Chapter 20.

August 2008

Welcome to the August 2008 edition of IBM InfoSphere Information Server Developer’s Notebook. This month we answer the question;

> Part II of: What are Web services, and how and why would I use them with DataStage and/or QualityStage?
>
> *(Which when expanded, also becomes; What are loose coupling, SOA, SCM, UDDI, SOAP, model-view-controller, and named frameworks.)*

Part I of this article arrived in July 2007, where all of the relevant terms and ideas were explained. In July 2007 we *published* and tested a Web service that was written in DataStage. In this article (Part II), we build upon earlier work, and *consume* Web services programmatically with DataStage and/or QualityStage. When you publish Web services, you use certain DataStage/QualityStage stages, and when you consume Web services you use other DataStage/QualityStage stages.

Software versions

All of these solutions were *developed and tested* on (IBM) InfoSphere Information Server (IIS) version 8.1, using the Microsoft Windows XP/SP2 platform to support IIS client programs, and a RedHat Linux Advanced Server 4 (RHEL 4) FixPak U6 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server components.

IBM InfoSphere Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM InfoSphere Information Server product contains the following major components;

> WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM InfoSphere Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

20.1 Terms and core concepts

As stated above, this edition of (IBM) InfoSphere Information Server Developer’s Notebook (IISDN) is a continuation from July 2007. In the July 2007 article, we explained the following terms and core concepts;

> Design patterns, design principles, model-view-controller (MVC), frameworks, coupling (loose coupling, decreased coupling, tight coupling, coupling by time, format or location), service requestor and service provider, Web services, Web service protocols (HTTP/SOAP, JMS, others), WSDL files, UDDI, LDAP, Java/J2EE EAR files, SOA, CORBA, SCA, SCM, and BPEL.

Also in the July 2007 article, we detailed how to complete the following tasks;

– How to create a DataStage/QualityStage Job that *provides* Web service.

  - A DataStage/Quality Job that *provides* a Web service uses different stages than a DataStage/QualityStage Job that *consumes* Web services.

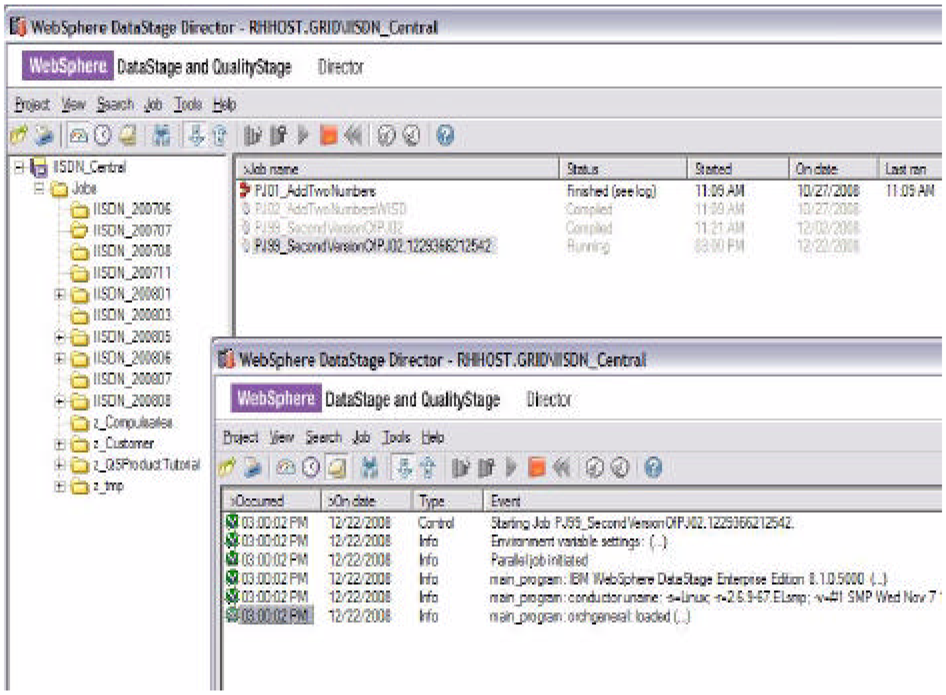
    DataStage/QualityStage Jobs that provide Web service use the WISD Input and/or WISD Output stages. DataStage/QualityStage Jobs that consume a Web service use the Web Services Transformer, and/or the Web Services Client stages.

    All of the stages listed above are found in the Real Time drawer of the Palette view, inside the DataStage/QualityStage Designer program.

  - DataStage/QualityStage Jobs with a WISD Input, or a WISD Input and WISD Output stages, are (in InfoSphere Information Server parlance) referred to as; Topology III Jobs. Topology III Jobs are (Web) service enabled Jobs that are *always on*. Topology III Jobs are deployed, and are hosted within a Java/J2EE compliant application server, normally IBM WebSphere Application Server (WAS).

    The DataStage/QualityStage Director program may be used to monitor Topology III Jobs, as shown in Figure20-1 below. Topology III Jobs are also displayed with a different (real time) icon in the Job list.

    DataStage/QualityStage Jobs that don’t have a WISD Input stage, but do have a Web Services Transformer stage or Web Services Client stage, are (in InfoSphere Information Server parlance) referred to as; Topology I + II Jobs. Topology I + II Jobs are not always on, and are loaded/run on demand.



Job name	Status	Started	On date	Last ran
PJ01_AddTwoNumbers	Finished (see log)	11:05 AM	10/27/2008	11:05 AM
PJ02_AddTwoNumbersWISD	Compiled	11:09 AM	10/27/2008	
PJ99_SecondVersionOfPJ02	Compiled	11:21 AM	12/02/2008	
PJ99_SecondVersionOfPJ02.1229366212542	Running	03:00 PM	12/22/2008	

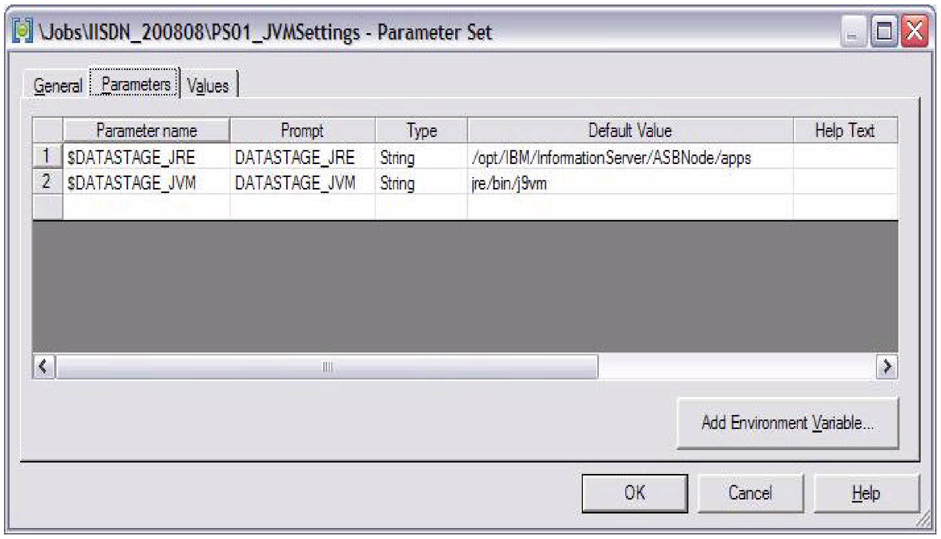
Occured	On date	Type	Event
03:00:02 PM	12/22/2008	Control	Starting Job PJ99_SecondVersionOfPJ02.1229366212542.
03:00:02 PM	12/22/2008	Info	Environment variable settings: (...)
03:00:02 PM	12/22/2008	Info	Parallel job initiated
03:00:02 PM	12/22/2008	Info	main_program: IBM WebSphere DataStage Enterprise Edition 8.1.0.5000 (...)
03:00:02 PM	12/22/2008	Info	main_program: conductor uname: -s=Linux; -r=2.6.9-67.ELsmp; -v=#1 SMP Wed Nov 7
03:00:02 PM	12/22/2008	Info	main_program: orchgeneral: loaded (...)

Figure 20-1 DataStage/QualityStage Director program displaying Topology III job.

- Because Topology III Jobs are deployed and run inside a Java/J2EE compliant application server, Topology III Jobs have everything they need with regards to a Java Virtual Machine (JVM), the Java runtime environment.

  Topology I + II Jobs are loaded at time of execution, and need two additional Environment Variables defined so that they may locate their JVM at run time. These Environment Variables are displayed in Figure20-2, below.

  The settings displayed in Figure 20-2 are accurate for a default location of a Linux installation of InfoSphere Information Server. DATASTAGE_JRE requires an absolute pathname, whereas DATASTAGE_JVM is a relative pathname to DATASTAGE_JRE.


\Jobs\IISDN_200808\PS01_JVMSettings - Parameter Set
General
Parameters
Values
Add Environment Variable...
OK
Cancel
Help

	Parameter name	Prompt	Type	Default Value	Help Text
1	$DATASTAGE_JRE	DATASTAGE_JRE	String	/opt/IBM/InformationServer/ASBNode/apps	
2	$DATASTAGE_JVM	DATASTAGE_JVM	String	jre/bin/j9vm	

Figure 20-2 Environment Variables required to operate Topology I + II Jobs.

– How to enable a DataStage/QualityStage Job to support Web services.

  Specifically this is Page-11, Steps,3.a-c of the July 2007 edition of this document. Its not enough to have the Web service enabling stages, one must also set properties within the Job to support same.

  This activity must be performed once for every new DataStage/QualityStage Job that provides Web services. This activity is performed inside the DataStage/QualityStage Designer program.

– How to create an Information Services Connection.

  Specifically this is Page 12, Steps,5-6 of the July 2007 edition of this document.

  An Information Services Connection is the resource that allows connectivity between the Java/J2EE compliant application server, and the DataStage/QualityStage Engine proper. While administrative and internal connections of this type already exist, this connection is specific to the collection of Web services we are about to deploy, host, and manage.

  This activity needs to be performed only once, unless additional user ids or security conditions are required. This activity is performed within the InfoSphere Information Server (Administrative) Console program.

– How to create a WISD Project and Application.

  Specifically this is Page 14, Steps,7-10 of the July 2007 edition of this document.

A WISD Project is a logical term; it serves as a grouping of WISD Applications. In the context of InfoSphere Information Server, a WISD Application equals one DataStage/QualityStage Topology III type job.

A WISD Project only needs to be created one time, as it can contain numerous WISD Applications (numerous DataStage/QualityStage Jobs).

A WISD Application needs to be created for every DataStage/QualityStage Topology III type Job you wish to provide Web service.

These activities are performed in the InfoSphere Information Server (Administrative) Console program.

– How to deploy a WISD (Project) Application.

  Specifically this is Page 21, Step,12 of the July 2007 edition of this document.

  Deployment is the actual event that places the given Web service inside the Java/J2EE compliant application, and makes it ready for use. One can deploy, un-deploy, and re-deploy (WISD Applications).

  This activity needs to be performed only once, unless one has made changes to the associated DataStage/QualityStage Job, or the accompanying WISD Application. This activity is performed within the InfoSphere Information Server Console program.

Note: At this point, our Web service is deployed and ready. (Its functioning.) The steps that follow are diagnostic, or used for further discovery.

– How to test and/or reverse engineer a deployed Web service.

  Specifically this is Page 21, Steps, 13-14 of the July 2007 edition of this document.

  Each deployed/active Web service is associated with a WSDL file. A WSDL file is located by URL (a Web address, local or remote), and tells you pretty much everything you need to know about a given Web service.

  First, we use the InfoSphere Information Server (Administrative) Web Console to find our Web service's WSDL file, then we use Rational Data Architect as our WSDL, and/or Web services explorer.

  If we were consuming/testing/using a Web service other than our own, we'd use another means to capture the WSDL file. Because Web services are industry standard, any WSDL, Web services explorer would do.

Note: If you are going to perform the activities contained in this document (creating and deploying additional Web services, or consuming Web services), you must first complete *all* of the activities detailed in the July 2007 edition of (IBM) InfoSphere Information Server Developer's Notebook.

20.2 Using the DS/QS Web Services Transformer stage

Figure20-3 displays the Real Time drawer of the DataStage Designer program, and the DataStage/QualityStage Job we create in this section of this document; a Job to demonstrate configuration and use of the DataStage/QualityStage Web Services Transformer stage.

A DataStage/QualityStage Web Services Transformer stage functions just like our everyday (Parallel) Transformer stage; data passes through this stage and the (Transformer) generally does stuff to said data, whatever that stuff may be.

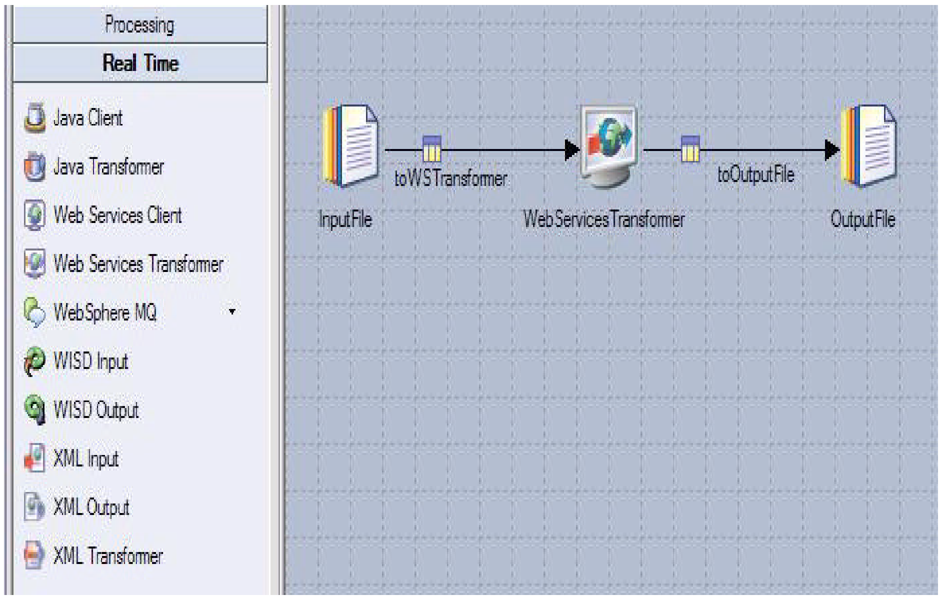


Figure 20-3 DS/QS Job with Web Services Transformer stage.

Continuing a discussion of the Real Time drawer of the DataStage/QualityStage Designer program, the following is offered;

– The XML Input, XML Output, and XML Transformer stages are detailed in the June 2007 edition of this document, (IBM) InfoSphere Information Server Developer's Notebook (IISDN).

– The WISD Input and WISD Output stages are detailed in the July 2007 edition of IISDN.

– The Java Transformer stage is detailed in the July 2008 edition of IISDN.

  We *discussed* the Java Client stage, but did not implement anything; a Java Client stage acts as the end points (data input or data output) of a DataStage/QualityStage Job, and hence requires a given amount of Java code to (minimally: read and/or write files or JDBC data sources). That sub-topic becomes too much of a Java primer/tutorial for the purposes of this document.

– This edition of IISDN details the Web Services Transformer stage (here in this section), and the Web Services Client stage (in the next section of this document.

– An upcoming edition of IISDN will give focus to all things MQ/Series and messaging related.

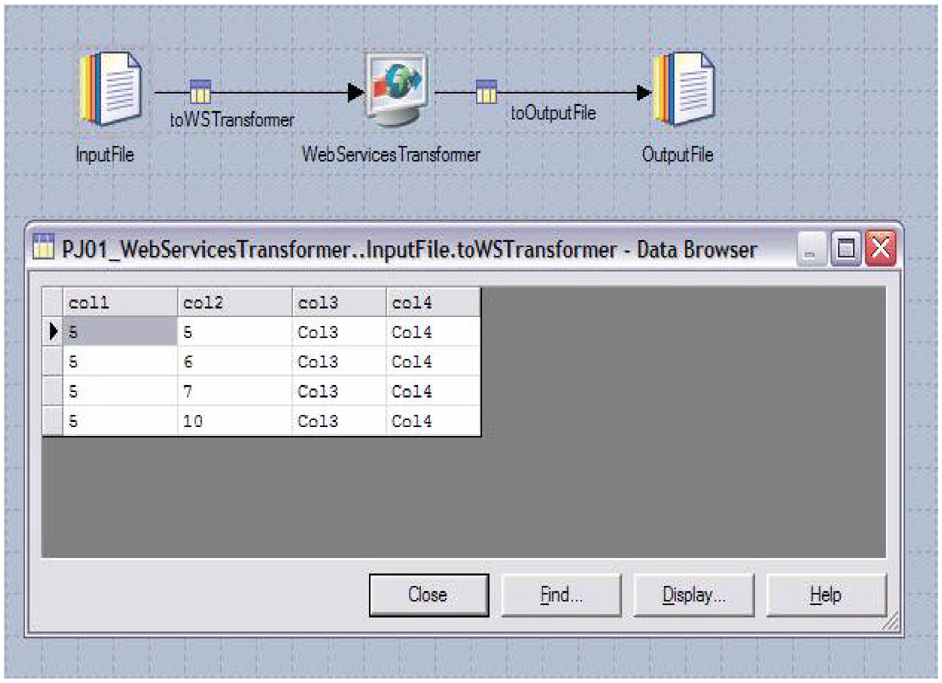
Create DS/QS Job to use the Web Services Transformer stage

1. Using your favorite editor, create an ASCII text (data) file to be used as an input data source. This text file should have 4 columns; 2 with integer values, two with character values.

   Figure20-4 displays a sample input data file we use throughout this first example. We created sample data with 4 columns and 4 rows.

2. Launch the DataStage/QualityStage Designer component of IBM InfoSphere Information Server.

   As mentioned above, the steps we are about to complete have a dependency; you must first complete all steps in the July 2007 edition of this document. Otherwise, you will be trying to call a Web service that does not exist.



col1	col2	col3	col4
5	5	Col3	Col4
5	6	Col3	Col4
5	7	Col3	Col4
5	10	Col3	Col4

Figure 20-4 Input Data source we use throughout this Job.

3. Create table definitions for both the input and output columns of the Web service we will be invoking.

   Normally we don’t care too much about table definitions in these simple exercises; however, in this case we specifically want/need the XPath expressions for the input and output columns to this Web service, and we don’t want to write them manually.

   > The primary benefit of table definitions is code reuse. Having the InfoSphere Information Server product generate XPath expressions for us is a bonus.

   a. From the July 2007 edition of IISDN, complete Page 22, Steps, 13.a-f of that document.

You are done when you produce Figure 7-16 of that document; we want to be able to copy (and later paste) the URL address of this Web service's WSDL document.

Note: To complete the step above, we use the InfoSphere Information Server (Administrative) Web Console. In the real world, you might not have access to this console, or have administrator rights. That's fine. In the real world we'd have a WSDL explorer configured, or someone would just give you the WSDL file.

If you don't have an icon for the InfoSphere Information Server (Administrative) Web Console on your desktop, this program operates inside a Web browser. What you need then is the URL to this (Web page). The default value for this URL is,

```
http://hostname:9080/loginForm.jsp
```

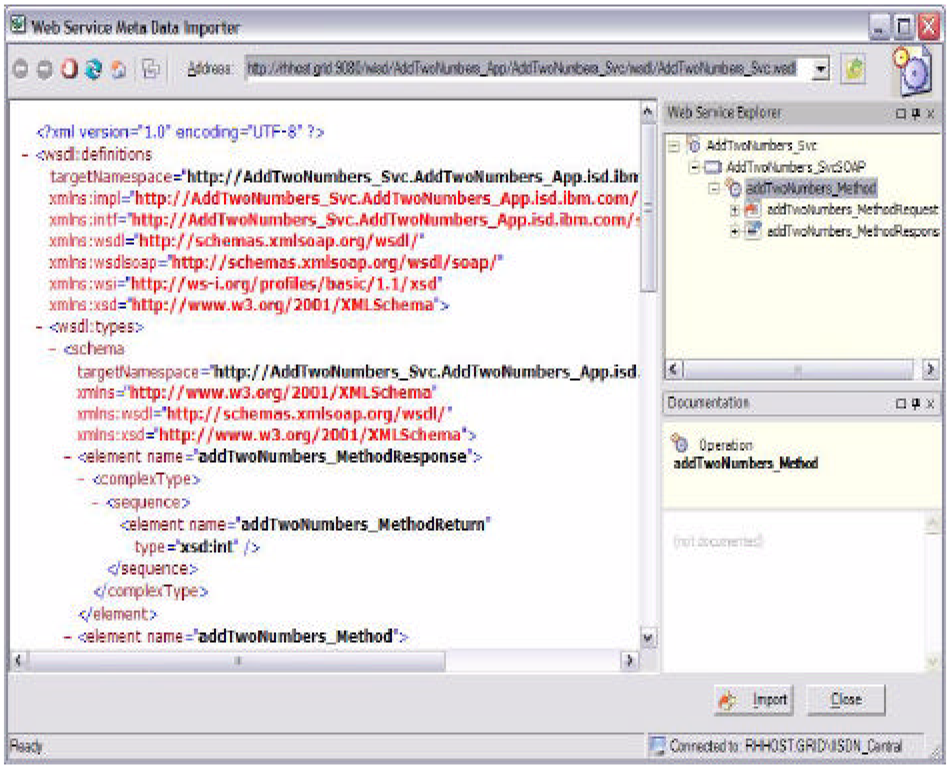
... where *hostname* is replaced by the actual hostname where the Java/J2EE compliant application server component of InfoSphere Information Server is operating. 9080 is the default port number, and may vary. The remainder of that URL is constant.

Again, you are done with the above step when you have copied the URL of the WSDL page for the Web service we wish to consume in this example.

b. Back in the DataStage/QualityStage Designer program, select the following from the Menu Bar,

Import -> Table Definitions -> Web Services WSDL Definitions...

This action produces the display as shown in Figure20-5.


Web Service Meta Data Importer
Address: http://rhhost.grid:9080/wsd/AddTwoNumbers_App/AddTwoNumbers_Svc/wsdl/AddTwoNumbers_Svc.wsdl
Web Service Explorer
AddTwoNumbers_Svc
AddTwoNumbers_SvcSOAP
addTwoNumbers_Method
addTwoNumbers_MethodRequest
addTwoNumbers_MethodRespons
Documentation
Operation
addTwoNumbers_Method
(not documented)
Import
Close
Ready
Connected to: RHHOST.GRID\ISDN_Central

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
    targetNamespace="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm
    xmlns:impl="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm.com/
    xmlns:intf="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm.com/
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsi="http://ws-i.org/profiles/basic/1.1/xsd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <schema
        targetNamespace="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      - <element name="addTwoNumbers_MethodResponse">
        - <complexType>
          - <sequence>
              <element name="addTwoNumbers_MethodReturn"
                type="xsd:int" />
            </sequence>
          </complexType>
        </element>
      - <element name="addTwoNumbers_Method">
```

Figure 20-5   Menu Bar: Import -> Table Definitions -> Web Services WSDL Definitions...

c. In Figure20-5, in the text entry field labeled, Address, paste the URL to the WSDL file we copied in Step-3.a above.

   Then click the (Go button) a green arrow just to the right of the Address bar. The hover help/label for the (Go button) displays, “Open a local (or UNC) file”.

   Based on your specific InfoSphere Information Server configuration, you may optionally receive one or more dialog boxes asking you to user verify; generally you may Cancel out of those.

   You are done with this step when your displays equal the upper right portion of Figure20-5.

d. Based on the naming conventions outlined in the July 2007 edition of this document, we wish to highlight the entry in Figure20-5 entitled, “addTwoNumbers_Method”.

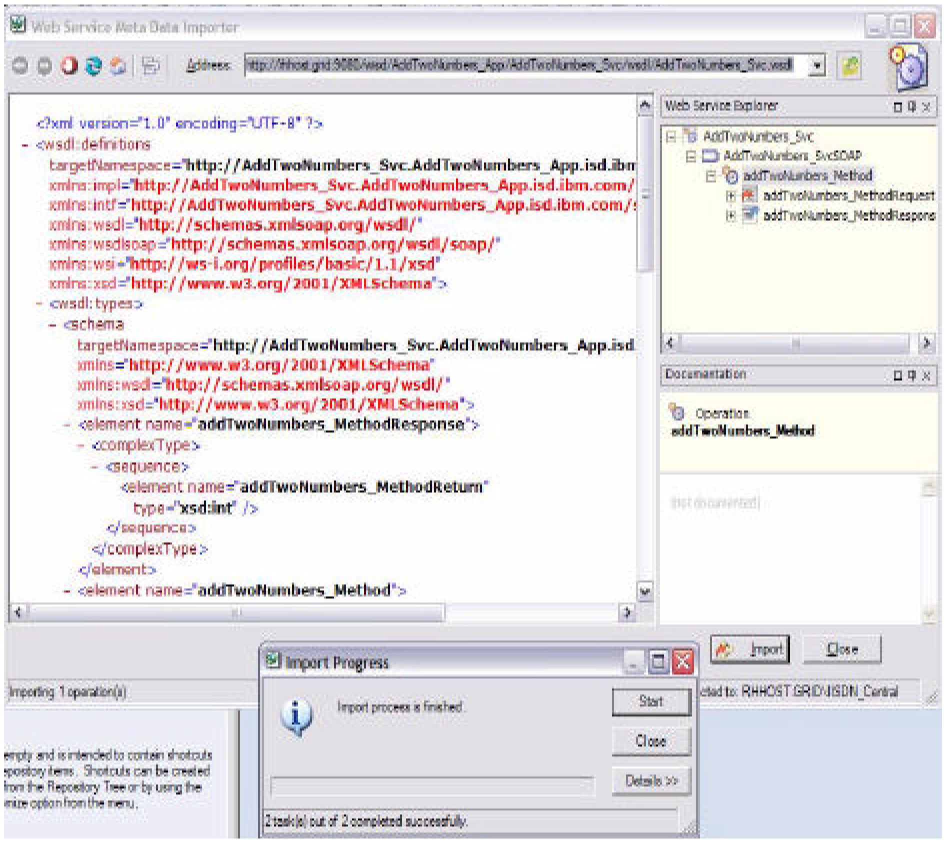
   As a rule, this entry will start with a lowercase letter, and its member elements will include both “request” and “response” items.

You are done with this step when you highlight this entry with a single Click.

e. Then Click the Import Button at the bottom of Figure20-6.

This action will produce the Import Progress dialog box, also as shown in Figure 20-6.

When this process completes, Click Close, and then Click Close again.


Web Service Meta Data Importer
Address: http://rhhost.grid:9080/wsd/AddTwoNumbers_App/AddTwoNumbers_Svc/wsdl/AddTwoNumbers_Svc.wsdl
Web Service Explorer
AddTwoNumbers_Svc
AddTwoNumbers_SvcSOAP
addTwoNumbers_Method
addTwoNumbers_MethodRequest
addTwoNumbers_MethodRespons
Documentation
Operation
addTwoNumbers_Method
(not documented)
Import
Close
Import Progress
Start
Close
Details >>

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
    targetNamespace="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm
    xmlns:impl="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm.com/
    xmlns:intf="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm.com/s
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsi="http://ws-i.org/profiles/basic/1.1/xsd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <schema
        targetNamespace="http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      - <element name="addTwoNumbers_MethodResponse">
        - <complexType>
          - <sequence>
              <element name="addTwoNumbers_MethodReturn"
                type="xsd:int" />
            </sequence>
          </complexType>
        </element>
      - <element name="addTwoNumbers_Method">
```

```
Importing 1 operation(s)
Import process is finished.
2 task(s) out of 2 completed successfully.
...cted to: RHHOST.GRID\ISDN_Central
```

```
...empty and is intended to contain shortcuts
...pository items. Shortcuts can be created
from the Repository Tree or by using the
...mize option from the menu.
```

Figure 20-6   Tale Importer, part 2.

f. The above action will produce several new DataStage/QualityStage Table Definitions in the Repository view, under,

Table Definitions -> Web Services -> {Web service name}

Example as shown in Figure20-7.

Specifically we will make direct use of the “IN” and OUT” table definitions; the columns that are passed in, and received out of this Web service invocation.

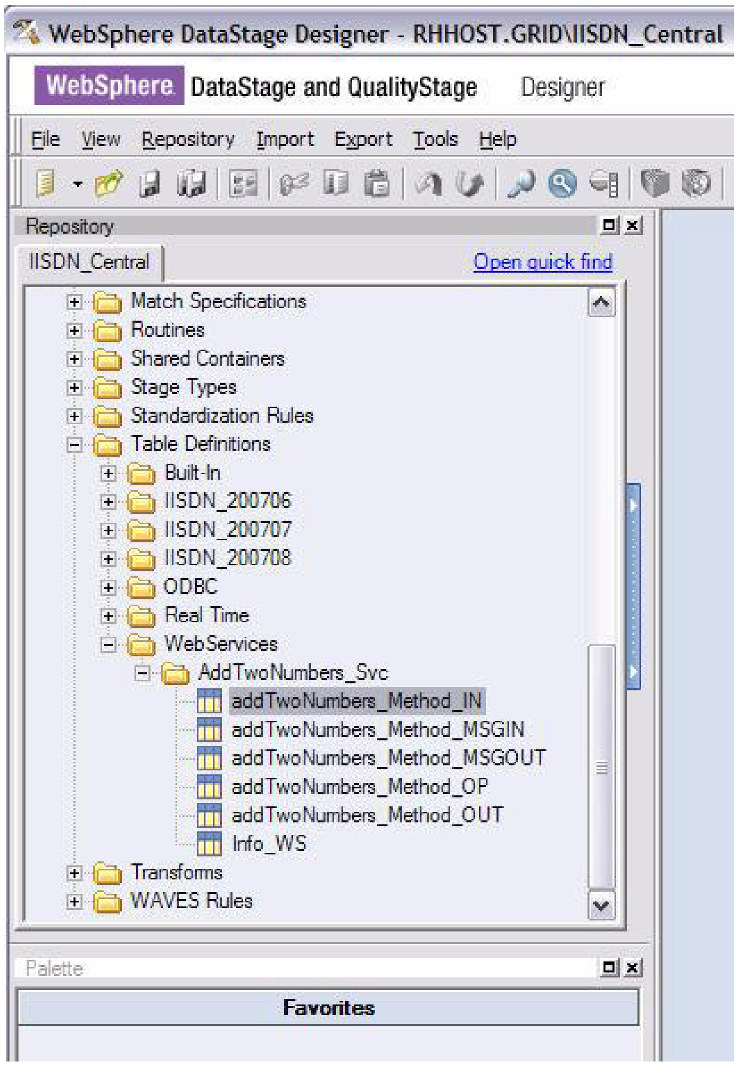


Figure 20-7 Repository view of Table Definitions created above.

Note: Remember the technique above. We use it later with another, newly created Web service.

4. Create a new parallel Job, equal to what is shown in Figure20-3 and Figure20-4.

    a. Save this Job with a name of your choosing.

    b. This example Job contains 2 (count) Sequential File stages from the File drawer, and 1 (count) Web Services Transformer stage from the Real Time drawer.

        Position, connect, and rename the stages and links as displayed in Figure 20-3 and Figure20-4.

    c. *As always*, the Sequential File stages require that the following properties be set;

        - Output/Input TAB -> Properties TAB -> Source -> File (name)
        - Output/Input TAB -> Format TAB -> Record level -> Final delimiter
        - Output/Input TAB -> Format TAB -> Record level -> Record delimiter
        - Output/Input TAB -> Format TAB -> Field defaults -> Delimiter
        - Output/Input TAB -> Format TAB -> Field defaults -> Quotes

    Don't worry about setting the input or output columns on the Sequential File stages, we will do those soon.

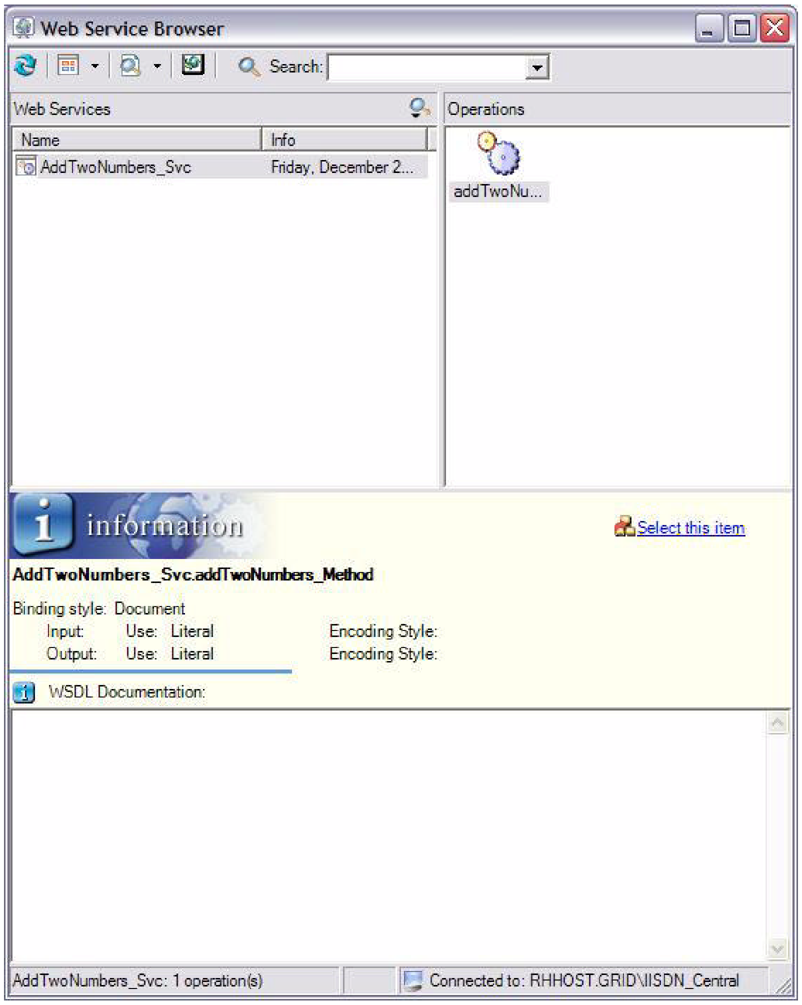
5. Configure the Web Services Transformer stage.

    a. Open this stage by Double-clicking it, to access its Properties.

    b. Under the Stage TAB -> General TAB, click the button entitled, Select Web Service Application.

        This action produces the Web Service Browser dialog box, example as shown in Figure20-8.

Note: The dialog box displayed in Figure20-8 is populated from entries located under the Repository view, Table Definitions -> Web Services.



Name	Info
AddTwoNumbers_Svc	Friday, December 2...

AddTwoNumbers_Svc.addTwoNumbers_Method

Binding style: Document
Input: Use: Literal Encoding Style:
Output: Use: Literal Encoding Style:

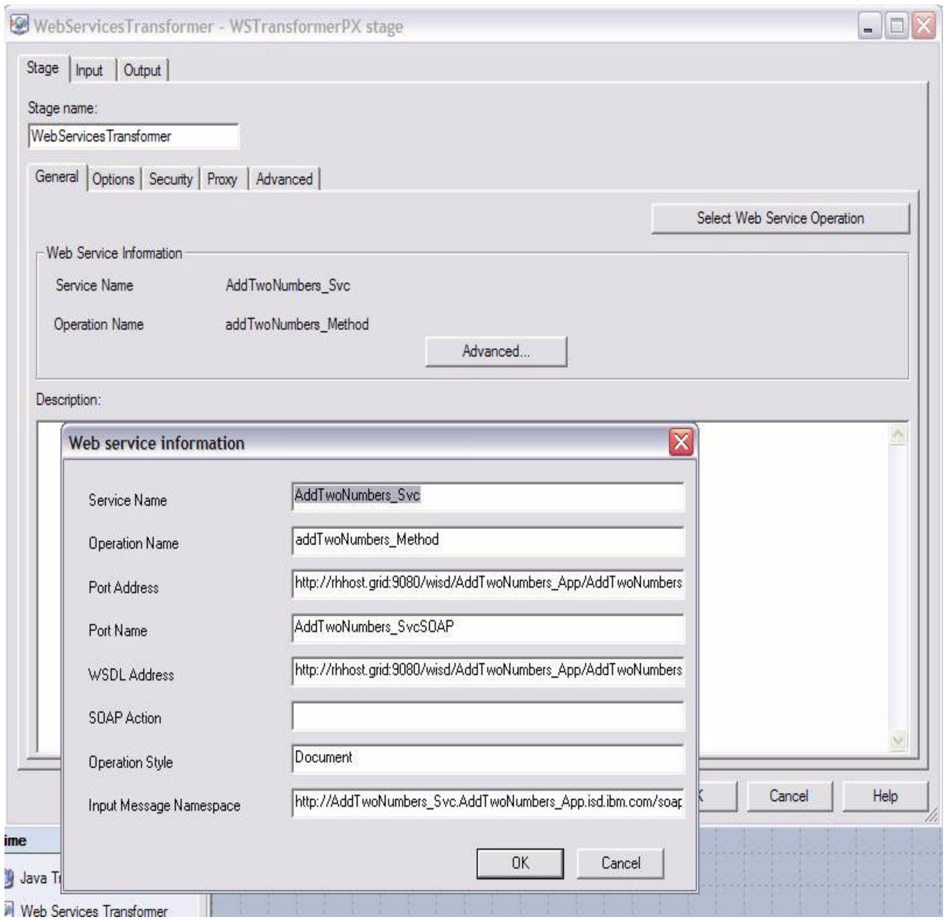
AddTwoNumbers_Svc: 1 operation(s)
Connected to: RHHOST.GRID\IISDN_Central

Figure 20-8 Web Service Browser

c. In Figure20-8, highlight the addTwoNumbers_Method entry in the Operations portion of the display, then Click “Select this item”.

This action will close the Web Service Browser dialog box.

If you Click the Advanced button in Figure20-9, you can see some of the values that were populated as the result of your work. Ultimately this data came from the WSDL file we located earlier.


WebServicesTransformer - WSTransformerPX stage
Stage
Input
Output
General
Options
Security
Proxy
Advanced
Select Web Service Operation
Advanced...
OK
Cancel
Help

Stage name: WebServicesTransformer

Web Service Information

Service Name	AddTwoNumbers_Svc
Operation Name	addTwoNumbers_Method

Description:

Web service information

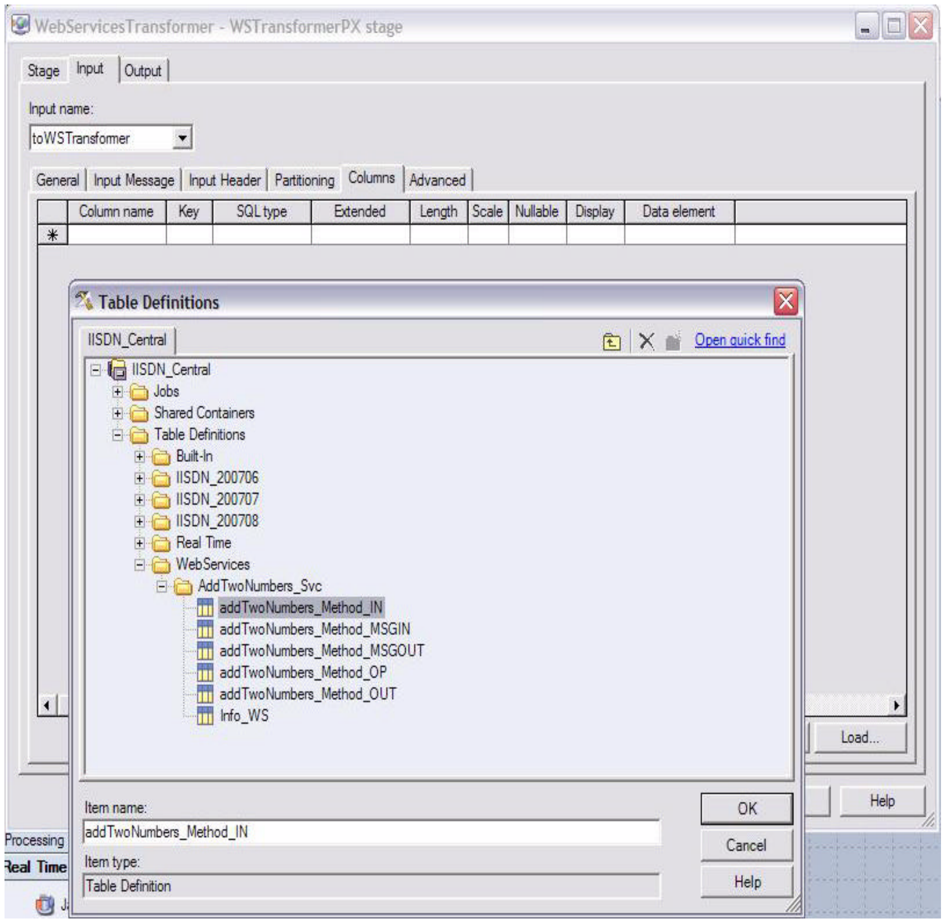
Service Name	AddTwoNumbers_Svc
Operation Name	addTwoNumbers_Method
Port Address	http://rhhost.grid:9080/wisd/AddTwoNumbers_App/AddTwoNumbers
Port Name	AddTwoNumbers_SvcSOAP
WSDL Address	http://rhhost.grid:9080/wisd/AddTwoNumbers_App/AddTwoNumbers
SOAP Action	
Operation Style	Document
Input Message Namespace	http://AddTwoNumbers_Svc.AddTwoNumbers_App.isd.ibm.com/soap

Figure 20-9   Web Service Browser, results of.

d. Still inside the Web Services Transformer properties dialog box, move to the Input TAB -> Columns TAB.

   Click the Load button, and navigate to the table definitions we created above. Select the table definition labeled "Something_IN", example as shown in Figure20-10.

   Click OK, and click through to select all columns.



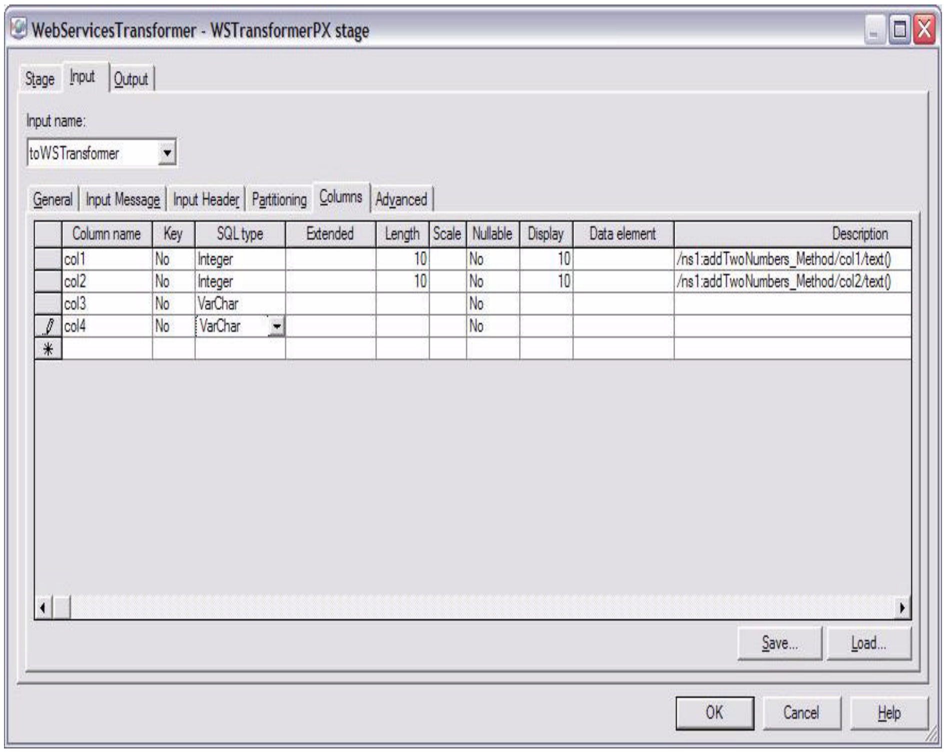
Item name: addTwoNumbers_Method_IN

Item type: Table Definition

Figure 20-10   Loading columns into the Input side of our Web service.

e. Manage the display to add two additional columns, as shown in Figure 20-11.

   The final product should have 4 total columns, in the order displayed in Figure 20-11. This column order matches our input data source, a Sequential File stage.



Column name	Key	SQL type	Extended	Length	Scale	Nullable	Display	Data element	Description
col1	No	Integer		10		No	10		/ns1:addTwoNumbers_Method/col1/text()
col2	No	Integer		10		No	10		/ns1:addTwoNumbers_Method/col2/text()
col3	No	VarChar				No			
col4	No	VarChar				No			

Figure 20-11   Adding two additional columns to support our Job as designed.

Note: Due to a product feature of DataStage/QualityStage called Run Time Column Propagation, these 4 columns (created in our middle stage) are now listed in the input file stage, (Sequential File) which starts/leads our Job.

When you finish the work of configuring this stage, return to the input file stage and Click, View Data, to ensure that the input file stage is configured correctly.

The column names are not significant in this case, but column order is. This is because we are using a Sequential File operator, which processes columns in their physical sequence/order.

The entries in Figure 20-11, under the Description column, are called XPath expressions. Only the columns with XPath expressions will be sent to our Web service invocation, which is our intent. Without an XPath expression, the run time environment wouldn't know how or where to send any additional columns.

The columns, named col3 and col4, are not passed to the Web service, but are passed through this stage, as you will see.

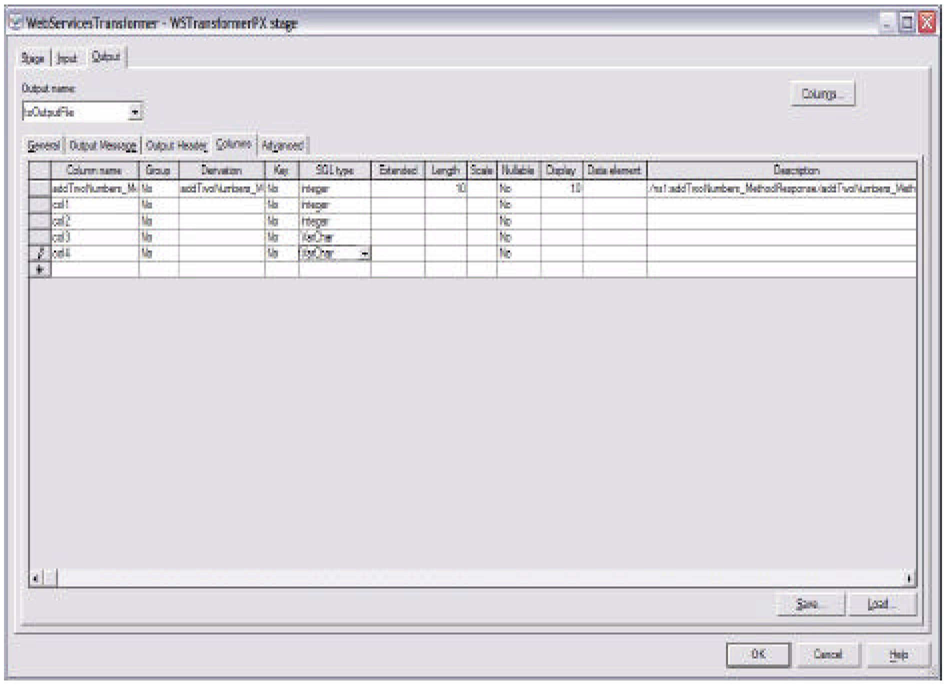
f. Move to the Output TAB -> Columns TAB.

   Load the columns as before, however, this time load the columns from the OUT side of this Web service invocation. There should be one column in this list, of type Integer.

   Add 4 columns manually to equal the 4 input columns to this stage. We do this purely for style in this case. The actions above also set the (input) columns to our final stage, a Sequential File stage used for output.

   You are done when your display equals that as shown in Figure 20-12.

   Click OK to close this dialog.



Column name	Group	Derivation	Key	SQL type	Extended	Length	Scale	Nullable	Display	Data element	Description
addTwoNumbers_M	No	addTwoNumbers_M	No	Integer		10		No	10		/ns1:addTwoNumbers_MethodResponse/addTwoNumbers_Meth
col1	No		No	Integer				No			
col2	No		No	Integer				No			
col3	No		No	VarChar				No			
col4	No		No	VarChar				No			

Figure 20-12   Output columns specification.

g. Set/add two new Environment Variables to this Job.

Figure 20-2, above, lists two Environment Variables that must be added to this Job. These two Environment Variables are required for all Jobs that use the Web Service Transformer and/or the Web Service Clients stages.

Because these variables are used often, you may choose to add them to a Parameter Set; set them in one location and re-use as needed.

For simplicity here, we will say use simple Job Variables.

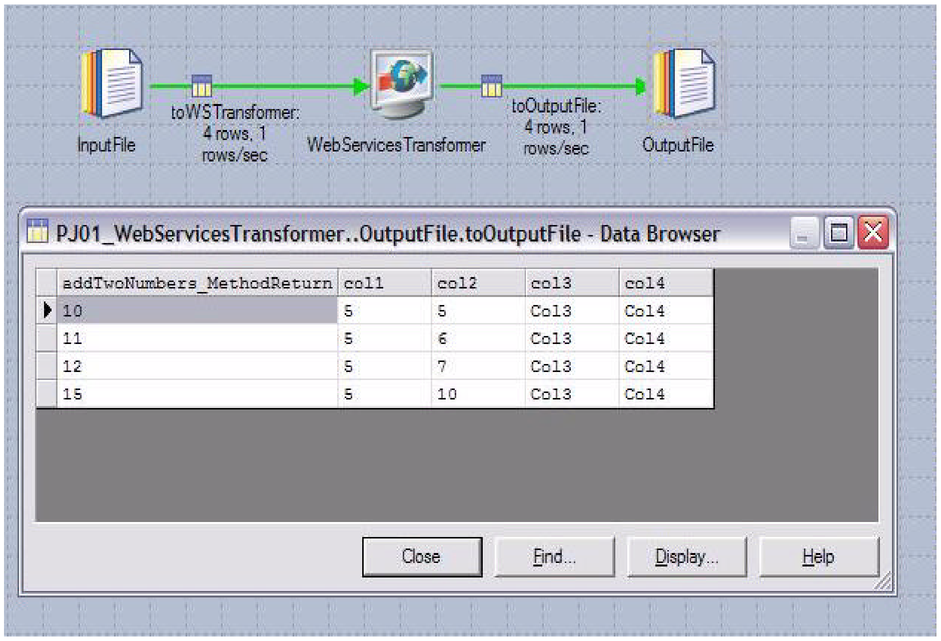
From the Menu Bar, select, Edit -> Job Properties -> Parameters, and add two Environment Variables as displayed in Figure 20-2.

Click OK when you are done.

h. Save, compile and test.

This Job should now be complete; save, compile and run.

A successful test looks like the display in Figure20-13.


InputFile
toWSTransformer: 4 rows, 1 rows/sec
WebServicesTransformer
toOutputFile: 4 rows, 1 rows/sec
OutputFile
PJ01_WebServicesTransformer..OutputFile.toOutputFile - Data Browser
Close
Find...
Display...
Help

addTwoNumbers_MethodReturn	col1	col2	col3	col4
10	5	5	Col3	Col4
11	5	6	Col3	Col4
12	5	7	Col3	Col4
15	5	10	Col3	Col4

Figure 20-13 Job One, successful result.

20.3 Using the DS/QS Web Service Client stage

For the sample DataStage/QualityStage Job above, (the first Job, the Web Services Transformer stage example), we had an existing Web service that would suit our needs; we had a Web service we could consume with both input and output, a requirement for any Transformer stage.

The DataStage/QualityStage Web Service Client stage, however, can consume a Web service for *input or output only*; *not both input and output*, not like a Transformer. Since we don't have a Web service of that type handy, we will make one. And, since we're at it, let's go with the super fabulous version that handles the most complex use case possible; one that sends and receives arrays of data.

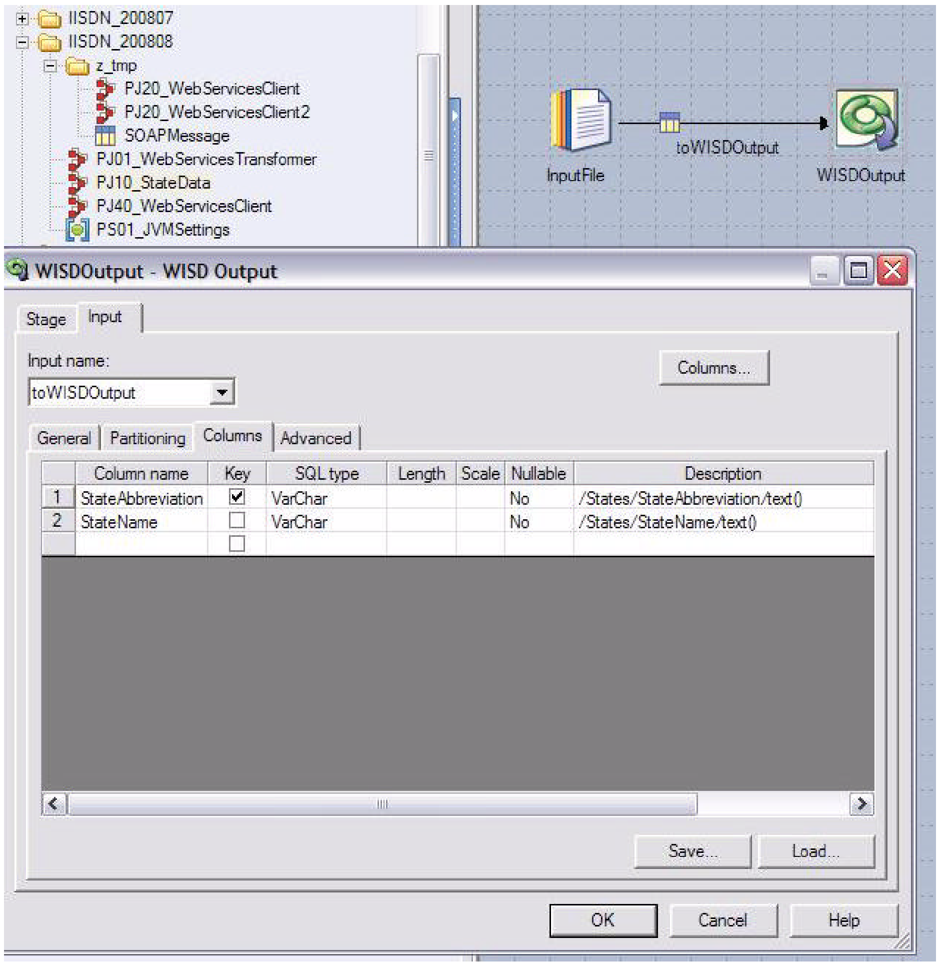
Web services aren't limited to two dimensional (flat) records; two dimensional (flat) records as in SQL based systems, rows and columns only. Web services allow nested records, complex data types, other. The example that follows is most capable, and while we will demonstrate arrays only, you will then be fully enabled for much more complex use cases. Who's your buddy? IISDN, that's who.

Advanced case: Step 1, Make a new Web service provider

For this section of this document, we are going to rely heavily on the July 2007 edition of (IBM) InfoSphere Information Server Developer's Notebook; Web Services Part I. (To a smaller extent, we also rely on the June 2007 edition of IISDN to get us the information towards using the XML Input stage.)

In this section of this document, we create and deploy a new DataStage/QualityStage Job to act as a Web service provider. This Web service differs from the one used above in that this new Web service only outputs data.

6. Create a new DataStage/QualityStage Parallel Job.
    a. As displayed in Figure20-14, this new Job has two stages; a Sequential File input, and a WISD Output.



Input name: toWISDOutput

	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	StateAbbreviation	✓	VarChar			No	/States/StateAbbreviation/text()
2	StateName	☐	VarChar			No	/States/StateName/text()
		☐					

Figure 20-14   New DS/QS Job providing Web service, to support Web Services Client.

b. The Sequential File input should read from a two column file with numerous records. We used the example of State Abbreviations and State Names.

   You will need to create that source input data file on your own, and populate it with data.

c. The WISD Output stage will list both columns from the Sequential File input, and be certain to contain XPath expressions as listed in Figure 20-14. Be certain too to Check the StateAbbreviation column as the Key field.

d. Follow the July 2007 edition of the IISDN document, specifically Pages 9-12, Steps,1-4.

e. *Do not perform the July 2007 edition of IISDN, Steps 5-6*; we don't need a new or additional Information Services Connection.

f. Continue in the July 2007 edition of IISDN at Pages 15-21, Steps 9-12.

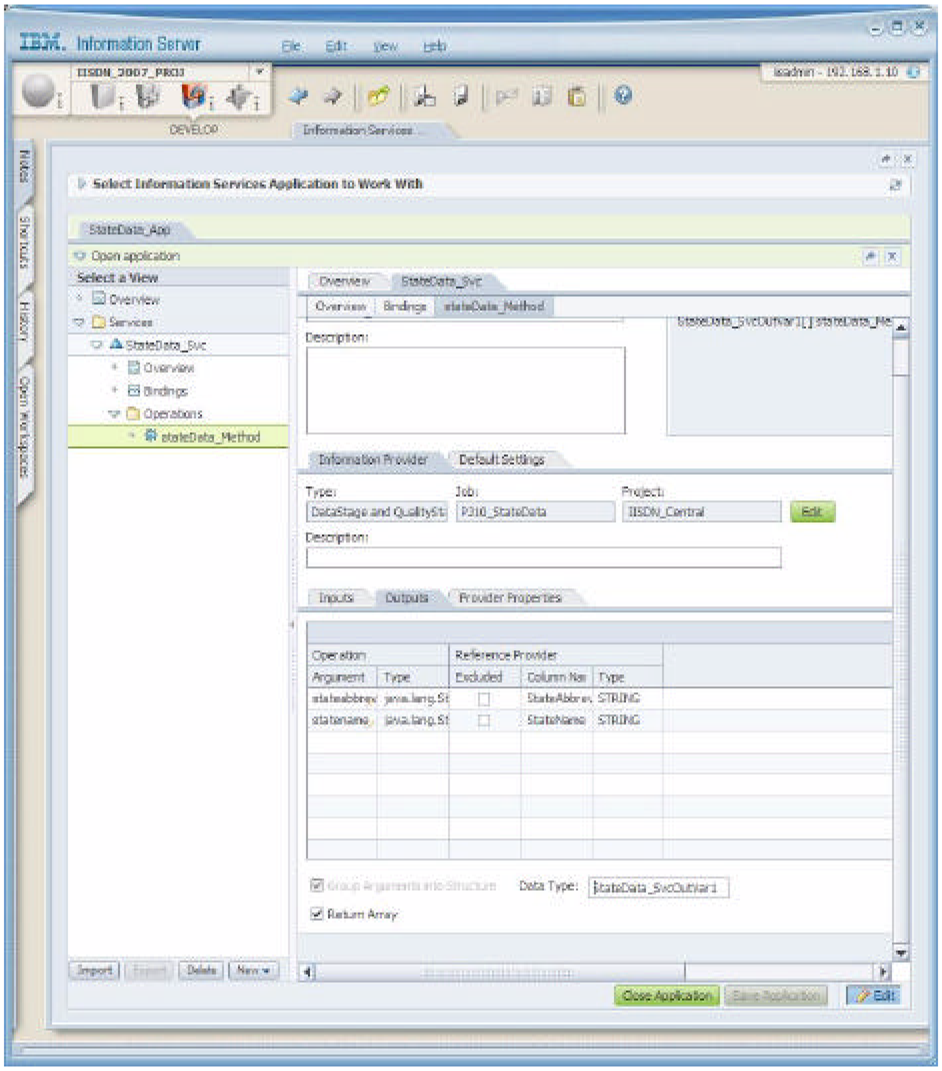
   Besides using different identifiers (names of things) in each of these steps, there is one change we wish to make in this example.

   The step we need to change is Step10.g, displayed in Figure 7-14.

   As written for the July 2007 IISDN example, Step 10.g Figure 7-14, is just a click through. Our use here, however, is different. Before leaving Step 10.g, we wish also to Check the visual control entitled, Return Array.

   Example as shown in Figure20-15.

   Checking this control allows this Web service to return multiple State records. If we don't Check this visual control, we only get one State record in return, which is not our design intent.


Type: DataStage and QualitySt
Job: P310_StateData
Project: IISDN_Central

Operation		Reference Provider		
Argument	Type	Excluded	Column Na	Type
stateabbrev	java.lang.St	☐	StateAbbrev	STRING
statename	java.lang.St	☐	StateName	STRING

☑ Group Arguments into Structure   Data Type: StateData_SvcOutVar1
☑ Return Array

Figure 20-15   Information Server Console, State Data Application.

g. You are done with this section when you have deployed this new Web service, Page 21, Steps 12,a-d, from the July 2007 of the IISDN document.

Test this newly deployed Web service

Following the July 2007 edition of the IISDN document, complete Pages 21-25, Steps 13-14, to test this new Web Service. Success has you receiving multiple output records, every line from your source file of State data above.

Advanced case: Step 2, Get the SOAP message for analysis

Every successful Web service invocation issued via the HTTP/SOAP communication protocol results in a response in the form of a SOAP message. A SOAP message has header information, and a message payload (the data proper). Both of those data sets, and the ability to read each, can be a very handy and unique skill to possess.

SOAP messages are in ASCII text, and XML formatted. Because the data proper being returned is unique to every application, each SOAP message offers a unique XML schema definition (XSD). (The SOAP message header is standard, the data proper is not; the data proper is specific to your application.)

In this section of this document, we are going to create a throw away DataStage/QualityStage Job; a Job just to land our Web service invocation SOAP message response somewhere. From the landed SOAP message, we may use automated tools to generate an XML schema definition (XSD) for us. (In this context, the XSD is like a DataStage/QualityStage table definition. It has columns, data types, other.)

Figure20-16 displays the DataStage/QualityStage Job we will make next in this next section.

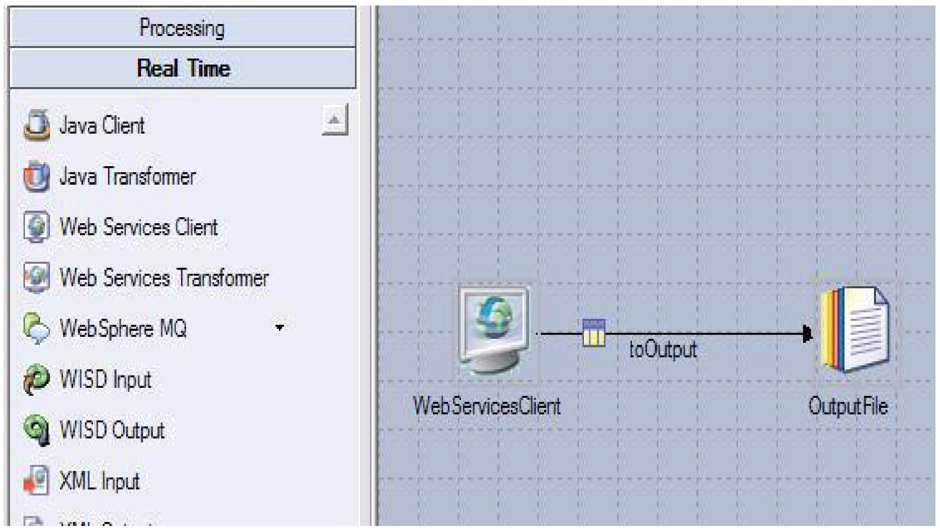


Figure 20-16 Simple DS/QS Job to capture SOAP message.

7. Create a DataStage/QualityStage table definition for the State list Web service we created in Step 6, of this document, above.

   This work is similar to what we did in Steps 3.a-f above in this document.

You are done with this step when you have table definition entries similar to what is displayed in Figure 20-17.

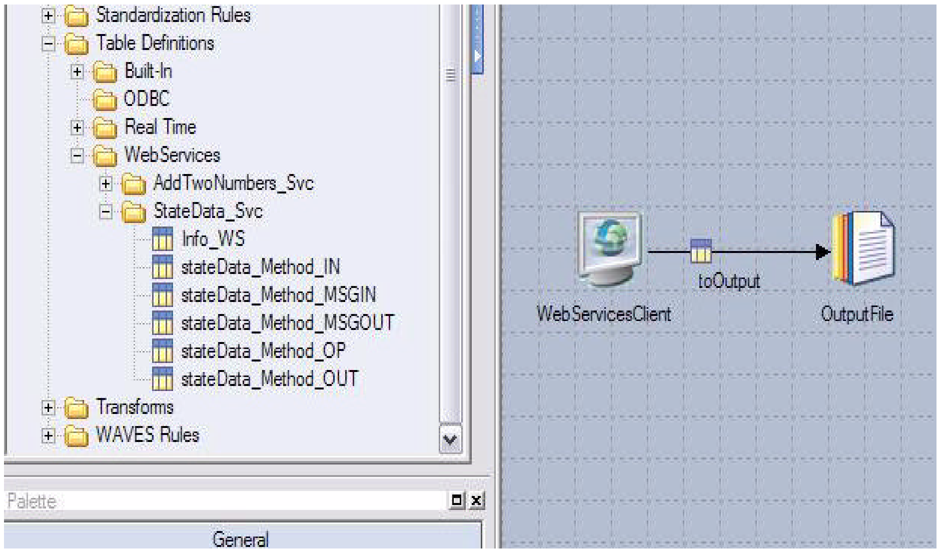


Figure 20-17 Importing the State Data Web Service Table Definitions.

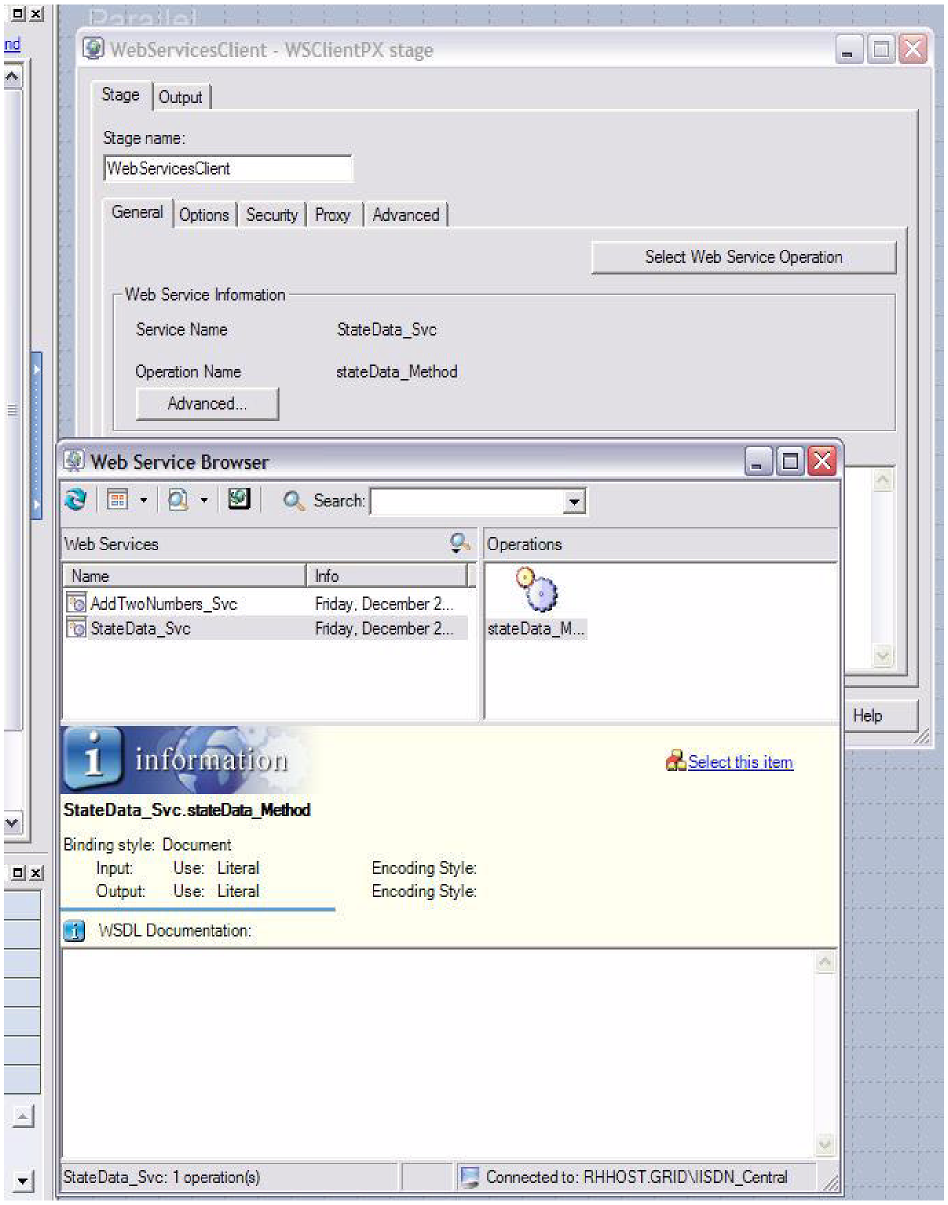
8. Create a new DataStage/QualityStage Parallel Job.
   a. As displayed in Figure20-16, this new Job has two stages; a Web Service Client stage for input, and a Sequential File output.
   b. Double-click the Web Service client stage to access its properties.

      As displayed in Figure20-18, access the Stage TAB -> General TAB, and Click the button entitled, Select Web Service Operation.

      This action produce the Web Service Browser dialog box.
   c. In the Operations area of this display, select the stateData_Method, or equal Web service invocation method, (you may have used a different service name).

      Then Click the visual control entitled, "Select this item".

      This action closes the Web Service Browser dialog box.


WebServicesClient - WSClientPX stage

Stage | Output

Stage name: WebServicesClient

General | Options | Security | Proxy | Advanced

Select Web Service Operation

Web Service Information

Service Name StateData_Svc

Operation Name stateData_Method

Advanced...

Web Service Browser

Search:

Web Services | Operations

Name	Info
AddTwoNumbers_Svc	Friday, December 2...
StateData_Svc	Friday, December 2...

stateData_M...

information

Select this item

StateData_Svc.stateData_Method

Binding style: Document

Input: Use: Literal Encoding Style:

Output: Use: Literal Encoding Style:

WSDL Documentation:

StateData_Svc: 1 operation(s)

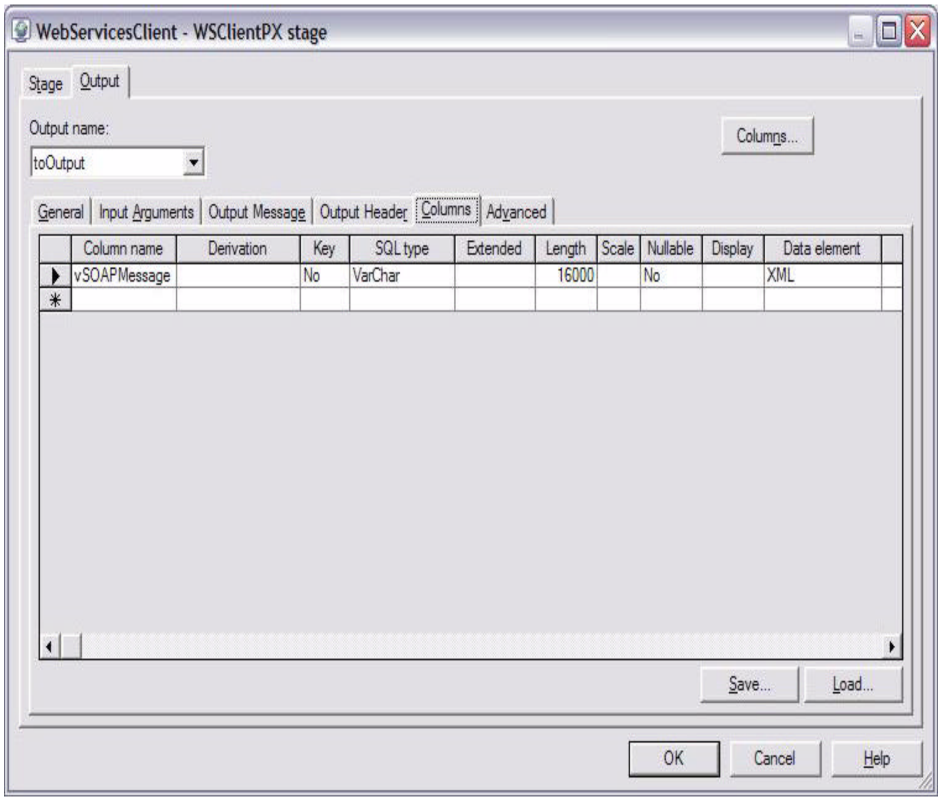
Connected to: RHHOST.GRID\IISDN_Central

Help

Figure 20-18 Setting Properties within Web Service Client.

d. Still inside the Properties dialog for the Web Service client stage, move to the Output TAB -> Columns TAB.

Create a new single line entry equal to what is displayed in Figure20-19.



Column name	Derivation	Key	SQL type	Extended	Length	Scale	Nullable	Display	Data element
vSOAPMessage		No	VarChar		16000		No		XML

Figure 20-19 Web Services Client stage, Output TAB -> Columns TAB.

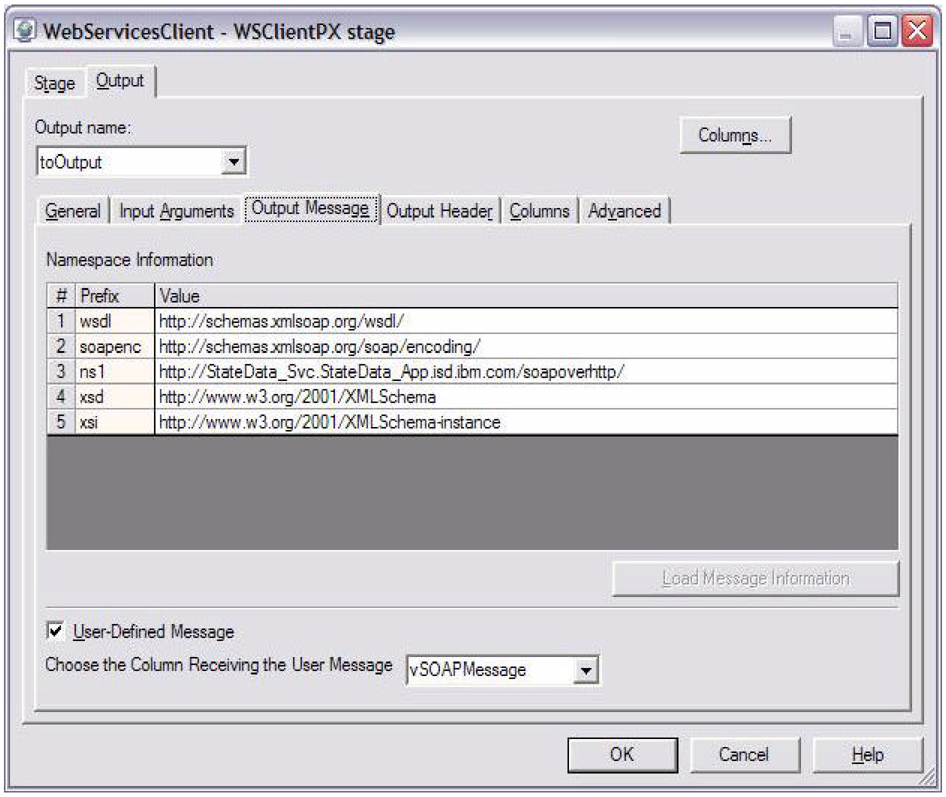
e. Move to the Output TAB -> Output Message TAB.

Check the visual control entitled, User-Defined Message, and use the associated Drop Down List Box entitled, "Choose the Column...".

Select the column, vSOAPMessage we defined in Step 8.d above. Example as displayed in Figure20-20.

Click OK when you are done.

The Web Services Client stage is configured.



Output name: toOutput

Namespace Information

#	Prefix	Value
1	wsdl	http://schemas.xmlsoap.org/wsdl/
2	soapenc	http://schemas.xmlsoap.org/soap/encoding/
3	ns1	http://StateData_Svc.StateData_App.isd.ibm.com/soapoverhttp/
4	xsd	http://www.w3.org/2001/XMLSchema
5	xsi	http://www.w3.org/2001/XMLSchema-instance

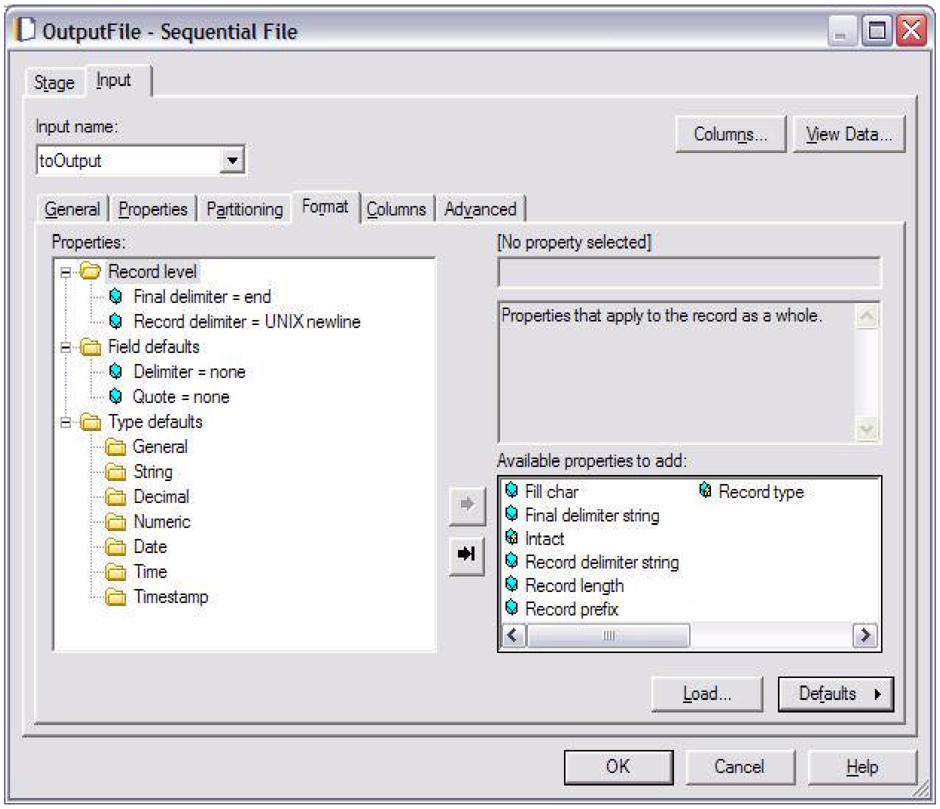
[✓] User-Defined Message

Choose the Column Receiving the User Message vSOAPMessage

Figure 20-20   Web Services Client stage, Output TAB -> Output Message TAB.

f. Complete configuration of the Sequential File stage used for output. To save yourself a brief step later, use an output file name with a ".xml" file name suffix.

   Choose to format the contents of this file as shown in Figure20-21; you don't want to add double quotes or unexpected line feeds, messing up your XML file.



Properties:
- Record level
  - Final delimiter = end
  - Record delimiter = UNIX newline
- Field defaults
  - Delimiter = none
  - Quote = none
- Type defaults
  - General
  - String
  - Decimal
  - Numeric
  - Date
  - Time
  - Timestamp

[No property selected]

Properties that apply to the record as a whole.

Available properties to add:
- Fill char
- Final delimiter string
- Intact
- Record delimiter string
- Record length
- Record prefix
- Record type

Figure 20-21 Sequential File stage, Input TAB -> Format TAB.

g. Repeat the instructions from Step 5.g in this document, above. This step sets the required JVM associated Environment Variables for this new Job.

h. Save, compile and test.

This DataStage/QualityStage Job will run slightly slower than the previous DataStage/QualityStage Jobs in this document, because this is our first Web Service invocation Job, and as such, needs to create its own Java run time environment.

A successful test creates only one line of output; the SOAP message response to our successful Web service invocation. As a further test, open this XML file response with a Web browser. The Web browser should not complain that the document has errors, or is poorly formatted. Also, you should see numerous State data lines, not just one State data line.

Advanced case: Step 3, Parse the SOAP message

The last DataStage/QualityStage Job we created was really just a throw away; we really just wanted an easy way to capture the outputted SOAP message from our State data Web service. To save steps, we can now add to that last Job. If you wish a more complete record of your work, first open that last Job, and then Select, File -> Save As..., from the Menu Bar.

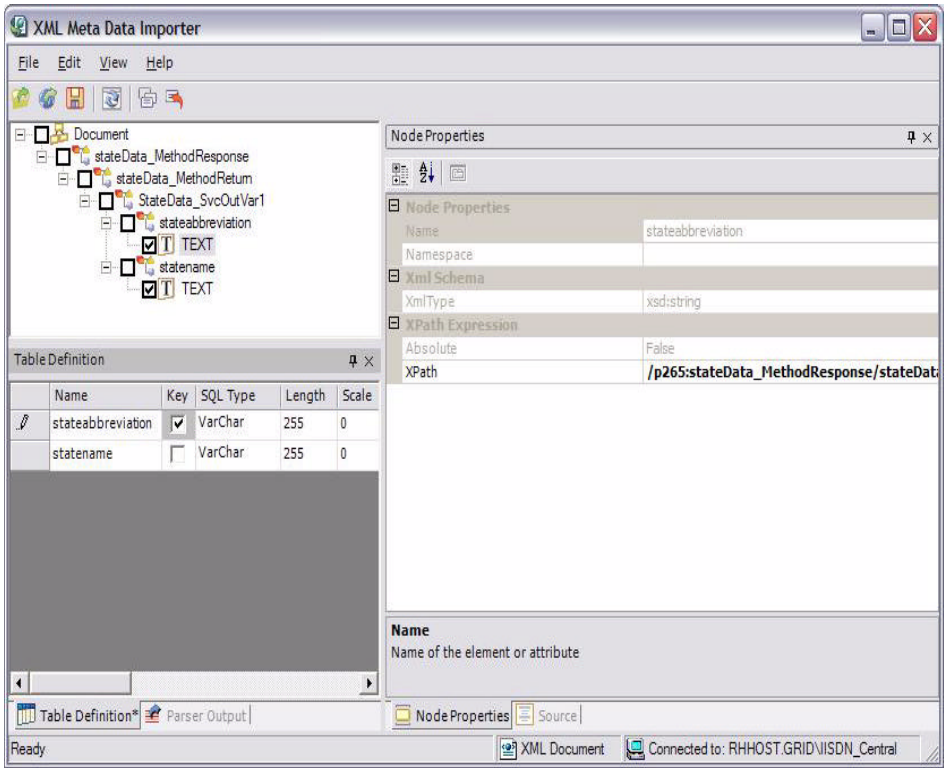
The SOAP message response we captured above is standard in the area of its message header, but variable (unique to that Job) in its formatting and hence the modelling of, its data proper. In this section of this document, we will first create a DataStage/QualityStage table definition of the SOAP message, then use the XML Input stage to parse the data proper of that SOAP message.

Note: This is a hugely powerful technique; capturing the full Web service invocation SOAP message response, generating an XML schema definition (XSD) for it, and then parsing exactly what we want out of that message, be it data or response header.

In Java, which we adore, the above is hundreds of lines of source code, and very difficult to learn to do from scratch.

9. Create a DataStage/QualityStage table definition for the SOAP message response we received from the State data Web service invocation.
    a. Still inside the DataStage/QualityStage Designer program, and from the Menu Bar, Select, Import -> “XML Table Definitions...”.

       This action produces the XML Meta Data Importer dialog box, as displayed in Figure 20-22.



```
Document
  stateData_MethodResponse
    stateData_MethodReturn
      StateData_SvcOutVar1
        stateabbreviation
          TEXT
        statename
          TEXT
```

Table Definition

Name	Key	SQL Type	Length	Scale
stateabbreviation	✓	VarChar	255	0
statename		VarChar	255	0

Node Properties

Node Properties	
Name	stateabbreviation
Namespace	
Xml Schema	
XmlType	xsd:string
XPath Expression	
Absolute	False
XPath	/p265:stateData_MethodResponse/stateDat

Name
Name of the element or attribute

Figure 20-22   XML Meta Data Importer, Menu Bar, Import -> XML Table Definitions...

b. On the Tool Bar to this dialog box, Click the left most icon, Open File.

   Browse to the XML file we outputted in the DataStage/QualityStage Job above.

   If you are running a Client/Server installation of InfoSphere Information Server, you may be prompted to enter a user name and password to access the server.

c. From Figure20-22, expand the tree in the upper left portion of the display so that you may Check both the Text entries for “stateabbreviation” and “statename”.

   Also in Figure20-22, check the Key attribute for “stateabbreviation”.

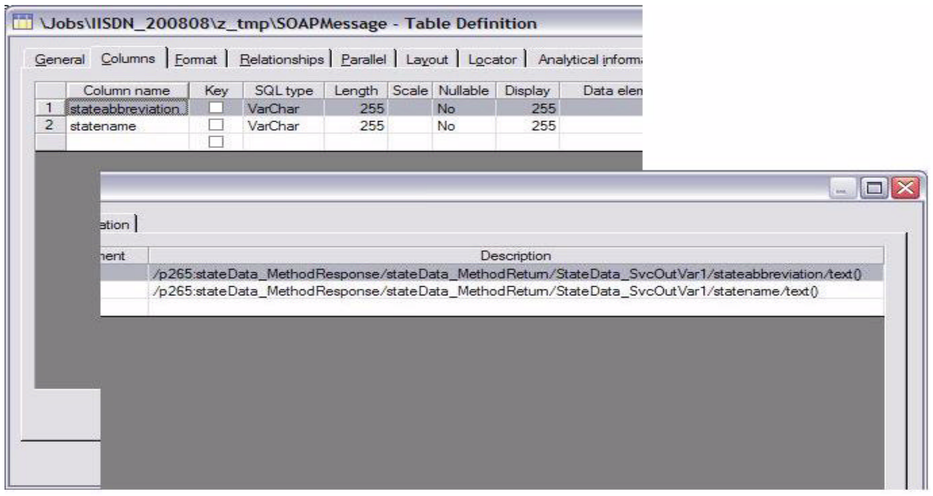
   You are done when your displays equals what is shown in Figure20-22.

d. From the Menu Bar to this dialog box, Select, File -> Save.

   This action produces the Save Table Definition dialog box.

You may browse to a new Folder Path, but do not change the “DataStage table definition name” at this time. (That name has to be precise and multi-part formatted. After you save this resource, you may rename it in the Repository view. We renamed ours “SOAPMessage”.

Note: The whole purpose of this step, Step 9, is to produce the XPath expressions displayed in Figure20-23. XML and XPath expressions can be at times unforgiving; if you don’t get these expressions just right, you don’t get the output you desire. Better to generate these expressions than to try and create them by hand.



	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data elen
1	stateabbreviation	☐	VarChar	255		No	255	
2	statename	☐	VarChar	255		No	255	
		☐						

nent	Description
	/p265:stateData_MethodResponse/stateData_MethodReturn/StateData_SvcOutVar1/stateabbreviation/text()
	/p265:stateData_MethodResponse/stateData_MethodReturn/StateData_SvcOutVar1/statename/text()

Figure 20-23 SOAP message table definition created above.

Create Web Service Client, with an XML formatter (XML Input)

At this point we have what we need to both invoke the State data Web service, and parse its SOAP message response.

10. To save time, continue with the DataStage/QualityStage Job we created above.

    Optionally perform a Menu Bar, File -> Save As..., to save the prior version of this Job for your review at a later time.

    a. Alter this Job to equal what is displayed in Figure 20-24.

       If you first drop the new stage, XML Input, on the Parallel Canvas, and then move the output link from Web Services Client to XML Input, you will preserve the properties of Web Services Client stage, and save time.

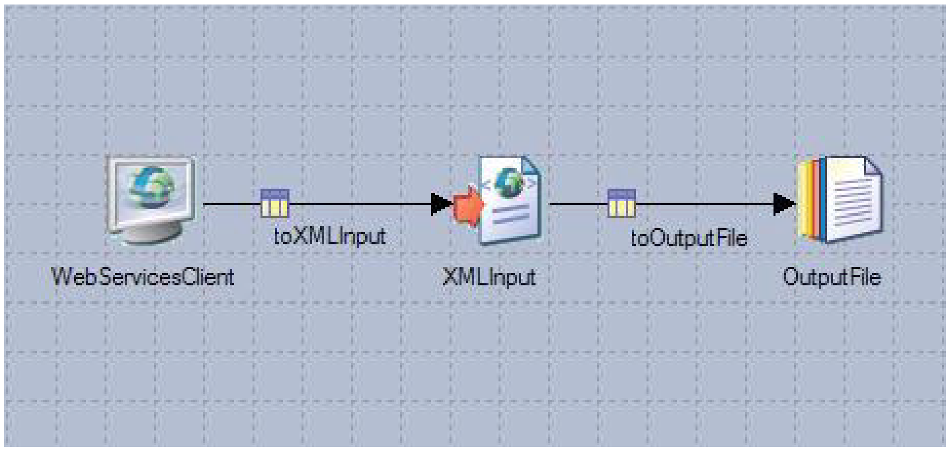


Figure 20-24 Next evolution of our DS/QS Job.

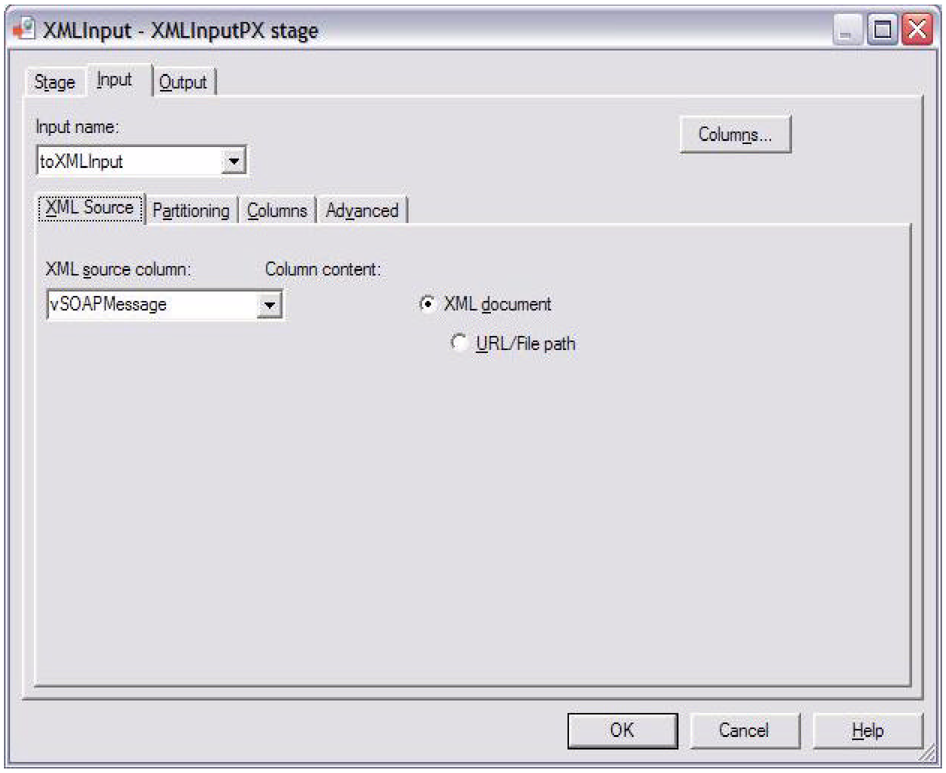
b. The Web Services Client stage should still be configured correctly. If not, repeat necessary steps/sub-steps from Steps 8.a-h, above.

c. Double-click the XML Input stage in order to access and set its Properties. Example as shown in Figure20-25.

d. Under the Input TAB -> XML Source TAB, press the XML document Radio Button.

e. For the XML source column, select what should be the only available choice; the only input column to this stage, vSOAPMessage.

   If this column is not available, you lost your properties from the preceding stage, the Web Services Client stage. If so, you must go back and accurately configure the Web Services Client stage.



Input name: toXMLInput

XML source column: vSOAPMessage

Column content: [✓ XML document] URL/File path

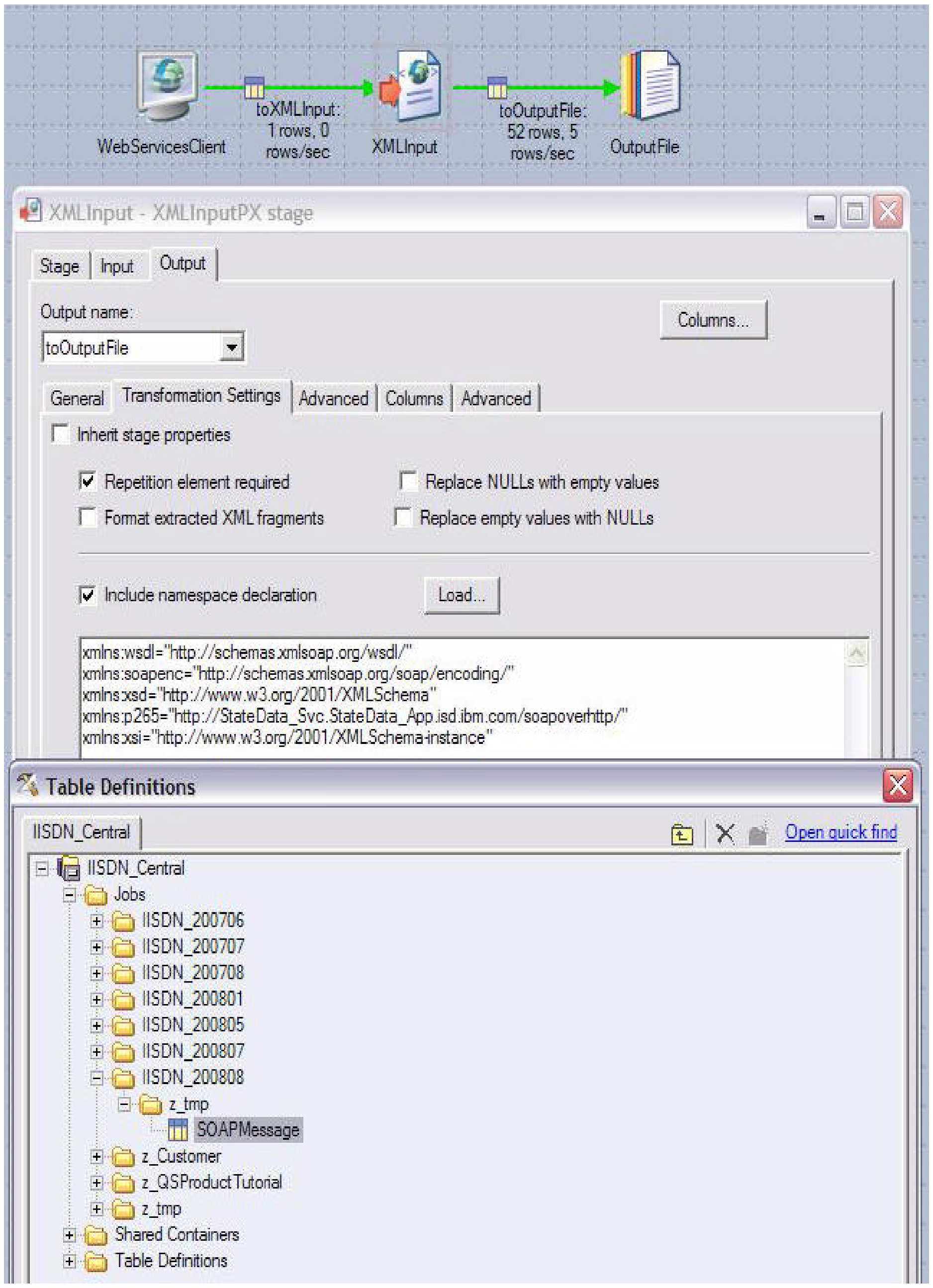
Figure 20-25 XML Input stage, Input TAB -> XML Source.

f. Moving to the Output TAB -> Transformation Settings TAB, Check the control entitled, "Include namespace declaration".

Then Click the button entitled, Load.

Browse to the saved table definition for the SOAP message we made earlier. *We have to include the SOAP message name space declaration, no other namespace declaration will work.* There is an XML tag in the SOAP that will not resolve without this step, and your Job will fail to run.

Example as shown in Figure20-26.



Output name: toOutputFile

- [ ] Inherit stage properties
- [x] Repetition element required
- [ ] Replace NULLs with empty values
- [ ] Format extracted XML fragments
- [ ] Replace empty values with NULLs
- [x] Include namespace declaration

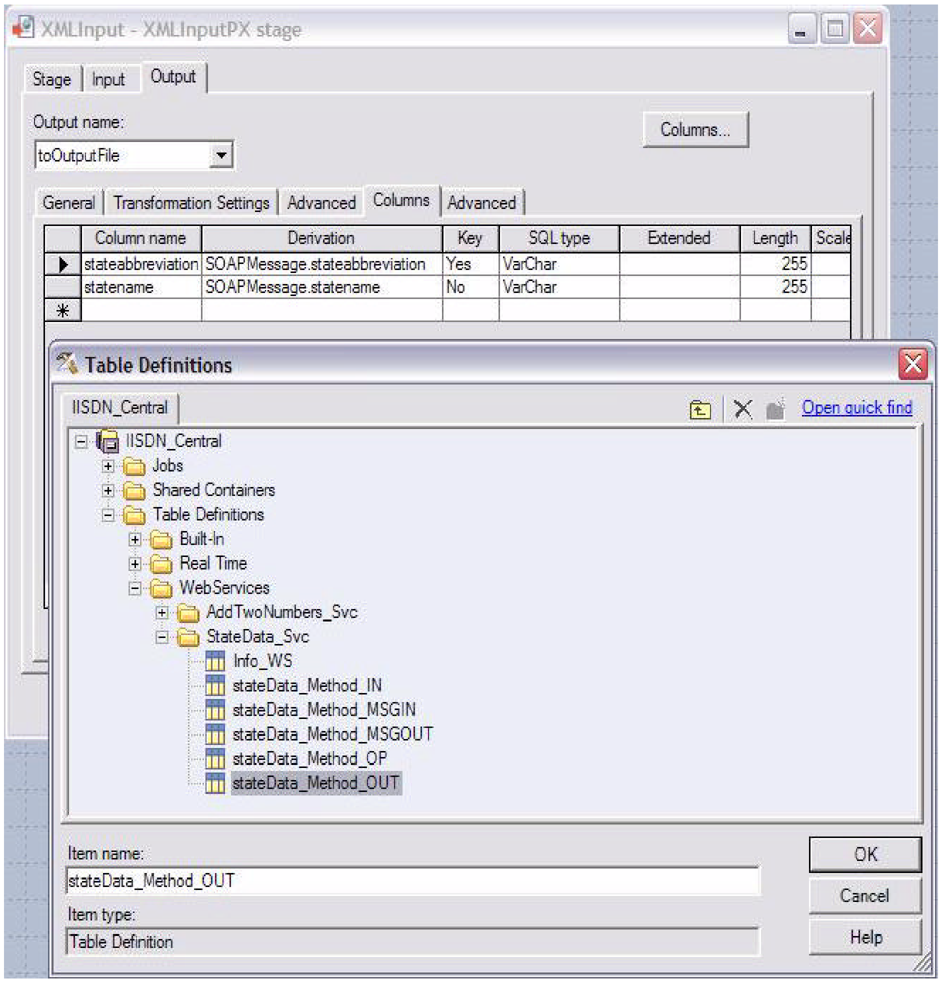
```
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:p265="http://StateData_Svc.StateData_App.isd.ibm.com/soapoverhttp/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
IISDN_Central
  Jobs
    IISDN_200706
    IISDN_200707
    IISDN_200708
    IISDN_200801
    IISDN_200805
    IISDN_200807
    IISDN_200808
      z_tmp
        SOAPMessage
    z_Customer
    z_QSProductTutorial
    z_tmp
  Shared Containers
  Table Definitions
```

Figure 20-26 XML Input stage, Output TAB -> Transformation Settings TAB

g. Moving to the Output TAB -> Columns TAB, Click Load, and browse to the saved table definition for the State data Web service output record format. Example as shown in Figure20-27.

h. Click OK to load this saved table definition, and Click OK to complete editing properties for the XML Input stage.


XMLInput - XMLInputPX stage

Output name: toOutputFile

Column name	Derivation	Key	SQL type	Extended	Length	Scale
stateabbreviation	SOAPMessage.stateabbreviation	Yes	VarChar		255	
statename	SOAPMessage.statename	No	VarChar		255	

Table Definitions

```
IISDN_Central
  Jobs
  Shared Containers
  Table Definitions
    Built-In
    Real Time
    WebServices
      AddTwoNumbers_Svc
      StateData_Svc
        Info_WS
        stateData_Method_IN
        stateData_Method_MSGIN
        stateData_Method_MSGOUT
        stateData_Method_OP
        stateData_Method_OUT
```

Item name: stateData_Method_OUT

Item type: Table Definition

Figure 20-27 XML Input stage, Output TAB -> Columns TAB.

i. Set the properties for the Sequential File stage used for output.

Change the output file name.

Also, the Format properties will be incorrect as previously we wanted to output one continuous, non-delimited line (our XML output file). Now we prefer to output many records, one per line, with proper a delimiter and related.

j. Check that the Environment Variables needed to run a Web Services Transformer and/or Web Service client job are still set.

k. Save, compile and test this Job.

A successful test will appear similar to that as displayed in Figure20-28.

Notice one record is produced from the Web Services Client stage, our SOAP message response from the Web service, and (n) records are produced from the XML Input stage; each of our output data records, the pay load from the SOAP message.

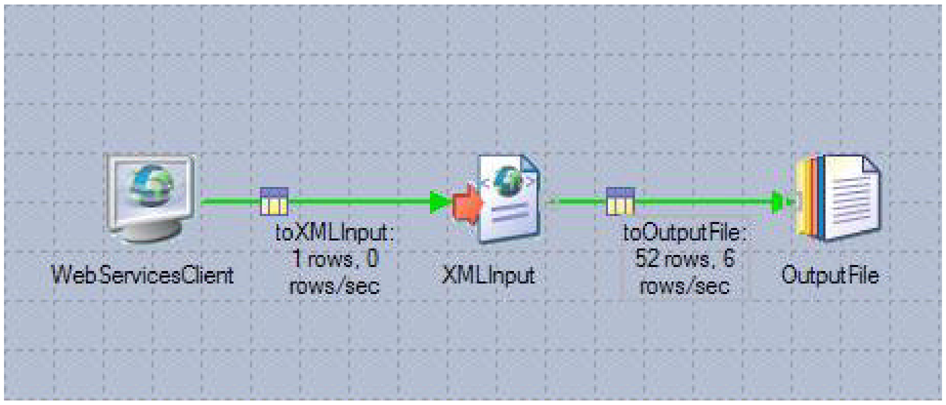

WebServicesClient
toXMLInput: 1 rows, 0 rows/sec
XMLInput
toOutputFile: 52 rows, 6 rows/sec
OutputFile

Figure 20-28   Successful Job from above.

Advanced case: Step 4, Build final Job, Web Services Client

From our work in the section above, our Web Services Client simple use case Job is complete. A more complete use case is displayed in Figure20-29.

Figure20-29 displays a Web Services Client stage in use with a Lookup stage; the Web Services Client stage supplying the Reference data to the Lookup, the valid States data. Since the Web Services Client stage outputs a Standard Link, and the Lookup stage requires a Reference Link, we use a Copy stage in between these two stages.

Using the DataStage/QualityStage Job we completed just above, work to complete the example displayed in Figure20-29.

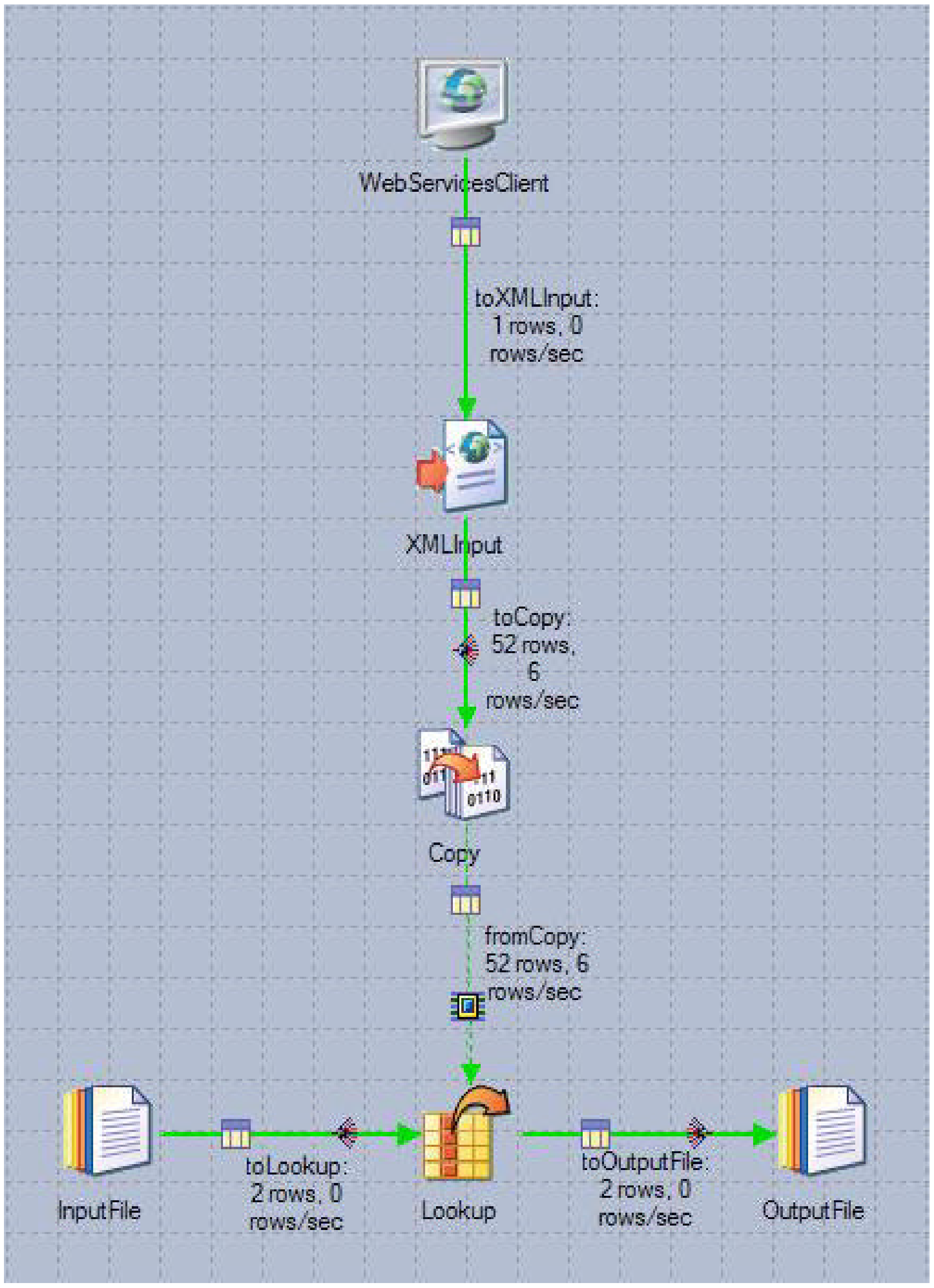

WebServicesClient
toXMLInput: 1 rows, 0 rows/sec
XMLInput
toCopy: 52 rows, 6 rows/sec
Copy
fromCopy: 52 rows, 6 rows/sec
InputFile
toLookup: 2 rows, 0 rows/sec
Lookup
toOutputFile: 2 rows, 0 rows/sec
OutputFile

Figure 20-29 More complete Web Services Client Job.

20.4 Summation

In this document, we reviewed or created:

We examined the (IBM) InfoSphere Information Server (IIS) DataStage/QualityStage Web Services Transformer, and Web Services Client stages. We used the DataStage/QualityStage product capabilities to import and manage XML record and Web service meta data. And, using a very deep and cool set of techniques, we demonstrated how to handle native SOAP response messages and complex return data types.

Persons who help this month.

Beate Porst, Aaron Titus, and Ernie Ostic.

Additional resources:

The IBM WebSphere DataStage (InfoSphere Information Server), Web Services Pack Guide, version 8.1. (Filename, i46dewsv.pdf.)

(IBM) InfoSphere Information Server Developer’s Notebook (IISDN), July 2007 edition; Web Services, Part 1.

(IBM) InfoSphere Information Server Developer’s Notebook (IISDN), June 2007 edition; XML Stages.

Legal statements:

Special attributions: