INSTRUCTION MANUAL FOR ESTABLISHING CONNECTION TO CREDIT AGRICOLE'S XS2A PRODUCTION ENVIRONMENT

Version 1.4

CRÉDIT AGRICOLE

Table of Contents

INSTRUCTION MANUAL FOR ESTABLISHING CONNECTION TO CREDIT AGRICOLE'S XS2A PRODUCTION ENVIRONMENT 1
Prerequisites 3
JWS signature generation 4
AIS service call example 4
Authorization 4
Token generation 6
Service execution 7
Changes and additional details vs PolishAPI standard 9
Acquiring permissions to retrieve payment status 9
Specifying the consent scope 10
Retrieving account information with account selection on the ASPSP side 12
Renewal of SCA for AIS consent 12
Request Limits 13
Restrictions on consent validity 13

Prerequisites

Steps to complete before starting work with the production environment:

- Secure qualified certificates necessary for communication with XS2A API endpoints.
- Read the technical specification of services published by the API Portal at: https://www.credit-agricole.pl/apiportal.

Things to keep in mind while working with the production environment:

- Connections to the endpoints are established using two-way (mutual) TLS authentication with QWAC certificates.
- Each and every request and response must be signed with QSEAL certificates. As per the PolishAPI standard (https://polishapi.org/#docs), when you call a service, the request must include a JWS-SIGNATURE header with the JWS signature of the request content. The process of generating a JWS signature is detailed later on.
- The `"requestId"` field must be unique in each call.
- The value of the `"tppId"` field in a call must be consistent with the value of field "Organization Identifier" (organisationIdentifier - 2.5.4.97) in the test certificates used.
- The content of the `"client_id"` field in requests must match the value of the `"tppId"` field.
- The solution does not support a dedicated onboarding service. The first, correct call to the /authorize service using valid certificates is equivalent to registering the TPP application.

JWS signature generation

In order to ensure integrity and immutability of transmitted messages, each message must have a `X-JWS-SIGNATURE` header containing the JWS signature of the request. The JWS signature must be generated in line with RFC 7515. What is more, the JWS signature should be prepared without the payload (i.e. it should be detached) and it should be generated based on the unencoded payload (Unencoded Payload Option - RFC 7797).

The JWS signature header should include the following parameters

- `"alg"` – algorithm used for signing – this field should equal `"RS256"`
- `"x5c"` – certificate or certificate chain corresponding to the key used to generate the signature
- `"x5u"` – URL address of the certificate corresponding to the key used to generate the signature
- `"x5t#S256"` – base64url-encoded thumbprint of the certificate corresponding to the key used to generate the signature
- `"b64"` – information whether the signature was generated based on an encoded payload – it should equal `false`
- `"kid"` – identifier of the key used to generate the signature

Please note that either the `"x5c"` or `"x5u"` parameter (but not both) can be used to indicate the certificate used.

AIS service call example

Below is an example of calling the AIS getAccount service in the production environment available at: https://xs2a.credit-agricole.pl/CaPolishAPI/prod/individual. This process consists of three steps, which are further detailed below.

Authorization

The first step is to retrieve an authorization code. To do this, you need to call the `/authorize` service with the following payload:

```
{
    "requestHeader": {
        "requestId": "8a740673-c751-4558-86e7-9fab31d91c4e",
        "tppId": " YYYYY-ZZZZ-TPPIdentificator",
        "userAgent": "SOAP-UI accounts.0-ais-getAccount",
        "isCompanyContext": false,
        "ipAddress": "127.0.0.1",
        "sendDate": "2019-09-06T09:36:47.536Z"
    },
    "response_type": "code",
    "client_id": " YYYYY-ZZZZ-TPPIdentificator ",
    "redirect_uri": "http://example.com/",
    "state": "252a5f94-dc2a-4260-9070-af91e3eb3cde",
    "scope": "ais",
    "scope_details": {
        "scopeGroupType": "ais",
        "consentId": "ffd4954c-e2c5-488d-b7e9-eeffad0c64ac",
        "scopeTimeLimit": "2019-10-06T11:28:50.000+02:00",
        "throttlingPolicy": "psd2Regulatory",
```

```
        "privilegeList": [
            {
                "accountNumber": "PL78194000086704648427357299",
                "ais:getAccount": {
                    "scopeUsageLimit": "single"
                }
            }
        ]
    }
}
```

The payload listed above must be signed with a QSEAL certificate and placed in the X-JWS-SIGNATURE header of the request for a REST service. A request must contain the following headers to be processed successfully:

```
Accept-Language: pl
X-REQUEST-ID: {value consistent with "requestId" field in the payload}
Accept: application/json
Accept-Charset: utf-8
Accept-Encoding: deflate
X-JWS-SIGNATURE: {JWS signature of the request}
```

In this example, the call made to a REST service is the following:

```
POST https://xs2a.credit-
agricole.pl/CaPolishAPI/prod/individual/v3_0.1/auth/v3_0.1/authorize HTTP/1.1
Accept-Encoding: gzip,deflate
Accept-Language: pl
X-REQUEST-ID: 8a740673-c751-4558-86e7-9fab31d91c4e
Accept: application/json
Accept-Charset: utf-8
Accept-Encoding: deflate
X-JWS-SIGNATURE: eyJ4NWMiOlsiTUlJRnl6Q0NCTE9nQXdJQkFnSUpBSUU1eGo5RC9CdzdN(...)
Content-Type: application/json;charset=UTF-8
Content-Length: 874
Host: xs2a.credit-agricole.pl
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

The expected response is an URL address to CABP's online banking website, where the user can authorize consent for the execution of the service.

Example response with redirection address:

```
{
    "responseHeader": {
        "requestId": "8a740673-c751-4558-86e7-9fab31d91c4e",
```

```
        "sendDate": "2019-09-06T09:36:48.834+02:00",
        "isCallback": false
    },
    "aspspRedirectUri": "https://ca24.credit-
agricole.pl/login?key=bQpG2SInmCEjQjYs&hash=%2B8s4CtTcrEiicV5N5aOxCHkpQ3k%3D"
}
```

When you go to the address defined in the `"aspspRedirectUri"` field you will arrive at CABP's online banking website. After authorization of consent in the online banking website, you will be redirected to the address specified in the `"redirect_uri"` field of the request. In the above example this will be `"Http://example.com/"`. This address will be appended with the `"code"` parameter containing the authorization code required in the next step and the `"state"` parameter with a value provided in the authorize request. In this example the full redirection address is the following:

`http://example.com?code=zJAjnsTwTPkTAekm&state=252a5f94-dc2a-4260-9070-af91e3eb3cde`

Token generation

In the next step, you need to generate an access token. To do this, you need to call the `/token` service with a payload containing the obtained authorization code in the `"code"` field. In this example, the payload is the following:

```
{
    "requestHeader": {
        "requestId": "7aa46703-1cac-457e-b616-a50f9f514713",
        "tppId": "YYYYY-ZZZZ-TPPIdentificator",
        "userAgent": "SOAP-UI accounts.0-ais-getAccount",
        "ipAddress": "127.0.0.1",
        "isCompanyContext": true,
        "sendDate": "2019-09-06T09:37:42.632Z"
    },
    "grant_type": "authorization_code",
    "code": "zJAjnsTwTPkTAekm",
    "client_id": "YYYYY-ZZZZ-TPPIdentificator",
    "redirect_uri": "http://example.com/"
}
```

A request must contain the following headers to be processed successfully:

```
Accept-Language: pl
X-REQUEST-ID: {value consistent with "requestId" field in the payload}
Accept-Charset: utf-8
Accept-Encoding: deflate
X-JWS-SIGNATURE: {JWS signature of the request}
```

The expected response is an access token for the execution of the service. In this example, the response is the following:

```
{
```

```
    "responseHeader": {
        "requestId": "7aa46703-1cac-457e-b616-a50f9f514713",
        "sendDate": "2019-09-06T09:37:46.313+02:00",
        "isCallback": false
    },
    "access_token": "O3/D6+rfvDOTnUjMQx2EU6AQxGk=",
    "token_type": "bearer",
    "expires_in": "120",
    "refresh_token": "4oL6qdF86tHj48k2",
    "scope": "psd2-ais",
    "scope_details": {
        "privilegeList": [
            {
                "accountNumber": "PL78194000086704648427357299",
                "ais:getAccount": {
                    "scopeUsageLimit": "single"
                }
            }
        ],
        "consentId": "ffd4954c-e2c5-488d-b7e9-eeffad0c64ac",
        "scopeTimeLimit": "2019-10-06T11:28:50.000+02:00",
        "throttlingPolicy": "psd2Regulatory"
    }
}
```


The response includes the token in the `"access_token"` field. If the token expires, a new one must be generated using the token passed in the `"refresh_token"` field, provided that it has been issued.

Service execution

The last step is the execution of the business service. In this example, this is the retrieval of the customer's account information. To do this, you need to call the `/getAccount` service with a payload containing in the `"token"` field a value that in the previous step has been received in the `"access_token"` field. In this example, the payload is the following:


```
{
    "requestHeader": {
        "requestId": "fa884132-d089-11e9-bb65-2a2ae2dbcce4",
        "userAgent": "SOAP-UI accounts.0-ais-getAccount",
        "ipAddress": "127.0.0.1",
        "sendDate": "2019-09-06T09:37:50.583+02:00",
        "tppId": "YYYYY-ZZZZ-TPPIdentificator",
        "token": "O3/D6+rfvDOTnUjMQx2EU6AQxGk=",
        "isDirectPsu": true
    },
    "accountNumber": "PL78194000086704648427357299"
}
```


A request must contain the following headers to be processed successfully:

```
Accept-Language: pl
Authorization: {value consistent with the "token" field in the payload}
X-REQUEST-ID: {value consistent with the "requestId" field in the payload}
Accept-Charset: utf-8
Accept-Encoding: deflate
X-JWS-SIGNATURE: {JWS signature of the request}
```

The expected response is detailed account information. In this example, the response is the following:


```
{
    "responseHeader": {
        "requestId": "fa884132-d089-11e9-bb65-2a2ae2dbcce4",
        "sendDate": "2019-09-06T09:37:56.148+02:00",
        "isCallback": false
    },
    "account": {
        "accountNumber": "PL78194000086704648427357299",
        "nameAddress": {
            "value": [
                "ul. Niska 1, 50-000 Wrocław"
            ]
        },
        "accountType": {
            "code": "5555",
            "description": "description"
        },
        "accountTypeName": "Account Type Name",
        "accountHolderType": "individual",
        "accountNameClient": "Client Name",
        "currency": "PLN",
        "availableBalance": "99999",
        "bookingBalance": "99999",
        "bank": {
            "bicOrSwift": "AGRIPLPR",
            "name": "Credit Agricole Bank Polska SA",
            "address": [
                "Credit Agricole Bank Polska SA",
                "Legnicka 48 bud. C-D",
                "54-202 Wrocław"
            ]
        },
        "auxData": {
            "additionalProp1": "",
            "additionalProp2": "",
            "additionalProp3": ""
        }
    }
}
```

In this example, the above-listed response proves that the AIS getAccount service has been called correctly. Any calls to all the other AIS, PIS and CAF services should be executed in a similar fashion The specification of calls for individual services can be found in the technical documentation published at the API Portal https://apiportal.credit-agricole.pl.

Changes and additional details vs PolishAPI standard

In some aspects, the PolishAPI standard is fairly imprecise and is subject to inconsistent interpretation. Below are specifications regarding selected areas in the context of how they were implemented by the Bank. Scope of consent

In the call to the `/authorize` service, consents to retrieve the payment status are not supported, i.e.

- pis:getPayment
- pis:getBundle
- pis:getRecurringPayment

In addition, the following rules were adopted for potential permission combinations:

- in the service, pis group permissions can only be provided individually (they cannot be merged with other permissions)
- permissions from the ais group can be provided under any combination
- ais-accounts permissions must be provided individually (they cannot be merged with other permissions)

Acquiring permissions to retrieve payment status

In accordance with the Polish API standard, acquiring permissions to retrieve the payment status, the status of a payment bundle or recurring payment is possible by refreshing the token (refreshToken) issued for the consent for payment, payment bundle or a recurring payment. The token is refreshed using the token service in refreshToken mode after instructing a payment, payment bundle or recurring payment (after using the permission from the original consent), but before the lapse of the expiry date of the initial consent. The following transitions are possible:

Initial permission	Acquired permission
pis:domestic	pis:getPayment
pis:EEA	pis:getPayment
pis:nonEEA	pis:getPayment
pis:tax	pis:getPayment
pis:bundle	pis:getBundle
pis:recurring	pis:getRecurringPayment

Below is an example of a call to the `/token` service acquiring permissions to execute the `/getPayment` service based on a previously obtained consent to execute the `/domestic` service

```
{
    "requestHeader": {
        "requestId": "2bc25652-e8d6-11e9-81b4-2a2ae2dbcce4",
        "tppId": "YYYYY-ZZZZ-TPPIdentificator",
```

```
        "userAgent": "SOAP-UI refreshToken",
        "isCompanyContext": false,
        "ipAddress": "127.0.0.1",
        "sendDate": "2019-10-04T19:38:22.139Z"
    },
    "grant_type": "refresh_token",
    "refresh_token": "wkqLFectV5WsXPmd",
    "scope_details": {
        "scopeGroupType": "pis",
        "throttlingPolicy": "psd2Regulatory",
        "consentId": "72839b4e-e8ec-11e9-81b4-2a2ae2dbcce4",
        "scopeTimeLimit": "2019-10-16T11:28:50.000+02:00",
        "privilegeList": [
            {
                "pis:getPayment": {
                    "scopeUsageLimit": "multiple",
                    "paymentId": "3243564",
                    "tppTransactionId": "85638563"
                }
            }
        ]
    }
}
```

In the above call, you must remember that

- the value of the `"refresh_token"` field must match the value returned in the `"refresh_token"` field after calling the `/token` service acquiring the token to execute the initial service, i.e. in this case `/domestic`
- the value of the `"consentId"` field must be equal to the value provided when the `/authorize` service was called for the initial consent, i.e. in this case `/domestic`
- the value of the `"paymentId"`, `"bundleId"` or `"recurringPaymentId"` field must be equal to the value returned after calling the initial service, i.e. in this case `/domestic`
- the value of the `"tppTransactionId"`, `"tppBundleId"` or `"tppRecurringPaymentId"`" field must be equal to the value provided after calling the service from the initial consent, i.e. in this case `/domestic`. This field is optional.
- in the `"scopeTimeLimit"` field you can specify a new consent expiry date. This date is not validated against the expiry date from the initial consent (you can specify a wider time horizon for the new permission)
- in the `"scopeUsageLimit"` field you can specify a different value than in the initial consent (you can obtain multiple permissions to retrieve the status of a payment, payment bundle or a recurring payment)

Specifying the consent scope

In `exchangeToken` mode, the `/token` service allows you to exchange a token with a ais-accounts:getAccounts consent for a new token with a new, detailed scope of the AIS consent for selected accounts. In this mode, you can obtain the following permissions:

- ais:getAccount
- ais:getHolds
- ais:getTransactionsDone
- ais:getTransactionsPending
- ais:getTransactionsRejected
- ais:getTransactionsCancelled
- ais:getTransactionsScheduled
- ais:getTransactionDetail

Below is an example of a call acquiring the getAccount permission based on the getAccounts consent:

```
{
    "requestHeader": {
        "requestId": "8b62b3d0-54cd-404d-8daa-918013599cc9",
        "tppId": "YYYYY-ZZZZ-TPPIdentificator",
        "userAgent": "SOAP-UI exchangeToken",
        "ipAddress": "127.0.0.1",
        "isCompanyContext": false,
        "sendDate": "2019-10-16T19:38:22.139Z"
    },
    "grant_type": "exchange_token",
    "exchange_token": "ImOUbP6_AOSuVjFMz3GD177SQjs=",
    "scope": "ais",
    "scope_details": {
        "scopeGroupType": "ais",
        "throttlingPolicy": "psd2Regulatory",
        "consentId": "e798d37c-c92c-41bf-806e-e0a61fb4811a",
        "scopeTimeLimit": "2019-11-16T11:28:50.000+02:00",
        "privilegeList": [
            {
                "accountNumber": "PL68146000095180629309555036",
                "ais:getAccount": {
                    "scopeUsageLimit": "multiple"
                }
            }
        ]
    }
}
```

In the above call, you must remember that

- the value of the `"exchange_token"` field must be equal to the value returned in the `"access_token"` field after calling the `/token` service acquiring a token to execute the `/getAccounts` initial service
- account numbers indicated in the call must be included in the list received from the call to the `/getAccounts` service
- the consent expiry date provided in the `"scopeTimeLimit"` field may not exceed the expiry date of the initial consent

- the value of the `"consentId"` field should contain a new consent identifier
- in the `"scopeUsageLimit"` field you can specify a different value than in the initial consent

To modify the scope of the consent acquired in this manner, e.g. remove one of the acquired permissions or remove one of the accounts from the consent, the `/token` service should be called again in `exchangeToken` mode, in accordance with the above guidelines, with the indication of the new scope of the AIS consent. The AIS consent obtained previously will be automatically invalidated.

Please note that removing the initial ais-accounts:getAccounts consent using the `/deleteConsent` service will also remove all consents with the detailed scope generated on its basis.

Retrieving account information with account selection on the ASPSP side

In the scenario, when the selection of accounts for which permissions will be granted is made on the bank's side, information on selected accounts is provided in the response to the call to the `/token` service in the `"accountNumber"` field, which is an element of the `"privilegeList"` structure. Below is an example of a response:


```
{
    "responseHeader": {
        "requestId": "7aa46703-1cac-457e-b616-a50f9f514713",
        "sendDate": "2019-09-06T09:37:46.313+02:00",
        "isCallback": false
    },
    "access_token": "O3/D6+rfvDOTnUjMQx2EU6AQxGk=",
    "token_type": "bearer",
    "expires_in": "120",
    "refresh_token": "4oL6qdF86tHj48k2",
    "scope": "psd2-ais",
    "scope_details": {
        "privilegeList": [
            {
                "accountNumber": "PL78194000086704648427357299",
                "ais:getAccount": {
                    "scopeUsageLimit": "single"
                }
            }
        ],
        "consentId": "ffd4954c-e2c5-488d-b7e9-eeffad0c64ac",
        "scopeTimeLimit": "2019-10-06T11:28:50.000+02:00",
        "throttlingPolicy": "psd2Regulatory"
    }
}
```


The account number obtained in this fashion should be later provided when calling the target service for which the consent was given.

Renewal of SCA for AIS consent

The option of renewing the SCA for an AIS consent was implemented using the `/authorize` service. To renew a consent, you must call the `/authorize` service by indicating the identifier of the

previously given consent without providing the list of permissions in the `"consentId"` field. On this basis, the system detects that a consent is being renewed. Below is an example of a call made in this mode:

```
{
    "requestHeader": {
        "requestId": "86340673-b751-4558-8357-9fab21d91c8e",
        "tppId": "YYYYY-ZZZZ-TPPIdentificator",
        "userAgent": "SOAP-UI accounts.0-ais-getAccount-SCA",
        "isCompanyContext": false,
        "ipAddress": "127.0.0.1",
        "sendDate": "2019-10-04T11:16:56.536Z"
    },
    "response_type": "code",
    "client_id": "YYYYY-ZZZZ-TPPIdentificator",
    "redirect_uri": "http://example.com/",
    "state": "267a5f94-d32a-1860-9070-af37e3eb27de",
    "scope": "ais",
    "scope_details": {
        "scopeGroupType": "ais",
        "consentId": "0b5159d2-030f-4ad5-8591-0b489b935ade",
        "scopeTimeLimit": "2019-10-14T11:16:56.536Z",
        "throttlingPolicy": "psd2Regulatory"
    }
}
```

In the above call, you must remember that:

- the value of the `"scopeTimeLimit"` field should not exceed the date provided when giving the consent (in the case of a shorter expiry date, the consent is updated)

Moreover, the renewal of the SCA for an AIS consent, which has been specified by means of the `/token` service in `exchangeToken` mode does not automatically trigger the renewal of the subordinate consent.

Request Limits

The PolishAPI standard imposes a limit on calls to AIS services to a maximum of 4 calls without user interaction within 24 hours. Queries are counted at the level of a single permission in the consent.

For example, if the consent contains the ais:getTransactionsDone and ais:getAccount permission, it will allow for 4 history queries within 24 hours and 4 account details queries within 24 hours.

Therefore, query limits are not verified when the token is issued, but only when the business service is called. Thus, in addition to the setting of the `"is_user_session"` header in the call to `/token` service, the content of `"isDirectPsu"` header in the respective business methods calls is key here.

Restrictions on consent validity

The maximum validity of PIS consents for a payment, payment bundle or recurring payment instruction was capped at 15 minutes.