# Squeezol API RESTful 1.3.3

Split as a Service

Ing. Davide Costa, Ing. Fabio Fiori

11 Luglio 2016

---

## Contents

# 1 Change Log

## 1.1 From 1.0

Squeezol API 1.3.1 admit the Authenticated User to modify his shopping Cart with the help of two simple functionality:

- **Check the shopping cart:** Get the number of item contained in the shopping cart
- **Modify the shopping cart**: Update products and or product quantity
- **Products Quantity Specification**: Specify products quantity during Split creation

## 1.2 From 1.3.1

- **Terminology:** "Split" instaed of misleading "Wallet"
- **Terminology:** Section 4.1 "Integration with ecommerce shopping cart" become "Integration with products database"
- **Modify the shopping cart**: Moved section 4.2 "Modify the shopping cart" to section 4.2.1 "Modify Split Shopping cart"
- **Exhaustive explanation**: how to modify the shopping cart related to a Split section 4.2.1

## 1.3 From 1.3.2

- **Terminology:** Added Required/Optional information to each field in each API call

# 2 Introduction

This guide discusses how to use the Squeezol Documents List API version 1.0. What can this API do?

The Squeezol RESTful API allows developers to create, invite and make a payment to Squeezol Splits.

You will find this API useful if you need to allow group Payments for your Business or to your customers. All interactions with the API require a valid Squeezol Account, which can be a "consumer" account, or a Squeezol Business or Developer account.

## 2.1 Audience

This developer's guide is intended for software developers needing a technical reference for using the Squeezol RESTful API. The information you find in this guide is written by Squeezol engineers, and is authoritative on how the API behaves.

## 2.2 Terminology used in this guide

Throughout this guide, we refer to a series of terms like Split, Invited, Participant and Administrator which we intend to have specific meanings.

- **Split:** an item created by a user that become the Split Administrator: every Split can accept payments from one or more people through the payment method selected by the business partner. The Administrator can invite someone to participate to fund the Split. Each Split can have different status code: WAC(Waiting For Acceptance), WPA(Waiting For Payments), DES(Deserted), CWS(Closed With Success)
- **Administrator**: the user that decided to create and open the Squeezol Split. He is the only Participant that can take decisions about his Split.
- **Invited**: an item deriving from an invitation process through Squeezol platform or Squeezol RESTful API.
- **Participant**: an Invited become Participant when he/she accept to participate or not to a specific Split. A unique identifier is allocated for each Split. This is of the format $group\_id = id$, e.g. $group\_id : 12345$.
- **Ecommerce**: represent the organization and or company which will install and use Squeezol APIs.
- **Client**: is the Oauth2 client instance corresponding to a specific partner, e.g. if you are a software house installing Squeezol services you will create one Partner for each of your customer and one or more Client associated with it.

Occasionally, we make a mistake in this documentation, or unknowingly break a feature that you depend on. If this has happened to you, sorry! You can help us fix the issue by posting in the forum or by filing a bug. We appreciate your help!

# 3 Setting up your client library

A number of client libraries are provided in various languages. These client libraries make it easier to interact with the RESTful API. We are currently supporting the following server side API:

- Python-Django 1.7
- PHP
- ASP .NET 2.0

Additionaly, Squeezol API ships with a javascript library, integrating JQuery as well, as a front end plug and play solution. Go to the CMS plugin page to find out if it already exist a plugin for your CMS flavor.

## 3.1 Authorizing requests

When your application requests non-public user data, it must include an authorization token. The token also identifies your application to Squeezol. About authorization protocols We assume mandatory the use of OAuth 2.0 to authorize requests.

If your application has certain unusual authorization requirements, such as logging in at the same time as requesting data access (hybrid) or domain-wide delegation of authority (2LO), then you cannot currently use OAuth 2.0 tokens.

Authorizing requests with OAuth 2.0

Requests to the Squeezol RESTful API for non-public user data must be authorized by an authenticated user.

The details of the authorization process, or "flow," for OAuth 2.0 vary somewhat depending on what kind of application you're writing. The following general process applies to all application types:

1. When you create your application, you register it with Squeezol. Squeezol then provides information you'll need later, such as a client ID and a client secret.

2. In order to make authorized requests you need the explicit authorization by Squeezol staff for production. If you want to try the sandbox platform the authorization is not required.

3. When your application needs access to user data, it asks Squeezol for a particular scope of access. Squeezol displays an OAuth dialog to the user, asking them to authorize your application to request some of their data.

4. If the user approves, then Squeezol gives your application a short-lived access token. Your application requests user data, attaching the access token to the request. If Squeezol determines that your request and the token are valid, it returns the requested data.

Some flows include additional steps, such as using refresh tokens to acquire new access tokens. For detailed information about flows for various types of applications, see Squeezol's OAuth 2.0 documentation.

## 3.2 Tutorial

### 3.2.1 Server-to-Server

Squeezol client side APIs use **JSON data-format** to perform server-to-server calls to Squeezol backend server. Feel free to use our server side APIs or perform raw calls in JSON format!

Inside each example, the following notiation is assumed:

- **Required**: The field is required and can not be left null or blank
  i.e.Bad call

```
{
  ...
  'name': ''                                                    // Required
  ...
}
```

- **Optional**: The field can be left null or blank but must be present
  i.e.Good call

```
{
  ...
  'description': ''                                             // Optional
  ...
}
```

### 3.2.2 Setup

Download and install the required APIs based on your server technology. In order to make authorized requests you have to register and login to https://www.squeezol.com and then go to the ecommerce dashboard and setup a ecommerce instance. After that, clicking on the desired ecommerce you need to create an Oauth2 Client instance providing mandatory information:

- **Name**: Client name
- **Web site URL**: ecommerce website URL
- **CallBack URL**: Redirect URL after Oauth2 authorization
- **IPN URL**: Instant Payment Notification URL

**Warning**: URLs must return HTTP status-code: 200.

Then Squeezol will provide the CLIENT_ID(or SQUEEZOL_ID) and CLIENT_SECRET and copy-paste them into your API configuration file. Keep this information safe!

**Remember**: In sandbox and production environment you need Squeezol Staff approval.

### 3.2.3 Authorization Request

Before start performing requests you need to request and obtain a valid Access Token which will authorize you to perform calls depending on the requested SCOPE.

- 'create': authorization to create a Split
- 'pay': authorization to pay into a Split
- 'create+pay': creation and pay permissions

Every request must be authorized via Access Token using the following header construct:

```
{
  "Authorization":"Bearer 1e7c27bb6dee449ebf2eaf4b9a7242b1a6e1865c",
  "Content-Type":"application/json"
}
```

### 3.2.4 Login with Squeezol

This section gives directives for Squeezol login integration. **targetUrl** which is used to start the standard Oauth2 autorization process

```
AUTHORIZATION ENDPOINT: /plugin/authorize/
<script>
var authorization_url;
function Authorize(targetUrl, access_token) {
  var popupWin;
  if(window.addEventListener)
    window.addEventListener("message", PostMessageReceiver,false);
  else if(window.attachEvent)
    window.attachEvent("onmessage", PostMessageReceiver);
  if(access_token != "")
    window.location.replace(targetUrl);
  else
    popupWin = PopupCenter(targetUrl, "sq_login", 600, 550);
}
// When the HTML5 Postmessage contains the "code" needed to Oauth2 in order
// to complete the authorization process and get the Access Token.
// When is received, the popup is closed automatically by Squeezol
function PostMessageReceiver(e) {
  var DecodedString=decodeURIComponent(e.data);
  var data=JSON.parse(DecodedString);
  if (data.action == 'close-pop'){
    window.location.replace(authorization_url+'?'+data.code);
  }
}
function PopupCenter(url, title, w, h) {
  var dualScreenLeft = window.screenLeft != undefined ? window.screenLeft :
      screen.left;
  var dualScreenTop = window.screenTop != undefined ? window.screenTop :
      screen.top;
  width = window.innerWidth ? window.innerWidth :
      document.documentElement.clientWidth ?
                                  document.documentElement.clientWidth :
                                      screen.width;
  height = window.innerHeight ? window.innerHeight :
      document.documentElement.clientHeight ?
                                      document.documentElement.clientHeight :
                                          screen.height;
  var left = ((width / 2) - (w / 2)) + dualScreenLeft;
  var top = ((height / 2) - (h / 2)) + dualScreenTop;
  var newWindow = window.open(url, title, 'scrollbars=yes, width=' + w + ',
                                      height=' + h + ', top=' + top + ',
                                          left=' + left);
  if (window.focus) {
    newWindow.focus();
  }
  return newWindow;
}
</script>
```

# 4 Raw APIs

Here's the OAuth 2.0 information for the Squeezol RESTful API:

## 4.1 Integration with products database

Add product to Squeezol Database(mandatory): If a product is already present in the database an update of fields is performed. In case of error, the validation error message is present in the message returned.

```
Request: POST /api/partner/add_product/
{'products':[
             {
               'product_id': 'ecommerce id for this product',   // Required
               'name': 'Product name',                          // Required
               'description': 'Product description',            // Required
               'url': 'http://your-web-site.com/url_of_product_page',// Required
               'price_currency': 'EUR or USD',                  // Required
               'price': '100.00',                               // Required
               'partner': 'CLIENT_ID'                           // Required
             },
             {...}
            ]
}

Response:

HTTP status-code: 200 OK

{
  'status': 'ok',
  'message': 'Products added with success',
  'products': [
               {'saved': 'product_id'}, // if new product
               {'already_saved': 'product_id'} // if updated/modified
              ]
}

HTTP status-code: 200 OK

{
  'status': 'error',
  'products': [
               {'saved': 'product_id'},
               {'already_saved': 'product_id'},
               {'error': 'product_id', 'message': 'Validation error'}
              ]
}
```

## 4.2 split Setup

Split Creation. In this phase products previously added to the database must be associated to the Split as shown below. Starting from version 1.3 the quantity of each product can be specified:

Date format: dd-mm-YYYY

```
Request: POST /api/group/create/
{
  'name': 'Split name',                                          // Required
  'description': 'Split description',                            // Optional
  'occurrence': 'R',                                             // Optional
  'max_acceptance_date': '23-7-2014',                            // Optional
  'max_payment_date': '9-8-2014',                                // Optional
  'isOpen': 'false',                                             // Optional
  'promo': 'Promo code(if any)',                                 // Optional
  'hide_invitation': 'false',                                    // Optional
  'hide_contribution': 'false',                                  // Optional
  'alert_email': 'false',                                        // Optional
  'currency': 'EUR or USD',                                      // Required
  'amount': '100.00',                                            // Required
  'order_id': 'Unique identifier for your order',                // Optional
  // 'S': split equally, 'F': fixed amount, 'D': suggested amount
  'fundraising': 'S',                                            // Optional
  // NEW in 1.3(*) In case you want to keep track of product quantity, both
     solution are allowed
  'products': ['B001', 'B002'],                                  // Required(*)
  // OR
  'products': {'B001': 1, 'B002': 2 },                           // Required(*)
  'partner': 'CLIENT_ID',                                        // Required
  // Required
  'invitation_url': 'http://your-web-site.com/invitations.php?page=group_id',
  // Required
  'digest_url': 'http://your-web-site.com/digest.php?page=group_id'
}

Response:
HTTP status-code: 200 OK
{
  'status': 'ok',
  'url': 'http://your-web-site.com/invitations.php?page=100',
  'group_id': 100
}

Response(validation errors)
HTTP status-code: 200 OK
{
  'status': 'error',
  'form_error': ['name': 'error name', 'description': 'error description' ]
}
```

### 4.2.1 Shopping cart update (new in 1.3)

This functionality allow an authorized user to dinamically modify the shopping cart and automatically update the total amount associated with it.

Get products quantity in the shopping cart

```
Request: GET /api/group/cart/modify/
{
  'group_id': 100                                                    // Required
}
Response:
HTTP status-code: 200 OK
{ "status":"ok",
  "products":[{"pid":"B001","qty":1.0},
              {"pid":"B002","qty":2.0},
              {"pid":"B003","qty":3.0}, ... ]
}

Response(validation errors)
HTTP status-code: 200 OK
{
  "status": "error", // Without parameter group_id
  "message": "Insert group ID"
}

HTTP status-code: 200 OK
{
  "status": "error", // Group corresponding with group_id not found
  "message": "Group not found"
}
```

After getting the correct information about the shopping cart is mandatory to modify each product to the desired quantity(even if not updated).

```
Request: POST /api/group/cart/modify/
{
  "group_id": 100,                                                    // Required
  "products" : [{
                 "pid":"B001",                                        // Required
                 "qty":2.0                                            // Required
                },
                {
                 "pid":"B002",
                 "qty":3.0
                },
                .....
                {
                 "pid":"B00N",
                 "qty":M
                }
                ]}
}
Response:
HTTP status-code: 200 OK
{
  "status":"ok",
  "message": "Shopping cart updated correctly"
}

Response(validation errors)
HTTP status-code: 200 OK
{
  "status":"error",
  "message": "Product 'Ref_004' not found, add it before updating"
  // In this case you are required to add the product in to the Data base with
  // POST /api/partner/add_product/ (Section 4.1)
}

HTTP status-code: 200 OK
{
  "status":"error",
  "message": "The resulting amount is less the one already paid"
}

Response(Bad Request)
HTTP status-code: 400 BAD REQUEST
{
  "status":"error", // If the Group is already concluded
  "message": "Operation not Allowed"
}
```

## 4.3 Invitation to a Split

Get invitation list

```
GET /api/group/invitation/

{
 'group_id': '100',                                          // Required
 'partner': 'CLIENT_ID'                                      // Required
}
```

Response:

```
{
 "status":"ok",
 "p_admin":{ //Administrator
           "id":110,"user":3,
           "group":100,"status":"A",
           "single_amount":"100.00",
           "paypal_pay":null
          },
 "group": { // Split data
          "name":"Split name",
          "description":"Split Description",
          "max_acceptance_date":"2014-07-22T22:00:00Z",
          "max_payment_date":"2014-08-09T22:00:00Z",
          "status":"WAC",
          "fundraising":"S",
          "occurrence":"R",
          "alert_email":false,
          "isOpen":false,
          "amount":"100.00",
          "currency":"EUR or USD",
          "hide_contribution":false,
          "hide_invitation":false,
          "promo_code":null,
          "partner":"Partner Name"
         },

 "invited": [ // Invited list
             {
              "group_id": "100", "email": "jhon@doe.com", "fb_id":"",
              "name": "Jhon Doe", "avatar_url": "path_to_picture"
             }
            ],
 "participants":[ // Participant list
                 {
                   "id":110,"user":3,"group":100,"status":"A",
                   "single_amount":"10.00","paypal_pay":null
                 }
                ],
 "params":{ // Extra parameters
           "group_id": 100, "social_providers": ["google", "facebook"],
           "admin_email": "nafta86@gmail.com", "google_friends": "True",
           "fb_url":
              "https://sandbox.squeezol.com/api/group/fb_invitation/100/"
          },
 "gdata": [{  // Google friends picture and email
            "picture":"https:48f0b78bd77ea2?access_token=ya29.xGMpIW5TvSmlfBZNvXDbM",
            "value":"jhon@doe.com"
          }]
}
```

INVITATION to join a Split:

Email invitation: an email is sent to each valid email address.

```
POST /api/group/invitation/
{
  'group_id': '100',                                          // Required
  'action': 'email',                                          // Required
  // Default amount proposed to participants
  'admin_contrib': '10.00',                                   // Required
  'fb_array': '[]',
  'mailArray': '[{"email": "jhon@doe.com",                    // Required
                  "avatar_url": "full_path_2_avatar"},        // Optional
                 {...}]',
  'participant_id': '99'                          // (Group admin ID) Required
}

Response:

HTTP status-code: 200 OK

{
  'status': 'ok',
  'action': 'invite'
}
```

Update the amount single amount supposed to be paid:

```
POST /api/group/invitation/
{
  'group_id': '100',                                          // Required
  'action': 'next',                                           // Required
  'admin_contrib': '10.00',                                   // Required
  'fb_array': '[]',                                           // Optional
  'mailArray': '[]',                                          // Optional
  'participant_id': '99'                                      // Required
}

Response:

HTTP status-code: 200 OK
{
  'status': 'ok',
  'action': 'next',
  'redirect_url': 'invitation_url'
}
```

### 4.3.1 Frontend: Facebook dialog iframe

This API allow to open the Private Message Facebook Dialog in order to provide an extra invitation tool for the user who logged in with Squeezol Facebook App.

Opening an iframe at the GET parameter: `params.fb_url`, for example in a bootstrap modal dialog will allow the user to send an invitation through Squeezol Facebook App. The invitation list will be sent back by Squeezol through HTML5 PostMessage.

```
Twitter Bootstrap example
<script>
var socIframe = '<iframe scrolling="auto" class="embed-responsive-item"
    src="'+params.fb_url+'"></iframe>'
jquery("#appendSqIframe").append(socIframe);
jquery("#fb-modal-iframe").modal();
</script>

<body>
  <div class="modal fade" id="fb-modal-iframe">
    <div class="modal-dialog">
      <div class="embed-responsive embed-responsive-4by3" id="appendSqIframe">
      </div>
    </div>
  </div>
</body>
```

The return parameters from the invitation allow to close the modal(ore hide the iframe) and provide the list of invited users(already recorded by Squeezol) as follow:

```
var PostMessageReceiver = function(e) {
  var DecodedString=decodeURIComponent(e.data);
  var data=JSON.parse(DecodedString);
  var modal;
  if (data.action == 'facebook'){
      modal = jquery('#fb-modal-iframe');
      modal.modal('hide');
    if (data.fbList)
      showInvitationList(data.fbList);
  }
}
```

## 4.4 Payments

Payment page:

```
GET /api/group/digest/
{
 'group_id': '100',                                          // Required
 'partner': 'ac375f47918c5e838e83'                           // Required
}
Response:
{  "status":"ok",
   "group": { // Split data
          "name":"Split name",
          "description":"Split Description",
          "max_acceptance_date":"2014-07-22T22:00:00Z",
          "max_payment_date":"2014-08-09T22:00:00Z",
          "status":"WAC",
          "fundraising":"S",
          "occurrence":"R",
          "alert_email":false,
          "isOpen":false,
          "amount":"100.00",
          "currency":"EUR or USD",
          "hide_contribution":false,
          "hide_invitation":false,
          "promo_code":null,
          "partner":"ecommerce Name"
        },
   "participants": [{ // Participant list
                   "status": "A",
                   "group": 100,
                   "name": "Jhon Doe",
                   "avatar_url": "https://photo.jpg",
                   "user": 3,
                   "single_amount": 3.00,
                   "id": 110}],
   "params": { // Parameters for payments
             "totalPerc": 0, "p_status": "A", "canFinish": false,
             "totalPay": 0, "p_single_amount": 10.00,
             "daysLeft": 24, "squeezol_site": "sandbox.squeezol.com",
             "is_admin": true, "pAdminId": 110, "participant_id": 99,
                "group_id": 100,
             "invitation_url": "http://your-web-site/invitations.php?page=100",
             "canOpenPay": true, "openPay": false
           },
   "invited":[{ // Invited list
              "id":26,"group":100,"email":"jhon@doe.org",
              "fb_id":null,"name": "Jhon Doe","avatar_url":null
           }]
}
```

### 4.4.1 Administrator: Open Payments

The Administrator must open payments before anyone could perform any payments. The Administrator is the only Participant allowed to open, close, refund payments.

Open Payments

```
POST /api/group/digest/
{
  "action": "OP",                                          // Required
  "group_id": "100",                                       // Required
  "participant_id": 110,                                   // Required
  "single_amount": 10,                                     // Required
  "partner": "CLIENT_ID"                                   // Required
}

Response:
{
  "status":"ok",
  "squeezol_site":"sandbox.squeezol.com",
  "group_id":"100",
  "response":"OP" // Split status: WPA (Waiting For Payments)
}
```

### 4.4.2 Administrator: Close Payments

Close Split

```
POST /api/group/digest/
{
  "action": "CG",                                          // Required
  "group_id": "100",                                       // Required
  "participant_id": 110,                                   // Required
  "single_amount": 10.00,                                  // Required
  "partner": "CLIENT_ID"                                   // Required
}

Response:
{
  "status":"ok",
  "squeezol_site":"sandbox.squeezol.com",
  "group_id":"100",
  "response":"CG" // Split status: CWS (Closed With Success)
}
```

### 4.4.3 Administrator: Refund Payments

Refund Payments(every payment will be refunded):

```
POST /api/group/digest/
{
  "action": "RG",                                                    // Required
  "group_id": "100",                                                 // Required
  "participant_id": 110,                                             // Required
  "single_amount": 10.00,                                            // Required
  "partner": "CLIENT_ID"                                             // Required
}

Response:
{
  "status":"ok",
  "squeezol_site":"sandbox.squeezol.com",
  "group_id":"100",
  "response":"RG" // Split status: DES (Deserted)
}
```

### 4.4.4 Participant: Modify amount

Each participant can modify its personal amount to be paid. **Warning:** the amount the user have to pay must be saved in the Squeezol Data Base using this specific action.

```
POST /api/group/digest/
{
  "action": "SA",                                                    // Required
  "group_id": "100",                                                 // Required
  "participant_id": 110,                                             // Required
  "single_amount": 10.00,                                            // Required
  "partner": "CLIENT_ID"                                             // Required
}

Response:
{
  "status":"ok",
  "squeezol_site":"sandbox.squeezol.com",
  "group_id":"100",
  "response":"SA" // Split status not modified: WPA or WAC
}
```

### 4.4.5 Participant: Pay

Pay through redirect to Squeezol POS: 1. Prepare:

```
POST /api/group/digest/
{
  "action": "P",                                                    // Required
  "group_id": "100",                                                // Required
  "participant_id": 110,                                            // Required
  "single_amount": 10.00,                                           // Required
  "partner": "CLIENT_ID"                                            // Required
}

Response:
{
  "status":"ok",
  "group_id":"100",
  "response":"P" // Split status not modified: WPA or WAC
  "squeezol_site":"https://sandbox.squeezol.com",
  "redirect_url": "/payments/pay/GestPay/"
}
```

2. Pay: The user will be redirected after POST and the returned again on your website

```
POST https://sandbox.squeezol.com/payments/pay/GestPay/
{
  "participant_id": "110",                                          // Required
  "group_id": "100"                                                 // Required
}
```

## 4.5 Participant: Accept invitation to Group (New in 1.3.1)

The request is an authorized POST as follow. Subsequent calls to modify the amount and pay for the participant are handled as defined in section 4.5.5, Participant: Pay

```
POST /api/group/participant/
Request:
{
  "group_id": "group_id",                                           // Required
  // Proposed amount to be paid, can be modified later on
  "single_amount": "50.00"                                          // Required
}

Response:
{
  "status":"ok",
  "participant_id" : "1234",
  "message": "Partecipante aggiunto con successo"
}
```

## 4.6 Participant: Get the list of Split opened through a partner

The request is an authorized GET without parameters.

```
GET /api/groups/

Response:
{ "status":"ok",
  "groups":
    [{
      "id":"group_id",
      "name": "Split name",
      "status": "Split status",
      "amount": "Target amount",
      "total_paid": "Amount paid",
      "days_left": "Days left to complete the fund",
      "picture": "Split picture"
    },
    {...}]
}
```

# 5 Payment Notification (IPN)

Once each Participant has paid his share and the goal amount is reached, the group administrator can complete the payment. In that moment a notification is sent to the IPN URL provided during Client creation.

```
Example of IPN notifications
1. payment status: Success(the payment has been confirmed by the administrator)
{
  'status': u'S',
  'currency': 'EUR',
  'amount': 100.00,
  'group': 100,
  'creation_date': '2015-05-18 09:03:32.434618+00:00',
  'checksum': '774de6314260dd4634d3fdad776809faa5c2b29e67ff5b98783c0bee0320414b'
}

2. payment status: failure(the payment has been canceled by the administrator)
{
  'status': u'I',
  'currency': 'EUR',
  'amount': 100.00,
  'group': 100,
  'creation_date': '2015-05-18 09:03:32.434618+00:00',
  'checksum': '774de6314260dd4634d3fdad776809faa5c2b29e67ff5b98783c0bee0320414b'
}
```

If the checksum security check is passed:

```
// pseudo code:
if (SHA256(group,status,CLIENT_ID,SECRET_ID) == checksum)
```

The server must reply to the IPN notification with one of the following JSON response:

```
1. payment status: Success
{
  'status': u'S',                                          // Required
  'group': 100,                                            // Required
  'order_id': 'ORDER_ID'                                   // Optional
}

2. payment status: Failure
{
  'status': u'I',                                          // Required
  'group': 100,                                            // Required
  'order_id': 'ORDER_ID'                                   // Optional
}
```

# 6 Frontend solution: plug and play Javascript

The javascript library integrates with every client side library provided by Squeezol s.r.l. The javasccript library ships integrating JQuery and Bootstrap v3(.js and .css). This solution does not provide flexibility and customization for UI/UX but it guarantees quick and smart integration with Squeezol APIs client. The library provide Squeezol split payment service, as follow:

1. Creation of "Pay With Squeezol" button and Oauth2 request:

```
SqueezolApi.createSqButton( Squeezol_auth_request_URL, buttonSize,
   access_token );
```

2. Split cration page:

```
SqueezolApi.createGroup( TotalAmount, // cart amount
                         Currency,   // Currency
                         codProducts, // List of product IDs
                         crea_endpoint_URL, // Creation endpoint URL
                         gestisci_URL, // Payment page URL
                         invita_URL); // Invitation page URL
```

3. Invitation page, Email, Google contacts and Facebook:

```
SqueezolApi.getInvitationData( id_colletta, invitation_endpoint_URL );
```

4. Payment page:

```
SqueezolApi.getDigestData( id_colletta, digest_endpoint_URL );
```