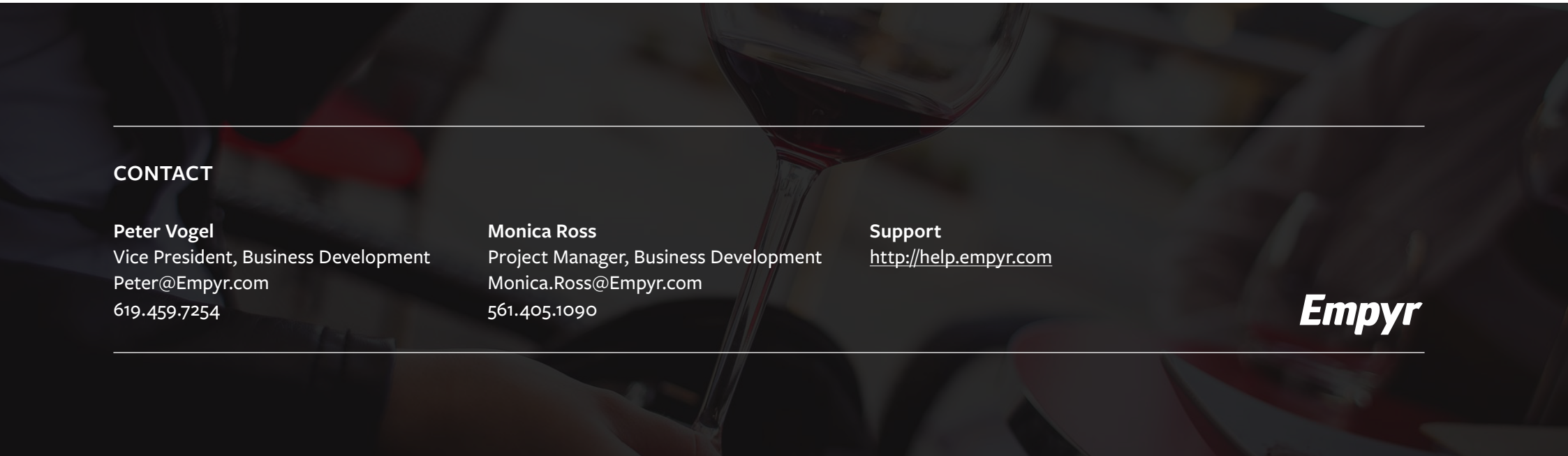
Empyr

API Guide - Publishers

Get up and running with the Empyr API



CONTACT

Peter Vogel
Vice President, Business Development
Peter@Empyr.com
619.459.7254

Monica Ross
Project Manager, Business Development
Monica.Ross@Empyr.com
561.405.1090

Support
http://help.empyr.com

Empyr

Empyr

Table of Contents

Click to jump to specific section.

Overview

Introduction

This guide is intended to educate developers on the basics of integration with the Empyr API. This guide, and the accompanying "Try it Now" tool available for all endpoints in the Mogl API, provides all the necessary information for a standard integration. For non-standard integrations, you should contact your account manager for further details. To find the "Try it Now" tool and Mogl API details, browse to the section titled, "Accessing the API" on page 3.

Audience

This document is intended for developers who will be integrating the Empyr API into their applications.

A note about compliance

You and Empyr jointly bear the PCI responsibility of the user's card information. There are several functionalities that Empyr has which will help to reduce your exposure to the full scope of PCI compliance, but it is important to understand, even if you are simply hosting a web page and soliciting card information from consumers which will be passed directly to the Empyr system, PCI compliance still applies to you.

Empyr

10,000 Foot View

A general publisher integration will involve several main components:

Enrollment

This stage includes enrollment of consumer cards.
- **Typical API endpoinit: /users/signupWithCard**
- or: Hosted Fields provides a web based solution without card data hitting your server
- or: client_usertoken grants provide a way to use the signupWithCard without data hitting your server.

Offer Display

This stage includes displaying the various offer content to your users.
- **Typical API endpoinit: /venues/search**
- In some cases, partners will choose to cache the data from /venues/search for 1 hour

Transact

Once the consumer has their card enrolled and sees the offers they will transact at the businesses.
- **Empyr will communicate to the partner webhook with the transaction details**
- Partner will typically send notification (push, email, sms) to their consumer informing them of the reward.

Payment

In this stage the consumer is paid. There are typically three approaches:
- Use the default Empyr pay schedule. Empyr will pay the “primary” card when the user’s balance in a month exceeds $10.
- Partner will use the /payables/add endpoint to pay a card. Empyr pays partner. Partner pays consumer.
- Partner will convert the payment to an app currency (e.g. points). Empyr pays partner. Partner pays consumer.

Empyr

10,000 Foot View

As illustrated, although the API has many methods the basic integration will only use a very small subset of the functionality available. Specifically, a typical integration will use one of several mechanisms to:

- Enroll Users
- Display Offers
- Capture Transactions
- Pay Consumers

It is up to the individual publisher how many additional methods they would like to interact with. The following page shows a small list of additional methods available in the API that may prove useful to a publisher.

10,000 Foot View

Endpoint Name	Availability	Description
/cards/setPrimary	test, prod	Sets the “primary” card. This method is only relevant when electing to use the Empyr payment schedule as this is the card that Empyr will pay credits to.
/cards/list	test, prod	This provides a list of the cards registered to the user.
/cards/delete	test, prod	Provides the ability to remove a card from a user.
/users/transactions	test, prod	Provides a transaction history for the user. Limit is 12 months of data.
/utilities/delete	test	Provides a mechanism to remove a user, the associated cards etc. This is useful for testing/unit tests.
/utilities/categories	test, prod	Returns a list of all the categories that businesses might belong to.
/utilities/features	test, prod	Returns a list of all the features that a business may have.
/utilities/txAuth	test	Provides a mechanism to test your webhook by sending an authorization transaction to your webhook. Must be a VISA card registered by your application.

Accessing the API

First, we need to make sure your code is working, and for this purpose, we have a test environment. In order to access the test environment we will need:

- **Contact information for a user who will receive notices about the API and credentials.**
- **IP addresses that will access the test environment.**
- **Holes in your firewall for us to access your environment for webhook events**

Once you have provided the above information we will make the appropriate changes in our environment and can provision you with a client key and client secret. These are used to authenticate with the Empyr API. Please note that the client key and secret will be different between the test and production environment. The client secret should be considered highly privileged information and its distribution should be limited and protected (especially the production secret).

API documentation can be found at https://test.mogl.com/api/docs/v2/. Access to test.mogl.com is limited to whitelisted IPs but can be browsed from https://www.mogl.com/api/docs/v2/ (although your test key and secret will not work):

All of the Empyr APIs have the ability to "Try It Now" and we encourage you to explore the API interaction using this functionality particularly when you are stuck or having difficulties.

Once you're seeing 200s, your webhook is getting the responses, and your code is correctly parsing the responses, you're nearly ready for production. The good news is that very little changes. You'll have a different client id and client secret, and instead of the urls all starting with test.mogl.com they'll all use www.mogl.com, but everything else remains the same. This makes it easy to use the same code for both environments, and just toggle a few values based on your target environment.

There are some additional questions that are required before we can move your application to production. Your account manager will collect that information from you and you will be assigned production credentials.

Test Environment Notes

There are some important notes that should be taken into consideration when working with the test environment:

- Contact information for a user who will receive notices about the API and credentials.
- **The test environment has restricted access from the public internet. You must provide us the IP address(es) that you will use to access this environment.**
- **DO NOT UPLOAD ANY REAL CARD DATA TO THIS ENVIRONMENT.** Test credit card numbers may be generated by using an online tool such as: http://www.getcreditcardnumbers.com/
- Images within the test environment are currently not accessible by your applications. However, many of these images are available by rewriting the image urls to point to the static production server. So, if an image url is "https://test.mogl.com:444/..." you can rewrite this to be "https://d10ukqbetc20km.cloudfront.net/...".
- **Any user registrations in the test environment MUST contain the word "mogl" somewhere in the email.**
- The txAuth utility will only simulate properly for cards that are "VISA" bins. This card needs to be registered by your application so our system knows that it is one of your cards and to send to your application hook. Any active business that has VISA as a payment type in the test environment will work but we suggest sticking with a known business such as business id 1014.
- The test environment is by definition a "test" environment. We routinely update this environment with new features and functionalities and although we do our best to ensure that there is no downtime in this environment the nature of our pushes won't alway guarantee it is up and functioning without bugs.
- Any data that you register in this environment may be removed without notice. We routinely sanitize production data and place it in this environment for testing purposes.
- Not all methods are directly linked from the API documents main page. For example, the documentation for the txAuth endpoint is not exposed under utilities on the main page but can be accessed fairly simply by knowing the name and group of the method (/api/docs/v2/Utilities/txAuth).
- If you attempt to use the "Try It Now" forms within the documentation pages you should note that the "Endpoint" in the form may need to be changed (e.g. from https://www.mogl.com to https://test.mogl.com).

Authentication

One additional area that is very important is the subject of authenticating with the API. Most of the details about API authentication are covered in the API documentation linked here: https://www.mogl.com/api/docs/v2/Authentication.

All applications are issued an API client id and secret which can be used to interact with the client backend. Client ids are considered non-private tokens and it is not necessary to keep them confidential. Client secrets are obviously privileged and care should be taken to ensure that they are not compromised. In the event of compromise you should notify Empyr immediately so that new credentials may be issued.

Overall, the API makes use of OAuth 2.0 authentication flows. Specifically, the following flows are supported:

- **password -- This flow is used when the user password is known. It is only used by applications internal to Empyr.**
- **client_credentials -- The flow that will most commonly be used by publishers. The client key and secret are exchanged for an access token. The access token is passed to the server and it authenticates the application with privileges on all objects that the application has ever created (e.g. users, businesses etc.). At no time should a client_credentials access token be distributed to end user devices.**
- **client_usertoken -- This is a custom flow that is implemented by the Empyr backend which allows the publisher to request an access token on behalf of a single user account. Essentially, the application is granting an individual user the ability to interact with the API but the user is restricted to only that specific user resource. This is commonly used in scenarios where the publisher wants to issue an access token to give to a mobile app which would then interact with the Empyr API directly (for example registering credit cards).**

Here's an example of one way (of many) to get an Access Token:

curl --insecure "https://test.mogl.com/oauth/token?grant_type=client_credentials&client_id=$1&client_secret=$2"

where you'll substitute your client_id for $1 and your client_secret for $2.

Obviously you don't have to use curl, you can use anything that can send a GET request and enable you to use the response. The response should look similar to this:

```
<oauth>
   <access_token>37392fc6-ae8g-4d66-93e1-347a0cfe9c93</access_token>
   <expires_in>3599</expires_in>
</oauth>
```

This access token expires in just under an hour, as you can see in the expires_in element, and of course the access_token will likely be different.

User Enrollment and Identification

The first task when interacting with the API is the step of enrolling users and their cards into the system. To this end, there are several different ways to accomplish this task and the approach taken is typically guided by the publishers desire to limit scope for PCI compliance.

Approach	Description
Direct	Direct integration with the API creates the highest level of compliance scope. The publisher is responsible for ensuring that credit card data is collected and transmitted securely to the Empyr API.
Hosted Fields	This approach limits the compliance scope by allowing consumers to enter card information into their browser and have it transmitted directly to the Empyr backend. To use this approach a small Empyr javascript is included and initialized by the publisher on the page that will collect details from a dynamically created form element. Once the consumer enters their card information the data is transmitted to Empyr and a card token is transmitted back which may be stored by the publisher.
client_usertoken grant	Similar to hosted fields, the client_usertoken grant is used by the partner to transmit card data directly to the Empyr API. The distinction here is that this approach is more commonly used when on a mobile device.

User Enrollment and Identification - Direct Approach

Direct

This guide is intended to educate developers on the basics of integration with the Empyr API. This guide is meant as a supplement to the existing API documentation and provides some background and explanation specifically framed around integrating as a Publisher.

- **/users/signupWithCard --** https://www.mogl.com/api/docs/v2/Users/signupWithCard

As its name implies the signupWithCard API will both signup a user and add a card to that user's account. Additionally, if the user already exists in the system (determined by the provided user.email field being identical) then calls would append the card to the existing account.

The benefit of appending the card to the existing account is that the publisher application has one endpoint that it will use for creating users and adding the first card as well as adding any future cards so this removes the need to determine if the user already exists and call the /card/add endpoint.

The fields on the following page are the inputs required for the signupWithCard endpoint.

User Enrollment and Identification - Direct Approach

Approach	Description	Recommended Value
user.firstName	First name of the user. This is primarily used by Empyr's own applications and partners where Empyr communicates directly to the consumer.	Anonymous
user.lastName	See above.	User
user.address.postalCode	The postal code associated with the user. This is not used for card verification purposes but we request this be as accurate as possible to help inform sales efforts and merchant acquisition.	
user.email	This will become the "user_token" for the user used in many other API requests. The Empyr platform uses it for its own properties and user communication but is not commonly reflecting a real email address for partners.	partnerUserId@partnername-mogl.com
user.password	A password for the user. Used by Empyr's internal application/properties.	Not set. Use "generatePassword=true" to generate a default.
user.sendNotifications	Used to indicate whether this user should be excluded from notifications.	false
billingdetail.cardNumber	The PAN for tracking and crediting purposes.	
billingDetail.expirationMonth	Not required but validated if provided.	
billingDetail.expirationYear	Not required but validated if provided.	
user. userWhoInvited	Used by Empyr applications.	
user.donatePercent	Used by Empyr applications.	

The response from the signupWithCard API will be a "user" and "card" object as defined in the API documentation. **Each of these will carry an identifier that should be stored by the partner for the purposes of reconciling transaction events back to the user account.**

Empyr

User Enrollment and Identification - Hosted Fields

Hosted Fields was created as a mechanism to reduce the scope of PCI requirements that the partner is exposed to. This approach is commonly used by applications that have a web presence and allows them to create a form within their application and control a great deal of the user interface design while still subsequently passing the credit card data directly to the Empyr platform.

For a detailed description of Hosted Fields please contact your account manager who may provide the documentation.

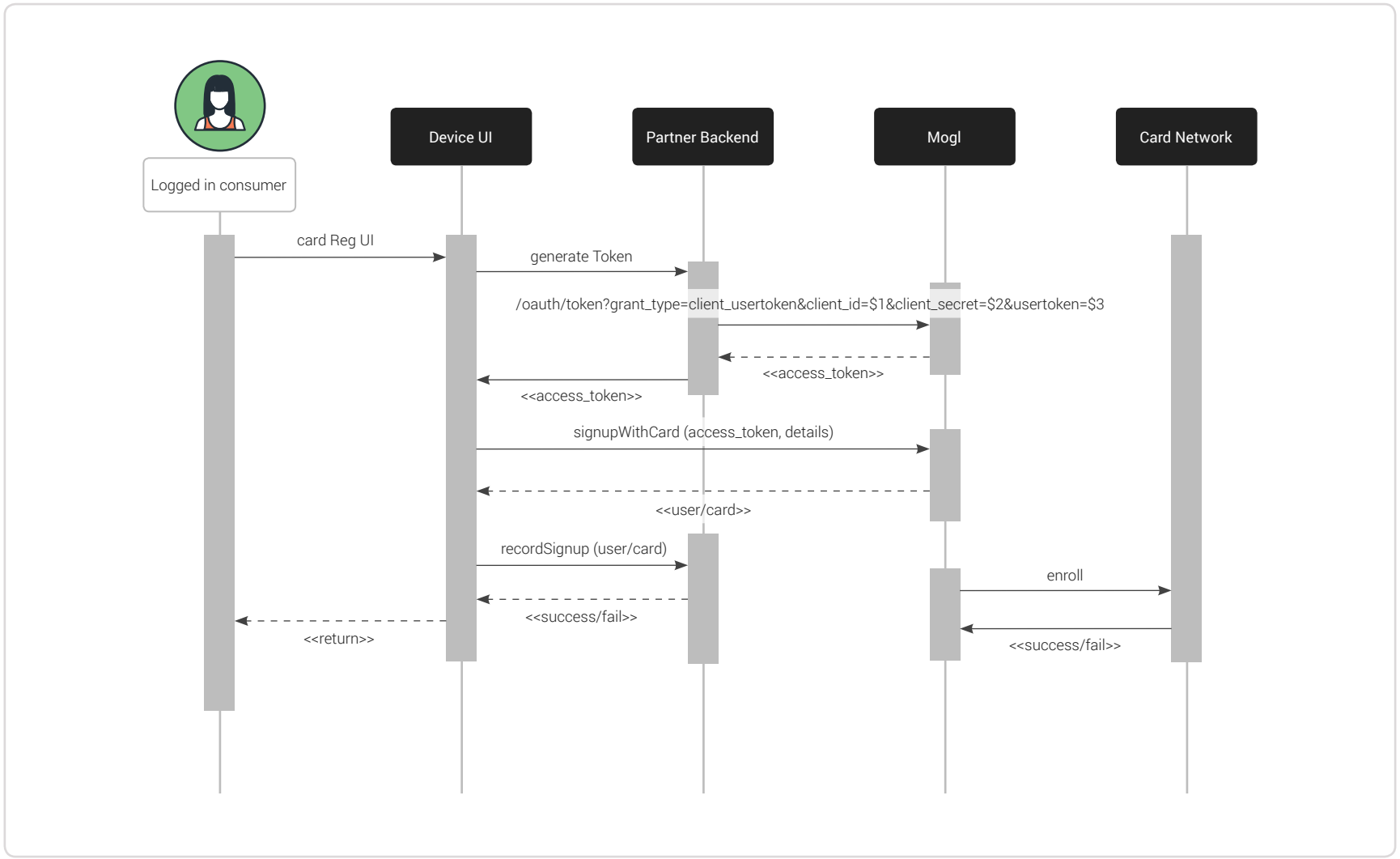
User Enrollment and Identification - client_usertoken Grant

While Hosted Fields allows web applications (and mobile web) a convenient way of providing card data directly to the Empyr platform it does not easily allow mobile applications to easily benefit from it. The client_usertoken grant was created to solve the same problem that web applications face for mobile devices.

At a very high level the basic steps involved in leveraging the client_usertoken grant are as follows:

1. Mobile application shows view where card data should be entered.
2. Application requests a "token" from the partner server to interact with the Empyr API on behalf of the logged in user.
3. The partner server validates its own authentication of the user that is interacting with its API. The partner server then requests from the Empyr platform a "client_usertoken" access token by providing the client key, client secret and the user_token for this user. The "user_token" in this case would be the identifier that the partner would like to use to identify this user in future API requests and is commonly deterministically generated based on the internal id of that user by the partner. It must look like an email. For example "partnerUserId@partnername-mogl.com".
4. The partner server will now give this access token to the mobile device which may use it to interact directly with the Empyr API. This would include making calls to the /users/signupWithCard endpoint explained in the "Direct" integration earlier. Please note that the call to the /users/signupWithCard endpoint requires a user.email which MUST MATCH the user_token provided in step 3.

User Enrollment and Identification - client_usertoken Grant



User Enrollment and Identification - client_usertoken Grant

By using the client_usertoken grant the partner's mobile application will provide card data DIRECTLY to the Empyr platform and removes the partners server from scope of PCI compliance because it is never storing, processing, or transmitting the card data.

Note that it is most common to use the client_usertoken grant for the purposes of handling sensitive calls to the Empyr API (calls which would involve credit card data). Other calls that the partner's application might want to make would typically be handled by the partners mobile application interacting with the partner server and then, if necessary, the partner server would use the client_credentials grant to interact with the Empyr API. For example, if the mobile application wanted to get a list of registered cards the flow would be:

1. Mobile application shows view where card list should be.
2. Application requests list of enrolled cards from partner server API.
3. Partner server API authenticates its own user and then makes a request to the Empyr API for /cards/list using the client_credentials access token and providing the user_token for the given user.
4. Partner server returns the card list to the device.
5. The device shows the registered cards.

There are multiple advantages to this approach including:

1. Limiting the need to frequently get new client_usertoken grants. These grants are for individual account access and expire in one hour. The partner will presumably have longer authentication intervals and can use the client_credentials grant over a number of users on the server side.
2. Dependencies on the Empyr API are more encapsulated to the partner's server so any changes made to interact with the Empyr API are largely localized instead of requiring changes which might be necessary on Android/iOS etc.
3. The ability to augment the data with the partners own internal data.

Empyr

Offer Display

In order for users to benefit from the program they need to be informed about the available offers. The way that offer data retrieved is through:

- /venues/search -- https://www.mogl.com/api/docs/v2/Venues/search

When called with no parameter information the API will simply return a list of all the available offers in the system. Please note that the results are paged so you will only be able to retrieve 500 results at a time.

Partners may choose to either:

- Retrieve the offer data and cache it locally in their own database.
- Retrieve the offer data directly from the API with provided category/query/location criteria for display to the user.

Note that if the partner elects to cache the offer data locally they will be responsible for “removing” any venues that may no longer appear in the feed from display.

Capturing Events

Once users have been enrolled and have been able to see the available offers the next logical activity is that they will go to a participating location. To this end, once the user has transacted at the participating location the Empyr platform will notify the partner of the transaction as well as the associated reward at that location. The partner may then communicate with the user through any means available (e.g. email, sms, push notification etc.).

The events that the Empyr API will send to a partner are documented here: https://www.mogl.com/api/docs/v2/events

In order to receive events partners will provide Empyr with a "webhook" url that Empyr will POST event data to. The webhook must be protected by SSL and in the production environment is required to be on one or more static IP addresses.

Several notes about the events are important for a partner to understand:

1. **AUTHORIZED** events are a notification that we received and processed an authorization of a transaction. It is not uncommon for some authorizations to not clear or settle. For example, if a waiter mistakenly authorizes a user's credit card for the wrong amount then they will void this transaction and the transaction will not clear. What happens in this case is that after 7 days of not receiving the settlement the Empyr platform will send a REMOVED event informing the partner that we have removed the transaction.
2. **CLEARED** events are notifications when transactions have actually settled. Usually, although not always, this would be preceded by an AUTHORIZED event. A CLEARED event will nearly never be subsequently REMOVED except in rare circumstances where there was a processing error.
3. **REMOVED** events are notifications to roll a transaction back and repeal any credits/payment balance accrued by the user.
4. **PAYMENT** events are sent when a user has actually been paid through a credit to their card.

Empyr

Simulating an AUTHORIZED Event

In order to test out your webhook it can be useful to simulate an event. To this end there is a utility in the API that can be used to simulate an authorization event. This utility can be found at:

https://www.mogl.com/api/docs/v2/Utilities/txAuth

Please note the following:

- **Card -- The card MUST have been registered by your application through our API (e.g. /users/signupWithCard). It also MUST be a VISA.**
- **Business -- This needs to be an active business returned by /venues/search. Note that 1014 is known to work.**

Payout Users

Once your users are transacting there will come a time when those users need to be paid. The payout of users will depend upon the partner. At a high level there are three ways to pay out users:

1. **Partner Points --** Some partners will choose to convert reward values into "points" that their consumers earn instead of statement credits.
2. **/payments/add API --** The payments add API enables partners to issue statement credits directly to a consumer's card. The ability to perform this function is not enabled by default and is closely regulated/throttled. Please contact your account manager if you are interested in this approach.
3. **Empyr Funding --** This approach allows Empyr to fund a consumer's card using a predetermined schedule. The basic schedule is as follows:
   A.) Whenever a user's account balance for a prior month exceeds $10 we will pay out that consumer on the 9th.
   B.) If the user went active for the first time in the prior month then the balance will be paid out as long as there has been a tracked transaction and the balance is greater than 0. Again, this payment will be on the 9th.

Empyr recommends the following in regards to payments:

1. Pay only settled transactions -- Authorizations may not settle and these are not able to be billed to merchants.
2. Allow at least 9 days after a settled transaction to pay the consumer. There are rare circumstances that would require a settlement to be rolled back but it is possible.
3. Be **VERY CAREFUL** if you choose to use the /payments/add API. Once a statement credit has been issued it will not be possible to roll it back (unlike CAPTURE transactions where users are charged there is no facility to roll back a refund.).

Miscellaneous

Below are a few miscellaneous notes which may be valuable in your buildout of the application.

- Most images for offers are served through our CDN which then passes to our origin servers. You will notice in the url that there are dimensions provided for many images (e.g. /imagesr/w-325_h-192). It is possible to provide your own width and height parameters if you desire differently sized images.

Peter Vogel - Vice President, Business Development
Peter@Empyr.com - 619.459.7254

Monica Ross - Project Manager, Business Development
MonicaRoss@Empyr.com - 561.405.1090

Support
http://help.empyr.com

9645 Scranton Rd #110
San Diego, CA 92121

888.664.5669 - www.Empyr.com