KARTE REST API仕様書

株式会社プレイド

v 1.2

目次

1. はじめに

本ドキュメントはWeb接客サービス「KARTE」における、サーバーAPI仕様について記載したものです。
サーバーAPIの利用に関しては、KARTEサポートもしくは担当営業にお問い合わせください。

2. 認証方式

HMACによるメッセージ認証方式。
通信は全てSSLで行います。

3. 注意事項

KARTEでは、ユーザのサイト上における行動ログを取得する、というその特性上、特定のユーザー情報以外のデータに対するDELETE、UPDATE処理を行うことができません。大量データの送信前に、必ず少量のデータでのテストを行ってください。

4. 免責事項

株式会社プレイドはシステム上の不具合を除き、パートナーが本APIの利用の結果生じたデータに対して一切の責任を負いません。

5. 制限事項

本APIでは、以下の制限事項があります。制限事項に触れてしまった場合には、エラーコードが返されます。

項目	制限内容
アクセス回数	60回/分
bodyのデータサイズ	8192bytes
timestamp	現在時間とあまりにもずれが大きい場合

6. API仕様

6.1. リクエスト形式

項目	内容
リクエストURL	https://api.karte.io/v0/track
Method	POST

6.2. header部

パラメータ	内容
ContentType	text/plain; charset=utf-8
Authorization	Authorization:KARTE0-HMAC-SHA256 Credential=”<API_KEY>”, TimeStamp=”<ISO8601形式>”, Signature=”<SIGNATURE>”

Authorizationに含まれる値の説明

API_KEY：管理画面からご確認ください（128bitのHASH値）
ISO8601形式のタイムスタンプ：例 2015-07-15T08:20:15+09:00
SIGNATURE：送信データのbody部に対して、SECRET_KEYを秘密鍵として用いてHMAC-SHA256にて符号化したデータをbase64エンコードしたもの

6.3. body部

body部は全て、JSON形式でエンコーディングされている必要があります。
なお、KARTEの特性上、values内のパラメータに関しては、任意のパラメータを追加することが可能です。

パラメータ	説明	データ例
event_name	KARTEでイベントとして識別される文字列※1	“identify”
api_key	プロジェクトごとに発行されるユニークな識別子（管理画面から確認が可能	“ea703e7aa1efda0064eaa507d9e8ab7e”
timestamp	ISO8601形式のタイムスタンプ	“2015-07-15T08:20:15+09:00”

keys	送信データのキーとなるデータ※2	{
		“user_id” : “20”
		}

values	送信データの中身を格納するデータ※3	{
		“rank” : “gold”,
		“gender” : “male”
		}

※1：event_name で指定できる定義済みの文字列については6.4で詳述
※2：keys で指定できる文字列については6.5で詳述
※3：values内のkey-valueデータのバリデーションついては6.6で詳述

6.4. event_name で指定可能な文字列

KARTEに標準で設定されているイベントのうち、本API指定可能なイベントは下記の通りです。

指定可能な定義済みイベント名

文字列	発生タイミング
identify	ユーザ情報の送信時（ユーザータグの読み込み時）
buy	購入の発生時（buyタグの読み込み時）

また、以下のイベント名はKARTEの集計等に影響を与えるおそれがあるため、指定することができません。

指定不可能な定義済みイベント名

page, req, enter_group, leave_group, view, group, message_send, date, massage_open, message_click, message_clicked, message_close

6.5. keys で指定可能な文字列

KARTEではユーザーおよびセッションの識別に複数のIDを用いており、以下の文字列がユーザーとの紐付けを行うキーとして利用可能です。
なお、一般的な用途では、user_id 以外をkeyで指定する必要はありません。

文字列	用途	必須
user_id	ユーザーを識別するユニークキー。ユーザータグにて送信されるもの	○
visitor_id	全てのユーザーに対してKARTEが自動で付与するキー	
session_id	セッションごとにKARTEが自動で付与するキー	
ktid	マルチドメイン対応環境下で、サードパーティークッキーが有効なユーザーに対して、KARTEが自動で付与するキー	

6.6. values内のバリデーション

以下の条件に一致する文字列を含むデータについては、バリデーションエラーとなり、KARTEを内には取り込まれません。

種別	バリデーションエラーとなる条件
keys	● null bytesが含まれる ● “.”が含まれる ● “$”で始まる
values	● null bytesが含まれる ● 空文字（””） ● null ● undefined

6.7. データ例

header部

```
‘Content-Type’: ‘text/plain; charset=utf-8’
‘Content-Length’: 283
‘Authorization’: ‘KARTE0-HMAC-SHA256
Credential="684dc221098a660375584574af32b3b9",TimeStamp="2015-07-17
T11:57:32+09:00",Signature="f4Ry9NnPWrJkmbChhMLj1YDOAyYPxlJRob8iXvW
rM1U="’
```

body部

```
{
    "event_name":"buy",
    "api_key":"ea703e7aa1efda0064eaa507d9e8ab7e",
    "timestamp":"2015-07-17T11:57:32+09:00",
    “keys”:{
        “user_id”: “20”
    },
    "values":{
        "transaction_id":"12",
        "revenue":9999,
        "shipping":500,
        "tax":100,
        "items":[{
            "item_id":"99",
            "name":"\u307b\u3052\u307b\u3052",
```

```
            "price":9999,
            "quantity":1
        }]
    }
}
```

7. エラーコード

code	name	発生するケース
200	OK	リクエストが正常に終了した場合
400	Bad Request	リクエストの形式に異常がある場合
401	Unauthorized	リクエストが許可されていない場合
404	Not Found	リクエスト対象が見つからない場合
413	Too Large Request	送信データ量が大きすぎる場合
429	Too Many Request	リクエストの送信頻度が多すぎる場合
500	Something Broken	データに破損がある場合

8. サンプルコード

8.1. node.js

```
https = require 'https'
crypto = require 'crypto'

timestamp = "2015-07-17T11:57:32+09:00"
api_key = "ea703e7aa1efda0064eaa507d9e8ab7e"
secret_key = 'BDncn7PMyRwA6tHZAy9obAYOjn9VPcfb'

data = JSON.stringify({
    'event_name': 'buy'
    'api_key': api_key
    'timestamp': timestamp
    'keys': {
        'user_id': '20'
    }
    'values': {
        "transaction_id": "12",
        "revenue": 9999,
        "shipping": 500,
        "tax": 100,
        "items": [{
                "item_id": "99"
                "name": "\u307b\u3052\u307b\u3052"
                "price": 9999
```

```
                "quantity": 1
            }]
        }
    })


_calc_signature = (data) ->
  return crypto.createHmac('sha256',
secret_key).update(data).digest('base64')

signature = _calc_signature(data)

console.log 'Calculated signature:' + signature

options = {
  hostname: 'api.karte.io'
  path: '/v0/track'
  method: 'POST'
  headers: {
    'Content-Type': 'text/plain; charset=utf-8'
    'Content-Length': data.length
    'Authorization': 'KARTE0-HMAC-SHA256 Credential="' + api_key +
'",TimeStamp="' + timestamp + '",Signature="' + signature + '"'
  }
}

req = https.request options, (res) ->
  console.log('STATUS: ' + res.statusCode)
  console.log('HEADERS: ' + JSON.stringify(res.headers));
  res.setEncoding('utf8');
  res.on 'data', (chunk) ->
    console.log('BODY: ' + chunk)

req.on 'error', (e) ->
  console.log('problem with request: ' + e.message)

req.write(data);
req.end();
```

8.2. PHP

```
<?php

  $objDateTime = new DateTime('NOW');
  $timestamp = $objDateTime->format('c');
  $api_key = "ea703e7aa1efda0064eaa507d9e8ab7e";
  $secret_key = 'BDncn7PMyRwA6tHZAy9obAYOjn9VPcfb';
```

```
  function _calc_signature ($data, $secret_key) {
    $hmac = hash_hmac ( 'sha256', $data, $secret_key, true);
    return base64_encode($hmac);
  }

  $data = json_encode([
    'event_name' => 'identify',
    'api_key' => $api_key,
    'timestamp' => $timestamp,
    'values' => [
      'gender' => 'M',
      'user_id' => 1
    ]
  ]);
  $length = strlen($data);

  $signature = _calc_signature($data, $secret_key);

  $options = array(
      'Content-Type: text/plain; charset=utf-8',
      'Content-Length: ' . $length,
      'Authorization: KARTE0-HMAC-SHA256 Credential="' . $api_key .
'",TimeStamp="' . $timestamp . '",Signature="' . $signature . '"'
  );

  $curl = curl_init();
  curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($curl, CURLOPT_POST, true);
  curl_setopt($curl, CURLOPT_URL, "https://api.karte.io/v0/track");
  curl_setopt($curl, CURLOPT_HTTPHEADER, $options);
  curl_setopt($curl, CURLOPT_POSTFIELDS, $data);
  $result = curl_exec($curl);
  curl_close($curl);

 ?>
```

8.3. Python

```
# -*- coding: utf-8 -*-

# pythonの構築時、SSLをサポートするようにsocketモジュールをコンパイルしている
必要がある

import httplib
import hashlib
```

```
import hmac
import json
import base64
from datetime import datetime

api_key = 'mock'
secret_key = 'BDncn7PMyRwA6tHZAy9obAYOjn9VPcfb'

timestamp = datetime.now().isoformat()

data = json.dumps({
  'event_name': 'identify',
  'api_key': api_key,
  'timestamp': timestamp,
  'values': {
    'user_id': '0001',
    'gender': 'M',
  }
})

def _calc_signature(data):
    digest = hmac.new(secret_key, data, hashlib.sha256).digest()
    return base64.b64encode(digest)

signature = _calc_signature(data)

hostname = 'api.karte.io'
path = '/v0/track'

req = httplib.HTTPSConnection(hostname)
req.putrequest('POST', path)
req.putheader('content-type', 'text/plain; charset=utf-8')
req.putheader('content-length', str(len(data)))
req.putheader('authorization',
              'KARTE0-HMAC-SHA256 Credential="' + api_key +
              '",TimeStamp="' + timestamp +
              '",Signature="' + signature + '"')
req.endheaders()
req.send(data)

res = req.getresponse()

print 'STATUS:', res.status
if res.msg:
    print 'MESSAGE:', res.msg
print 'HEADERS:', json.dumps(res.getheaders())
print 'BODY:', res.read()
```

8.4. Ruby

8.5. JAVA

```
package sample;

import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpPost;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang3.time.FastDateFormat;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;

import com.fasterxml.jackson.databind.ObjectMapper;

public class Main {

      //プロジェクト固有のKey
      static final String api_key = "mock";
      static final String signatureKey =
"weoLrGQdhsadfachyxpLhGhE7dad8C0FH";

      public static void main(String[] args)
                  throws NoSuchAlgorithmException,
InvalidKeyException, ClientProtocolException, IOException {

            //データ作成部分
            Date now = new Date();
            FastDateFormat fdfIso8601ExtendedFormat =
FastDateFormat.getInstance("yyyy-MM-dd'T'HH:mm:ss.SSSXXX");
            String timestamp =
fdfIso8601ExtendedFormat.format(now);
            String event_name = "identify";
            String user_id = "X0002";
            String gender = "F";

            Map<String, Object> values = new HashMap<String,
Object>();
            values.put("uesr_id", user_id);
            values.put("gender", gender);

            Body body = new Body(event_name, api_key, timestamp,
```

```
user_id, values);

            //JSON形式に変換
            ObjectMapper mapper = new ObjectMapper();
            String json = mapper.writeValueAsString(body);

            //暗号化
            String algorithm = "HmacSHA256";
            SecretKey secretKey = new
SecretKeySpec(signatureKey.getBytes(), algorithm);
            Mac mac = Mac.getInstance(algorithm);
            mac.init(secretKey);
            mac.update(json.getBytes());
            byte[] encryptedData = mac.doFinal();
            String signature = new
String(Base64.encodeBase64(encryptedData));

            String apiurl = "https://api.karte.io/v0/track/";
            CloseableHttpClient client =
HttpClientBuilder.create().build();
            HttpPost post = new HttpPost(apiurl);

            //ヘッダのセット
            post.setHeader("Content-Type", "text/plain;
charset=utf-8");
            post.setHeader("Authorization", "KARTE0-HMAC-SHA256
Credential=\"" + api_key + "\",TimeStamp=\"" + timestamp
                        + "\",Signature=\"" + signature + "\"");

            //Bodyのセット
            StringEntity stringEntity = new StringEntity(json,
HTTP.UTF_8);
            post.setEntity(stringEntity);

            //送信
            final HttpResponse response = client.execute(post);
            client.close();

            //結果のステータスコードを表示

System.out.println(response.getStatusLine().getStatusCode());
      }

      static class Body {
            public String event_name;
            public String api_key;
            public String timestamp;
            public Keys keys;
            public Map<String, Object> values;

            Body(String event_name, String api_key, String
timestamp, String user_id, Map<String, Object> values) {
                  this.event_name = event_name;
                  this.api_key = api_key;
                  this.timestamp = timestamp;
                  this.values = values;
                  this.keys = new Keys(user_id);
```

```
            }

            static class Keys {
                  public String user_id;

                  Keys(String user_id) {
                        this.user_id = user_id;
                  }
            }
      }
}
```

9. よくあるエラー原因と確認箇所

400エラーが返ってくるが、形式は正しいように思える
→ヘッダーのフォーマットが誤っている可能性があります。ダブルクオートでくくられているか、大文字・小文字の間違いはないかを確認してください。

タイムスタンプでエラーが発生する
→形式が誤っている、もしくは現在時間と大幅に時間がずれている可能性があります。また、header部とbody部のタイムスタンプが一致していることを確認してください。形式についてはISO8601フォーマットになっていることを確認してください。

200が返ってくるのに、管理画面ではデータが反映されていない
→APIサーバに到達する前の処理でエラーが発生している可能性があります。この場合レスポンスのbodyに{“status”:2}が返されます。6.6に記載のあるバリデーション情報に触れている可能性があります。

シークレットキーとは何ですか？
→「サイト設定」内にある「暗号化キー」のことです。

全て正しく設定されているのに、それでもエラーが発生する
→KARTEサーバAPI機能はオプション機能です。機能を有効化するには株式会社プレイド側での設定が必要になります。