

IKT-Basisdienst E-Payment Schnittstellenbeschreibung

Zur Anbindung eines Fachverfahrens an den IKT-Basisdienst E-Payment.

Version:	2.0.37
Autor:	Steffen Gransow
Firma:	BerlinOnline Stadtportal GmbH & Co. KG Stefan-Heym-Platz 1, 10367 Berlin
Datum:	13.12.2023

1. Inhaltsverzeichnis

- [IKT-Basisdienst E-Payment Schnittstellenbeschreibung](#)
 - [Inhaltsverzeichnis](#)
 - [Historie](#)
 - [Begriffe](#)
 - [Zahlungsablauf](#)
 - [Zahlungsarten aus Kundensicht](#)
 - [SEPA aus Sicht der Nutzenden](#)
 - [Giropay aus Sicht der Nutzenden](#)
 - [Kreditkarte aus Sicht der Nutzenden](#)
 - [PayPal aus Sicht der Nutzenden](#)
 - [Grober Ablauf von Aufrufen zwischen Shop, Kunde und API im Erfolgsfall](#)
 - [Abfrage der Berlin.de-Payment-Schnittstelle](#)
 - [Technische Vorbedingungen](#)
 - [Allgemeine Hinweise](#)
 - [Anfragen an die API-Funktionen](#)
 - [Antworten der API-Funktionen](#)
 - [API-Funktionen](#)
 - [Ablauf API-Aufrufe](#)
 - [GetBasketId](#)
 - [PayInit](#)
 - [PayConfirm](#)
 - [PayComplete](#)
 - [Benachrichtigung an den Shop \(Notify\)](#)
 - [Hinweis](#)
 - [Wichtig](#)
 - [GetStatus](#)
 - [Wartungsmodus](#)
 - [Anhang](#)
 - [Formatbeschreibung der Parameter](#)
 - [Ergebnismeldungen / Antworten](#)
 - [API - Errorcodes](#)
 - [Testdaten](#)
 - [Zahlung mit Kreditkarte](#)
 - [Zahlungen mit SEPA](#)
 - [Zahlungen mit Giropay](#)
 - [Zahlungen mit PayPal](#)
 - [Tests](#)
 - [Support-Anfragen](#)

2. Historie

Datum	Version	Autor	Änderung
2023/12/13	2.0.37	TA, SG	PayOne hat angekündigt, auf der PayPage für Kreditkarten die ggfs. fehlenden zusätzlichen Adress- und Kontaktdaten abzufragen, falls der Shop diese nicht übergibt. <code>customerid</code> bei <code>PayInit</code> erlaubt auch Kleinbuchstaben. Der bei <code>PayInit</code> übergebbare <code>backlink</code> ist abgekündigt/deprecated. Der Zwischenschritt wird den <code>faillink</code> stattdessen nutzen. Für Shops wird damit nicht mehr unterscheidbar, ob eine Transaktion "einfach abgebrochen" wurde von Nutzenden oder tatsächlich aus anderen Gründen fehlschlug. Das war allerdings auch vorher schon nicht gut voneinander unterscheidbar. Am Ausgang einer Transaktion ändert zudem die Art des Abbruchs nichts, daher wird konsequent der möglicherweise bekannte <code>faillink</code> genutzt. Konkretisierungen zu <code>successlink</code> / <code>faillink</code> und dem dort mitgelieferten <code>status</code> -Parameter.
2023/12/01	2.0.36	TA, SG	Neue Parameter bei <code>PayInit</code> für die Zahlarten Kreditkarte und Giropay: <code>payerfirstname</code> , <code>payerlastname</code> , <code>payeremail</code> , <code>payerphone</code> , <code>payerphonemob</code> , <code>payerphonework</code> , <code>addrcity</code> , <code>addrcountry</code> , <code>addrpostcode</code> , <code>addrstreet</code> , <code>addrstreetnumber</code> , <code>shipcity</code> , <code>shipcountry</code> , <code>shippostcode</code> , <code>shipstreet</code> , <code>shipstreetnumber</code> . Angabe der nutzbaren Felder bei <code>PayInit</code> pro Zahlart. Hinweis auf neue Pflichtfelder ab Januar 2024 (Giropay) bzw. Mitte Februar 2024 (Kreditkarte).
2023/10/12	2.0.35	SG	Giropay-Testdaten für erfolgreiche Zahlungen und zur Auslösung von Fehlerzuständen entfernt, da die Hintergrundsysteme dies nicht mehr anbieten (BICs WELADED1AUS und WELADED1ZWI etc.). Giropay ist jetzt Paydirekt und das dort angebotene Sandbox-System ist derzeit problematisch.
2023/10/10	2.0.34	SG	Diverse Anpassungen und Konkretisierungen bei Beschreibung der Payment-API, insbesondere bei <code>PayConfirm</code> und <code>Notify</code> . Klarstellung, dass <code>data</code> und <code>signature</code> als URL-Parameter sowohl bei Rückkehr von Kunden via <code>successlink</code> als auch beim Aufruf des <code>notifylink</code> mitgeliefert werden, sofern die Zahlung noch nicht fehlgeschlagen ist und damit der normale API-Ablauf mit <code>PayConfirm</code> erfolgen soll. Hinweis ergänzt bei <code>PayInit</code> , dass Redirects von Kunden zum Zahlungsdienstleister nicht via HTTP-Statuscode 307 erfolgen sollten. Konsistentere Nutzung von Basket-ID/Transaktions-ID statt Warenkorb-ID. Paypal bei Grobablauf-Sequenzdiagramm ergänzt.
2023/07/27	2.0.33	PD	Wording angepasst, Merchant wurde in Mandant umbenannt
2023/07/26	2.0.32	SG	SEPA <code>holder</code> Spezifikation bei <code>PayInit</code> konkretisiert. Direktzahlungen via SEPA ohne Zwischenseite sind nicht möglich. Errorcodes 421 und 424 aus Dokumentation entfernt, da nicht genutzt/unterstützt. Hinweise und

Datum	Version	Autor	Änderung
			Konkretisierungen bei Abschnitten Zwischenschritt, PayInit und Notify ergänzt.
2023/01/18	2.0.31	SG	Veraltete Testdaten für Kreditkarte entfernt. Allgemeine Hinweise ergänzt. Typos korrigiert. Konkretisierungen.
2022/10/11	2.0.30	SG	Testdaten für Zahlungen mit Kreditkarte aktualisiert.
2022/09/22	2.0.29	SG	Anpassungen Inhaltsverzeichnis/Überschriften und optisch konsistentere Ablaufgrafiken.
2022/09/21	2.0.28	SG	Abschnitt Tests hinzugefügt. Hinweis zu PENDING Transaktionen beim Ablauf API-Aufrufe ergänzt.
2022/08/26	2.0.27	SG	Inhaltsverzeichnis an aktuellen Aufbau des Dokuments angepasst. Kleine Änderungen an Hierarchie und Formatierung.
2022/07/28	2.0.26	SG	Ablauf von API-Aufrufen in Abhängigkeit von Rückgabewerten und Benachrichtigungen bzw. rückkehrenden Nutzern hinzugefügt
2022/05/09	2.0.25	SG	Zahlungsablauf für Zahlungsarten aus Kundensicht ergänzt. Formatierungen angepasst. Firmenadresse aufgrund Umzug angepasst.
2022/05/09	2.0.24	SG	Hinweise zum Umgang mit <code>status</code> -Parameter des <code>notifylink</code> ergänzt.
2022/05/09	2.0.23	SG	Testsystem ist ab sofort unter <code>https://test-secure.berlin.de/</code> verfügbar. Alte URL unter <code>test.secure.berlin.de</code> wird irgendwann abgeschaltet.
2022/05/05	2.0.22	SG	Allgemeine Hinweise zum API-Ablauf ergänzt und <code>status</code> -Werte beim <code>notifylink</code> -Aufruf korrigiert/ergänzt
2021/12/07	2.0.21	SG	Abschnitt zum Wartungsmodus überarbeitet
2021/09/17	2.0.20	SG	Typo gefixt und kleine Konkretisierung zu gewünschter zeitlicher Nähe von GetBasketId/PayInit Aufrufen
2021/08/18	2.0.19	StS	Neue Kreditkarten Testdaten ; Zahlungsmethode "Vorkasse" wird nicht mehr angeboten Präzisierung Error-Codes
2021/07/19	2.0.18	TA	Präzisierung GetBasketId ; Präzisierung Feld-Spezifikationen; Hinzufügen Sequenzdiagramm Zahlungsablauf; Umstellung Dokumentenformat
2021/03/17	2.0.17	StS	Klarstellung der Zeichensätze für <code>A</code> , <code>N</code> und <code>s</code> ; Neuer API ERROR 321 „no payment url“
2021/01/11	2.0.16	StS	PSD2.0 ist Pflicht, dazu Möglichkeit die Adresse des Karteninhabers mitzuschicken. <code>holder</code> Karten-Inhaber; <code>address</code> Adresse „streetnumber:zip:town:countrycode“; Testdaten für Fehlerfälle ergänzt

Datum	Version	Autor	Änderung
2020/03/18	2.0.15	StS	Giropay Test präzisiert.
2019/11/20	2.0.14	StS	Giropay Testdaten geändert (neue IBAN, optisches ChiptanVerfahren), Erklärung Wartungsmodus (Soft-Aus)
2019/09/10	2.0.13	StS	Klarstellung GiroPay Test-Zahlung
2019/07/03	2.0.12	StS	Test-Zugang für Paypal-Zahlung-Testen
2019/03/11	2.0.11	StS	Neue Test-Mastercard-Nummer mit 3d-secure
2019/03/07	2.0.10	StS	PayInit nun mit „notifylink“
2019/02/26	2.0.9	JB	Formatierungen angepasst
2019/02/22	2.0.8	AM	Adresse BerlinOnline geändert, Inhaltsverzeichnis aktualisiert
2019/02/01	2.0.7	StS	Begin Testbetrieb PayPal
2018/09/12	2.0.6	StS	Schreibweise nicht mehr „GiroPAY“ sondern „Giropay“
2017/10/05	2.0.5	StS	Klarstellung URL maxlen: 1024 Bytes
2016/12/14	2.0.4	StS	URL zum test.secure.berlin.de
2016/04/28	2.0.3	StS	Beim Testen von Giropay ist BIC Pflichtfeld
2016/04/12	2.0.2	StS	Neue Parameter „IBAN“ und „BIC“ für SEPA und Giropay; Testdaten für Giropay ergänzt
2016/01/27	2.0.1	StS	Ergänzungen für SEPA und Giropay-Zahlungen; Testdaten für alle Zahlarten
2015/06/15	2.0.0	StS	Anpassungen an zusätzliches Backend PayPlace von B+S; Umstellung „amount“ auf Cent (Integer-Wert); Error-Meldungen nun auch URL-kodiert für besseres Parsen
2015/03/20	1.3.3	StS	Ergänzung der Ergebnismeldungen/Antworten
2013/06/20	1.3.2	StS	neuer API-Call GetStatus , Antwort von PayInit präzisiert
2012/01/25	1.3.1	StS	Klarstellung des Workflow
2010/05/19	1.3.0	StS	neue Version unterstützt „trans_prefix“ Parameter (2 Zeichen), dadurch Verkürzung des Grund-Prefix auf zwei Zeichen.
2010/04/28	1.2.1	StS	Transaktions-ID Anfangskürzel auf 3 Zeichen erweitert (ges. 13 Zeichen); anzusprechender Payment-Server soll https://secure.berlin.de sein
2010/02/10	1.2.0	StS	Umstellung intern auf Java-Connector
2009/10/10	1.1.2	StS	Rückgabe-Werte verbessert für externe REST-API-Frameworks
2009/09/26	1.1.1	StS	Erweiterung der PayInit Methode um zusätzliche Parameter

Datum	Version	Autor	Änderung
2009/08/09	1.1	StS	Dokumentation angepasst auf die programmierte TestUmgebung, derzeit nur implementiert: REST-API
2008/09/26	1.0	StS	Spezifikation der Basisparameter

3. Begriffe

Übersetzung von Berlin.de-Vokabular in Landes-Verwaltungsbegriffe

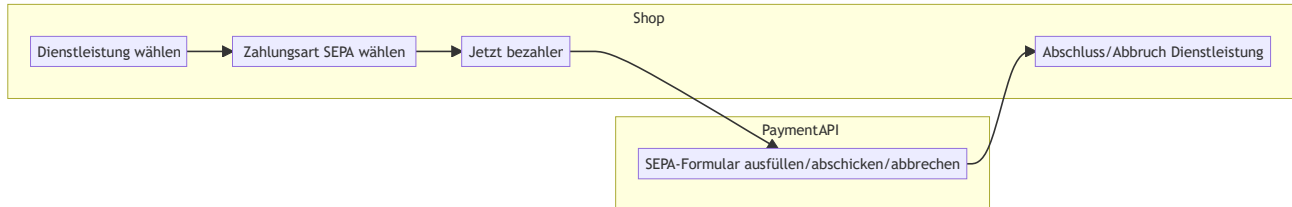
- Shop := System, welches Dienstleistungen eines Fachverfahrens im Internet abrufbar bzw. bezahlbar macht.
- Mandant := Kann pro Behörde oder auch pro Standort separat sein. Jeder Mandant hat eine 2 oder 3-stellige Kennung (Kürzel). Normalerweise entspricht ein Shop einem Mandanten in der abgesprochenen Konfiguration.
- Transaktion := Eine Zahlung bzw. ein Zahlversuch eines Kunden im Shop. Jeder Zahlversuch ist eine neue Transaktion. Liste der Transaktionen eines Mandanten ist in der Transaktionsdatenbank einsehbar.
- Transaktions-ID bzw. Basket-ID := ID einer Transaktion an der API zur Zahlungsabwicklung. Entspricht nicht der internen Warenkorb-ID eines Shops, sondern der ID des Zahlungsversuchs bei der Transaktionsdatenbank der Payment-API.
- Transaktionsdatenbank := Datenbank hinter der Payment-API mit Liste aller Transaktionen/Zahlversuche.
- trans_prefix := Kennung zur Unterscheidung von Submandanten eines Mandanten. Wird genutzt, um für einen Shop mehrere Konfigurationen bereitzustellen (unterschiedliche Kassenzeichen, Limits etc); siehe [GetBasketId](#).

4. Zahlungsablauf

4.1 Zahlungsarten aus Kundensicht

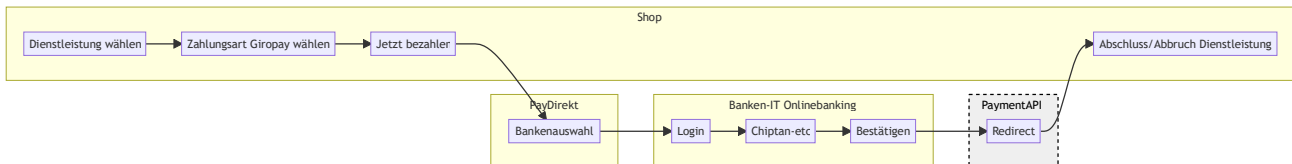
4.1.1 SEPA aus Sicht der Nutzenden

Auswahl im Shop => SEPA Dialog beim ePayment => Abschluss im Shop



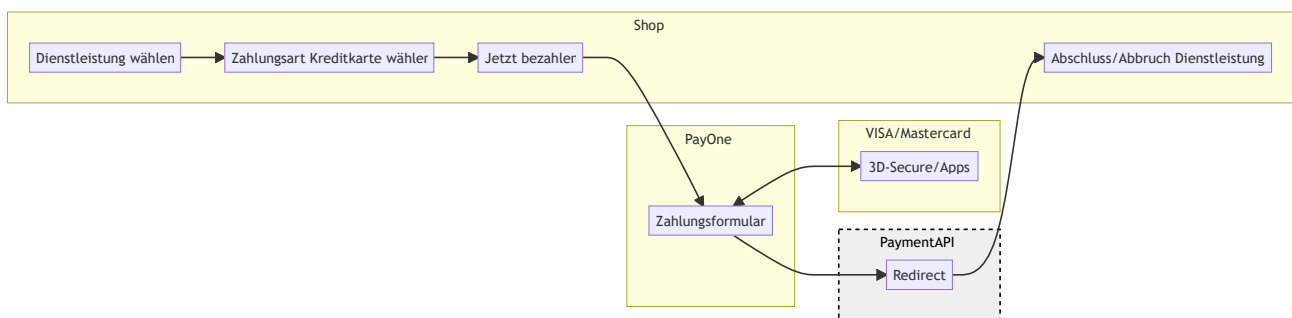
4.1.2 Giropay aus Sicht der Nutzenden

Auswahl im Shop => Bankenauswahl bei PayDirekt => Login/Chiptan-etc/Bestätigung bei teilnehmender Bank => Redirect über ePayment => Abschluss im Shop



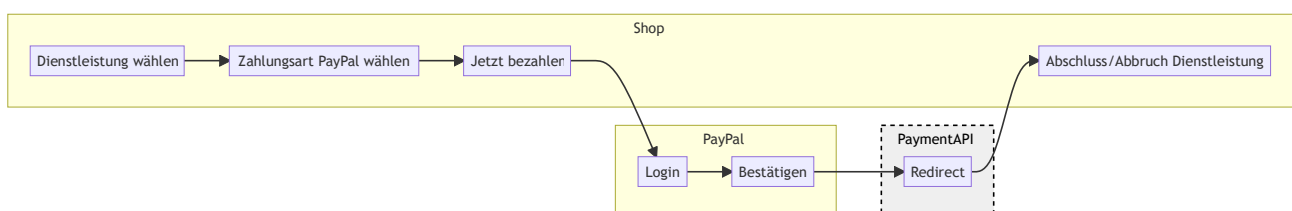
4.1.3 Kreditkarte aus Sicht der Nutzenden

Auswahl im Shop => Zahlungsformular bei PayOne => ggfs. 2FA bei VISA/Mastercard => Redirect über ePayment => Abschluss im Shop



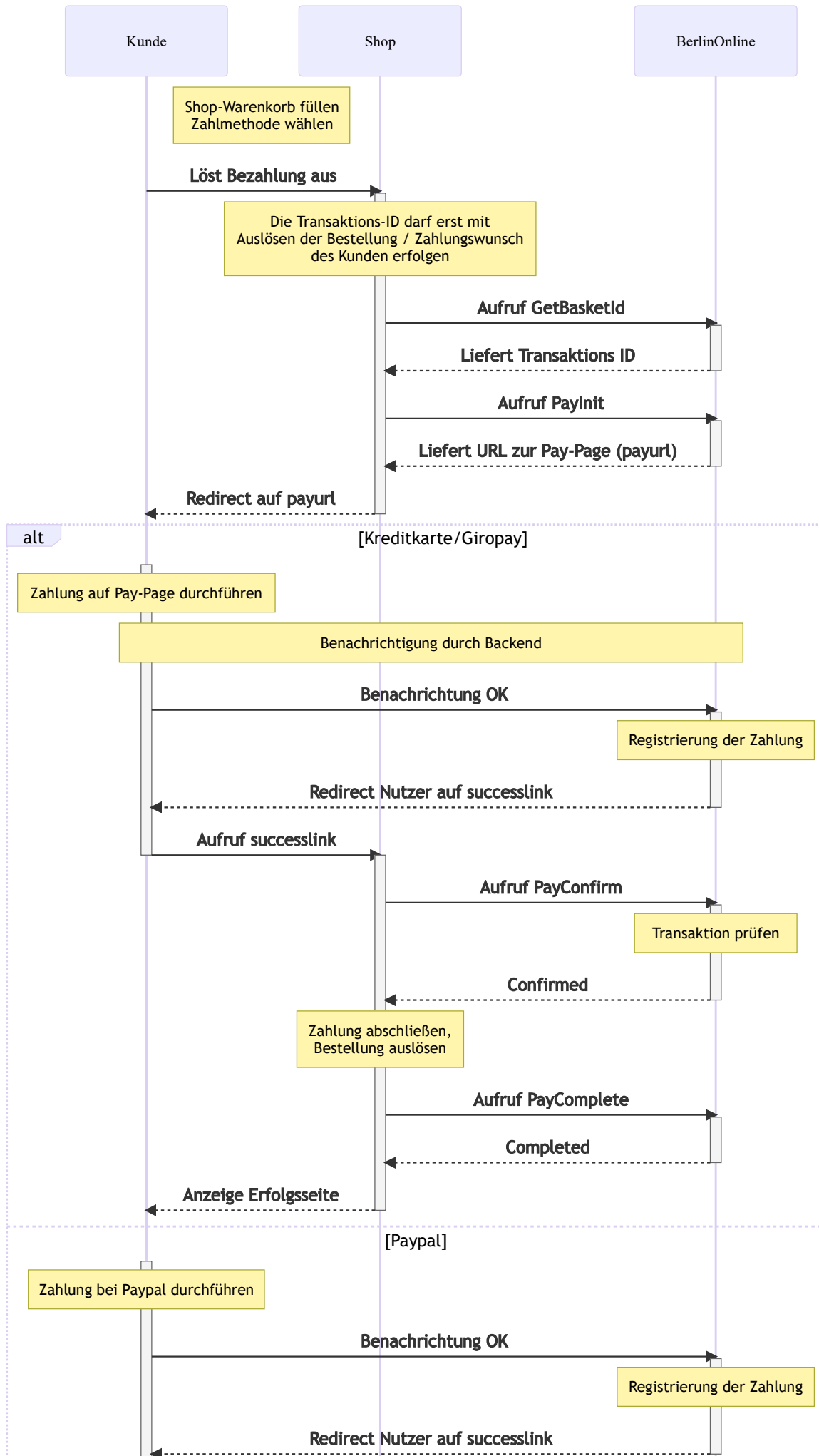
4.1.4 PayPal aus Sicht der Nutzenden

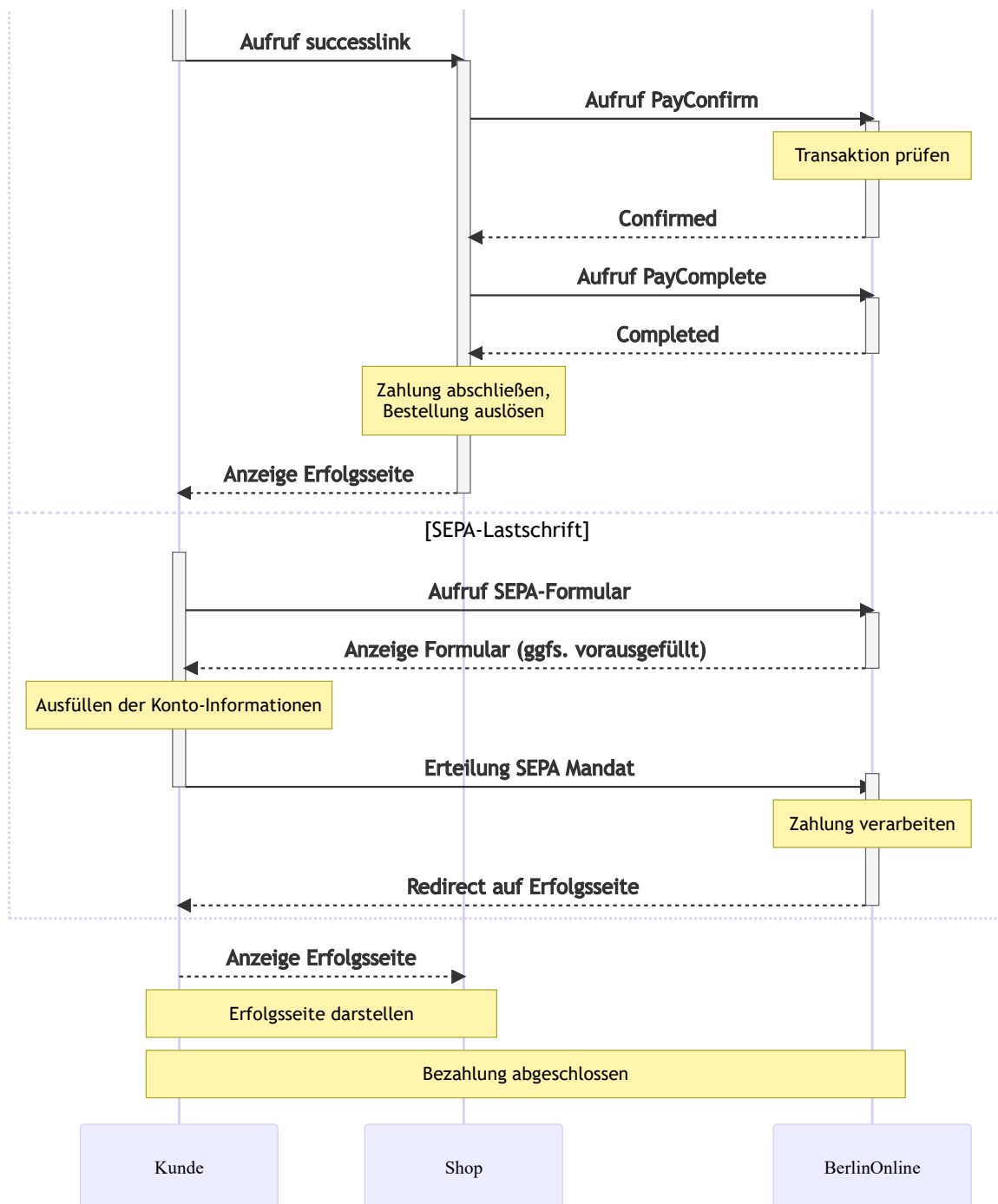
Auswahl im Shop => Login/Bestätigung bei PayPal => Redirect über ePayment => Abschluss im Shop



4.2 Grober Ablauf von Aufrufen zwischen Shop, Kunde und API im Erfolgsfall

Folgender Ablauf stellt exemplarisch die erfolgreiche Durchführung von Zahlungen durch Nutzende dar. Fehlerfälle und -Zustände werden hierbei nicht betrachtet.





5. Abfrage der Berlin.de-Payment-Schnittstelle

5.1 Technische Vorbedingungen

Beim Berlin.DE-Payment wird für jeden Shop fest hinterlegt:

- http-Authentisierung (HTTP Basic Auth)
- interne auth-Authentifizierung (Parameter via HTTP POST)
- interne Shop-Daten für Zahlungsabwicklung und Reporting (Festlegung der erlaubten Zahlungsarten usw.)
- OK-URL (`successlink`): welche URL soll aufgerufen werden, um dem Shop die Kundenbestätigung der Zahlung mitzuteilen (kann auch pro Transaktion beim `PayInit` -Request übermittelt werden)
- FAIL-URL (`faillink`): welche URL vom Shop soll aufgerufen werden, um dem Shop einen Fehler mitzuteilen (kann auch pro Transaktion beim `PayInit` -Request übermittelt werden)
- NOTIFY-URL (`notifylink`): Server-to-Server Information nach Beendigung der Zahlung direkt an den Shop des Fachverfahrens, ohne Einbindung des Nutzers (kann auch pro Transaktion beim `PayInit` -Request übermittelt werden)
- Bei Benutzung des Kontos der Landes-Hauptkasse Hinterlegung eines "Kassenzeichens" zur Übermittlung und Zuordnung (1 pro Mandant/Submandant)
- Bei Nutzung der Zahlart "SEPA" muss eine Gläubiger-ID hinterlegt werden.

Werden unterschiedliche Konfigurationen für den Shop/Mandanten benötigt (verschiedene Gläubiger-Ids, Kassenzeichen etc.), müssen diese abgesprochen und von BerlinOnline eingerichtet werden. Der Shop wählt die entsprechende Konfiguration für seine Submandanten via `trans_prefix` beim `GetBasketId` aus. Der Shop/Mandant hat in Verträgen festgelegte Referenzkonten, auf denen das Geld erfolgreicher Transaktionen letztendlich eingeht.

5.2 Allgemeine Hinweise

Die für einen Shop/Mandanten verfügbaren Zahlarten hängen von der Vertragsgestaltung und Konfiguration ab. Nur die mit BerlinOnline abgesprochenen Zahlarten sind an der Payment-API zu verwenden bzw. verwendbar.

Die API bietet derzeit keine Paypage an, auf der Kunden eine der mit dem Mandanten abgesprochenen Zahlarten auswählen kann. Die Auswahl der angebotenen und abgesprochenen Zahlarten obliegt dem Shop.

Es gibt keine Warenkorbfunktionalität an der Payment-API. D.h. ob eine Kundentransaktion via API einer oder mehreren Leistungen (Betrag als Summe) entspricht, hängt von der Implementierung des Shops ab.

Es ist dringend empfohlen das Feld `customerid` (Kundenreferenz) beim `PayInit` sinnvoll zu nutzen, um Payment-API Transaktionen zu kennzeichnen. Nach dieser Id kann in der Transaktionsdatenbank gesucht werden. Shops oder nachgelagerte Fachverfahren können damit aufgrund eigener Ids Zahlungen gegebenenfalls einfacher zuordnen.

Die Payment-API ist bei der Transaktionsabwicklung nicht daran interessiert personenbeziehbare oder sensible Daten zu verarbeiten. Achten Sie darauf bitte insbesondere bei Warenkorbtext und Kundenreferenz.

5.3 Anfragen an die API-Funktionen

API-Aufrufe erfolgen durch die Shops der Fachverfahren.

Für alle API-Aufrufe ist die HTTP-Methode `POST` zu benutzen.

Aufgerufen werden die Funktionen einzeln, verschlüsselt durch eine TLS-Verbindung, gesichert durch die http-Authentisierung und zusätzlich durch die auth-Authentifizierung in den POST-Parametern.

Die Anmeldedaten (http und auth) werden separat pro Shop vergeben.

Allgemein hat der Aufruf einer Funktion folgendes Aussehen:

```
https://secure.berlin.de/payment/v20/FunktionsName.php
```

Das Test-System ist erreichbar unter der URL

```
https://test-secure.berlin.de/payment/v20/FunktionsName.php
```

Auf dem Test-System funktionieren nur die Test-Kennungen, bitte kontaktieren Sie uns.

Allgemeine Parameter

Parameter	Wertebereich	Beispiel	Erläuterung
auth_id	AN	an49KZ12nK	dem Shop fest zugeordnete ID zur Authentifizierung
auth_pw	ANS	S/cher12%3	Kennwort

5.4 Antworten der API-Funktionen

Eine *erfolgreiche Antwort* besteht immer aus:

```
OK Weiterer_Text_oder_URL
```

Weiterer_Text_oder_URL = alle Informationen werden wie ein Query-String URL-kodiert übermittelt
name1=val1&name2=val2&name3=val3&... so dass Sie sie mit Standardfunktionen in jeder Programmiersprache parsen können.

Beispiel:

```
OK id=ABC1234567890
```

Eine *Fehler-Antwort* ist ebenfalls URL-kodiert und hat immer die Form:

```
ERROR error=NNN&errortext=Fehlermeldung
```

NNN ist ein dreistelliger Code. Fehlermeldung ist standardisiert (siehe [Ergebnismeldungen / Antworten](#))

Beispiel:

```
ERROR error=223&errortext=missing%20or%20wrong%20validation%20date
```

Generell können ab Zeile zwei zusätzliche Informationen beschreibender Natur mitgeliefert werden, die aber für eine Auswertung der Nachricht nicht erforderlich sind, diese sind dann auch als Fließtext zu lesen und nicht speziell kodiert. Alles nach Zeile 1 ist nicht für maschinelle Entscheidungen oder eine sonstige Weiterverwendung gedacht.

Achtung: Bei OK Parametern bitte nicht ein eventuell vorhandenes Zeilenende-Zeichen weiterverarbeiten.

6. API-Funktionen

Der generelle Reihenfolge der API-Aufrufe von Shops ist wie folgt:

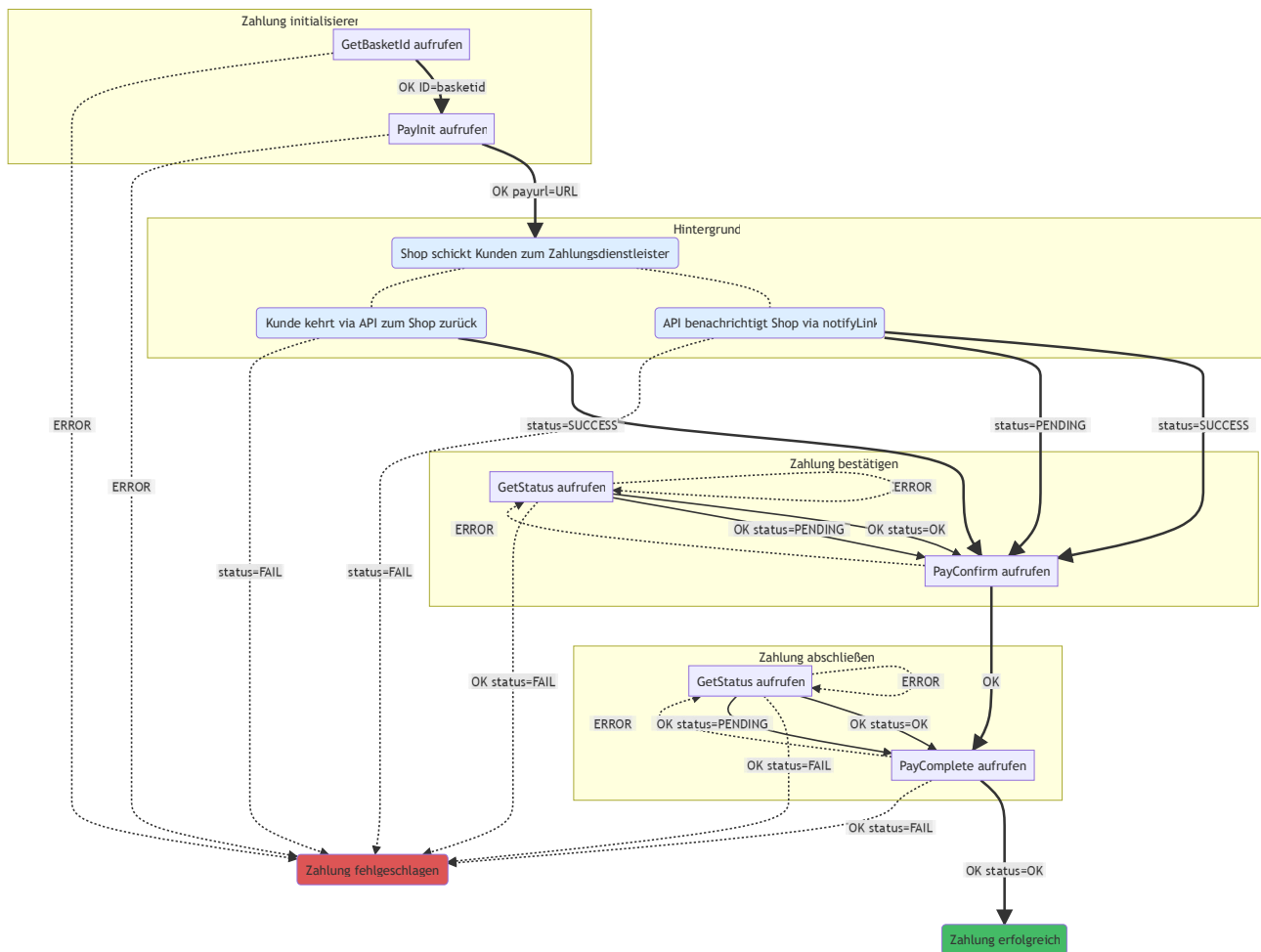


Der Shop muss im Fall einer erfolgreich bestätigten Zahlung alle diese Schritte durchlaufen. Es wird zunächst eine Basket-ID (Transaktions-ID) abgeholt und anschließend die Zahlung initialisiert (mit notwendigen Zahlungsdaten). Dieser `PayInit`-Aufruf liefert dem Shop eine URL für Kunden zurück. Der Shop schickt Kunden zu dieser URL via Redirect oder Link. Nachdem Kunden erfolgreich beim Zahlungsdienstleister Zahlungen angestoßen haben, kommen die Kunden im Normalfall via Redirect-Kette im Browser vom Zahlungsdiensteanbieter über die API zum Shop zurück (zum `successlink`; alternativ `faillink`). Die API kann den Shop zusätzlich im Hintergrund über den Ausgang einer Zahlung benachrichtigen (siehe `notifylink`). Der API-Call `GetStatus` ist ab einer gewissen Zeit nach `PayInit` sinnvoll. Bei Zahlungen mit Vorauthorisierung ist ohne erfolgreichen Aufruf von `PayComplete` die Zahlung nicht komplett.

Achtung: Aufgrund der Erfahrung mit unterschiedlichen Implementationen/Konfigurationen ist es stets empfehlenswert die Funktion `GetStatus` zu verwenden, um sich über den Ausgang einer Zahlung zu vergewissern.

6.1 Ablauf API-Aufrufe

Darstellung des Ablaufs der Aufrufe der API-Endpunkte in Abhängigkeit von Rückgabewerten und Benachrichtigungen bzw. rückkehrenden Nutzern.



Achtung: Es kann passieren, dass Kunden nicht zum E-Payment und damit auch nicht zum Shop zurückkehren (Inaktivität, Netzwerkprobleme, Browser am Ende der Bezahlung zu schnell geschlossen, Redirect-Ketten unterbrochen, Geräteabsturz etc.). In diesen Fällen ist der Ausgang des Bezahlvorgangs der E-Payment API nicht direkt bekannt. Es ist unbekannt, ob Kunden überhaupt zum Zahlungsdienstleister (oder gar der dahinterliegenden Bank) kamen. Daraus folgt, dass sowohl die E-Payment API als auch der Shop für einen gewissen Zeitraum eine `PENDING` Transaktion sehen. `PENDING` ist der normale Status einer Transaktion bis zur Entscheidung über den erfolgreichen oder fehlgeschlagenen Ausgang eines Zahlungsveruchs. Aufgrund von Abfragen im Hintergrund an die Zahlungsdienstleister wird der Zustand der Transaktion erst nach einiger Zeit auf `FAIL` oder `OK` gehen.

6.2 GetBasketId

Erster Schritt generiert eine gültige Basket-ID (Transaktions-ID) im Backend.

GetBasketId ist erst mit dem tatsächlichen Auslösen der Bestellung durch Kunden aufzurufen. Also zum Beispiel nach dem Klicken eines Buttons wie *Zahlungspflichtig bestellen*. Damit soll sichergestellt werden, dass alle für die Bestellung und Bezahlung notwendigen Daten vorliegen und die Zahlung in minimaler Zeit abgewickelt werden kann.

Neben den obligatorischen Parametern zu Authentifizierung (u.a. auth_id und auth_pw im POST-Body) unterstützt die API-Methode GetBasketId einen optionalen Parameter trans_prefix.

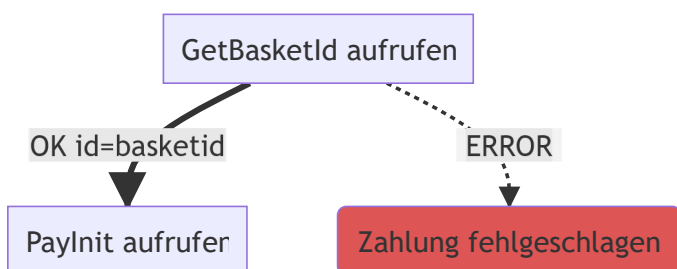
Parameter	Wertebereich	Beispiel	Erläuterung
trans_prefix	A-2	GA	2-stellige Kennung eines Submandanten für Mandanten mit 2-stelligen Kennung (Kürzel). Bei einem Mandanten mit 3-stelliger Kennung (Kürzel) wird der Parameter nicht verwendet.

Rückgabe:

```
OK id=basketid
```

Die basketid (immer 13 Zeichen AN-13) und hat derzeit den folgenden Aufbau:

- Mandant mit 3-stelligem Kürzel; „ohne trans_prefix“
 - 3 Zeichen Kürzel für den Mandanten (dient der Zuordnung der hinterlegten Kontoverbindung)
 - 10 Zeichen (derzeit aus zufälligen Ziffern)
 - AAANNNNNNNNN
- Mandant mit 4-stelligem Kürzel; „mit trans_prefix“
 - 2 Zeichen Kürzel für den Mandanten (dient der Zuordnung der hinterlegten Kontoverbindung)
 - 2 Zeichen Trans-Prefix eines Submandanten (Unterzuordnung des Mandanten zur Verteilung; Bsp.: Bezirkskürzel)
 - die restlichen 9 Zeichen (gesamt 13 Zeichen) werden derzeit mit einer Zufallszahl aus Ziffern aufgefüllt.
 - AAAANNNNNNNNN



6.3 PayInit

Übergabe aller gewünschten Parameter des Warenkorbs. Der Aufruf von `PayInit` sollte direkt im Anschluss an `GetBasketId` erfolgen. Bitte vermeiden Sie personenbeziehbare oder sensible Daten in Warenkorbbeschreibung oder Kundenreferenz.

`PayInit` initialisiert eine Transaktion an der Payment API. Eine Transaktion besteht mindestens aus `basketid`, `paymethod`, `amount`, `currency`, `langid` und `description`. Empfohlen ist zusätzlich stets eine Kundenreferenz via `customerid`. Sollten keine allgemein gültigen Rücksprung-URLs mit BerlinOnline für den Shop konfiguriert worden sein, muss mindestens `successlink` übergeben werden. Möchte der Shop unabhängig von rückkehrenden Kunden über den Zahlungsfortschritt informiert werden, sollte der `notifylink` übergeben werden.

Zusätzlich zu den allgemein notwendigen Feldern gibt es zusätzliche Felder je nach Zahlart. Es folgt eine Referenzliste aller Parameter. Im Anschluss werden die Parameter pro unterstützter Zahlart zusammengefasst.

6.3.1 Liste unterstützter PayInit-Parameter

Im Folgenden eine Liste aller unterstützten Parameter mit ihren Wertebereichen (unabhängig von gewählter Zahlungsmethode).

Parameter	Wertebereich / Pflicht	Beispiel	Erläuterung
<code>basketid</code>	AN-13	ZZZ1234567890	13-stellige Basket-ID/Transaktions-ID, aus dem Schritt GetBasketId
<code>amount</code>	N-8	1492	ganzzahlige Zahl-Beträge in der kleinsten Einheit der Währung (Beispiel: 14.92 Euro werden zu „1492“). Keine Tausenderzeichen, oder Kommas erlaubt. Maximalbetrag pro Shop ist einstellbar.
<code>currency</code>	AN-3	EUR	Pflichtfeld: Währung, derzeit nur EUR möglich (ISO 4217)
<code>description</code>	ÄäNLS-120	Plakette Nummernschild	Beschreibender Text des Warenkorbs. Die erste Zeile wird an einigen Stellen hervorgehoben dargestellt und sollte auch auf dem Kontoauszug sofort zu erkennen geben, wofür Geld abgezogen wurde. Bitte keine

Parameter	Wertebereich / Pflicht	Beispiel	Erläuterung
			personenbeziehbaren oder Daten mit besonderem Schutzbedarf verwenden. Zeichensatz für alle Parameter ist UTF-8.
kunde	ÄäNLS-120	–	<i>abgekündigt/deprecated</i> – wird nicht verarbeitet
paymethod	[ZAHLUNGSART]	Kreditkarte	Gewünschte Zahlungsart. Derzeit verfügbar: Kreditkarte , SEPA , Paypal , Giropay . Die Auswahl der Zahlart muss vom Shop im Vorfeld abgefragt werden.
langid	AN-2	de	Sprache des angezeigten Backends voreinstellen, derzeit unterstützt (de , en , fr , it , ru)
successlink	URI?-1024	https://abc.de/dl/return.php?orderid=1234	Allgemeiner Rück-Link, zu dem Kunden nach Freigabe einer Zahlung geschickt werden. Wird um URL-Parameter wie basketid und status erweitert. Aufruf des successlink mit status=SUCCESS bedeutet, dass normal mit den API-Aufrufen fortgefahren werden soll - deshalb werden die Parameter data und signature mitgeliefert, damit mit PayConfirm fortgefahren werden kann. Der mitgelieferte Status ist semantisch NICHT der Status der Transaktion. Übergibt man keinen faillink , so kann der übergebene Status auch status=FAIL sein - siehe faillink Dokumentation.
backlink	URI?-1024	https://abc.de/dl/return.php?orderid=1234	<i>abgekündigt/deprecated</i> Voreinstellung ist der successlink . Abbruch durch den User.

Parameter	Wertebereich / Pflicht	Beispiel	Erläuterung
			Rückgabe erfolgt mit zusätzlichem Parameter <code>status=BACK</code> .
faillink	URI?-1024	<code>https://abc.de/dl/return.php?orderid=1234</code>	Voreinstellung ist der <code>successlink</code> . Aufruf erfolgt im Falle einer fehlgeschlagenen Freigabe der Zahlung durch Kunden oder eines Abbruchs oder eines Fehlers bei Hintergrundsystemen. Rückgabe erfolgt mit Parametern <code>basketid</code> und <code>status=FAIL</code> .
notifylink	URI?-1024	<code>https://abc.de/dl/notify.php?orderid=1234</code>	Über den Notify-Link wird der Shop des Fachverfahrens Server-to-Server über den Ausgang/Status der Zahlung informiert. Die URL wird um diverse Parameter wie <code>basketid</code> und <code>status</code> erweitert. Der Parameter <code>status</code> kann <code>SUCCESS</code> , <code>FAIL</code> oder <code>PENDING</code> sein. Die Parameter <code>data</code> und <code>signature</code> werden geliefert, wenn mit <code>PayConfirm</code> fortgefahren werden kann. Der mitgelieferte Status ist semantisch NICHT der Status der Transaktion. Per GetStatus kann der tatsächliche Status der Transaktion abgefragt werden.
customerid	AaN[-]-32	CUST-1234567890	selbst gewählter Identifier zur Zuordnung dieser Zahlung im Shop oder Fachverfahren (für z.B. den internen Vorgang des Kundens im Fachverfahren oder eine Warenkorb-ID). Kann insbesondere bei Supportfällen und Kundenanfragen sehr hilfreich sein und wird daher explizit empfohlen zu

Parameter	Wertebereich / Pflicht	Beispiel	Erläuterung
			übergeben. Die <code>customerid</code> kann in der Transaktionsdatenbank gesucht werden.
sepamandatreference	AN[_]-35	Kunde123-Warenkorb3712	Mandatsreferenz; (bei SEPA) Es wird dringend empfohlen eine eigene Mandatsreferenz zu übermitteln, um Kundenanfragen im Shop zu Zahlungen einfacher bearbeiten zu können. Wenn keine Mandatsreferenz übergeben wurde, wird automatisch eine anhand der <code>basketid</code> gebildet.
iban	AN34	DE00123456781234567890	IBAN; (bei SEPA, Giropay)
bic	AN11	TESTDETT421	BIC Bankcode; (bei SEPA, Giropay)
holder	AÄäNLS-27	Max Mustermann	Kontoinhaber; (bei SEPA, Giropay)
address	ÄäNL[-/().,&']-50 „:“ ANL-11 „:“ ÄäNL[-/().,&']-50 „:“ A2	Alte Jakobstr. 105:10969:Berlin:DE	<i>abgekündigt/deprecated</i> Adresse des Karteninhabers, um die erneute Abfrage bei der Bezahlung wegen PSD2.0 zu vermeiden. Format: <code>StrasseUndNr:PLZ:Ort:Landescode</code> also „Strasse Nr“ und „PLZ“ und „Ort“ mit Doppelpunkt voneinander getrennt. Nur vollständige Adressen akzeptiert. StrNr und Stadt je maximal 50 Zeichen, PLZ in D genau 5 Ziffern, Landescode genau 2 Zeichen (ISO-3166); Bitte die Einzelfelder nutzen.

Parameter	Wertebereich / Pflicht	Beispiel	Erläuterung
addrcity	AÄääNLS-50	Berlin	Stadt der Adresse der zahlenden Person; (bei Kreditkarte, Giropay)
addrcountry	A2	DE	Zweibuchstabiges Länderkürzel nach ISO 3166 der Adresse der zahlenden Person; (bei Kreditkarte, Giropay)
addrpostcode	ANL[-]-11	10969	Postleitzahl der Adresse der zahlenden Person; (bei Kreditkarte, Giropay)
addrstreet	AaÄäänLS-30	Alte Jakobstr.	Straße der Adresse der zahlenden Person; (bei Kreditkarte, Giropay)
addrstreetnumber	ANLS-8	105	Hausnummer der Adresse der zahlenden Person; (bei Kreditkarte, Giropay)
shipcity	AÄääNLS-50	Berlin	Ortsname der Versandadresse; (bei Kreditkarte)
shipcountry	A2	DE	Zweibuchstabiges Länderkürzel nach ISO 3166 der Versandadresse; (bei Kreditkarte)
shippostcode	ANL[-]-11	10969	Postleitzahl der Versandadresse; (bei Kreditkarte)
shipstreet	AaÄäänLS-30	Alte Jakobstr.	Straße der Versandadresse; (bei Kreditkarte)
shipstreetnumber	ANLS-8	105	Hausnummer der Versandadresse; (bei Kreditkarte)
payerfirstname	AÄääNLS-30	Susi	Vorname der zahlenden Person; (bei Giropay; ggfs SEPA)
payerlastname	AÄääNLS-30	Sorglos	Nachname der zahlenden Person; Giropay, ggfs SEPA)

Parameter	Wertebereich / Pflicht	Beispiel	Erläuterung
payeremail	AaÄäN[_.+ -@]-254	Susi.Sorglos@example.com	E-Mail-Adresse der zahlenden Person; (bei Kreditkarte)
payerphone	N[+]-20	+4930123456789	Telefonnummer (Zuhause) der zahlenden Person; (bei Kreditkarte)
payerphonemob	N[+]-20	+4930123456789	Telefonnummer (mobil) der zahlenden Person; (bei Kreditkarte)
payerphonework	N[+]-20	+4930123456789	Telefonnummer (Arbeit) der zahlenden Person; (bei Kreditkarte)

Die Kombination notwendiger Felder beim `PayInit` ergibt sich nach gewählter Zahlungsmethode.

6.3.2 Notwendige Zahlungsangaben bei PayInit

Folgende Parameter sind bei Zahlungsinitialisierungen via `PayInit` global immer zu nutzen bzw. nutzbar:

- **Zahlungsdaten** (Pflichtfelder zur Zahlung)
 - basketid
 - paymethod
 - amount
 - currency
 - description
 - langid
- **Kundenreferenz** (optional, aber empfohlen)
 - customerid
- **Shopdaten**
 - successlink (Pflicht)
 - faillink
 - notifylink

Zusätzlich zu diesen notwendigen Feldern kommen je nach gewählter Zahlart weitere Felder hinzu. Diese sind in den folgenden Abschnitten beschrieben.

6.3.3 PayInit-Parameter für Giropay / Paydirekt

Zusätzlich zu den notwendigen Zahlungsangaben bei `PayInit` sind bei Giropay-Transaktionen folgende Felder nutzbar:

- **Adressangaben** zur Adresse der zahlenden Person:
 - payerfirstname

- payerlastname
- addrpostcode
- addrcity
- addrcountry

- **Kontoangaben** zum Konto der zahlenden Person

- iban
- bic
- holder (wird mit payerfirstname + payerlastname befüllt, falls angegeben)

Ab dem 01. Januar 2024 sind laut Paydirekt Adressangaben der zahlenden Personen Pflicht. Bei Angabe eines Felds aus der Gruppe der Adressangaben *müssen* auch alle anderen Felder dieser Gruppe angegeben werden (Keiner- oder Alle-Prinzip). D.h. es reicht nicht aus, nur den Namen des Zahlenden oder nur die Adresse des Zahlenden anzugeben. Die Kontoangaben sind optional.

Aus den Giropay-Anforderungen ergeben sich folgende Kombinationen für valide `PayInit` -Requests:

1. [notwendige Zahlungsangaben]
2. [notwendige Zahlungsangaben] + [Adressangaben] (empfohlen)
3. [notwendige Zahlungsangaben] + [Adressangaben] + [Kontoangaben]
4. [notwendige Zahlungsangaben] + [Kontoangaben]

Aufgrund von Ankündigungen von Paydirekt ergibt sich ein hohes Risiko für fehlschlagende Zahlungen aufgrund fehlender übermittelter Daten zur zahlenden Person ab dem 01. Januar 2024. **D.h. ab dem Jahr 2024 sind die zweite und dritte Variante die einzig sinnvollen Feldkombinationen bei Giropay-Zahlungsinitialisierung.**

Die obigen Adress-Felder ersetzen den abgekündigten (veralteten/deprecated) Parameter `address`. Eine Umstellung auf die Einzelparameter ist empfohlen.

6.3.4 PayInit-Parameter für Kreditkarte

Zusätzlich zu den notwendigen Zahlungsangaben bei `PayInit` sind bei Kreditkarten-Transaktionen folgende Felder zum Zweck der Transaktionsprüfung durch VISA/Mastercard nutzbar:

- **Adressangaben** zur Adresse der zahlenden Person (falls verwendet, müssen alle Felder verpflichtend gefüllt werden):
 - addrstreet
 - addrstreetnumber
 - addrpostcode
 - addrcity
 - addrcountry
- **Kontaktdaten** der zahlenden Person (zusätzlich zu Adressangaben; nicht alle Felder müssen verwendet werden, aber eine Mindest-Angabe von Email und einer Telefonnummer ist bei Verwendung notwendig)
 - payeremail
 - payerphone
 - payerphonemob

- payerphonework
- **Versandadresse** (falls bekannt und verwendet, müssen alle Felder verpflichtend gefüllt werden)
 - shipstreet
 - shipstreetnumber
 - shippostcode
 - shipcity
 - shipcountry

Achtung: Wenn beim `PayInit` Felder zu Kontaktdaten (Email und Telefonnummer) oder eine Versandadresse übergeben werden, sind die Adressangaben Pflicht.

Die Anforderungen zu Kreditkartenzahlungen bei Prüfungen nach 3-D Secure 2.0 sind unterschiedlich. Es ist daher empfehlenswert, alle im Shop bekannten bzw. hinterlegten Daten zur zahlenden Person zu übergeben, da dies die Ablehnungsrate von Kreditkartenzahlungen positiv beeinflussen wird.

Aus den Anforderungen für Kreditkarten ergeben sich folgende Kombinationen für valide `PayInit` - Requests:

1. [notwendige Zahlungsangaben]
2. [notwendige Zahlungsangaben] + [Adressangaben]
3. [notwendige Zahlungsangaben] + [Adressangaben] + [Kontaktdaten] (empfohlen)
4. [notwendige Zahlungsangaben] + [Adressangaben] + [Kontaktdaten] + [Versandadresse] (empfohlen, falls Daten vorhanden bzw. bekannt)

Hinweis Für Zahlungen mit VISA-Karte wird ab dem 12. Februar 2024 vorausgesetzt, dass Adressangaben *und* Kontaktdaten übergeben werden. Dabei wird mindestens die Emailadresse *und* eine Telefonnummer der zahlenden Person erwartet. Bei Mastercard sind ähnliche Schritte für Pflichtfelder geplant (plus Versandadresse), aber noch nicht terminiert. **D.h. ab Mitte Februar 2024 sind die dritte und vierte Variante die einzig empfohlenen Feldkombinationen bei Kreditkarten-Zahlungsinitialisierung.** PayOne hat angekündigt, auf der PayPage für Kreditkarten die ggfs. fehlenden zusätzlichen Adress- und Kontaktdaten abzufragen, falls der Shop diese nicht übergibt.

Die obigen Felder zur Adressangabe ersetzen den abgekündigten (veralteten/deprecated) Parameter `address`. Eine Umstellung auf die Einzelparameter ist empfohlen.

6.3.5 PayInit-Parameter für SEPA-Lastschrift

- **Mandatsreferenz** (optional, aber empfohlen)
 - `sepamandatereference`
- **Kontoangaben** (optional; nützlich zur Vorbelegung der Felder auf der SEPA-PayPage)
 - `iban`
 - `bic`
 - `holder` (oder `payerfirstname` + `payerlastname`)

Die Kontoangaben sind optional, da diese auf der PayPage durch Kunden mindestens bestätigt oder komplett eingegeben werden müssen. Für die Mandatsreferenz (`sepamandatereference`) empfehlen wir dringend die Festlegung und Übergabe durch den Shop, damit bei Supportfällen durch Kunden die Mandatsreferenz zur Transaktion auch beim Shop bekannt ist. Wird keine Mandatsreferenz übergeben,

wird ohne Kenntnis des Shops automatisch eine Mandatsreferenz für die Zahlung aus `basketid` und einem Zufallswert generiert.

6.3.6 PayInit-Parameter für Paypal

Paypal benötigt neben den notwendigen Zahlungsangaben keine zusätzlichen Felder.

6.3.7 PayInit-Antwort bzw. Response

Rückgabe im Erfolgsfall:

```
OK payurl=URL_Zur_ZahlungsSeite
```

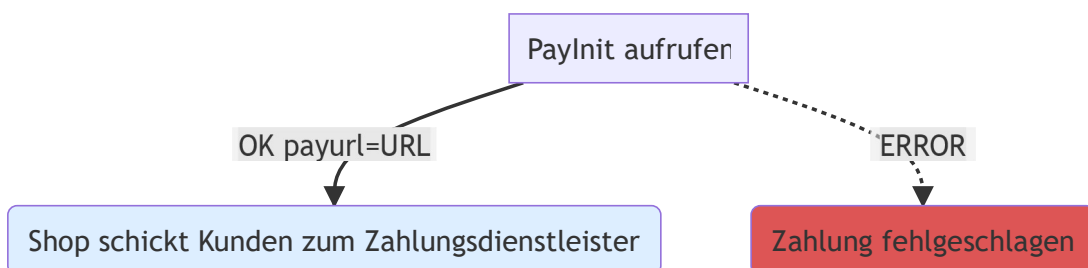
Die `payurl` ist die URL bei der Kunden die Zahlung in der beim `PayInit` übergebenen Zahlart freigeben/bestätigen können.

Der Shop muss die von `PayInit` zurückgegebene URL dem Kunden als Redirect oder Link präsentiert werden. Im Normalfall verwenden Shops Redirects mit HTTP-Statuscode 303 (oder historisch 302). Kunden landen dann beim Zahlungsdienstleister via GET-Request. Redirects von Kunden via HTTP-Statuscode 307 sind NICHT zulässig, da z.B. die Bankenauswahlseite bei Giropay-Zahlungen keine POST-Requests akzeptiert (durch weitergeleitete Kunden).

Anmerkung `description`: Der Wertebereich für die Ausgabe auf dem Kontoauszug entspricht den im Datenträgeraustausch-Format (DTAUS-Format) festgelegten Zeichen: Buchstaben (mit den Umlauten Ä, Ö, Ü, ä, ö, ü sowie ß), Ziffern, Leerzeichen, ".", ",", "&", "-", "+", "*", "%", "/", "'", "\$".

Bei SEPA-Zahlungen muss mit BerlinOnline abgesprochen werden, wie der Kunde die Mandats-Referenz mitgeteilt bekommt. Für SEPA können `iban`, `sepa Mandatsreferenz`, `holder` und ggfs. `bic` (nur notwendig bei Konten außerhalb von Deutschland und Österreich) mitgeschickt werden.

Gibt der `PayInit`-Aufruf einen Fehler zurück via `ERROR error=NNN&error text=...`, ist die Initialisierung der Transaktion bzw. Zahlung fehlgeschlagen. Die Transaktion bzw. der Zahlversuch ist dann fehlgeschlagen (Status `FAIL` bei `GetStatus`) und der Shop kann einen neuen Zahlversuch mit einer neuen Transaktion (via `GetBasketId`) beginnen. `PayInit` mehrfach aufzurufen ist normalerweise nicht zulässig.

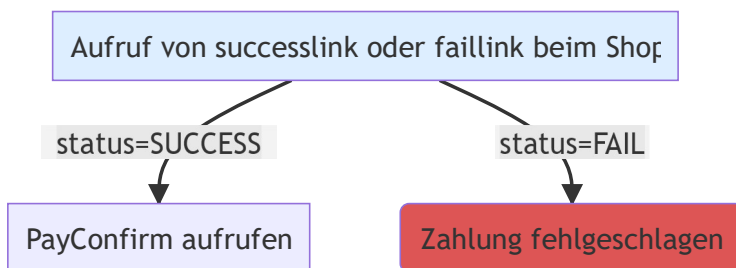


Zwischenschritt

Die beim `PayInit` zurückgelieferte URL müssen Shops ihren Kunden an dieser Stelle als Redirect (z.B. HTTP-Statuscode 303) oder zum Beispiel auf einer Webseite als Link präsentieren (Kurze Gültigkeit des Links beachten), damit diese mittels dieser URL den eigentlichen Bezahl-Vorgang anfangen können. Wenn der Nutzer mit der Bezahlung fertig ist (oder abbricht) kommt er zuerst zurück zur Payment-API, von

der er an die von Shops zur Verfügung gestellten Rück-URLs weitergeleitet wird. Im OK-Fall (Zahlungsfreigabe durch Kunden) erfolgt nach Redirect zum Berlin.de-Payment ein Redirect zur `successlink`-URL und im Abbruch- oder Fehlerfall zum `faillink` (ggfs. `successlink`, falls `faillink` nicht übergeben wurde). Zusätzlich zum zurückkehrenden Kunden kann es eine Server-to-Server Notifikation geben, falls der Shop eine `notifylink`-URL hinterlegt hat. Alle weiteren Schritte passieren nun wieder zwischen Ihrem Server und der Payment-API.

Die Rückkehr der Kunden via `successlink` entspricht semantisch einem “continue” bzw. “Bitte weitere API-Schritte durchführen”. D.h. bei nicht fehlgeschlagenen Zahlungsbestätigungen von Kunden (`status`-Wert am `successlink` ist nicht `FAIL`) ist der nächste API-Schritt `PayConfirm` durchzuführen. Dazu werden am `successlink` (oder `notifylink`) die notwendigen Queryparameter mitgeliefert. Ein `GetStatus`-Aufruf nach Rückkehr von Kunden und vor einem `PayConfirm`-Aufruf kann sinnvoll sein, wenn man sich des bei der API hinterlegten Zahlungsstatus versichern möchte.



6.4 PayConfirm

Nachdem der User im Zahlfenster alles erfolgreich absolviert hat, wird erst jetzt die Zahlungsantwort überprüft, die über den Nutzer-Browser zum Shop zurückgekommen ist. Die Zahlung ist je nach Zahlungsmethode möglicherweise nur vorbereitet, aber noch nicht ausgeführt (im Fall Giropay ist z.B. die Hausbank der Nutzer involviert). Um auch die interne Abrechnung korrekt zu gewährleisten, müssen PayConfirm und PayComplete bei erfolgreicher Nutzerrückkehr via `successlink` (oder Aufruf der `notifylink`-URL) trotzdem aufgerufen werden! PayConfirm braucht nur dann nicht mehr aufgerufen zu werden, wenn Nutzer über den `faillink` (`status=FAIL` am `successlink` oder `faillink`) zurückkehren, keine für PayConfirm notwendigen Parameter `data` und `signature` am `successlink` und/oder `notifylink` mitgeliefert wurden oder wenn `GetStatus` eine fehlgeschlagene Transaktion meldet.

Die für PayConfirm erforderlichen Parameter `data` und `signature` werden beim Aufruf der `successlink`- und/oder `notifylink`-URL übermittelt. Der vom Shop beim PayInit übergebene `successlink` und/oder `notifylink` darf GET-Parameter für eigene Zwecke enthalten. Achten Sie bitte auf eine sinnvolle Gesamtlänge der übergebenen URLs und die sich möglicherweise überschneidenden Parameter-Namen (insbesondere `basketid` und `status`).

Beispiel für einen `successlink` / `notifylink` (beim PayInit):

```
https://your-server.example.com/continue?your=param1&something=youneed&basketid=XXXX1234567896&status=SUCCESSorFAIL&data=somedata&signature=somesignature&ck=string&more=params
```

Eigene Parameter (oben `your=param` etc) können für eigenen Informationen genutzt werden.

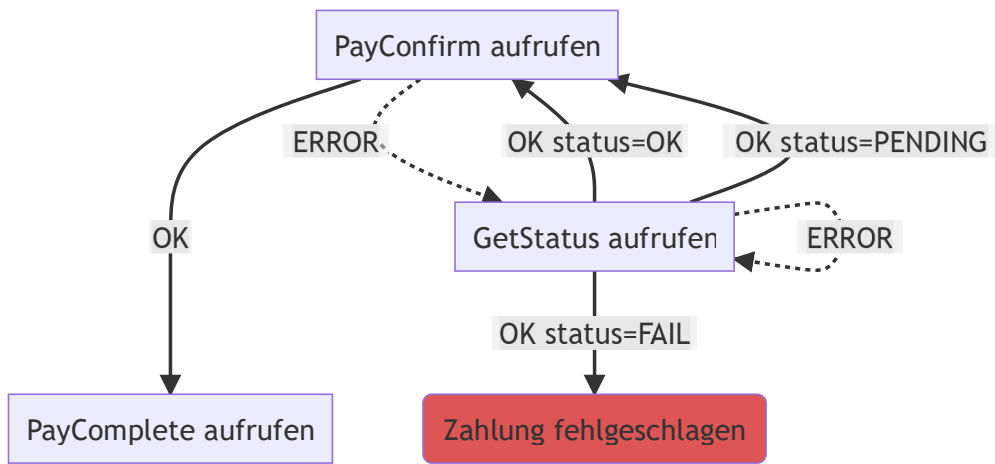
PayConfirm erwartet folgende Parameter:

Parameter	Wertebereich	Beispiel	Erläuterung
<code>basketid</code>	AN-13	ZZZ1234567890	13-stellige <code>basketid</code> (Basket-ID/Transaktions-ID), aus dem Schritt GetBasketId
<code>data</code>	ANS-680	siehe oben	Zahlungsdaten die von der Payment-API benutzt werden.
<code>signature</code>	ANS-128	siehe oben	Checksumme

Antwort (Bsp):

```
OK id=JK2781237864JLIOSIIJWIJIW87381378623&token=(none)
```

Fehlschlagende oder mit `ERROR` antwortende PayConfirm -Aufrufe bedeuten nicht zwangsläufig, dass die Zahlung beim Zahlungsdiensteanbieter fehlgeschlagen ist oder an der Payment-API als nicht erfolgreich markiert wurde. Eine Wiederholung des Aufrufs ergibt Sinn, wenn der tatsächliche Status mit einem `GetStatus` -Call geprüft wurde:



Nach erfolgreichem PayConfirm fährt man mit PayComplete fort.

6.5 PayComplete

Mittels der ID aus `PayConfirm`, wird nun die Zahlung abgeschlossen.

Parameter	Wertebereich	Beispiel	Erläuterung
basketid	AN-13	ZZZ1234567890	13-stellige basketid (Basket-ID/Transaktions-ID), aus dem Schritt GetBasketId
id	ANS-64	JK2781237864JLI0SIIJWIJIW87381378623	ID der vorbereiteten Zahlung
token	ANS-64	(none)	token aus dem <code>PayConfirm</code> -Aufruf

Rückgaben von `PayComplete` sind `OK status=OK` bzw. `ERROR error=fehlercode&errortext=fehlercode-als-text`.

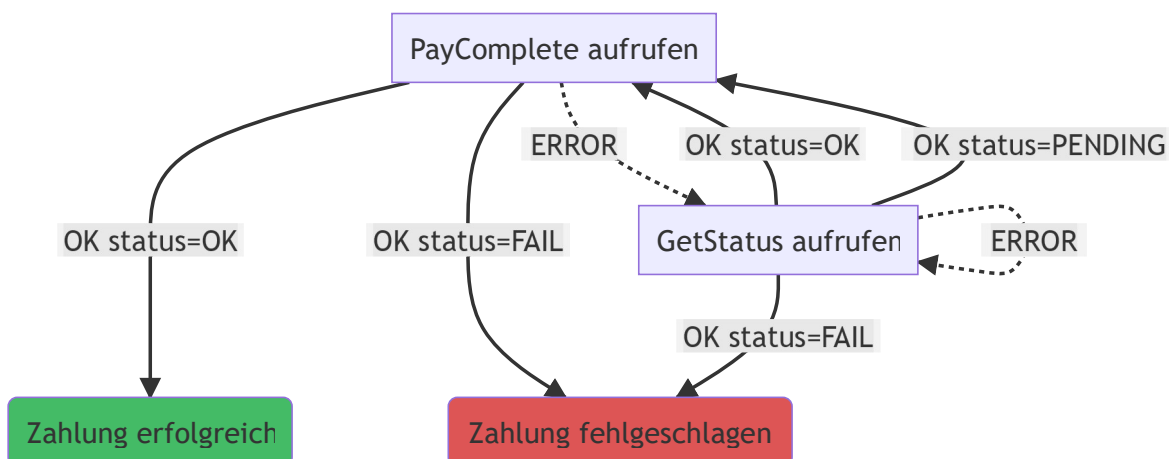
Erfolgreich bestätigte und durchgeführte Zahlung:

```
OK status=OK
```

oder ein Fehler, z.B.:

```
ERROR error=261&errortext=this%20payment%20step%20is%20not%20allowed
```

Ein `PayComplete`-Aufruf kann insbesondere bei Zahlungen mit Vorauthorisierung fehlschlagen. Ein allgemein (z.B. Timeout oder HTTP-Fehler) oder mit `ERROR` fehlschlagender Aufruf von `PayComplete` kann wiederholt werden, wenn dies laut `GetStatus` noch sinnvoll ist.



6.6 Benachrichtigung an den Shop (Notify)

Wenn Sie eine serverseitige Shop-Benachrichtigung über den Ausgang der Zahlung wünschen (unabhängig vom Nutzer und dessen Browser), dann kontaktieren Sie BerlinOnline. Andernfalls verweisen wir auf die Möglichkeit des [GetStatus-Call](#). Es gibt beim `PayInit`-Schritt die Möglichkeit einen `notifylink` zu übertragen. Diese URL wird nach beendeter Zahlungsbestätigung durch Kunden als **Server-to-Server Aufruf** ausgeführt und übermittelt u.a. den `status=SUCCESS`, `status=PENDING` oder `status=FAIL` als Query-Parameter.

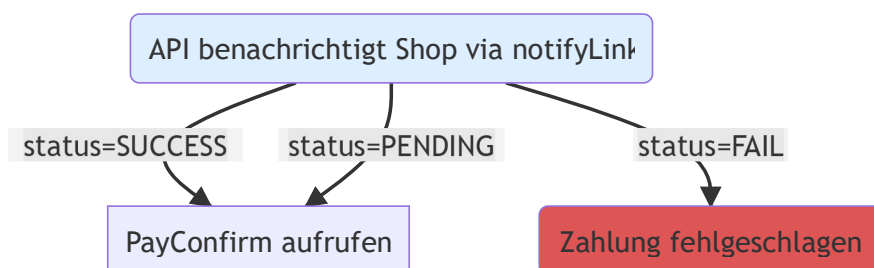
Der Wert des `status`-Parameters am `notifylink` ist in jedem Fall zu interpretieren. Erwartbare Werte sind `SUCCESS`, `FAIL` oder `PENDING` und bedeuten Folgendes:

- bei `SUCCESS`: Nutzer hat Zahlung freigegeben. Der normale API-Ablauf mit Aufruf von `PayConfirm` und `PayComplete` ist durchzuführen.
- bei `PENDING`: Zahlung beim Zahlungsdiensteanbieter wurde noch nicht komplett durchlaufen. Der normale API-Ablauf mit Aufruf von `PayConfirm` und `PayComplete` ist durchzuführen. Ein `GetStatus` in gewissen Abständen kann Gewissheit über den Transaktionsstatus schaffen.
- bei `FAIL`: Zahlung beim Zahlungsdiensteanbieter ist fehlgeschlagen. Ein API-Aufruf von `PayConfirm` wird fehlschlagen. `GetStatus` wird dies bestätigen. Zahlung durch Nutzer ist fehlgeschlagen (unabhängig davon, ob Nutzer via Redirect z.B. zum `faillink` zurückkehrt). URL-Parameter `data` und `signature` sind nicht mitgeliefert, weil ein `PayConfirm`-Aufruf nicht mehr notwendig ist.

Weitere Query-Parameter am `notifylink` entsprechen den Parametern des `successlink` (über den Kunden zum Shop zurückkehren), d.h. der `notifylink` enthält ebenfalls die Parameter `data` und `signature`, welche für den Aufruf von `PayConfirm` notwendig sind.

Hinweis: Der Notifylink-Aufruf wird nicht wiederholt bzw. erneut versucht. Der HTTP-Status oder Antworten werden nicht interpretiert.

ACHTUNG: Um unnötige Kopplung von Systemen zu vermeiden, sollten Sie den Eingang der Notifikation nur kurz bestätigen (z.B. mit HTTP Status 200 oder 201), bevor Folgeaufrufe Ihrerseits von `GetStatus` oder `PayConfirm` erfolgen. D.h. ein asynchroner Ablauf der weiteren API-Aufrufe nach Notifikation ist empfohlen!



Fehlschlagende oder mit `ERROR` antwortende `PayConfirm`-Aufrufe bedeuten nicht zwangsläufig, dass die Zahlung beim Zahlungsdiensteanbieter fehlgeschlagen ist oder an der Payment-API als nicht erfolgreich markiert wurde. Eine Wiederholung des Aufrufs ergibt Sinn, wenn der tatsächliche Status mit einem `GetStatus`-Call geprüft wurde.

Wenn `GetStatus` eine erfolgreiche Zahlung meldet, können Sie von einer erfolgreichen Zahlung ausgehen (unabhängig vom durchgeführten API-Schritt).

Ein `PayComplete` kann insbesondere bei Zahlungen mit Vorauthorisierung ebenfalls fehlschlagen. Auch hier hilft im Zweifelsfall ein `GetStatus` -Call bei Entscheidungen.

6.6.1 Hinweis

Für die TLS-verschlüsselte Übertragung der Shopbenachrichtigung benötigen Sie ein TLS-Zertifikat, das von einer anerkannten Zertifizierungsstelle auf den Server ausgestellt wurde, den Sie in der URL für die Shopbenachrichtigung angeben haben. Selbst generierte Zertifikate werden nicht akzeptiert. Sie erhalten in diesem Fall keine Shopbenachrichtigung. Sie erhalten ebenfalls keine Shopbenachrichtigung, wenn der Servername in der URL für die Shopbenachrichtigung nicht mit dem Server übereinstimmt, für den das Zertifikat ausgestellt wurde. Das gilt z.B. auch, wenn Sie in der URL für die Shopbenachrichtigung die IP-Adresse statt des Servernamens angeben.

Nach Abwicklung der Zahlung wird der Shop über das Ergebnis informiert. Dazu wird die vom Shopbetreiber eingestellte URL für die Shopbenachrichtigung aufgerufen und die in der folgenden Tabelle enthaltenen Parameter übertragen. Die Benachrichtigung des Shops erfolgt auch bei Auftreten eines Fehlers, Timeouts oder nach Abbruch der Zahlung durch den Kunden.

6.6.2 Wichtig

Wenn Ihre Shopbenachrichtigung SSL-verschlüsselt übertragen wird, testen Sie bitte nach einem Wechsel des SSL-Zertifikats, ob die Benachrichtigung nach wie vor funktioniert. An die hinterlegten `successlink` und/oder `notifylink` URLs werden nach dem Bezahlen durch den Nutzer einige Zusatzangaben (je nach Zahlungsart und nachgelagertem Dienstleister) übermittelt. Alle Felder sind dabei prinzipiell optional und sind auch nicht bei jeder Art der Bezahlung verfügbar.

Parameter	Wertebereich	Beispiel	Erläuterung
basketid	AN-13	ZZZ1234567890	13-stellige basketid (Basket-ID/Transaktions-ID), aus dem Schritt GetBasketId
status	A-7	SUCCESS	Status im OK-Fall SUCCESS oder OK
amount	N-8	1492	bezahlter Betrag (Bsp „1492“ entspricht 14,92 EUR)
currency	AN-3	EUR	Währung; siehe PayInit
customerid	AN[-]-32	CUST-1234567890	selbst gewählter Identifier; siehe PayInit

6.7 GetStatus

Mittels der Transaktions-ID bzw. Basket-ID kann jederzeit der aktuelle Status einer Transaktion/Zahlung erfragt werden. Er stellt sich so dar wie man es auch im Administrationsinterface der Transaktionsdatenbank (separat anfragen) sieht.

Parameter	Wertebereich	Beispiel	Erläuterung
basketid	AN-13	ZZZ1234567890	13-stellige basketid (Basket-ID/Transaktions-ID), aus dem Schritt GetBasketId

Rückgabe ist der aktuelle Status der Transaktion aus Sicht der Payment-API.

Rückgabewerte für das Feld `status` können sein:

- OK
- PENDING
- FAIL

Antwort Abfrage war korrekt und Status ist OK:

```
OK status=OK
```

Status `OK` bedeutet, dass die Transaktion aus Sicht der API erfolgreich war. D.h. Kunden haben die Zahlung freigegeben und dies wurde intern auch bestätigt/geprüft. Falls `PayConfirm` noch nicht durchgeführt wurde, ist dieser Schritt durchzuführen.

Weitere mögliche korrekte Antworten:

```
OK status=PENDING
```

Status `PENDING` bedeutet, dass der Ausgang der Transaktion aus API-Sicht noch nicht klar ist. Dies kann der Fall sein, falls `GetStatus` zu früh aufgerufen wurde oder weil weder Kunden zur API zurückgekehrt sind, noch Notifikationen von Zahlungsanbietern eingegangen sind oder Überprüfungen bei Zahlungsanbietern noch keinen endgültigen Zahlungsstatus liefern.

Sollten Sie als Shop nach einem `PayInit` Kunden zur Zahlungsdienstleister-URL geschickt haben und nach einigen Minuten Kunden weder via hinterlegtem `successlink` oder `faillink` zurückbekommen und auch bei dem optional hinterlegten `notifylink` keinen Aufruf registriert haben, empfiehlt sich ein regelmäßiger Aufruf von `GetStatus`. Gegebenenfalls mit exponential Backoff. Die Payment-API überprüft den Status offener Transaktionen regelmäßig im Hintergrund. Aussagen dazu, wann offene Transaktionen endgültig als erfolgreich oder fehlgeschlagen angesehen werden, sind aufgrund der verschiedenen Zahlmethoden und involvierten Hintergrundsysteme schwierig. Die API versucht Transaktionen möglichst schnell zu beenden.

```
OK status=FAIL
```

In nachfolgenden Zeilen können evtl. zusätzliche Informationen stehen, die allerdings nicht für die maschinelle Auswertung gedacht sind. Wenn `GetStatus` den Status `FAIL` liefert, gilt die Transaktion bei

der Payment-API als fehlgeschlagen. Weitere Aufrufe der API (wie `PayConfirm`) sind nicht mehr sinnvoll.

Im Fehlerfall wird die übliche Fehlermeldung ausgegeben, mit dem entsprechenden Fehlercode (Bsp.):

```
ERROR error=123&errortext=Fehlermeldung
```

Der Aufruf von `getStatus` kann dann korrigiert wiederholt werden.

6.8 Wartungsmodus

Der Wartungsmodus wird für geplante API-Abschaltungen genutzt. Im Rahmen von Wartungen, Migrationen oder angekündigten Auszeiten nachgelagerter Systeme (Zahlungsdienste), wird die Schnittstelle in einen Modus gesetzt, bei dem bestimmte Aktionen mit einem Fehler 099 antworten.

Die API-Methoden `GetBasketId` und `PayInit` antworten je nach Zustand mit:

```
ERROR error=099&errortext=maintenance+modus
```

Der Wartungsmodus kennt derzeit für API-Abnehmer zwei relevante Zustände:

- API global wird abgeschaltet: `GetBasketId` und `PayInit` antworten mit API-Fehler 099
- Eine oder mehrere Zahlungsarten werden abgeschaltet: `PayInit` antwortet mit API-Fehler 099

D.h. bei geplanter API-Downtime können keine neuen Transaktions-IDs via `GetBasketId` abgerufen werden und damit auch keine neuen Zahlungen begonnen werden.

Bei geplanter Downtime einzelner Zahlungsarten wird ein `PayInit` der jeweiligen Zahlart mit Fehlermeldung enden.

Bereits begonnene Zahlungen können in beiden Zuständen noch abgeschlossen werden. API-Abrufe außer `GetBasketId` und `PayInit` werden weiterhin im Rahmen ihrer Möglichkeiten benutzbar bleiben (`PayConfirm`, `PayComplete`, `GetStatus`, ...).

7. Anhang

7.1 Formatbeschreibung der Parameter

In der folgenden Tabelle sind alle Zeichen aufgeführt und erklärt, die das Format eines Parameters definieren.

Zeichen	Bedeutung
A	Buchstaben [A-Z]
a	Buchstaben [a-z] ; nur Kleinbuchstaben
Ä	wie A zusätzlich [ÄÖÜß] ; UTF-8 Kodierung
ä	wie a zusätzlich [äöüß] ; nur Kleinbuchstaben; UTF-8 Kodierung
N	Ziffern [0-9]
S	Sonderzeichen [*,;,:&-+!/%\$]
L	Leerzeichen []
Xn	genau <i>n</i> Zeichen
X-n	maximal <i>n</i> Zeichen
FIX	feste Zeichenkette
[...]	Eine Anzahl erlaubter Zeichen wird zwischen eckigen Klammern aufgelistet. (*)

Beispiele

- N5 Ziffer mit genau 5 Zeichen
- ANS10 Zeichenkette mit bis zu 10 Zeichen. Diese Zeichenkette kann Buchstaben, Ziffern und Sonderzeichen enthalten, aber keine Leerzeichen
- AN[_]-10 Zeichenkette mit bis zu 10 Zeichen. Diese Zeichenkette kann Buchstaben, Ziffern und das Zeichen _ enthalten, aber keine Leerzeichen
- URI Gültige URL nach dem Muster `https://www.berlin.de/dienst1/dl.php`
- URI? Gültige URL mit erlaubtem Query-String nach dem Muster `https://www.berlin.de/dienst1/dl.php?orderid=1234`
- Ist auch die eckige Klammer erlaubt, wird diese doppelt aufgeführt z.B. `[[]]`.

7.2 Ergebnismeldungen / Antworten

7.2.1 API - Errorcodes

Meldungen der REST-API kommen immer an als

- **ERRORMELDUNG** := ERROR error=[0-9]{3}&errortext=Fehlermeldung
- **OKMELDUNG** := OK OKMeldungAlsQueryString

„ERRORMELDUNG“ ist in der ersten Zeile kurz und prägnant (genormt)

In Folgezeilen können zusätzlich Ausgaben/Fehlermeldungen von hinteren Backends oder Debug-Meldungen durchkommen

Beispiele:

```
ERROR error=101&errortext=Authorization%20failed
```

```
OK basketid=ZZZ1234567890&text=Irgendwas&status=OK
```

Bei der Angabe der Codes unten, sind Kommentare hinten in eckigen Klammern “[]” angegeben

Code	Meldung
0xx	Allgemeine Fehler
001	http connection error
099	maintenance modus [Wartungs-Modus]
199	action aborted by user
-	-
1xx	Backend-Fehler (allgemein)
101	authorization failed
102	no valid api version found
103	api version does not match allowed api version
-	-
2xx	Backend-Fehler (berlin.de-blackbox eigene)
201	no available basketid found
202	settings for basketid incomplete
203	wrong trans_prefix for basketid
220	manipulated data found
221	missing or wrong account data

Code	Meldung
222	missing or wrong cvc control code
223	missing or wrong validation date
229	missing or wrong parameter for Mailing [required: mailto, mailheader, mailbody]
251	request to long [please shorten your description]
261	this payment step is not allowed
262	payment initializing failed
264	payment completion failed
291	valid CVC-code has to be given [test driven error]
-	-
3xx	Backend-Fehler (nachgelagerte Systeme)
301	no backend provider selected
302	unknown url to backend provider
303	http connection error to backend provider
320	error while sending data to one of our service providers
321	no payurl from backend
399	unspecified error from backend provider [Details finden Sie dann in den nachfolgenden Zeilen]
-	-
4xx	Eingabe-Fehler (Parameter)
401	missing or wrong parameter basketid
402	missing or wrong parameter amount
403	missing or wrong parameter currency
404	missing or wrong parameter description
405	missing or wrong parameter address (format: "streetno:zip:town:countrycode")
406	missing or wrong parameter payerfirstname
407	missing or wrong parameter payerlastname
408	missing or wrong parameter addrstreet, shipstreet
409	missing or wrong parameter addrstreetnumber, shipstreetnumber
410	missing or wrong parameter addrpostalcode, shippostalcode
411	missing or wrong parameter addrcity, shipcity

Code	Meldung
412	missing or wrong parameter addrcountry, shipcountry
414	missing or wrong parameter payeremail
415	missing or wrong parameter payerphone, payerphonemob, payerphonework
422	wrong parameter paymethod
423	wrong parameter successlink or backlink or faillink or notifylink
425	wrong parameter customerid
426	wrong parameter kunde
427	wrong parameter iban
428	wrong parameter bic
429	wrong parameter holder
430	wrong parameter sepamandatereference
451	request to long
-	-
8xx	External Checking
801	no check specified
802	accountcheck manual check required
803	accountcheck not authorized
809	accountcheck system error
-	-
9xx	Unknown / System Fehler
901	generic test error
998	unknown api call detected
999	unknown error

7.3 Testdaten

In der Payment-Testumgebung (test-secure) werden Zahlungen nur simuliert. Die Test- oder Sandboxsysteme der Zahlungsdienstleister sind hier angebunden. Bitte benutzen Sie keine echten Kontonummern oder Kreditkartennummern!

7.3.1 Zahlung mit Kreditkarte

Im Testsystem führen volle Euro-Beträge zu einer erfolgreichen Zahlung. Beträge mit Cent-Beträgen werden mit einem Fehler abgewiesen.

Für Kreditkartenzahlungen können folgende Kreditkartennummern genommen werden:

Methode	Nummer
Ohne CardHolder Autorisierung	4012001037167778
Mit CardHolder Autorisierung	4012001037664444
3DS-Methode	4005559876540
3DS-Methode und CardHolder Autorisierung	4012001036853337

Achtung: Die CVC-Nummer ist beliebig dreistellig und das Gültigkeitsdatum muss in der Zukunft liegen.

Um einen Fehler zu erzwingen:

Eine bestimmte Kreditkartennummer für absichtlich fehlschlagende Transaktionen gibt es nicht. Um fehlschlagende Kreditkarten-Transaktionen zu simulieren empfehlen sich: - Zahlendreher bei den angegebenen Nummern für eine ungültige Karte, - ein Gültigkeitsdatum in der Vergangenheit oder - ein Betrag mit Cent-Betrag (z.B. 1.05€)

7.3.2 Zahlungen mit SEPA

Im Gegensatz zur Kreditkarte können hier auch Centbeträge getestet werden.

Für eine gültige Zahlung benutzen Sie:

- Kontonummer: 1234567890
- Bankleitzahl: 12345678
- Bzw. als IBAN: DE87123456781234567890

Um einen Fehler zu erzwingen (Bankleitzahl ist anders):

- Kontonummer: 1234567890
- Bankleitzahl: 12345677
- Bzw. folgende fehlererzeugende IBAN: DE52123456771234567890

7.3.3 Zahlungen mit Giropay

Das Testsystem schickt Nutzer zu einer Paydirekt-Sandbox. Dort wählt man erst "Testbank", dann "Test-

und Spielbank AG" aus, wählt dann "Weiter zum Bezahlen" und kann anschließend die Zahlung bestätigen.

7.3.4 Zahlungen mit PayPal

Zum Testen benötigen Sie keinen Developer-PayPal-Account.

Im Test-System können Sie einen von uns bereitgestellten "buyer"-Account benutzen um die Nutzer-Seite zu testen.

Sandbox-Account:

- E-Mail-Adresse: `bdetest-buyer@test.berlinonline.net`
- Paypal-Kennwort: `Land,WieEinBall`

7.4 Tests

Bei der Anbindung der API ist es empfehlenswert manuelle und automatische Tests zu etablieren. Integrationstests sollten gegen das unter <https://test-secure.berlin.de/> erreichbare Testsystem laufen. Die test-secure-Systemumgebung hat stets die Sandbox-Systeme der Zahlungsdienstleister angebunden, damit keine Echtgeldzahlungen durchgeführt werden. Da diese Testsysteme manchmal nicht verfügbar sind, kann es allerdings zu unerwarteten Fehlern bei der API-Nutzung durch Integrationstests kommen. Solche Abhängigkeiten sollten auf Seite von Fachverfahrensherstellern bedacht werden. Eine E-Payment API Sandbox-Systemumgebung, welche vorher festgelegte Antworten liefert oder nicht die Sandbox-Systeme der Zahlungsdienstleister im Hintergrund nutzt, ist derzeit nicht verfügbar.

Empfehlenswerte Testfälle für Shops beinhalten je unterstützter Zahlungsart:

- erfolgreiche Zahlung mit gültigen Testdaten
- fehlgeschlagene Zahlung mit ungültigen Testdaten
- Abbruch durch Nutzende vor/nach Login bei Banken/Zahlsystemen
- Timeout durch Inaktivität bzw. Abbruch durch Nutzende (d.h. Simulation unterbrochener Redirect-Ketten bzw. erfolglose Rückkehr Nutzender aufgrund technischer Probleme oder Inaktivität)
- Umgang mit Fehler-Antworten der Payment-API
- Simulation des Nicht-Erhalts/Timeouts von Responses der Payment-API
- Simulation des Nicht-Erhalts von Notifikationen der Payment-API
- Doppelte Benachrichtigung über Ausgang von Zahlungen via Nutzenden-Rückkehr und Server-to-Server Benachrichtigung (in unterschiedlicher Reihenfolge; bei Nutzung von `notifylink`)
- Neustart von Bezahlvorgängen im Shop (mit erneutem Aufruf von `GetBasketId` !) nach Abbrüchen durch Nutzer oder Fehler-Rückgaben der Payment-API

7.5 Support-Anfragen

Allgemeine Supportanfragen können an die E-Mail-Adresse `payment-support@berlin.de` gestellt werden.