

## Technical report

Author: Johannes Nicolai <johannes.nicolai@hpi.uni-potsdam.de>  
Date: 03/01/2005

### **Evaluation of certain technologies supporting the transport of binary data in the scope of the ASG Deployment Web service**

#### 1. Intent / Motivation

The ASG has specified a deployment web service to enable its customers to deploy J2EE component resource files such as .war and .ear files from remote. These components are typically compressed and therefore have to be treated as binary data by the web service interface. The question that arises in this context is how to encode and transport this data in conjunction with the SOAP protocol. There are various technologies available that concentrate on answering that question. These technologies will be evaluated in this document according to the following criteria.

#### 2. Forces

When talking about binary data transportation and encoding methods, you have to consider at least five different forces: compatibility / interoperability, size of the resulting message / payload, performance and memory consumption while generating, transporting and parsing the message, ability to apply xml encryption / signing on the resulting message and the clarity of the corresponding WSDL file to the involved tools.

Of course, it is not possible to fulfil all criteria because some forces have a negative impact on others (e. g. compatibility versus performance), so the final solution will be a reasonable trade off and is determined by the importance of the specific points. In our context, interoperability is the primary goal.

#### 3. Pre- Conditions / Work already done

While evaluating the different technologies, we used the JBoss application server [10] that relies on the Apache Axis libraries to process web service requests in the SOAP format. We implemented both a "Soap with Attachments" and a "WS attachment" compatible web service and the corresponding client in order to get familiar with the Axis framework, its benefits and limitations.

#### 4. Alternatives

During the evaluation, we concentrated on the following technologies:

##### 4.1. Base 64 encoding

The Base 64 encoding is the simplest way to transport binary data via a web services. The data in question is transformed in a well defined way to the Base 64 format that only consists of 7 bit ASCII characters. These characters can be easily transported in an XML tag. This approach is interoperable with all implementations of web services because base64 is one of the basic types you can use to define the signatures of your web service operations.

Unfortunately, the encoding bloats the data to four thirds of the original size. As a consequence, the generation, sending and parsing processes tend to last a very long time and will consume a lot of memory and cpu resources (see [1] and [8]).

If you want to encrypt or sign your resulting message, you will have no problems with this approach since the output is "pure SOAP". The WSDL files are understood by any WSDL processing tool so that the generation of appropriate client and server stubs can be automated.

##### 4.2. Soap with Attachments

Soap with attachments is a standard proposed by Microsoft and Hewlett Packard (W3C note, see [3]) and defines how to transport SOAP and binary attachments in a MIME document.

With MIME encoding, the binary data is sent as a separate MIME part that is referenced by some elements of the SOAP MIME part. Due to the fact that binary data is transported as it is in the MIME part, the resulting message is not much bigger than the size of the binary data itself. Thus, processing and transportation are quite fast and do not consume a lot of memory since the MIME part containing the binary data has not to be parsed by the XML parser (see [1] and [8]).

This fact implies some problems with XML - signing and encrypting the resulting message because it is not clear how to treat the different MIME parts that do not contain XML.

This approach is not compatible to JAX RPC, WS -I or WSE .NET but Axis is able to handle this format. Unfortunately, it is neither capable to generate code from the corresponding WSDL file nor to generate a valid WSDL file from a valid implementation.

#### 4.3. WS Attachments

WS attachments are another technology proposed by Microsoft that is similar to Soap with Attachments but uses DIME instead of MIME (see [7]). DIME is not as flexible as MIME but supports message chunking and defines in its headers how many bytes the following part will consume.

Processing and generating DIME messages is considered to be the fastest method in this evaluation especially by large binary files, also the memory usage was the lowest in the whole field (see [1] and [8]).

XML encryption and signing is problematic due to the same reasons as with Soap with Attachments. The WSE .NET framework understands this format with the installed Web Service Extensions 2.0 and is able to generate code out of the corresponding WSDL files. Axis is also able to implement clients and servers that implement this technology but will not generate the WSDL files in the format specified by Microsoft (see [9]).

#### 4.4. WS - I Attachments

WS - I Attachments ([5]) is the successor of the Soap with Attachments standard. It also uses MIME bindings and the same WSDL format but defines another way to reference to the binary content.

Its performance and the problems with cryptographical treatment are comparable to the Soap with Attachment approach. JAX RPC supports WS - I attachments and provide tools to parse and generate the corresponding WSDL files / the corresponding code stubs.

Neither Axis nor WSE .NET support WS -I attachments.

#### 4.5. MTOM

MTOM (Message Transmission Optimization Mechanism, see [2]) is a new idea of Microsoft and was proposed to the W3C. The basic idea is to encode binary data as Base 64 in the information set but transform it to binary data during the transport.

With this approach, you can encrypt and sign the message but also have a little message size during the transport. Thus, processing time and memory consumptions lies in between of Base 64 and the MIME / DIME approaches.

MTOM is currently not supported by any tool. Microsoft plans to integrate it in WSE 3.0 (see [2] and [4]). The format of the corresponding WSDL files is already specified.

#### 4.6. Other approaches (Binary XML, Grid FTP, XSoap, ...)

Of course, there are a lot of other ways to cope with the binary transportation problem ([2]). Most of the industrial software platforms in use do not understand these protocols.

The procedure how to XML- encrypt and sign the corresponding messages is unspecified in almost all cases. The resulting messages tend to be smaller than Base 64 encoding but larger than DIME encoding. This also applies to the processing and generation time. Widely used tools do not support the generation or parsing of the corresponding WSDL files if these files are present at all.

## 5. Solution / Consequences

Since the J2EE components we plan to deploy tend to be not too big and the is not likely that the deployment service will be used too often, interoperability is our primary goal and we do not want to miss the possibility to have a well defined XML encryption and signing method, we have decided to use the simple Base 64 encoding approach.

The consequences are that we have to implement another web service and develop a suitable client for it. This web service will tend to consume a lot memory and will need a lot of time to process requests with a large amount of binary data. In contrast to this drawback, we will be compatible with every platform and can chose to use XML encryption / signing in a later phase of the project.

During a meeting it was decided to implement the new web service as a Java Service Endpoint Servlet, so that we can deploy our service on virtually every J2EE application server.

We decided in favour of a servlet and not of a stateless session bean because we have to store the received data on disk, an action that is not likely for a component in the business layer.

## 6. References

[1]

[http://www.atl.external.lmco.com/projects/QoS/compare/dist\\_oo\\_compare\\_ipc.html](http://www.atl.external.lmco.com/projects/QoS/compare/dist_oo_compare_ipc.html)  
Some detailed data of performance measurements comparing the presented alternatives

[2]

[http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnwebsrv/html/opaquedata.asp#binary\\_topic5](http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnwebsrv/html/opaquedata.asp#binary_topic5)  
A good comparison of the different technologies

[3]

<http://www.w3.org/TR/SOAP-attachments>  
Soap messages with attachments

[4]

<http://www.winfx247.com/247reference/messages/0/1206.aspx>  
Microsoft's position in this question

[5]

<http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html>  
WS -I attachment profile 1.0

[6]

<https://jax-rpc.dev.java.net/whitepaper/1.1.2/attachments.html>  
How JAX RPC supports attachments

[7]

<http://msdn.microsoft.com/archive/default.asp?url=/msdnmag/issues/02/12/dime/TOC.asp>  
Description of the DIME format and how to use it with web services

[8]

[www.wesc.ac.uk/resources/publications/pdf/AHM04/206.pdf](http://www.wesc.ac.uk/resources/publications/pdf/AHM04/206.pdf)  
A performance comparison of Base 64, MIME and DIME in conjunction with Grid FTP

[9]

[http://www.gotdotnet.com/team/xml\\_wsspecs/dime/WSDL-extension-for-DIME.htm](http://www.gotdotnet.com/team/xml_wsspecs/dime/WSDL-extension-for-DIME.htm)  
WSDL format description for WS - Attachments

[10]

<http://www.jboss.org/products/index>  
JBoss application server