

How to

Integrate Web Services

Tools: Application Manager,
Designer

Duration: 25 minutes

Skill Level: Medium

Summary: An introduction to using Web Services
with AltioLive Studio.

Integra SP

88 Wood Street London
EC2V 7RS
United Kingdom

www.altio.com
tel: +44 (0) 20 8528 1045

Contents

Introduction	4
Welcome.....	4
What you will need	4
Should you need help	5
License Terms	5
Web Services	6
Using the Web Services Demo Wizard	7
Accessing the Web Services Demo Wizard.....	7
Use the Web Services Demo Wizard to test a web service that uses RPC binding	9
Saving a Service using the Web Services Demo Wizard	12
Modifying the Web Services Demo Wizard	12
Using the Web Service Application.....	13
Open the WS application	13
Test the WS application	15

View the Datatypes associated with the web service	16
Explore the View.....	18
Using the WS Application	20
The SOAP Service Function	22
The Request Tab	23
Request headers section.....	24
Message Body section (This defines the structure of the request):	25
The Response Tab.....	27
Aliases	27
When Services is Used for Initial Data Request	28
Returned Data.....	29
SOAP Response Processing.....	29
The Acknowledgement Tab	33
The Test Tab.....	34
Test Properties (Displays the parameters available for this SOAP Service).....	34
Test Results:.....	34

Find SOAP Service	35
Find and add an operation from a web service	36
Troubleshooting.....	39
General	39
Other sources of help	40
Further Information / References.....	41

Introduction

Welcome

As a new AltioLive developer you should use this document to help you:

- Understand the core concepts of integrating Web Services using the AltioLive Application Manager
- Use the Application Manager to link to a Web Service using the Web Services Demo Wizard.

This document provides an overview of the processes involved in using the Application Manager to link to Web Services. It contains tutorial exercises.


What you will need

- You should have already completed the Introduction to the AltioLive Smart Client document.
- Before progressing with this document, you should study: **Getting Started with the AltioLive Designer** and **Getting Started with the Application Manager**.
- You will need a basic understanding of web services, web applications and XML.

Remember that you must be connected to the Internet to access the example web services described in this document.

***Please note:** also that the examples are not controlled by Altio and may be unavailable without notice.*

Should you need help

See the Troubleshooting chapter at the end of the document. The online help (available from the AltioLive console or by clicking the  button in any of the Altio tools) gives detailed instructions on all features of AltioLive, and includes a more detailed troubleshooting section.

Also at the end of this document are contact details for Altio Technical Support in the event that you are unable to resolve an issue.

License Terms

The software is licensed under the terms committed to when the software was explicitly licensed or under the terms you (or someone else) committed to when they installed the software.

Web Services

AltioLive Studio provides access to web service operations described as Remote Procedure Calls (RPC) and document passing in a WSDL (Web Service Definition Language) file, using standard SOAP (Single Object Access Protocol) encoding via HTTP data transfer.

The web service must use a WSDL file that includes a SOAP binding for the service definition to be invoked. WSDL is an XML-based language used to define Web services and describes how to access them.

Web Services integrate as an AltioLive Service Function that can be invoked by the AltioLive Smart Client or used with a Datapool to publish updates.

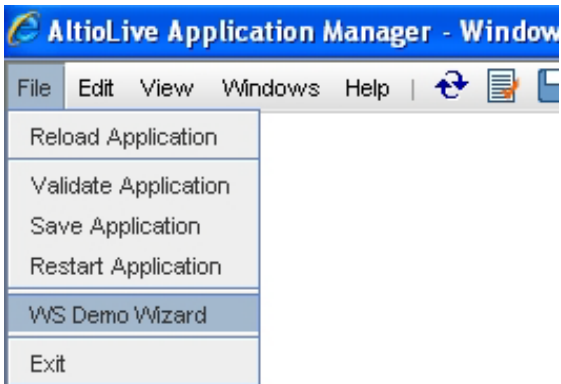
Using the Web Services Demo Wizard

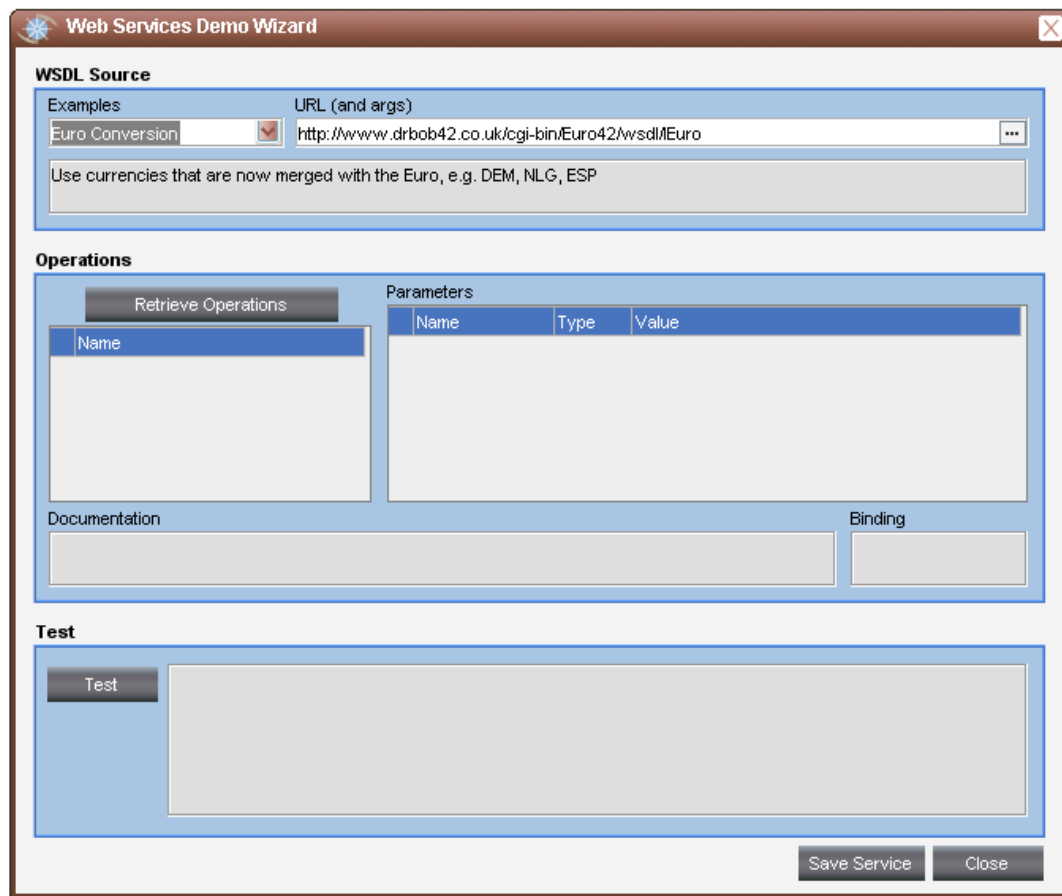
The Web Services Demo Wizard provides an ideal starting point for using web services with AltioLive. The Demo Wizard allows you to explore pre-selected web services and become familiar with the configuration options available with the AltioLive Application Manager.

Accessing the Web Services Demo Wizard

1. To access the Web Services Demo Wizard:
2. Right-click a project and select the **Application Manager** tool in the context menu
3. Click the **File** menu and select the **WS Demo Wizard** option:

The Web Services Demo Wizard window is displayed:





Web Services Demo Wizard

WSDL Source

Examples: Euro Conversion | URL (and args): http://www.drbob42.co.uk/cgi-bin/Euro42/wsdlMEuro

Use currencies that are now merged with the Euro, e.g. DEM, NLG, ESP

Operations

Retrieve Operations

Name

Parameters

Name	Type	Value
------	------	-------

Documentation

Binding

Test

Test

Save Service | Close

There are 4 regions to the Web Services Demo Wizard window:

- **WSDL Source:** this allows you to select a web service from a list. When selected, the location of the web service is displayed along with a brief description (note that this is not obtained from the WSDL source). URLs to other WSDL files may be entered at the URL (and args) field.
- **Operations:** this allows you to view the operations that are available as web services along with supporting documentation available as part of a WSDL definition. Any parameters required by an operation are displayed; including the name, type, and value. Parameter values may be entered for use in testing a service
- **Test:** The Test region displays values or XML returned from a web service operation
- **Save Service:** Once you are satisfied with the web service, you may save it to the WS application where a test window will be generated. This can be modified and liked to any AltioLive application that needs to use it.

Use the Web Services Demo Wizard to test a web service that uses RPC binding

With RPC binding, each parameter sent to an operation is sent as a method parameter. The **Roman Numeral Conversion** example uses RPC:

1. At the **WSDL Source** region, make sure the **Roman Numeral Conversion** example is selected.
2. At the **Operations** region,

3. Click the **Retrieve Operations** button to access the remote WSDL file.
Please note: the time taken for this operation varies and depends on network conditions and the performance of the remote server.

The operations available at the web service are listed. As each operation is selected, the parameters required are shown in the **Parameters** table:

- the **Name** of the parameter,
 - the parameter **Type**: for RPC it can be: Int, Float, Double, Long, Boolean, String and Literal.
 - the **Value** field to enter the value in.
4. Enter a test value for a selected operation. As an example, select the **RomanToInt** operation and enter the value:
III
 5. Click the **Test** button, in the **Test** area to test the web service operation:

Operations

Retrieve Operations

Name
IntToRoman
RomanToInt

Parameters

Name	Type	Value
Int	int	5

Documentation

No documentation supplied

Binding

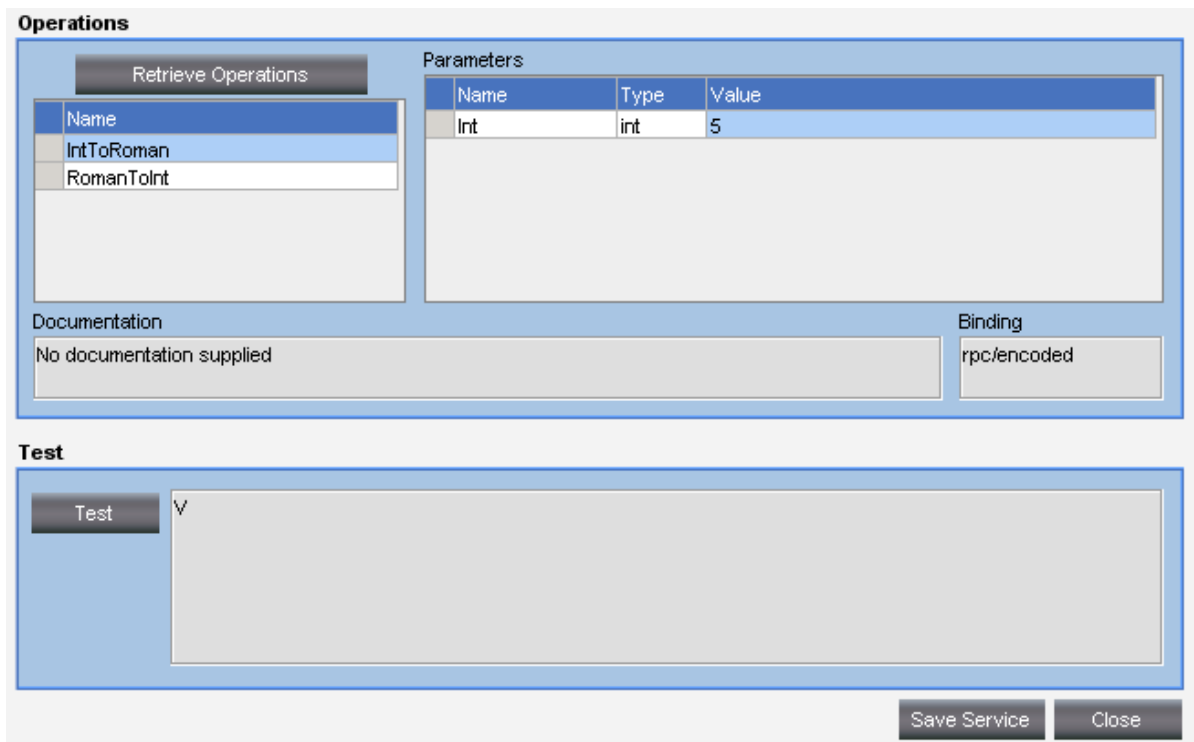
rpc/encoded

Test

Test

V

Save Service Close



There may be a delay while the remote service is called.

Saving a Service using the Web Services Demo Wizard

A web service operation selected at the Web Services Demo Wizard may be saved to the WS application. This allows you to explore and understand how a web service is invoked from a View.

To save the Service to the WS application:

1. Select the service you want to save, (in this example **RomanToInt**).
2. Click the **Save Service** button.
3. Click **OK** when a message displays to advise that a view will be created.

The **Save Service** creates a default view for the Web Service application and an AltioLive SOAP Service Function. These are described in **The Web Service Application** topic of this document.

Modifying the Web Services Demo Wizard

If you want to change the Web Services listed in the Web Services Demo Wizard, this may be achieved by editing the file **wsdemo_db.xml** at the **/tools/admin/dbdata** directory. Simply adjust the **NM**, **WSDL**, and **HINT** attributes for the **WS_DEMO_SERVICE** element (or add new elements as required); then restart the Server.

Using the Web Service Application

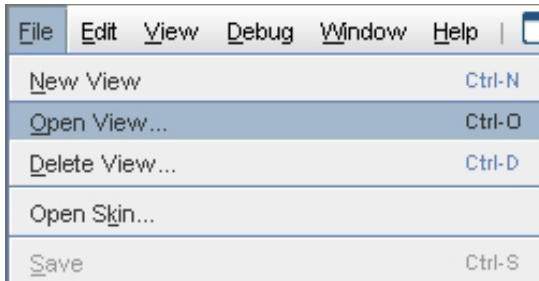
The Web Service application contains views and AltioLive Service Functions generated automatically from the Web Services Demo Wizard. It allows you to explore and understand how a web service is invoked from a View

If you have not already done so, access the Application Manager/ Web Services Demo Wizard and save the 'From Euro' operation available at the 'Euro Conversion' example.

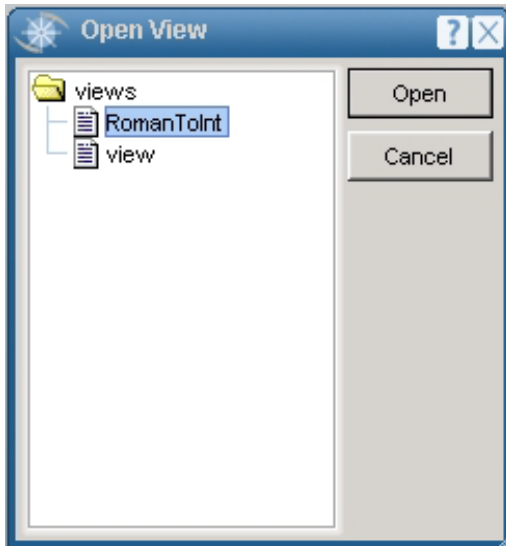
Open the WS application

To explore the view created from the Web Services Demo Wizard:


1. Access the Designer for the **WS** application.
2. At the Designer, click the **File** menu and select **Open View**.

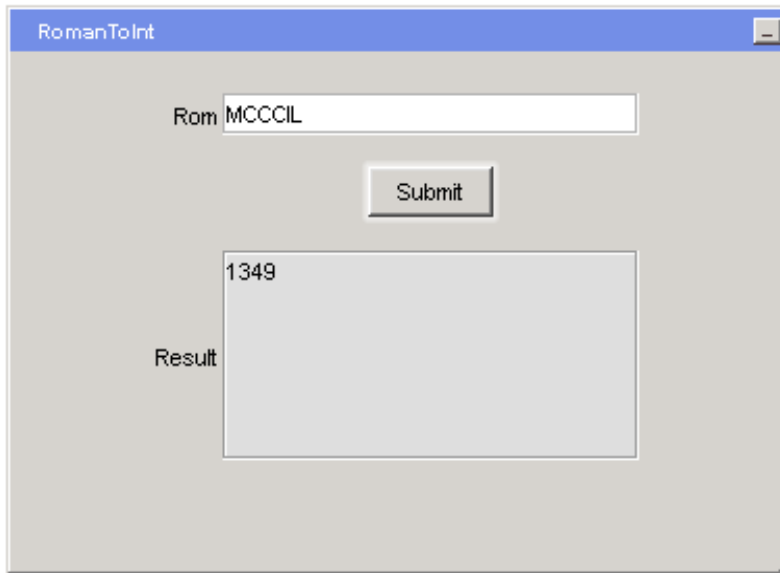


3. Select the **RomanToInt** View and click the **Open** button.



Test the WS application

1. Test the view by clicking the Run button  at the Designer toolbar.



Notice that the Web Services Demo Wizard has created a test window for the web service.


2. In the **Rom** field, enter a roman number (for example: MCIL).
3. Click **Submit** button.
4. The conversion displays.

5. Exit the test by clicking the **Return to Designer Mode** button:

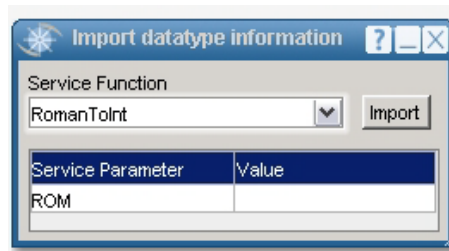


View the Datatypes associated with the web service

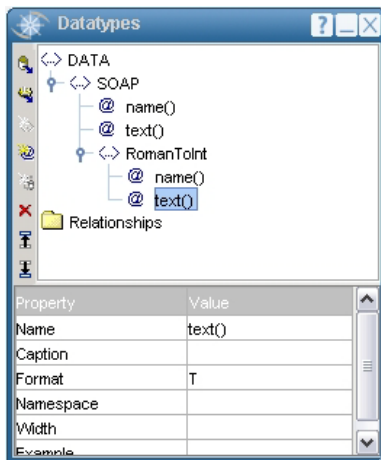
Once the web service has been tested successfully and has returned data, you may view the Datatypes associated with the Operation. Datatypes show a representation of the data used in the application and enable you to associate particular types of data with controls.

1. Access the Datatypes window by clicking the **View | Datatypes** menu items.
2. Click the **Import Datatypes from a service function** .
3. At the **Import Datatypes** pop-up window, select the **RomanToInt** in the ServiceFunction drop-down menu.

- Click on the **Import** button.



- Expand the XML hierarchy and click the **RomanToInt** node to view the attributes returned from the web service:





The return from the web service are contained in element:
/SOAP/RomanToInt, attribute @text().

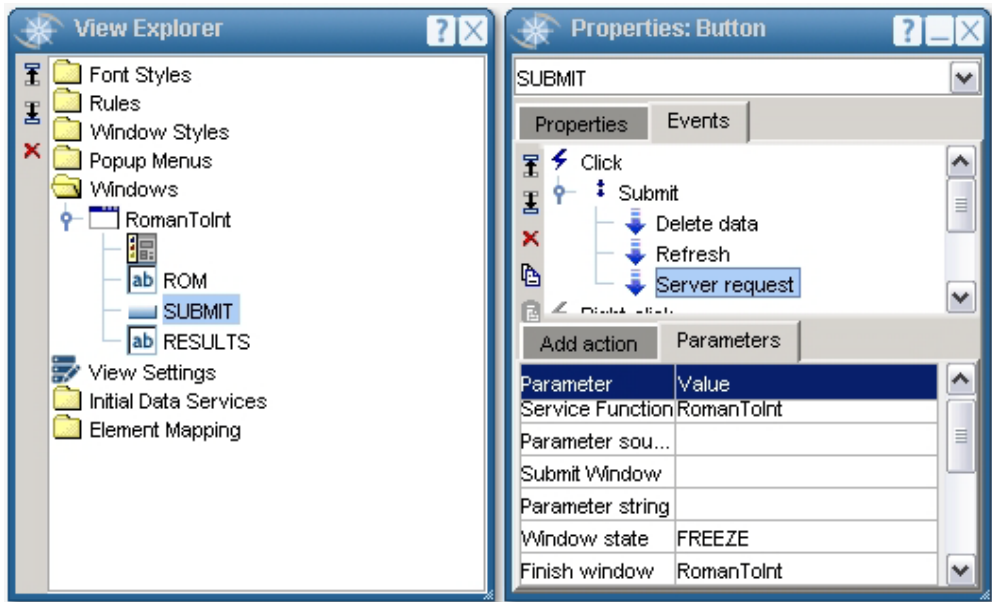
- Close the **Datatypes** window.

Explore the View

Use the Designer to explore how the **RomanToInt** window works with the web service. In particular, observe how a Server Command action is used to access the web service.

1. Open up the **View Explorer** button by clicking the **View | Explorer** menu item or by clicking on the View Explorer icon:  .
2. Ensure the **Properties** window is visible by clicking on the **View | Properties** menu item or by clicking on the View Properties icon:  .
3. At the View Explorer, Expand the **Windows** folder, the **RomanToInt** window, and select the **SUBMIT** button. Observe that the Properties window displays the properties for the SUBMIT button.

- From the **Properties** window, Select the **Events** tab, expand the **Click** event so that the three actions associated with clicking the button are shown:




- Select the **Parameters** tab.
Here is a brief overview of the available actions:

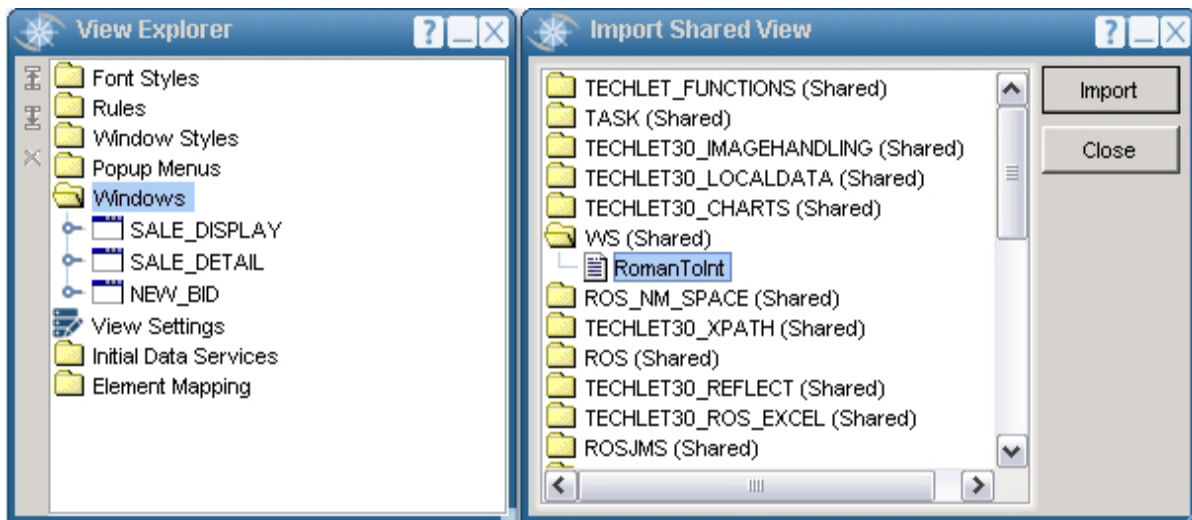
- **Delete:** this action removes data at the Client. The element removed is: **/SOAP/RomanToInit** (i.e. the data returned from the AltioLive Service Function)
 - **Refresh:** The Results control Data Source is: **/SOAP/RomanToInit**. As this has been removed, the control is effectively cleared.
 - **Server request:** The calls the **RomanToInit** Service Function (defined in the Application Manager). The window state is set to **FREEZE** until the action is complete; the Finish Window is a name of a window to display when the action is complete – in this case the **RomanToInit** window is refreshed (as it is already displayed).
6. Once you have completed exploring the View, close the Designer.

Using the WS Application

Windows created in the WS application are available to any other AltioLive application as **Linked Windows**. To link a window:

1. Open another application, such as **ROS**, with the Designer.
2. Display the **View Explorer** window by clicking on the **View explorer** icon:  .
3. Right-click on the **Windows** folder and select the **Link View...** option in the context menu.

4. The **Import Shared View** window will open:



5. Open the **WS (Shared)** folder and import the **RomanToInt** View. This will link the windows available at that View. **Please note:** window properties for the linked window may be adjusted; for example the Skin may be changed to match that of application loaded in the Designer.

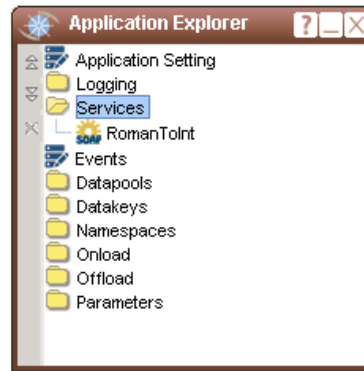
The SOAP Service Function

The View created by the Web Services Demo Wizard uses the **RomanToInt** Service Function, which was also created by the wizard.

It is unlikely that you would create a SOAP Service ‘from scratch’, as the Application Manager can assist you with configuration though the **Find SOAP service** feature (which is described later in this document).

To investigate the AltioLive Service Function created by the Web Services Demo Wizard:

1. Access the Application Manager for application **WS**.
2. At the **Application Explorer** window, expand the **Services** folder and double-click on the **RomanToInt** service to open it:



Configuration of a SOAP Service occurs automatically from the remote WSDL service definition. Brief descriptions of the service parameters follow.

The Request Tab

The Request tab configures the parameters required for a remote web service operation:

The screenshot shows the 'Request' tab of the 'SOAP Service RomanTolnt' configuration window. The window title is 'SOAP Service RomanTolnt' and it features a 'Simple Object Access Protocol' header with an envelope icon. The 'Service name' is 'RomanTolnt'. The 'Location' is 'http://www.ebob42.com/cgi-bin/Romulan.exe/soap/Roman'. The 'SOAPAction header' is 'urn:Roman-IRoman#RomanTolnt'. The 'Server side cookie scope' is set to 'Send client cookie'. The 'Request headers' table is empty. The 'Message body' section shows a 'Binding style' of 'RPC', a 'Namespace' of 'urn:Roman-IRoman', and an 'Operation' of 'RomanTolnt'. The 'Parts' table contains one entry: 'Rom' with type 'string' and value '{\$client.ROM}'. The window includes 'Add' and 'Delete' buttons for both headers and parts, and 'Offline Configuration...', 'OK', and 'Cancel' buttons at the bottom.

Service name: RomanTolnt

Request | Response | Acknowledgement | Test

Location: http://www.ebob42.com/cgi-bin/Romulan.exe/soap/Roman

SOAPAction header: urn:Roman-IRoman#RomanTolnt

Server side cookie scope: Send client cookie

Request headers

Must understand	Actor	Value
-----------------	-------	-------

Add Delete

Message body

Binding style: RPC DOC

Namespace: urn:Roman-IRoman

Operation: RomanTolnt

Parts

Name	Type	Value
Rom	string	`\${client.ROM}`

Add Delete

Offline Configuration... OK Cancel

- **Service Name:** The name by which the AltioLive Service Function is referenced.
- **Location:** The URL of the server hosting the webservice.
- **SOAP Action Header:** Sometimes used by the host server to direct the SOAP webservice header to the implementation. If omitted, defaults to an empty string.
- **Server side cookie scope:** When invoking a HTTP service that returns Set-Cookie or Set-Cookie2 header Presentation Server saves the Cookie information so that when subsequent calls to services of same domain and path, this cookie information can be sent back. From the Server side cookie scope drop-down menu you can choose how the Presentation Server stores the cookie information and how cookie information will be shared among services.
- **Send client cookie:** You can choose whether you want the Presentation Server to pass on cookie(s) from client browser to backend service.

Request headers section

A SOAP message can have zero or more message headers, which can be used to control how the message is routed and processed. Each header is configured via the following information:

- **Must Understand:** Boolean indicating whether the target actor must understand this header or not. This attribute is optional, and default is set to false.
- **Actor:** The URI location of a target SOAP actor. The purpose of an actor is to redirect the SOAP packet to another location on its route to the webservice server. This attribute is optional.

- **Value:** String representing well-formed XML. This is a required part of a SOAP header.

Message Body section (This defines the structure of the request):

- **Binding Style: RPC or DOCUMENT.** One of these should be set according to the request structure expected by the SOAP web service.
 - **RPC** stands for Remote Procedure Call. As the name suggests, this means that a method is being invoked, and that each 'Part' being sent is a method parameter.
 - **DOCUMENT** style invocation means that each 'Part' being sent is an XML fragment.
- **Namespace: (RPC binding only):** The namespace of the request element.
- **Operation: (RPC binding only):** The name of the method being invoked.

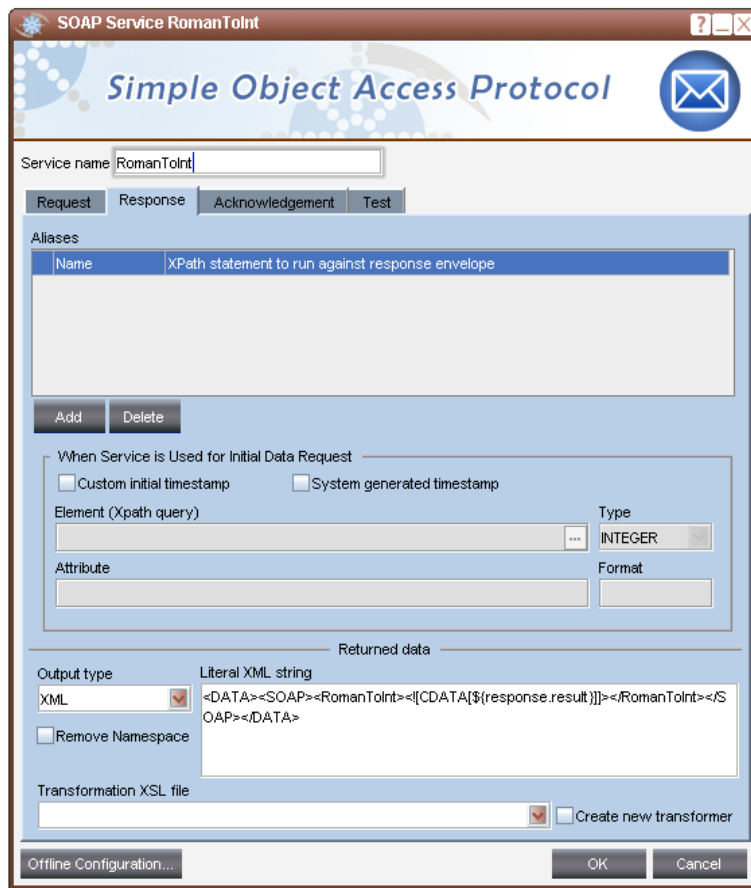
- **Parts:** A block of information to be sent to the web service. There can be zero or more parts.
 - **Name:** The name of the part. This is used by **RPC** binding, with a type other than LITERAL. Otherwise it is discarded.
 - **Value:** The value to use. It will normally contain references to the client data. If the type is LITERAL, then this field must evaluate to well-formed XML.
 - **Type:** The type of the value that is being sent, and the allowed value depends on the binding being used.
 - **RPC:** Possible types include INT, FLOAT, DOUBLE, LONG, BOOLEAN, STRING, BYTE, DATETIME, BIGDECIMAL, BIGINTEGER, SHORT.
 - **DOCUMENT:** Only a LITERAL type may be used.

The Response Tab

The Response tab configures how returned values from a remote web service operation are processed. If XML is to be returned from the service call, then the **Literal XML string** edit box contains the XML to be returned. The literal XML string can contain zero or more references to data from the SOAP response; these references take the form **`${response.result}`**, **`${response.body}`** or **`${response.alias.XP1}`** etc. See the section on SOAP response processing below.

Aliases

- **Name:** The name of this alias
- **XPath statement to run against response envelope:** XPath to obtain the data represented by this alias. (See The following ‘SOAP Response Processing’ topic)



When Services is Used for Initial Data Request

- **Custom Initial Timestamp:** This only applies to XML Service Functions. Check this box if you want the Timestamp to be extracted from a customized location. If this option is selected you can set the Element and Attribute for locating timestamp information. By default Timestamp are extracted from a Timestamp Attribute on the root element of the XML returned by the Service.
- **System Generated Timestamp:** Check this box if you want the Presentation Server to use system clock for the Timestamp value.
- **Element (XPath query):** an XPath statement pointing to the element containing Timestamp information. By default the Presentation Server continues to find Timestamp information from root element. If more than one element matches, only the first element is used.
- **Attribute:** attribute name containing the Timestamp info or **TEXT()** to indicate that the element's text contains the timestamp. By default the Presentation Server finds Timestamp information from Timestamp default attribute.
- **Type:** whether the Timestamp field is to be read as a Date/Time or as an Integer. By default the Type is set to Integer.
- **Format:** the Format to be used to parse the Timestamp from the specified Element and Attribute.
 - For a Date/Time this should be a format recognized by **java.text.SimpleDateFormat**.
 - For an Integer this should be a format recognized by **java.text.DecimalFormat**.

By default if the **Type** is set to **Integer** then the Timestamp Attribute is read as an Integer.

Returned Data

- **Output Type: XML or EMPTY.** If this is set to **XML**, then **Literal XML String** is evaluated to construct the XML to return.
- **Literal XML String:** Defaults to **\${response.body}**. This constructs a well-formed block of XML, which represents the returned results. It can contain a generic string with property references.
- **Transformation XSL Stylesheet:** If the return type is XML, then the resulting XML of **Literal XML String** is passed through the specified stylesheet.

SOAP Response Processing

There are three properties available for SOAP response processing you can enter in the **Literal XML string** field:

- `${response.body}`
- `${response.alias.xxx}` (where xxx is the name of the alias)
- `${response.result}`.

`${response.body}`

The `${response.body}` property is always available. It maps onto the first child element of the response body element.

Example: Body

Using `${response.body}` in the FromEuro example, the Literal XML string could be:

```
<DATA><SOAP><RomanToInt>${response.body}</RomanToInt></SOAP></DATA>
```

Test Values of **LXV** would return:

```
<DATA>
  <SOAP>
    <RomanToInt>
      <a:RomanToIntResponse xmlns:a='urn:Roman-IRoman'>
        <return b:type='xsd:int' xmlns:b='http://www.w3.org/2001/XMLSchema-instance'>65</return>
      </a:RomanToIntResponse>
    </RomanToInt>
  </SOAP>
</DATA>
```

`${response.alias.xxx}`

The **`${response.alias.xxx}`** property is an XPath statement referred to by name. The root of the XPath statement is the envelope element of the SOAP response, so any information contained by the SOAP response is available via this mechanism.

Example: Alias

Using `${response.alias.xxx}` in the RomanToInt example, the Literal XML string could be:

```
<DATA><SOAP><RomanToInt>${response.alias.XXX}</RomanToInt></SOAP></DATA>
```

With the Aliases configured as:

Name	XPath statement run against response envelope
XXX	//return

Test Values of **LXV** would return:

```
<DATA>
  <SOAP>
    <RomanToInt>
      <return a:type='xsd:int' xmlns:a='http://www.w3.org/2001/XMLSchema-instance'>65</return>
    </RomanToInt>
  </SOAP>
</DATA>
```

`${response.result}`

Finally, the `${response.result}`. Here the result is configured according to its expected result type (i.e. as described in the attribute 'a:type'). As the result could contain invalid XML, `${response.result}` can be wrapped in CDATA which prevents the string from being parsed.

Example: Result

Using `${response.result}` in the RomanToInt example, the Literal XML string could be:

```
<DATA><SOAP><RomanToInt><![CDATA[ ${response.result} ]]></RomanToInt></SOAP></DATA>
```

Test Values of **LVX** would return:

```
<DATA>  
  <SOAP>  
    <RomanToInt><![CDATA[ 65 ]]></RomanToInt>  
  </SOAP>  
</DATA>
```

The Acknowledgement Tab

The Acknowledgement tab configures messages that can be displayed on success or failure of the remote web service Operation.

On Success:

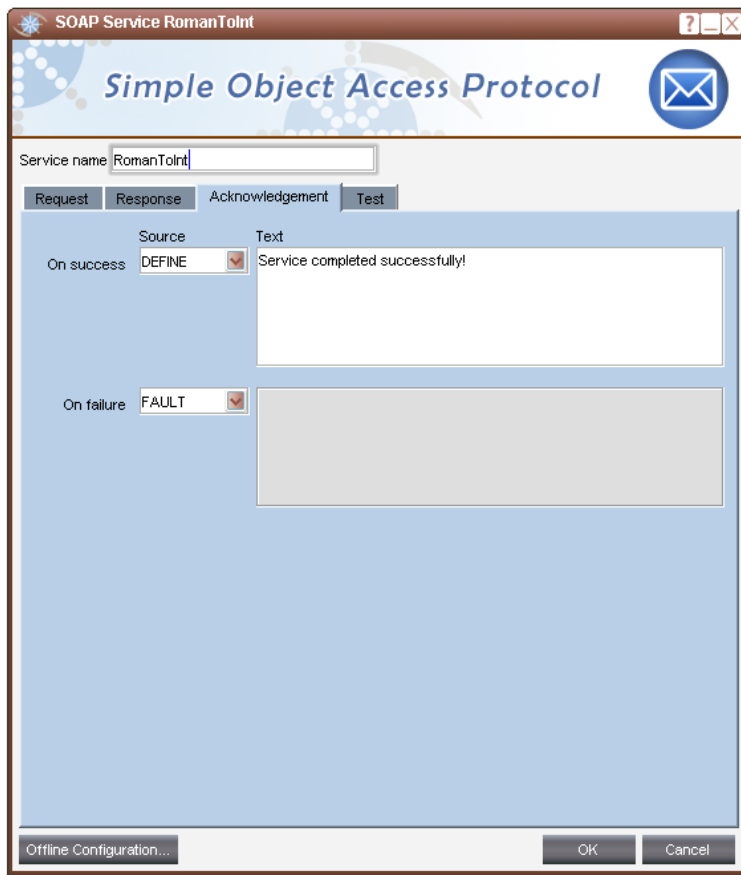
Source: How to create the success message to send to the client, if any. Possible values are NONE and DEFINE.

Text: If Source is set to DEFINE, then this is the message that is sent back.

On Failure:

Source: How to create the fail message to send to the client, if any. Possible values are NONE, DEFINE and FAULT. If FAULT is specified, then the message returned is that from the SOAP response fault element.

Text: If Source is set to DEFINE, then this is the message that is sent back.



The Test Tab

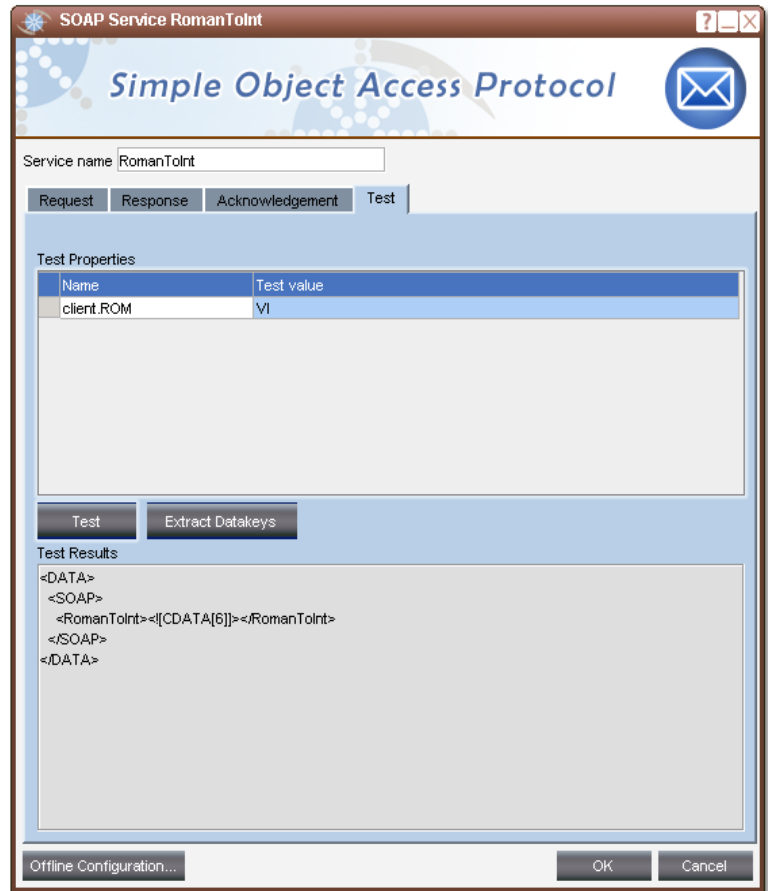
The Test tab allows parameter values to be sent to a remote web service Operation; any return values are displayed.

Test Properties (Displays the parameters available for this SOAP Service)

- **Name:** The name of the parameter being used.
- **Test Value:** A value to be entered by the user for the purpose of testing this SOAP service.

Test Results:

- The results of executing a call to the SOAP web service using the parameters in the Test Properties table. This is the XML that would be returned to the client.



Find SOAP Service

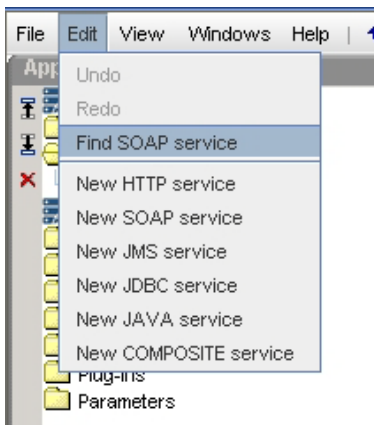
The Web Services Demo Wizard is an ideal starting point for understanding how web services can be used with AltioLive. Once you have seen the examples, you will be ready to find other web services available and integrate them into any AltioLive application. The Find SOAP Service available in the Application Manager makes adding a SOAP service from a WSDL file a simple task.

To use the Find SOAP Service you will require a URL to a WSDL file. You can find many sample services at <http://www.xmethods.com>.

To demonstrate Find SOAP Service, you can use one of the known URLs available at the Web Services Demo Wizard. For example: Country Webservice at <http://www.websvicex.net/country.asmx?wsdl> is an example where document binding styles are used.

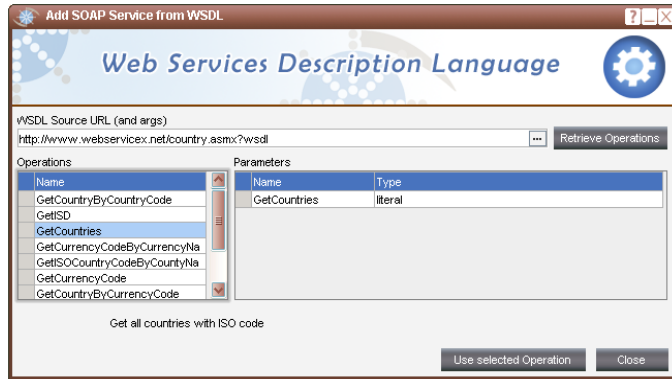
Find and add an operation from a web service

1. Access the Application Manager for application **WS**, or another application to which you want to add a web service operation.
2. At the **Application Explorer** window, select the **Services** folder, and then choose the **Find SOAP** service option from the **Edit** menu:



The Add SOAP Service from WSDL window opens.

3. Enter the URL to the WSDL file at the **WSDL Source URL (and args)** field and press the **Retrieve Operations** button:



4. The Operations are listed; Click an operation to view **Documentation**, the **Binding Style**, and **Parameters** required.
5. Click the **Use selected Operation** button to configure automatically an AltioLive Service Function for the selected Operation.
The newly created AltioLive Service Function will display.

The screenshot shows a window titled "SOAP Service GetCountries1" with a "Simple Object Access Protocol" header. The "Service name" is "GetCountries1". The "Request" tab is selected. The "Location" is "http://www.webserviceX.net/country.asmx". The "SOAPAction header" is "http://www.webserviceX.NET/GetCountries". The "Server side cookie scope" is set to "All" and "Send client cookie" is checked. The "Request headers" table is empty. The "Binding style" is "DOC". The "Parts" table has one entry:

Name	Type	Value
GetCountries	literal	<tt:GetCountries xmlns:tt="http://www.webserviceX.NET">\${client.GE...

Please note: the XML fragment to be sent to the Operation. Here the value contained in the client control `${client.COUNTRYCODE}` will be passed to the web service.

6. Select the **Test** tab and enter a value for the parameter.
7. Click the **OK** button, the SOAP Service Function is now configured.

Troubleshooting

General

- **Can't log in to Designer / Cannot run an application:**

Check that you have the Designer or Demonstration login options visible on the console when you enter your login details.

- **Browser window not showing new changes / Browser window does not load new tool:**

This happens because the browser will cache data if it has not been shut down. Ensure you have closed all your browser windows. Stop and restart your application server. Then restart your browser.

- **'You are no longer logged on' error message:**

This error message occurs when you attempt to save your work after not saving it within the servlet engine's timeout period. Check the servlet engine documentation for details on changing this.

In the meantime, you will have to close all your browser windows and then stop and restart your application server. When you restart, you will return to the point at which you last saved you work.

Other sources of help

- **Online help**

More troubleshooting help is available in the online help, which can be accessed by clicking the question mark icon on any Altio window, or from the Altio Console or the Windows start menu.

- **Developer Center**

<http://www.altio.com/developer/> - a community-gathering place where you'll find the technical resources, knowledge, and peer advice that will assure your success in building and deploying Altio applications.

- <http://www.altio.com/forums/> - a wealth of questions and answers from members of the Altio community.

Further Information / References

Other manuals that should be read in conjunction with this one are:

- Getting Started with the Altio Smart Client
- Getting Started with the Synchronization Engine Admin Tool
- Getting Started with the Application Manager
- Getting Started with the AltioLive Designer
- Getting Started with the AltioDB
- Advanced GUI Design Techniques
- Markets Tutorial

Detailed information on the AltioLive API can be found in the online help (click the link from the Altio Console).

Document Information

Integra SP – Altio

Telephone: +44 (0) 20 8528 1045 Internet: www.altio.com

Copyright © 2010 Integra SP

Copyright in this document is vested in Integra SP. The contents of the document (wholly or in part) must not be reproduced, distributed, used or disclosed without the prior written permission of Integra SP.

Integra recognizes the trademarks or registered trademarks of any third party product or company name referenced in this document at the time of its publication.