Debit API

Versionsgeschichte

Version 1.0	07.01.2009	H. Heyne

Inhaltsverzeichnis

1. Einleitung

1.1. Allgemein

Die Debit API bietet dem Partner die Möglichkeit, die Micropayment Zahlart "Lastschrift" für seine Kunden anzubieten und gleichzeitig maximale Kontrolle über die Kundeninteraktionen zu erhalten. Hierfür ist der Partner selbst verantwortlich, dem Kunden die benötigten Eingabeformulare und Informationen anzuzeigen und ihn durch den Bezahlvorgang zu führen.

Im wesentlichen erfolgt die Bezahlung in den folgenden Schritten:

1. Eingabe der Kunden- und Bankdaten
2. Auftrag zum Einzug per Lastschrift erzeugen
3. Bestätigung des Lastschrifteinzugs.
4. Benachrichtigung über den erfolgreichen Bankeinzug
5. Benachrichtigung über evtl. Rücklastschrift.
6. Benachrichtigung über evtl. Nachzahlungen

2. Schnittstelle

2.1. Allgemein

2.1.1. Serviceprotokolle

Die Schnittstelle ist derzeit mit Hilfe zweier alternativer Technologien implementiert.

2.1.1.1. Simple HTTP

Die Webservice-URL wird um die Parameterliste als HTTP Querystring (GET-Methode) erweitert. Für den Funktionsnamen ist der Parameter `action` vorgesehen.

```
http://webservice-url/?action=func-name&param-name=param-value&param-name=param-value&...
```

Die Rückgabewerte werden zeilenweise als Name-Wert-Paare im Ergebnisdokument geliefert:

```
error=0
result-name=result-value
result-name=result-value
...
```

Im Fehlerfall werden nur die beiden Standardrückgaben `error` und `errorMessage` geliefert:

```
error=error-code
errormessage=error-message
```

Die jeweiligen Parameter- und Rückgabewerte sind URL-codiert. Sonderzeichen sind nach ISO-8859-1 Standard zu codieren.

Bei Listenelementen (Arrays) werden die Indizes in eckige Klammern eingeschlossen. Die Eigenschaften strukturierter Typen (Objekte) werden mit Punkt vom Namen getrennt:

```
http://webservice-url/?...&param[index]=value&param.property=value
```

bzw.

```
result[index]=value
result.property=value
```

2.1.1.2. SOAP Webservice

An die Service-URL werden Funktion und Parameter als Soap-Envelop gesendet (HTTP Methode POST). Alle Parameter sind hierbei in einem strukturierten Typ enthalten:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope
  soap-env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:dbt="http://webservices.micropayment.de/public/debit/version1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap-env:Body>
    <dbt:func-name>
      <param xsi:type="dbt:Dbtfunc-nameRequestTyp">
        <param-name xsi:type="param-type">param-value</param-name>
        <param-name xsi:type="param-type">param-value</param-name>
        ...
      </param>
    </dbt:func-name>
  </soap-env:Body>
</soap-env:Envelope>
```

Das Ergebnisdokument enthält im Erfolgsfall die Rückgabestruktur, ...

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope
  soap-env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:dbt="http://webservices.micropayment.de/public/debit/version1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap-env:Body>
    <dbt:func-nameResponse>
      <return xsi:type="dbt:Dbtfunc-nameResponseTyp">
        <result-name xsi:type="result-type">result-value</result-name>
        <result-name xsi:type="result-type">result-value</result-name>
        ...
      </return>
    </dbt:func-nameResponse>
  </soap-env:Body>
</soap-env:Envelope>
```

... oder einen Soap-Fault mit den Standardrückgaben `error` und `errorMessage`:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope
  soap-env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
  <soap-env:Body>
    <soap-env:Fault>
      <faultcode>error-code</faultcode>
      <faultstring>error-message</faultstring>
      <faultactor></faultactor>
      <detail></detail>
    </soap-env:Fault>
  </soap-env:Body>
</soap-env:Envelope>
```

2.1.2. Standardparameter und -rückgaben für Funktionen

Jeder Aufruf erfolgt mittels definierter URL der Serviceschnittstelle und dem erforderlichen Parameter `accessKey` zur Identifizierung des Aufrufers. Der AccessKey wird automatisch bei der Partnerregistrierung im Micropayment-System erzeugt, und kann im Controlcenter ermittelt werden. Optional erwarten alle Funktionen den Parameter `testMode`. Mit seiner Hilfe lässt sich die integrierte Testumgebung aktivieren.

Scheitert eine Funktion, wird der aufgetretene Fehler durch die Rückgaben `error` und `errorMessage` beschrieben. `error` enthält hier einen numerischen Code, der programmseitig ausgewertet werden kann. `errorMessage` hingegen enthält die nähere Beschreibung im Klartext und sollte im Fehlerfall mitprotokolliert werden.

2.1.3. Standardparameter und -rückgaben für Benachrichtigungen

Die API ist in der Lage verschiedene Ereignisse an eine Schnittstelle des Projektbetreibers zu senden. Dafür muss in der Projektkonfiguration die URL der Schnittstelle eingetragen werden. Alle Benachrichtigungen übermitteln den Parameter `testMode`, der angibt, ob sie von der Testumgebung ausgelöst wurden.

Wird beim Aufruf der Benachrichtigungsurl ein Fehler zurückgegeben, wird eine automatische Email an den Betreiber mit den Aufrufparametern versandt.

2.1.4. Integrierte Testumgebung

Alle Funktionen erwarten den optionalen Parameter `testMode`, der festlegt, dass die Daten in einer abgetrennten Umgebung (der sogenannten Sandbox) erzeugt und abgerufen werden sollen. Keiner der hier angelegten Bezahlvorgänge wird jemals ausgeführt, es können aber mit speziellen Funktionen einige externe Ereignisse simuliert werden.

Die Testumgebung soll es ermöglichen, während und nach Integration der API, alle Abläufe realitätsnah auszutesten oder sogar automatisierte Testfälle zu implementieren (Unittests).

2.1.5. Fehlercodes

Um Fehlerbehandlung zu vereinfachen, sind die Fehlercodes der Rückgabe `error` in vier verschiedene Klassen mit eigenem Nummernkreis unterteilt.

permanenter Serverfehler `1xxx`	Bei diese Fehlerklasse liegt meist ein permanentes Problem beim Webdienst vor. Bei Fehlern dieser Klasse sollte der Support informiert werden.
temporärer Serverfehler `2xxx`	Fehler dieses Typs sind auch vom Webdienst bedingt, haben aber nur eine vorübergehenden Ursache, wie Wartungsarbeiten oder

	erschöpfte Ressourcen. Der Kunde kann hierbei auf spätere Nutzung des Dienstes vertröstet werden.
Clientfehler `3xxx`	Die Ursache der Fehler aus dieser Klasse liegt typischerweise bei der aufrufende Applikation. Es ist sinnvoll diese Fehler mit dem Text aus `errorMessage` mitzuloggen, und die Anwendung entsprechend zu modifizieren. Zur Behebung kann die Dokumentation oder der Support zu Rate gezogen werden.
Userfehler `4xxx`	Zu dieser Klasse gehören alle Fehler, die typischerweise aus falschen Eingaben des Kunden resultieren, z.B. falsches Passwort etc. Dem User ist hier eine aussagekräftige Fehlerbeschreibung anzuzeigen, damit er die Falscheingabe korregieren kann. Die Ursache kann aber auch hier bei der Applikation liegen, die z.B. eingegebene Werte fehlerhaft formatiert.

2.2. Funktionen

Im folgenden wird auf die Verwendung und die Besonderheiten der Schnittstellenfunktionen eingegangen. Die jeweils vollständige Parameter- und Rückgabenliste sind in der Kurzreferenz zusammengefasst.

2.2.1. Testumgebung löschen mit `resetTest`

Die Funktion löscht alle Customer und Sessions in der Testumgebung. Mit ihrer Hilfe können u.a. Kollisionen für `customerId` und `sessionId` zu vermieden werden im Zusammenhang mit automatischen Unittests.

2.2.2. Kunden anlegen, ändern und abrufen mit `customerCreate`, `customerSet` und `customerGet`

Vor jedem Bezahlvorgang muss ein Kunde angelegt werden. Zur Identifizierung kann mit `customerId` eine eigene Kundennummer, der Username oder die Emailadresse übergeben werden, andernfalls wird von der API eine eindeutige ID erzeugt und zurückgegeben. Die Funktion scheitert, wenn der Wert in `customerId` bereits für den Account existiert.

Achtung

Wenn mehrere Projekte betrieben werden, in denen sich die Kunden getrennt anmelden müssen, sollten verschiedene Präfixe vor den Wert in `customerId` gesetzt werden, z.B: "prj1:max@muster.de".

An den Kundensatz können mit der assoziativen Liste `freeParams` beliebig viele zusätzliche Daten als freie Parameter gebunden werden. Diese können später mit `customerGet` abgerufen oder mit `customerSet` verändert bzw. erweitert werden. Dadurch ist prinzipiell eine Anbindung der API ohne eigene Datenbank möglich, indem alle anderen Kundendaten wie Email, Passwort oder Sperrstatus als freie Parameter hinterlegt werden.

Einem Kunden können mit `customerSet` jederzeit weitere freie Parameter hinzugefügt werden, oder bestehende geändert werden. Dabei werden nur Werte erzeugt oder überschrieben, die im Parameter `freeParams` enthalten sind; bestehende bleiben unverändert. Zum Löschen eines Wertes kann einfach ein leerer String ("") übergeben werden.

2.2.3. Bankdaten hinterlegen und abrufen mit `bankaccountSet` und `bankaccountGet`

Nachdem ein Kunde angelegt wurde und vor der Bezahlung, muss die Bankverbindung mit `bankaccountSet` hinterlegt werden. Sollte sich diese einmal ändern kann auch die neue Bankverbindung mit

dieser Funktion übergeben werden. Der Parameter `customerId` identifiziert dabei den Kunden. Die Funktion prüft Kontonummer und Bankleitzahl auf Plausibilität nach den Vorgaben der Bundesbank und scheitert gegebenenfalls. Im Erfolgsfall wird der ermittelte Name der Bank zurückgeliefert.

Die hinterlegten Bankdaten des Kunden können mit `bankaccountGet` wieder ermittelt werden, z.B. um sie vom Kunden bearbeiten zu lassen. Beide Funktionen liefern zusätzlich den Sperrstatus der Bankverbindung (siehe `bankaccountBar`).

2.2.4. Bankverbindungen prüfen und sperren mit `bankCheck`, `bankaccountCheck` und `bankaccountBar`

Mit Hilfe der Funktionen `bankCheck` und `bankaccountCheck` können Bankname ermittelt und die Kontonummer geprüft werden, ohne dass sie einem Kunden zugeordnet werden. Die Funktion `bankaccountBar` ermöglicht es, eine bestimmte Bankverbindung generell zu sperren bzw. wieder freizugeben. Ein Aufruf mit `sessionCreate` würde dann scheitern, wenn der zugehörige Kunde eine gesperrte Bankverbindung besitzt. `bankaccountSet` und `bankaccountGet` liefern mit `barStatus` den jeweiligen Sperrstatus

2.2.5. Zusätzliche Adress- und Kontaktdaten hinterlegen und abrufen

Mit den Funktionen `addressSet` und `contactDataSet` kann ein Kundendatensatz um Namen und Anschrift erweitert werden. Für den Bezahlvorgang ist dies nicht unbedingt notwendig, kann aber für evtl. Forderungsmaßnahmen sinnvoll sein. Zum Abruf der Daten stehen die Funktionen `addressGet` sowie `contactDataGet` zur Verfügung.

Während bei `addressSet` alle Daten angegeben werden müssen, kann bei `contactDataSet` jeder Bestandteil separat übermittelt werden. Die anderen Parameter sind in diesem Fall wegzulassen, und werden nicht gelöscht.

2.2.6. Lastschriftauftrag erzeugen mit `sessionCreate`

Nachdem Kunde und Bankverbindung angelegt wurden, kann ein neuer Bezahlvorgang erzeugt werden. Der Parameter `sessionId` muss eindeutig sein und wird ggf. generiert und zurückgegeben - ähnlich, wie `customerId` in der Funktion `customerCreate`. Es ist sinnvoll hier z.B. die aktuelle Sitzungsnummer des Users anzugeben, um Doppelbuchungen zu vermeiden.

Die Parameter `project` und `projectCampaign` legen fest, welchem Projekt und welcher Kampagne der Erlös zugeordnet werden soll. Die Parameter `account` und `webmasterCampaign` werden benötigt, wenn das Projekt webmasterfähig ist.

Wichtig

Die Funktion scheitert, wenn im Parameter `projectCampaign` eine ungültige oder gesperrte Kampagne angegeben wurde. Ist hingegen `webmasterCampaign` ungültig, wird der Parameter lediglich ignoriert.

Mit `amount`, `currency` und `title` werden Artikel und Preis (in Cent) festgelegt. Werden die Parameter weggelassen, werden die Standardwerte aus der Projekteinstellung angenommen. Der Verwendungszweck entspricht dem Parameter `payText`, bzw. wird aus dem Projektnamen und dem Wert in `title` zusammengesetzt. Mit der Funktion `sessionGet` können alle übergebenen oder generierten Werte zurückgegeben werden, um sie z.B. dem Kunden zur Bestätigung anzuzeigen.

Wichtig

Wurde in der Konfiguration festgelegt, dass NUR die Standardwerte verwendet werden sollen, werden die Parameter `amount` und `title` ignoriert.

Ähnlich wie bei der Funktion `customerCreate`, können auch dem Bezahlvorgang freie Parameter übergeben und mit der Funktion `sessionGet` wieder ermittelt werden. Der separate Parameter `ip`

wird zusammen mit dem Zeitstempel für Missbrauchsfälle vorgehalten und sollte die IP des Benutzers enthalten, der die Bezahlung veranlasst.

Die Funktion liefert typischerweise den Status `"INIT"` zurück. Wurde allerdings für den Kunden ein unbestätigter Vorgang gefunden, wird dieser mit den Werten dieses Aufrufs überschrieben und es wird der Status `"REINIT"` zurüchgegeben.

Bei erfolgreichem Aufruf von `sessionCreate`, wird die Benachrichtigung `sessionStatus` ausgelöst. Wurde die Bankverbindung des Kunden mit `bankaccountBar` generell gesperrt, scheitert der Aufruf.

Achtung

Die Benachrichtigung `sessionStatus` erfolgt sofort, noch bevor die Rückgabe geliefert wird.

2.2.7. Vorgang abfragen mit `sessionGet`

Die Funktion ermittelt den aktuellen Status eines Vorgangs, sowie alle Daten entsprechend der Parameter aus der Funktion `sessionCreate`. Die Rückgabe `status` kann folgende Zustände annehmen:

`"INIT"`	Vorgang wurde initialisiert und wartet auf Bestätigung.
`"REINIT"`	Bestehender Vorgang wurde neu erzeugt und wartet auf Bestätigung.
`"APPROVED"`	Bezahlung wurde bestätigt, der Betrag wird im Laufe der nächsten Tage eingezogen.
`"CHARGED"`	der Betrag wurde erfolgreich eingezogen.
`"FAILED"`	die Lastschrift ist gescheitert. Kosten sind dadurch nicht entstanden.
`"REVERSED"`	die Lastschrift wurde zurückgebucht oder vom Kunden storniert. Dadurch sind weitere Rücklastgebühren entstanden.
`"RECHARGED"`	Die stornierte Lastschrift wurde vollständig (inklusive der Rücklastgebühr) nachgezahlt.

In den Fällen `"FAILED"` und `"REVERSED"`, wird mit `statusDetail` die detaillierte Ursache im Klartext zurückgegeben.

Zusätzlich werden alle Daten entsprechend der Parameter aus der Funktion `sessionCreate` zurückgegeben. Für optionale Parameter werden dabei die Voreinstellungen bzw. Vorgaben aus der Konfiguration geliefert.

2.2.8. Lastschriftbestätigung mit `sessionApprove`

Angelegte Bezahlvorgänge müssen mit dieser Funktion bestätigt werden. Sie sollte unmittelbar nach dem ausdrücklichen Abbuchungauftrags des Kunden aufgerufen werden, oder sogar nach der Bereitstellung des gekauften Artikels, z.B. nach abgeschlossenem Download eines Dokuments.

Bei Aufruf von `sessionApprove`, wird die Benachrichtigung `sessionStatus` ausgelöst.

Achtung

Die Benachrichtigung `sessionStatus` erfolgt sofort, noch bevor die Rückgabe geliefert wird.

2.2.9. Vorgänge abrufen mit `sessionList`

Mit Hilfe dieser Funktion können alle offenen, sowie abgeschlossenen Vorgänge eines Kunden ermittelt werden. Die Rückgabe besteht aus einer numerisch indizierten Liste aller zugehöriger SessionID's. Die Details der Sessions können wiederum mit `sessionGet` ermittelt werden.

2.2.10. Bestätigte Bezahlvorgänge abbuchen mit `sessionChargeTest`

Alle bestätigten Zahlungen werden zusammengefasst und der Bank zur Abbuchung vorgelegt. Der anschließende Geldeingang löst typischerweise die Statusänderung `"CHARGED"` und die entsprechenden Benachrichtigung `sessionStatus` aus. Ausserdem wird eine Transaktion des Typs `"BOOKING"` angelegt und die Benachrichtigung `transactionCreate` ausgelöst. Die Funktion dient dazu, um diesen Geldeingang in der Testumgebung zu simulieren.

2.2.11. Einzelne Buchungen stornieren mit `sessionReverseTest`

Die Funktion simuliert im Testmodus eine Rückbelastung des Betrages. In der Realität kann das z.B. durch Widerspruch des Kontoinhabers oder mangelnde Deckung ausgelöst werden. Auch hier wird mittels `sessionStatus` und `transactionCreate` das Zielsystem benachrichtigt, sowie eine Transaktion des Typs `"REVERSAL"` erzeugt.

2.2.12. Beträge ganz oder Teilweise nachzahlen mit `sessionRechargeTest`

Wurde mit `sessionReverseTest` ein Storno in der Testumgebung simuliert, kann anschliessend die Nachzahlung simuliert werden. Der optionale Parameter `amount` enthält den bezahlten (Teil-)Betrag, wird er weggelassen, wird die Nachzahlung des gesamten offenen Betrags simuliert und als Rückgabe `amount` zurückgeliefert. `sessionRechargeTest` sowie das reale Ereignis eines Zahlungeingangs erzeugt eine Transaktion vom Typ `"BACKPAY"` und löst die zugehörige Benachrichtigung `transactionCreate` aus. Wurde der gesamte offene Betrag bezahlt, wird zusätzlich `sessionStatus` auf `"RECHARGED"` gesetzt und die Benachrichtiung `sessionStatus` ausgelöst.

2.2.13. Eigene Zahlungseingänge verbuchen mit `transactionCreate`

Mit Hilfe dieser Funktion kann ein eigener Zahlungseingang, oder eine nachträgliche Forderungsminderung verbucht werden. Sie erzeugt eine neue Transaktion vom Typ `"EXTERNAL"` und löst zusätzlich die Benachrichtigung `transactionCreate` aus. Neben Datum und Betrag kann zusätzlich ein Beschreibungstext zur eigenen Verwendung übermittelt werden. Wird der gesamte offene Betrag oder mehr verbucht ändert sich der Status der Session in `"RECHARGED"`. Es besteht auch die Möglichkeit mit einem negativem Betrag die offene Summe wieder zu erhöhen oder überzahlte Sessions auszugleichen.

2.2.14. Transaktionen ermitteln mit `transactionList` und `transactionGet`

Diese Funktionen dienen der Ermittlung aller Transaktionen einer Session. `transactionList` liefert eine Liste der Transactions-IDs, die wiederum als Eingangsparameter für `transactionGet` genutzt werden können, um die Details abzurufen. Der Typ der Transaktion in der Rückgabe `type` kann folgende Werte enthalten:

`"BOOKING"` Die Hauptbuchung der Session wird erzeugt bei Bankeinzug bzw. durch `sessionChargeTest`.

`"REVERSAL"` Die Rücklastschrift, erzeugt durch die Rücklastschrift bzw. `sessionReverseTest`. Der Wert `amount` ist negativ und enthält zusätzlich die Rücklastgebühr.

`"BACKPAY"` Eine (Teil-)Nachzahlung. Sie wird bei Zahlunsgeingang bzw. `sessionRechargeTest` erzeugt.

`"EXTERNAL"` Externe Buchung oder Forderungsminderung. Sie wird ausschließlich durch `transactionCreate` erzeugt. `amount` ist positiv bei externen Zahlungeingang, kann aber auch negativ bei einer Erhöhung der Forderung sein.

2.3. Benachrichtigungen

2.3.1. Statusänderung durch `sessionStatus`

Jede Statusänderung, auch die Erzeugung, eines Bezahlvorgangs führt zu jeweils einer Benachrichtigung mit dieser Funktion z.B. wenn der Betrag bestätigt, eingezogen oder storniert wird. Es werden `sessionId` und der aktuelle Status, sowie alle freien Parameter mitgeschickt.

Als Ergebnis können neue oder veränderte `freeParams` zurückgegeben werden, die anschließend zur Session hinzugefügt werden.

2.3.2. Neue Transaktion durch `transactionCreate`

Wie oben beschrieben, können mehrere verschiede Transaktionen je Session erzeugt werden. Die Benachrichtigung dient dazu das Zielsystem darüber zu informieren. Bei den Transaktionsarten `"BOOKING"` und `"REVERSAL"` geht dies immer mit einer Statusänderung der Session einher und der zugehörigen Benachrichtigung `sessionStatus`. Wurde bei `"BACKPAY"` und `"EXTERNAL"` nicht der komplette offene Betrag gebucht, ändert sich der Status der Session jedoch nicht.

A. Kurzreferenz

1. Funktion `resetTest`

löscht alle Kunden und Transaktionen in der Testumgebung

Parameter	Typ	Standard *(Pflicht)	Beschreibung
`accessKey`	string	*	AccessKey aus dem Controlcenter
`testMode`	integer	*	Muss 1 sein

Tabelle A.1. Funktion `resetTest` Parameter

2. Funktion `customerCreate`

legt neuen Kunden an

Parameter	Typ	Standard *(Pflicht)	Beschreibung
`accessKey`	string	*	AccessKey aus dem Controlcenter
`testMode`	integer	`0`	aktiviert Testumgebung
`customerId`	string	`null`	eigene eindeutige ID des Kunden, wird anderenfalls erzeugt
`freeParams`	array	`null`	Liste mit freien Parametern, die dem Kunden zugeordnet werden

Tabelle A.2. Funktion `customerCreate` Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
`customerId`	string	*	eigene oder erzeugte eindeutige ID des Kunden

Tabelle A.3. Funktion `customerCreate` Rückgabe

3. Funktion `customerSet`

ordnet weitere freie Parameter dem Kunden zu, oder ändert sie

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	eindeutige ID des Kunden
freeParams	array	null	Liste mit zusätzlichen freien Parametern

Tabelle A.4. Funktion customerSet Parameter

4. Funktion customerGet

ermittelt alle freien Parameter des Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	ID des Kunden

Tabelle A.5. Funktion customerGet Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
freeParams	array	*	Liste mit allen freien Parametern

Tabelle A.6. Funktion customerGet Rückgabe

5. Funktion bankaccountSet

erzeugt oder ändert Bankverbindung eines Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	ID des Kunden
country	string	'DE'	Sitz der Bank
bankCode	string	*	Bankleitzahl
accountNumber	string	*	Kontonummer
accountHolder	string	*	Kontoinhaber

Tabelle A.7. Funktion bankaccountSet Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
bankName	string	*	der ermittelte Name der Bank
barStatus	string	*	Sperr-Status der Kontoverbindung

Tabelle A.8. Funktion bankaccountSet Rückgabe

6. Funktion bankaccountGet

ermittelt die Bankverbindung des Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	ID des Kunden

Tabelle A.9. Funktion bankaccountGet Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
country	string	*	Sitz der Bank
bankCode	string	*	Bankleitzahl
bankName	string	*	Name der Bank
accountNumber	string	*	Kontonummer
accountHolder	string	*	Kontoinhaber
barStatus	string	*	Sperr-Status der Konto-verbindung

Tabelle A.10. Funktion bankaccountGet Rückgabe

7. Funktion bankCheck

prüft Bankleitzahl und ermittelt Banknamen

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
country	string	'DE'	Sitz der Bank
bankCode	string	*	Bankleitzahl

Tabelle A.11. Funktion bankCheck Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
bankName	string	*	Name der Bank

Tabelle A.12. Funktion bankCheck Rückgabe

8. Funktion bankaccountCheck

prüft Bankverbindung und ermittelt Banknamen

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
country	string	'DE'	Sitz der Bank
bankCode	string	*	Bankleitzahl
accountNumber	string	*	Kontonummer

Tabelle A.13. Funktion bankaccountCheck Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
bankName	string	*	der ermittelte Name der Bank
barStatus	string	*	Sperr-Status der Konto-verbindung

Tabelle A.14. Funktion bankaccountCheck Rückgabe

9. Funktion bankaccountBar

Sperrt Bankverbindung oder gibt sie frei

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
country	string	'DE'	Sitz der Bank
bankCode	string	*	Bankleitzahl
accountNumber	string	*	Kontonummer
barStatus	string	*	Sperr-Status BARRED, ALLOWED

Tabelle A.15. Funktion bankaccountBar Parameter

10. Funktion addressSet

erzeugt oder ändert Adressdaten eines Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	ID des Kunden
firstName	string	*	Vorname
surName	string	*	Nachname
street	string	*	Strasse und Hausnum-mer
zip	string	*	Postleitzahl
city	string	*	Ort
country	string	'DE'	Land

Tabelle A.16. Funktion addressSet Parameter

11. Funktion addressGet

ermittelt die Adresse des Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	ID des Kunden

Tabelle A.17. Funktion addressGet Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
`firstName`	string	*	Vorname
`surName`	string	*	Nachname
`street`	string	*	Strasse und Hausnummer
`zip`	string	*	Postleitzahl
`city`	string	*	Ort
`country`	string	*	Land

Tabelle A.18. Funktion `addressGet` Rückgabe

12. Funktion `contactDataSet`

erzeugt oder ändert Kontaktdaten eines Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
`accessKey`	string	*	AccessKey aus dem Controlcenter
`testMode`	integer	`0`	aktiviert Testumgebung
`customerId`	string	*	ID des Kunden
`email`	string	`null`	Emailadresse des Kunden
`phone`	string	`null`	Festnetzanschluss
`mobile`	string	`null`	Handynummer

Tabelle A.19. Funktion `contactDataSet` Parameter

13. Funktion `contactDataGet`

ermittelt die Kontaktdaten des Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
`accessKey`	string	*	AccessKey aus dem Controlcenter
`testMode`	integer	`0`	aktiviert Testumgebung
`customerId`	string	*	ID des Kunden

Tabelle A.20. Funktion `contactDataGet` Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
`email`	string	*	Emailadresse
`phone`	string	*	Festnetzanschluss
`mobile`	string	*	Handynummer

Tabelle A.21. Funktion `contactDataGet` Rückgabe

14. Funktion `sessionCreate`

erzeugt einen neuen Bezahlvorgang

Parameter	Typ	Standard *(Pflicht)	Beschreibung
`accessKey`	string	*	AccessKey aus dem Controlcenter
`testMode`	integer	`0`	aktiviert Testumgebung
`customerId`	string	*	ID des Kunden
`sessionId`	string		eigene eindeutige ID des Vorgangs, wird anderenfalls erzeugt
`project`	string	*	das Projektkürzel für den Vorgang
`projectCampaign`	string		ein Kampagnenkürzel des Projektbetreibers
`account`	string		Account des beteiligten Webmasters
`webmasterCampaign`	string		ein Kampagnenkürzel des Webmasters
`amount`	integer	`0`	abzurechnender Betrag in Cent
`currency`	string	`'EUR'`	Währung
`title`	string		Bezeichnung der zu kaufenden Sache
`payText`	string		Abbuchungstext der Lastschrift
`ip`	string		IP des Benutzers
`freeParams`	array	`null`	Liste mit freien Parametern, die dem Vorgang zugeordnet werden

Tabelle A.22. Funktion `sessionCreate` Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
`sessionId`	string	*	eigene oder erzeugte eindeutige ID des Vorgangs
`status`	string	*	Vorgangsstatus "INIT" oder "REINIT"
`expire`	string	*	Ablaufzeit der Bestätigung

Tabelle A.23. Funktion `sessionCreate` Rückgabe

15. Funktion `sessionSet`

ordnet weitere freie Parameter der Session zu, oder ändert sie

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
sessionId	string	*	ID des Vorgangs
freeParams	array	null	Liste mit zusätzlichen freien Parametern

Tabelle A.24. Funktion sessionSet Parameter

16. Funktion sessionGet

ermittelt Daten eines Bezahlvorgangs

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
sessionId	string	*	ID des Vorgangs

Tabelle A.25. Funktion sessionGet Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
status	string	*	Vorgangsstatus "INIT", "REINIT", "EXPIRED", "APPROVED", "FAILED", "CHARGED", "REVERSED" oder "RECHARGED"
expire	string	*	Ablaufzeit bzw. Bestätigung des Vorgangs
statusDetail	string	*	Beschreibung für gescheiterte Transaktionen
customerId	string	*	ID des Kunden
project	string	*	zugeordnetes Projekt
projectCampaign	string	*	zugeordnete Projektkampagne
account	string	*	zugeordneter Webmasteraccount
webmasterCampaign	string	*	zugeordnete Webmasterkampagne
amount	integer	*	übergebener Betrag bzw. Standard aus Konfiguration in Cent
openAmount	integer	*	offener, noch zu zahlender Betrag der Session
currency	string	*	übergebene Währung bzw. "EUR"
title	string	*	übergebene Kaufsache bzw. Standard aus Konfiguration
payText	string	*	Abbuchungstext der Lastschrift
ip	string	*	übergebene IP des Benutzers
freeParams	array	*	Liste mit allen freien Parametern

Tabelle A.26. Funktion `sessionGet` Rückgabe

17. Funktion `sessionApprove`

bestätigt den Lastschrifteinzug eines Vorgangs

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
sessionId	string	*	ID des Vorgangs

Tabelle A.27. Funktion `sessionApprove` Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
status	string	*	Vorgangsstatus "APPROVED" oder "FAILED"
expire	string	*	Zeitpunkt der Bestätigung

Tabelle A.28. Funktion sessionApprove Rückgabe

18. Funktion sessionList

ermittelt alle Bezahlvorgänge eines Kunden

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
customerId	string	*	ID des Kunden

Tabelle A.29. Funktion sessionList Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
count	integer	*	Anzahl der Einträge in sessionIdList
sessionIdList	array	*	0-indizierte Liste mit Vorgang-IDs

Tabelle A.30. Funktion sessionList Rückgabe

19. Funktion sessionChargeTest

simuliert die Abbuchung für alle bestätigten Vorgänge

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	muss 1 sein

Tabelle A.31. Funktion sessionChargeTest Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
count	integer	*	Anzahl der gebuchten Vorgänge

Tabelle A.32. Funktion sessionChargeTest Rückgabe

20. Funktion sessionReverseTest

simuliert Stornierung eines einzelnen Vorgangs

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	muss 1 sein
sessionId	string	*	ID des Vorgangs

Tabelle A.33. Funktion sessionReverseTest Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
amount	integer	*	stornierter Betrag inkl. Gebühr

Tabelle A.34. Funktion sessionReverseTest Rückgabe

21. Funktion sessionRechargeTest

simuliert die komplette Nachzahlung eines stornierten Vorgangs

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	muss 1 sein
sessionId	string	*	ID des Vorgangs
amount	integer	null	optional der nachgezahlte Teilbetrag

Tabelle A.35. Funktion sessionRechargeTest Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
amount	integer	*	gebuchter Betrag

Tabelle A.36. Funktion sessionRechargeTest Rückgabe

22. Funktion transactionCreate

erstellt eine Transaktion vom Typ "EXTERNAL"

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
sessionId	string	*	ID der zugehörigen Session
date	string	null	Datum der Transaktion
amount	integer	*	Transaktionsbetrag
description	string	'""'	Beschreibungstext

Tabelle A.37. Funktion transactionCreate Parameter

23. Funktion transactionList

ermittelt alle Transaktionen für einen Bezahlvorgang

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
sessionId	string	*	ID des Vorgangs

Tabelle A.38. Funktion transactionList Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
count	integer	*	Anzahl der Einträge in transactionIdList
transactionIdList	array	*	0-indizierte Liste mit Transaktions-IDs

Tabelle A.39. Funktion transactionList Rückgabe

24. Funktion transactionGet

ermittelt Daten einer Transaktion

Parameter	Typ	Standard *(Pflicht)	Beschreibung
accessKey	string	*	AccessKey aus dem Controlcenter
testMode	integer	0	aktiviert Testumgebung
transactionId	string	*	ID des Vorgangs

Tabelle A.40. Funktion transactionGet Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
sessionId	string	*	ID des Vorgangs
date	string	*	Datum der Transaktion
type	string	*	Art der Transaktion "BOOKING", "REVERSAL", "BACKPAY", "EXTERNAL"
amount	integer	*	Transaktionsbetrag
description	string	*	Beschreibungstext

Tabelle A.41. Funktion transactionGet Rückgabe

25. Benachrichtigung sessionStatus

Benachrichtigt bei jeder Änderung des Vorgangsstatus, und fügt weitere freie Parameter hinzu

Parameter	Typ	Standard *(Pflicht)	Beschreibung
testMode	integer	*	1 bei aktivierter Testumgebung
sessionId	string	*	ID des Vorgangs
status	string	*	Vorgangsstatus "INIT", "REINIT", "EXPIRED", "APPROVED", "FAILED", "CHARGED", "REVERSED" oder "RECHARGED"
freeParams	array	null	Liste mit allen freien Parametern

Tabelle A.42. Benachrichtigung sessionStatus Parameter

Rückgabe	Typ	Standard *(Pflicht)	Beschreibung
freeParams	array	null	Liste mit zusätzlichen freien Parametern

Tabelle A.43. Benachrichtigung sessionStatus Rückgabe

26. Benachrichtigung `transactionCreate`

Benachrichtigt bei Erstellung einer neuen Transaktion

Parameter	Typ	Standard *(Pflicht)	Beschreibung
`testMode`	integer	*	1 bei aktivierter Testumgebung
`sessionId`	string	*	ID des Vorgangs
`transactionId`	string	*	ID der Transaktion
`date`	string	*	Datum der Transaktion
`type`	string	*	Transaktionsart "BOOKING", "REVERSAL", "BACKPAY", "EXTERNAL"
`amount`	string	*	Transaktionsbetrag
`description`	string	*	Beschreibungstext

Tabelle A.44. Benachrichtigung `transactionCreate` Parameter