secupay

Flex 2.0 API Payment Schnittstelle

Version 2.3.16

15. April 2015

Inhaltsverzeichnis

1 Allgemeines **3**
1.1 Das einfach.sicher.zahlen-System . . . 3
1.2 Demoanfragen . . . 3

2 Systembeschreibung **4**
2.1 REST . . . 4
2.2 Pfade . . . 4
2.3 HTTP-Header . . . 4
2.3.1 Content-Type und Accept . . . 5
2.4 language . . . 5
2.5 Zeichensatz . . . 6
2.6 Datenstruktur . . . 6
2.6.1 Antwort: Status . . . 6
2.6.2 Antwort: Data . . . 7
2.6.3 Antwort: Errors . . . 7

3 Payment **8**
3.1 Liste der verfügbaren Zahlungsarten . . . 8
3.2 Init/Einreichung einer Transaktion . . . 9
3.3 Apikey Erzeugung per API . . . 10
3.3.1 Spezifische Parameter . . . 11
3.4 Statusabfrage . . . 12
3.5 Capture vorautorisierter Zahlungen . . . 13
3.6 Storno vorautorisierter/nicht final eingereichter Zahlungen . . . 13
3.7 Invoice (Kauf auf Rechnung) . . . 14
3.8 Wiederkehrende Zahlungen (Abo) . . . 16
3.8.1 Erstellen einer Subscription durch einen gegebenen Hash . . . 16
3.8.2 Verwendung der Subscription-Id . . . 17
3.8.3 Besonderheiten bei Wiederkehrenden Rechnungskauf Transaktionen 18

4 Push API **18**

5 Parameter **20**
5.1 Besonderheiten bei einzelnen Parametern . . . 21
5.2 Basket . . . 21
5.3 Userfields . . . 22
5.4 Lieferadresse . . . 22

6 Fehlercodes **22**

1 Allgemeines

Dieses Handbuch beschreibt die flex.API v2.3 zur Onlineabwicklung von Zahlungen.

1.1 Das einfach.sicher.zahlen-System

Das einfach.sicher.zahlen-System von secupay ermöglicht die Onlineabwicklung von Zahlungsvorgängen in Internetshops und auf Internetportalen.

Bei fehlerhaften Daten oder unzureichender Bonität des Endkunden erfolgt eine Ablehnung mit entsprechender Fehlermeldung. Der Kunde sollte dann auf die Seite mit der Auswahl der Bezahlsysteme zurück geleitet werden.

Zur erfolgreichen Durchführung einer Demo-Transaktion benötigt Ihr Vertrag die entsprechenden Konditionen. Sollte die Anfrage mit dem Fehlercode 0007 abgelehnt werden, melden Sie sich bitte beim Kundendienst für die Freischaltung.

1.2 Demoanfragen

Um Anfragen als Demo zu kennzeichnen und damit eine reale Buchung der Transaktion zu verhindern, ist bei den payment Requests der Parameter “demo” vorgesehen. Wird der Parameter mit dem Wert “1” oder “true” als String oder boolean übermittelt, wird die Transaktion simuliert und es entstehen weder Transaktionskosten noch reale Buchungen.

2 Systembeschreibung

2.1 REST

Die secupay Flex 2.0 API, nachfolgend flex.API genannt, ist im Groben als REST (Representational State Transfer) Interface umgesetzt und lässt sich deshalb besonders leicht mit einfachen https-Anfragen ansprechen.

Der Hostname der API ist api.secupay.ag, das Protokoll https://, die Basis URL für alle Anfragen lautet deshalb:

```
https://api.secupay.ag/
```

Aus Sicherheitsgründen sind Anfragen auf Port 80 nicht verfügbar und werden serverseitig auf Port 443/HTTPS umgeschrieben.

Für initiale Entwicklungen empfehlen wir, anstatt des Live Systems, das Dist-System zu verwenden. Dieses ist unter

```
https://api-dist.secupay-ag.de
```

erreichbar. Der Vorteil hierbei liegt darin, dass etwaige Fehler bei der Entwicklung keine Auswirkungen auf Ihren Live Vertrag haben. Transaktionen werden hierbei letztlich nicht gebucht und es fallen keinerlei Anfragekosten an.

Einen Zugang zum Dist-System erhalten Sie auf Anfrage von unserem Kundendienst.

2.2 Pfade

Die Funktionen der API sind über den anfragenden Pfad zu erreichen. Diese sind in verschiedene Namensräume unterteilt. Die Zahlfunktionen sind dabei unter /payment/<FUNKTION> zu erreichen.

Beispiele:

Namensraum: payment, Funktion: init

```
https://api.secupay.ag/payment/init
```

Namensraum: payment, Funktion: gettypes

```
https://api.secupay.ag/payment/gettypes
```

2.3 HTTP-Header

Anfragen an die API benötigen HTTP Header um verschiedene Ein- und Ausgabeformate zu unterstützen.

Des Weiteren ist es möglich die Sprache der gemeldeten Fehler zu steuern.

2.3.1 Content-Type und Accept

Die Schnittstelle unterstützt zwei Ausgabeformate – JSON und XML. Innerhalb des secupay-Systems wird mit JSON gearbeitet und falls gewünscht in XML umgewandelt.

HTTP-Header Content-Type Mit dem HTTP-Header Content-Type teilt der Client der flex.API mit, in welchem Format die Daten im HTTP-Body (PUT- oder POST-Methode) gesendet werden

Daten werden im XML-Format gesendet

Content-Type: text/xml; charset=utf-8;

Daten werden in JSON gesendet

Content-Type: application/json; charset=utf-8;

Die Angabe charset=utf-8; hat keine Bedeutung, da die flex.API immer UTF-8 verwendet. Die Angabe „charset“ wird von der flex.API automatisch in utf-8 geändert.

HTTP-Header Accept Mit dem HTTP-Header Accept bestimmt der Client in welchen Format die flex.API antworten soll.

Beispiel für das Empfangen von XML Ergebnissen:

```
Accept: text/xml;
```

Daten werden in JSON empfangen:

```
Accept: application/json;
```

Beim HTTP-Header Accept ist es üblich mehrere Datenformate anzugeben. Die flex.API wertet aber nur den ersten Wert aus. Es sollte also davon abgesehen werden mehr als ein Format im Accept-Header zu senden.

Die HTTP-Header Accept und Content-Type können beliebig gemischt werden. Es ist also möglich eine POST Anfrage in JSON zu senden und die Antwort im XML-Format zu erhalten.

2.4 language

Die flex.API unterstützt mehrere Sprachen. So gibt es beispielsweise, bei falsch übermittelten Daten an die API, Fehlercodes die vom Client-Programm abgefangen werden können. Zum Anderen gibt es aber auch qualifizierte Fehlermeldungen, die dem Benutzer direkt in der entsprechenden Oberfläche angezeigt werden sollen. Ein typischer Validierungsfehler ist eine leere Eingabe bei einem Pflichtfeld.

Derzeit gültige Werte:

```
de_DE
```

sowie

```
en_US
```

2.5 Zeichensatz

Die flex.API verwendet als Zeichensatz ausschließlich UTF-8, sowohl für eingehende als auch für ausgehende Daten. Eine Überprüfung der eingehenden Daten hinsichtlich des Zeichensatzes findet nicht statt, das heißt der Client ist selbst für eine eventuell notwendige Konvertierung verantwortlich.

2.6 Datenstruktur

Unabhängig von dem angefragten Namensraum, sowie vom Format sind alle eingehenden und ausgehenden Daten einheitlich strukturiert. Alle Anfragen mit HTTP-Body kapseln die übertragenen Nutzdaten in data. Alle Antworten der flex.API sind in drei Bereiche untergliedert: status, data und errors.

Nachfolgend ein Beispiel zur Verdeutlichung:

```
{
    "status": "ok",
    "data": {
        "hash": "...",
        "iframe_url": "..."
    },
    "errors": null
}
```

2.6.1 Antwort: Status

Der Status zeigt an, ob die Anfrage erfolgreich war. Hierbei gibt es drei unterschiedliche Status:

ok - Die Anfrage an die flex.API war erfolgreich und die gewünschte Aktion wurde durchgeführt.

error - Die gewünschte Aktion konnte nicht durchgeführt werden, die Ursache ist technischer Natur. Solche Fehler deuten auf Bugs der flex.API oder des Clients hin. In diesem Fall ist der Eingriff eines Entwicklers erforderlich.

failed - Die gewünschte Aktion konnte nicht durchgeführt werden. Meist handelt es sich um Validierungsfehler, deshalb sollte dem Benutzer das Formular mit den Fehlermeldungen angezeigt werden.

2.6.2 Antwort: Data

Der Bereich data enthält die eigentlichen Nutzdaten. Die Struktur der Nutzdaten ist abhängig von dem Namensraum sowie der Funktion. Meist handelt es sich um eine einfache Name-Wert-Zuordnung oder Listen von Name-Wert-Zuordnungen.

Beachten: Die hier zurückgegebenen Daten können zukünftig um zusätzliche Wertpaare erweitert werden. Dies darf bei der Verarbeitung nicht zu Fehlern führen.

2.6.3 Antwort: Errors

Der Bereich wird nur beim Status „Failed“ oder „Error“ gesetzt. Dann enthält er eine Liste mit Fehlercodes und Fehlermeldungen.

3 Payment

3.1 Liste der verfügbaren Zahlungsarten

```
https://api.secupay.ag/payment/gettypes
```

Dieser Aufruf gibt eine Liste aller möglichen Zahlungsarten zurück. Diese Liste ist abhängig vom apkiey und kann sich dynamisch ändern. Die Zahlarten sind bei jedem Kunden der secupay in den entsprechenden Verträgen hinterlegt.

Es empfiehlt sich, diese Abfrage zur Auflistung der verfügbaren Zahlungsarten auszuführen, damit es nicht zu ungewollten Fehlermeldungen kommt. Wenn die entsprechende Zahlungsart nicht verfügbar sein sollte. Die Abfrage gibt die jeweils verfügbaren Zahlungsarten in Echtzeit wieder.

Anfrage

```
{
    "data": {
        "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace"
    }
}
```

Antwort

```
{
    "status": "ok",
    "data": [
        "creditcard",
        "debit"
    ],
    "errors": null
}
```

3.2 Init/Einreichung einer Transaktion

```
https://api.secupay.ag/payment/init
```

Zu Beginn muss der Zahlvorgang dem secupay-System bekannt gemacht werden. Dabei werden die Informationen über die Zahlung vom Shop-System übertragen. secupay initialisiert damit den Zahlvorgang, legt die internen Transaktionsdaten an und erzeugt den Hash, der bei der weiteren Kommunikation verwendet wird.

Die übertragenen Informationen können je nach Zahlungsart und Shop-System unterschiedlich sein. Deshalb sind viele Angaben optional oder werden in manchen Fällen nicht berücksichtigt. Damit der Zahlvorgang initialisiert werden kann, sind daher vor allem die technischen Parameter wichtig.

- apikey - Der apikey dient zur Anmeldung am secupay-System
- amount - Der Betrag in der kleinsten Einheit der Währung (Eurocent)
- payment_type - Dieser Wert gibt die Zahlart des Vorgangs an. Typische Werte sind debit oder creditcard
- url_success - Nach erfolgreicher Transaktion wird der Kunde auf diese Seite weitergeleitet
- url_failure - Nach Abbruch bzw. aufgetretenen Fehlern wird der Kunde auf diese Seite weitergeleitet
- url_push - Auf diese wird bei wesentliche Statusänderungen an der Transaktionen der neue Status per HTTP POST übermittelt (siehe Punkt 4 Push API)

Die Zahlung wird mit den in Abschnitt 5 beschriebenen Parametern eingereicht. Die flex.API prüft die Daten auf Vollständigkeit. Wurden alle für die Transaktion benötigten Daten übergeben, liefert die flex.API den hash und den iFrame link der Transaktion zurück.

Zahlungsmitteldaten werden in der Regel von secupay in einem separat zur Verfügung gestellten iFrame abgefragt. Dazu antwortet die flex.API mit dem Hash und dem Link zum Öffnen des iFrames zur Eingabe. Im Folgenden wird zunächst von secupay der Genehmigungsprozess durchgeführt. Danach liefert die flex.API den Hash und den Status der Transaktion zurück.

Der Parameter “labels” ist optional. Wird er nicht angegeben werden die Standard Texte von uns verwendet.

```
{
    "data": {
      "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
      "demo": "0",
      "payment_type": "creditcard",
      "payment_action": "sale",
      "apiversion": "2.11",
      "amount": 100,
```

```
      "url_success": "https://shop.example.com/success",
      "url_failure": "https://shop.example.com/failed",
      "url_push": "https://shop.example.com/push.php",
      "labels": {
        "en_US": {
          "basket_title": "Your Order",
          "submit_button_title": "Submit",
          "cancel_button_title": "Return to Basket"
        },
        "de_DE": {
          "basket_title": "Ihre Bestellung",
          "submit_button_title": "Daten Übermitteln",
          "cancel_button_title": "Zum Warenkorb"
        }
     }
  }
}
```

Antwort

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303"
    },
    "errors": null
}
```

Über die zurückgegebene URL lässt sich nun die Zahlung vom Kunden beenden, es steht Ihnen offen die URL entweder als iFrame einzubetten oder per Weiterleitung aufzurufen. Die Eingabemasken sind grundsätzlich so aufgebaut, dass sie sich entsprechend des Platzangebots von selbst in der Darstellung anpassen.

Der Parameter “payment_action” steuert die Verarbeitung der Transaktion durch uns, vorerst gibt es hier die Werte “sale” und “authorization”. Sale ist dabei eine direkte Zahlung. Um die Transaktion erst später durchzuführen muss hier “authorization” übermittelt werden.

3.3 Apikey Erzeugung per API

Diese Funktion bietet eine Möglichkeit dynamisch einen APIKey zu erzeugen. Für den Normalen Shop-Betrieb sollte diese Funktion nicht notwendig sein und ist auch nur für explizit freigeschaltete APIKeys gültig.

```
https://api.secupay.ag/payment/requestapikey
```

```
{
    "data": {
        "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
        "user": {
            "title": "Herr",
            "company": "Firma",
            "firstname": "Test FN",
            "lastname": "Test LN",
            "street": "Test Street",
            "housenumber": "5t",
            "zip": "12345",
            "city": "TestCity",
            "telephone": "+4912342134123",
            "dob_value": "01.02.1903",
            "email": "test@ema.il"
        },
        "payout_account": {
            "accountowner": "Test Inhaber",
            "iban": "DE00012345678912345678",
            "bic": "COBADEFF103"
        },
        "project": {
            "name": "APITest Shop"
        },
        "iframe_opts": {
            "show_basket": true,
            "basket_title": "Ihr Warenkorb",
            "submit_button_title": "Kostenpflichtig Bestellen!"
        }
    }
}
```

3.3.1 Spezifische Parameter

Bis auf iframe_opts sind alle Parameter Pflichtangaben.

- user - Vertragsdaten des Projektinhaber
- payout_account - Konto auf welches die Transaktionen ausgezahlt werden sollen
- project - Projekt spezifische Eigenschaften, vorerst nur der Name des Projekts
- iframe_opts - definiert eine Liste von Standards welche die Darstellung des iFrames beeinflussen
  - show_basket - Soll im Iframe ein Basket angezeigt werden, true/false

- basket_title - Der Titel des Warenkorbes
- submit_button_title - Der Text des Absenden Buttons
- logo_base64 - Das zu hinterlegende Logo für das iFrame als base64 Darstellung
- cession - Ansprache im iFrame, 'personal' oder 'formal'

logo_base64 sowie cession werden falls nicht explizit übermittelt aus dem vorhandenen Vertrag übernommen.

Antwort

```
{
    "status": "ok",
    "data": {
        "apikey": "48411xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx744e",
        "contract_id": "1234"
    },
    "errors": null
}
```

3.4 Statusabfrage

```
https://api.secupay.ag/payment/status
```

Über diese Funktion kann der Status und weitere Informationen zu einem Hash abgefragt werden.

In Zukunft kann die Antwort um weitere Informationen erweitert werden.

```
{
  "data": {
    "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
    "hash":"tujevzgobryk3303"
  }
}
```

Antwort

```
{
  "status": "ok",
  "data": {
    "hash": "tujevzgobryk3303",
    "payment_status": "acccepted",
    "status": "accepted",
    "created": "2013-06-10 10:27:42",
    "demo": 0,
```

```
    "trans_id": "1831201",
    "amount": 100,
    "opt": {

    }
  },
  "errors": null
}
```

Der im data Teil der Antwort enthaltene status muss nach der Weiterleitung auf url_success auf accepted stehen. Eine Ausnahme kann bei vorautorisierten Transaktionen bestehen.

Es wird empfohlen, nachdem die Weiterleitung auf url_success erfolgt ist, durch eine Statusabfrage den Status “accepted” zu verifizieren.

Im Property “opt” werden Payment-Type spezifische Rückantworten übermittelt. Zum Beispiel: Bei Invoice werden hier die Kontodaten für die Überweisung der Rechnung übermittelt, um dem Kunden z.B. in einer E-Mail darüber zu informieren. Vorerst wird das Property nur bei Rechnungskauftransaktionen zurückgeliefert, in der Zukunft macht dies aber auch bei anderen Zahlarten Sinn. Neue Implementierungen sollten diese Property daher grundsätzlich beachten.

3.5 Capture vorautorisierter Zahlungen

Vorautorisierte Zahlungen müssen, um die Zahlung durchzuführen, gecleared werden, dies kann auf 2 Wegen erfolgen, entweder per POST auf

```
https://api.secupay.ag/payment/<hash>/capture
```

oder per GET auf

```
https://api.secupay.ag/payment/<hash>/capture/<apikey>
```

Mittels der GET Variante erhält der Anwender im Browser eine Übersicht über die Transaktion und kann per Button die Zahlung durchführen.

In der POST Version muss wie gewohnt der Apikey im data array übermittelt werden.

3.6 Storno vorautorisierter/nicht final eingereichter Zahlungen

Auch beim Storno sind 2 Wege vorgesehen, einmal mit Interaktion des Anwenders per GET, sowie eine reine Server zu Server Übermittlung. Storniert werden können vorerst nur Transaktionen welche vorautorisiert sind oder noch nicht final eingereicht wurden (Bei Lastschriften ist dies in der Regel der nachfolgende Tag, etwa 7 Uhr).

POST:

```
https://api.secupay.ag/payment/<hash>/cancel
```

GET:

```
https://api.secupay.ag/payment/<hash>/cancel/<apikey>
```

In der POST Variante wird bei Erfolg/Misserfolg entsprechend status failed/ok zurückgeliefert, bei einem Fehler inklusive entsprechender Fehlermeldung im error array.

3.7 Invoice (Kauf auf Rechnung)

Bei der Zahlart invoice/Kauf auf Rechnung gibt es zusätzlich zu den normalen API-Funktionen noch die Möglichkeit, das Versanddatum zu setzen.

Capture (Versanddatum der Rechnung Setzen)

Um das Versanddatum einer Rechnung zu setzen, muss folgende URL aufgerufen werden:

```
https://api.secupay.ag/payment/{hash}/capture/{apikey}
```

Die Werte für den hash sowie des Apikey des Händlers sind entsprechend zu ersetzen.

Beispiel einer URL:

```
https://api.secupay.ag/payment/oxaijkeaucbl2041/capture/6801
    fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace
```

Alternativ ist auch ein Setzen des Versanddatums per POST ohne eine direkte Nutzerinteraktion möglich, in dieser Variante wird der Zeitpunkt der Anfrage als Versanddatum verwendet.

Hierzu muss auf folgende URL eine Post Anfrage gesendet werden:

```
https://api.secupay.ag/payment/{hash}/capture/
```

Im Data Objekt der Anfrage muss zusätzlich noch der Parameter Apikey übermittelt werden:

```
{
  "data": {
    "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
    "tracking": {
          "provider": "DHL",
          "number": "TC123456789"
    },
    "invoice_number": "RN 0001"
  }
}
```

Die Parameter “tracking” (Versanddienstleister und Tracking-Nummer) und “invoice_number” (Rechnungsnummer) sind optional. Die hier angegebene “invoice_number” ist bei einer späteren Statusanfrage in der Antwort enthalten.

Zusätzliche Rückgabeparameter bei einer Statusanfrage

Bei einer Statusanfrage auf Invoice Transaktionen bekommen Sie zusätzliche Felder im opt Property zurückgeliefert:

Antwort


```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "status": "accepted",
        "payment_status": "accepted",
        "created": "2013-06-10 10:27:42",
        "demo": 0,
        "trans_id": "1831201",
        "amount": 100,
        "opt": {
            "recipient_legal": "secupay AG, Goethestraße 6, 01896
               Pulsnitz",
            "payment_link": "https://api.secupay.ag/payment/
               tujevzgobryk3303",
            "payment_qr_image_url": "https://api.secupay.ag/qr?d=http
               %3A%2F%2F....",
            "transfer_payment_data": {
                "purpose": "TA 123456 DT 20130610",
                "accountowner": "Secupay AG",
                "iban": "DE00012345678912345678",
                "bic": "COBADEFF103",
                "accountnumer": "0123456789",
                "bankcode": "01234567",
                "bankname": "Test Bank"
            },
      "invoice_number": "RN 0001"
        }
    },
    "errors": null
}
```


Zahlung der Rechnung durch den Endkunden

Um die Rechnung zu begleichen hat der Kunde mehrere Möglichkeiten. Diese werden ihm in einem iFrame angeboten. Die URL hierfür ist die selbe wie die zurückgegebene iFrame url in

init. Diese url kann direkt per QR Code oder per Link in E-Mails bzw Rechnungs-PDFs integriert werden, um dem Kunden eine einfache Möglichkeit zum Begleichen der Rechnung zu bieten.

3.8 Wiederkehrende Zahlungen (Abo)

Um wiederkehrende Zahlungen durchführen zu können müssen Sie zuerst Ihren Vertrag dafür freischalten lassen. Anschließend gibt es 2 Methoden um eine initiale ABO Zahlung durchzuführen. Die bevorzugte Möglichkeit ist das Anfragen auf /payment/getSubscription für eine bereits erfolgreiche Zahlung. Alternativ können Sie auch im init der 1. ABO Transaktion ein Subobjekt "subscription" übermitteln, woraufhin Sie in der Antwort den Parameter "subscription_id" zurück bekommen. Mit diesem werden die Folgetransaktionen ausgelöst. Wichtig ist hierbei nur, dass es das Objekt gibt, falls kein ABO Verwendungszweck gewünscht ist also ein leeres Subobjekt "subscription".

Die Subscription-Id kann verwendet werden, um eine neue Zahlung unter Verwendung der vom Zahler eingegebenen Zahlungsmitteldaten zu erzeugen.

Die Interaktion zwischen Ihnen und secupay läuft hierbei ohne Interaktion durch den Endkunden ab.

Init Request:

```
{
  ...
  "data": {
    "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
    "subscription": {
      "purpose": "ABO Monatlich bei ..."
    },
    "amount": 2350,
    "purpose": "Ihre Bestellung bei ..."
  }
}
```

3.8.1 Erstellen einer Subscription durch einen gegebenen Hash

Falls Sie nachträglich zu einer bereits getätigten Zahlung ein Abo erzeugen möchten, ist dies mit der action getSubscription möglich. Hierbei wird per POST auf /payment/getSubscription der zu verwendende Apikey sowie der Hash der Ursprungstransaktion übermittelt.

Als Antwort erhalten Sie daraufhin eine Abo-Id mit welcher nun Abo Zahlungen durchgeführt werden können.

POST

```
https://api.secupay.ag/payment/getSubscription
```

```
{
  "data": {
    "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
    "hash": "tglswdmhziwb84171",
    "subscription": {
      "purpose": "ABO Monatlich bei ..."
    }
  }
}
```

Antwort

```
{
    "status": "ok",
    "data": {
        "subscription_id": 1234
    },
    "errors": null
}
```

3.8.2 Verwendung der Subscription-Id

Verwendungszweck Den Verwendungszweck von Abo Transaktionen können Sie an mehreren Stellen definieren, hierbei gilt folgende Rangabfolge:

1. purpose im POST auf payment/subscription
2. purpose im Unterobjekt “subscription” bei payment[init|getSubscription]
3. purpose der Ursprungstransaktion

Um das Rücklastschriftrisiko zu minimieren, achten Sie bitte auf einen sinnvollen Verwendungszweck den der Kunde eindeutig zuordnen kann.

Abo-Zahlung ausführen

Um aus der aus /payment/init zurückgelieferten Subscription-Id eine Abo Zahlung zu erzeugen, ist die nachfolgende POST Anfrage auf /payment/subscription auszuführen.

POST

```
https://api.secupay.ag/payment/subscription
```

Im data Objekt der Anfrage müssen zusätzlich noch die Parameter “amount”, “subscription_id” sowie “apikey” übermittelt werden:

```
{
  "data": {
    "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
    "subscription_id": 1234,
    "amount": 2350,
    "purpose": "Ihre Bestellung bei ..."
  }
}
```

Antwort

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303"
    },
    "errors": null
}
```

3.8.3 Besonderheiten bei Wiederkehrenden Rechnungskauf Transaktionen

Bei einer wiederkehrenden Rechnungskauf Transaktion sollte dem Kunden bei Erstellung der neuen Zahlung eine Mitteilung gesendet werden mit der Angabe der Überweisungsdaten, welche als Antwort auf die Transaktionserstellung zurückgeliefert werden.

4 Push API

Aus Sicherheitsgründen bietet secupay die Möglichkeit jede über die API abgesetzte Transaktion zusätzlich über eine reine Server zu Server Kommunikation zu bestätigen.

Dabei wird beim Einreichen der Transaktion der Parameter url_push übermittelt. Auf diesen wird bei Statusänderungen an der Transaktionen der neue Status per HTTP POST übermittelt.

Dies ermöglicht z.B. den Einsatz einer eigenen WaWi welche automatisch den aktuellen Status des Zahlvorgangs erhält und damit die Abarbeitung und den Versand der Pakete vereinfacht und sicherer gestaltet.

Hierbei wird bei Umstellung des Transaktionsstatus im secupay System eine HTTP(S) POST Anfrage auf die übermittelte Push url erstellt:

Url:

```
https://push.example.net/push_client.php
```

```
POST /push_client.php HTTP/1.1

hash=jtnjpfgrbrqk3300&amount=1199&status_id=6&status_description=
    abgeschlossen&changed=1365444092&payment_status=accepted&apikey
    =6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&hint=
```

- hash: den betroffenen Hash
- amount: Gesamtbetrag der Transaktion
- status_id: die detaillierte secupay Status-Id
- status_description: der detaillierte secupay Status-Text
- changed: UTC Unix Timestamp wann die besagte Statusänderung eingetreten ist
- apikey: der bei der Transaktion verwendete Apikey, dieser sollte vor Annahme des Push's abgeglichen werden
- hint: (todo) Hinweistext, vorgesehen um z.B. im RLS Fall zusätzliche Informationen beizulegen
- payment_status
  - Vereinfachte Form des Secupay Status, dies vereinfacht die Einordnung in Kategorien, möglich sind hierbei
    * accepted - Transaktion wurde erfolgreich durchgeführt
    * authorized - Transaktion ist autorisiert
    * denied - Transaktion wurde abgelehnt
    * issue - Es gibt ein Problem mit der Transaktion, z.B. Rücklastschrift Fall
    * void - Die Transaktion wurde storniert oder gutgeschrieben
- (Zusatz bei Abo Zahlungen)
  - subscription_id - Die für die Abo Zahlung verwendete Id

Die Push Notification muss von der empfangenden Seite nun noch quittiert werden. Dazu muss sie als einfachen Text mit dem kompletten Anfrage-Text inklusive ack=Approved antworten.

Für das oben genannte Beispiel erwartet die API dabei folgende Antwort:

```
ack=Approved&hash=jtnjpfgrbrqk3300&status_id=6&
    status_description=abgeschlossen&changed=1365444092&
    payment_status=accepted&apikey=6801
    fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&hint=
```

Können Sie den Hash auf Ihrer Seite nicht zuordnen, sollten Sie für die Fehleranalyse und spätere Auswertung mit ack=Disapproved senden. Bei Bedarf kann hier auch noch der Rückgabeparameter error mit einem nützlichen Hinweistext ergänzt werden.

Beispiel einer Abweisung eines Push Vorganges:

```
ack=Disapproved&error=no+matching+order+found+for+hash&hash=
    jtnjpfgrbrqk3300&status_id=6&status_description=abgeschlossen&
    changed=1365444092&payment_status=accepted&apikey=6801
    fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&hint=
```

Aus Sicherheitsgründen ist zu empfehlen alle Eingaben der Push API zu escapen und auf den Referrer api.secupay.ag zu prüfen. Im Idealfall akzeptiert die PUSH API in Ihrem System Anfragen nur per https (Port 443).

5 Parameter

```
apikey              => "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace"
payment_type        => "debit"
demo                => "0"
url_success         => "http://shop.example.com/success.php"
url_failure         => "http://shop.example.com/failure.php"
url_push            => "http://shop.example.com/shop/push"
language            => "de_DE"
shop                => "APITest Shop"
shopversion         => "1.0"
moduleversion       => "1.0"
title               => "Herr"
firstname           => "Test FN"
lastname            => "Test LN"
company             => "Test Company"
street              => "Test Street"
housenumber         => "5t"
zip                 => "12345"
city                => "TestCity"
country             => "DE"
telephone           => "+4912342134123"
dob                 => "01.02.1903"
email               => "test@ema.il"
ip                  => "172.31.6.49"
amount              => "1199"
currency            => "EUR"
purpose             => "Test Order #1"
basket              => <siehe extra Definition #5.2>
```

```
userfields          => <siehe extra Definition #5.3>
delivery_address    => <siehe extra Definition #5.4>
order_id            => "100203"
note                => "default note text"
apiversion          => "2.11"
labels              => <siehe extra Definition #3.2>
```

5.1 Besonderheiten bei einzelnen Parametern

- amount - hierbei handelt es sich jeweils um die kleinste Einheit der entsprechend gewählten Währung. Wird keine Währung explizit übermittelt, gehen wir von Eurocent aus.

5.2 Basket

Beim Basket handelt es sich im Gegensatz zu den anderen Parametern um keine flache Struktur, sondern um ein Array aus Objekten. Dies ermöglicht die einfache Übermittlung von beliebigen Paramtern für die verschiedenen Positionen im Warenkorb.

Die korrekte Übermittlung des Warenkorbes hilft uns beim Support der Transaktionen sowie der Risikoabschätzung der Bestellung.

Nachfolgende Struktur wird von uns nativ aufgenommen und kann bei Bedarf erweitert werden.

```
[
    {
        "article_number": "1234",
        "name": "Testname1",
        "model": "test model",
        "ean": "2309842",
        "quantity": "2",
        "price": "50",
        "total": "100",
        "tax": "19"
    },
    {
        "article_number": "1235",
        "name": "Testname2",
        "model": "test model",
        "ean": "2309843",
        "quantity": "4",
        "price": "199",
        "total": "796",
        "tax": "19"
```

```
    }
]
```

5.3 Userfields

Mit Hilfe der userfields ist es möglich zusätzliche Parameter zu übermitteln, welche im Nachgang die Filterung in unserem Backend erlauben (todo). Sinnvoll sind hier ggfs. Notizen, besondere Eigenschaften des Käufers oder der Bestellung.

```
[
  "userfield_1": "test 1",
  "userfield_2": "test 2...",
  "userfield_3": "test 3"
]
```

5.4 Lieferadresse

Bei der delivery_address handelt es sich um ein Objekt das die Elemente der Lieferadresse aufnimmt.

Die korrekte Übermittlung der Lieferadresse hilft uns beim Support der Transaktionen sowie der Risikoabschätzung der Bestellung.

Nachfolgende Struktur wird von uns nativ aufgenommen und kann bei Bedarf erweitert werden.

```
[
  "firstname": "Test FN",
  "lastname": "Test LN",
  "company": "Test Company",
  "street": "Test Street",
  "housenumber": "5t",
  "zip": "12345",
  "city": "TestCity",
  "country": "DE"
]
```

6 Fehlercodes

Die Fehlerausgaben bestehen immer aus einem code und einer dazugehörigen Message. Die Sprache der ausgegebenen Fehlermeldungen richtet sich nach dem in der Anfrage gesetzten language Parameter. Derzeit werden “en_us” (Standard) sowie “de_de” unterstützt.

Nachfolgend eine Auflistung der möglichen Fehlermeldungen in Englisch:

```
0001 => Invalid apikey
0002 => Invalid hash
0003 => Cannot capture unauthorized payment
0004 => Cannot cancel/void unaccepted payment
0005 => Invalid amount
0006 => Cannot refund unaccepted payment
0007 => No payment type available
0008 => Hash has already been processed
0009 => Scoring invalid
0010 => Payment denied by Scoring
0011 => Payment couldnt be finalized
0012 => Selected payment type is not available
0013 => Apikey mismatch
0014 => Cannot capture specified payment
0015 => Cannot cancel/void specified payment
0016 => Cannot refund specified payment
0017 => Invalid Paymentdata
0018 => Missing Parameter
0019 => Couldnt create the user
0020 => Username/email unavailable
0021 => Username or password invalid
0022 => Userauth Token invalid
0023 => Could not connect the card to the user
0024 => Invalid value for parameter
0025 => Cannot process specified payment
0026 => Cannot create payment information
0027 => Invalid data
0028 => Payment Provider declined the payment
0029 => No transaction available for this terminal
0030 => Cannot create transaction
0031 => Unknown parking zone
0032 => The password you entered is too short
0033 => The passwords are not equal
0034 => The data provided is not sufficient
0035 => There was an error creating the Apikey
0036 => The Plate is not valid
0037 => Payment data missing
0038 => Cannot find any active ticket for this zone and car
0039 => There is already valid ticket for this car in this zone
0040 => There is not any tariff valid for your parking time
```

und auf Deutsch:

```
0001 => Ungültiger apikey
0002 => Ungültiger hash
0003 => nicht autorisierte Zahlung kann nicht eingezogen werden
0004 => nicht abgeschlossene Zahlung kann nicht storniert werden
0005 => Ungüliger Betrag
0006 => nicht abgeschlossene Zahlung kann nicht rückabgewickelt werden
0007 => Keine Zahlart verfügbar
```

```
0008 => Hash wurde schon verarbeitet
0009 => Scoring ungülig
0010 => Zahlung durch Scoring abgelehnt
0011 => Zahlung konnte nicht abgeschlossen werden
0012 => ausgewählte Zahlart ist nicht verfügbar
0013 => Apikey stimmt nicht überein
0014 => Zahlung konnte nicht abgeschlossen werden
0015 => Zahlung konnte nicht storniert bzw erstattet werden
0016 => Zahlung konnte nicht erstattet werden
0017 => Zahlungsmitteldaten ungültig
0018 => Fehlernder Parameter
0019 => Konnte den Nutzer nicht erstellen
0020 => Nutzername/E-Mail bereits vergeben
0021 => Nutzername oder Passwort ungültig
0022 => Userauth Token ungültig
0023 => Konnte die Karte nicht verknüpfen
0024 => Ungültiger Wert für einen Parameter
0025 => Angegebene Zahlung konnte nicht verarbeitet werden
0026 => Zahlungsinformationen konnten nicht erstellt werden
0027 => Daten fehlerhaft
0028 => Fehler beim einreichen der Transaktion beim Zahlungsanbieter.
0029 => Für dieses Terminal ist keine Transaktion hinterlegt.
0030 => Konnte die Transaktion nicht erstellen
0031 => Unbekannte Parkzone
0032 => Das eingegebene Passwort ist zu kurz
0033 => Die eingegebenen Passwörter sind nicht gültig
0034 => Übermittelte Daten nicht vollständig
0035 => Fehler beim anlegen des APIKeys
0036 => Das Kennzeichen ist ungültig
0037 => Zahlungsmitteldaten fehlen
0038 => Es konnte kein aktives Ticket für diese Zone und dieses Fahrzeug gefunden werden
0039 => Es existiert bereits ein Ticket für diese Zone und dieses Fahrzeug
0040 => Für die angegebene Parkzeit existiert kein gültiger Tarif
```