Trade Republic

PSD2 OPEN BANKING

TPP Integration Guide

Technical documentation for Third-Party Providers integrating with the Trade Republic PSD2 API.

Version 2.1 · July 2026

open-banking@traderepublic.com

Welcome

The Trade Republic PSD2 Open Banking API lets a registered Third-Party Provider (TPP) act on behalf of a Trade Republic customer — within the strict boundaries the customer has approved. Once your TPP is onboarded, you can:

- **Initiate SEPA payments** from a customer's Trade Republic cash account (PIS — Payment Initiation Service).
- **Read account information** — the list of accounts the customer holds with us, current and available balances, and booked transactions (AIS — Account Information Service).

Both capabilities are gated by the same fundamental principle: **every action that touches customer data or moves customer money requires a valid bearer token**, and that token can only be obtained by proving (a) who you are, using your eIDAS QSeal certificate, and (b) where the customer authorised you to operate, using PSD2-compliant Strong Customer Authentication (SCA).

This guide walks you, end-to-end, from your first onboarding email to a working integration in production. Each step explains *what* you do, *why* it is required, *what you get back*, and *what to do next*.

> **Audience**: engineers integrating a licensed PISP or AISP with Trade Republic. We assume familiarity with HTTPS, JSON, JWTs, and the broad shape of PSD2. No prior Trade Republic-specific knowledge is required.

How to read this guide

The sections are ordered the way an integration is built. Read them in order the first time; afterwards, treat the table of contents below as a reference.

Table of Contents

Before You Begin

Onboarding

Onboarding itself is **self-service**. There is no pre-flight email, manual review, or approval step at the application layer: Trade Republic registers your TPP automatically on the **first signed token request** that presents a valid eIDAS QSeal certificate. Subsequent requests from the same TPP use the registration created on that first call.

Everything we need to identify and authorise your TPP is taken directly from the certificate:

- **TPP identifier** — read from the `organizationIdentifier` field (OID `2.5.4.97`), e.g. `PSDDE-BAFIN-100003`. This becomes the durable key for your TPP.
- **Roles** — read from the PSD2 `qcStatements` extension (ETSI TS 119 495). Only the roles your QTSP has attested in the certificate are granted. Requesting a token for a `scope` that maps to a role not attested by the certificate is rejected with 401.
- **Legal entity name** and **country of registration** — read from the subject DN.
- **Certificate fingerprint** — stored on first call and bound to your TPP identifier. Every subsequent request must present a certificate with the same fingerprint; a mismatch is rejected with 401.

Once registered, your TPP can immediately call any endpoint allowed by the roles attested in its certificate. **Sandbox and production remain independent registrations** — the first call you make in each environment performs that environment's registration, and the two sets of state (TPP

records, consents, payments) are not shared.

> **TPP status**. Trade Republic retains the right to deactivate a TPP administratively — for example after a confirmed incident or upon licence revocation. Token requests from a deactivated TPP are rejected with 401. Reactivation requires a ticket to **open-banking@traderepublic.com**.

Environments

Environment	Base URL
Sandbox	`https://sandbox.dev.open-banking.traderepublic.com`
Production	`https://open-banking.traderepublic.com`
Production (fallback)	`https://fallback.open-banking.traderepublic.com`

Sandbox is a fully isolated environment intended for integration testing. Real customer data is never exposed there. To make iteration fast, the SCA flow is simplified: no real second factor is required, and consents and payments are auto-approved within up to 120 seconds of the customer being redirected. Use sandbox to validate request signatures, error handling, and the complete lifecycle of every flow before you go live.

Production serves real customers and real money. All security, regulatory, and rate-limit controls are enforced strictly. The fallback URL is an active-passive standby: it shares the same TPP registry, certificates, and consents as the primary, but should only be used when the primary endpoint is unreachable.

> **Failover guidance**. Direct all traffic to the primary production URL by default. Switch to the fallback only when the primary is genuinely unavailable, and switch back as soon as the primary recovers. Do not load-balance across the two — you may observe a consistency lag of a few seconds during a failover event.

What you need

Before writing any code, gather the following:

Item	Description
QSeal certificate	Your eIDAS Qualified Seal Certificate. Sent on every API request via the `TPP-Signature-Certificate` header — used to sign the request and to identify your TPP at the protocol layer.
QSeal private key	The private key paired with your QSeal certificate. **This never leaves your systems.**
TPP-Redirect-URI	An HTTPS endpoint on your server where customers are returned after completing SCA. Must be a fully qualified URL using TLS 1.2 or higher.

Certificate requirements

Your QSeal certificate is the cryptographic anchor of every request. To be accepted it must:

- Be a **valid, non-expired X.509 certificate** issued by a Qualified Trust Service Provider (QTSP) listed in the EU Trusted List.
- Contain your TPP identifier in the organizationIdentifier field (OID `2.5.4.97`), formatted as `PSD<CC>-<NCA>-<ID>` — for example, `PSDDE-BAFIN-100003`.
- **Not revoked**. Every API request triggers a live **OCSP check** against the certificate's issuer. A revoked certificate is rejected immediately with HTTP 401, regardless of whether the signature is otherwise correct.

> **Operational tip**. Certificate rotation requires a coordinated handover with our team. Open a rotation ticket via **open-banking@traderepublic.com** at least **30 days** before the existing certificate expires.

Authentication

Why signed requests

The token endpoint is the only door through which a TPP can obtain access. To make sure the request truly originates from a registered TPP — and that no proxy, log, or attacker has modified the body in transit — every call must carry an **HTTP Signature** built from your QSeal certificate. Three headers, taken together, provide that proof.

Header	Description
`TPP-Signature-Certificate`	Your QSeal certificate, Base64-encoded (full PEM including the BEGIN/END lines). Identifies the signer.
`Digest`	The SHA-256 hash of the raw request body bytes, Base64-encoded. Binds the signature to the exact body sent.
`Signature`	RSA-SHA256 HTTP Signature over the `Digest` header value, signed with your QSeal private key. Proves possession of the private key.

The server verifies all three: the certificate must match a registered TPP and pass an OCSP check; the digest must match the body it received byte-for-byte; the signature must verify against the certificate's public key. **Any mismatch fails the request with 401**.

Building the headers — step by step

Step 1 — Serialise the request body

Use compact JSON with no extra whitespace. The `Digest` is computed over the **exact bytes sent on the wire**, so any difference — a trailing newline, a re-ordered key, a pretty-print indent introduced by middleware — will invalidate the signature.

```
{"grantType":"CLIENT_CREDENTIALS","scope":"PAYMENT_INITIATION"}
```

> **Common pitfall**. Do not let your HTTP client re-serialise the body after you compute the digest. Compute the bytes, hash them, and send those exact bytes.

Step 2 — Compute the Digest

SHA-256 hash the raw body bytes, then Base64-encode the result. Prefix the value with the algorithm identifier `SHA-256=` so the server knows which hash function to verify.

```
Digest: SHA-256=<base64(sha256(body_bytes))>
```

Step 3 — Build the signing string

The signing string is a single line: the header name `digest` in lowercase, a colon and a single space, then the full `Digest` header value. Do not append a newline.

```
digest: SHA-256=<base64(sha256(body_bytes))>
```

We deliberately sign only the `Digest` header. Because `Digest` is bound to the body, signing it transitively binds the signature to the body — no need to enumerate every header.

Step 4 — Sign

Sign the UTF-8 bytes of the signing string using **RSA-SHA256 with PKCS#1 v1.5 padding**, with your QSeal private key and Base64-encode the resulting signature bytes.

Step 5 — Set the Signature header

Assemble the header from four named parameters. `keyId` is the SHA-256 fingerprint of your certificate as lowercase hexadecimal — the server uses it to look up which registered certificate to verify against. `headers="digest"` declares which headers are covered by the signature; this must be exactly `digest`.

```
Signature: keyId="<SHA-256 fingerprint of
cert>",algorithm="rsa-sha256",headers="digest",signature="<base64-sign
ature>"
```

Step 6 — Set the TPP-Signature-Certificate header

Base64-encode the entire PEM file — including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines and any newlines between them — and place the result in a single header value with no line breaks.

```
TPP-Signature-Certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...
```

> **Verify before you ship**. Replay the exact bytes you send on the wire through your verification routine offline. If your own verifier accepts them, the server's will too.

Generate an Access Token

Every protected endpoint — payment initiation, consent creation, account data retrieval — requires a **bearer access token**. You obtain that token from a single endpoint by presenting a signed request that declares which capability you intend to use.

There are two grant types, used in different parts of the lifecycle:

- `CLIENT_CREDENTIALS` authenticates **your TPP**. The token it produces is bound to your TPP identifier and lets you initiate payments and create consent requests. It does not, on its own, grant access to any specific customer's data.
- `AUTHORIZATION_CODE` is exchanged after a customer has completed SCA and approved a consent. It produces a token bound to **that specific customer** and is the only token that can read the customer's accounts, balances, and transactions.

Endpoint

Request

```
POST /api/v1/external/psd2/token
Content-Type: application/json
TPP-Signature-Certificate: <base64-encoded-pem>
Digest: SHA-256=<base64-sha256-of-body>
Signature:
keyId="<fingerprint>",algorithm="rsa-sha256",headers="digest",signatur
e="<sig>"
```

Every request to this endpoint must be signed exactly as described in Section 2 — Authentication.

Request body

Field	Type	Required	Values
`grantType`	string	yes	`CLIENT_CREDENTIALS`, `AUTHORIZATION_CODE`
`scope`	string	yes	`PAYMENT_INITIATION`, `ACCOUNT_INFORMAT`

Field	Type	Required	Values
			`ION`
`authCode`	UUID	only for `AUTHORIZATION_CODE`	The single-use authorisation code received after the customer completed SCA.

Which combination to use

Use case	grantType	scope
Initiate a payment	`CLIENT_CREDENTIALS`	`PAYMENT_INITIATION`
Create an account consent request	`CLIENT_CREDENTIALS`	`ACCOUNT_INFORMATION`
Read account data after the customer approved consent	`AUTHORIZATION_CODE`	`ACCOUNT_INFORMATION`

> `AUTHORIZATION_CODE` combined with `PAYMENT_INITIATION` is not supported — payment initiation never issues an authorisation code, because the customer's confirmation is bound to a specific payment, not to an ongoing relationship.

Response

For `CLIENT_CREDENTIALS`:

```
{
  "access_token":
"eyJhbGciOiJFUzI1NiIsImtpZCI6InRyLXRc3Qta2V5LTIwMjUifQ...",
  "token_type": "Bearer",
  "expires_in": 900
}
```

For `AUTHORIZATION_CODE`, a `refresh_token` is also included:

```
{
  "access_token": "eyJhbGciOiJFUzI1NiIs...",
  "refresh_token": "8xLOxBtZp8...Zjfr8mRg",
  "token_type": "Bearer",
  "expires_in": 900
}
```

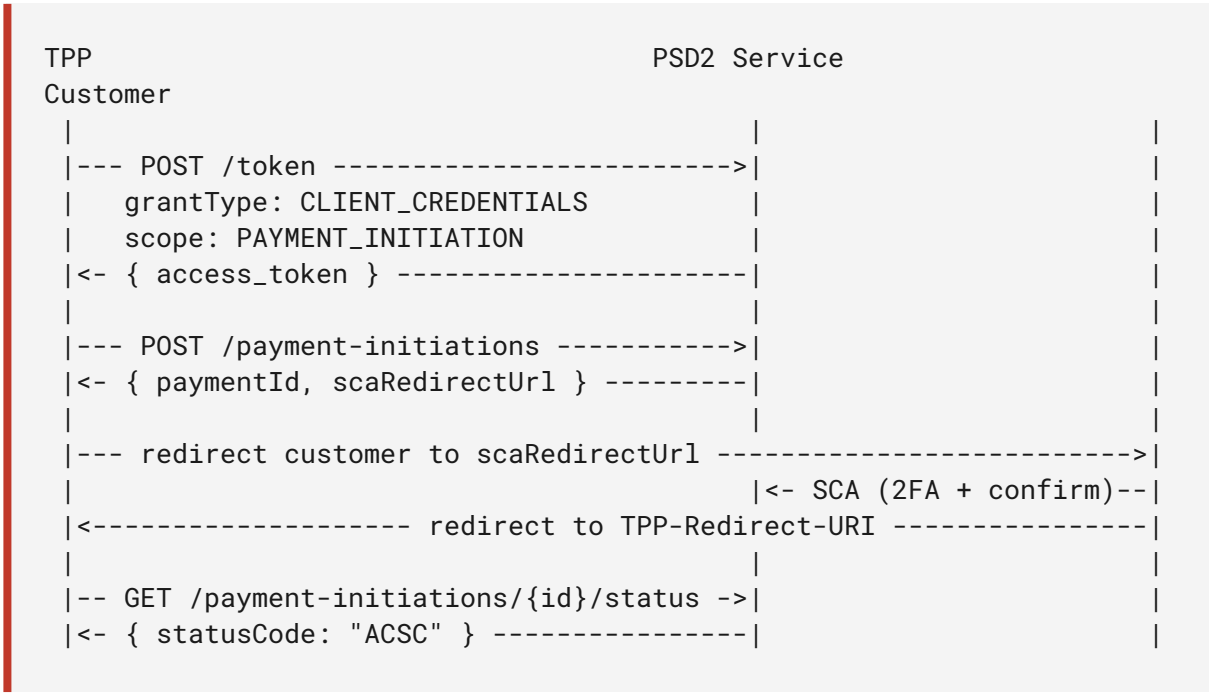
Field	Description
`access_token`	The bearer token. Send it as `Authorization: Bearer <token>` on every subsequent API call.
`token_type`	Always `Bearer`.
`expires_in`	Seconds until the access token expires — **900 seconds (15 minutes)**.
`refresh_token`	Only present for `AUTHORIZATION_CODE` grants. Used to mint new customer-scoped access tokens for the lifetime of the consent (up to 90 days). See Section 7 — Token Refresh.

Token hygiene. Treat every access token as a high-value secret. Store it in memory only, never in logs or URLs, and discard it the moment it expires. Re-mint a fresh token shortly before each batch of work rather than holding one for the entire 15-minute window.

Payment Initiation Flow

This flow allows a TPP to initiate a SEPA Credit Transfer from a customer's Trade Republic cash account. Every payment must be approved or rejected by the customer from their Trade Republic mobile application — PSD2 does not allow recurring or batch payments to be initiated by a TPP without per-payment consent.

Overview



Step 1 — Get a payment token

Purpose. Mint a TPP-scoped access token that authorises you to call the payment-initiation endpoint. The token does not yet identify any particular customer — that binding happens later, during SCA.

Request

```
POST /api/v1/external/psd2/token
Content-Type: application/json
TPP-Signature-Certificate: <base64-encoded-pem>
Digest: SHA-256=<base64-sha256-of-body>
Signature:
keyId="<fingerprint>",algorithm="rsa-sha256",headers="digest",signatur
e="<sig>"

{"grantType":"CLIENT_CREDENTIALS","scope":"PAYMENT_INITIATION"}
```

Response

```
{
  "access_token": "eyJhbGciOiJFUzI1NiIs...",
  "token_type": "Bearer",
  "expires_in": 900
}
```

Next: Carry this token as `Authorization: Bearer <access_token>` on Step 2. Without it, Step 2 fails with 401.

Step 2 — Initiate the payment

Purpose. Submit the payment instruction to Trade Republic. We validate the IBANs, the amount, and the SEPA character set, then return both a paymentId for tracking and a `scaRedirectUrl` that you will hand the customer.

Request

```
POST /api/v1/external/psd2/payment-initiations
Authorization: Bearer <access_token>
Content-Type: application/json
TPP-Redirect-URI: https://your-tpp.example.com/callback
```

The `TPP-Redirect-URI` header tells us where to send the customer's browser after they complete SCA. It must be HTTPS, must match the exact URI you registered, and must be reachable by the customer's browser.

Request body

```
{
  "instructedAmount": {
    "currency": "EUR",
    "amount": "50.00"
  },
  "creditorAccount": {
    "iban": "DE75512108001245126199"
  },
  "creditorName": "Merchant GmbH",
  "debtorAccount": {
    "iban": "DE89370400440532013000"
  },
  "remittanceInformationUnstructured": "Invoice #12345"
}
```

Field	Required	Description
`instructedAmount.currency`	yes	ISO 4217 currency code — only `EUR` is accepted.
`instructedAmount.amount`	yes	Decimal string, e.g. `"50.00"`. Must be greater than `0` and at most `100000.00`, with no more than two decimal places.
`creditorAccount.iban`	yes	The beneficiary's IBAN.
`creditorName`	yes	Beneficiary name, max 70 characters, SEPA character set (`A-Z a-z 0-9 - ? : ( ) . , ' + /`).
`debtorAccount.iban`	no	The customer's source IBAN. Defaults to their primary account if omitted. Must differ from `creditorAccount.iban`.

Field	Required	Description
`remittanceInformationUnstructured`	no	Free-text payment reference shown to the beneficiary. Max 140 characters, SEPA character set.

Response (HTTP 201 Created)

```
{
  "paymentId": "7a6b0c3d-2e1f-4b8a-9c5d-1e2f3a4b5c6d",
  "scaRedirectUrl":
"https://<auth-host>/psd2/payment?paymentId=7a6b0c3d..."
}
```

`paymentId` is your durable handle for the payment — store it; you will need it to poll status. `scaRedirectUrl` is one-time-use and must be opened in the customer's own browser.

Next. Redirect the customer's browser to `scaRedirectUrl` in Step 3.

Step 3 — Redirect the customer

Purpose. Hand control to Trade Republic so the customer can review and authorise the payment with their credentials and second factor. This step is mandatory: under PSD2, a TPP may **never** see, store, or transmit the customer's authentication factors.

Redirect the customer's browser to the `scaRedirectUrl` returned in Step 2. The customer will:

1. Authenticate with two factors (SCA — typically password plus device confirmation).
2. Review the payment details — amount, recipient, and reference — exactly as you submitted them.
3. Confirm or cancel the payment.

Once SCA completes, the customer is redirected back to the `TPP-Redirect-URI` you supplied in Step 2. **No query parameters are appended to the payment redirect** — the outcome is discovered by polling the status endpoint in Step 4.

> **Why no query parameters**? Payment confirmation is not an OAuth-style code exchange; the result is settled inside the bank, not at the redirect. Treat the redirect as a signal to start polling, not as a source of truth.

Step 4 — Poll payment status

Purpose. Determine the terminal outcome of the payment. Settlement may be instantaneous in the sandbox environment but takes seconds to a few minutes in production.

Request

```
GET /api/v1/external/psd2/payment-initiations/{paymentId}/status
Authorization: Bearer <access_token>
```

Response

```
{
  "paymentId": "7a6b0c3d-2e1f-4b8a-9c5d-1e2f3a4b5c6d",
  "statusCode": "ACSC",
  "rejectReason": null
}
```

Field	Description
`paymentId`	The payment identifier.
`statusCode`	The current payment status — see table below.
`rejectReason`	Present (and non-null) only when `statusCode` is `RJCT`. A short, human-readable rejection reason.

statusCode	Meaning	Terminal?
`RCVD`	Payment received by the service.	no
`ACCP`	Awaiting customer SCA and authorisation.	no
`ACFC`	Customer authorised — awaiting bank processing.	no
`ACSC`	Settlement completed —	**yes ✓**

statusCode	Meaning	Terminal?
	funds transferred.	
RJCT	Rejected by customer or bank.	**yes ✓**

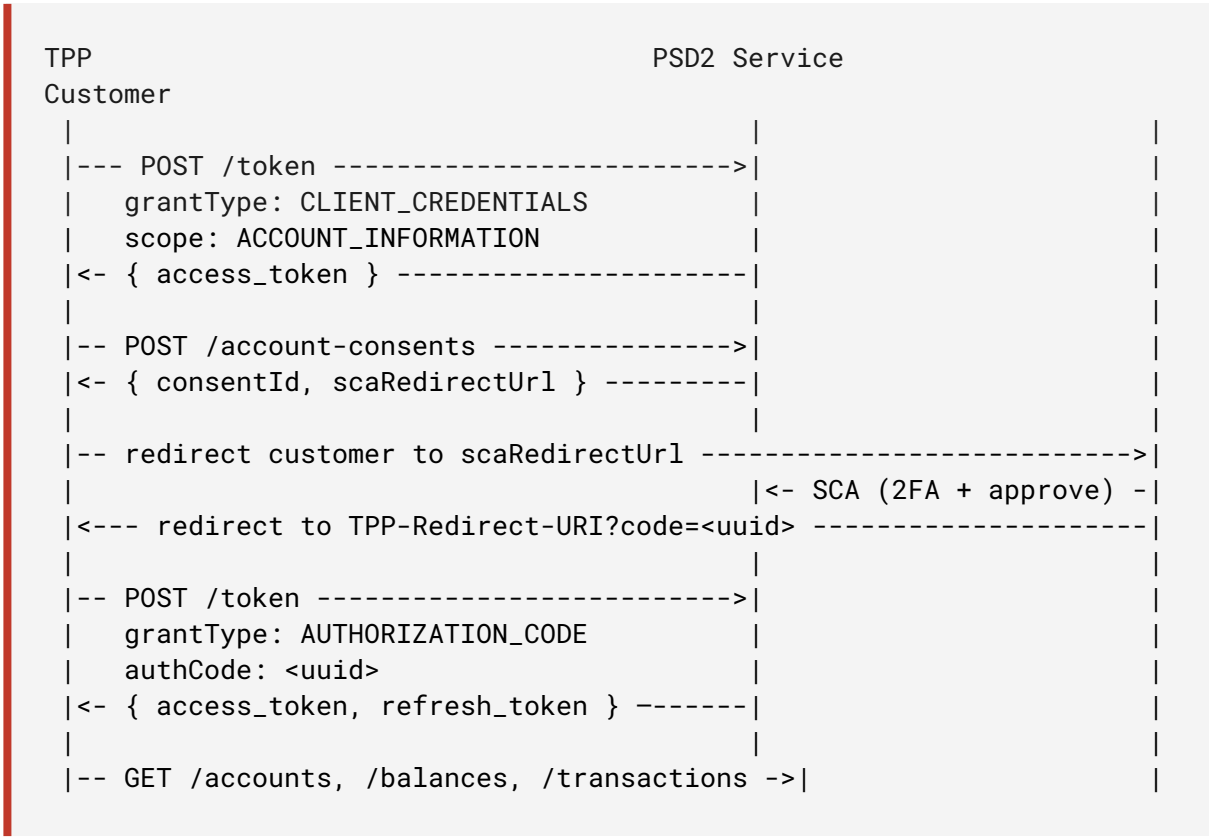
Poll until ACSC or RJCT is returned. **Do not fulfill customer orders, ship goods, or release services until ACSC is observed**. A non-terminal status indicates the payment is still in flight and may yet be rejected.

> **Polling cadence**. Start with a 1-second interval, double on each failed poll up to a 30-second ceiling, and give up after 15 minutes. Combine with the rate-limit guidance in Section 8.

Account Consent Flow

This flow allows a TPP to request, and the customer to grant, durable read access to a customer's account data — the list of accounts, current balances, and transaction history. Unlike payment initiation, where the customer authorises one specific transfer, account consent grants ongoing access (up to 90 days) within explicit limits the customer reviews and approves.

Overview



Step 1 — Get an account information token

Purpose. Mint a TPP-scoped token authorising you to create consent requests. Same shape as the payment-token request, but with `scope=ACCOUNT_INFORMATION`.

Request

```
POST /api/v1/external/psd2/token
Content-Type: application/json
TPP-Signature-Certificate: <base64-encoded-pem>
Digest: SHA-256=<base64-sha256-of-body>
Signature:
keyId="<fingerprint>",algorithm="rsa-sha256",headers="digest",signatur
e="<sig>"

{"grantType":"CLIENT_CREDENTIALS","scope":"ACCOUNT_INFORMATION"}
```

Response

```
{
  "access_token": "eyJhbGciOiJFUzI1NiIs...",
  "token_type": "Bearer",
  "expires_in": 900
}
```

Step 2 — Create the account consent

Purpose. Describe the access you are requesting on the customer's behalf. Be precise: the customer will see this scope verbatim during SCA and the server enforces it on every subsequent read.

Request

```
POST /api/v1/external/psd2/account-consents
Authorization: Bearer <access_token>
Content-Type: application/json
TPP-Redirect-URI: https://your-tpp.example.com/callback
```

There are two consent shapes. Pick whichever matches your actual data needs — the principle of least privilege applies.

Option A — access to all customer accounts

```
{
  "access": {
    "availableAccounts": true
  },
  "isOneTime": false,
  "validityInDays": 90,
  "frequencyPerDay": 4
}
```

Option B — access to specific accounts only

```
{
  "access": {
    "accounts":     [{ "iban": "DE89370400440532013000" }],
    "balances":     [{ "iban": "DE89370400440532013000" }],
    "transactions": [{ "iban": "DE89370400440532013000" }]
  },
  "isOneTime": false,
  "validityInDays": 90,
  "frequencyPerDay": 4
}
```

Field	Required	Description
`access.availableAccounts`	no	`true` grants access to all accounts the customer holds. Mutually exclusive with the IBAN lists below.
`access.accounts`	no	Specific IBANs that may be returned by `/accounts`. Cannot be combined with `availableAccounts: true`.

Field	Required	Description
`access.balances`	no	Specific IBANs whose balances may be read. Cannot be combined with `availableAccounts: true`.
`access.transactions`	no	Specific IBANs whose transactions may be read. Cannot be combined with `availableAccounts: true`.
`isOneTime`	yes	`true` = consent is consumed after one token exchange (single read). `false` = recurring access until expiry. When `true`, `validityInDays` must be `1`.
`validityInDays`	yes	Consent duration in days, minimum `1`. PSD2 caps recurring consents at **90 days**; the TPP is responsible for staying within that cap.
`frequencyPerDay`	yes	Maximum number of account-data API calls permitted per calendar day, minimum `1`. Beyond this, requests fail with 429.

> Either `availableAccounts: true` or at least one IBAN list (`accounts`, `balances`, or `transactions`) must be provided. Omitting both returns 400.

Response

```
{
  "consentId": "8b112cf1-c73e-4a3e-a7c1-82c4f0b4f3e6",
  "scaRedirectUrl":
"https://<auth-host>/psd2/account-information?consentId=8b112cf1..."
}
```

`consentId` is your durable handle for this consent. Persist it so you can later check status and exercise the consent on subsequent sessions.

Step 3 — Redirect the customer

Purpose. Hand control to Trade Republic so the customer can authenticate and approve the exact scope you requested.

Redirect the customer's browser to `scaRedirectUrl`. The customer will:

1. Authenticate with two factors (SCA).
2. Review the precise scope of data access being requested — accounts, balances, transactions, for either all accounts or the IBANs you listed.
3. Approve or deny the consent.

Once the customer acts, they are redirected back to the `TPP-Redirect-URI` with a `code` query parameter appended:

```
https://your-tpp.example.com/callback?code=a1b2c3d4-e5f6-7890-abcd-ef1
234567890
```

> **Time-sensitive**. The `code` is single-use and valid for **10 minutes**. Exchange it for a token (Step 4) as soon as your callback receives it. After 10 minutes, or after one successful exchange, the code is permanently invalid.

Step 4 — Exchange the auth code for a customer token

Purpose. Trade the short-lived authorisation code for a customer-scoped access token (and its long-lived refresh token). This is the moment your TPP transitions from "authorised by the customer" to "able to read their data".

Request

```
POST /api/v1/external/psd2/token
Content-Type: application/json
TPP-Signature-Certificate: <base64-encoded-pem>
Digest: SHA-256=<base64-sha256-of-body>
Signature:
keyId="<fingerprint>",algorithm="rsa-sha256",headers="digest",signatur
e="<sig>"

{
  "grantType": "AUTHORIZATION_CODE",
  "scope": "ACCOUNT_INFORMATION",
  "authCode": "a1b2c3d4-e5f6-7890-abcd-ef1234567890"
}
```

Response

```
{
  "access_token": "eyJhbGciOiJFUzI1NiIs...",
  "refresh_token": "8xLOxBtZp8...Zjfr8mRg",
  "token_type": "Bearer",
  "expires_in": 900
}
```

This `access_token` is **scoped to the specific customer** who approved the consent. Use it for every account-data request in Section 6. The `refresh_token` remains valid for the full consent duration (up to 90 days) and lets you mint new access tokens without sending the customer through SCA again — see Section 7.

Step 5 — Check consent status

You can query a consent's lifecycle state at any time. This is useful for diagnostics, for re-prompting the customer when a consent has been denied, and for proactive renewal as expiry approaches.

Request

```
GET /api/v1/external/psd2/account-consents/{consentId}/status
Authorization: Bearer <access_token>
```

Response

```
{
  "consentId": "8b112cf1-c73e-4a3e-a7c1-82c4f0b4f3e6",
  "statusCode": "ACCP"
}
```

statusCode	Meaning
RCVD	Consent created — awaiting customer action.
PDNG	The customer redirected — SCA in progress.
ACCP	The customer approved — consent is active and usable.
RJCT	The customer denied the consent request.
USED	Authorisation code was consumed — token was already issued.

Account Information Endpoints

These endpoints become available **after** the customer has approved a consent and you have exchanged the authorisation code for a customer-scoped token (Section 5, Steps 3–4). Every request requires the `AUTHORIZATION_CODE` token in the `Authorization` header. Any other token — for example, a `CLIENT_CREDENTIALS` token — fails with 403.

List accounts

```
GET /api/v1/external/psd2/accounts
Authorization: Bearer <access_token>
```

Returns every account covered by the active consent. Use the `accountId` from the response to address the per-account endpoints below.

Get account details

```
GET /api/v1/external/psd2/accounts/{accountId}
Authorization: Bearer <access_token>
```

Returns the IBAN, currency, holder name, and product type for a single account. Useful when you store an `accountId` long-term and need to refresh display attributes.

Get balances

```
GET /api/v1/external/psd2/accounts/{accountId}/balances
Authorization: Bearer <access_token>
```

Returns the current and available balances for the account. Only callable when the consent included balance access for this IBAN.

Get transactions

```
GET /api/v1/external/psd2/accounts/{accountId}/transactions
Authorization: Bearer <access_token>
```

Returns the booked transaction history. The endpoint supports cursor-based pagination via two optional query parameters:

Query parameter	Type	Description
`nextPageStartBookingDate`	date (`YYYY-MM-DD`)	Cursor — return transactions whose booking date is on or after this date.
`nextPageStartTransactionId`	UUID	Cursor — return transactions starting from (exclusive) this transaction ID.

Each parameter is an independent cursor returned by the previous page. Include whichever ones appear in the response to fetch the next page. **When the response omits a cursor parameter, there are no more pages in that dimension.**

Token Refresh

Customer-scoped (`AUTHORIZATION_CODE`) access tokens expire after **15 minutes**. Use the refresh token to obtain a fresh access token without requiring the customer to repeat SCA.
The refresh endpoint is unusual in that it requires **two** tokens, sent on different headers:

1. **A `CLIENT_CREDENTIALS` token with `ACCOUNT_INFORMATION` scope** in the `Authorization` header. This identifies your TPP. Obtain it exactly as in Step 1 of the consent flow. Like any access token, it expires after 15 minutes — mint a fresh one if your cached one has gone stale.
2. **The refresh token** passed as an HTTP cookie named refresh_token. This is the long-lived secret that ties the call to the customer's consent.

Request

```
POST /api/v1/external/psd2/token/refresh
Authorization: Bearer <CLIENT_CREDENTIALS_ACCOUNT_INFORMATION_token>
Cookie: refresh_token=8xLOxBtZp8...Zjfr8mRg
```

Response

```
{
  "access_token": "eyJhbGciOiJFUzI1NiIs..."
}
```

The refresh token is **not rotated** on each use — it stays valid until either the consent expires or the customer revokes it. Treat it as a high-value, long-lived secret: store it encrypted at rest, scope it to a single customer, and discard it the moment a 401 indicates the consent has ended.

> **Refresh proactively**. Mint a new access token at the start of each work session rather than catching `401`s from an expired one. This avoids partial-failure modes where the first request in a batch fails while the rest succeed.

Rate Limiting

Every public TPP endpoint is rate-limited per client. Exceeding the limit returns HTTP `429 Too Many Requests`.

Parameter	Value
Sustained rate	200 requests per second
Burst capacity	1 request

Design your integration to operate **comfortably below** the sustained rate. The limits exist to keep the platform stable for every TPP that shares it; they are not a target.

When you receive a `429`:

1. **Stop sending new requests immediately**.
2. Wait at least 1 second before the first retry.
3. Apply exponential backoff with jitter for any further retries.
4. Never run a tight retry loop — it will only delay your recovery.

Error Reference

HTTP Status	Scenario	Resolution
`400`	Malformed JSON, missing required field, or unsupported `grantType` / `scope` combination.	Review the request body against the field definitions in this guide.
`401`	Missing, expired, or invalid `access_token`.	Refresh the token or generate a new one.
`401`	Invalid HTTP Signature.	Verify that the `Digest` is computed over the exact bytes you send and that the signing key matches the registered certificate.
`401`	Certificate revoked.	The QSeal certificate failed the OCSP check — contact your certificate authority and re-onboard.
`403`	TPP not authorised for the requested scope.	Confirm your TPP registration includes the required role (`PAYMENT_INITIATION` or `ACCOUNT_INFORMATION`).
`404`	`paymentId` or `consentId` not found.	Verify the identifier is correct and was issued to this TPP.
`429`	Rate limit exceeded.	Back off and retry after a short delay. See Section 8.

Glossary

Term	Meaning
AISP	Account Information Service Provider — a TPP licensed to read account data.
ASPSP	Account Servicing Payment Service Provider — the bank, e.g. Trade Republic.
PISP	Payment Initiation Service Provider — a TPP licensed to initiate payments.
QSeal	Qualified electronic Seal — eIDAS-grade certificate that identifies a legal entity.
OCSP	Online Certificate Status Protocol — live revocation check against the issuing CA.
SEPA	Single Euro Payments Area — the regulatory zone for euro-denominated transfers.
`scaRedirectUrl`	One-time URL that delivers the customer to Trade Republic for SCA.
`TPP-Redirect-URI`	Pre-registered HTTPS URL where the customer is returned to your TPP after SCA.