Interoperability issues seen in Web Services

Sujala D Shetty * , Dr S Vadivel**
*Senior Lecturer, BITS, Pilani - Dubai, Dubai UAE
** Associate Professor, BITS, Pilani - Dubai, Dubai UAE

Abstract
Web services are applications that enable internet based distributed computing. There are broadly two types of tools say J2EE and .NET for hosting and consuming web services. Because of the heterogeneous nature of the internet it is mandatory that a J2EE client should be able to invoke a .Net web service or the other way round. But in reality the feasibility of above is not always possible. In this article the state of art in web services interoperability issues have been extensively discussed and good practices for interoperability have been suggested.
Keywords: J2EE, .NET, web services, interoperability

1. Introduction

Today, distributed computing together with the growth of the Internet make the Web services possible for their users.Component based technologies like CORBA and RMI are connection oriented and use non standard communication ports, which may cause problems for accessing due to the security issues.Furthermore these technologies cannot handle the network interruption successfully so it costs a lot for reconnecting to the remote server after corruption.While Java RMI supports cross platform interoperability, it is difficult to achieve interlanguage interoperability with Java RMI[1].
On the other hand, when it comes to connecting applications together that were written for different languages, Web services technology come to the picture.Web services are peices of functionalities that can be accessed by sending them messages formatted in XML over a network.All messages are sent through standard internet protocols like HTTP.Messages sent between the client and server are encoded in a XML formatted protocol , say Simple Object Access Protocol (SOAP), which defines the standardized way of accessing web services on remote machines.Web services provide interoperability across platforms and languages.The Web Service Description Language(WSDL) can be used to describe the interface a web service offers, also based on XML[2].

1.1. Defination of Interoperable Web Services

An interoperable web service is one which can work across platforms, languages, applications and with web services from different vendors.

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

1.2. Why Interoperability ?

Once achieved, the ability to seamlessly integrate Java Enterprise Edition (Java EE) and .NET environments will help developers create applications on a diverse range of operating systems including the Solaris Operating System (OS), Windows and Linux, that can co-exist and interoperate across heterogeneous computing environments. Seamless integration will also enable greater collaboration for enterprises, by allowing them to leverage a larger ecosystem of partners in application development. Additionally, interoperability between the two platforms will help pave the way for greater adoption of web services and service-oriented architecture (SOA) -based application development by reducing the associated cost, complexity and risk [3].

1.3. Examples of Interoperability

e.g.1
In a real life scenario it is seen that organizations comfortable in .NET would develop their web service using a .NET platform.Now the client which invokes the web service could be a ASP.Net client, a pearl cient or a java client, similarly organizations comfortable in Java could have a web service developed in Java and this service could be invoked by a client developed on any programming language platform.As a simple example we could consider a simple web service for currency conversion which is developed in Java with Axis which is consumed by a ASP.Net client created in C# .

e.g.2
We could also consider multiple web services interacting with each other as an example we could consider a supply chain application where we have the supplier, warehouse, manufacturer and the retail stores interacting with each other. Developers could be using a Java2 Enterprise Edition (J2EE) based web service deployed on Oracle 9i application server with web services deployed on other platforms such as .Net and J2EE application servers from other vendors.
e.g.3
We could consider a purchase order scenario. In this scenario, a potential buyer (that is, the "Customer") retrieves a catalog of products offered by a particular buyer and selects which products will be purchased in

which quantities. When a purchase is complete, an order is sent to the supplier with the types and amounts of products that the buyer requested.
The supplier will then check if the requested products are available to be shipped to the customer. This is achieved by querying the warehouse. If there is insufficient stock to fulfill the order, the order's status is set appropriately and an error is returned. If the available stock is sufficient, the supplier will then check the customer's credit to execute the purchase. Each buyer is associated with a particular bank that can provide information about the account status, as well as deduct the required funds from that account.
Finally, assuming that the bank account status is good, the supplier sends a request to the warehouse to ship the order. The customer, in the meantime, can check on the status of a particular order with the supplier, or retrieve an invoice for a confirmed shipment.
Each of the participants in the application scenario described above, namely Customer, Supplier, Warehouse and Bank, is implemented as a Web service. These services can run on the same machine or on different machines, with different implementations, showing the value of Web services technology in a heterogeneous environment.
Which implementation of a role is used when running the application is determined by a set of XML files that describe combinations of, for example, a particular warehouse service with a particular supplier. This way, any permutation of role implementations by the different vendors can be configured. The XML configuration files are kept in a central place and, typically, cannot be changed [4].

2. BASIC PROFILE

Interoperability is an important factor in the success of solutions that are based on Web Services and Service Oriented Architecture (SOA), along with other key factors such as contracts, loose coupling, and reuse. Interoperability is generally accomplished by developing your Web Services using the well-established guidelines for implementing Web Services and by following industry standards such as XML, WSDL, *SOAP*, and UDDI. However, just following Web Services standards and guidelines during the development phase of a project isn't sufficient to achieve interoperability.
The different products used for development also have to comply with many requirements such as the need to have similar implementations (data types, formats, and schemas) of the standards that you want to use. As different products are provided by different vendors, developed by several sets of people, and employ various types of underlying technologies, achieving a common understanding often becomes very difficult, which makes the products likely to be non-interoperable with each other.
Over the last few years, the basic Web Services standards like XML, WSDL, and SOAP have matured a lot and WS-I have released a Basic Profile that contains implementation guidelines for basic Web Services standards. Today, most vendors provide products that comply with the Basic Profile and support the standards included in the profile. With the wide adoption of the Basic Profile, software vendors have been able to make their products interoperable to a great extent.
There are four deliverables produced by the WS-I for the Basic Profile version 1.0. Briefly, they are:

- The Basic Profile, which contains requirements and guidelines for writing interoperable Web services.
- The Basic Profile usage scenarios, which describe fundamental ways that providers and consumers interact.
- The sample application, which is an implementation of an interoperable Web service that demonstrates the requirements and guidelines presented in the Basic Profile.
- The testing tools, which help developers verify that their Web service implementations conform to the requirements in the Basic Profile [5].

Achieving interoperability for scenarios involving only basic standards is relatively easy if you follow the guidelines set by the Basic Profile (BP) 1.0 or 1.1 of the Web Services Interoperability Organization (WS-I). The Basic Profile consists of implementing guidelines recommending how a set of core Web Services specifications should be used together to develop interoperable Web Services. The guidelines address technologies that cover four core areas: Messaging, Description, Discovery, and Security. BP1.0 covers the following core Web Services specifications and provides constraints and clarifications to these base specifications, along with conventions about how to use them together [6].

2.1. Interoperability issues seen with Java applications developed with different tools

Since Java can be developed by different vendors there is a problem with interoperability between these platforms itself. There are a number of proposals to increase interoperability.

a. Make it possible for Java engineers to use the same add on tools with programming applications from different Java providers, but most of the large java providers including IBM are not in favor of this suggestion.
b. Leading Java tool companies have tried to encourage independent software providers to build plug-ins for their respective products. But a standardized system for plug-in interoperability does not exist.
c. A Java tool called Eclipse is an open source effort started by IBM . The eclipse project has created a 'framework' in which several development tools can operate. The eclipse tool allows a Java programmer to combine a coding tool with a source code management system from different providers, but eclipse is not pursuing outside its own.
d. Oracle spearheaded Java Specification request (JSR) which creates a standardized way to plug third party utilities into java tools which are already under way.
e. It was noted that the technical approach of JSR 198, which reflects Oracle's beliefs, differs from that of Eclipse. Rather than advocate one single foundation that third-party tools can connect to, the JSR 198 is proposing a plug-in system that would allow Java programmers to choose between different frameworks from tools companies such as BEA, Borland Software, IBM and Oracle. While Eclipse wants to be the framework for all tools. Oracle disagrees with that [7].

3. .NET and J2EE

Similarities

1. Both technologies provide a number of API's that serve a common purpose e.g. IO, reflection, serialization, networking etc.
2. Both technologies support primitive data types, e.g. integer, float, Boolean, double, long etc. However since .NET supports multiple languages primitive data types have been mapped to a specific class in the .NET framework.
3. Both technologies do not support multiple inheritances.
4. Applications written in both technologies get compiled to an intermediate language.
5. Both technologies have a garbage collector for managing their resources. The garbage collector deletes resources once they go out of scope. Though the garbage collector performs the same function in both the technologies their approach is very different**.**

Differences

The differences in approach between .NET and J2EE and the technical challenges accompanying them are sometimes significant hindrances to interoperability. The following are some of the differences between .NET and J2EE.

1. J2EE is a set of open standards not a product. .NET on the other hand, is a product suite with some features built on standards and other features that extend standards.
2. .NET provides runtime support for SOAP and UDDI as native .NET protocols.
3. Integrated support is provided in .NET to build and deploy XML based web services.J2EE vendors must provide integration between J2EE products and an IDE offering.
4. .NET provides business process management and e-commerce capabilities. These capabilities may be provided in J2EE implementation but are not part of the standard.
5. J2EE is focused on application portability and connectivity between platforms supporting Java..NET targets application integration using XML.
6. Application and backend integration approaches differ. Java uses JCA (Java connector architecture) to connect to specific systems and applications. Connections across disparate applications is through JMS..NET provides integration through several mechanisms, the host integration server 2000, COM Transaction Integrator (COM TI), Microsoft Messaging Queue (MSMQ) and Biztalk Server 2000[8].

3.1.Interoperability Issues seen between Java and .Net Web Services

- Using vendor tools to derive the Web services semantics in WSDL from implementation code is convenient, but this approach ignores the design of the message schemas which is central to Web services interoperability in heterogeneous environments (J2EE technology versus .NET, for example).
- The ease, flexibility, and familiarity of the popular *RPC/encoded* style makes it an attractive choice for developers; however, the difficulty in synchronizing the implementations of the abstract SOAP encoding data model

among vendors presents a difficult challenge for Web services interoperability.

- Weakly-typed collection objects, arrays containing null elements, and certain native data types all pose special problems for interoperability. Specifically:
  - It is impossible for vendor tools to accurately interpret XML Schemas representing weakly-typed collection objects and map them to the correct native data types.
  - The XML representations of an array with null elements differ between .NET and WebSphere.
  - Because native data types and XSD data types do not share a one-to-one mapping, information or precision can be lost during the translation.
- Different naming conventions in .NET and Java technology can result in namespace conflicts, as can the use of relative URI references [9].

4. Test Results

Results seen in implementation of Interoperability Issues

Web services exchange data by exchanging XML documents. As soon as data objects are pushed into the Web service stack they are represented as XML documents. Thus, the Web service stack on the receiving end should know how to interpret the XML document sent by the sender. The XML Schema, which provides an outline of the XML document, helps the receiver to map the data which is represented in XML. But the implementation difference in the underlying technologies of J2EE and .NET results in different mappings between the schema and native data types on both the platforms. This may lead to information distortion and de-serialization failure. [10]

We have tested interoperability issues by creating a reliable web service in Java using Netbeans 6.5 and deploying it on Glassfish server (v2). We than create a C# client in .NET and a Java web client, in order to compare the performance of a Java web service and a Java client and a Java web service and a .NET client we allow the client to invoke the web service and pass data to the web service the web service processes this data and sends it back to the client. We shall see the output for various cases like primitive data types, arrays with null elements, and complex data types. We can check the communication between the web service and the client by the exchange of SOAP messages using the TCP Monitor.

4.1. An Array with Null Elements

The XML representations of an array with null elements are different between .NET and Java. Consider a Java web service which returns an array with a null element. A java client can correctly interpret the null string in an array. However, a .NET client interprets the null string as a string of length zero or an empty string. Empty and null strings are completely different from each other in object oriented programming language[11].

Output from Java and .NET clients

The screenshots below show the difference in the interpretation of null values by Java and .NET clients. The output of the Java client is Disha, null, Vinita. Hence, we infer that Java clients infer the null values correctly. Whereas the .NET client displays null as an empty string and cannot deseialize null values correctly.

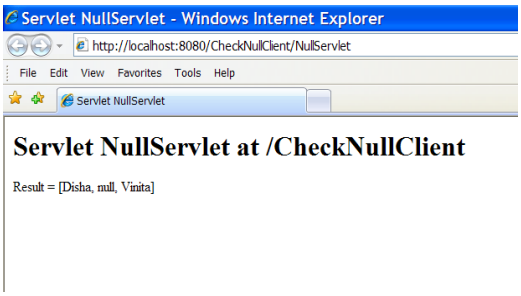

Fig.1. Output of an array with null element when invoked by a Java client

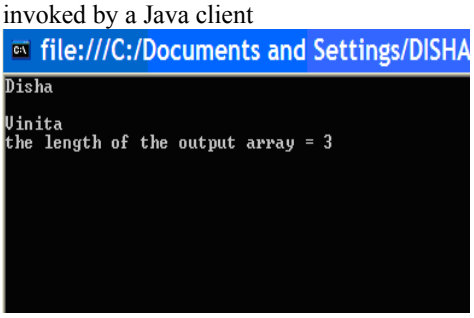

Fig.2. Output of an array with null element when invoked by a .NET client

4.2. Primitive Types

Primitive data types can cause trouble. Each programming language has a set of native data types. A one-to-one mapping is not available between native data types and XSD data types. Therefore, information can be lost during the translation, or the receiver is not able to do the mappings for certain native data types[11].

Unsigned Numbers

For example, unsigned numerical types, such as xsd:unsignedInt, xsd:unsignedLong, xsd:unsignedShort , and xsd:unsignedByte , are the typical examples of xsd types. In .NET, the uint, ulong, ushort , and ubyte types map directly to the xsd types, in Java language unsigned types are not defined.

[WebMethod]

Public uint getUint(uint ui)

{ Return ui; }

This is a .NET Web Service which returns the unsigned integer passed to it. Since unsigned types are not defined in Java, it leads to an interoperability issue when a Java client tries to call this Web service.

To solve this, use the WebServicesAssembler tool to map the request input type to the Java native type long and then call the web service. Another thing to do is use wrapper methods to convert these unsigned data types to xsd:string type so that interoperability is achieved.

4.3 Precision issues

For xsd:decimal, xsd:double , and xsd:float types, each platform might have different precision support. This may lead to loss of precision. Let's consider the following example in which a Java Web Service returns the sum of two float numbers. Java has a precision of 6 digits after decimal whereas .NET has a precision of 5 digits after decimal. Therefore, rounding off takes place in the .NET client and it loses precision[12].

Java Client

The Java client which calls the add method of the testprecsion web service and passes the float values 4.111111 and 8.888888 to the web service and dislays the sum that is returned by the web service i.e. 12.999999.

.NET client

The .NET client also passes the same values and displays the sum returned by the service. But because.NET is less precise the value 12.999999 gets rounded of to 13 and this is displayed.

Output from Java and .NET clients:

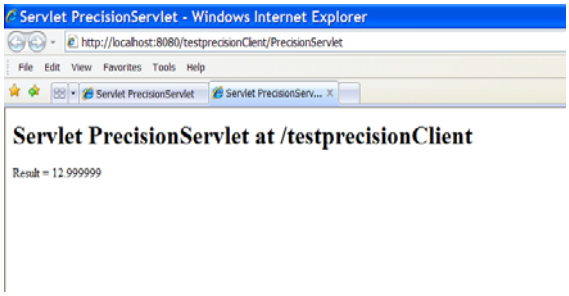

Fig.3. Precision testing with a Java client

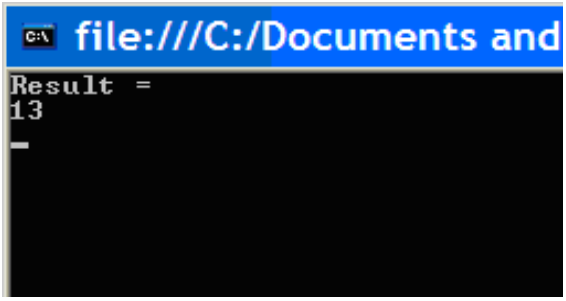

Fig.4. Precision testing with a .NET client

4.4 Collection of complex data types

In both Java and C# there are rich libraries of collection types. For example, Java supports collection types like java.util.Hashtable, vectors, Set, ArrayList, etc. Whereas, in C# there's Systems.Collections.Hastable, SortedList, Queue, Stack, ArrayList,etc.][12] These collection objects contain elements of different data types. Due to this, they may also be considered as weakly typed data structures.

When exposed across Web services they create problems. The receiving side may not be able to understand the SOAP messages that contain weakly-typed object elements and native data types. For example, an ArrayList in a .NET web service is taken to be data of 'anytype' in the XML Schema. This makes it ambigious. Therefore, when the Java client sees the Schema, he wont know which collection type to map the data to at the receiving side. This can be resolved by sticking to simple data types as much as possible and avoiding the use of complex data types.

c. Collection of complex data types

Collection objects might contain elements of any data types. Thus, many consider them as weakly-typed data structures. That makes them a wonderful programming

tool. In object-oriented programming, there are rich libraries of collection types. In Java for example, there are:

- java.util.Hashtable
- Vectors
- Hashmap
- Set
- ArrayList

While in C#, there are:

- System.Collections.Hashtable
- SortedList
- Queue
- Stack
- ArrayList

If exposed across Web services, these collection types can cause insurmountable problems. The problem lies in how the receiving side is able to understand the serialized Simple Object Access Protocol (SOAP) messages that contain the weakly-typed object elements and native data types. Even though some collection types look extremely similar between languages, such as System.Collections.ArrayList in C# and java.util.ArrayList in Java, remember that the elements in the collections are generic references. To accurately unmarshall the XML representation of a collection, consumers must have prior knowledge of the original concrete types. The burden is on the toolkit developers to interpret the XML Schemas published by the Web services providers and map the SOAP messages to the native data is not an easy task for the weakly-typed collections[10].

An object of student was created, the object contained details like id_no which was of type int, name of type string, dob of type date, gender of type char, subjects an array of type string, marks an array to store float values and a Boolean value, the object containing the mixed data types could be successfully sent from the client to the web service and the object could be successfully displayed in the hash map table on the web service.

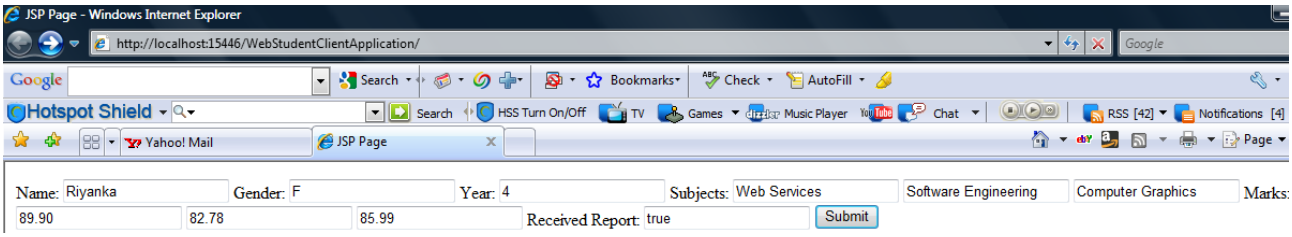

Fig. 5. Initial Screen to enter Student Details (Java Client)

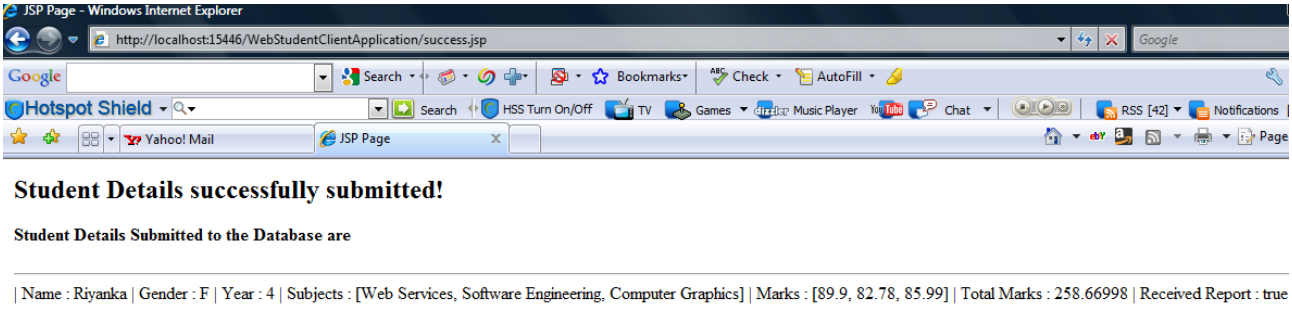

Fig.6. Result displayed after successful invocation (Java Client)

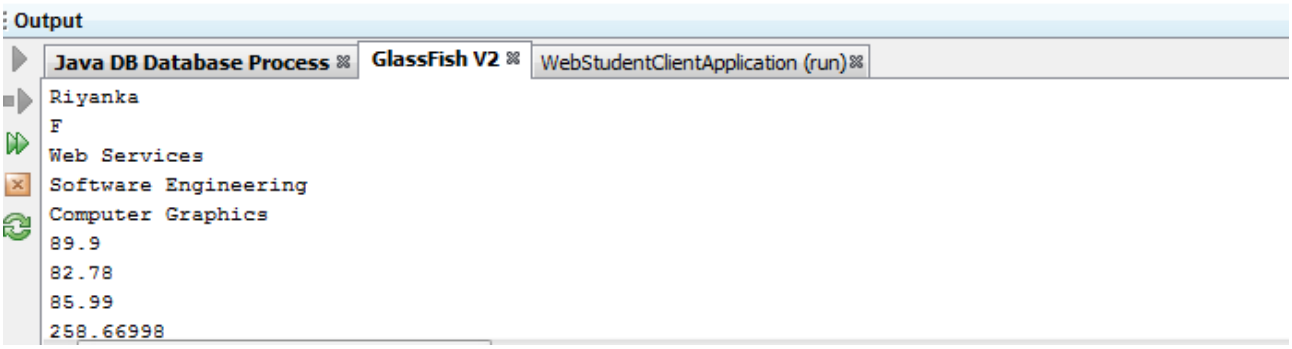

Fig.7. Output showing hash map contents of StudentDetails (Java Client)

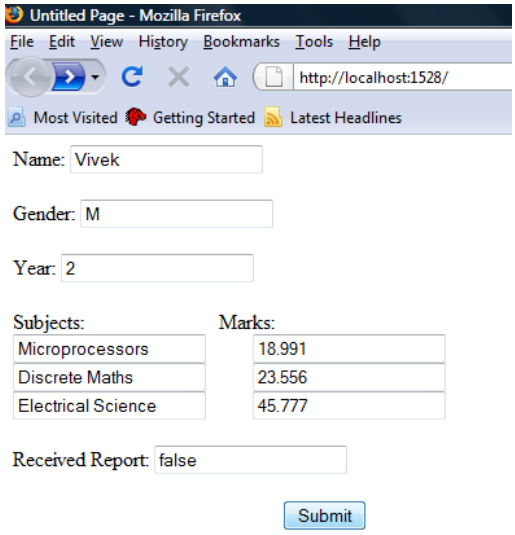

Fig.8. Initial Screen for entering Student Details(.NET Client)

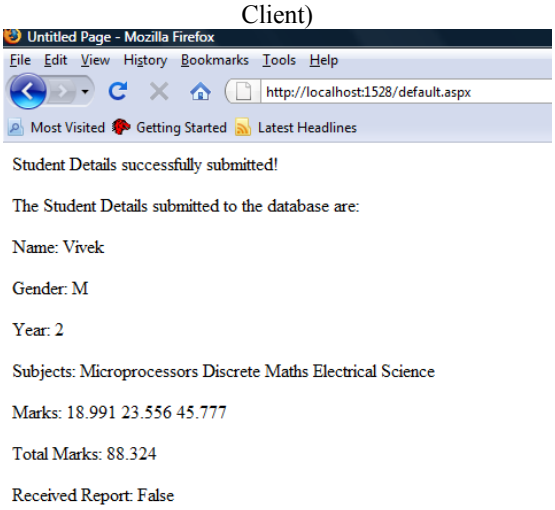

Fig.9. Result displayed after successful invocation(.NET Client)

Fig.10. Output showing hash map contents of StudentDetails(.NET Client)

4.5 Relative URI reference as a namespace declaration in WSDL

XML namespaces help in creating universally unique URIs. They resolve naming conflicts in the XML documents. However, the way that URIs are interpreted and mapped in the native code differs between platforms. It is usually relative URIs which cause a problem. In Java, it's not a problem when the WSDL file is generated by the Web service itself. This is because the target namespace is derived from the package and the tool automatically qualifies them with the schemas. But, when the web service is on .NET and it generates the WSDL then the target namespace comes directly from what is mentioned in the code. In .NET, the process of qualifying with the schema is not done and the relative URIs sometimes cause conflict when the target namespace is the same. Therefore, to avoid this, the best practice is to always make the namespace unique by qualifying it with its own organization domain name[11].

4.6. DateTime Issues

We have a schema data type called xsd:dateTime. This too is one of the primitive data types, but is discussed as a separate issue here due to a variety of problems

occurring when this schema type is used, if not careful[12].

4.7. Null Values in Date data type

The communicating parties could pose problems if one of their data types is a reference type and the other is a value type. The xsd:dateTime is mapped to System.dateTime in .NET. This is a value type, whereas it is mapped to java.util.Calendar or java.util.Date, which is a reference type, in Java. We know that the object of a value type is in a stack and the object of a reference type is in a heap. Hence a null reference is allowed as it signifies that the object has a null pointer but a value type cannot have a null value.
In Java when the reference type is not referencing any object, we can assign a null value to it. Whereas, .NET Web services will throw a System.FormatException in case it receives a null value to its value type of data from a Java client.
If the Calendar or Date object is initialized with a null value in a Java client, then a null xsd:dateTime is sent in the SOAP message. When the Web service built on the .NET platform receives the SOAP message, correct deserialization of the message is not possible. This is because the System.DateTime type is not nillable.

4.8. Precision problem in date data type

Different platforms use different precisions when interpreting the native dateTime types. When translating values of an XML dateTime simple type to different platforms, loss of precision can occur. The .NET platform uses four digits for the year value and seven digits for the milliseconds and the Java platform uses five digits for the year value and three digits for milliseconds. This can be cleary illustrated in the following example:
Here is a .NET Web method that returns a System MAX_VALUE of the DateTime data type.

The Java client then gets a SOAP Response from the .Net Web Method returning the MAX_VALUE of the DateTime datatype.

```
<?xml version="1.0" encoding="utf-8" ?>
  <dateTime xmlns="http://tempuri.org/">9999-12-31T23:59:59.9999999 08:00</dateTime>
```

Since the Java platform uses only 3 digits for the milliseconds and the MAX_VALUE has seven digits, it rounds up the date. Therefore on the receiving side we get the output as

January 1, 10000

5. Best interoperability practices for developing web services.

a.Use XSD First

Always define the data first. Than decide on what data will be sent and create the XSD than use tools to generate classes from the XSD file which will guarantee interoperability

b. Use Unit Tests to Test Interoperability

It is always a good practice to test the units separately (using NUnit for .Net and JUnit for Java) in a web service, if datatypes change, than we can rerun the unit tests.

c. Ensure Document/Literal when generating Web Services

As per the WS-I profile 1.0 only Document / Literal should be used as the default encoding mechanism.

d**.** Add Option to Change Host and Port

In order to make it easy to change the location of the web service it is a good practice to add a helper method to change the host and port value of the web service location.

e. Use Trace Tool to Investigate

Trace tools are invaluable for investigating SOAP requests and responses between web services.

f. Always use compareTo() when comparing dates/times

If sending dates and times over a Web Service between .NET and Java, always use the appropriate compareTo() method in Java to compare dates (as opposed to date == value). This will help ensure accuracy for date comparisons between the platforms, especially when trying to compare milliseconds values.

g. Null Dates and Times are recognized by Java, but not by .NET

In Java, java.util.Date and java.util.Calendar are classed as reference types. In .NET, System.DateTime is considered a value type. Reference types can be null, whereas value types cannot. If you are planning to send null date values across a Web Service, always send the

value in a complex type, and set the value of the complex type to null. This will help prevent the null date value being interpreted incorrectly (and raising an exception).

h. Watch out for empty arrays

Some toolkits recognize an empty array as a single null value, but if you are sending an array of objects over a web service, always ensure they contain valid data[13].

6. Conclusion

Web services today are provided by the core UDDI, WSDL, and SOAP protocols. On the immediate horizon are a second layer of protocols that define workflow automation (BPEL), Web service management services, and vertical market protocols. Web services greatly help developers build highly integrated solutions. So it should be no surprise to see interoperability problems arrive when workflow automation Web services are mixed with vertical market Web services. For example when a client developed using Microsoft .NET consumes a Web Service supplied by an Apache-Axis server, probably either because of ambiguities in SOAP specification or server provider's implementation differences interoperability problems do happen. Generally the problem happens if response of the service contains empty arrays. If Web service toolkits are continually improved to solve interoperability problems then customers, users and businesses will more efficiently solve system integration problems than using the existing standards (CORBA, DCOM, and RMI). The more serious Web service toolkits vendors have been diligent at solving interoperability problems. If interoperability problems linger or get worse then we are in for slower adoption of web services and will lead to much bigger professional services costs to implement intranet systems In this article core issues involved in web services interoperability among j2ee tools as well as .net Vs J2ee tools have also been discussed and some issues involved with interoperability like null elements in an array , primitive data types and complex data types have been implemented. In addition to that best practices for web services interoperability among j2ee and .NET also have been suggested. Further efforts taken by vendors to achieve web services interoperability also have been highlighted.

7. References

[1] Ashish Banerjee, Arvind Corera, Zach Greenvoss, C# Web Services building Web Services with .NET remoting and ASP.net, Wrox pres Ltd, Birmingham, UK 2001.
[2] James Snell, 'Web Services Interoperability" , http://www.xml.com/pub/a/2002/01/30/soap.html]
[3] [http://www.sun.com/smi/Press/sunflash/2005-11/sunflash.20051104.1.xml]
[4] [http://www.ibm.com/developerworks/library/ws-bpinter/]
[5] Web Services Specifications and SOA InteroperabilityAchieving interoperability is neither simple nor straightforward By: Sanjay Narang
[6] Building Interoperable web services WS-I basic Profile 1.0 , Microsoft
[7] Narayana Rao Surapaneni, Dhananjay Khatre, "Java and .NET a developers guide to Interoperability and Migration, Prentice Hall India"
[8] David A Chappel , Tyler Jewell, 'Java Web services', O'Reilly publications
[9] [http://www.ibm.com/developerWorks] Wangming Ye Web Services Interoperability between J2EE and .NET
[10] http://www.developerfusion.com/article/5155/web-services-interoperability-between-j2ee-and-net-part-2/
[11] http://www.developerfusion.com/article/5155/web-services-interoperability-between-j2ee-and-net-part-3/
[12] http://download.oracle.com/docs/cd/B25221_04/web.1013/b25603/interop.htm
[13] Arun Gupta, Project Tango : Adding quality of service and .NET Interoperability to the Metro Web services stack . http://blogs.sun.com/arungupta

Sujala Shetty has finished her MTech from MIT, Manipal in 2002. She is currently pursuing her PhD from BITS, Pilani. She has worked as lecturer in the Computer Science Dept of MIT, Manipal from 1997 to 2002. She is currently working as Senior Lecturer in the Computer Science Dept of BITS, Pilani-Dubai Campus from 2002.She has three publications in International conferences .Her current areas of interest are web services and security.

Dr.S.Vadivel has got his PhD degree in C*omputer Science* and Engg from I.I.T Madras, India by 1989.After that he worked in *Crompton Greaves* in Bombay as research executive for 3 years. Then he worked as A*ssistant Professor* in engg in Govt college at *Tamil Nadu*, *India* for 4 years. Then he joined as Research Lead in Think business networks a multinational software company in Tamil Nadu. He has joined BITs,Pilani-Dubai campus as faculty in CSE by Jan 2003 and currently working as *associate professor* in CSE in the same institute .He has 10 publications in various international journal and conferences. His current research interest are in web services and security, Embedded controllers ,data mining, and Architecture of enterprise software applications.