Web Services-Problems

Contents

Problems

This section describes some problems and bugs that sometimes occur when using *WebObjects Web Services*.

DirectToWebService can't return a WSDL with secure HTTPS references in it

Authors: Francis Labrie
Affected products: *WebObjects* 5.2.x, 5.3.x
Bug reference: rdar://3546304

Problem:

A *DirectToWebService* defined Web Services doesn't return correct WSDL document, even if the correct procedure (see Secure Web Services) is followed. So only classes oriented Web Services manually registered with the `com.webobjects.appserver.WOWebServiceRegistrar` class seems to generate a correct WSDL.

Solution:

Darel Lee from Apple told me that right now, the dynamic WSDL generation is not exposed to developers so there currently isn't a clean solution to perform this. One workaround is to hardcode rules (of `com.webobjects.directtoweb.Assignment` type) with the `serviceLocationURL` key for each operation that you want to use secure HTTPS references. For instance:

```
((operationName="anOperation") and (serviceName="Service")) ->

serviceLocationURL="https://host.net/cgi-bin/Service.woa/ws/Service"
```

If you need all operation to be called using the secure protocol, you can also define a more generic rule like this one:

```
(serviceName="Service") ->

serviceLocationURL="https://host.net/cgi-bin/Service.woa/ws/Service"
```

SOAP serializers and deserializers registered with `WOWebServiceRegistrar` class doesn't appear in the WSDL schema

Authors: Francis Labrie
Affected products: *WebObjects* 5.2.x, 5.3.x
Bug reference: rdar://3546330

Problem:

Custom SOAP serializers and deserializers registered to Web Services with `com.webobjects.appserver.WOWebServiceRegistrar` class are never added to the types / schema definition of the WSDL. The only type definitions shown are the following:

```
<types>
    <schema targetNamespace="http://lang.java/" xmlns:soapenc=
      "http://schemas.xmlsoap.org/soap/encoding/" xmlns=
      "http://www.w3.org/2001/XMLSchema">
      <complexType name="Class">
         <sequence/>
      </complexType>
      <complexType name="ArrayOf_xsd_any">
        <complexContent>
          <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType" wsdl:arrayType=
              "xsd:any[]"/>
          </restriction>
        </complexContent>
      </complexType>
      <element name="ArrayOf_xsd_any" nillable="true" type=
        "lang:ArrayOf_xsd_any"/>
    </schema>
    <schema targetNamespace="http://www.apple.com/webobjects/
      webservices/soap/" xmlns:soapenc="http://schemas.xmlsoap.org/
      soap/encoding/" xmlns="http://www.w3.org/2001/XMLSchema">
      <complexType name="EOGlobalID">
        <element name="entityName" type="xsd:string"/>
        <element name="primaryKeys" type="lang:ArrayOf_xsd_any"/>
      </complexType>
      <element name="EOGlobalID" type="tns:EOGlobalID"/>
      <complexType name="EOEnterpriseObject">
        <element name="entityName" type="xsd:string"/>
        <element name="globalID" type="webobjects:EOGlobalID"/>
        <element name="properties" type="soapenc:Struct"/>
      </complexType>
    </schema>
  </types>
```

Solution:

You must know that all complexe data types referred in operations will be added to the WSDL types definition. But if a complexe type makes references to others complexe types, you should make sure you add proper calls in the `writeSchema(Types)` method of the class serializer. For example:

```
public boolean writeSchema(Types types)
throws Exception {
 ...
 // Add type for Foo
 String prefixedName = types.writeType(Foo.class, _FooQName);
 ...

 return true;
}
```

Unfortunately, I don't know a dynamic workaround for all possible cases right now. At least a static and complete WSDL can be shared through a direct action, but it's not very handy though...

DirectToWebService can't return a WSDL with custom namespace and definitions name in it

Authors: Francis Labrie
Affected products: *WebObjects* 5.2.x, 5.3.x
Bug reference:

Problem:

A *DirectToWebService* defined Web Services can't return a WSDL with custom values for properties like namespaces and definitions name. Worse, the generated namespace can even contains WebObjects application instance number or the wrong hostname.

Solution:

Base on a tips from Darel Lee, I've found in the `com.webobjects.webservices.generation._private.WOWSDLTemplate` class some extras key definitions read from the `user.d2wmodel` *DirectToWebService* rule file. For instance:

- **serviceLocationURL:** a key that allow the setting of the location URL for an operation. This is usefull if you need your WebServices to be reached using a secure HTTPS reference;
- **WSDLDefinitionName:** a key that allow the definitions name change. So instead of having "ServiceNameDefinition", you can set this value;
- **WSDLTargetNamespace:** a key that allow the namespace change. This is the most usefull: you can avoid dynamic generation depending on WebObjects HTTP Adaptor with this.

Here is an example of rule definition changing the above values:

```
(se(serviceName="Service") ->

WSDLTargetNamespace="https://host.net/cgi-bin/Service.woa/ws/Service"
(serviceName="Service") ->
        WSDLDefinitionName="AnotherDefinition"
((operationName="anOperation") and (serviceName="Service")) ->

serviceLocationURL="https://host.net/cgi-bin/Service.woa/ws/Service"
```

WOWebServiceClient class can't connect to a server that requires an authentication

Authors: Francis Labrie
Affected products: *WebObjects* 5.2.x, 5.3.x
Bug reference: rdar://3568441

Problem:

The `com.webobjects.webservices.client.WOWebServiceClient` class can't connect to a server that requires a Basic HTTP Authentication despite the fact this class offers a way to register a security delegate (see `setSecurityDelegateForServiceNamed(Object, String)` instance method).

Normally, the `processClientRequest(MessageContext)` delegate method (see `com.webobjects.webservices.support.WOSecurityDelegateinterface` documentation) would allow an easy way to set a username and a password to the message context. But there is a problem related to the design of the class: to register a security delegate, the `WOWebServiceClient` class has to fetch the Web Services Definition Language (WSDL) XML document. But to get access to this WSDL, an authentication header must be set. This is the classic chicken and egg problem...

Solution:

The best would be to add a default method to `WOWebServiceClient` class to register a default security delegate that is not related to a service name before the class fetch the WSDL. But unfortunately, all key methods that would allow this kind of behavior change are privates, so subclassing is not a solution...

But a workaround is still possible:

1. Fetch the WSDL document yourself and store it to the local filesystem, using the java.net.URL instance and setting up the Basic HTTP Authentication header field of the `java.net.URLConnection` yourself;
2. Instanciate another `java.net.URL` class that refer to the local WSDL document file;
3. Instanciate the `com.webobjects.webservices.client.WOWebServiceClient` class using the file URL;
4. Set for each service a security delegate that will add the proper credential information for the Basic HTTP Authentication.

That's it. It looks like a big hack, but it works...

Web Services can't return a WSDL with secure HTTPS references specifying port other than the default 443

Authors: Francis Labrie
Affected products: *WebObjects* 5.2.x, 5.3.x
Bug reference: rdar://4196417

Problem:

HTTPS protocol references can be published in Web Services WSDL. But unfortunately, WebObjects seems to ignore ports other than the default 443.

This problem is related to the bad way `com.webobjects.appserver.WORequest` builds the URL prefix: if the protocol is secure and no port (i.e. 0) is set when calling the `_completeURLPrefix(StringBuffer, boolean, int)` method, the port will always be 443. Unfortunately, Web Services `com.webobjects.appserver._private.WOWebService` class seems to call this method without setting the port number.

Solution:

To work around this bug, you can subclass the `com.webobjects.appserver.WORequest` class like this:

```
package com.smoguli.appserver;

import com.webobjects.appserver.WORequest;
import com.webobjects.foundation.NSData;
import com.webobjects.foundation.NSDictionary;

/**
 * This class provide fixed {@link com.webobjects.appserver.WORequest}
```

```
methods.
 * To use it, just overload the {@link
com.webobjects.appserver.WOApplication.
 * createRequest(String,String,String,NSDictionary,NSData,NSDictionary)}
method
 * to instanciate this class instead.
 *
 * @author              Francis Labrie <francis.labrie at smoguli.com>
 */
public class WOFixedRequest extends WORequest {

        /**
         * @see
com.webobjects.appserver.WORequest#WORequest(String,String,String,
         * NSDictionary,NSData,NSDictionary)
         */
        public WOFixedRequest(String method, String url, String
httpVersion, NSDictionary headers, NSData content, NSDictionary info) {
                    super(method, url, httpVersion, headers, content,
info);
        } // WOFixedRequest

        /**
         * This method builds the URL prefix into the
<code>urlPrefix</code> buffer
         * with the appropriate protocol (<code>http</code> or
<code>https</code>)
         * and the right TCP port. But unlike the {@link
com.webobjects.appserver.
         * WORequest#_completeURLPrefix(StringBuffer,boolean,int} method,
it
         * supports secure HTTP protocol (<code>https</code>) with port
other than
         * <code>443</code>, even if the <code>port</code> parameter is set
         * <code>0</code>.
         *
         * @param urlPrefix                             the buffer that
receives the contructed URL.
         * @param isSecure                              a flag indicating
if the protocol is secure.
         * @param port                                  the port number.
         */
        public void _completeURLPrefix(StringBuffer urlPrefix, boolean
isSecure, int port) {
                if(isSecure && (port == 0)) {
                        String serverPort;

                        serverPort = _serverPort();
                        if((serverPort != null) &&
!serverPort.equals("443")) {
                                try {
                                        port =
Integer.parseInt(serverPort);
```

```
                                } catch(NumberFormatException exception)
{} // catch
                        } // if
                } // if

                super._completeURLPrefix(urlPrefix, isSecure, port);
```

```
        } // _completeURLPrefix
} // WOFixedRequest
```