PASS CONSULTING GROUP

GETTING STARTED

PASS OPEN BANKING - UMWELTBANK XS2A

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 Aufgliederung der Autorisierung der Kontoabfrage in EMBEDDED und DECOUPLED verfahren	02.12.2022

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 werden zwei Verfahren-abgebildet. „Mtan“ (Embedded-Approach) und „PushApp“ (Decoupled-Approach). Um beide 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: `123456` (EMBEDDED SCA Approach)
`DE32760350000001111111` (Pluskonto)
`DE98760350000002222222` (Sparbuch)

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

Demo PSU-ID 3: `131415` (DECOUPLED SCA Approach)
`DE05760350000005555555` (Girokonto)

Möchte man zwischen den beiden Verfahren auswählen können, wird folgende PSU-ID benötigt:

Demo PSU-ID 2: `7891011` (EMBEDDED oder DECOUPLED SCA Approach)
`DE67760350000003333333` (Girokonto)
`DE36760350000004444444` (Darlehen)

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 beide Verfahren innerhalb der Sandbox-Umgebung eingeschränkt sind. Die Einschränkungen werden im Detail in den beiden folgenden Kapiteln beschrieben.

1.1.1 AUTORISIERUNG MIT EMBEDDED-APPROACH

Im „Embedded-Approch“ ist die Einschränkung in der TAN-Abfrag (3.1.1.7) zu finden. Kommt es zu einer TAN-Abfrage bzw. Eingabe im „Embedded-Approach“, ist die gültige TAN immer „123456“. In der Produktionsumgebung wird im vorherigen Schritt eine TAN an den Kunden versendet, welche dann anstelle der Simulations-TAN „123456“ zu ersetzen ist. Nach Eingabe und Absenden der gültigen TAN, kommt von der Xs2a-Schnittstelle eine Antwort mit dem „scaStatus: finalised“. Achtung: Die Simulation erkennt Fehleingaben der TAN. Bei einer falschen TAN, antwortet die Schnittstelle mit einem Error. Ist der „scaStatus: finalised“ können im Folgenden die Kontoumsätze der zur Autorisierung verwendeten PSU-ID abgefragt werden.

1.1.2 AUTORISIERUNG MIT DECOUPLED-APPROACH

Im „Decoupled-Approach“ ist die Einschränkung in der Statusabfrage (3.1.2.7) der Autorisierung zu finden. Bei der Statusabfrage wird im Kernbankensystem angefragt, ob der Kunde sich authentifiziert hat. 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 der Umweltbank 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://open-banking.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 EMBEDDED-APPROACH

3.1.1.1 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
}
```

3.1.1.2 Response – POST /consents

```
201
ASPSP-SCA-Approach: EMBEDDED
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"
    }
  }, {
    "startAuthorisationWithPsuAuthentication" : {
      "href" : "<URL für Authorisierung>"
    }
  } ]
}
```

Die Antwort enthält die URL zum Starten der Autorisierung.

3.1.1.3 Request – POST /consents/{consentId}/authorisations

Mit dieser URL kann nun die Autorisierung gestartet werden.

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

{ }
```

3.1.1.4 Response – POST /consents/{consentId}/authorisations

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

{
  "scaStatus" : "psuIdentified",
  "authorisationId" : "<Authorisation-ID>",
  "_links" : [ {
    "updatePsuAuthentication" : {
      "href" : "<URL für Autorisierung>"
    }
  } ]
}
```

In der Antwort ist die URL enthalten, an der die Autorisierung fortgesetzt wird.

3.1.1.5 Request – PUT /consents/{consentId}/authorisations

Beim Embedded-Verfahren ist zunächst das Kundenpasswort zu übergeben.

```
PUT https://.../api/v1/consents/<Consent-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>"
  }
}
```

3.1.1.6 Response – PUT /consents/{consentId}/authorisations

```
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" : [ ]
}
```

Durch den Aufruf wird der Versand einer TAN ausgelöst.

3.1.1.7 Request – PUT /consents/{consentId}/authorisations

Nachdem der Kunde die TAN empfangen und eingegeben hat, kann diese an die Schnittstelle übergeben werden.

```
PUT https://.../api/v1/consents/<Consent-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>

{
  "scaAuthenticationData" : "<Vom Endkunden eingegebene TAN>"
}
```

3.1.1.8 Response – PUT /consents/{consentId}/authorisations

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

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

Ist die TAN richtig, so ist die Autorisierung abgeschlossen und die Einwilligung des Kunden gültig.

3.1.2 AUTORISIERUNG MIT DECOUPLED-APPROACH

3.1.2.1 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
}
```

3.1.2.2 Response – POST /consents

```
201
ASPSP-SCA-Approach: DECOUPLED
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"
    }
  }, {
    "startAuthorisationWithPsuAuthentication" : {
      "href" : "<URL für Autorisierung>"
    }
  } ]
}
```

Die Antwort enthält die URL zum Starten der Autorisierung.

3.1.2.3 Request – POST /consents/{consentId}/authorisations

Mit dieser URL kann nun die Autorisierung gestartet werden.

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

{ }
```

3.1.2.4 Response – POST /consents/{consentId}/authorisations

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

{
  "scaStatus" : "started",
  "chosenScaMethod" : {
    "authenticationType": "PUSH_OTP",
    "authenticationVersion": "1",
    "authenticationMethodId": "PushApp",
    "name": "Freigabe des Vorgangs mittels PushApp",
    "explanation": "Freigabe des Vorgangs erfolgt mittels PushApp."
  },
  "authorisationId" : "<Authorisation-ID>",
  "psuMessage" : "Please use your BankApp for transaction Authorisation.",
  "_links" : [ {
    "status" : {
      "href" : "<URL für Statusabfrage>"
    }
  } ]
}
```

In der Antwort ist die URL enthalten, an der die Statusabfrage fortgesetzt wird.

3.1.2.5 Optional Request – PUT /consents/{consentId}/authorisations

Beim Decoupled-Verfahren kann das Kundenpasswort zur Validierung übergeben werden.

```
PUT https://.../api/v1/consents/<Consent-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" : "<Passwort des Endkunden>"
  }
}
```

3.1.2.6 Response – PUT /consents/{consentId}/authorisations

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

{
  "scaStatus" : "started",


  "_links" : [ {
    "scaStatus" : {
      "href" : "<URL für Statusabfrage>"
    }
  } ]

}
```

Durch den Aufruf wird das Passwort validiert.

3.1.2.7 Request – GET /consents/{consentId}/authorisations

Mit dieser URL wird der Status der Autorisierung überprüft und verifiziert.

```
POST https://.../api/v1/consents/<Consent-ID>/authorisations/<Authorisation-ID>
accept: application/json
content-type: application/json
x-request-id: <UUID für diesen Request>

{ }
```

3.1.2.8 Response – POST /consents/{consentId}/authorisations

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

{
  "scaStatus" : "finalised"
}
```

Ist der Status der Antwort „finalised“, so ist die Autorisierung abgeschlossen und die Einwilligung des Kunden gültig.

Achtung:

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

3.1.3 KONTOABFRAGE

Die Kontoabfrage kann erst durchgeführt werden wenn eine der beiden, 3.1.1 oder 3.1.2, Autorisierungspfade erfolgreich durchlaufen wurden.

3.1.3.1 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>
```

3.1.3.2 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.