PASS CONSULTING GROUP

GETTING STARTED

PASS OPEN BANKING - UMWELTBANK XS2A

Getting Started

An Introduction to the xs2a-Interface

Version: 1.3

State: Published

Confidentiality: Public

DOCUMENT INFORMATION

Document name:	**Getting Started**
Description:	An Introduction to the xs2a-Interface
Current version number:	1.3
Content responsibility:	PASS
Confidentiality:	Public
Current State:	Published

CHANGE HISTORY

Version	Author	Changes	Date
1.0	PASS Consulting	Initiale Anlage	2019-06-11
1.1	PASS Consulting	Added an example pain message	2020-10-15
1.2	PASS Consulting	Updated requests and responses	2021-02-02
1.3	PASS Consulting	Added explanation of the limitations of the sandbox environment Breakdown of account query authorisation into EMBEDDED and DECOUPLED procedures	2022-12-02

CONTENTS

1 NOTES ON TESTING WITH THE SANDBOX ENVIRONMENT

The sandbox environment is a simulation that does not provide any interactions with the core banking system. This means that requests to the core banking system are simulated with static responses.

1.1 AUTHORISATION PROCEDURE

Two procedures are mapped in the sandbox environment. “Mtan” (embedded approach) and “PushApp” (decoupled approach). In order to be able to test both procedures, the following test data must be considered:

To test the embedded approach implicitly, the following PSU ID is required:

Demo PSU-ID 1: 123456 (EMBEDDED SCA Approach).
DE32760350000001111111 (Pluskonto)
DE98760350000002222222 (Sparbuch)

To test the Decoupled-Approach, the following PSU-ID is required:

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

If you want to be able to choose between the two methods, the following PSU-ID is required:

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

When testing, make sure that the IBANs are not swapped among the PSU IDs, otherwise errors will be returned by the interface. Furthermore, it should be noted that both procedures are restricted within the sandbox environment. The restrictions are described in detail in the following two chapters.

1.1.1 AUTHORISATION WITH EMBEDDED-APPROACH

In the "embedded approach", the restriction is found in the TAN query (3.1.1.6). If a TAN is requested or entered in the "embedded approach", the valid TAN is always "123456". In the production environment, a TAN is sent to the customer in the previous step, which is then to be substituted for the simulation TAN "123456". After the valid TAN has been entered and sent, the Xs2a interface responds with the "scaStatus: finalised". Attention: The simulation recognises incorrect TAN entries. If the TAN is incorrect, the interface responds with an error. If the "scaStatus: finalised" is given, the account transactions of the PSU ID used for authorisation can be queried in the following.

1.1.2 AUTHORISATION WITH DECOUPLED APPROACH

In the "decoupled approach" the restriction is found in the status query (3.1.2.7) of the authorisation. During the status query, the core banking system asks whether the customer has authenticated himself. If this is the case, the "scaStatus: finalised" is returned as the answer and in the next steps the account transactions can be queried, as in the embedded approach. In the sandbox environment, the status query is always answered with "scaStatus: finalised", presuming that the previous steps have been executed correctly.

1.2 ACCOUNT TRANSACTIONS AND PAYMENT INSTRUCTION QUERIES

The queries for account transactions or payment instructions are also restricted within the sandbox environment, or do not communicate with the core banking system. If queries are sent here, they are answered statically. This means that the account transactions do not receive any changes and payment instructions are not executed.

2 PREQUISITS

2.1 CONTACT THE REGULATORY BODY

To gain access to the live XS2A-interface as a TPP (third party provider), you have to register with a regulatory body (the BaFin in Germany).

2.2 TRUST CENTER CERTIFICATE

To use the live XS2A-interface you need a certificate ready for production usage (QWAC). Such a certificate is issued by trust services listed at https://webgate.ec.europa.eu/tl-browser/#/. More information on this matter is available at your trust service, i.e. https://www.bundesdruckerei.de/en/PSD2 in Germany.

You will need an approval from your regulatory body first.

2.3 REGISTRATION

After receiving your QWAC, send us an e-mail at xs2a_umweltbank@pass-consulting.com containing the following information:

- An e-mail-address to contact you
- Name of your organisation
- Webpage of your organisation
- Roles in your certificate (for example “PSD_PI”)
- PSP-ID of your certificate (for example PSDDE-BAFIN-123456 – listed at OID 2.5.4.97)
- The trust services that signed your certificate

If you have any questions regarding executed API-calls, please send us a message containing the following additional informations:

- X-Request-ID
- URL (including any identifiers contained in the URL)
- Date and Time of the request

3 USING THE INTERFACE

Most of the interface’s endpoints are transmitting JSON over HTTP. At https://open-banking.pass-consulting.com/ additional information about each endpoint and a sandbox is available.

Nevertheless, this document describes two typical use cases of the API. Triggering a payment and accessing a list of accounts.

3.1 ACCESSING THE ACCOUNT LIST

A customer wants to track his accounts and balances using a TPP for an extended period of time.

3.1.1 AUTHORISATION WITH EMBEDDED APPROACH

3.1.1.1 Request – POST /consents

First we need the consent of the customer. We initiate the creation of the consent by calling this endpoint.

```
POST https://.../api/v1/consents
accept: application/json
content-type: application/json
psu-id: <Customer's Login-ID>
x-request-id: <UUID for this request>

{
  "access" : {
    "allPsd2" : "allAccounts"
  },
  "recurringIndicator" : true,
  "validUntil" : <Some time in the future - example: "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>
x-request-id: <UUID for this request>

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

The response contains the URL to start the authorisation.

Request – POST /consents/{consentId}/authorisations
We post an empty body to the URL

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

{ }
```

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

```
201
ASPSP-SCA-Approach: EMBEDDED
Content-Type: application/json
x-request-id: <UUID for this request>

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

The response contains the URL for continuing the authorisation process.

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

The embedded approach requires us to send the customer's password first.

```
PUT https://.../api/v1/consents/<Consent-ID>/authorisations/<Authorisation-ID>
accept: application/json
content-type: application/json
psu-id: <Customer's Login-ID> (Optional)
x-request-id: <UUID for this request>

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

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

```
200
ASPSP-SCA-Approach: EMBEDDED
Content-Type: application/json
x-request-id: <UUID for this 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" : [ ]
}
```

Sending the password will send an SMS containing the TAN to the customer.

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

After the customer received his TAN, we can hand it to the API.

```
PUT https://.../api/v1/consents/<Consent-ID>/authorisations/<Authorisation-ID>
accept: application/json
content-type: application/json
psu-id: <Customer's Login-ID> (Optional)
x-request-id: <UUID for this request>

{
  "scaAuthenticationData" : "<Customer's TAN>"
}
```

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

```
200
Content-Type: application/json
x-request-id: <UUID for this request>

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

If the TAN is correct, the authorisation is complete and the consent will be valid.

3.1.2 AUTHORISATION WITH DECOUPLED APPROACH

3.1.2.1 Request – POST /consents

First we need the consent of the customer. We initiate the creation of the consent by calling this endpoint.

```
POST https://.../api/v1/consents
accept: application/json
content-type: application/json
psu-id: <Customer's Login-ID>
x-request-id: <UUID for this request>

{
  "access" : {
    "allPsd2" : "allAccounts"
  },
  "recurringIndicator" : true,
  "validUntil" : <Some time in the future - example: "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>
x-request-id: <UUID for this request>

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

The response contains the URL to start the authorisation.

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

We post an empty body to the URL

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

{ }
```

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

```
201
ASPSP-SCA-Approach: DECOUPLED
Content-Type: application/json
x-request-id: <UUID for this request>

{
  "scaStatus" : "started",
  "chosenScaMethod" : {
    "authenticationType": "PUSH_OTP",
    "authenticationVersion": "1",
    "authenticationMethodId": "PushApp",
    "name": " Approval of the process via PushApp",
    "explanation": " The process is released via PushApp."
  },
  "authorisationId" : "<Authorisation-ID>",
  "psuMessage" : "Please use your BankApp for transaction Authorisation.",
  "_links" : [ {
    "status" : {
      "href" : "<URL for status request>"
    }
  } ]
}
```

The response contains the URL at which the status query is continued.

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

With the Decoupled procedure, the customer password can be transferred for validation.

```
PUT https://.../api/v1/consents/<Consent-ID>/authorisations/<Authorisation-ID>
accept: application/json
content-type: application/json
psu-id: <Customer’s Login-ID>
x-request-id: <UUID for this request>

{
  "psuData" : {
    "password" : "<Customer`s password>"
  }
}
```

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

```
200
ASPSP-SCA-Approach: DECOUPLED
Content-Type: application/json
x-request-id: <UUID for this request>

{
  "scaStatus" : "started",


  "_links" : [ {
    "scaStatus" : {
      "href" : "<URL for status request>"
    }
  } ]

}
```

The call validates the password.

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

This URL is used to check and verify the status of the authorisation.

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

{ }
```

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

```
200
Content-Type: application/json
x-request-id: <UUID for this request>

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

If the status of the response is "finalised", the authorisation is complete and the client's consent is valid.

Attention:

When testing with the sandbox, the result "finalised" is always returned, presuming the correct authorisation ID is specified in the link.

3.1.3 ACCOUNT QUERY

The account enquiry can only be carried out when one of the two authorisation paths, 3.1.1 or 3.1.2 has been successfully completed.

3.1.3.1 Request – GET /accounts

We can now query the customer’s accounts.

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

3.1.3.2 Response – GET /accounts

```
Content-Type: application/json
x-request-id: <UUID for this request>

{ "accounts" : [ { … } ] }
```

The response contains the customer’s accounts. We can repeat this query once a day until the consent expires.

3.2 TRIGGERING A PAYMENT

A customer wants to trigger a single payment.

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

We start with a POST-call to /{payment-service}/{payment-product}. First we will replace the parameters contained in the endpoint's path. Since we have a single payment in a pain-xml-file, we will use /payments/pain.001-sepa-credit-transfers.

The header of the request should contain any information you have about the customer in the PSU-* fields. The customer identification number PSU-ID is mandatory. Your message may be signed using the header fields Digest, Signature, and TPP-Signature-Certificate.

The message body, just contains the payment in the pain-xml-format.

```
POST https://.../api/v1/payments/pain.001-sepa-credit-transfers
accept: application/json
content-type: application/xml
psu-id: <Customer's Login-ID>
x-request-id: <UUID for this 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 for this 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"
    }
  } ]
}
```

The response contains two important pieces of information. On the one hand, we receive a unique identifier called “payment-ID”. This payment-ID allows us to start the authorisation and query the current state of our payment. On the other hand the response contains a hint about the SCA (strong customer authorisation) approach we will be using (EMBEDDED in this case).

Furthermore we note that our payment is currently in state RCVD. So the payment provider has received our payment message, but has not executed it yet. The payment will be executed, once the authorisation is done.

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

With this request we initiate the authorisation procedure.

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

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

The authorisation is started.

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

After asking the customer's password, we can transmit it to the API.

```
PUT                                 https://.../api/v1/payments/pain.001-sepa-credit-transfers/<Payment-ID>/authorisations/<Authorisation-ID>
accept: application/json
content-type: application/json
psu-id: <Customer's Login-ID> (Optional)
x-request-id: <UUID for this request>

{
  "psuData" : {
    "password" : "<Customer's password>"
  }
}
```

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 for this 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" : [ ]
}
```

The password was accepted and the customer's predefined SCA method will begin (in this example we use SMS)

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

The user will receive a TAN, which can be given to the API to finish the authorisation process.

```
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: <Customer's Login-ID> (Optional)
x-request-id: <UUID for this request>

{
  "scaAuthenticationData" : "<Customer's TAN>"
}
```

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

```
200
Content-Type: application/json
x-request-id: <UUID for this request>

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

If the TAN is correct, the authorisation is completed.

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

The current state of the payment can be queried whenever necessary (even before the payment is authorized).

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

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

```
200
Content-Type: application/json
x-request-id: <UUID for this request>

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

In this example the payment was rejected.