CFML Web Services
Tips and Tricks

Charlie Arehart
charlie@carehart.org

(presented June 2006, updated Oct 2007)
Topics

- Testing Your Web Services
- Consuming Public Web Services
  - Including Amazon web services demo
- Dealing with Data Returned From Web Services
  - Including Processing .NET Datasets
- Web Service Details and Caveats
- Where to Learn More
About Your Speaker

- Independent consultant since April 2006
- 10 yrs CF experience (25 in Enterprise IT)
  - Member, Adobe Community Experts
  - Certified Adv CF Developer (4 - 7), Cert. Adobe Instructor
  - Co-author, ColdFusion MX Bible (Wiley)
  - Contributor to upcoming CF8 WACK books
  - Frequent contrib. to ColdFusion Dev Journal, blogs, lists
  - Until recently, President, Atlanta ColdFusion User Group
  - Now co-lead (w/ Ray Camden) Online ColdFusion Meetup
    (http://coldfusionmeetup.com)
  - Frequent speaker: UGs, conf’s worldwide
  - Living in Alpharetta, Georgia (north of Atlanta)
- Web home at www.carehart.org
  - Hosts my blog; lists all my past articles, presentations
  - UGTV: recordings of presentations by nearly 100 CFUG speakers
  - AskCharlie: per-minute telephone & web-based CF support
Quick Review

- Talk presumes you know basics to create/consume web services in CFML
  - Feature of CFMX, BlueDragon
- Simple example of invoking a web service in CFML:
  
  ```
  <cfinvoke webservice="http://localhost/demo/hello.cfc?wsdl"
         returnvariable="fromhello" method="GetHello">
    <cfoutput>#fromhello#</cfoutput>
  </cfinvoke>
  ```

- CFMX/BlueDragon handle details
  - Generates SOAP objects and publisher/consumer communications
  - Translates data into and back from XML for trip across the wire
Testing A Web Service

- Whether you’re trying to call a web service and wonder how it works, or writing a web service:
  - Can use several things, short of writing CFML, to test/invoke your web service
    - On browser URL
    - Using web-based tools
    - Using Dreamweaver MX
  - Some show web service available methods and the kind of data returned
    - Some allow you to run the web service
Viewing WSDL

- If URL for CFML web service is entered on browser, we see the resulting WSDL
Viewing Web Svcs in DW

- Can also view any web service or CFC using Dreamweaver MX/2004/8/CS3
  - Open a CFML page
  - From Application panel, select Components tab
  - Choose “web services” from drop down
  - Provide URL, and it identifies all the methods and input/output arguments of the web service

June 28th – July 1st 2006
Browsing Web Svcs in DW

- Can then drag/drop webservice method to create CFML code
  - CFINVOKE
  - CFINVOKEARGUMENTS
Browsing Web Svcs in Eclipse

- Same feature offered in Eclipse by way of Adobe’s ColdFusion Extensions for Eclipse (not CF Eclipse)
  - http://www.adobe.com/support/coldfusion/downloads.html#cfdevtools
  - Use Services Browser (Window>Show View>Other>ColdFusion>Services Browser)
    - It will take time to first load all CFCs on your system
    - But then click the icon just left of the minimize icon (“Show Web Services”)
      - Click red + to add a new WSDL URL
    - Will explore the web service like in DW example
    - Can right-click on WS URL or method to choose option to create CFINVOKE, CFObject, and/or createObject code
Other Web Service Viewing Tools

- [http://xmethods.net/ve2/Tools.po](http://xmethods.net/ve2/Tools.po)
  - Validates web service’s WSDL, see methods and properties
- [http://www.mindreef.net/tide/scopeit/start.do](http://www.mindreef.net/tide/scopeit/start.do)
  - Commercial tool with free web-based sample
  - Not only can view web service methods and properties
    - But also offers simulated input screens to execute a given method
    - Can view results as raw SOAP, xml, tree view, etc
Other Web Service Viewing Tools

- http://www.gmorpher.com/Morph/dynamo/main.jsp
  ✓ Works with any web service URLs
  ✓ Analyze and execute web services (execute requires login)
- Consider Microsoft InfoPath to test web services
- Also, Altova XMLSpy
  ✓ http://www.altova.com/products_ide.html
Web Service Viewing Tools (cont)

- Of course, with web-based tools your web service must be web-accessible
  - Can’t test code on your localhost unless you can offer public IP address, have turned off firewall, etc.
- Can use all these tools to test various publicly available web services, discussed next
Executing Web Service Via URL

- Can even execute the service by specifying method in URL:

  http://[host]/[dir]/[file].cfc?wsdl&method=[methodname]

  http://127.0.0.1/demo/hello.cfc?wsdl&method=GetHello
Executing Web Service Via URL

- Curiously, and perhaps as a surprise, CF returns result as WDDX packet
  ✓ use “view>source” in browser to see this
    
    ```xml
    <wddxPacket version='1.0'><header/><data><string>Hello World</string></data></wddxPacket>
    ```

- BlueDragon instead returns SOAP (XML) packet, just as it would to any web service client that would invoke it
  ```xml
  <?xml version="1.0" encoding="UTF-8" ?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getHelloResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <getHelloReturn xsi:type="xsd:string">Hello World!</getHelloReturn>
      </getHelloResponse>
    </soapenv:Body>
  </soapenv:Envelope>
  ```
Creating Web Service Alias

- In CFMX, can create an alias for the web service
  - Similar to how we define DSN names to abstract details of DB access
  - See Admin console’s “Web Services” link under “Data & Services” in left nav bar
Creating Web Service Alias (cont.)

- Can then use that alias for web service name wherever would have used full URL (CFINVOKE, CFOBJECT, CreateObject)
- Will see that CFMX already put an entry in for any web service that’s been called
  - Can edit it and simply provide an alias in the “web service name” field
- That admin console listing of an entry for each web service browsed has another use…
Internal Auto-Caching of WSDL

- It reflects the fact that CF has cached the WSDL (created a proxy/stub) for the web service browsed
  - Sometimes, it’s helpful (indeed needed) to refresh/remove that cache proxy/stub
    - Especially when building/testing your own CFCs as web services, with frequent changes
  - Simply use the “refresh” icon/link on list of web services
    - Also option to “Update web service” when editing svc
  - What if no access to Admin console?...
Refresh WSDL Stub in Code

- Do it programmatically

<!--- may want to place in secured directory to prevent denial of service attacks or other mischief --->

<cfparam name="url.wsdlurl"
default="http://soap.amazon.com/schemas2/AmazonWebServices.wsdl">

<cfscript>
// get the web service wsdl file as URL String
wsdl = url.wsdlurl;
factory = CreateObject('JAVA', "coldfusion.server.ServiceFactory");
RpcService = factory.XmlRpcService;
RpcService.refreshWebService(wsdl);
</cfscript>


June 28th – July 1st 2006
Consuming Public Web Services

- Commercial examples
  - http://www.google.com/apis/
  - http://www.usps.com/webtools/

- Exposing services/data, often even permitting transactions
  - Sometimes free, sometimes for a fee
Amazon Web Services

- Exposes Amazon’s product data and E-Commerce functionality
  - Detailed Product Information on all Amazon.com Products
  - Access to Amazon.com Product Images
  - All Customer Reviews associated with a Product
  - Extended Search
  - Remote Shopping Cart
  - Amazon Wish List Search
  - And more
Amazon Web Services

- Leverage Amazon data and functionality to power your own E-Commerce business
  - Requires Account, Account Number, Access Key ID, Subscription ID, Developer Token
  - Free of charge
    - Limited to one request per second per IP address
- Amazon supports both SOAP and REST approaches
  - Each will require different URL for web service
SOAP vs REST

- Focus in most CFML docs is on creating/consuming SOAP-based web services
  - SOAP packages data in a special XML envelope
  - SOAP servers and clients (like CF and BD) handle the details
  - Client gets back data as same datatype sent (string, number, etc.)
SOAP vs REST

- Some web services instead serve data via HTTP requests and return plain XML (XML over HTTP)
  - Could call such web services using CFHTTP
    - Or test via URL in browser
    - Or use CFXMLRPC tag in BlueDragon
  - You then process result using CFMX/BlueDragon functions for XML
- Approaches:
  - Xml-rpc: http://www.xmlrpc.com/
- Benefit: easy to do via browser URLs, returns XML (IE formats nicely)
Calling Amazon Web Service

- Docs at their site describe the many available methods (and their properties)
  - And how to call their web service
  - Available “XML ScratchPad” tool shows how to build a URL to use for various services
- Is about more than just books now
  - Here’s how to find seafood restaurants in Seattle
Calling Amazon Web Service

- **Via URL (browser or CFHTTP)**
  
  http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&SubscriptionId=[id]&Operation=ItemSearch&SearchIndex=Restaurants&Cuisine=seafood&Neighborhood=downtown&City=Seattle

- **Via CFHTTP**

  ```cfhttp
  url="http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&SubscriptionId=[id]&Operation=ItemSearch&SearchIndex=Restaurants&Cuisine=seafood&Neighborhood=downtown&City=Seattle">
  <cfdump var="#cfhttp.filecontent#"/>
```
Traditional Book Request

<!--- invoke_amazon_soap.cfm --->
<!--- part 1 of 3 (join together to run as one) --->
<cfparam name="url.keyword" default="coldfusion">
<cfscript>
  aKeywordRequest = structnew();
  aKeywordRequest.devtag="yourtag";
  aKeywordRequest.keyword=url.keyword;
  aKeywordRequest.mode="books";
  aKeywordRequest.page="1";
  // aKeywordRequest.sort="+salesrank";
  aKeywordRequest.tag="webservices-20";
  aKeywordRequest.type="lite";
</cfscript>
Traditional Book Request

<!--- part 2 of 3 (join together to run as one)--->

<cftry>
  <cfinvoke webservice="http://soap.amazon.com/schemas2/AmazonWebServices.wsdl"
    method="KeywordSearchRequest" returnvariable="aProductInfo">
    <cfinvokeargument name="KeywordSearchRequest" value="#aKeywordRequest#"/>
  </cfinvoke>
  <cfcatch>
    <cfif cfcatch.message contains "Could not perform web service invocation">
      No records found.
      <Cfdump var="#cfcatch#">
      <cfelse>
      Unexpected error in invocation of web service.
      <cfdump var="#cfcatch#">
    </cfif>
  </cfcatch>
</cftry>

June 28th – July 1st 2006
Traditional Book Request

<!---- part 3 of 3 (join together to run as one)--->
<cfoutput>
Total Found: #aProductInfo.TotalResults#<br>
Searched for: #url.keyword#
</cfoutput>

<table border="2">
<tr><td></td><td><strong>Title</strong></td><td><strong>Author</strong></td></tr>
<cfoutput>
<cfloop index="i" from="1" to="#arraylen(aProductInfo.details)#" step="1" >
<tr>
<td><img src="#aProductInfo.details[i].ImageUrlSmall#"></td>
<td>#aProductInfo.details[i].ProductName#</td>
<td>#aProductInfo.details[i].authors[1]#</td>
</tr>
</cfloop>
</cfoutput>
</table>
<table>
<thead>
<tr>
<th>Title</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macromedia ColdFusion MX 7 Web Application Construction Kit</td>
<td>Ben Forta</td>
</tr>
<tr>
<td>(Web Application Construction Kit)</td>
<td></td>
</tr>
<tr>
<td>Learning PHP 5</td>
<td>David Sklar</td>
</tr>
<tr>
<td>ColdFusion MX Web Application Construction Kit, Fifth Edition</td>
<td>Ben Forta</td>
</tr>
<tr>
<td>Advanced Macromedia ColdFusion MX 7 Application Development</td>
<td>Ben Forta</td>
</tr>
<tr>
<td>Advanced Macromedia ColdFusion MX Application Development, Third Edition</td>
<td>Ben Forta</td>
</tr>
<tr>
<td>Macromedia Dreamweaver 8 with ASP, ColdFusion, and PHP : Training from the Source (Training from the Source)</td>
<td>Jeffrey Bardzell</td>
</tr>
</tbody>
</table>
Other Public Web Services

- Other Public Services - Directories
  - http://www.xmethods.net
  - http://www.webservicex.net/
XMethods.NET

Welcome to XMethods.

Emerging web services standards such as SOAP, WSDL, and UDDI will enable system-to-system integration that is easier than ever before. This site lists publicly available web services.

Programmatic Interfaces

Access XMethods through a variety of interfaces:
- UDDI v2
- WCF-Inspection
- RDSL
- SOAP
- DISCO

Read about the TRY IT feature.

Recent Listings [View the FULL LIST]

<table>
<thead>
<tr>
<th>Publisher</th>
<th>Style</th>
<th>Service Name</th>
<th>Description</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbulNasheed</td>
<td>RPC</td>
<td>Try It</td>
<td>Get Values from the Queue</td>
<td>SOAPLite</td>
</tr>
<tr>
<td>XWebServices</td>
<td>DOC</td>
<td>Try It</td>
<td>XML/Soap based Web Service which validates Email addresses for client</td>
<td>MS .NET</td>
</tr>
<tr>
<td>nulllogone.net</td>
<td>DOC</td>
<td>Try It</td>
<td>matches the description of a message against the filter profiles</td>
<td>MS .NET</td>
</tr>
<tr>
<td>nulllogone.net</td>
<td>DOC</td>
<td>Try It</td>
<td>All services for the Wotraman (semantic word space)</td>
<td>MS .NET</td>
</tr>
<tr>
<td>nulllogone.net</td>
<td>DOC</td>
<td>Try It</td>
<td>All services for the messages in the OLI it network</td>
<td>MS .NET</td>
</tr>
</tbody>
</table>

June 28th – July 1st 2006
Calling ValidateEmail Service

- ValidateEmail is at webservicesx.net

```cfs
<cfobject
webservice="http://www.webservicex.net/ValidateEmail.asmx?WSDL" name="checkmail">

<cfparam name="email"
default="charlie@carehart.org">
<cfoutput>#email# is</cfoutput>
<cfif checkmail.IsValidEmail(email)>
good email
<cfelse>
bad email
</cfif>
address<p>
```
Calling ValidateEmail Service

- You can call that as filename.cfm?email=x@y.com
- Beware
  - some mail servers will return true ("good") for any email address, even if not a valid one at that domain. At least the domain name is validated in such instances.
Dealing with Data Returned From Web Services

- Can publish many datatypes via web svc
  - Simple string
  - Array
  - Structure
  - Array of structures
  - CFML query result set (with a caveat, discussed later)
  - XML object (using CFMX and BlueDragon’s support of XML)
  - To name a few
Determining Data Types

- How can CFML web svc client know the type of data returned from a web service?
  - It it a query? an array? a structure? something else?
    - Must know type to determine how to process
- See typeof UDF:
  - http://cflib.org/udf.cfm?ID=689
- Reports if something is a array, struct, query, string, date, numeric, boolean, binary, wddx, xml object, or even a custom function (udf)
Parsing a more elaborate web service result

```xml
<cfinvoke
    webservice="http://ws.invesbot.com/stockquotes.asmx?WSDL"
    method="getQuote"
    returnvariable="a_GetQuoteResponse_GetQuoteResult">
    <cfinvokeargument name="symbol" value="msft"/>
</cfinvoke>

<cfset resultarray = a_GetQuoteResponse_GetQuoteResult.get_any()>
<cfset arrayelement = resultarray[1]>
<cfoutput>
#htmlcodeformat(arrayelement.toString())#
</cfoutput>

<cfset xml = xmlparse(arrayelement.toString())>
<cfdump var="#xml#">
```
Returning CF Query Results

- What if you try to return a CF Query resultset over web services to a non-CF consumer?
  - They won’t understand it
  - Can instead convert into an array of structures
  - Consider following UDFs at the cflib.org site
    - QueryToArrayOfStructures: http://cflib.org/udf.cfm?ID=10
    - ArrayOfStructuresToQuery: http://cflib.org/udf.cfm?ID=287
    - QueryToStructOfArrays: http://cflib.org/udf.cfm?ID=470
Calling a .NET Web Service

- What if you want to call a .NET web service?
  - See next slide for code of such an .ASMX
- No difference in invocation in CFML

```cfinvoke
  webservice="http://127.0.0.1/demo/HelloWorld.asmx?wsdl" method="HelloWorld"
  returnvariable="gethello">

  <cfdump var="#gethello#">
```
Calling a .NET Web Service

- Saved in /demo/HelloWorld.asmx

```csharp
<%@ WebService Language="C#" Class="Demo.HelloWorldService" %>
using System.Web.Services;
namespace Demo
{
    public class HelloWorldService: WebService
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```
.NET Web Service Datasets

- What if .NET web service returns DataSet?
  - From CFMX, quite painful. Object returned is a complex object of ill-defined xml entries
    - See CFDJ article, “Crossing the .NET Divide: CFMX, Web Services, and .NET”
      - http://coldfusion.sys-con.com/read/47199.htm
    - He offers a UDF to help with the problem
    - He also covers moving arrays, structs from .NET to CFMX over web services
  - An argument can be made that .NET web services **should not** return datasets
    - http://www.theserverside.net/articles/showarticle.tss?id=Top5WSMistakes
.NET Web Service Datasets

- From BlueDragon.NET, the issue of consuming a .NET web service returning a dataset is pure simplicity
  - No need to manually convert xml into CF query result. BD.NET does it for you

```cfm
<cfinvoke
    webservice="http://127.0.0.1/demo/GetInfo.asmx?wsdl"
    method="ShowSuppliers" returnvariable="suppliers"
    str="USA">
<cfdump var="#suppliers#">
```
### Debugging Information

BlueDragon .NET Evaluation 6,2,1,302
Web Service Details/Caveats

- **Exception Handling**
  - Web service requests may fail
  - Consider `cftry/cfcatch` to detect/handle errors

- **Timeout**
  - CFMX 6.1 added ability to timeout web service requests
    - how long you’re willing to wait for a reply
Web Service Details/Caveats

**Security**

- Can secure CFC using either web server authentication
  - just as you can limit access to any web page
  - CFInvoke offers USERNAME/PASSWORD
- Can secure in CFML using ROLE attribute on CFFUNCTION
  - Tied to CFLOGIN/CFLOGINUSER tags
  - See CFMX documentation for more details
Web Service Details/Caveats

- Caching Web Service object
  - ✓ As we might cache a query resultset if it doesn’t change often, can do with web svc
  - ✓ No current feature to cache web service results
    - Can do it yourself, storing result in shared scopes (session/application/server)
    - Use some timing mechanism to determine when to refresh result, re-execute web service invocation
More You Can Learn

- CFMX docs elaborate on many additional topics
  - Working with WSDL files
  - Consuming web svcs not generated by CFMX
  - Calling web services from a Flash client
  - Catching errors when consuming web services
  - Configuring web svcs in CFMX Administrator
  - Conversions between CF/WSDL datatypes
  - Defining data types for web services
  - Handling complex data types
  - Integrating with Dreamweaver MX
  - **New in CF7: Using SOAP Request/Response Headers**
Learning More

- Macromedia Documentation
  - 6.1: *Developing ColdFusion MX Applications*, Chapter 32
  - 7: *ColdFusion MX Developer's Guide*, Chapter 36
  - Available at livedocs.macromedia.com

- Books
  - CFMX Bible, Wiley (Churvis, Helms, Arehart), Chapter 25
  - Programming ColdFusion MX (Brooks-Bilson), Chapter 25
  - And others
Learning More

- See articles offered in Wednesday talk, “Creating and Consuming Web Services with CFML”
- Also CommunityMX Articles
  - Oct 04, “Calculating Shipping Costs with UPS Online Tools”
  - Apr 03, “Consuming the CMX Web Service with ColdFusion”
  - May 03, “Using the Google API from ColdFusion”
Learning More

- Setting/getting cookies in SOAP request

- Dave Shuck’s CF Web Services examples

- Using CFMX Sniffer to watch request/response
  - See last section of WS chapter in “Developing CFMX Applications” (CFMX 7)

- Ron West’s Advanced WS CFUnited presentation
Summary

- Very easy to create/consume web svcs in CFML
- Several useful tools for testing them
  - Some in DWMX, some in browser, some as web sites
- Several public web services to explore
  - Including Amazon and many others
- Useful techniques for processing returned data
  - Including sharing CF data to other clients, .NET integ.
- Keep security, stub/output caching, other tips in mind

Questions: charlie@carehart.org