

# Technical standards

---

The technical standards are what pension providers and schemes will use to interface with the central technical architecture and/or each other. This includes the connectivity mechanisms; protocols for authorising the sharing of information; the methodology for the generation of pension identifiers, tokens, globally unique identifiers used in ecosystem transactions; and the rules for registration of pension identifiers for pensions found.

Further technical documentation to accompany the technical standards will be released at a later date.

## Version 2.0

These standards are approved by the Secretary of State for Work and Pensions and the Department for Communities (Northern Ireland) and were published on **13 March 2025**.

Pension providers and schemes must align with this version of the technical standards.

These standards were approved by the Secretary of State for Work and Pensions on 4 March 2025 and by the Department for Communities (Northern Ireland) on 13 March 2025.

How changes to standards will be managed will be outlined in PDP's [approach to standards governance](#).

## Changelog

Refer to the [changelog](#) for updates since the last publication.

## Downloads

Download the APIs and schemas related to the technical standards.

[Download technical-standards-apis-and-schemas-v2-0.zip](#)

This zip folder contains:

- view-data v1\_2.yaml
- find-requests v1\_3.yaml
- introspect v1\_1.yaml
- perm v1\_1.yaml
- rqp v1\_1.json
- rreguri v1\_3.yaml
- token v2\_1.yaml

# Introduction

## Background

1. Pensions dashboards are apps, websites or other tools which help individuals view information about their multiple pensions in one secure place online, at a time of their choosing. They bring together information on all a user's (in-scope) pensions, including their State Pension, as well as any occupational and personal pensions. This supports individuals' engagement with their pensions and their planning for retirement.
2. The Money and Pensions Service (MaPS) set up the Pensions Dashboards Programme (PDP) in 2019 to design and build the central digital architecture (CDA) and services that make pensions dashboards possible. PDP are also responsible for the supporting governance framework, service design and operating model for the pensions dashboards ecosystem.
3. The pensions dashboards ecosystem enables millions of individuals to connect with their pensions information through multiple dashboards across thousands of pension providers and schemes. Find out more about [the pensions dashboards ecosystem and its components](#).
4. MaPS is responsible for operating its own non-commercial, pensions dashboard as a public service.

## Purpose

5. These technical standards are issued by the Money and Pensions Service (MaPS) under delegated powers given by the [Pensions Dashboards Regulations 2022](#) and the [Pensions Dashboards \(No. 2\) Regulations \(Northern Ireland\) 2023](#) (referred to hereafter as 'Regulations') and the [Rules of the Financial Conduct Authority \(FCA\)](#) (hereafter 'Rules').
6. The technical standards provide the basis for interoperability across the pensions dashboards ecosystem. They provide a common set of connectivity mechanisms and interfacing rules for pension providers and schemes, determining how parties are to interact with and communicate with the central digital architecture and each other.
7. The technical standards therefore cover:
  - connectivity mechanisms
  - protocols for authorizing the sharing of information
  - the methodology for the generation of pension identifiers, tokens, globally unique identifiers used in ecosystem transactions
  - the rules for registration of pension identifiers for pensions found
  - definitions of APIs to be used by ecosystem participants
  - details of how the APIs must be used during the expected functioning of the ecosystem
8. The technical standards facilitate pension providers' and schemes' compliance.

## Areas for future inclusion

**9.** There are a number of areas of functionality that will be addressed in later versions of the technical standards, namely:

- standards that apply only to pension dashboard providers
- mechanism to enable refresh of PAT tokens to allow pension providers and schemes to manage resources that they have registered
- PAT refresh notifications issued by the CDA to the pension providers or schemes
- general error response formats and codes

## Audience

**10.** These standards apply legally to the trustees or managers of occupational pension schemes (pension schemes), the managers of stakeholder and personal pension schemes (pension providers), connected to, or required to connect to, the pensions dashboards ecosystem. This version does not include any standards that apply to dashboard providers. The standards for dashboard providers will be published separately.

**11.** Where pension providers and schemes connect to the ecosystem using a third-party supplier, such as a third-party administrators or software providers, the third parties will apply the technical standards on behalf of their client pension providers and schemes. We expect much of the implementation of our standards will be undertaken by these third parties on behalf of multiple clients. A pension provider or scheme connecting via an already-connected third party will use the third party's processes to meet the standards. However, as the standards apply to the pension provider or scheme, the latter are responsible for compliance with them, even if implementation is delegated to a third party. When we refer to pension providers and schemes, this includes any of these third parties.

## Jurisdiction

**12.** These standards apply to all United Kingdom pension providers subject to the dashboard duties in the FCA Rules, and all United Kingdom pension schemes subject to the dashboard duties in the DWP Regulations.

## Other guidance

**13.** These standards should be read in conjunction with the other [PDP standards](#) (data standards, code of connection, reporting standards).

## Use and evidence

**14.** Standards are mandatory requirements and, therefore, compliance by pension providers and schemes is compulsory.

## Version

**15.** This is version 2.0 of the technical standards.

16. Refer to the [changelog](#) for updates since the last publication.

## Technical overview

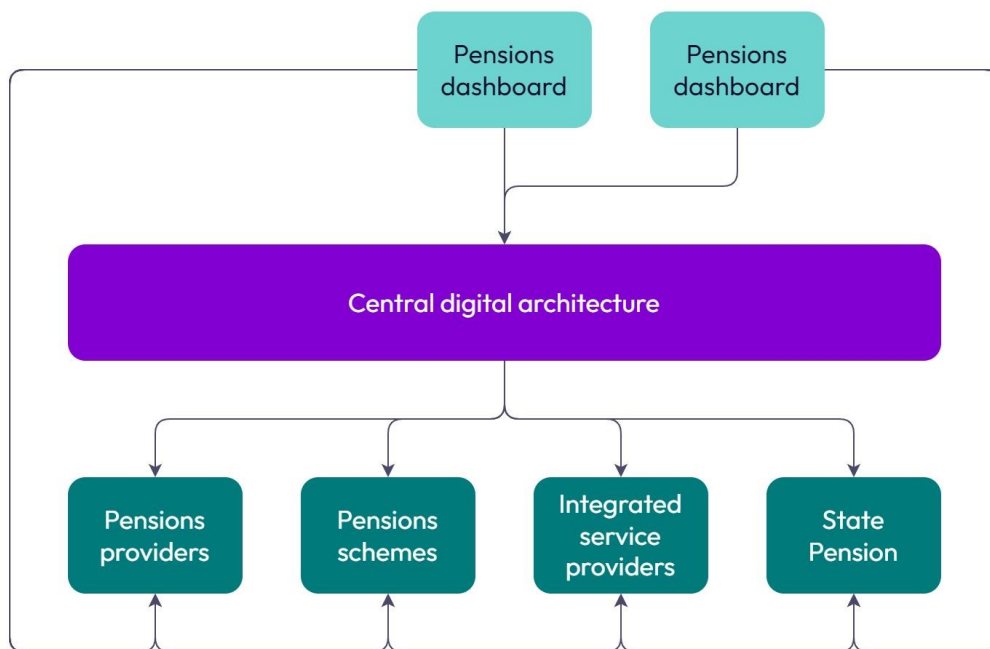
### Client registration

17. Registration of OAuth clients (pension providers and schemes and pension dashboard providers) is required for the operation of the user-managed access (UMA) profiles. The governance register (which includes the UMA authorization server as the software entity managing software client registration) will provide services for client registration.

18. Static client registration is being used.

### Overview of the pensions dashboards ecosystem participants

19. The following diagram shows each participant involved in the protocol interactions detailed in this document.



### Pensions dashboards ecosystem participants

20. The participants in the ecosystem are:

- pension dashboard providers (MoneyHelper and commercial dashboards)
- central digital architecture (this encompasses the consent and authorization service, the pension finder service, the identity service and the governance register)
- pension providers, pension schemes, integrated service providers (ISPs), State Pension service

## OpenAPI and JSON schema specifications

**21.** This version of the technical standards includes, and should be read in conjunction with the OpenAPI specifications contained in the [downloads section](#) and, with the json schemas detailing the find\_request\_token\_payload and view\_data\_token\_payload that are issued as part of the data standards.

## GUID creation protocols

**22.** All GUIDs used within the ecosystem are 32 hex digits (128 bits) allocated 'randomly' by standard methods and must be profiled using the approach in rfc4122 (<https://www.ietf.org/rfc/rfc4122.txt>). As per RFC4122, all GUIDs should be generated in lowercase.

## Identifiers

**23.** This section defines specific identifiers that must be understood by users of the ecosystem.

### find\_correlation\_id

**24.** A correlation identifier provided in find-requests. This allows pension providers and schemes to identify where repeated find-requests are issued in relation to the same citizen user. Its format is a 256-bit hash value, represented as a 64-digit hexadecimal number.

### holdernameGuid

**25.** holdernameGuids are identifiers provided by the pension provider or scheme. They are associated with one or more parts of regulated pension providers or schemes.

### assetGuid

**26.** An assetGuid is a globally unique identifier (GUID) allocated by the pension provider or scheme in relation to a pension asset, on the pension provider or scheme systems, and find\_correlation\_id.

### Pension identifier (Pel)

**27.** Within the ecosystem, the Pel is a unique resource location identifier for a pension asset that has been registered with the CDA. It is composed of two GUIDs separated by a colon, <holdernameGuid>:<assetGuid>.

**28.** They are generated by pension providers or schemes (or their ISPs) and must be globally unique. HoldernameGUIDs are registered by pension providers or schemes during the connection process. The holdernameGuid is registered against a view\_data\_host\_url which identifies a base URL and endpoint to be used when attempting to retrieve data using the view-data API. More than one holdernameGuid can be registered with the same view\_data\_host\_url.

**29. Operational retention period:** pension providers and schemes are required by the legislation to create pension identifiers in accordance with PDP technical standards for a positive match. The legislation also sets out requirements in respect of de-registration where a match is made by the member ceases to be a relevant member. A pension identifier is therefore a long-lived identifier which persists and remains registered with the consent and authorization service for as long as the pension asset to which it relates belongs to a 'relevant member' of a pension scheme in scope of the dashboards service as defined in legislation. The pension provider or scheme that registered the pension identifier has obligations under the dashboards legislation to de-register the pension identifier where the member ceases to be a relevant member. Where a possible match is registered but the user either does not make contact within 30 days to resolve the match, or where the user does make contact, but the provider or scheme is unable to resolve the possible match.

**30.** The persistence of the pension identifier at the consent and authorization service, however, is subject to the user's account at the consent and authorization service remaining active. If the user's account is deleted as a result of a period of non-use, or by the active decision of the user to delete their account, the pension identifiers will then expire.

## **holdernameViewDataUrl**

**31.** The view\_data\_host\_url is the base URL to be used when accessing view-data. It is associated with one or more holdernameGuids.

## **Connection (onboarding) information**

**32.** Participants will work through a number of processes to connect to the PDP ecosystem. These processes are defined outside of the technical standards, but this section provides a summary of the technical information which will need to be supplied by participants to PDP, or by PDP by participants, to enable participants to interface with the PDP ecosystem. This information is also intended to support the API definitions and sequence diagrams defined in the later sections of the technical standards.

### **Provided by PDP**

*token\_host\_url*

**Format:** URL.

**Description:** The host for the token resource endpoint. For example: https://[CDA URL]/ig/token. Also known as the as\_uri.

**Scope:** One per shared environment.

*rreguri\_host\_url*

**Format:** URL.

**Description:** The host for the rreguri resource registration endpoint For example: https://[CDA URL]/ig/rreguri.

**Scope:** One per shared environment.

*introspect\_host\_url*

**Format:** URL.

**Description:** The host for the introspect resource endpoint. For example: https://[CDA URL]/ig/introspect.

**Scope:** One per shared environment.

*perm\_host\_url*

**Format:** URL.

**Description:** The host for the permission endpoint. For example: https://[CDA URL]/ig/perm.

**Scope:** One per shared environment.

*authorize\_host\_url*

**Format:** URL.

**Description:** The host for the authorization endpoint. For example: https://[CDA URL]/ig/authorize.

**Scope:** One per shared environment.

*jwks\_host\_url*

**Format:** URL.

**Description:** The host for the jwks endpoint. For example: https://[CDA URL]/ig/jwk\_uri.

**Scope:** One per shared environment.

*JWT\_audience*

**Format:** String.

**Description:** Authentication server. AS identifier to be supplied/provided in JWTs where the AS is the audience for the token.

**Scope:** One per shared environment.

*Crypto material: certificate package*

**Format:** Package (zip).

**Description:** Contains the participant certificate, certificate chain and the private and public keys to be used to secure the mTLS connection between parties.

**Scope:** One per shared environment.

*Crypto material: signing key package*

**Format:** Package (zip).

**Description:** Contains the private and public keys and the key identifier (kid), to be used to sign and verify the signature of JWTs.

**Scope:** One per shared environment.

**Provided by each pension provider or scheme**

*find\_request\_path\_url*

**Format:** URL.

**Description:** The path to the service that the pension provider or scheme provides to implement the find-requests API. For example:  
<https://api.mypensionsservice/dashboards/find>.

**Scope:** One per shared environment.

*view\_data\_host\_url*

**Format:** URL.

**Description:** The host for the view\_data resource endpoint. For example:  
<https://api.mypensionsservice/dashboards/view-data1/view-data>.

**Scope:** One per shared environment. A view\_data\_host\_url may be registered against multiple holdernameGuids.

*holdernameGuid*

**Format:** Hex string (GUID).



**Description:** The holdernameGuid is the identifier for the pension provider or scheme from which pension details are returned to users at dashboards.

**Scope:** One per shared environment.

## Tokens

### Token summary

**33.** The PDP ecosystem is built on user managed access (UMA 2.0) and thus utilises the tokens required/permitted by UMA 2.0:

- **Requesting party access token:** RPT
- **Permissions ticket:** PMT
- **Protection API token:** PAT
- **Persisted claims token:** PCT

The PDP UMA 2.0 profile adds a custom claims token:

- **Requesting party:** RQP

**34.** In addition, the PDP technical standards also define the following tokens:

- user token
- user account token
- view data token

**35.** The information for each token is summarised in the below, with further details provided in the subsequent sections.

#### *RQP*

**Issuer:** Dashboard.

**Signed:** Yes.

**Encrypted:** No.

**Time to live:** 60 seconds and one time use.

**Purpose:** Asserts the user at the dashboard.

#### *PMT*

**Issuer:** CDA.

**Signed:** No.

**Encrypted:** Yes.

**Time to live:** 60 seconds and one time use.

**Purpose:** Used to request an RPT.

*RPT*

**Issuer:** CDA.

**Signed:** No.

**Encrypted:** Yes.

**Time to live:** 5 days.

**Purpose:** Authorizes the dashboard to retrieve Pels from the CDA and pensions view data from pension providers or schemes. Each RPT relates to a single UMA resource.

*PCT*

**Issuer:** CDA.

**Signed:** No.

**Encrypted:** Yes.

**Time to live:** 90 days.

**Purpose:** Represents the association of the identity of the requesting party (user) at dashboard and at the Authorization Server. Used to 'refresh' RPTs.

*PAT*

**Issuer:** CDA.

**Signed:** Yes.

**Encrypted:** No.

**Purpose:** Authorizes pension providers and schemes to create/update/delete Pels, introspect RPTs and obtain PMTs.

*user\_token*

**Issuer:** CDA.

**Signed:** Yes.

**Encrypted:** No.

**Time to live:** 60 seconds and one time use.

**Purpose:** Encapsulates the verified and self-asserted “find” data of an individual citizen user.

*user\_account\_token*

**Issuer:** CDA.

**Signed:** Yes.

**Encrypted:** Yes.

**Time to live:** 60 seconds.

**Purpose:** Authorizes pension providers and schemes to request a PAT.

*view\_data\_token*

**Issuer:** Pension provider or scheme.

**Signed:** Yes.

**Encrypted:** No.

**Time to live:** 60 seconds.

**Purpose:** Encapsulates the pension “view” data returned by a pension provider or scheme for a pension asset.

## General token principles

- unless stated otherwise, all tokens are JSON Web Tokens (JWT) as per [RFC7519](#)
- JWTs contain base64 encoded JSON
- all tokens are signed using RS256
- certain tokens are encrypted as summarised in the table above and detailed in the individual token sections below
- where the token receiver (dashboard or pension provider or scheme) is only required to “store and forward” and not process the token contents, then on receipt of the token:
  - the token **does not** need to be decrypted (if applicable)
  - the token **does not** need to be decoded
  - the token signature **does not** need to be validated (if applicable)
  - the token schema **does not** need to be validated
- where the token receiver (dashboard or pension provider or scheme) is required to process the token contents, then on receipt of the token:
  - the token **must** be decoded
  - the token signature **must** be validated
  - the token schema **must** be validated
  - token validity periods (IAT, EXP and NBF where present) **must** be respected

Where applicable, this behaviour is specified and detailed in the individual token sections below.

## JSON web token (JWT) signing and verification

**36.** As part of the process to onboard to the CDA the participant will receive a crypto package. This will contain the signing key material. Included is the private key which must be used to generate the signature. This uses RS256 as the signing algorithm to sign a JWT.

**37.** To verify the signature of a JWT the CDA will expose a centralised JWKS endpoint. The participant must be able to call this endpoint to retrieve a jwk for the kid provided in the JWT header. This can be converted to a pem which can then be used to verify the signature.

**38.** Participants must ensure they pass their assigned “kid” parameter. This will be part of the JWT header when returning pension details back to a dashboard. The “kid” will be generated as part of connection and its format will be a GUID. Participants will receive this as part of the crypto package.

### Example of JWT header containing “kid” parameter

```
{  
  
  "alg": "RS256",  
  
  "typ": "JWT",  
  
  "kid": "ec1abf89-225b-49c2-ab87-1d425ac70f8d"  
  
}
```

## Requesting party (RQP)

- **PDP UMA profile name:** pension\_dashboard\_rqp

### Purpose

**39.** The RQP (Requesting Party) represents the pensions dashboard’s assertion at the time of authorization. It confirms of the identity of the user, the dashboard instance and the user’s role. The PDP CDA uses the RQP to link the user@dashboard to the CDA user record.

### Description

**40.** For the schema see “rqp v1\_1.json”. The RQP token is a JWT defined as per the referenced schema, containing the following claims.

- **REQUIRED iss:** Unique identifier within dashboard ecosystem of the dashboard instance issuing the JWT. Provided to the dashboard provider during connection (onboarding).
- **REQUIRED sub:** Unique identifier within scope of iss, of the requesting party (user) which is authenticated to iss at the time the JWT is issued. Format [uniqueUserId]@[iss].

- **REQUIRED aud:** Unique identifier within the scope of the dashboard ecosystem of the Authorization Server. Provided to the dashboard provider during connection (onboarding).
- **REQUIRED iat:** Time of issue.
- **REQUIRED exp:** Time of expiry.
- **REQUIRED jti:** Unique token identifier.
- **REQUIRED role:** States the role in which the requesting party is acting. String value. Must be set to "owner".

## Usage

**41.** The client (dashboard) must issue an RQP aligning with the supplied schema. This must contain claims to identify the user, the dashboard instance, and the role the user is playing in using the dashboard. It must be presented with every authorization request. It must also assert that the user acting in a role is controlling the dashboard instance. The RQP token does not need to be bound to the client (Authorization Server).

- for each authorization attempt the issuer is required to mint a new RQP token
- the token must be signed by the issuer (dashboard) using the private key supplied by PDP as part of Connection (onboarding)

## Permissions ticket (PMT)

- **PDP UMA profile name:** pension\_dashboard\_pmt

## Purpose

**42.** The permissions ticket (PMT) is a correlation handle representing requested permissions. The Authorization Server creates and maintains the PMT. It enables the client to request an RPT.

## Description

**43.** The PMT token is a JWT as defined in the PDP UMA profile. However, as these tokens are issued and consumed by the Authorization Server, they should be treated as opaque by other parties. These other parties would include dashboard clients and pension providers or schemes. Consequently, no schema is provided.

## Usage

**44.** The PMT is generated by the PDP CDA (Authorization Server). It can be passed to different destinations based on how it was generated. It is either passed to the dashboard client directly by the CDA, or to the dashboard by the pension provider or scheme (Resource Server).

**45.** The PMT must be presented by the dashboard client, at the token endpoint and during requesting party redirects. Permission tickets have very short lifetimes. They are single use and they are protected from replay.

- the token will be encrypted by the issuer (Authorization Server)
- the client receiver (dashboard) does not need to decrypt the token
- PMTs are single use; the Authorization Server will issue a new token with a new JTI for every iteration of the permission process

- the client (dashboard) must discard any existing RPT for the resources owned by the pension owner (to which the permission ticket relates), whether a new RPT is issued or not

## Requesting party access token (RPT)

- **PDP UMA profile name:** pension\_dashboard\_rpt

### Purpose

**46.** The requesting party access token (RPT) contains the details of an authorization request. It will detail the specific requesting party (pension owner), and also the specific dashboard client. This will allow access to specific resources at a Resource Server (pension provider or scheme) with stated scope.

### Description

**47.** The RPT token is a JWT as defined in the PDP UMA profile. However, as these tokens are issued and consumed by the Authorization Server, they should be treated as opaque by other parties. These other parties would include dashboard clients and pension provider or schemes. Consequently no schema is provided.

### Usage

**48.** The RPT is generated by the PDP CDA (Authorization Server). It is returned to the dashboard as a result of the successful processing of a PMT.

**49.** RPT is used by the dashboard as the authorization token for requests to access UMA protected resources. The UMA protected resources could be Pels hosted by the CDA provided via the “Retrieve Pel” flow. They could also be pensions asset information hosted by pension providers or schemes, provided via the “Get View Data Request” flow. Separate RPTs are required for retrieving Pels and for each accessed pensions asset.

**50.** After receiving a request containing an RPT, pension providers and schemes must introspect the RPT against the CDA Introspect API. The pension provider or scheme uses the PAT to authorize the introspect API interaction.

- The token will be encrypted by the issuer (Authorization Server).
  - The client receiver (dashboard, pension provider or scheme) does not need to decrypt the token.
- The token may be persisted by the dashboard in accordance with policy (if the client is capable of suitably protecting the token). The token should be deleted by the dashboard if or when the dashboard makes an unsuccessful attempt to access a resource using it (indicating the token has exceeded its lifetime or has been revoked).
- The dashboard client should provide its existing RPT for the resource it requested in the previous call to the same resource server for the same requesting party, if it has one.
- The RPT is bound via OMTLS to the dashboard client by the Authorization Server when it is issued.
- The Authorization Server may revoke a token for its own reasons at any time (including resource owner revocation of policy).
- The token should be presented to the resource server (pension provider or scheme) by the dashboard, the token must be introspected by the resource server at the Authorization Server introspection endpoint, the results of introspection must not be cached at the resource server .
- Introspection of the RPT is used to:

- validate the OAuth MTLS Token Binding fingerprint (of the client, by the Authorization Server)
  - support revocation of the RPT at any time
- Resource servers are responsible for access to the resource. The introspection response needs to be compared with what is stored internally for the resource to ensure access request is authorized.

## Persisted claims token (PCT)

- **PDP UMA profile name:** pension\_dashboard\_pct

### Purpose

**51.** The persisted claims token (PCT) represents the association of the identity of the requesting party (user). The PCT is represented at the dashboard and at the Authorization Server, where it retains the assured user state. This allows the PCT to serve as an 'enhanced' refresh token over longer timeframes.

**52.** The use of the PCT removes the need for the user to step up their authentication level for every authorization request. It does this by creating a persistent association between the user and role at the dashboard instance with their identity at the Authorization Server assured to (a higher) standard.

### Description

**53.** The PCT is a JWT as defined in the PDP UMA profile. However, these tokens are issued and consumed by the Authorization Server. Therefore, they should be treated as opaque by other parties such as the dashboard clients. Consequently, no schema is provided.

### Usage

**54.** The PCT is issued from, and presented at, the Authorization Server token end point. This is part of an authorization request and/or response by the dashboard.

**55.** The PCT supports the user experience of being able to silently re-authorize across all the user's resource servers. This would be based on their assured identity for a defined period (the PCT Time to live).

- the token will be encrypted by the issuer (Authorization Server)
- the client receiver (dashboard) does not need to decrypt the token
- the PCT is bound via OMTLS to the dashboard client by the Authorization Server when it is issued

**56.** The token may be persisted by the dashboard to support future authorization requests. This would apply in the case of the same requesting party in the same role. The client must be capable of suitably protecting the token.

**57.** The dashboard must delete the PCT when it is presented with a new PCT by the Authorization Server, for the same user, in the same role. Or should delete the PCT if or when the dashboard makes an unsuccessful attempt to access a resource using it. This would indicate the token has exceeded its lifetime or has been revoked.

**58.** The token must be presented to the token endpoint at the Authorization Server by the dashboard client.

## Protection API token (PAT)

- **PDP UMA profile name:** pension\_dashboard\_pat

### Purpose

**59.** The protection API token (PAT) allows Resource Servers to register UMA protected resources. This would be on behalf of the resource owner, a pensions owner. Note in this case Resource Servers are pension providers or schemes. UMA protected resources are pension assets.

**60.** The PAT is also used to authorize Resource Servers. This allows them to introspect RPTs received with pensions data view requests.

**61.** It can also be used to retrieve PMTs from the CDA. This would be useful where the RPT in the pensions data view request is missing or expired.

### Description

**62.** The PAT is a JWT as defined in the PDP UMA profile. However, these tokens are issued and consumed by the Authorization Server. Thus, they should be treated as opaque by other parties such as the pensions providers or schemes. Consequently, no schema is provided.

### Usage

**63.** Pension owners grant permission for the Authorization Server to determine their access policy. This controls access to pension owner's protected resources. This would be at one or more Resource Servers holding pension assets managed by pension providers. Pension owners are able to do this in their role as resource owners.

**64.** The PAT is an OAuth token of scope uma\_protection.

**65.** PATs must be persisted by the Resource Server to which they were issued. They must be associated with the resource owner's record(s) at that location.

- the token will be signed by the issuer (Authorization Server)
- the client receiver (pension provider or scheme) does not need to validate the signature
- the PAT is bound to the Resource Server client when it is issued
- the Authorization Server may revoke a PAT for its own reasons at any time (including resource owner revocation of policy)

**66.** The token may be deleted by the Resource Server if it is revoked or when the Resource Server makes an unsuccessful attempt to access an Authorization Server API using it.

## User token

### Purpose



**67.** The user token encapsulates the verified and self-asserted personal data for an individual user performing a request to find their pensions.

### **Description**

**68.** For the schema see `find_request_token_payload.json` (issued as part of the data standards).

**69.** The user token is a JWT as defined in the referenced schema, issued by the PDP CDA. Pension providers and schemes are required to parse and process the user token contents. This would be as part of processing the find request.

### **Usage**

**70.** A user token is included with every find request sent by the PDP CDA to pension providers or schemes:

- the token will be signed by the issuer (PDP CDA)
- the client (pension provider or scheme) must validate the signature of the token
- the client must validate the token contents against the published version of the token schema
- The token is issued for the sole purpose of triggering the find process and its contents must not be persisted by the pension provider or scheme if: (a) the pension provider/scheme determines that the token does not relate to any relevant member of the scheme, (b) if a possible match is registered but the dashboard user does not make contact within 30 days, or (c) a possible match is registered and the user makes contact but the pension provider/scheme is not able to resolve the possible match within such time as may be reasonably allowed by the provider/scheme

## **User account token**

### **Purpose**

**71.** The user account token enables a Resource Server (pension provider or scheme) to obtain a PAT on behalf of the resource owner. It is a temporary credential.

### **Description**

**72.** The user account token is a JWT. These tokens are issued and consumed by the Authorization Server. Thus they should be treated as opaque by other parties such as the dashboard clients. Consequently, no schema is provided.

### **Usage**

**73.** The user account token is an OAuth2 authorization grant. This is expressed as a JWT, which can be exchanged for the PAT. It is provided alongside the user token with every find request sent by the PDP CDA to pension providers or schemes.

**74.** The pension provider or scheme is required to use the user account token to request a PAT. This is following a successful find operation where it needs to register one or more found pensions assets with the PDP CDA.

- the token will be signed by the issuer (PDP Authorization Server)

- the client receiver does not need to validate the signature
- the token will be encrypted by the issuer (Authorization Server)
- the client receiver (pension provider or scheme) does not need to decrypt the token
- the token must not be persisted by the pension provider or scheme. They must not be stored beyond the time period required to process the find request and a PAT

## View data token

### Purpose

**75.** The view data token encapsulates the data related to the matched pension asset. It includes associated retirement illustrations. This is returned with the pensions view data request.

### Description

For the schema see `view_data_token_payload json` (issued as part of the data standards).

**76.** The view data token is returned by a pension provider or scheme to a dashboard in response to a view pensions data request. It is a JWT as defined in the referenced schema. Pension providers and schemes are required to structure the view data token to align with the schema. Dashboards are also required to parse and process the view data token contents to ensure alignment with the schema. The view data token contains the following claims.

- **REQUIRED sub:** The assetGUID of the returned pension.
- **REQUIRED iss:** The `view_data_host_url` value.
- **REQUIRED aud:** `https://pensionsdashboards.org.uk/`.
- **REQUIRED exp:** Set to 24 hours from issuance (86400 seconds).
- **REQUIRED iat:** Set to the time of issue of the token. This is an epoch date value to the nearest second.
- **REQUIRED jti:** Set to a randomly generated GUID.
- **REQUIRED view\_data:** the encapsulated view data for the returned pension

### Usage

**77.** A view data token is returned in the view pensions data response. This is sent from a pension provider or scheme to the requesting dashboard.

- the token will be signed by the issuer (pension provider or scheme)
- the client (dashboard) must validate the signature of the token
- the issuer and client must ensure the token structure and contents align with the published version of the token schema
- the token and its contents must not be persisted by the dashboard beyond the user session at the dashboard

## API technical standards

### Transaction monitoring

**78.** Transaction monitoring and correlation will be achieved in the following way:

- All interfaces will carry a unique transaction identifier GUID (X-Request-ID) for logging, audit and monitoring purposes.
- The transaction identifier must be unique to each request, including retried requests.
- The transaction identifier is issued by the party which starts the transaction. Both parties to the transaction must retain the same transaction identifier in their respective audit logs.
- For transactions which PDP initiates, PDP will generate the identifier; for transactions which the pension provider, scheme or dashboard initiates, it must generate the identifier.
- For API calls the transaction ID must be transmitted via the HTTP header: X-Request-ID.
- When redirecting to the consent and authorization service, dashboard providers must include the transaction identifier (request\_id) in the URL as a query parameter. For example `https://{url}?request_id={request_id}`.

## Caching

**79.** Pension providers and schemes must not cache API responses.

## Third-party standards

**80.** These standards are built upon a number of third-party standards:

### *UMA grant 2.0*

- **Issuing authority:** Kantara Initiative
- **Contact details:** <https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html>
- **Version:** 2.0

### *UMA federated authorization 2.0*

- **Issuing authority:** Kantara Initiative
- **Contact details:** <https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html>
- **Version:** 2.0

## API definitions

### Authorization

**81.** In general permission to access API endpoints is governed by mTLS. This is possible because the PDP ecosystem is a closed ecosystem. Within the ecosystem participants are registered and endpoints are secured using private PKI certificates issued on registration. This allows connections to be established via a mutual TLS connection with the central infrastructure. It also allows connections between dashboards providers and pension providers and schemes using the same mechanism.

**82.** Additional authorization may be required for specific endpoints. This is detailed in the documentation for specific APIs.

## Encoding

**83.** Request and response bodies must be UTF-8 encoded.

## Definitions

### *find-requests (find) API*

The find-requests API is hosted by pension providers and schemes. The purpose of the find-request API is to accept and act upon find requests issued by the CDA.

These requests contain the pension owner's PII data. This data must be used by each pension provider or scheme to determine matches against their internal records.

In addition to the PII data, the find request will also provide a user\_account\_token, user account token. This user\_account\_token is used to retrieve a PAT (protection API token), PAT. The PAT is required by the pension providers and schemes to register and maintain Pel resources.

**Hosted by:** Pension providers or schemes.

**Called by:** CDA.

**Used in:**

- pension provider or scheme:match pensions and register Pels

**OpenAPI specification:** See "find-requests v1\_3.yaml" and the find\_request\_token\_payload (issued as part of the data standards).

**Performance and SLA's :** See code of connection (CoCo2.1.1, CoCo2.1.2, CoCo2.1.4, CoCo2.1.5)

### *token API*

The token API is hosted by the CDA. It is used by pension providers and schemes and dashboard providers to obtain access tokens in the form of JWTs. The access token JWTs are used throughout the system to allow access to controlled resources.

The token endpoint is an implementation of the standard access token endpoint. This is described at <https://datatracker.ietf.org/doc/html/rfc6749#section-3.2>.

**Hosted by:** CDA.

**Called by:** Pension providers or schemes or pensions dashboard providers.

**Used in:**

- pension provider or scheme: match pensions and register Pels

**OpenAPI specification:** See “token v2\_1.yaml”.

### *rreguri API*

The rreguri API is hosted by the CDA. Pension providers and schemes use this API to register and maintain registered Pels relating to pensions matched as a result of find-requests. The permitted states and transition paths are documented in the pensions identifier (Pel) status and transition section.

Once a Pel has been registered, responsibility for maintaining the Pel belongs with the pension providers or schemes. This responsibility remains with pension providers and schemes until they have deleted the registered Pel.

**Hosted by:** CDA.

**Called by:** Pension providers or schemes.

**Used in:**

- pension provider or scheme:match pensions and register Pels
- pension provider or scheme:update match status
- pension provider or scheme:delete registered Pel

**OpenAPI specification:** See “rreguri v1\_3.yaml”.

**Authorization:** In addition to the standard use of mTLS, a valid PAT must be sent as bearer authentication with each request.

### *view-data API*

The view-data API is hosted by pension providers or schemes. It is called by dashboard providers to retrieve pension and pension illustration details. These details will be associated with a pension asset that has been registered under a Pel.

**Hosted by:** Pension providers or schemes.

**Called by:** Pensions dashboard providers.

**Used in:**

- pension provider or scheme:get view data request

**OpenAPI specification:** See “view-data v1\_2.yaml” and the view\_data\_token\_payload schema (issued as part of the data standards).

**Authorization:** In addition to the standard use of mTLS, a valid RPT must be sent as bearer authentication for the GET request.

**Performance and SLA's:** See code of connection (CoCo2.1.3, CoCo2.1.4, CoCo2.1.5).

## *introspect API*

The introspect API is hosted by the CDA. It is used by pension providers and schemes to check whether access tokens provided in requests provide sufficient permissions to allow access to requested data. Specifically, the pension providers and schemes will check if an RPT provided in a view-data request grants sufficient access for the view data to be returned.

**Hosted by:** CDA.

**Called by:** Pension providers or schemes.

**Used in:**

- pension provider or scheme:get view data request

**OpenAPI specification:** See “introspect v1\_1.yaml”.

**Authorization:** In addition to the standard use of mTLS, a valid PAT must be sent as Bearer authentication for the GET request.

## *perm API*

The perm (permissions) API is hosted by the CDA. It is used by pension providers and schemes to retrieve permission tokens. These permission tokens are returned to view-data API clients so that they can obtain the correct access token (RPT) to retrieve data that they have requested.

**Hosted by:** CDA.

**Called by:** Pension providers or schemes.

**Used in:**

- pension provider or scheme:get view data request

**OpenAPI specification:** See “perm v1\_1.yaml”.

**Authorization:** In addition to the standard use of mTLS, a valid PAT must be sent as bearer authentication for the GET Request.

## Sequences diagrams and API usage

### Pension provider or scheme – match pensions and register Pels

#### Scenario

**84.** This section details the interactions that occur between the ecosystem participants in the scenario where a find request is issued by the CDA.

**85.** The find request must be immediately acknowledged by the pension provider or scheme, which will then enact a search based on the match criteria provided for the citizen user in the request and the pension asset owner details held on the pension provider or scheme systems.

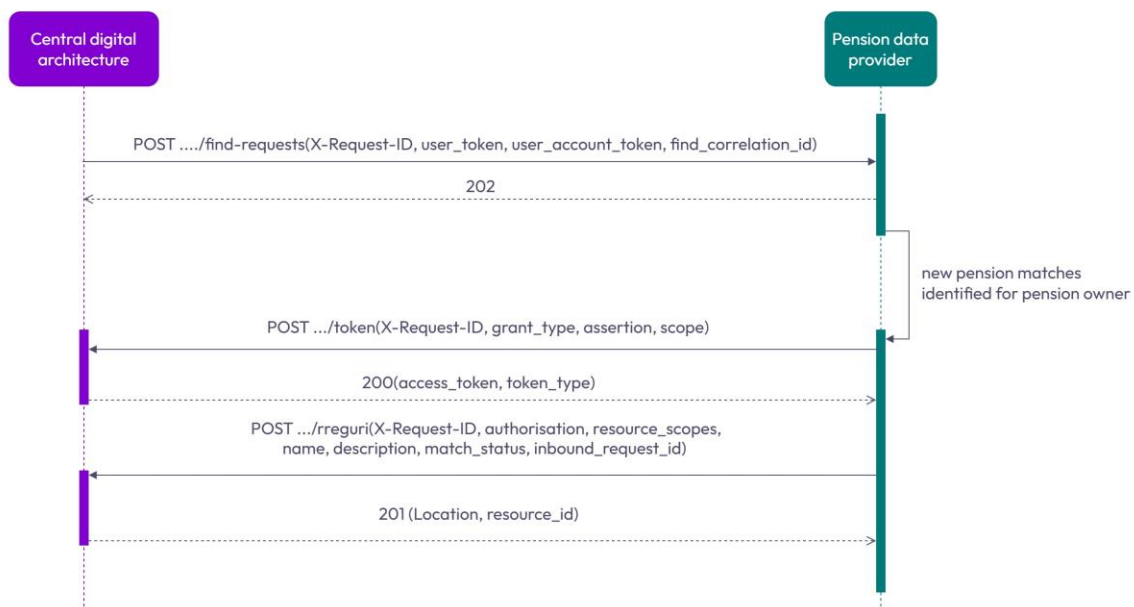
**86.** For each match identified, the pension provider or scheme will register an appropriate Pel with the CDA.

## Ecosystem participants

- CDA
- pension provider or scheme

## APIs

- **find-requests:** Hosted by pension provider or scheme.
- **token:** Hosted by CDA.
- **rreguri:** Hosted by CDA.



## API usage details

**87.** The sections below describe the detailed usage of the APIs in this scenario.

### CDA issues find request

**88.** The CDA issues a request for the pension provider or scheme to initiate a search for pensions matching based on criteria provided in the user token.

**89.** A pension provider or scheme will use the information provided in the user token to determine if there any matches within the internal records of pension asset owners.

---

#### Call Data Provider POST find-requests Request

Name	Sent as	Source	Logic
<b>BaseUrl&amp;Endpoint</b>	Base URL.	Provided by the pension provider or scheme during onboarding.	See find_request_path_url.
<b>X-Request-ID</b>	Header.	Generated by CDA.	A unique ID for the request to allow tracing of request flows.
<b>find_correlation_id</b>	Header.	Generated by CDA.	A unique identifier to link find-requests that are issued in relation to a particular citizen user.
<b>user_token</b>	Request body.	Generated by CDA.	See user token. The user token payload must be validated against the expected schema.
<b>user_account_token</b>	Request body.	Generated by CDA.	See user account token. This is a JWT token which the pension provider or scheme does not need to interpret but will use to obtain a PAT, see PAT. The pension provider or scheme should not attempt to validate the token. The user account token will expire 60 seconds after issue. This aligns with the SLA for registering a PeI after a find request has been issued.

---

#### Call Data Provider POST find\_requests Response (202 Success)

Name	Sent as	Source	Logic
<b>Not applicable.</b>	Provided by the pension provider or scheme.	Not applicable.	The pension provider or scheme must immediately respond to the request with an acknowledgement of the request.

---

#### Call Data Provider POST find\_requests Response (400 Bad Request)



Name	Sent as	Source	Logic
<b>Not applicable.</b>	Provided by the pension provider or scheme.	Not applicable.	Pension provider or scheme must immediately respond to invalid requests. This may include if the request has invalid, or missing, parameters or the user token is not correctly signed or its payload does not conform to the schema. The pension provider or scheme must immediately respond to the request with the 400, Bad Request, response.

### Call Data Provider POST find\_requests Response (400 Bad Request or 403 Forbidden or connection refused/reject handshake)

Name	Sent as	Source	Logic
<b>Not applicable.</b>	Provided by the pension provider or scheme.	Not applicable.	For mTLS related errors where the client certificate for the request is missing or invalid, the find-requests and view-data APIs should return a 403 or 400 error. If this isn't possible due to technology constraints, a lower-level transport layer error (for example "connection refused", "reject handshake") should be returned

### Pension provider or scheme obtains PAT

**90.** The pension provider or scheme will identify any full or possible matches between the citizen user details provided in the find request and the pension asset owner details on their systems. Any possible or full matches identified must be registered with the CDA, see repeated match behaviour for special behaviour in repeat match scenarios.

**91.** Before registering matching assets, the pension provider or scheme will need to obtain a PAT. Per find request, a single PAT should be used to register all the pension assets for a particular pension asset owner/citizen user. If, when a new match is identified, pension asset owner already has a PAT associated with them then a new PAT should be requested to register the new match and this PAT should replace the stored PAT for previous registrations for the pension asset owner/citizen user.

**92.** The User Account Token provided in the find request is an OAuth2 authorization grant. This is expressed as a JWT (JSON web token) which can be exchanged for the PAT as described below. This is an OAuth2 access token, as per the OAuth2 standard. For the POST token interaction the request body must be sent with a content type of "x-www-form-urlencoded".

## Call CDA POST token Request

Name	Sent as	Source	Logic
<b>BaseUrl&amp;Endpoint</b>	Base URL.	Provided to the pension provider or scheme in the onboarding pack during the Connection process.	See token_host_url.
<b>X-Request-ID</b>	Header.	Generated by the pension provider or scheme.	A unique ID for the request to allow tracing of request flows.
<b>grant_type</b>	Request body parameter.	Provided by the pension provider or scheme.	In this scenario the pension provider or scheme will set this to "urn:ietf:params:oauth:grant-type:jwt-bearer".
<b>assertion</b>	Request body parameter.	Provided by the CDA.	The user account token provided in the find_requests call.
<b>scope</b>	Request body parameter.	Provided by the pension provider or scheme.	In this scenario the pension provider or scheme will set to "uma_protection".

## Call CDA POST token Response (200 Success)

Name	Sent as	Source	Logic
<b>access_token</b>	Response body.	Provided by the CDA.	The owner PAT expressed as a JWT. The pension provider or scheme does not need to interpret the token but will use it when accessing the rreguri API endpoints.
<b>token_type</b>	Response body.	Provided by the CDA.	The pension provider or scheme must check that this is "pension_dashboard_pat". They should throw an exception if the value is not correct.

## Pension provider or scheme registers Pel(s)

**93.** Once the pension provider or scheme has obtained a PAT they must register any identified matches with the CDA.

**94.** The pension provider or scheme will record sufficient information about a successful registration to allow the pension provider or scheme to maintain the registered Pels and to service calls to retrieve the view-data associated with the associated assetGuid.

**95.** In order to enable this the recorded information will need to include the following:

- **assetGuid:** A unique identifier for a pension asset owner/ find\_correlation\_id. Where there are multiple matches against a pension asset owner for find-requests with different find\_correlation\_ids the pension provider or scheme must generate a unique ID for each so that they can identify which match is associated with a particular view-data request, see pension provider or scheme – get view data request.
- **resource\_id:** The ID for the registered Pel resource provided in the success response returned when the Pel resource is registered.
- **PAT:** The current PAT that can be user to maintain the registered Pel and introspect RPTs in relation to view-data requests (initially this will be the PAT used to register the Pel).
- **match\_status:** The current status of the match.
- **find\_correlation\_id:** As passed in the find request, this is required to resolve repeated match scenarios, see repeated match behaviour.

**96.** It may also need to include these attributes to link to the internal data held by the pension provider or scheme:

- **pensionOwnerIdentifier (the internal ID for a pension owner):** The pension provider or scheme internal identifier for a pension asset owner.
- **internalAssetIdentifier (the internal ID for a pension asset):** The pension provider or scheme internal identifier for a pension asset.

---

### Call CDA POST rreguri Request

Name	Sent as	Source	Logic
<b>BaseUrl&amp;Endpoint</b>	Base URL.	Provided to the pension provider or scheme in the onboarding pack during the connection process.	See rreguri_host_url.
<b>X-Request-ID</b>	Header.	Generated by the pension provider or scheme.	A unique ID for the request to facilitate tracing.
<b>Authorization:</b>	Header.	Provided by the CDA.	The access_token provided in the Post tokens response. Provided as "Bearer " + access_token.
<b>resource_scopes</b>	Request body.	Provided by the pension provider or scheme.	In this scenario the pension provider or scheme will set the scopes to ["value", "owner", "delegate"].

<b>name</b>	Request body.	Provided by the pension provider or scheme.	The urn for the PeI to be registered in the form "urn:pei:<PeI>.
<b>description</b>	Request body.	Provided by the pension provider or scheme.	This must be the "scheme name" that will be returned in response to the GET pension-details (view) response for the PeI.
<b>match_status</b>	Request body.	Determined by the pension provider or scheme.	The initial match status for the registered PeI. Either "match-yes" for a full match or "match-possible" for a possible match.
<b>inbound_request_id</b>	Request body.	Not applicable.	In this instance the X-Request-ID sent in the find request.

### Call CDA POST rreguri Response (201 Success)

Name	Sent as	Source	Logic
<b>Location</b>	Header.	Provided by the CDA.	The location of the resource. For example, https://cdapath/registered-peis/resource_id.
<b>resource_id</b>	Response body.	Provided by the CDA.	The unique_id of the newly created registered PeI.

## Pension provider or scheme – update match status

### Scenario

**97.** This section details the interactions between the ecosystem participants when pension provider or scheme processes determine that the match status needs of a registered PeI needs to be updated. This could be because:

- A possible match has been resolved to a full match either due to a manual process or a repeated match scenario, see behaviour where there is a pre-existing current match.

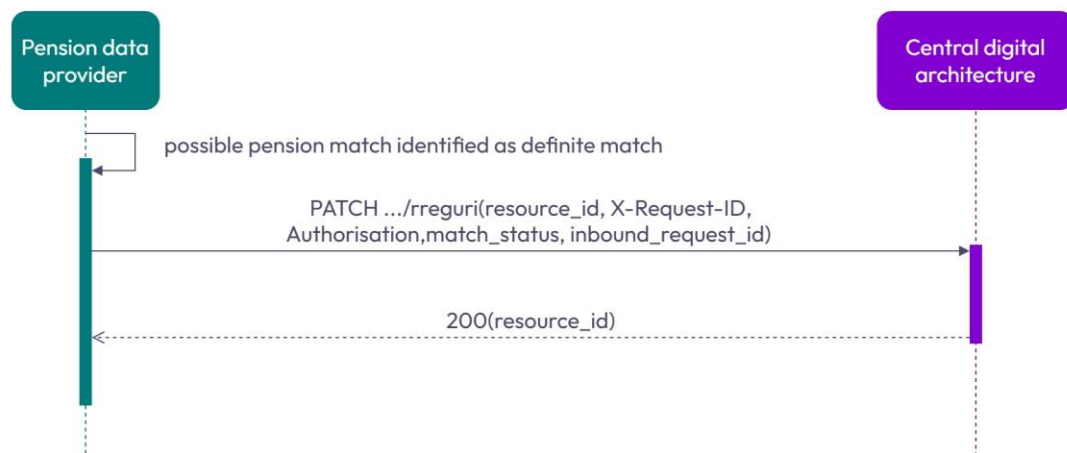
### Ecosystem participants

- CDA
- pension provider or schemes

### APIs

- **rreguri**: Hosted by CDA.

## Sequence diagram



## API usage details

98. The section below describes the detailed usage of the APIs in this scenario.

### Call CDA PATCH rreguri Request

Name	Sent as	Source	Logic
BaseUrl&Endpoint	Base URL.	Provided to the pension provider or scheme in the onboarding pack during the connection process.	See rreguri_host_url.
X-Request-ID	Header.	Generated by pension provider or scheme.	A unique ID for the request to facilitate tracing.
Authorization:	Header.	The access_token (PAT) provided in the Post token response received during the PeI registration process, access_token.	"Bearer " + access_token. The PAT token held against the pei in relation to the pension owner whose match status has been clarified.

<b>resource_id</b>	Path parameter.	Returned by the CDA in the POST rreguri response.	Not applicable.
<b>match_status</b>	Request body.	Determined by the pension provider or scheme.	The new match status, this may only be "match-yes".
<b>inbound_request_id</b>	Request body.	As per logic.	Only required to be populated if the PeI status change was triggered by a repeat find request. Where it is required to be populated, the inbound_request_id must be set to the X-Request-ID sent in the find-request which triggered the update (likely the latest find request). The inbound_request_id must be formatted as a GUID, matching the format of X-Request-ID. If the update was not triggered by the CDA this should not be provided.

### Call CDA Patch rreguri Response (200 Success)

Name	Sent as	Source	Logic
<b>resource_id</b>	Response body.	Not applicable.	The unique_id of the updated rreguri.

## Pension provider or scheme – delete registered PeI

### Scenario

**99.** This section details the interactions between the ecosystem participants when the pension provider or scheme determines that registered PeI needs to be deleted. This may be triggered in the following scenarios:

- a possible match is determined to not be a match
- a possible match was not confirmed as full (no citizen contact within 30 days or the match could not be resolved)
- the pension provider or scheme identifies that a PeI was registered erroneously
- registered PeI relates to a crystallised or transferred pension asset

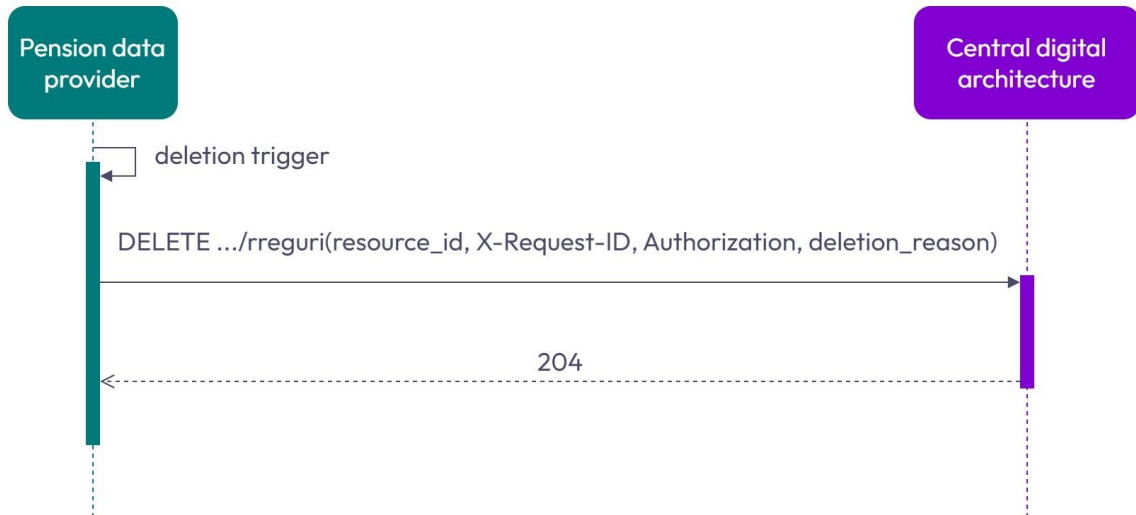
### Ecosystem participants

- CDA
- pension provider or schemes

## APIs

- **rreguri**: Hosted by CDA.

## Sequence diagram



## API usage details

**100.** The section below describes the detailed usage of the APIs in this scenario.

### Call CDA DELETE rreguri Request

Name	Sent as	Source	Logic
<b>BaseUrl&amp;Endpoint</b>	Base URL.	Provided to the pension provider or scheme in the onboarding pack during the connection process.	See rreguri_host_url.
<b>X-Request-ID</b>	Header.	Generated by the pension provider or scheme.	A unique ID for the request to facilitate tracing.
<b>deletion_reason</b>	Query parameter.	Determined by the pension provider or scheme.	The reason that the PeI resource is being deleted: match-no – a match-possible has previously

			been registered but found not to be a match; match-timeout – a possible match has previously been registered but has since been removed (for example, if the pension owner did not confirm a match within the required time period); match-withdrawn – an erroneous match made or possible match has now been withdrawn; asset-removed – a previously registered Pel for a match made or possible match has been removed (for example, benefit crystalised or transferred out).
<b>Authorization:</b>	Header.	The access_token provided in the Post token response, access_token.	"Bearer " + access_token. For information, an expired PAT (access_token) is permitted to be used to authorize the delete Pel rreguri API call. This is applicable when, for example, a pension user hasn't visited the platform for a while and the PAT has expired.
<b>resource_id</b>	Path parameter.	The resource ID return in the POST rreguri response, resource_id.	Not applicable.

### CDA DELETE rreguri Request Response (204 Success)

Name	Sent as	Source	Logic
Not applicable.	Not applicable.	Not applicable.	Not applicable.

## Pension provider or scheme – get view data request

### Scenario

**101.** This section details the interactions between the ecosystem participants in the scenario where a pension dashboard provider requests the pension details related to a particular assetGuid from a pension provider or scheme.

### Ecosystem participants

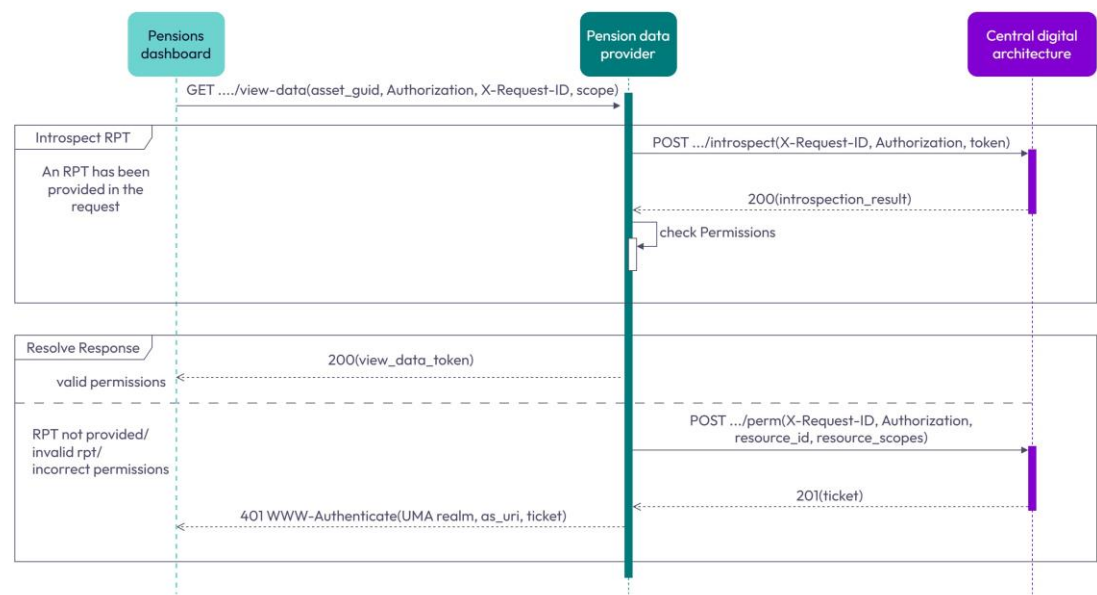


- pension dashboard providers
- pension providers or schemes

APIs

- **view-data**: Hosted by pension provider or scheme.
- **introspect**: Hosted by CDA.
- **perm**: Hosted by CDA.

Sequence diagram



API usage details

**102.** The section below describes the detailed usage of the APIs in this scenario.

**Pensions dashboard provider issues a request to retrieve the view data**

**103.** The pensions dashboard provider issues a request for the pension provider or scheme to retrieve the pension details associated with a Pel in relation to a particular user.

Call Data Provider GET view-data (Request)

Name	Sent as	Source	Logic
BaseUrl&Endpoint	Base URL.	Provided by the pension provider or scheme for each holdernameGuid.	See view_data_host_url.

<b>X-Request-ID</b>	Header.	Generated by the dashboard provider.	A unique ID for the request to facilitate tracing.
<b>Authorization:</b>	Header.	Dashboard provider.	The RPT relating to the citizen user seeking access.
<b>asset_guid</b>	Path parameter.	The assetGuid of the PeI who's details are required.	See assetGuid.
<b>scope</b>	Query parameter	"owner".	Currently only owner is supported. Any other value must result in the pension provider or scheme responding with a 400 bad request. If not provided the pension provider or scheme will assume "owner".

#### Call Data Provider GET view-data (200 Response)

Name	Sent as	Source	Logic
<b>view_data_token</b>	Response body.	Provided by the pension provider or scheme.	The pension provider or scheme will construct a JWT token containing the view data derived by the pension provider or scheme for the pension identified by the passed asset_guid. The schema for the view_data_token_payload is issued as part of the data standards.

#### Call Data Provider GET view-data (401 Response)

Name	Sent as	Source	Logic
<b>WWW-Authenticate</b>	Header.	Provided in the error response.	The WWW-authenticate header see ( <a href="https://datatracker.ietf.org/doc/html/rfc7235?utm_source=localhost%3A8080#section-4.1">https://datatracker.ietf.org/doc/html/rfc7235?utm_source=localhost%3A8080#section-4.1</a> ), containing the following claims: realm, in this scenario the returned realm must always be "PensionDashboard"; as_uri, the path to the token endpoint; ticket, the PMT provided as a JWT token. The dashboard does not need to interpret this ticket. The challenge must be set to "UMA". Example: WWW-Authenticate: UMA realm="PensionDashboard", as_uri="example", ticket="example"

---

### Call Data Provider GET view-data Response (400 Bad Request or 403 Forbidden or connection refused/reject handshake)

Name	Sent as	Source	Logic
Not applicable	Provided by the pension provider or scheme.	Not applicable	For mTLS related errors where the client certificate for the request is missing or invalid, the find-requests and view-data APIs should return a 403 or 400 error. If this isn't possible due to technology constraints, a lower-level transport layer error (for example "connection refused", "reject handshake") should be returned

---

### Pension provider or scheme introspects the RPT

**104.** If an RPT has been provided in the request then, prior to responding to the request, it is the pension providers' and schemes' responsibility to ensure that the requesting party has current, valid permissions to access the requested pension details.

---

### Call CDA POST introspect (Request)

Name	Sent as	Source	Logic
BaseUrl&Endpoint	Base URL.	Provided by the CDA for each shared environment during connection.	See introspect_host_url.
X-Request-ID	Header.	Generated by the pension provider or scheme.	A unique ID for the request to facilitate tracing.
Authorization:	Header.	Provided by the pension provider or scheme.	"Bearer " + access_token. The PAT token held against the PeI in relation to the pension owner of the asset.
token	Request body.	Provided in the view-data request Authorization: header.	Not applicable.

---

### Call CDA POST introspect Response (200 Success)

Name	Sent as	Source	Logic
<b>active</b>	Response body.	CDA.	Denotes if the passed token (RPT) is active. If this is not true, the pension provider or scheme must attempt to obtain a PMT and return this in a 401 error response.
<b>token_type</b>	Response body.	CDA.	The type of the passed token. If provided the pension provider or scheme must check that it is "pension_dashboard_rpt". If the token type is not correct, the pension provider or scheme must reject the view-data request with HTTP 400 Bad Request.
<b>exp</b>	Response body.	CDA.	If provided the pension provider or scheme must check that the RPT has not expired. If the RPT has expired, the pension provider or scheme must attempt to obtain a PMT and return this in a 401 error response.
<b>iss</b>	Response body.	CDA.	Not applicable.
<b>permissions</b>	Response body.	CDA.	The CDA will return a single set of resource information for each resource granted under the RPT for the account denoted by the PAT. If no permissions are returned, the pension provider or scheme must attempt to obtain a PMT and return this in a 401 error response.
<b>permissions.resource_id</b>	Response body.	CDA.	The resource_id of the registered Pel. The pension provider or scheme must identify the permission details for the resource_id associated with the asset_guid provided in the view-data request. If a permission is not returned for the expected resource_id, the pension provider or scheme must attempt to obtain a PMT and return this in a 401 error response.

<b>permissions.resource_scopes</b>	Response body.	CDA.	The access scopes allowed under the RPT for the resource. The pension provider or scheme must check that the resource_scopes for the identified permission details contains the scope provided in the view-data request. If the requested scope is not present, the pension provider or scheme must attempt to obtain a PMT and return this in a 401 error response.
<b>permissions.exp</b>	Response body.	CDA.	If provided the pension provider or scheme must check that the permission has not expired. If the permission has expired, the pension provider or scheme must attempt to obtain a PMT and return this in a 401 error response.

### Pension provider or scheme retrieves PMT

**105.** Where the pension provider or scheme cannot confirm that the requester has permission to access the view data for the passed asset\_guid, it must retrieve a PMT from the CDA to pass back to the requestor to enable the requester to generate an appropriate RPT prior to a subsequent view data request.

**106.** This will be triggered:

- if no RPT is presented in the view\_data request
- if the permissions returned in by the introspection results do not confirm access to the resource in the requested scope

### Call CDA POST perm (Request)

Name	Sent as	Source	Logic
<b>BaseUrl&amp;Endpoint</b>	Base URL.	Provided by the CDA for each shared environment during connection.	See perm_host_url.
<b>X-Request-ID</b>	Header.	Generated by the pension provider or scheme.	A unique ID for the request to facilitate tracing.

<b>Authorization:</b>	Header.	Provided by the pension provider or scheme.	"Bearer " + access_token. The PAT token held against the PeI in relation to the pension owner of the asset.
<b>resource_id</b>	Request body.	Provided by the pension provider or scheme.	The PAT resource_id held against the PeI provided by the CDA when the PeI was registered.
<b>resource_scopes</b>	Request body.	Provided by the pension provider or scheme.	Set to ["value", "owner"].

### Call CDA POST perm Response (201 Success)

Name	Sent as	Source	Logic
ticket	Response body.	CDA.	A JWT representing a PMT. The pension provider or scheme will not need to examine the contents of the JWT but will simply return it in the view_data 401 response.

## Repeated match behaviour

**107.** It is the pension provider or scheme's responsibility to define their criteria in terms of what defines a full or possible match between the citizen user data, provided in the user token in a find request, and pension asset owner details held on its systems. However, to ensure consistent behaviour within the ecosystem, it is necessary for the pension providers and schemes to behave in a consistent way when repeat matches are made against a single pension asset owner as a result of multiple find requests.

**108.** Repeat matches against a single pension asset owner can occur in 2 scenarios:

- multiple matches for find-requests received in relation to the same actual citizen user. This can be identified by the fact that identical find\_correlation\_id header information has been provided in the find requests
- multiple matches for find-requests received in relation to different citizens. This can be identified by the fact that different find\_correlation\_id header information has been provided in the find requests

## Multiple CDA accounts for the same citizen user

**109.** The possibility of the same citizen user creating 2 CDA accounts cannot be ruled out. In this situation the ecosystem will treat them as 2 different citizen users (with different find\_correlation\_ids).

## Repeat matches for the same citizen user

**110.** Over time, multiple find requests may be issued by the CDA in relation to the same citizen user.

**111.** When a find-request is received in relation to a citizen user, there may be previous matches already recorded for the citizen user against a particular pension asset owner.

**112.** There may be a single current active match which may be either a full match, “match-yes”, or a possible match, “match-possible”.

**113.** Additionally, there may be any number of historical matches where a PeI registration has been deleted by the pension provider or scheme, namely:

- a historic possible match that has been resolved as not a match (deleted with reason match-no)
- historic possible matches (deleted with reason match-timeout)
- historic matches that have been withdrawn having been erroneously registered (deleted with reason match-withdrawn)
- historic matches that have been removed as the asset has been, for example, transferred, crystallised, begun drawdown (deleted with reason asset-removed)

### Behaviour where there is a pre-existing current match

**114.** Regardless of any historic matches, where there is current registered PeI for a citizen user/pension asset owner match, the pension provider or scheme must behave as described in the following table.

Scenario	Existing full match	Existing possible match
<b>Full match resulting from a subsequent find-request.</b>	The pension provider or scheme leaves the existing registered PeI and does not register a new PeI resource.	The pension provider or scheme updates the existing registered PeI to indicate a full match.
<b>Possible match resulting from a subsequent find-request.</b>	The pension provider or scheme leaves the existing registered PeI and does not register a new PeI resource.	The pension provider or scheme leaves the existing registered PeI and does not register a new PeI resource. The expiry date (to resolve the match status) of the original possible match should not be extended.
<b>Subsequent find-request does not result in a match.</b>	The pension provider or scheme leaves the existing registered PeI.	The pension provider or scheme leaves the existing registered PeI.

### Behaviour where there is no pre-existing current match

**115.** Where there is no current match for the pension asset owner, but there are historic matches, the pension provider or scheme must behave as described in the following table.

Scenario	Historic possible match resolved to no match (regardless of other historic matches)	Any other combination of historical matches
Full match resulting from a subsequent find-request.	The pension provider or scheme will create a new registered Pel. resource with match status “match-yes”.	The pension provider or scheme will create a new registered Pel resource with match status “match-yes”.
Possible match resulting from a subsequent find-request.	The pension provider or scheme will not create a new registered Pel.	The pension provider or scheme will create a new registered Pel resource with match status “match-possible”.
Subsequent find-request does not result in a match.	The pension provider or scheme will not create a new registered Pel.	The pension provider or scheme will not create a new registered Pel.

## Matches between multiple citizen users and a single pension asset owner

**116.** It is possible for pension provider or scheme systems to identify matches, to a single pension asset owner, as a result of find requests related to multiple citizen users.

**117.** Where this scenario arises, the pension provider must behave as described in the following table.

Scenario	Current possible match for citizen user 1	Current full match for citizen user 1
Possible match resulting from a subsequent find-request for citizen user 2.	The pension provider or scheme will leave the existing Pel registration linked to citizen user 1 and generate a new assetGuid and creates a new registered Pel resource using the new assetGuid with match status “match-possible” linking this to citizen user 2.	The pension provider or scheme will leave the existing Pel registration linked to citizen user 1 and generate a new assetGuid and creates a new registered Pel resource using the new assetGuid with match status “match-possible” linking this to citizen user 2. In this scenario, it is the pension provider or scheme’s responsibility



		to determine whether to return view-data with a “CONT” status (data item 2.004 set to true) and limited data for the full matches until the conflict has been resolved.
<b>Full match resulting from a subsequent find-request for citizen user 2.</b>	The pension provider or scheme leaves the existing Pel registration linked to citizen user 1 and generates a new assetGuid and creates a new registered Pel resource using the new assetGuid with match status “match-yes” linking this to citizen user 2. In this scenario, it is the pension provider or scheme’s responsibility to determine whether to return view-data with a “CONT” status (data item 2.004 set to true) and limited data for the full match until the conflict has been resolved.	The pension provider or scheme leaves the existing Pel registration linked to citizen user 1 and generates a new assetGuid and creates a new registered Pel resource using the new assetGuid with match status “match-yes” linking this to citizen user 2. In this scenario, it is the pension provider or scheme’s responsibility to determine whether to return view-data with a “CONT” status (data item 2.004 set to true) and limited data for the full matches until the conflict has been resolved.

## Pensions Identifier (Pel) status and transition

**118.** After matching, Pels are registered with a match\_status of either match-possible or match-yes.

**119.** Subsequently the match status may need to be updated, for example from match-possible to match-yes, or the match may need to be removed.

**120.** When matches are removed, the deletion reason denotes the reason for removal.

121. The permitted states and transition paths are documented below:

