Working with EDI Messages

In this section:
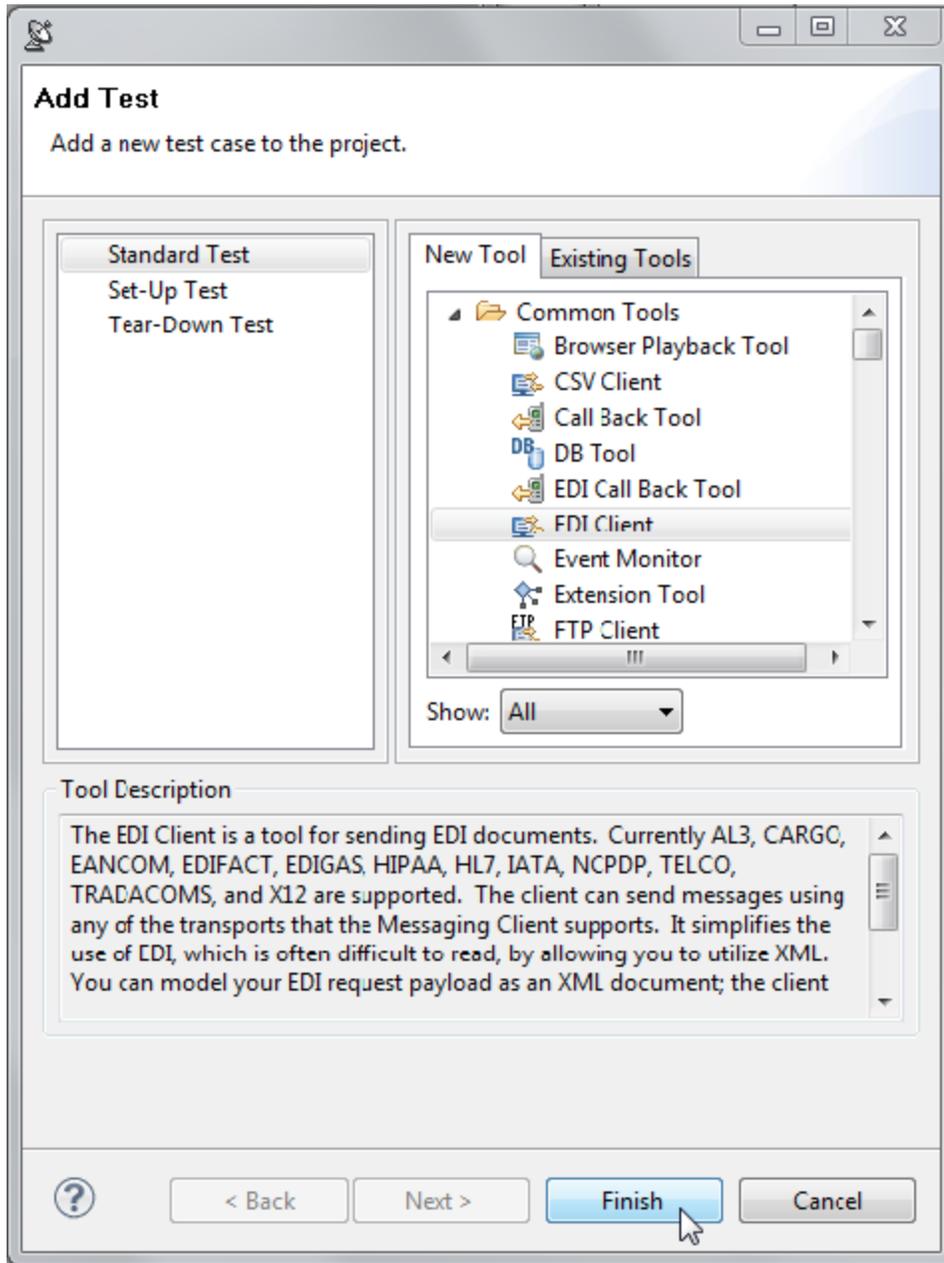
- Creating a New EDI Client
- Setting EDI Conversion Options
- Configuring EDI Messages
- Validating EDI Content

The EDI Client is a tool for sending EDI documents. Currently AL3, CARGO, EANCOM, EDIFACT, EDIGAS, HIPAA, HL7, IATA, NCPDP, TELCO, TRADACOMS, and X12 are supported. It simplifies the use of EDI, which is often difficult to read, by allowing you to utilize XML. You can model your EDI request payload as an XML document; the client then automatically converts the XML to EDI before sending the message. If the server returns an EDI message, the client can convert the message to XML so that you can attach tools such as the XML Assertor and XML Data Bank to the response.
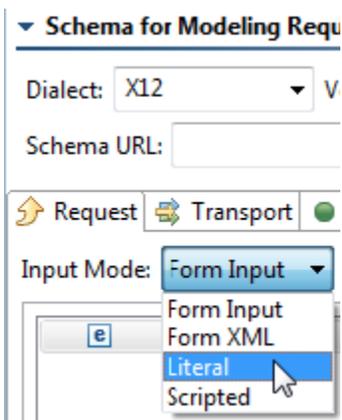
Creating a New EDI Client

To create a new EDI Client:

1. Create a new empty .tst file as follows:
   a. Choose **File> New> Project**.
   b. Select **SOAtest> Empty Project**, then click **Next**.
   c. Enter a project name, then click **Finish**.
2. Create a new EDI Client as follows:
   a. Right-click the new test suite node, then choose **Add New> Test**.
   b. Select **EDI Client**, then click **Finish**.

Add Test

Add a new test case to the project.

| Standard Test | New Tool | Existing Tools |

Standard Test
Set-Up Test
Tear-Down Test

New Tool | Existing Tools

◢ 📂 Common Tools
　　🖳 Browser Playback Tool
　　📇 CSV Client
　　📇 Call Back Tool
　　ᴰᴮ DB Tool
　　📇 EDI Call Back Tool
　　📇 EDI Client
　　🔍 Event Monitor
　　✨ Extension Tool
　　ꜰᵀᴾ FTP Client

Show: All

Tool Description

The EDI Client is a tool for sending EDI documents. Currently AL3, CARGO, EANCOM, EDIFACT, EDIGAS, HIPAA, HL7, IATA, NCPDP, TELCO, TRADACOMS, and X12 are supported. The client can send messages using any of the transports that the Messaging Client supports. It simplifies the use of EDI, which is often difficult to read, by allowing you to utilize XML. You can model your EDI request payload as an XML document; the client

(?)　　< Back　　Next >　　**Finish**　　Cancel

3. Go to the new tool's **Request** tab and change the Input Mode to **Literal**.

▼ **Schema for Modeling Requ**

Dialect: X12　　　　▼　V

Schema URL:

🔁 Request | 📇 Transport | ●

Input Mode: Form Input　▼
　　　　　Form Input
　ⓔ　Form XML
　　　　Literal
　　　　Scripted

4. Paste the following text:

```
UNA:+.? ~
UNB+IATB:1+6XPPC+LHPPC+940101:1950+1~
UNH+1+PAORES:93:1:IA~
MSG+1:45~
IFT+3+XYZCOMPANY AVAILABILITY~
ERC+A7V:1:AMD~
IFT+3+NO MORE FLIGHTS~
ODI~
TVL+240493:1000::1220+FRA+JFK+DL+400+C~
PDI++C:3+Y::3+F::1~
APD+74C:0:::6++++++6X~
TVL+240493:1740::2030+JFK+MIA+DL+081+C~
PDI++C:4~
APD+EM2:0:1630::6+++++++DA~
UNT+13+1~
UNZ+1+1~
```

5. Change the Input Mode from **Literal** to **Form Input**, then confirm that you want to override the values. Note that **Dialect**, **Version**, and **Message t ype** were automatically completed.

6. Go to the **Transport** tab, enter **http://localhost:8080/parabank/echo.jsp** into the **Router Endpoint** field, and save the editor.

Run the EDI Client.
7. Open the Traffic Viewer.

Note that the request was sent in the EDI format. However, also note that each line ends with a quote character instead of the "~" character that was in our original message. By default, the EDI Client is using a different segment separator character than the one that we want to send in our message.
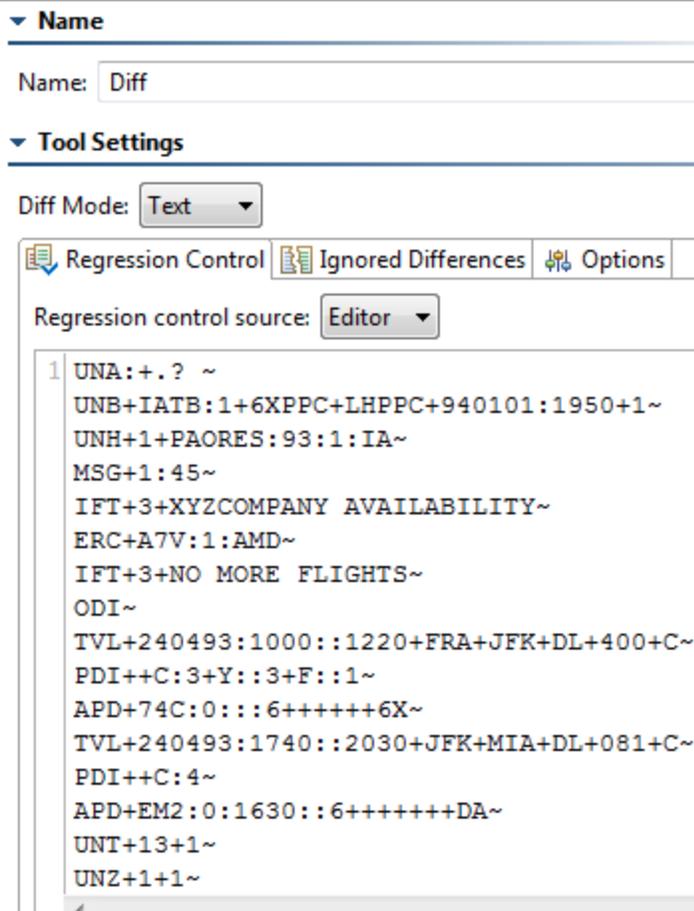
Setting EDI Conversion Options

Now, let's ensure that we're sending the EDI content with the correct segment separator (the character that is at the end of the line).

1. Add a Diff output as follows:
   a. Right-click the EDI Client, then choose **Add Output**.
   b. Select the **Request> Payload** and **Diff**, then click **Finish**.
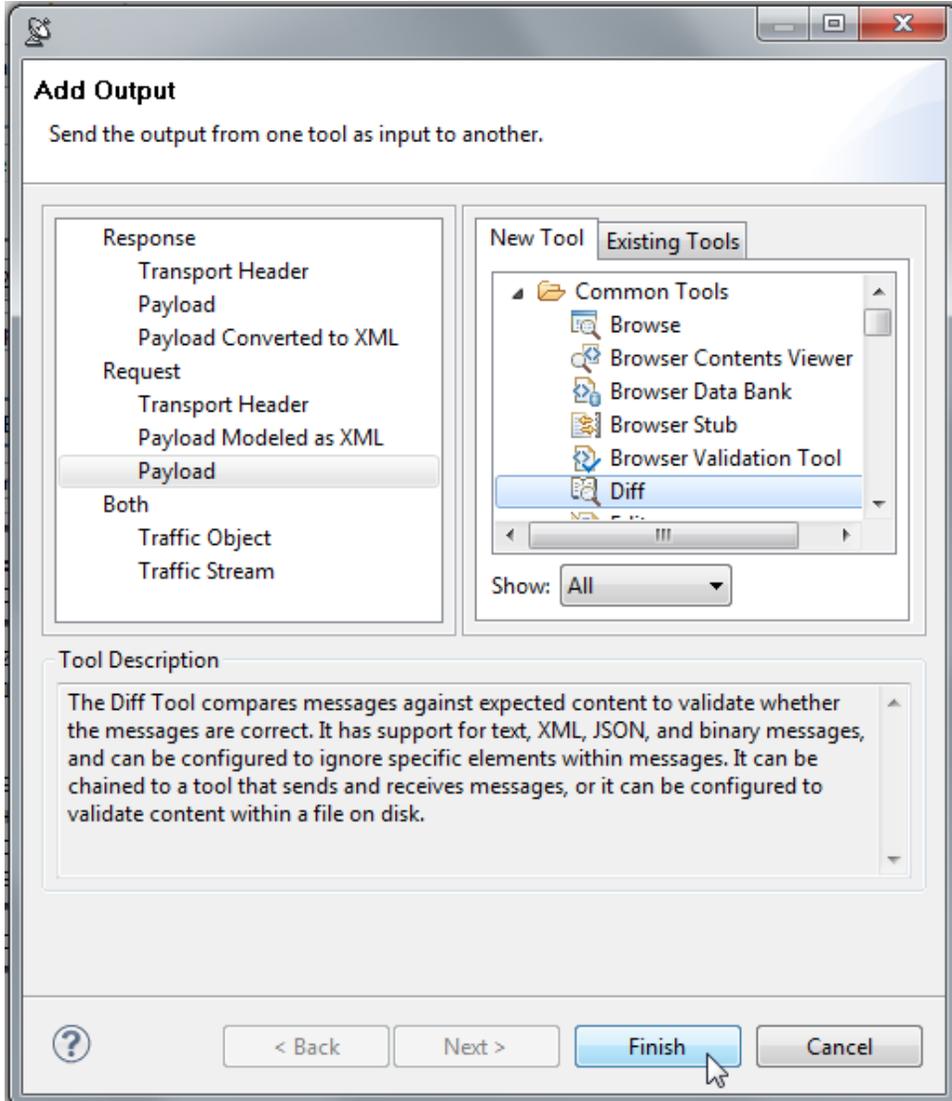
2. Paste the same text (shown above) into the Diff Tool.

3. Open the **Options** tab and enable **Ignore Whitespace**. When the above text is copied and pasted into the Diff tool, new line characters are inserted. Enabling this option will prevent those characters from being reported as differences.
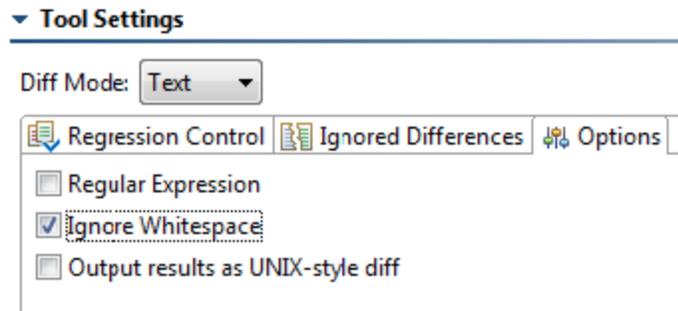
4. Save the Diff tool.
5. Rerun the EDI Client. The test should fail with an error that 16 lines were changed. This is expected since the EDI Client is using a different segment separator than the original message.
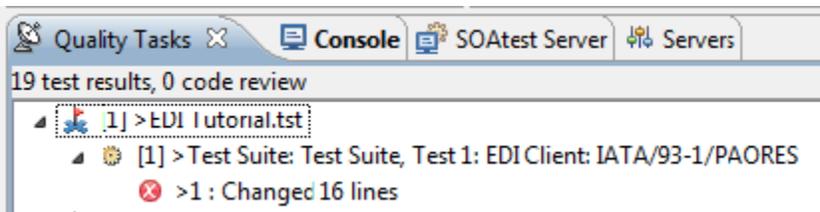
6. Change the EDI Client's conversion options as follows:
   a. Open the tool's **Conversion Options** tab.
   b. Find the **Segment separator** option, and type in ~.

Schema for Modeling Request Payload Using Form Input

Dialect: IATA   Version: 93-1   Message type: PAORES: Pro

Schema URL: file:///C:/Users/cynthia/AppData/Local/Temp/edi-901259999

| Request | Transport | Success Criteria | Conversion Options |

Component value separator:

Line continuation character:

Decimal character:

Element separator:

Segment name/content separator (TRADACOMS):

Invalid character replacement:

Release (escape) symbol:

Repeat symbol:

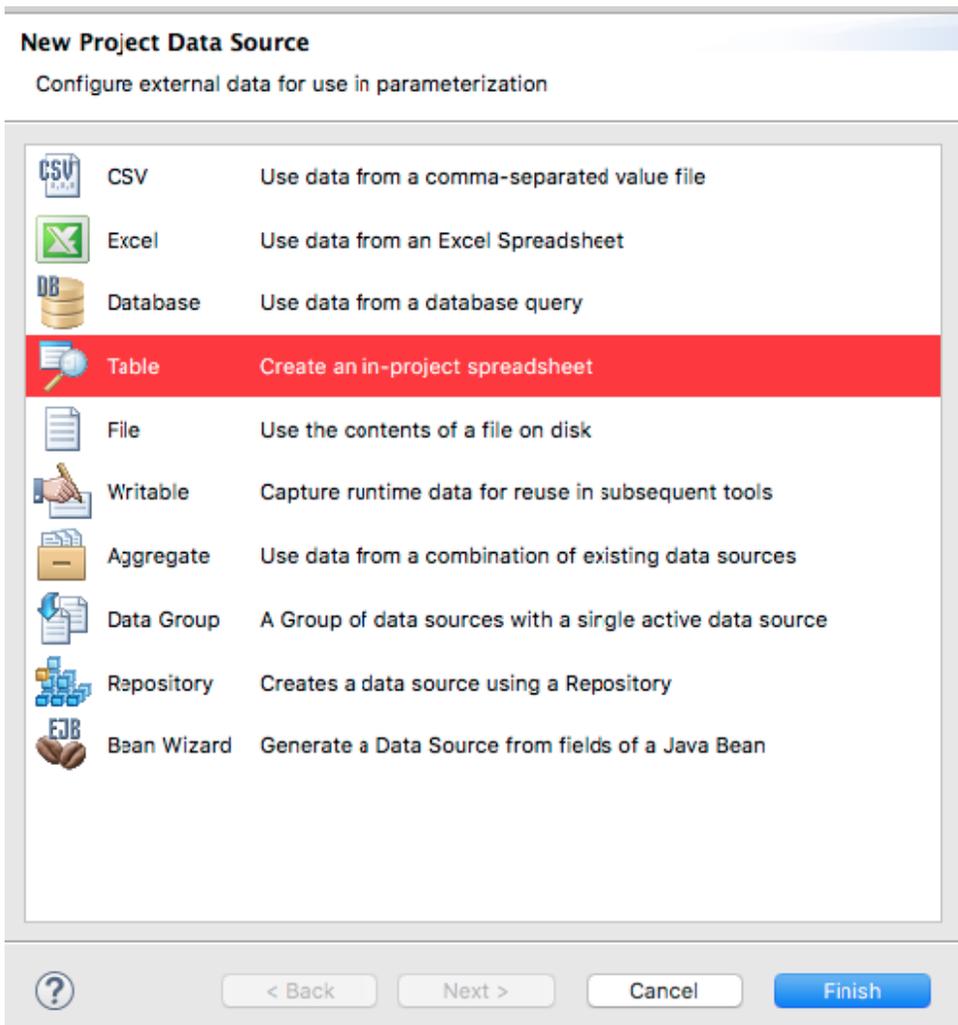Segment separator: ~

      c. Save the EDI Client.
   7. Re-run the EDI Client. Now it will succeed.

Note that in locales where the decimal separator character is a comma (',') rather than a period ('.'), you will see a single test failure on the first line. This is due to the difference in the decimal separator character.

Configuring EDI Messages

One of the main reasons to use the EDI Client is its ability to let you easily configure EDI messages. Let's parameterize one of the fields in our EDI message.

   1. Create a new Table Data Source as follows:
      a. Right click the test suite node, then choose **Add New> Data Source**.
      b. Select **Table**, then click **Finish**.

New Project Data Source

Configure external data for use in parameterization

CSV	CSV	Use data from a comma-separated value file
Excel	Excel	Use data from an Excel Spreadsheet
DB	Database	Use data from a database query
Table	Table	Create an in-project spreadsheet
File	File	Use the contents of a file on disk
Writable	Writable	Capture runtime data for reuse in subsequent tools
Aggregate	Aggregate	Use data from a combination of existing data sources
Data Group	Data Group	A Group of data sources with a single active data source
Repository	Repository	Creates a data source using a Repository
EJB Bean Wizard	Bean Wizard	Generate a Data Source from fields of a Java Bean

[< Back] [Next >] [Cancel] [Finish]

   c. Name the data source `Companies`.
   d. Enable **First row specifies column names**.
   e. In the first column, enter `Company Availability` as the column name, then enter `XYZCOMPANY AVAILABILITY` and `PARASOFT AV` `AILABILITY` in rows 1 and 2, respectively.

▼ Table

☑ First row specifies column names

	A
	Company Availability
1	XYZCOMPANY AVAILABILITY
2	PARASOFT AVAILABILITY
3	
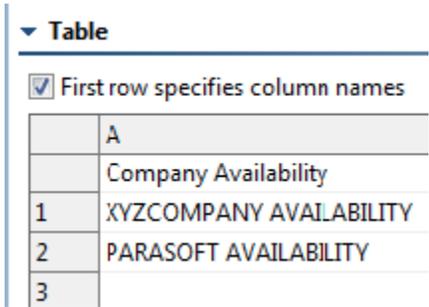
   f. Save the data source.
2. Copy the first EDI Client you created (right-click it, then choose **Copy**) and paste it into the same test suite.
**3.** In the new EDI Client, delete the Diff tool that is attached to the Request Payload.
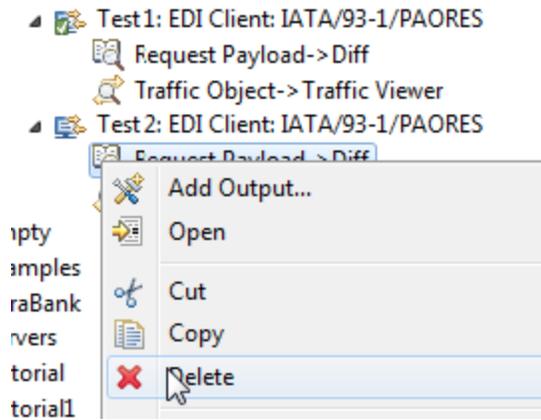
4. In the new EDI Client's **Request** tab of the new EDI Client, ensure that Input Mode is set to **Form Input**.
5. In the tree shown in Form Input, select the first **IFT02-FreeText** node (the full path to the node is "IATA/PAORES/IFT-InteractiveFreeText/IFT02-FreeText").
6. On the right, change **Fixed** to **Parameterized**.
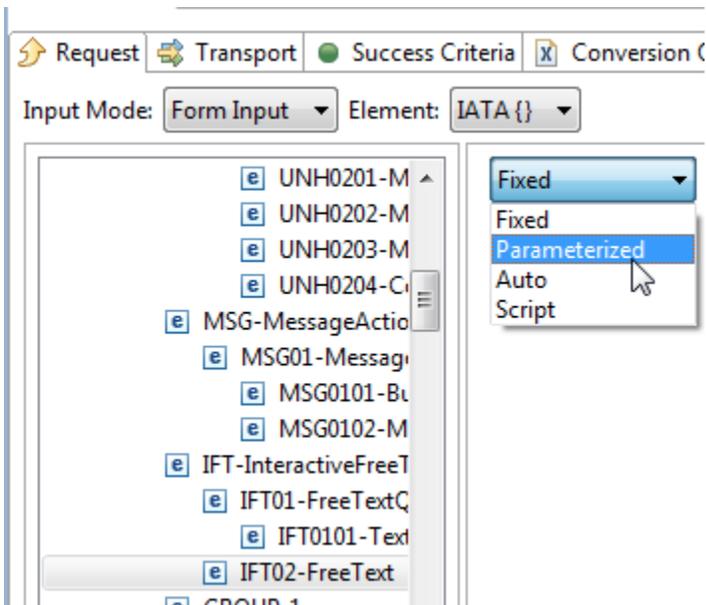
The combo box to the right should show the data source column named **Company Availability**.

7. Save the EDI Client.
8. Run the test.
9. Open the Traffic Viewer for **Test 2: EDI Client** and notice that the test was run twice: once for each row in the data source.
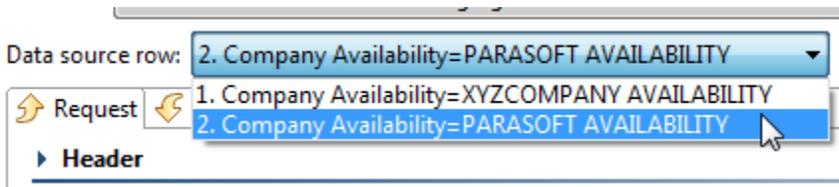

Validating EDI Content

We have configured our EDI Client to send the request to a service that simply echoes back the exact same request that was sent. Thus, the EDI Client is getting the same EDI content in the response as it sent in the request. Let's set up a validation on this EDI content.

1. Add a new XML Assertor Output as follows:

a. Right-click the parameterized EDI Client (the second one), then choose **Add Output**.
b. Select the **Response Payload Converted to XML** and **XML Assertor**, then click **Finish**.
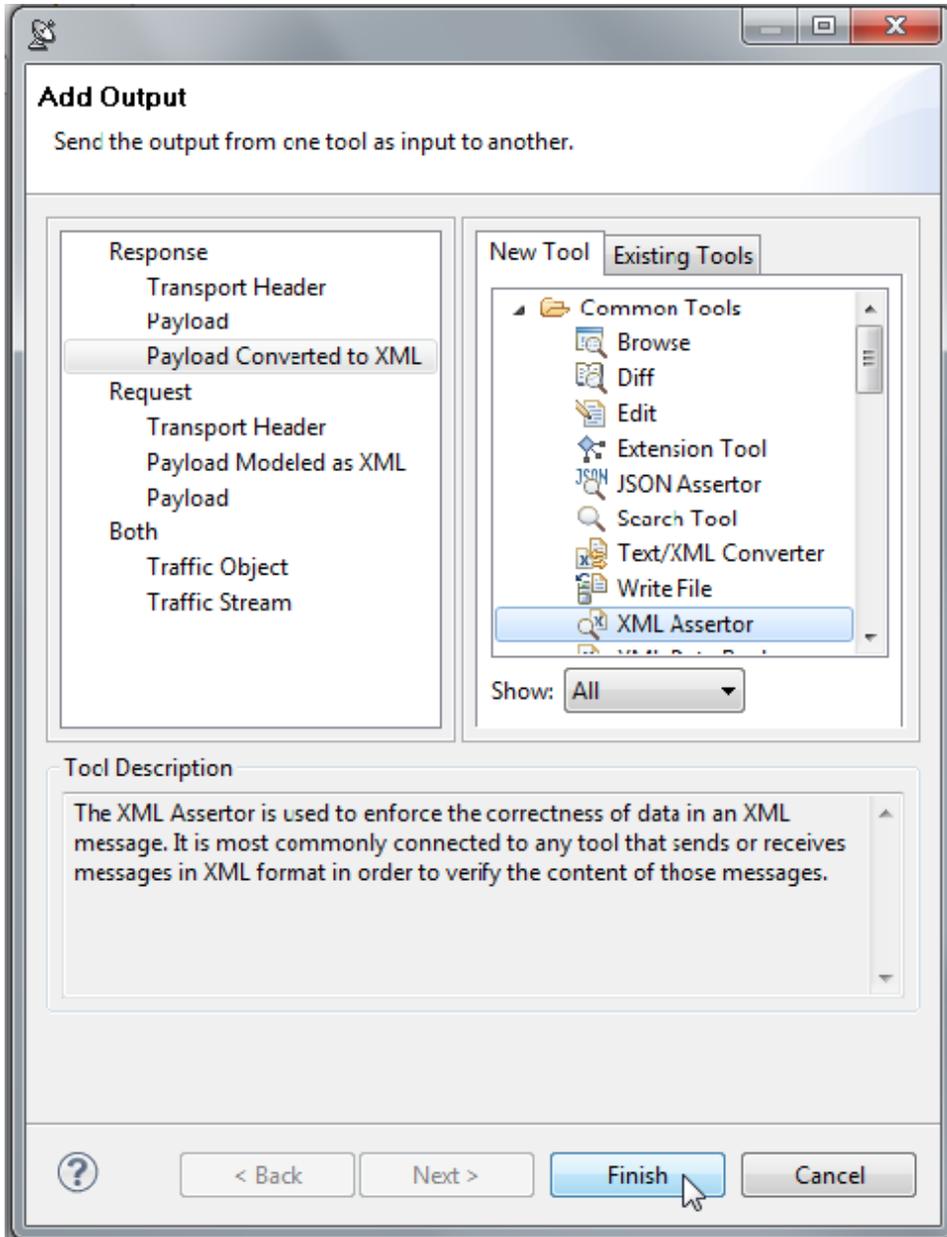
2. Rerun the test to populate the contents of the XML Assertor.
3. Configure the XML Assertor as follows:
   a. Go to the XML Assertor's **Configuration** tab.
   b. Click the **Add** button, select **Value Assertions** and click **Next**.
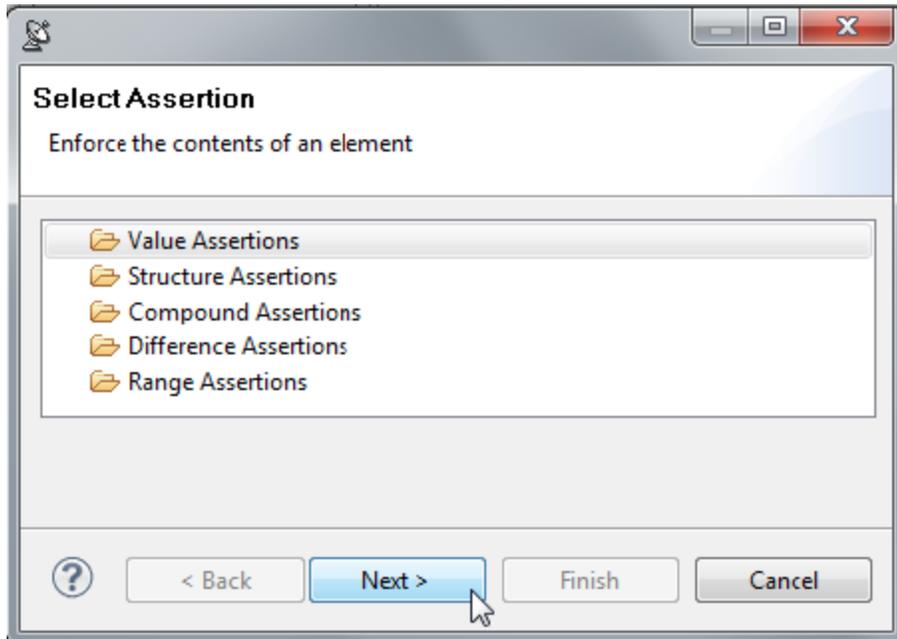
c. Select **Value Assertion** and click **Next**.
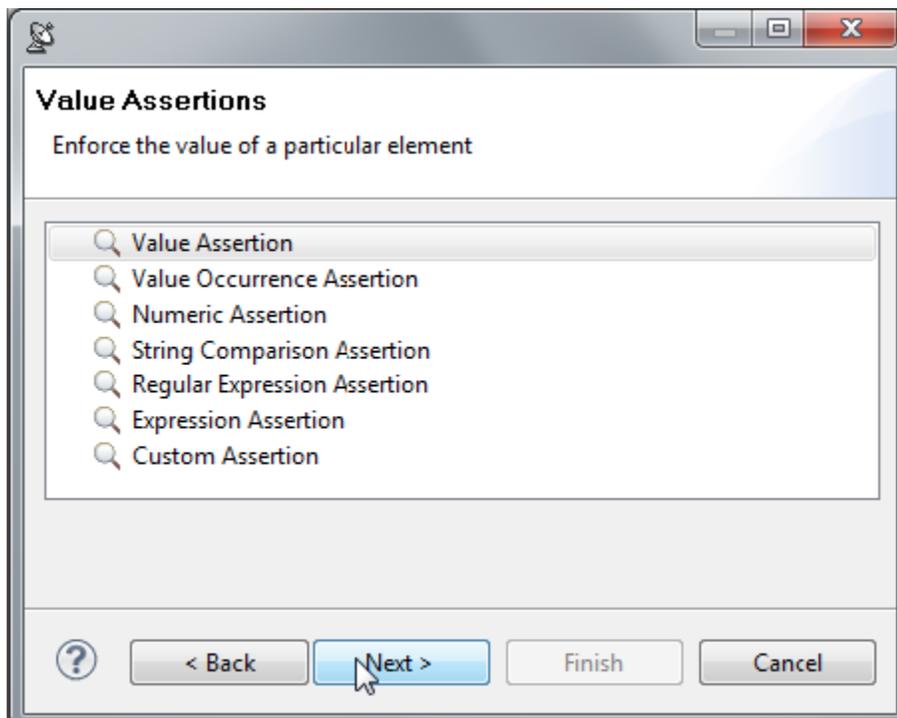
d. In the tree that is shown, select the first **IFT02-FreeText** node (the full path to node is "IATA/PAORES/IFT-InteractiveFreeText/IFT02-FreeText") and click **Finish**.
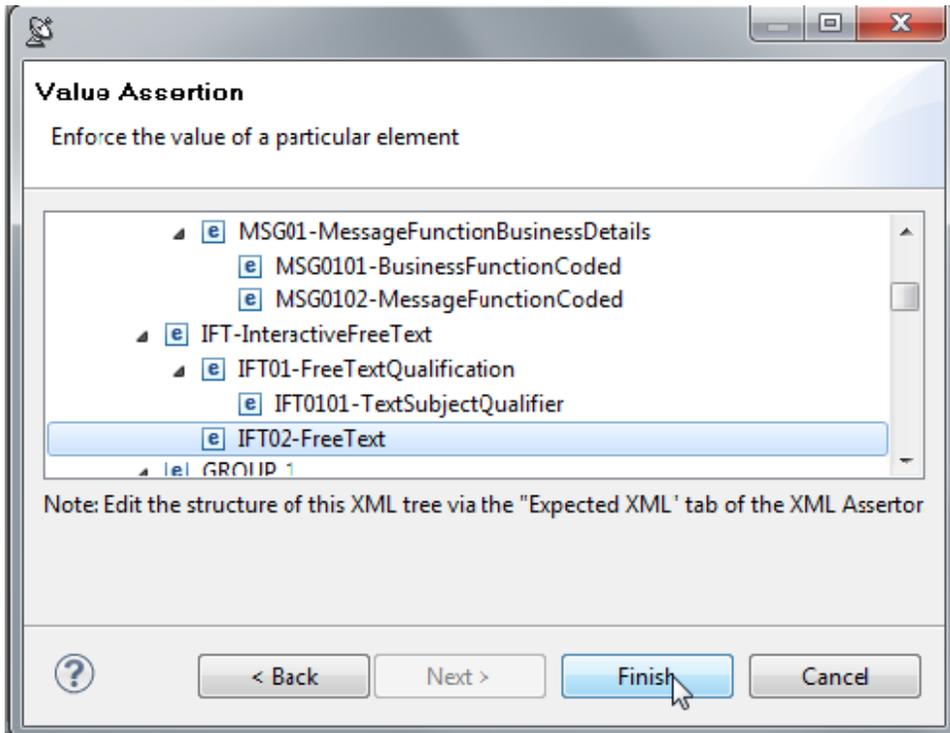
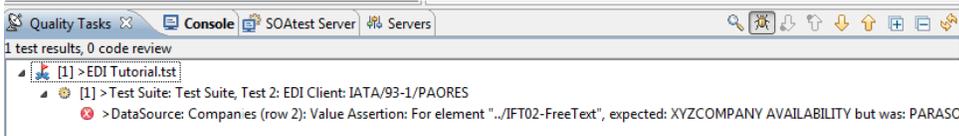   **e.** Enter `XYZCOMPANY AVAILABILITY` into the **Expected Value** field.

       f. Save the XML Assertor.

4. Run the test. It should fail with the message 'DataSource: Companies (row 2): Value Assertion: For element "../IFT02-FreeText", expected: XYZCOMPANY AVAILABILITY but was: PARASOFT AVAILABILITY '.

This mismatch occurs because the request is parameterized, but the XML Assertor is not.

5. Change the Expected Value field in the XML Assertor to **Parameterized**. The combo box to the right should show the column in our data source named "Company Availability."

6. Save the XML Assertor.
7. Run the test. It should now succeed.

The .tst file corresponding to these exercises can be found at `<SOAtest install dir>/examples/tests/EDI`.