PASS CONSULTING GROUP

GETTING STARTED

**PASS OPEN BANKING - XS2A DOKUMENTATION**

# Getting Started

Eine Einführung in die Verwendung der xs2a-Schnittstelle

Version: 1.3

Status: Freigegeben

Vertraulichkeit: Öffentlich

## DOKUMENTENINFORMATIONEN

| | |
|---|---|
| ***Dokumentenname:*** | **Getting Started** |
| ***Beschreibung:*** | Eine Einführung in die Verwendung der xs2a-Schnittstelle |
| ***Aktuelle Versionsnummer:*** | 1.3 |
| ***Verantwortlich für den Inhalt:*** | PASS |
| ***Klassifizierung der Vertraulichkeit:*** | Öffentlich |
| ***Aktueller Status:*** | Freigegeben |



## BEARBEITUNGSHISTORIE

| Version | Autor | Gegenstand | Datum |
|---|---|---|---|
| 1.0 | PASS Consulting | Initiale Anlage | 11.06.2019 |
| 1.1 | PASS Consulting | Beispiel pain-Nachricht eingefügt | 15.10.2020 |
| 1.2 | PASS Consulting | Aktualisieren der Requests und Responses | 02.02.2021 |
| 1.3 | PASS Consulting | Erläuterung der Einschränkungen der Sandbox-Umgebung hinzugefügt<br>Aufgliederung der Autorisierung der Kontoabfrage in EMBEDDED und DECOUPLED verfahren | 02.12.2022 |
| 1.4 | PASS Consulting | REDIRECT-Ansatz hinzugefügt (oauth2) | 15.10.2024 |

## INHALT

# 1 HINWEISE ZUM TESTEN MIT DER SANDBOX-UMGEBUNG

Die Sandbox-Umgebung ist eine Simulation, welche keine Interaktionen mit dem Kernbankensystem bietet. Im Detail bedeutet das, dass Anfragen an das Kernbankensystem mit statischen Antworten simuliert werden.

## 1.1 AUTORISIERUNGSVERFAHREN

In der Sandbox-Umgebung wird lediglich das Oauth2 Verfahren-abgebildet. Um diese Verfahren testen zu können, sind folgende Testdaten zu beachten:

Um den Embedded-Approach implizit zu testen, wird folgende PSU-ID benötigt:

Demo PSU-ID 1: 654321 (Supports only OAuth2 pre-step authentication)

- `6666666` (Zahlungskonto)

Beim Testen muss darauf geachtet werden, dass die IBANs nicht unter den PSU-IDs vertauscht werden, ansonsten werden Fehler von der Schnittstelle zurückgeliefert. Weiterhin ist zu beachten, dass das Verfahren innerhalb der Sandbox-Umgebung eingeschränkt sind. Die Einschränkungen werden im Detail in den beiden folgenden Kapiteln beschrieben.

### 1.1.1 AUTORISIERUNG MIT OAUTH2

Bei der Authorisierung mit Oauth2 wird der Kunde auf die Authorisierungsseite der Bank umgeleitet, muss sich dort authorisieren und erhält danach ein gültiges Token mit dem ein consent angefragt werden kann. Ist das der Fall wird der „scaStatus: finalised“ als Antwort zurückgeliefert und in den nächsten Schritten können die Kontoumsätze, wie auch im Embedded-Approach, abgefragt werden. In der Sandbox-Umgebung wird, die korrekte Ausführung der vorherigen Schritte vorausgesetzt, die Statusabfrage immer mit „scaStatus: finalised“ beantwortet.

### 1.2 ABFRAGE KONTOUMSÄTZE UND ZAHLUNGSANWEISUNGEN

Auch die Abfragen der Kontoumsätze oder Zahlungsanweisungen sind innerhalb der Sandbox-Umgebung eingeschränkt, bzw. kommunizieren nicht mit dem Kernbankensystem. Werden hier Abfragen versendet, werden diese statisch beantwortet. Im Detail heißt das, dass die Kontoumsätze keine Änderungen erhalten und Zahlungsanweisungen nicht ausgeführt werden.

## 2 VORAUSSETZUNGEN

### 2.1 KONTAKT ZUR REGULIERENDEN STELLE

Damit Sie als TPP (Third Party Provider) die produktiven XS2A-Schnittstellen nutzen können, müssen Sie sich bei der regulierenden Stelle (in Deutschland ist dies die BaFin) registrieren oder eine Erlaubnis einholen.

### 2.2 TRUST CENTER ZERTIFIKAT

Um die XS2A-API nutzen zu können, benötigen Sie ein Echtzertifikat, welches von einem zugelassenen Trust Center ausgestellt ist. In Deutschland werden diese Zertifikate z.B. durch die Bundesdruckerei ausgestellt. Weitere Informationen sind beispielsweise unter https://www.bundesdruckerei.de/de/PSD2 zu finden. Eine Liste der internationalen Trust Center finden Sie unter https://webgate.ec.europa.eu/tl-browser/#/, diese Trust-Center werden im produktiven Kontext unterstützt.

Dieses Zertifikat können Sie erst nach der Zulassung durch die regulierende Stelle beantragen.

### 2.3 REGISTRIERUNG

Neben dem QWAC-Zertifikat ist eine Registrierung notwendig. Hierzu genügt eine formlose E-Mail an xs2a@pass-consulting.com mit folgenden Informationen:

- E-Mail-Adresse zu Kontaktzwecken
- Name ihrer Organisation
- Webseite ihrer Organisation
- Rollen ihres Zertifikates (z.B. „PSP_PI“)
- PSP-ID ihres Zertifikates (z.B. PSDDE-BAFIN-123456 – entspricht OID 2.5.4.97)
- Aussteller (inkl. Land) ihres Zertifikates

Bei konkreten Fragen zu durchgeführten API-Aufrufen, senden Sie uns bitte in der Nachricht folgende zusätzliche Informationen:
X-Request-ID, aufgerufener API-Endpunkt, paymentId oder ConsentId und Datum/Uhrzeit des Requests.

## 3 VERWENDUNG DER SCHNITTSTELLE

Bei der Schnittstelle handelt es sich bei den meisten Endpunkten um einen HTTP-JSON-Schnittstelle. Unter https://dz-privatbank-xs2a.pass-consulting.com/ sind weiterführende Informationen und eine Sandbox-Umgebung zu finden.

Anhand von zwei Beispielen – der Auslösung einer Zahlung und der Abfrage einer Kontoübersicht – soll im Folgenden erläutert werden, wie die Schnittstelle verwendet werden kann.

## 3.1 ABFRAGE EINER KONTOÜBERSICHT

Der Kunde möchte seine Kontostände über einen längeren Zeitraum über einen Drittanbieter abfragen

### 3.1.1 AUTORISIERUNG MIT OAUTH2-APPROACH

#### 3.1.1.1 Authorisation Request

Für die „Autorisierungsanfrage“ an den Autorisierungsendpunkt sind die folgenden Parameter definiert:

| Attribute | Condition | Description |
|---|---|---|
| response_type | Mandatory | "code" is recommended as response type |
| client_id | Mandatory | organizationIdentifier as provided in the eIDAS certificate. The organizationIdentifier attribute shall contain information using the following structure in the presented order:<br>- "PSD" as 3 character legal person identity type reference;<br>- 2 character ISO 3166 country code representing the NCA country;<br>- hyphen-minus "-" and<br>- 2-8 character NCA identifier (A-Z uppercase only, no separator) - hyphen-minus "-" and<br>- PSP identifier (authorization number as specified by NCA). |
| Scope | Mandatory | PIS: The scope is the reference to the payment resource in the form "PIS:". AIS: The scope is the reference to the consent resource for account access in the form "AIS:" PIIS: The scope is the reference to the consent resource for granting consent to confirmation of funds in the form "PIIS:". Note: The resource ids chosen by the ASPSP need to be unique to avoid resource conflicts during the SCA process. |
| state | Mandatory | A dynamical value set by the TPP and used to prevent XSRF attacks. |
| redirect_uri | Mandatory | the URI of the TPP where the |

| | | |
|---|---|---|
| | | OAuth2 server is redirecting the PSU's user agent after the authorization. |
| code_challenge | Mandatory | PKCE challenge according to cryptographic RFC 7636 (https://tools.ietf.org/html/rfc7636) used to prevent code injection attacks. |
| code_challenge_method | Optional | Code verifier transformation method, is "S256" or "plain". "S256" is recommended by this specification |

**Example**

GET /authorise?response_type=code&client_id="PSDES-BDE-3DFD21" & scope=ais%3A1234-wertiq-983+offline_access& state= S8NJ7uqk5fY4EjNvP_G_FtyJu6pUsvH9jsYni9dMAJw& redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb& code_challenge_method="S256" code_challenge=5c305578f8f19b2dcdb6c3c955c0aa709782590b4642eb890b97e43917cd 0f36

HTTP/1.1

Host: api.testbank.com

#### 3.1.1.2 Authorisation Response

Die Autorisierungsantwort des ASPSP liefert die folgenden Daten: Bemerkung: Da die Anfrage nicht vom TPP, sondern vom PSU-Benutzeragenten gesendet wird, wird sie nicht durch den QWAC des TPP gesichert.

**http Response Code**

302

**Query Parameters**

| Attribute | Condition | Description |
|---|---|---|
| Location: | Mandatory | redirect URI of the TPP |
| Code | Mandatory | Authorisation code |
| state | Mandatory | Same value as for the request. |

#### 3.1.1.3 Token Request

Der TPP sendet eine POST-Anfrage an den Token-Endpunkt, um den in der Autorisierungsantwort bereitgestellten Autorisierungscode gegen ein Zugriffstoken und optional gegen ein Refresh-Token auszutauschen. Die folgenden Parameter werden verwendet:

| Attribute | Condition | Description |
|---|---|---|

| | | |
|---|---|---|
| grant_type | Mandatory | "authorization_code" is recommended as response type. |
| client_id | Mandatory | See Authorisation Request |
| code | Mandatory | Same value as for the request. |
| redirect_uri | Mandatory | the exact uri of the TPP where the OAuth2 server redirected the user agent to for this particular transaction |
| code_verifier | Mandatory | PKCE verifier according to cryptographic RFC 7636 (https://tools.ietf.org/html/rfc7636) used to prevent code injection attacks. |

**Beispiel**:

POST /token HTTP/1.1

Host: https://api.testbank.com

Content-Type: application/x-www-form-urlencoded client_id="PSDES-BDE-3DFD21" &grant_type=authorisation_code &code=SplxlOBeZQQYbYS6WxSbIA &redirect_uri= https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb &code_verifier=7814hj4hjai87qqhjz9hahdeu9qu771367647864676787878

Der TPP wird bei dieser Anfrage durch die Verwendung von „Oauth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens“ in Verbindung mit dem eIDAS-Zertifikat des TPP authentifiziert.

#### 3.1.1.4 Token Response

Die ASPSPS antwortet mit den folgenden Parametern:

| Attribute | Condition | Description |
|---|---|---|
| access_token | Mandatory | Access Token bound to the scope as requested in the authorisation request and confirmed by the PSU. |
| token_type | Mandatory | Set to "Bearer" |
| expires_in | Optional | The lifetime of the access token in seconds |
| refresh_token | Optional | Refresh Token, which can be utilised to obtain a fresh access tokens in case the previous access token expired or was revoked. Especially useful in the context of |

| | | |
|---|---|---|
| | | AIS. |
| scope | Mandatory | the scope of the access token |

Beispiel:

HTTP/1.1 200 OK

```
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "expires_in": 3600,
   "refresh_token": "tGzv3JokF0XG5Qx2TlKWIA",
   "scope": "exampleScope"
}
```

#### 3.1.1.5 Request – POST /consents

Zunächst wird die Einwilligung des Benutzers benötigt. Dazu wird ein Request an diesen Endpunkt geschickt.

```
POST https://.../api/v1/consents
accept: application/json
content-type: application/json
psu-id: <Login-ID des Endkunden>
x-request-id: <UUID für diesen Request>

{
  "access" : {
    "allPsd2" : "allAccounts"
  },
  "recurringIndicator" : true,
  "validUntil" : <Zeitpunkt in der Zukunft – Formatbeispiel: "2019-07-13">,
  "frequencyPerDay" : 1,
  "combinedServiceIndicator" : false
}
```

#### 3.1.1.6 Response – POST /consents

```
201
ASPSP-SCA-Approach: REDIRECT
Content-Type: application/json
Location: /api/v1/consents/<Consent-ID>
```

```
{
  "consentStatus" : "received",
  "consentId" : "<Consent-ID>/",
  "_links" : [ {
    "self" : {
      "href" : "/api/v1/consents/<Consent-ID>/"
    }
  }, {
    "status" : {
      "href" : "/api/v1/consents/<Consent-ID>/status"
    }
  }
 ]
}
```


Die Antwort enthält die URL zum Starten der Autorisierung.

Achtung:

Beim Testen mit der Sandbox, wird immer das Ergebnis „finalised“ zurückgeliefert, sofern im Link die richtige Authorisation-ID angegeben ist.

### 3.1.2 KONTOABFRAGE

Die Kontoabfrage kann erst durchgeführt werden wenn eine der beiden, **Fehler! Verweisquelle konnte nicht gefunden werden.** oder 3.1.1, Autorisierungspfade erfolgreich durchlaufen wurden.

#### 3.1.2.1 Request – GET /accounts

Es können nun die Konten des Kunden abgerufen werden.

```
GET https://.../api/v1/accounts?withBalance=true
accept: application/json
consent-id: <Consent-ID>
x-request-id: <UUID für diesen Request>
```

#### 3.1.2.2 Response – GET /accounts

```
Content-Type: application/json
x-request-id: <UUID für diesen Request>

{ "accounts" : [ { … } ] }
```

Die Antwort enthält die Kundenkonten. Diese Anfrage kann bis zum Ablauf der Einwilligung einmal täglich abgesetzt werden.

## 3.2 AUSLÖSUNG EINER ZAHLUNG

Ein Endkunde möchte eine einmalige Zahlung auslösen.

### 3.2.1 REQUEST - POST /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}

Zunächst wird ein POST-Aufruf an /{payment-service}/{payment-product} getätigt. Dabei sind die Variablen im Pfad zu setzen. Da es sich um eine einmalige Zahlung handelt und die Zahlung im pain-xml-Format übergeben wird, lautet der Pfad /payments/pain.001-sepa-credit-transfers.

Im Header sollten – soweit vorhanden – die Daten über die Anfrage des Endkunden in den Feldern PSU-* erfasst werden. Die Identifikationsnummer PSU-ID muss auf jeden Fall übergeben werden. Die Nachricht kann auch noch über die Header-Felder Digest, Signature und TPP-Signature-Certificate signiert werden.

Im Body der Anfrage wird die Zahlung im pain-xml-Format übergeben.

```
POST https://.../api/v1/payments/pain.001-sepa-credit-transfers
accept: application/json
content-type: application/xml
psu-id: <Login-ID des Endkunden>
x-request-id: <UUID für diesen Request>

<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03">
   <CstmrCdtTrfInitn>
    <GrpHdr>
      <MsgId>ABC/090928/CCT001</MsgId>
      <CreDtTm>yyyy-mm-ddThh:mm:ss</CreDtTm>
      <NbOfTxs> … </NbOfTxs>
      <CtrlSum>11500000</CtrlSum>
      <InitgPty>
        <Nm>ABC Corporation</Nm>
        <PstlAdr>
          <StrtNm> … </StrtNm>
          <BldgNb> … </BldgNb>
          <PstCd> … </PstCd>
          <TwnNm> … </TwnNm>
          <Ctry> … </Ctry>
        </PstlAdr>
      </InitgPty>
    </GrpHdr>
    <PmtInf>
      <PmtInfId>ABC/086</PmtInfId>
      <PmtMtd>TRF</PmtMtd>
      <BtchBookg>false</BtchBookg>
      <ReqdExctnDt>2020-03-03</ReqdExctnDt>
      <Dbtr>
        <Nm>ABC Corporation</Nm>
        <PstlAdr>
          <StrtNm> … </StrtNm>
          <BldgNb> … </BldgNb>
          <PstCd> … </PstCd>
```

```
      <TwnNm> … </TwnNm>
      <Ctry> … </Ctry>
    </PstlAdr>
  </Dbtr>
  <DbtrAcct>
    <Id>
      <IBAN> … </IBAN>
    </Id>
  </DbtrAcct>
  <DbtrAgt>
    <FinInstnId>
      <BIC> … </BIC>
    </FinInstnId>
  </DbtrAgt>
  <CdtTrfTxInf>
    <PmtId>
      <InstrId>ABC/090928/CCT001/01</InstrId>
      <EndToEndId>ABC/4562/yyyy-mm-dd</EndToEndId>
    </PmtId>
    <Amt>
      <InstdAmt Ccy="EUR">0.01</InstdAmt>
    </Amt>
    <ChrgBr>SHAR</ChrgBr>
    <CdtrAgt>
      <FinInstnId>
        <BIC> … </BIC>
      </FinInstnId>
    </CdtrAgt>
    <Cdtr>
      <Nm>DEF Electronics</Nm>
      <PstlAdr>
        <AdrLine> -street- </AdrLine>
        <AdrLine> -postalcode + city- </AdrLine>
        <AdrLine> -country- </AdrLine>
      </PstlAdr>
    </Cdtr>
    <CdtrAcct>
      <Id>
        <IBAN> … </IBAN>
      </Id>
    </CdtrAcct>
    <Purp>
      <Cd>CINV</Cd>
    </Purp>
    <RmtInf>
      <Strd>
        <RfrdDocInf>
          <Nb> -number- </Nb>
```

```
                <RltdDt> yyyy-mm-dd </RltdDt>
              </RfrdDocInf>
            </Strd>
          </RmtInf>
        </CdtTrfTxInf>
      </PmtInf>
    </CstmrCdtTrfInitn>
</Document>
```

### 3.2.2 RESPONSE - POST /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}

```
201
ASPSP-SCA-Approach: EMBEDDED
Content-Type: application/json
Location: /api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>
x-request-id: <UUID für diesen Request>

{
  "transactionStatus" : "RCVD",
  "paymentId" : "<Payment-ID>",
  "_links" : [ {
    "self" : {
      "href" : "/api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>"
    }
  }, {
    "status" : {
      "href" : "/api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/status"
    }
  }, {
    "startAuthorisationWithPsuAuthentication" : {
      "href" : "/api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/authorisations"
    }
  } ]
}
```

In der Antwort sind zwei wichtige Informationen enthalten. Einerseits eine eindeutige ID der Zahlung (im Response-Body unter „paymentId“ und im Header unter „Location“) und andererseits das Verfahren für die SCA (im Header unter „ASPSP-SCA-Approach“ sowie im Body). Im Folgenden wird davon ausgegangen, dass das EMBEDDED-Verfahren für den SCA verwendet wird.

Die Zahlung befindet sich im Zustand „received“. Die Zahlung wurde also vom Zahlungsdienstleister empfangen. Ausgeführt wird die Zahlung allerdings erst nach der Autorisierung.

### 3.2.3 REQUEST - POST /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/AUTHORISATIONS

Mit diesem Request wird das Autorisierungsverfahren gestartet

```
POST https://.../api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/authorisations
accept: application/json
content-type: application/json
x-request-id: <UUID für diesen Request>

{ }
```

### 3.2.4 RESPONSE - POST /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/AUTHORISATIONS

```
201
ASPSP-SCA-Approach: EMBEDDED
Content-Type: application/json
x-request-id: <UUID für diesen Request>

{
  "scaStatus" : "psuIdentified",
  "authorisationId" : "auth-wS4lLbMLE5IZ38p",
  "_links" : [ {
    "updatePsuAuthentication" : {
      "href"               :               "/api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/authorisations/<Authorisation-ID>"
    }
  } ]
}
```

Die Autorisierung ist nun angelegt.

### 3.2.5 REQUEST - PUT /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/AUTHORISATIONS/{AUTHORISATIONID}

Nachdem das Passwort des Endkunden abgefragt wurde, kann es an die Schnittstelle übertragen werden.

```
PUT                                 https://.../api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/authorisations/<Authorisation-ID>
accept: application/json
content-type: application/json
psu-id: <Login-ID des Endkunden> (Optional)
x-request-id: <UUID für diesen Request>

{
  "psuData" : {
    "password" : "<Password des Endkunden>"
  }
}
```

### 3.2.6 RESPONSE - PUT /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/AUTHORISATIONS/{AUTHORISATIONID}

```
200
ASPSP-SCA-Approach: EMBEDDED
Content-Type: application/json
x-request-id: <UUID für diesen Request>

{
  "scaStatus" : "psuAuthenticated",
  "chosenScaMethod" : {
    "authenticationType" : "SMS_OTP",
    "authenticationVersion" : "1",
    "authenticationMethodId" : "Mtan",
    "name" : "Mtan an die registrierte Handynummer",
    "explanation" : "Generiert eine Mtan und verschickt diese an die registrierte Handynummer"
  },
  "challengeData" : {
    "otpMaxLength" : 6,
    "otpFormat" : "integer"
  },
  "_links" : [ ]
}
```

Das Passwort wurde akzeptiert und das hinterlegte SCA-Verfahren (hier ein TAN-Verfahren per SMS) wird gestartet.

### 3.2.7 REQUEST - PUT /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/AUTHORISATIONS/{AUTHORISATIONID}

Der Benutzer bekommt seine TAN per SMS und diese TAN kann nun zur Autorisierung der Zahlung verwendet werden.

```
PUT https://.../api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/authorisations/<Authorisation-ID>
accept: application/json
connection: keep-alive
content-type: application/json
psu-id: <Login-ID des Endkunden> (Optional)
x-request-id: <UUID für diesen Request>

{
  "scaAuthenticationData" : "<TAN des Endkunden>"
}
```

### 3.2.8 RESPONSE - PUT /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/AUTHORISATIONS/{AUTHORISATIONID}

```
200
Content-Type: application/json
x-request-id: <UUID für diesen Request>

{
  "scaStatus" : "finalised",
  "_links" : [ ]
}
```

Bei korrekter TAN bestätigt der Server das Ergebnis. Die Autorisierung ist abgeschlossen.

### 3.2.9 REQUEST – GET /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/STATUS

Der Status der Zahlung kann jederzeit abgefragt werden (auch bevor die Zahlung autorisiert ist).

```
GET https://.../api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/status
accept: application/json
x-request-id: <UUID für diesen Request>
```

### 3.2.10 RESPONSE – GET /{PAYMENT-SERVICE}/{PAYMENT-PRODUCT}/{PAYMENTID}/STATUS

```
200
Content-Type: application/json
x-request-id: <UUID für diesen Request>

{
  "transactionStatus" : "RJCT"
}
```

In diesem Beispiel wurde die Zahlung leider abgelehnt.