Parcel Data Exchange (PDX)

Application Program Interface (API)

User Guide

Version: 50

Table of Contents

Contents

1. Introduction 6
1.1 Purpose of Technical Specifications 6
1.2 Intended Audience 6
1.3 PDX Access 6
1.3.1 Account Inactivity/Deactivation 6
1.4 API Calls 7
1.5 Best Common Practices 7
2. Types of API Clients 9
2.1 REST Client - POSTMAN 9
2.2 cURL 11
2.3 Java 11
3. PDX API Calls 13
PDX Customer Acceptance Test (CAT) Environment 13
3.1 Upload a Manifest File 14
curl Sample: 15
Sample response: 15
Response elements: 15
POSTMAN Sample 15
Java Sample: 16
3.2 Get a List of Uploaded Files 18
curl Sample: 19
Sample response: 19
Response elements: 19
Java Sample: 21
3.3 Get an Uploaded File 23
curl Sample: 23
Sample response: 23
Java Sample: 24

3.4 Get a List of all Outbound Files 25
curl Sample: 26
Sample response: 26
Response elements: 27
POSTMAN Sample 28
Java Sample: 28
3.5 Get an Outbound File 30
curl Sample: 30
Sample response: 30
POSTMAN Sample: 30
Java Sample: 30
3.6 Get Multiple Outbound Files as a ZIP 32
curl Sample: 33
POSTMAN Sample 34
3.7 DEPRECATED: Get Multiple Outbound Files into a Zip Folder 35
curl Sample: 36
Sample response: 36
POSTMAN Sample: 36
Java Sample: 37
3.8 DEPRECATED: Get a List of Outbound Files by MID 39
curl Sample: 40
Sample response: 40
POSTMAN Sample 41
Java Sample: 41
4. Appendix 44
4.1 POM.xml 44
4.2 Error Messages 45
Authentication 45
PDX Unavailable 45
Manifest Upload 45
API Parameter Validation Errors 46

4.3 Customer Outbound File Types 48
4.4 Manifest File naming 49
4.5 .NET Code Sample for API File Upload 50
4.6 User Agent Information 52
4.7 Protocol / Communication Errors 53
4.8 Support Information 54

REVISION HISTORY

Date	Revision #	Details
02/03/2021	34	Updated copyright
02/22/2021	35	Updated errors messages and simultaneous Zips note
03/24/2021	36	Changed fromDate in Extracts Zip to be optional; changed extract job processing to once every 15 minutes; added deactivation information; updated error message for day of month to [1,31]
07/01/2021	37	Added HTTP 304 status information
07/28/2021	38	Added explanation of API results; update screenshots and API responses
08/18/2021	39	Added PDX unavailable error message; added access failure retry
01/04/2022	40	Added authentication error messages to section 4.2.
01/17/2022	41	Added more authentication error messages to section 4.2.
01/31/2022	42	Added manifest upload error section
03/07/2022	43	Use either List of Extracts or Extracts ZIP was added to the Best Common Practices section.
03/16/2022	44	Added Best Common Practice: Wait until extract download completes
04/06/2022	45	When the list returned is less than the pageSize, the data is complete; update the library versions in the POM appendix.
05/25/2022	46	Added new error messages for manifest upload rejection; added hyperlinks.
11/02/2022	47	Changed manifest name length to 88 characters; replaced deprecated Get Multiple Outbound Files as a ZIP with new version and moved the old to the deprecated section.
01/18/2023	48	Added USPS Ship fileTypes.
04/19/2023	49	Added warning to aggregate manifest contents. Deleted 2020 revision history.
09/14/2023	50	Updated error messages (Forbidden)

1. Introduction

1.1 Purpose of Technical Specifications

Parcel Data Exchange (PDX) is a web and Application Programming Interface (API) based application that provides the capability for customers to send manifest files to the United States Postal Service® (USPS®) and receive outbound files from USPS, pertaining to domestic and international parcel shipments.

1.2 Intended Audience

This document is intended for users of PDX who use or plan to use a Representational State Transfer (REST) API based software application to connect to PDX. Users can automate their PDX requests utilizing REST clients.

1.3 PDX Access

All PDX Web Service calls require a USPS® Business Customer Gateway (BCG) username and password. If you do not already have a BCG account, go to https://gateway.usps.com and click the 'Register for free' button to create a username and password.

Obtaining a BCG account allows you to upload manifests to PDX. No additional configuration is required. If you need to download Outbound Files (Extracts), your BCG account must be registered to a program and configured to Web Services.

For additional information about creating a BCG account, resetting a BCG account password or configuring a BCG account to receive Outbound Files (Extracts), refer to the PDX Online External User Guide (Section 2).

For each of the API requests described in this document, the BCG username and password are sent to the server using the BASIC AUTH mechanism of Hypertext Transfer Protocol (HTTP). These requests return a suitable HTTP Status Code. A status code of 200 will be returned for all successful requests.

To assist with the API setup, users can obtain a REST client. Numerous REST clients are available for free. A popular such client is POSTMAN (formerly an Extension of the Chrome web browser). Native POSTMAN is available for free at https://www.getpostman.com.

1.3.1 Account Inactivity/Deactivation

A PDX account that is inactive for more than 90 days will be automatically deactivated.

Using a deactivated account to access PDX via the API will return the following error:

Error 401: API Access Denied. Your account has been temporarily deactivated. Please contact the PDX helpdesk at DTS-PDX@usps.gov for assistance.

1.4 API Calls

There are six Web Service (WS) calls available with the API application:

- Upload a File
- Get a List of Uploaded Files
- Get an Uploaded File
- Get a List of Outbound Files
- Get an Outbound File
- Get Multiple Outbound Files in a Zip Folder

1.5 Best Common Practices

The current best common practices provide rules to utilize PDX efficiently to upload, list and download the most timely and accurate information with the least amount of API calls.

General:

1. Create a Customer Acceptance Test (CAT) BCG account. Use this account to test access, application changes, etc.
2. Add User Agent information to API requests to log company information.
3. Review all messages returned from PDX. Take appropriate corrective actions to avoid overloading the system with repeated erroneous requests.
4. Too many consecutive API calls to PDX can delay responses. Provide adequate pause/sleep between successive API calls (e.g. sleep 5 seconds after every 25 API calls).
5. Retry failed API calls with adequate pause/sleep between retries (e.g. sleep 10 minutes between access failures).

Manifests:

1. Manifest filenames should be unique (to assist in possible debugging; example: add a date/timestamp). Uploading the same named manifest repeatedly can delay processing and potentially cause data loss.
2. Upload each manifest once. Aggregate data and send larger manifests less frequently. Manifest filenames should be 88 or less alphanumeric characters. Manifest file contents must be alphanumeric text; special characters or file encoding will cause manifest rejection and/or processing to fail.
3. Listing and/or downloading manifests should not be used frequently. Once a manifest ID is returned from a manifest upload request, that manifest has been successfully uploaded to PDX and does not require listing or downloading.

Extracts/Outbound Files:

1. Either execute List of Extracts/Extract download requests -or- execute Extracts ZIP requests. Executing both may result in duplicate extract downloads.
2. Limit "List of Extracts" requests to every fifteen (15) minutes or **more**. Extract data is processed by PDX once every 15 minutes. Requesting lists repeatedly within a 15-minute window provides no additional information and may slow down responses. Frequency of "List of Extracts" requests should be based on the number of extracts received. Some customers find that requesting a list once a day is adequate.

3. Rather than repeatedly looping through all extracts using the "List of Extracts" call, limit the response data using optional parameters (click on the link to view). These parameters include pageSize, fromDate, lastId and notDownloaded.
4. When the results of a list request are less than the pageSize requested, there is no more data to retrieve. Do not request the "next page" as it will return zero results.
5. To retrieve more data per response, set the pageSize parameter in the "List of Extracts" API call to the maximum (i.e., pageSize=500).

Extract/Outbound File Download

1. Extract content does not change. Each extract should only be downloaded **once**.
2. Wait for each "Extract Download" request to finish before submitting other requests.
3. Wait for each "Get Multiple Outbound Files into a Zip Folder" request to finish before submitting other requests; zip requests of hundreds of files may take minutes to process. Simultaneous Zip requests will be denied.

2. Types of API Clients

Samples are provided here for users who prefer to send API requests directly using API or REST clients. The samples include POSTMAN, cURL and Java, though many others may be used.

2.1 REST Client - POSTMAN

Once POSTMAN is installed, the user may start POSTMAN from the Windows start menu as seen in Figure 1. Actual POSTMAN screens may vary with different versions, but the content/fields are similar.

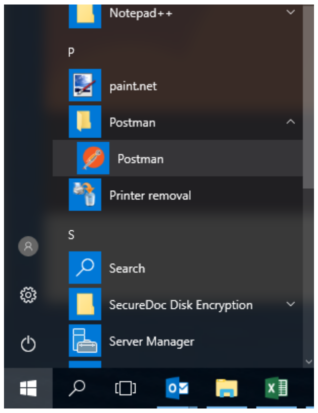
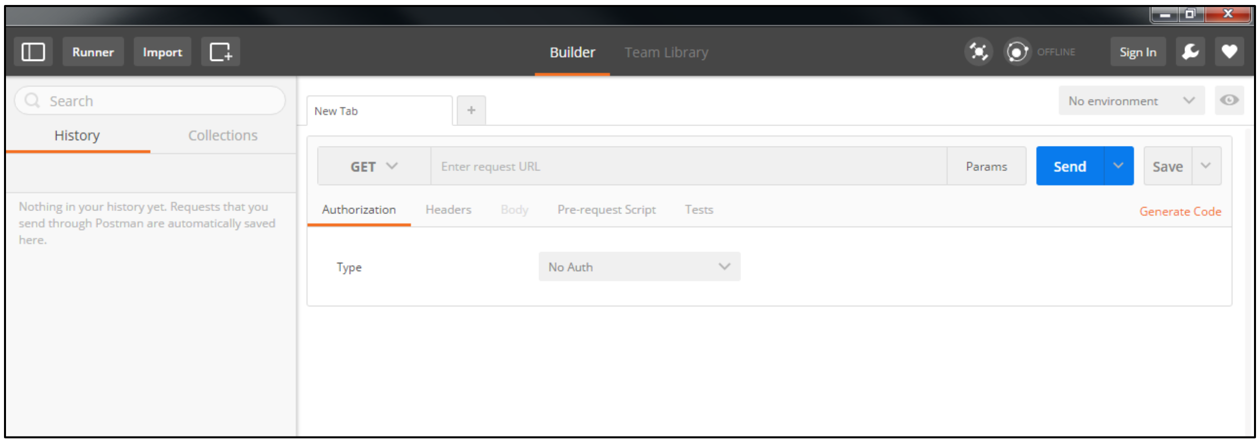


Figure 1: Starting Native POSTMAN on Windows



Nothing in your history yet. Requests that you send through Postman are automatically saved here.

Figure 2: Initial POSTMAN screen

Before making any request, users should select 'Basic Auth' from the type dropdown as seen in figure 3.

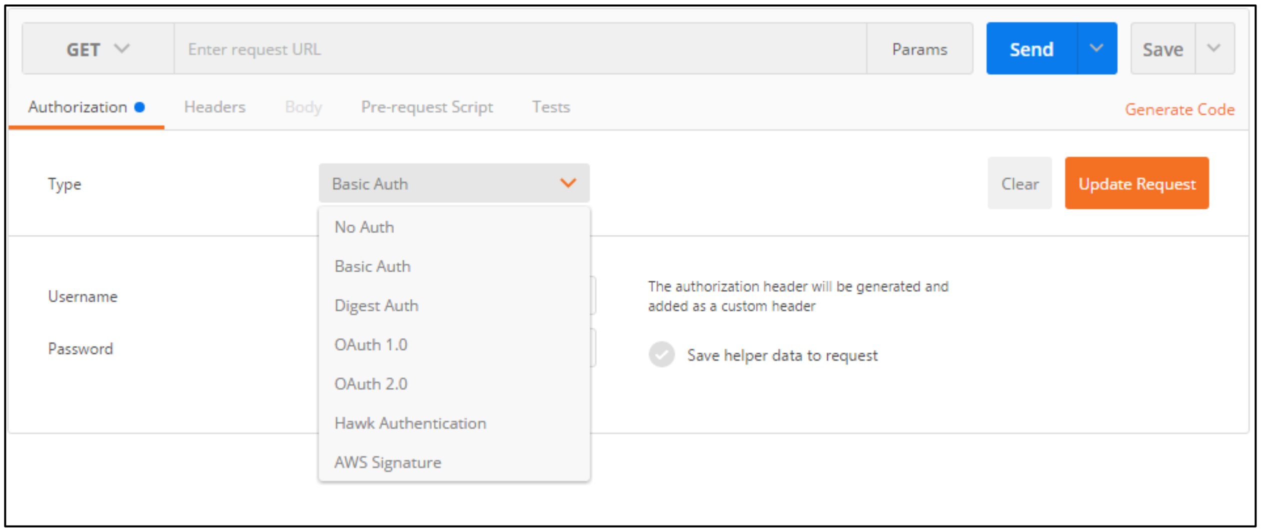


Figure 3: Creating a Basic Auth header, Select Type

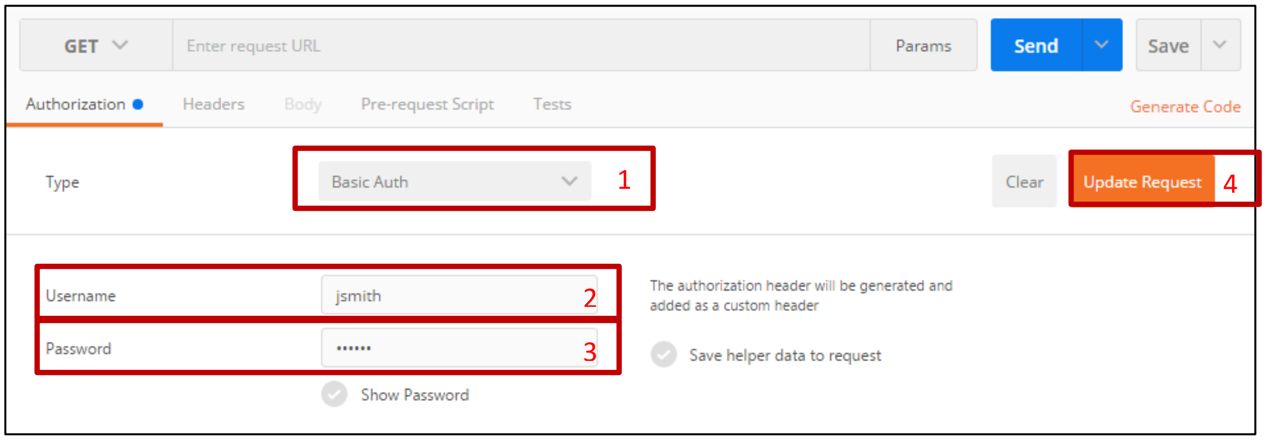
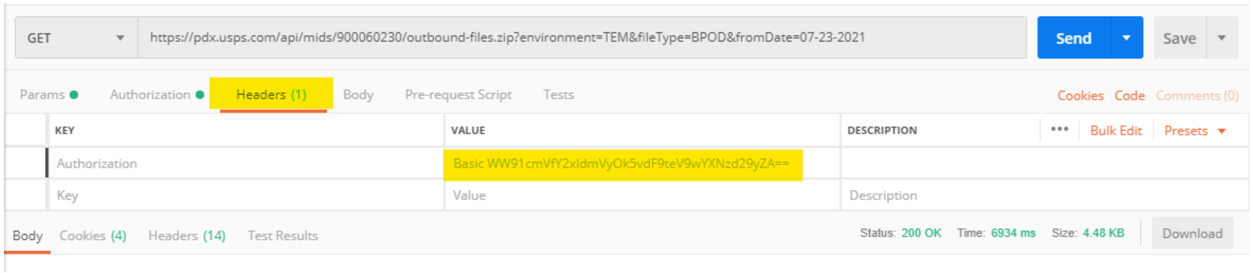


Figure 4: Creating a Basic Auth header (continued)

After selecting "Basic Auth" you will be prompted for your authorization information. Insert your BCG username and password, then click "Update Request." This creates the generated header displayed in Figure 5.


GET https://pdx.usps.com/api/mids/900060230/outbound-files.zip?environment=TEM&fileType=BPOD&fromDate=07-23-2021

KEY	VALUE	DESCRIPTION
Authorization	Basic WW91cmVfY2xldmVyOk5vdF9teV9wYXNzd29yZA==	
Key	Value	Description

Status: 200 OK Time: 6934 ms Size: 4.48 KB

Figure 5: Basic Auth header

2.2 cURL

The Client URL (cURL) command is run from the command line. Some arguments include:

- –u {BCG Username:Password}
- –X {POST, GET}
- -F {variables}
- URL (https://pdx.usps.com/api/…)

Example to upload a manifest named "test.manifest" to the TEM environment for user jsmith:

```
$ curl -u jsmith:jsmith -X POST -F "environment=TEM" -F
"filename=test.manifest" -F "multipartFile=@test.manifest"
'https://pdx.usps.com/api/manifests'
```

2.3 Java

The Java programming language can be used to make API calls to PDX. Example to download a manifest with an ID number of 17390, by user APIUser:

```
public class DownloadManifests {

	public static void main(String[] args) {
		// Set the ID of the file to be downloaded.
		final int ID = 17390;
		// Set the credentials for the PDX user
		final String USERNAME = "APIUser";
		final String PASSWORD = "APIUserPassword";
		// The credentials are converted to a basic auth format.
		final String CREDENTIALS = USERNAME + ":" + PASSWORD;
		final String AUTHORIZATION = "Basic "
				+ Base64.encodeBase64String(CREDENTIALS.getBytes());
		// This is the request URL.
		final String requestURL = "https://pdx.usps.com/api/manifests/{id}";
		// Declare the Rest Template as a variable.
		RestTemplate rstTemplate = new RestTemplate();
		// Declare the HTTP Headers variable.
		HttpHeaders headers = new HttpHeaders();
		// Set the basic auth user credentials.
		headers.set("Authorization", AUTHORIZATION);
		// Set the acceptType as octet-stream
		headers.set("Accept", "application/octet-stream");
```

```
        // Create the request with the headers set above.
        HttpEntity<String> request = new HttpEntity<String>(headers);
        // Make an API call to the REQUEST_URL as a GET method, using the
        // request set above, and with the response as type String.
        // The URL parameters are defined in order at the end of the
        // parameters list. The response is put in a Response Entity.
        ResponseEntity<String> response = rstTemplate.exchange(requestURL,
                     HttpMethod.GET, request, String.class, ID);
        // Print out the response as a string.
        System.out.println(response.getBody());
}}
```

3. PDX API Calls

NOTE: Unless specified as *optional*, all HTTP parameters are required.

PDX Customer Acceptance Test (CAT) Environment

It is recommended that customers create an account in the PDX Customer Acceptance Test (CAT) environment. The PDX CAT environment is separate from PDX Production and provides an isolated testing environment or “sandbox” for customers to test PDX file uploads, lists and downloads without impacting any Production data.

To create a CAT account, customers must log into the Business Customer Gateway (BCG) using the URL: “https://gateway-cat.usps.com”. A new username and password for the CAT environment are required to create the new CAT account.

PDX CAT API calls mimic PDX Production API calls, except for changing the URL of the Production API calls from “pdx.usps.com” to “pdx-cat.usps.com” and updating the username/password to the BCG CAT account credentials. Users will need to request new MIDs to use in the CAT environment.

3.1 Upload a Manifest File

NOTE: Overloading PDX with five or more manifest uploads per minute may cause your account to be deactivated. Manifest content should be aggregated (i.e., concatenated together) into larger files (less than 256MB).

PDX supports the following file extensions: .manifest, .consolid, .delivery, .subscribe and .podsub.
Example: 20170712_080123_shipservices.manifest

Manifests ending in extensions other than these will be rejected. If in doubt, use .manifest. File extensions should be in lower case. Uploaded files should not exceed the filename length of 88 characters. Manifest filenames should be unique (to assist in possible debugging). Manifest filenames should be alphanumeric characters. Manifest file contents must be alphanumeric text without encoding; special characters or encoding will cause manifest processing to fail. PDX provides no validation on file contents – downstream systems perform content checking. Manifest upload failure messages are detailed in Appendix 4.2 Error Messages.

URL: `https://pdx.usps.com/api/manifests`

HTTP Method: `POST`

HTTP enctype: `multipart/form-data`

HTTP Headers: `Accept: application/json`

HTTP Parameters:

Name	Type	Description
`environment`	String	This field can refer to one of the two supported environments: **"TEM"** This is used for non-production data for testing purposes. It refers to "Test Environment for Mailers". **"PROD"** This is used for production data. It refers to the "Production" environment.
`filename`	String	The name assigned to the uploaded file (used by PDX)
`multipartFile`	Binary	This is the data representation of the uploaded file (the contents or data stream that is being uploaded)

Output: A JavaScript Object Notation (JSON) formatted string with information about the uploaded file.

curl Sample:

In the curl example below, the contents of test.manifest will be sent to PDX as file: test.manifest. The extension of the multipartFile will be validated.

```
$ curl -u jsmith:jsmith -X POST -F "environment=TEM" -F
"filename=test.manifest" -F "multipartFile=@test.manifest"
'https://pdx.usps.com/api/manifests'
```

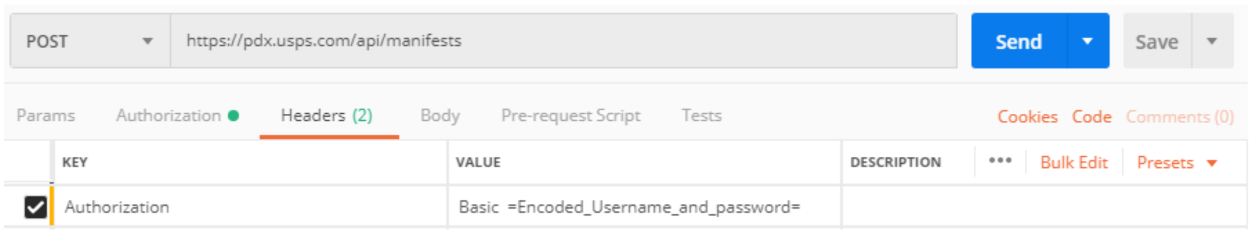
Sample response:

```
   {
      "id":4717562,
      "environment":"TEST ENVIRONMENT FOR MAILERS",
      "createdTime":"Jul 27, 2021 05:50:54 AM",
      "sentTime":null,
      "filename":"000.TEST.podsub"
   }
```

Response elements:

Name	Type	Description
`id`	Integer	The PDX manifest identifier; this value is used in a manifest download request.
`environment`	String	The environment the manifest was uploaded into; either “PRODUCTION” or “TEST ENVIRONMENT FOR MAILERS”.
`createdTime`	Date	The date and time (in Central Time) that the manifest was uploaded into PDX.
`sentTime`	Date	The date and time (Central Time) that the manifest was forwarded by PDX to internal USPS processing (this field is not populated at creation time).
`filename`	String	The name that the manifest was saved as (some encoding of special characters may occur).

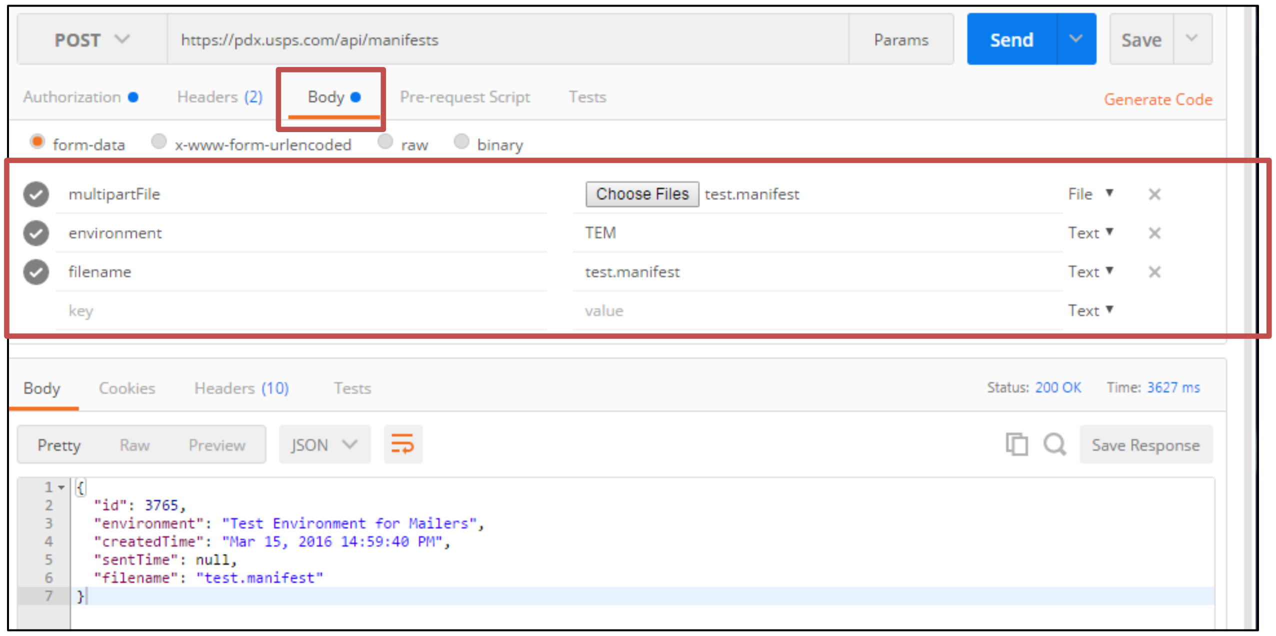
POSTMAN Sample:



KEY	VALUE	DESCRIPTION
Authorization	Basic =Encoded_Username_and_password=	

Figure 6: POSTMAN uploading a manifest file, Part 1

Add parameters to the request in the Body tab. See figure 7.



multipartFile	Choose Files test.manifest	File
environment	TEM	Text
filename	test.manifest	Text
key	value	Text

Status: 200 OK Time: 3627 ms

```
{
    "id": 3765,
    "environment": "Test Environment for Mailers",
    "createdTime": "Mar 15, 2016 14:59:40 PM",
    "sentTime": null,
    "filename": "test.manifest"
}
```

Figure 7: POSTMAN uploading a manifest file, Part 2

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

Please see Appendix 3.4 for a .NET code example for file uploads to the PDX API.

```
import java.io.IOException;
import org.springframework.core.io.FileSystemResource;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestClientException;
```

```java
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;

/**
 *
 * Uses Spring Rest Template to call the PDX API to upload a manifest with a
 * specified environment and filename
 */

public class UploadManifests {

	public static void main(String[] args) {
		// Set the credentials for the PDX user
		final String USERNAME = "APIUser";
		final String PASSWORD = "APIUserPassword";
		// The credentials are converted to a basic auth format.
		final String CREDENTIALS = USERNAME + ":" + PASSWORD;
		final String AUTHORIZATION = "Basic "
				+ Base64.encodeBase64String(CREDENTIALS.getBytes());
		// Set the name assigned to the uploaded file (used by PDX)
		final String FILENAME = "test.manifest";
		// Choose which environment to upload the manifest file to. Options are
		// "TEM" and "PROD".
		final String ENVIRONMENT = "TEM";
		// The data representation of the uploaded file
		// (the contents or data stream that is being uploaded)
		final String FILE_PATH = "C:/Users/test/workspace/client/src/main/"
				+ "files/test.manifest";
		// This is the request URL.
		final String REQUEST_URL = "https://pdx.usps.com/api/manifests";
		// Declare the Rest Template variable.
		RestTemplate rstTemplate = new RestTemplate();
		// Create the request body.
		// Declare the body as a Multi Value Map.
		MultiValueMap<String, Object> body =
		new LinkedMultiValueMap<String, Object>();
		// Add the filename as a parameter.
		body.add("filename", FILENAME);
		// Add the environment as a parameter.
		body.add("environment", ENVIRONMENT);
		// Add the multipartFile as a parameter.  The multipartFile will either
		// be the contents or data stream that is being uploaded
		body.add("multipartFile", new FileSystemResource(FILE_PATH));
		// Set HTTP headers
		// Declare the HTTP Headers variable.
		HttpHeaders headers = new HttpHeaders();
		// Set the basic auth user credentials.
		headers.set("Authorization", AUTHORIZATION);
		// Set the acceptType as JSON
		headers.set("Accept", "application/json");
		// Create the request with the request body and headers set above.
		HttpEntity<Object> request = new HttpEntity<Object>(body, headers);
		// Make an API call to the REQUEST_URL as a POST method, using the
```

```
                // request
                // created above, and with the response as type String.
                // The response is put in a Response Entity.
                ResponseEntity<String> response = rstTemplate.exchange(REQUEST_URL,
                                HttpMethod.POST, request, String.class);
                // Print out the response as a string.
                System.out.println(response.getBody());
        }
}
```

3.2 Get a List of Uploaded Files

URL: `https://pdx.usps.com/api/manifests`

HTTP Method: `GET`

HTTP Headers: `Accept: application/json`

HTTP Parameters:

Note: Those manifests marked as "Hide" (hidden) on the manifest UI will not be displayed in the JSON output.

Name	Type	Description
`pageNumber`	Integer	*Optional* Manifest responses are paginated since there can be many entries in the response. This field specifies the page number. The first page is 0.
`pageSize`	Integer	*Optional* The default is 100; the maximum page size request is 500.
`sumCount`	Boolean	*Optional* **"true"** – The value of "totalResults" returned will be the total number of uploaded manifests. **"false"** – The value of "totalResults" returned will be the value of pageSize.

Output: A JSON formatted string with information about the files.

curl Sample:

```
$ curl -u jsmith:jsmith  -X GET
'https://pdx.usps.com/api/manifests?pageNumber=0&pageSize=500'
```

Sample response:

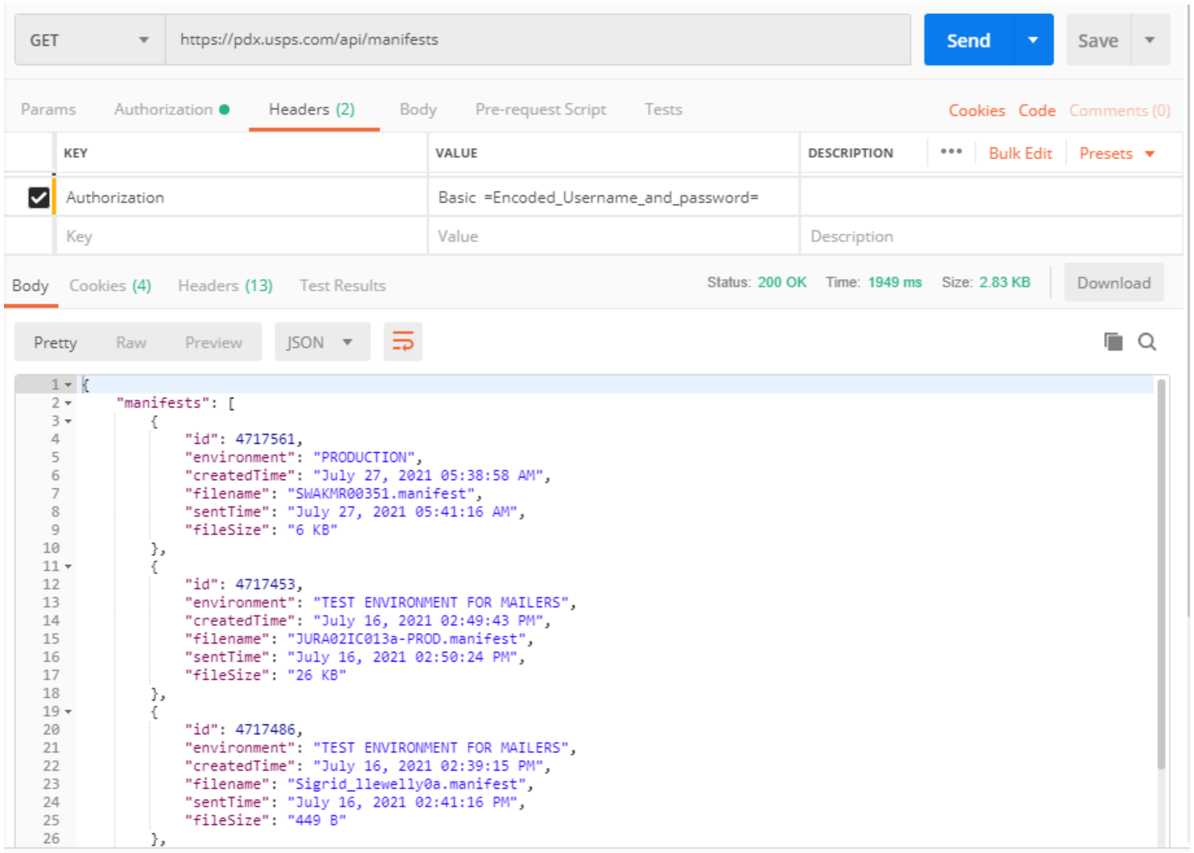
```
   {
      "manifests":[
      {
            "id":4716062,
            "environment":"TEST ENVIRONMENT FOR MAILERS",
            "createdTime":"June 23, 2021 01:35:21 PM",
            "filename":"23D_969005363_10230106.subscribe",
            "sentTime":"June 23, 2021 01:36:22 PM",
            "fileSize":"26 KB"
      },{
            "id":4717561,
            "environment":"PRODUCTION",
            "createdTime":"July 27, 2021 05:38:58 AM",
            "filename":"SWAKMR00351.manifest",
            "sentTime":"July 27, 2021 05:41:16 AM",
            "fileSize":"6 KB"
   }
   …
      ],
      "totalResults":4,
      "pageNumber":1,
      "pageSize":4
   }
```

Response elements:

Name	Type	Description
`id`	Integer	The PDX manifest identifier; this value is used in a manifest download request.
`environment`	String	The environment the manifest was uploaded into; either “PRODUCTION” or “TEST ENVIRONMENT FOR MAILERS”.
`createdTime`	Date	The date and time (in Central Time) that the manifest was uploaded into PDX.
`filename`	String	The name that the manifest was saved as (some encoding of special characters may occur).
`sentTime`	Date	The date and time (Central Time) that the manifest was forwarded by PDX to internal USPS processing.
`fileSize`	String	The size of the file uploaded.

`totalResults`	Integer	Refer to the “sumCount” explanation above.
`pageNumber`	Integer	The number of the page requested (default: 0).
`pageSize`	Integer	The size of the page requested (affected by “sumCount”).

POSTMAN Sample:



https://pdx.usps.com/api/manifests

KEY	VALUE	DESCRIPTION
✓ Authorization	Basic =Encoded_Username_and_password=	
Key	Value	Description

Status: 200 OK Time: 1949 ms Size: 2.83 KB

```
{
    "manifests": [
        {
            "id": 4717561,
            "environment": "PRODUCTION",
            "createdTime": "July 27, 2021 05:38:58 AM",
            "filename": "SWAKMR00351.manifest",
            "sentTime": "July 27, 2021 05:41:16 AM",
            "fileSize": "6 KB"
        },
        {
            "id": 4717453,
            "environment": "TEST ENVIRONMENT FOR MAILERS",
            "createdTime": "July 16, 2021 02:49:43 PM",
            "filename": "JURA02IC013a-PROD.manifest",
            "sentTime": "July 16, 2021 02:50:24 PM",
            "fileSize": "26 KB"
        },
        {
            "id": 4717486,
            "environment": "TEST ENVIRONMENT FOR MAILERS",
            "createdTime": "July 16, 2021 02:39:15 PM",
            "filename": "Sigrid_llewelly0a.manifest",
            "sentTime": "July 16, 2021 02:41:16 PM",
            "fileSize": "449 B"
        },
```

Figure 8: POSTMAN getting a list of uploaded files

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

```
import java.io.IOException;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;

/**
 * Uses Spring Rest Template to call the PDX API to view a manifests from a user
 * with specified pageNumber and pageSize
 */

public class ViewManifests {

      public static void main(String[] args) {
```

```
		// The response will be paginated. Use PAGE_SIZE to choose the number of
		// outbound files returned per page and PAGE_NUMBER to choose which page
		// to return.
		final int PAGE_NUMBER = 0;
		final int PAGE_SIZE = 500;
		// Set the credentials for the PDX user
		final String USERNAME = "APIUser";
		final String PASSWORD = "APIUserPassword";
		// The credentials are converted to a basic auth format.
		final String CREDENTIALS = USERNAME + ":" + PASSWORD;
		final String AUTHORIZATION = "Basic "
				+ Base64.encodeBase64String(CREDENTIALS.getBytes());
		// This is the request URL. Optional fields notDownloaded and fileType
		// may be removed as desired.
		final String REQUEST_URL = "https://pdx.usps.com/api/manifests"
				+ "?pageNumber={pageNumber}&pageSize={pageSize}";
		// Declare the Rest Template variable.
		RestTemplate rstTemplate = new RestTemplate();
		// Set HTTP headers and declare the HTTP Headers variable.
		HttpHeaders headers = new HttpHeaders();
		// Set the basic auth user credentials.
		headers.set("Authorization", AUTHORIZATION);
		// Set the acceptType as JSON
		headers.set("Accept", "application/json");
		// Create the request with the headers set above.
		HttpEntity<String> request = new HttpEntity<String>(headers);
		// Make an API call to the REQUEST_URL as a GET method, using the
		// request above with the response as type String. The URL parameters
		// (curly bracketed in the REQUEST_URL) are defined in order at the end
		// of the parameters list. The response is put in a Response Entity.
		ResponseEntity<String> response = rstTemplate.exchange(REQUEST_URL,
				HttpMethod.GET, request, String.class, PAGE_NUMBER,
PAGE_SIZE);
		// Print out the response as a string.
		System.out.println(response.getBody());
	}}
```

3.3 Get an Uploaded File

URL: `https://pdx.usps.com/api/manifests/{id}`

Replace `{id}` with the ID of a previously uploaded file. This ID can be found in the list of uploaded files, which can be retrieved as described in Section 3.2 Get a List of Uploaded Files.

HTTP Method: `GET`

HTTP Headers: `Accept: application/octet-stream`

HTTP Parameters: None

Output: The requested file is returned as a stream of data.

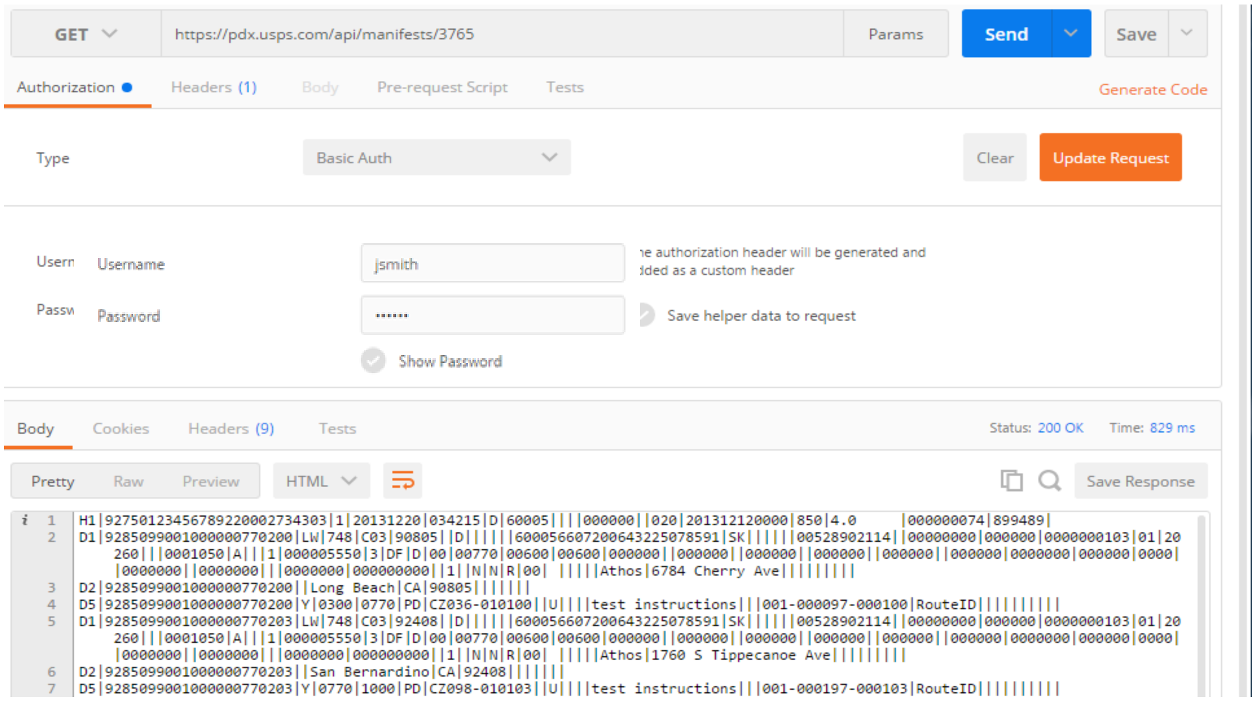
curl Sample:

$ curl -u jsmith:jsmith –X GET 'https://pdx.usps.com/api/manifests/3765'

Sample response:

The data returned will be the contents of the manifest.

POSTMAN Sample:



https://pdx.usps.com/api/manifests/3765

Type: Basic Auth

Username: jsmith

Password: ••••••

ɪe authorization header will be generated and ɪded as a custom header

Save helper data to request

Show Password

Status: 200 OK Time: 829 ms

```
H1|92750123456789220002734303|1|20131220|034215|D|60005|||000000||020|201312120000|850|4.0      |000000074|899489|
D1|92850990010000007702000|LW|748|C03|90805||D|||||60005660720064322507859 1|SK|||||00528902114||00000000|000000|0000000103|01|20
    260|||0001050|A|||1|000005550|3|DF|D|00|00770|00600|00600|000000||000000||000000||000000||000000||000000|0000000|000000|0000|
    |0000000||0000000|||0000000|000000000||1||N|N|R|00| |||||Athos|6784 Cherry Ave|||||||||
D2|92850990010000007702000||Long Beach|CA|90805||||||
D5|92850990010000007702000|Y|0300|0770|PD|CZ036-010100||U|||test instructions|||001-000097-000100|RouteID||||||||||
D1|92850990010000007702003|LW|748|C03|92408||D|||||60005660720064322507859 1|SK|||||00528902114||00000000|000000|0000000103|01|20
    260|||0001050|A|||1|000005550|3|DF|D|00|00770|00600|00600|000000||000000||000000||000000||000000||000000|0000000|000000|0000|
    |0000000||0000000|||0000000|000000000||1||N|N|R|00| |||||Athos|1760 S Tippecanoe Ave|||||||||
D2|92850990010000007702003||San Bernardino|CA|92408||||||
D5|92850990010000007702003|Y|0770|1000|PD|CZ098-010103||U|||test instructions|||001-000197-000103|RouteID||||||||||
```

Figure 9: POSTMAN getting a previously uploaded file

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

```
import java.io.IOException;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;

/**
 * Uses Spring Rest Template to call the PDX API to download a manifest by a
specified manifest ID
 */

public class DownloadManifests {

	public static void main(String[] args) {
		// Set the ID of the file to be downloaded. You can find the ID by using
		// the View Manifests API.
		final int ID = 17390;
		// Set the credentials for the PDX user
		final String USERNAME = "APIUser";
		final String PASSWORD = "APIUserPassword";
		// The credentials are converted to a basic auth format.
		final String CREDENTIALS = USERNAME + ":" + PASSWORD;
		final String AUTHORIZATION = "Basic "
				+ Base64.encodeBase64String(CREDENTIALS.getBytes());
		// This is the request URL.
		final String requestURL = "https://pdx.usps.com/api/manifests/{id}";
		// Declare the Rest Template as a variable.
		RestTemplate rstTemplate = new RestTemplate();
		// Set HTTP headers; Declare the HTTP Headers variable.
		HttpHeaders headers = new HttpHeaders();
		// Set the basic auth user credentials.
		headers.set("Authorization", AUTHORIZATION);
		// Set the acceptType as octet-stream
		headers.set("Accept", "application/octet-stream");
		// Create the request with the headers set above.
		HttpEntity<String> request = new HttpEntity<String>(headers);
		// Make an API call to the REQUEST_URL as a GET method, using the
		// request above with the response as type String. The URL parameters
		// (curly bracketed in the REQUEST_URL) are defined in order at the end
		// of the parameters list. The response is put in a Response Entity.
		ResponseEntity<String> response = rstTemplate.exchange(requestURL,
				HttpMethod.GET, request, String.class, ID);
		// Print out the response as a string.
		System.out.println(response.getBody());
	}}
```

3.4 Get a List of all Outbound Files

This API call supersedes the "DEPRECATED: Get a List of Outbound Files by MID" API call found later in this document. The value of "totalResults" returned will be the total number of extracts.

URL: `https://pdx.usps.com/api/extracts`

HTTP Method: `GET`

HTTP Headers: `Accept: application/json`

HTTP Parameters:

Name	Type	Description
`environment`	String	*Optional* There are two available environments: **"TEM"** (Test Environment for Mailers) for testing purposes. **"PROD"** (Production) for accessing production data. environment defaults to both "TEM" and "PROD".
`filename`	String	*Optional* full or partial filename; select all extracts whose name matches the full or partial filename provided
`fileType`	String	*Optional* **Reference the "File Type" column Appendix 4.3 for valid options**
`fullList`	Boolean	*Optional* **"true"** – list all available extracts, regardless of download status. **"false"** – only list those extracts that were not previously downloaded If not provided, fullList defaults to "false"
`fromDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created since the value of fromDate.
`toDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created before the value of toDate.

`lastId`	Integer	*Optional* Select all extracts whose identifier is greater than the lastId value.
`mid`	String	*Optional* Select all extracts for the specified MID (must be complete MID, will not match on partial). If not provided, mid defaults to processing all MIDs associated with the user account.
`pageNumber`	Integer	*Optional* Extract responses are paginated since there can be many entries in the response. This field specifies the page number. The first page is 0.
`pageSize`	Integer	*Optional* The default is 100; the maximum page size request is 500.

Output: A JSON formatted string with information about the files.

curl Sample:

```
$ curl -u jsmith:jsmith -X GET
'https://pdx.usps.com/api/extracts?environment=TEM&fileType=eVS%20Post
age'
```

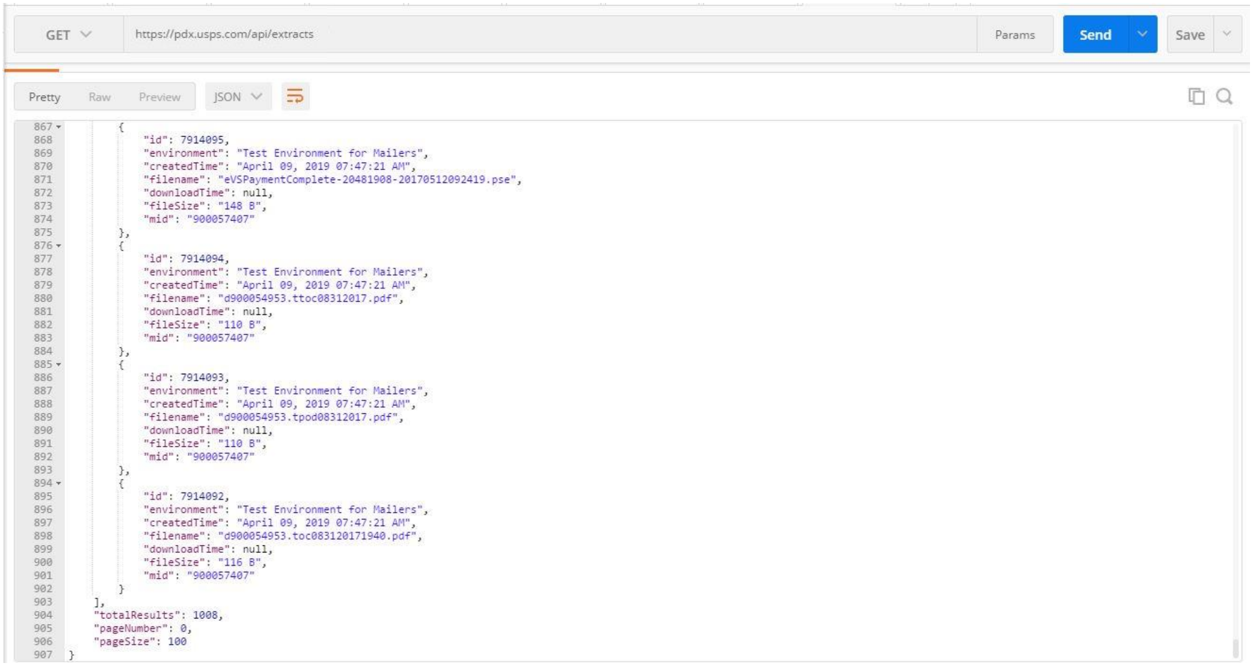
Sample response:

```
{
   "outboundFiles":[
   {"id":14059762,
   "environment":"TEST ENVIRONMENT FOR MAILERS",
   "createdTime":"July 27, 2021 06:43:46 AM",
   "filename":"eVSReconciliationExtract-901649158-20191114101325.rxt",
   "downloadTime":null,
   "fileSize":"9 KB",
   "mid":"900060230"
   },{
   "id":14059748,
   "environment":"PRODUCTION",
   "createdTime":"July 27, 2021 06:43:46 AM",
   "filename":"WKEXTR01.V15.RPT.31114101328",
   "downloadTime":"July 27, 2021 07:23:56 AM",
   "fileSize":"3 KB",
   "mid":"900060230"},
   …
   ],"totalResults":74,"pageNumber":0,"pageSize":28
}
```

Response elements:

Name	Type	Description
`id`	Integer	The PDX outbound file identifier; this value is used in an outbound file download request.
`environment`	String	The environment the outbound file was uploaded into; either “PRODUCTION” or “TEST ENVIRONMENT FOR MAILERS”.
`createdTime`	Date	The date and time (in Central Time) that the outbound file was created by internal USPS processing.
`filename`	String	The name of the outbound file.
`downloadTime`	Date	The date and time (Central Time) when the outbound file was last downloaded or “null” if it has not been downloaded.
`fileSize`	String	The size of the outbound file.
`mid`	Integer	The Mailer ID that the outbound file was uploaded into.
`totalResults`	Integer	The total number of outbound files that meet the criteria.
`pageNumber`	Integer	The number of the page requested (default: 0).
`pageSize`	Integer	The size of the page requested.

POSTMAN Sample:



https://pdx.usps.com/api/extracts

```
        {
            "id": 7914095,
            "environment": "Test Environment for Mailers",
            "createdTime": "April 09, 2019 07:47:21 AM",
            "filename": "eVSPaymentComplete-20481908-20170512092419.pse",
            "downloadTime": null,
            "fileSize": "148 B",
            "mid": "900057407"
        },
        {
            "id": 7914094,
            "environment": "Test Environment for Mailers",
            "createdTime": "April 09, 2019 07:47:21 AM",
            "filename": "d900054953.ttoc08312017.pdf",
            "downloadTime": null,
            "fileSize": "110 B",
            "mid": "900057407"
        },
        {
            "id": 7914093,
            "environment": "Test Environment for Mailers",
            "createdTime": "April 09, 2019 07:47:21 AM",
            "filename": "d900054953.tpod08312017.pdf",
            "downloadTime": null,
            "fileSize": "110 B",
            "mid": "900057407"
        },
        {
            "id": 7914092,
            "environment": "Test Environment for Mailers",
            "createdTime": "April 09, 2019 07:47:21 AM",
            "filename": "d900054953.toc083120171940.pdf",
            "downloadTime": null,
            "fileSize": "116 B",
            "mid": "900057407"
        }
    ],
    "totalResults": 1008,
    "pageNumber": 0,
    "pageSize": 100
}
```

Figure 10: POSTMAN getting a list of outbound files available for download

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

```java
import java.io.IOException;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;

/**
 * Uses Spring Rest Template to call the PDX API to view a paginated list of
 * extracts.
 */

public class ViewOutboundFiles {

	public static void main(String[] args) {
		// The response will be paginated. Use PAGE_NUMBER (maximum = 500)
		// to choose the number of outbound files returned per page and
		// PAGE_SIZE (minimum = 0) to choose which page to return.
		final int PAGE_NUMBER = 0;
		final int PAGE_SIZE = 500;

		// Set the credentials for the PDX user
```

```
        final String USERNAME = "APIUser";
        final String PASSWORD = "APIUserPassword";
        // The credentials are converted to a basic auth format.
        final String CREDENTIALS = USERNAME + ":" + PASSWORD;
        final String AUTHORIZATION = "Basic "
                     + Base64.encodeBase64String(CREDENTIALS.getBytes());
        // Set the optional field FILE_TYPE to restrict outbound files to a
        // specific type.
        final String FILE_TYPE = "BPOD";
        // This is the request URL. Optional fields may be removed as desired.
        final String REQUEST_URL = "https://pdx.usps.com/api/extracts"
                     + "?pageNumber={pageNumber}&pageSize={pageSize}"
                     + "&fileType={fileType}";
        // Declare the Rest Template variable.
        RestTemplate = new RestTemplate();
        // Set HTTP headers
        // Declare the HTTP Headers variable.
        HttpHeaders headers = new HttpHeaders();
        // Set the basic auth user credentials.
        headers.set("Authorization", AUTHORIZATION);
        // Set the acceptType as JSON
        headers.set("Accept", "application/json");
        // Create the request with the headers set above.
        HttpEntity<String> request = new HttpEntity<String>(headers);
        // Make an API call to the REQUEST_URL as a GET method, using the
        // request above and the response as type String. The URL parameters
        // (curly bracketed in the REQUEST_URL) are defined in order at the end
        // of the parameters list. The response is put in a Response Entity.
        ResponseEntity<String> response = restTemplate.exchange(REQUEST_URL,
                     HttpMethod.GET, request, String.class, PAGE_NUMBER,
                     PAGE_SIZE, FILE_TYPE);
        // Print out the response as a string.
        System.out.println(response.getBody());
}}
```

3.5 Get an Outbound File

URL:	`https://pdx.usps.com/api/outbound-files/{id}` Replace `{id}` with the ID of the outbound file. This ID can be found in the list of outbound files, which can be retrieved as described in Section 3.4 Get a List of Outbound Files. Each time an extract is downloaded the count of downloads is increased.
HTTP Method:	`GET`
HTTP Headers:	`Accept: application/octet-stream`
HTTP Parameters:	None
Output:	The requested file is returned as a stream of data. An HTTP 304 Not Modified status will be returned if the same extract is downloaded more than 20 times. The 304 status indicates that there is no need to retransmit the data since the client has a previously downloaded copy. If necessary, files can be downloaded via the UI, regardless of the number of times previously downloaded.

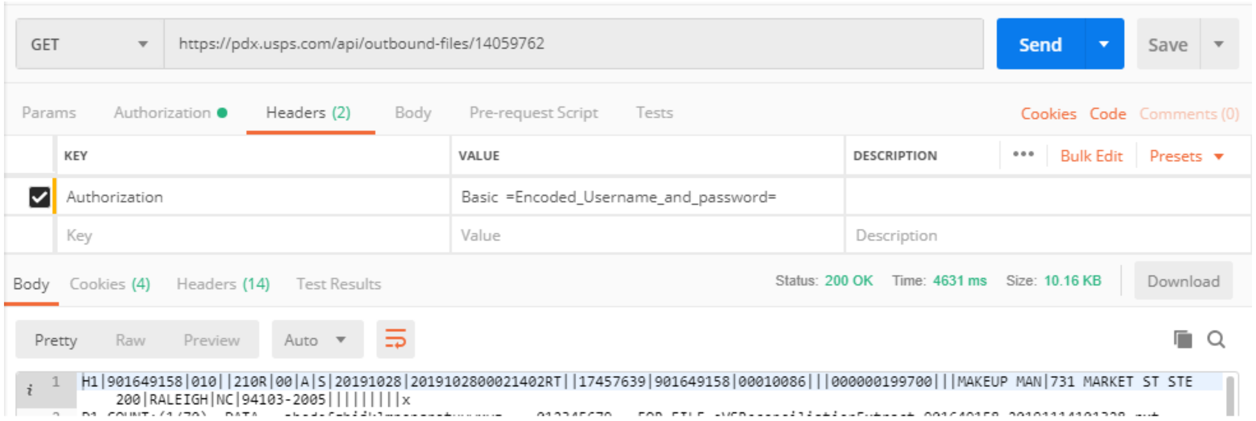
curl Sample:

```
$ curl -u jsmith:jsmith -X GET 'https://pdx.usps.com/api/outbound-files/2'
```

Sample response:

The data returned will be the contents of the extract.

POSTMAN Sample:



https://pdx.usps.com/api/outbound-files/14059762

KEY	VALUE	DESCRIPTION
✓ Authorization	Basic =Encoded_Username_and_password=	
Key	Value	Description

Status: 200 OK Time: 4631 ms Size: 10.16 KB

```
H1|901649158|010||210R|00|A|S|20191028|2019102800021402RT||17457639|901649158|00010086|||000000199700|||MAKEUP MAN|731 MARKET ST STE
    200|RALEIGH|NC|94103-2005|||||||||x
```

Figure 11: POSTMAN getting an outbound file

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

```java
import java.io.IOException;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;
/**
 * Uses Spring Rest Template to call the PDX API to download an extract by a
specified extract Id
 */
public class DownloadOutboundFiles {

	public static void main(String[] args) {
		// Set the ID of the file to be downloaded. You can find the ID by using
		// the View Outbound Files API.
		final int ID = 33562;
		// Set the credentials for the user whose outbound file you would like
		// to download.
		final String USERNAME = "APIUser";
		final String PASSWORD = "APIUserPassword";
		// The credentials are converted to a basic auth format.
		final String CREDENTIALS = USERNAME + ":" + PASSWORD;
		final String AUTHORIZATION = "Basic "
				+ Base64.encodeBase64String(CREDENTIALS.getBytes());
		// Set the MID to which the file belongs.
		final int MID = 123456789;
		// This is the request URL.
		final String REQUEST_URL = "https://pdx.usps.com/api/outbound-
		files/{id}";
		// Declare the Rest Template as a variable.
		RestTemplate = new RestTemplate();
		// Set HTTP headers
		// Declare the HTTP Headers variable.
		HttpHeaders headers = new HttpHeaders();
		// Set the basic auth user credentials.
		headers.set("Authorization", AUTHORIZATION);
		// Set the acceptType as octet-stream
		headers.set("Accept", "application/octet-stream");
		// Create the request with the headers set above.
		HttpEntity<String> request = new HttpEntity<String>(headers);
		// Make an API call to the REQUEST_URL as a GET method, using the
		// request
		// set above, and with the response as type String. The URL parameters
		// (curly bracketed in the REQUEST_URL) are defined in order at the end
		// of the parameters list. The response is put in a Response Entity.
		ResponseEntity<String> response = restTemplate.exchange(REQUEST_URL,
				HttpMethod.GET, request, String.class, ID);
		// Print out the response as a string.
		System.out.println(response.getBody());
	}
}
```

3.6 Get Multiple Outbound Files as a ZIP

This API call supersedes the "DEPRECATED: Get Multiple Outbound Files into a Zip Folder" API call found later in this document. If the same parameters are used in the ExtractsList and ZIP API calls, the resultant files should be the same. However, new extracts may be received or existing extracts may change state (not downloaded → downloaded), causing results to differ. If the ZIP limits are exceeded (total file size or the number of extracts), reduction in contents can be achieved by making multiple calls using fullList=false, pageNumber=0, pageSize=100 until all extracts are downloaded.

URL: `https://pdx.usps.com/api/extracts/zip`

HTTP Method: `GET`

HTTP Headers: `Accept: application/json`

HTTP Parameters:

Name	Type	Description
`environment`	String	*Optional* There are two available environments: **"TEM"** (Test Environment for Mailers) for testing purposes. **"PROD"** (Production) for accessing production data. environment defaults to both "TEM" and "PROD".
`filename`	String	*Optional* full or partial filename; select all extracts whose name matches the full or partial filename provided
`fileType`	String	*Optional* **Reference the "File Type" column Appendix 4.3 for valid options**
`fullList`	Boolean	*Optional* **"true"** – list all available extracts, regardless of download status. **"false"** – only list those extracts that were not previously downloaded If not provided, fullList defaults to "false"
`fromDate`	Date	*Optional*

		"MM-DD-YYYY" formatted date; select all extracts created since the value of fromDate.
`toDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created before the value of toDate.
`lastId`	Integer	*Optional* Select all extracts whose identifier is greater than the lastId value.
`mid`	String	*Optional* Select all extracts for the specified MID (must be complete MID, will not match on partial). If not provided, mid defaults to processing all MIDs associated with the user account.
`pageNumber`	Integer	*Optional* Extract responses are paginated since there can be many entries in the response. This field specifies the page number. The first page is 0.
`pageSize`	Integer	*Optional* The default is 100; the maximum page size request is 500.

Output: The requested outbound files are returned as a binary stream representation of zip data.

NOTE:
Creating Zip files is inherently slow since the processing and compression of many files is required. Simultaneous Zip requests will be denied and an error message similar to the following will be displayed:

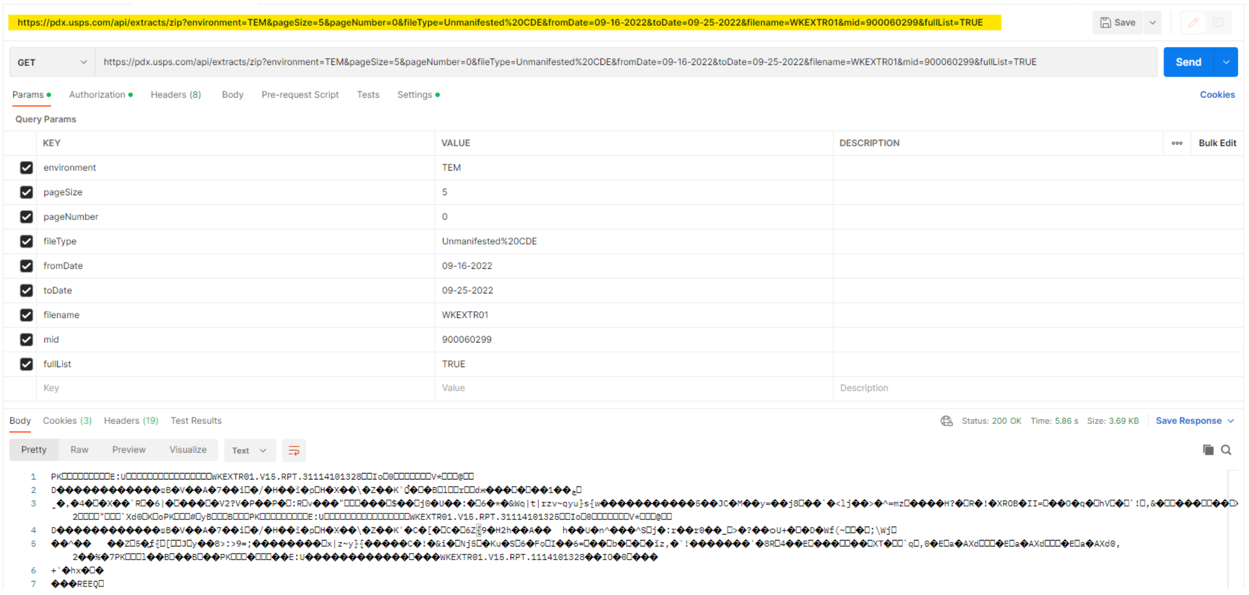
A zip is still in progress. Please try later. Last started at: MM/DD/YYYY HH:MI:SS

If the Zip fails to complete (e.g. receiving disk is full/write failure) new Zip requests will immediately fail for up to 10 minutes (the restriction is automatically cleared after 10 minutes); new Zip requests will then be accepted again.

curl Sample:

```
$ curl -X GET
'https://pdx.usps.com/api/extracts/zip?fileType=Unmanifested%20CDE&pag
eNumber=0&pageSize=7&environment=TEM&fromDate=09-16-2022&toDate=09-25-
2022&filename=WKEXTR01&mid=900060299&fullList=TRUE'
```

POSTMAN Sample:



https://pdx.usps.com/api/extracts/zip?environment=TEM&pageSize=5&pageNumber=0&fileType=Unmanifested%20CDE&fromDate=09-16-2022&toDate=09-25-2022&filename=WKEXTR01&mid=900060299&fullList=TRUE

GET https://pdx.usps.com/api/extracts/zip?environment=TEM&pageSize=5&pageNumber=0&fileType=Unmanifested%20CDE&fromDate=09-16-2022&toDate=09-25-2022&filename=WKEXTR01&mid=900060299&fullList=TRUE

	KEY	VALUE	DESCRIPTION
✓	environment	TEM	
✓	pageSize	5	
✓	pageNumber	0	
✓	fileType	Unmanifested%20CDE	
✓	fromDate	09-16-2022	
✓	toDate	09-25-2022	
✓	filename	WKEXTR01	
✓	mid	900060299	
✓	fullList	TRUE	
	Key	Value	Description

Status: 200 OK Time: 5.86 s Size: 3.69 KB

```
PK□□□□□□□□□□E:U□□□□□□□□□□□□□□□□□WKEXTR01.V15.RPT.31114101328□□Io□0□□□□□□□□V*□□□@□□
D��������������B�V��A�7��i□�/�H��ï�p□H�X��\�Z��K'Ç�□�B□l□□r□□dж����□�□���1��ₔ□
_�,�4�□�X��`R□�6|�□����□�V2?V�P��P�□:R□v���"□□□����□$��□j0�U��:�□6�*�&Wq|t|rzv~qyu}s{w�������������5��JC�M��y=��j8□��`�<lj��>�^=mz□����H?�□R�!�XROB�II=□���O�q�□hV□�□`!□,&�□□����□□���□
    2□□□□"□□□`Xd0□K□oPK□□□#□yB□□□B□□□PK□□□□□□□□□□E:U□□□□□□□□□□□□□□□□□WKEXTR01.V15.RPT.31114101325□□Io□0□□□□□□□□V*□□□@□□
D��������������B�V��A�7��i□�/�H��ï�p□H�X��\�Z��K'�C�[�□C�□6Z{9�H2h��A��  h��U�n^���^S□j�:r��r0��_□>�?��oU+�□□D�Wf(~□□�□;\Wj□
��^��   ��Z□5�f{□[□□J□y��8>:>9=;���������□x|z~y}{�����C�!�&i�□Nj5□�Ku�S□6�Fo□I��6=□���□b�□�□�ĭz,�'!��������'�8R□4��E□�����□□���□XT�□□`q□,0�E□a�AXd□□□�E□a�AXd□□□�E□a�AXd0,
    2��%�7PK□□□l��B□��B□��PK□□□�□□□��E:U�������������□���WKEXTR01.V15.RPT.1114101328��IO�0□���
+'�hx�□�
���REEQ□
```

Figure 12: POSTMAN getting a ZIP of outbound files

3.7 DEPRECATED: Get Multiple Outbound Files into a Zip Folder

NOTE: This API call is deprecated. Please refer to the preferred API call in section "3.6 Get Multiple Outbound Files as a ZIP". This API call does not match List of Extract parameters, and results cannot be easily adjusted if the maximum file size or count is exceeded. The size and number of files that can be downloaded are limited. See Section 4.2 Error Messages for details on these zip limits.

URL: `https://pdx.usps.com/api/mids/{mid}/outbound-files.zip`

Replace `{mid}` with the MID of interest. This call returns all outbound files created since the 'fromDate' date indicated as a parameter in a zip folder. Each time an extract is downloaded the count of downloads is increased.

HTTP Method: `GET`

HTTP Headers: `Accept: application/zip`

HTTP Parameters:

Name	**Type**	**Description**
`environment`	String	This field can refer to one of the two supported environments: **"TEM"** This is used for non-production data for testing purposes. It refers to "Test Environment for Mailers". **"PROD"** This is used for production data. It refers to the "Production" environment.
`fromDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created since the value of fromDate.
`fileType`	String	*Optional* **Reference the "File Type" column Appendix 4.3 for valid options**
`filename`	String	*Optional* Full or partial filename (case insensitive) of the desired extract(s)
`notDownloaded`	Boolean	*Optional* **"true"** – All files, whether or not previously downloaded, will be returned in the zip data. **"false"** – Only files not previously downloaded will be returned in the zip data.

		If not provided, notDownloaded defaults to "true".
`toDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created after the value of fromDate and before the value of toDate.

Output: The requested outbound files are returned as a binary stream representation of zip data.

NOTE:
Creating Zip files is inherently slow since the processing and compression of many files is required. Simultaneous Zip requests will be denied and an error message similar to the following will be displayed:

A zip is still in progress. Please try later. Last started at: MM/DD/YYYY HH:MI:SS

If the Zip fails to complete (e.g. disk write failure) new Zip requests will continue to fail for up to 10 minutes (the restriction is automatically cleared after 10 minutes); Zip requests will then be accepted again.

curl Sample:

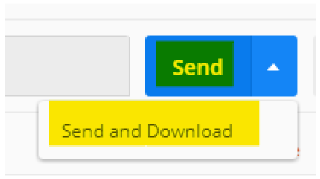
```
$ curl -u jsmith:jsmith -X GET
'https://pdx.usps.com/api/mids/901017163/outbound-
files.zip?environment=PROD&fromDate=02-01-2017&fileType=Other' > file.zip
```

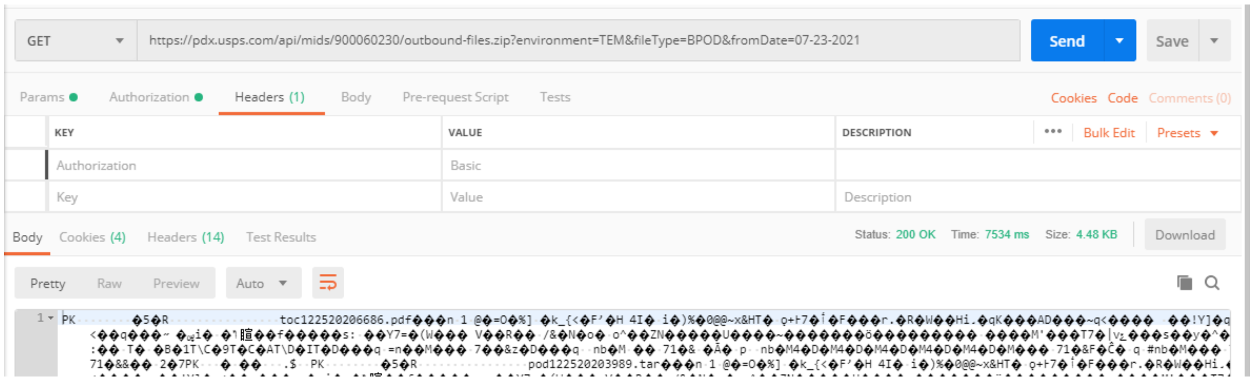
Sample response:

The data returned will be the contents of the zip file (those extracts selected by the filters provided).

POSTMAN Sample:

NOTE: Click on the down arrow next to Send and select "Send and Download". It will prompt to choose a location/filename of the ZIP to be saved:





GET https://pdx.usps.com/api/mids/900060230/outbound-files.zip?environment=TEM&fileType=BPOD&fromDate=07-23-2021

KEY	VALUE	DESCRIPTION
Authorization	Basic	
Key	Value	Description

Status: 200 OK Time: 7534 ms Size: 4.48 KB

```
PK        �5�R              toc122520206686.pdf���n 1 @�=O�%] �k_{<�F'�H 4I� i�)%�0@@~x&HT� ǫ+Ͱ7�ǐ�F���r.�R�W��Hi.�qK���AD���~q<����   ��!Y]�q
  <��q���~ ��i� �ȉ瞔��f�����s: ��Y7=�(W��� V��R�� /&�N�o� o^��ZN�����U����~��������ö�������� ����M'���T7�|v��s��y�^�
  :�� T� �B�1T\C�9T�C�AT\D�IT�D���q =n��M��� 7��&z�D���q  nb�M �� 71�& �Ã� p  nb�M4�D�M4�D�M4�D�M4�D�M4�D�M��� 71�&F�Ĉ� q #nb�M���
  71�&&�� 2�7PK   � ��   .$  PK        �5�R              pod122520203989.tar���n 1 @�=O�%] �k_{<�F'�H 4I� i�)%�0@@~x&HT� ǫ+Ͱ7�ǐ�F���r.�R�W��Hi.
```

Figure 13: POSTMAN getting multiple outbound files in a ZIP folder

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

```java
import java.io.IOException;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;

/**
 * Uses Spring Rest Template to call the PDX API to bulk download outbound files
 * from a specified MID, created after fromDate and with a specified fileType.
 */

public class BulkDownloadOutboundFiles {

	public static void main(String[] args) {
		// Set the credentials for the user whose outbound files you would like
		// to bulk download.
		final String USERNAME = "APIUser";
		final String PASSWORD = "APIUserPassword";
		// The credentials are converted to a basic auth format.
		final String CREDENTIALS = USERNAME + ":" + PASSWORD;
		final String AUTHORIZATION = "Basic "
				+ Base64.encodeBase64String(CREDENTIALS.getBytes());
		// Set the MID to which the file belongs.
		final int MID = 123456789;
		// Choose the environment you would like to download files from.
		// The options are "TEM" or "PROD".
		final String ENVIRONMENT = "TEM";
		// Chose the FROM_DATE. All files created after this date will
		// be bulk downloaded. This date cannot be more than 45 days
```

```
                // before today's date.
                final String FROM_DATE = "07-12-2017";
                // Only files of the file type defined as FILE_TYPE will be
                // downloaded. If you would like to download all file types,
                // please remove this field from the request.
                final String FILE_TYPE = "BPOD";
                // This is the request URL.
                final String REQUEST_URL = "https://pdx.usps.com/api/mids"
                        + "/{mid}/outbound-files.zip?environment={environment}"
                        + "&fromDate={fromDate}&fileType={fileType}";
                // Declare the Rest Template as a variable.
                RestTemplate = new RestTemplate();
                // Set HTTP headers
                // Declare the HTTP Headers variable.
                HttpHeaders headers = new HttpHeaders();
                // Set the basic auth user credentials.
                headers.set("Authorization", AUTHORIZATION);
                // Set the acceptType as zip.
                headers.set("Accept", "application/zip");
                // Create the request with the headers set above.
                HttpEntity<String> request = new HttpEntity<String>(headers);
                // Make an API call to the REQUEST_URL as a GET method, using the
                // request
                // set above, and with the response as type String. The URL parameters
                // (curly bracketed in the REQUEST_URL) are defined in order at the end
                // of the parameters list. The response is put in a Response Entity.
                ResponseEntity<String> response = restTemplate.exchange(REQUEST_URL,
                        HttpMethod.GET, request, String.class, MID, ENVIRONMENT,
                        FROM_DATE, FILE_TYPE);
                // Print out the response as a string.
                System.out.println(response.getBody());
}}
```

3.8 DEPRECATED: Get a List of Outbound Files by MID

NOTE: This API call is deprecated. Please refer to the preferred API call in section "3.4 Get a List of all Outbound Files" for the API call to retrieve extracts for all MIDs

Do not forget the hyphen between "outbound" and "files" (i.e. "outbound-files").

URL: `https://pdx.usps.com/api/mids/{mid}/outbound-files`

Replace `{mid}` with the Mailer ID (MID) of interest.

HTTP Method: `GET`

HTTP Headers: `Accept: application/json`

HTTP Parameters:

Name	**Type**	**Description**
`environment`	String	This field can refer to one of the two supported environments: **"TEM"** This is used for non-production data for testing purposes. It refers to "Test Environment for Mailers". **"PROD"** This is used for production data. It refers to the "Production" environment.
`filename`	String	*Optional* full or partial filename; select all extracts whose name matches the full or partial filename provided
`fileType`	String	*Optional* **Reference the "File Type" column Appendix 4.3 for valid options**
`fromDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created since the value of fromDate.
`toDate`	Date	*Optional* "MM-DD-YYYY" formatted date; select all extracts created before the value of toDate.
`lastId`	Integer	*Optional*

		Select all extracts whose identifier is greater than the lastId value.
`notDownloaded`	Boolean	*Optional* **“true”** – Only files not previously downloaded will be returned in the generated list of outbound files. **“false”** – All files, whether or not previously downloaded, will be returned in the generated list of outbound files.
`pageNumber`	Integer	*Optional* The output may contain many extracts so the response is paginated. This field specifies the page number of the pagination. The first page is page 0.
`pageSize`	Integer	*Optional* The default is 100; the maximum page size request is 500.
`sumCount`	Boolean	*Optional* **“true”** – The value of “totalResults” returned will be the total number of extracts. **“false”** – The value of “totalResults” returned will be the value of pageSize.

Output: A JSON formatted string with information about the files.

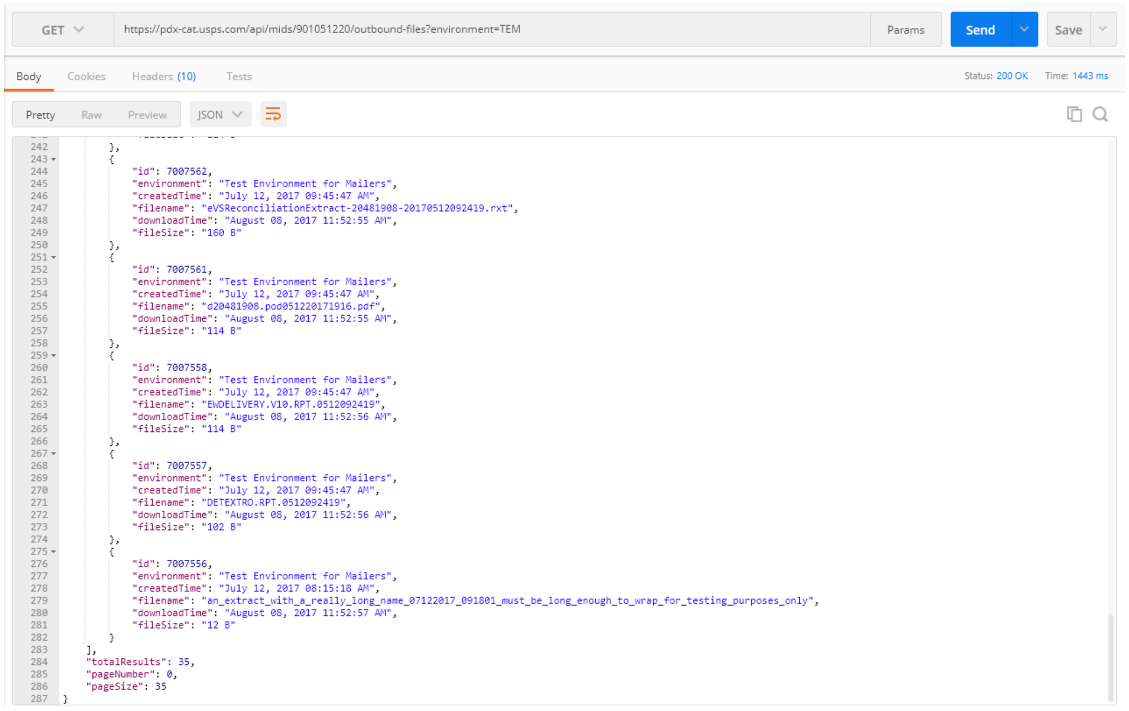
curl Sample:

```
$ curl -u jsmith:jsmith -X GET
'https://pdx.usps.com/api/mids/440149001/outbound-
files?environment=TEM&notDownloaded=false&fileType=Other'
```

Sample response:

```
{
  "outboundFiles": [
    {
      "id": 200,
      "environment": "Test Environment for Mailers",
      "filename": "extract_200",
      "createdTime": "April 29, 2017 12:42:12 PM",
      "downloadTime": "June 03, 2017 09:27:35 AM",
      "fileSize": "255 KB"
    },
    {
      "id": 199,
      "environment": "Test Environment for Mailers",
      "filename": "extract_199",
      "createdTime": "April 29, 2017 12:40:12 PM",
      "downloadTime": "June 03, 2017 09:27:40 AM"
      "fileSize": "1 MB"
    }
}
```

POSTMAN Sample:



```
https://pdx-cat.usps.com/api/mids/901051220/outbound-files?environment=TEM
```

```
        },
        {
            "id": 7007562,
            "environment": "Test Environment for Mailers",
            "createdTime": "July 12, 2017 09:45:47 AM",
            "filename": "eVSReconciliationExtract-20481908-20170512092419.rxt",
            "downloadTime": "August 08, 2017 11:52:55 AM",
            "fileSize": "160 B"
        },
        {
            "id": 7007561,
            "environment": "Test Environment for Mailers",
            "createdTime": "July 12, 2017 09:45:47 AM",
            "filename": "d20481908.pod051220171916.pdf",
            "downloadTime": "August 08, 2017 11:52:55 AM",
            "fileSize": "114 B"
        },
        {
            "id": 7007558,
            "environment": "Test Environment for Mailers",
            "createdTime": "July 12, 2017 09:45:47 AM",
            "filename": "EWDELIVERY.V10.RPT.0512092419",
            "downloadTime": "August 08, 2017 11:52:56 AM",
            "fileSize": "114 B"
        },
        {
            "id": 7007557,
            "environment": "Test Environment for Mailers",
            "createdTime": "July 12, 2017 09:45:47 AM",
            "filename": "DETEXTRO.RPT.0512092419",
            "downloadTime": "August 08, 2017 11:52:56 AM",
            "fileSize": "102 B"
        },
        {
            "id": 7007556,
            "environment": "Test Environment for Mailers",
            "createdTime": "July 12, 2017 08:15:18 AM",
            "filename": "an_extract_with_a_really_long_name_07122017_091801_must_be_long_enough_to_wrap_for_testing_purposes_only",
            "downloadTime": "August 08, 2017 11:52:57 AM",
            "fileSize": "12 B"
        }
    ],
    "totalResults": 35,
    "pageNumber": 0,
    "pageSize": 35
}
```

Figure 14: POSTMAN getting a list of outbound files available for download

Java Sample:

The libraries needed for this sample can be downloaded automatically by using Maven. The accompanying pom.xml file is shown in Appendix A.

```
import java.io.IOException;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;
import org.apache.commons.codec.EncoderException;
import org.apache.commons.codec.binary.Base64;

/**
 * Uses Spring Rest Template to call the PDX API to view a paginated list of
 * extracts for a specific MID. Can specify page number, page size, environment,
 * download status and file type.
 */

public class ViewOutboundFiles {
```

```
public static void main(String[] args) {
        // The response will be paginated. Use PAGE_SIZE to choose the number of
        // outbound files returned per page and PAGE_NUMBER to choose which page
        // to return.
        final int PAGE_NUMBER = 0;
        final int PAGE_SIZE = 500;
        // Choose which environment to return outbound files from. Options are
        // "TEM" and "PROD".
        final String ENVIRONMENT = "TEM";
        // Set the credentials for the PDX user
        final String USERNAME = "APIUser";
        final String PASSWORD = "APIUserPassword";
        // The credentials are converted to a basic auth format.
        final String CREDENTIALS = USERNAME + ":" + PASSWORD;
        final String AUTHORIZATION = "Basic "
                        + Base64.encodeBase64String(CREDENTIALS.getBytes());
        // Choose which of the above user's MIDs you would like to
        // view outbound files for.
        final int MID = 123456789;
        // Set the optional field NOT_DOWNLOADED to "true" if you would like to
        // view only files not previously downloaded. If you would like to view
        // all files regardless of download status, either set this field to
        // false or remove the parameter from the URL.
        final String NOT_DOWNLOADED = "true";
        // Set the optional field FILE_TYPE if you would like to view only
        // outbound files of a specific type. Otherwise, remove this parameter
        // from the URL.
        final String FILE_TYPE = "BPOD";
        // This is the request URL. Optional fields notDownloaded and fileType
        // may be removed as desired.
        final String REQUEST_URL =
        "https://pdx.usps.com/api/mids/{mid}/outbound-files"
                        + "?pageNumber={pageNumber}&pageSize={pageSize}"
                        +
                        "&environment={environment}&notDownloaded={notDownloaded}"
                        + "&fileType={fileType}";
        // Declare the Rest Template variable.
        RestTemplate = new RestTemplate();
        // Set HTTP headers
        // Declare the HTTP Headers variable.
        HttpHeaders headers = new HttpHeaders();
        // Set the basic auth user credentials.
        headers.set("Authorization", AUTHORIZATION);
        // Set the acceptType as JSON
        headers.set("Accept", "application/json");
        // Create the request with the headers set above.
        HttpEntity<String> request = new HttpEntity<String>(headers);
        // Make an API call to the REQUEST_URL as a GET method, using the
        // request above and the response as type String. The URL parameters
        // (curly bracketed in the REQUEST_URL) are defined in order at the end
        // of the parameters list. The response is put in a Response Entity.
        ResponseEntity<String> response = restTemplate.exchange(REQUEST_URL,
                        HttpMethod.GET, request, String.class, MID, PAGE_NUMBER,
                        PAGE_SIZE, ENVIRONMENT, NOT_DOWNLOADED, FILE_TYPE);
```

```
            // Print out the response as a string.
            System.out.println(response.getBody());
    }}
```

4. Appendix

4.1 POM.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.usps.pdx.api</groupId>
	<artifactId>client</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>
	<name>client</name>

		<dependencies>
			<dependency>
				<groupId>org.springframework</groupId>
				<artifactId>spring-web</artifactId>
				<version>${org.springframework.version}</version>
			</dependency>
			<dependency>
				<groupId>ch.qos.logback</groupId>
				<artifactId>logback-classic</artifactId>
				<version>${ch.qos.logback.version}</version>
			</dependency>
			<dependency>
				<groupId>org.apache.httpcomponents</groupId>
				<artifactId>httpclient-cache</artifactId>
				<version>${httpcomponents-client.version}</version>
				<exclusions>
					<exclusion>
						<groupId>commons-logging</groupId>
						<artifactId>commons-logging</artifactId>
					</exclusion>
				</exclusions>
			</dependency>
			<dependency>
				<groupId>commons-collections</groupId>
				<artifactId>commons-collections</artifactId>
				<version>${commons-collections.version}</version>
			</dependency>
			</dependencies>

	<properties>
		<build.java.version>1.8</build.java.version>
		<browser>chrome</browser>
		<ch.qos.logback.version>1.2.5</ch.qos.logback.version>
		<commons-collections.version>3.2.1</commons-collections.version>
		<httpcomponents-client.version>4.5.13</httpcomponents-client.version>
		<org.springframework.version>5.3.9</org.springframework.version>
	</properties>
</project>
```

4.2 Error Messages

Authentication

PDX interfaces with the Business Customer Gateway (BCG) for API user authentication (username and password). Access the gateway via https://gateway.usps.com. The most common errors are:

Scenario	API Error Message
The user account has been deactivated.	Error 401: API Access Denied. Your account has been temporarily deactivated. Please contact the PDX helpdesk at DTS-PDX@usps.gov for assistance.
The user or password is invalid.	Error 401: Invalid username or password
The username does not exist.	Error 401: User Alias Not Found
The user/password combination is invalid. X attempts remain.	Invalid User Id or Password. You have X of 3 remaining login attempts.
The user/password combination was invalid 3 or more times. The next state will be "locked".	Maximum Invalid Passwords Exceeded.
The authentication was successful, but the account is locked because 3 or more invalid login attempts were made within the last 24 hours.	Your account has been locked.
The account has been disabled and authentication is not allowed.	Your account has been disabled.
PDX is unable to authenticate the user/password; the authentication system is not responding.	Error 401: Server maintenance.

PDX Unavailable

If the PDX application is unavailable, retry later.

Scenario	API Error Message
The PDX application is unavailable during maintenance or other planned down time.	Service Unavailable.

Manifest Upload

The following are PDX API/UI errors displayed during manifest upload failures. Allowable sizes are subject to change. Currently, the largest size a manifest can be is 250MB and the filename must be 88 characters or less in length. Minimally, manifests must have header and details information.

Scenario	API Error Message
The manifest had no contents. Correct content and resubmit.	Error 417: File is EMPTY!
The manifest is too small. Correct content and resubmit.	Error 412: File size below allowed size of {minimum size}
The manifest is too large. Correct content and resubmit.	Error 413: File exceeds maximum allowed size of {maximum size}
The extension on the manifest was invalid. The list of valid extensions (e.g. ".manifest") is details in the Section: Upload a Manifest File. Correct the extension and resubmit.	Error 417: File Extension not valid
The filename is too long. Shorten the filename and resubmit.	Error 413: File name exceeds maximum allowed number of char in the file name size of {size} digits
The file is encoded (e.g. UTF-8-BOM). Files must be ASCII. Encoded files are rejected at upload time. Save as UTF-8 or ANSI and resubmit.	Error 415: Wrong encoding of the file. Char at position: X is Y=Z
Processing is not provided for TEM .subscribe manifests so they are rejected at upload time.	Error 415: File upload rejected. Cause: subscribe file type (extension); not accepted in TEM

API Parameter Validation Errors

The following are PDX API errors displayed for parameter issues.

Parameter	Scenario	API Error Message
environment	environment = null	No environment specified, please modify this criteria and try again.
environment	environment != TEM, environment != PROD	The environment specified is invalid, please select either TEM or PROD as the value for the environment parameter.
fileType	fileType = null	No fileType specified, please modify this criteria and try again.
fileType	fileType = invalid fileType	The value for the fileType parameter you have specified is invalid, please modify this criteria and try again.
filename	filename length > 88 characters	File name exceeds maximum allowed number of char in the filename size of 88 chars
fromDate/toDate	fromDate = null	No {fromDate/toDate} specified, please modify this criteria and try again.
fromDate	fromDate > 45 days from today	The specified fromDate parameter is past the 45 day limit, please modify this criteria and try again. -or- fromDate {date} cannot be older than 45 days. Please verify your request and try again.
fromDate/toDate	fromDate/toDate > today	Date cannot be in the future.
fromDate/toDate	invalid date ranges	Cannot parse '{date}': Value {month} for monthOfYear must be in the range [1,12] Cannot parse '{date}': Value {day} for dayOfMonth must be in the range [1,31] Date cannot be before 2017
fromDate/toDate	invalid format	Invalid format: '{invalid date}' -or- Invalid format: '{invalid date}' is malformed at '{invalid portion}'
fromDate/toDate	out of sequence	The specified toDate parameter is less than the fromDate parameter, please modify these criteria and try again.
id	id = (extract or manifest) does not belong to the user	Forbidden
id	id = string	Unrecognized {manifest/extract} id, please specify an Integer value for the id parameter and try again.
id	id not in database	Unable to find specified {manifest/extract} id, please modify this criteria and try again.
id	id = valid, but out of 45 day range	Unable to download {manifest/extract} file with id:{id} because this file is older than 45 days.
lastId	lastId = null	No lastId specified, please modify this criteria and try again.
lastId	lastID = invalid identifier	lastId with value {invalid} is invalid. Expected numeric value. Please verify your request and try again.
MID	MID = not accessible by the user	Your account does not have access to the mid {mid}
MID	MID = String, MID = not found	The specified MID is not recognized, please modify this criteria and try again. –or—The MID is not a number, please modify this criteria and try again.
multipartFile	multipartFile = null	Please select a file to upload
multipartFile	manifest file extension = invalid	File Extension Not valid
notDownloaded	notDownloaded = null	No notDownloaded specified, please modify this criteria and try again.
notDownloaded	notDownloaded = invalid	Parameter notDownloaded is not set to [true|false]. The notDownloaded was set to:{value}
pageNumber	pageNumber = null	No pageNumber specified, please modify this criteria and try again.
pageNumber	pageNumber = String	Please specify an Integer value for the pageNumber parameter greater than or equal to 0.
pageSize	pageSize = null	No pageSize specified, please modify this criteria and try again.

pageSize	pageSize = String	Please specify an Integer between 1 and 500 for the pageSize parameter.
sumCount	sumCount = null	No sumCount specified, please modify this criteria and try again.
sumCount	sumCount is not ‘true’ or ‘false’	Parameter sumCount parameter is not set to [true|false]. The sumCount was set to:{value}
toDate	content/value is missing	Date cannot be empty.
ZIP results	Too many files	Total count of Selected Files exceeds the Max ZIP File Count Limit {limit (currently 1,000)}
ZIP results	Too much data	Total byte size of Selected Files Exceeds the Max ZIP Bytes Size Limit {limit (currently 1,000,000,000 bytes)}
ZIP results	No files	No records for criteria
ZIP requests	Attempting multiple simultaneous Zip requests	Error 429: A zip is still in progress. Please try later. Last started at: MM/DD/YYYY HH:MI:SS
Invalid	Invalid parameter(s)	Parameters that are wrong:{name=value}

4.3 Customer Outbound File Types

NOTE: When using the fileType parameter in API calls, it may be necessary to insert special characters instead of whitespace (e.g. “Customer%20Daily%20Extract”) to preserve functionality. Check with the documentation of the REST API used (e.g. curl, POSTMAN, java, etc.) for the correct format to specify special characters.

File Type	Pattern	Examples
Customer Daily Extract	DETEXTRO.%	DETEXTRO.V15.RPT.0630160158
Delivery Partner CEW	EWDELIVERY.%	EWDELIVERY.V10.RPT.0804154956
eVS Postage	eVSPayment%.pse	eVSPaymentComplete-901278585-20140630161255.pse
eVS Reconciliation	eVSReconciliation%.rxt	eVSReconciliationExtract-901352737-20140901040544.rxt
Firm Sheet	FRMEXTRO1.%	FRMEXTRO1.RPT.0226113000
Shipping Partner CEW	ERRWRNO1.%	ERRWRNO1.V20.1217002622
Shipping Services CEW	ERRWRNO.%	ERRWRNO.V15.RPT.0701124626, ERRWRNO.RPT.0630155249, ERRWRNO.V17.RPT.0630160619
Subscription Scan CEW	EWSUBSCRIBE.%	EWSUBSCRIBE.V10.RPT.0730162735
Subscription Scan	SUBEXTRO.%	SUBEXTRO.V10.RPT.0730170643
Unmanifested CDE	WKEXTR01.%	WKEXTR01.V15.RPT.0819075737, WKEXTR01.RPT.0930080621
BPOD	toc%	toc050415.pdf
BPOD	%.toc%	d901560805.toc021516.pdf
BPOD	%.pod%	d901560805.pod0215160001.pdf, d901818616.pod0907160001.tar
BPOD	pod%	pod0504150001.pdf
Subscription Scan Event	GSXUSPS.%	GSXUSPS.V01.RPT.911111a.0628161320, GSXUSPS.V01.RPT.mmgg11.0908160002
Other	all other file types	eVSPaymentComplete-999999999-20180206081048.pse.zip, eVSReconciliationExtract-999999999-20180201061735.rxt.ZIP
TPOD	ttoc%	ttoc0504150001.pdf
TPOD	tpod%	tpod0504150001.pdf
TPOD	%.ttoc%	d901560805.ttoc021516.pdf
TPOD	%.tpod%	d901560805.tpod0215160001.pdf, d901818616.tpod0907160001.tar
USPS Ship	%_PPC_%	20230118080248_30198397_PPC_FA_EX_LUPV.txt
Enterprise Payment System	%_EPS_%	20230118080252_51784251_EPS_MC_LA_EWSX.txt
USPS Ship PEW	NONE	92750900060286152833394-15564732545528557848326344463593128 76.txt

4.4 Manifest File naming

Reserved characters found in manifest filenames will be converted by PDX to underscore (_) characters. The following reserved characters will be converted:

1. Pipe (|)
2. Percent sign (%)
3. Dollar sign ($)
4. Greater than (>)
5. Less than (<)
6. Ampersand (&)
7. Single quote (‘)

For example, a manifest file named **Sprockets&01%06082022083100.manifest** will be renamed **Sprockets_01_06082022083100.manifest** once uploaded to PDX. Manifest filenames should be 88 or less alphanumeric characters. Manifest filenames should be unique (to assist in possible debugging).

4.5 .NET Code Sample for API File Upload

```
public class PDXUploadAFileRequest
      {
         public string filename { get; set; }
         public string environment { get; set; }
         public string multipartFile { get; set; }
      }

      public static bool SendFileUsingPDXTransfer(ShippingOrigin so, string filepath, string filename)
      {
         try
         {
            var url = "https://pdx.usps.com/api/manifests";
            var request = (HttpWebRequest)WebRequest.Create(url);
            request.Method = "POST";

            var credentialCache = new CredentialCache();
            var username = so.GetValueForSOCarrierConstant("APIUser");
            var password = so.GetValueForSOCarrierConstant("APIUserPassword");
            credentialCache.Add(new Uri(url), "Basic", new NetworkCredential(username, password));
            request.Credentials = credentialCache;
            request.PreAuthenticate = true;
            request.Accept = "application/json";
            request.AllowWriteStreamBuffering = true;

            var pdxData = new PDXUploadAFileRequest();
            pdxData.environment = so.GetValueForSOCarrierConstant("TEM");
            pdxData.filename = filename;
            pdxData.multipartFile = File.ReadAllText(filepath + @"\" + filename, UTF8Encoding.UTF8);

            var boundaryText = "USPS-PDX";
            var bodyText = new StringBuilder();
            bodyText.Append("--" + boundaryText + "\r\n");
            bodyText.Append("Content-Disposition: form-data; name=\"filename\"" + "\r\n\r\n");
            bodyText.Append(pdxData.filename + "\r\n");
            bodyText.Append("--" + boundaryText + "\r\n");
            bodyText.Append("Content-Disposition: form-data; name=\"environment\"" + "\r\n\r\n");
            bodyText.Append(pdxData.environment + "\r\n");
            bodyText.Append("--" + boundaryText + "\r\n");
            bodyText.Append("Content-Disposition: form-data; name=\"multipartFile\"; filename=\"" +
pdxData.filename + "\"" + "\r\n");
            bodyText.Append("Content-Type: text/plain" + "\r\n\r\n");
            bodyText.Append(pdxData.multipartFile + "\r\n");
            bodyText.Append("--" + boundaryText + "--");
            var bodyBytes = UTF8Encoding.UTF8.GetBytes(bodyText.ToString());

            request.ContentType = "multipart/form-data; boundary=" + boundaryText;
            request.ContentLength = bodyBytes.Length;
            var requestStream = request.GetRequestStream();
            requestStream.Write(bodyBytes, 0, bodyBytes.Length);
            requestStream.Close();

            using (var webResponse = (HttpWebResponse)request.GetResponse())
            {
               if (webResponse.StatusCode == HttpStatusCode.OK)
               {
                  using (var responseReader = new StreamReader(webResponse.GetResponseStream()))
                  {
                     var responseText = responseReader.ReadToEnd();
                     if (logger.IsDebugEnabled)
                     {
                        logger.Debug("HttpWebResponse: \r\n" + responseText + "\r\n");
                     }
                  }
                  return true;
               }
               else
               {
                  if (logger.IsErrorEnabled)
                  {
                     logger.Error("HttpWebResponse Error SendFileUsingPDXTransfer(), Server: " +
webResponse.Server + ", StatusCode: " + webResponse.StatusCode + ", StatusDescription: " +
webResponse.StatusDescription);
                  }
                  return false;
               }
            }
         }
         catch (WebException webExc)
         {
            if (logger.IsErrorEnabled)
            {
               logger.Error("Web Error SendFileUsingPDXTransfer(), Source: " + webExc.Source + ", Status:
" + webExc.Status + ", Message: " + webExc.Message);
            }
            if (webExc.Status == WebExceptionStatus.ProtocolError)
            {
               HttpWebResponse response = webExc.Response as HttpWebResponse;
               if (response != null)
               {
                  // Process response
               }
            }
            return false;
         }
         catch (Exception exc)
         {
            if (logger.IsErrorEnabled)
            {
            usa    logger.Error("Error SendFileUsingPDXTransfer(): " + exc.Message + ", " +
exc.ToString());
            }
            return false;
         }
      }
```

4.6 User Agent Information

User Agent information can be added to PDX API requests. This information will be logged during initial request processing, which is useful if API calls fail prior to reaching the PDX application. For example, executing an API request using old protocols (e.g. HTTP1.0) will fail before PDX receives the request. Without the User Agent information, the PDX support team does not know who is having the issue since, by default, the User Agent information only provides the software or browser being used. PDX suggests that this information contain the SSF header record for Vendor Code and Software Version Number (i.e. 9999/v2.7.x).

Each REST implementation has its own method for setting User Agent information (e.g. setRequestProperty("User-Agent", "Company")). Refer to the documentation or online references to determine how to set or override the User Agent value for the REST implementation chosen.

Examples:

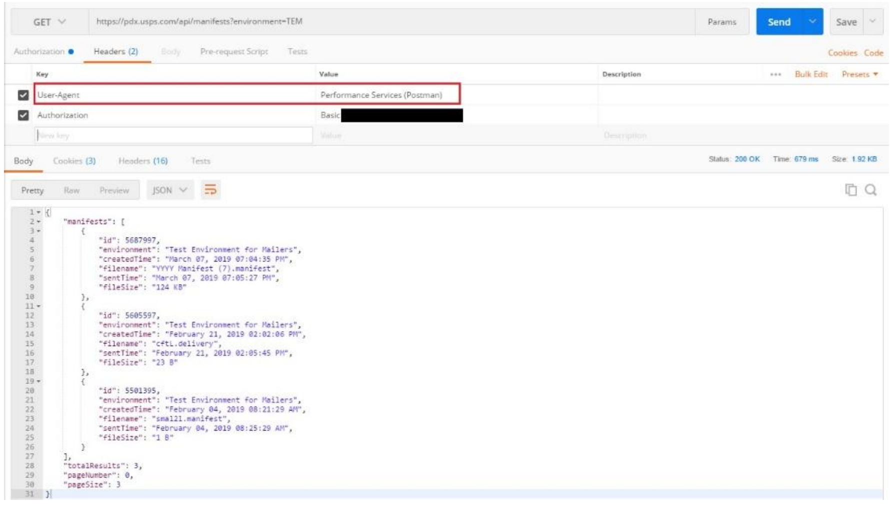
- Executing a curl request with a default User Agent creates a log entry similar to the following (which does not indicate the customer having the issue, only that they used curl/7.58.0):

  "GET /manifests/api?environment=TEM HTTP/1.0" 403 "**curl/7.58.0**"

- Executing a curl request with a User Agent defined (-A option):
  curl **-A "Performance Services (curl) 9999/v2.7"** -X GET …
  creates a log entry similar to the following:

  "GET /manifests/api?environment=TEM HTTP/1.0" 403 "**Performance Services (curl) 9999/v2.7**"

- Executing a Postman request with the User-Agent key defined as "**Performance Services (Postman)**":



GET https://pdx.usps.com/api/manifests?environment=TEM

Key	Value	Description
User-Agent	Performance Services (Postman)	
Authorization	Basic [redacted]	

Status: 200 OK Time: 679 ms Size: 1.92 KB

```
{
    "manifests": [
        {
            "id": 5687997,
            "environment": "Test Environment for Mailers",
            "createdTime": "March 07, 2019 07:04:35 PM",
            "filename": "YYYY Manifest (7).manifest",
            "sentTime": "March 07, 2019 07:05:27 PM",
            "fileSize": "124 KB"
        },
        {
            "id": 5605597,
            "environment": "Test Environment for Mailers",
            "createdTime": "February 21, 2019 02:02:06 PM",
            "filename": "cftL.delivery",
            "sentTime": "February 21, 2019 02:05:45 PM",
            "fileSize": "23 B"
        },
        {
            "id": 5501395,
            "environment": "Test Environment for Mailers",
            "createdTime": "February 04, 2019 08:21:29 AM",
            "filename": "smal21.manifest",
            "sentTime": "February 04, 2019 08:25:29 AM",
            "fileSize": "1 B"
        }
    ],
    "totalResults": 3,
    "pageNumber": 0,
    "pageSize": 3
}
```

  creates in a log entry similar to the following:

  "GET /api/manifests?environment=TEM HTTP/1.0" 403 "**Performance Services (Postman)**"

4.7 Protocol / Communication Errors

The PDX application is converting to newer protocols. Attempting to access PDX using older protocols (HTTP1.0 or TLS1.1, TLS1.0, SSL) via the GUI or API will generate errors and requests will fail. Detailed explanations regarding these protocol issues are described below:

Requests Using HTTP1.0

The request will process successfully yet return results containing the message: "403 Forbidden".

Example:
```
curl  --http1.0  -X GET  'https://pdx.usps.com/manifests/api?environment=TEM'
```

Results:
```
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /manifests/api/ on this server.</p>
</body></html>
```

Requests Using Pre-TLSv1.2

The request will fail without results and the return code will be set to an error.

Example:
```
curl --tlsv1.0  -X GET  'https://pdx.usps.com/manifests/api?environment=TEM'
```

There are no results. The return code from curl was set to 35 (CURLE_SSL_CONNECT_ERROR) indicating that a problem occurred during the SSL/TLS handshake.

4.8 Support Information

Questions or comments can be sent directly to the PDX team via dts-pdx@usps.gov. Alternatively, the USPS help desk can be reached at: (800) 522-9085.