Visual Introduction to WebService with Tivoli Directory Integrator

(Part 1)

Document Last Updated

July 4th 2007

From TDI Support Team ibmdi@us.ibm.com

Visual Introduction to WebService with Tivoli Directory Integrator

Introduction to WebService

A Web Service (Server) is a software component that is described via WSDL and is capable of being accessed (from a web Service Client) via standard network protocols such as but not limited to SOAP over HTTP.

[adapted from http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm#_Toc198211]

While a Web Service supports four types of patterns of communication between the Server and the Client (as defined in SOAP 1.2), we are going to consider only the Request-Response Message Exchange Pattern, which is depicted below.



As you can see from this diagram, UDDI and WSDL are really optional. If you know the location and the description of a Web service, you don't need to query the UDDI server and fetch the WSDL document. All you need to do is to build a request in SOAP format, and send it via HTTP to the provider.

Note: The Web Service Server is usually referred as Web Service Provider and Web Service Client as Web Service Consumer in standards literature. Here we will use Server and Client instead to make the tutorial compatible with naming conventions used by TDI Components. Pre-requisites :

1. TDI version 6.1.1

Additional Considerations:

Ensure that the port used in the tutorial for the Web Service Server is available for use. Screen shots show Windows directory naming conventions, so please modify to suit your OS.

The Successful creation of a Web Service Server and Client requires successful completion of the following tasks (which are discussed in detail).

Section #1 – Creating a Web Service Server (provider)

Step#1 – Create an AL which will contain a WebService Server component

Step#2 – Create WebService Server Component

Step#3 – Create & Expose WebService Functionality

Step#4 – Create the WSDL file

Step#5 – Configure WebService Server parameters

Step#6 – Define Input & Output Maps for WebService Server

Step#7 – Define WebService Server Logic Flow

Section #2 – Creating a Web Service Client (consumer)

Step#8 – Create an AL which will contain a WebService client component

Step#9 – Configure the WebService Client Component

Step#10 – Add post processing logic (after the response has arrived)

Section #3 – Test the Web Services Server and Client

Step#11 – Start the Web Services Server (Provider)

Step#12 – Test WSDL file from a Browser

Step#13 – Invoke the Web Services Client (to consume the Web Service)

Step#14 – Check the Server logs to see if the results are as expected

Section #1 – Creating a Web Service Server (provider)

Step#1 – Create an AL which will contain a WebService Server component

Step#2 – Create WebService Server Component

This example will use an AxisEasyWSServerConnector to provide the core WebService Server functionality.

Advantages of Connector :

The Axis Easy Web Service Server Connector is part of the TDI Web Services suite. It is a simplified version of the <u>Web Service Receiver Server Connector</u> in that it internally instantiates, configures and uses the <u>AxisSoapToJava</u> and <u>AxisJavaToSoap</u> FCs.

Create a connector which embeds a Webserver: AxisEasyWSServerConnector

Name: WSServer

Select Connector				×
Name	Comment	Template	Include from	
ibmdi.ACTConnector		ibmdi.ACTCon	system	~
ibmdi.ADChangelogv2		ibmdi.ADChan	system	
ibmdi.AssemblyLineConnector		ibmdi.Assembl	system	1
ibmdi.AxisEasyWSServerConnector		ibmdi.AxisEas	system	
ibmdi.BTreeObjectDB		ibmdi.BTree0	system	
ibmdi.CommandLine		ibmdi.Comma	system	
ibmdi.DSMLv2S0APConnector		ibmdi.DSMLv	system	
ibmdi.DSMLv2S0APServerConnector		ibmdi.DSMLv	system	1
ibmdi.DominoChangeDetectionConnector		ibmdi.Domino	system	1
ibmdi.DominoUsersConnector		ibmdi.Domino	system	1
ibmdi.ExchangeChangelog		ibmdi.Exchan	system	
ibmdi.FTPClient		ibmdi.FTPClient	system	
ibmdi.FileSystem		ibmdi.FileSyst	system	
ibmdi.GLAConnector		ibmdi.GLACon	system	
ibmdi.HTTPClient		ibmdi.HTTPCli	system	
ibmdi.HTTPServer		ibmdi.HTTPS	system	
ibmdi.IBM-MQ		ibmdi.IBM-MQ	system	
ibmdi.IBMDirectoryServerChangelog		ibmdi.IBMDire	system	
ibmdi.ITIMAgentConnector		ibmdi.ITIMAg	system	
ibmdi.JDBC		ibmdi.JDBC	system	
ibmdi.JMS		ibmdi.JMS	system	
ibmdi.JMXConnector		ibmdi.JMXCon	system	
ibmdi.JNDI		ibmdi.JNDI	system	
ibmdi.LDAP		ibmdi.LDAP	system	
ibmdi.LDAPServer		ibmdi.LDAPS	system	~
Name: WSServer	Mode: Server		Cancel	

The functionality provided is the same as if you chain and configure these FCs in an AssemblyLine which hosts the <u>Web Service Receiver Server Connector</u>. When using this Connector you forgo the possibility of hooking custom processing before parsing the SOAP request and after serializing the SOAP response. That is, you are tied to the processing and binding provided by Axis, but you gain simplicity of setup and use.

Another limitation of the *AxisEasyWSServerConnector* is that it can only be configured to handle one SOAP operation--it cannot service several SOAP operation requests. But apart from that, the customer is free to use whatever components he likes in the AssemblyLine that contains the Connector.

Limitations :

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_6.1.1/referenceguide14.htm# axiswsrecvconn

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_6.1.1/usersguide51.htm

Step#3 – Create & Expose WebService Functionality

In TDI, AL operations are automatically exposed as the WebService functions and can be invoked remotely. In this example we will create 2 operations using the Operation Tab of the AL.

HebServiceServer							
💤 WebServiceServer 🛛 🔽 🔽 🕅 🔽 🕅 🖓							
🛷 Hooks 🔟 Data Flow 🐘 Config 👪 Operations 🛃 Checkpoint 🥳 Sandbox 🔟 Logging 🚺 Description							
Add New Operation Remove Selected Operation							
Published AssemblyLine Initialize Parameters							
The values of these parameters are configurable in the Config panel of any AssemblyLine Connector or AssemblyLine Function that calls this AssemblyLine. Their values are available to the AssemblyLine through task.getOpEntry() while scripting, or through Expressions as {op-entry.paramName}.							
Name Syntax Information							

There are no Operations associated with the AL when the AL is created. Please NOTE that this behavior is different then AL's imported from TDIv6.0.

You need to create a "Default" operation for the AL to be able to run the AL.

Input Text	
Enter new name	
Default	
OK Can	icel

And this particular operation will not have any Input or Output defined.

🖸 Hooks 🔟 Data Flow 🔐 Config 🎆 Operations 📴 Checkpoint 🥵 Sandbox 🛄 Logging 🕤 Description					
Add New Operation Remove Selected Operation					
Published AssemblyLine Initialize Paramete	ers Default				
🔏 Input Attributes Output Attributes					
🍄 💢 🐵 🔖 Null					
Work Attribute	Work Attribute 😪 Schema				
🗢 💢					
Name Java Class Native Syntax Sample					

In addition to the Default Operation, we need to create an operation called "GetFullName".

Shown below are the two Input Attributes called "FirstName" & "LastName" which are the input parameters that are being passed in from the WebService Client into this WebService Server.

_{ଙ୍କ} ମ୍ମି WebServiceServer			Run mode 🜔 Standard (Run to	completion) 🔽 🕨 😰 💌
🛷 Hooks 🔢 Data Flow 🔒 Config	😹 Operations 📝 Checkpoint	💰 Sandbox 🔟 Logging 🚺	Description	
Add New Operation Remove Selected O	peration			
Published AssemblyLine Initialize Paramete	ers Default GetFullName			
🔏 Input Attributes Output Attributes				
🕼 🌠 👁 🔖 Null				
Work Attribute	🝰 Schema			
	Ø 🕱			
	Add an attribute to the schema	Java Class	Native Syntax	Sample
	FirstName			
	LastName			

The next step in the process of defining the operation is to enumerate the list of attributes which will be passed back to the WebService Client. (This is defined in the "Output Attributes" Tab of the Operation).

က်မှာ WebServiceServer		Run mode 🕨 Standard (Run to d	completion) 💌 🕨 😰 💌	
🔊 Hooks 🔟 Data Flow 鼠 Config	👪 Operations 🛃 Checkpoint 🥵 Sandbox 🛄 Logging 🚺	Description		
Add New Operation Remove Selected Op	peration			
Published AssemblyLine Initialize Paramete	rs Default GetFullName			
🔗 Input Attributes Output Attributes				
🕼 🎘 👁 🔖 Null				
Work Attribute	🔗 Schema			
🖉 💢				
	Add an attribute to the schema Java Class	Native Syntax	Sample	
	FullName			

Step#4 – Create the WSDL file

On the config Tab of the WebService Server Connector (AxisEasyWSServerConnector) enter the WSDL File name and the URL that the WebService Client will be communicating to. Example, if the WebService Server will be running on a machine called "tdihawk" and the port is 1998, you would define as <u>http://tdihawk:1998/</u>

WSDL Output to Filename	c:\temp\WSv611.wsdl
Web Service provider URL	http://localhost:1998/
	Generate WSDL

Ensure you press the "Generate WSDL", and successfully generate the WSDL file.

Step#5 – Configure WebService Server parameters

- a. Configure Port
 b. Enable "Tag Op-Entry"
 c. Define the path to the WSDL file that you previously generated in Step4.
 d. Choose a SOAP Operation

🔒 Config 🕵 Input Map 👔 Output Ma	p 📷 Link Criteria 🔊 Hooks 🛕 Delta 🚺 Description 💋 Connection Errors 🐚 Pool	
Initialize at startup	✓	
Connection Parser		
Axis Easy Web Service Server	Connector	
TCP Port	1998	
Connection Backlog		
WSDL File	c:\temp\WSv611.wsdl	Select
SOAP Operation	WebServiceServer	Operations
Complex Types		
Tag Op-Entry		
Use SSL		
Require Client Authentication		
Auth Realm	IBM Tivoli Directory Integrator	
Use HTTP Basic Authentication		
Comment		
Detailed Log		
		Help
WSDL Output to Filename	c:\temp\WSv611.wsdl	
Web Service provider URL	http://localhost:1998/	
		Generate WSDL

IBM Tivoli Directory Interview 100 (1998)				🗖 🗖 🔀		
File Object Store Remote	<u>Window Tools Help</u>					
🖻 🖥 🖻 📡 🔝						
webservice101.xml	WebServiceServer					
Config AssemblyLines WebServiceServer	WebServiceServer		Run mode 🕞 Standard (Run to	completion) 🚽 🕨 😰 💌		
Functions	W Hooks No Data How St U	onlig My Uperations 🔐 Checkpoint (Sandbox 🛄 Logging 😈 Description			
AttributeMaps	WSServer Flow	Mode Server V State Enabled V Delta Inherit from: ibmdi.AxisEasyWSServerConnector				
		🚯 Config 🕵 Input Map 🔹 Outp	ut Map 📷 Link Criteria 🛷 Hooks 🗥 Delta 🚺 Description	Connection Errors here Pool		
Java Libraries		Initialize at startup	~			
		Connection Parser				
		Axis Easy Web Service Ser	ver Connector	<u>^</u>		
		TCP Port	1998			
		Connection Backlog				
		WSDL File	c:\temp\W/Sv611.wsdl	Select		
		SOAP Operation		Operations		
		Select 🔀		<u></u>		
m - 92	57 X m A	Select Web Service Operation		Ξ		
	Work Entry Operations	webserviceserver V		~		
Resources Packages	Name Source	OK Cancel				
dinorary		Use SSL				
		Require Client Authentication				
		Auth Realm	IBM Tivoli Directory Integrator			
		Use HTTP Basic Authentication				
		Comment		a _		
				Ξ		
				~		
		Detailed Log				
				Laborit from forward		
° 🚥 💥				mmenic notic (parent)		
	la.	1				

Step#6 – Define Input & Output Maps for WebService Server

Input Map: add "*"

Hooks 10 Data Flow 10 Config 10 I	Dperations 📝 Checkpoint (💰 Sandbox 🔲 Loggin	g 🚺 Description		
W Hooks Image: Config Image:					
Work Att	Select Enter name for new a OK Ca	Name host Name host word name bjArray dResource on t uest ponse wsdlRequested	Java Class String String String String String String String String String String String	Native Syntax	Sample

Output Map: The attributes responseContentType and responseObjArray are used to define the contents which will be sent back to the WebService Client. The additional three attributes, soapFault, soapResponse, and wsdlRequested, provide default responses needed for the WebService Server

हरू WebServiceServer			Run mode 🜔 S	itandard (Run to comple	etion) 🔽 🕨 😰
💞 Hooks 🔟 Data Flow 🐘 Config 🖓 Operations 🗹 Checkpoint 🥳 Sandbox 🔟 Logging 🚺 Description					
E	Mode Server V State Enabled V Delta Inherit from: ibmdi.AxisEasyWSServerConnector				
💦 Config 🚳 Input Map 😰 Output Map 📷 Link Criteria 🔊 Hooks 🖾 Delta 🚺 Description 💕 Connection Errors					Connection Errors 🛯 🐚 Pool
	Connector Attribute	😂 🕱 🐌 🕨 🛠	K 💋 📎		
	responseContentType	Name	Java Class	Native Syntax	Sample
	soapFault	http.credentialsValid	Boolean		
	soapResponse	responseContentType	String		
	wsdlRequested	responseUbjArray	String		
		soapResponse	String		
		wsdlRequested	String		

Step#7 – Define WebService Server Logic Flow

Click on Flow, create Script Component. Name: ProcessInputParameters



```
And Add the Following Script to the Script Component
```

Add a Switch Component, name: ChooseOperation

Choose "AssemblyLine Operations" Choose "Add Case Components" - will choose the 2 operations created.

Note: if click again, get a message already added.





Note – You can add additional Case Components by clicking on "Add Case components" button during solution development.

In the ChooseOperation_GetFullname, click to add a script component.

Add script to setup the responseObjArray.



The Script that needs to be added to the response Object is ..

var obj = new java.lang.Object(); var responseObjArray = java.lang.reflect.Array.newInstance(obj.getClass(), 1);

// set up the STRING response
responseObjArray[0] = new java.lang.String("This is the String Returned Back");

work.setAttribute("responseObjArray", responseObjArray);

And Lastly...

```
Add script component to end of flow, name: DumpWork
main.logmsg(" *** " + work);
```

Section #2 - Creating a Web Service Client (consumer)

Step#8 – Create an AL which will contain a WebService Client component

Add a new AL named: WebServiceClient



Step#9 – Create WebService Client Component

Add a Function Component of type AxisEasyInvokeSoapWS

The Axis EasyInvokeSoapWS Function Component (FC) is part of the TDI Web Services suite.

This is a "simplified" web service invocation component: it is a stand-alone FC with its own Config screen, but internally instantiates, configures and uses the following three FCs: <u>AxisJavaToSoap</u>, <u>InvokeSoapWS</u> and <u>AxisSoapToJava</u>.

The functionality provided is the same as if you chain and configure these three FCs in an AssemblyLine. When using this FC you lose the possibility of hook custom processing, that is, you are tied to the processing and binding provided by Axis. However, you gain simplicity of setup and use.

 $http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_6.1.1/referenceguide119.htm\ \#axiseasyinvokesoapwsfc$

IBM Tivoli Directory Integrate								\mathbf{X}
File Object Store Remote Window	w Tools Help							
WebServiceClient								
WebServiceClient					Run moo	de 😤 Step (Break on error)	?	
🔊 Hooks 🔟 Data Flow 🔒 Confi	ig 🔼 Operations	🕼 Checknoint 🥵 Sandbox 👖	Lloadina 🔒 De	scription				-1
Feeds		Select Function						
Elow		Name	Comment	Template	Include from			- 11
		ibmdi Assemblul ineEC		ihmdi Assemblul i	sustem			- 18
		ibmdi.AxisEasulnyokeSoapWS		ibmdi.AxisEasvIn	system			
		ibmdi.AxisJavaToSoap		ibmdi.AxisJavaT	system			
		ibmdi.AxisSoapToJava		ibmdi.AxisSoapT	system			
		ibmdi.CBEGeneratorFC		ibmdi.CBEGener	system			
		ibmdi.CastorJavaToXML		ibmdi.CastorJava	. system			- 11
		ibmdi.CastorXMLToJava		ibmdi.CastorXML	. system			
		ibmdi.ComplexTypesGenerator		ibmdi.ComplexTy	. system			
		ibmdi.EmfSdoTcXml		ibmdi.EmfSdoTo	system			
		ibmdi.Emf⊠mlToSdo		ibmdi.EmfXmlTo	system			- 11
		ibmdi.InvokeSoapWS		ibmdi.InvokeSoa	. system			
		ibmdi.JavaClassFC		ibmdi.JavaClassFC	; system			
		ibmdi.MemQueueFC		ibmdi.MemQueu	system			- 11
		ibmdi.ParserFC		ibmdi.ParserFC	system			
		ibmdi.HemoteCmdLineFC		ibmdi.HemoteCm	system			
😼 💥 🚥 🛕		ibmdi.SapH3HtcFC		ibmdi.SapH3HtcH0	. system			
		ibmdi. ScriptedHU		Ibmdi. ScriptedFU	system			
Work Entry Operations		ibmdi.SendEMaliFC		ibmdi. SendE Mall	system			
Name Source		ibmdi aQSCommandi inoEC		ibmdi aQSComm	system			
		ibilidi.203CommandLinerC		Ibindi.203Comm	system			
		Name: CallWS	OK	Cancel				
p							 _	

This Function Component (FC) provides a relatively simple way of invoking SOAP over HTTP web services.

This is how the communication flows:

Web service client <-> AxisEasyInvokeSoapWS FC <-> org.apache.axis.client.Call <-> Web service

The Function Component takes either an Object Array or an Entry Type:

Object[] -> AxisEasyInvokeSoapWS FC -> Object[]

or

Entry -> AxisEasyInvokeSoapWS FC -> Entry

In our Example, we will use the Entry Type to define the Output Attributes of the Function Component. Note the Output of the Function Component is what is SENT to the Web Service Server, and the returned values come in the Input Map.

Step#10 – Configure WebService Client Component

Set the WSDL File, SOAP Operation, and the Operation Parameters (as shown in the example below)

IBM Tivoli Directory Integra	tor			
<u>File Object Store Remote Wind</u>	ow <u>T</u> ools <u>H</u> elp			
1				
🖌 🚽 WebServiceClient				
ക്രീ WebServiceClient			Run mode 춛 Step (Break on error) 💌	
👌 Hooks 脑 Data Flow 🔒 Con	nfig 🞊 Operations 👩 Checkp	oint 🥵 Sandbox 🛄 Logging 🚹 Description		
Feeds	State Enabled 🛩 Inherit f	rom: ibmdi.AxisEasyInvokeSoapWS		Help
CallwS	🐘 Config 🛃 Input Map 👔	0utput Map 🛷 Hooks 👔 Description		
	Initialize at startup	×		
	Axis Easy Web Service	Invoke		
	WSDL URL	c:\temp\WSv611.wsdl		
	SOAP Operation	GetFullName	Operations	
	Login username			
	Login password			
	Complex Types			
		String1 String?	<u>×</u>	
Work Entry Operations	Uperation Parameters			
Name Source	Detailed Log			
	Lomment			

Define the Input Map

(Basically drag and drop the available Attribute called "return" on the Work side)

B WebServiceClient	
<mark>, m</mark> ⊉ WebServiceClient	
🔊 Hooks 10 Data Flow 🔒 C	nfig 😹 Operations 🗹 Checkpoint 💰 Sandbox 🔟 Logging 🚹 Description
Feeds	State Enabled Inherit from: ibmdi.AxisE asylnvokeSoapWS Image: State Config Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State

Define the Values for the Output Map

(The Output Map should have the two Attributes that are expected by the Web Service Server)

HebServiceClient		
<mark>₩</mark> WebServiceClient		
🔊 Hooks 🌆 Data Flow 🔒 Co	onfig 😹 Operations 🛃 Checkpoint 👔	🕵 Sandbox 🚺 Logging 🚺 Description
Feeds General Flow CalW/S	State Enabled V Inherit from: i	ibmdi.AxisEasyInvokeSoapWS at Map & Hooks Description Schema Schema Name Java Class

And set the Value for each the Output Map Attributes

(The example shown below uses literal values)

₩ebServiceClient	
👌 Hooks 😥 Data Flow	🛿 Config 🦓 Operations 👩 Checkpoint 🥳 Sandbox 🛄 Logging 🚺 Description
Feeds	State Enabled Inherit from: ibmdi.AxisE asyInvokeSoapWS Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Image: State Imag
	Image: String 2 Image: String 2

എ WebServiceClient	
🔊 Hooks 🕫 Data Flow 🚦	🕽 Config 👪 Operations 🛃 Checkpoint 🥵 Sandbox 🛄 Logging 👔 Description
Flow CallwS	State Enabled Inherit from: ibmdi.AxisE asylnvokeSoapWS Config Input Map Hooks Description Image: String 1 Image: String 2 Image: String 2 Image: String 2 String 2 Image: String 2 Image: String 2 Image: String 2 Image: String 2 String 2 Image: String 2 String 2 Image: String 2

Section #3 – Test the Web Services Server and Client

Step#11 – Start the Web Services Server (Provider)



Please Note: If you get any Error in starting the Server Assembline Line, you need to fix this before proceeding further. If you get an error like the one shown below, ensure that you have an AssemblyLine Operation called Default (not "default").

14:56:52	CTGDIS7851
14:56:53	CTGDIS040I Loading configuration from stdin.
14:56:53	CTGDIS028I Starting AssemblyLine AssemblyLines/WebServiceServer.
14:56:53	CTGDIS015E Error while auto starting AssemblyLines.
java.lang	.Exception: CTGDIS184E This AssemblyLine was invoked with operation '(null)', which does not match any of the
defined of	perations for this AssemblyLine.
	at com.ibm.di.server.Log.exception(Unknown Source)
	at com.ibm.di.server.AssemblyLine.logException(Unknown Source)
	at com.ibm.di.server.AssemblyLine. <init>(Unknown Source)</init>
	at com.ibm.di.server.RS.startAL(Unknown Source)
	at com.ibm.di.server.RS.startAL(Unknown Source)
	at com.ibm.di.server.RS.runServer(Unknown Source)
	at com.ibm.di.server.RS.run(Unknown Source)
CTGDIS015	E Error while auto starting AssemblyLines.: java.lang.Exception: CTGDIS184E This AssemblyLine was invoked with
operation	'(null)', which does not match any of the defined operations for this AssemblyLine.
14:56:55	CTGDIS037I Server terminates because only main thread is left.
14:56:55	CTGDIS174I Config Instance C:\IBM\TDI611 Solutions\webservice101.xml exited with status 0.
14:56:55	CTGDIS228I Unregister server: C:\IBM\TDI611 Solutions\webservice101.xml.
14:56:55	CTGDIS627I TDI Shutdown.
*******	****
Process es	xit code = 0

Step#12 – Test WSDL file from a Browser

Use a browser to go to http://webservice_server_url?WSDL (check the webservice_server_url defined in Section 1 of Step#4) In this example - Bring up a browser, with url: <u>http://localhost:1998/?WSDL</u>

◎ http:////h// 000/20//CDL	
Intep://iocalinosci.1998/rwsbit - Microsoft Internet Explorer	
Ele Edit View Favorites Tools Help	~~
🕜 Back 🔹 🕥 🐇 😰 🏠 🔎 Search 🤺 Favorites 🚱 🔗 🚽 🍃 📄 😵 😐 🥸	
Address 🙋 http://localhost: 1998/?WSDL	🖌 ᠫ Go
i Links 💩 IBM Business Transformation Homepage 🛛 👸 IBM Standard Software Installer 🛛 👸 IT Help Central 🖉 Join World Community Grid	»
<pre>classical control of the contro</pre>	
z/wedt-bindings	
Cocal Intra	net 📑

Step#13 – Invoke the Web Services Client (to consume the Web Service)

It should provide a log like this..as it runs to completion..

```
15:46:36 [CallWS] CTGDIS504I *Result of attribute mapping*
15:46:36 [CallWS] CTGDIS505I The 'conn' object
15:46:36 [CallWS] CTGDIS003I *** Start dumping Entry
15:46:36 Operation: generic
15:46:36 Entry attributes:
15:46:36
                  String2 (replace): 'DemoLastName'
15:46:36
                  Stringl (replace): 'DemoFirstName'
15:46:36 [CallWS] CTGDIS004I *** Finished dumping Entry
15:46:36 [CallWS] CTGDIS506I The 'work' object
15:46:36 [CallWS] CTGDIS003I *** Start dumping Entry
15:46:36 Operation: generic
15:46:36 Entry attributes:
15:46:36 [CallWS] CTGDIS004I *** Finished dumping Entry
15:46:36 [CallWS] CTGDIZ613I About to call web service...
15:46:38 [CallWS] CTGDIZ601I Web service called successfully.
15:46:38 [CallWS] CTGDIZ614I SOAP response Java Object: This is the String Returned Back.
15:46:38 [CallWS] CTGDIS504I *Result of attribute mapping*
15:46:38 [CallWS] CTGDIS505I The 'conn' object
15:46:38 [CallWS] CTGDIS003I *** Start dumping Entry
15:46:38 Operation: generic
15:46:38 Entry attributes:
15:46:38
                  return (replace): 'This is the String Returned Back'
15:46:38 [CallWS] CTGDIS004I *** Finished dumping Entry
15:46:38 [CallWS] CTGDIS506I The 'work' object
15:46:38 [CallWS] CTGDIS003I *** Start dumping Entry
15:46:38 Operation: generic
15:46:38 Entry attributes:
15:46:38
                                        'This is the String Returned Back'
              return (replace):
15:46:38 [CallWS] CTGDIS004I *** Finished dumping Entry
15:46:38 *** [return:This is the String Returned Back]
15:46:38 CTGDIS088I Finished iterating.
15:46:38 CTGDIS100I Printing the Connector statistics.
15:46:38 [CallWS] CallReply:1
15:46:38 [DumpWork] (No statistics for script component.)
15:46:38 CTGDIS104I Total: CallReply:1.
```

And on the Server Side,

```
15:46:38 [WSServer] Get:1, reply:1
15:46:38 [ProcessInputParameters] (No statistics for script component.)
15:46:38 [ChooseOperation] Switches:1
15:46:38 [ChooseOperation_Default] Match:0, NoMatch:1
15:46:38 [ChooseOperation_GetFullName] Match:1, NoMatch:0
15:46:38 [ProcessGetFullName] (No statistics for script component.)
15:46:38 [DumpWork] (No statistics for script component.)
15:46:38 CTGDIS104I Total: Get:1, reply:1, Switches:1.
15:46:38 CTGDIS101I Finished printing the Connector statistics.
15:46:38 CTGDIS080I Terminated successfully (0 errors).
15:46:38 CTGDIS079I AssemblyLine AssemblyLines/WebServiceServer terminated successfully.
```

Step#14 – Check the Server logs to see if the results are as expected

Enable Detailed Long on the STEP 5 – for the Web Services Server Connector, then the Server logs would be similar to this..

15:46:38 [WSServer] CTGDIS506I The 'work' object
15:46:38 [WSServer] CTGDIS003I *** Start dumping Entry
15:46:38 Operation: generic
15:46:38 Entry attributes:
15:46:38 soapAction (replace): 'ns:WebServiceServer_thisNamespace#GetFullName'
15:46:38 host (replace): 'localhost:1998'
15:46:38 requestedResource (replace): ''
15:46:38 soapRequest (replace): ' xml version="1.0" encoding="UTF-8"? <soapenv:envelope< td=""></soapenv:envelope<>
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:body><ns1:getfullname< td=""></ns1:getfullname<></soapenv:body>
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:nsl="ns:WebServiceServer_thisNamespace"> <firstname< td=""></firstname<>
xsi:type="xsd:string">DemoFirstName <lastname< td=""></lastname<>
xsi:type="xsd:string">DemoLastName'
15:46:38 requestObjArray (replace): '[Ljava.lang.Object;@2040204'
15:46:38 responseObjArray (replace): '[Ljava.lang.Object;@ea20ea2'
15:46:38 wsdlRequested (replace): 'false'
15:46:38 [WSServer] CTGDIS004I *** Finished dumping Entry
15:46:38 [WSServer] CTGDI2008I Response Object values: [This is the String Returned Back].
15:46:38 [WSServer] CTGDIZ127I EventHandler response HTTP status code set to '200 OK'.
15:46:38 [WSServer] CTGDIZ128I EventHandler response content type set to 'text/xml;charset=UTF-8'.
15:46:38 [WSServer] CTGDIZ129I EventHandler response content length set to '510'.
15:46:38 [WSServer] CTGDIZ130I EventHandler response set to ' xml version="1.0" encoding="UTF-8"? <soapenv:envelope< td=""></soapenv:envelope<>
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:body></soapenv:body>
<nsl:getfullnameresponse xmlns:nsl="ns:WebServiceServer_thisNamespace"></nsl:getfullnameresponse>
<nsl:arg0 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:string">This is the String Returned Back</nsl:arg0>
'.
15:46:38 CTGDIS088I Finished iterating.
15:46:38 CTGDIS315I AssemblyLine worker thread: AssemblyLines/WebServiceServer.WSServer.751578316 is stopped.
15:46:38 CTGDIS100I Printing the Connector statistics.
15:46:38 [WSServer] Get:1, reply:1

The End.

For any question about this document, contact ibmdi@us.ibm.com

Useful TDI Links

On-line Documentation

Fix Packs

<u>ITDI Wiki</u>

News Group

Remote Assistance Tool

Internal Site

Internal News Group