hiku External API Documentation v1.5.1

Nils Gura, Rajan Bala
hiku labs, inc

Change log

Version	Changes
1.5	Introduced event API including PRODUCT_ENTRY, EAN_NOT_FOUND, and DEVICE_BATTERY_LEVEL events
1.5.1	Made fields “ean” and “audioPath” required fields for PRODUCT_ENTRY event to provide backward compatibility. Updated data type for “batteryLevel” in DEVICE_BATTERY_LEVEL to be a string that represents an integer from 0-100.

API Basics

All requests go to baseUrl/api/apiVersion
base url = https://hiku.herokuapp.com/
API Version = 1
>> https://hiku.herokuapp.com/api/v1/

All requests must contain the following parameters
app_id = yourAppId
sig = a sha256 hash of (app_id+secret+time).
time = String version of the current utc time. The time field should follow ISO-8601 standard with the following format: “YYYY-MM-DD hh:mm:ss.SSSSSS”. If your programming language of choice does not support seconds to 6 decimal places, then set any trailing digits to zeros. Of course the time value should be the same for the time parameter and the time used in the signature. Note: while all parameters should be URL encoded in https requests, make sure you do *NOT* use the URL encoded time when generating the signature.

Email devrel@hiku.us to register for an app_id and secret.

All successful responses follow the format of

```
{
"response": {
        "status": "ok",
        "data": {
                "somedata": []
                }
        }
}
```

data is optional

Any HTTP status code may be returned based on service reachability, however all valid hiku service errors return HTTP 200 status with a payload as follows describing the error:

```
{
"response": {
        "status": "error",
        "errMsg": "some reason for the error"
        }
}
```

Event API
Receiving device data (aka beeps) and events

Any time a user scans or speaks an item with hiku, it is called a beep. These beeps trigger a corresponding event that can be forwarded to third parties using a URL webhook. In addition to the beep events, other events can selectively be forwarded such as device battery levels or notifications of unrecognized barcodes. To register your third party application webhook, send the URL and your app_id to devrel@hiku.us.

Once a user is registered for a third party app, all events the app is configured to receive trigger a webhook POST, i.e. the event data is sent to a URL specific to the app. The body of the POST request contains a data object in JSON format, where the data encoding can be configured per app to be either 'application/json' or 'application/x-www-form-urlencoded'. Event data objects have the following structure:

{"data" : {"eventId": <eventId>, "eventDesc": <eventDesc>, <required parameters>, <optional parameters>}}

Parameters eventId and eventDesc uniquely identify an event, where eventId is a positive integer specifying the event type and eventDesc is a string providing a human-readable description of the event. Each event type has required parameters, which are always present, and optional parameters, which may be present only in some event instances. Parameters may appear in the JSON object in any order. To allow for future extensions of the event API, implementations handling events should ignore additional, unrecognized parameters.

The table below lists the data types used by the event API, which are a subset of the JSON data types.

Type	**JSON Type**	**Description**
Int	Number	32-bit signed integer
String	String	ASCII string
UString	String	Unicode string
HString	String	String containing a hexadecimal number

Beeps/product entry event

Beeps/product entry events are generated whenever a user within the hiku system adds a product to a shopping list. Products can be entered with hiku through a barcode scan or voice entry, or within the hiku or third-party apps through textual entry, selection from favorites, barcode scan, or voice entry.

Parameter	Req/Opt	Type	Description/Value
eventId	Req	Int	1
eventDesc	Req	String	PRODUCT_ENTRY
token	Req	HString	Unique token identifying the user in the third party system
id	Req	Int	Shoplist entry id of the product
name	Req	UString	Product name/description
ean	Req	String	Barcode/EAN, variable length. Set to the EAN when the item was scanned, set to NULL otherwise.
audioPath	Req	String	URL to a WAV file containing the voice recording for playback. Set when voice entry was used, set to the empty string otherwise.
serialNumber	Opt	HString	hiku serial number, only present if a hiku device was used for product entry

EXAMPLES:

Product entry with barcode scan, format application/json
{"data": {"token":
"f4fef8ad14f46595be4c3f62420b08294f68de320b5a2cbab367589a7022e549", "eventDesc":
"PRODUCT_ENTRY", "name": "Caress Velvet Bliss Ultra Silkening Beauty Bar - 6 CT", "ean":
"11111065925", “audioPath”: “”, "serialNumber": "46229f72bce7b25ee82aafb85bc65ede",
"eventId": 1, "id": 8423}}
Product entry with voice, format application/x-www-form-urlencoded
data=%7B%22token%22%3A+%220c4ea3833226aee570bc35397cde804bdbcdbffeb52bcb6
82fc6ff9e2679df9f%22%2C+%22audioPath%22%3A+%22http%3A%2F%2Ftests3.hiku.us.s3.
amazonaws.com%2Ftmp%2Fdb2b75f61cfd0fa7.wav%22%2C+%22eventDesc%22%3A+%22

PRODUCT_ENTRY%22%2C+%22name%22%3A+%22Paper+towels%22%2C+%22serialNumber%22%3A+%2246229f72bce7b25ee82aafb85bc65ede%22%2C+%22eventId%22%3A+1%2C+%22ean%22%3A+NULL%2C+%22id%22%3A+8417%7D

Barcode-not-found event

In some cases, a scanned barcode may not be found in the databases hiku sources from. This may be the case for private-label products or products otherwise not registered in publicly accessible databases. Apps can subscribe to the EAN_NOT_FOUND event to be notified of unrecognized barcodes.

Parameter	Req/Opt	Type	Description/Value
eventId	Req	Int	2
eventDesc	Req	String	EAN_NOT_FOUND
token	Req	HString	Unique token identifying the user in the third party system
eanNotFound	Req	String	Unrecognized barcode, variable length

EXAMPLES:

Barcode not found, format application/json

{"data": {"eventDesc": "EAN_NOT_FOUND", "eventId": 2, "token": "f4fef8ad14f46595be4c3f62420b08294f68de320b5a2cbab367589a7022e549", "eanNotFound": "111111111111"}}

Barcode not found, format application/x-www-form-urlencoded

data=%7B%22eventDesc%22%3A+%22EAN_NOT_FOUND%22%2C+%22eventId%22%3A+2%2C+%22token%22%3A+%220c4ea3833226aee570bc35397cde804bdbcdbffeb52bcb682fc6ff9e2679df9f%22%2C+%22eanNotFound%22%3A+%221111111111111%22%7D

Battery level event

hiku devices periodically report their battery levels, which can be displayed in an app to remind users to charge their devices. The battery level is provided as a percentage, where 100% indicates a full charge and 0% indicates an empty battery. It is recommended that users be reminded to charge their devices once the battery level drops below 20%.
A battery level event is generated whenever a device reports its battery level and the value differs from the previously reported value.

Parameter	Req/Opt	Type	Description/Value
eventId	Req	Int	3
eventDesc	Req	String	DEVICE_BATTERY_LEVEL
token	Req	HString	Unique token identifying the user/device owner in the third party system
batteryLevel	Req	String	Battery level, percentage from 0-100 as a string; can be converted to an integer value
serialNumber	Req	HString	hiku serial number

EXAMPLES:
Battery level, format application/json
{"data": {"token":
"f4fef8ad14f46595be4c3f62420b08294f68de320b5a2cbab367589a7022e549", "eventId": 3,
"serialNumber": "62dfb41be54951c1", "eventDesc": "DEVICE_BATTERY_LEVEL",
"batteryLevel": "1"}}
Battery level, format application/x-www-form-urlencoded
data=%7B%22token%22%3A+%220c4ea3833226aee570bc35397cde804bdbcdbffeb52bcb6
82fc6ff9e2679df9f%22%2C+%22eventId%22%3A+3%2C+%22serialNumber%22%3A+%226
2dfb41be54951c1%22%2C+%22eventDesc%22%3A+%22DEVICE_BATTERY_LEVEL%22
%2C+%22batteryLevel%22%3A+%221%22%7D

/apps

POST /apps - Add a third party app for a user

This is how a third party app initiates the process for adding their app to a user's hiku account. /apps POST returns a frob and barcode image. Present the barcode image to the user to scan with their hiku device. After scanning the barcode, the device is authenticated to the third party app and the device will emit the 'happy up-beep'.

The third party app then calls /apps PUT with the frob to receive a unique user token which will be sent with subsequent webhook POST's for that user.

Form-encoded POST parameters:
None

Sample response:

```
{
"response": {
        "data": {
                "frob": "aUniqueFrob",
                “barcode”: “svgBarcodedImageOfTheFrob”
                },
        "status": "ok"
        }
}
```

PUT /apps/<frob> - Update a third party app to authenticate it

This returns a token that uniquely identifies a user. This token should be stored by the third party application and mapped to their corresponding user, since it will be sent with beep data in the webhook.

For the frob, use the one returned from /apps POST.

Sample response:

```
{
"response": {
        "data": {
                "status": "Authorized"
                "token": "aUniqueUserToken"
                },
        "status": "ok"
        }
}
```

token is only present when ‘status’: ‘Authorized’. If the frob is still awaiting verification, it will return ‘status’: ’New’ and the token will not be present.

```
{
"response": {
        "data": {
                "status": "New"
                },
        "status": "ok"
        }
}
```

hiku user control of Apps

Once a third party adds an app for a user, it will appear in the hiku mobile app under Settings / Apps. To stop data from flowing to the third party, the user simple unchecks the checkbox next to the third party app. Checking the checkbox restarts data flow to the third party.

/list

GET /list - Retrieve all list items for a user.

By default returns items with both active and crossed off status.

GET parameters:
token - the session token

Sample response:


```
{
"response": {
        "data": {
                "list": [
                        {
                        "aisle_id": 70,
                        "audioPath": null,
                        "flFavorite": false,
                        "id": 35,
                        "listTags": [
                                9
                                ],
                        "name": "pumpkin seeds",
                        "product_id": 36,
                        "quantity": 2,
                        "status": 1
                        }
                ]
                },
        "productOrder": [
              {
```

```
            "aisle_id": 13519,
            "name": "No Aisle",
            "productOrder": [
              14073,
              14072,
              14071
            ]
          },
          {
            "aisle_id": 13520,
            "name": "Baby",
            "productOrder": [
              14078,
              14077
            ]
          },
      "regulars": [
          {
            "aisle_id": 1992,
            "name": "Napkins",
            "product_id": 975
          },
          {
            "aisle_id": 1989,
            "name": "Club soda",
            "product_id": 978
          }
      ],
      "status": "ok"
      }
}
```


/list GET also includes a comprehensive list of all regulars, as well as the sort order of the current list by aisle.

NOTE: product_id uniquely identifies a product for that user, while id identifies the incarnation of that product in the users shoplist.

So when you want to update the status of the shoplist item, use the 'DELETE' method with id's.

DELETE /list - Cross off, clear, or reactive list items

GET parameters:
token - the session token
id - comma separated list of id’s
action - accepts ‘clear’, ‘crossOff’, ‘reActivate’ (aka un-delete)

Sample response, note no data payload is returned with /list DELETE:

```
{
"response": {
        "status": "ok"
        }
}
```