Appendix A: Part D Pricing File Submission (PDPFS) API Pilot

Contents

Background

Beginning with Contract Year (CY) 2021, CMS will provide Part D sponsors with two mechanisms for submitting their Part D pricing files in HPMS. Sponsors may continue to submit pricing files by uploading files via the PDPFS module; alternatively, sponsors may submit pricing files via a new application programming interface (API).

This document provides logistical information regarding the pilot process. Please note that these instructions were developed to facilitate the PDPFS pilot. CMS will update these instructions, as needed, using lessons learned during the pilot process.

What is an API?

An API, or application programming interface, (API) provides an interface that allows two pieces of software or systems to communicate with each other. HPMS will follow the Partner API release policy, where the API is shared only with users that have access to HPMS.

In the case of the Part D pricing files, the API can be used to set up protocols that would submit the pricing files automatically to HPMS on a scheduled basis, without an individual logging into the system itself. To make this happen, your organization (or industry partner) would be responsible for creating the API that would (1) send the drug pricing files through the HPMS API, and (2) call the HPMS API to determine the status of your submitted drug pricing files.

Overview of the PDPFS API Pilot

In the pilot, your organization will:

- Submit pricing files (PF), pharmacy cost files (PC), and optional ceiling price files (CP) to HPMS using the PDPFS API.
- Retrieve the status of submission(s) for a given contract using the PDPFS API.

Please note the following regarding the pilot:

- Submissions made via the PDPFS API pilot will not impact the production Medicare Plan Finder (MPF).
- The retrieval of submission statuses will be limited to the submissions made via the PDPFS API pilot window.
- The pilot will be administered using the HPMS implementation environment.
- Each sponsor will be limited to five (5) contracts only. Please note that the 5 contract restriction only applies to the pilot process.

Roles and Responsibilities for the PDPFS API Pilot

CMS (or its designee) will:

- Be responsible for HPMS API key management, including the issuance of API keys to eligible responsible parties.
- Ensure that the necessary technical support resources are available to facilitate the implementation of the API on the HPMS side.

The Part D sponsor (or designee) will:

- Be responsible for establishing and maintaining the API to submit their pricing files to HPMS and retrieve submission statuses.
- Ensure that the necessary technical support resources (e.g., IT department, DevOps, etc.) are available to facilitate the implementation of the API on the sponsor side.
- Safeguard the API secret key.
- Follow the HPMS rules of behavior for the API secret key and tokens.

Obtaining Access for the PDPFS API Pilot

Your organization will need to establish **one or more responsible parties** to obtain access to the PDPFS API pilot. Each responsible party must have:

1. A CMS-issued EUA user ID with HPMS access;
2. API-participating contract number(s) assigned to the user ID in HPMS; and
3. An official letter from your organization documenting your role as the responsible party for the PDPFS API pilot for the selected contract number(s). The letter must include the following:
    a. Name of the responsible party (you may establish more than one)
    b. CMS user ID
    c. Company name
    d. Contract number(s) that are part of the pilot (up to 5)
    e. Contact information for the **technical** point of contact that will be developing the API on your organization's behalf (if different than the responsible party)

One letter may be submitted for responsible parties participating in the pilot. The official letters must be sent to Sara Walters at sara.walters@cms.hhs.gov no later than Friday, May 22, 2020.

Upon receipt and processing of these submissions, the responsible party will receive further information regarding the API key(s) to be used for PDPFS API pilot.

Important Notes:

- If you are participating as a consultant or third party vendor on behalf of one or more Part D sponsors, then you must submit an official letter from each of the sponsor organizations that are part of the PDPFS API pilot.

- If you are not currently contracted with a Part D sponsor, please contact Sara Walters for further guidance.

API Key Management

- The HPMS API key request process for the pilot will be different than the HPMS production API key process.

- As such, the API key(s) issued as part of this pilot are **only** valid during this pilot window. New API keys will be issued for the CY 2021 test and production submission windows.

- One or more API keys may be requested from CMS.

    - Each API key is associated to a responsible party user. Conversely, one responsible party can be associated with multiple API keys.
    - Each API key is associated with a defined set of contract numbers.

Security Practices

- The API secret should be restricted to only the responsible party for each key. This individual is responsible for ensuring the security of the secret key. It must not be shared with other staff and should be stored in a location that is only accessible to the user's machine.

    - If more than one user requires access using a single key, then you should create a service that provides the designated users with an authorization token only. With this approach, the user does not need to know the secret, but will be able to perform the action.

- The API key ID can be shared with internal partners with a need to utilize the new API.

- The API key ID and secret will rotate periodically. New IDs and secrets will be reissued automatically at that time.

    - Contract assignments and other authorization components will not be affected.

- If a key is compromised, the responsible party must inform CMS immediately so that your key can be deactivated.

- If needed, CMS can deactivate and reissue a new secret at the request of the Part D sponsor.

CMS Resources for the PDPFS API Pilot

The HPMS Help Desk should **not** be contacted regarding the PDPFS API pilot.

Rather, please use the following e-mail resource for technical communications regarding the pilot: HPMS-PDPFS.API@softrams.io

For general questions regarding the pilot, please contact Sara Walters at sara.walters@cms.hhs.gov.

Supporting Materials

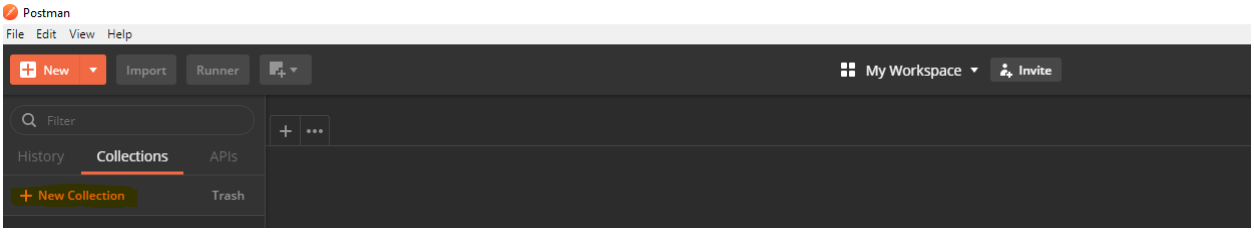
To upload pricing file submissions to HPMS via the PDPFS API for one or more contract numbers

Step 1: Using the API key, the API client will send a request to the IDM service API to get an Authorization token (JWT).

A CMS user ID is required in order to obtain the token. The token would be active for 30 minutes, and the process of requesting a new token is automatic.

Step 2: Using the Authorization token, the API client will make the request call for the HPMS Upload API. The Upload API verifies the token, and if everything is passed, the API will return an upload success with a confirmation ID or a failure response.

Steps to access the PDPFS file upload and submission status endpoints



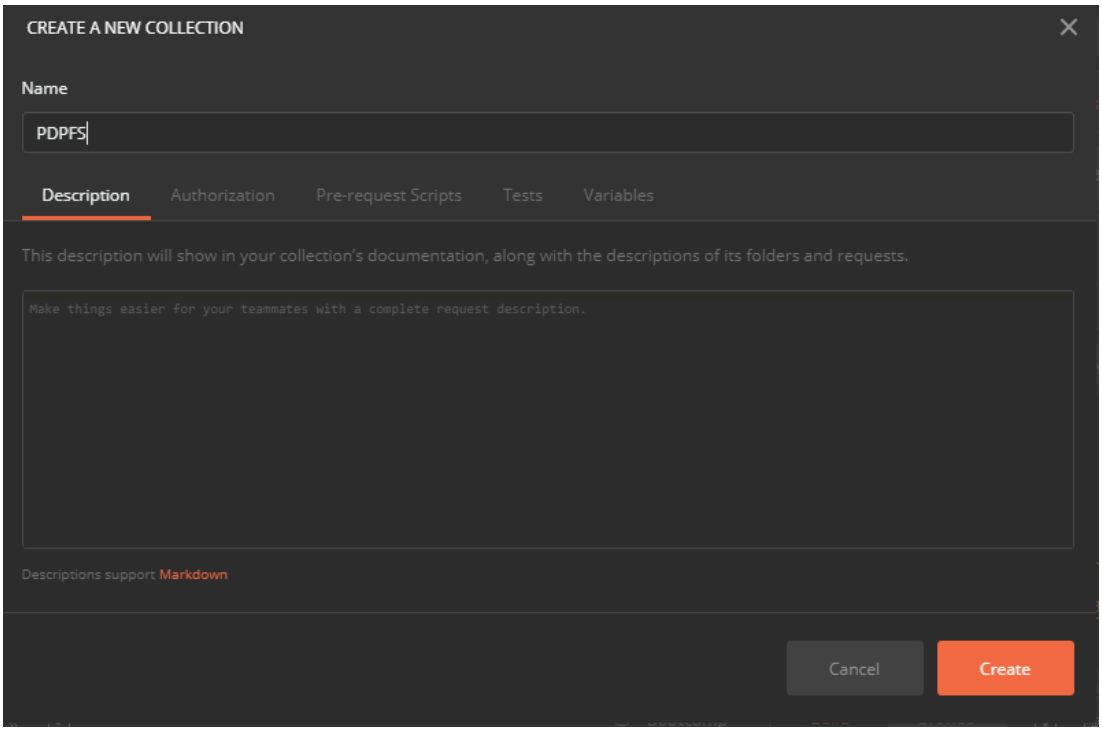
The following are **examples** of software options that can be used for calling the endpoint:

- Postman
- Fiddler
- SoapUI

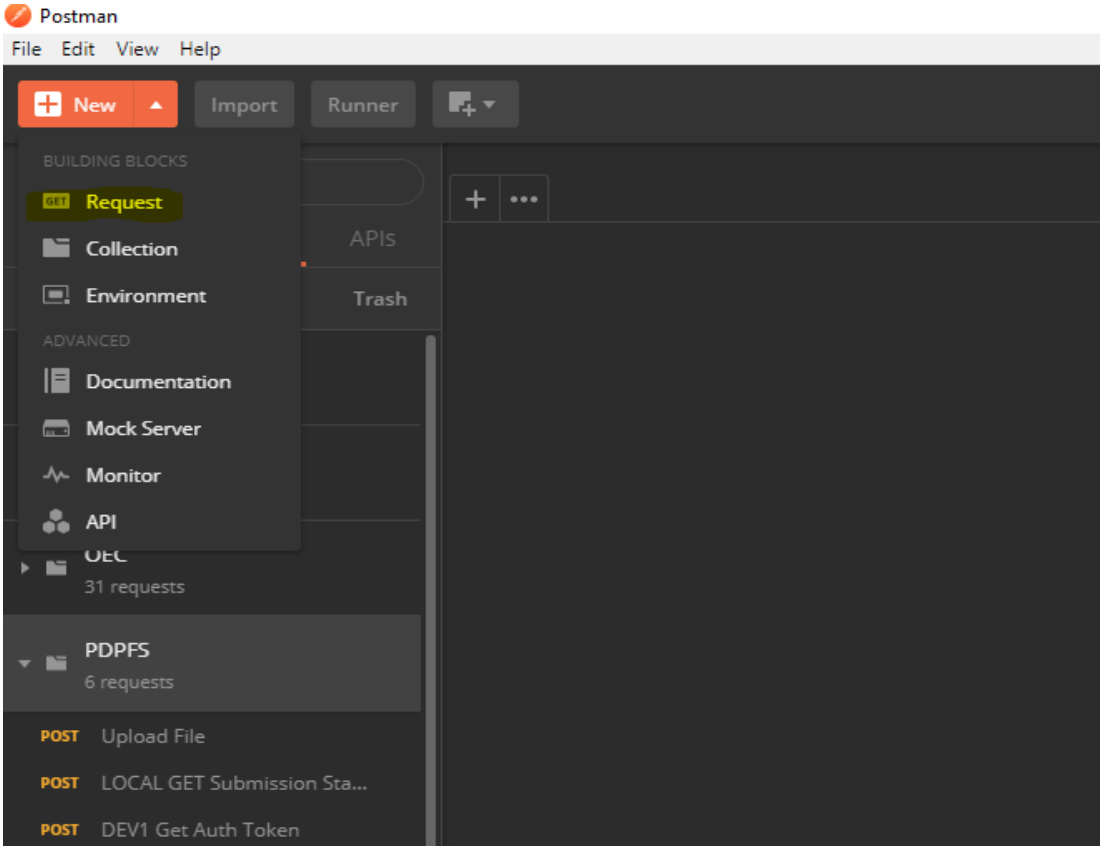
Likewise, there are various programming languages that may be used. This document provides code snippets for NodeJS, cURL, and GoLang. **You may use whichever technical solutions work for your organization.**

Step 1: Retrieving an authorization token from the IDM service, including the steps using Postman.

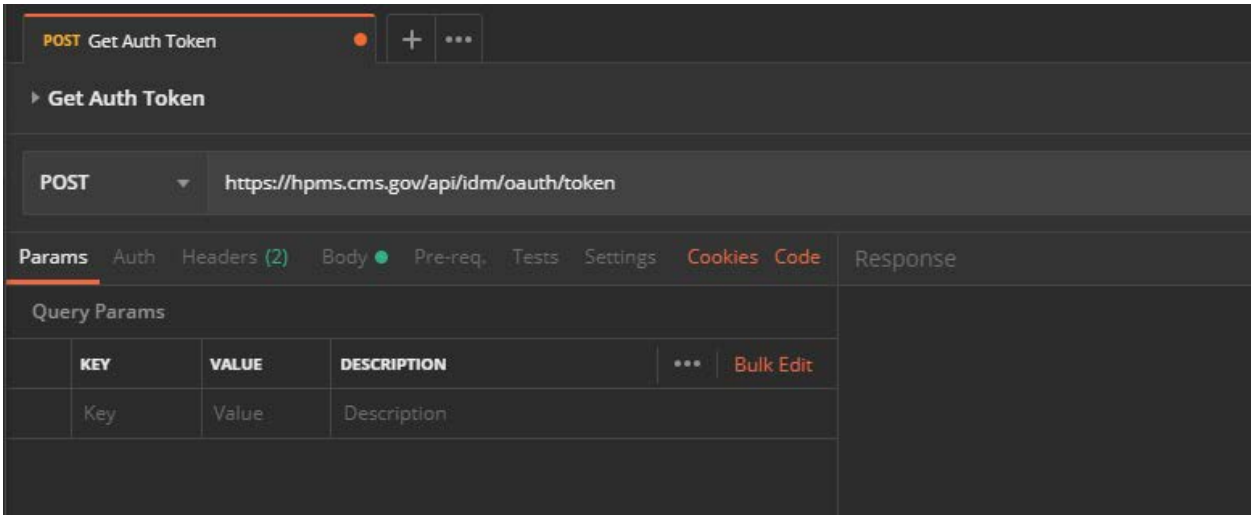
1. Create and label a new collection. This document will refer to the collection as PDPFS going forward.



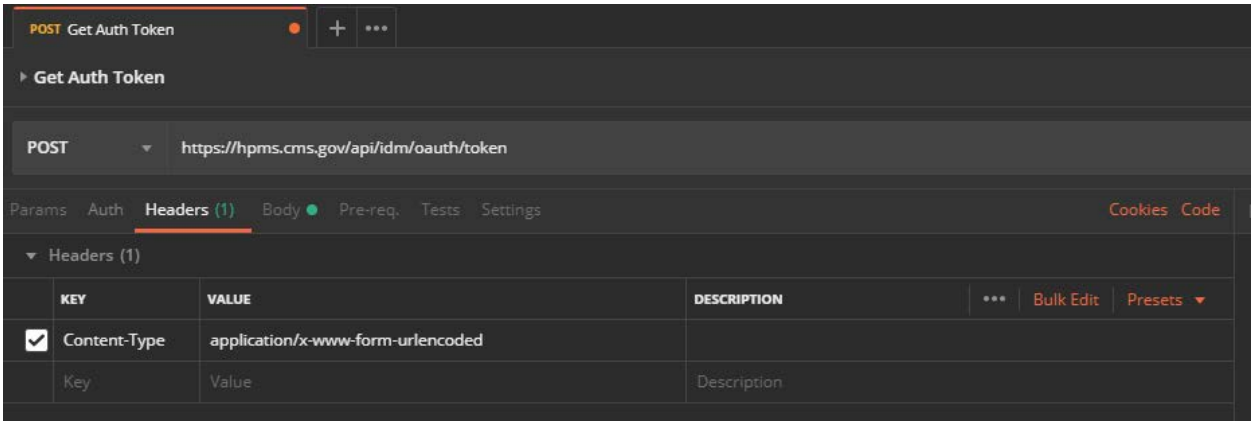
2. Create a new request and save it to the PDPFS collection.



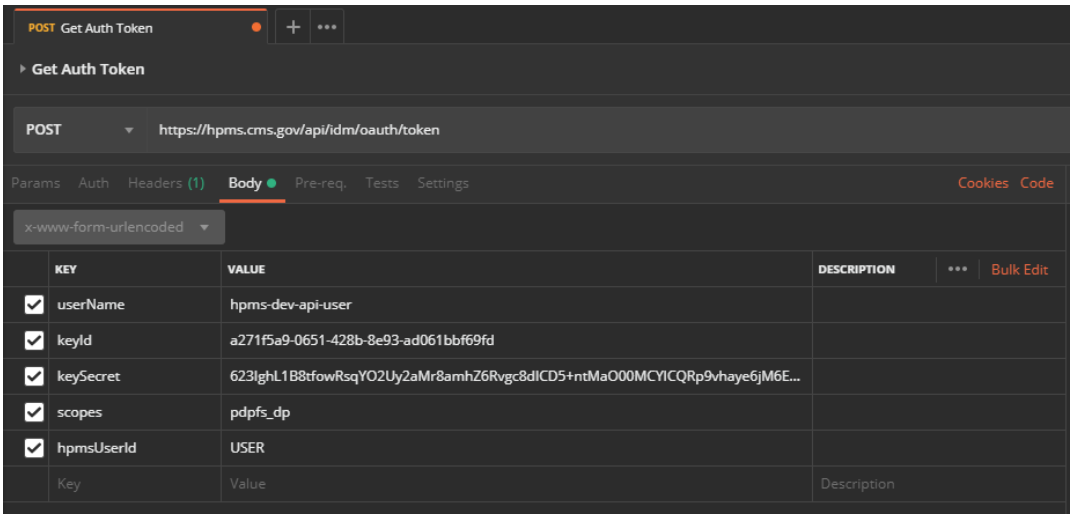
3. Change the type to POST and add the IDM endpoint URL.



4. Add a header with a key of Content-Type and value of application/x-www-form-urlencoded.



5. In the Body, add the following fields. The associated values will be supplied to you once you are granted access to use the endpoints.
    a. username
    b. keyId
    c. keySecret
    d. scopes
    e. hpmsUserId



6. Click the “Send” button. The response should provide you with a temporary access token which you will use in subsequent calls to the PDPFS API.

IDM Code Snippets Examples

NodeJs (using Request):

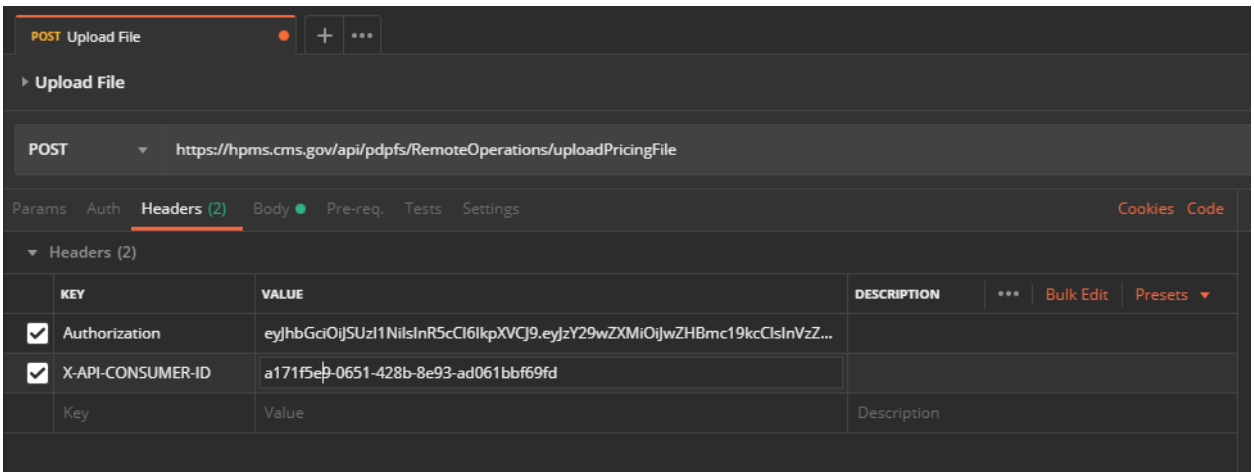
```
var request = require('request');
var options = {
  'method': 'POST',
  'url': 'https://hpms.cms.gov/api/idm/oauth/token',
  'headers': {
    'Content-Type': 'application/x-www-form-urlencoded'
  },
  form: {
    'userName': 'mct-dev-api-user',
    'keyId': 'a171f5e9-0651-428b-8e93-ad061bbf69fd',
    'keySecret':
'623IghL1B8teowRsqYO2Uy2aMr3amhZ6Rvgc8dICD5+ntMaO00MCYICQRp9vhaye6jM6EyREiJdpvcXmCC13Qw==',
    'scopes': 'pdpfs_dp',
    'hpmsUserId': 'USER'
  }
};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

cURL:

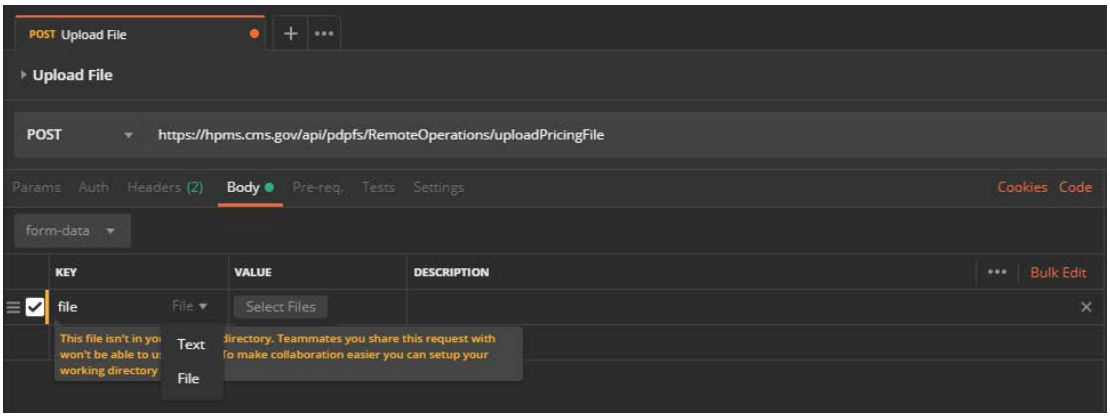
```
curl --location --request POST 'https://hpms.cms.gov/api/idm/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'userName=mct-dev-api-user' \
--data-urlencode 'keyId=a171f5e9-0651-428b-8e93-ad061bbf69fd' \
--data-urlencode
'keySecret=623IghL1B8teoeRsqYO2Uy2aMr8amhZ6Rvgc8dICD5+ntMaO00MCYICQRp9vhaye6jM6EyREiJdpvcXmCC13Qw
==' \
--data-urlencode 'scopes=pdpfs_dp' \
--data-urlencode 'hpmsUserId=USER
```

Step 2: Uploading the pricing files (PC,PF and optional CP file)

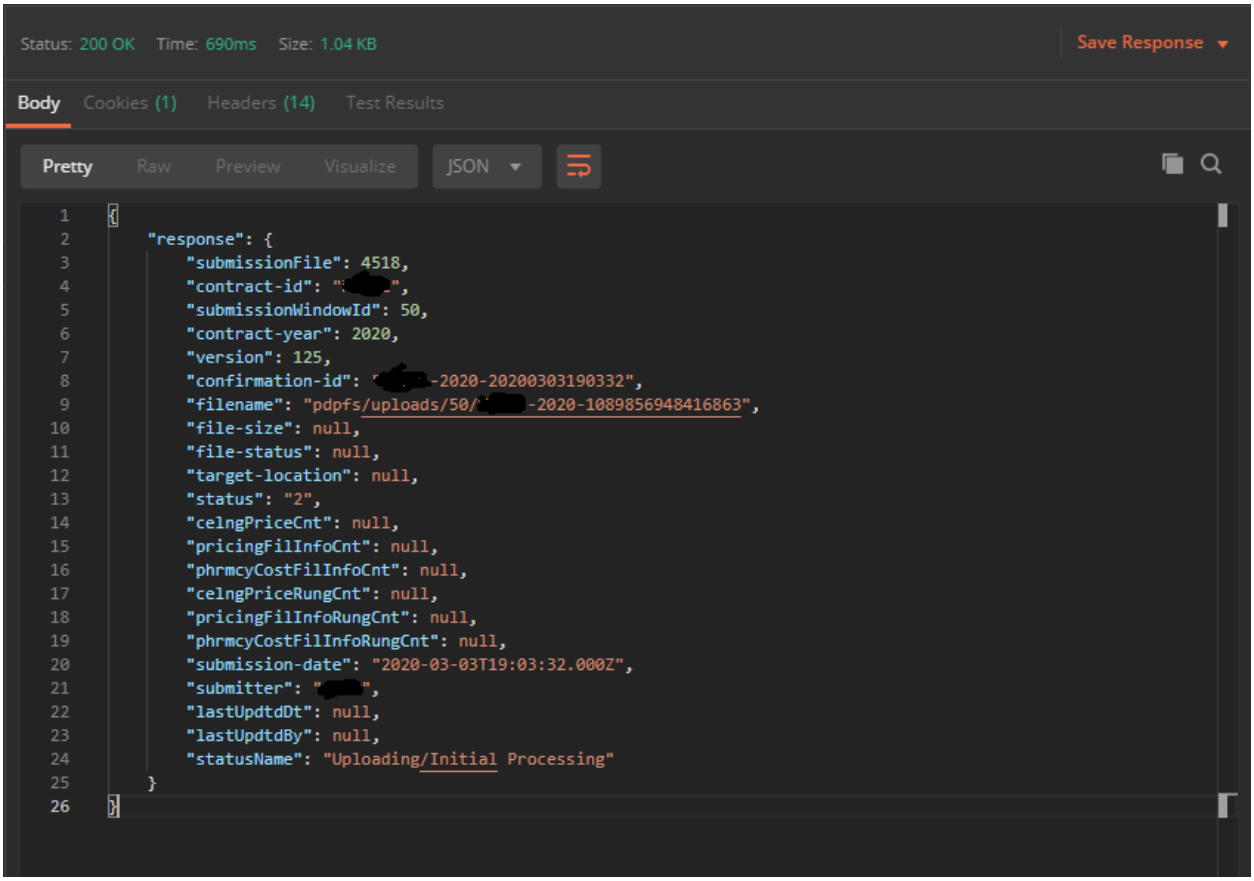
1. Create a new request POST request and add the upload pricing file endpoint.
2. Add the following headers:
    a. Authorization: This will be the value of the authorization token you retrieve from the previous IDM call.
    b. X-API-CONSUMER-ID: This is the keyId from the previous IDM call which will be supplied to you when you are granted access to the endpoints.
    c. Contractyear: The contract year you wish to submit for.
    d. ContractId: The ID of the contract you are submitting for.



3. In the body add a key called file and select file from the drop-down box that appears to the right of the text box.



4. Press the “Select Files” button in the value column and navigate to the pricing file you wish to upload and select it. Press send and you will receive a similar response as shown below.



Code Snippet Examples for Uploading the Pricing Files

NodeJS

```
var request = require('request');
var fs = require('fs');
var options = {
  'method': 'POST',
  'url': 'https://hpms.cms.gov/api/pdpfs/RemoteOperations/uploadPricingFile',
  'headers': {
    'Authorization':
'eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzY29wZXMiOiJwZHBmc19kcCIsInVzZXJJZCI6Im1jdC1kZXYxLWFwcCIsImhwbXN
Vc2VySWQiOiJqeXhhIiwia2V5IjoiRnhaN1dUTXdCUS84dFNyQW5KMytZNC9NdUw3bUp6VU5QbGJuSEtEVlE2bmpnaHJHe
W9GSVVPS1A5VEtONFpHbFNEK2lPL2RWWkZwUzM2cW1tV3pydXc9PSIsImlhdCI6MTU4MzI2MjAwMSwiZXhwIjoxNTgzMj
Y1NjAxLCJhdWQiOiJhcGkuaHBtcy5jbXMuZ292IiwiaXNzIjoiaHBtcy5jbXMuZ292In0.qGBzhAg9AUU8yKSR7E6n-
6fHToy4NlLGqfRgSAILTT36TS7s40Qh0whg0qE8Y8e1lLXRKO66vwF2YhtNPPbblA',
    'X-API-CONSUMER-ID': 'a171f5e9-0651-428b-8e93-ad061bbf69fd',
    'contractyear': '2020',
    'contractid': 'H0001',
    'Content-Type': 'multipart/form-data; boundary=--------------------------428460616493473796797408'
  },
  formData: {
    'file': {
      'value': fs.createReadStream('/C:/PATH/TO/FILE/H0001.zip'),
      'options': {
        'filename': '/C:/PATH/TO/FILE/H0111.zip',
        'contentType': null
      }
    }
  }
};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

cURL

```
curl --location --request POST 'https://hpms.cms.gov/api/pdpfs/RemoteOperations/uploadPricingFile' \

--header 'Authorization:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXeCJ9.eyJzY29wZXMiOiJwZHBmc19kcCIsInVzZXJJZCI6Im1jdC1kZXYxLWFwcCIsImhwbXN
Vc2VySWQiOiJqeXhhIiwia2V5IjoiRnhaN1dUTXdCUS84dFNyQW5KMytZNC9NdUw3bUp6VU5QbGJuSEtEVlE2bmpnaHJHe
W9GSVVPS1A5VEtONFpHbFNEK2lPL2RWWkZwUzM2cW1tV3pydXc9PSIsImlhdCI6MTU4MzI2MjAwMSwiZXhwIjoxNTgzMj
Y1NjAxLCJhdWQiOiJhcGkuaHBtcy5jbXMuZ292IiwiaXNzIjoiaHBtcy5jbXMuZ292In0.qGBzhAg9AUU8yKSR7E6n-
6fHToy4NlLGqfRgSAILTT36TS7s40Qh0whg0qE8Y8e1lLXRKO66vwF2YhtNPPbblA' \

--header 'X-API-CONSUMER-ID: a171f5e9-0651-428b-8e93-ad061bbf69fd' \

--header 'contractyear: 2020' \

--header 'contractid: H0001' \

--header 'Content-Type: multipart/form-data; boundary=--------------------------428460616493473796797408' \

--form 'file=@/C:/PATH/TO/FILE/H0001.zip'
```

Error Handling / Submission Status Messages

The following error messages are currently delivered via the HPMS PDPFS module. Additional error messages may be added based on user experience during the pilot. Each error message with its corresponding input fields are mapped for easy reference. This is the current list, but it is subject to change.

Required Fields	Response Messages
Headers	

Authorization	When missing or invalid:
	{
	"error": {
	"statusCode": 401,
	"name": "Error",
	"message": "Authorization Required",
	"code": "AUTHORIZATION_REQUIRED"
	}
	}

X-API-CONSUMER-ID	When missing:
	{
	"error": {
	"statusCode": 401,

	"name": "Error", "message": "Authorization Required", "code": "AUTHORIZATION_REQUIRED" } }

contractYear	When missing:
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "Missing contract year"
	}
	}

contractId	When missing:
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "Missing contract id"
	}
	}
	
	Not authorized to operate on contract:
	
	{
	"error": {
	"statusCode": 401,
	"name": "Unauthorized",
	"message": "API user: cms-dev1-app associated with HPMS user: jyxa is not authorized to operate on contract: R1234. "
	}
	}

Body	

file	When missing:
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "No file detected"
	}
	}

	ContractId doesn't match file name:
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "ContractId: R1234 does not match name of file: H4321"
	}
	}
	
	File is not a Zip
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "Incorrect file type: text/plain detected application/zip is required"
	}
	}
	
	File name is incorrectly formatted
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "File name: fileName not in proper format"
	}
	}
	
	File too large
	
	{
	"error": {
	"statusCode": 400,
	"name": "Bad Request",
	"message": "File too large"
	}
	}

General Msgs	

Submission window	Good request
	
	{
	"response": {

```
      "submissionFile": 4501,
      "contract-id": "R1234",
      "submissionWindowId": 50,
      "contract-year": 2020,
      "version": 108,
      "confirmation-id": "R1234-2020-20200212160548",
      "filename": "pdpfs/uploads/50/R1234-2020-9176610945964541",
      "file-size": null,
      "file-status": null,
      "target-location": null,
      "status": "2",
      "celngPriceCnt": null,
      "pricingFilInfoCnt": null,
      "phrmcyCostFilInfoCnt": null,
      "celngPriceRungCnt": null,
      "pricingFilInfoRungCnt": null,
      "phrmcyCostFilInfoRungCnt": null,
      "submission-date": "2020-02-12T16:05:48.000Z",
      "submitter": "jyxa",
      "lastUpdtdDt": null,
      "lastUpdtdBy": null,
      "statusName": "Uploading/Initial Processing"
   }
}
```


Contract did not submit in window 1


```
{
   "error": {
      "statusCode": 401,
      "name": "Unauthorized",
      "message": "Contract did not submit in window 1."
   }
}
```


Window 1 submission not in proper status


```
{
   "error": {
      "statusCode": 401,
      "name": "Unauthorized",
      "message": "Window 1 submission is not in review required or resubmission
required status."
   }
}
```

Status Check

Required Fields	Response Messages
Headers	

Authorization	When missing or invalid:
	
	{
	"error": {
	"statusCode": 401,
	"name": "Error",
	"message": "Authorization Required",
	"code": "AUTHORIZATION_REQUIRED"
	}
	}

X-API-CONSUMER-ID	When missing or invalid:
	
	{
	"error": {
	"statusCode": 401,
	"name": "Error",
	"message": "Authorization Required",
	"code": "AUTHORIZATION_REQUIRED"
	}
	}

Body	

Confirmation Numbers	Good request:
	
	{
	"submissionStatuses": [
	{
	"ConfirmationNumber": "R1234-2020-20200129154043",
	"SubmissionStatus": "Level Two Validation - Fatal Errors"
	}
	]
	}
	
	Good request:
	
	{
	"submissionStatuses": [
	{

	"ConfirmationNumber": "R1234-2020-20200129154043",
	"SubmissionStatus": "Level Two Validation - Fatal Errors"
	},
	{
	"ConfirmationNumber": "R1234-2020-20200212141659",
	"SubmissionStatus": "Uploading/Initial Processing"
	}
	]
	}
	
	Not authorized to access requested contract:
	
	{
	"error": {
	"statusCode": 401,
	"name": "Unauthorized",
	"message": "User not authorized to retrieve status for contractIds: R1234"
	}
	}
	
	Confirmation number not found:
	
	{
	"submissionStatuses": [
	{
	"ConfirmationNumber": "H0111-2020-2020012914043",
	"SubmissionStatus": "Confirmation number does not exist"
	}