How to publish an OpenEdge® Web Service – an overview

Progress User Group India Let's do PROGRESS together!	SKIL evolving thoughts... changing mindsets
www.pugindia.org	www.skilglobal.com
pugadmin@pugindia.org	info@skilglobal.com
Author: Raghuraman Kadambi	

Contents

Introduction to Web Services and OpenEdge® Support for Web Service

A Web service is a method of communication between two electronic devices over a network (Machine-to-Machine).

The W3C defines a Web service generally as:
"a software system designed to support interoperable machine-to-machine interaction over a network". (Source: https://en.wikipedia.org/wiki/Web_service)

History

Web services evolved from previous technologies that served the similar purpose such as Remote Procedure Call (RPC), DCOM, CORBA and JAVA Remote Method Invocation (RMI).

Web Services were intended to solve three main problems:

1. Interoperability
2. Firewall traversal
3. Complexity

Interoperability

Earlier distributed systems suffered from interoperability issues because each software vendor implemented its own format for distributed object messaging.

For example, development of DCOM apps strictly bound to Windows Operating system. And even, development of RMI bound to Java programming language.

Firewall

Collaboration across companies was an issue because distributed systems such as CORBA and DCOM used non-standard ports.

However, Web Services use HTTP as a transport protocol and most of the firewalls allow access though port 80 (HTTP), leading to easier and dynamic collaboration.

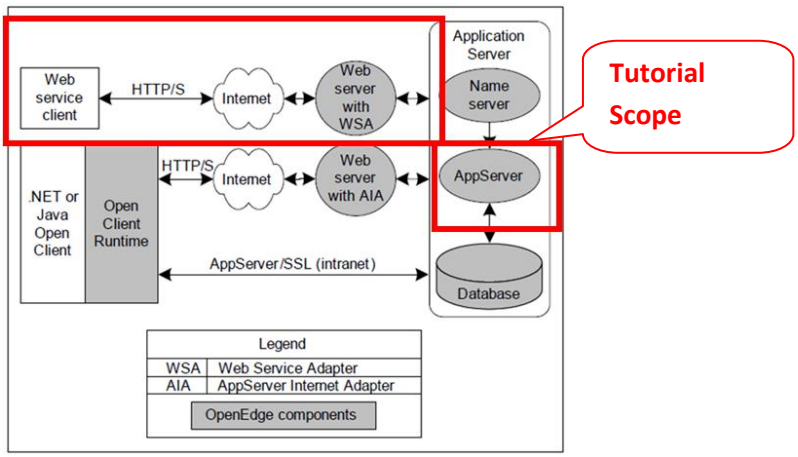
Complexity

Web Services is a developer-friendly service system.

Most of the above-mentioned technologies such as RMI, COM, and CORBA involve a whole learning curve. New technologies and languages have to be learnt to implement these services.

Web services in OpenEdge®

In OpenEdge®, a Web service is usually an AppServer® application that is accessible to a client application through a Web server.



In OpenEdge®, you could:

- Create new Web services that you build as ABL (Advanced Business Language) applications and deploy on an AppServer®.
- Expose existing AppServer® applications as Web services.
- Create an interface to your Web services and deploy it on a Web server.
- Create the client-side applications that interact with your Web services.

OpenEdge® includes support for Web services that are based either on SOAP (Simple Object Access Protocol), or on REST (Representation State Transfer).

Both SOAP and REST are industry standards, but there are many variations on how they are implemented.

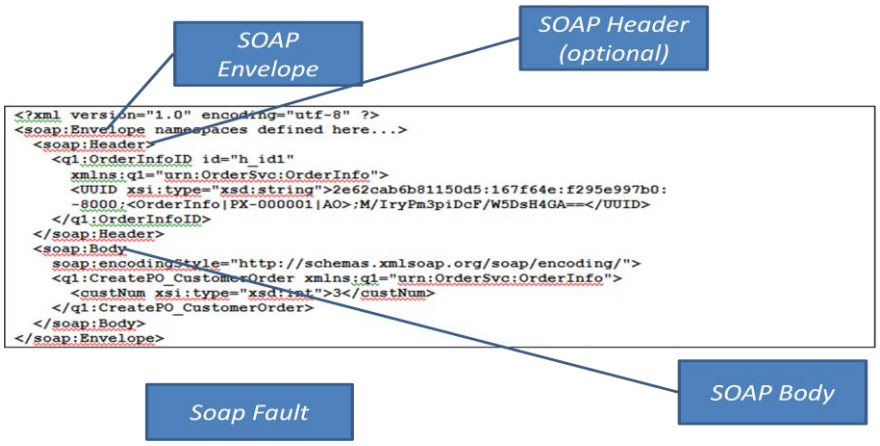
Loads of information available on web for these topics!

SOAP (Simple Object Access Protocol)

The Simple Object Access Protocol (SOAP) is the basis for enabling applications to communicate over the Internet, independent of how they are programmed or what platform they are deployed on.

The W3C standard for SOAP defines the format that a message must have when it is sent over the Internet for Web service requests and responses.

OpenEdge® support for Web services includes the functionality of a SOAP processor, which is used to manage SOAP messages on behalf of your application.



WSDL (Web Services Definition Language)

WSDL, the Web Services Definition Language, is another W3C standard.

A WSDL document is a body of metadata in the form of an XML-based description of how to communicate with a Web service. It describes the SOAP messages that a Web service accepts and generates.

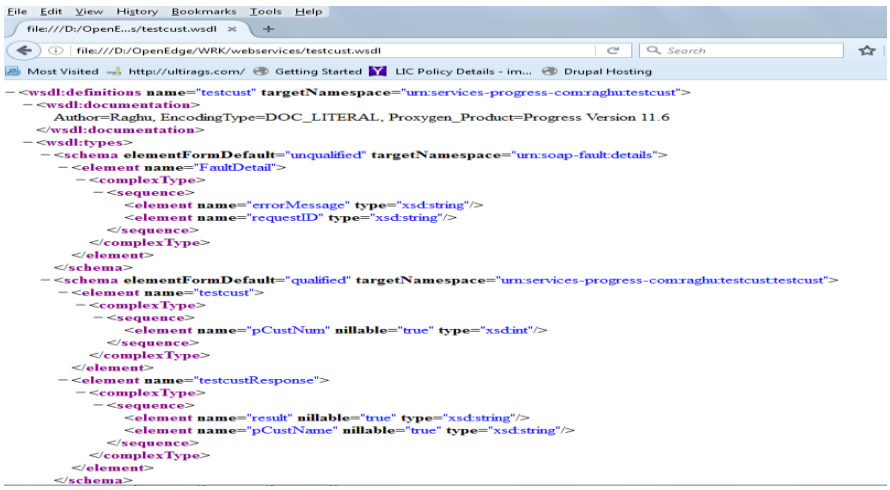
The WSDL document contains all the information a Web services client needs to make a request of, or consume, a Web Service. A WSDL document also contains the information used to locate the Web service on the Internet.

If an organization wishes to allow access to its Web services, it must create and host WSDL documents that describe them.

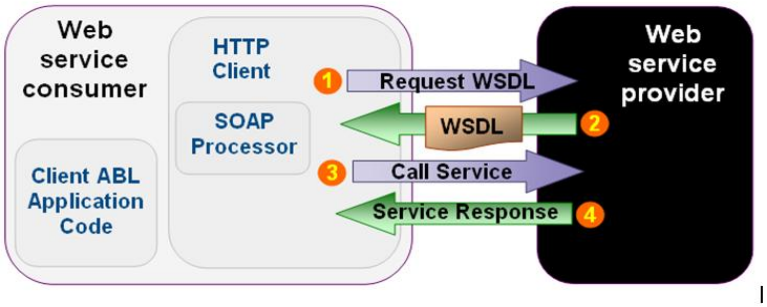
To exposing ABL procedures as Web services, you have to use the OpenEdge® ProxyGen tool. (Not in the scope of this document)

But if you are consuming someone else's Web service, you need only know the location of the WSDL that describes it. (In the scope of this document)

For the learning purpose, testcust Web Services is used in this document.



Web Service request and service response cycle in OpenEdge®



Steps to publish your own Web Service in OpenEdge® – an overview

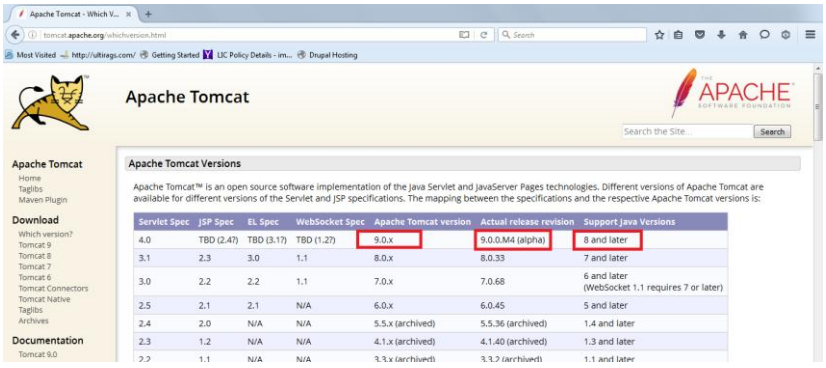
Key Components (used for this demo)

1. Apache Tomcat – WebServer with Java Servelets Processing capability
2. WSA (Web Service Adapter) – OpenEdge® Java Servelets communicates with AppServer®
3. AppServer® – Hosts the OpenEdge® application and OpenEdge® database
4. OpenEdge® ABL Code (*testcust.r*) – to be deployed as web service

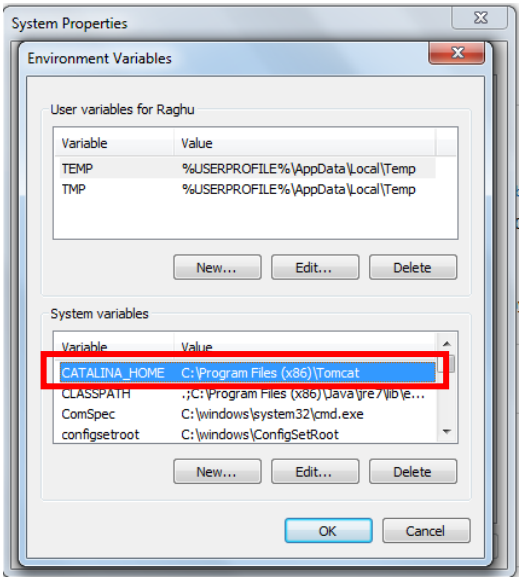
```
Procedure - D:\OpenEdge\WRK\webservices\testcust.p
File  Edit  Search  Compile  Help
/* TestCust.p:
* Test procedure for AppServer and WebServices call.
*/
DEFINE INPUT PARAMETER pCustNum AS INTEGER NO-UNDO.
DEFINE OUTPUT PARAMETER pCustName AS CHARACTER NO-UNDO.
DEFINE NEW GLOBAL SHARED VARIABLE giCallCount AS INTEGER NO-UNDO.

giCallCount = giCallCount + 1.

IF pCustNum = 1 THEN
DO:
  pCustName = "Test Name".
  RETURN "Success!".
END.
ELSE DO:
  pCustName = "No Name".
  RETURN "Failure!".
END.
```

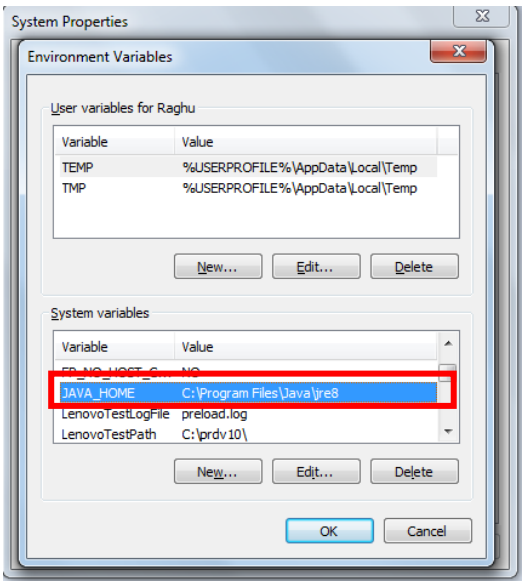
Steps

1. *Install Apache Tomcat* – [download from tomcat.apache.org] [This would serve as WebServer as well as Java Servelets processor]. The Progress OpenEdge® Web Service Adapter (WSA) is a Java Servelet and hence requires Tomcat. This tutorial uses Apache Tomcat 9.0.0.M4 and Java 8.



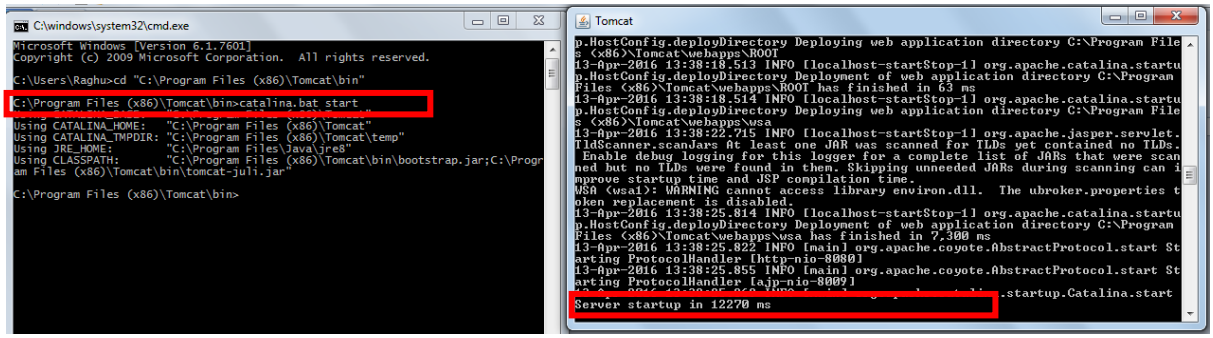
2. Once the download is done, you need to define a few of environment variables for the configuration script to use. In Windows Explorer, you can right-click on the My Computer node, select Properties, and then the Advanced tab, and Environment Variables.



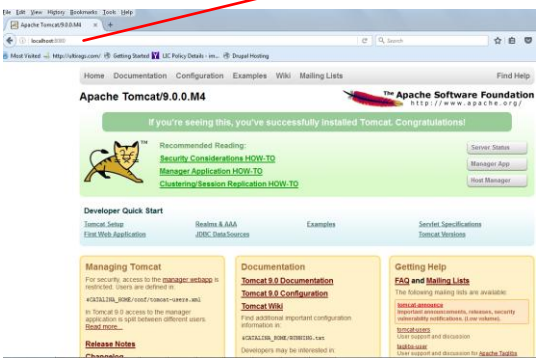


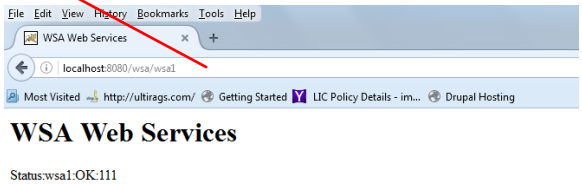
3. The WSA is a program supplied with OpenEdge® that runs as a Java Servlet in the Web server. It provides the connectivity between the OpenEdge® AppServer and the procedures. Copy the directory “wsa” under the progress install dierectory “servlets” [example D:\Progress\servlets] to the “Webapps” directory of the tomcat install directory [ex. C:\Program Files (x86)\Tomcat\webapps].

4. Start Tomcat web server – from the command prompt run catalina.bat start

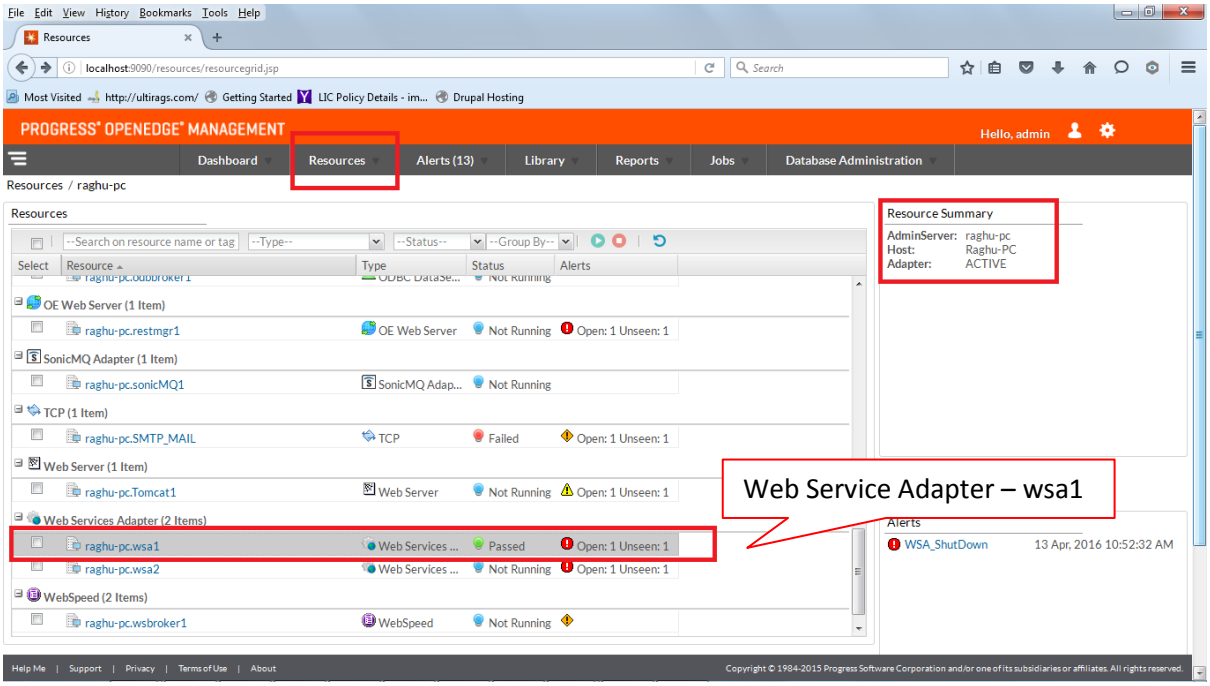


5. Verify the Tomcat WebServer and WSA (Refer to the URL in the images below)

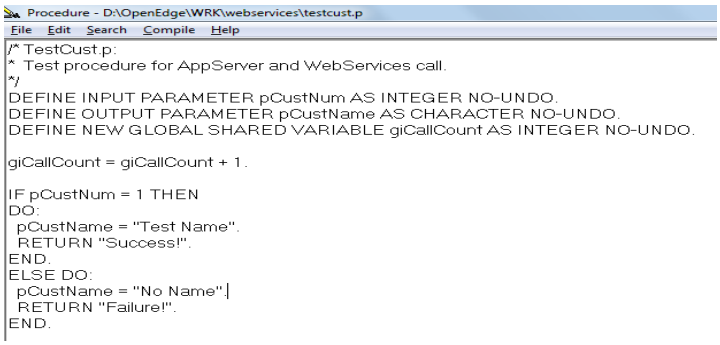




6. Launch OpenEdge® Management (usually http://localhost:9090) or OpenEdge® Explorer

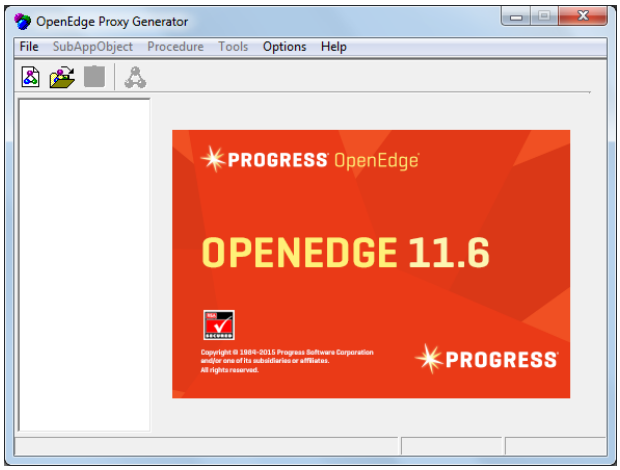


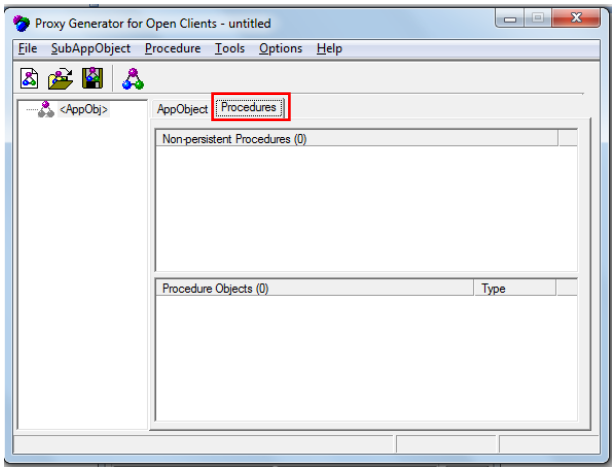
7. Till this point your Web Services infrastructure is ready. And now, it's time to create and deploy your own web services on this infrastructure you just created.
8. Deploying and Running an ABL procedure as a web service.
    a. ABL procedure, called ***testcust.p*** as below

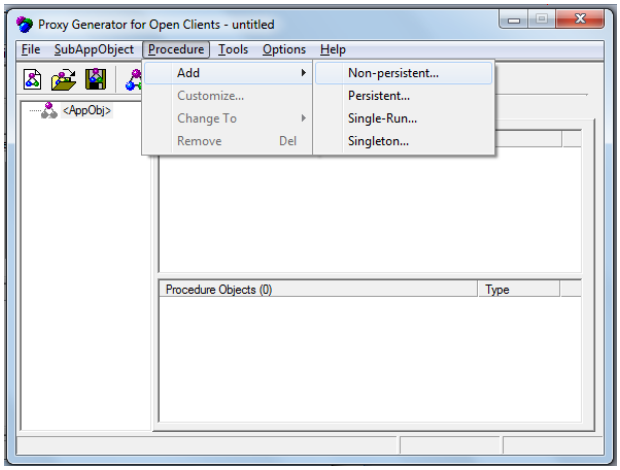


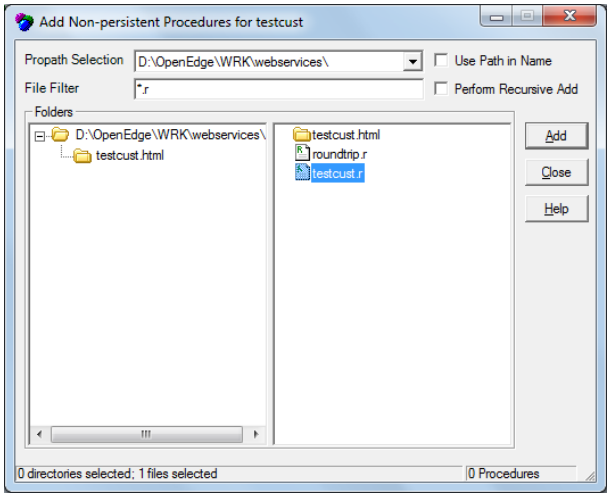
    b. Now, you need to make your ABL procedures accessible through an OpenEdge® AppServer to make them callable as Web services. The AppServer is designed to handle requests that come directly from OpenEdge® clients. To adapt it to receive requests as Web services, you need to insert a Web server into the mix. As shown earlier, this example uses the Apache Tomcat Web server. The Web server holds the Web Services Adapter (WSA), a program supplied with OpenEdge® that runs as a Java Servlet in the Web server.
    c. The OpenEdge® Proxy Generator, ProxyGen for short, generates the Web Service Application Descriptor (WSAD) and a test WSDL (Web Service Descriptor Language document) to use in deploying and testing the service.

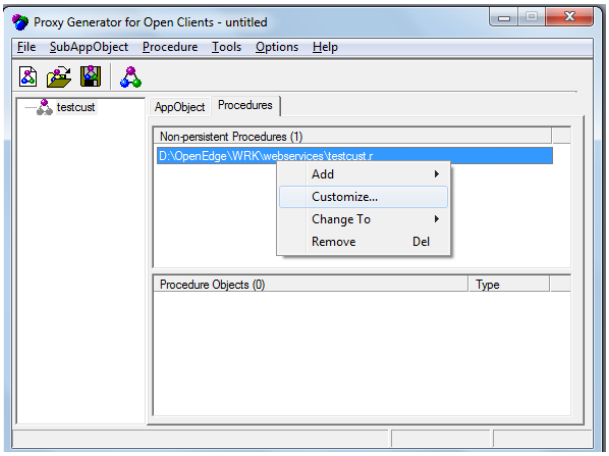
d. You start ProxyGen from the OpenEdge® Start group, and under the File menu, select New.

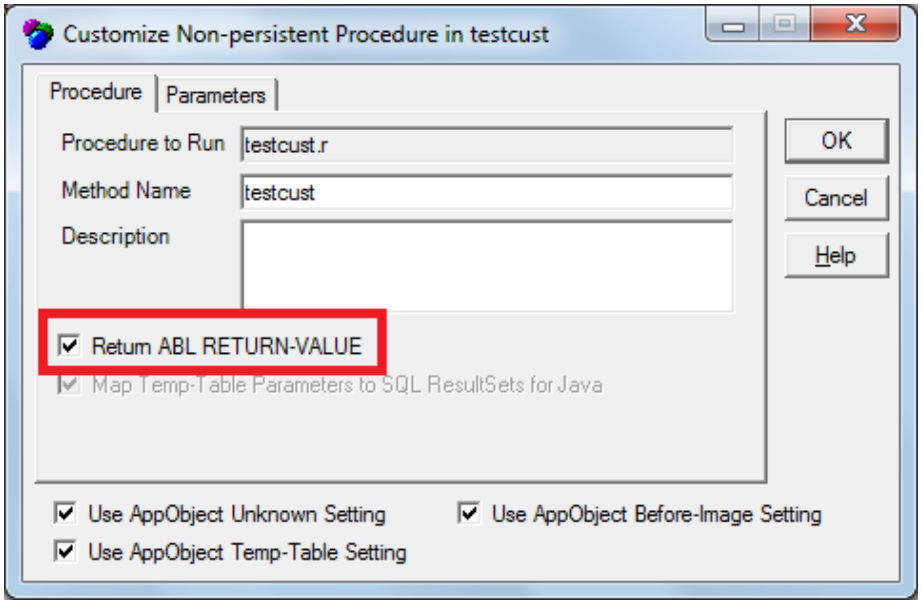


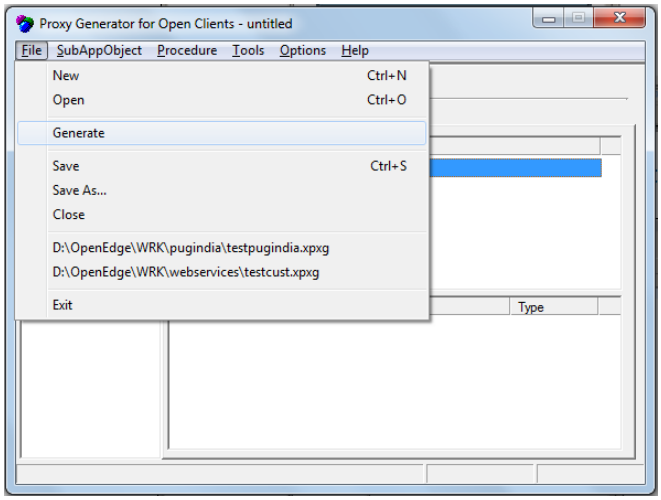


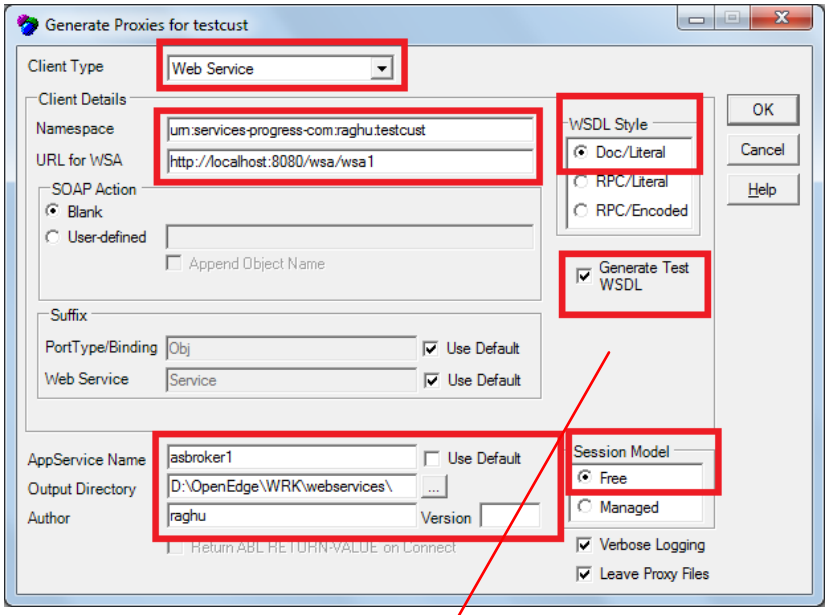



Add Non-persistent Procedures for testcust
Propath Selection
D:\OpenEdge\WRK\webservices\
Use Path in Name
File Filter
*.r
Perform Recursive Add
Folders
D:\OpenEdge\WRK\webservices\
testcust.html
testcust.html
roundtrip.r
testcust.r
Add
Close
Help
0 directories selected; 1 files selected
0 Procedures

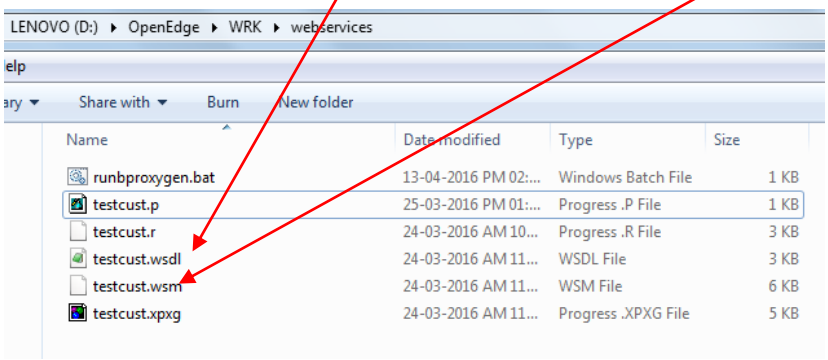

Proxy Generator for Open Clients - untitled
File SubAppObject Procedure Tools Options Help
testcust
AppObject
Procedures
Non-persistent Procedures (1)
D:\OpenEdge\WRK\webservices\testcust.r
Add
Customize...
Change To
Remove
Del
Procedure Objects (0)
Type


Customize Non-persistent Procedure in testcust
Procedure
Parameters
Procedure to Run
testcust.r
Method Name
testcust
Description
OK
Cancel
Help
Return ABL RETURN-VALUE
Map Temp-Table Parameters to SQL ResultSets for Java
Use AppObject Unknown Setting
Use AppObject Before-Image Setting
Use AppObject Temp-Table Setting





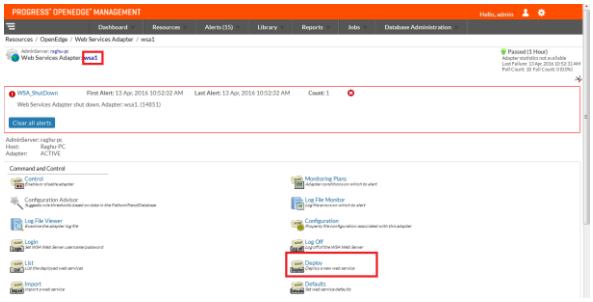
The job ProxyGen is creating the descriptor files the Web server will use to route a request to the right AppServer and the right ABL procedure. OpenEdge® Explorer reads the **Web Service Mapping** file that is part of what ProxyGen created, and uses this to deploy the procedure as a service.



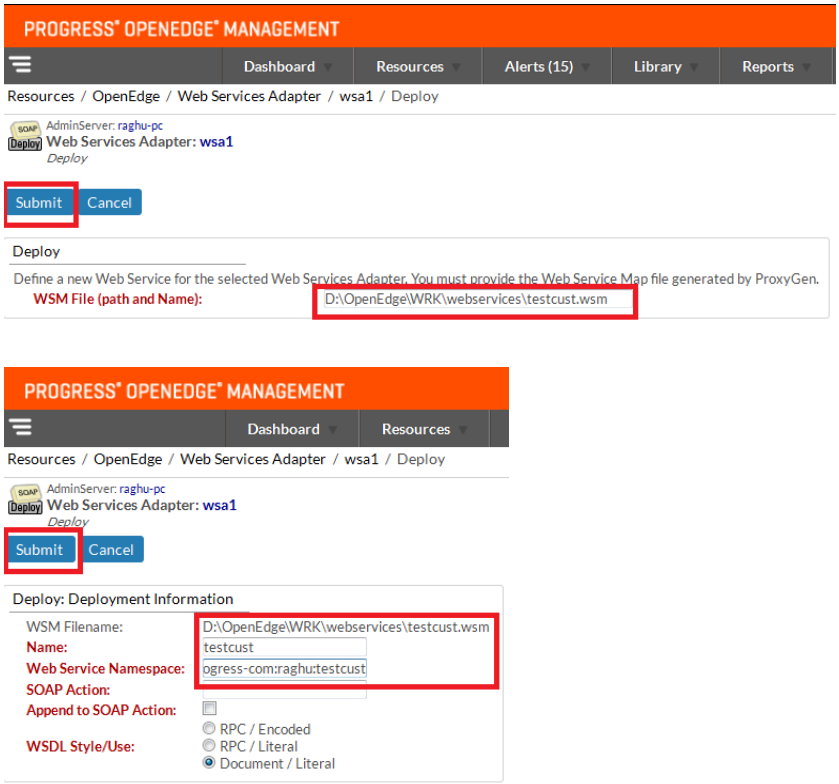
e. At this time the ProxyGen has created everything that a Web service call requires. ProxyGen can create proxies for many different kinds of external access to an OpenEdge® session, but this one

is a Web service. It's time now to deploy the web services using the files created above.

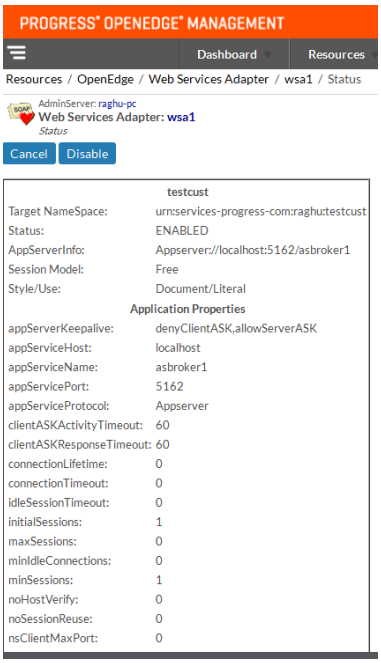
f. Launch the wsa1 resource from OpenEdge® Management / Explorer and click on the *deploy* button shown as below.



g. Select the *web services mapping file* (*testcust.wsm*) and submit the deployment.

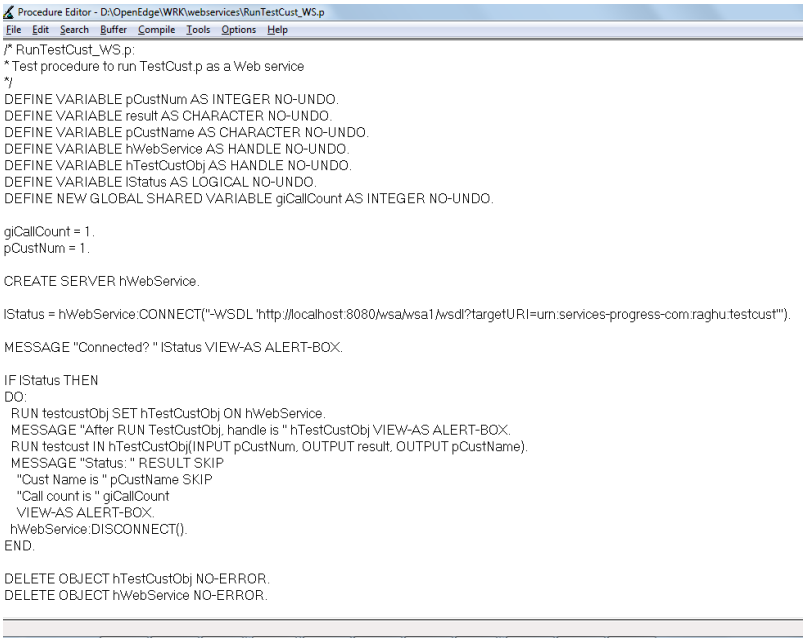


h. Scrolling down in the **wsa1** page, you can see a list of all the deployed services. If there's more than one deployed service, you can select one from the dropdown list and click **Select**. Then you click the **Status Enablement** link.



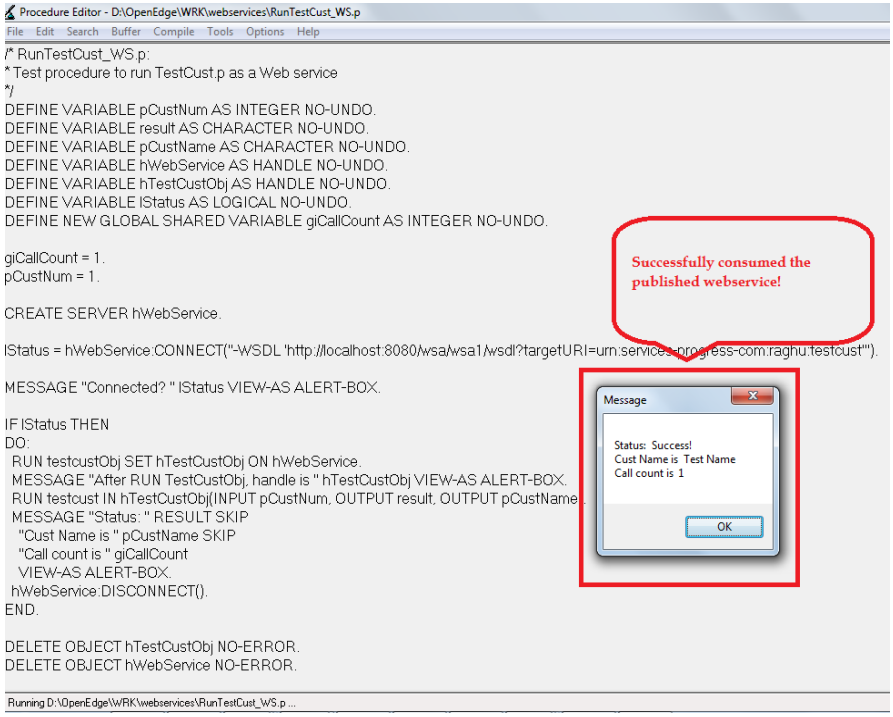
i. We have successfully have deployed the web service called *testcust*. This sequence shows the role Explorer plays: to read the output from ProxyGen and use that to set up the procedure to be run on an AppServer, accessed through a Web server using the OpenEdge® Web Services Adapter.

9. Consume the web service *testcust* created and deployed in the above sequence.
Write another procedure *RunTestCust_WS.p* for consuming the web services created and deployed above. For more information on consuming a web services refer to the article [http://ultirags.com/Accessing-non-Progress-Application-via-WebService]




```
/* RunTestCust_WS.p:
* Test procedure to run TestCust.p as a Web service
*/
DEFINE VARIABLE pCustNum AS INTEGER NO-UNDO.
DEFINE VARIABLE result AS CHARACTER NO-UNDO.
DEFINE VARIABLE pCustName AS CHARACTER NO-UNDO.
DEFINE VARIABLE hWebService AS HANDLE NO-UNDO.
DEFINE VARIABLE hTestCustObj AS HANDLE NO-UNDO.
DEFINE VARIABLE lStatus AS LOGICAL NO-UNDO.
DEFINE NEW GLOBAL SHARED VARIABLE giCallCount AS INTEGER NO-UNDO.

giCallCount = 1.
pCustNum = 1.

CREATE SERVER hWebService.

lStatus = hWebService:CONNECT("-WSDL 'http://localhost:8080/wsa/wsa1/wsdl?targetURI=urn:services-progress-com:raghu:testcust'").

MESSAGE "Connected? " lStatus VIEW-AS ALERT-BOX.

IF lStatus THEN
DO:
  RUN testcustObj SET hTestCustObj ON hWebService.
  MESSAGE "After RUN TestCustObj, handle is " hTestCustObj VIEW-AS ALERT-BOX.
  RUN testcust IN hTestCustObj(INPUT pCustNum, OUTPUT result, OUTPUT pCustName).
  MESSAGE "Status: " RESULT SKIP
    "Cust Name is " pCustName SKIP
    "Call count is " giCallCount
    VIEW-AS ALERT-BOX.
  hWebService:DISCONNECT().
END.

DELETE OBJECT hTestCustObj NO-ERROR.
DELETE OBJECT hWebService NO-ERROR.
```

10. Run the above procedure. The output demonstrates that the web services *testcust* has been successfully consumed.



References, Credits, Trademarks and Copyrights

1. Apache and Tomcat is trademark of Apache Software Foundation.
2. Creating and Deploying ABL Web Services, by John Sadd (2010)
3. Accessing Web Services From OpenEdge®, by John Sadd (2007)
4. Progress Software Corporation, USA – OpenEdge® Web Services documentation.
5. Definition of Web Service: https://en.wikipedia.org/wiki/Web_service
6. Progress®, OpenEdge®, AppServer® & WebSpeed® are trademarks of Progress Software Corporation, USA.
7. "PMP" & "PMI" are registered marks of the Project Management Institute (PMI), USA.
8. All other logo and trademarks referenced in this material are belong to their respective owners.
9. Copyright content belongs to its rightful owners.
10. Additional reading - Accessing non Progress Application via WebService – An overview, by Raghuraman Kadambi [URL - http://ultirags.com/Accessing-non-Progress-Application-via-WebService]