UPS Web Service Sample Code Guide

February 26th, 2018

Important Information

UPS Developer Kit APIs

 Your development of an application using UPS Developer Kit APIs is governed by the UPS Technology Agreement you entered into with UPS. The following are key legal requirements from these agreements for the UPS Developer Kit APIs. For more information on all requirements for the UPS Developer Kit APIs, please refer to the UPS Technology Agreement.

Defined terms used but not defined in this document have the meaning set forth in the UPS Technology Agreement.

Key Legal Requirements for UPS Developer APIs

Permitted Territories

This document can only be used in the countries of the Permitted Territory as defined in the UPS Technology Agreement, as applicable.

Use

The application must not be designed to allow distribution of information received through the UPS Developer Kit APIs to third parties, other than to persons having a bona fide interest in such information (e.g., the shipper, receiver, or the third party payer, or to your service providers authorized by UPS).

Consent to Use of UPS Mark

- All screens or forms generated by your application including information received through the UPS Developer Kit APIs must include (1) the UPS Mark positioned in reasonable proximity to the Information and of an appropriate size to readily identify the source of the Information as UPS and (2) the following language at the bottom of every screen that displays the UPS Mark: "UPS, the UPS brand mark, and the Color Brown are trademarks of United Parcel Service of America, Inc. All Rights Reserved." Except as set forth in the preceding sentence, you have no right to use the UPS Mark without the prior written approval of UPS.

- You shall not use the UPS Mark in association with any third party trademarks in a manner that might suggest co-branding or otherwise create potential confusion as to source or sponsorship of the application, or ownership of the UPS Mark.

- The UPS Mark shall be used only as provided by UPS electronically or in hard copy form. The UPS Mark may not be altered in any manner, including proportions, colors, elements, etc., or animated, morphed or otherwise distorted in perspective or dimensional appearance.

- The UPS Mark may not be combined with any other symbols, including words, logos, icons, graphics, photos, slogans, numbers, or other design elements. A minimum amount of empty space must surround the UPS Mark separating it from any other object, such as type, photography, borders, edges, etc. The required area of empty space around the UPS Mark must be 1/3x, where x equals the height of the UPS Mark.

Copyright and Proprietary Notice

In your application and any POD Letters you prepare, you must include a prominent reproduction of UPS's copyright and proprietary notices in a form and format specified by UPS (See the Copyright section of this document).

Display of Information

The application must not display information concerning any other provider of shipping services or such other shipping services on any page, whether comprising one or more frames, displaying information your application receives from the UPS Developer Kit APIs. Your application must present all data within each field received through the UPS Developer Kit APIs without amendment, deletion, or modification of any type.

Notice

In all communications with UPS concerning this document, please refer to the document date located on the cover.

Copyright

Trademarks

Some of the UPS corporate applications use United States city, state, and postal code information obtained by United Parcel Service of America, Inc. under a non-exclusive license from the United States Postal Service.

Table of Contents

Chapter 1: Introduction

The samples included in the UPS Developer Kits can be used to reference how to create, populate, and invoke various UPS Web Services.

UPS samples are available in Axis2 1.4, JAX-WS 2.1, .NET C#, Perl, and PHP formats.

📄 **IMPORTANT**: To successfully test transactions, the Client will need to submit transactions containing their own:

- Access License
- UserID
- Password
- Shipper Account
- 1Z numbers

Chapter 2: Axis2 Samples

Naming Convention

Sample Java code classes use following naming convention:

Technology Name + Web Service Name + Client

Examples: Axis2ShipClient.java, Axis2VoidClient.java

Axis2	Technology name.
Ship, Void	UPS Web Service name.
Client	'Client' indicates that it is a client program used to invoke the UPS Web Service.

UPS Web Service Axis 2 sample code is provided with build.xml.

Axis directory structure:

- Root directory (name of root directory web service name)
  - src directory (Axis 2- contains java sample client code)
  - build.xml (ant build script to clean, generate stubs, compile and execute sample code)
  - build.properties (is used by build.xml and client class for reading configurable properties)

Ship Web Service JAXB directory structure example:
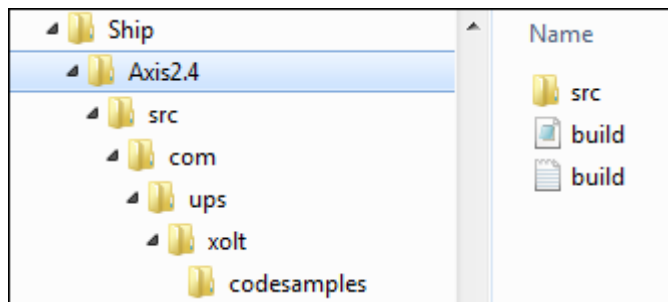
Running the Axis build.xml file using Apache Ant

In order to run **build.xml**, Apache Ant must be installed and configured on the same machine. Apache Ant is a Java based build tool. For additional information, refer to the Apache Ant web site at: http://ant.apache.org/.

The response and response status can be found in the **XOLTResult.xml** file. The **build.properties** file lists the location of **XOLTResult.xml** file.

The default build target runs the complete sample. Optionally, you may select a specific target in the build.xml to run.

Axis.home should have all required axis2 jars.

build.properties

Ship example:

```
url=https://wwwcie.ups.com/webservices/Ship or https://onlinetools.ups.com/webservices/Ship
accesskey=Your CIE Accesskey for wwwcie, or production AccessKey for onlinetools
username=Your User Name
password=Your Password
axis.home=Your Axis24 home dir
wsdl.home=Your WSDL dir
wsdl=Ship.wsdl
clientname=com.ups.xolt.codesamples.Axis2ShipClient
build=build
src=src
build.classes=build/classes
deletefolder=com/ups/www
out_file_location =XOLTResult.xml
tool_or_webservice_name = Ship Web Service
```

XOLTResult.xml file response

```
<ExecutionAt>Wed Dec 09 15:38:45 IST 2015</ExecutionAt>
<TooloOrWebServiceName>Ship Web Service</TooloOrWebServiceName>
    <ResponseStatus>
        <Code>1</Code>
        <Description>Success</Description>
    </ResponseStatus>
```

Running Axis examples using Eclipse

This section explains how to use the Java samples using Eclipse IDE with Apache Axis 2-1.4.

The Eclipse IDE for Java EE Developers can be downloaded at: http://www.eclipse.org/downloads/

Configuring Eclipse

1. Download the Binary Distribution of Axis2 from: http://axis.apache.org/axis2/java/core/download.cgi

2. Place the WSDL and all associated schema files in a folder.

3. Create a new system environment variable named **AXIS2_HOME**, pointing to the directory where the Axis2 distribution was unzipped.

4. Create a new Java environment variable named **JAVA_HOME**, pointing to to the jdk installation directory (if not already set).

5. Run the wsdl2java tool (located in the "bin" folder) from the command prompt.

    <PATH-OF-AXIS2-BIN-FOLDER>\wsdl2java -g -ssi -uri <PATHOF-WSDL-FOLDER>\[wsdl name].wsdl

    Example : C:\axis2-1.6.2\bin\wsdl2java -g -ssi -uri C:\RateSampleCode\RateWS.wsdl

The above command will generate the stub java files (in a java package structure) in the same folder where the command is executed.

Using the stub and client in a new java project

1. Open Eclipse and create a new project.

    o   File > New > Other > Java Project

2. Copy the client sample code into the project.

3. Copy the generated stub code into the project.

4. Add the Axis libraries to Eclipse build path.

    o   Right-click project > properties > java build path > go to "libraries" tab > add external jars > browse and add all axis2* and axiom* jars from the Axis2 distribution's lib folder.

Chapter 3: JAX WS 2.1 Samples

Naming Convention

Sample Java code classes use following naming convention:

Technology Name + Web Service Name + Client

Examples: JaxwsShipClient.java, JaxwsVoidClient.java

Jaxws	Technology name.
Ship, Void	UPS Web Service name.
Client	'Client' indicates that it is a client program used to invoke the UPS Web Service.

UPS Web Services JAX WS 2.1 samples are provided with build.xml. Samples have following directory structure:

- Root directory - name of root directory web service name
  - src directory - JAX WS 2.1 contains java sample client code
  - build.xml - ant build script to clean, generate stubs, compile and execute sample code
  - build.properties - used by build.xml and client class for reading configurable properties)

Additional binding.xml is needed for ship, freight ship and void web services.

Ship Web Service JAXWS2.1 directory structure example:
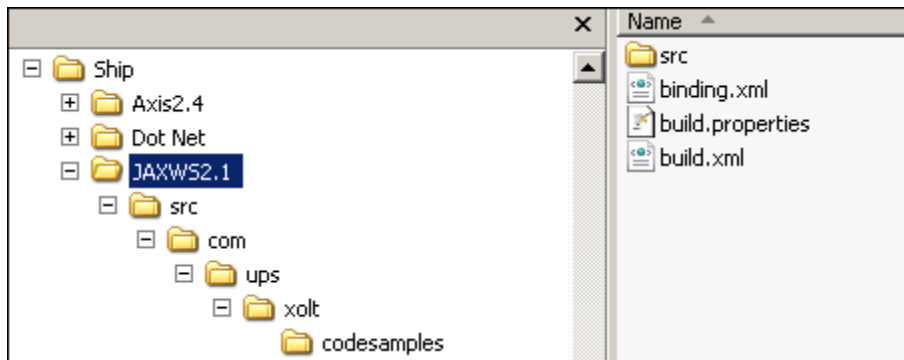
Running the JAX build.xml file using Apache Ant

In order to run **build.xml**, Apache Ant must be installed and configured on the same machine. Apache Ant is a Java based build tool. For additional information, refer to the Apache Ant web site at: http://ant.apache.org/.

The response and response status can be found in **XOLTResult.xml** file. The **build.properties** file lists the location of **XOLTResult.xml** file.

The default build target runs the complete sample. Optionally, you may select a specific target in the build.xml to run.

📄 **NOTE**: The Lib folder in build.properties should contain all of the JAXWS 2.1 related jars.

build.properties

Ship example:

```
url= https://wwwcie.ups.com/webservices/Ship or https://onlinetools.ups.com/webservices/Ship
accesskey=Your Access Key
username=Your user name
password=Your password
client.class=com.ups.xolt.codesamples.JaxwsShipClient
build.classes=build/classes
src=src
lib=JAXWS2.1 root
debug=true
keep=true
verbose=true
wsdl.home=WSDL Root
jaxws21=JAXWS2.1
wsdlfilename=Ship.wsdl
gensrc1=com/ups/wsdl
gensrc2=com/ups/xmlschema
out_file_location =XOLTResult.xml
tool_or_webservice_name =Ship Web Service
```

XOLTResult.xml file response:

```
<ExecutionAt>Wed Dec 09 15:38:45 IST 2015</ExecutionAt>
<TooloOrWebServiceName>Ship Web Service</TooloOrWebServiceName>
    <ResponseStatus>
        <Code>1</Code>
        <Description>Success</Description>
    </ResponseStatus>
```

Ship, Void, and Freight Ship Binding

When generating Ship, Void, and Freight Ship stub files you must use binding.xml to avoid errors (see below).

```
[wsimport] Consider using <depends>/<produces> so that WSimport won't do unnecessary compilation

[wsimport] parsing WSDL...

[wsimport] [ERROR] XPath evaluation of "wsdl:definitions/wsdl:types/xsd:schema[@targetNamespace='http://.....

[wsimport] line 10 of file:......

[wsimport] [ERROR] The package name 'com.ups.xmlschema.xoltwe.if.v1' used for this schema is not a valid..

[wsimport] line 3 of file:....
```

External binding files are semantically equivalent to embedded binding declarations. UPS provides the binding.xml to map a name space to an explicit package. This ensures that auto generated package names do not conflict with Java key words, such as "if" and "void".

Using binding.xml in build.xml

```
<target name="genclients" depends="clean">
    <wsimport fork="true"
              sourcedestdir="${basedir}/src"
              wsdl="${wsdl.home}/${wsdl}"
              binding="${basedir}/binding.xml"/>
</target>
```

Ship Web Service binding.xml example

In this example the child **jaxws:bindings** applies package customization. An XPath expression in the node attribute refers to the root node of the WSDL document, which is **wsdl:definitions** and declares the package **com.ups.IFS** for all the generated Java classes mapped from the WSDL file.

```
<jxb:bindings version="1.0" xmlns:jxb="http://java.sun.com/xml/ns/jaxb">
    <jxb:bindings xmlns:xs="http://www.w3.org/2001/XMLSchema" schemaLocation="wsdl/IFWS.xsd"
    node="/xs:schema">
        <jxb:schemaBindings>
            <jxb:package name="com.ups.IFS"/>
        </jxb:schemaBindings>
    </jxb:bindings>
</jxb:bindings>
```

Void

Build.xml in the Void Web Service JAX-WS sample makes use of the "package" attribute in the wsimport element for auto code generation. This ensures that auto generated package names do not conflict with Java key words, such as "if" and "void".

Thefollowing is an example of the genclient target using the "package" attribute in wsimport:

```
<target name="genclient" depends="init">
    <wsimport fork="true" debug="${debug}" verbose="${verbose}" keep="${keep}" destdir="${build.classes}"
              sourcedestdir="${src}"
              wsdl="${wsdl.home}/${wsdlfilename}"
              package="com.ups.wsdl.xoltws.voidws.v1"/>
</target>
```

Chapter 4: .Net Samples

Naming Convention

Sample Java code classes use following naming convention:

Web Service Name + Client

Examples: ShipWSClient.cs, VoidWSClient.cs

ShipWS, VoidWS	UPS Web Service name.
Client	'Client' indicates that it is a client program used to invoke the UPS Web Service.

Running .NET samples using MS Visual Studio

UPS Web Services .NET samples are provided in Microsoft Visual Studio Solution (.sln) format.

All of the samples have a directory structure similar to the Shipping Package Web Service example below:
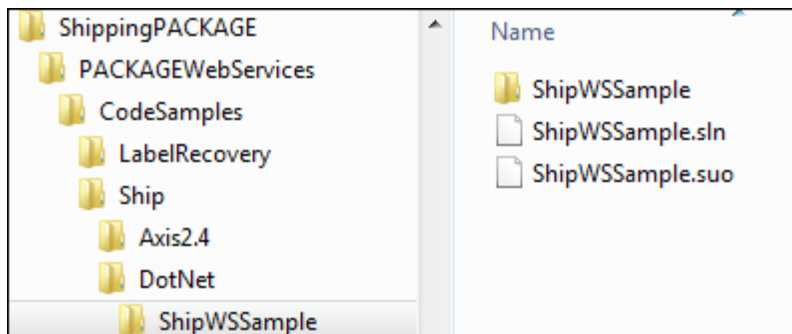
1. Open Microsoft Visual Studio.
2. Add your User Name, Access Key, and Password in the [apiname].cs file.
3. Execute the sample code.

Results display in the console. You can modify the app.config to point to the URLs to be tested.

Chapter 5: Perl Samples

Requirements

1. Perl 5.8 or above
2. Perl Package Manager
3. XML::Compile::WSDL11 version 2.24 or above
4. XML::LibXML::Simple version 0.91 or above
5. HTTP::Request version 6.02 or above
6. HTTP::Response version 6.02 or above
7. Data::Dumper version 2.131 or above

Naming Convention

Sample Perl code use following naming convention:

Technology Name +Web Service Name + Client

Examples: XMLCompileWsdl11ShipWSClient.pl, XMLCompileWsdl11VoidWSClient.pl

XMLCompileWsdl11	Technology name.
ShipWS, VoidWS	UPS Web Service name.
Client	'Client' indicates that a client program is used to invoke the UPS Web Service.

Installing Perl

ActivePerl distribution should be used for Perl development. To download the msi executable, refer to the Perl Reference topic.

📄 **NOTE**: We strongly recommend consulting with your IT administrator to have this configured.

📄 **NOTE**: You must be logged into the system at root level to install Perl.

Installing Perl Package Manager (PPM)

We recommend using Perl Package Manager (PPM) that comes included with ActivePerl distribution. The Perl Package Manager (PPM) can be used to download and install Perl modules automatically.

Firewall/Proxy Configuration

📄 **NOTE**: Please consult with your IT administrator to determine if your system is behind a firewall that will prevent downloading Perl modules.

Using PPM requires Internet access to download Perl packages from Perl repository sites. When PPM cannot connect to Internet, the following message displays:

Downloading ActiveState Package Repository packlist ... failed 407 Proxy Authentication Required

To prevent this error message you must define the http_proxy SYSTEM variable.

1. Display the System Properties dialog.

2. Click **Environment Variables**.
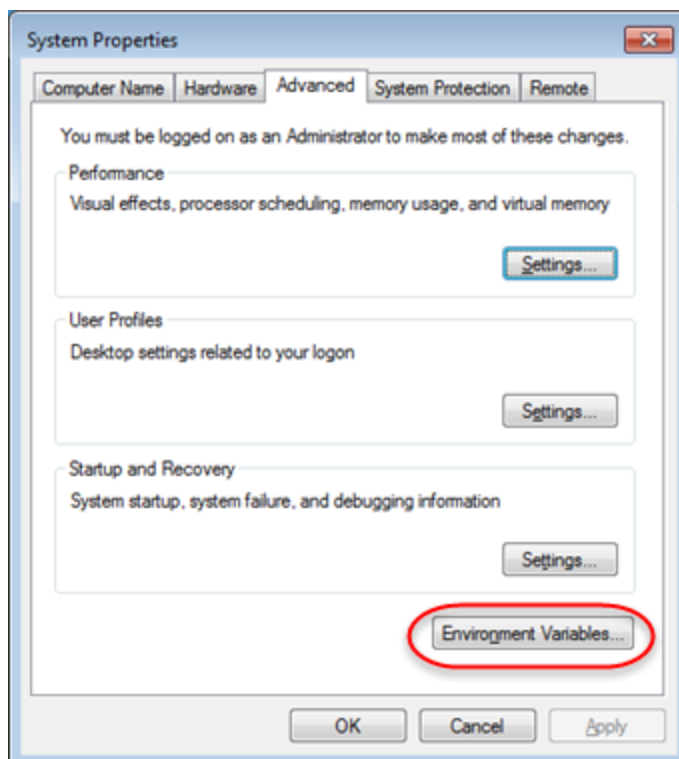
3. Click **New**.
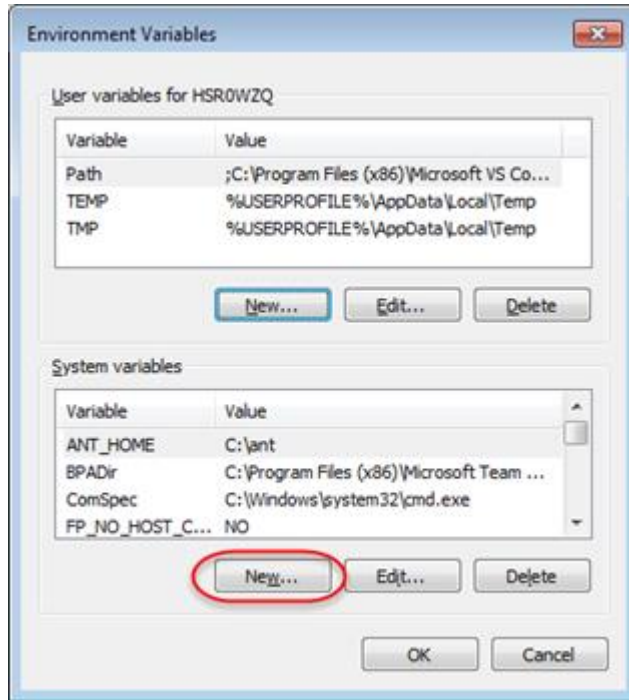
4. Enter the Variable name and value.

- Variable name: http_proxy

- Variable value:

  o Replace the **Userid** and **Password**, inthe example below, with the network id and password used to log into your network or computer.

  o **ProxyURL** is the hostname used to connect to Internet. Please consult your IT administrator to get proxy details.
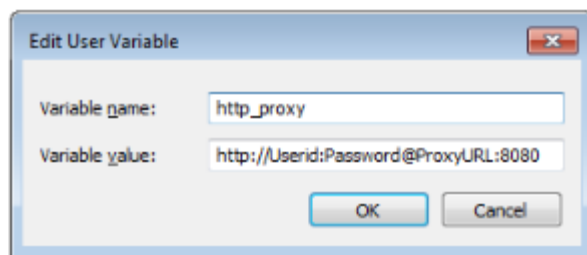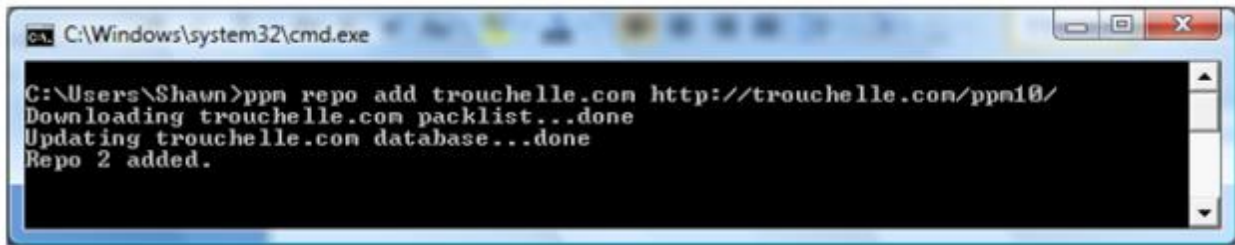

5. Restart your computer.

Adding a repository

To add a repository, open a command prompt and type the following command:

ppm repo add trouchelle.com http://trouchelle.com/ppm10/

Installing a Perl module

To install a Perl module, open a command prompt and execute the commands (blue text) in chronological order.
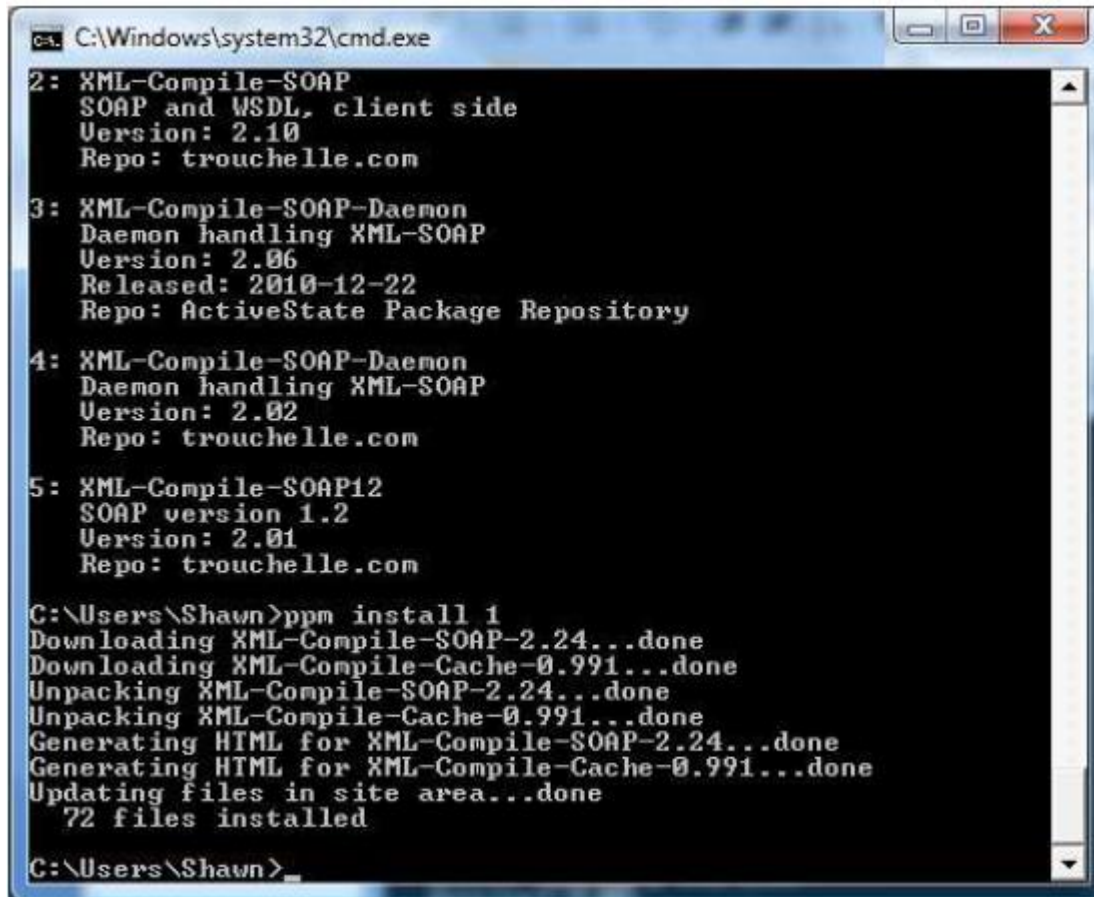
1. Enter the Search command: ppm search XML::Compile::Soap

Module search for XMLCompileSoap result example:

2. Enter the Install command: ppm install 1

XMLCompileSOAP using ActiveState Package Repository site example:

```
C:\Windows\system32\cmd.exe

2: XML-Compile-SOAP
   SOAP and WSDL, client side
   Version: 2.10
   Repo: trouchelle.com

3: XML-Compile-SOAP-Daemon
   Daemon handling XML-SOAP
   Version: 2.06
   Released: 2010-12-22
   Repo: ActiveState Package Repository

4: XML-Compile-SOAP-Daemon
   Daemon handling XML-SOAP
   Version: 2.02
   Repo: trouchelle.com

5: XML-Compile-SOAP12
   SOAP version 1.2
   Version: 2.01
   Repo: trouchelle.com

C:\Users\Shawn>ppm install 1
Downloading XML-Compile-SOAP-2.24...done
Downloading XML-Compile-Cache-0.991...done
Unpacking XML-Compile-SOAP-2.24...done
Unpacking XML-Compile-Cache-0.991...done
Generating HTML for XML-Compile-SOAP-2.24...done
Generating HTML for XML-Compile-Cache-0.991...done
Updating files in site area...done
   72 files installed

C:\Users\Shawn>_
```
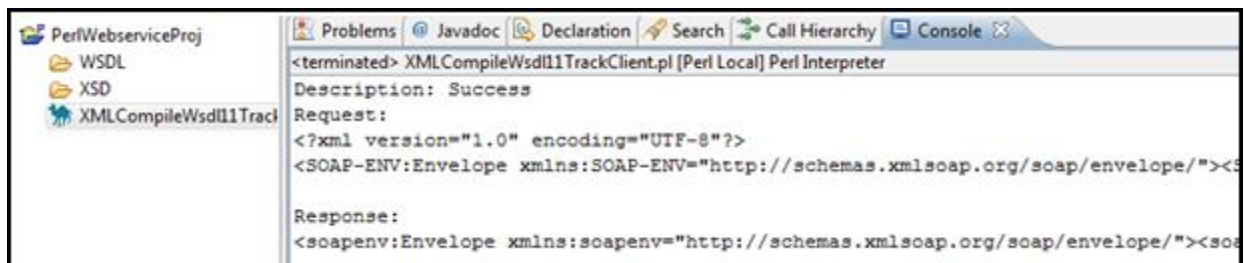
Running Perl samples using Unix Visual Editor (VI)

1. Open the sample Perl sample in VI.

   o Web Service file name example: XMLCompileWsdl11ShipClient.pl

2. Define the values for the following variables:

   o access license, userid and password

   o endpoint url

   o wsdl file

   o operation name

   o location of the schema directory

   o outputFileName(response file name)

3. Run the Perl file using the following command: perl *PerlFileName*.pl

   o Replace *PerlFileName.pl* withh file name used in Step 1.

Running Perl samples using Eclipse

To run a Perl code sample using Eclipse you must install the EPIC plugin. For plugin download and documentation, refer to the Perl Reference section. On the download page, choose the latest EPIC plugin.

1. Create a new Perl project in Eclipse.

2. Import the Perl sampleinto the project.

   o Example: XMLCompileWsdl11ShipWSClient.pl

3. Define the values for the following variables:

   o access license, userid and password

   o endpoint url

   o wsdl file

   o oprationname

   o schema file location

   o outputFileName (Response file name)

4. Run the Perl file imported in step 2.

Successful run:

Perl References

Description	Link
Download site for ActivePerl distribution	http://www.activestate.com/activeperl/downloads
Tools for installing Perl modules	http://www.cpan.org/modules/INSTALL.html
XML::Compile::WSDL11 module reference site	http://search.cpan.org/~markov/XMLCompile- SOAP-2.24/lib/XML/Compile/WSDL11.pod
XML::Compile::Schema module reference site	http://search.cpan.org/~markov/XMLCompile-1.22/lib/XML/Compile/Schema.pod
EPIC plugin tutorial site	http://www.epic-ide.org/
EPIC plugin download site	http://sourceforge.net/projects/e-p-i-c/files/e-p-i-c/
Perl repository site to download and install XML::Pastor:	http://trouchelle.com/perl/ppmrepview.pl
Installing Perl modules (Manual Process)	http://www.thegeekstuff.com/2008/09/howto- install-perl-modules-manually-and-using-cpan-command/

Chapter 6: PHP Samples

Requirements

- PHP 5.3 or above
- Soap extension

📄 **NOTE**: PHP highly recommends building PHP extensions in a UNIX/Linux environment.

Naming Convention

Sample PHP code use following naming convention:

Soap + Web Service Name + Client

Examples:SoapShipWSClient.php, SoapVoidWSClient.php

Soap	Technology name.
Name	UPS Web Service name.
Client	'Client' indicates that a client program is used to invoke the UPS Web Service.

Installing PHP in a UNIX/Linux environment

Please consult with your IT administrator regarding the UNIX/Linux binaries needed for PHP installation.

📄 **NOTE**: To install PHP you must log into the system at the root level.

Installing the SOAP Extension

Please consult with your IT administrator about installing PHP extensions on your UNIX/Linux environment.

Before installing the SOAP extension please check the "**ext**" directory under PHP folder to verify if the SOAP directory is present. Example: "/php5-3-6/ext"

Inside this directory is a hidden folder called ".**libs**" and it will contain a **soap.so** file. If this file in this directory exists then SOAP extension has been built and may need to be enabled. If this file does not exist or the directory itself is not present then the SOAP extension may need to be built.

Build SOAP

📄 **NOTE**: To build the SDO extension you must log into the system at root level.

A link to php.net site is provided in the PHP References section that can help with building and installing PHP extensions on UNIX/Linux system. We recommend using the phpize command for simplicity.

After build, the extension will be installed to a specific directory. This directory is mention after the build finishes. This extension contains the SoapClient program and has to be enabled for PHP to use.

Configure PHP.ini

1. Once the SOAP extension has been built successfully, locate and open **php.ini** in Unix Visual Editor (VI).

2. Enable the SOAP extension by adding the file location where the **soap.so** extension is installed.

3. To verify if the SOAP extension was enabled successfully run the following command: php –m

📄 **NOTE**: To determine what dependencies are needed for UNIX/Linux environment, we strongly recommend consulting with your IT administrator to gather this information before installing PHP and the SOAP extension.

SOAP extension enabled:

Running PHP samples using Unix Visual Editor

1. Open the PHP file in Unix Visual Editor (VI).

   o File name example: SoapShipWSClient.php

2. Define the values for the following variables:

   o access license, userid and password

   o endpoint url

   o wsdl

   o operation name

   o outputFileName (Response file name)

3. Save the file.

4. Run the PHP file using the following command: php *FileName*.php

   o Example: php SoapShipWSClient.php

PHP References

Description	Link
Build and compile PHP extensions	http://www.php.net/manual/en/install.pecl.phpize.php
SoapClient Guide	http://us.php.net/manual/en/class.soapclient.php