# Payment Gateway API

**v0.3.2**

**BitPay, Inc.**

**https://bitpay.com**

©2011-2014 BITPAY, Inc. All Rights Reserved.

## 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.

| **State** | **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, partia 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 confirmed 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 (fo 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 confirme 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

| **Name** | **Description** |
|---|---|
| “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

| **Name** | **Description** |
|---|---|
| “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. Maximum string length is 100 characters.<br><br>This passthru variable can be a JSON-encoded string, for example<br><br>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).<br><br>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, the default value set i your merchant dashboard is “medium”.<br><br>● “high”: An invoice is considered to be "confirmed" immediately upon receipt of payment.<br><br>● “medium”: An invoice is considered to be "confirmed" after 1 block |

| | |
|---|---|
| | confirmation (~10 minutes).<br><br>● "low": An invoice is considered to be "confirmed" after 6 block confirmations (~1 hour).<br><br>**NOTE:** Orders are posted to your Account Summary after 6 block confirmations regardless of this setting. |
| "fullNotifications" | default value: false<br><br>● true: Notifications will be sent on every status change.<br><br>● false: Notifications are only sent when an invoice is confirmed (accordin the the transactionSpeed setting). |
| "notificationEmail" | Bitpay.com will send an email to this email address when the invoice status changes. |

## Optional Order Handling fields

| **Name** | **Description** |
|---|---|
| "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

| **Name** | **Description** |
|---|---|
| "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. Maximum string length is 100 characters. |
| "itemDesc" | Used to display an item description to the buyer. Maximum string length is 100 characters. |
| "itemCode" | Used to display an item SKU code or part number to the buyer. Maximum string length is 100 characters. |
| "physical" | default value: false<br><br>● true: Indicates a physical item will be shipped (or picked up)<br><br>● false: Indicates that nothing is to be shipped for this order |
| "buyerName"<br>"buyerAddress1"<br>"buyerAddress2" | These fields are used for display purposes only and will be shown on the invoice if provided. Maximum string length of each field is 100 characters. |

|  |  |
|---|---|
| “buyerCity”<br>“buyerState”<br>“buyerZip”<br>“buyerCountry”<br>“buyerEmail”<br>“buyerPhone” |  |

## 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.<br><br>● “new”<br>● “paid”<br>● “confirmed”<br>● “complete”<br>● “expired”<br>● “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 th 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 (i milliseconds since midnight January 1, 1970). Currently, all invoices are valid fo 15 minutes. |
| “currentTime” | The current time on the BitPay.com system (by subtracting the current time fro 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).

## Bitcoin Best Bid (BBB) Rates

BitPay consolidates market depth from multiple exchanges to provide buyers with a Bitcoin Best Bid (BBB) exchange rate. BitPay currently calculates BBB based on Bitcoin/US Dollar rates because of the maximum liquidity.  For a complete list of available currencies see https://bitpay.com/bitcoin-exchange-rates.

To calculate the exchange rate for US Dollars, we pull the market depth from exchanges with adequate liquidity and withdrawal capability in USA and the Eurozone. The exchange order books are merged into a Consolidated Level II table.

The BBB is calculated by simulating an auto-routing market sell order, across all exchanges, with zero commission fees. Buyers will always get a better value by spending their bitcoins at a BitPay merchant than by selling them on an exchange.

The BBB is available via JSON API at https://bitpay.com/api/rates. Rates are updated every 1 minute.

### BitPay Server Response

The response to a request for BBB is a list of identical JSON representations of the rate object.  The fields are described below:

| **Name** | **Description** |
|---|---|
| “name” | The full display name of the currency. |
| “code” | The three letter code for the currency, in all caps. |
| “rate” | The numeric exchange rate of this currency provided by the BitPay server. |

# HTTP Integration Concerns

### Embedded Invoice (iframe) Post to Parent Window

When an invoice is presented in an iframe you have an option to receive invoice status updates in the parent window. This option is useful for updating the parent window presentation or redirecting the parent window to another URL after the invoice has been paid.

When the invoice iframe receives a status update from the BitPay server the new status is posted from the invoice iframe to the parent window via the Window.postMessage method and passing the following event data:

> {status: string}
> where 'status' can be any of the Invoice States according to the descriptions provided.

Your implementation should take into consideration the browser support for this method. See http://caniuse.com/#feat=x-doc-messaging for a list of browsers supporting Window.postMessage.

Following is an HTML code example illustrating a simple interaction between an invoice iframe and its parent document.

```
<html>
<head>
<title>Parent</title>
</head>
<body>
<p>Invoice status: <span id="s1"></span>
<iframe src="https://bitpay.com/invoice?id=UbAd3j1E7ivCe5i9t88obd"
style="width: 800px; height: 800px;"></iframe>
<script language="javascript">
window.addEventListener("message", function(event) {
    document.getElementById("s1").innerHTML=event.data.status;
}, false);
</script>
</body>
</html>
```

## 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/bitpay/nodejs-client/archive/master.zip. 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.

## Testing

Testing transaction processing is easily accomplished using the BitPay production server.

### Merchant Account Setup

In order to test you must have an established BitPay merchant account. It is perfectly safe to test using a single production merchant account, however, this may not be suitable for some organizations due to security and ledger accounting visibility concerns. If you require a test environment to be completely isolated from your production environment then you should request a second BitPay merchant account.

Complete the following steps to setup your merchant account for testing:

1. Set your payout to 100% BTC - see https://bitpay.com/accounting
2. Set your bitcoin address to be the same address from which you send bitcoin while paying invoices
3. Generate an API key for your account - see https://bitpay.com/api-keys

Setting up your merchant account this way allows the bitcoins you send for payment to be circulated back to you the following day when merchant payouts are run.

### Testing Considerations

Testing may involve not only the integration of the BitPay service and the payment of invoices but also how you handle common payment exceptions. See the BitPay Customer Support Guide for more information about payment exceptions.

You should consider testing the following kinds of transactions:

- A fully paid invoice, paid on time (within 15mins)
- A fully paid invoice, paid late (after 1 hour, after 24 hours)
- An under paid invoice (pay an amount less than the invoice requires)
- An overpaid invoice (pay an amount more than the invoice requires)
- An invoice with and without wallet fees
    - Try to force an invalid transaction using a low value invoice with no wallet fees
    - Invalid transactions are invoices that have been paid but which have zero confirmations within one hour of being received by the BitPay server

## Business Integration Scenarios

The BitPay Payment Gateway API may be integrated into your business using any available technology or language that can interact with HTTP servers by performing HTTP GET and PUT operations. This section presents some additional key concepts and usage patterns that you may encounter while deploying this API. For additional information about available integration options see https://bitpay.com/bitcoin-payment-gateway-api. For an overview (including video examples) of the business use cases supported by this API see https://bitpay.com/faq.

Service integration with BitPay may or may not require your business to implement this API directly. We recommend that you evaluate and select the best alternatives for your business. Generally, you will be making selections from one or more of the following methods:

- Shopping cart plugins (with available Fulfillment by Amazon capability)
    - https://bitpay.com/bitcoin-for-ecommerce#plugins
    - https://bitpay.com/bitcoin-for-ecommerce#amazon (for Fulfillment by Amazon)

- BitPay.com hosted shopping cart with “buy now” or ”add to cart” buttons/links
    - https://bitpay.com/catalog-item-list

- BitPay.com hosted checkout without a shopping cart (with available donation capability)
    - https://bitpay.com/create-checkout
    - https://bitpay.com/create-donate (for accepting donations)

- Source code library (e.g., PHP, Node.js, Ruby, Java, C#)
    - https://bitpay.com/bitcoin-for-ecommerce#code-libraries

In most cases you’ll be able to begin accepting bitcoin payments on the very same day that your merchant account is approved. We have developed tools and information for you to understand and easily integrate BitPay services into your business process. Relative to each other, some of the available integration scenarios require more work than others to implement. The following table describes how we have scored each integration scenario relative to the others to give you an idea of what you can expect.

| | |
|---|---|
| Selection Criteria | When this scenario / option might be a good choice for your business. |
| Time to Availability | Estimated amount of time required to enable payment processing for this scenario. |
| Required Effort | ● **None** - nothing for you to do<br>● **Low** - execute documented installation instructions, GUI driven configuration<br>● **High** - Potentially much work needs to be done, perhaps requiring multiple persons |
| Required Skill | ● **Low** - Able to navigate websites and follow simple point & click instructions<br>● **Medium** - Administrative understanding of server-side tools and navigation of files and file systems for installation and configuration<br>● **High** - Creative thinking and problem solving; developer level understanding of server-side implementation, able to construct, debug, and test software systems |

## Transaction Speed and Order Fulfillment

The concept of transaction speed is tightly coupled to your order fulfillment business process and should not be overlooked. However unlikely, it is possible for the bitcoin network to experience a double-spend attempt. Transaction speed is designed to protect your business from product loss during such an attempt. The transaction speed you choose reflects the level of risk associated with any potential loss of product between the time a BitPay invoice is "paid" and the time the invoice is "confirmed".

There are two places that you can set the transaction speed for invoice processing; (1.) on your BitPay merchant account dashboard at https://bitpay.com/order-settings, and (2.) on a per invoice basis as each invoice is created using this API. The transaction speed setting on your merchant dashboard serves as the default transaction speed for all invoices that do not explicitly set (or override) the transaction speed. The default value is initialized to "medium" when your merchant account is created. The ability to set the transaction speed on a per invoice basis provides you with flexibility and control over when invoices may be fulfilled.

The choice of transaction speed is really dependent on your tolerance for fulfillment risk during the period of time between when the payment is received by BitPay and when you fulfill the order.

**Note:** If you choose to wait for BitPay invoices to become "complete" prior to beginning your order fulfillmen process then you are guaranteed to collect the full amount of the invoice and any further consideration of fulfillment logistics is unimportant relative to BitPay payment processing.

### Fulfillment of Hard Goods

For your business the fulfillment logistics for hard goods (any shippable product) may typically exceed the 6-block confirmation time (approximately 1 hour). In this case it's safe to set your transaction speed to "low"; i.e., orders will "complete" prior to fulfillment. Note that setting transaction speed to "low" means that you are (depending on the details of your business process) delaying order confirmation and subsequent fulfillment processing by approximately 1 hour (the 6-block confirmation time).

If your fulfillment process is highly automated and can accept order fulfillment processing nearly immediately after receiving a "paid" invoice then there is a decision to be made regarding how much business risk you are willing to accept. If your downstream order fulfillment process is not easily interrupted (e.g., to cancel a pending order) or the value of the product to ship is very high then it may be best to purposefully delay order confirmation by using either "low" or "medium" transaction speed.

To avoid potential customer service problems you may choose to provide an explanation or order progression mechanism about when you expect to be able to confirm and ship the customers order.

### Fulfillment of Soft Goods

For soft goods (digital downloads) you may choose to vary the transaction speed based on the cost of the item being purchased. If a customer is buying an inexpensive item (e.g., a single song) then one alternative may be to set the transaction speed to "high" to allow your fulfillment process to complete immediately allowing for the download to occur without delay.

If the customer is buying an expensive item (e.g., software needing license keys) then you might want the transaction speed set to "medium". This allows you to know that the network has at least one confirmation for the transaction signalling that it's safe to complete fulfillment by emailing product license keys, etc.

## Scenario: Accept eCommerce payment using BitPay Catalog Items and Shopping Cart

| | |
|---|---|
| Selection Criteria | ● Desire a shopping cart experience<br>● Relatively small product catalog<br>● No existing commercial shopping cart integration on your website |
| Time to Availability | 60 to 90 minutes |
| Required Effort | Low |
| Required Skill | Medium |

In this scenario you can begin accepting bitcoin payments after minimal configuration of your website to associate buttons or links with your items.

1. Create buttons or links for each catalog item on your website. See https://bitpay.com/catalog-item-list.
2. For each button or link, modify your website to associate the payment link with the product.
3. Your customers can now accept bitcoin payment by adding products to the BitPay shopping cart and checking out.

## Scenario: Accept eCommerce payment using a Commercial Shopping Cart Plugin

| | |
|---|---|
| Selection Criteria | ● Desire a shopping cart experience<br>● Have an existing commercial shopping cart integration on your website<br>● Large product catalog |
| Time to Availability | 60 to 90 minutes |
| Required Effort | Low |
| Required Skill | Medium |

In this scenario you can begin accepting bitcoin payments after minimal configuration of your existing shopping cart implementation.

1. Review and select the appropriate shopping cart plugin from https://bitpay.com/bitcoin-for-ecommerce#plugins.
2. Install the plugin as prescribed by the plugin installation instructions.
3. Configure the plugin as prescribed by the plugin installation instructions.
4. Your customers can now accept bitcoin payment by choosing the bitcoin option during checkout.

## Scenario: Accept eCommerce payment using proprietary shopping experience

| | |
|---|---|
| Selection Criteria | ● Desire a shopping cart experience<br>● Have an existing proprietary shopping cart integration on your website<br>● Large product catalog |
| Time to Availability | Several days |
| Required Effort | High |
| Required Skill | High |

In this scenario you must integrate the BitPay Payment Gateway using one of the available source code libraries (or write a custom integration using a language not currently offered by BitPay). The following outline of activities should be considered for this scenario.

1. Review and understand this details of this API in the context of your technology and language constraints. Obtain or create the desired API library. You can download supported code libraries from https://bitpay.com/bitcoin-for-ecommerce#code-libraries.
2. Understand the business requirements for your implementation. Generally, your implementation will integrate with the BitPay Payment Gateway API during the customer checkout process at the step when the customer will select from among the available payment options. Your implementation should include allowing the customer to select “Bitcoin” as an option. Consider the following information as you define your implementation.

   a. Payment information - BitPay only requires three pieces of information to process a payment; the bitcoin payment address (a public encryption key), the payment amount, and currency units. Any other information should be managed by your implementation via posData, for example.
   b. IPN (Instant Payment Notification) - This server-to-server communication (see notificationURL) can help you integrate your bitcoin transactions with other types of transactions in your order book. It can also be used to provide feedback to your customer (i.e., presenting the details of bitcoin transaction in the customers order history).
   c. Send minimal POS data - When posting to the BitPay server for creating an invoice you can provide optional posData to read from the IPN later. Adding your internal invoice id to the posData is useful to match your invoice with the BitPay invoice.
   d. Full page or embedded (iframe) invoice presentation - BitPay.com can present a full page invoice to your customer. When the full page invoice is presented your customer is no longer on your website. After the order is paid the customer may optionally click a button that brings them back to your website (see redirectURL). In contrast, the embedded iframe invoice is presented within the context of your website. The customer can send a payment using the embedded invoice (click-to-pay, scan QR, or copy/paste). Once the invoice has been paid it will update within about 1 second (typical) to show that it has been paid. You can detect this invoice state change in your parent web page and perform some action as a result (i.e., display a thank you page).
   e. Customer experience following invoice payment - You can control where the customer lands after an invoice has been paid.
   f. Helping the customer understand their payment action options:

    i. Click-to-pay - If a wallet installed on the customers PC the button click will launch their wallet application and transfer the bitcoin payment address and amount to the “send payment” wallet screen.  If the customer does not have a wallet installed and they click-to-pay the browser will not be able to launch an application and may respond poorly (display a generic page) and confuse the customer.
    ii. Scanning the invoice QR code - If the customer desires to pay with a mobile device they can scan an invoice QR code which transfers the bitcoin payment address and the amount to the mobile device wallet send form.
    iii. Copy/paste your payment address - If the customer uses a wallet accessed by logging into a website, or they simply desire to copy the address to their wallet, they can do so by clicking a link on the BitPay invoice and using the right-click context menu to copy the address.  They must also manually enter the amount into their wallet send form.

3. Design and implement changes to your eCommerce website by integrating appropriate API calls.
4. Your customers can now accept bitcoin payment by choosing the bitcoin option during checkout.

## Scenario: Accept in-person payment without a product catalog

| | |
|---|---|
| Selection Criteria | ● Desire an in-person purchase experience<br>● Do not require a product catalog<br>● Have existing POS used by service professionals |
| Time to Availability | Zero |
| Required Effort | None |
| Required Skill | Low |

In this scenario you can be accepting bitcoin payments immediately. This is a very simple process and requires little to zero setup or configuration. Only a few minutes of merchant service professional training is necessary. This process uses the Checkout Now feature of your BitPay merchant account (see My Payment Tools at https://bitpay.com/home). The following scenario describes the process of accepting payment for food and beverage service (for example).

1. When the customer is ready to pay they say "I'd like to pay with bitcoin!"
2. The service professional replies "That's great!", then obtains a mobile device (smart phone, tablet, or similar), accesses your bookmarked "Checkout Now" webpage at BitPay.com and generates an invoice for the amount of the bill by entering:
   a. An invoice number (from your POS)
   b. The price in USD
3. The service professional clicks a button and BitPay.com responds by displaying a QR code (the invoice) for the customer to scan.
4. The service professional then shows the QR code to the customer who scans the QR code using his or her own mobile device (with a bitcoin wallet installed).
5. The customer clicks the send payment button in their wallet.
6. The service professional waits a few seconds until the invoice on his or her own mobile device updates to show payment received.
7. Done.

The only thing you may need to be able to do is to mark a sale in your POS as paid with bitcoin. This could be done simply by performing a cash transaction with $0 tendered. The POS transaction would be matched up later with the transaction log from BitPay.com using your POS invoice number.

For reconciliation you can access your BitPay.com account to view and download your transactions. The key between your POS and your BitPay transactions is your POS invoice number (which is entered into the BitPay invoice when a customer payment is made).

This whole process may be supplemented by either or both of the following (each of these require a one time configuration/setup effort):

- Optional - an email of your received payment may be sent to any email address you choose (human readable email format)
- Optional - an instant payment notification (IPN) may be posted from BitPay.com to your server (content is the payment transaction information in a machine readable format).

## Scenario: Accept billed invoice payment

| | |
|---|---|
| Selection Criteria | ● Desire to bill client for purchase (pre or post service rendering)<br>● Do not require a product catalog<br>● Service professionals have login access to BitPay.com merchant account |
| Time to Availability | Zero |
| Required Effort | None |
| Required Skill | Low |

In this scenario you can be accepting bitcoin payments immediately. This is a very simple process and requires zero setup or configuration. Only a few minutes of merchant service professional training is necessary. This process uses the My Bills feature of your BitPay merchant account (see My Payment Tools at https://bitpay.com/home). The following scenario describes the process of accepting payment for holding a hotel room reservation (for example).

### Part A - Accept a pending room reservation

1. A customer calls the hotel reservation desk and asks to pay their room deposit using bitcoin.
2. The reservation assistant logs in to their BitPay.com merchant account using the merchant account user id and password (access can be via smart phone, tablet, or PC), clicks on "My Bills" and clicks the "Create New Bill" button.
3. The reservation assistant enters the following information into the bill:
    a. The bill or folio number (e.g., "12345.1")
    b. An item description (e.g., "1st night room and tax for res 12345.1")
    c. The price (e.g., "$125.98")
    d. The quantity (e.g, "1")
    e. The customers information including:
        i. Email address
        ii. Name
        iii. Mailing address
4. The reservation assistant clicks the "Send Bill" button.
5. Optionally, the reservation assistant makes a notation in the customers' folio that payment has been billed.
6. Done.

### Part B - Customer pays for a room reservation

1. The customer receives an email from BitPay.com.
2. The customer clicks the link provided in the email to access the BitPay.com hosted bill form.
3. The customer reviews the bill and clicks the "Checkout Now" button on the bill form.
4. The customer is presented an itemized invoice from BitPay.com and chooses one of the following three options to make payment:
    a. Click-to-pay - If a wallet installed on the customers PC, smartphone, or tablet the button click will launch their wallet application and transfer the bitcoin payment address and amount to the "send

payment" wallet screen.

   b. Scanning the invoice QR code - If the customer desires to pay with a mobile device they can scan an invoice QR code which transfers the bitcoin payment address and the amount to the mobile device wallet send form.
   c. Copy/paste your payment address - If the customer uses a wallet accessed by logging into a website, or they simply desire to copy the address to their wallet, they can do so by clicking a link on the BitPay invoice and using the right-click context menu to copy the address. They must also manually enter the amount into their wallet send form.

5. The customer may re-access the bill (via the emailed link), view the paid status, and print for a receipt.
6. Done.

## Part C - Reconcile a pending room reservation

1. The hotel night audit specialist logs in to their BitPay.com merchant account using the merchant account user id and password (access can be via smart phone, tablet, or PC), clicks on "My Bills" and views bills that have been paid during the prior day.
2. For each bill marked "Complete" the night audit specialist accesses the customers' folio and marks the room reservation as paid.
3. Done.

For reconciliation you can access your BitPay.com account to view and download your transactions. The key between your folio system and your BitPay transactions is your folio number (which is entered into the BitPay bill when the bill is created).

## Troubleshooting

Community help and support can be found on the BitPay support forums at https://support.bitpay.com/forums.

| **Symptom** | **Possible Cause** | **Solution** |
|---|---|---|
| Customer order status is not updated after payment. | The immediate payment notification (IPN) URL is specified incorrectly. The BitPay API supports sending IPN POSTs to only secure socket layer (SSL) URLs. Additionally, the IPN URL must have a valid SSL certificate. | 1. Verify that your "notificationURL" for the invoice is "https://" (not "http://")<br>2. Ensure a valid SSL certificate is installed on your server.<br>3. Verify that your callback handler at the "notificationURL" is properly receiving POSTs. You can verify this by POSTing your own messages to the server from a tool like Chrome Postman.<br>4. Verify that the POST data received is properly parsed and that the logic toward updating order status on the merchants eCommerce server is as expected.<br>5. Verify that the merchants eCommerce server is not blocking POSTs from servers it may not recognize. |

## 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 |
| 0.3.1 | December 2013 | Added Business Integration Scenarios |
| 0.3.2 | January 2014 | Added explanation of transaction speed and fulfillment<br>Added BBB API section<br>Added Troubleshooting section<br>Added Testing section<br>Added new implementation scenario |

©2011-2014 BITPAY, Inc. All Rights Reserved.