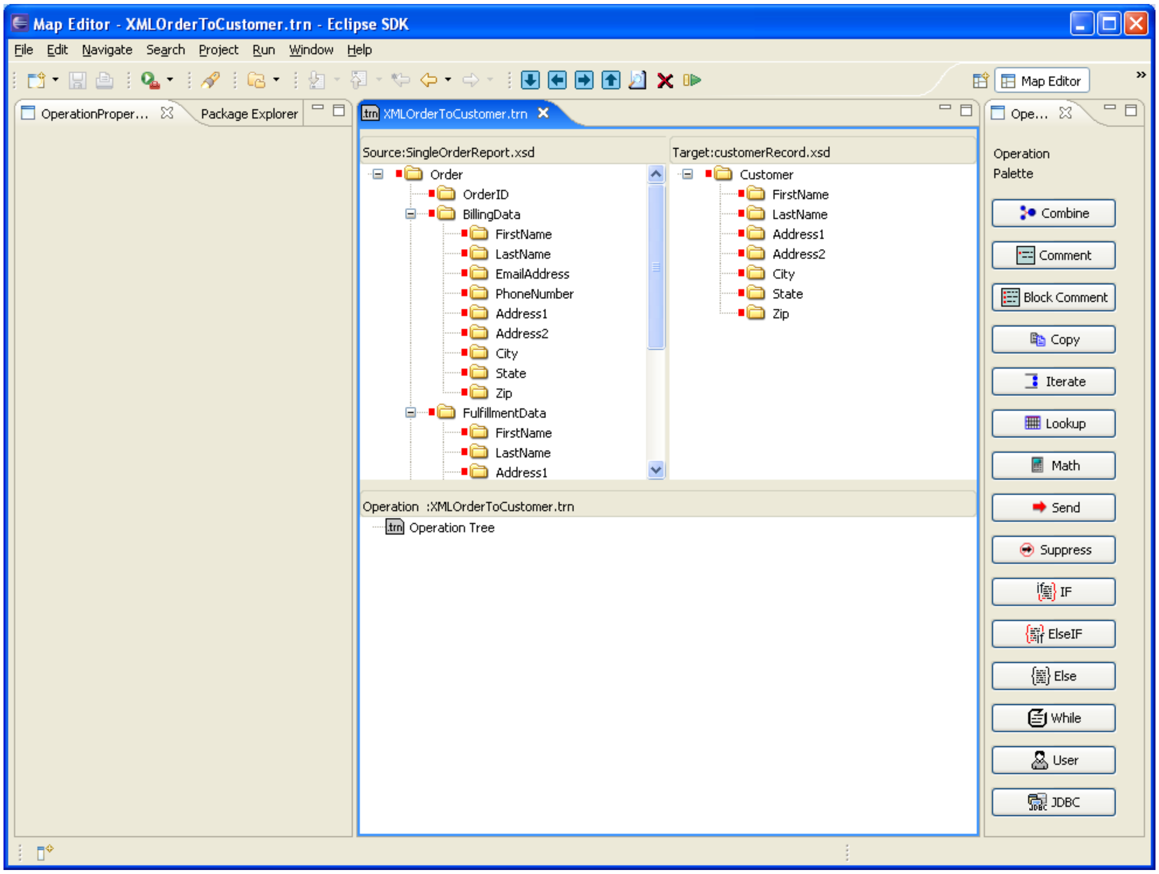
Property	Value
Info	
derived	false
editable	false
last modifie	<not local>
linked	false
location	<not local>
name	singleOrder1.txt
path	/UseCaseXmlOrder/src/f...
size	<not local>

singleOrder1.txt - UseCaseXmlOrder/src/formats

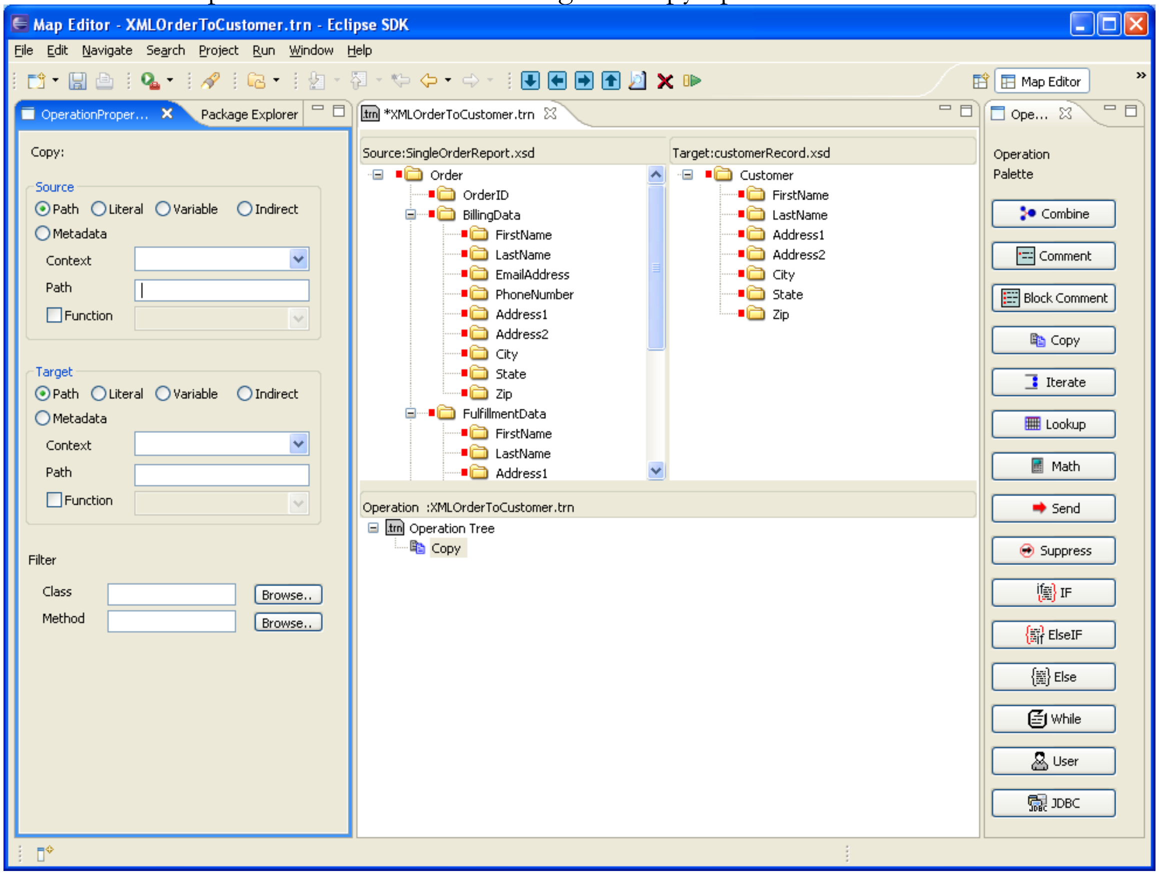
Step 4. Create Transformation File. Create a Tranformation file named "XMLOrderToCustomer" using the Map Editor. Select the source definition as: "UseCaseXMLOrder::src/formats/SingleOrderReport.xsd" and the destination definition as "UseCaseXMLOrder::src/formats/customerRecord.xsd".

This transformation file will map data from the SingleOrderReport (order record) format to the Customer Record format.

The Map Editor starts out blank as depicted below. Note the locations of the Operation Properties tab on the left (you may need to click on this to open it), the Source and Target trees in the upper middle, the operation Tree in the lower middle, and the Operation Palette on the right.



To start the mapping, add a Copy operation to the Operation Tree by clicking the Copy button on the Operation Palette. An unconfigured copy operation is added as shown below:



Source: ● Path ○ Literal ○ Variable ○ Indirect ○ Metadata

□ Function

Target: ● Path ○ Literal ○ Variable ○ Indirect ○ Metadata

□ Function

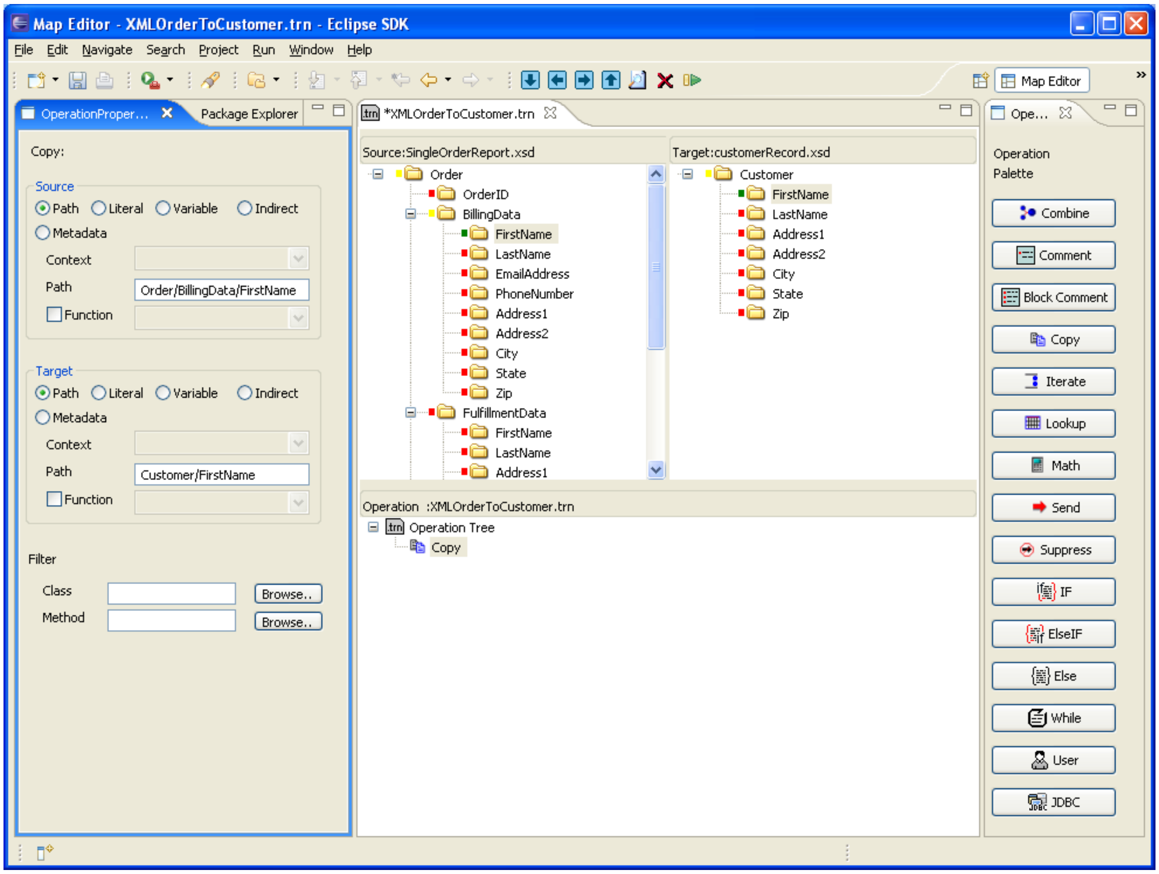
The Copy operation is the most basic mapping instruction; it tells the mapper to reproduce the data found in the selected source field into the selected target field.

Configure the copy operation.

Left-click the FirstName field (under BillingData in the Source Tree) and drag it to the Path box in the Source section of the Operation Properties tab. This configures the source field for the Copy operation.

The Left-click the FirstName field (in the Target Tree) and drag it to the Path box in the Target section of the Operation Properties tab. This step configures the Target field for the Copy operation.

The result is as shown below:



Map Editor - XMLOrderToCustomer.trn - Eclipse SDK

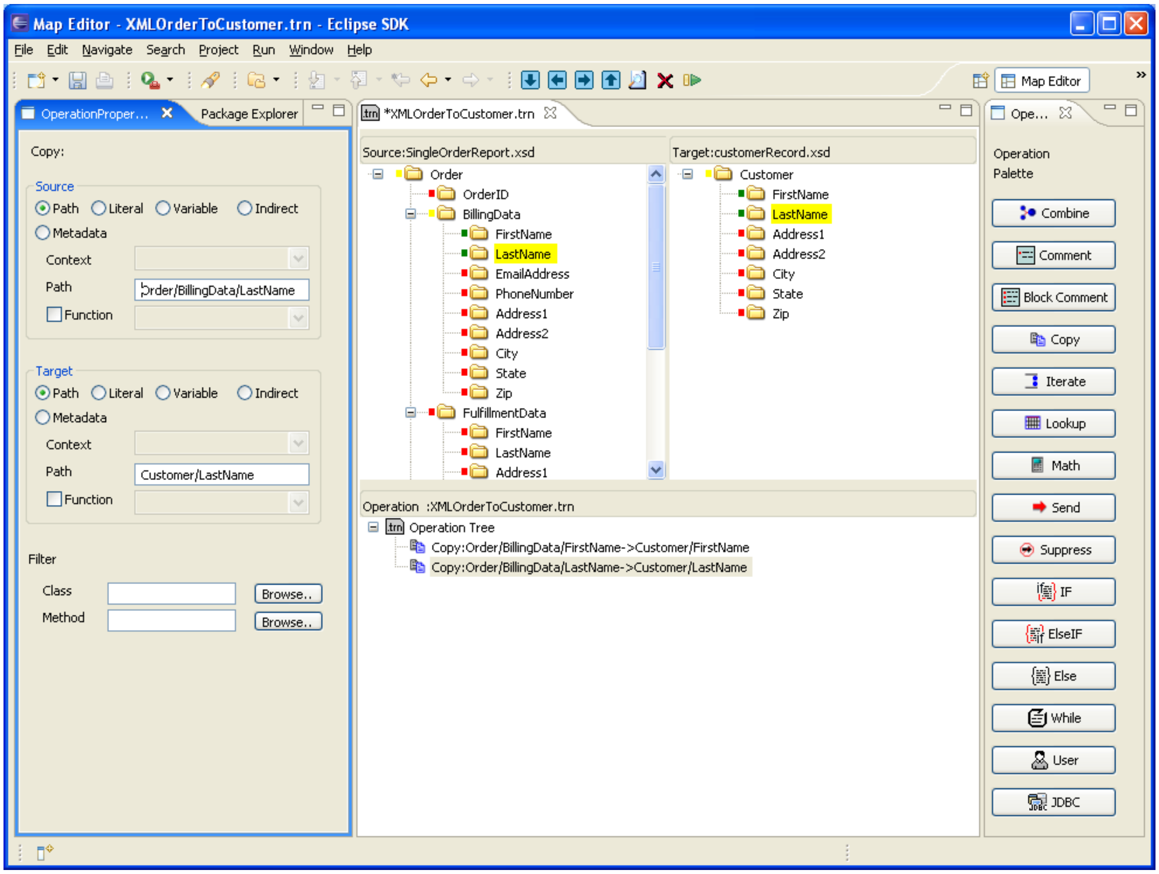
Copy:

Source
● Path ○ Literal ○ Variable ○ Indirect
○ Metadata
Context
Path Order/BillingData/FirstName
□ Function

Target
● Path ○ Literal ○ Variable ○ Indirect
○ Metadata
Context
Path Customer/FirstName
□ Function

Filter
Class
Method

There is an alternate method to create copy statements in ChainBuilder ESB. For the next copy statement, simply Left-click the LastName field (under Billing Data in the Source Tree) and drag it to the LastName field in the Target Tree. A copy statement between those two fields is created.



Copy:

Source

● Path ○ Literal ○ Variable ○ Indirect
○ Metadata

Context

Path: Order/BillingData/LastName

□ Function

Target

● Path ○ Literal ○ Variable ○ Indirect
○ Metadata

Context

Path: Customer/LastName

□ Function

Filter

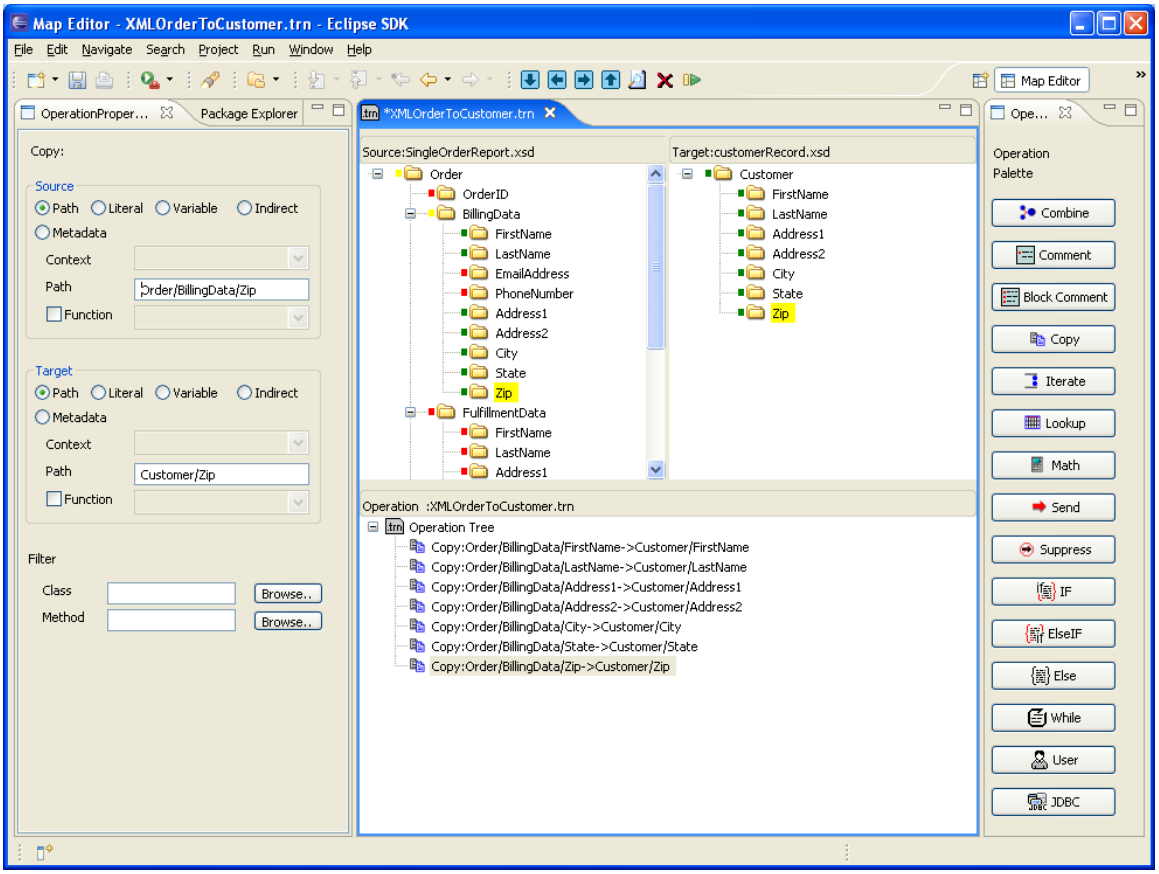
Class Browse..

Method Browse..

Operation :XMLOrderToCustomer.trn

Operation Tree
- Copy:Order/BillingData/FirstName->Customer/FirstName
- Copy:Order/BillingData/LastName->Customer/LastName

Complete the map, by creating copy operations for the Address1, Address2, City, State, and Zip fields. The result should appear as shown below:



Copy:

Source
● Path ○ Literal ○ Variable ○ Indirect
○ Metadata
Context
Path Order/BillingData/Zip
□ Function

Target
● Path ○ Literal ○ Variable ○ Indirect
○ Metadata
Context
Path Customer/Zip
□ Function

Filter
Class Browse..
Method Browse..

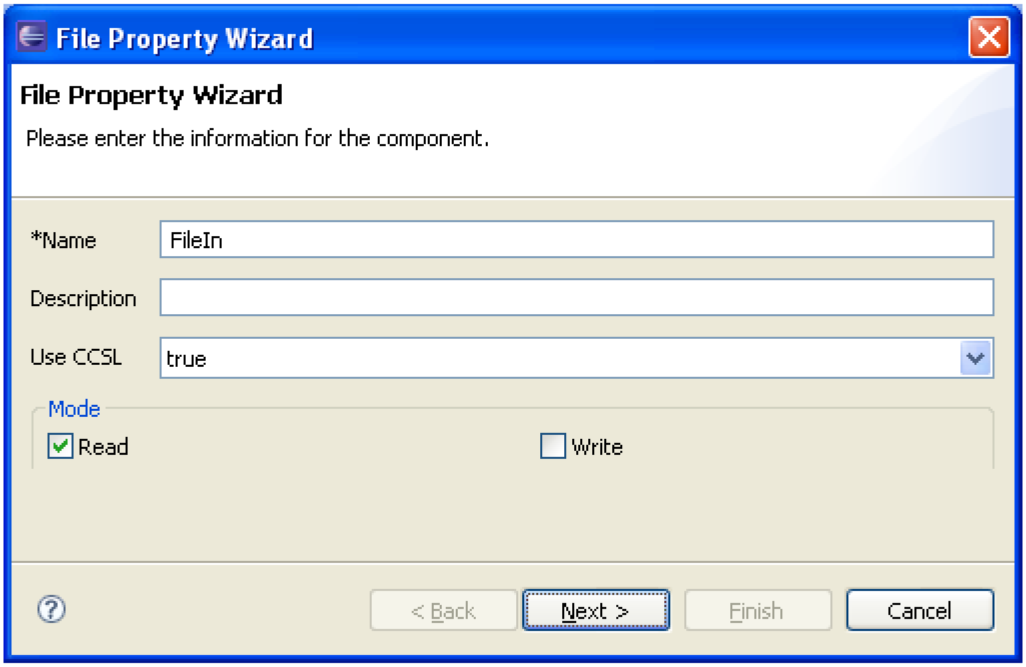
Operation :XMLOrderToCustomer.trn

```
Operation Tree
    Copy:Order/BillingData/FirstName->Customer/FirstName
    Copy:Order/BillingData/LastName->Customer/LastName
    Copy:Order/BillingData/Address1->Customer/Address1
    Copy:Order/BillingData/Address2->Customer/Address2
    Copy:Order/BillingData/City->Customer/City
    Copy:Order/BillingData/State->Customer/State
    Copy:Order/BillingData/Zip->Customer/Zip
```

Save the Map using the disk icon; the map is complete.

Step 5. Create the Flow Definition. Create the the flow definition using the Component Flow Editor.

Left-click on the File Binding component, then left-click, drag and release within the component flow diagram to create a File binding component. This will start the File Property Wizard. Follow the wizard to create the file component, naming it FileIn. FileIn should be configured for read mode, as shown below.



File Property Wizard

File Property Wizard

Please enter the information for the component.

*Name: FileIn

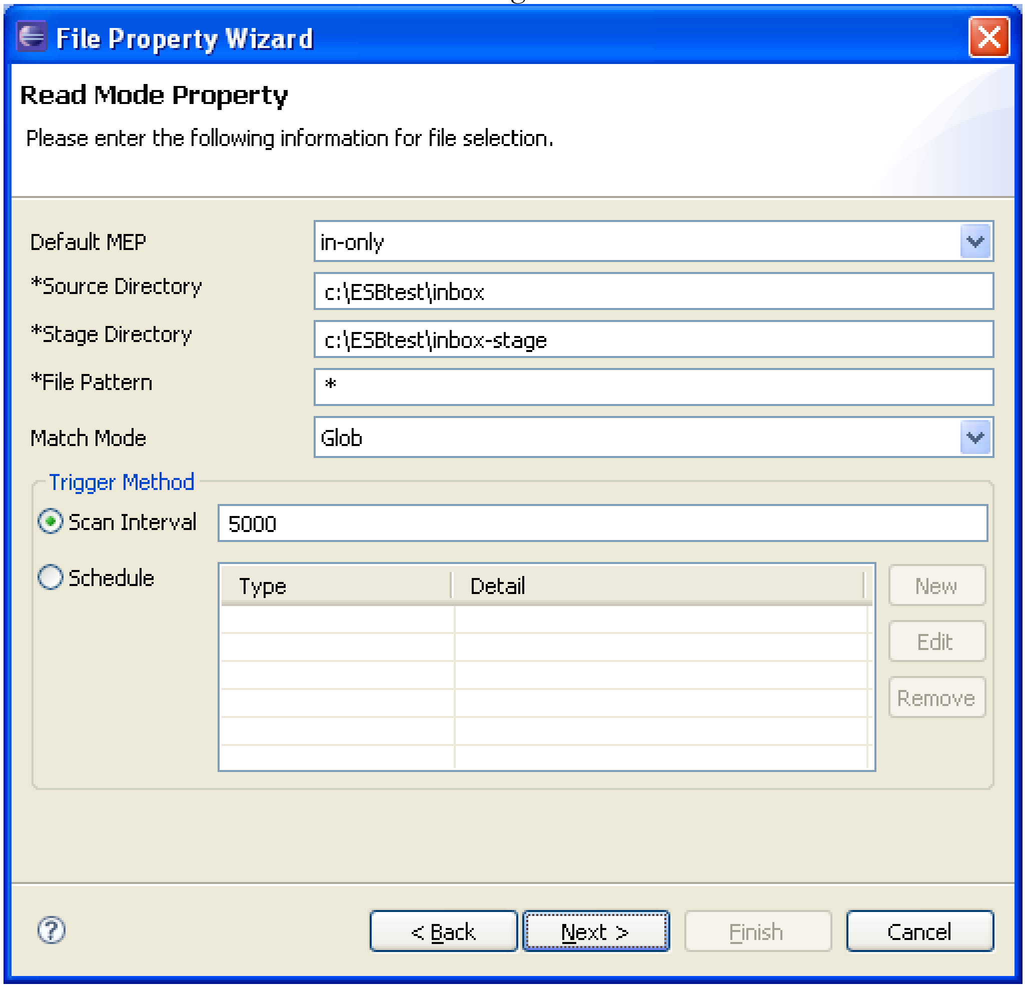
Description:

Use CCSL: true

Mode

☑ Read ☐ Write

Click Next then fill in the Source and Stage directories as shown:



File Property Wizard

Read Mode Property

Please enter the following information for file selection.

Default MEP: in-only

*Source Directory: c:\ESBtest\inbox

*Stage Directory: c:\ESBtest\inbox-stage

*File Pattern: *

Match Mode: Glob

Trigger Method

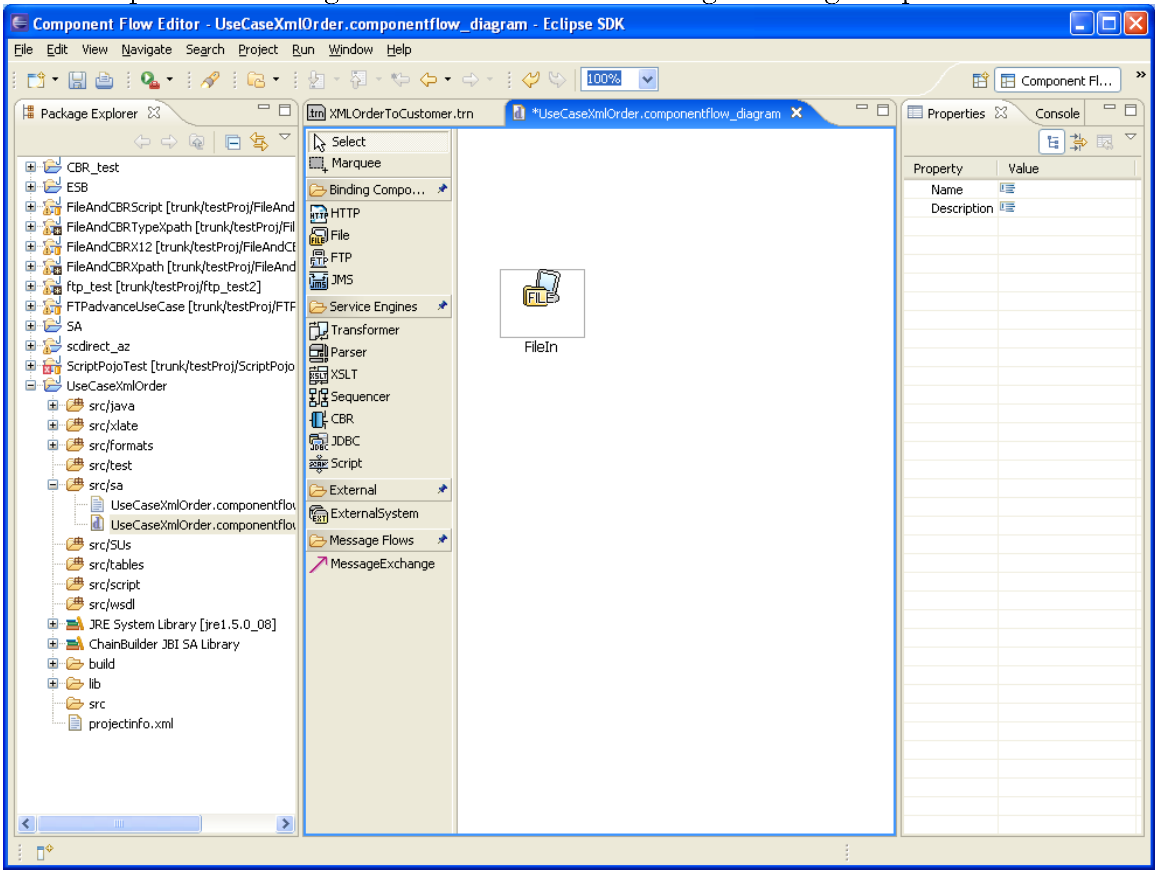
● Scan Interval: 5000

○ Schedule

The File component will search for files in the Source directory and move them to the Stage directory for safe processing. The file will be deleted after it has been read from the Stage directory.

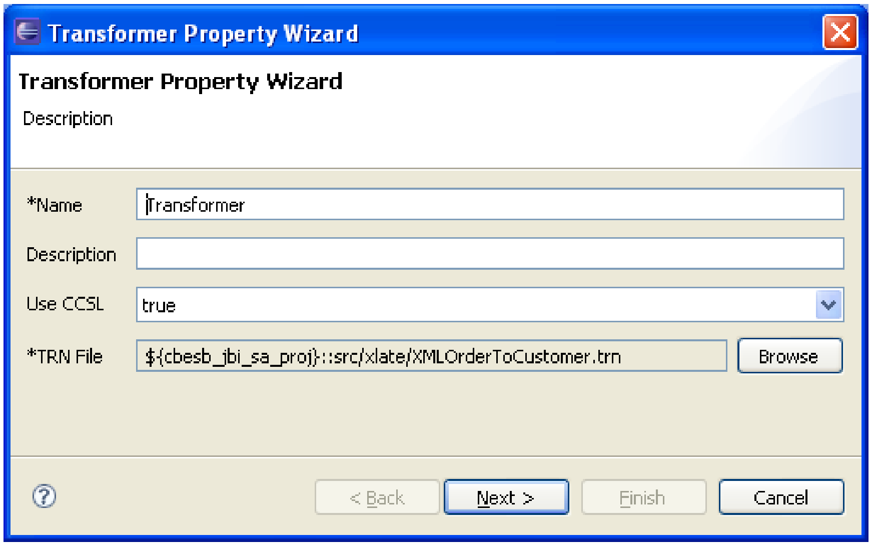
Click Next 3 times then Finish to accept the remaining defaults.

Your component flow diagram should now include a single binding component:


Component Flow Editor - UseCaseXmlOrder.componentflow_diagram - Eclipse SDK
File Edit View Navigate Search Project Run Window Help
100%
Component Fl...
Package Explorer
CBR_test
ESB
FileAndCBRScript [trunk/testProj/FileAnd
FileAndCBRTypeXpath [trunk/testProj/Fil
FileAndCBRX12 [trunk/testProj/FileAndCE
FileAndCBRXpath [trunk/testProj/FileAnd
ftp_test [trunk/testProj/ftp_test2]
FTPadvanceUseCase [trunk/testProj/FTF
SA
scdirect_az
ScriptPojoTest [trunk/testProj/ScriptPojo
UseCaseXmlOrder
src/java
src/xlate
src/formats
src/test
src/sa
UseCaseXmlOrder.componentflo
UseCaseXmlOrder.componentflo
src/SUs
src/tables
src/script
src/wsdl
JRE System Library [jre1.5.0_08]
ChainBuilder JBI SA Library
build
lib
src
projectinfo.xml
XMLOrderToCustomer.trn
*UseCaseXmlOrder.componentflow_diagram
Select
Marquee
Binding Compo...
HTTP
File
FTP
JMS
Service Engines
Transformer
Parser
XSLT
Sequencer
CBR
JDBC
Script
External
ExternalSystem
Message Flows
MessageExchange
FileIn
Properties
Console
Property
Value
Name
Description

Next, create a sequencer below the FileIn component (click Finish to accept the defaults in the wizard).

Under the sequencer (and a little to the left), create a Transformer. Use the Browse button in the Transformer Property Wizard to select the TRN file that you previously created.



Transformer Property Wizard

Description

*Name: Transformer

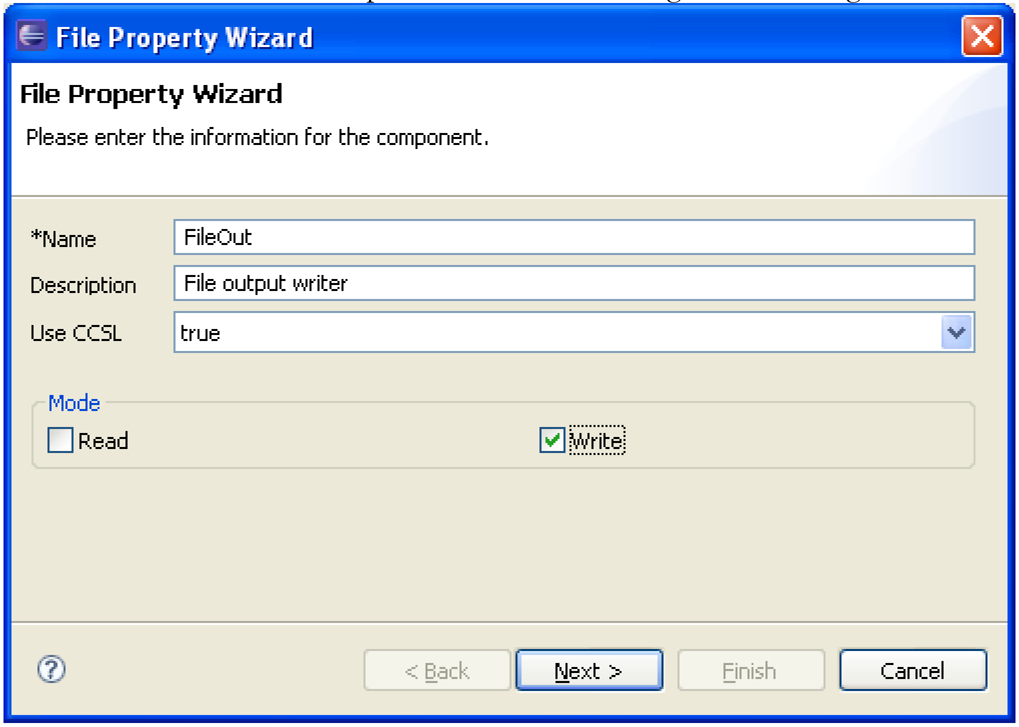
Description:

Use CCSL: true

*TRN File: ${cbesb_jbi_sa_proj}::src/xlate/XMLOrderToCustomer.trn

Click Next then Finish to complete the Transformer.

Now create a second File binding component under the sequencer but to the right of the Transformer. Name this componet FileOut and configure for writing it as shown below:



File Property Wizard

Please enter the information for the component.

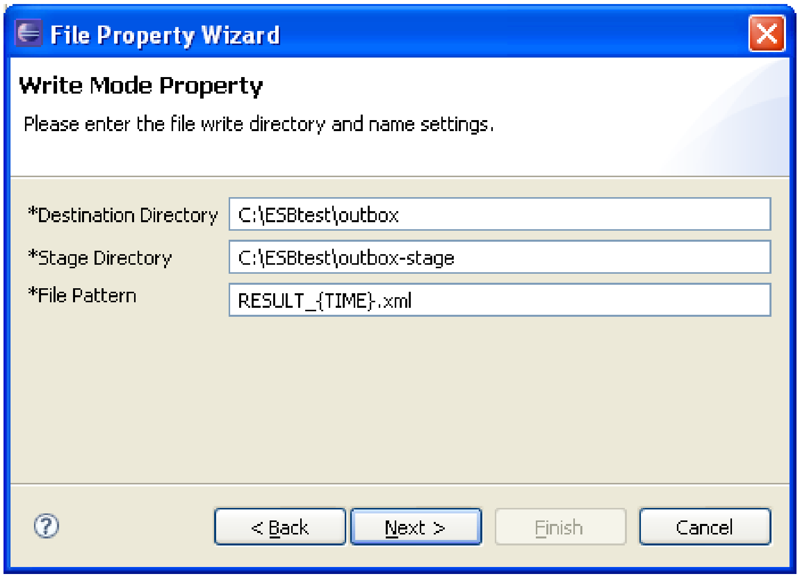
*Name: FileOut

Description: File output writer

Use CCSL: true

Mode

☐ Read ☑ Write



Write Mode Property

Please enter the file write directory and name settings.

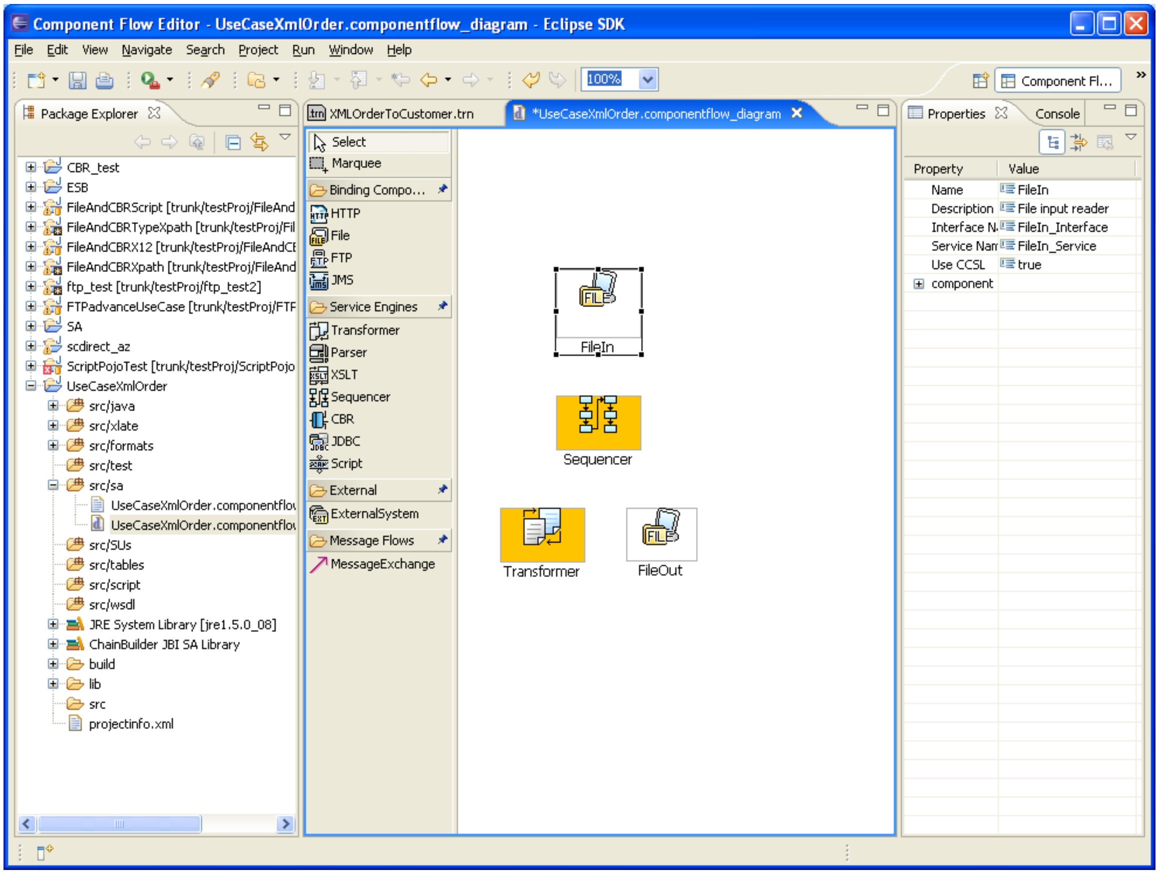
*Destination Directory: C:\ESBtest\outbox

*Stage Directory: C:\ESBtest\outbox-stage

*File Pattern: RESULT_{TIME}.xml

The file contents will be written to the Stage directory and moved to the Destination directory once the file is complete. Any other systems wishing to read the output file should look in the Destination directory.

Click Next then Finish. Your Component Flow should now look like this:


Component Flow Editor - UseCaseXmlOrder.componentflow_diagram - Eclipse SDK
File Edit View Navigate Search Project Run Window Help
100%
Component Fl...
Package Explorer
CBR_test
ESB
FileAndCBRScript [trunk/testProj/FileAnd
FileAndCBRTypeXpath [trunk/testProj/Fil
FileAndCBRX12 [trunk/testProj/FileAndCI
FileAndCBRXpath [trunk/testProj/FileAnd
ftp_test [trunk/testProj/ftp_test2]
FTPadvanceUseCase [trunk/testProj/FTF
SA
scdirect_az
ScriptPojoTest [trunk/testProj/ScriptPojo
UseCaseXmlOrder
src/java
src/xlate
src/formats
src/test
src/sa
UseCaseXmlOrder.componentflo
UseCaseXmlOrder.componentflo
src/SUs
src/tables
src/script
src/wsdl
JRE System Library [jre1.5.0_08]
ChainBuilder JBI SA Library
build
lib
src
projectinfo.xml
XMLOrderToCustomer.trn
*UseCaseXmlOrder.componentflow_diagram
Select
Marquee
Binding Compo...
HTTP
File
FTP
JMS
Service Engines
Transformer
Parser
XSLT
Sequencer
CBR
JDBC
Script
External
ExternalSystem
Message Flows
MessageExchange
FileIn
Sequencer
Transformer
FileOut
Properties
Console

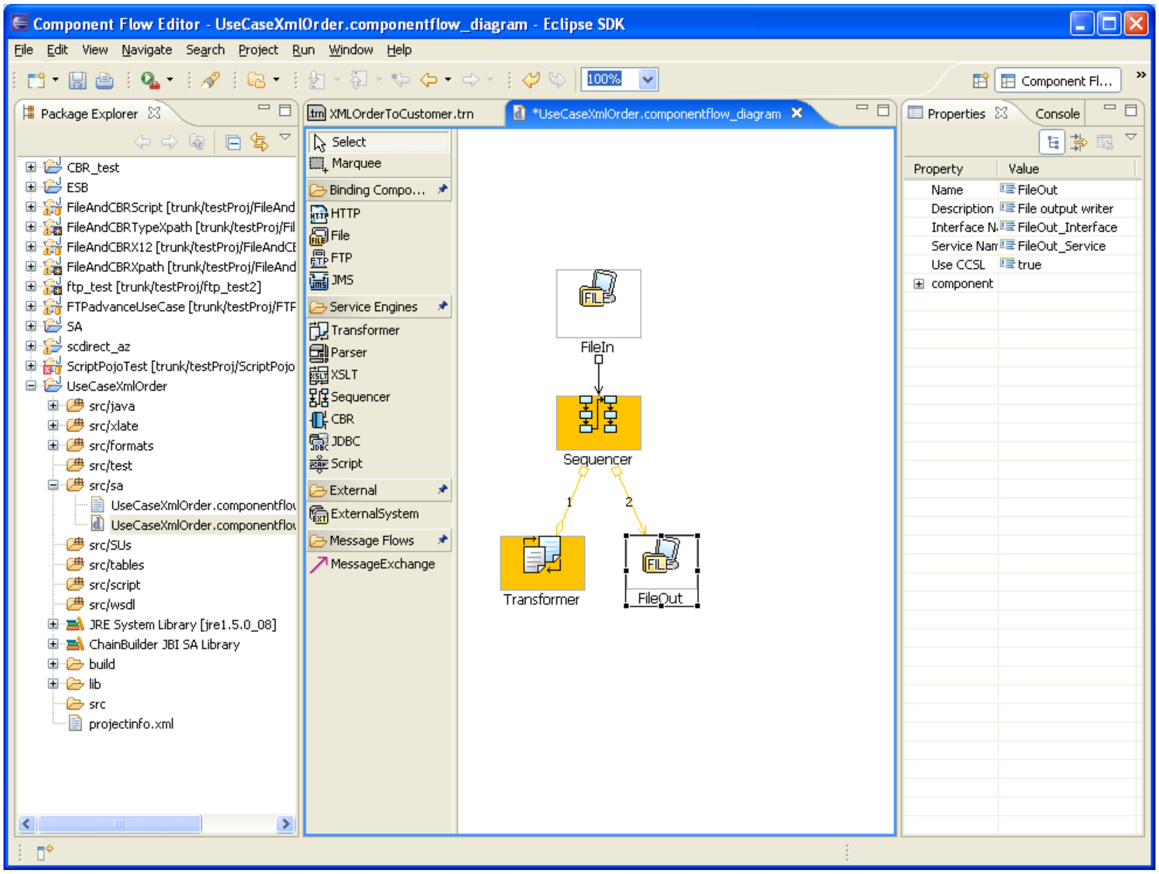
Property	Value
Name	FileIn
Description	File input reader
Interface N.	FileIn_Interface
Service Nam	FileIn_Service
Use CCSL	true
component	

Now create the Message Flows (arrows) between the various objects. Click on MessageExchange, then click on FileIn then on Sequencer, to create a black line. The black line indicates a basic data route.

Click on MessageExchange, then on Sequencer, then on Transformer. This will create an orange line numbered "1" indicating that this is the first event to be performed by the sequencer.

Click on MessageExchange, then on Sequencer, then on FileOut. This will create a second orange line labeled "2" which goes from Sequencer to FileOut. This will be the second event performed by the sequencer.

The end result should appear as shown below:


Component Flow Editor - UseCaseXmlOrder.componentflow_diagram - Eclipse SDK
File Edit View Navigate Search Project Run Window Help
100%
Component Fl...
Package Explorer
CBR_test
ESB
FileAndCBRScript [trunk/testProj/FileAnd
FileAndCBRTypeXpath [trunk/testProj/Fil
FileAndCBRX12 [trunk/testProj/FileAndCI
FileAndCBRXpath [trunk/testProj/FileAnd
ftp_test [trunk/testProj/ftp_test2]
FTPadvanceUseCase [trunk/testProj/FTF
SA
scdirect_az
ScriptPojoTest [trunk/testProj/ScriptPojo
UseCaseXmlOrder
src/java
src/xlate
src/formats
src/test
src/sa
UseCaseXmlOrder.componentflo
UseCaseXmlOrder.componentflo
src/SUs
src/tables
src/script
src/wsdl
JRE System Library [jre1.5.0_08]
ChainBuilder JBI SA Library
build
lib
src
projectinfo.xml
XMLOrderToCustomer.trn
*UseCaseXmlOrder.componentflow_diagram
Select
Marquee
Binding Compo...
HTTP
File
FTP
JMS
Service Engines
Transformer
Parser
XSLT
Sequencer
CBR
JDBC
Script
External
ExternalSystem
Message Flows
MessageExchange
FileIn
Sequencer
1
2
Transformer
FileOut
Properties
Console

Property	Value
Name	FileOut
Description	File output writer
Interface N	FileOut_Interface
Service Nan	FileOut_Service
Use CCSL	true
⊞ component	

The flow starts when the FileIn component finds a file to read. The contents of the file will be sent to the sequencer (as a message in a message exchange) then first to the Transformer. The result of the transformation will be returned to the sequencer and sent on to the FileOut component to be written out as the final result of this example SA.

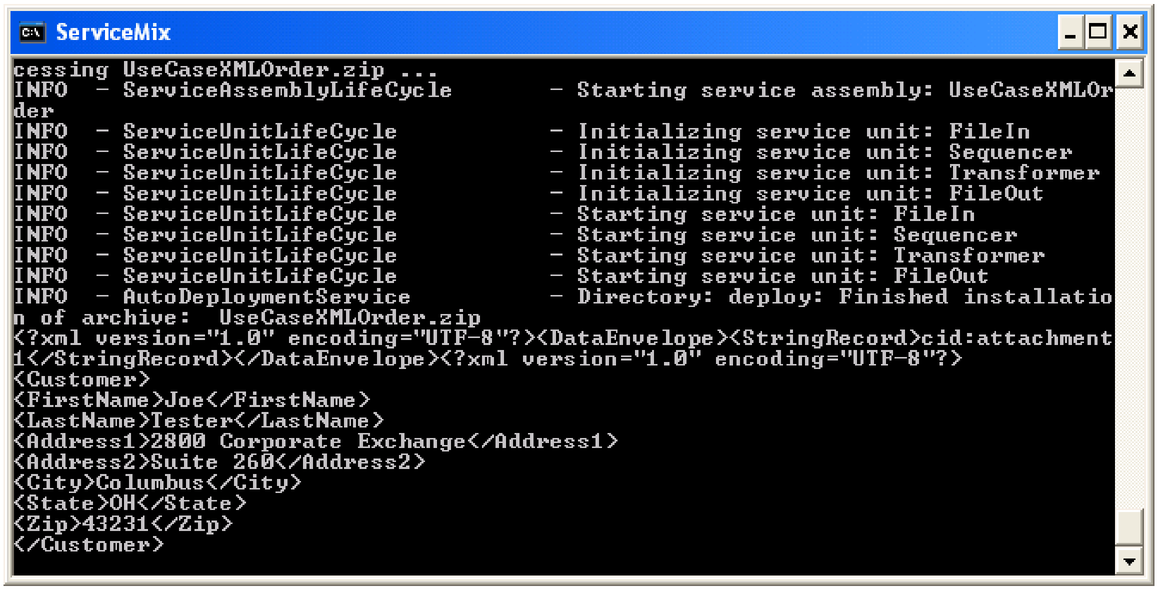
Click the disk icon to save your component flow. Before you move the next step, right-click on the component flow editor canvas and select “Deploy” from the drop-down menu, and wait for the confirmation window. The “Deploy” action will create the relevant JBI actificats and WSDL files for ChainBuilder ESB server runtime to use.

Step 6. Run the Project.

Copy the **singleOrder1.txt** from the downloadable zip file into the inbox directory configured in the file component in the Component Flow Editor.

Start the server to run UseCaseXMLOrder project using **cbesb_run UseCaseXMLOrder.** Note: this command should be executed from the drive on which ChainBuilder ESB was installed.

The following shows the result screen:



```
cessing UseCaseXMLOrder.zip ...
INFO  - ServiceAssemblyLifeCycle       - Starting service assembly: UseCaseXMLOr
der
INFO  - ServiceUnitLifeCycle           - Initializing service unit: FileIn
INFO  - ServiceUnitLifeCycle           - Initializing service unit: Sequencer
INFO  - ServiceUnitLifeCycle           - Initializing service unit: Transformer
INFO  - ServiceUnitLifeCycle           - Initializing service unit: FileOut
INFO  - ServiceUnitLifeCycle           - Starting service unit: FileIn
INFO  - ServiceUnitLifeCycle           - Starting service unit: Sequencer
INFO  - ServiceUnitLifeCycle           - Starting service unit: Transformer
INFO  - ServiceUnitLifeCycle           - Starting service unit: FileOut
INFO  - AutoDeploymentService          - Directory: deploy: Finished installatio
n of archive:  UseCaseXMLOrder.zip
<?xml version="1.0" encoding="UTF-8"?><DataEnvelope><StringRecord>cid:attachment
1</StringRecord></DataEnvelope><?xml version="1.0" encoding="UTF-8"?>
<Customer>
<FirstName>Joe</FirstName>
<LastName>Tester</LastName>
<Address1>2800 Corporate Exchange</Address1>
<Address2>Suite 260</Address2>
<City>Columbus</City>
<State>OH</State>
<Zip>43231</Zip>
</Customer>
```

After the input file is picked up from the inbox directory, the result file is written to the outbox directory with a timestamped file name.

5. ChainBuilder ESB Community

ChainForge.net is the internet's premier destination to share ChainBuilder and JBI knowledge with your peers.

Join the ChainBuilder ESB Community:
http://www.chainforge.net/community

As a member you can view content and contribute to a Forum:
http://www.chainforge.net/community/forums.html

Read ChainBuilder ESB related Blogs:
http://www.chainforge.net/blogs