TPA Stream Developer Documentation

Sep 15, 2020

Contents

Welcome to TPA Stream! Here you'll find comprehensive information for linking your application with the TPA Stream API, allowing you to focus on building your product, not integrating with ours. If you're planning to use our API in Production, please review our Privacy Policy.

TPA Stream's developer tools consist of a **JavaScript SDK** Client Library to embed consumer enrollment into your web or mobile application, **Webhooks** to provide your application with realtime Claim updates, and a **REST API** for getting the most up-to-date information.

CHAPTER 1

Guide

1.1 Getting Started

1.1.1 Connecting the Health Insurance API to your Application

In order to get Claim data into your application, a participant must first connect their Health Insurance account with your application using TPA Stream's API. The easiest way to accomplish this is by using our *Connect Client Library*.

Our Client Library is highly configurable and meant to be built directly into your application. Once embedded into your application, the client will allow the participant to select an Insurance Carrier from our list, enter their credentials, and verify them in realtime. Once they are connected, we will start receiving their Claim data asynchronously.

1.1.2 Retrieving Claim Data

Typically we begin receiving a participants claims within a minute or so after they connect their account using our Client Library. It can take anywhere from several minutes to several hours for us to receive all their claims, which are usually ordered by newest claims first.

There are several ways for you to get Claim Data into your application. If you want them as soon as possible, the quickest way to get them is by implementing our *Connect Claim Webhook*. You can also pull claims on demand using our *REST API*. Additionally, we can setup a daily CSV file feed to send all new claims via sFTP, FTPS, or to an S3 bucket. Contact sales@tpastream.com to learn more.

1.2 TPA Stream Connect

1.2.1 Client Library

Usage

Our JavaScript Client Library is designed to allow users of your Web or Mobile application to connect their Health Insurance accounts with your Application.

To get up and running, you can simply copy and paste the JavaScript snippets below, and run it in *isDemo* mode today. You can use the provided callbacks to customize the look, feel, and overall experience for your users.

JavaScript Client

Source Code

You can find the source code to our JavaScript Client here: https://github.com/TPAStream/stream-connect-js-sdk

NPM Example (recommended)

```
npm i stream-connect-sdk

import StreamConnect from 'stream-connect-sdk';

const sdk = StreamConnect({
    el: '#react-hook',
    isDemo: true
});
```

CDN Example

```
<script src="https://app.tpastream.com/static/js/sdk.js"></script>
<script>
    window.StreamConnect({
        el: '#react-hook',
        tenant: {
            systemKey: 'test',
            vendor: 'internal'
        },
        employer: {
            systemKey: 'some-system-key',
            vendor: 'internal',
            name: 'some-name'
        },
        user: {
            firstName: 'Joe',
            lastName: 'Sajor',
            email: 'some-email@place.com',
            memberSystemKey: 'some-system-key',
            phoneNumber: '333333333',
            dateOfBirth: '11-11-1121'
        },
        apiToken: 'Some Provided Key → 21poi34kjqf21j1poi1d2po', // We'll provide␣
↪this.
        isDemo: false,
        realTimeVerification: true,
        renderChoosePayer: true,
```

(continues on next page)

(continued from previous page)

```
        doneGetSDK: ({ user, payers, tenant, employer }) => {},
        doneChoosePayer: ({ choosePayer, streamPayers }) => {},
        doneTermsOfService: () => {},
        doneCreatedForm: () => {},
        donePostCredentials: ({ params }) => {},
        doneRealTime: () => {},
        donePopUp: () => {},
        doneEasyEnroll: ({ employer, payer, tenant, policyHolder, user }) => {},
        handleFormErrors: (error, {response, request, config}) => {},
        userSchema: {}
    })
</script>
```

As of SDK version 0.4.7 the CDN provider is now versioned and will support up to 10 minor versions behind.

- **Importing the various versions of the SDK is handled in `src` attribute on your script tag**
  - "https://app.tpastream.com/static/js/sdk.js" –> Grabs the latest version of the SDK
  - **"https://app.tpastream.com/static/js/sdk-v-<VersionNumber>.js" –> For a specific version. Examples below.**
    - "https://app.tpastream.com/static/js/sdk-v-0.4.7.js"

Supported Parameters

The SDK currently supports the following parameters:

- `el` (This is where the SDK will render: Note -> This is a 'css selector')
- **tenant**
  - systemKey
  - vendor
- `employer`
  - `systemKey`
  - `vendor` (This will usually be 'internal')
  - `name`
- `user`
  - `firstName`
  - `lastName`
  - `email`
  - `memberSystemKey`
  - `phoneNumber`
- `apiToken`
- `realTimeVerification` -> Bool
- `renderChoosePayer` (If this is set to false doneChoosePayer* will pass all the required methods to create your own module)
- `isDemo` -> Bool (This is recommended for sandboxing before you hook the SDK up for real)

- `userSchema` (This is an object {} following react-jsonschema-form for making ui:schema)
- `doneGetSDK` * (Below are args passed into the func)
  - `user`
  - `payers`
  - `tenant`
  - `employer`
- `doneChoosePayer` * (The following params only appear when renderChoosePayer is `false`)
  - `streamPayers`
  - `choosePayer` * (The function to be called to render the next SDK step)
    - `payer` (An obj value from streamPayers)
    - Your functional call should look like `choosePayer({payer: streamPayers[some_index]})`
- `doneTermsOfService` *
- `doneCreatedForm` *
- `donePostCredentials` *
  - `params` (All submitted params to our API)
- `donePopUp` *
- `doneRealTime` *
- `doneEasyEnroll` *
  - `employer`
  - `payer`
  - `policyHolder`
  - `user`
  - `tenant`
- `handleFormErrors` *
  - `error`
  - `error_parts`
    - `response`
    - `request`
    - `config`

Note: only 'el' is required for demo mode

Function (`() => {}`) parameters are Starred*

Android Example

In Android, you can implement the JavaScript SDK by creating an html file in your assets folder that loads the SDK, then loading it in a WebView. Here's a simple example implementation in Java.

```
public class ViewWeb extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.content);
        WebView webview = (WebView) findViewById(R.id.webView);
        webview.loadUrl("file:///android_asset/stream-connect.html");
    }
}
```

iOS Example

In iOS, you can implement the JavaScript SDK by creating a directory in your project ("stream-connect" in the below example), and putting an html file in it (index.html in the below example) that loads the SDK. You can then load it into a WKWebView. Here's a simple example implementation in Objective-C.

```
import UIKit
import WebKit

class ViewController: UIViewController, WKUIDelegate, WKNavigationDelegate {

    @IBOutlet weak var webView: WKWebView!
    override func viewDidLoad() {
        super.viewDidLoad()
        webView.uiDelegate = self
        webView.navigationDelegate = self
        let url = Bundle.main.url(forResource: "index", withExtension: "html",␣
↪subdirectory: "stream-connect")!
        webView.loadFileURL(url, allowingReadAccessTo: url)
        let request = URLRequest(url: url)
        webView.load(request)
    }
}
```

Change Log

v0.4.8 (Latest)

- Add individual endpoints for the following: payer and terms of service
- Seperate the versions of the api and create a version manager.
- Drastically improve initial endpoint loadtime by bringing down less info.

v0.4.7

- Add versioning to the CDN provider
- Append version to all request headers for underlying api to read.

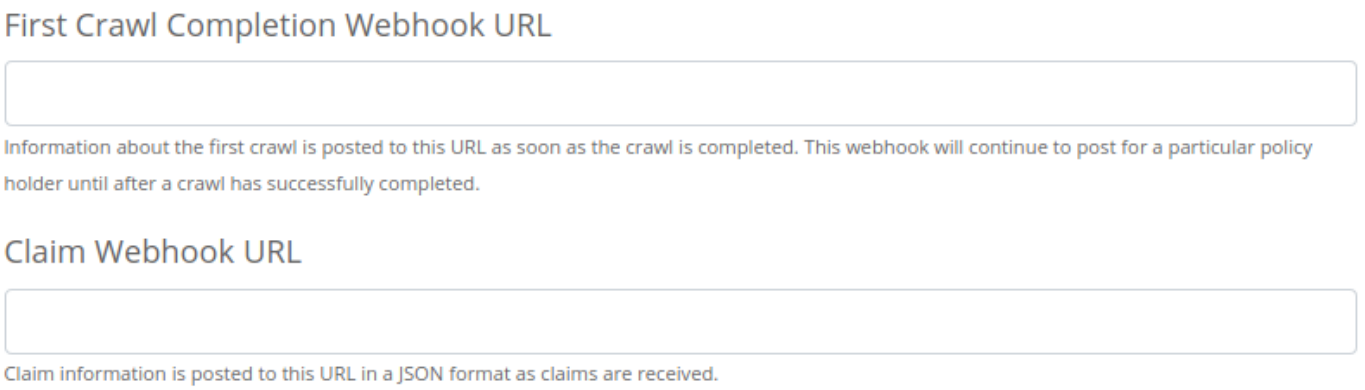
1.2.2 Webhooks

Claim Webhook

TPA Stream offers a claim webhook feature in which TPA Stream will post new claims to a customer-provided endpoint. We will POST any new claim that comes into TPA Stream via this webhook immediately after it is processed.

Claim Webhook URL

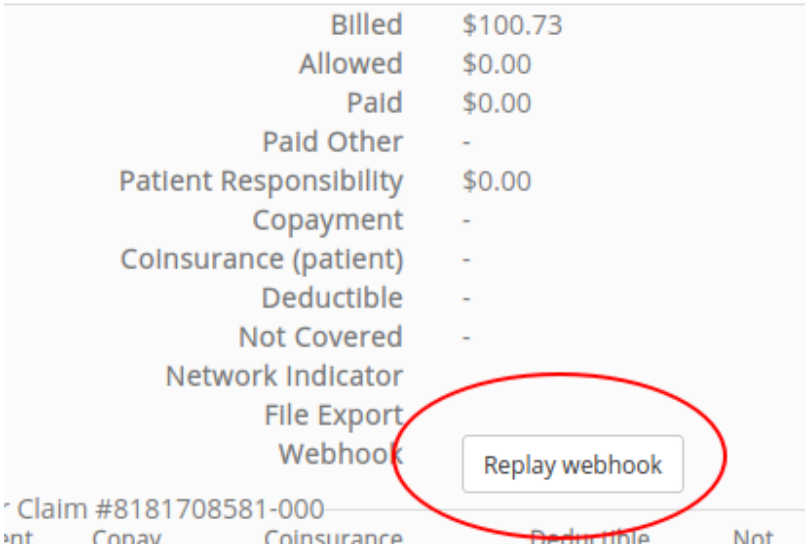
To edit the claim webhook URL, click on "Account Settings" on the settings page.



Note that you will only see this setting if the claim webhook feature is enabled.

Once the webhook URL has been updated, all future posts will go to that URL.

Replaying a Claim Post



To manually replay a claim post, find the appropriate claim on the claims page and click the "Replay webhook" button. If the button is not shown, please verify that the webhook feature is enabled and a URL is set as described above. This "replay" functionality is useful for testing, and can also be used to trigger a webhook for any pre-existing claims that are in the system, if desired.

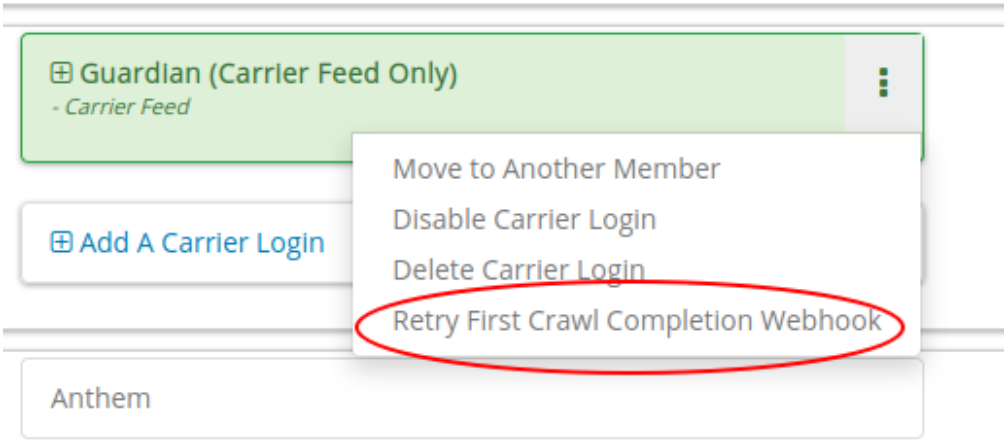
First Crawl Completion Webhook

TPA Stream also offers a crawl webhook feature that posts details about the first crawl of a policyholder to a customer-provided endpoint. It will POST this information after all the claims for the crawl have been processed. The last POST for a policy holder will occur when the crawl is successful for the first time. For example, if the first two crawls fail and next two attempts are successful, 3 POST requests will be made. Two for the failures and a third, final POST for the first success.

First Crawl Completion Webhook URL

To edit the first crawl completion webhook URL, click on "Account Settings" on the settings page similar to editing the claim webhook URL.

Replaying a Crawl Completion Post



To manually replay a first completion webhook post, find the appropriate member on the member page. Under policy holders, there will be a button to replay the webhook request. If the button is not shown, please verify that the webhook feature is enabled and a URL is set as described above. This "replay" functionality is useful for testing. If a crawl for that policy holder has not happened yet, it will return a failure. Note that the replay will not have `crawl_claim_ids` and will not be retried upon failure.

Request Retries

The request will be an HTTP POST with Content-Type header of application/json. An example of the JSON you can expect can be found at the end of this document. For Webhook POSTs, TPA Stream listens for the following codes from your server and reacts accordingly:

- If TPA Stream receives a `200` or `2xx` (Success) code it will determine the webhook POST is successful and not retry.
- If TPA Stream receives a `406` (Not Acceptable) code, TPA Stream will determine the POST is rejected and not retry.
- For any other code, TPA Stream will retry POSTing with an exponential backoff delay for up to 4 hours.

Security

TPAStream-Signature Verification

Also included in the request is a JWT signature that can be used to verify that the request has originated from TPA Stream, and not any other party. This header is passed in the TPAStream-Signature header of the request. The signature can be verified using our SSH RSA public key. The key can be obtained from https://app.tpastream.com/keys. The JWT hashing algorithm used is RS256.

We strongly recommend that you verify our JWT signature as a part of your webhook. Examples of how to decode a JWT in several common programming languages can be found at https://jwt.io. Note that the JWT library you choose must support RS256 (nearly all of them do), and should also support an exp check (although you could easily perform this simple expiration date check yourself using a UTC timestamp).

Example Claim Webhook JSON Request


```
{
   "data": {
      "service_provider_billing_npi_number": null,
      "computed_coinsurance_patient": null,
      "group_name": null,
      "members": [{
          "id": 99999999,
          "email": "johnny@appleseed.com",
          "first_name": "Johnny",
          "last_name": "Appleseed",
          "employer": {
              "id": 999,
              "name": "Dunder Mifflin"
          }
      }],
      "group_number": null,
      "eob_date": "2017-07-01T16:51:16.701956",
      "date_column": "2017-07-01T16:51:16.701956+00:00",
      "service_provider_number": "1aMhYxXFh",
      "service_provider_billing_number": null,
      "coinsurance_patient": null,
      "modifieddate": "2017-06-01T04:14:29.348875+00:00",
      "service_provider_billing_address": null,
      "status": "Partially Approved",
      "amount_billed": 228,
      "reduction": null,
      "claim_medical_lines": [{
         "claim_medical_line_id": 10004,
         "procedure_code": "87254 - VIRUS INOCULATION SHELL VIA",
         "days_supply": null,
         "copayment": null,
         "date_of_service": {
         "bounds": "[]",
         "start": "2020-01-16",
         "end": "2020-01-17"
         },
         "vendor_system_id": "2b7e0936",
         "discount": null,
         "coinsurance_patient": null,
         "amount_allowed": null,
```


(continues on next page)

(continued from previous page)


```
            "national_drug_code": null,
            "prescription_type_id": null,
            "patient_responsibility": 0,
            "procedure_name": "Preventive Visit-Ages 18-39 - see note E23",
            "prescription_type_str": null,
            "amount_billed": 228,
            "reduction": null,
            "amount_paid": null,
            "amount_not_covered": null,
            "diagnosis_code": "M19.172",
            "quantity": null,
            "amount_paid_other": null
        }],
        "amount_paid_other": null,
        "computed_reduction": null,
        "vendor_system_id": "28284fbbaa164d02",
        "discount": null,
        "computed_amount_billed": null,
        "service_provider_npi_number": null,
        "patient_responsibility": 0,
        "dataobject_id": 104,
        "remarks": "leverage ubiquitous users",
        "service_provider_billing_name": null,
        "network": null,
        "amount_not_covered": null,
        "copayment": null,
        "computed_copayment": null,
        "policy_holder": {
            "id": 104,
            "first_name": "Parrish",
            "last_name": "Appleseed"
        },
        "patient_account_number": null,
        "tpafiles": [],
        "date_of_service": {
            "bounds": "[]",
            "start": "2020-01-28",
            "end": "2020-01-28"
        },
        "incurred_value": null,
        "is_incomplete": null,
        "id": 10004,
        "type": null,
        "amount_allowed": null,
        "service_provider_address": "1097 Dicha Garden",
        "claim_requests": [],
        "computed_amount_paid": null,
        "check_number": "8858906",
        "last_updated_status": "2020-02-11T13:56:45.249767+00:00",
        "processed_on": "2017-06-21T00:30:46.791615",
        "createddate": "2016-03-17T07:11:30.580677+00:00",
        "computed_amount_allowed": null,
        "dependents": [{
            "id": 999,
            "member_id": 9999,
            "first_name": "Sally",
            "last_name": "Appleseed",
```


(continues on next page)

(continued from previous page)


```
            "email": "sally@appleseed.com",
            "ssn": "999999999",
            "relationship": "spouse",
            "createddate": "2017-07-01T16:51:16.701956",
            "modifieddate": "2017-07-01T16:51:16.701956",
        }],
        "patient_payer_number": null,
        "service_provider_billing_phone": null,
        "claim_medical_id": 10004,
        "check_date": "2020-01-24T23:49:50+00:00",
        "computed_patient_responsibility": null,
        "amount_paid": null,
        "service_provider": "Salary.com",
        "patient_name": "Abram",
        "policy_holder_id": 104
    }
}
```


Example Crawl Webhook JSON Request


```
{
    "data": {
        "members": [
            {
                "id": 63167
            }
        ],
        "policy_holder": {
            "id": 189162,
            "login_correction_message": "The login information you provided for␣
↪Anthem is invalid. Please re-enter your login information.",
            "login_problem": "invalid"
        },
        "success": false
    }
}
```


1.3 REST API

1.3.1 Authentication

TPA Stream's REST API supports Basic Authentication over HTTPS using your Secret API Key.

To implement, simply put your user name (email address) and Secret API key in the Authorization header on each API request.

Your API key is a Secret Password! Do not share this key with anyone, including TPA Stream. TPA Stream will NEVER ask for your Secret API Key.

A simple example using curl:

```
curl -L --user me@example.com:MY_API_KEY https://app.tpastream.com/api/claims
```

1.3.2 Security

For additional security, you must provide an IP CIDR address range. This range may be `0.0.0.0/0`, meaning no restriction, however we HIGHLY recommend a more restrictive address range to provide an extra layer of security in addition to your Secret API Key. This range can be provided on the Manage Users page under Settings.

1.3.3 API Response Format

Each API endpoint, upon success, will return an object with the entity (or entities) in the data key. If the endpoint is paginated, it will also contain a number of other pagination related fields:

```
{
  "data": [],  // or {} for a single entity
  "has_next": true,
  "has_prev": false,
  "next_num": 2,
  "page": 1,
  "pages": 123,
  "per_page": 1000,
  "prev_num": 0,
  "total": 122901
}
```

1.3.4 API Errors

In the event that there is an error, our response will have a key of message. It may contain some other additional information depending on the nature of the error. If the error is an unexpected error (and raises an exception in our system), it will also contain an event_id. If you see an event_id, then our engineers are notified that there is an issue.

We make every effort to provide meaningful HTTP status codes and messages when there is an error. Please do not hesitate to contact us regarding your usage of our API by emailing support@tpastream.com.

Error responses will look something like this:

```
{
  "message": "There was an error and hopefully this is useful info.",
  "event_id": "xxxxxxxxxx"
}
```

1.3.5 A note about Date Ranges

Types of data that are date ranges are serialized to JSON as an object with bounds:

```
{
  "bounds": "[)",
  "end": "2016-09-16",
  "start": "2016-09-15"
}
```

Note that since date of service can span over multiple days (for example, a hospital stay), it is stored as a range.

> *In the text form of a range, an inclusive lower bound is represented by "[" while an exclusive lower bound is represented by "(". Likewise, an inclusive upper bound is represented by "]", while an exclusive upper bound is represented by ")"*

—For more details and examples, please consult the PostgreSQL documentation
https://www.postgresql.org/docs/12/static/rangetypes.html#RANGETYPES-INCLUSIVITY

1.3.6 Filtering and Paging Through Results

Most API endpoints have enforced pagination. There are currently no globally defined upper limits to how much can be requested per_page, but please try to maintain a healthy balance between number of requests and amount of data requested via per_page with each request. When pulling large amounts of data, we recommend starting with per_page=1000 and optimizing as necessary. Please contact support@tpastream.com if you intend to pull large amounts of data, as we can help define a strategy or custom endpoint to better serve both your needs & ours.

Here are a list of endpoints. GET, PUT, POST, and DELETE are supported for most endpoints, however may or may not be enabled for each type of user.

GET All Claims (starts with page 1. The response will tell you if there are more pages available)

```
/api/claims
```

GET All Employers

```
/api/employer
```

GET All Members

```
/api/member
```

GET All Policy Holders

```
/api/policy_holder
```

GET All Claims where Employer ID is 99999

```
/api/employer/99999/claims
```

GET All Claims where Policy Holder ID is 99999

```
/api/policy_holder/99999/claims
```

GET All Claims where Member ID is 99999

```
/api/member/99999/claims
```

GET Page 3 of Claims with 1000 Claims per page for Employer 999

```
/api/employer/999/claims?per_page=1000&page=3
```

GET Page 3 of Claims with 10 Claims per page for Employer 99. Do not include claims that are marked as "Read", and do not include claims before the Employer's "Effective Date"

```
/api/employer/99/claims?per_page=10&page=3&hide_read=on&hide_before_effective_date=on
```

Claims Response

```
{
  "data": [
     {
```

(continues on next page)

(continued from previous page)


```
        "amount_allowed": null,
        "amount_billed": 4509.00,
        "amount_not_covered": null,
        "amount_paid": 487.90,
        "amount_paid_other": null,
        "check_date": null,
        "check_number": null,
        "claim_medical_id": 9999999,
        "claim_medical_lines": [
        {
           "amount_allowed": 69.11,
           "amount_billed": 85.00,
           "amount_not_covered": null,
           "amount_paid": 0.00,
           "amount_paid_other": null,
           "claim_medical_line_id": 999999,
           "coinsurance_patient": 0.00,
           "copayment": 0.00,
           "date_of_service": {
              "bounds": "[)",
              "end": "2016-10-26",
              "start": "2016-10-25"
           },
           "diagnosis_code": null,
           "discount": null,
           "patient_responsibility": null,
           "polymorphic__amount_allowed": null,
           "polymorphic__amount_billed": null,
           "polymorphic__amount_paid": null,
           "polymorphic__coinsurance_patient": null,
           "polymorphic__copayment": null,
           "polymorphic__patient_responsibility": null,
           "polymorphic__reduction": null,
           "procedure_code": null,
           "procedure_name": "Office/outpatient Visit, Est",
           "reduction": 69.11,
           "total_patient_responsibility": 69.11,
           "vendor_system_id": "0"
        }
        ],
        "claim_requests": [],
        "coinsurance_patient": 209.10,
        "copayment": 0.00,
        "createddate": "2017-05-28T06:47:16.361817-04:00",
        "dataobject_id": 9999,
        "date_of_service": null,
        "dependents": [
        {
           "alegeus_key": null,
           "createddate": "2018-03-29T08:47:12.044480-04:00",
           "datapath_key": null,
           "email": null,
           "first_name": "Johnny",
           "generic_key": null,
           "id": 99999,
           "last_name": "Appleseed",
           "modifieddate": "2018-03-29T08:47:12.044480-04:00",
```


(continues on next page)

(continued from previous page)


```
            "ssn": null,
            "wex_key": null
        }
        ],
        "discount": null,
        "eob_date": null,
        "group_name": null,
        "group_number": null,
        "id": 476877,
        "last_updated_status": "2017-05-28T06:47:16.361817-04:00",
        "members": [
        {
            "email": "johnny@appleseed.com",
            "employer_id": 99999,
            "employer": {
                "id": 99999,
                "name": "Fruit Tree Planting Services, LLC",
                "reimbursement_policy": "off"
            },

            "full_name": "Johnny Appleseed",
            "id": 888888
        }
        ],
        "modifieddate": "2017-05-28T06:47:16.361817-04:00",
        "network": null,
        "patient_account_number": null,
        "patient_name": "Jimmy Appleseed",
        "patient_responsibility": 3959.10,
        "policy_holder": {
        "fullname": "Johnny Appleseed",
        "policy_holder_id": 888888
        },
        "policy_holder_fullname": "Johnny Appleseed",
        "policy_holder_id": 888888,
        "polymorphic__amount_allowed": null,
        "polymorphic__amount_billed": null,
        "polymorphic__amount_paid": null,
        "polymorphic__coinsurance_patient": null,
        "polymorphic__copayment": null,
        "polymorphic__patient_responsibility": null,
        "polymorphic__reduction": null,
        "processed_on": "2016-10-15",
        "read": [],
        "read_all": [],
        "reduction": 0.00,
        "remarks": null,
        "service_provider": "Dr. Suess",
        "service_provider_address": null,
        "service_provider_billing_address": null,
        "service_provider_billing_name": null,
        "service_provider_billing_npi_number": null,
        "service_provider_billing_number": null,
        "service_provider_billing_phone": null,
        "service_provider_npi_number": null,
        "service_provider_number": null,
        "status": "Processed",
```


(continues on next page)

(continued from previous page)


```
        "total_patient_responsibility": 69.11,
        "status": "Processed",
        "tpafiles": [
        {
           "extension": ".png",
           "tpafile_id": 99999,
           "url": "/claim_medical/99999/tpafile/88888"
        },
        {
           "extension": ".pdf",
           "tpafile_id": 44444,
           "url": "/claim_medical/99999/tpafile/88888"
        }
        ],
        "type": {
        "name": "dental",
        "type_id": 2
        },
        "vendor_system_id": "xxxxx122344"
     }
  ],
  "has_next": true,
  "has_prev": false,
  "next_num": 2,
  "page": 1,
  "pages": 9999,
  "per_page": 1,
  "prev_num": 0,
  "total": 9999
  }
```


Employer Response


```
{
  "data": [
     {
        "accounts": [],
        "alegeus_key": null,
        "can_request_reimbursements": false,
        "can_use_portal": false,
        "createddate": "2016-12-10T12:17:09.497104-05:00",
        "datapath_key": "99999",
        "easy_enroll_ssn_required": true,
        "effective_date": "2016-05-01",
        "email_automation": true,
        "employer_id": 5555555,
        "generic_key": null,
        "is_demo": false,
        "modifieddate": "2017-02-28T18:07:03.799519-05:00",
        "name": "Dunder Mifflin Paper Company",
        "onboard_field_send_reimbursement": "all",
        "onboard_url": "https://www.easyenrollment.net/enroll/ddddd",
        "payers": [
        {
           "logo_url": "https://s3.amazonaws.com/tpastream-public/HorizonBlue-Logo-
↪Updated-Jan15.jpg",
           "name": "Horizon Blue Cross Blue Shield of New Jersey",
```


(continues on next page)

(continued from previous page)


```
            "payer_id": 33,
            "retriever": "horizon_bluecross.HorizonBlue",
            "short_name": "Horizon BCBS NJ"
        }
        ],
        "send_new_claim_emails": false,
        "slug": "dddd",
        "support_email": null,
        "support_email_derived": "support@my-tpa.com",
        "support_phone": null,
        "support_phone_derived": "(800) 999-9999",
        "team_primary": null,
        "team_primary_id": null,
        "teams": [],
        "tenant": {
        "logo_url": "https://s3.amazonaws.com/tpastream-public/xxxxxxx.png",
        "name": "My TPA",
        "tenant_id": 99999
        },
        "unread_count": 333,
        "wex_key": null
    }

],
"has_next": true,
"has_prev": false,
"next_num": 2,
"page": 1,
"pages": 66,
"per_page": 1,
"prev_num": 0,
"total": 66
}
```

CHAPTER 2

Indices and tables

- genindex
- search