



UPS XML Sample Code Guide

March 18, 2020



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 $\frac{1}{3}x$, 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

© 2020 United Parcel Service of America, Inc. All Rights Reserved. Confidential and Proprietary

The use, disclosure, reproduction, modification, transfer, or transmittal of this work for any purpose in any form or by any means without the written permission of United Parcel Service is strictly prohibited.

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	5
Chapter 2: JAXB Samples	6
Naming Convention.....	6
Running the build.xml file using Apache Ant.....	7
Chapter 3: Perl Samples.....	8
Requirements.....	8
Naming Convention.....	8
Installing Perl	9
Installing Perl Package Manager (PPM)	9
Running Perl samples using Unix Visual Editor (VI)	11
Running Perl samples using Eclipse	11
Perl References	12
Chapter 4: PHP Samples.....	13
Requirements.....	13
Naming Convention.....	13
Installing PHP in a UNIX/Linux environment	14
Installing SDO Extension	14
Build SDO	14
Configure PHP.ini.....	14
Running a PHP sample in Unix VI Editor	15
PHP References.....	15

Chapter 1: Introduction

The samples included in the UPS Developer Kits can be used to reference how to create, populate, and invoke various UPS XML/API requests.

UPS samples are available in JAXB 2.1, 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: JAXB Samples

Naming Convention

Sample Java code classes use following naming convention:

Technology Name + UPSToolName +Client

Examples: JaxbShipConfirmClient.java, JaxbVoidClient.java

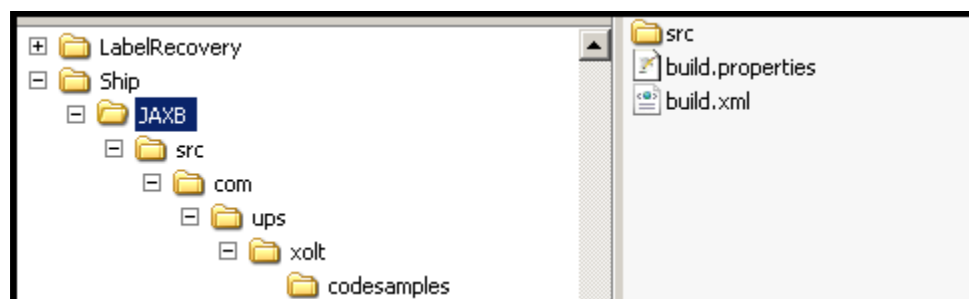
Jaxb	Technology name
Ship	UPS XML Name
Client	'Client' indicates that it is a client program used to invoke the UPS XML/API.

UPS XML JAXB sample code is provided with build.xml.

Sample code directory structure:

- Root directory - name of root directory web service name
 - src directory - contains JAXB sample client java 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

Ship XML JAXB directory structure example:



Running the build.xml file using Apache Ant

In order to run the **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.

Jaxb.home contains all of the required jaxb jars files.

build.properties

The build.properties file contains the following information:

```
url=Ship XML URL
accesskey=your access key
username=your user name
password=your password
build=build
src=src
build.classes=build/classes
jaxb.home=your jaxb library home
requestxsd=Tool Request schema
responsexsd=Tool Response schema
accessrequestxsd=yTool Access Request schema
requestpackage=com.ups.xolt.codesamples.request.jaxb
responsepackage=com.ups.xolt.codesamples.response.jaxb
accessrequestpackage=com.ups.xolt.codesamples.accessrequest.jaxb
out_file_location=XOLTResult.xml
deletedir1=src/com/ups/xolt/codesamples/accessrequest
deletedir2=src/com/ups/xolt/codesamples/request
deletedir3=src/com/ups/xolt/codesamples/response
```

XOLTResult.xml file response example

```
<ShipmentConfirmResponse>
  <Response>
    <TransactionReference>
      <CustomerContext>Your Customer Context</CustomerContext>
    </TransactionReference>
    <ResponseStatusCode>1</ResponseStatusCode>
    <ResponseStatusDescription>Success</ResponseStatusDescription>
  </Response>
</ShipmentConfirmResponse>
```

Chapter 3: Perl Samples

Requirements

1. Perl 5.8 or above
2. Perl Package Manager
3. XML::Compile version 1.22 or above
4. XML::LibXML::Simple version 0.91 or above
5. LWP::UserAgent version 6.03 or above
6. HTTP::Request version 6.02 or above
7. Data::Dumper version 2.131 or above

Naming Convention

Sample Perl code use following naming convention:

Technology Name + UPSToolName +Client

Examples: XMLCompileShipConfirmClient.pl, XMLCompileVoidClient.pl

XMLCompile	Technology name
Ship	UPS Tool Name
Client	'Client' indicates that a client program is used to invoke the UPS XML/API.

Perl sample code directory structure:

- Root directory (name of root directory)
 - Perl directory (contains Perl sample client code)

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



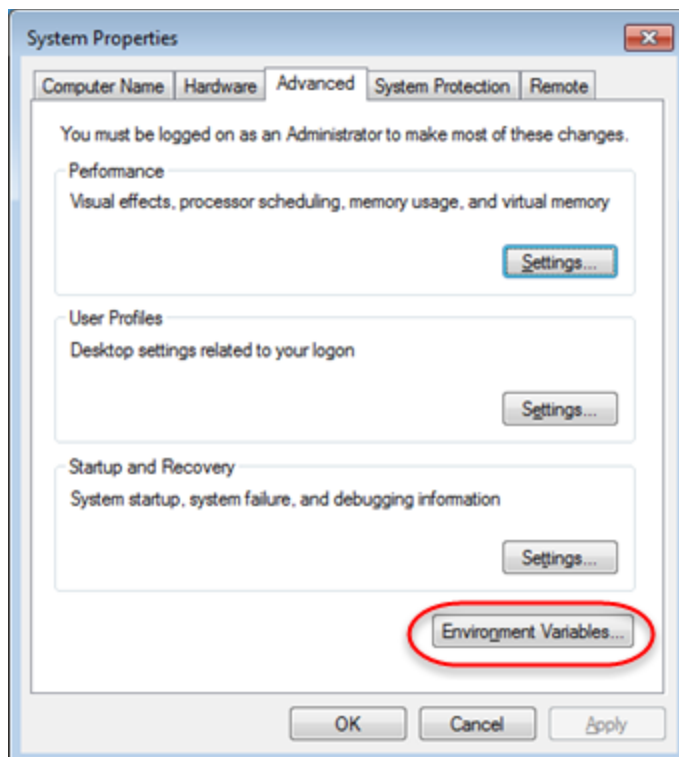
NOTE: Please consult with your IT administrator to determine if your system is behind a firewall that will prevent downloading Perl modules.

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

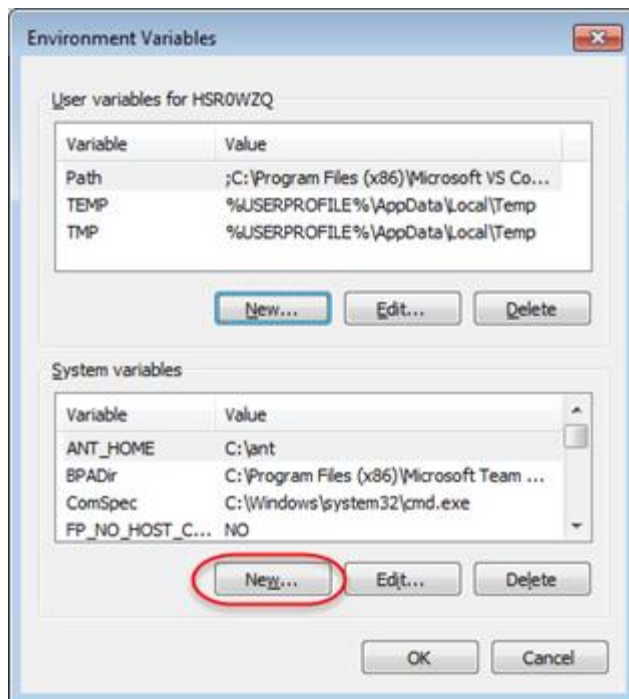
Downloading ActiveState Package Repository packlist ... failed 407 Proxy Authentication Required

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

1. Display the System Properties dialog.
2. Click **Environment Variables**.

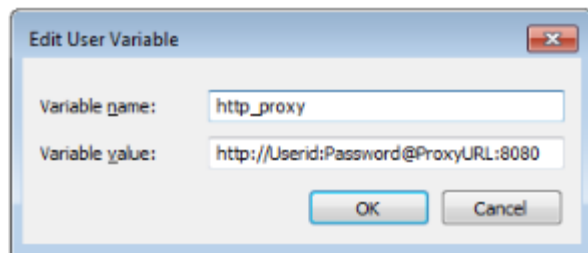


3. Click **New**.



4. Enter the Variable name and value.

- Variable name: http_proxy
- Variable value:
 - Replace the **Userid** and **Password**, in the example below, with the network id and password used to log into your network or computer.
 - **ProxyURL** is the hostname used to connect to Internet. Please consult your IT administrator to get proxy details.



5. Restart your computer.

Running Perl samples using Unix Visual Editor (VI)

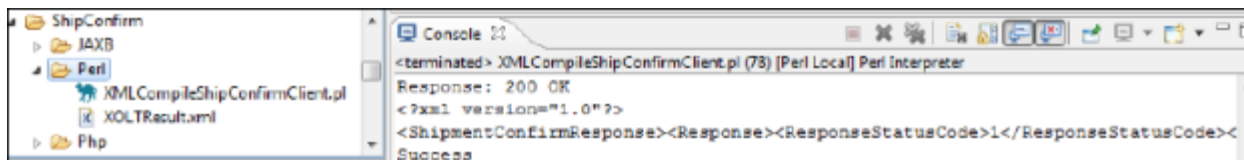
1. Open the sample Perl sample in VI.
 - o XML file name example: [XMLCompileShipConfirmClient.pl](#)
2. Define the values for the following variables:
 - o access license, userid and password
 - o endpoint url
 - o schema file location
 - ▲ accessSchemaFile
 - ▲ requestSchemaFile
 - ▲ responseSchemaFile
 - ▲ importedSchemaFile
 - o outputFileName (response file name)
3. Run the Perl file using the following command: [perl PerlFileName.pl](#)
 - o Replace [PerlFileName.pl](#) with 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 file into the project.
 - o Example: [XMLCompileShipConfirmClient.pl](#)
3. Define the values for the following variables:
 - o access license, userid, and password
 - o endpoint url
 - o schema file location
 - ▲ accessSchemaFile
 - ▲ requestSchemaFile
 - ▲ responseSchemaFile
 - ▲ importedSchemaFile
 - o outputFileName (Response file name)
4. Run the Perl file imported in step 2.
 - o Example: [perl XMLCompileShipConfirmClient.pl](#)

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 module reference site	http://search.cpan.org/~markov/XML-Compile-1.22/lib/XML/Compile.pod
XML::Compile::Schema module reference site	http://search.cpan.org/~markov/XML-Compile-1.22/lib/XML/Compile/Schema.pod
XML::LibXML::Simple module reference site	http://search.cpan.org/~markov/XML-LibXML-Simple-0.91/lib/XML/LibXML/Simple.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/ppmreview.pl
Installing Perl modules (Manual Process)	http://www.thegeekstuff.com/2008/09/how-to-install-perl-modules-manually-and-using-cpan-command/

Chapter 4: PHP Samples

Requirements

- PHP 5.3 or above
- SCA_SDO version 1.2.4 or above

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

Naming Convention

Sample PHP code use following naming convention:

Technology Name + UPSToolName +Client

Examples: SDOShipConfirmClient.php, SDOVoidClient.php

SDO	Technology name
Ship	UPS Tool Name
Client	'Client' indicates that a client program is used to invoke the UPS XML/API.

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.

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

Installing SDO Extension

The SCA_SDO download link can be found in the [PHP Reference](#) section.

Build SDO

A link to the php.net site is provided in the [PHP Reference](#) section that can help with building and installing PHP extensions on UNIX/Linux systems. We recommend using the [phpize](#) command for simplicity.

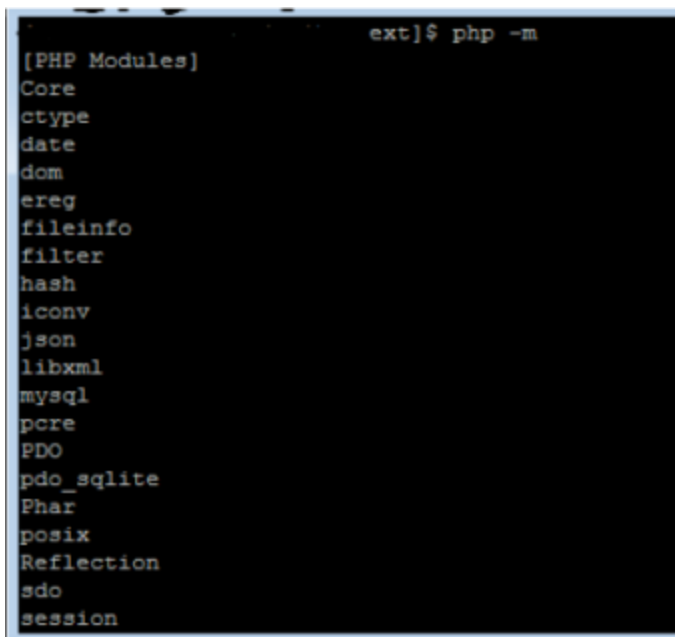
Please consult with your IT administrator for this process.

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

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

Configure PHP.ini

1. Once the SDO extension has been built successfully, locate and open **php.ini** in Unix Visual Editor (VI).
2. Enable the SDO extension by adding the file location where the sdo.extension is installed.
3. To verify if the SDO extension was enabled successfully run the following command: [php -m](#)



```
ext]$ php -m
[PHP Modules]
Core
ctype
date
dom
ereg
fileinfo
filter
hash
iconv
json
libxml
mysql
pcre
PDO
pdo_sqlite
Phar
posix
Reflection
sdo
session
```

Running a PHP sample in Unix VI Editor

1. Open the PHP file in VI editor.
 - File name example: [SDOShipConfirmClient.php](#)
2. Define the values for the following variables:
 - access license, userid and password
 - endpoint url
 - schema file location
 - ▲ `accessSchemaFile`
 - ▲ `requestSchemaFile`
 - ▲ `responseSchemaFile`
 - `outputFileName` (Response file name)
3. Save the file.
4. Run PHP file using the following command: `php FileName.php`
 - Example: `php SDOShipConfirmClient.php`

PHP References

Description	Link
Build and compile PHP extensions	http://www.php.net/manual/en/install.pecl.phpize.php
SDO package download site	http://pecl.php.net/get/SCA_SDO
SDO package reference	http://www.php.net/manual/en/sdo-das-xml.examples.php