

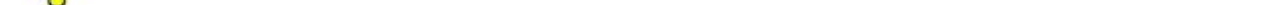
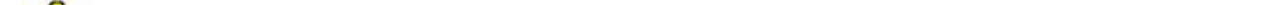


OpenSpan Elective Training

- Using the Service Client Component
- Add and Configure the Service Client Component
- Supplying a Simple Input to a Service
- Use a Service Method to Return Data in a Table Format
- Service Result Returned as a Complex Data Type
- Extracting Data from Complex Data Type Result
- Data Input and SOAP Headers

Prerequisites	5
Background Information	5
Working with the Service Client Component.....	6
State Management	7
Editing WSDL Definitions	7
Solutions Used in this Training Module	7
Project Outline	8
 Exercise – Simple Result – getSuppliers & getProducts.....	 11
Exercise – Simple Input/Simple Result - getsupplierForProduct	17
Exercise – Simple Input/DataTable Result — getProductDetail	19
Exercise – Simple Input/Complex Result getProductDetailsasObject	21
Exercise – Simple Input/Complex Result-getProductsForSupplier	25

You can save time using this training guide by understanding how screen elements, input data, and definitions are shown.



This guide describes how to use the Service Client component in OpenSpan Studio.

This document is intended for readers familiar with OpenSpan Studio to design and create solutions. The document also assumes that you are familiar web services and WSDL (Web Service Description Language) definitions.

OpenSpan Studio contains a ServiceClient component. This component provides web service capabilities within an OpenSpan Studio solution. In OpenSpan, the term web service describes a set of functions or procedures that can be accessed and executed remotely using NTP, TCP, HTTP, HTTPS and WS*protocol.

In OpenSpan Studio, web services can contain inputs/outputs with simple or complex data types. A complex data type is a data structure that contains multiple pieces of information. For web services containing complex types, a new Toolbox group is added for the web service where all of the complex types are listed. When you select a complex type, you can see all of the properties for that type. This allows you to easily access and manipulate the complex types contained in web services. Refer to of this document for more information on working with complex types.


OpenSpan's ServiceClient component also supports [State Management](#) and SOAP Headers. Refer to of this document for more information on SOAP headers.

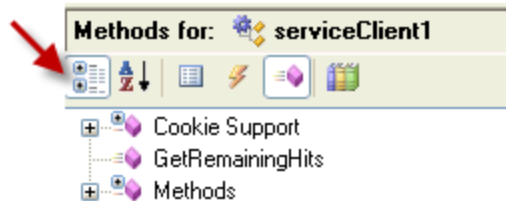
The steps below outline the tasks required to add web service capability to an OpenSpan Studio solution.

1. Add a Global Container project item to the solution.
2. Add the ServiceClient Component to the Global Container
3. Set the WSDL property for the component
4. Query the WSDL to expose the available methods of the service
5. Set SOAP Header / State management for the service
6. Set input parameters for the service
7. Trigger the service event point
8. Get ServiceClient result directly or create Proxy for complex data results

Additionally, you may need to set specific properties for the component depending on the service you are accessing. For a list of properties and their definitions, refer to the OpenSpan on-line Help topic:

The ServiceClient component can use a cookie-based state management type. When a user logs in, an ID is issued to the user and stored on the client side. That ID is attached to the service method call and pushed to the service for authentication purposes. Note cookies require the http communication protocol.

OpenSpan Studio lists both the service methods and cookie methods in the Object Explorer. You can click on the  button to see the methods categorized by type.






There are two Helper methods for cookies:

- SetCookie – Sets the cookie on the client side
- GetCookie – Gets the cookie on the web service

The session ID is the same as a cookie, but it has service component tracking via ASP.net. If you use an ASP.net service, you will most likely use session ID as the state.

OpenSpan Studio offers the following three commands to help edit and manage WSDL definitions.

-  – Allows you to query the WSDL to see the available service methods. If you've already started using the WSDL, the editor is read-only.
-  – Displays the WSDL XML in a browser.
-  – If you add service methods and want to refresh the WSDL so you can use those methods, you must close and restart OpenSpan Studio after clicking Refresh Web Methods. You will see a confirmation message when you click Refresh Web Methods. If you have errors in your automation, you will see a warning message instead of a confirmation message, and you should correct the errors before restarting OpenSpan Studio.

Practice files you will use while working through the step-by-step exercises presented in this training guide can be [downloaded here](#). Finished solutions are available from the same download location as a reference for checking your work after completing the exercises on your own.

To learn more about the ServiceClient component, this guide provides steps for building the following example solutions.

- ServiceClient Example 1: Simple Result - getSuppliers & getProducts
- ServiceClient Example 2: Simple Input/Simple Result - getsupplierForProduct
- ServiceClient Example 3: Simple Input/DataTable Result - getProductDetail
- ServiceClient Example 4: Simple Input/ Complex Result 1 – getProductDetailsasObject
- ServiceClient Example 5: Simple Input/ Complex Result 2 - getProductsForSupplier

The sample solutions use the following web service:

<http://demows.openspan.com:5304/Openspan?wsdl>

This web service is based on queries made to the Microsoft sample Northwinds database and has the following methods available:

getProductDetail	Product as String	DataTable of Product Details
getProductDetailsasObject	Product as String	Complex DB of Product Details
getProducts	None	String Array of Products
getProductsForSupplier	Supplier as String	Complex DB Array or Product Details
getsupplierForProduct	Product as String	Supplier as String
getSuppliers	None	String Array of Suppliers
SayHello	Name as String	String

All five examples use the sample Windows Form

that you can [download here](#).

Service Client Example

How To use the Service Client Component Sample Solution

<http://demows.openspan.com:5304/Openspan?wsdl>

Products Suppliers

Northwind Product List

lbxProd

Listbox populated in first solution.

Product Co. Name:

Product ID:

Product Name:

Quantity Per Unit:

Unit Price:

Units In Stock:

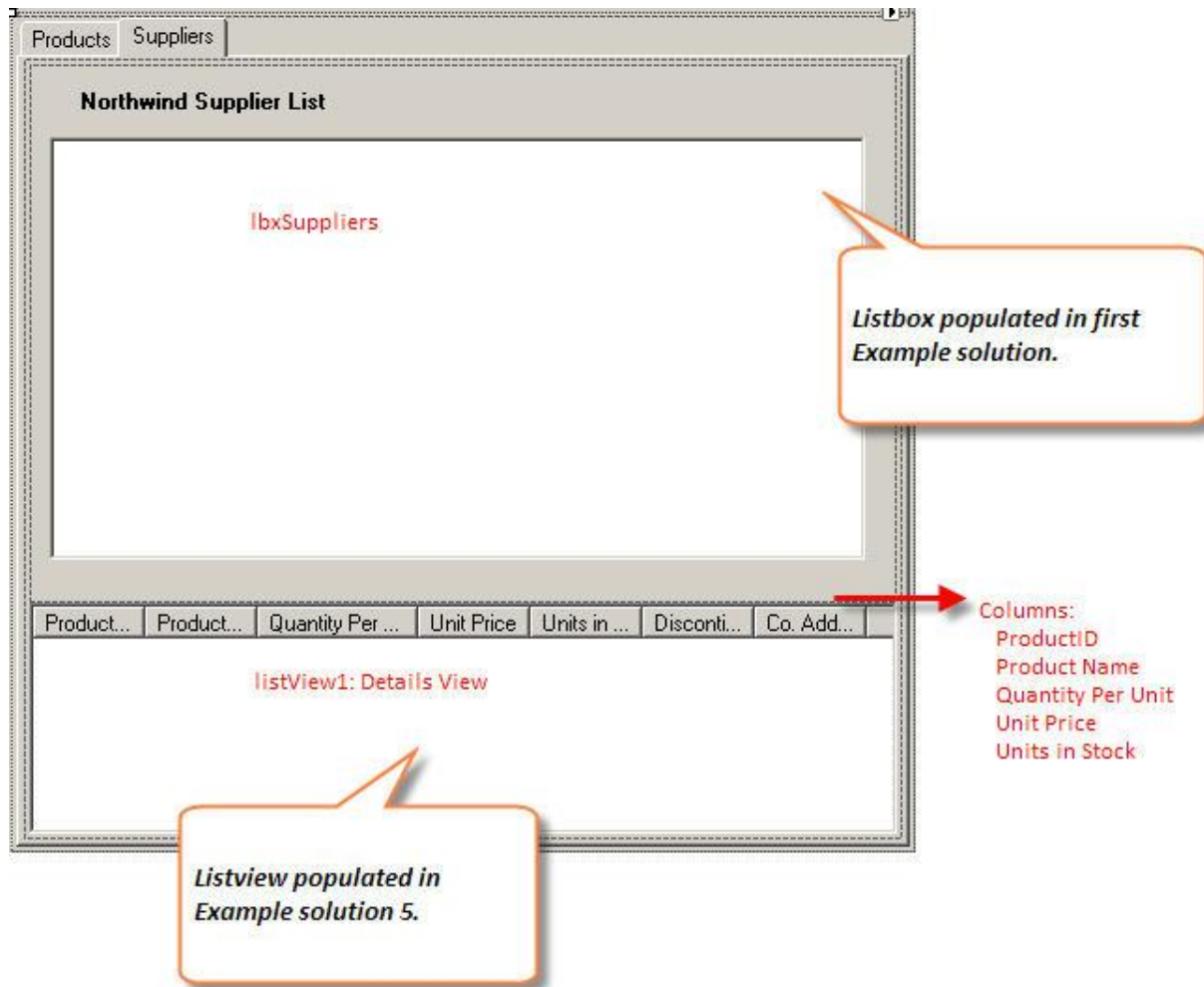
Discontinued?

Company Address:

These labels are hidden in the first two example solutions.

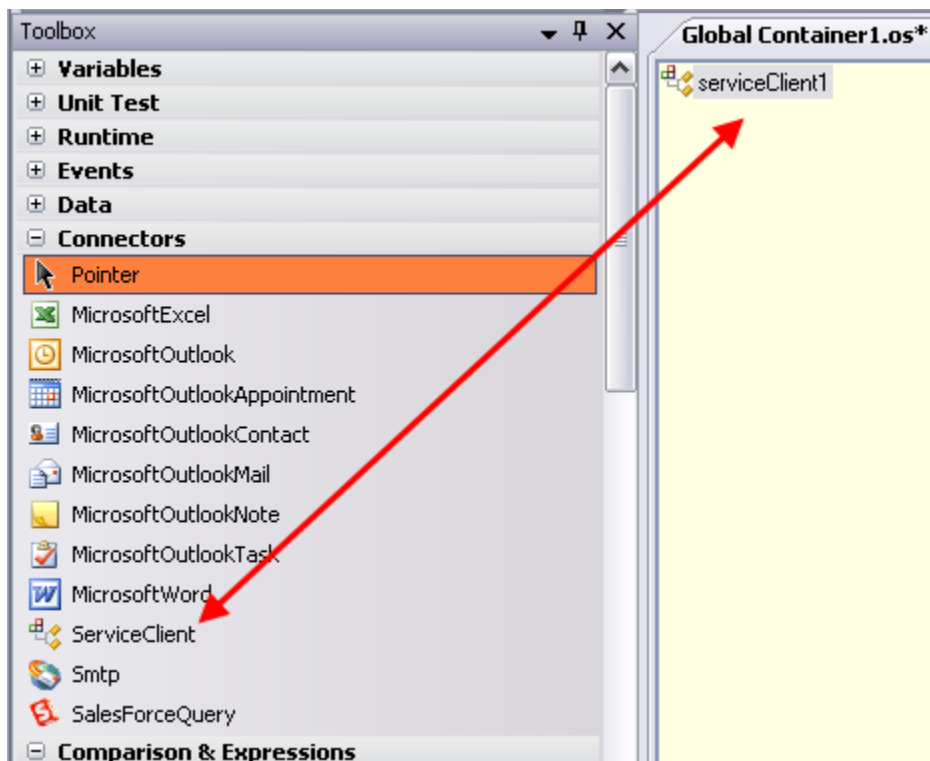
DataGrid

The DataGrid is populated in the third solution.



This example solution shows how to add and configure the Service Client component and how to use the component to retrieve a simple list. The first automation in the project will add the Service Client and retrieve two lists of data – products and suppliers from the Northwinds database.

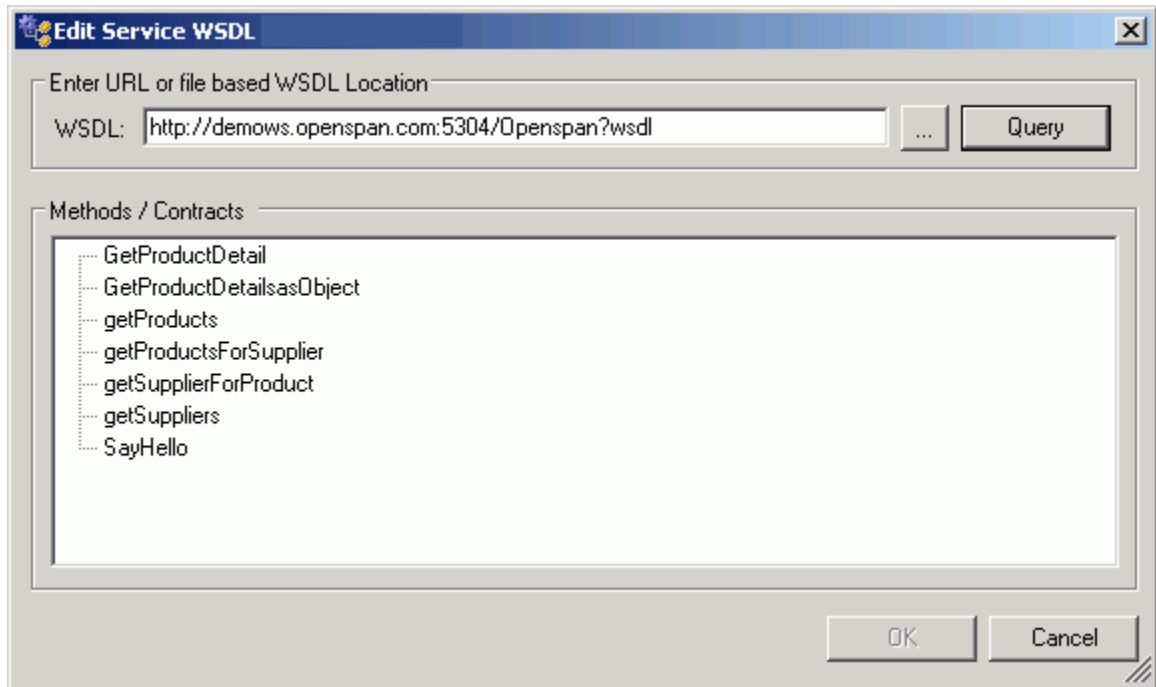
1. Open the _____ solution to get started. Save the solution as _____.
2. Add a _____ project item.
3. Add a _____ component to the _____. The ServiceClient component is located on the _____ tab of the OpenSpan Toolbox.



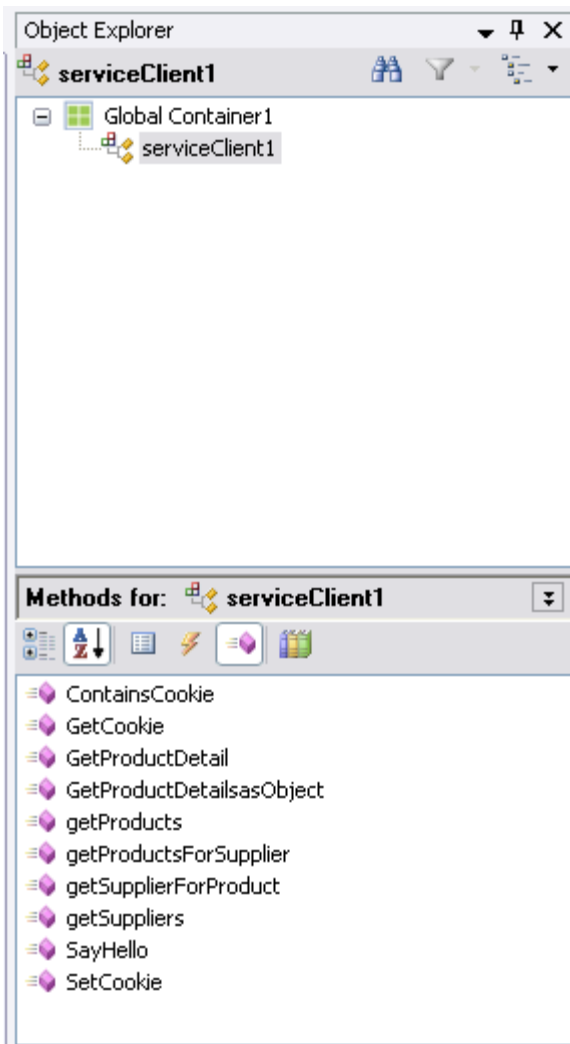
4. Browse the _____ property of the Service Client component to open the dialog. Enter the following WSDL in the _____ field:

<http://demows.openspan.com:5304/Openspan?wsdl>

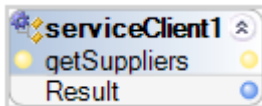
On the Edit Service WSDL dialog, click the _____ button. A list of methods is retrieved once you are connected to the service. Your dialog should look like the following:



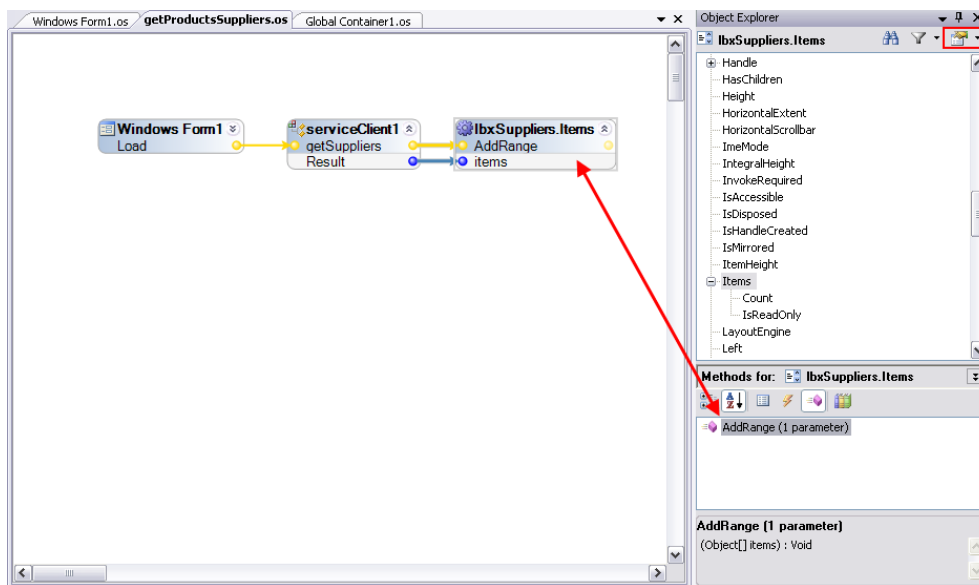
5. Click to continue. The service and methods are added to your solution.



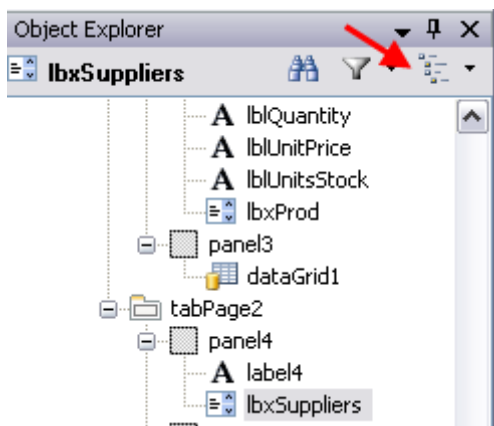
6. Add a new automation to the solution. Name the automation: .
7. Add the method to the automation.



8. To trigger the method, use the event for the of suppliers, use the property of . Complete the automation as follows:

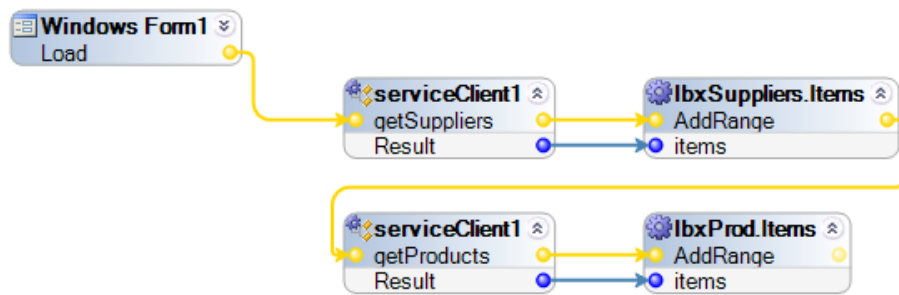


: To locate the method for the property, select the Suppliers listbox and click the button in the . Next, highlight the property in the Properties treeview and then browse the Properties, Methods, and Events for the method.

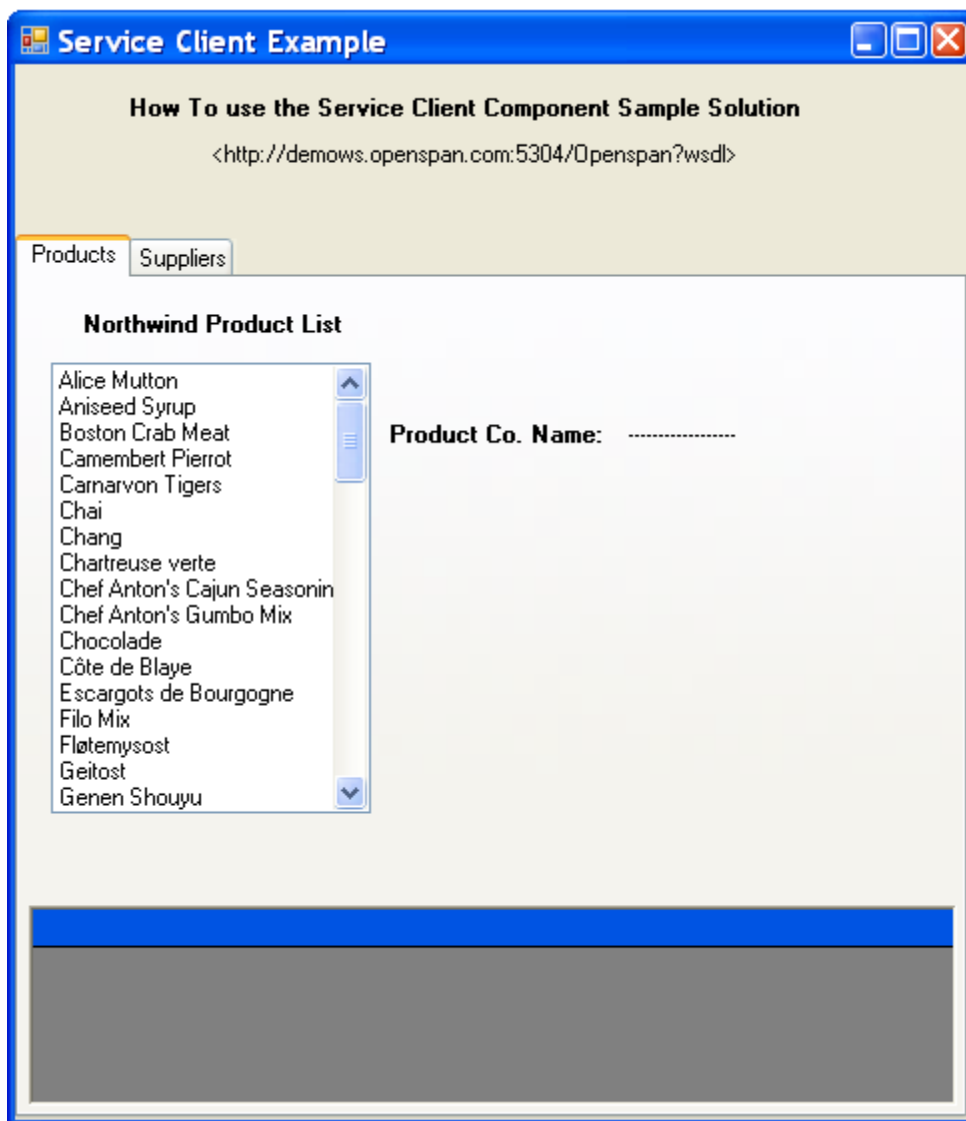


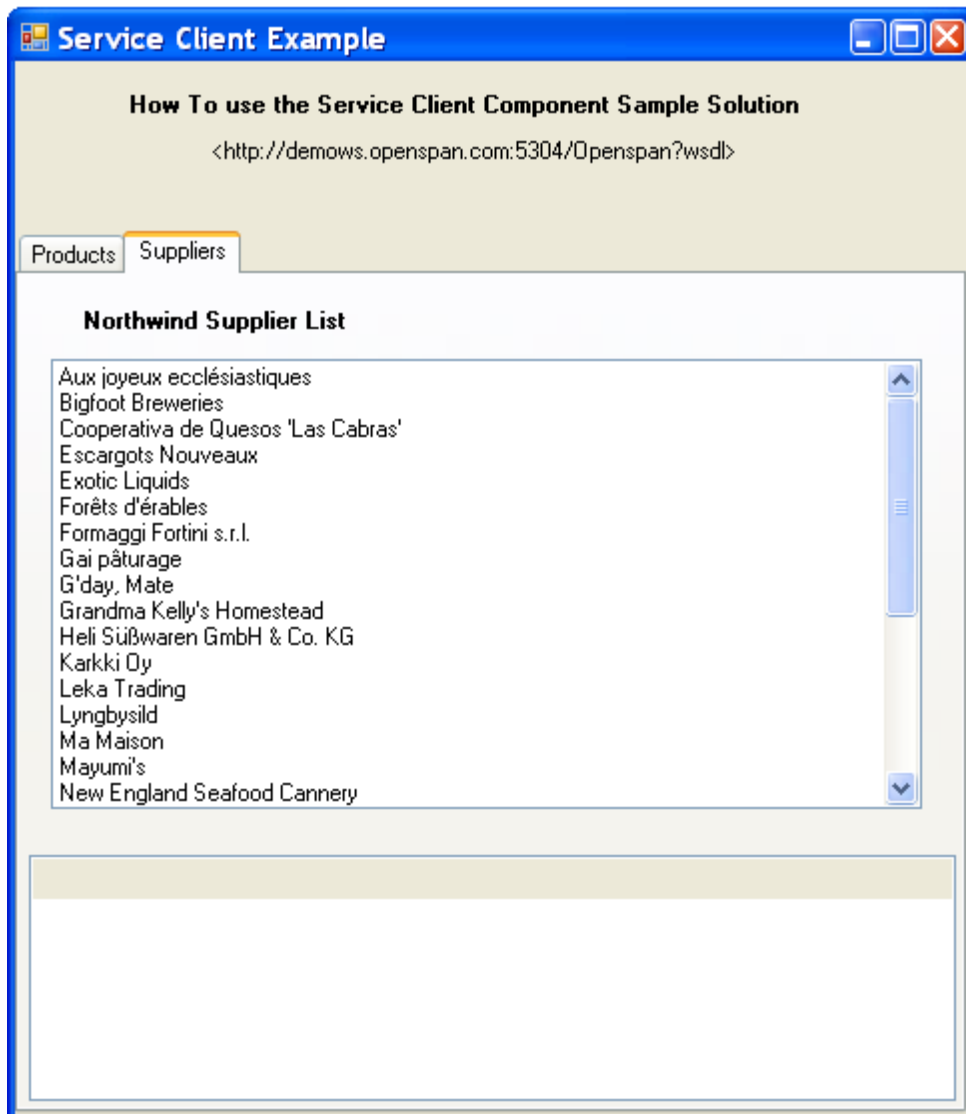
9. Add similar logic to the automation to populate the Products listbox. Add the method for the service to the automation.
10. Add the method to the automation.

11. Connect the components as follows:



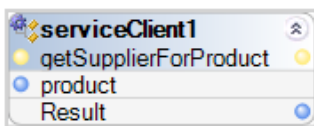
12. Save and run the solution. When the Windows form opens the products list should populate. Click the **Suppliers** tab to view a list of suppliers. Your form should look similar to the following:



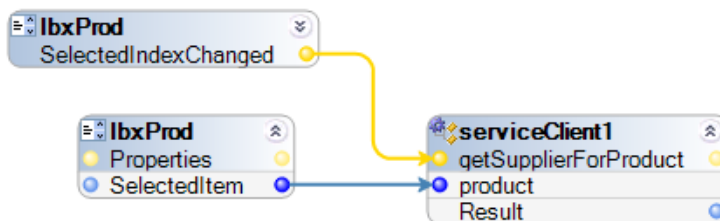


The next example solution shows how to supply simple input to a service. In this project, a product (string) is input to the service and the Supplier name (string) is returned as the result.

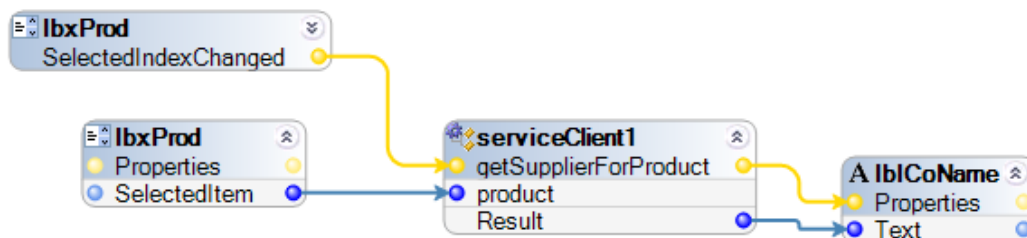
1. Save the solution you created in Example 1 as _____.
2. Add a new automation to the solution. Name the automation: _____.
3. Add the _____ method for the service to the automation.



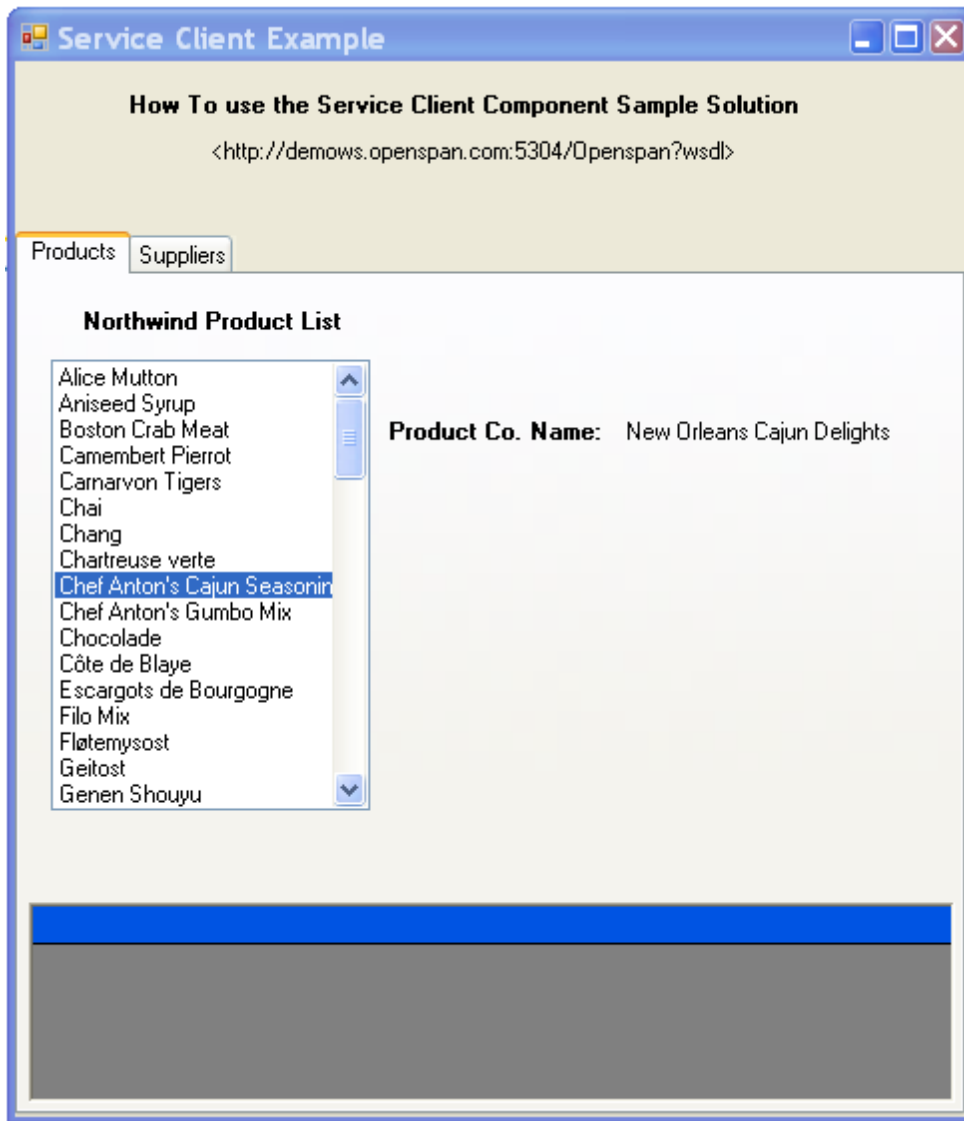
4. The input for the _____ method requires the Product Name in string format. We can use the _____ from the Products listbox as the input. The triggering event for this automation will be when the user selects an item, or, _____. Add these components to your automation as follows:



5. The Result from the _____ is a string containing the Supplier Name. This output can be displayed in the _____ control. Complete the automation as follows:

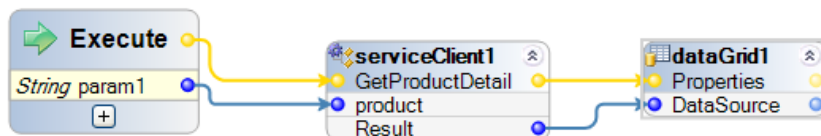


6. Save and run the solution. After the Windows form opens, select a product from the Product List. A supplier name should appear in the field as shown in the following example.

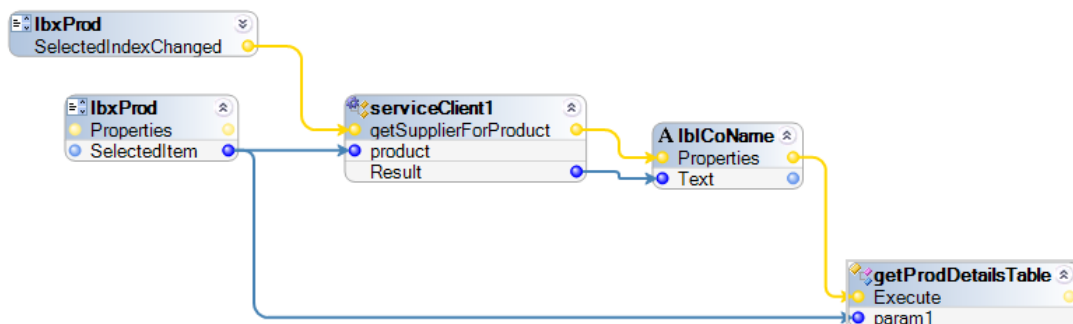


This example shows how to use a service method which returns data in data table format. Building on the solution built in Example 2, we will use the `getProductDetail` method of the service to return details for a product selected from the Products list box.

1. Save the solution built in Example 2 as _____.
2. Add a new automation to the solution. Name the automation: _____.
3. Add the _____ method for the service to the automation.
4. The _____ method requires a Product Name as an input parameter. Since this has already been set in the _____ automation, we can use an entry point on the _____ and pass the value from the _____ automation to this automation. Add an Entry point with 1 string parameter to the _____ automation.
5. The Result of the _____ method is in DataTable format. We can use the _____ on the windows form to display the data in this format. Add the _____ property for the _____ to the automation. Your automation should look like the following.



6. To trigger the _____ entry point, return to the _____ automation and add the _____ method for the _____ automation. Connect the data port to the _____ and pass the event from the _____ as shown in the following automation:



- Save and run the solution. When you select a product from the Product List, the supplier Company Name displays and details about the product display in the grid at the bottom of the form as shown in the following example:

Service Client Example

How To use the Service Client Component Sample Solution

<http://demows.openspan.com:5304/Openspan?wsdl>

Products Suppliers

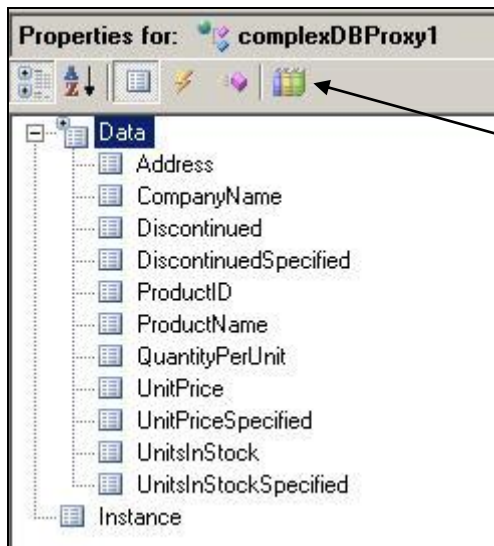
Northwind Product List

Alice Mutton
Aniseed Syrup
Boston Crab Meat
Camembert Pierrot
Carnarvon Tigers
Chai
Chang
Chartreuse verte
Chef Anton's Cajun Seasonin
Chef Anton's Gumbo Mix
Chocolade
Côte de Blaye
Escargots de Bourgogne
Filo Mix
Fløtemysost
Geitost
Genen Shouyu

Product Co. Name: Aux joyeux ecclésiastiques
Product ID:
Product Name:
Quantity Per Unit:
Unit Price:
Units In Stock:
Discontinued?
Company Address:

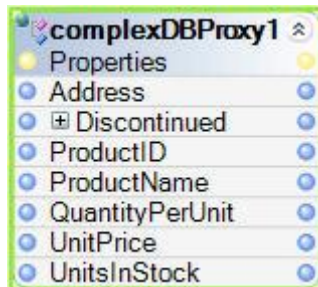
	ProductID	ProductName	QuantityPerU	UnitPrice	UnitsInStock	Discontin
▶	38	Côte de Blay	12 - 75 cl bott	263.5000	17	<input type="checkbox"/>
*						

6. In the Object Explorer, click on the object and then explore the Properties for the object. Note that the details for the Product are listed as properties:



If the properties do not appear by default, click the browse button and add them to the Filter.

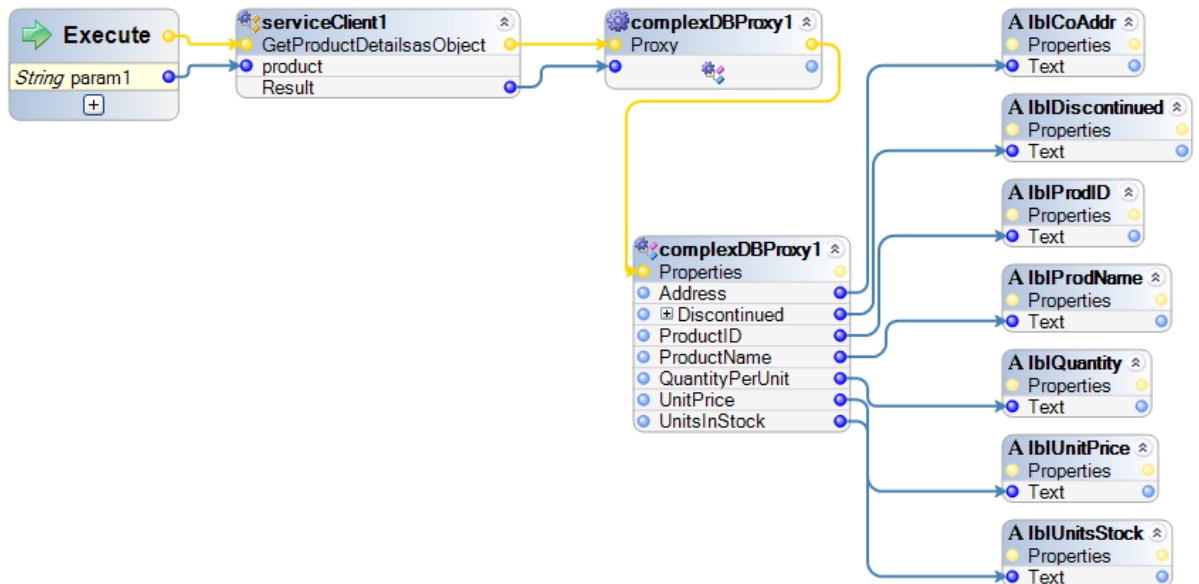
7. Select the following Properties for the object and add them to the automation:



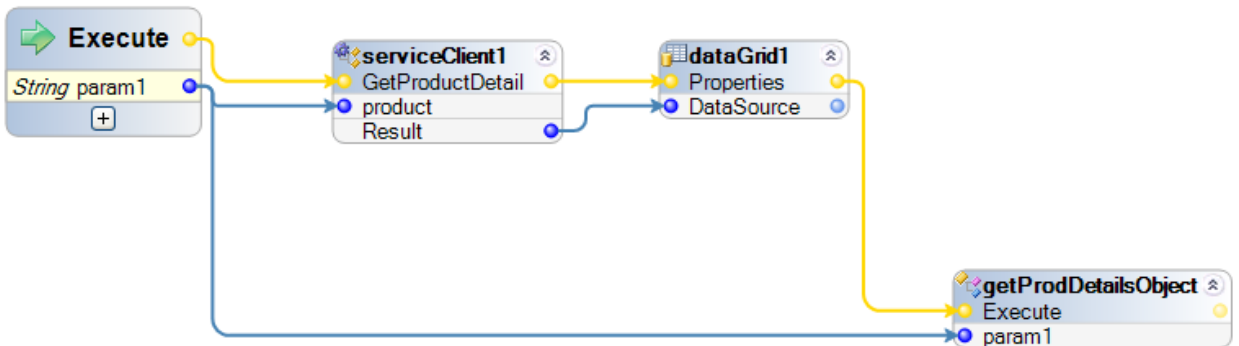
If you hold down the CTRL key while selecting Properties, OpenSpan will place all of the Properties in a single design block as shown above.

8. Add the Windows Form labels for the corresponding Product Details properties to the automation.

9. Pass the event link from the service object, and then to the method to the object. Your completed automation should look like the following:



10. To trigger the entry point, we will use the method for the automation. Return to the automation and add the method.
11. Connect the data parameter for the method from the automation's entry point and pass the event link from the control. The automation should look like the following:



12. Save and run the project. After you select a product from the Product List, details about the item display in the data grid as well as in the labels on the Products tab as shown in the following example:

Service Client Example

How To use the Service Client Component Sample Solution

<http://demows.openspan.com:5304/Openspan?wsdl>

Products Suppliers

Northwind Product List

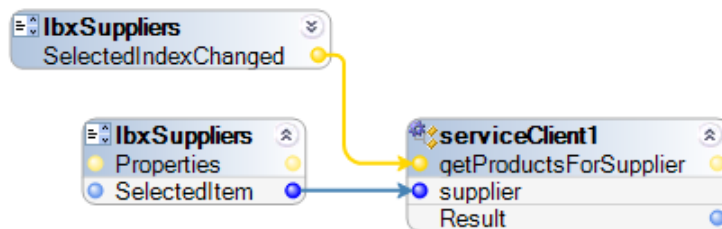
Alice Mutton
Aniseed Syrup
Boston Crab Meat
Camembert Pierrot
Carnarvon Tigers
Chai
Chang
Chartreuse verte
Chef Anton's Cajun Seasonin
Chef Anton's Gumbo Mix
Chocolate
Côte de Blaye
Escargots de Bourgogne
Filo Mix
Fløtemysost
Geitost
Genen Shouyu

Product Co. Name: New England Seafood Cannery
Product ID: 40
Product Name: Boston Crab Meat
Quantity Per Unit: 24 - 4 oz tins
Unit Price: 18.4000
Units In Stock: 123
Discontinued? False
Company Address: Order Processing Dept.
2100 Paul Revere Blvd.

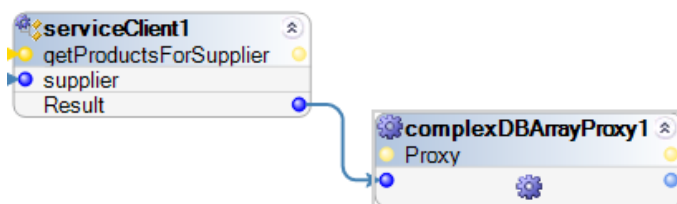
	ProductID	ProductName	QuantityPerU	UnitPrice	UnitsInStock	Discontin
▶	40	Boston Crab	24 - 4 oz tins	18.4000	123	<input type="checkbox"/>
*						

This example solution shows how to extract data from a complex data type result where the service returns multiple datasets. The `getProductsForSupplier` method for the service returns a set of product records. A Proxy is used to access the group of records and a ForLoop control is used to iterate through the Proxy to access each product record. The individual product records are displayed in a ListView on the Windows form in the solution.

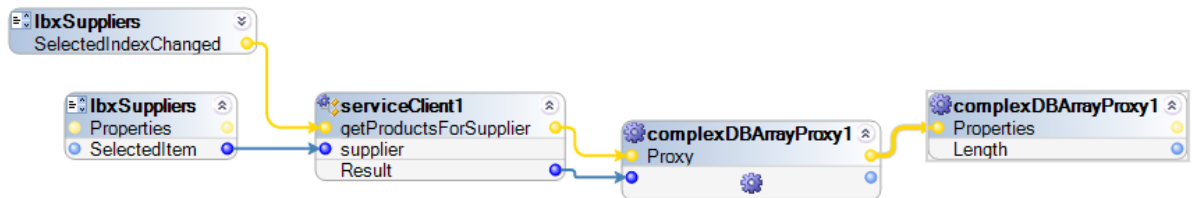
1. Save the solution created in Example 4 as _____.
2. Add a new automation to the solution. Name the automation: _____.
3. Add the _____ method for the service to the automation.
4. The _____ method requires a supplier name input as a string parameter. We can use the _____ from the Suppliers listbox as the input. The triggering event for this automation will be when the user selects an item, or, _____. Add and connect these components on your automation as follows:



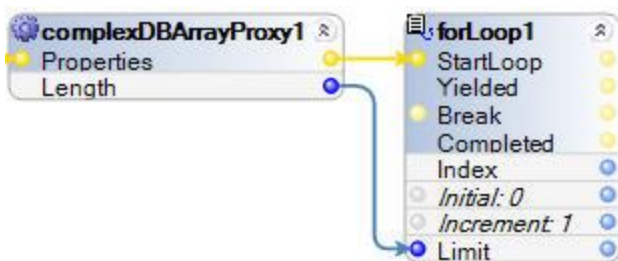
5. The Result of the _____ method is a complex data type. To access this data, we need to create a _____ object. Right-click the Result output port and select _____. The _____ object is created.



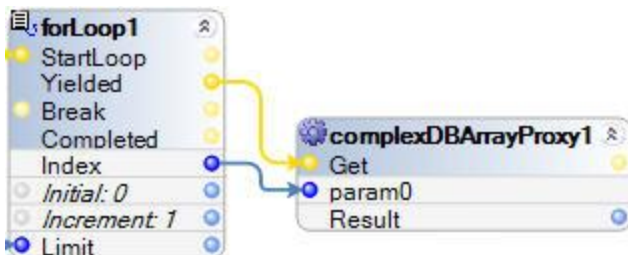
6. The `IbxSuppliers` object contains a set of product detail records. To display these records we will add each one to the `ComplexDBArrayProxy1` on the Suppliers tab. In order to iterate through the records we will use the `forLoop1` control. The `forLoop1` control requires a `Limit` parameter input. In this case, the `Limit` is the number of records contained in the `ComplexDBArrayProxy1`. This value can be obtained from the `Length` property of the `ComplexDBArrayProxy1`. Add the `ComplexDBArrayProxy1` property to the automation and pass the event link from the `SelectedIndexChanged` method to the `getProductsForSupplier` as follows:



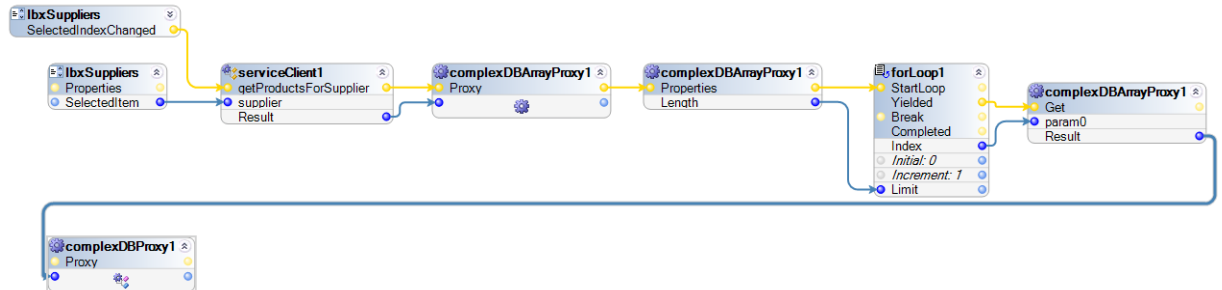
7. Add the `forLoop1` component to the automation. Note that the `Limit` input parameter is set to 0 as default. Hover the mouse over this setting, right-click and select `ComplexDBArrayProxy1` from the local menu. Connect the `Length` property data output to the `Limit` input. Pass the event link from the `SelectedIndexChanged` event to the `getProductsForSupplier` as follows:



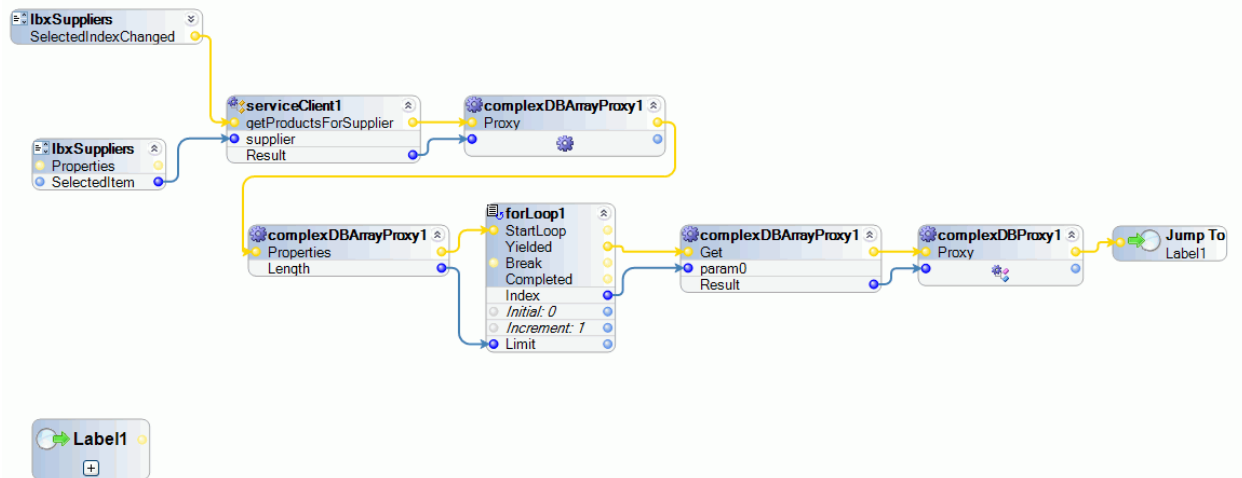
8. As the `forLoop1` starts each iteration, we want to retrieve a specific row in the array. The `Get` method for the `ComplexDBArrayProxy1` retrieves a row in the array for a given index value. Add this method to the automation and connect the `Index` data output from the `forLoop1` to the `param0` input of the `Get` method. Pass the event link from the `Completed` event to the `NextIteration` method as follows:



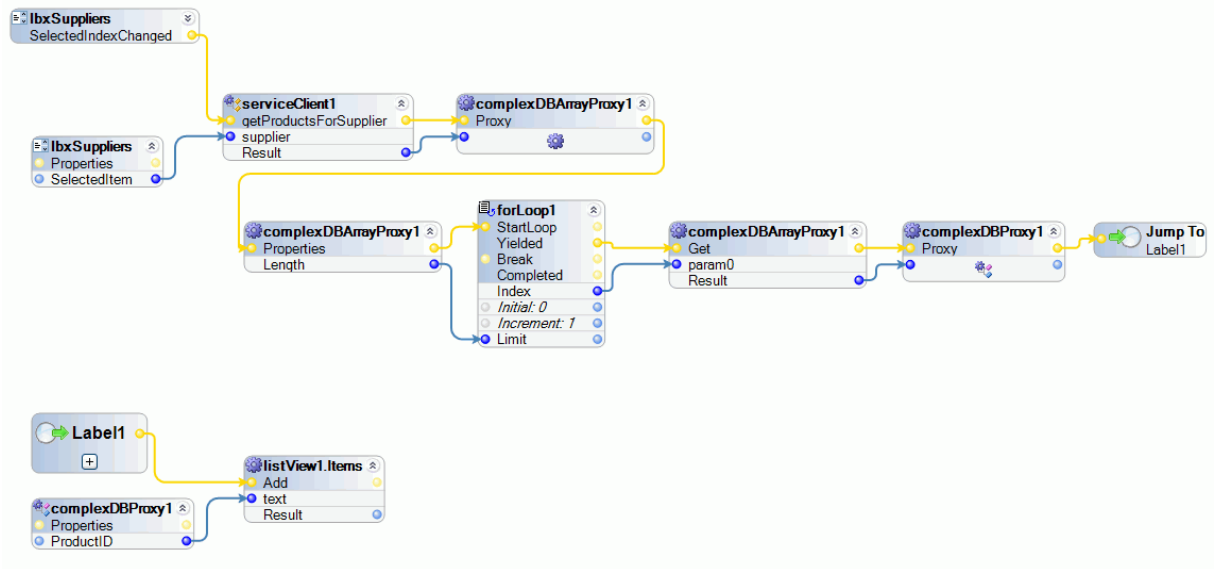
9. The Result of the method is a complex type – a data record or row. To use the record data to populate the elements. Create a for loop for the Result of the method. The automation up to this point should look like the following:



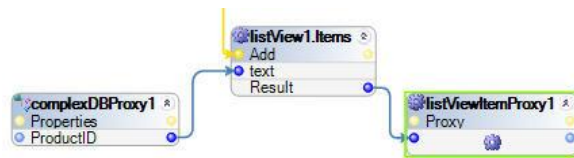
10. To enhance readability of the automation, add a Jump – Label (see the on-line Help topic: [Label and Jump To](#)) and connect as follows:



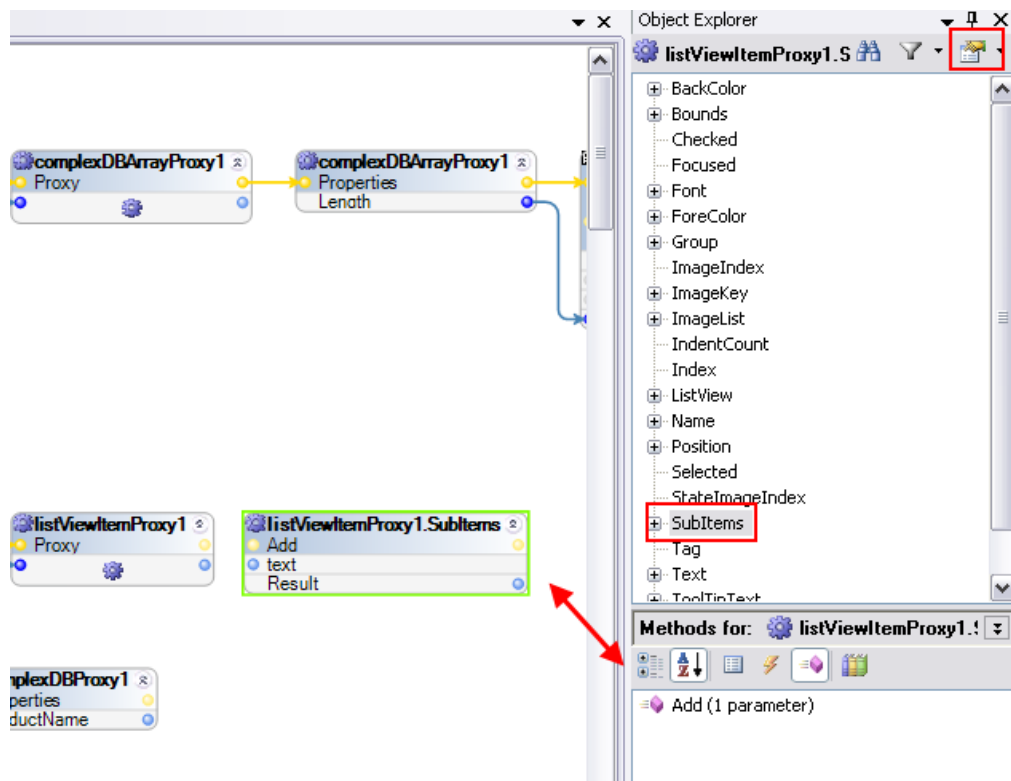
11. We are ready to add the key field to the . For the data set we are working with, the key field is the ProductID. Using the property, (String Text parameter) method (accessed from the tree) connect the property data output to the method and pass the event link from the method to the method as follows:



12. Create a from the Result of the method to access the sub-items for the ListView.



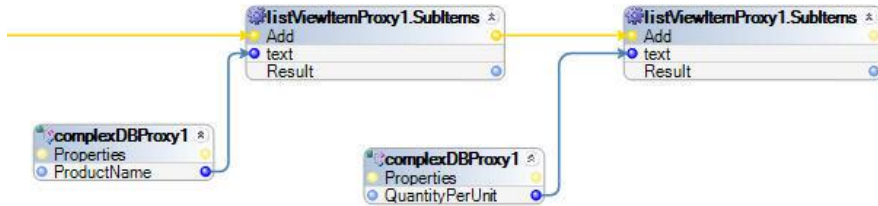
13. Add the `listViewItemProxy1.SubItems` (String text parameter) method to the automation.
: Make sure to use the method having string parameter input. Access this method from the tree.



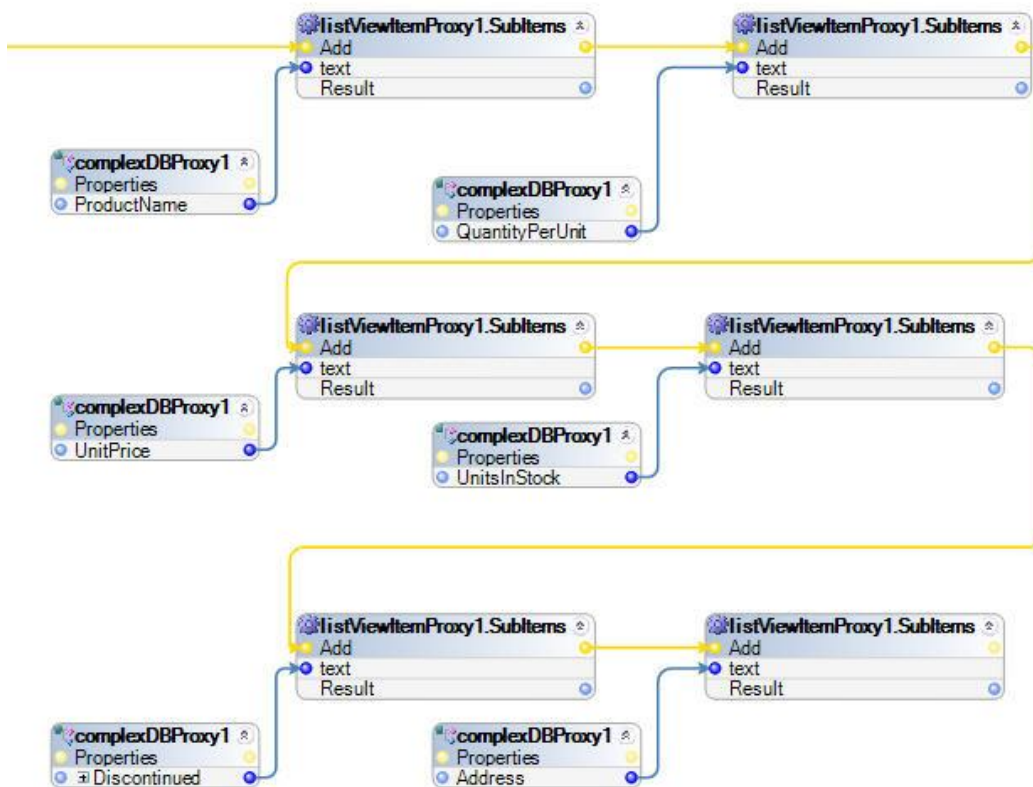
14. Add the `listViewItemProxy1.SubItems` property to the automation and connect it to the `listView1.Items` method. Pass the event link from the `listView1.Items` method as follows:



15. Use the `Add` method again, this time connect the `text` property to the `Result` method:



16. Continue using the `Add` method with the following properties: `UnitPrice`, `UnitsInStock`, and `Discontinued`. This part of the automation should look like the following:



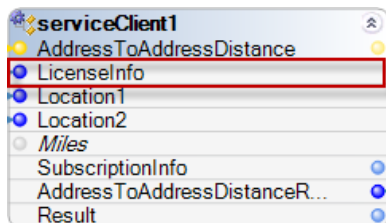
17. Save and run the automation. Once the Product List appears, click the Suppliers tab. Select a supplier from the list. Products for the supplier should appear in the ListView at the bottom of the Windows form as shown in the following example:

Northwind Supplier List

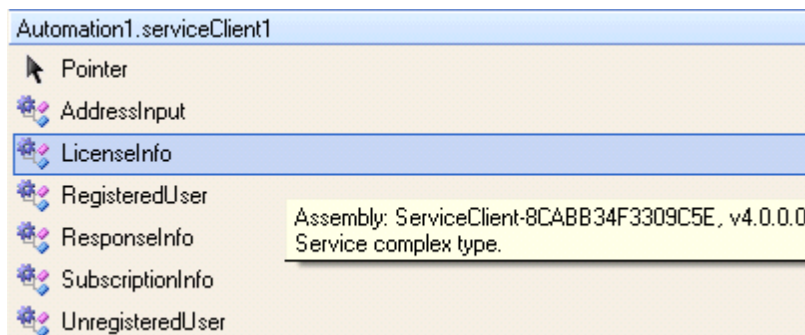
Aux joyeux ecclésiastiques
Bigfoot Breweries
Cooperativa de Quesos 'Las Cabras'
Escargots Nouveaux
Exotic Liquids
Forêts d'érables
Formaggi Fortini s.r.l.
Gai pâturage
G'day, Mate
Grandma Kelly's Homestead
Heli Süßwaren GmbH & Co. KG
Karkki Oy
Leka Trading
Lyngbysild
Ma Maison
Mayumi's
New England Seafood Cannery

Product ID	Product Name	Quantity Per Unit	Unit Price	Units in Stock	Discontinued	Co. Address
1	Chai	10 boxes x 20 bags	18.0000	39	False	49 Gilbert St.
2	Chang	24 - 12 oz bottles	19.0000	17	False	49 Gilbert St.
3	Aniseed Syrup	12 - 550 ml bottles	10.0000	13	False	49 Gilbert St.

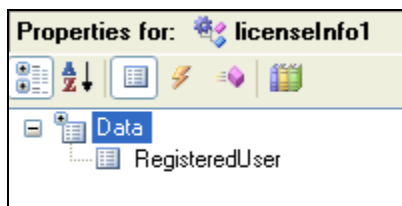
Some services require SOAP Header input. When using a service method that requires a SOAP Header, the header parameter(s) is included directly on the connection block for the method. In the following example, a service requires SOAP Header input in order for the service to be accessed.



In this case, the License is of a complex type. The License is input by a Registered User value. The Registered User Value is made up of a username and password. When the service is queried, the following complex types are shown on the Toolbox:



To set the on the automation, we must first set the input.



The contains two input parameters: Password and UserID.



So the automation starts by setting the `password` and `userid` for the `registeredUser1` object:



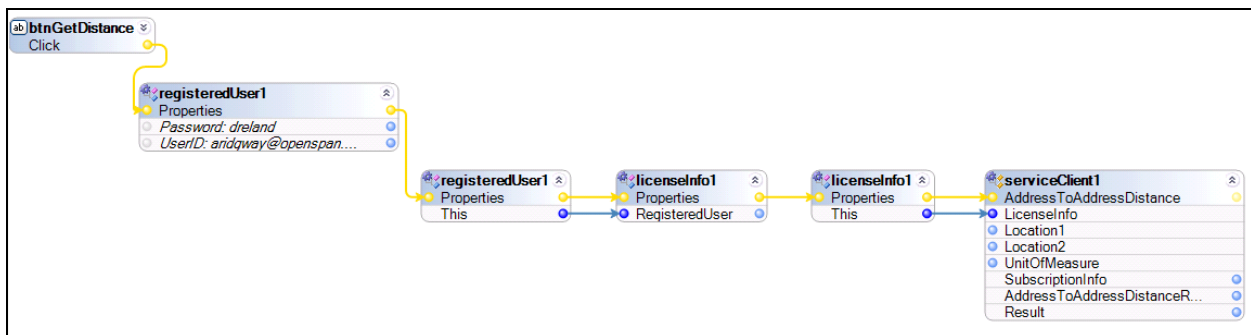
Next, the `registeredUser1` object is sent to the `licenseInfo1` object:



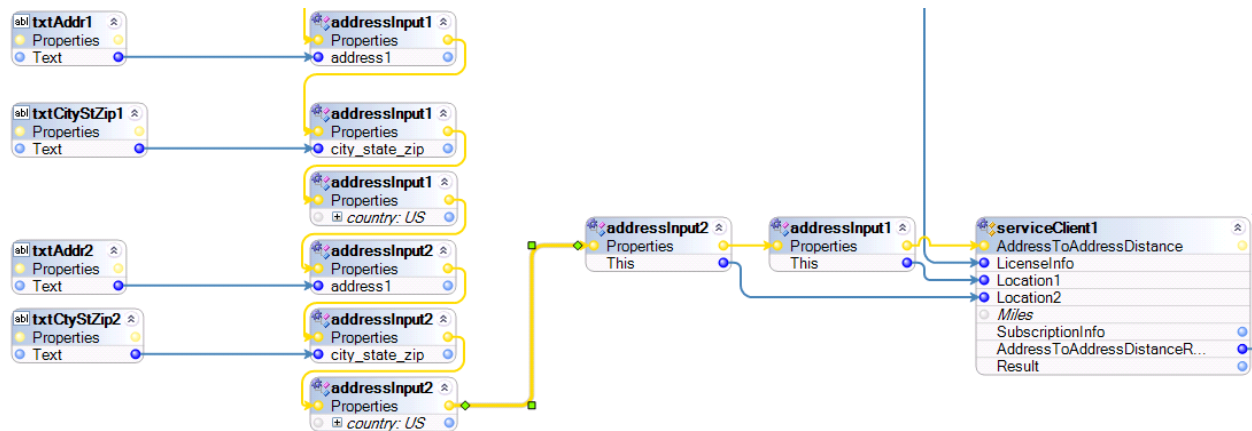
The complete part of the automation, which sets the `password` and `userid` for the `registeredUser1` object, follows:



Next, the `licenseInfo1` object must be sent to the `ServiceClient` `AddressToAddressDistance` parameter on the service call `btnGetDistance` method. See the following automation segment:



For this sample service, the Location 1 and Location 2 parameters are set in a similar way – using complex type input parameters. See the following automation segment:



The rest of the input parameters to the method can be set as usual. Note that triggering any of the service methods before setting the will result in an error.