

CF, Flex and the wonderful (?) world of web services and SOA

Kai Koenig
Software Solutions Architect

Ventego Creative Ltd
Wellington, NZ



About me...

- **Kai Koenig, Software Solutions Architect with Ventego Creative Ltd in Wellington, NZ**
- **Fields of work:**
 - Development: CF, Flex, Flash, Java
 - Consulting: Architecture, Frameworks, Performance, Code Reviews, Mentoring
 - Training: Adobe Certified Master Instructor for Flex, CF and Connect



Agenda (I)

- Why this session?
- Terminology
- SOA and web services
- Web service specifications
- Interoperability and how to achieve it – is it possible at all?



Agenda (II)

- CF, Flex and web services
- Behind the scenes
- Common pitfalls
- Some case studies
- Resources
- Contact



Why this session? (and a warning)

- During the last few months I worked on interesting integration projects leveraging web services and XML in CF and Flex.
- I've learned a lot about things that work well and other things that don't (but not just in CF and Flex) in the web services world and I've particularly learned that there's a significant lack of understanding for SOA and web services in ICT.



Why this session? (and a warning)

- **This session will NOT:**
 - Teach you how to write a CFC that exposes itself as a web service
 - Teach you how to call a webservice
 - Contain HUGE amounts of CF and Flex code
- **This session WILL:**
 - Hopefully remove some of the biggest confusion about the possibilities of web services and SOA
 - Give you an idea of common pitfalls when trying to build interoperable web service architectures
- **Vendors: don't take my comments personal! ☺**



Terminology

Web service:

1. Very general: The W3C defines a Web service as a software system designed to support interoperable Machine to Machine interaction over a network.
2. Common usage: the term refers to clients and servers that communicate XML messages that follow the SOAP-standard. Common is the assumption that there is also a machine readable description of the operations supported by the server, a description in the WSDL.



Terminology

- **SOAP is a communication protocol that is based on XML messages, WSDL is an XML description language for web service operations.**
- **Web services (as we want to understand them today) use both specifications mentioned above, but depending on who you ask, you'd get different and partly contradicting answers**



Terminology

- **SOA (service oriented architecture):**
 - The new buzzword (to be used together with Web 2.0, RIA, Social Recommendation Network etc. ☺)
 - Veeeeery high-level: A software design principle that assumes that a software architecture relies on service orientation, i.e. loosely coupled services that could be used as necessary at compile time or at run time of a software system
 - Note: SOA does not necessarily mean web services and XML!
 - Note 2: SOA is a fundamental principle and could be applied to various types of systems



SOA and web services

- **Pretty often, people talking about SOA mean web services or vice versa**
- **At lot of technologies and development platforms support web services nowadays, so do ColdFusion and Flex.**
- **Big players: .NET and Java**



SOA and web services

- **Funny story (via Ben Forta's blog):**

- CIO Magazine, "Are you ready for web services?"
- Question 8 out of 12:

I plan to develop Web services in:

- a. Java (5 points)
- b. .Net (5 points)
- c. other (such as Macromedia ColdFusion, Ruby on Rails, Ajax) (3 points)



SOA and web services

- **The reality:**

- There is no right or wrong when it comes to service orientation and SOA, but...
- ...SOA is way more than web services, XML, SOAP, WSDL, UDDI etc.
- Some keywords to be mentioned in this context:
 - Management of business processes
 - Change management
 - Corporate culture
 - Security, Reliability
 - etc.



Web service specifications

- **That's the fun part...**
- **The easiest ones:**
 - SOAP and WSDL (we've heard of those!) – pretty common
 - CF: <http://www.server.co.nz/abc.cfc?wsdl>
 - Flex: <mx:WebService>
 - At some point people recognized that web services are pretty basic and insecure



Web service specifications

- **More interesting stuff:**
 - WS-Security – to make sure your web service architecture is safe and secure
 - More than just https...
 - WS-Security applies encryption and certificates in the SOAP header by leveraging industry standards such as X.509, Kerberos etc.
 - WS-ReliableMessaging and WS-Reliability
 - Competition about which one to become "THE" standard
 - General idea: To ensure that a message that has been sent arrives at its destination – it's a reliable communication



Web service specifications

- WS-Addressing
 - XML structure for encapsulating addressing and routing information for web services messages
 - Message destinations, endpoints for reply and fault etc.
- WS-I
 - Interoperability... the dream of hooking up different web service engines with each other and it just works smoothly...



Interoperability

- **Development of interoperable web services still a challenge**
 - Type conflicts
 - Unsupported message formats
- **WS-I and the basic profile**
 - Guideline for using web service technology
 - Analysis tools for web service artifacts
 - Hints and tips for developers



Interoperability

- **Reasons for issues are the way how WS apps are developed:**
 1. Write code for a web service application in the language you're used to
 2. Created WSDL document for this code (automatic?)
 3. Generate client proxies from generated WSDL
- **Particularly step 2 causes the interoperability issues**
- **This approach is called "code first" – the web service interface is derived from the implementation**



Interoperability

- **Trouble: Code First is simple – and works great in homogeneous environments**
 - Tempts to use platform-specific types such as Vector (Java), DataSet (.NET)
 - There's no defined mapping between those types and XML data types



The golden Interoperability rule!

- XML is the least common denominator
- The solution for interoperability issues is “Contract First” – define the WSDL interface first and develop the web service implementation from the WSDL



Interop in “sandpit” projects

- No predefined schemas
- Code First works great
 - Just simple data types (String, int etc.)
 - A lot of tools (also CF) supports that approach
- Things WILL change in a real integration project
 - “But it used to work in the test project”



Interop in the “real world”

- **In big integration projects you'd need to have the WSDL schemas first**
 - Insurance industry: Origo and Accord standards
 - Tourism industry: OTA (Open Travel Alliance)
- **Involved dev teams could then develop tools towards the existing spec in “their” language**
- **The alternative:**
 - Generate WSDL automatically
 - Hope that generated WSDL fits pre-existing schema



CF, Flex and web services

- **Using web services with CF is a safe bet as long as you use simple data types or stay within CF and Java resp. connect to Flash/Flex clients.**
- **To be fair: Most other technologies work reasonable as well... but...**
- **There are some pitfalls!**



CF, Flex and web services

- In Flex, we have to deal with the web service implementation within the Flash Player.
- Again: Not really bad, no particular issue – BUT: black box



CF, Flex and web services

- **Common issue with both Flex and CF:**
 - The vendor basically decides which of the web service specifications are supported.
 - i.e. if you currently need WS-security, you basically can't achieve it with today's ColdFusion and Flex in a reasonable way
 - The same is true for other more sophisticated web services specifications and for other vendors (MS is much worse...)



Behind the scenes

- ColdFusion uses Apache Axis 1.x internally as its web service engine
- This is very good – de facto industry standard, open source etc. – and there are ways of updating the Axis engine
- **BUT: Axis is a tricky product and it's not really easy to get to the core of an issue when necessary.**



Behind the scenes

- **How does <CFINVOKE> really work?**
 - CF retrieves the target WSDL
 - WSDL is run through a tool named WSDL2Java (in your CF bin folder), this outputs Java source code that map to the data types defined in the WSDL XML
 - CF compiles the Java code
 - CF calls the stub methods of those Java classes and automatically converts any input arguments to XML documents of the appropriate format
- **This is more tricky as it sounds**



Behind the scenes

- **We have to deal with two issues:**
 - ColdFusion is itself an un-typed language, so the Axis integration has to take care of this issue in regards to converting the data to typed values.
 - Apache Axis is just ONE player in the web services toolkit market and there are known issues with .NET...



Common pitfalls

- **CF: dealing with complex nested data structures in web services**
- **In WSDL: complexType containing other complexType(s)**
- **The general solution:**
 - Eventually works fine with CF structs
 - Use Java2WSDL and WSDL2Java, those tools help to figure out how a Java stub needed to look and you could build you CF along those lines
 - Not perfect though as structs are basically untyped objects



Common pitfalls

- **Reasonably easy:**

```
<s:complexType name="Employee">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="fname" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="lname" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="age" type="s:int" />
  </s:sequence>
</s:complexType>

<Employee>
  <fname>John</fname>
  <lname>Smith</lname>
  <age>25</age>
</Employee>

<cfscript>
  stUser = structNew();
  stUser.fname = "John";
  stUser.lname = "Smith";
  stUser.age = 23;
  ws = createObject("webservice", "http://somehost/echosimple.asmx?wsdl");
  ws.domSomething(stUser);
</cfscript>
```



Common pitfalls

- **Flex: If you work with web services, and want to transport potential SOAP Faults into your Flex application, you will be surprised...**
- **The Flash player cannot read the body of the response if the HTTP response code != 200**
 - Solution: Use Flex Data Services WS proxy...
 - OR: force your web service to return a valid status code 200 in case it runs into a SOAP Fault (via a servlet filter in the case of Java)



Common pitfalls

- Besides the common RPC/encoded web services, there are document/literal web services.
- The industry is heading towards accepting this as the new standard, RPC/encoded web services might phase out rather sooner than later



Common pitfalls

- **Flex:** With some document/literal-style web services and particular Java web service engines, you have to specify your request as below:

```
<mx:operation name="openConnection"
  result="openConnectionRH(event)"
  fault="openConnectionFH(event)" resultFormat="object">
  <mx:request format="xml">
    <openConnection xmlns="http://www.a.com/webServices">
      <userName>abc</userName>
      <password>def</password>
      <progname>this</progname>
      <proglib>that</proglib>
      <apphost>myserver</apphost>
    </openConnection>
  </mx:request>
</mx:operation>
```



Common pitfalls

- **Problem:** `<mx:request format="xml">` doesn't enforce the new internal E4X XML data type in Flex 2 and you'd have to use the "deprecated" XML engine in Flex.



Some case studies (I)

- **Systems integration via web services between two telecommunication providers:**
- **A: Major Telco providing line, toll and broadband services to resellers**
 - Technologies: .NET backend system, complex web service API, secured with WS-security/encryption via X.509 certificates
- **B: reseller that needs to integrate with the services of A.**
 - Technologies: ColdFusion MX 7 Enterprise



Some case studies (I)

- **Major issue: ColdFusion MX 7's web service engine doesn't support WS-security features.**
- **Some ideas for a solution:**
 - Extend ColdFusion's Apache Axis engine with an Axis plugin for WS-security (wss4j)
 - Preferred way, but didn't work with ColdFusion MX 7!
 - Build a proxy that ColdFusion can talk to easily by leveraging a different level of security (potentially in .NET)
 - We went down that road!
 - Move away from ColdFusion at integration partner B
 - No way! ☹



Some case studies (I)

- **Final solution:**
 - We've built a Java client-server application running on the ColdFusion integration server of B.
 - CF application talks to an external Java wrapper via the client-server application
 - The external Java application (un)wraps all the encryption/security features and talks to integration partner A.



Some case studies (I)

- **Issues:**

- The external Java application leverages Java 5 features – so we couldn't integrate it directly with the ColdFusion JRE (CF 7 doesn't support Java 5 properly).
- We had to run two different JREs on the ColdFusion server – not optimal



Some case studies (II)

- **Integration of a Flex client application with a web service that wraps – you might believe it or not – a legacy COBOL application**
- **The project itself went rather smoothly, but we had to go through a pretty steep learning curve in terms of the tiny, little issues around the web service communication.**



Resources

- About not nested, complex data types in CF:
http://livedocs.adobe.com/coldfusion/7/htmldocs/wwhelp/wwhimpl/coomon/html/wwhelp.htm?context=ColdFusion_Documentation&file=00001554.htm
- About nested, complex data types in CF:
http://hcc.musc.edu/research/shared_resources/xml_complex_types_t_o_cf_structure_notes.cfm
- Tom Jordahl's blog (the CF/Axis bible):
<http://tjordahl.blogspot.com/>
- Thilo Frotscher: Best practices for interoperable web service apps:
<http://www.frotscher.com>
- Top ten tips for web service interoperability:
<http://blogs.msdn.com/smguest/archive/2004/08/12/213659.aspx>



Contact

Kai Koenig

Ventego Creative Ltd
154 Parkvale Rd
Wellington, New Zealand



Phone: +64 4 476 6781

Mobile: +64 21 928 365

kai@ventego-creative.co.nz

Blog: <http://www.bloginblack.de>

