PERKY BENEFITS

Integration Guide

June 2022

perky.co

GETTING STARTED

I. DATA REQUIREMENTS

II. REQUIRED FILES

III. ESTABLISH A CONNECTION

IV. SETUP THE BENEFITS MANAGER

V. INITIALIZE THE GUIDE

VI. CUSTOMIZE YOUR INTEGRATION

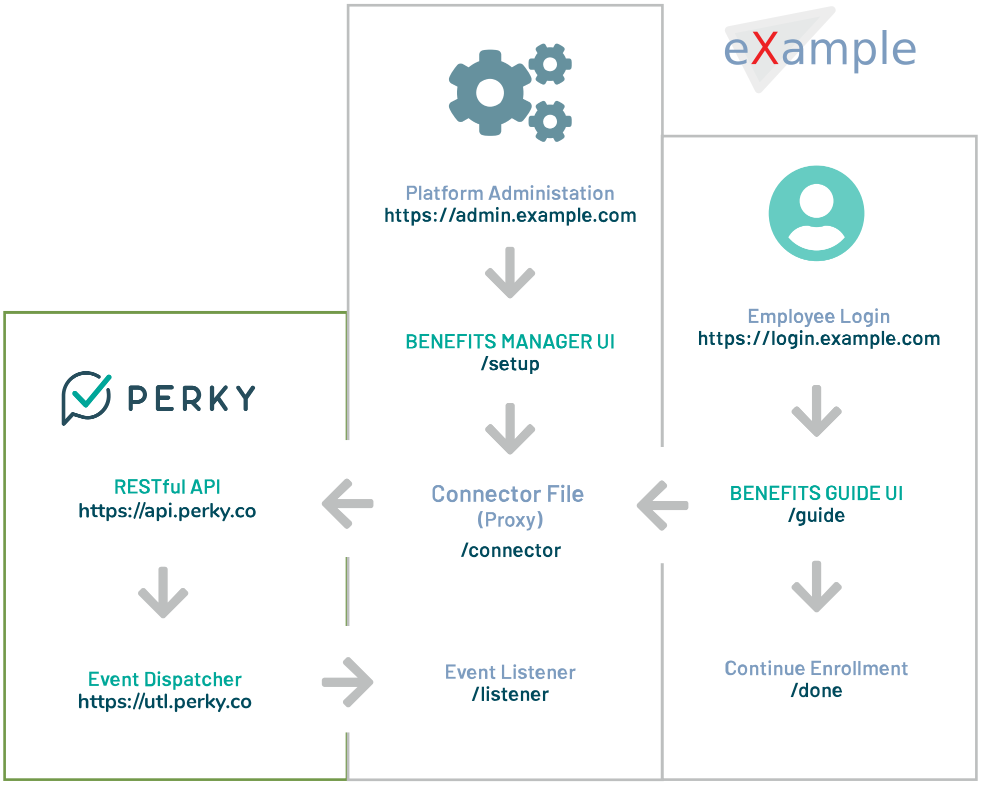
VII. CAPTURING EVENTS

VIII. USER EXPERIENCE EXAMPLES

IX. PROGRAMMATIC ACCESS

X. DEVELOPER PORTAL

Getting Started



THIRD-PARTY PLATFORM INTEGRATIONS

Our API provides programmatic access to all data AND can render the interfaces required to configure benefits information as well as deliver the guide experience to your end-users.

This allows you to leverage a RESTful API to setup benefits and save and retrieve data stored in our system at any time. You can rebrand our decision support tools and host them under any domain.

You can bring the Benefits Guide directly into your employee portal and use the outputs as you see fit.

The primary driver to this implementation is the API Connector which is a file hosted by the integrating platform that acts as a proxy for the other components. This file contains your API key and retrieves data requested by the Benefits Manager and Benefits Guide using established industry HTTP protocols.

The source files to get started are shown throughout this section but if you want, you can clone our sample repository to help you get started.

```
$ git clone git@bitbucket.org:perkytech/perky-sample.git
```

ENVIRONMENT

The API is hosted on Amazon Web Services (AWS) and deployed using Aptible's HIPAA compliant architecture. We maintain two isolated environments for development and production purposes.

Environment	URL	PHI-Ready
Development / Staging	https://api.perky.dev	No
Production / Live Application	https://api.perky.co	Yes

All development & testing should be done using https://api.perky.dev and updated to https://api.perky.co when ready for production. A Production-Ready API key will be provided upon completion of the integration.

I. DATA REQUIREMENTS

REQUIRED INFORMATION

Benefits Package & Plan Identifiers

First, we require that a “package” object containing the plan options available for a group of users be created and stored in our system via the API. The package can be referenced by any unique identifier used by the third-party (typically the id that is used to provide a group of employees their enrollment options).

User Identification

Similar to synchronizing package identifiers, we require a user id to be passed to initialize the request. This identifier can be used to associate results back to the platform as well as be included in any session results that are queried via the API directly. One instance of a guide can be initialized per user; however, each package will maintain its own dataset per user (identified by an immutable session key).

Benefits Eligibility

Each benefit type can be configured by eligibility group (consisting of a unique key). For example, if an employer has different premium contributions for Full-Time and Part-Time employees; these can be defined as groups under the health benefit. You can then include or exclude these from different plan options and/or define different premiums and contributions for each. When a user runs the guide, their “group” eligibility property should be passed in (e.g., _fulltime, _parttime) with the request. It is also possible to send in an array of plan ids to determine an employee’s eligibility. See Initialize the Guide for more information.

Required User Data / Optional Dependents

Enrollment platforms already have most of the personal information required to run the guide stored in their system. Passing key demographic and dependent data in the initial request would eliminate the need for it to be keyed in again by the user but is not required. We require relation, birthdate, and gender for each member passed which is used to determine eligible coverage types and average health utilization.

First Name	Relationship [self, spouse, child]	Sex [male, female, other]	Date of Birth
John	self	male	1970-01-01
Jane	spouse	female	1970-01-01
Jack	child	male	2000-01-01

ADDITIONAL INFORMATION

These can be used to soft default user inputs with values already known by the system of record.

Description	Property	Example
Eligibility Object	elig	{“health”:{“group”:”_fulltime”, “plans”:[123, 456]}}
Zip Code	zip	14450
Annual Compensation	comp	50000
Estimated Household Income	income	150000
401(k)/403(b) Deferral Percentage	retire	3.2
Last Known HSA Balance	hsa_balance	6500
Annual HSA Contribution	hsa	1350
Pay Periods (a Year)	pay_periods	26
Hire Date	hire_date	2020-01-01

II. Required Files

INTEGRATION NEEDS

There are four files required to facilitate a complete integration.

File / Sample URL	Description
API Connector /connector	This file will authorize and facilitate communications between the API and your server using your unique API_KEY.
Benefits Manager /setup	This file will call and render the management UI to create and store the necessary JSON object.
Benefits Guide /guide	This file will initialize a token and render the guide UI for the end-user.
Webhook (optional) /listener	You can specify a URL to receive event notifications to update your own records.

The URLs of these files can be whatever you choose but we will use the following in our examples and highlight where you can change them to fit your own system requirements.

III. Establish a Connection

The integration requires a server-side script to send authenticated requests to the API. This script can utilize cURL or another server based technology to pass data to the endpoint and return JSON based responses. Requests are authorized by setting the HTTP_APIKEY header to the environments authorization key.

The connector file should be placed on it’s own web accessible path behind a user login. In our example, this file at /connector and should be updated to match your configuration. You will also want to include any user authentication methods used by your system to control access to this file (e.g., session/cookie).

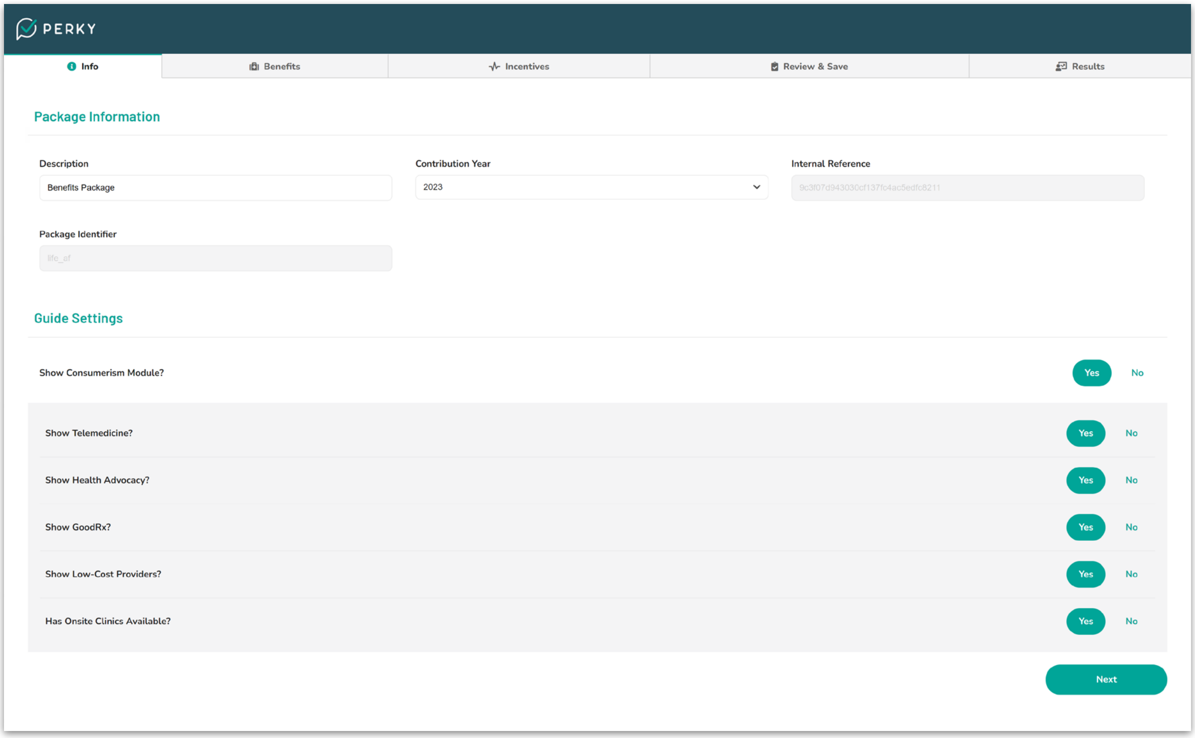
```
<?php
      //API PARAMETERS (Developer Supplied API Key)
      define(‘C_APIKEY’, ‘YOUR_KEY’);
      define(‘C_APIURL’, ‘https://api.perky.co/’);
      define(‘C_HOOK’, ‘https://www.example.com/listener’);

      //Simple cURL Based Request
      function ProcessRequest($endpoint, $data=’’) {
            $curl = curl_init();
            curl_setopt_array($curl, array(
                  CURLOPT_RETURNTRANSFER => 1,
                  CURLOPT_URL => C_APIURL.$endpoint,
                  CURLOPT_HTTPHEADER => array(‘APIKEY:’.C_APIKEY),
                  CURLOPT_POST => 1,
                  CURLOPT_POSTFIELDS => array(‘data’=>$data, ‘webhook’=>C_HOOK)
                  )
            );

            $resp = curl_exec($curl);
            curl_close($curl);

            return $resp;
      }

      //IF AJAX BASED REQUEST && ENDPOINT/ACTION PRESENT (INCLUDE USER AUTHORIZATION)
      if(isset($_SERVER[‘HTTP_X_REQUESTED_WITH’])
         && ($_SERVER[‘HTTP_X_REQUESTED_WITH’] == ‘XMLHttpRequest’)
         && isset($_POST[‘endpoint’])) {

            //OUTPUT/RETURN JSON RESPONSE
            echo ProcessRequest($_POST[‘endpoint’], $_POST[‘data’]);
      }
```

IV. Setup the Benefits Manager

BENEFITS SETUP

This is a file that takes a value representing the internal enrollment id and passes it to the initial AJAX request. The included JavaScript will receive the response and render the benefits setup screens to the specified container using the included script and styles automatically.

The package ID used will maintain a 1-to-1 relationship, so any existing data associated with that ID will be populated and/or created at that time. When the package is saved; a webhook event is triggered and the window will can be closed or returned to the referring URL using the “onsave” callback method. Additional user authentication should be added to this page. You can also include your own theme file to customize it to match your administrative environment.



Package Information

Description: Benefits Package

Contribution Year: 2023

Internal Reference: 9c3f07d943030cf137fc4ac5edfc8211

Package Identifier: life_af

Guide Settings

Show Consumerism Module? [✓ Yes] No

Show Telemedicine? [✓ Yes] No

Show Health Advocacy? [✓ Yes] No

Show GoodRx? [✓ Yes] No

Show Low-Cost Providers? [✓ Yes] No

Has Onsite Clinics Available? [✓ Yes] No

BENEFITS MANAGER

Configuration Options	Default	Description
url	/connector	The location to your connector file
con	body	HTML container to append the UI
init	/package/_{id}/edit	Initial API endpoint to call. Replace {id} with your value
on_start	pkg.set	The initial callback to render package UI w/ retrieved data
on_save	function()	Callback for the save function. You can use it to close the window or return to the original location.
lib	false	Allow plan library management (store/delete plans)

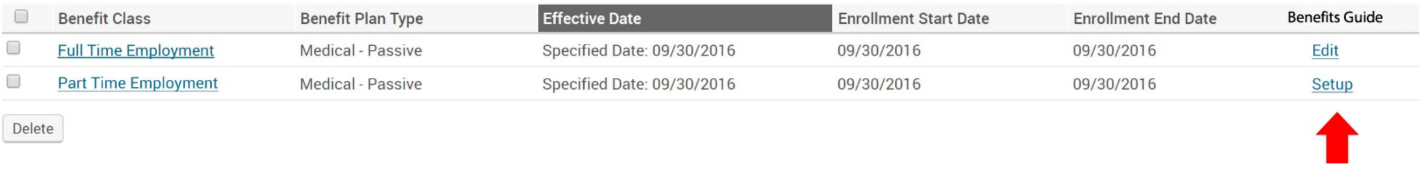
Example Code [/setup]

```
<!doctype html>
<html>
      <head>
            <meta charset=”utf-8”>
            <title>Benefits Manager</title>
            <script>
                  var pm_settings = {
                        ‘url’:’/connector’,
                        ‘init’:’package/_{id}/edit’,
                        ‘on_start’: ‘pkg.set’,
                        ‘on_save’: function() { location.href=’/enrollments’; },
                        ‘lib’: true
                  };
            </script>
            <script src=’https://api.perky.co/pm-js’></script>
            <link href=’https://api.perky.co/pm-css’ rel=’stylesheet’ type=’text/css’>

            <!--INCLUDE YOUR OWN CUSTOM THEME VARIABLES HERE -->
            <link href=’/theme.css’ rel=’stylesheet’ type=’text/css’>

      </head>
      <body>
            <div id=’header’><div id=’logo’></div></div>
      </body>
</html>
```

EXAMPLE ADMINISTRATION INTEGRATION

The process for integrating the Benefits Manager is going to differ from platform to platform. In this example, the third-party administration has a list of active enrollment periods.

Each of these has their own unique ID. This ID can be POSTed using a standard HTML form to the setup page which will then use this to associate data.



☐	Benefit Class	Benefit Plan Type	Effective Date	Enrollment Start Date	Enrollment End Date	Benefits Guide
☐	Full Time Employment	Medical - Passive	Specified Date: 09/30/2016	09/30/2016	09/30/2016	Edit
☐	Part Time Employment	Medical - Passive	Specified Date: 09/30/2016	09/30/2016	09/30/2016	Setup

Simple HTML Form Button to POST row ID 977736 to /setup

This button triggers a form POST containing your unique package identifier to the /setup page within your domain. The button text could also change to reflect presence of data using a /package/_{id} request and reading the response code (200 vs 401).

Example Code: Button Implementation

```
<form action=’/setup’ method=’POST’ target=’perky’>
      <input type=’hidden’ name=’id’ value=’977736’ />
      <input type=’submit’ value=’Setup’ />
</form>
```

V. Initialize the Guide

REQUEST A TOKEN

For a user to interact with the guide, they need to request an authorization token from our API. This request will contain the user data needed to initialize the guide including the package ID, a unique user identifier, dependent information, and if necessary, the user's eligibility object. To do so, you will need to package all information in a json-encoded data string and use cURL (or applicable) to POST this information to the /connector file using the /session endpoint.

The API will validate the information and return a client-side token which is valid for 24 hours. Any repeated requests for that user and package combination will renew the token, and information that was previously submitted will be replaced under that identifier (i.e., dependents changed). If the guide is partially completed, it will resume where the user left off. Information can be retrieved either though the session reference that is generated or by receiving the webhook for the user after the guide has been completed.

Required Data

Property	Example Value	Description
elig	object	The eligibility definition consisting of the employee's contribution group and eligible plans per benefit type.
type	health	Benefit type (health, dental, vision, accident, hospital, critical, std, ltd, dcap, retire, life).
group	_fulltime	Contribution / Eligibility Group
plans	[123,456]	Array of eligible plan id's
members	array	An array containing the user's demographic information.
name	John	John
relation	self, spouse, child	Relation to "self"
gender	male, female, other	Gender
dob	1970-01-01	Date of Birth
id	123	An internal identifier for the dependent (optional)
life	[12000,30000]	Existing Life Coverage [employer, outside] (optional)

Optional Configuration

Property	Values	Description
manage_dependents	**true**, false	Allow entry of new dependents, if false, depends_url required
dependents_url	/dependents	Where can the user go to update dependents?
return_url	/done	Where should the user return if they go back?
company	Sample Company	Company Name
pay_periods	52|26|24|12|	Number of pay periods per year
version	**none**, compare_only, skip_contributions, skip_contributions_keep_dental, skip_savings	Version of the guide to run
audio	**true**, false	Use the audio guided experience?
always_show_dv	true, **false**	Always show dental /vision (even if recommendation is to waive)
pdf	**true**, false	Allow PDF export?
feedback	**true**, false	Collect feedback from the user at the end?
hire_date	2020-02-01	Date of Hire
privacy_url	https://example.com/privacy	Include external link to company privacy policy?
terms_url	https://example.com/terms	Include external link to company terms of use?

Example Code: Request a Token

```
<?php
      //SETUP DATA
      $data = (object) array(‘ref’=>’{ref}’, ‘package’=>’{package}’, ‘user’=>’{user}’,
‘manage_dependents’=>true ‘return_url’=>’/’, ‘company’=>’{company}’);

      //WHAT IS THE EMPLOYEE’S ELIGIBILITY IF REQUIRED?
      $data->elig = (object) array(
            ‘health’=>(object) array(‘group’=>’_fulltime’, ‘plans’=>array(123, 456)),
            ‘dental’=>(object) array(‘group’=>’default’, ‘plans’=>array(ABC, DEF)),
            ‘vision’=>(object) array(‘group’=>’_corporate’, ‘plans’=>array(789, XYZ))
      );

      //PRIMARY MEMBER (Required)
      $data->members[] = array(‘name’=>’{name}’, ‘relation’=>’self’,
‘gender’=>’{gender}’, ‘dob’=>’{dob}’);

      //ADD OPTIONAL DEPENDENTS IF AVAILABLE
      //$data->members[] = array(‘name’=>’Jane’, ‘relation’=>’spouse’,
‘gender’=>’female ‘, ‘dob’=>’1970-01-01’);
      //$data->members[] = array(‘name’=>’Jack’, ‘relation’=>’child’, ‘gender’=>’male
‘, ‘dob’=>’2000-01-01’);

      //MAKE CALL TO /CONNECTOR
      $curl = curl_init();
      curl_setopt_array($curl,
            array(
                  CURLOPT_RETURNTRANSFER => 1,
                  CURLOPT_URL => ‘/connector’,
                  CURLOPT_POST => 1,
                  CURLOPT_HTTPHEADER => array(“X-Requested-With: XMLHttpRequest”),
                  CURLOPT_POSTFIELDS => array(‘endpoint’=>’session’, ‘data’=>json_
encode($data))
            )
      );

      $res = json_decode(curl_exec($curl));

      curl_close($curl);

      //$res->token now contains a valid client token (string);
```

STARTING THE GUIDE

A page with the following code needs to be created on a path accessible to the employee (/guide). This will accept the token and render the guide automatically using the externally loaded js and css files.

Example Code

Additional content should be added to this page for user authentication, as well as your own css theme file.

```
<!doctype html>
<html>
      <head>
            <meta charset=”utf-8”>
            <title>Benefits Guide</title>
            <script>
            var pg_settings = {
                  ‘url’:’/connector’,
                  ‘token’: ‘{token}’,
                  ‘on_complete’: function() { location.href = ‘/done’; },
                  ‘on_error’: function(e) { console.log(e); }
            };
            </script>
            <script src=’https://api.perky.co/pg-js’>/script>
            <link href=’https://api.perky.co/pg-css’ rel=’stylesheet’ type=’text/css’>

            <!--INCLUDE YOUR OWN CUSTOM THEME VARIABLES HERE -->
            <link href=’/theme.css’ rel=’stylesheet’ type=’text/css’>
      </head>
      <body></body>
</html>
```

Configuration Options

Property	Values	Description
url	/connector	Your package identifier.
con	body|#guide	Container to render the UI in /guide HTML
token	A Valid Token	Client-side token
on_complete	function()	Callback function when the guide is complete (e.g., close the window, return to enrollment)
on_error	function()	Callback function when an error has occurred

VI. CUSTOMIZE YOUR INTEGRATION

ADJUST YOUR THEME VARIABLES

Because the HTML files reside on your own server; you can add or overwrite any of the existing css variables as you see fit to match your integration. You can also add any surrounding HTML to inject these modules into your pages. The variables are the same for both the guide and setup pages.

```
/*!

theme.css
Copyright (c) 2021 PERKY
By setting the following variables, you can quickly customize the the look of the
Benefits Manager, without directly editing the CSS.

/*IMPORT GOOGLE FONTS */
@import url(“https://fonts.googleapis.com/css2?family=Barlow:wght@300;400;500;600”);
@import url(“https://fonts.googleapis.com/css2?family=Nunito:wght@300;400;500;700”);

/* STYLE VARS */
:root {
--pc-primary: #00a598;
--pc-primary-50: hsl(175, 100%, 96%);
--pc-secondary: #244C5A;
--pc-overlay-text: #FFFFFF;
--pc-error: #DF3030;
--pc-error-50: rgba(223, 48, 48, .13);
--pc-warning: #F18532;
--pc-success: #4F8235;
--pc-success-50: rgba(79, 130, 53, .13);
--pc-info: #08789D;
--pc-logo: url(https://cdn.perky.co/logo.svg);
--pc-font-heading: ‘Barlow’;
--pc-font-body: ‘Nunito’;
--pc-border-radius: 24px;
--pc-border-radius-card: 12px;
}
```

VII. Capturing Events

SETTING UP A WEBHOOK

If you specify a webhook url in your connector file, it will be recorded with each request. Specific events are setup to report back to the specified url when they occur. You can capture these events and update your database and system acordingly. Webhooks are dispatched every minute, and will try up to 5 times to receive a 200 HTTP response from the specified url. You can view pending and dispatched webhooks for your implementation in Event Notifications in the Developer Portal.

Event Types

Webhook Event	Action	Matches
Package	Add	package/add
Package	Delete	package/%/delete
Package	Update	package/%/update
Session	Init	session
Session	Delete	session/%/delete
Guide	Step	guide/%/compare, guide/%/review

Event Format

Format	Type	Values	Description
type	string	package, session, guide	The type of event
action	string	add, update, delete	The action of the event
ref	string	a6c5e214962bdb975fae4bd9eed690fc	The reference of the package
package	string	demo	The unique identifier of the package
session	string	55a8c7f2911b74ce40f5c38a8a684d06	The session reference
user	string	123	The value supplied to identify the user
ts	datetime	2020-03-14 09:08:04	Time the event occurred

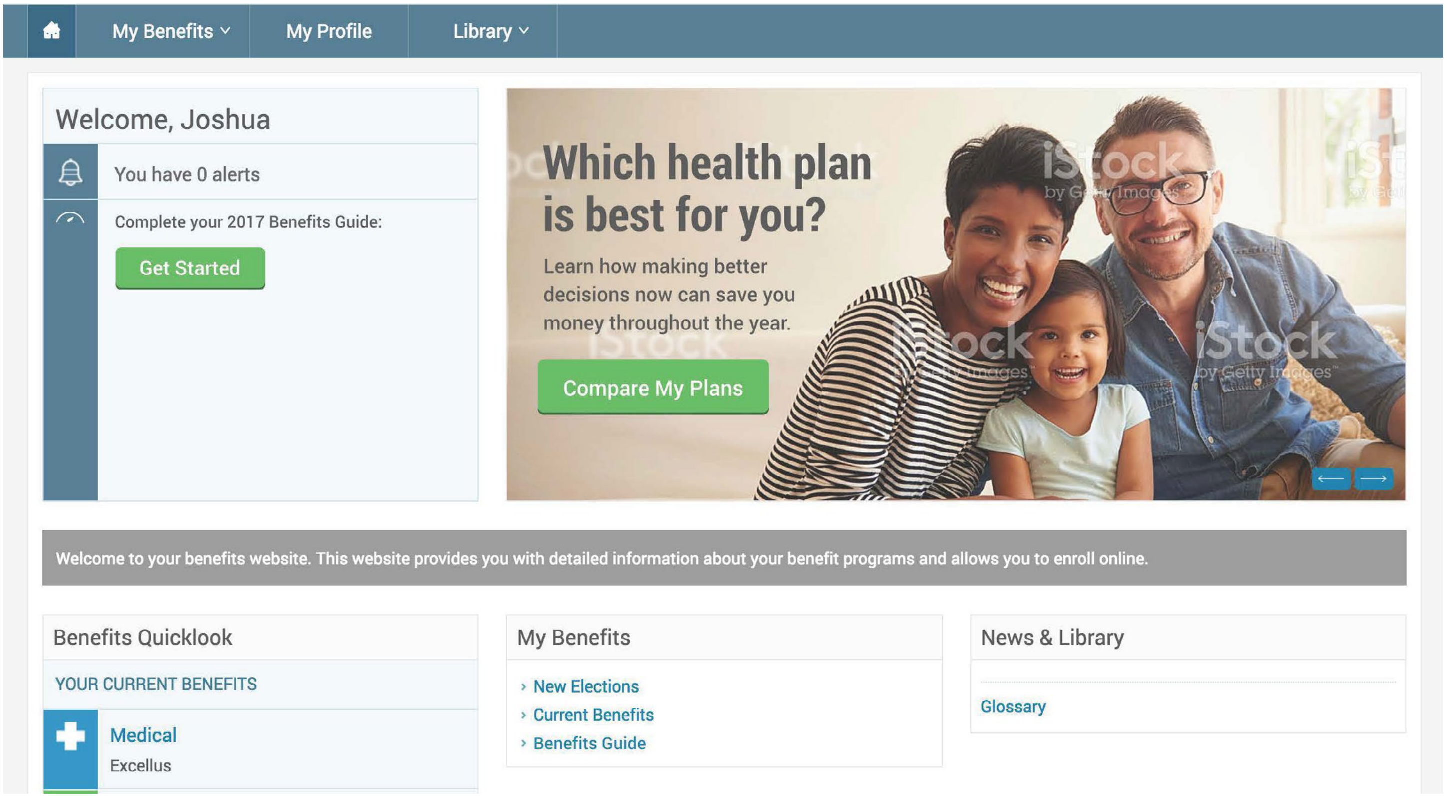
Example Event Notification

```
[
   {
      “type”: “package”,
      “action”: “add”,
      “ref”: “ff4f42c6608e404799fb9f969658c652”,
      “package”: “demo”,
      “ts”: “2018-03-14 09:08:04”
   },
   {
      “type”: “guide”,
      “action”: “review”,
      “ref”: “ff4f42c6608e404799fb9f969658c652”,
      “package”: “demo”,
      “session”: “55a8c7f2911b74ce40f5c38a8a684d06”,
      “user”: “123”,
      “ts”: “2020-03-14 11:09:32”
   }
]
```

Webhook Listener Code Example

```
<?php

   if(isset($_POST[‘events’])) {
      $events = json_decode(stripslashes($_POST[‘events’]));

      if(is_array($events) && count($events)) {

         foreach($events as $event) {

            //CHECK EVENT TYPE & ACTION
            if($event->type==’guide’) {

               //GET SESSION REFERNCE
               $session = get_guide_results(‘/session/’.$event->session.’/results’);

               //FIND LOCAL USER BY ID
               $user_id = get_user($event->user);

               //FIND LOCAL ENROLLMENT
               $package_id = get_package($event->package);

               //UPDATE LOCAL USER SETTINGS
               $health_id = $session->results->health->sel->id;
               $hsa_contrib = $session->results->contrib->hsa;

               //SAVE LOCAL USER
               save_user();

            }

         }

      }

   }
```

XIII. User Experience Examples



Welcome, Joshua

You have 0 alerts

Complete your 2017 Benefits Guide:

Which health plan is best for you?

Learn how making better decisions now can save you money throughout the year.

Welcome to your benefits website. This website provides you with detailed information about your benefit programs and allows you to enroll online.

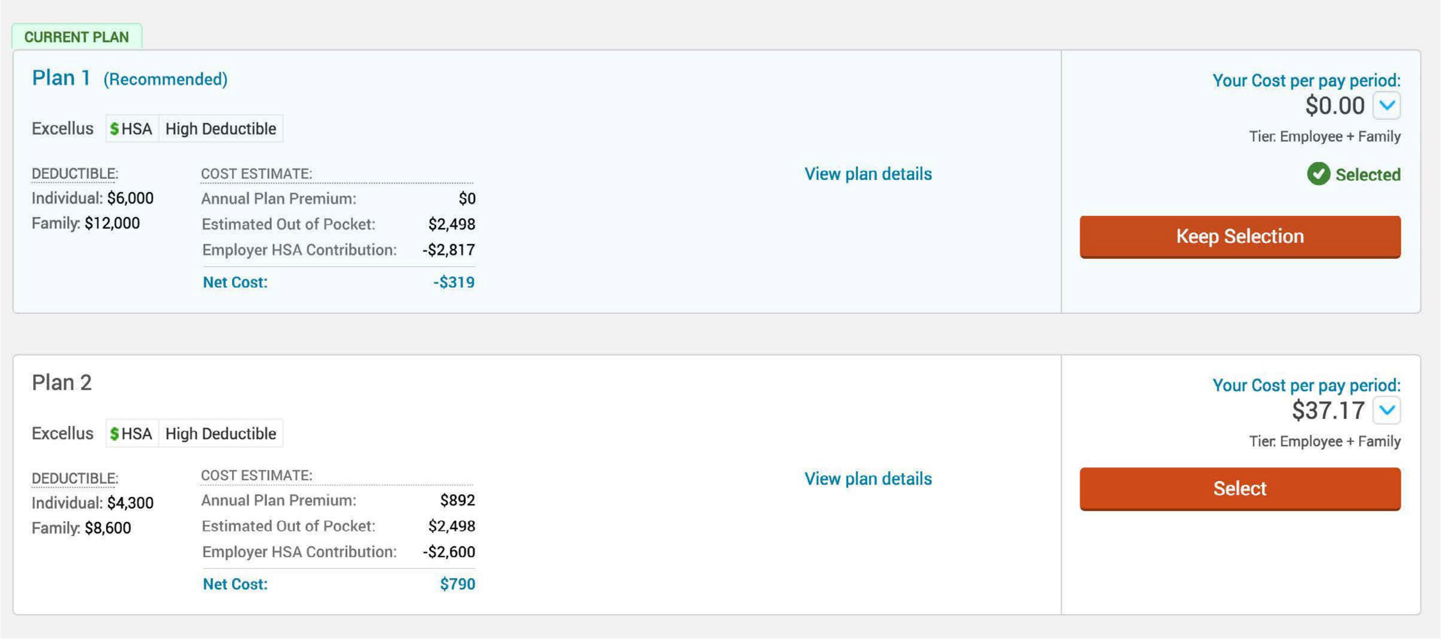
DASHBOARD / WELCOME PAGE

When the Benefits Guide is enabled for a client's enrollment period, you could add a banner and side bar task to the main page which links to /guide. Triggering this page will render/restore the guide and associated data for that user. If the primary enrollment task could be disabled until the user have completed the guide, that would be the most effective in driving plan adoption. Once the user completes the guide, they can be redirected to the enrollment confirmation page.

ENHANCING THE ENROLLMENT PROCESS

Recommended Plan

Highlight the recommended and/or selected plan for the user during their comparison and expose a total estimated cost alongside those details. It is important to note that many plan comparisons do not account for the Employer's HSA contribution which helps offset the net cost of each plan. We hope this can become a factor in the employee's decision-making process.



Plan 1 (Recommended)

Excellus $HSA High Deductible

DEDUCTIBLE:	COST ESTIMATE:	
Individual: $6,000	Annual Plan Premium:	$0
Family: $12,000	Estimated Out of Pocket:	$2,498
	Employer HSA Contribution:	-$2,817
	Net Cost:	**-$319**

Your Cost per pay period: $0.00

Tier: Employee + Family

✓ Selected

Plan 2

Excellus $HSA High Deductible

DEDUCTIBLE:	COST ESTIMATE:	
Individual: $4,300	Annual Plan Premium:	$892
Family: $8,600	Estimated Out of Pocket:	$2,498
	Employer HSA Contribution:	-$2,600
	Net Cost:	**$790**

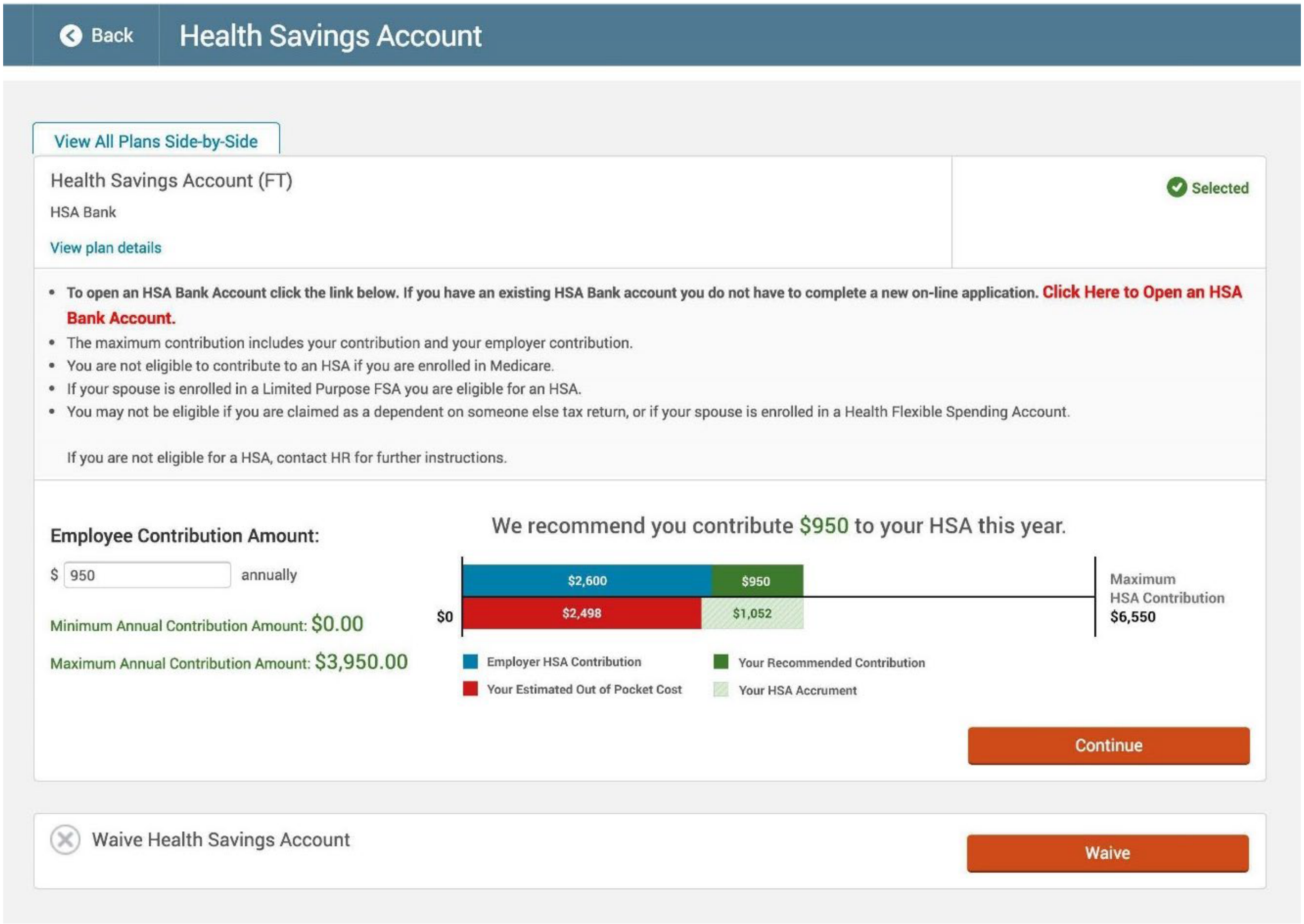
Your Cost per pay period: $37.17

Tier: Employee + Family

Note: The above content demonstrates a 1/1 enrollment period. You will need to determine prorated values for new hires and off-year enrollment while adjusting out of pockets for partial year utilization.

Pre-Populate Contributions

Pre-populate the employee’s recommended HSA contribution (showing both employer and employee contributions as it relates to the annual maximums) alongside plan metrics.



Health Savings Account (FT)

HSA Bank

- **To open an HSA Bank Account click the link below. If you have an existing HSA Bank account you do not have to complete a new on-line application. Click Here to Open an HSA Bank Account.**
- The maximum contribution includes your contribution and your employer contribution.
- You are not eligible to contribute to an HSA if you are enrolled in Medicare.
- If your spouse is enrolled in a Limited Purpose FSA you are eligible for an HSA.
- You may not be eligible if you are claimed as a dependent on someone else tax return, or if your spouse is enrolled in a Health Flexible Spending Account.

If you are not eligible for a HSA, contact HR for further instructions.

Employee Contribution Amount:

$ 950 annually

Minimum Annual Contribution Amount: $0.00

Maximum Annual Contribution Amount: $3,950.00

We recommend you contribute $950 to your HSA this year.

$0
$2,600
$950
$2,498
$1,052
Maximum HSA Contribution $6,550
Employer HSA Contribution
Your Recommended Contribution
Your Estimated Out of Pocket Cost
Your HSA Accrument

Waive Health Savings Account

Note: This graph demonstrates the potential to visually represent the data, as it relates to estimated out-of-pocket and long-term savings. Any data visualizations contained outside of the guide experience would be the responsibility of the integrating platform.

IX. PROGRAMMATIC ACCESS

REST API EXAMPLES

Additional methods are available if you wish to pull data directly from our API for reporting purposes. Here are a few common use request and response examples. You can test these calls at https://docs.perky.co/browse.

Get All Packages

This endpoint will allow you to retrieve all packages stored in our system. A list of all tags and other helpful properties will also be returned in order to further display and filter results.

Endpoint: /packages

API Response:

```
{
      “version”: “1.0”,
      “endpoint”: “packages”,
      “code”: 200,
      “packages”: [
        {
            “ref”: “80d885f5ffd9fc43ca8a18f49e768e4d”,
            “desc”: “Package 1”,
            “package”: “125”,
            “tags”: [“34”],
            “created”: “2016-01-01 00:00:00”
        },
        {
            “ref”: “8a1db45971948145cfd53e46cfc2d247”,
            “desc”: “Package 2”,
            “package”: “456”,
            “tags”: [“35”, “36”],
            “created”: “2016-01-01 00:00:00”
        }
      ],
      “tags”: [“34”, “35”, “36”]
}
```

Get Packages (Filtered by Tag)

This endpoint will allow you to retrieve all packages stored in our system under a given tag. This can be used if packages are setup under employer identifiers and an additional layer of management is needed.

Endpoint: /packages/34

API Response:

```
{
      “version”: “1.0”,
      “endpoint”: “packages/34”,
      “code”: 200,
      “packages”: [
        {
            “ref”: “80d885f5ffd9fc43ca8a18f49e768e4d”,
            “desc”: “Package 1”,
            “package”: “125”,
            “tags”: [“34”],
            “created”: “2016-01-01 00:00:00”
        }
      ],
      “tags”: [“34”, “35”, “36”],
      “tag”: “34”
}
```

Get Package

This can be used to retrieve the entire package object. You can get the package either by reference, or by using your own internal package ID when it was created (designated by an underscore).

Endpoint: /package/80d885f5ffd9fc43ca8a18f49e768e4d

Alternate Endpoint: /package/_125

API Response: Upon Request

Delete Package

You can remove a package from our system by calling the delete endpoint by reference or package ID.

Endpoint: /package/80d885f5ffd9fc43ca8a18f49e768e4d/delete

Alternate Endpoint: /package/_125/delete

API Response:

```
{
      “version”: “1.0”,
      “endpoint”: “package/80d885f5ffd9fc43ca8a18f49e768e4d/delete”,
      “code”: 200,
      “result”: “true”,
      “ref”: ”80d885f5ffd9fc43ca8a18f49e768e4d”,
      “package”:”125”
}
```

Get All Sessions for a Package

Get a list of all sessions created under a package. This can be used to find a session key for a particular user or to determine how many individuals within a group have started a session.

Endpoint: /package/80d885f5ffd9fc43ca8a18f49e768e4d/sessions

Alternate Endpoint: /package/_125/sessions

API Response:

```
{
      “version”: “1.0”,
      “endpoint”: “package/80d885f5ffd9fc43ca8a18f49e768e4d/sessions”,
      “code”: 200,
      “ref”: “80d885f5ffd9fc43ca8a18f49e768e4d”,
      “package”: “125”,
      “sessions”: [
        {
            “ref”: “3a45526dbd942f737303dac9aecdd55e”,
            “token”:”sxGcJazN1597FBB60Fou72FPl1tW44T2”,
            “user”:”1”,
            “created”: “2016-09-21 12:24:03”,
            “expires” “2016-09-22 12:24:03”
        },
        {
            “ref”: “b4e0ccc28b0c6d1e353211f65ee96ef2”,
            “token”:”8a1db45971948145cfd53e46cfc2d247”,
            “user”:”2”,
            “created”: “2016-09-24 11:02:25”,
            “expires” “2016-09-25 11:02:25”
        }
      ]
}
```

Package Results

Get a broken-down grid of all plan recommendations for the requested package.

Endpoint: /package/80d885f5ffd9fc43ca8a18f49e768e4d/results

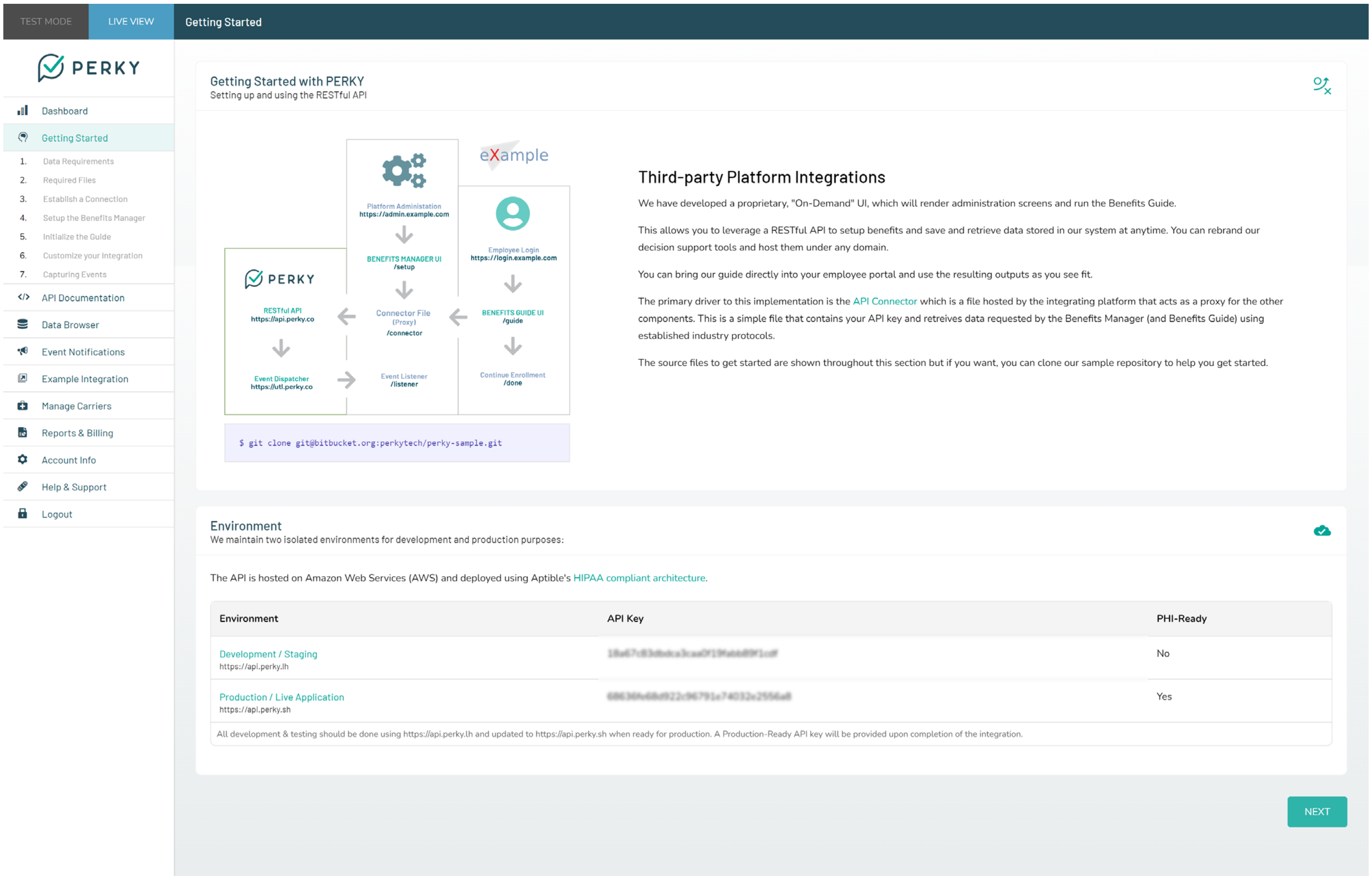
API Response:

```
{
  “version”: “82b59edd63a1df208adf475365099f7fa708895a”,
  “endpoint”: “package/532e3e4af9542740448e07821c474cdf/results”,
  “code”: 200,
  “ref”: “532e3e4af9542740448e07821c474cdf”,
  “desc”: “Full Guide”,
  “package”: “vb”,
  “year”: “2020”,
  “total”: 0,
  “legend”: {
    “health”: {
      “types”: {
        “single”: “Single”,
        “two_person”: “Two Person”,
        “family”: “Family”
      },
      “plans”: {
        “1576609204557”: “Gold Plan”,
        “1576609716422”: “Silver Plan”,
        “1576609789884”: “Bronze Plan”
      }
    },
    “accident”: {
      “types”: {
        “single”: “Single”,
        “employee_spouse”: “Employee + Spouse”,
        “employee_child”: “Employee + Child”,
        “family”: “Family”
      },
      “plans”: {
        “1573675330720”: “Accident Plus”
      }
    },
    “hospital”: {
      “types”: {
        “single”: “Single”,
        “employee_spouse”: “Employee + Spouse”,
        “employee_child”: “Employee + Child”,
        “family”: “Family”
      },
      “plans”: {
        “1568933005897”: “Hospital Plus”
      }
    },
    “critical”: {
      “types”: {
        “single”: “Single”,
        “two_person”: “Two Person”,
        “family”: “Family”
      },
      “plans”: {
        “1576610084863”: “Group Disease”
      }
    },
    “dental”: {
      “types”: {
        “single”: “Single”,
        “employee_spouse”: “Employee + Spouse”,
        “employee_children”: “Employee + Children”,
        “family”: “Family”
      },
      “plans”: {
        “1567795172149”: “Dental Plus”
      }
    },
    “vision”: {
      “types”: {
        “single”: “Single”,
        “employee_spouse”: “Employee + Spouse”,
        “employee_children”: “Employee + Children”,
        “family”: “Family”
      },
      “plans”: {
        “1568747137376”: “Vision Plus”
      }
    },
    “groups”: {
      “default”: “All Employees”
    }
  },
  “created”: “2019-08-07 20:06:56”,
  “modified”: “2020-09-16 17:48:12”,
  “completed”: 0,
  “grid”: {},
  “results”: {}
}
```

X. Developer Portal

DOCUMENTATION

All implementation documentation, sample resources, data-browser and API endpoints are available on our developer Access Portal located at https://docs.perky.co (login required).



Getting Started with PERKY

Setting up and using the RESTful API

```
$ git clone git@bitbucket.org:perkytech/perky-sample.git
```

Third-party Platform Integrations

We have developed a proprietary, "On-Demand" UI, which will render administration screens and run the Benefits Guide.

This allows you to leverage a RESTful API to setup benefits and save and retrieve data stored in our system at anytime. You can rebrand our decision support tools and host them under any domain.

You can bring our guide directly into your employee portal and use the resulting outputs as you see fit.

The primary driver to this implementation is the API Connector which is a file hosted by the integrating platform that acts as a proxy for the other components. This is a simple file that contains your API key and retreives data requested by the Benefits Manager (and Benefits Guide) using established industry protocols.

The source files to get started are shown throughout this section but if you want, you can clone our sample repository to help you get started.

Environment

We maintain two isolated environments for development and production purposes:

The API is hosted on Amazon Web Services (AWS) and deployed using Aptible's HIPAA compliant architecture.

Environment	API Key	PHI-Ready
Development / Staging https://api.perky.lh	[redacted]	No
Production / Live Application https://api.perky.sh	[redacted]	Yes

All development & testing should be done using https://api.perky.lh and updated to https://api.perky.sh when ready for production. A Production-Ready API key will be provided upon completion of the integration.