Tokenisation und IFramelösung v1.0

G. Franke, Micropayment™

Juli 2016

Contents

Einleitung

Für Zahlungen per Kreditkarte benötigen Sie einen Account bei Micropayment und ein eingerichtetes Project.

Eine Zahlung per Kreditkarte kann auf 2 unterschiedlichen Wegen erfolgen. Wir unterscheiden Folgendes:

- Purchase die Zahlung wird sofort durchgeführt.
- Authorization / Capture die Zahlung erfolgt in 2 Schritten. Mit Authorization wird ein Betrag für einen gewissen Zeitraum reserviert, d.h. das Kreditlimit des Kunden wird um diesen Betrag reduziert. Der reservierte oder ein kleinerer Betrag wird dann zu einem späteren Zeitpunkt mit Capture gebucht.

Bei der Kreditkartenzahlung unterscheiden wir weiterhin zwischen der Einmal- bzw. Erstzahlung und möglichen Folgezahlungen nach einer Erstzahlung.

Grundsätzlich erfordert jede Zahlung (Purchase / Authorization) den cvc Code. Bei Folgezahlungen nach einer erfolgreichen Erstzahlung kann auf den cvc Code verzichtet werden.

Kreditkartenzahlungen laufen bei Micropayment wie folgt ab (siehe API-Funktionen):

Erstzahlung:

1. Customer erzeugen. Beschrieben mit customerId.
2. Kreditkartendaten dem Customer zuweisen
3. optionale Daten dem Customer zuweisen (Anschrift etc.)
4. Session erzeugen. wichtige Parameter: customerId, amount, currency
5. Transaction erzeugen in Form einer Zahlung per Purchase oder Authorization/Capture. Parameter: cvc

Kreditkarteninformationen werden also immer über den Customer referenziert. Für eine Folgebuchung ohne Eingabe des cvc Codes benötigen Sie also einen Customer, referenziert über die customerId, dem Kreditkartendaten zugeordnet sind und mit dem bereits eine erfolgreiche Buchung gemacht wurde.

Folgezahlung:

1. Session erzeugen. wichtige Parameter: customerId, amount, currency
2. Zahlung per Purchase oder Authorization/Capture. optionaler Parameter: cvc

Wenn Kreditkartendaten durch Scripte des Händlers erfaßt/bearbeitet werden sollen, ist eine entsprechende PCI Zertifizierung erforderlich. Um dies zu umgehen, stellt Micropayment eine IFramelösung bereit, wodurch die Händlerscripte nie unmittelbar mit pan und cvc in Berührung kommen. Die IFramelösung ist als JsBridge implementiert.

Bei der Entgegennahme der Kreditkartendaten wird ein temporärer Token erstellt. Dieser wird bei einigen Funktionen als Referenz auf die Kreditkartendaten benötigt. Somit ergeben sich folgende Schritte.

Erstzahlung:

1. Tokenisation per JsBridge
2. Initialisierung der Zahlung wahlweise per
    - JsBridge “prepare”/“amount”
    - JsBridge “session” (siehe API-Funktion tokenSessionCreate())
3. Zahlung per JsBridge oder per API-Funktion
    - per JsBridge “process” als Purchase oder Authorization (letzteres muss dann per API-Funktion Capture abgeschlossen werden)
    - per API-Funktionen (Purchase oder Authorization/Capture)

3Dsecure für Purchase und Authorization ist momentan nur per JsBridge, also nicht über die API-Funktionen möglich!

Nach der Erstzahlung wird an die von ihnen definierte Rücksprungadresse eine ID übermittelt, die cryptId. Mit dieser ID können Sie für einen kurzen Zeitraum (ca. 2 Stunden) den Transaktionsstatus abfragen (API-Funktion transactionGetCrypt()). Im Ergebnis dieser Abfrage erhalten Sie neben der customerId (benötigen Sie für mögliche Folgezahlungen) die transactionId und die sessionId. Diese Daten werden ihnen auch per Notification mitgeteilt. Mit der transactionId können Sie per API-Funktion transactionGet() zeitlich unbegrenzt den Transaktionsstatus abfragen.

Folgezahlung (wie gehabt):

1. Session erzeugen. wichtige Parameter: customerId, amount, currency
2. Zahlung per Purchase oder Authorization/Capture. optionaler Parameter: cvc

Erfassung der Kreditkartendaten

Die Kartendaten werden per Formular entgegengenommen.

Relevante Felder sind

- holder
- year
- month
- pan
- cvc

Diese Felder sind Pflichfelder und müssen immer in einem Schritt verarbeitet werden.

Typischerweise sind die Felder Bestandteil eines größeren Formulars bei dem z.B. auch noch email etc. erfaßt werden sollen. Aus Sicherheitsgründen muss zumindest pan und cvc in einem iframe der bei Micropayment gehostet ist, erfaßt werden.

Begrifflich ergibt sich somit das Händlerformular und das/die IFrame(s) wobei die IFrames ausschließlich per JavaScript mittels JsBridge erstellt und verwaltet werden. Im weiteren wird deshalb immer von JsBridge gesprochen.

JsBridge

Funktionen

Folgende Funktionen werden über “Micropayment” bereitgestellt. Die Billingfunktion dienen dazu, nach der Tokenisation direkt eine Buchung vorzunehmen.

- **init**

  Syntax: *Micropayment.init();*

  Initialisiert die IFrames pan/cvc für die Tokenisation und wird normalerweise automatisch beim laden des Scriptes ausgeführt (siehe Param tokenize). Parameter: keine.

- **ready**

  Syntax: *Micropayment.ready();*

  Prüft ob die Iframes geladen wurden. Parameter: keine.

- **addEvent**

  Syntax: *Micropayment.addEvent(Eventname, Eventfunktion);*

  Fügt einen Eventhandler hinzu (siehe Events). Parameter:

  - Eventname
  - Eventfunktion

- **makeToken**

  Syntax: *Micropayment.makeToken();*

  Tokenisation ausführen. Siehe Parameter “form_submit”. Wenn “form_submit” den Wert “on-success” oder “auto” hat, wird nach der Tokenisation das Formular mit zusätzlichen “hidden” Feldern angereichert, nämlich token, pan und cvc. Wenn ein Prefix definiert wurde, erhalten die 3 Felder ebenfalls diesen Prefix. Die Werte von pan und cvc sind maskiert und dürfen gespeichert bzw. angezeigt werden. Parameter: keine.

- **validate**

  Syntax: *Micropayment.validate();*

  Validierung der Felder wird angestoßen (siehe Event validate). Parameter: keine.

- **complete**

  Syntax: *Micropayment.complete();*

  Prüfung des Feldes auf Vollständige Eingabe wird angestoßen (siehe Event complete/uncomplete). Parameter:

  - Feldname. Wenn leer werden alle Felder geprüft.

- **style**

Syntax: *Micropayment.style(Elementname, Value, Feldname);*

Styleanpassung von pan und cvc. Parameter:

- Elementname, Name des Styleelements
- Value, Value des Styleelements
- Feldname, pan oder cvc. Wenn leer werden alle Felder genommen.

- **reset**

  Syntax: *Micropayment.reset();*

  pan und cvc zurücksetzen. Parameter: keine

- **prepare** [*Billingfunktion*]

  Syntax: *Micropayment.prepare(token, url, target, alias);*

  Zahlvorgang mit token initialisieren. Parameter:

  - token
  - url, Rücksprungadresse, d.h. die Adresse auf die nach dem Zahlvorgang weitergeleitet wird. Wenn nicht gesetzt wird die Url aus dem Response der Notification nach der Zahlung genommen! Dazu controlcenter: Setup/Payment methods/Projectname//Card - Event ->API URL beachten.
  - target der Rücksprungadresse, wenn leer wird _parent angenommen. Achtung, die Adressierung bezieht sich auf das eingebundene IFrame, d.h._parent bedeutet, dass die Ausgabe an der Stelle des Formulars erfolgt.
  - alias, Parameteralias von cryptId. An die Rücksprungadresse wird die verschlüsselte Transaktionsnummer als cryptId gehängt. Die wird für die API-Funktion transactionGetCrypt() benötigt. Bei Namenskollisionen kann hier ein anderer Name für cryptId festgelegt werden.

- **amount** [*Billingfunktion*]

  Syntax: *Micropayment.amount(amount, currency, id, secret, methode);*

  Zahlvorgang mit amount/currency initialisiern. *Billingfunktion*. Parameter:

  - amount als Centbetrag, also bei 5,99 EUR würde hier 599 stehen, bei $6.77 würde 677 erwartet.
  - currency nach ISO 4217, also 3 stellig, z.B. EUR
  - time als Requesttime in der Form ‘Y-m-d-H:i:s’, also z.B. 2016-01-01-00:00:00
  - id als belibiege, optionale Vorgangsnummer. Wenn gesetzt entspricht er dem Parameter ‘title’ bei Zahlfenstern von Micropayment.
  - secret als hash über die Eingangsparameter und dem accesskey des Projectaccounts. Achtung! Aus Sicherheitsgründen darf das secret NICHT mit javascript erzeutgt werden.

    Bildung von secret:

1) accesskey: controlcenter: Setup/Overview ->AccessKey
2) Stringaddition der Parameter in der folgenden Reihenfolge mit dem Seperator ";" und dem Stringadditionsoperator "+". Die Parameterreihenfolge darf nicht geändert werden und über das Ergebnis wird ein md5 hash gebildet.
3) secretData = project + Seperator + amount + Seperator + currency + Seperator + id + Seperator + time + Seperator + accesskey
4) secret = md5(secretData)

in PHP ergibt sich:

$secret = md5($project . ';' . $amount . ';' . $currency . ';' . $id . ';' . $time . ';' . $accesskey);

- methode gibt die Art der Zahlung an. Mögliche Werte sind "Purchase" und "Authorization". Wenn leer wird "Purchase" angenommen, also die sofortige Zahlung. Wenn "Authorization" verwendet wird, müssen Sie später dann über die API-Funktion transactionCapture() die eigentliche Zahlung vornehmen.

- **session** [*Billingfunktion*]

  Syntax: *Micropayment.session(sessionId, url, target, alias, methode);*

  Zahlvorgang mit sessionId initialisieren, alternativ zu prepare/amount. Die sessionId muß über die API-Funktion tokenSessionCreate() erzeugt werden. Parameter:

  - sessionId
  - url, Rücksprungadresse, d.h. die Adresse auf die nach dem Zahlvorgang weitergeleitet wird. Wenn nicht gesetzt wird die Url aus dem Response der Notification nach der Zahlung genommen! Dazu controlcenter: Setup/Payment methods/Projectname//Card - Event ->API URL beachten.
  - target der Rücksprungadresse, wenn leer wird _parent angenommen. Achtung, die Adressierung bezieht sich auf das eingebundene IFrame, d.h. _parent bedeutet, dass die Ausgabe an der Stelle des Formulars erfolgt.
  - alias, Parameteralias von cryptId. An die Rücksprungadresse wird die verschlüsselte Transaktionsnummer als cryptId gehängt. Die wird für die API-Funktion transactionGetCrypt() benötigt. Bei Namenskollisionen kann hier ein anderer Name für cryptId festgelegt werden.
  - methode gibt die Art der Zahlung an. Mögliche Werte sind "Purchase" und "Authorization". Wenn leer wird "Purchase" angenommen, also die sofortige Zahlung. Wenn "Authorization" verwendet wird, müssen Sie später dann über die API-Funktion transactionCapture() die eigentliche Zahlung vornehmen.

- **process** [*Billingfunktion*]

  Syntax: *Micropayment.process();*

Zahlvorgang nach der Initialisierung auslösen. Im Ergebnis wird eine Notification (siehe controlcenter/API URL) vorgenommen und danach die Rücksprungadresse aufgerufen. Für ihren Zahleingang sollten Sie immer die Notification auswerten. Wenn die Rücksprungadresse übergeben wurde und nicht aus der Notification stammt, sollten Sie immer die Zahlung mit der API-Funktion transactionGetCrypt() prüfen. Parameter: keine

Events

- **ready**

  Scripts und Iframes wurden geladen. Initialisierungen, z.B. Styling kann vorgenommen werden. Parameter KEINE.

- **on**

  Sammelevent für Ereignisse bei pan/cvc. Parameter Eventobject Typ *OnEvent*.

- **type**

  Der Typ der Kreditkarte, auch wenn noch nicht komplett übermittelt. Parameter Typ *Brand*

- **complete**

  ein Feld wurde komplett mit Daten gefüllt. Parameter Typ *FieldName*.

- **uncomplete**

  wenn ein komplettes Feld nicht mehr vollständig ist. Parameter Typ *FieldName*.

- **error**

  bei einem Fehler. Parameter Eventobject Typ *Field*. Siehe auch Parameter “error” und “error_off”.

- **success**

  im Erfolgsfall. Parameter Eventobject Typ *Field*.

Event Parametertypen

Die Eventobjecte sind Jsoncodiert. Folgende Typen gibt es:

- Field
- OnEvent
- Brand
- FieldName

Event Parameter Feldbeschreibungen:

Field enthält die Werte

- **token** wenn Tokenisation erfolgreich war, steht hier der token
- **code** der Statuscode des Vorganges. mögliche Werte sind:
  - *ok* Verarbeitung erfolgreich
  - *fielderror* Die Felder enthalten Fehler. 'fields' untersuchen.
  - *syserror* Ein Verarbeiutngsfehler ist aufgetreten
- **fields** Liste der Felder. Für jedes Feld gibt es code und val, optional kann noch msg gesetzt sein.
  - code: Statuscode für das Feld. mögliche Werte sind:
    * *ok* Verarbeitung erfolgreich
    * *empty* Feld ist leer
    * *expire* Feldinhalt ist abgelaufen (year / month)
    * *luhn* Pan ist syntaktisch falsch
    * *bad* Feld hat ungültiges Format
    * *system* Feldprüfung undefiniert gescheitert
  - val: Der Wert des Feldes der angezeigt werden kann, bei pan und cvc sind diese Werte maskiert.
  - msg: zusätzliche Informationen

```
# Beispiel 1
{
  "token": "92ade7d1c46ec0a2dc1b505e40b2b6b3",
  "code": "ok",
  "fields": {
    "holder": {
      "code": "ok",
      "val": "Mustermann"
    },
    "month": {
      "code": "ok",
      "val": "01"
    },
    "year": {
      "code": "ok",
      "val": "2017"
    },
    "pan": {
      "code": "ok",
```

```
      "val": "411111******1111"
    },
    "cvc": {
      "code": "ok",
      "val": "***"
    }
  }
}

# Beispiel 2
{
  "token": "",
  "code": "fielderror",
  "fields": {
    "holder": {
      "code": "ok",
      "val": "Mustermann"
    },
    "month": {
      "code": "ok",
      "val": "01"
    },
    "year": {
      "code": "ok",
      "val": "2017"
    },
    "pan": {
      "code": "luhn",
      "val": "611111******1111"
    },
    "cvc": {
      "code": "ok",
      "val": "***"
    }
  }
}
```

OnEvent enthält die Werte

- **event** Name des aufgetretenen Events. Mögliche Werte sind:
  - focus
  - input

  - submit
  - blur
  - mouseover
  - mouseout
- **field** der Feldname pan oder cvc der das Event auslößt. Prefix wird nicht beachtet!

Brand mögliche Werte sind

visa, *master* und *amex*

FieldName mögliche Werte sind

holder, *year*, *month*, *pan* und *cvc*

Parameter

Dem JsBridge können einige Parameter bei der Einbindung mitgegeben werden. Beispiel:

```
<script src="//sipg.micropayment.de/public/bridge/v1/mp.js" project=demo></script>
```

- **project**

  Pflichtparameter, Projektkürzel von Micropayment

- **testmode**

  Default 0, aktiviert bzw. deaktiviert den Testmodus.

- **panformat**

  Default 1, Kreditkarte in 4er Blöcken formatieren

- **tokenize**

  Default 1, Iframe Tokenisation sofort initialisieren

- **error**

  Default “#ffa0a0”, Hintergrundfarbe von pan/cvc im Fehlerfall. Wenn leer, wird keine Error-markierung vorgenommen und somit wird auch “error_off” ignoriert.

- **error_off**

  Default “#fff”, Hintergrundfarbe von pan/cvc nach einem behobenen Fehlerfall

- **prefix**

  Default “”: Prefix der Feldenamen und FeldId‘s (holder, year, month, pan, cvc) sowie der FormularId des Händlerformulars. Durch Angabe eines Alias für ein Feld, wird ein vorhandener Prefix für dieses Feld überschrieben.

- **holder**

  Default “holder”: Alias für den gleichnamigen Feldnamen bzw. FeldId. Parameter Prefix wird hierfür ignoriert.

- **year**

  Default “year”: Alias für den gleichnamigen Feldnamen bzw. FeldId. Parameter Prefix wird hierfür ignoriert.

- **month**

  Default “month”: Alias für den gleichnamigen Feldnamen bzw. FeldId. Parameter Prefix wird hierfür ignoriert.

- **pan**

  Default “pan”: Alias für den gleichnamigen Feldnamen bzw. FeldId. Parameter Prefix wird hierfür ignoriert.

- **cvc**

  Default “cvc”: Alias für den gleichnamigen Feldnamen bzw. FeldId. Parameter Prefix wird hierfür ignoriert.

- **form**

  Default “token-form”: Alias für FormularId. Parameter Prefix wird hierfür ignoriert.

- **autobill**

  Default 0, automatisch nach der Tokenisation den Zahlvorgang starten

- **form_submit**

  Default “off”, mögliche Werte “onsuccess”, “auto”.

  - off

    Die Tokenisation muß manuell gestartet werden mit makeToken(). Im Fehlerfall wird Event “error” generiert, im Erfolgsfall “success”. Das Formular darf erst nach diesen Events abgeschickt werden.

  - onsuccess

    Das Händlerformular wird im Erfolgsfall abgeschickt, bei manuellem Aufruf von makeToken().

  - auto

    Das JsBridge hängt sich automatisch in onsubmit. Das Händlerformular kann also ganz normal mit einem “submit” Button abgeschickt werden.

Scenarien

Grundsätzlich

Das Händlerformular läuft mit SSL und die JsBridge von Micropayment wird eingebunden. Dem Script können Parameter übergeben werden, wobei lediglich der Parameter “project” Pflicht ist.

```
<script src="//sipg.micropayment.de/public/bridge/v1/mp.js"
  form_submit="auto"
  autobill=1
  prefix="card_"
  project="demo">
</script>
```

Es wird ein Formular mit der id “token-form” erwartet. Dieses Formular muss Inputfelder mit den IDs holder, year und month enthalten, wobei year und pan select-Felder sein können. Für pan und cvc müssen DIVs angelegt werden. In diese DIVs werden dann die Frames vom JsBridge geladen. Die Einbindung der pan würde z.B. so erfolgen:

```
<div id="card_pan"></div>
```

Sollte es Namenskonflikte geben, können diese Feldnamen bzw. IDs mit einem Prefix versehen werden (Parameter prefix).

Ein simples Formular würde dann so aussehen:

```
<form id="card_token-form"  action="echo.php"  method="POST">
    Name on Card:
    <input type="text" name="card_holder"
            id="card_holder" value="Mustermann" />
    <br>
    Credit Card Number: <div id="card_pan"></div><br>
    Expiration Date:
            <select name="card_month" id="card_month">
                <option>Month</option>
                <option value="01" selected>Jan (01)</option>
                <option value="02">Feb (02)</option>
                <option value="03">Mar (03)</option>
                <option value="04">Apr (04)</option>
                <option value="05">May (05)</option>
                <option value="06">June (06)</option>
                <option value="07">July (07)</option>
                <option value="08">Aug (08)</option>
                <option value="09">Sep (09)</option>
                <option value="10">Oct (10)</option>
```

```
                <option value="11">Nov (11)</option>
                <option value="12">Dec (12)</option>
            </select>
            <select name="card_year" id="card_year">
                <option>Year</option>
                <option value="15">2015</option>
                <option value="16">2016</option>
                <option value="17"  selected>2017</option>
                <option value="18">2018</option>
                <option value="19">2019</option>
                <option value="20">2020</option>
                <option value="21">2021</option>
                <option value="22">2022</option>
                <option value="23">2023</option>
            </select><br><br>
    Card CVV: <div id="card_cvc"></div>
    <br><br><br>
    <button type="submit" id="submit-button">Pay Now</button>
</form>
```

Die gewünschten Events müssen registriert werden und das Layout kann je nach Anforderung gestaltet werden. Gestaltung erfolgt mittels der Funktion “style” und sollte frühestens bei dem Event “ready” erfolgen. Ein Beispiel:

```
<script type="text/javascript">
    setupStyle = function() {
        Micropayment.style('border-style', 'none');
        Micropayment.style('border-color', '#cccccc');
        Micropayment.style('box-sizing', 'border-box');
        Micropayment.style('border-width', '0px');
        Micropayment.style('background-color', '#fff');
        Micropayment.style('padding', '6px 12px');
        Micropayment.style('font-size', '15px');
        Micropayment.style('font-weight', 'normal');
        Micropayment.style('color', '#333333');
        Micropayment.style('width', '99%'); // no calc support
    // mit calc support - override 99%
        Micropayment.style('width', 'calc(100% - 1px)');
        Micropayment.style('border-radius', '5px');
    }
    Micropayment.addEvent('ready', setupStyle);
</script>
```

Buchung in einem Schritt, ohne Zwischenseite

eine mögliche Lösung sieht so aus:

1. Formular wie unter “Grundsätzlich” beschrieben erstellen.
2. JsBridge Parameter
   - form_submit=“auto”
   - autobill=1
   - project=“demo”
3. Events registrieren
   - Event: ready

     Bsp.: Micropayment.addEvent(‘ready’, isReady);
4. Zahlung initialisieren.

   Initialisierung erfolgt mit den Funktionen amount() und prepare(), Bsp.:

```
isReady = function() {
    Micropayment.amount(
            "505", // amount
            "EUR", // currency
            "2016-02-22-16:15:31", // time
            "1456154131",   // id
            "8e8849263930b39cafbfcbf75cb338fa" // secret
    );
    // Rücksprungadresse als Bsp. "https://localhost/after_buy.php"
    // alias ist 'dieId'
    Micropayment.prepare('', "https://localhost/after_buy.php", '', 'dieId');
}
```

5. Wenn das Formular abgeschickt wird und alle Daten syntaktisch richtig waren, wird die Buchung ausgeführt und an die Rücksprungadresse weitergeleitet.

Buchung in 2 Schritten, mit Zwischenseite. Betrag/Währung wird versiegelt und per Javascript gesetzt

eine mögliche Lösung sieht so aus:

1. Formular wie unter “Grundsätzlich” beschrieben erstellen. Step A.
2. JsBridge Parameter
   - form_submit=“auto”
   - autobill=0

- project=“demo”

3. Events registrieren
   - Event: error, success

     Bsp.: Micropayment.addEvent(‘error’, onError); Micropayment.addEvent(‘success’, onSuccess);
4. Formular wird abgeschickt. Verarbeitung im Erfolgsfall durch Step B.
5. Step B Seite.

   POST Variablen von Step A entgegennehmen, u.a. Parameter token. Ausgabe der Zwischenseite (z.B. eine Übersichtsseite mit den Zahlungsdetails).
6. Wie bei Step A wird die JsBridge eingebunden. JsBridge Parameter:
   - tokenize=0
   - project=“demo”
7. Events registrieren
   - Event: error, success

     Bsp.: Micropayment.addEvent(‘error’, onError); Micropayment.addEvent(‘success’, onSuccess);
8. Initialisierung der Buchung erfolgt mit den Funktionen amount() und prepare(). Die eigentliche Buchung dann mit process(). Dazu wird z.B. eine Funktion billing() erstellt und dann bei Klick auf den “Kaufen” - Button ausgeführt. Bei einem Token mit dem Wert ‘1b62b706f4c620b0e43a652632b5b567’ ergibt sich z.B.:

```
billing = function() {
    Micropayment.amount("505",
            "EUR",
            "2016-04-04-15:07:28",
            "1459775248",
            "6fee5c1a270086eba69676c5dbb421a4");
    Micropayment.prepare(
            "1b62b706f4c620b0e43a652632b5b567",
            "https://localhost/workspace/iframedemo/after_buy.php",
            '',
            'dieId');
    Micropayment.process();
}
```

9. Die Buchung wird ausgeführt und an die Rücksprungadresse weitergeleitet.

Buchung in 2 Schritten, mit Zwischenseite. Betrag/Währung wird durch API-Aufruf gesetzt (tokenSessionCreate())

eine mögliche Lösung sieht so aus:

1. Formular wie unter “Grundsätzlich” beschrieben erstellen. Step A.
2. JsBridge Parameter
   - form_submit=“auto”
   - autobill=0
   - project=“demo”
3. Events registrieren
   - Event: error, success

     Bsp.: Micropayment.addEvent(‘error’, onError); Micropayment.addEvent(‘success’, onSuccess);
4. Formular wird abgeschickt. Verarbeitung im Erfolgsfall durch Step B.
5. Step B Seite.

   POST Variablen von Step A entgegennehmen, u.a. Parameter token. Ausgabe der Zwischenseite (z.B. eine Übersichtsseite mit den Zahlungsdetails).
6. Wie bei Step A wird die JsBridge eingebunden. JsBridge Parameter:
   - tokenize=0
   - project=“demo”
7. Events registrieren
   - Event: error, success

     Bsp.: Micropayment.addEvent(‘error’, onError); Micropayment.addEvent(‘success’, onSuccess);
8. Mit dem Token von Step A wird die API Funktion tokenSessionCreate() aufgerufen. Damit wird eine Session initialisiert und erzeugt. Hierbei kann vor allem der Buchungsbetrag und die Währung gesetzt werden. Die dabei erhaltene sessionId wird als Parameter bei der JsBridge Funktion session() erwartet. Für die Verwendung der API siehe Doku unter https://sipg.micropayment.de/public/creditcard/v1.6/nvp/ . Bsp. in PHP:

```
// MCP__CREDITCARDSERVICE_INTERFACE z.B. IMcpCreditcardService_v1_6
// MCP__CREDITCARDSERVICE_NVP_URL
//      z.B. https://sipg.micropayment.de/public/creditcard/v1.6/nvp/
// MCP__ACCESSKEY je nach account bzw. project

$dispatcher = new TNvpServiceDispatcher(MCP__CREDITCARDSERVICE_INTERFACE,
                                        MCP__CREDITCARDSERVICE_NVP_URL);
```

```php

  $ip = $_SERVER["REMOTE_ADDR"];

  $res = $dispatcher->tokenSessionCreate(MCP__ACCESSKEY, $testMode,
      $token,              //token
      'Mustermann',        //firstname
      $holder,             //surname
      null,                //doublet
      null,                //email
      null,                //culture
      null,                //customerId
      null,                //freeParams
      $project,            //project
      null,                //projectCampaign
      null,                //account
      null,                //webmasterCampaign
      $amount,             //amount
      $currency,           //currency
      'Die ganze Welt',    //title
      null,                //paytext
      $ip,                 //ip
      0,                   //sendMail
      null,                //freeParamsSession
      null,                //sessionId
      null                 //update
  );

  if(empty($res)) throw new Exception('Session invalid');
  $sessionId = $res['sessionId'];
  if(empty($sessionId)) throw new Exception('Session empty');
```

8. Initialisierung der Buchung erfolgt mit der Funktion session() (alternativ zu amount() und prepare()). Die eigentliche Buchung dann mit process(). Dazu wird z.B. eine Funktion billing() erstellt und dann bei Klick auf den “Kaufen” - Button ausgeführt. Bei einer sessionId mit dem Wert ‘CC9e1e63961de1baab8f845b41522dadade9e1db’ ergibt sich z.B.:

```php
    billing = function() {
        Micropayment.session("CC9e1e63961de1baab8f845b41522dadade9e1db",
                "https://localhost/workspace/iframedemo/after_buy.php");
        Micropayment.process();
    }
```

9. Die Buchung wird ausgeführt und an die Rücksprungadresse weitergeleitet.

Buchung in einem Schritt. Buchung durch API-Aufruf (tokenPurchase())

eine mögliche Lösung sieht so aus:

1. Formular wie unter “Grundsätzlich” beschrieben erstellen. Step A.
2. JsBridge Parameter
   - form_submit=“auto”
   - autobill=0
   - project=“demo”
3. Events registrieren
   - Event: error, success

     Bsp.: Micropayment.addEvent(‘error’, onError); Micropayment.addEvent(‘success’, onSuccess);
4. Formular wird abgeschickt. Verarbeitung im Erfolgsfall durch Step B.
5. Step B Seite.

   POST Variablen von Step A entgegebnehmen, u.a. Parameter token.
6. Mit dem Token von Step A wird die API Funktion tokenPurchase() aufgerufen. Damit wird direkt eine Buchung ausgeführt, allerdings sind z.Z. nur Buchungen ohne 3DSecure möglich. Für die Verwendung der API siehe Doku unter https://sipg.micropayment.de/public/creditcard/v1.6/ . Bsp. in PHP:

```php
// MCP__CREDITCARDSERVICE_INTERFACE z.B. IMcpCreditcardService_v1_6
// MCP__CREDITCARDSERVICE_NVP_URL
//      z.B. https://sipg.micropayment.de/public/creditcard/v1.6/nvp/
// MCP__ACCESSKEY je nach account bzw. project

$dispatcher = new TNvpServiceDispatcher(MCP__CREDITCARDSERVICE_INTERFACE,
                                        MCP__CREDITCARDSERVICE_NVP_URL);

$ip = $_SERVER["REMOTE_ADDR"];

$res = $dispatcher->tokenPurchase(MCP__ACCESSKEY, $testMode,
    $token,             //token
    null,               //firstname
    $holder,            //surname
    null,               //doublet
    null,               //email
    null,               //culture
    null,               //customerId
    null,               //freeParams
```

```
        $project,           //project
        null,               //projectCampaign
        null,               //account
        null,               //webmasterCampaign
        $amount,            //amount
        $currency,          //currency
        'Die ganze Welt',   //title
        null,               //paytext
        $ip,                //ip
        0,                  //sendMail
        null,               //freeParamsSession
        null,               //sessionId
        null,               //cvc2
        1,                  //fraudDetection
        0,                  //avsCheck
        null                //update
    )
    if(empty($res)) throw new Exception('Transaction invalid');

    if($res['transactionStatus'] == 'SUCCESS') {
        echo 'Buchung war erfolgreich.';
    } else {
        echo 'Buchung ist gescheitert.';
    }
```

Appendix

API-Funktion

damit sind Funktionen der Kreditkarten API gemeint. Tokenfunktionen sind ab Version 1.6 verfügbar https://sipg.micropayment.de/public/creditcard/v1.6/nvp/ .

Weiter Einzelheiten sind unter https://techdoc.micropayment.de/ verlinkt.

Tokenisation

Damit ist der Vorgang der Tokenerstellung gemeint. Der Token repräsentiert die erfaßten Kartendaten und ist eine bestimmte Zeit gültig (z.Z. 2 Stunden). In dieser Zeit müssen die Daten weiterverarbeitet werden. Die Verarbeitung des Token kann auf mehrere Arten erfolgen. Zum einen stehen die Funktionen der API zur Verfügung, vor allem die Funktionen mit dem Prefix “token”, z.B. tokenCustomerCreate, zum anderen stellt das JsBridge eine Möglichkeit zur direkten Buchung bereit (Billingfunktionen). Momentan ist eine 3DSecurebuchung nur über die Billingfunktionen des JsBridgees möglich.

3DSecure

3-D Secure ist ein Verfahren, das für zusätzliche Sicherheit bei Online-Kreditkartentransaktionen eingesetzt wird. Es steht stellvertretend für die dementsprechenden Dienste von VISA und MASTER.

Händlerformular

NICHT gehostest bei Micropayment. Pan und cvc werden hier NICHT erfaßt, allerdings gibt es hier die Inputfelder für holder, year und month. Hier wird das JsBridge eingebunden. Parameter ist das Micropayment project. Das Formular kann normal gestaltet werden allerdings setzt man an die gewünschten Stellen von pan und cvc entsprechend gestaltete DIV‘s die dann zur Laufzeit mit den IFrames und den dort enthaltenen input-Feldern gefüllt werden (vom JsBridge). Das JsBridge stellt diverse Steuerungsfunktionen bereit.

IFramelösung

siehe JsBridge

JsBridge

Script und IFrame gehostet bei Micropayment https://sipg.micropayment.de/public/bridge/v1/mp.js Mindestens pan und cvc werden hiermit erfaßt, wobei für jedes Feld ein eigenes IFrame generiert wird. Das Javascript stellt beim Einbinden das Object “Micropayment” bereit. Dies stellt zur IFramesteuerung einige Funktionen, Events und Parameter bereit.

Rücksprungadresse

Das ist die Adresse auf die nach dem Zahlvorgang mit der Billingfunktion “process” weitergeleitet wird.

cryptId

An die Rücksprungadresse wird die verschlüsselte Transaktionsnummer als GET-Parameter cryptId gehängt. Die wird für die API-Funktion transactionGetCrypt() benötigt. Die erste Stelle der cryptId gibt den Testmodus an. Wenn cryptId mit 1 beginnt wurde cryptId im Testmodus erzeugt, bei 0 war der Testmodus aus. Diese Stelle der cryptId können Sie auswerten, wenn Sie in der Rücksprungadresse den Testmodus nicht extra angeben wollen. Für die API-Funktion transactionGetCrypt() wird der richtige Wert für Testmodus erwartet.

Account

Anmeldung bzw. Registrierung bei Micropayment erforderlich. siehe controlcenter.

controlcenter

Verwaltungsprogramm wenn Sie sich unter https://www.micropayment.de/ einloggen mit ihrem Account.

accesskey

accesskey des Projectaccounts. Sie finden ihn unter: controlcenter: Setup/Overview ->AccessKey Achtung! Aus Sicherheitsgründen darf der accesskey nicht mit javascript verwendet werden sondern nur auf Serverseite.

Testmodus

Wenn aktiviert, wird die Tokenisation und eine eventuelle Buchung mit dem Token nur simuliert, also keine reale Buchung ausgelößt. Die resultierende Transaktion erscheint nicht in der Statistik. Bei der Notification ist testmode ebenfalls entsprechend gesetzt. Für das Project muß im controlcenter der Testmodus aktiviert sein.

Project

Pflichtparameter. Muss über das controlcenter für die Zahlart “CreditCard - Event” eingerichtet werden. Der Testmodus kann sofort benutzt werden, muss aber aktiviert werden. Bei “API URL” sollte ein Script hinterlegt sein, dass die Notifications entgegennimmt und verarbeitet.

Notification

Direkt nach der Zahlung wird von Micropayment die “API URL” aufgerufen, siehe Project.

Customer

Datensatz der den Kunden representiert. Neben Kreditkartendaten kann er noch Informationen wie email oder Anschrift haben.

customerId

Mit der customerId wird der Customer referenziert und damit indirekt die Kreditkartendaten.

Purchase

Sorfortige Buchung bei Zahlung mit Kreditkarte.

Authorization

Ein bestimmter Betrag wird reserviert, d.h. der Kreditrahmen des Kunden wird um diesen Betrag reduziert. Der reservierte oder ein kleinerer Betrag kann dann mit Capture zu einem späteren Zeitpunkt gebucht werden.

Capture

Ein (vor)autorisierter Betrag, siehe Authorization, wird gebucht.

pan

die Kreditkartennummer.

cvc

steht für Card Validation Code und tritt auch noch unter anderen Bezeichnungen auf. Gemeint ist die auf der Kreditkarte aufgedruckte Prüfnummer.

holder

steht für credit card holder, also der Kreditkarteninhaber

Session

Datensatz der den Zahlvorgang beschreibt, enthält Informationen über den zu buchenden Betrag, Währung und sonstiger Informationen nebst einer Refernz auf den Customer. Einer Transaktion können mehrere Transaktonen je nach Art zugeordnet sein, z.B. Authorization, Capture und dann ein Refund. Referenziert über sessionId.

Transaction

beschreibt den konkreten Vorgang oder die Vorgänge einer Session, also Purchase, Authorization, Capture etc.. Je Vorgang wird ein Transaktonssatz erzeugt. Referenziert über transactionId.

transactionId

siehe Transaction.

sessionId

siehe Session.

title

bei allen Zahlfenstern von Micropayment kann mit diesem Parameter ein eindeutiger Bezeichner für den Vorgang mitgegeben werden. Der Wert erscheint in der Statistik, wird für die Sesessionsuche indiziert und wird bei der Notification nach der Zahlung mitgeschickt.