# Punchh Integration proxy HTTP API

There is test server created for at 192.198.218.90:8080 (this is public IP!) which is accessible for your testing.

## Requesting barcode based on the currently open check number

POST TO http://server:port/receipt\_details

where

* server address is the address of the integration proxy. Normally it is local host.
* port is specified in the Punchh.cfg file of the integration proxy

ATTENTION: use <http://192.198.218.90:8080/receipt_details> for testing. Allow changing to localhost for deployment.

**Parameters:**

**required:**

- **transaction\_no**: receipt number (should be same for check reprints)

The call returns the 12 digit barcode as a raw string. Example [**12345678910**](tel:12345678910).

If the return value is ‘none’, don’t print the barcode

## Sending check data

Note: that this call can be combined with the previous one as it also returns the barcode (it just takes more parameters)

POST TO http://server:port/receipt\_details

**Parameters:**

**required:**

- **transaction\_no**: receipt number (should be same for check reprints)

- **amount**: subtotal amount (total - tips - taxes - discounts)

- **employee\_id**

- **employee\_name**

- **receipt\_datetime**: example 12/31/2013 20:12:45

- **menu\_items[]**: multiple receipt line items. Each item includes the following pipe separated fields:

**name | quantity | amount | type | id | category | subcategory**

where

   type is M for regular item, D for discount, T for tax

   id is menu item ID

   category is menu item category

subcategory is menu item subcategory (or any other descriptor)

   amount is total for the menu item line (quantity \* rate)

The call returns the 12 digit barcode as a raw string. Example [**12345678910**](tel:12345678910).

If the call fails set the barcode to be **999999999999**. Or better make that number configurable with all nines as default.

## Redemption

POST TO http://server:port/redemption\_codes/<redemption code>

where <redemption code> is the code provided by the customer’s iPhone application

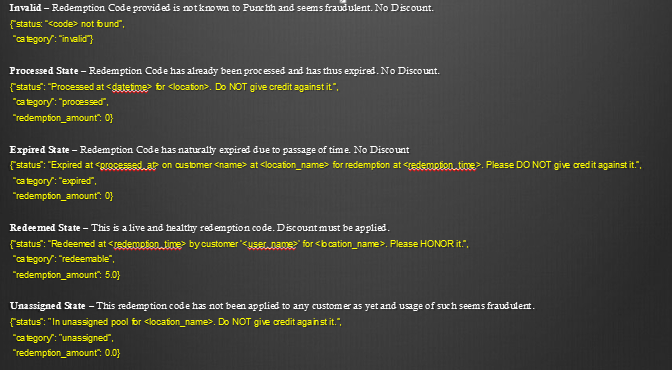
Parameters:

* Same as in [Sending Check Data](#_Sending_check_data), plus:
* either “process=true” or “query=true” (in the 1st case the redemption is commited, in the 2nd case it is just a validation)

JSON response:

* status
* category
* redemption\_amount (optional)

Example:



## Check in

POST TO http://server:port/checkins

Parameters:

* Same as in [Sending Check Data](#_Sending_check_data), plus:
* user\_token=<user id from the barcode presented on mobile app>

JSON Response:

* first\_name
* last\_name
* points
* avatar\_url
* checkins

## Configuration

The following is provided to get some settings from the back end. Make sure to make that call at least each time the system or terminals start and save the settings for future use (until the next time you call the API).

POST TO

<http://server:port/punchhconfig>

We may add more settings to the response so make sure your code doesn’t crash if there are extra JSON properties there.

Parameters:

* none

JSON Response:

* the content of the corresponding config file

Examples:

<http://server:port/punchhconfig>

{"POS":"generic","LogLevel":4,"API\_Key":"5320d2bf22a366b60a893eb952a6897e","Staging":false,"PrintBarcodes":true,"Trailer1":"Download Punchh app on iPhone or","Trailer2":"Android and scan to earn rewards","Trailer3":"for your visits and referrals","Trailer4":" No smartphone? Punchh at","Trailer5":" http://punchh.com"}

Important configuration parameters (these parameters can be updated from the punchh.com dashboard so make sure you re-read them at least when the terminals restart):

* PrintBarcodes:

(true or false)

If false don’t print both the punchh barcode and the trailer message.

* Trailer1 to Trailer5:

These strings should be printed right after the barcode

## Swipe Card Check in

POST TO http://server:port/card\_checkins

Parameters:

* Same as in [Sending Check Data](#_Sending_check_data), plus:
* Card\_number=<swipe card number >

JSON Response:

* first\_name
* last\_name
* registered – flag telling if the card is registered by the user (we support unregistered cards too)
* unredeemed\_checkins – number of check ins available for redemption
* avatar\_url
* redeemable\_cards – number of available redemptions
* value\_per\_redeemable\_card – how much user can redeem at once
* points\_added – number of points added with the check in

Example:

{"first\_name":"Punchh","last\_name":"Farmer Boys Card","registered":false,"avatar\_url":"https://assets.punchh.com/images/default/avatar.png","unredeemed\_checkins":8,"redeemable\_cards":1,"value\_per\_redeemable\_card":5.0,"points\_added":1}

## Swipe Card Redeem

POST TO http://server:port/card\_redeem

Parameters:

* Same as in [Sending Check Data](#_Sending_check_data), plus:
* Card\_number=<swipe card number >

response "{\"status\":\"Not enough Punches to redeem\",\"category\":\"invalid\",\"redemption\_amount\":0.0}" string

## Swipe Card Balance

GET FROM http://server:port/card\_redeem/possible

Parameters:

* Card\_number=<swipe card number >
* Amount

Response:

* status: text string to display or print
* category: if category is “redeemable” then process
* redemption\_amount: the amount to redeem

{"status":"Possible maximum discount amount", "category":"redeemable","redemption\_amount":5.0}

## Additional Requirements

* Minimize setup steps if possible
* See if it is possible to automate software updates