IMPLEMENTING A SOAP BASED WEB SERVICE USING DELPHI: A COMPLETE TUTORIAL[1]

Rico R. Pamplona[2]

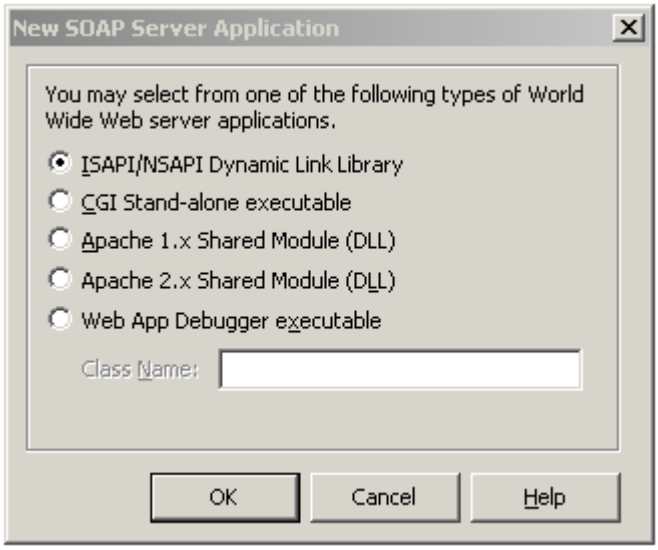
INTRODUCTION

The Simple Object Access Protocol or SOAP (*SOAP 1.1*, 2003) has become a popular standard protocol for TCP/IP communications. Web services developed using SOAP allow intercommunication of different platforms and development tools. Hence, popular tools have integrated SOAP as one of the basic features. Delphi for one has a built-in SOAP feature for the development of both the service and the client (Delphi, 2005).

This document outlines the step by step procedure for building a web service using Delphi. Similarly, the procedure for consuming the developed web service is also given.

CREATING THE SERVICE

Open the Delphi IDE then on the menu, click File > New > Other (Select SOAP Server Application from WebServices Tab), then choose deployment type ISAPI/NSAPI DLL, CGI Stand-alone executable, Apache Shared Module (depending on how you want your service to be deployed and what kind of web server you are using).



For the sake of example, we choose ISAPI for deployment to IIS. Click YES to create interface for SOAP Module > Service Name (e.g. MyCar).

Let's choose a Per Request on the 'Choose Service Activation Model' part. We want the application to create a new instance of your implementation class in response to each request it receives. That instance is

---

[1] *Submitted contribution to XMETHODS (http://www.xmethods.org)*

[2] *Senior Software Engineer, TSS-AVSOFT, 2724 Kilgore Rd, Rancho Cordova, CA 95670*

freed after the request is handled. One may choose the Global Object if he wants the application to create a single instance of your implementation class, which is used to handle all requests.

Lets create a web service that allows input of some variables (mpg, tank capacity etc.) then returns full tank miles and gas expense. In this example, we'll create two SOAP calls: GetFullTankMiles and GetGasExpense.

Let's type in the codes for the *.Int.pas and the *Impl.pas in the Delphi IDE.

//Type the codes of the SOAP methods in *Intf.pas (CarWebServiceIntf.pas in this example)

```
unit CarWebServiceIntf;

interface

uses InvokeRegistry, Types, XSBuiltIns;

type

  TCarInfo = class(TRemotable)
  private
    FModel: AnsiString;
    FYear: Integer;
    FMilesPerGallon: Double;
    FTankCapacity: Double;
  published
    property Model: AnsiString read FModel write FModel;
    property Year: Integer read FYear write FYear;
    property MilesPerGallon: Double read FMilesPerGallon write FMilesPerGallon;
    property TankCapacity: Double read FTankCapacity write FTankCapacity;
  end;

  { Invokable interfaces must derive from IInvokable }
  ICarWebService = interface(IInvokable)
  ['{87D671A3-9CDD-4869-9E5F-0D30CE0357F7}']

    { Methods of Invokable interface must not use the default }
    { calling convention; stdcall is recommended }
    function GetFullTankMiles(const aCar: TCarInfo): Double; stdcall;
    function GetGasExpense(const aCar: TCarInfo; const aMiles:Double; const
aPricePerGallon:Double ): Double; stdcall;
  end;

implementation

initialization
  { Invokable interfaces must be registered }
  InvRegistry.RegisterInterface(TypeInfo(ICarWebService));

end.
```

// Type the codes of the SOAP methods in *Impl.pas (CarWebServiceImpl.pas in this example)

```
{ Invokable implementation File for TCarWebService which implements ICarWebService }
unit CarWebServiceImpl;

interface

uses InvokeRegistry, Types, XSBuiltIns, CarWebServiceIntf;

type
  { TCarWebService }
  TCarWebService = class(TInvokableClass, ICarWebService)
  public
    function GetFullTankMiles(const aCar: TCarInfo): Double; stdcall;
    function GetGasExpense(const aCar: TCarInfo; const aMiles:Double; const
aPricePerGallon:Double): Double; stdcall;
  end;

implementation

function TCarWebService.GetFullTankMiles(const aCar: TCarInfo): Double; stdcall;
begin
  Result :=aCar.MilesPerGallon * aCar.TankCapacity;
end;

function TCarWebService.GetGasExpense(const aCar: TCarInfo; const aMiles:Double;
const aPricePerGallon:Double ): Double; stdcall;
begin
   Result := aPricePerGallon * aMiles / aCar.MilesPerGallon;
end;

initialization
  { Invokable classes must be registered }
  InvRegistry.RegisterInvokableClass(TCarWebService);
end.
```

Save and compile the project. Deploy as isapi in IIS (or to your favorite web server). If the web service was deployed in a directory called isapi with a domain of http://tstsvr.achworks.com, then the WSDL would be http://tstsvr.achworks.com/isapi/carwebservice.dll/wsdl/ICarWebService and would look similar to this when accessed using a browser:

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    name="ICarWebServiceservice" targetNamespace="http://tempuri.org/"
    xmlns:tns="http://tempuri.org/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:ns1="urn:CarWebServiceIntf">
  - <types>
    - <xs:schema targetNamespace="urn:CarWebServiceIntf"
        xmlns="urn:CarWebServiceIntf">
      - <xs:complexType name="TCarInfo">
        - <xs:sequence>
            <xs:element name="Model" type="xs:string" />
            <xs:element name="Year" type="xs:int" />
```

```
            <xs:element name="MilesPerGallon" type="xs:double" />
            <xs:element name="TankCapacity" type="xs:double" />
          </xs:sequence>
        </xs:complexType>
      </xs:schema>
    </types>
- <message name="GetFullTankMiles0Request">
      <part name="aCar" type="ns1:TCarInfo" />
    </message>
- <message name="GetFullTankMiles0Response">
      <part name="return" type="xs:double" />
    </message>
- <message name="GetGasExpense1Request">
      <part name="aCar" type="ns1:TCarInfo" />
      <part name="aMiles" type="xs:double" />
      <part name="aPricePerGallon" type="xs:double" />
    </message>
- <message name="GetGasExpense1Response">
      <part name="return" type="xs:double" />
    </message>
- <portType name="ICarWebService">
  - <operation name="GetFullTankMiles">
        <input message="tns:GetFullTankMiles0Request" />
        <output message="tns:GetFullTankMiles0Response" />
      </operation>
  - <operation name="GetGasExpense">
        <input message="tns:GetGasExpense1Request" />
        <output message="tns:GetGasExpense1Response" />
      </operation>
    </portType>
- <binding name="ICarWebServicebinding" type="tns:ICarWebService">
      <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http" />
  - <operation name="GetFullTankMiles">
        <soap:operation soapAction="urn:CarWebServiceIntf-
          ICarWebService#GetFullTankMiles" style="rpc" />
    - <input message="tns:GetFullTankMiles0Request">
          <soap:body use="encoded"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:CarWebServiceIntf-ICarWebService" />
        </input>
    - <output message="tns:GetFullTankMiles0Response">
          <soap:body use="encoded"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:CarWebServiceIntf-ICarWebService" />
        </output>
      </operation>
  - <operation name="GetGasExpense">
        <soap:operation soapAction="urn:CarWebServiceIntf-
          ICarWebService#GetGasExpense" style="rpc" />
```

```
      - <input message="tns:GetGasExpense1Request">
          <soap:body use="encoded"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:CarWebServiceIntf-ICarWebService" />
        </input>
      - <output message="tns:GetGasExpense1Response">
          <soap:body use="encoded"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:CarWebServiceIntf-ICarWebService" />
        </output>
      </operation>
    </binding>
  - <service name="ICarWebServiceservice">
    - <port name="ICarWebServicePort" binding="tns:ICarWebServicebinding">
        <soap:address
          location="http://tstsvr.achworks.com/isapi/carwebservice.dll/soap
          /ICarWebService" />
      </port>
    </service>
  </definitions>
```

CONSUMING THE WEB SERVICE IN DELPHI

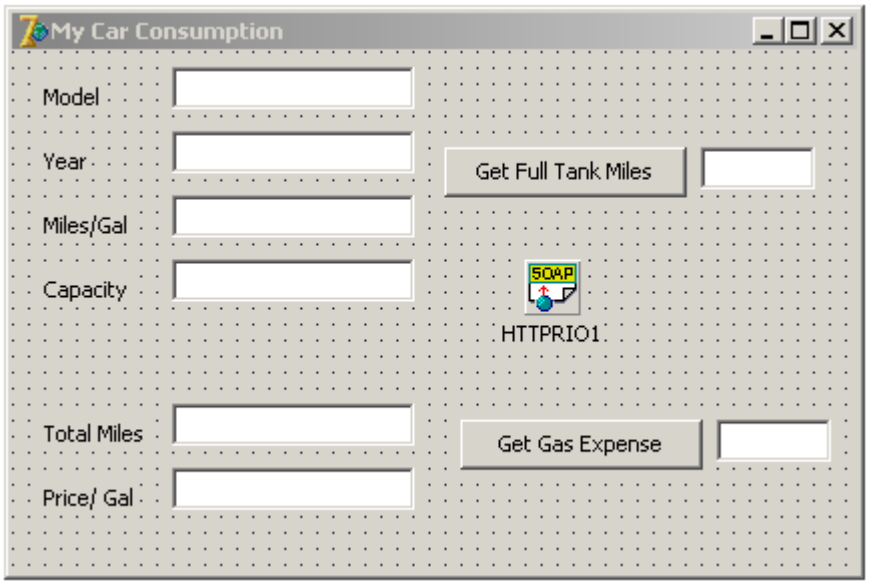
Now, we are ready to consume the web service we have created. The first step is to use the Delphi wizard utility for importing the WSDL or XML schema. This utility generates all the interface and class definitions you need for calling on the web service. Let's assume that the web service was deployed in a directory called isapi with a domain of http://tstsvr.achworks.com, then the WSDL would be http://tstsvr.achworks.com/isapi/carwebservice.dll/wsdl/ICarWebService . In the Delphi menu, click File > New > Other (Select WSDL Importer from the WebServices Tab) and put this as the location of the WSDL file or URL. Save the generated file in a convenient directoy. In this example, the file is saved as ICarWebService1.pas. The file will look something like this:

```
// ************************************************************************ //
// The types declared in this file were generated from data read from the
// WSDL File described below:
// WSDL     : http://tstsvr.achworks.com/isapi/carwebservice.dll/wsdl/ICarWebService
// Encoding : utf-8
// Version  : 1.0
// (5/12/2006 4:09:41 PM - 1.33.2.5)
// ************************************************************************ //

unit ICarWebService1;

interface

uses InvokeRegistry, SOAPHTTPClient, Types, XSBuiltIns;

type

  // ************************************************************************ //
  // The following types, referred to in the WSDL document are not being represented
  // in this file. They are either aliases[@] of other types represented or were
referred
```

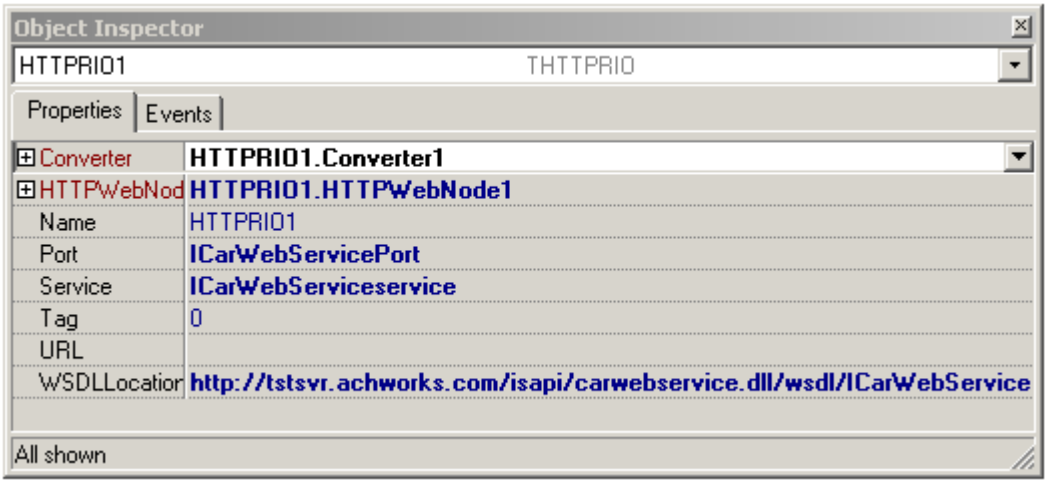
```
  // to but never[!] declared in the document. The types from the latter category
  // typically map to predefined/known XML or Borland types; however, they could also
  // indicate incorrect WSDL documents that failed to declare or import a schema
type.
  // ************************************************************************ //
  // !:string          - "http://www.w3.org/2001/XMLSchema"
  // !:int             - "http://www.w3.org/2001/XMLSchema"
  // !:double          - "http://www.w3.org/2001/XMLSchema"

  TCarInfo             = class;                { "urn:CarWebServiceIntf" }



  // ************************************************************************ //
  // Namespace : urn:CarWebServiceIntf
  // ************************************************************************ //
  TCarInfo = class(TRemotable)
  private
    FModel: WideString;
    FYear: Integer;
    FMilesPerGallon: Double;
    FTankCapacity: Double;
  published
    property Model: WideString read FModel write FModel;
    property Year: Integer read FYear write FYear;
    property MilesPerGallon: Double read FMilesPerGallon write FMilesPerGallon;
    property TankCapacity: Double read FTankCapacity write FTankCapacity;
  end;


  // ************************************************************************ //
  // Namespace : urn:CarWebServiceIntf-ICarWebService
  // soapAction: urn:CarWebServiceIntf-ICarWebService#%operationName%
  // transport : http://schemas.xmlsoap.org/soap/http
  // style     : rpc
  // binding   : ICarWebServicebinding
  // service   : ICarWebServiceservice
  // port      : ICarWebServicePort
  // URL       :
http://tstsvr.achworks.com/isapi/carwebservice.dll/soap/ICarWebService
  // ************************************************************************ //
  ICarWebService = interface(IInvokable)
  ['{0E5D16BA-72E9-17CD-6E61-061B2C98DE8B}']
    function  GetFullTankMiles(const aCar: TCarInfo): Double; stdcall;
    function  GetGasExpense(const aCar: TCarInfo; const aMiles: Double; const
aPricePerGallon: Double): Double; stdcall;
  end;

function GetICarWebService(UseWSDL: Boolean=System.False; Addr: string=''; HTTPRIO:
THTTPRIO = nil): ICarWebService;


implementation

function GetICarWebService(UseWSDL: Boolean; Addr: string; HTTPRIO: THTTPRIO):
ICarWebService;
const
  defWSDL = 'http://tstsvr.achworks.com/isapi/carwebservice.dll/wsdl/ICarWebService';
  defURL  = 'http://tstsvr.achworks.com/isapi/carwebservice.dll/soap/ICarWebService';
  defSvc  = 'ICarWebServiceservice';
  defPrt  = 'ICarWebServicePort';
var
  RIO: THTTPRIO;
begin
```

```
  Result := nil;
  if (Addr = '') then
  begin
    if UseWSDL then
      Addr := defWSDL
    else
      Addr := defURL;
  end;
  if HTTPRIO = nil then
    RIO := THTTPRIO.Create(nil)
  else
    RIO := HTTPRIO;
  try
    Result := (RIO as ICarWebService);
    if UseWSDL then
    begin
      RIO.WSDLLocation := Addr;
      RIO.Service := defSvc;
      RIO.Port := defPrt;
    end else
      RIO.URL := Addr;
  finally
    if (Result = nil) and (HTTPRIO = nil) then
      RIO.Free;
  end;
end;


initialization
  InvRegistry.RegisterInterface(TypeInfo(ICarWebService), 'urn:CarWebServiceIntf-
ICarWebService', 'utf-8');
  InvRegistry.RegisterDefaultSOAPAction(TypeInfo(ICarWebService),
'urn:CarWebServiceIntf-ICarWebService#%operationName%');
  RemClassRegistry.RegisterXSClass(TCarInfo, 'urn:CarWebServiceIntf', 'TCarInfo');

end.
```

The next step is to create a simple application to use the service. Create a new application as shown in the figure below:



Put an HTTPRIO component (under the WebServices pallete). This component uses HTTP messages to call remote interfaced objects using SOAP. Populate the component's properties in the Object Inspector; First put the WSDL in the WSDLLocation as shown in the figure below, then click Service and Port dropdown boxes, the properties will be automatically provided (this assumes that your computer has an internet connection).



- Open the previously saved IcarWebService1.pas and add this to the application (Project > Add to project on the menu).
- In the main application form, use this unit (File > Use unit on the menu, select IcarWebService1)
- Double click the Get Full Tank Miles button then type the codes similar to the one below:

```
procedure TForm1.Button1Click(Sender: TObject);
var
   MyService:ICarWebService; //declare a service
   MyCar:TCarInfo; //declare a TcarInfo type
begin
   MyService:=HTTPRIO1 as ICarWebService; //create the service
   MyCar:=TCarInfo.Create; //create MyCar

   //assign values to MyCar
   MyCar.Model:=Edit1.Text;
   MyCar.Year:=StrToInt(Edit2.Text);
   MyCar.MilesPerGallon:=StrToFloat(Edit3.Text);
   MyCar.TankCapacity:=StrToFloat(Edit4.Text);

   //Get Result
   Edit5.Text:=FloatToStr(MyService.GetFullTankMiles(MyCar));

   //Free MyCar
   MyCar.Free;
end;
```

Similarly, for the Get Gas Expense Button:

```
procedure TForm1.Button2Click(Sender: TObject);
var
   MyService:ICarWebService; //declare a service
```

```
   MyCar:TCarInfo; //declare a TcarInfo type
begin
   MyService:=HTTPRIO1 as ICarWebService; //create the service
   MyCar:=TCarInfo.Create; //create MyCar

   //assign values to MyCar
   MyCar.Model:=Edit1.Text;
   MyCar.Year:=StrToInt(Edit2.Text);
   MyCar.MilesPerGallon:=StrToFloat(Edit3.Text);
   MyCar.TankCapacity:=StrToFloat(Edit4.Text);

   //Get Result

Edit6.Text:=FloatToStr(MyService.GetGasExpense(MyCar,StrToFloat(Edit7.Text),StrToF
loat(Edit8.Text)));

   //Free MyCar
   MyCar.Free;
end;
```

REFERENCES

Delphi. 2005. Borland Software Corporation. Cupertino, California, USA. http://www.borland.com/Delphi.

SOAP 1.1 . 2003. Recommendations from the World Wide Web Consortitum (W3C) produced by the XML Working Group. http://www.w3.org/TR/soap/