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

A Technical Guide to

Priority Mail® Service Standards

Application Programming Interface

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

Version 2.2 (1/6/01)

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

Registered trademarks of the U.S. Postal Service	Trademarks of the U.S. Postal Service
Express Mail	Delivery Confirmation
First-Class Mail	Global Express Guaranteed
Global Priority Mail	Global Express Mail
Priority Mail	GXG
ZIP Code	International Parcel Post
	Parcel Post
	Priority Mail Global Guaranteed

Microsoft, Visual Basic, and Word are registered trademarks of Microsoft Corporation.

Adobe Acrobat is a trademark of Adobe Systems Incorporated.

Table of Contents

Introduction to the Priority Mail® Service Standards API

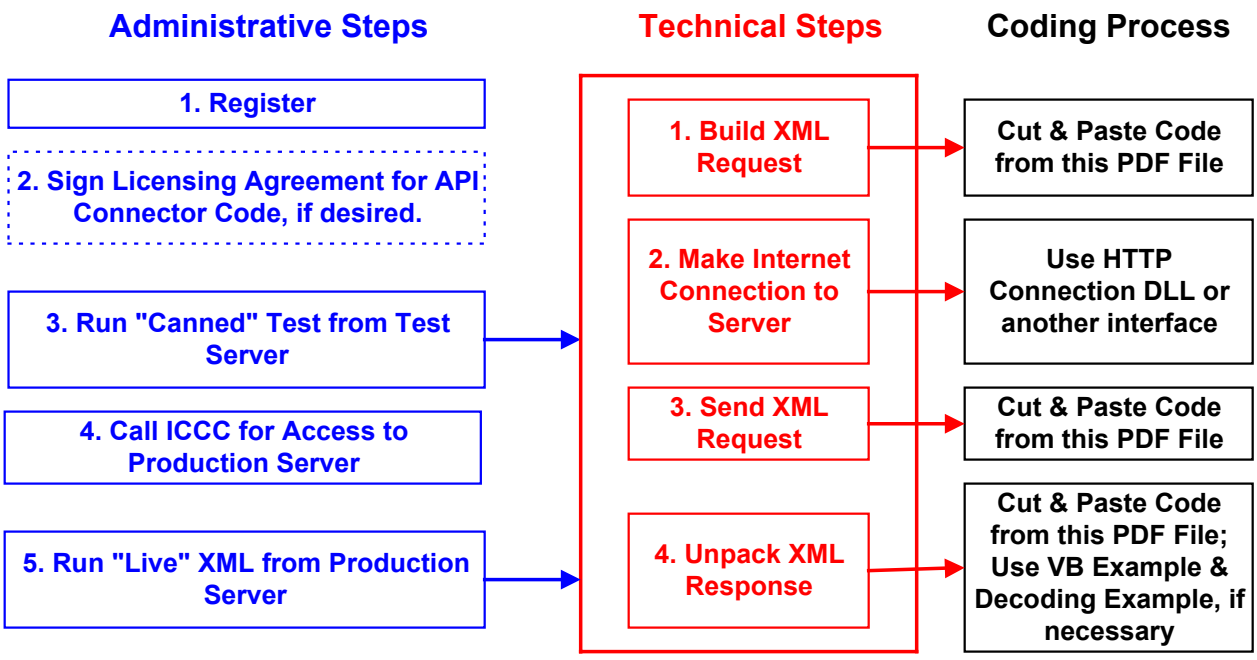
This API can be used as either an Internet or Intranet application to help answer questions about projected Priority Mail® delivery times between ZIP Codes®.

The Priority Mail® Service Standards API receives requests and returns the number of days (on average) it will take a Priority Mail® package to arrive at its destination. ***Ensure that end-users understand that these are service standards and not guaranteed commitments.*** The Priority Mail® Service Standards API processes a single request.

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 Priority Mail® Service Standards 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.



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

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.

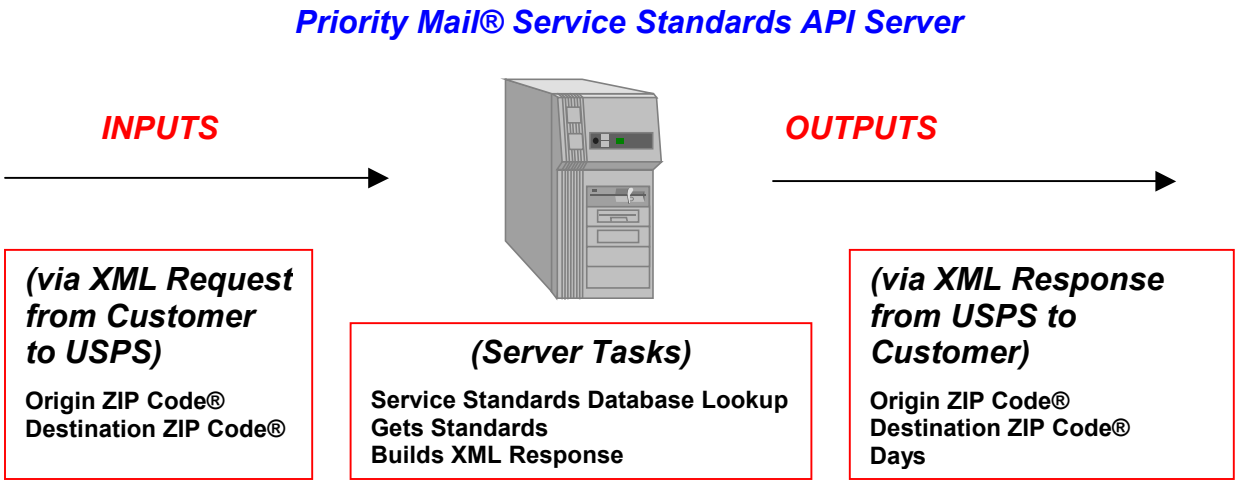
The documentation and sample code contained in the *Web Tool Kit User Guide* series may be reused and/or distributed to your customers or affiliates to generate awareness, encourage web tool use, or provide ease-of-use. However, it is your responsibility to ensure that your customers do not use your 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.

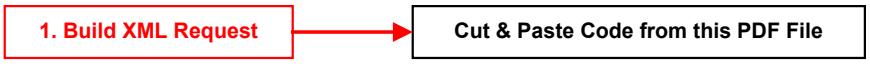
Installation

The illustration below shows the transactional flow of information to and from the USPS Priority Mail® Service Standards API server.



Technical Steps

Step 1: Build the XML Request



"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

or

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 ten valid requests included in this procedure:

Valid Request #1

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>4</OriginZip>
<DestinationZip>4</DestinationZip></PriorityMailRequest>
```

Valid Request #2

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>4</OriginZip>
<DestinationZip>5</DestinationZip></PriorityMailRequest>
```

Valid Request #3

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>4</OriginZip>
<DestinationZip>6</DestinationZip></PriorityMailRequest>
```

Valid Request #4

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>6450</OriginZip>
<DestinationZip>6371</DestinationZip></PriorityMailRequest>
```

Valid Request #5

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>90210</OriginZip>
<DestinationZip>6371</DestinationZip></PriorityMailRequest>
```

Valid Request #6

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>9021</OriginZip>
<DestinationZip>43671</DestinationZip></PriorityMailRequest>
```

Valid Request #7

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>50000</OriginZip>
<DestinationZip>90210</DestinationZip></PriorityMailRequest>
```

Valid Request #8

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>500</OriginZip>
<DestinationZip>90210</DestinationZip></PriorityMailRequest>
```

Valid Request #9

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>50000</OriginZip>
<DestinationZip>902</DestinationZip></PriorityMailRequest>
```

Valid Request #10

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>1</OriginZip>
<DestinationZip>2</DestinationZip></PriorityMailRequest>
```

Pre-Defined Error Requests

There are two 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 Origin ZIP Code®"*

The pre-defined error in this request is using input less than 1 or greater than 99999 for `<OriginZip>`.

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>500000</OriginZip>
<DestinationZip>902</DestinationZip></PriorityMailRequest>
```

Pre-defined Error Request #2: *"Invalid Destination ZIP Code®"*

The pre-defined error in this request is using input less than 1 or greater than 99999 for `<DestinationZip>`.

```
http://SERVERNAME/ShippingAPITest.dll?API=PriorityMail&XML=<PriorityMailReque
st" "USERID="xxxxxxx"" "PASSWORD="xxxxxxx"><OriginZip>50000</OriginZip>
<DestinationZip>9022020</DestinationZip></PriorityMailRequest>
```

"Live" Request

Refer to the *"Canned" Test Requests* section above for instructions on how to cut and paste the sample code from this PDF file.

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.

Developers: For sample code utilizing Perl and ASP, refer to the *Domestic Rates Calculator API* and *Track/Confirm API* user's guides.

Input	XML Tag	Values Allowed
Type of Request	`<PriorityMailRequest…`	Input tag exactly as presented.
Username	`…USERID="userid"…`	Use user ID provided with registration.
Password	`…PASSWORD="password">`	Use password provided with registration.
Origination ZIP Code®	`<OriginZip>`	Origination and destination ZIP Codes® must be valid. Only the first 3 digits of the Zip Code® are entered between the open tag and close tag. If a 1- or 2-digit ZIP Code® is entered, it will be treated the same as a 3-digit zip prefixed with 2 or 1 zeros, respectively. If a 4- or 5-digit ZIP Code® is entered, the last 1 or 2 digits will be ignored.

Destination ZIP Code®	<DestinationZip>	Origination and destination ZIP Codes® must be valid. Only the first 3 digits of the Zip Code® are entered between the open tag and close tag. If a 1- or 2-digit ZIP Code® is entered, it will be treated the same as a 3-digit zip prefixed with 2 or 1 zeros, respectively. If a 4- or 5-digit ZIP Code® is entered, the last 1 or 2 digits will be ignored.

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

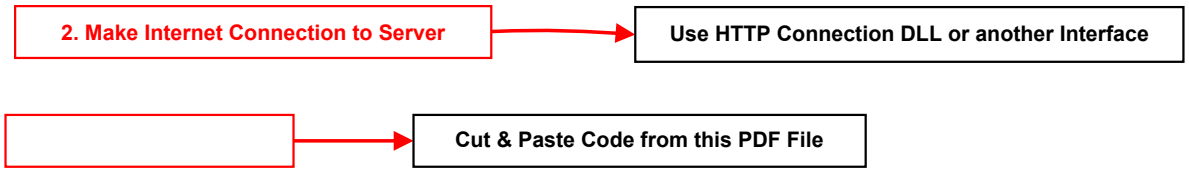
```
<PriorityMailRequest USERID="xxxxxxxx" PASSWORD="xxxxxxxx">
      <OriginZip>902</OriginZip>
      <DestinationZip>211</DestinationZip>
</PriorityMailRequest>
```

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 oXMLDocument As DOMDocument
    Dim oRequestLevel As IXMLDOMElement
    Dim oCommitmentElementLevel As IXMLDOMElement

    ' Build the XML Request

    ' Create a new XML document
    Set oXMLDocument = New DOMDocument

    ' Build the top-level (request)
    Set oRequestLevel = oXMLDocument.createElement
    ("PriorityMailRequest")
    oRequestLevel.setAttribute "USERID", "xxxxxxxx"
    oRequestLevel.setAttribute "PASSWORD", "xxxxxxxx"

    ' Origin Zip Code
    Set oCommitmentElementLevel = oXMLDocument.createElement
    ("OriginZip")
    oCommitmentElementLevel.appendChild
    oXMLDocument.createTextNode("90210")
    oRequestLevelappendChild oCommitmentElementLevel

    ' Destination Zip Code
    Set oCommitmentElementLevel =
    oXMLDocument.createElement("DestinationZip")
    oCommitmentElementLevel.appendChild
    oXMLDocument.createTextNode("20770")
    oRequestLevel.appendChild oCommitmentElementLevel<BR>

    ' Append request level to document
    oXMLDocument.appendChild oRequestLevel
```

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



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 two 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 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:

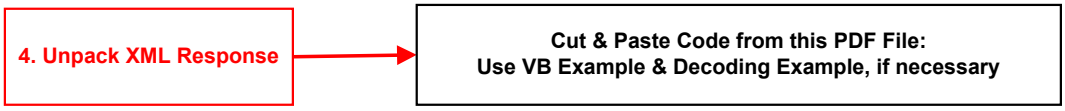
```
File = "/ShippingAPI.dll?"
```

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=PriorityMail&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)
```

Step 4: Unpack the XML Response



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 DOMDocument
Dim nodeList As IXMLDOMNodeList
Dim n As IXMLDOMNode, e As IXMLDOMNode
Dim j As Long

Dim lDays As Long

Dim lErrorNumber As Long
Dim sDescription As String
Dim sSource As String
Dim sHelpFile As String
Dim sHelpContextId As String

Set xmlDoc = New DOMDocument
xmlDoc.validateOnParse = False
xmlDoc.loadXML (sXML_RESPONSE) 'Response
If xmlDoc.documentElement.nodeName = "Error" Then
'Top-level Error
   Set nodeList = xmlDoc.getElementsByTagName("Error")
   Call UnpackErrorNode(nodeList.Item(0), lErrorNumber,
   sDescription, sSource, sHelpFile, sHelpContextId)
   ' Add code here to display the error
Else  'no Top-level Error
```

```
      Set nodeList = xmlDoc.getElementsByTagName
      ("PriorityMailResponse")
      Set n = nodeList.Item(0)
      For j = 0 To n.childNodes.length - 1
         Set e = n.childNodes.Item(j)
            Select Case e.nodeName
               Case "Days"
                  If e.hasChildNodes Then
                     lDays = e.firstChild.nodeValue
                  End If
            End Select
      Next j
   End If
```

The UnpackErrorNode common subroutine that is referred to in the above code examples unpacks an XML Error node into individual variables.

```
' Input:
'    oNode - XML Error Node
' Output:
'    lErrorNumber - Error Number
'    sDescription - Error Description
'    sSource - Error Source
'    sHelpFile - Help File Name
'    sHelpContextId - Help Context Id

Private Sub UnpackErrorNode(ByRef oNode As IXMLDOMNode, ByRef lErrorNumber As
Long, ByRef sDescription As String, ByRef sSource As String, ByRef sHelpFile
As String, ByRef sHelpContextId As String)

   Dim oNodeError As IXMLDOMNode

   Dim lIndex As Long

   lErrorNumber = 0
   sSource = ""
   sDescription = ""
   sHelpFile = ""
   sHelpContextId = ""

   For lIndex = 0 To oNode.childNodes.length - 1
      Set oNodeError = oNode.childNodes.Item(lIndex)
      Select Case oNodeError.nodeName
         Case "Source"
            If oNodeError.hasChildNodes Then
               sSource = oNodeError.firstChild.nodeValue
            End If
         Case "Number"
            If oNodeError.hasChildNodes Then
               lErrorNumber = oNodeError.firstChild.nodeValue
            End If
         Case "Description"
            If oNodeError.hasChildNodes Then
               sDescription = oNodeError.firstChild.nodeValue
            End If
         Case "HelpFile"
            If oNodeError.hasChildNodes Then
```

```
               sHelpFile = oNodeError.firstChild.nodeValue
            End If
         Case "HelpContext"
            If oNodeError.hasChildNodes Then
               sHelpContextId =
               oNodeError.firstChild.nodeValue
            End If
      End Select
   Next
End Sub
```

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.

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
Response Type	`<PriorityMailResponse>`	
Origination ZIP Code®	`<OriginZip>`	Only the first 3 digits of the ZIP Code® are returned.

Destination ZIP Code®	<DestinationZip>	Only the first 3 digits of the ZIP Code® are returned.
Average number of days it will take the package to arrive	<Days>	

"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"?>
<Error><Number>-2147219040</Number>
      <Source>SOLServerTest;SOLServerTest.PriorityMail_Respond</Source>
      <Description>This Information has not been included in this Test
      Server.</Description>
      <HelpFile></HelpFile>
      <HelpContext></HelpContext>
</Error>
```

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"?>
<PriorityMailResponse>
      <OriginZip>4</OriginZip>
      <DestinationZip>4</DestinationZip>
      <Days>1</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #2

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>4</OriginZip>
      <DestinationZip>5</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #3

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>4</OriginZip>
      <DestinationZip>6</DestinationZip>
      <Days>3</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #4

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>645</OriginZip>
      <DestinationZip>637</DestinationZip>
      <Days>2</Days>
```

```
</PriorityMailResponse>
```

Response to Valid Test Request #5

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>902</OriginZip>
      <DestinationZip>637</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #6

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>902</OriginZip>
      <DestinationZip>436</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #7

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>500</OriginZip>
      <DestinationZip>902</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #8

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>500</OriginZip>
      <DestinationZip>902</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #9

```
<?xml version="1.0"?>
<PriorityMailResponse>
      <OriginZip>500</OriginZip>
      <DestinationZip>902</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```

Response to Valid Test Request #10

```
<PriorityMailResponse>
      <OriginZip>1</OriginZip>
      <DestinationZip>2</DestinationZip>
      <Days>No Data</Days>
</PriorityMailResponse>
```

Response to Pre-defined Error Request #1: *"Invalid Origin ZIP Code®"*

```
<?xml version="1.0"?>
<Error>
      <Number>-2147219200</Number>
      <Source>SOLServerTest;SOLServerTest.PriorityMail_Respond</Source>
      <Description>You must use a 3 digit zip code for Origin Zip.  You sent:
      500000</Description>
      <HelpFile></HelpFile>
      <HelpContext>1000440</HelpContext>
</Error>
```

Response to Pre-defined Error Request #2: *"Invalid Destination ZIP Code®"*

```
<?xml version="1.0"?>
<Error>
      <Number>-2147219200</Number>
      <Source>SOLServerTest;SOLServerTest.PriorityMail_Respond</Source>
      <Description>You must use a 3 digit zip code for Destination Zip.  You
      sent: 9022020</Description>
      <HelpFile></HelpFile>
      <HelpContext>1000440</HelpContext>
</Error>
```

"Live" Responses

XML Output Example

```
<PriorityMailResponse>
      <OriginZip>902</OriginZip>
      <DestinationZip>211</DestinationZip>
      <Days>2</Days>
</PriorityMailResponse>
```