Coventry Building Society

Security Profile v3.0

Contents

Version control

Version	Date	Updated by	Changes made
1.0	07 Feb 2018	Coventry Building Society	Baseline version
2.0	20 Feb 2019	Coventry Building Society	Update to include new Funds Confirmation APIs and future support of OBWAC and eIDAS QWAC certificates
3.0	03 Mar 2020	Coventry Building Society	Update to include support of OBWAC and eIDAS QWAC certificates Update examples to include client id passed in request

Release Note

This release note explains what's new in The CBS Security Profile between versions.

Version 3.0 – As per the RTS, TPPs with eIDAS certificates must be allowed access to CBS APIs without requiring a further certificate. On that basis CBS has decided to only accept eIDAS QWAC or OB WAC certificates from June 2020

There will be a 3 month period before the above deadline where a TPP with an existing CBS certificate will be able to use either their CBS, OpenBanking or eIDAS certificate. To allow this CBS will be communicating with existing on-boarded TPPs the information required to allow dual access.

CBS provided certificates will be revoked and only eIDAS QWAC or OB WAC certificates accepted from June 2020.

The migration process to eIDAS or OB WAC certificates will be communicated to TPPs who have already on-boarded with CBS to facilitate a smooth transition.

As part of this change the client_id provided to a TPP when they on-board with CBS must be sent in every request.

Overview

This specification describes the authentication and authorisation given to Third Party Providers (TPPs) to receive payments, obtain funds confirmation or access account information from Coventry Building Society (CBS) accounts by our customers.

The API endpoints described here allow an AISP to:

- Create and retrieve TPP payment authorisations
- Create, retrieve and revoke TPP account access authorisations
- Create, retrieve and revoke TPP confirmation of funds authorisations

CBS has adopted the same standards as have been implemented by Open Banking. These can be found here: https://www.openbanking.org.uk/standards/

Authentication

Consent leverages the OAuth 2.0 authorization framework, allowing customers of CBS to log into applications to grant authorisation to access their account data or to initiate payments from their accounts without exposing their credentials to the TPP.

In addition to OAuth 2.0, OpenID Connect identity layer has been used to pass the AccountRequestId, PaymentId and ConsentId (created by the TPP when registering an intent to access data) within the Hybrid Flow, allowing CBS to link the intent created by the TPP to the customer who will authenticate and authorize the intent.

Request Header

Every request must include a header field called client_id with the value set to the clientId provided by CBS

```
POST https://resourcema.coventrybuildingsociety.co.uk/mga/sps/oauth/oauth20/token HTTP/1.1
client_id: {clientId value}
```

Client Types

As per OAuth2 specification, the **Confidential Client** Type has been implemented. Access to CBS API's is based on TPPs authenticating securely with our Authorization Server. TPP's must maintain the confidentiality of the client credentials which CBS will provide once a TPP successfully on-boards with CBS.

All communication between the TPP and CBS is over TLS 1.2 MA using eIDAS QWAC or OB WAC PSD2 certificates.

Grant Types

OIDC Hybrid Flow (response_type=code id_token)

Both the Payments, Funds Confirmation and Accounts APIs illustrate the use of request_type=code id_token for the OIDC Hybrid Flow implementation.

Client Credentials Grant Type using multiple scopes (scope=accounts payments)

- The Client Credentials Grant Type is used across both Payments, Funds Confirmation and Account APIs only when the TPP (AISP/PISP/CBPII) requires an Access Token (on behalf of itself) in order to access a Payment, Funds Confirmation or Accounts API resource e.g.
    - ***Payments***:

      ```
      POST /payments

      GET /payment-submissions/{PaymentSubmissionId}
      ```

    - ***Accounts:***

      ```
      POST /account-requests
      ```

- ***Funds Confirmation:***

  POST /funds-confirmation-consents

- A TPP may therefore choose to request for either a single scope e.g. accounts or for multiple scope(s) e.g. accounts payments as the TPP may want to use the same Access Token across both APIs.
- Only valid API scopes will be accepted when generating an Access Token (*accounts payments fundsconfirmations*).
- Access tokens generated by a Client Credentials grant may not return any refresh tokens (as per the OAuth 2.0 specification)
- Access tokens generated by a Client Credentials grant will expire after 3600 seconds.

Example – Client Credentials:

https://resourcema.coventrybuildingsociety.co.uk/mga/sps/oauth/oauth20/token

Request must include:

grant_type=”Client Credentials”

scope=”openId accounts”

client_id={clientId provided by CBS when TPP on-boarded}

client_secret={client secret provided by CBS when TPP on-boarded}

ID Token

- ID Tokens must be validated by the TPP (AISP/PISP/CBPII) as outlined within the OIDC Errata 1 Specification
- TPPs must use the *openbanking_intent_id* claim to populate and retrieve the IntentID (PaymentID for Payments API and AccountRequestId for the Accounts API) for any required validation.
- The full set of claims that can be represented within an ID Token are documented in the Request Object and ID Token Section of the Security Profile.
- ID Token claims (*exp* and *iat)* determine its validity.
- Returned with the Authorization Code when the Hybrid flow (code id_token) is initiated.

Access Tokens issued through Client Credentials Grant

- Only valid API scopes will be accepted when generating an Access Token (*accounts payments fundsconfirmations*).
- Access tokens generated by a Client Credentials grant may not return any refresh tokens (as per the OAuth 2.0 specification)
- Access tokens generated by a Client Credentials grant for will expire after 3600 seconds (1 hour).

Access Tokens issued through Authorization Code Grant

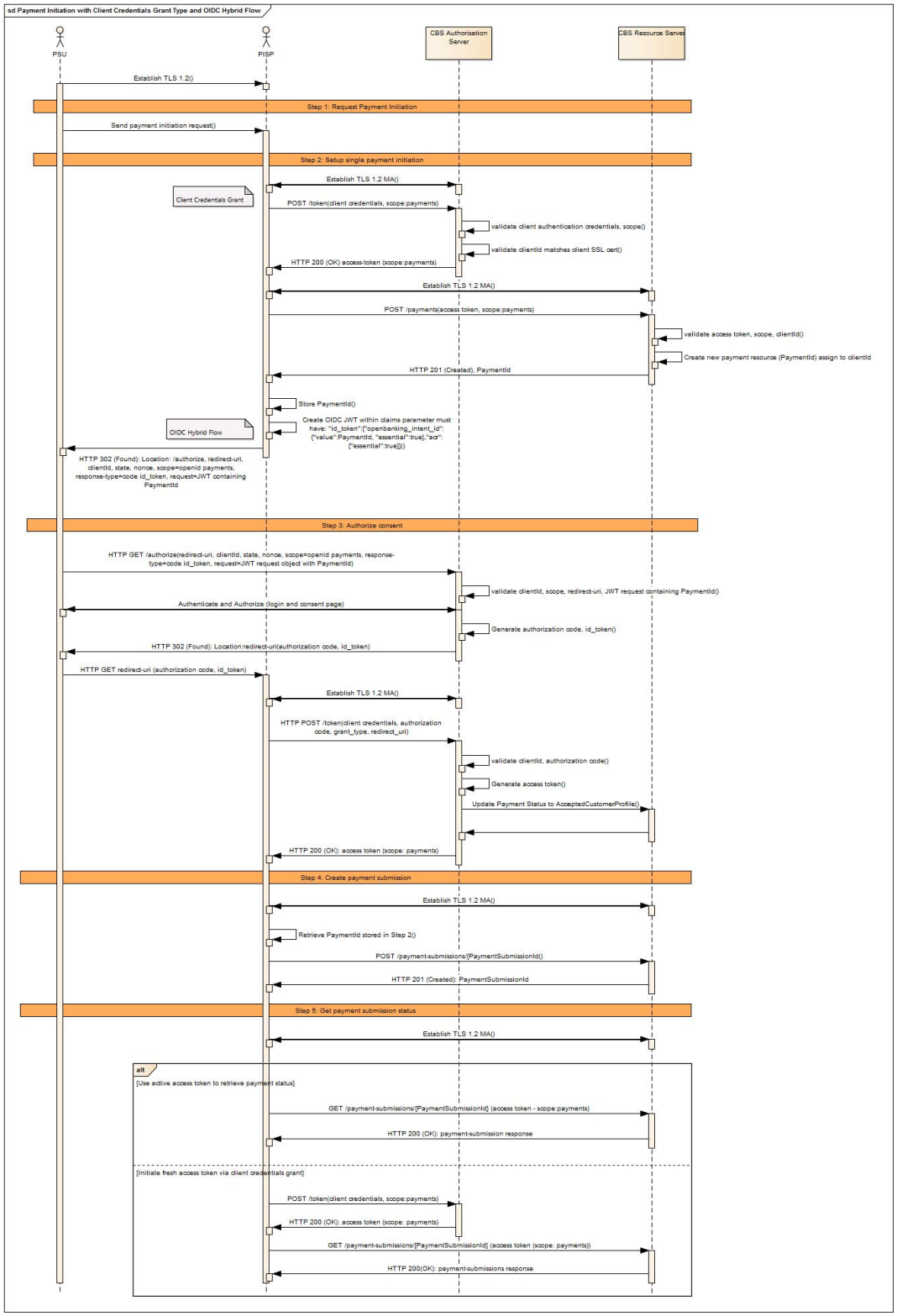
- For the Payments and Accounts APIs, the Access Token must be obtained within a Secure, Server Side Context between the TPP (AISP/PISP/CBPII) and CBS.
- Access Tokens must be validated by the TPP (AISP/PISP/CBPII) as outlined within the OIDC Errata 1 Specification
- The ***expires_in*** attribute returned by the Authorization Server when an Access Token is generated determines its validity.
- Our Access Tokens for Payment Initiation are set to expire after 3600 seconds (1 hour)
- Our Access Tokens for Account Information and Funds Confirmation are set to expire after 90 days, after which a new account or funds confirmation request should be initiated. We do not currently support Refresh Tokens.

Authorization Codes

- Authorization Codes must be validated by the TPP (AISP/PISP) as outlined within the OIDC Errata 1 Specification
- OAuth 2.0 Specification suggests an Authorization Code should be short lived to a maximum of 10 minutes. Any codes exceeding this limit to be rejected.
- **CBS authorization codes will expire after 5 minutes.**

Success Flows - Payment API Specification

Payment Initiation with Client Credentials Grant Type and OIDC Hybrid Flow



Client Credentials Grant Type (OAuth 2.0)

Summary

This grant type is used by the PISP in Step 2 to setup a single payment with CBS.

1. The client_id must be included within the Request Header
2. The PISP initiates an Authorization request using valid Client Credentials Grant type and scope(s)
3. The CBS Authorization Server validates the Client Authentication request from the PISP and generates an Access Token response where the request is valid
4. The PISP uses the Access Token to create a new Payment resource against the CBS Resource Server
5. The CBS Resource server responds with the PaymentId for the resource it has created.
6. The Client Credentials Grant may optionally be used by the PISP in Step 5 to retrieve the status of a Payment or Payment-Submission where no active Access Token is available.

OIDC Hybrid Flow

Summary

- The client_id must be included within the Request Header
- The Hybrid flow is the recommendation from the OB Security Profile and the FAPI Specification for R/W. The Hybrid flow prevents IdP mixup attacks as documented by Nat Sakimura - Cut and Paste OAuth 2.0 Attack
- This is initiated at the end of Step 2 by the PISP after the PaymentId is generated by CBS and returned to the PISP.
- This is used in a redirect across the PSU and CBS in Step 3 in order for the PSU to authorize consent with CBS - for the PISP to proceed with the Payment.
- This is used across the PISP and CBS in Step 4 by exchanging the Authorization Code for an Access Token in order to create the Payment-Submission resource.

Non-Normative HTTP Request and Response Examples

Step 1 - Request Payment Initiation

There are no Requests and Responses against the Payments API in this Step for the PSU, PISP and CBS.

Step 2 - Setup Single Payment Initiation

1. PISP obtains an Access Token using a Client Credentials Grant Type. The scope *payments* must be used. When an Access Token expires, the PISP will need to re-request for another Access Token using the same request below.

Request: Client Credentials using private_key_jwt

```
POST  / mga/sps/oauth/oauth20/token
HTTP/1.1
Host: https://
resourcema.coventrybuildingsociety.co.uk
client_id: tppclientid
Content-Type: application/x-www-form-
urlencoded
Accept: application/json
grant_type=client_credentials
&scope=payments
```

Response: Client Credentials

```
Content-Length: 1103
Content-Type: application/json
Date: Mon, 26 Jun 2017 15:18:28 GMT
{
    "access_token":
"2YotnFZFEjr1zCsicMWpAA",
    "expires_in": 3600,
    "token_type": "bearer",
    "scope":"payments"
}
```

```
&client_id=tppclientid
&client_secret=tppclientsecret
```

2. PISP uses the Access Token (with *payments* scope) from CBS to invoke the Payments API.

Request: Payments API

```
POST /payments HTTP/1.1
Authorization: Bearer
2YotnFZFEjr1zCsicMWpAA
x-idempotency-key: FRESCO.21302.GFX.20
x-fapi-financial-id: OB/2017/001
x-fapi-customer-last-logged-time: 2017-
06-13T11:36:09
x-fapi-customer-ip-address: 104.25.212.99
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id: tppclientid
Content-Type: application/json
Accept: application/json

{
  "Data": {
    "Initiation": {
      "InstructionIdentification":
"ACME412",
      "EndToEndIdentification":
"FRESCO.21302.GFX.20",
      "InstructedAmount": {
        "Amount": "165.88",
        "Currency": "GBP"
      },
      "CreditorAccount": {
        "SchemeName":
"SortCodeAccountNumber",
        "Identification":
"08080021325698",
        "Name": "ACME Inc",
        "SecondaryIdentification": "0002"
      },
      "RemittanceInformation": {
        "Reference": "FRESCO-101",
        "Unstructured": "Internal ops
code 5120101"
      }
    }
  },
  "Risk": {
    "PaymentContextCode":
"EcommerceGoods",
    "MerchantCategoryCode": "5967",
    "MerchantCustomerIdentification":
"053598653254",
    "DeliveryAddress": {
```

Response: Payments API

```
HTTP/1.1 201 Created
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "PaymentId": "612d9e8e-074b-490b-
bc8a-0df5287a0dbc",
    "Status":
"AcceptedTechnicalValidation",
    "CreationDateTime": "2017-06-
05T15:15:13+00:00",
    "Initiation": {
      "InstructionIdentification":
"ACME412",
      "EndToEndIdentification":
"FRESCO.21302.GFX.20",
      "InstructedAmount": {
        "Amount": "165.88",
        "Currency": "GBP"
      },
      "CreditorAccount": {
        "SchemeName":
"SortCodeAccountNumber",
        "Identification":
"08080021325698",
        "Name": "ACME Inc",
        "SecondaryIdentification": "0002"
      },
      "RemittanceInformation": {
        "Reference": "FRESCO-101",
        "Unstructured": "Internal ops
code 5120101"
      }
    }
  },
  "Risk": {
    "PaymentContextCode":
"EcommerceGoods",
    "MerchantCategoryCode": "5967",
    "MerchantCustomerIdentification":
"053598653254",
    "DeliveryAddress": {
      "AddressLine": [
        "Flat 7",
        "Acacia Lodge"
```

```
      "AddressLine": [
        "Flat 7",
        "Acacia Lodge"
      ],
      "StreetName": "Acacia Avenue",
      "BuildingNumber": "27",
      "PostCode": "GU31 2ZZ",
      "TownName": "Sparsholt",
      "CountySubDivision": [
        "Wessex"
      ],
      "Country": "UK"
    }
  }
}
```

```
      ],
      "StreetName": "Acacia Avenue",
      "BuildingNumber": "27",
      "PostCode": "GU31 2ZZ",
      "TownName": "Sparsholt",
      "CountySubDivision": [
        "Wessex"
      ],
      "Country": "UK"
    }
  },
  "Links": {
    "Self":
"https://resourcema.coventrybuildingsociety.co.uk/open-
banking/v1.0/payments/58923"
  },
  "Meta": {}
}
```

Step 3 - Authorize Consent

1. PISP receives a PaymentId from CBS. The PISP then creates an Authorization request (using a signed JWT Request containing the PaymentID as a claim) for the PSU to consent to the Payment directly with CBS. The request is an OIDC Hybrid flow (requesting for Code and id_token). The same redirect URL which was submitted to CBS when the TPP on-boarded must be used.

Request: OIDC Hybrid Flow

```
GET /cbs/authorize?
client_id: tppclientid
response_type=code id_token
&state=af0ifjsldkj
&scope=openid payments
&nonce=n-0S6_WzA2Mj
&redirect_uri=https://api.mytpp.com/cb
&request=CJleHAiOjE0OTUxOTk1ODd.....JjVqs
Duushgpwp0E.5leGFtcGxlI
iwianRpIjoiM....JleHAiOjE0.olnx_YKAm2J1rb
pOP8wGhi1BDNHJjVqsDuushgpwp0E
```

Response: OIDC Hybrid Flow

```
HTTP/1.1 302 Found
  Location: https://api.mytpp.com/cb#
    code=SplxlOBeZQQYbYS6WxSbIA
    &id_token=eyJ0 ... NiJ9.eyJ1c ...
I6IjIifX0.DeWt4Qu ... ZXso
    &state=af0ifjsldkj
```

Non-Base64 encoded example of the request parameter object

```
{
    "alg": "",
    "kid": "GxlIiwianVqsDuushgjE0OTUxOTk"
}
.
{
   "iss": "https://
resourcema.coventrybuildingsociety.co.uk
",
   "aud": "s6BhdRkqt3",
   "response_type": "code id_token",
   "client_id": "s6BhdRkqt3",
```

```
   "redirect_uri":
"https://api.mytpp.com/cb",
   "scope": "openid payments accounts",
   "state": "af0ifjsldkj",
   "nonce": "n-0S6_WzA2Mj",
   "max_age": 86400,
   "claims":
    {
     "userinfo":
      {
       "openbanking_intent_id": {"value":
"612d9e8e-074b-490b-bc8a-0df5287a0dbc",
"essential": true}
      },
     "id_token":
      {
       "openbanking intent id": {"value":
"612d9e8e-074b-490b-bc8a-0df5287a0dbc",
"essential": true},
       "acr": {"essential": true,
               "values":
["urn:openbanking:psd2:sca",

"urn:openbanking:psd2:ca"]}}}
      }
    }
}
```

2. The PSU is then redirected to the PISP. The PISP will now possess the Authorization Code and ID Token from CBS. Note at this point, there is no Access Token. The PISP will now introspect the ID Token and use it to check:

- The hash of the Authorization Code to prove it hasn't been tampered with during redirect (comparing the hash value against the c_hash attribute in ID Token)
- The hash of the State to prove it hasn't been tampered with during redirect (comparing the state hash value against the s_hash attribute in the ID Token)

Example: ID Token

```
{
  "alg": "RS256",
  "kid": "12345",
  "typ": "JWT"
}
.
{
   "iss": "https://
resourcema.coventrybuildingsociety.co.uk
",
   "iat": 1234569795,
   "sub": "urn:alphabank:payment:58923",
   "acr": "urn:openbanking:psd2:ca",
```

```
   "openbanking_intent_id":
"urn:alphabank:payment: 612d9e8e-074b-
490b-bc8a-0df5287a0dbc",
   "aud": "s6BhdRkqt3",
   "nonce": "n-0S6_WzA2Mj",
   "exp": 1311281970,
   "s_hash": "76sa5dd",
   "c_hash": "asd097d"
  }
.
{
```

2. Once the state and code validations have been confirmed as successful by use of the ID token, the PISP will proceed to obtain an Access Token from CBS using the Authorization Code they now possess. The PISP will present its Authorization Code together with the private_key_jwt. The Access Token is required by the PISP in order to submit the Payment on behalf of the PSU. The *payments* scope should already be associated with the Authorization Code generated in the previous step.

Request: Access Token Request using Authorization Code and private_key_jwt

```
POST / mga/sps/oauth/oauth20/token
HTTP/1.1
Host: https://
resourcema.coventrybuildingsociety.co.uk
client_id: tppclientid
Content-Type: application/x-www-form-
urlencoded
Accept: application/json
grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https://api.mytpp.com/cb
&client_assertion_type=
    urn%3Aietf%3Aparams%3Aoauth%3Aclient-
assertion-type%3Ajwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpc3MiOiJodHRw
czovL2p3dC1pZHAuZXhhbXBsZS5jb20iLCJzdWIiO
iJtYWlsdG86bWlrZUBleGFtcGxlLmN
vbSIsIm5iZiI6MTQ5OTE4MzYwMSwiZXhwIjoxNDk5
MTg3MjAxLCJpYXQiOjE0OTkxODM2MD
EsImp0aSI6ImlkMTIzNDU2IiwidHlwIjoiaHR0cHM
6Ly9leGFtcGxlLmNvbS9yZWdpc3Rlc
iJ9.SAxPMaJK_wYl_W2idTQASjiEZ4UoI7-
P2SbmnHKr6LvP8ZJZX6JlnpK_xClJswAni1T
p1UnHJslc08JrexctaeEIBrqwHG18iBcWKjhHK2Tv
5m4nbTsSi1MFQOlMUTRFq3_LQiHqV2
M8Hf1v9q9YaQqxDa4MK0asDUtE_zYMHz8kKDb-jj-
Vh4mVDeM4_FPiffd2C5ckjkrZBNOK0
01Xktm7xTqX6fk56KTrejeA4x6D_1ygJcGfjZCv6K
nki7Jl-6MfwUKb9ZoZ9LiwHf5lLXPuy
_QrOyM0pONWKj9K4Mj7I4GPGvzyVqpaZUgjcOaZY_
rlu_p9tnSlE781dDLuw
```

Response: Access Token

```
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
}
```

Non-Base64 encoded example of the request parameter object

```
{
  "alg": "RS256",
  "kid": "12345",
  "typ": "JWT"
}
.
{
  "iss": "s6BhdRkqt3",
  "sub": "s6BhdRkqt3",
  "exp": 1499187201,
  "iat": 1499183601,
  "jti": "id123456",
  "aud":
"https://resourcema.coventrybuildingsociety.co.uk/mga/sps/oauth/oauth20/token "
}
```

Step 4 - Create Payment-Submission

1. The PISP has an Access Token which can be used to Create a Payment-Submission (Step 4). The PISP must obtain the PaymentId (Intent-ID) so that the Payment request is associated with the correct PaymentId. This can be sourced from:

1. The PaymentId claim from the ID Token (default). The PISP will need to locate the claim attribute associated with the PaymentId.

The PISP can now invoke the /payment-submissions endpoint to commit the Payment using the Access Token and PaymentId in the payload of the request. This example is sourced from the Payment Initiation API Specification

Request: payment-submissions

```
POST /payment-submissions HTTP/1.1
Authorization: Bearer S1AV32hkKG
x-idempotency-key: FRESNO.1317.GFX.22
x-fapi-financial-id: OB/2017/001
x-fapi-customer-last-logged-time: 2017-
06-13T11:36:09
x-fapi-customer-ip-address: 104.25.212.99
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id: tppclientid
Content-Type: application/json
Accept: application/json

{
  "Data": {
    "PaymentId": "612d9e8e-074b-490b-
bc8a-0df5287a0dbc",
```

Response: payment-submissions

```
HTTP/1.1 201 Created
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "PaymentSubmissionId": "58923-001",
    "PaymentId": "612d9e8e-074b-490b-
bc8a-0df5287a0dbc",
    "Status":
"AcceptedSettlementInProcess",
    "CreationDateTime": "2017-06-
05T15:15:22+00:00"
  },
  "Links": {
```

```
    "Initiation": {
      "InstructionIdentification":
"ACME412",
      "EndToEndIdentification":
"FRESCO.21302.GFX.20",
      "InstructedAmount": {
        "Amount": "165.88",
        "Currency": "GBP"
      },
      "CreditorAccount": {
        "SchemeName":
"SortCodeAccountNumber",
        "Identification":
"08080021325698",
        "Name": "ACME Inc",
        "SecondaryIdentification": "0002"
      },
      "RemittanceInformation": {
        "Reference": "FRESCO-101",
        "Unstructured": "Internal ops
code 5120101"
      }
    }
  },
  "Risk": {
    "PaymentContextCode":
"EcommerceGoods",
    "MerchantCategoryCode": "5967",
    "MerchantCustomerIdentification":
"053598653254",
    "DeliveryAddress": {
      "AddressLine": [
        "Flat 7",
        "Acacia Lodge"
      ],
      "StreetName": "Acacia Avenue",
      "BuildingNumber": "27",
      "PostCode": "GU31 2ZZ",
      "TownName": "Sparsholt",
      "CountySubDivision": [
        "Wessex"
      ],
      "Country": "UK"
    }
  }
}
```

```
    "Self": "https://
resourcema.coventrybuildingsociety.co.uk
/open-banking/v1.0/payment-
submissions/58923-001"
  },
  "Meta": {}
}
```

Step 5 - Get Payment-Submission Status

1. The PISP can query for the status of a Payment-Submission by invoking the /payment-submissions using the known PaymentSubmissionId. This can use an existing access token with *payments* scope or the PISP can obtain a fresh access token by replaying the client credentials grant request as per Step 2 - Setup Single Payment Initiation.

Request: payment-submissions/{PaymentSubmissionId}

```
GET /payment-submissions/58923-001
HTTP/1.1
Authorization: Bearer S1AV32hkKG
x-fapi-financial-id: OB/2017/001
x-fapi-customer-last-logged-time: 2017-
06-13T11:36:09
x-fapi-customer-ip-address: 104.25.212.99
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id: tppclientid
Accept: application/json
```

Response: payment-submissions

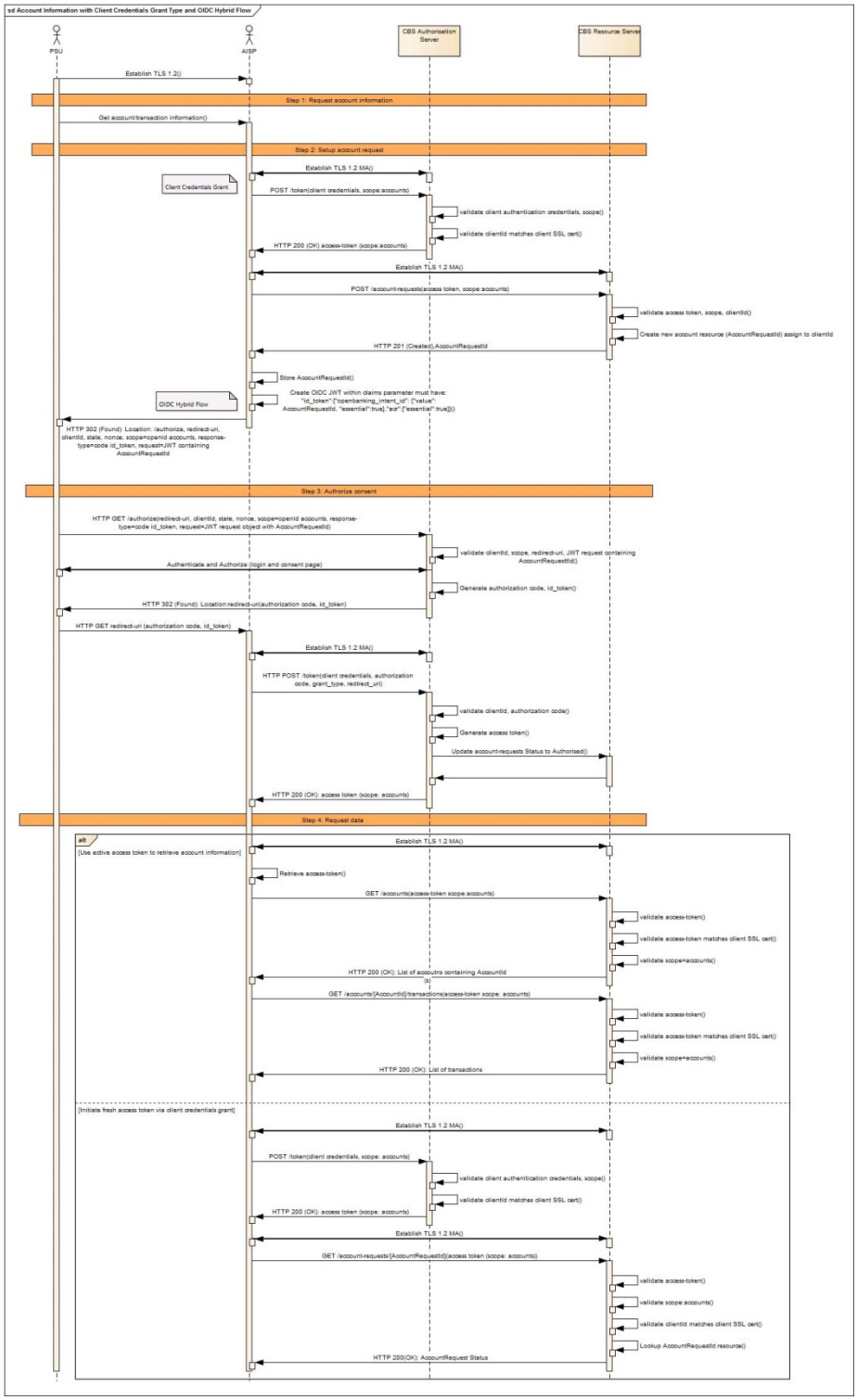
```
HTTP/1.1 200 OK
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "PaymentSubmissionId": "58923-001",
    "PaymentId": "612d9e8e-074b-490b-
bc8a-0df5287a0dbc",
    "Status":
"AcceptedSettlementInProcess",
    "CreationDateTime": "2017-06-
05T15:15:22+00:00"
  },
  "Links": {
    "Self": "https://
resourcema.coventrybuildingsociety.co.uk
/open-banking/v1.0/payment-
submissions/58923-001"
  },
  "Meta": {}
}
```

2. A PISP can also optionally query for the status of a Payment resource by invoking /payments/{PaymentId}. This can use an existing access token with *payments* scope or the PISP can obtain a fresh access token by replaying the client credentials grant request as per Step 2 - Setup Single Payment Initiation.

Success Flows - Account API Specification

Account and Transaction Information with Client Credentials Grant Type and OIDC Hybrid Flow



Client Credentials Grant Type (OAuth 2.0)

Summary

This grant type is used by the AISP in Step 2 to register an intent for the PSU to allow the AISP to retrieve their Account information from CBS.

1. The client_id must be included within the Request Header
2. The AISP initiates an Authorization request using valid Client Credentials Grant type and scope(s)
3. The CBS Authorization Server validates the Client Authentication request from the AISP and generates an Access Token response where the request is valid
4. The AISP uses the Access Token to create a new Account Request resource against the CBS Resource Server
5. The CBS Resource server responds with the AccountRequestId representing the resource it has created.

OIDC Hybrid Flow

Summary

1. The client_id must be included within the Request Header
2. This is initiated at the end of Step 2 by the AISP after the AccountRequestId is generated by CBS and returned to the AISP.
3. This is used in a redirect across the PSU and CBS in Step 3 in order for the PSU to authorize consent with CBS - for the AISP to proceed with the requesting Account information.
4. This is used across the AISP and CBS in Step 4 by swapping the Authorization Code for an Access Token in order to retrieve PSU Account information.

Non-Normative HTTP Request and Response Examples

Step 1 - Request Account Information

There are no Requests and Responses against the Accounts and Transactions API in this Step for the PSU, AISP and CBS.

Step 2 - Setup Account Request

1. AISP obtains an Access Token using a Client Credentials Grant Type. The scope accounts must be used. When an Access Token expires, the AISP will need to re-request for another Access Token using the same request below.

Request: Client Credentials using private_key_jwt

```
POST  / mga/sps/oauth/oauth20/token
HTTP/1.1
Host:
https://resourcema.coventrybuildingsociet
y.co.uk
client_id: tppclientid
Content-Type: application/x-www-form-
urlencoded
Accept: application/json
grant_type=client_credentials
&scope=accounts
&client_id=tppclientid
&client_secret=tppclientsecret
```

Response: Client Credentials

```
Content-Length: 1103
Content-Type: application/json
Date: Mon, 26 Jun 2017 15:18:28 GMT
{
    "access_token":
"2YotnFZFEjr1zCsicMWpAA",
    "expires_in": 3600,
    "token_type": "bearer",
    "scope":"accounts"
}
```

2. AISP uses the Access Token (with *accounts* scope) from CBS to invoke the Accounts API. This example is sourced directly from the Account and Transactions API Specification

Request: Accounts API

```
POST /account-requests HTTP/1.1
Authorization: Bearer
2YotnFZFEjr1zCsicMWpAA
x-fapi-financial-id: OB/2017/001
x-fapi-customer-last-logged-time: 2017-
06-13T11:36:09
x-fapi-customer-ip-address: 104.25.212.99
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id: tppclientid
Content-Type: application/json
Accept: application/json

{
  "Data": {
    "Permissions": [
      "ReadAccountsDetail",
      "ReadBalances",
      "ReadBeneficiariesDetail",
      "ReadDirectDebits",
      "ReadProducts",
      "ReadStandingOrdersDetail",
      "ReadTransactionsCredits",
      "ReadTransactionsDebits",
      "ReadTransactionsDetail"
    ],
    "ExpirationDateTime": "2017-05-
02T00:00:00+00:00",
    "TransactionFromDateTime": "2017-05-
03T00:00:00+00:00",
    "TransactionToDateTime": "2017-12-
03T00:00:00+00:00"
  },
  "Risk": {}
}
```

Response: Accounts API

```
HTTP/1.1 201 Created
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "AccountRequestId": "612d9e8e-074b-
490b-bc8a-0df5287a0dbc",
    "Status": "AwaitingAuthorisation",
    "CreationDateTime": "2017-05-
02T00:00:00+00:00",
    "Permissions": [
      "ReadAccountsDetail",
      "ReadBalances",
      "ReadBeneficiariesDetail",
      "ReadDirectDebits",
      "ReadProducts",
      "ReadStandingOrdersDetail",
      "ReadTransactionsCredits",
      "ReadTransactionsDebits",
      "ReadTransactionsDetail"
    ],
    "ExpirationDateTime": "2017-08-
02T00:00:00+00:00",
    "TransactionFromDateTime": "2017-05-
03T00:00:00+00:00",
    "TransactionToDateTime": "2017-12-
03T00:00:00+00:00"
  },
  "Risk": {},
  "Links": {
    "Self": "/account-requests/612d9e8e-
074b-490b-bc8a-0df5287a0dbc"
  },
  "Meta": {
    "TotalPages": 1
  }
}
```

Step 3 - Authorize Consent

1. AISP receives a AccountRequestId from CBS. The AISP then creates an Authorization request (using a signed JWT Request containing the AccountRequestId as a claim) for the PSU to consent to the Account request directly with CBS. The request is an OIDC Hybrid flow (requesting for Code and id_token)

Request: OIDC Hybrid Flow

```
GET /cbs/authorize?
response_type=code id_token
&client_id=s6BhdRkqt3
&state=af0ifjsldkj
&scope=openid accounts
&nonce=n-0S6_WzA2Mj
&redirect_uri=https://api.mytpp.com/cb
&request=CJleHAiOjE0OTUxOTk1ODd.....JjVqs
Duushgpwp0E.5leGFtcGxlI
iwianRpIjoiM....JleHAiOjE0.olnx_YKAm2J1rb
pOP8wGhi1BDNHJjVqsDuushgpwp0E
```

Response: OIDC Hybrid Flow

```
HTTP/1.1 302 Found
  Location: https://api.mytpp.com/cb#
    code=SplxlOBeZQQYbYS6WxSbIA
    &id_token=eyJ0 ... NiJ9.eyJ1c ...
I6IjIifX0.DeWt4Qu ... ZXso
    &state=af0ifjsldkj
```

Non-Base64 encoded example of the request parameter object

```
{
    "alg": "RS256",
    "kid": "GxlIiwianVqsDuushgjE0OTUxOTk"
}
.
{
   "iss":
"https://resourcema.coventrybuildingsocie
ty.co.uk",
   "aud": "s6BhdRkqt3",
   "response_type": "code id_token",
   "client_id": "s6BhdRkqt3",
   "redirect_uri":
"https://api.mytpp.com/cb",
   "scope": "openid accounts",
   "state": "af0ifjsldkj",
   "nonce": "n-0S6_WzA2Mj",
   "max_age": 86400,
   "claims":
    {
     "userinfo":
      {
       "openbanking_intent_id": {"value":
"612d9e8e-074b-490b-bc8a-0df5287a0dbc",
"essential": true}
      },
     "id_token":
      {
       "openbanking_intent_id": {"value":
"612d9e8e-074b-490b-bc8a-0df5287a0dbc",
"essential": true},
       "acr": {"essential": true,
                "values":
["urn:openbanking:psd2:sca",

"urn:openbanking:psd2:ca"]}}
      }
    }
}
```

2. The PSU is then redirected to the AISP. The AISP will now possess the Authorization Code and ID Token from CBS. Note at this point, there is no Access Token. The AISP will now introspect the ID Token and use it as a detached signature to check:

- The hash of the Authorization Code to prove it hasn't been tampered with during redirect (comparing the hash value against the c_hash attribute in ID Token)
- The hash of the State to prove it hasn't been tampered with during redirect (comparing the state hash value against the s_hash attribute in the ID Token)

Example: ID Token

```
{
   "alg": "RS256",
  "kid": "GxlIiwianVqsDuushgjE0OTUxOTk"
}
.
{
   "iss": "https://
resourcema.coventrybuildingsociety.co.uk
",
   "iat": 1234569795,
   "sub":
"urn:alphabank:accountRequestId:88379",
   "acr": "urn:openbanking:psd2:ca",
   "openbanking_intent_id":
"urn:alphabank:accountRequestId:612d9e8e-
074b-490b-bc8a-0df5287a0dbc",
   "aud": "s6BhdRkqt3",
   "nonce": "n-0S6_WzA2Mj",
   "exp": 1311281970,
   "s_hash": "76sa5dd",
   "c_hash": "asd097d"
  }
```

3. Once the state and code validations have been confirmed as successful by use of the ID token, the AISP will proceed to obtain an Access Token from CBS using the Authorization Code they now possess. The AISP will present its Authorization Code together with the private_key_jwt. The Access Token is required by the AISP in order to access PSU Account information. The accounts scope should already be associated with the Authorization Code generated in the previous step.

Request: Access Token request using Authorization Code and private_key_jwt

```
POST /mga/sps/oauth/oauth20/token
HTTP/1.1
Host: https://
resourcema.coventrybuildingsociety.co.uk
client_id: tppclientid
Content-Type: application/x-www-form-
urlencoded
Accept: application/json
```

Response: Access Token

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "access_token": "SlAV32hkKG",
 "token_type": "Bearer",
```

```
grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https://api.mytpp.com/cb
&client_assertion_type=
    urn%3Aietf%3Aparams%3Aoauth%3Aclient-
assertion-type%3Ajwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpc3MiOiJodHRw
czovL2p3dC1pZHAuZXhhbXBsZS5jb20iLCJzdWIiO
iJtYWlsdG86bWlrZUBleGFtcGxlLmN
vbSIsIm5iZiI6MTQ5OTE4MzYwMSwiZXhwIjoxNDk5
MTg3MjAxLCJpYXQiOjE0OTkxODM2MD
EsImp0aSI6ImlkMTIzNDU2IiwidHlwIjoiaHR0cHM
6Ly9leGFtcGxlLmNvbS9yZWdpc3Rlc
iJ9.SAxPMaJK_wYl_W2idTQASjiEZ4UoI7-
P2SbmnHKr6LvP8ZJZX6JlnpK_xClJswAni1T
p1UnHJslc08JrexctaeEIBrqwHG18iBcWKjhHK2Tv
5m4nbTsSi1MFQOlMUTRFq3_LQiHqV2
M8Hf1v9q9YaQqxDa4MK0asDUtE_zYMHz8kKDb-jj-
Vh4mVDeM4_FPiffd2C5ckjkrZBNOK0
01Xktm7xTqX6fk56KTrejeA4x6D_1ygJcGfjZCv6K
nki7Jl-6MfwUKb9ZoZ9LiwHf5lLXPuy
_QrOyM0pONWKj9K4Mj7I4GPGvzyVqpaZUgjcOaZY_
rlu_p9tnSlE781dDLuw
```

```
 "expires_in": 7776000
}
```

Non-Base64 encoded example of the request parameter object

```
{
  "alg": "RS256",
  "kid": "12345",
  "typ": "JWT"
}
.
{
  "iss": "s6BhdRkqt3",
  "sub": "s6BhdRkqt3",
  "exp": 1499187201,
  "iat": 1499183601,
  "jti": "id123456",
  "aud": "https://
resourcema.coventrybuildingsociety.co.uk
/as/token.oauth2"
}
```

Step 4 - Request Account Data

1. The AISP can use the Access Token to retrieve Accounts (bulk or specific). The following examples are from the Account and Transaction API Specification

Where the initial Access Token expires, the AISP can use the Refresh token in order to obtain a fresh Access Token.

Example request against Accounts resource

Request: GET /Accounts API

```
GET /accounts HTTP/1.1
Authorization: Bearer S1AV32hkKG
x-fapi-financial-id: OB/2017/001
x-fapi-customer-last-logged-time: 2017-
06-13T11:36:09
x-fapi-customer-ip-address: 104.25.212.99
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id: tppclientid
Accept: application/json
```

Response: GET /Accounts API

```
HTTP/1.1 200 OK
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "Account": [
      {
        "AccountId": "22289",
        "Currency": "GBP",
        "Nickname": "Bills",
        "Account": {
          "SchemeName":
"SortCodeAccountNumber",
          "Identification":
"80200110203345",
          "Name": "Mr Kevin",
          "SecondaryIdentification":
"00021"
      }
    ]
  },
  "Links": {
    "Self": "/accounts/"
  },
  "Meta": {
    "TotalPages": 1
  }
}
```

Example request for a specific Account Id

Request: GET /Accounts/22289 API

```
GET /accounts/22289 HTTP/1.1
Authorization: Bearer S1AV32hkKG
x-fapi-financial-id: OB/2017/001
x-fapi-customer-last-logged-time: 2017-
06-13T11:36:09
x-fapi-customer-ip-address: 104.25.212.99
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id: tppclientid
Accept: application/json
```

Response: GET /Accounts/22289 API

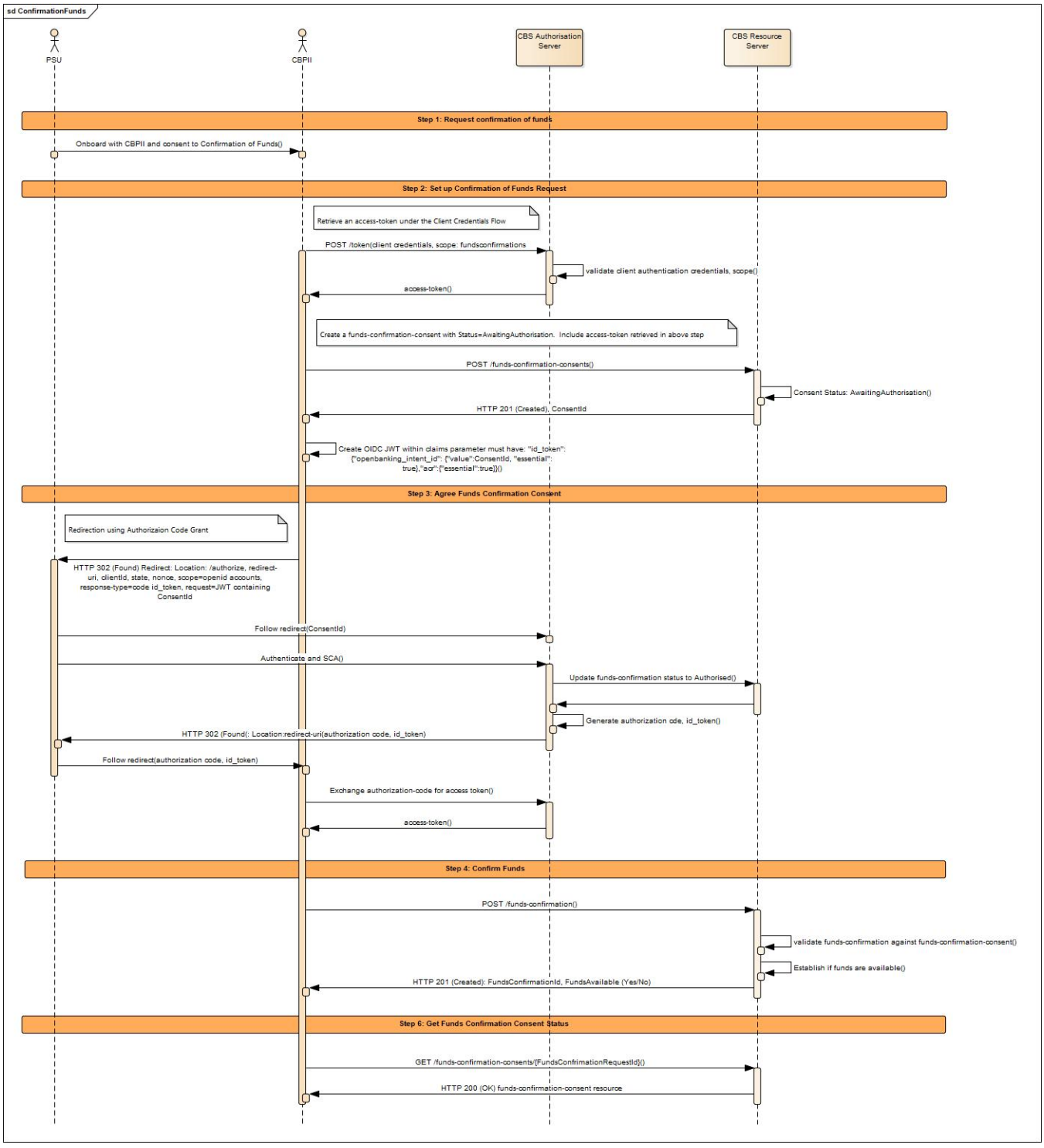
```
HTTP/1.1 200 OK
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "Account": [
      {
        "AccountId": "22289",
        "Currency": "GBP",
        "Nickname": "Bills",
        "Account": {
          "SchemeName":
"SortCodeAccountNumber",
          "Identification":
"80200110203345",
```

```
          "Name": "Mr Kevin",
          "SecondaryIdentification":
"00021"
        }
      }
    ]
  },
  "Links": {
    "Self": "/accounts/22289"
  },
  "Meta": {
    "TotalPages": 1
  }
}
```

Success Flows – Funds Confirmation API Specification

Funds Confirmation with Client Credentials Grant Type and OIDC Hybrid Flow



Client Credentials Grant Type (OAuth 2.0)

Summary

This grant type is used by the CBPII in Step 2 to register an intent for the PSU to allow the CBPII to retrieve their Funds Confirmation information from CBS.

1. The client_id must be included within the Request Header
2. The CBPII initiates an Authorization request using valid Client Credentials Grant type and scope(s)
3. The CBS Authorization Server validates the Client Authentication request from the CBPII and generates an Access Token response where the request is valid
4. The CBPII uses the Access Token to create a new Funds Confirmation Consent Request resource against the CBS Resource Server
5. The CBS Resource server responds with the ConsentId representing the resource it has created.

OIDC Hybrid Flow

Summary

1. The client_id must be included within the Request Header
2. This is initiated at the end of Step 2 by the CBPII after the ConsentId is generated by CBS and returned to the CBPII.
3. This is used in a redirect across the PSU and CBS in Step 3 in order for the PSU to authorize consent with CBS - for the CBPII to proceed with the requesting Funds Confirmation information.
4. This is used across the CBPII and CBS in Step 4 by swapping the Authorization Code for an Access Token in order to retrieve PSU Funds Confirmation information.

Non-Normative HTTP Request and Response Examples

Step 1 - Request Funds Confirmation

There are no Requests and Responses against the Funds Confirmation API in this Step for the PSU, CBPII and CBS.

Step 2 - Setup Funds Confirmation Request

1. CBPII obtains an Access Token using a Client Credentials Grant Type. The scope fundsconfirmations must be used. When an Access Token expires, the CBPII will need to re-request for another Access Token using the same request below.

Request: Client Credentials

```
POST  / mga/sps/oauth/oauth20/token
HTTP/1.1
Host:
https://resourcema.coventrybuildingsociet
y.co.uk
client_id: tppclientid
Content-Type: application/x-www-form-
urlencoded
Accept: application/json
grant_type=client_credentials
&scope= fundsconfirmations
&client_id=tppclientid
&client_secret=tppclientsecret
```

Response: Client Credentials

```
Content-Length: 1103
Content-Type: application/json
Date: Mon, 26 Jun 2017 15:18:28 GMT
{
    "access_token":
"2YotnFZFEjr1zCsicMWpAA",
    "expires_in": 3600,
    "token_type": "bearer",
    "scope":"fundsconfirmations"
}
```

2. CBPII uses the Access Token (with *fundsconfirmations* scope) from CBS to invoke the Funds Confirmation API. This example is sourced directly from the Funds Confirmation API Specification

Request: Funds Confirmation API

```
POST /funds-confirmation-consents
HTTP/1.1
Content-Type: application/json
Authorization: Bearer
2YotnFZFEjr1zCsicMWpAA
Accept: application/json; charset=utf-8
x-fapi-financial-id: I4mth3R3-4p3r-411t-
hing-5withh33dfu1
x-fapi-customer-last-logged-time: Mon, 13
Nov 2017 19:49:37 GMT
x-fapi-customer-ip-address: 92.11.92.11
x-fapi-interaction-id: hook5i13-ntIg-
4th3-rP41-3ro535touch3
X-Client-Id:tppclientid


{
  "Data": {
    "DebtorAccount": {
      "SchemeName": "UK.OBIE.IBAN",
      "Identification": "GB76LOYD30949301
273801
    },
    "ExpirationDateTime": "2017-05-
02T00:00:00+00:00"
  }
}
```

Response: Funds Confirmation API

```
HTTP/1.1 201 Created
Content-Type: application/json
x-fapi-interaction-id: hook5i13-ntIg-
4th3-rP41-3ro535touch3

{
  "Data": {
    "ConsentId": "123456",
    "CreationDateTime": "2017-05-
02T00:00:00+00:00",
    "Status": "AwaitingAuthorisation",
    "StatusUpdateDateTime": "2017-05-
02T00:00:00+00:00",
    "ExpirationDateTime": "2017-05-
02T00:00:00+00:00",
    "DebtorAccount": {
      "SchemeName": "UK.OBIE.IBAN",
      "Identification": "GB76LOYD30949301
273801
    }
  },
  "Links": {
    "Self": "https://
resourcema.coventrybuildingsociety.co.uk
/open-banking/v2.0/funds-confirmation-
consents/88379"
  },
  "Meta": {}
}
```

Step 3 – Agree Funds Confirmation Consent

2. CBPII receives a ConsentId from CBS. The CBPII then creates an Authorization request (using a signed JWT Request containing the ConsentId as a claim) for the PSU to consent to the Funds Confirmation request directly with CBS. The request is an OIDC Hybrid flow (requesting for Code and id_token)

Request: OIDC Hybrid Flow

```
GET /cbs/authorize?
response_type=code id_token
&state=af0ifjsldkj
&scope=openid fundsconfirmations
&nonce=n-0S6_WzA2Mj
&redirect_uri=https://api.mytpp.com/cb
&request=CJleHAiOjE0OTUxOTk1ODd.....JjVqs
Duushgpwp0E.5leGFtcGxlI
```

Response: OIDC Hybrid Flow

```
HTTP/1.1 302 Found
  Location: https://api.mytpp.com/cb#
    code=SplxlOBeZQQYbYS6WxSbIA
    &id_token=eyJ0 ... NiJ9.eyJ1c ...
I6IjIifX0.DeWt4Qu ... ZXso
    &state=af0ifjsldkj
```

```
iwianRpIjoiM....JleHAiOjE0.olnx_YKAm2J1rb
pOP8wGhi1BDNHJjVqsDuushgpwp0E
```

Non-Base64 encoded example of the request parameter object

```
{
    "alg": "RS256",
    "kid": "GxlIiwianVqsDuushgjE0OTUxOTk"
}
.
{
   "iss":
"https://resourcema.coventrybuildingsocie
ty.co.uk",
   "aud": "s6BhdRkqt3",
   "response_type": "code id_token",
   "client_id": "s6BhdRkqt3",
   "redirect_uri":
"https://api.mytpp.com/cb",
   "scope": "openid fundsconfirmations",
   "state": "af0ifjsldkj",
   "nonce": "n-0S6_WzA2Mj",
   "max_age": 86400,
   "claims":
    {
     "userinfo":
      {
       "openbanking_intent_id": {"value":
"123456", "essential": true}
      },
     "id_token":
      {
       "openbanking_intent_id": {"value":
"123456", "essential": true},
       "acr": {"essential": true,
                "values":
["urn:openbanking:psd2:sca",

"urn:openbanking:psd2:ca"]}}
      }
    }
}
```

2. The PSU is then redirected to the CBPII. The CBPII will now possess the Authorization Code and ID Token from CBS. Note at this point, there is no Access Token. The CBPII will now introspect the ID Token and use it as a detached signature to check:

- The hash of the Authorization Code to prove it hasn't been tampered with during redirect (comparing the hash value against the c_hash attribute in ID Token)
- The hash of the State to prove it hasn't been tampered with during redirect (comparing the state hash value against the s_hash attribute in the ID Token)

Example: ID Token

```
{
   "alg": "RS256",
  "kid": "GxlIiwianVqsDuushgjE0OTUxOTk"
}
.
{
   "iss": "https://
resourcema.coventrybuildingsociety.co.uk
",
   "iat": 1234569795,
   "sub": "urn:alphabank:consentId:
123456",
   "acr": "urn:openbanking:psd2:ca",
   "openbanking_intent_id":
"urn:alphabank:consentId: 123456",
   "aud": "s6BhdRkqt3",
   "nonce": "n-0S6_WzA2Mj",
   "exp": 1311281970,
   "s_hash": "76sa5dd",
   "c_hash": "asd097d"
  }
```

3. Once the state and code validations have been confirmed as successful by use of the ID token, the CBPII will proceed to obtain an Access Token from CBS using the Authorization Code they now possess. The CBPII will present its Authorization Code together with the private_key_jwt. The Access Token is required by the CBPII in order to access PSU Funds Confirmation information. The fundsconfirmations scope should already be associated with the Authorization Code generated in the previous step.

Request: Access Token request using Authorization Code and private_key_jwt

```
POST
https://resourcema.coventrybuildingsociet
y.co.uk/mga/sps/oauth/oauth20/token
HTTP/1.1
Host: https://
resourcema.coventrybuildingsociety.co.uk
client_id: tppclientid
Content-Type: application/x-www-form-
urlencoded
Accept: application/json
grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https://api.mytpp.com/cb
&client_assertion_type=
    urn%3Aietf%3Aparams%3Aoauth%3Aclient-
assertion-type%3Ajwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpc3MiOiJodHRw
czovL2p3dC1pZHAuZXhhbXBsZS5jb20iLCJzdWIiO
iJtYWlsdG86bWlrZUBleGFtcGxlLmN
```

Response: Access Token

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "access_token": "SlAV32hkKG",
 "token_type": "Bearer",
 "expires_in": 7776000
}
```

```
vbSIsIm5iZiI6MTQ5OTE4MzYwMSwiZXhwIjoxNDk5
MTg3MjAxLCJpYXQiOjE0OTkxODM2MD
EsImp0aSI6ImlkMTIzNDU2IiwidHlwIjoiaHR0cHM
6Ly9leGFtcGxlLmNvbS9yZWdpc3Rlc
iJ9.SAxPMaJK_wYl_W2idTQASjiEZ4UoI7-
P2SbmnHKr6LvP8ZJZX6JlnpK_xClJswAni1T
p1UnHJslc08JrexctaeEIBrqwHG18iBcWKjhHK2Tv
5m4nbTsSi1MFQOlMUTRFq3_LQiHqV2
M8Hf1v9q9YaQqxDa4MK0asDUtE_zYMHz8kKDb-jj-
Vh4mVDeM4_FPiffd2C5ckjkrZBNOK0
01Xktm7xTqX6fk56KTrejeA4x6D_1ygJcGfjZCv6K
nki7Jl-6MfwUKb9ZoZ9LiwHf5lLXPuy
_QrOyM0pONWKj9K4Mj7I4GPGvzyVqpaZUgjcOaZY_
rlu_p9tnSlE781dDLuw
```

Non-Base64 encoded example of the request parameter object

```
{
  "alg": "RS256",
  "kid": "12345",
  "typ": "JWT"
}
.
{
  "iss": "s6BhdRkqt3",
  "sub": "s6BhdRkqt3",
  "exp": 1499187201,
  "iat": 1499183601,
  "jti": "id123456",
  "aud": "https://
resourcema.coventrybuildingsociety.co.uk
/as/token.oauth2"
}
```

Step 4 – Confirm Funds

1. The CBPII can use the Access Token to create a Funds Confirmation Resource. The following examples are from the Funds Confirmation API Specification

Where the initial Access Token expires, the CBPII will need to create a new request, CBS have not implemented Refresh Tokens.

Example request against Funds Confirmations resource

Request: POST /Funds Confirmations API

```
POST /funds-confirmations HTTP/1.1
client_id:tppclientid
Content-Type: application/json
Authorization: Bearer
1t1satruthun1v3rs4lly
Accept: application/json; charset=utf-8
```

Response: POST / Funds Confirmations API

```
HTTP/1.1 201 Created
Content-Type: application/json
x-fapi-interaction-id: hook5i13-ntIg-
4th3-rP41-3ro535touch3

{
  "Data": {
```

```
x-fapi-financial-id: I4mth3R3-4p3r-411t-
hing-5withh33dfu1
x-fapi-interaction-id: hook5i13-ntIg-
4th3-rP41-3ro535touch3
X-Client-Id:tppclientid
{
  "Data": {
    "ConsentId": "123456",
    "Reference": "Purchase01",
    "InstructedAmount": {
       "Amount": "20.00",
       "Currency": "GBP"
    }
  }
}
```

```
    "FundsConfirmationId": "789012",
    "ConsentId": "123456",
    "CreationDateTime": "2017-05-
02T00:00:00+00:00",
    "FundsAvailable": true,
    "Reference": "Purchase01",
    "InstructedAmount": {
       "Amount": "20.00",
       "Currency": "GBP"
    }
  },
  "Links": {
    "Self": "https://
resourcema.coventrybuildingsociety.co.uk
/open-banking/v2.0/funds-
confirmations/789012"
  },
  "Meta": {}
}
```

Step 5 – Get Funds Confirmation Consent Status

The CBPII can use the Access Token to retrieve Funds Confirmation Consent Resource. The following examples are from the Funds Confirmation Consents API Specification

Where the initial Access Token expires, the CBPII will need to create a new request, CBS have not implemented Refresh Tokens.

Example request against Confirm Funds Consent resource

Request: GET /Funds Confirmations API

```
GET /funds-confirmation-consents/123456
HTTP/1.1
Authorization: Bearer Jhingapulaav
x-fapi-financial-id: OB/2017/001
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
X-Client-Id:tppclientid
Accept: application/json
```

Response: POST / Funds Confirmations API

```
HTTP/1.1 200 OK
x-fapi-interaction-id: 93bac548-d2de-
4546-b106-880a5018460d
Content-Type: application/json

{
  "Data": {
    "ConsentId": "123456",
    "CreationDateTime": "2017-05-
02T00:00:00+00:00",
    "Status": "AwaitingAuthorisation",
    "StatusUpdateDateTime": "2017-05-
02T00:00:00+00:00",
    "ExpirationDateTime": "2017-05-
02T00:00:00+00:00",
    "DebtorAccount": {
      "SchemeName": "UK.OBIE.IBAN",
      "Identification": "GB76LOYD30949301
273801",
```


```
      "SecondaryIdentification": "Roll
56988"
    }
  },
  "Links": {
    "Self": "https://
resourcema.coventrybuildingsociety.co.uk
/open-banking/v2.0/funds-confirmation-
consents/123456"
  },
  "Meta": {}
}
```

Edge Cases

This section provides further information on potential edge cases that may arise via the implementation of Accounts and Payments API Specifications.

PSU Consent Authorization Interrupt with CBS

API	Scenario	Workflow Step	Impact	
Payments	Due to an interruption, the PSU does not complete the Authorization of the Payment with CBS when redirected by the PISP (after creating a PaymentId)	Step 3: Authorize Consent	Payment Status remains as Pending or AcceptedTechnicalValidation	The PISP may choose to implement a separate follow up process which reminds the PSU to complete their Authorization consent steps with CBS.
Accounts	Due to an interruption, the PSU does not complete the Authorization of the Accounts request with CBS when redirected by the AISP (after creating an AccountRequestId)	Step 3: Authorize Consent	Account Status remains as AwaitingAuthorisation	The AISP may choose to implement a separate follow up process which reminds the PSU to complete their Authorization consent steps with CBS