Bitcoin Payment Gateway API

v0.3

BitPay, Inc.

https://bitpay.com

Table of Contents

Introduction

The BitPay.com Bitcoin Payment Gateway API is designed for merchants that need full control over their customers' shopping and checkout experience. An eCommerce site can make use of this API to transmit invoice information to BitPay.com from their back-end server, and receive server notifications when the customer has completed payment and the invoice total has been credited to the merchant account.

A merchant can elect to receive notifications immediately upon receipt of a payment, or when the payment has been completed and credited to the merchant account.

There are three interactions with the BitPay.com service that this API enables:

- create an invoice
- fetch an invoice status
- receive invoice status updates

Note: For the documentation on the older version of the API that uses SSL client certificates for authentication, see https://bitpay.com/downloads/bitpayApi-0.2.pdf (we have not yet updated all shopping cart plugins to use the new authentication method). While the old authentication method is now deprecated, it is still supported (we don't have any immediate plans to disable it if you're already setup to use it).

Activating API Access

The BitPay.com JSON API is accessible at https://bitpay.com/api/.

The merchant must obtain an API key from the bitpay website by logging into their merchant account and clicking on My Account, API Access keys. A merchant can create multiple keys for use with different e-commerce stores or API functions. Once an API key has been created, BitPay will use this API key to authenticate your API connections.

The merchant's API key must remain private and should never be visible on any client-facing code. Should it ever be compromised, the merchant can generate a new key in their BitPay account.

When connecting to BitPay, use HTTP Basic Authentication with the username as your API key and leave the password blank (the following page describes the HTTP Basic authentication protocol in detail: http://www.ietf.org/rfc/rfc2617.txt). You should also only communicate with the server if you can validate the bitpay.com SSL certificate with a certificate authority. Most HTTPS client libraries make this as simple as setting a switch. Similarly, inbound notification connections should only be recognized when the SSL certificate is validated. Taking both of these steps will ensure that you are always communicating with the Bitpay server and that your API key will never be exposed.

Invoice States

A BitPay.com invoice can be in one of the following states: “new”, “paid”, “confirmed”, “complete”, “expired” or “invalid”. Payments sent to the bitcoin address associated with an invoice will only be credited to the invoice when it is in the “new” state.

“status”	**Description**
“new”	An invoice starts in this state. When in this state and only in this state, payments to the associated bitcoin address are credited to the invoice. If an invoice has received a partial payment, it will still reflect a status of new to the merchant (from a merchant system perspective, an invoice is either paid or not paid, partial payments and over payments are handled by bitpay.com by either refunding the customer or applying the funds to a new invoice.
“paid”	As soon as full payment (or over payment) is received, an invoice goes into the paid status.
“confirmed”	The transaction speed preference of an invoice determines when an invoice is confirmed. For the high speed setting, it will confirmed as soon as full payment is received on the bitcoin network (note, the invoice will go from a status of new to confirmed, bypassing the paid status). For the medium speed setting, the invoice is confirmed after the payment transaction(s) have been confrimed by 1 block on the bitcoin network. For the low speed setting, 6 blocks on the bitcoin network are required. Invoices are considered complete after 6 blocks on the bitcoin network, therefore an invoice will go from a paid status directly to a complete status if the transaction speed is set to low.
“complete”	When an invoice is complete, it means that BitPay.com has credited the merchant’s account for the invoice. Currently, 6 confirmation blocks on the bitcoin network are required for an invoice to be complete. Note, in the future (for qualified payers), invoices may move to a complete status immediately upon payment, in which case the invoice will move directly from a new status to a complete status.
“expired”	An expired invoice is one where payment was not received and the 15 minute payment window has elapsed.
“invalid”	An invoice is considered invalid when it was paid, but payment was not confirmed within 1 hour after receipt. It is possible that some transactions on the bitcoin network can take longer than 1 hour to be included in a block. In such circumstances, once payment is confirmed, BitPay.com will make arrangements with the merchant regarding the funds (which can either be credited to the merchant account on another invoice, or returned to the buyer).

Creating an Invoice

An invoice is created by sending an http POST message to https://bitpay.com/api/invoice with the details of the invoice passed in the body of the request. The body of the message must be JSON encoded and the content-type should be set to “application/json”.

On successful creation, the invoice details will be provided in a JSON encoded response. If there is an error, you will receive a JSON encoded error response. All error responses will have an “error” field that is an object with two fields called “type” and “message”. A merchant is restricted to creating no more than 100 invoices per hour (there are also per second and per minute limits). The fields in the request are described below:

Required POST fields

“price”	This is the amount that is required to be collected from the buyer. Note, if this is specified in a currency other than BTC, the price will be converted into BTC at market exchange rates to determine the amount collected from the buyer.
"currency”	This is the currency code set for the price setting. The pricing currencies currently supported are USD, EUR, BTC, and all of the codes listed on this page: https://bitpay.com/bitcoin-exchange-rates

Optional Payment Notification (IPN) fields

“posData”	A passthru variable provided by the merchant and designed to be used by the merchant to correlate the invoice with an order or other object in their system. This passthru variable can be a JSON-encoded string, for example posData: ‘ { “ref” : 711454, “affiliate” : “spring112” } ‘
“notificationURL”	A URL to send status update messages to your server (this must be an https URL, unencrypted http URLs or any other type of URL is not supported). Bitpay.com will send a POST request with a JSON encoding of the invoice to this URL when the invoice status changes.
“transactionSpeed”	default value: set in your https://bitpay.com/order-settings “high” : An invoice is considered to be "confirmed" immediately upon receipt of payment. “medium” : An invoice is considered to be "confirmed" after 1 block confirmation (~10 minutes). “low” : An invoice is considered to be "confirmed" after 6 block confirmations (~1 hour). **NOTE:** Orders are posted to your Account Summary after 6 block confirmations regardless of this setting.
“fullNotifications”	default value: false

	true: Notifications will be sent on every status change. false: Notifications are only sent when an invoice is confirmed (according the the transactionSpeed setting).
“notificationEmail”	Bitpay.com will send an email to this email address when the invoice status changes.

Optional Order Handling fields

“redirectURL”	This is the URL for a return link that is displayed on the receipt, to return the shopper back to your website after a successful purchase. This could be a page specific to the order, or to their account.

Optional Buyer Information to display

“orderID”	Used to display your public order number to the buyer on the BitPay invoice. In the merchant Account Summary page, this value is used to identify the ledger entry.
“itemDesc”	Used to display an item description to the buyer.
“itemCode”	Used to display an item SKU code or part number to the buyer.
“physical”	default value: false true : Indicates a physical item will be shipped (or picked up) false : Indicates that nothing is to be shipped for this order
“buyerName” “buyerAddress1” “buyerAddress2” “buyerCity” “buyerState” “buyerZip” “buyerCountry” “buyerEmail” “buyerPhone”	These fields are used for display purposes only and will be shown on the invoice if provided.

BitPay Server Response

The response to a create invoice request, the response to a get invoice request, and the content of a status update notification are all identical JSON representations of the invoice object. The fields are described below:

Name	Description
“id”	The unique id of the invoice assigned by bitpay.com
“url”	An https URL where the invoice can be viewed.
“posData”	The passthru variable provided by the merchant on the original invoice creation.
“status”	The current invoice status. The possible states are described earlier in this document. “new” “paid” “confirmed” “complete” “expired” “invalid”
“price”	The price set by the merchant (in terms of the provided currency).
“currency”	The 3 letter currency code in which the invoice was priced.
“btcPrice”	The amount of bitcoins being requested for payment of this invoice (same as the price if the merchant set the price in BTC).
“invoiceTime”	The time the invoice was created in milliseconds since midnight January 1, 1970.
“expirationTime”	The time at which the invoice expires and no further payment will be accepted (in milliseconds since midnight January 1, 1970). Currently, all invoices are valid for 15 minutes.
“currentTime”	The current time on the BitPay.com system (by subtracting the current time from the expiration time, the amount of time remaining for payment can be determined).

Getting an Invoice Status

To get the current state of an invoice, an http GET request can be sent to https://bitpay.com/api/invoice/<id> where the id is the invoice id provided when the invoice was created. The format of the response is exactly the same as that which is returned when creating an invoice.

Receiving Invoice Status Updates

Invoice status updates can be sent either via email, https or both. The “notificationEmail” and “notificationURL” settings control the destination for the notification. Note, email notification is a human readable format and not intended for use as a system interface. For https notification, BitPay.com sends a POST request to the given URL with a JSON encoding of the invoice that is identical to the format returned from a create invoice or get invoice request. If “fullNotifications” are set to true, then an update will be sent for every change in status. If “fullNotifications” are false, then an update is only sent when an invoice is confirmed (according to the “transactionSpeed” setting).

Sample Client Library

For convenience, a sample client library is provided to demonstrate how to interact with the BitPay.com JSON API. You can use this client library as is on your server, you can customize it, or you can use it as a guide for developing a client library in another language. The sample client library is written in JavaScript and is designed to run using nodejs. Nodejs can be downloaded form http://nodejs.org. The examples have been tested on version 0.8.9, but should work on later versions as well. The sample client library can be obtained from https://github.com/gasteve/bitpayNodejs/zipball/master. The zip file contains 3 utilities: createInvoice, getInvoice, invoiceListener. These files are executable and invoke the node runtime using typical Unix shebang notation. They can also be started by passing them as the first argument to the "node" runtime. The files themselves are JavaScript source code.

To use the utilities, modify the config.js file and copy and paste an API key from your merchant account into the apiKey setting. This will associate your API calls with your merchant account. Also, there is a sample SSL key and certificate file that is used by the invoiceListener to setup the HTTPS server that listens for incoming invoice notifications. While these example credentials will work fine, you may want to create your own unique SSL key and certificate.

To create an invoice, run the createInvoice utility and pass in an invoice description on stdin. A sample invoice description is provided in the file sampleInvoice.json. To create an invoice using this sample, run the following command:

```
$ ./createInvoice <sampleInvoice.json
```

The newly created invoice will be output on a single line in JSON format.

To get an invoice, run the getInvoice utility and pass the invoice id as the sole argument as follows:

```
$ ./getInvoice 5_TU2V-M0glicVcZuQkkkq9aiA7qP0MjxRkhdc1MRSY=
```

Just as before, the invoice will be output on a single line in JSON format.

To receive notifications of invoice status updates, use the invoiceListener utility. It takes a single parameter on the command line to specify the port number (or it can be specified in config.js) and listens for incoming notifications from BitPay.com. If you create an invoice with a notificationURL to your server and port, notifications of status changes on that invoice will be delivered to this utility. When a notification is received, the utility will print the JSON encoded invoice on stdout (one line per notification).

With these utilities, it is easy to craft a solution that can create a BitPay.com invoice and receive payment notifications.

Revision History

0.1	September 2011	Original Release
0.2	December 2011	Updated SSL info
0.3	September 2012	Changed Authentication from SSL fingerprint to API token method