**PWS Java sample program**

This document will show the user how to consume the PWS wsdl to create Java proxy classes. Sample code for the Authorize transaction will be provided also.

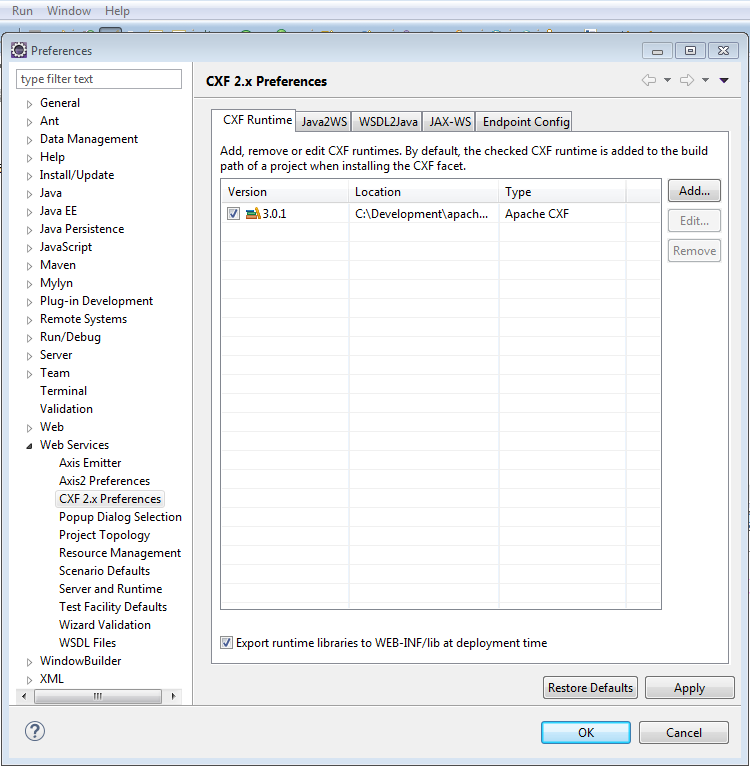
**Resources used:**

* Eclipse Luna - <https://www.eclipse.org/downloads/>
* Java 6 - <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html>
* Included in Maven POM.xml file:
  + Apache CXF 3.01 – used to consume WSDL and build proxy classes for sending/receiving SOAP messages.
    - <http://cxf.apache.org/download.html>
  + GSON 2.2.4 – used to convert/revert JSON objects.
    - <http://mvnrepository.com/artifact/com.google.code.gson/gson/2.2.4>
  + Log4j – used for logging.
    - <http://mvnrepository.com/artifact/log4j/log4j>

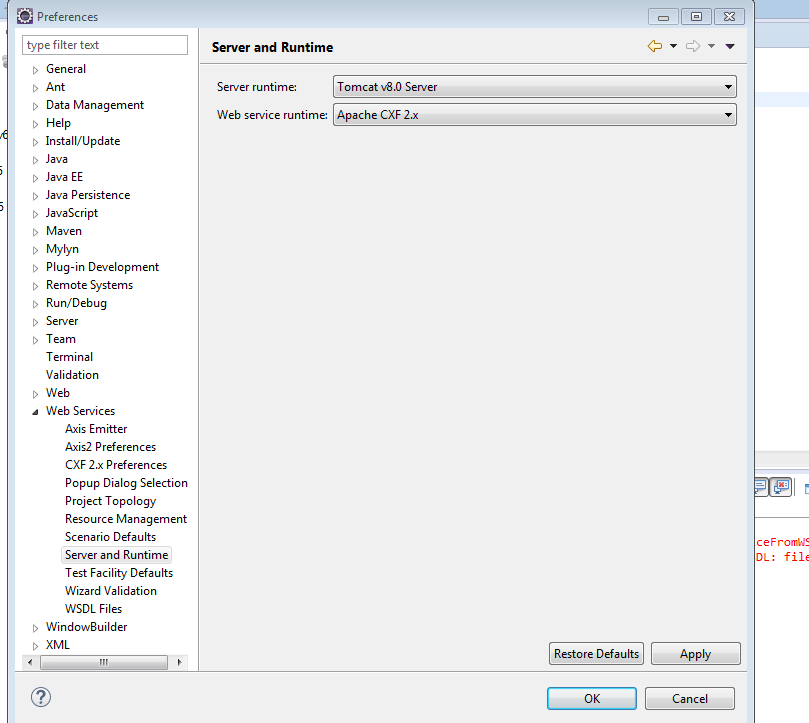
**Setting up Apache CXF**

**NOTE: this is only required if you are going to PWS direct. If going through Apigee, this is not required.**

1. Download CXF to your local environment.
2. In Eclipse navigate to Window -> Preferences -> Web Services -> CXF2.x Preferences.
   1. Under the CXF Runtime tab, add the location of the CXF folder.



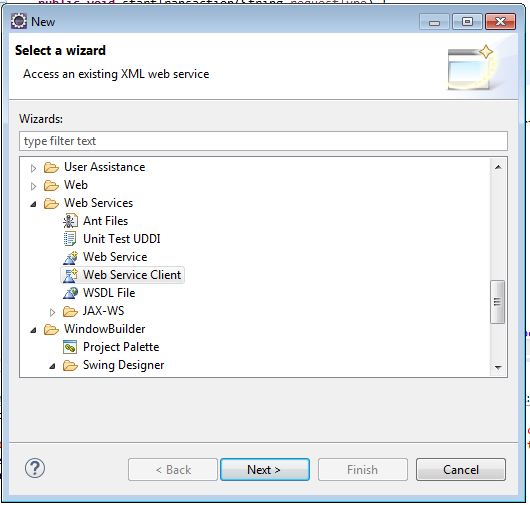
* 1. In the same location, navigate to Server and Runtime (Window -> Preferences -> Web Services -> Server and Runtime. Choose your server and then set the Web Service Runtime environment to Apache CXF 2.x.



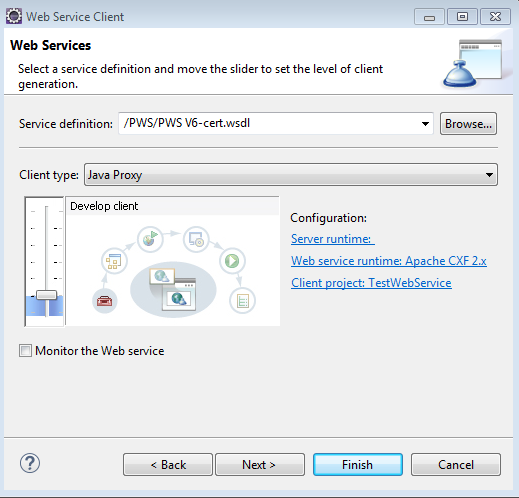
* 1. Now you are ready to create your web service client.

**Create JAX-WS web service client using Apache CXF**

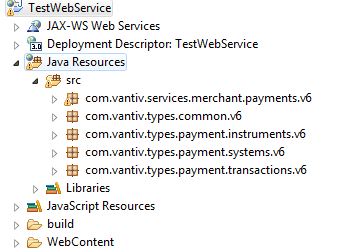
1. Create a new Dynamic Web Project within eclipse.
2. Right click on your project, New -> Other -> Web Services -> Web Service Client



1. In Service definition, browse to your WSDL file. (Must be in your workspace)
   1. Set the slider to “Develop Client”
   2. Make sure Web Service Runtime is set to Apache CXF 2.x



1. Make sure the proxies were created under Java Resources -> src in the project navigator.



1. That’s it, you’ve now consumed the WSDL and are ready to start using the PWS api.

**Sample code initializing the client**

1. Sample code initializing the client.
2. You will need the handlers to set the username/password within the header unless you prefer to use a different way. The two handlers needed are provided below.

**import** java.util.Set;

**import** javax.xml.namespace.QName;

**import** javax.xml.soap.SOAPElement;

**import** javax.xml.soap.SOAPEnvelope;

**import** javax.xml.soap.SOAPHeader;

**import** javax.xml.ws.handler.MessageContext;

**import** javax.xml.ws.handler.soap.SOAPHandler;

**import** javax.xml.ws.handler.soap.SOAPMessageContext;

/\*\*

\*/

**public** **class** WsseHeaderHandler **implements** SOAPHandler<SOAPMessageContext> {

**public** **static** **final** String ***WSSE\_SECEXT\_XSD\_URL*** = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd";

**public** **static** **final** String ***WSSE\_UTIL\_XSD\_URL*** = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd";

**public** **static** **final** String ***WSSE\_PASSWORD\_XSD\_URL*** = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText";

**private** String username;

**private** String password;

**public** WsseHeaderHandler(String username, String password) {

**this**.username = username;

**this**.password = password;

}

**public** **boolean** handleMessage(SOAPMessageContext smc) {

Boolean outboundProperty = (Boolean) mc.get(MessageContext.***MESSAGE\_OUTBOUND\_PROPERTY***);

**if** (outboundProperty.booleanValue()) {

**try** {

SOAPEnvelope envelope = smc.getMessage().getSOAPPart().getEnvelope();

SOAPHeader header = envelope.getHeader();

**if** (header == **null**)

header = envelope.addHeader();

SOAPElement security =

header.addChildElement("Security", "wsse", ***WSSE\_SECEXT\_XSD\_URL***);

SOAPElement usernameToken = security.addChildElement("UsernameToken", "wsse");

usernameToken.addAttribute(**new** QName("xmlns:wsu"),***WSSE\_UTIL\_XSD\_URL***);

SOAPElement usernameElement =

usernameToken.addChildElement("Username", "wsse");

// System.out.println("setting username to " + username );

usernameElement.addTextNode(username);

SOAPElement passwordElement =

usernameToken.addChildElement("Password", "wsse");

passwordElement.setAttribute("Type", ***WSSE\_PASSWORD\_XSD\_URL***);

//System.out.printf("setting password to %s%n", password);

passwordElement.addTextNode(password);

} **catch** (Exception e) {

e.printStackTrace();

}

}

**return** outboundProperty;

}

**public** Set getHeaders() {

// throw new UnsupportedOperationException("Not supported yet.");

**return** **null**;

}

**public** **boolean** handleFault(SOAPMessageContext context) {

// throw new UnsupportedOperationException("Not supported yet.");

**return** **true**;

}

**public** **void** close(MessageContext context) {

// throw new UnsupportedOperationException("Not supported yet.");

}

}

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.xml.ws.handler.Handler;

**import** javax.xml.ws.handler.HandlerResolver;

**import** javax.xml.ws.handler.PortInfo;

**import** javax.xml.ws.handler.soap.SOAPHandler;

**import** javax.xml.ws.handler.soap.SOAPMessageContext;

/\*\*

\*/

**public** **class** PaymentsHandlerResolver **implements** HandlerResolver {

**private** **final** List<Handler> handlerChain = **new** ArrayList<Handler>();

**public** PaymentsHandlerResolver(String username, String password) {

WsseHeaderHandler loginHeaderHandler = **new** WsseHeaderHandler(username, password);

handlerChain.add(loginHeaderHandler);

}

**public** **void** addMessageHandler( SOAPHandler<SOAPMessageContext> handler )

{

handlerChain.add(handler);

}

**public** List<Handler> getHandlerChain(PortInfo portInfo) {

**return** handlerChain;

}

}

1. To test the client connection, you can use a simple echo test:

**public** **boolean** invokeEchoTest() {

System.***out***.println("Validating connection with echo test...");

**boolean** b = **false**;

EchoRequest echo = **new** EchoRequest();

Date d = **new** Date();

String test = "This is my echo test: " + d.getTime();

echo.setTest(test);

// This command can be used to see the raw xml request.

// com.sun.xml.internal.ws.transport.http.client.HttpTransportPipe.dump

// = true;

EchoResponse resp = **new** EchoResponse();

**try** {

resp = client.echo(echo);

} **catch** (Exception e) {

e.printStackTrace();

}

// System.out.println(resp.getResponse());

**if** (resp.getResponse().equals(test)) {

System.***out***.println("Echo test successful");

b = **true**;

} **else**

System.***out***.println("Echo test did not return the expected value");

**return** b;

}

1. To send an authorize request, you must create an AuthorizeRequest object:
   1. In this method, I use a variable globals, which is another class that contains all the values.

**public** AuthorizeRequest createAuthorizeRequest() {

auth.setDraftLocatorId(globals.getDraftLocatorId());

auth.setMerchantRefId(globals.getMerchantRefId());

// auth.setNetworkResponseCode(globals.getNetworkResponseCode());

auth.setPurchaseOrder(globals.getPurchaseOrder());

auth.setReferenceNumber(globals.getReferenceNumber());

auth.setReportGroup(globals.getReportGroup());

auth.setSystemTraceId(globals.getSystemTraceId());

auth.setBillPaymentPayee(globals.getBillPaymentPayeeType());

auth.setCredit(globals.getCreditInstument());

// auth.setGift(value);

// auth.setIncrementalAuthorization(value);

auth.setMerchant(globals.getMerchant());

auth.setPaymentType(PaymentType.*fromValue*(globals.getPaymentType()));

**if** (globals.isTaxPresent())

auth.setTax(globals.getTax());

// auth.setTokenRequested(value);

auth.setTransactionAmount(globals.getTransactionAmountType());

auth.setTransactionTimestamp(util.stringToXMLGregorian(globals

.getTransactionTimestamp()));

auth.setTransactionType(TransactionTypeType.*fromValue*(globals

.getTransactionType()));

**return** auth;

}

* 1. Call the client.authorize(AuthorizeRequest) method to send the authorize message.

**public** **void** invokeAuthorize(PaymentPortType client) {

AuthorizeResponse resp = **new** AuthorizeResponse();

// com.sun.xml.internal.ws.transport.http.client.HttpTransportPipe.dump

// = true;

**try** {

resp = client.authorize(auth);

} **catch** (ServerFault e) {

System.***out***.println("Server Fault: " + e.getFaultInfo().getMessage());

} **catch** (RequestValidationFault e) {

System.***out***.println("Validation Fault: " + e.getFaultInfo().getMessage());

}

System.***out***.println(resp.getTransactionStatus());

}