United States Postal Service®
Web Tool Kit User's Guide

A Technical Guide to

Domestic Rates Calculator

Application Programming Interface

Before implementing this API, the *Administrative Guide for Application Programming Interfaces* must be read.

Version 3.2 (6/30/02)

To Our Customers

In the e-mail that accompanied this guide you received a password and user ID that will allow you to begin sending calls to the "test server" when you are ready. Any additional documentation or contact with you will be made through the contact person indicated on the registration form.

If you require technical support, contact the USPS Internet Customer Care Center (ICCC). This office is manned from 7:00AM to 11:00PM EST.

E-mail: icustomercare@usps.com

Telephone: 1-800-344-7779 (7:00AM to 11:00PM EST)

USPS Customer Commitment

The United States Postal Service fully understands the importance of providing information and service anytime day or night to your Internet and e-commerce customers. For that reason, the USPS is committed to providing 7 x 24 service from our API servers, 365 days a year.

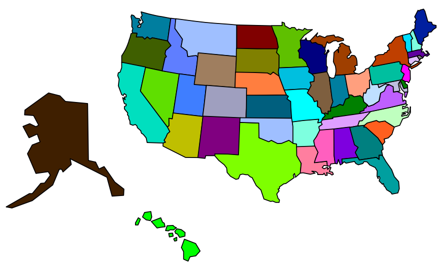
Thank you for helping the U.S. Postal Service provide new Internet services to our shipping customers.

Internet Shipping Solutions Team
U.S. Postal Service
475 L'Enfant Plaza, SW
Washington, DC 20260-2464

Trademarks

Table of Contents

Introduction to the Domestic Rates Calculator API

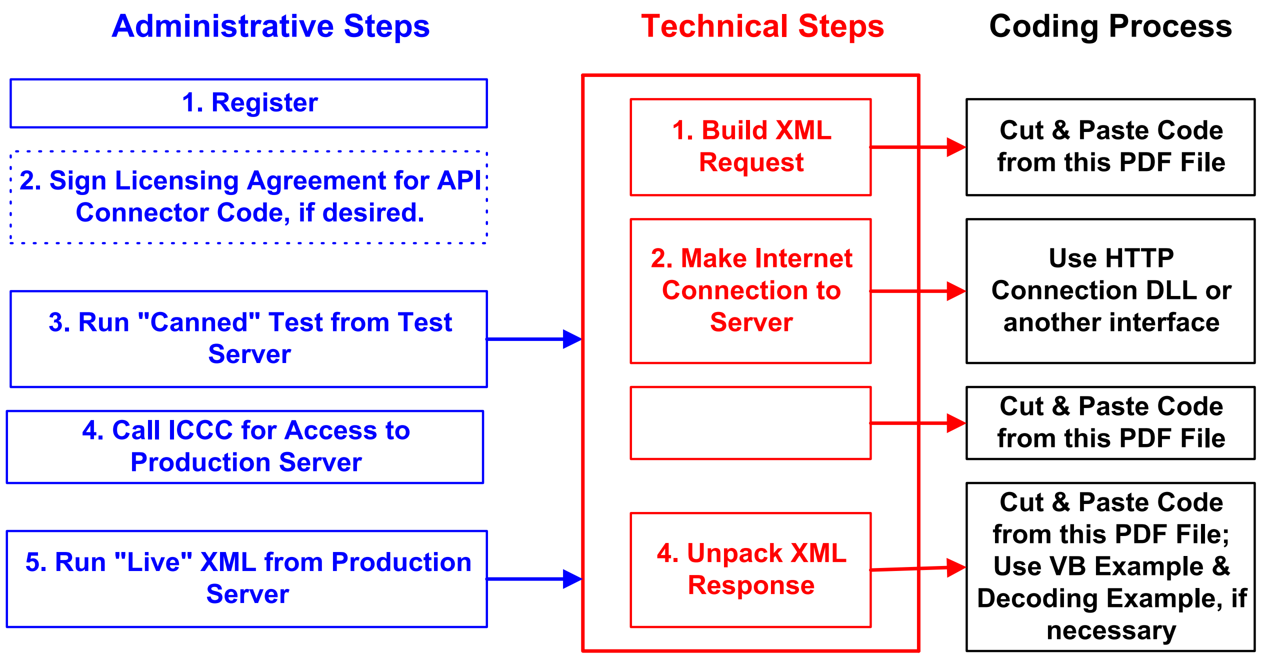


The Domestic Rates API provides automated online access to domestic rates for Express Mail, First-Class mail and Priority Mail, as well as single-piece rates for all four Package Services: Parcel Post, Bound Printed Matter, Library Mail, and Media Mail. APO/FPO addresses are identified and mailing restrictions, when applicable, are also provided. Requests for shipping rates will be processed for up to 25 packages per request.

Note to Developers: USPS shipping rates cannot be misrepresented on web sites as "Handling Charges." USPS rates can, however, be included in total "Shipping and Handling Charges." The intent is to not mislead consumers by disguising handling charges as USPS shipping charges.

As shown below, implementing USPS Shipping APIs requires a series of *Administrative Steps*. The *Administrative Guide for APIs*, also available at www.uspswebtools.com, provides necessary information and procedures prior to installation. The illustration also shows the *Technical Steps* required to run XML transactions for the Domestic Rates Calculator API to either the test server or the production server, as well as the *Coding Process* to be followed for each *Technical Step*. This document provides step-by-step instructions for both the Technical Steps and Coding Process illustrated below. As each step is presented throughout this guide, appropriate portions of the illustration below will be repeated as a reference point in the implementation process.

Implementing these APIs requires experienced programmers who are familiar with Internet and web site development tools and techniques.


Administrative Steps
Technical Steps
Coding Process
1. Register
2. Sign Licensing Agreement for API Connector Code, if desired.
3. Run "Canned" Test from Test Server
4. Call ICCC for Access to Production Server
5. Run "Live" XML from Production Server
1. Build XML Request
2. Make Internet Connection to Server
4. Unpack XML Response
Cut & Paste Code from this PDF File
Use HTTP Connection DLL or another interface
Cut & Paste Code from this PDF File
Cut & Paste Code from this PDF File; Use VB Example & Decoding Example, if necessary

Before implementing this API, the *Administrative Guide for Application Programming Interfaces* must be read.

It is also recommended that you review specific sections of the Domestic Mail Manual (DMM) for mailing standards, including:

- Eligibility (what items can be mailed at Bound Printed Matter, Media Mail, and Library Mail rates)
- Characteristics (physical size and weight limits for packages)
- Rate Markings (marking on package indication the class of mail)
- Rates (rate charts that will be calculated by the APIs)

The DMM is available on the U.S. Postal Service Postal Explorer Home Page at http://pe.usps.gov and is updated monthly. The following sections of the DMM contain specific information that will assist you in understanding the eligibility, characteristics, markings, and rates for the mail classes listed below, and may be useful to review before implementing any APIs. The USPS also has a series of Quick Service Guides that summarize the information in the DMM. The Quick Service Guides are available on the Postal Explorer Home Page and are linked to the DMM for easy reference. All information is available in both PDF and HTML.

References below are to Domestic Mail Manual sections:

- *Priority Mail*
  (E120) Rate marking on packages, Flat Rate envelope, and balloon rate information.
  (R100.8) Priority Mail rate chart.
  Quick Service Guide 120 -- Priority Mail
- *Parcel Post*
  (C700.2) Nonmachinable surcharge criteria.
  (E711.2) Parcel Post packages less than 15 pounds and 84 to 108 inches (balloon rates), and Oversized rates.
  (R700.1) Parcel Post rate charts
  Quick Service Guide 710 -- Parcel Post
- *Bound Printed Matter*
  (E712) Content eligibility standards and 15 pound maximum weight limit.
  (M712) Required rate marking "Bound Printed Matter."
  (R711.2) Bound Printed Matter rate charts
  Quick Service Guide 720 -- Bound Printed Matter
- *Media Mail*
  (E713) Content eligibility standards.
  (M730) Required rate marking "Media Mail"
  (R711.3) Media Mail rate chart.
  Quick Service Guide 730 Media Mail
- *Library Mail*
  (E714) Qualifying organizations and items that mail at Library Mail rates.
  (M740) Required rate marking "Library Mail"

(R711.4) Library Mail rate chart.
Quick Service Guide 740 -- Library Mail

User ID and Password Restrictions

The user ID and password that you have received is for you or your company to use in accordance with the Terms and Conditions of Use to which you agreed during the registration process. *This user ID and password is not to be shared with others outside your organization, nor is it to be packaged, distributed, or sold to any other person or entity.* Please refer to the Terms and Conditions of Use Agreement for additional restrictions on the use of your user ID and password, as well as this document and the APIs contained herein.

Warning: If the U.S. Postal Service discovers use of the same user ID and password from more than one web site, all users will be subject to immediate loss of access to the USPS server and termination of the licenses granted under the Terms and Conditions of Use.

The documentation and sample code contained in the *Web Tool Kit User Guide* series may be reused and/or distributed to your customers or affiliates to generate awareness, encourage web tool use, or provide ease-of-use. However, it is your responsibility to ensure that your customers do not use your password and user ID. Direct them to www.uspswebtools.com so that they can register, agree to the Terms and Conditions of Use agreement, and receive their own unique password and user ID.

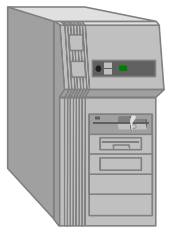
Note to Software Distributors: The User ID and password restrictions discussed above are intended for e-tailers that use the USPS Web Tools exclusively within their own web sites. If you plan to distribute software with the USPS Web Tools embedded, you must refer to the "Software Developers' Terms and Conditions of Use" available at www.uspswebtools.com.

For more information regarding the USPS Web Tool Kit password and user ID policy, or for questions regarding the distribution of documentation, send e-mail to icustomercare@usps.com.

Installation

The illustration below shows the transactional flow of information to and from the USPS Domestic Rates Calculator API server.

Domestic Rates Calculator API Server



INPUTS

(via XML Request from Customer to USPS)

Origination & Destination ZIP Codes
Service Type
Package Size & Weight
Container Type

SERVER TASKS

Ensures Valid Package
Looks Up Rate
Builds XML Response

OUTPUTS

(via XML Response from USPS to Customer)

Origination & Destination ZIP Codes
Service Type
Package Size & Weight
Container Type
Postage
Zone

Technical Steps

Step 1: Build the XML Request

$$\begin{tikzcd}\boxed{\text{1. Build XML Request}} \arrow[r] & \boxed{\text{Cut \& Paste Code from this PDF File}}\end{tikzcd}$$

"Canned" Test Requests

For testing purposes, the only values in the test code in this section that you should change are the "USERID," "PASSWORD," and "SERVERNAME." Enter the user ID, password, and server name you received in the registration e-mail for ***testing***. Your user ID and password never change, but the server name will change later when you send "live" requests. The "live" server name will be provided when the ICCC provides you with access to the production server. ***All remaining code in the test scripts provided below must remain unchanged.***

All of the test script code contained in this document can be cut and pasted for your use in testing the software. To copy the test script code from this PDF file, click on the icon for "Text Selector" and highlight the code. (The icon will look like abc or T

depending on your version of Adobe Acrobat.) You can then copy the code and paste it into your test document.

Valid Test Requests

There are six valid requests for this procedure. Be sure to note the request numbers so you can match up the responses you will receive as provided in the *"Canned" Test Responses* section.

Valid Test Request #1

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
EXPRESS</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>20852<
/ZipDestination><Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Contain
er><Size>REGULAR</Size><Machinable></Machinable></Package></RateRequest>
```

Valid Test Request #2

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest USERID=
"xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>Priority</Service>
<ZipOrigination>20770</ZipOrigination><ZipDestination>90210</ZipDestination><
Pounds>5</Pounds><Ounces>1</Ounces><Container>0-1096</Container><Size>
REGULAR</Size><Machinable></Machinable></Package></RateRequest>
```

Valid Test Request #3

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest USERID=
"xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>Parcel</Service>
<ZipOrigination>20770</ZipOrigination><ZipDestination>90210</ZipDestination><
Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Container><Size>Regular<
/Size><Machinable>True</Machinable></Package></RateRequest>
```

Valid Test Request #4

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest USERID=
"xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>Parcel</Service>
<ZipOrigination>20770</ZipOrigination><ZipDestination>90210</ZipDestination><
Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Container><Size>Regular<
/Size><Machinable>False</Machinable></Package></RateRequest>
```

Valid Test Request #5

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>Parcel
</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>09007</ZipDes
tination><Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Container><Siz
e>Regular</Size><Machinable>False</Machinable></Package></RateRequest>
```

Valid Test Request #6

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxx" PASSWORD="xxxxxxx"><Package ID="0"><Service>Priority
</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>09021</ZipDes
tination><Pounds>5</Pounds><Ounces>1</Ounces><Container>None</Container><Size
>Regular</Size><Machinable>False</Machinable></Package></RateRequest>
```

Pre-Defined Error Requests

There are eight pre-defined errors included for this procedure. Be sure to note the request numbers so you can match up the responses you will receive as provided in the *"Canned" Test Responses* section.

Pre-defined Error Request #1: "*Invalid ZIP Code for Sender*"

The pre-defined error in this request is using "99999" for the `<ZipOrigination>` input.

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest USERID=
"xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Parcel</Service><ZipOrigination>99999</ZipOrigination><ZipDestination>90210</
ZipDestination><Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Containe
r><Size>Regular</Size><Machinable>False</Machinable></Package></RateRequest>
```

Pre-defined Error Request #2: *"Invalid ZIP Code for Receiver"*

The pre-defined error in this request is using "99999" for the `<ZipDestination>` input.

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Parcel</Service><ZipOrigination>90210</ZipOrigination><ZipDestination>99999</
ZipDestination><Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Containe
r><Size>Regular</Size><Machinable>False</Machinable></Package></RateRequest>
```

Pre-defined Error Request #3: *"Invalid Package Size for Parcel Post"*

The pre-defined error in this request is using anything other than "Regular," "Large," or "Oversize" for `<Size>` ***and*** the `<Service>` input is "Parcel."

```
Http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Parcel</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>90210</
ZipDestination><Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Containe
r><Size>Normal</Size><Machinable>False</Machinable></Package></RateRequest>
```

Pre-defined Error Request #4: *"Invalid Container for Priority Mail"*

The pre-defined error in this request is using anything other than "None," "0-1095," "0-1096," "0-1097," "0-1098," "EP14," or "EP14F" for `<Container>` ***and*** the `<Service>` input is "Priority."

```
http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxx"><Package ID="0"><Service>
Priority</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>90210
</ZipDestination><Pounds>10</Pounds><Ounces>0</Ounces><Container>Mailbox</Con
tainer><Size>Regular</Size><Machinable></Machinable></Package></RateRequest>
```

Pre-defined Error Request #5: *"Invalid Machinable Input for Parcel Post"*

The pre-defined error in this request is using anything other than "True" or "False" for `<Machinable>` ***and*** the `<Service>` input is "Parcel."

```
http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Parcel</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>90210</
ZipDestination><Pounds>10</Pounds><Ounces>0</Ounces><Container>None</Containe
r><Size>Regular</Size><Machinable>None</Machinable></Package></RateRequest>
```

Pre-defined Error Request #6: *"Invalid Weight for Express Mail® and Priority Mail"*

The pre-defined error in this request is using "Express" or "Priority" for `<Service>`*and*:

- the `<Pounds>` and `<Ounces>` inputs are both set to "0"
- the `<Pounds>` input is set to "0" and the `<Ounces>` input is greater than 1120

```
http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Express</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>90210<
/ZipDestination><Pounds>0</Pounds><Ounces>0</Ounces><Container>None</Containe
r><Size>Regular</Size><Machinable></Machinable></Package></RateRequest>
```

Pre-defined Error Request #7: *"Invalid Weight for Parcel Post"*

The pre-defined error in this request is using "Parcel" for <Service> *and* both the `<Pounds>` and `<Ounces>` inputs are "0." ***Both*** values cannot be "0."

```
http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Parcel</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>90210</
ZipDestination><Pounds>0</Pounds><Ounces>0</Ounces><Container>None</Container
><Size>Regular</Size><Machinable>False</Machinable></Package></RateRequest>
```

Pre-defined Error Request #8: *"Invalid Weight for Pounds"*

The pre-defined error in this request is using a number greater than 70 for `<Pounds>`.

```
http://SERVERNAME/ShippingAPITest.dll?API=Rate&XML=<RateRequest
USERID="xxxxxxxx" PASSWORD="xxxxxxxx"><Package ID="0"><Service>
Express</Service><ZipOrigination>20770</ZipOrigination><ZipDestination>90210<
/ZipDestination><Pounds>200</Pounds><Ounces>0</Ounces><Container>None</Contai
ner><Size>Regular</Size><Machinable></Machinable></Package></RateRequest>
```

"Live" Request

All of the sample code contained in this document can be cut and pasted for your use in building the software. To copy the sample code from this PDF file, click on the icon for "Text Selector" and highlight the code. (The icon will look like

abc or T

depending on your version of Adobe Acrobat.) You can then copy the sample code and paste it into your code document.

Remember that all data and attribute values in this document are for illustration purposes and are to be replaced by your actual values. For instance, a line of sample code may be:

```
<FromName>Joe Smith</FromName>
```

In this instance, you will replace "Joe Smith" with the name of the person sending the package when making your request. ***Also remember that you are provided with a different server name to send "live" requests.***

When building the XML request, pay particular attention to the ***order and case*** for tags.

The table below presents the ***required*** XML input tags for generating "Live" requests and the restrictions on the values allowed. An error message will be returned if the tag does not contain a value or if an incorrect value is entered. Also, be aware of the maximum character amounts allowed for some tags. If the user enters more than those amounts, an error will not be generated. ***The API will simply pass in the characters up to the maximum amount allowed and disregard the rest.*** This is important since the resulting value could prevent delivery.

Input	**XML Tag**	**Values Allowed**
Type of Request	`<RateRequest…`	Input tag exactly as presented.
User ID	`…USERID="userid"…`	Use user ID provided with registration.
Password	`…PASSWORD="password">`	Use password provided with registration.
Package ID Number	`<Package ID="#">`	No restrictions on number or type of characters.
Type of Service Requested	`<Service>`	The service type must be one of the following: "Express" "First Class" "Priority" "Parcel" "BPM" (Bound Printed Matter) "Library" "Media" The API validates the entry to the service type.
Origination ZIP Code	`<ZipOrigination>`	Input tag exactly as presented. ZIP Codes must be valid. Maximum characters allowed: 5
Destination ZIP Code	`<ZipDestination>`	Input tag exactly as presented. ZIP Codes must be valid. Maximum characters allowed: 5
Package Weight in Pounds	`<Pounds>`	Value must be numeric. Package weight cannot exceed 70 pounds. (First-Class Mail cannot exceed 13 ounces. Bound Printed Matter cannot exceed 15 pounds).
Package Weight in Ounces	`<Ounces>`	Value must be numeric. Package weight cannot exceed 70 pounds. (First-Class Mail cannot exceed 13 ounces. Bound Printed Matter cannot exceed 15 pounds).
Shipping Container	`<Container>`	See below for valid entries.
Package Size	`<Size>`	The API validates the descriptions entered to the package size. See below for valid entries.
Machinable	`<Machinable>`	The Machinable tag is required for Parcel Post only. (Machinable criteria do not apply to Priority Mail, Express Mail, Bound Printed Matter, Library Mail, or Media Mail.) The size, content, and weight of a package can all determine whether a Parcel Post package is machinable or nonmachinable. The value entered must be either "True" or "False." The "True" or "False tag specifically applies to the size and content of the package. Based on weight, the calculator API will automatically apply the nonmachinable surcharge on Parcel Post packages weighing less than 6 ounces or over 35 pounds.

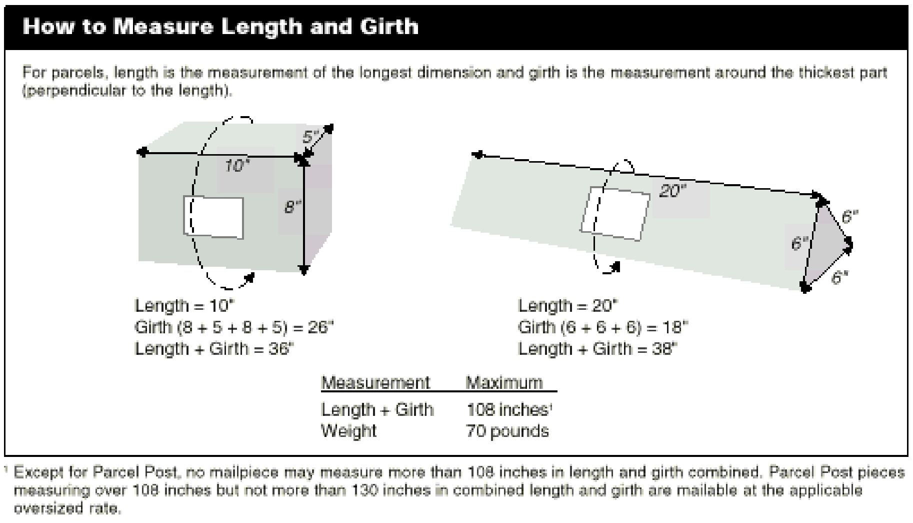
The `<Container>` field must contain one of the following valid packaging type names:

Package Name	Description
Express Mail	
None	For someone using their own package
0-1093	Express Mail Box, 12.25 x 15.5 x
0-1094	Express Mail Tube, 36 x 6
EP13A	Express Mail Cardboard Envelope, 12.5 x 9.5
EP13C	Express Mail Tyvek Envelope, 12.5 x 15.5
EP13F	Express Mail Flat Rate Envelope, 12.5 x 9.5
Priority Mail	
None	For someone using their own package
0-1095	Priority Mail Box, 12.25 x 15.5 x 3
0-1096	Priority Mail Video, 8.25 x 5.25 x 1.5
0-1097	Priority Mail Box, 11.25 x 14 x 2.25
0-1098	Priority Mail Tube, 6 x 38
EP14	Priority Mail Tyvek Envelope, 12.5 x 15.5
EP14F	Priority Mail Flat Rate Envelope, 12.5 x 9.5
First-Class Mail	
None	For someone using their own package
Parcel Post	
None	For someone using their own package
Bound Printed Matter	
None	For someone using their own package
Library Mail	
None	For someone using their own package
Media Mail	
None	For someone using their own package

Package `<Size>` must be one of the following:

Package Size	Description	Service(s) Available
Regular	package length plus girth (84 inches or less) (See below for how to measure length and girth.)	Parcel Post Priority Mail First-Class Mail Express Mail Bound Printed Matter Library Mail Media Mail
Large	package length plus girth (Priority Mail and Parcel Post parcels that weigh less than 15 pounds but measure more than 84 inches (but less than 108 inches) in combined length and girth are charged the applicable rate for a 15-pound parcel.) (This tag does not affect the rate for Express Mail, Bound Printed Mater, Media Mail or Library Mail.)	Parcel Post Priority Mail First-Class Mail Express Mail Bound Printed Matter Library Mail Media Mail
Oversize	package length plus girth (more than 108 but not more than 130 inches) (Parcel Post packages that measure more than 108 inches but not more than 130 inches in combined length and girth are charged the oversized Parcel Post rate regardless of the weight of the package.)	Parcel Post ***only***

To determine the appropriate package size, measure the combined length and girth in inches as shown below:



How to Measure Length and Girth

For parcels, length is the measurement of the longest dimension and girth is the measurement around the thickest part (perpendicular to the length).

Length = 10"
Girth (8 + 5 + 8 + 5) = 26"
Length + Girth = 36"

Length = 20"
Girth (6 + 6 + 6) = 18"
Length + Girth = 38"

Measurement	Maximum
Length + Girth	108 inches[1]
Weight	70 pounds

[1] Except for Parcel Post, no mailpiece may measure more than 108 inches in length and girth combined. Parcel Post pieces measuring over 108 inches but not more than 130 inches in combined length and girth are mailable at the applicable oversized rate.

XML Request

The "Live" XML request should be in the form:

```
<RateRequest USERID="xxxxxxx" PASSWORD="xxxxxxxx">
      <Package ID="0">
            <Service>EXPRESS</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>54324</ZipDestination>
            <Pounds>2</Pounds>
            <Ounces>0</Ounces>
            <Container>None</Container>
            <Size>Regular</Size>
            <Machinable></Machinable>
      </Package>
      <Package ID="1">
            <Service>PRIORITY</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>02912</ZipDestination>
            <Pounds>20</Pounds>
            <Ounces>8</Ounces>
            <Container>None</Container>
            <Size>Regular</Size>
      </Package>
      <Package ID="2">
            <Service>PARCEL</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>02912</ZipDestination>
            <Pounds>20</Pounds>
            <Ounces>8</Ounces>
            <Container>None</Container>
            <Size>Regular</Size>
            <Machinable>True</Machinable>
      </Package>
</RateRequest>
```

Visual Basic Request

Using the Microsoft XML object model in Visual Basic, such a request can be built as shown below. In this code sample, the data needed to build the XML is obtained from a form. The `ServiceType` element is obtained from an option button control and the `ImageType` is from a combo box control. All other fields are obtained from text box controls.

```
Dim xmlDoc As New DOMDocument
Dim RequestLevel As IXMLDOMElement
Dim PackageLevel As IXMLDOMElement
Dim PackageElementLevel As IXMLDOMElement
Dim i as Integer
Dim t As Variant

Set RequestLevel = xmlDoc.createElement("RateRequest")
RequestLevel.setAttribute "USERID"
RequestLevel.setAttribute "PASSWORD"

For i = 0 To ?

    Set PackageLevel = xmlDoc.createElement("Package")
    PackageLevel.setAttribute "ID", i

    Set PackageElementLevel = xmlDoc.createElement("Service")
    Select Case cmbServiceType.ListIndex
    Case 0
        Set t = xmlDoc.createTextNode("EXPRESS")
    Case 1
        Set t = xmlDoc.createTextNode("PRIORITY")
    Case 2
         Set t = xmlDoc.createTextNode("PARCEL")
    End Select
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("ZipOrigination")
    Set t = xmlDoc.createTextNode(txtOriginZip.Text)
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("ZipDestination")
    Set t = xmlDoc.createTextNode(txtDestZip.Text)
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Pounds")
    Set t = xmlDoc.createTextNode(txtPounds.Text)
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Ounces")
    Set t = xmlDoc.createTextNode(txtOunces.Text)
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Container")
    Set t = xmlDoc.createTextNode
    (MapToPackageCode(cmbContainerDesc.Text))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Size")
    If OptOverSize(0) Then
        Set t = xmlDoc.createTextNode("REGULAR")
    ElseIf OptOverSize(1) Then
        Set t = xmlDoc.createTextNode("LARGE")
    Else
         Set t = xmlDoc.createTextNode("OVERSIZE")
    End If
    PackageElementLevelappendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Call RequestLevel.appendChild(PackageLevel)

Next i

Call xmlDoc.appendChild(RequestLevel)
```

Perl Request

This sample requires the Perl XML parser and libwww-perl. Both are available from the Comprehensive Perl Archive Network (CPAN) at http://www.perl.com/CPAN/. Build the XML request with the parameters passed from the HTML form.

```
$query = new CGI;
$query->use_named_parameters(1);

$rateReqDoc = new XML::DOM::Document;
$rateReqEl = $rateReqDoc->createElement('RateRequest');
$rateReqEl->setAttribute('USERID', 'xxxxxxxx');
$rateReqEl->setAttribute('PASSWORD', 'xxxxxxxx');
$rateReqDoc->appendChild($rateReqEl);

$packageEl = $rateReqDoc->createElement('Package');
$packageEl->setAttribute('ID', '0');
$rateReqEl->appendChild($packageEl);

$serviceEl = $rateReqDoc->createElement('Service');
$serviceText = $rateReqDoc->createTextNode($query->param('service'));
$serviceEl->appendChild($serviceText);
$packageEl->appendChild($serviceEl);

$zipOrigEl = $rateReqDoc->createElement('ZipOrigination');
$zipOrigText = $rateReqDoc->createTextNode($query->param('fromZip'));
$zipOrigEl->appendChild($zipOrigText);
$packageEl->appendChild($zipOrigEl);

$zipDestEl = $rateReqDoc->createElement('ZipDestination');
$zipDestText = $rateReqDoc->createTextNode($query->param('toZip'));
$zipDestEl->appendChild($zipDestText);
$packageEl->appendChild($zipDestEl);

$poundsEl = $rateReqDoc->createElement('Pounds');
$poundsText = $rateReqDoc->createTextNode($query->param('pounds'));
```

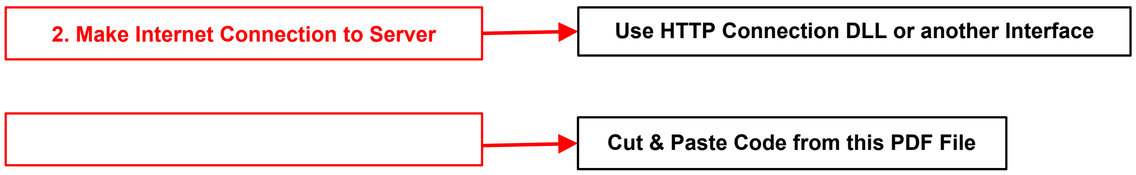
```
$poundsEl->appendChild($poundsText);
$packageEl->appendChild($poundsEl);

$ouncesEl = $rateReqDoc->createElement('Ounces');
$ouncesText = $rateReqDoc->createTextNode($query->param('ounces'));
$ouncesEl->appendChild($ouncesText);
$packageEl->appendChild($ouncesEl);

$containerEl = $rateReqDoc->createElement('Container');
$containerText = $rateReqDoc->createTextNode('NONE');
$containerEl->appendChild($containerText);
$packageEl->appendChild($containerEl);

$oversizeEl = $rateReqDoc->createElement('Size');
$oversizeText = $rateReqDoc->createTextNode('Regular');
$oversizeEl->appendChild($oversizeText);
$packageEl->appendChild($oversizeEl);
```

ASP Request

Using the Microsoft XML object model in Visual Basic script in an Active Server Page, a request can be built as follows:

```
<%@ Language=VBScript %>
<%
set xmlDoc = Server.CreateObject("MSXML.DOMDocument")

Set RequestLevel = xmlDoc.createElement("RateRequest")
RequestLevel.setAttribute "USERID", "xxxxxxxx"
RequestLevel.setAttribute "PASSWORD", "xxxxxxxx"

For i = 0 To ?

    Set PackageLevel = xmlDoc.createElement("Package")
    PackageLevel.setAttribute "ID", i

    Set PackageElementLevel = xmlDoc.createElement("Service")
    Set t = xmlDoc.createTextNode(Request.Form("selServiceType"))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("ZipOrigination")
    Set t = xmlDoc.createTextNode(Request.Form("txtOriginZip"))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("ZipDestination")
    Set t = xmlDoc.createTextNode(Request.Form("txtDestZip"))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Pounds")
    Set t = xmlDoc.createTextNode(Request.Form("txtPounds"))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Ounces")
    Set t = xmlDoc.createTextNode(Request.Form("txtOunces"))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Container")
    Set t = xmlDoc.createTextNode(Request.Form("selContainerDesc"))
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Set PackageElementLevel = xmlDoc.createElement("Size")
    Set t = xmlDoc.createTextNode(Request.Form("selSize")
    PackageElementLevel.appendChild (t)
    Call PackageLevel.appendChild(PackageElementLevel)

    Call RequestLevel.appendChild(PackageLevel)

Next

Call xmlDoc.appendChild(RequestLevel)
%>
.
.
.
```

Steps 2 & 3: Make the Internet Connection and Send the XML Request


2. Make Internet Connection to Server
Use HTTP Connection DLL or another Interface
Cut & Paste Code from this PDF File

These two steps are presented together to simplify things. The two steps actually involve four separate functions:

1. Making the connection to the USPS Shipping API server (test server or production server)
2. Sending the request (whether Visual Basic, Perl, ASP, or any other language)
3. Receiving the response from the API server
4. Closing the Internet connection

These steps are identical for sending *"Canned"* test requests or *"Live"* requests. ***Remember, however, that you are provided with a different server name to send "live" requests.***

This section provides three samples to make the Internet connection. This is not an all-inclusive list. It simply represents the most common and easiest ways to make the Internet connection.

- Using the USPS-supplied HTTP Connection DLL

  The HTTP Connection DLL is recommended for NT systems. This software, created specifically for the USPS API implementation, provides e-tailers with a thread-safe sockets interface to submit XML requests and receive XML responses from the API server.

- Using Microsoft's WinInet

  Although you can use the WinInet DLL to make the connection to the API server, it is not recommended for server applications due to limitations in the DLL. It is recommended that you either use the USPS-supplied HTTP Connection DLL or write your own sockets interface that can be used to make multiple connections and will remain thread-safe.

- Using Perl

  Perl is recommended for UNIX systems. This sample requires the Perl XML parser and libwww-perl. Both are available from the Comprehensive Perl Archive Network (CPAN) at http://www.perl.com/CPAN/.

Using HTTP Connection DLL

To obtain this code you must submit a Licensing Agreement. See the *Administrative Guide for APIs* for the agreement.

Using WinInet

This sample code shows how to use Microsoft's WinInet DLL to make the Internet connection, using either the "GET" or "POST" (necessary for requests over 2K in size) methods. XMLSTRING represents the URL-encoded XML request and SERVERNAME indicates the name of the USPS web site to which you are connecting.

For *"Canned"* test requests the code should read:

```
File = "/ShippingAPItest.dll?"
```

For *"Live"* requests the code should read:

```
File = "/ShippingAPI.dll?"
```

Input:

```
Dim hOpen As Long, hConnection As Long, hFile As Long, numread As Long
Dim File As String, xml As String, sHeader As String, htmlFile As String, tmp
As String * 2048
Dim bDoLoop As Boolean

File = "/ShippingAPI.dll?"
xml = "API=Rate&XML=" & XMLSTRING

hOpen = InternetOpen("", 1, vbNullString, vbNullString, 0)

hConnection = InternetConnect(hOpen, SERVERNAME, 0, _
        "", "", 3, 0, 0)

''''''''''''''''''''''''
'get
'File = File & xml
```

```
'hFile = HttpOpenRequest(hConnection, "GET", File, "HTTP/1.0", vbNullString,
0, 0, 0)
'OR
''''''''''''''''''''''

''''''''''''''''''''''
' post
hFile = HttpOpenRequest(hConnection, "POST", File, "HTTP/1.0", vbNullString,
0, 0, 0)

sHeader = "Content-Type: application/x-www-form-urlencoded" _
                       & vbCrLf

Call HttpAddRequestHeaders(hFile, _
              sHeader, Len(sHeader), 0)
''''''''''''''''''''''

bDoLoop = HttpSendRequest(hFile, vbNullString, 0, xml, Len(xml))

bDoLoop = True
    While bDoLoop
        tmp = vbNullString
        bDoLoop = InternetReadFile(hFile, tmp, Len(tmp), numread)
        If Not bDoLoop Then
            Exit Sub
        Else
            htmlFile = htmlFile & Left$(tmp, numread)
            If Not CBool(numread) Then bDoLoop = False
        End If
    Wend

If hFile <> 0 Then InternetCloseHandle (hFile)
If hConnection <> 0 Then InternetCloseHandle (hConnection)
If hOpen <> 0 Then InternetCloseHandle (hOpen)
```

Using Perl

"SERVERNAME" indicates the name of the USPS web site to which you are connecting:

For *"Canned"* test requests the code should read:

```
$req = new HTTP::Request 'POST', 'http://SERVERNAME/ShippingAPItest.dll';
```

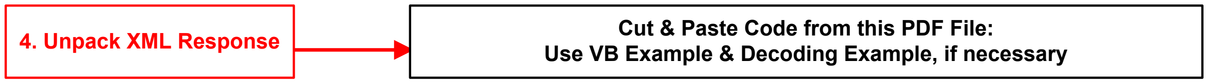
For *"Live"* requests the code should read:

```
$req = new HTTP::Request 'POST', 'http://SERVERNAME/ShippingAPI.dll';
```

Input:

```
print "content-type: text/html\n\n";
print $htmlBegin;

$ua = new LWP::UserAgent;
$req = new HTTP::Request 'POST', 'http://SERVERNAME/ShippingAPI.dll';
$req->content_type('application/x-www-form-urlencoded');
$req->content('API=Rate&XML=' . $rateReqDoc->toString);
$response = $ua->request($req);
if ($response->is_success) {
    $resp = $response->content;
} else {
    print "<p>There was an error processing your request</p>\n";
    print $htmlEnd;
    exit 0;
}
```

Step 4: Unpack the XML Response


4. Unpack XML Response
Cut & Paste Code from this PDF File:
Use VB Example & Decoding Example, if necessary

This step is identical for unpacking *"Canned"* test responses or *"Live"* responses.

Types of Responses

When the USPS Shipping API returns a response, it will either return a successful response document or an error document. Anytime you receive a response, you should check to see if the document is `<Error>.` Refer to the *Errors* section.

Using Visual Basic

Using the Microsoft XML object model in Visual Basic, such responses can be unpacked as follows:

```
Dim xmlDoc As New DOMDocument
Dim nodeList As IXMLDOMNodeList
Dim n As IXMLDOMNode, e As IXMLDOMNode, t As IXMLDOMNode
Dim i As Integer, j As Integer, k As Integer
Dim strname As String

xmlDoc.validateOnParse = False
xmlDoc.loadXML (xmlstr) 'Response
Set nodeList = xmlDoc.getElementsByTagName("Error")
If nodeList.length > 0 Then  'Top-level Error
    Call ParseError(nodeList.Item(0))
Else  'no Top-level Error
    Set nodeList = xmlDoc.getElementsByTagName("Package")
    For i = 0 To nodeList.length - 1
       Set n = nodeList.Item(i)
       For j = 0 To n.childNodes.length - 1
            Set e = n.childNodes.Item(j)
        If e.nodeName = "Error" Then 'Lower-level error
            Call ParseError(e)
        Else  'No error in Package
            Select Case e.nodeName
                        Case "Service"
                        TxtService(i).Text = e.firstChild.nodeValue
                        Case "ZipOrigination"
                        TxtOriginZip(i).Text = e.firstChild.nodeValue
                        Case "ZipDestination"
                        txtDestZip(i).Text = e.firstChild.nodeValue
                        Case "Pounds"
                        txtPounds(i).Text = e.firstChild.nodeValue
                        Case "Ounces"
                        txtOunces(i).Text = e.firstChild.nodeValue
                        Case "Zone"
                        txtZone(i).Text = e.firstChild.nodeValue
                        Case "Postage"
                        txtPostageDue(i).Text = e.firstChild.nodeValue
                        Case "RestrictionCodes"
                        txt RestrictionCodes (i).Text = e.firstChild.nodeValue
                        Case "RestrictionDescription"
                        txt RestrictionDescription (i).Text = e.firstChild.nodeValue
                    End Select
        End If
    Next j
    Next i
End If



Private Sub ParseError(ochild As IXMLDOMNode)

Dim t As IXMLDOMNode

    For i = 0 To ochild.childNodes.length - 1
        Set t = ochild.childNodes.Item(i)
        Select Case t.nodeName
        Case "Source"
        Case "Number"
        Case "Description"
                txtDesc.Text = t.firstChild.nodeValue
        Case "HelpFile"
        Case "HelpContext"
        End Select
    Next i
End Sub
```

Using Perl

This sample requires the Perl XML parser and libwww-perl. Both are available from the Comprehensive Perl Archive Network (CPAN) at http://www.perl.com/CPAN/. Unpack the response and display as HTML.

```
$parser = new XML::DOM::Parser;
$rateRespDoc = $parser->parse($resp);

if ($rateRespDoc->getDocumentElement->getNodeName eq 'Error') {
    print "<p>There was an error processing your request</p>\n";
    print $htmlEnd;
    exit 0;
}

$packageList = $rateRespDoc->getElementsByTagName('Package');
$n = $packageList->getLength;
```

```
for ($i = 0; $i < $n; $i++) {
    $packageNode = $packageList->item($i);
    $tmpList = $packageNode->getElementsByTagName('ZipOrigination');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $zipOrig = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $zipOrig = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple ZipOrigination tags in Package tag--
>\n";
    } else {
        $zipOrig = '';
        print "<!-- No ZipOrigination tag -->\n";
    }

    $tmpList = $packageNode->getElementsByTagName('ZipDestination');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $zipDest = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $zipDest = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple ZipDestination tags in Package tag--
>\n";
    } else {
        $zipDest = '';
        print "<!-- No ZipDestination tag -->\n";
    }

    $tmpList = $packageNode->getElementsByTagName('Service');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $service = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $service = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple Service tags in Package tag-->\n";
    } else {
        $service = '';
        print "<!-- No Service tag -->\n";
    }

    $tmpList = $packageNode->getElementsByTagName('Pounds');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $pounds = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $pounds = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple Pounds tags in Package tag-->\n";
    } else {
        $pounds = '';
        print "<!-- No Pounds tag -->\n";
    }

    $tmpList = $packageNode->getElementsByTagName('Ounces');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $ounces = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $ounces = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple Ounces tags in Package tag-->\n";
    } else {
        $ounces = '';
        print "<!-- No Ounces tag -->\n";
    }

    $tmpList = $packageNode->getElementsByTagName('Postage');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $postage = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $postage = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple Postage tags in Package tag-->\n";
    } else {
        $postage = '';
        print "<!-- No Postage tag -->\n";
    }
$tmpList = $packageNode->getElementsByTagName('RestrictionCodes');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $restcodes = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $restcodes = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple RestrictionCodes tags in Package tag-
->\n";
    } else {
        $restcodes = '';
        print "<!-- No RestrictionCodes tag -->\n";

$tmpList =
    $packageNode->getElementsByTagName('RestrictionDescription');
    $m = $tmpList->getLength;
    if ($m == 1) {
        $restdesc = $tmpList->item(0)->getFirstChild->getNodeValue;
    } elsif ($m > 1) {
        $restdesc = $tmpList->item(0)->getFirstChild->getNodeValue;
        print "<!-- XML Error: multiple RestrictionDescription tags in
Package tag-->\n";
    } else {
        $restdesc = '';
        print "<!-- No RestrictionDescription tag -->\n";

    print "                          <P>\n";
    print "                            ";
    print "Sending your package weighing " . ${pounds} . " pounds ";
    print "and " . ${ounces} . " ounces from ZIP code " . ${zipOrig};
    print " to ZIP Code " . ${zipDest} . " by " . ${service} . " Mail ";
    print "will cost: <BR>\n";
    print "                            ";
    print '<FONT SIZE="+1" COLOR="#CC0000">$' . ${postage} . "</FONT>\n";
    print "                          </P>\n";
}

print $htmlEnd;
```

Using ASP

Using the Microsoft XML object model in Visual Basic, such responses can be unpacked as follows

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
set xmlDoc = Server.CreateObject("MSXML.DOMDocument")

xmlDoc.validateOnParse = False
xmlDoc.loadXML (xmlStr) 'Response
If xmlDoc.documentElement.nodeName = "Error" then  'Top-level Error
Set nodeList = xmlDoc.getElementsByTagName("Error")
    Set n = nodeList.Item(0)
    For i = 0 To n.childNodes.length - 1
      Set e = n.childNodes.Item(i)
      Select Case e.nodeName
          Case "Source"
          Case "Number"
          Case "Description"
          Response.Write("Error: " & e.firstChild.nodeValue)
          Case "HelpFile"
          Case "HelpContext"
      End Select
    Next
Else  'no Top-level Error
    Set nodeList = xmlDoc.getElementsByTagName("Package")
    For i = 0 To nodeList.length - 1
       Set n = nodeList.Item(i)
       For j = 0 To n.childNodes.length - 1
            Set e = n.childNodes.Item(j)
            If e.nodeName = "Error" Then 'Lower-level error
                  For k = 0 To e.childNodes.length - 1
                        Set t = e.childNodes.Item(k)
                        Select Case t.nodeName
                              Case "Source"
                              Case "Number"
                              Case "Description"
                              Response.Write("Error: " &
                              t.firstChild.nodeValue)
                            Case "HelpFile"
                            Case "HelpContext"
                        End Select
                  Next
                  Select Case e.nodeName
                        Case "ZipOrigination"
                              Response.Write("From " &
                              e.firstChild.nodeValue)
                        Case "ZipDestination"
                              Response.Write("To " &
                              e.firstChild.nodeValue)
                        Case "Pounds"
```

```
                              Response.Write("Pounds: " &
                              e.firstChild.nodeValue
                        Case "Ounces"
                              Response.Write("Ounces: " &
                              e.firstChild.nodeValue)
                        Case "Zone"
                              Response.Write("Zone: " &
                              e.firstChild.nodeValue)
                        Case "Postage"
                              Response.Write("Postage: " &
                              e.firstChild.nodeValue)
                        Case "RestrictionCodes"
                              Response.Write("Restriction Codes: " &
                              e.firstChild.nodeValue)
                        Case "RestrictionDescription"
                        Response.Write("Restriction Description: " &
                              e.firstChild.nodeValue)
                  End Select
      End If
      Next
    Next
End If
%>
</BODY>
</HTML>
```

Errors

Error conditions are handled at the main XML document level. For APIs that can handle multiple transactions, the error conditions for requests for multiple responses to be returned together are handled at the response level. For example: an API developer sends a request for rates for two packages. If the addresses are non-existent, an "Error document" is returned to the user. On the other hand, if the address for the first package is acceptable but not the second, the response document contains the information for the first address, but under the XML tag for the second address there is an error tag.

Error documents follow the Visual Basic error standards and have following format:

```
<Error>
      <Number></Number>
      <Source></Source>
      <Description></Description>
      <HelpFile></HelpFile>
      <HelpContext></HelpContext>
</Error>
```

where:

- Number = the error number generated by the API server
- Source = the component and interface that generated the error on the API server
- Description = the error description
- HelpFile = [reserved]
- HelpContext = [reserved]

Errors that are further down in the hierarchy also follow the above format.

Some web items allow you to submit multiple requests within a single XML document. For instance, you may request multiple rate quotes, where each rate quote is identified by an "ID" attribute. Within a given package, an `<Error>` may be returned. For multiple request documents, you need to check if there is an `<Error>` within a given `<Package>`, `<Address>`, etc.

```
<RateRequest>
      <Package ID="0">
      …
      </Package>
      <Package ID="1">
            <Error>
            …
            </Error>
      </Package>
</RateRequest>
```

Output

After following Technical Step 4 and unpacking the XML response, you will have the output from your request. This section describes the different outputs resulting from "*Canned*" test requests, and "*Live*" requests. Both types of requests result in an XML response with the following tags:

Output	XML Tag	Comments
Type of Response	`<RateResponse>`	
Package Identification Number	`<Package ID="#">`	
Type of Service Required	`<Service>`	
Origination ZIP Code	`<ZipOrigination>`	
Destination ZIP Code	`<ZipDestination>`	
Package Weight (Pounds)	`<Pounds>`	
Package Weight (Ounces)	`<Ounces>`	
Shipping Container	`<Container>`	
Package Size	`<Size>`	
Postage Rate Charged	`<Postage>`	
Postal Zone	`<Zone>`	U.S. Postal Service Zones are used for Priority Mail packages over 5 lbs.
APO/FPO Restriction Codes	`<RestrictionCodes>`	Optional. This data is provided if the Destination ZIP Code is an APO/FPO ZIP Code.
APO/FPO Restriction Descriptions	`<RestrictionDescription>`	Optional. All of the APO/FPO codes and descriptions are available at www.uspswebtools.com.

"Canned" Test Responses

For your test to be successful, the following responses to Valid Test Requests and Pre-defined Test Requests should be ***verbatim***. If any values were changed in your request, the following default error will be returned:

```
<?xml version="1.0" ?>
<RateResponse>
```

```
      <Package ID="0">
          <Error>
          <Number>-2147219040</Number>
          <Source>SOLServerTest;SOLServerTest.CallRateDll</Source>
          <Description>This Information has not been included in this Test
          Server.</Description>
          <HelpFile />
          <HelpContext></HelpContext>
          </Error>
      </Package>
</RateResponse>
```

Although the input may be valid, the response will still raise this error, because those particular values have not been included in this test server. Refer to the *Errors* section for an explanation of any other returned errors.

Response to Valid Test Request #1

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Service>Express</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>20852</ZipDestination>
            <Pounds>10</Pounds>
            <Ounces>0</Ounces>
            <Container>None</Container>
            <Size>REGULAR</Size>
            <Zone>1</Zone>
            <Postage>33.65</Postage>
      </Package>
</RateResponse>
```

Response to Valid Test Request #2

```
<?xml version="1.0" ?>
<RateResponse>
   <Package ID="0">
      <Service>Priority</Service>
      <ZipOrigination>20770</ZipOrigination>
      <ZipDestination>90210</ZipDestination>
      <Pounds>5</Pounds>
      <Ounces>1</Ounces>
      <Container>0-1096</Container>
      <Size>REGULAR</Size>
      <Zone>8</Zone>
      <Postage>10.35</Postage>
   </Package>
</RateResponse>
```

Response to Valid Test Request #3

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Service>Parcel</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>90210</ZipDestination>
            <Pounds>10</Pounds>
            <Ounces>0</Ounces>
            <Container>None</Container>
            <Size>REGULAR</Size>
            <Machinable>TRUE</Machinable>
            <Zone>8</Zone>
            <Postage>15.19</Postage>
      </Package>
</RateResponse>
```

Response to Valid Test Request #4

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Service>Parcel</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>90210</ZipDestination>
            <Pounds>10</Pounds>
            <Ounces>0</Ounces>
            <Container>None</Container>
            <Size>REGULAR</Size>
            <Machinable>FALSE</Machinable>
            <Zone>8</Zone>
            <Postage>17.19</Postage>
      </Package>
</RateResponse>
```

Response to Valid Test Request #5

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
          <Service>Parcel</Service>
          <ZipOrigination>20770</ZipOrigination>
          <ZipDestination>09007</ZipDestination>
          <Pounds>10</Pounds>
          <Ounces>0</Ounces>
          <Container>None</Container>
          <Size>REGULAR</Size>
          <Machinable>FALSE</Machinable>
          <Zone>3</Zone>
          <Postage>7.68</Postage>
          <RestrictionCodes>B-B1-C</RestrictionCodes>
          <RestrictionDescription>
          B. Form 2976-A is required for all mail weighing 16 ounces or more,
          with exceptions noted below. In addition, mailers must properly
          complete required customs documentation when mailing any
          potentially dutiable mail addressed to an APO or FPO regardless of
          weight. B1. Form 2976 or 2976-A is required. Articles are liable
          for customs duty and/or purchase tax unless they are bona fide
          gifts intended for use by military personnel or their dependents.
          C. Cigarettes and other tobacco products are prohibited.
          </RestrictionDescription>
      </Package>
</RateResponse>
```

Response to Valid Test Request #6

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
        <Service>Priority</Service>
        <ZipOrigination>20770</ZipOrigination>
        <ZipDestination>09021</ZipDestination>
        <Pounds>5</Pounds>
        <Ounces>1</Ounces>
        <Container>None</Container>
        <Size>REGULAR</Size>
        <Machinable>FALSE</Machinable>
        <Zone>3</Zone>
        <Postage>7.90</Postage>
        <RestrictionCodes> B-B1-C-D-U</RestrictionCodes>
        <RestrictionDescription>
        B. Form 2976-A is required for all mail weighing 16 ounces or more,
        with exceptions noted below. In addition, mailers must properly
        complete required customs documentation when mailing any potentially
        dutiable mail addressed to an APO or FPO regardless of weight. B1.
        Form 2976 or 2976-A is required. Articles are liable for customs duty
        and/or purchase tax unless they are bona fide gifts intended for use
        by military personnel or their dependents. C. Cigarettes and other
        tobacco products are prohibited. D. Coffee is prohibited. U. Parcels
        must weigh less than 16 ounces when addressed to Box R.
        </RestrictionDescription>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #1: *"Invalid ZIP Code for Sender"*

```
<?xml version="1.0"?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number>-2147219453</Number>
            <Source>POL_RATE2001:clsRateEngine.CalcDomesticRate;SOLServer.Cal
            lRateDll</Source>
            <Description>No Origin City State found, Message Codes: -11001 ,
            0 , -11006 , 0</Description>
            <HelpFile></HelpFile>
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #2: *"Invalid ZIP Code for Receiver"*

```
<?xml version="1.0"?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number>2147219452</Number>
            <Source>POL_RATE2001:clsRateEngine.CalcDomesticRate;SOLServer.Cal
            lRateDll</Source>
            <Description>No Destination City State found, Message Codes: -
            11002 , 0 , -11006 , 0</Description>
            <HelpFile></HelpFile>
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #3: *"Invalid Package Size for Parcel Post"*

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number>-2147219440</Number>
            <Source>POL_RATE2001:clsRateEngine.CalcDomesticRate;SOLServer.Cal
            lRateDll</Source>
            <Description>Parcel Post package size must be 'Regular', 'Large',
            or 'Oversize'.</Description>
            <HelpFile />
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #4: *"Invalid Container for Priority Mail"*

```
<?xml version="1.0"?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number>2147219493</Number>
            <Source>POL_RATE2001:clsRateEngine.CalcDomesticRate;SOLServer.Cal
            lRateDll</Source>
            <Description>Please select a shipping package suitable for
            Priority Mail service.  </Description>
            <HelpFile></HelpFile>
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #5: *"Invalid Machinable Input for Parcel Post"*

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number>-2147219487</Number>
            <Source>POL_RATE2001:clsRateEngine.CalcDomesticRate;SOLServer.Cal
            lRateDll</Source>
            <Description>This is not a valid machinable value.</Description>
            <HelpFile />
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #6: *"Invalid Weight for Express Mail and Priority Mail"*

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number>-2147219500</Number>
            <Source>POL_RATE2001:clsRateEngine.bValidateWeight:clsRateEngine.
            CalcDomesticRate;SOLServer.CallRateDll</Source>
            <Description>Please enter a valid weight.</Description>
            <HelpFile/>
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #7: *"Invalid Weight for Parcel Post"*

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
            <Error>
            <Number -2147219497</Number>
            <Source>
            POL_RATE2001:clsRateEngine.bValidateWeight:clsRateEngine.CalcDome
            sticRate;SOLServer.CallRateDll</Source>
            <Description> Please enter the package weight. </Description>
            <HelpFile/>
            <HelpContext>1000440</HelpContext>
            </Error>
      </Package>
</RateResponse>
```

Response to Pre-defined Error Request #8: *"Invalid Weight for Pounds"*

```
<?xml version="1.0" ?>
<RateResponse>
      <Package ID="0">
        <Error>
        <Number>-2147219499</Number>
        <Source>
        POL_RATE2001:clsRateEngine.bValidateWeight:clsRateEngine.CalcDomesticR
        ate;SOLServer.CallRateDll</Source>
        <Description>Warning - The package weight cannot exceed 70 pounds.
        </Description>
        <HelpFile/>
        <HelpContext>1000440</HelpContext>
        </Error>
      </Package>
</RateResponse>
```

“Live” Responses

XML Output Example

```
<?xml version="1.0"?>
<RateResponse>
      <PackageID="0">
            <Service>EXPRESS</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>54324</ZipDestination>
            <Pounds>2</Pounds>
            <Ounces>0</Ounces>
            <Container>NONE</Container>
            <Size>REGULAR</Size>
            <Zone>5</Zone>
            <Postage>16.00</Postage>
      </Package>
      <Package ID="1">
            <Service>PRIORITY</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>02912</ZipDestination>
            <Pounds>20</Pounds>
            <Ounces>8</Ounces>
            <Container>NONE</Container>
            <Size>REGULAR</Size>
            <Zone>4</Zone>
            <Postage>16.35</Postage>
      </Package>
      <Package ID="2">
            <Service>PARCEL</Service>
            <ZipOrigination>20770</ZipOrigination>
            <ZipDestination>02912</ZipDestination>
            <Pounds>20</Pounds>
            <Ounces>8</Ounces>
            <Container>NONE</Container>
            <Size>REGULAR</Size>
            <Machinable>TRUE</Machinable>
            <Zone>4</Zone>
            <Postage>9.93</Postage>
      </Package>
</RateResponse>
```