



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



A Technical Guide to

Electronic Merchandise Return Service

&

***Electronic Merchandise Return Service with
Delivery Confirmation***

Application Programming Interfaces



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

Version 4.5 (02/27/03)

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: icustomer@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

Express Mail, First-Class Mail, Global Priority Mail, Parcel Post, Parcel Select, Priority Mail, USPS, and ZIP + 4 are registered trademarks of the U.S. Postal Service.

Delivery Confirmation, Global Express Guaranteed, Global Express Mail, GXG, International Parcel Post, Priority Mail Global Guaranteed, Signature Confirmation, and ZIP Code are trademarks of the U.S. Postal Service.

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

Adobe Acrobat is a trademark of Adobe Systems Incorporated.

©Copyright 2003 United States Postal Service

Table of Contents

Introduction to the Electronic Merchandise Return Service APIs.....	1
Optional Delivery Confirmation	3
Optional Special Services	5
Insurance	5
Mailing Acknowledgment	5
Return Materials Authorization (RMA) Number.....	6
RMA Barcode.....	6
Example with Different Options	6
User ID and Password Restrictions	7
Transaction Procedures	9
Technical Steps	9
Step 1: Build the XML Request	9
"Canned" Test Requests	9
"Canned" Test Requests (EMR with Delivery Confirmation only)	12
XML Tags and Values Allowed (Required & Optional)	15
"Sample" Request (EMR with Delivery Confirmation only).....	18
"Live" Request.....	19
Step 2 & 3: Make the Internet Connection and Send the XML Request.....	22
Using HTTP Connection DLL	23
Using WinInet.....	23
Step 4: Unpack the XML Response	24
Types of Responses	24
Using Visual Basic	25
Base64 Decoding in VBScript in an ASP	28
Errors	29
Output	29
"Canned" Test Responses	30
"Canned" Test Responses (EMR with Delivery Confirmation only).....	31
"Sample" Responses (EMR with Delivery Confirmation only)	33
"Live" Responses	35
Generating Your Own Label.....	35

Introduction to the Electronic Merchandise Return Service APIs



The Electronic Merchandise Return (EMR) APIs have been specifically designed for e-tailer web sites. These Internet applications facilitate returns by allowing customers to download and print a Merchandise Return label from your web site.

When a customer indicates through your web site that they would like to return an item, the API will return a completed Electronic Merchandise Return label that you can e-mail to your customer.

There are two options for EMR service: with or without Delivery Confirmation. EMR with Delivery Confirmation provides the convenience of accessing information on the Internet about the delivery status of EMR packages, including the date, time, and ZIP Code of delivery, as well as attempted deliveries. The label returned by the API is printed by the sender and attached to the package. EMR with Delivery Confirmation labels should be used the same day that they are released to the requestor. If you choose to implement the EMR with Delivery Confirmation API, you must test against our test server and have your barcode label printer certified by the USPS before use. Refer to Administrative Step 5 in the *Administrative Guide for APIs*.

The sample labels below illustrate the difference between EMR with (on the left) and without (on the right) Delivery Confirmation:

ID# 01272000

FROM: LINDA E. SHOPPER
100 MAIN ST.
LOS ANGELES, CA 90052

POSTAGE DUE COMPUTED BY ACCEPTANCE POST OFFICE

POSTAGE _____

DELIVERY CONFIRMATION FEE _____

INSURANCE FEE \$ 6.00

TOTAL POSTAGE AND FEES DUE \$ _____

INSURANCE DESIRED BY PERMIT HOLDER FOR \$ 500.00 (VALUE)

USPS DELIVERY CONFIRMATION

10054321

MERCHANDISE RETURN LABEL

PERMIT NO 987654321 NEW YORK NY 10128
XYZ CORPORATION 1234 ETAILER DR.

POSTAGE DUE UNIT
US POSTAL SERVICE
PO BOX 9998
NEW YORK NY 10128-1224

8580 5213 9079 2659 1855

RMA #: _____

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

PRIORITY MAIL

ID# 01272000

FROM: LINDA E. SHOPPER
100 MAIN ST., APT. 3C
NEW YORK, NY 10010

POSTAGE DUE COMPUTED BY ACCEPTANCE POST OFFICE

POSTAGE _____

INSURANCE FEE _____

TOTAL POSTAGE AND FEES DUE \$ 6.00

INSURANCE DESIRED BY PERMIT HOLDER FOR \$ 500.00 (VALUE)

10054321

MERCHANDISE RETURN LABEL

PERMIT NO 987654321 NEW YORK NY 10128
XYZ CORPORATION 1234 ETAILER DR.

POSTAGE DUE UNIT
US POSTAL SERVICE
PO BOX 9998
NEW YORK NY 10128-1224

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

PRIORITY MAIL

(The images above are not to scale.)

Note: The EMR with Delivery Confirmation API does not perform address validation. It is assumed that the address is valid.

Throughout this guide, sections with information specific to EMR with Delivery Confirmation have been labeled. If you do not use Delivery Confirmation, you do not need to follow the instructions in these sections.

Optional Courtesy Reply Mail Label. For e-tailers that elect to have their customers pay the postage on a return item, but still wish to provide a convenient return label, a Courtesy Label for Merchandise Return API is available. This API provides a label with the customer's address, the retailer's address and, if desired, inventory or shipping information, such as a Return Materials Authorization number. The guide for this API can be found at <http://www.uspswebtools.com/>.

EMR service is available to Merchandise Return permit holders for mailing to the postage due unit at any post office where authorized by an approved application. Refer to the *Administrative Guide for APIs* for instructions on obtaining a permit.

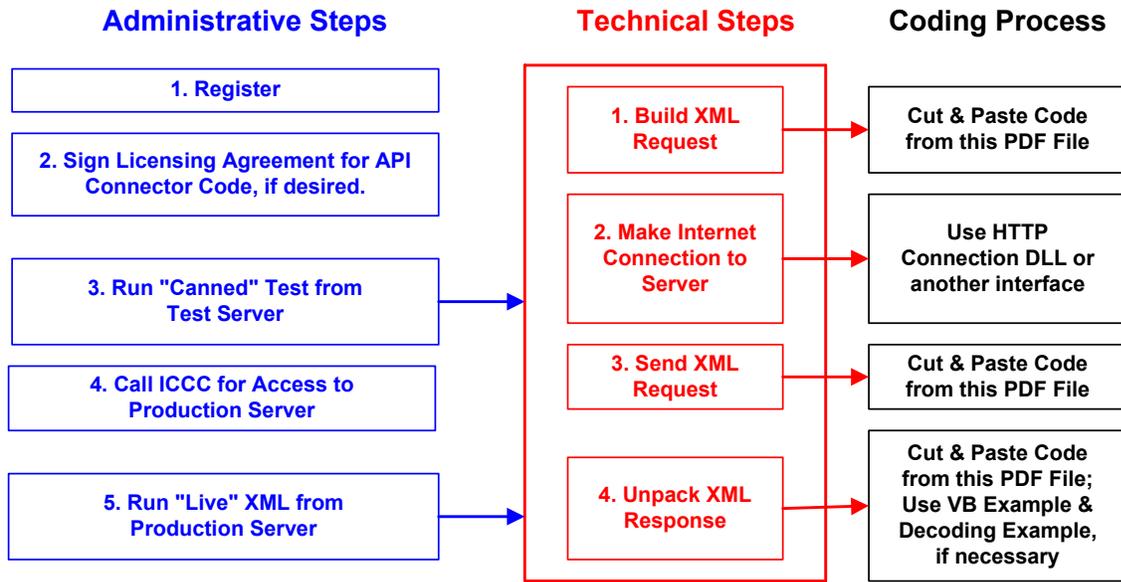
The standards for payment of postage and fees are:

- The permit holder guarantees payment of the proper postage and fees on all returned merchandise return service articles distributed under the permit holder's permit number. Charges are collected for each article as postage due at the time of delivery or from a centralized advance deposit account using Form 3582-C, Postage Due Invoice.
- When determining rates, postage for all Package Services will be based on the proper single-piece rate.
- If a request is made for a First-Class Mail label for a package that weighs 13 ounces or more, the label will be returned marked as "Priority Mail."

For more information about eligibility for all Package Services, including content and weight restrictions, refer to the Domestic Mail Manual (DMM), Section E630, located at the Postal Explorer web site <http://pe.usps.gov>. Also see DMM section S923, Merchandise Return Service.

As shown on the next page, implementing USPS Shipping APIs requires a series of *Administrative Steps*. The *Administrative Guide for APIs*, also available at <http://www.uspswebtools.com/>, provides necessary information and procedures prior to installation.

The illustration also shows the *Technical Steps* required to run XML transactions for either of the EMR APIs 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.



Implementing these APIs requires experienced programmers who are familiar with Internet and web site development tools and techniques. Before implementing either of these APIs, the Administrative Guide for Application Programming Interfaces must be read.

Optional Delivery Confirmation

EMR with Delivery Confirmation service may *only* be used for First-Class Mail parcels, Priority Mail and Package Services (Parcel Post, Media Mail, Bound Printed Matter, and Library Mail). EMR labels with Delivery Confirmation barcodes are charged the appropriate retail fees and do not qualify for the electronic option rate.

As shown below, EMR with Delivery Confirmation labels contain a barcode and corresponding 20-digit Delivery Confirmation number known as the Package Identification Code (PIC):



(The image above is not to scale.)

The components of the PIC number are as follows:

Digits	Description	Comments
1-2	Service Type Code	Valid Service Type Codes are: 82 for Priority Mail, First-Class Mail parcels, Package Services, and Standard Mail Parcels without insurance

		83 for Priority Mail, Package Services, and Standard Mail Parcels with insured value ≤ \$50 85 for Priority Mail, Package Services, and Standard Mail Parcels with insured value > \$50
3-11	API Program number	The API Program number is 805213907 (This number is used on every Delivery Confirmation label).
12-19	Package ID #	The Package ID # is an 8-digit number generated by the Delivery Confirmation API.
20	MOD 10 Check Digit #	The Check Digit is computed from the sequence number. For details on this check digit see Appendix G in the <i>Delivery Confirmation Technical Guide</i> , Publication 91, at http://www.usps.com/cpim/ftp/pubs/pub91/welcome.htm .

The PIC is used when requesting delivery status with Priority Mail or any of the Package Services. A daily manifest of all PICs issued is loaded by the USPS into the Web Tool database server. Customer inquiries made the following day through the Track/Confirm API on a package's delivery status will receive a response that states "An Electronic Merchandise Return Service label with Delivery Confirmation was created and transmitted" with date of occurrence. This response is **only** available through the Track/Confirm API. See the Track/Confirm API technical guide. All barcode scans made at the delivery point and in route to delivery will be recorded by the USPS PTS system. Delivery Confirmation information will be available through the <http://www.usps.com/shipping/trackandconfirm.htm> track/confirm site or the Track/Confirm API mentioned above.

Users also have the option of generating their own label. The Delivery Confirmation number that must be used is returned in the XML response. See the *Generating Your Own Label* section for restrictions on generating your own Delivery Confirmation label.

Whichever label option you choose, you must test against our test server and have your barcode label printer certified by the USPS before use. Refer to *Administrative Steps 4* and *7* in the *Administrative Guide for APIs*.

Optional Special Services

Along with Delivery Confirmation, the table below lists the special services available with the EMR APIs. The table shows six supported Mail Endorsements (First-Class Mail, Priority Mail, Parcel Post, Library Mail, Bound Printed Matter, and Media Mail) and five special services (Registered Mail, Insurance, Mailing Acknowledgment, Delivery Confirmation, and Return Materials Authorization). The description of Delivery Confirmation appears above. The descriptions of the other special services appear in the following table.

Merchandise Return Class of Mail Endorsement	Registered Mail	Insurance	Mailing Acknowledgement	Delivery Confirmation	Return Materials Authorization
First Class Mail	not available	not available	•	•	•
Priority Mail	not available	•	•	•	•
Parcel Post	not available	•	•	•	•
Library Mail	not available	•	•	•	•
Bound Printed Matter	not available	•	•	•	•
Media Mail	not available	•	•	•	•

Insurance

Insurance for packages that contain merchandise is available with Priority Mail and all four Package Services. Payment for insurance is collected at the Postage Due Unit along with postage. However, the customer receiving the EMR label must take the return package to a post office. The USPS retail clerk will apply an insurance stamp or affix the appropriate numbered insurance label to the EMR at the very top and directly to the left of the “No Postage Necessary if Mailed in the United States” box. Customers should be instructed *not* to apply tape over this area.

If insurance is desired, the e-tailer must supply an insurance dollar value in the XML request. This dollar value will be placed on the label along with the line “INSURANCE DESIRED BY PERMIT HOLDER FOR \$ ____”. The fee will also be imprinted next to a line that reads “INSURANCE FEE.” See the *Example with Different Options* section.

Mailing Acknowledgment

This service allows your customers to obtain a Mailing Acknowledgement form from the USPS at the time of mailing (see the *Example with Different Options* section.). This service provides documentation between the e-tailer and the mailing customer. The USPS charges no fee for this service nor does it maintain any records or provide further information about the

acknowledgement upon request. Mailing Acknowledgement service is available with First-Class Mail, Priority Mail, and all four Package Services.

When this service is requested, the EMR label returned to the e-tailer will include a detachable Mailing Acknowledgment form. The form and package must be presented to a USPS retail clerk. The clerk will initial the detachable form and place an official USPS date stamp on the form. Both the EMR label and the Mailing Acknowledgment will contain a unique parcel identification number that will be supplied by the e-tailer in the XML. The number could be an invoice number or anything the e-tailer chooses to make it. For more information on this service, refer to the DMM, Section S923, 4.12, located at the Postal Explorer web site <http://pe.usps.gov>.

Return Materials Authorization (RMA) Number

Assigning an RMA number to each package at the time an agreement is made with a customer to have merchandise returned enables retailers to provide a closed-loop correlation between receipt of returned packages and records of the order, typically kept in the order-processing database. For e-tailers that utilize Return Materials Authorization numbers in their order processing database system, the EMR API will accept an RMA in any combination of numeric and alpha characters and display the RMA on the EMR label. For example:

```
<RMA>RMA #123456789</RMA>
```

RMA Barcode

The RMA number entered by the user with the <RMA> tag can also appear as a US128C barcode on the label. If you wish to see the number displayed in a barcode using the <RMABarcode> tag, do not enter anything except the alpha/numeric number itself. As shown in *Example with Different Options* section, below, the API automatically enters “RMA #” on the label.

Example with Different Options

The image on the following page shows an EMR label that has all of the different options in place. The circled numbers correspond to notes below the graphic explaining the source of the item. Depending on the values entered in your request, the image returned to you will have any combination of these options. Also shown on the illustration is an example of the label mailing instructions that are provided with the label for your customer to follow. Do not remove them from the label.

FROM: LINDA E. SHOPPER
100 MAIN ST., APT. 3C
NEW YORK, NY 10010

POSTAGE DUE COMPUTED BY ACCEPTANCE POST OFFICE

POSTAGE
DELIVERY CONFIRMATION FEE
INSURANCE FEE
TOTAL POSTAGE AND FEES DUE \$ 1.10

INSURANCE DESIRED BY PERMIT HOLDER FOR \$ 50.00 (VALUE)

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

PRIORITY MAIL

USPS DELIVERY CONFIRMATION

1077309AE

MERCHANDISE RETURN LABEL

PERMIT NO. 307 OAKLAND CA 90011
XYZ CORPORATION 1234 ETAILER DR.

POSTAGE DUE UNIT
US POSTAL SERVICE
PO BOX 9998
LOS ANGELES CA 90052-1234

8580 5213 9079 6853 4902

RMA #

Customer Mailing Instructions

- Affix the label squarely onto the address side of the parcel, covering up any previous delivery address and barcode. If tape or similar material is used, it must not cover any part of the label where postage and fee information is to be recorded or where a Delivery Confirmation Barcode exists.
- Obliterate any other addresses and barcodes on the outside of the parcel.
- Take the parcel to a post office so that it may be handled by a USPS retail associate.

FROM: LINDA E. SHOPPER
100 MAIN ST., APT. 3C
NEW YORK, NY 10010

I.D. # 0000001

ROUND DATE
STAMP

ACCEPTANCE EMPLOYEE
INITIALS

MERCHANDISE RETURN MAILING ACKNOWLEDGMENT

PERMIT NO. 307 OAKLAND CA 90011
XYZ CORPORATION 1234 ETAILER DR.

(The image above is not to scale.)

- Mailing Acknowledgement Package Identifier number (as specified in the XML for <MailingAckPackageID>)
- Class of Service Marking (the value specified in the XML for <ServiceType>)
- Insurance Fee (computed cost of insurance based on the value specified in the XML for <InsuranceValue>)
- Value of package being insured (dollar amount of insurance for the article; the value specified in the XML for <InsuranceValue>)
- Return Merchandise Authorization (as specified in the XML for <RMA>) and the corresponding barcode (if requested in the XML for <RMABarcode>)
- Delivery Confirmation barcode (if requested in the XML for <DeliveryConfirmation>)

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 <http://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 Policy Guide* available at <http://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 icustomer@usps.com.

Transaction Procedures

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

Electronic Merchandise Return Service API Server



INPUTS

(via XML Request from Customer to USPS)

Customer Name & Address
 Retailer Name & Address
 Service Type
 Permit Information
 PDU Information
 Label Image Type
 Delivery Confirmation (optional)
 Insurance Value
 Package ID # (optional)
 Package Weight
 Return Materials Authorization (optional)
 E-mail Notification (optional)
 RMA Barcode (optional)

SERVER TASKS

Builds XML Response
 Generates Label
 Encodes Label

 Tracks Package *(with Delivery Confirmation option)*

OUTPUTS

(via XML Response from USPS to Customer)

Label
 Zone
 Insurance Cost

 Delivery Confirmation Number *(with Delivery Confirmation option)*

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

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

EMR Valid Request #1

```
http://SERVERNAME/ShippingAPITest.dll?API=MerchandiseReturnV2&XML=
<EMRSV2.0Request USERID="xxxxxxxx" PASSWORD="xxxxxxxx">
  <CustomerName>John Smith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20770</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
  <PDUPoBox>48374</PDUPoBox>
  <PDUCity>New York</PDUCity>
  <PDUState>NY</PDUState>
  <PDUZip5>10128</PDUZip5>
  <PDUZip4>0143</PDUZip4>
  <ServiceType>Priority</ServiceType>
  <DeliveryConfirmation>False</DeliveryConfirmation>
  <InsuranceValue>50</InsuranceValue>
  <MailingAckPackageID>1234</MailingAckPackageID>
  <WeightInPounds>10</WeightInPounds>
  <WeightInOunces>8</WeightInOunces>
  <RMA>RMA100</RMA>
  <ImageType>Tif</ImageType>
</EMRSV2.0Request>
```

EMR Valid Request #2 (with RMA barcode)

```
http://SERVERNAME/ShippingAPITest.dll?API=MerchandiseReturnV2&XML=
<EMRSV2.0Request USERID="xxxxxxxx" PASSWORD="xxxxxxxx">
  <CustomerName>John Smith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20770</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
  <PDUPoBox>48374</PDUPoBox>
  <PDUCity>New York</PDUCity>
  <PDUState>NY</PDUState>
  <PDUZip5>10128</PDUZip5>
  <PDUZip4>0143</PDUZip4>
```

```
<ServiceType>Parcel Post</ServiceType>
<DeliveryConfirmation>False</DeliveryConfirmation>
<InsuranceValue>50</InsuranceValue>
<MailingAckPackageID>1234</MailingAckPackageID>
<WeightInPounds>10</WeightInPounds>
<WeightInOunces>8</WeightInOunces>
<RMA>987654321</RMA>
<ImageType>PDF</ImageType>
<RMABarcode>TRUE</RMABarcode>
</EMRSV2.0Request>
```

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.

EMR Pre-defined Error Request #1: “Invalid Customer ZIP Code”

```
http://SERVERNAME/ShippingAPITest.dll?API=MerchandiseReturnV2&XML=
<EMRSV2.0Request USERID="xxxxxxxxx" PASSWORD="xxxxxxxxx">
  <CustomerName>JohnSmith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>99999</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
  <PDUPobox>48374</PDUPobox>
  <PDUcity>New York</PDUcity>
  <PDUstate>NY</PDUstate>
  <PDUzip5>10128</PDUzip5>
  <PDUzip4>0143</PDUzip4>
  <ServiceType>Priority</ServiceType>
  <DeliveryConfirmation>False</DeliveryConfirmation>
  <InsuranceValue>50</InsuranceValue>
  <MailingAckPackageID>1234</MailingAckPackageID>
  <WeightInPounds>10</WeightInPounds>
  <WeightInOunces>8</WeightInOunces>
  <RMA>RMA100</RMA>
  <ImageType>Tif</ImageType>
</EMRSV2.0Request>
```

EMR Pre-defined Error Request #2: “Invalid Image Type”

```
http://SERVERNAME/ShippingAPITest.dll?API=MerchandiseReturnV2&XML=
<EMRSV2.0Request USERID="xxxxxxxxx" PASSWORD="xxxxxxxxx">
  <CustomerName>JohnSmith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20770</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
```

```
<PermitNumber>987654321</PermitNumber>
<PermitIssuingPOCity>New York</PermitIssuingPOCity>
<PermitIssuingPOState>NY</PermitIssuingPOState>
<PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
<PDUPoBox>48374</PDUPoBox>
<PDUCity>New York</PDUCity>
<PDUState>NY</PDUState>
<PDUZip5>10128</PDUZip5>
<PDUZip4>0143</PDUZip4>
<ServiceType>Priority</ServiceType>
<DeliveryConfirmation>False</DeliveryConfirmation>
<InsuranceValue>50</InsuranceValue>
<MailingAckPackageID>1234</MailingAckPackageID>
<WeightInPounds>10</WeightInPounds>
<WeightInOunces>8</WeightInOunces>
<RMA>RMA100</RMA>
<ImageType>ABC</ImageType>
</EMRSV2.0Request>
```

“Canned” Test Requests (EMR with Delivery Confirmation only)

If you are implementing the EMR with Delivery Confirmation API, you should use these “canned” tests. The tests above can also be run, but it not necessary to do both.

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

EMR w/ Delivery Confirmation Valid Request #1

```
<EMRSV2.0Request USERID="xxxxxxx" PASSWORD="xxxxxxx">
  <CustomerName>John Smith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20770</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
```

```
<PermitIssuingPOState>NY</PermitIssuingPOState>
<PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
<PDUPoBox>48374</PDUPoBox>
<PDUCity>New York</PDUCity>
<PDUState>NY</PDUState>
<PDUZip5>10128</PDUZip5>
<PDUZip4>0143</PDUZip4>
<ServiceType>Priority</ServiceType>
<DeliveryConfirmation>True</DeliveryConfirmation>
<InsuranceValue>50</InsuranceValue>
<MailingAckPackageID>1234</MailingAckPackageID>
<WeightInPounds>10</WeightInPounds>
<WeightInOunces>8</WeightInOunces>
<RMA>RMA100</RMA>
<ImageType>Tif</ImageType>
</EMRSV2.0Request>
```

EMR w/ Delivery Confirmation Valid Request #2 (with RMA barcode)

```
<EMRSV2.0Request USERID="xxxxxxx" PASSWORD="xxxxxxx">
  <CustomerName>John Smith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20770</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
  <PDUPoBox>48374</PDUPoBox>
  <PDUCity>New York</PDUCity>
  <PDUState>NY</PDUState>
  <PDUZip5>10128</PDUZip5>
  <PDUZip4>0143</PDUZip4>
  <ServiceType>Parcel Post</ServiceType>
  <DeliveryConfirmation>True</DeliveryConfirmation>
  <InsuranceValue>50</InsuranceValue>
  <MailingAckPackageID>1234</MailingAckPackageID>
  <WeightInPounds>10</WeightInPounds>
  <WeightInOunces>8</WeightInOunces>
  <RMA>987654321</RMA>
  <ImageType>PDF</ImageType>
  <RMABarcode>TRUE</RMABarcode>
</EMRSV2.0Request>
```

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.

EMR w/ Delivery Confirmation Pre-defined Error Request #1: "Invalid Customer ZIP Code"

```
<EMRSV2.0Request USERID="xxxxxxx" PASSWORD="xxxxxxx">
  <CustomerName>John Smith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>99999</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
  <PDUPoBox>48374</PDUPoBox>
  <PDUCity>New York</PDUCity>
  <PDUState>NY</PDUState>
  <PDUZip5>10128</PDUZip5>
  <PDUZip4>0143</PDUZip4>
  <ServiceType>Priority</ServiceType>
  <DeliveryConfirmation>True</DeliveryConfirmation>
  <InsuranceValue>50</InsuranceValue>
  <MailingAckPackageID>1234</MailingAckPackageID>
  <WeightInPounds>10</WeightInPounds>
  <WeightInOunces>8</WeightInOunces>
  <RMA>RMA100</RMA>
  <ImageType>Tif</ImageType>
</EMRSV2.0Request>
```

EMR w/ Delivery Confirmation Pre-defined Error Request #2: "Invalid Image Type"

```
<EMRSV2.0Request USERID="xxxxxxx" PASSWORD="xxxxxxx">
  <CustomerName>John Smith</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20750</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1125 5th Avenue</RetailerAddress>
  <PermitNumber>987654321</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>10128</PermitIssuingPOZip5>
  <PDUPoBox>48374</PDUPoBox>
  <PDUCity>New York</PDUCity>
  <PDUState>NY</PDUState>
  <PDUZip5>10128</PDUZip5>
  <PDUZip4>0143</PDUZip4>
  <ServiceType>Priority</ServiceType>
  <DeliveryConfirmation>True</DeliveryConfirmation>
  <InsuranceValue>50</InsuranceValue>
  <MailingAckPackageID>1234</MailingAckPackageID>
  <WeightInPounds>10</WeightInPounds>
  <WeightInOunces>8</WeightInOunces>
  <RMA>RMA100</RMA>
  <ImageType>ABC</ImageType>
</EMRSV2.0Request>
```

XML Tags and Values Allowed (Required & Optional)

The table below presents the XML input tags for generating both “*Sample*” and “*Live*” requests and the restrictions on the values allowed. A second table is also included with *optional* tags for the EMR with Delivery Confirmation API. An error message will be returned 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.

Required Electronic Merchandise Return API Tags

Input	XML Tag	Values Allowed
“Sample” Request	<EMRSV2.0CertifyRequest ...	Input tag exactly as presented.
“Live” Request	<EMRSV2.0Request...	Input tag exactly as presented.
User ID	...USERID=“userid”...	Use user ID provided with registration.
Password	...PASSWORD=“password”>	Use password provided with registration.
Customer’s Name	<CustomerName>	Maximum characters allowed: 32
Customer’s Address	<CustomerAddress>	Maximum characters allowed: 32
Customer’s City	<CustomerCity>	Maximum characters allowed: 20
Customer’s State	<CustomerState>	Maximum characters allowed: 2
Customer’s ZIP Code	<CustomerZip5>	Input tag exactly as presented. Maximum characters allowed: 5
Retailer’s Name	<RetailerName>	Maximum characters allowed: 20
Retailer’s Address	<RetailerAddress>	Maximum characters allowed: 24
Post Office Permit Number	<PermitNumber>	Input permit number provided by your local post office.
City Issuing Post Office Permit	<PermitIssuingPOCity>	Maximum characters allowed: 15
State Issuing Post Office Permit	<PermitIssuingPOState>	Maximum characters allowed: 2
ZIP Code of Post Office Issuing Permit	<PermitIssuingPOZip5>	Input tag exactly as presented, not all caps. Maximum characters allowed: 5
Postage Due Unit PO Box	<PDUPoBox>	Maximum characters allowed: 24
Postage Due Unit City	<PDUCity>	Maximum characters allowed: 15
Postage Due Unit State	<PDUState>	Maximum characters allowed: 2
Postage Due Unit ZIP Code	<PDUZip5>	Input tag exactly as presented, not all caps. Maximum characters allowed: 5
Postage Due Unit ZIP Code + 4	<PDUZip4>	Input tag exactly as presented, not all caps. Maximum characters allowed: 4
Service Type Requested	<ServiceType>	Enter one of the valid entries: “Priority” (for Priority Mail) “First Class” “Parcel Post” “Bound Printed Matter” “Media Mail” “Library Mail”
Delivery Confirmation Service	<DeliveryConfirmation>	Enter “True” if using Delivery Confirmation. Enter “False” if not using Delivery Confirmation.

Insurance Desired by Permit Holder	<InsuranceValue>	Enter numeric currency with dollars and cents (no dollar sign). If insurance is not required, leave value blank. A value of "0.00" will result in an error being returned.
Unique Parcel Identification Number	<MailingAckPackageID>	<i>Value entry is optional.</i> E-tailer assignable number. Maximum characters allowed: 24. Refer to " <i>Optional Special Services,</i> " above.
Package Weight in Pounds	<WeightInPounds>	Value must be numeric. Estimated weight is allowed. The package weight is used to determine appropriate Service Type. Maximum characters allowed: 2.
Package Weight in Ounces	<WeightinOunces>	Value must be numeric. Maximum characters allowed: 4.
Return Materials Authorization	<RMA>	<i>Value entry is optional.</i> Any combination of alpha and numeric characters can be entered, up to a maximum of 30.
Label Image Type	<ImageType>	Enter one of the valid entries: " TIF" " JPG" " GIF" " PDF"

Optional Electronic Merchandise Return API Tags

The EMR API has a set of optional tags that may be sent with your request. Following a brief description of the functions of these tags is a table similar to the *Required Tags* table with values allowed.

For e-tailers that utilize Return Materials Authorization numbers in their order processing database system (<RMA>), the EMR API will produce a barcode of the RMA on the label with the <RMABarcode> tag. See the *Optional Special Services* section for a description of the RMA number and barcode, and their relationship to each other on the EMR label.

If you wish to have the USPS automatically e-mail the Electronic Merchandise Return label, there are four optional tags used for this request. If you are using EMR with Delivery Confirmation, the Delivery Confirmation number is on the label returned. Your customers can use this number to track the package via <http://www.usps.com> or on your web site with the Track/Confirm API. The following is a sample e-mail message transmitted with this feature:

```

From: eMerchandiseReturnAlert@USPSshippingapis.com
Sent: Tuesday, June 05, 2001 9:38 AM
To: <RecipientEMail>
CC: <SenderEMail>
Subject: Merchandise Return Label
    
```

Dear <RecipientName>:

Attached is your Electronic Merchandise Return Label, with mailing instructions included.

This email was automatically generated by the US Postal Service (www.usps.com) at the shipper's request. Any reply to this email will not be received by the USPS or shipper. The USPS has not collected or retained any personally identifying information about you or your purchase from this email.

Thank you,

```

<SenderName>
mailto:<SenderEMail>
    
```

The only tag **required** to use the e-mail feature is <RecipientEMail>. The other three are not required to use the e-mail feature.

- The name of the person or company sending the e-mail is entered with the <SenderName> tag. This name will appear in the text of the e-mail message.
- The address of the person or company sending the e-mail is entered with the <SenderEMail> tag. This address will appear in the text of the e-mail message. This address will be cc:'d when the e-mail is sent.
- The name of the person or company receiving the e-mail is entered with the <RecipientName> tag. This name will appear in the TO: field of the e-mail message as well as in the text of the message.
- The address of the person or company receiving the e-mail is entered with the <RecipientEMail> tag. This e-mail address will appear in the TO: field of the e-mail message. Although this field is considered optional, if e-mail is desired it is the only required field. Without this field, the e-mail will not be sent.

Input	XML Tag	Values Allowed
Name of E-mail Sender	<SenderName>	Name of e-mail sender.
E-mail Address of Sender	<SenderEMail>	Valid e-mail addresses must be used.
Name of E-mail Recipient	<RecipientName>	Name of e-mail recipient.
E-mail Address of Recipient	<RecipientEMail>	Valid e-mail addresses must be used.
Barcode of the User-assigned Number for Internal Use	<RMABarcode>	Enter "True" for the barcode to appear on the label. Enter "False" if no barcode is desired. If no value is entered, the API will assume "False."



Developers: For sample code utilizing Perl and ASP, refer to the Domestic Rates Calculator or Track/Confirm API technical guides.

“Sample” Request (EMR with Delivery Confirmation only)

When you are initially granted access to the production server, the only output you are able to receive is a “Sample” label. See the “Sample” Responses section for an example. When the ICCC has notified you of your printer approval, full production access is immediately granted. **Remember to start using valid data for the sample labels.** 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



or



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 below is:

```
<RecipientName>Joe Smith</RecipientName>
```

In this instance, you will replace “Joe Smith” with the name of the person receiving the package when making your request.

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

The “Sample” XML request should be in the form:

```
<EMRSV2.0CertifyRequest USERID="xxxxxxx" PASSWORD="xxxxxxx">
  <CustomerName>James Ingle</CustomerName>
  <CustomerAddress>6406 Ivy Lane</CustomerAddress>
  <CustomerCity>Greenbelt</CustomerCity>
  <CustomerState>MD</CustomerState>
  <CustomerZip5>20770</CustomerZip5>
  <RetailerName>XYZ Corp.</RetailerName>
  <RetailerAddress>1100 West Avenue</RetailerAddress>
  <PermitNumber>293829</PermitNumber>
  <PermitIssuingPOCity>New York</PermitIssuingPOCity>
  <PermitIssuingPOState>NY</PermitIssuingPOState>
  <PermitIssuingPOZip5>12345</PermitIssuingPOZip5>
  <PDUPobox>89899</PDUPobox>
  <PDUcity>New York</PDUcity>
  <PDUstate>NY</PDUstate>
  <PDUzip5>15395</PDUzip5>
  <PDUzip4>3939</PDUzip4>
  <ServiceType>Priority</ServiceType>
  <DeliveryConfirmation>True</DeliveryConfirmation>
  <InsuranceValue>50.00</InsuranceValue>
  <MailingAckPackageID>ID00001</MailingAckPackageID>
  <WeightinPounds>12</WeightinPounds>
  <WeightinOunces>3</WeightinOunces>
```

```

    <RMA>RMA 123456</RMA>
    <ImageType>PDF</ImageType>
    <SenderName></SenderName>
    <SenderEMail></SenderEMail>
    <RecipientName></RecipientName>
    <RecipientEMail></RecipientEMail>
    <RMABarcode></RMABarcode>
</EMRSV2.0CertifyRequest>

```

“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 “Live” XML request should be in the form:

```

<EMRSV2.0Request USERID="xxxxxxx" PASSWORD="xxxxxxx">
    <CustomerName></CustomerName>
    <CustomerAddress></CustomerAddress>
    <CustomerCity></CustomerCity>
    <CustomerState></CustomerState>
    <CustomerZip5></CustomerZip5>
    <RetailerName></RetailerName>
    <RetailerAddress></RetailerAddress>
    <PermitNumber></PermitNumber>
    <PermitIssuingPOCity></PermitIssuingPOCity>
    <PermitIssuingPOState></PermitIssuingPOState>
    <PermitIssuingPOZip5></PermitIssuingPOZip5>
    <PDUPoBox></PDUPoBox>
    <PDUCity></PDUCity>
    <PDUState></PDUState>
    <PDUZip5></PDUZip5>
    <PDUZip4></PDUZip4>
    <ServiceType></ServiceType>
    <DeliveryConfirmation></DeliveryConfirmation>
    <InsuranceValue></InsuranceValue>
    <MailingAckPackageID></MailingAckPackageID>
    <WeightInPounds></WeightInPounds>
    <WeightInOunces></WeightInOunces>
    <RMA></RMA>
    <ImageType></ImageType>
    <SenderName></SenderName>
    <SenderEMail></SenderEMail>
    <RecipientName></RecipientName>
    <RecipientEMail></RecipientEMail>
    <RMABarcode></RMABarcode>
</EMRSV2.0Request>

```

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 RequestLevel As IXMLDOMElement
Dim EMRSElementLevel As IXMLDOMElement
Dim inetGet As New clsInetGet
Dim response As String
Dim t As Variant
Dim iIdx As Integer

Set xmlDoc = New DOMDocument

Set RequestLevel = xmlDoc.createElement("EMRSV2.0Request")
RequestLevel.setAttribute "USERID", UserID.Text
RequestLevel.setAttribute "PASSWORD", Password.Text

'Load Merchandise Return values into request.
Set EMRSElementLevel = xmlDoc.createElement("CustomerName")
Set t = xmlDoc.createTextNode(FromName.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("CustomerAddress")
Set t = xmlDoc.createTextNode(FromAddress.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("CustomerCity")
Set t = xmlDoc.createTextNode(FromCity.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("CustomerState")
Set t = xmlDoc.createTextNode(FromState.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("CustomerZip5")
Set t = xmlDoc.createTextNode(FromZip5.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("RetailerName")
Set t = xmlDoc.createTextNode(CompanyName.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("RetailerAddress")
Set t = xmlDoc.createTextNode(CoStreetAddr.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("PermitNumber")
Set t = xmlDoc.createTextNode(PermitNumber.Text)
EMRSElementLevel.appendChild (t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PermitIssuingPOCity")
Set t = xmlDoc.createTextNode(PermitCity.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel =
xmlDoc.createElement("PermitIssuingPOState")
Set t = xmlDoc.createTextNode(PermitState.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PermitIssuingPOZip5")
Set t = xmlDoc.createTextNode(PermitZip.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PDUPOBox")
Set t = xmlDoc.createTextNode(PDUPOBox.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PDUCity")
Set t = xmlDoc.createTextNode(PDUCity.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PDUState")
Set t = xmlDoc.createTextNode(PDUState.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PDUZip5")
Set t = xmlDoc.createTextNode(PDUZip5.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("PDUZip4")
Set t = xmlDoc.createTextNode(PDUZip4.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("ServiceType")
iIdx = ServiceType.ListIndex
Set t = xmlDoc.createTextNode(ServiceType.List(iIdx))
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel =
xmlDoc.createElement("DeliveryConfirmation")
If DelivConfirm.Value = 1 Then
Set t = xmlDoc.createTextNode("True")
Else
Set t = xmlDoc.createTextNode("False")
End If
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)
```

```
Set EMRSElementLevel = xmlDoc.createElement("InsuranceValue")
Set t = xmlDoc.createTextNode(PkgValue.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("MailingAckPackageID")
Set t = xmlDoc.createTextNode(PkgID.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("WeightInPounds")
Set t = xmlDoc.createTextNode(WgtLbs.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("WeightInOunces")
Set t = xmlDoc.createTextNode(WgtOz.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("RMA")
Set t = xmlDoc.createTextNode(txtRMA.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)

Set EMRSElementLevel = xmlDoc.createElement("ImageType")
Set t = xmlDoc.createTextNode(ImageType.Text)
EMRSElementLevel.appendChild(t)
Call RequestLevel.appendChild(EMRSElementLevel)

Call xmlDoc.appendChild(RequestLevel)

Text1.Text = xmlDoc.xml

Set RequestLevel = Nothing
Set EMRSElementLevel = Nothing
```

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 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, “Sample” requests (EMR with Delivery Confirmation only), 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. (See the bolded code under "Input," below.)

For "Canned" test requests the code should read:

```
File = "/ShippingAPItest.dll?"  
xml = "API=MerchandiseReturnV2&XML=" & XMLSTRING
```

For "Sample" requests the code should read:

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

For "Live" requests the code should read:

```
File = "/ShippingAPI.dll?"  
xml = "API=MerchandiseReturnV2&XML=" & XMLSTRING
```

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=MerchandiseReturnV2&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, “*Sample*” responses (EMR with Delivery Confirmation only), 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 oChild As IXMLDOMNode
Dim nodeList As IXMLDOMNodeList
Dim i As Integer
Dim MerchandiseReturnLabel As String
Dim Zone As String
Dim DeliveryConfirmationNumber As String
Dim Error As String
Dim StreamLength As Long
Dim ByteStream() As Byte
Dim decodeByteStream() As Byte
Dim decodeByteStreamLength As Long
Dim filenumber As Integer
Dim status As Integer
Dim filename As String
Dim InsuranceCost As String

xmlDoc.loadXML (xmlstr)

Set nodeList = xmlDoc.getElementsByTagName("Error")
If nodeList.length > 0 Then 'error
    For i = 0 To nodeList.length - 1
        Set oChild = nodeList.Item(i)
        Set oChild = oChild.firstChild
        While Not oChild Is Nothing
            Select Case oChild.nodeName
                Case "Source"
                    Error = Error & oChild.firstChild.nodeValue & "source"
                Case "Number"
                    Error = Error & oChild.firstChild.nodeValue & "number"
                Case "Description"
                    Error = Error & oChild.firstChild.nodeValue & "description"
                Case "HelpFile"
                    'Error = Error & oChild.firstChild.nodeValue & "helpfile"
                Case "HelpContext"
                    Error = Error & oChild.firstChild.nodeValue & "helpcontext"
                Case Else
            End Select
            Set oChild = oChild.nextSibling
        Wend
    Next
    MsgBox (Error)
Else ' no error
    'Get the list of nodes.
    Set nodeList = xmlDoc.getElementsByTagName("EMRSV2.0Response")

    For i = 0 To nodeList.length - 1
        Set oChild = nodeList.Item(i)

        Set oChild = oChild.firstChild
        While Not oChild Is Nothing
            Select Case oChild.nodeName
```

```
Case "Zone"
  If oChild.hasChildNodes Then Zone = oChild.firstChild.nodeValue
  Else
    'Err.Raise msERR_MR_MISSING_Zone, "", msERR_MR_MISSING_Zone
  End If
Case "InsuranceCost"
  If oChild.hasChildNodes Then
    InsuranceCost = oChild.firstChild.nodeValue
  End If
Case "MerchandiseReturnLabel"
  If oChild.hasChildNodes Then
    MerchandiseReturnLabel = oChild.firstChild.nodeValue

    ' write out image to verify that it is correct
    StreamLength = Len(MerchandiseReturnLabel)
    ReDim decodeByteStream(StreamLength)
    decodeByteStreamLength = StreamLength
    status = Base64DecodeStringToByte(MerchandiseReturnLabel,
    StreamLength, decodeByteStream, decodeByteStreamLength)
    If status <> 0 Then
      Err.Raise status
    End If
    filenumber = FreeFile
    filename = "mrtest" & Format(Date, "yyyymmdd") &
    Format(Time, "hhmm")
    If (ImageType.Text = "PDF") Then
      filename = filename & ".pdf"
    ElseIf (ImageType.Text = "TIF") Then
      filename = filename & ".tif"
    End If
    ReDim Preserve decodeByteStream(decodeByteStreamLength)
    Open filename For Binary Access Write As #filenumber
    Put #filenumber, , decodeByteStream
    Close #filenumber
  Else
    'Err.Raise msERR_MR_MISSING_MerchandiseReturnLabel, "",
    msERR_MR_MISSING_MerchandiseReturnLabel
  End If
End Select
  Set oChild = oChild.nextSibling
Wend
Next i
End If

Set oChild = Nothing
Set nodeList = Nothing
.
.
.
Private Function Base64DecodeStringToByte(MerchandiseReturnLabel As String,
StreamLength As Long, dcOutput() As Byte, dcOutputLen As Long) As Long

Dim a As String, b As String, c As String, d As String
Dim j As Integer
Dim dcCopylen As Long
Dim dcCopy As String
```

```
Const sBase As String =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

On Error GoTo DecodeErr

'-----
'Strip cr/lf
dcCopylen = 0
dcCopy = ""
For j = 1 To StreamLength
    If ((Mid(MerchandiseReturnLabel, j, 1) <> vbCr) And
(Mid(MerchandiseReturnLabel, j, 1) <> vbLf)) Then
        dcCopy = dcCopy & Mid(MerchandiseReturnLabel, j, 1)
        dcCopylen = dcCopylen + 1
    End If
Next j
'-----

'Decode bulk of string
dcOutputLen = 0
For j = 1 To dcCopylen - 4 Step 4
    'map "A"-"/" to 0-63
    a = InStr(1, sBase, Mid(dcCopy, j, 1), vbBinaryCompare) - 1
    b = InStr(1, sBase, Mid(dcCopy, j + 1, 1), vbBinaryCompare) - 1
    c = InStr(1, sBase, Mid(dcCopy, j + 2, 1), vbBinaryCompare) - 1
    d = InStr(1, sBase, Mid(dcCopy, j + 3, 1), vbBinaryCompare) - 1
    'decode 0-63 to 0-255
    dcOutput(dcOutputLen) = (a * 4) Or ((b And 48) / 16)
    dcOutput(dcOutputLen + 1) = ((b And 15) * 16) Or ((c And 60) / 4)
    dcOutput(dcOutputLen + 2) = ((c And 3) * 64) Or (d And 63)

    dcOutputLen = dcOutputLen + 3
Next j
'-----

'Decode last 1-3 characters
a = InStr(1, sBase, Mid(dcCopy, j, 1), vbBinaryCompare) - 1
b = InStr(1, sBase, Mid(dcCopy, j + 1, 1), vbBinaryCompare) - 1
dcOutput(dcOutputLen) = (a * 4) Or ((b And 48) / 16)
If j + 2 <= dcCopylen Then
    c = InStr(1, sBase, Mid(dcCopy, j + 2, 1), vbBinaryCompare) - 1
    dcOutput(dcOutputLen + 1) = ((b And 15) * 16) Or ((c And 60) / 4)
    dcOutputLen = dcOutputLen + 1
End If
If j + 3 <= dcCopylen Then
    d = InStr(1, sBase, Mid(dcCopy, j + 3, 1), vbBinaryCompare) - 1
    dcOutput(dcOutputLen + 2) = ((c And 3) * 64) Or (d And 63)
    dcOutputLen = dcOutputLen + 1
End If
Base64DecodeStringToByte = 0
Exit Function
DecodeErr:
    Base64DecodeStringToByte = Err.Number
End Function
```

Base64 Decoding in VBScript in an ASP

```

<%@ Language=VBScript %>
<%
Dim dcOutput()
sBase = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
StreamLength = len(MerchandiseReturnLabel)

'-----
'Strip cr/lf
dcCopylen = 0
dcCopy = ""
For j = 1 To StreamLength
    If ((Mid(MerchandiseReturnLabel, j, 1) <> vbCr) And
Mid(MerchandiseReturnLabel, j, 1) <> vbLf)) Then
        dcCopy = dcCopy & Mid(MerchandiseReturnLabel, j, 1)
        dcCopylen = dcCopylen + 1
    End If
Next
'-----

'-----
'Decode bulk of string
dcOutputlen = 0
ReDim dcOutput((dcCopylen * 3) / 4)
    For j = 1 To dcCopylen - 4 Step 4
        'map "A"-"/" to 0-63
        a = InStr(1, sBase, Mid(dcCopy, j, 1), vbBinaryCompare) - 1
        b = InStr(1, sBase, Mid(dcCopy, j + 1, 1), vbBinaryCompare) - 1
        c = InStr(1, sBase, Mid(dcCopy, j + 2, 1), vbBinaryCompare) - 1
        d = InStr(1, sBase, Mid(dcCopy, j + 3, 1), vbBinaryCompare) - 1
        'decode 0-63 to 0-255
        dcOutput(dcOutputlen) = (a * 4) Or ((b And 48) / 16)
        dcOutput(dcOutputlen + 1) = ((b And 15) * 16) Or ((c And 60) / 4)
        dcOutput(dcOutputlen + 2) = ((c And 3) * 64) Or (d And 63)

        dcOutputlen = dcOutputlen + 3
    Next
'-----

'-----
'Decode last 1-3 characters
a = InStr(1, sBase, Mid(dcCopy, j, 1), vbBinaryCompare) - 1
b = InStr(1, sBase, Mid(dcCopy, j + 1, 1), vbBinaryCompare) - 1
dcOutput(dcOutputlen) = (a * 4) Or ((b And 48) / 16)
If j + 2 <= dcCopylen Then
    c = InStr(1, sBase, Mid(dcCopy, j + 2, 1), vbBinaryCompare) - 1
    dcOutput(dcOutputlen + 1) = ((b And 15) * 16) Or ((c And 60) / 4)
    dcOutputlen = dcOutputlen + 1
End If
If j + 3 <= dcCopylen Then
    d = InStr(1, sBase, Mid(dcCopy, j + 3, 1), vbBinaryCompare) - 1
    dcOutput(dcOutputlen + 2) = ((c And 3) * 64) Or (d And 63)
    dcOutputlen = dcOutputlen + 1
End If

```

```
Response.ContentType = "image/tiff"
for j = 0 to dcOutputlen - 1
    response.binarywrite chrB(dcOutput(j))
next
%>
```

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, “Sample” requests, and “Live” requests. All EMR requests result in an XML response with the following tags:

Output	XML Tag
“Sample” Response	<EMRSV2.0CertifyResponse>
“Live” Response	<EMRSV2.0Response>
Zone	<Zone>
Image of Merchandise Return Label	<MerchandiseReturnLabel>
Insurance Cost	<InsuranceCost>

If you are implementing EMR with Delivery Confirmation, you will also receive the following tag:

Delivery Confirmation Number	<DeliveryConfirmationNumber>
------------------------------	------------------------------

“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:

```
<Error>
  <Number>-2147219040</Number>
  <Source>SOLServerTest;SOLServerTest.MerchandiseReturnV2_Respond</Source>
  <Description>This Information has not been included in this Test
  Server.</Description>
  <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.

EMR Response to Valid Test Request #1

```
<?xml version="1.0" ?>
- <EMRSV2.0Response>
  <Zone>3</Zone>
  <MerchandiseReturnLabel>SUkqAAgAAAASAP4ABAABAAAAAAAAAAAAAAAAABAwABAAAAALwMAAAE
  BAwABAAAAHwQAAAIB
  AwABAAAAAQAAAAMBawABAAAAABAAAAAYBAwABAAAAAooooBAwABAAAAAgAAABEB
  BAAEAAAAzDAAABIBAwABAAAAAQAAABUBAwABAAAAAQAAABYBAwABAAAAANwEAABcB
  BAAEAAAAvDAAABoBBQABAAAApAIAABsBBQABAAAArAIAACUBBAABAAAAAAAAAACgB
  AwABAAAAAgAAADEBAGa6AAAAAtAIAAKaABAABAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  pB/5d0DY+XcCAAAAUOISAGoAbAAA7P1/agAIAg8AAAAeAAAAOz9fwAAAAAAAAAAAA
  .
  . (more data goes here)
  .
  FnFEXBzRiTxDGAYG008MZhbZErnuhginy226EwXW3QxROf/pkLP10AMCQSinGrK
  QQypihRKKZgq0Aui61PnP4X2b5BkKiRj4P+74/9LBv0FAv3/twySpD85wA8puZBn
  /KZcwRv6g/9LxGVJfwqHgh6CyEVYp2L8QfsH/ZdMTenfVGFRC/o/1f/9/y+Z6pKp
  AP2movwSs2bW/1v///9P4rI2aEuWE4es/0/2W81I/7Qr/f//////////x//
  P/7//////////8BEAABZJMCZrB8/P//////////B0AABABCDwAA
  LQgAAEMWAAAZAAAA8AIAADISAABfGgAAojAAAA==</MerchandiseReturnLabel>
  <DeliveryConfirmationNumber />
  <InsuranceCost>1.10</InsuranceCost>
</EMRSV2.0Response>
```

EMR Response to Valid Test Request #2

```
<?xml version="1.0" ?>
- <EMRSV2.0Response>
  <Zone>3</Zone>
  <MerchandiseReturnLabel>JVBERi0xLjINCjUgMChvYmoNCjw8DQovVHlwZSAvWE9iamV
  jda0KL1N1YnR5cGUg
  L0ltYWdlDQovTmFtZSAvU25vd2JvdW5kMA0KL1dpZHRoIDgxNQ0KL0hlaWdodCAx
  MDU1DQovQml0c1BlckNvbXBvbmVudCAxDQovQ29sb3JtCGFjZSAvRGV2aWNlR3Jh
  eQ0KL0ZpbHRlciAvQ0NJVFRGYXhEZWNVZGUNCi9EZWNvZGVQYXJtcyA8PA0KL0sg
  LTENCi9Db2x1bW5zIDgxNSAvUm93cyAxMDU1DQovRW5kT2ZCbG9jayBmYWxzZQ0K
  L0VuZE9mTGluZSBmYWxzZQ0KL0VuY29kZWRCeXRlQWxpZ24gZmFsc2UNCj4+DQov
```

```
TGVuZ3RoIDYgMCBSDQo+Pg0Kc3RyZWftDQrLgSwY+dmB//////////zuYLnMx
k+eIwIf/gzwfH6o9s01/7YQhr+v/bL5jDjpIER///6Rw0v//6RPpf//0m0If/iIe
.
. (more data goes here)
.
MzMgMDAwMDAagbg0KMDAwMDAxMTQ4OCwMDAwMCCBuDQowMDAwMDExNzcwIDAwMDAw
IG4NCjAwMDAwMTE2MzAgMDAwMDAagbg0KMDAwMDAwMDAxMCAwMDAwMCCBuDQowMDAw
MDExNDA5IDAwMDAwIG4NCjAwMDAwMTE1MjQgMDAwMDAagbg0KdHJhaWxlcg0KPDwN
Ci9TaXplIDgNCi9Sb290IDEgMCBSDQo+Pg0Kc3RhcncR4cmVmDQoxMTg2NQ0KJSVF
T0YnCG==</MerchandiseReturnLabel>
<DeliveryConfirmationNumber />
<InsuranceCost>1.10</InsuranceCost>
</EMRSV2.0Response>
```

EMR Response to Pre-defined Error Request #1: “Invalid Customer ZIP Code”

```
<Error>
  <Number>-2147219103</Number>
  <Source>SOLServerTest;SOLServerTest.MerchandiseReturnV2_Respond</Source>
  <Description>Please Enter a valid zip code.</Description>
  <HelpFile />
  <HelpContext></HelpContext>
</Error>
```

EMR Response to Pre-defined Error Request #2: “Invalid Image Type”

```
<Error>
  <Number>-2147219102</Number>
  <Source>SOLServerTest;SOLServerTest.MerchandiseReturnV2_Respond</Source>
  <Description>Invalid image type.</Description>
  <HelpFile />
  <HelpContext></HelpContext>
</Error>
```

“Canned” Test Responses (EMR with Delivery Confirmation only)

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:

```
<Error>
  <Number>-2147219040</Number>
  <Source>SOLServerTest;SOLServerTest.MerchandiseReturnV2_Respond</Source>
  <Description>This Information has not been included in this Test
  Server.</Description>
  <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.

EMR with Delivery Confirmation Response to Valid Test Request #1

```
<?xml version="1.0" ?>
<EMRSV2.0Response>
  <Zone>3</Zone>
```

```
<MerchandiseReturnLabel>SUKqAAgAAAAMAP8AAwABAAAAAQAAAAABBAABAAAAALwMAAAE
BBAABAAAAHwQAAAIB
AwABAAAAAQAAAAAMBawABAAAAAQAAAAAYBAwABAAAAAQAAABUBAwABAAAAAQAAABoB
BQABAAAANGAAABsBBQABAAAAPgAAABwBAwABAAAAAQAAABEBBAABAAAAArgAAAD0B
AwABAAAAQAAAAAAAABgAAAAAQAAAGAAAAABAAAA//////////
//////////+//////////
//////////+//////////
//////////+//////////
//////////+//////////
.
. (more data goes here)
.
//////////
//////////+//////////
//////////+//////////
//////////+//////////
//////////+//////////
//////////+//////////
//////////+//////////
//////////+//////////
//////////+</MerchandiseReturnLabel>
<DeliveryConfirmationNumber>85805213907937077621</DeliveryConfirmationN
umber>
<InsuranceCost>1.10</InsuranceCost>
</EMRSV2.0Response>
```

EMR with Delivery Confirmation Response to Valid Test Request #2

```
<EMRSV2.0Response>
<Zone>3</Zone>
<MerchandiseReturnLabel>JVBERi0xLjINCjUgMChvYmoNCjw8DQovVHlwZSAvWE9iamV
jdA0KL1N1YnR5cGUg
L0ltYWdlDQovTmFtZSAvU25vd2JvdW5kMA0KL1dpZHRoIDgxNQ0KL0hlaWdodCAx
MDU1DQovQml0c1BlckNvbXBvbmVudCAxDQovQ29sb3JTcGFjZSAvRGV2aWN1R3Jh
eQ0KL0ZpbHRlciAvQ0NJVFRGYXhEZWNVZGUNCi9EZWNVZGVQYXJtcyA8PA0KL0sg
LTENCi9Db2x1bW5zIDgxNSAvUm93cyAxMDU1DQovRW5kt2ZCbG9jayBmYWxzZQ0K
L0VuZE9mTGluZSBmYWxzZQ0KL0VuY29kZWRCeXRlQWxpZ24gZmFsc2UNCj4+DQov
TGVuZ3RoIDYgMChBSDQo+Pg0Kc3RyZWFTDQrLgSwY+dmB//////////zuYLnMx
.
. (more data goes here)
.
bnRzIDcgMChBSDQo+Pg0KZW5kb2JqDQozIDAgb2JqDQo8PA0KL1R5cGUgL1BhZ2Vz
DQovS2lkcyBbNCaWIFlGdQpdDQovQ291bnQgMQ0KL01lZG1hQm94IFsgMCAwIDYx
Mia3OTIgcXQ0KpJ4NCmVuZG9iaG0KeHJlZg0KMCA4DQowMDAwMDAwMDAwIDY1NTM1
IGYncjAwMDAwMTI5ODEgMDAwMDAgbg0KMDAwMDAxMzAzNiAwMDAwMChBSDQowMDAw
MDEzMzE5IDAwMDAwIG4NCjAwMDAwMTMxNzggMDAwMDAgbg0KMDAwMDAwMDAxMCAw
MDAwMChBSDQowMDAwMDEyOTU3IDAwMDAwIG4NCjAwMDAwMTMwNzIgMDAwMDAgbg0K
dHJhaWxlcg0KPDwNCi9TaXplIDgNCi9Sb290IDEgMChBSDQo+Pg0Kc3Rhcnc4cmVm
DQoxMzQxMw0KJSVFT0Yncg==</MerchandiseReturnLabel>
<DeliveryConfirmationNumber>86805213907989547423</DeliveryConfirmationN
umber>
<InsuranceCost>1.10</InsuranceCost>
</EMRSV2.0Response>
```

**EMR with Delivery Confirmation Response to Pre-defined Error Request #1:
“Invalid Customer ZIP Code”**

```
<Error>
  <Number>-2147219103</Number>
  <Source>SOLServerTest;SOLServerTest.MerchandiseReturnV2_Respond</Source>
  <Description>Please Enter a valid zip code.</Description>
  <HelpFile />
  <HelpContext></HelpContext>
</Error>
```

**EMR with Delivery Confirmation Response to Pre-defined Error Request #2:
“Invalid Image Type”**

```
<Error>
  <Number>-2147219102</Number>
  <Source>SOLServerTest;SOLServerTest.MerchandiseReturnV2_Respond</Source>
  <Description>Invalid image type.</Description>
  <HelpFile />
  <HelpContext></HelpContext>
</Error>
```

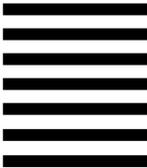
“Sample” Responses (EMR with Delivery Confirmation only)

The image returned is Base64-encoded in PDF, JPEG, “GIF”, or TIF format, according to your request (<ImageType>). It must be decoded before use. For sample code, see the *Base64 Decoding in VBScript in an ASP* section above. For additional information on Base64-encoding and decoding, consult the following URLs: <http://www.ietf.org/rfc/rfc1421.txt> (Section 4.3.2.4) and <http://www.ietf.org/rfc/rfc2045.txt> (Section 6.8).

<p>Important: When printing PDF files with barcodes, be sure that the “Fit to Page” option in the print dialogue box of Adobe Acrobat is unchecked.</p>

After decoding, the label should look like the graphic below. Instruct your customers to print out the EMR with Delivery Confirmation label on a laser or ink jet printer with 300 dpi or better. It is recommended (but not mandatory) that the EMR with Delivery Confirmation label be printed on a self-adhesive label at least 5½” x 8½”. The use of dot matrix printers is not recommended.

When you make your requests to the API Test Server, you will receive a canned response on the return address, the service requested, and delivery address for the Postage Due Unit. Once you begin sending calls to the API Production Server, you will then receive labels that have return addresses and delivery addresses with your requested data.

ID# 06052001		
FROM: LINDA E. SHOPPER 100 MAIN ST., APT. 3C NEW YORK, NY 10010		NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES
POSTAGE DUE COMPUTED BY ACCEPTANCE POST OFFICE		
POSTAGE		PRIORITY MAIL 
DELIVERY CONFIRMATION FEE		
INSURANCE FEE	\$ 6.00	
TOTAL POSTAGE AND FEES DUE	\$ 6.00	
INSURANCE DESIRED BY PERMIT HOLDER FOR	\$ 500.00 (VALUE)	
USPS DELIVERY CONFIRMATION		
		
8580 5213 9079 1618 8010		
RMA #:		
		10054321 MERCHANDISE RETURN LABEL PERMIT NO. 307 OAKLAND CA 90011 XYZ CORPORATION 1234 ETAILER DR.
		POSTAGE DUE UNIT US POSTAL SERVICE PO BOX 9998 LOS ANGELES CA 90052-1234



Customer Mailing Instructions

1. Affix the label squarely onto the address side of the parcel, covering up any previous delivery address and barcode. If tape or similar material is used, it must not cover any part of the label where postage and fee information is to be recorded or where a Delivery Confirmation Barcode exists.
2. Obliterate any other addresses and barcodes on the outside of the parcel.
3. Take the parcel to a post office so that it may be handled by a USPS retail associate.

LINDA E. SHOPPER FROM: 100 MAIN ST., APT. 3C NEW YORK, NY 10010
I.D. # 06052001
ROUND DATE _____ STAMP _____
ACCEPTANCE EMPLOYEE _____ INITIALS _____
MERCHANDISE RETURN MAILING ACKNOWLEDGMENT PERMIT NO. 307 OAKLAND CA 90011 XYZ CORPORATION 1234 ETAILER DR.

(The image above is not to scale.)

The “sample” label returned can be used for two purposes: the ten “sample” labels needed for printer certification (see Administrative Step 7 in the *Administrative Guide to APIs*), and to submit to your local post office to obtain a Merchandise Return Permit (see Administrative Step 3 in the *Administrative Guide to APIs*). When submitting “sample” labels for either printer certification or obtaining a permit, submit the entire label. **Do not** cut at the dotted line above “Customer Mailing Instructions.”

“Live” Responses

A request for an EMR label always results in the image of the label being generated in the requested format. The image returned is Base64-encoded and may be in PDF, JPEG, GIF, or TIF format. For sample code, see the *Base64 Decoding in VBScript in an ASP* section above. For additional information on Base64-encoding and decoding consult Section 4.3.2.4 in <http://www.ietf.org/rfc/rfc1421.txt> and Section 6.8 in <http://www.ietf.org/rfc/rfc2045.txt>.

Use the graphic in the size supplied. Do not resize the label.

Important: When printing PDF files with barcodes, be sure that the “Fit to Page” option in the print dialogue box of Adobe Acrobat is **unchecked**.

XML Output Example

```
<EMRSV2.0Response>
  <Zone>8</Zone>
  <MerchandiseReturnLabel>SUkqAAgAAAASAP4ABAABAAAAAAAAAAAAABBAABAAAAUgMAAAE
  BBAABAAAATAQAAAIBAwABAAAAAQAAAAMBawABAAAAABAAAAAYBAwABAAAAAooooBAwABAA
  AAAGAAABEBBAABAAAA5gEAABIBAwABAAAAAQAAABUBAwABAAAAAQAAABYBBAABAAAATAQAA
  .
  . (more data goes here)
  .
  1RTz/1ggCoeI679AoP///98gyR///wZt/fGG/uQYHPzJcWsLJOs/GPo/FfdPNeigoE816CX
  iWlRTTznVFPP/4//7////kini//+Z9f/////kinx6fr/Ww/B/1vJQXby7lZy62XLsv4jIi
  IiIiIiIiIiIuL/////////////////////////////////8//n/8////////////////////////////////
  //////////////////////////////////x8AARAAAQ==</MerchandiseReturnLabel>
  <InsuranceCost>1.10</InsuranceCost>
</EMRSV2.0Response>
```

XML Output Example (EMR with Delivery Confirmation)

```
<EMRSV2.0Response>
  <Zone>8</Zone>
  <MerchandiseReturnLabel>SUkqAAgAAAASAP4ABAABAAAAAAAAAAAAABBAABAAAAUgMAAAE
  BBAABAAAATAQAAAIBAwABAAAAAQAAAAMBawABAAAAABAAAAAYBAwABAAAAAooooBAwABAA
  AAAGAAABEBBAABAAAA5gEAABIBAwABAAAAAQAAABUBAwABAAAAAQAAABYBBAABAAAATAQAA
  .
  . (more data goes here)
  .
  1RTz/1ggCoeI679AoP///98gyR///wZt/fGG/uQYHPzJcWsLJOs/GPo/FfdPNeigoE816CX
  iWlRTTznVFPP/4//7////kini//+Z9f/////kinx6fr/Ww/B/1vJQXby7lZy62XLsv4jIi
  IiIiIiIiIiIuL/////////////////////////////////8//n/8////////////////////////////////
  //////////////////////////////////x8AARAAAQ==</MerchandiseReturnLabel>
  <DeliveryConfirmationNumber>85805213907937077621</DeliveryConfirmationN
  umber>
  <InsuranceCost>1.10</InsuranceCost>
</EMRSV2.0Response>
```

Generating Your Own Label

You may choose not to use the label provided in the output if you have the ability to generate your own. If you choose to generate your own label, the Delivery Confirmation number and

Insurance Cost (if applicable) returned in the response must be used. Refer to the *Example with Different Options* section to view the placement of the Delivery Confirmation number and the Insurance Cost.

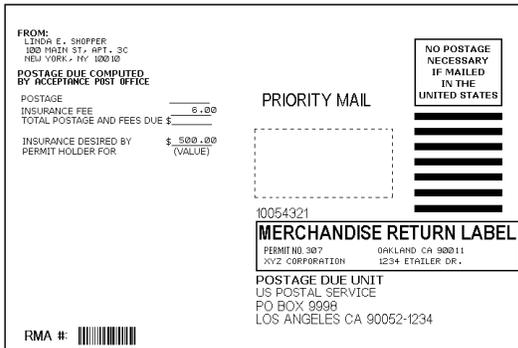
If you plan to generate your own Merchandise Return label, you must adhere to the requirements in DMM S923, located at the Postal Explorer web site <http://pe.usps.gov>. Also see USPS Publication 91 at <http://www.usps.com/cpim/ftp/pubs/pub91/welcome.htm>. This publication contains the specifications for the Delivery Confirmation label that must be met.

If you make multiple calls to the EMR with Delivery Confirmation API, you must be sure to match up the right Delivery Confirmation number and place it on a label and package with the correct from and to address information used on the input request.

Your logo (or any other information you choose, such as a shipping barcode or your company name) can be placed on the EMR label once it is generated and transmitted from the USPS in a PDF, JPEG, GIF, or TIF file. As shown in the dotted line box in the examples below (without Delivery Confirmation on the left, with Delivery Confirmation on the right), logos can be placed on the label in a 1" x 2" "free area" immediate above the Merchandise Return label box and immediate to the left of the black horizontal bars. Whatever is placed in the free area must not interfere with any EMR format elements or required markings. Should you desire to place inventory or shipping barcodes in the free area, the barcodes must not resemble the USPS 5-digit barcodes as described in DMM C850.

The image in the free area should be set at 96 dpi. The coordinates to place the image are:

- top left 370, 195
- bottom right 570, 295



(The images above are not to scale.)

Note: There is no tag that will allow the USPS to place the logo on the label through the XML request process. It can only be done after receiving the label from the API.