

Technical standards

Draft version: November 2022

Contents

Introduction	5
Executive summary.....	5
Background	5
Purpose.....	5
Audience	6
Jurisdiction.....	6
Other guidance	6
Use/evidence	6
Version	6
Technical overview	7
Client registration.....	7
Pensions dashboards ecosystem primary components.....	8
Sequence flows.....	10
Initial find.....	10
Internal register obtain Pels-URL	12
Pull Pels	14
Provider registers Pels	16
Identity process	19
View.....	21
Refresh process	24
API technical standards.....	27
Scope	27
Transaction monitoring.....	27
Third party standards	27
Find API.....	28
Summary of the find API	28
Pension finder service (PFS)	28
Hosting.....	28

Format	28
Authorization	28
HTTP method	28
Response.....	30
Error handling	30
<i>Obtain PAT API</i>	<i>32</i>
Summary of the obtain PAT API	32
Hosting.....	32
Format	32
Authorization	32
HTTP method	32
Response.....	33
Error handling	34
<i>Register Pel API</i>	<i>35</i>
Summary of the register Pel API.....	35
Hosting.....	36
Format	36
Authorization	36
HTTP method	36
Resource description	36
Create resource description	38
Read resource description.....	40
Update resource description	41
Delete resource description	42
List resource description	43
Error handling	44
<i>View API.....</i>	<i>45</i>
Summary of view API.....	45
Hosting.....	45
Format	45

Authorization	45
HTTP method	46
Response.....	47
Error handling	49
<i>Introspect API.....</i>	<i>51</i>
Summary of introspect API	51
Hosting.....	51
Format	51
Authorization	51
HTTP method	51
Response.....	52
Error handling	54
<i>Permission API.....</i>	<i>54</i>
Summary of permission API	54
Hosting.....	55
Format	55
Authorization	55
HTTP method	55
Response.....	56
Error handling	59
<i>PAT refresh API.....</i>	<i>59</i>
Summary of PAT refresh API	59
Hosting.....	59
Format	59
Authorization	59
HTTP method	60
Response.....	60
Error handling	61
<i>Authorise API.....</i>	<i>62</i>
Summary of authorise API	62

Hosting.....	62
Format	62
Authorization	62
HTTP method	62
Response.....	64
Error handling	65
Obtain Pels API.....	66
Summary of obtain Pels API	66
Hosting.....	66
Format	66
Authorization	66
HTTP method	66
Response.....	67
Error handling	68
Obtain Pel configuration API.....	68
Summary of obtain Pel configuration API	68
Hosting.....	69
Format	69
Authorization	69
HTTP method	69
Response.....	69
Error handling	70
Technical standards	71
Dashboard redirection protocols.....	71
JWT signing and verification.....	72
Pension identifier format	72
GUID creation protocols.....	73
Pension providers and schemes (UMA resource servers).....	73
Appendix	75
Glossary	75

Introduction

Executive summary

1. This document outlines the technical standards for pensions dashboards that need to be adopted by:
 - qualifying pensions dashboard services (QPDS)
 - pension providers (trustees and manager of occupational pension schemes as well as the managers of stakeholder and personal pension schemes), connected to, or are required to connected to, the pensions dashboards ecosystem

Background

2. Pensions dashboards are apps, website or other tools which will help individuals view their pensions information online. They will bring together an individual's pensions they haven't taken yet, including their State Pension as well as any occupational and personal pensions (including those with an insurer), to support better planning for retirement and growing financial wellbeing.
3. This standard is issued by the Money and Pensions Service (MaPS). MaPS set up the Pensions Dashboards Programme (PDP) in 2019 to design and create the pensions dashboards ecosystem and the supporting governance framework. The pension dashboards ecosystem contains the central digital architecture (CDA) that will make pensions dashboards work. It will connect millions of individuals to their information on thousands of pensions, via multiple pensions dashboards. For more information about the pensions dashboards ecosystem and its components, see <https://www.pensionsdashboardsprogramme.org.uk/ecosystem/>. MaPS is also responsible for operating its own pensions dashboard.
4. Standards are separate from, but designed to complement, the Financial Conduct Authority's (FCA) regulatory framework for pension dashboard service firms. Firms which operate a QPDS must need to be (or become) FCA authorised, get the regulatory permission to undertake this new regulated activity and meet any Handbook rules and guidance that the FCA may introduce for firms undertaking this activity.

Purpose

5. This standard covers the dashboard technical requirements on pensions providers, schemes and QPDS to connect to the pension dashboard ecosystem. It will help developers of dashboard connectivity solutions build a common set of tools to:
 - receive data from the pension finder service or dashboards
 - reply to dashboards with the appropriate data
 - comply with the security requirements
 - facilitate the central identity and authorisation requirements

Audience

6. This standard applies to pensions providers, schemes and QPDS. The term dashboard is used interchangeably with QPDS as this may also include the MaPS dashboard even though this standard does not apply to the MaPS dashboard. The MaPS will be adopting it to ensure technical interoperability with the pensions dashboard ecosystem.
7. Third parties (such as administrators or software providers) may apply our standards (and guidance) on behalf of their pension provider, scheme or QPDS clients. As the standards and guidance apply to the pension provider, scheme or the QPDS, they remain responsible for compliance with them, even if implementation is delegated to a contracted third party.

Jurisdiction

8. This standard applies to all United Kingdom pension providers, schemes and QPDS subject to the dashboard duties in the Pensions Dashboard Regulations 2022 (the Regulations) and FCA regulatory framework.

Other guidance

9. This standard should be read in conjunction with our data standards, design standards and the code of connection¹.

Use/evidence

10. Standards are mandatory requirements and, therefore, compliance by pension providers, schemes and QPDS's is compulsory.
11. Standards and guidance may be admitted in any proceedings relevant to pension providers, schemes and QPDS compliance with their dashboard duties, this also applies to the obligations owed by any other party (for example, a sponsoring employer or administrator). It will be the decision of the body hearing the proceedings (including any regulatory proceedings conducted by the FCA or the Pensions Regulator (TPR)) to assess the evidential weight to be attached to any standard or guidance admitted.

Version

12. This is the November 2022 version of the technical standards. Please refer to the [changelog](#) for updates since the last publication. The following core areas are covered in the document:
 - **part 1:** technical overview: sections 2 to 3
 - **part 2:**

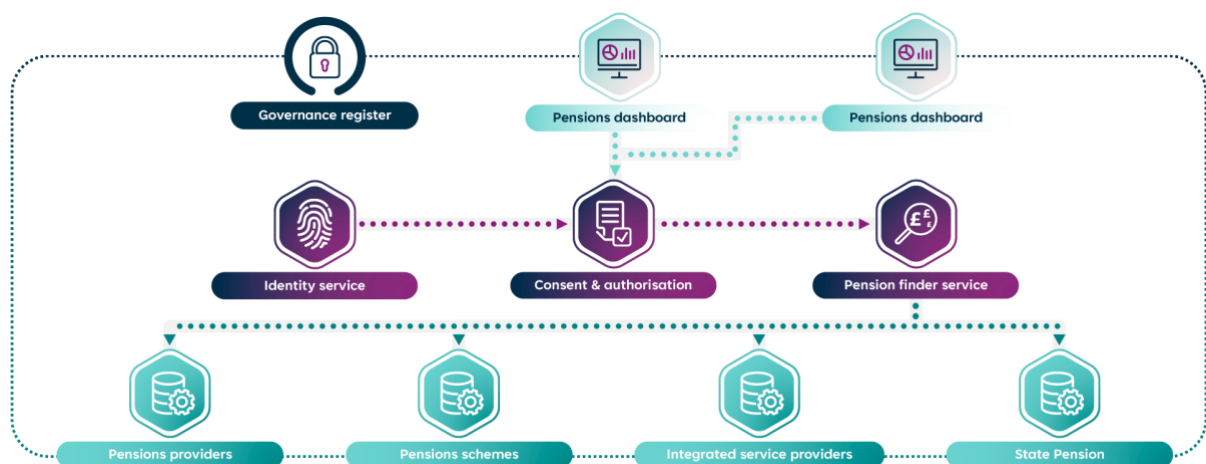
¹ <https://www.pensionsdashboardsprogramme.org.uk/standards/>

- API standards: sections 4 to 13
 - technical standards: section 16
 - **glossary:** section 17
13. Part one provides a technical overview of how the ecosystem operates. This is essential background to be able to understand the mandatory API and other technical standards detailed in part two.

Technical overview

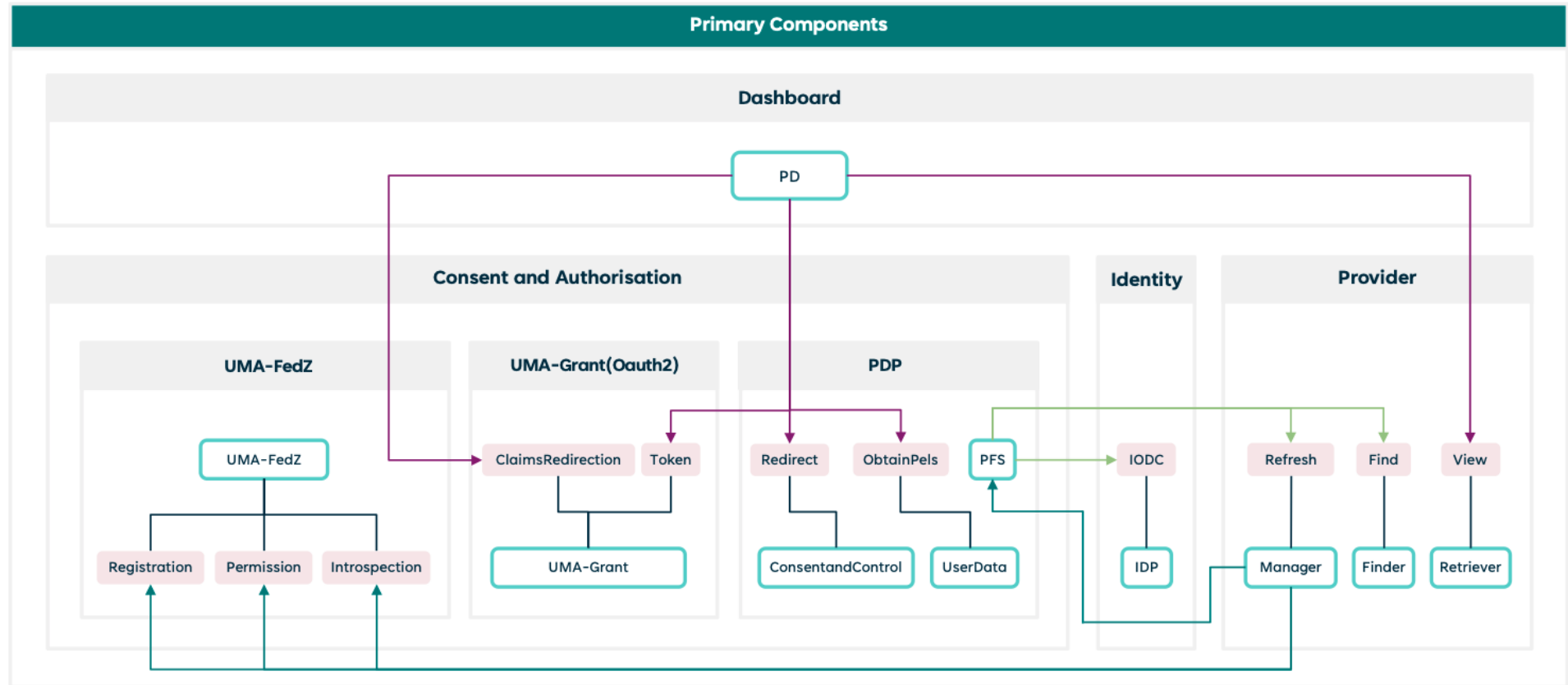
Client registration

14. Registration of OAuth clients (ie QPDS and pension providers and schemes) is required for the operation of the 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.
15. Static registration may be more secure but dynamic registration may be expected by industry participants, may involve fewer manual processes, and is necessary to support public client types (especially non-confidential ie SPAs, native apps).
16. In order for dynamic client registration to be effective it requires the AS to support dynamic registration [ODynClient] and to provision clients appropriately. The AS must support software statements to bootstrap the registration process securely.
17. Client authentication to the AS will be defined in accord with dynamic registration requirements.
18. Currently static client registration is being used. In future, dynamic client registration will be provided and this document updated accordingly.
19. **Overview of the pensions dashboards ecosystem primary components primary components**
This diagram shows each component involved in the protocol interactions in this document.



Pensions dashboards ecosystem primary components

20. Components are presented in packages and each package corresponds to a participant in the ecosystem: dashboard, consent and authorization (C&A) service, pension providers and schemes, identity service. Within a package the components expose interfaces (blue lines) and call interfaces of other components (lines of other colours dependent upon the caller), eg the pension provider and scheme interfaces are called by dashboards (red) and by C&A components (green).
21. Calls within packages are not shown on this diagram eg PDP.ConsentandControl – handling redirections – deeply interacts with UserData and the UMA authorization components eg PDP.ConsentandControl interacts with PFS when it initiates a search, passing consents and search parameters and authorization Server grants.
22. Dashboards expose no interfaces (other than their redirection endpoint which is used to unwind a previous redirection to the relevant consent and authorisation redirection interface).
23. PDs must redirect their user agents to either of C&A's interfaces – UMAGrant.ClaimsRedirection or ConsentandControl.Redirect. The ClaimsRedirection interface is part of the UMA authorization flow. The Consent.Redirect interface handles requests of type 'find', 'consent' and 'refresh'.



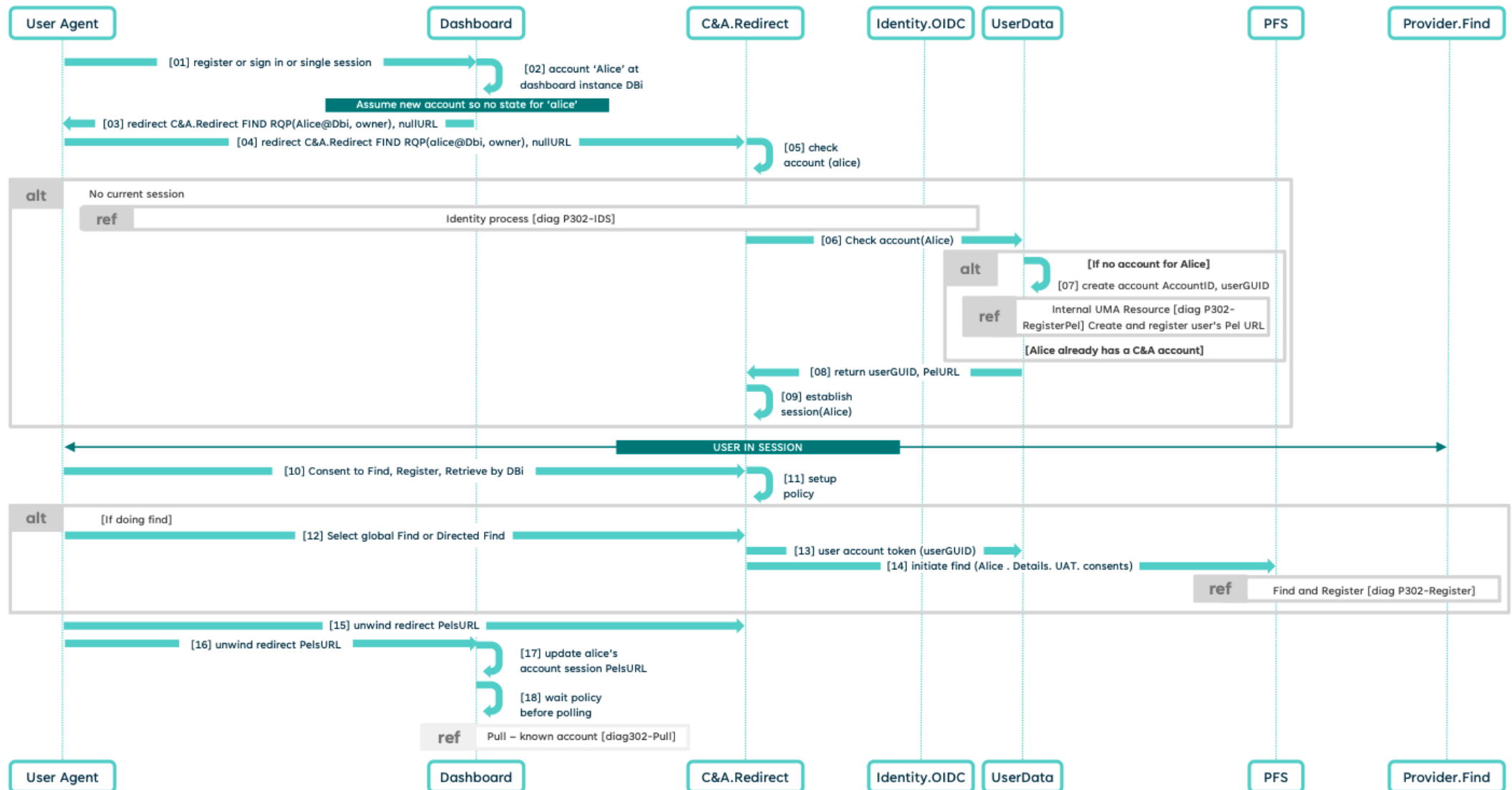
24. The authorization server is not explicitly shown on the diagram: it is part of the consent and authorisation package and comprised of the UMA components and *some* of the functions of consent and control and user data.
25. The **two UMA2 interfaces** can be seen: federated authorization is called by a pension provider or scheme, and UMA Grant by the dashboard.
26. Note that since an UMA authorization server is a specialisation of an OAuth2 authorization server, the Token interface, part of UMA-Grant(OAuth2), is *also* the same endpoint called by the pension provider or scheme when it needs to obtain a PAT (protection API access token), which is an OAuth2 access token with scope `uma_protection`, subsequently used against the UMA federated authorization protection API (UMAFedz).
27. Apart from UMA support itself, the ecosystem package shows the **other primary components of the C&A service**. The user facing consent services manage user interaction and policy. User data contains the policy and registered Pels for each user; it exposes the interface for dashboards to obtain Pels for their users. The consent and control component also manages user and authorization processes and repositories (user data , selectable names for user directed find, etc). PFS performs search orchestration after the consent and control component has established the context, which includes proving user identity using the identity service, establishing user data, and other tasks as necessary.
28. The **identity service** interface (IDP.OIDC) is simplified in this diagram (it is also a specialisation of the standard OAuth2 authorization service endpoints).

Sequence flows

Initial find

29. This section presents a flow for a new user at a dashboard, initiating a find activity at the C&A.
30. In this case the dashboard knows it has a new user (unless it is a stateless dashboard when it *assumes* it has a new user) and redirects to the C&A to process a find operation. The condition around step 12ff is discussed in a later section on obtaining Pels URL from dashboards. In the simple 'new find' variant, these find steps must be performed, specifically the PFS will be invoked, step 14, to orchestrate calls to several/all pension providers and schemes.

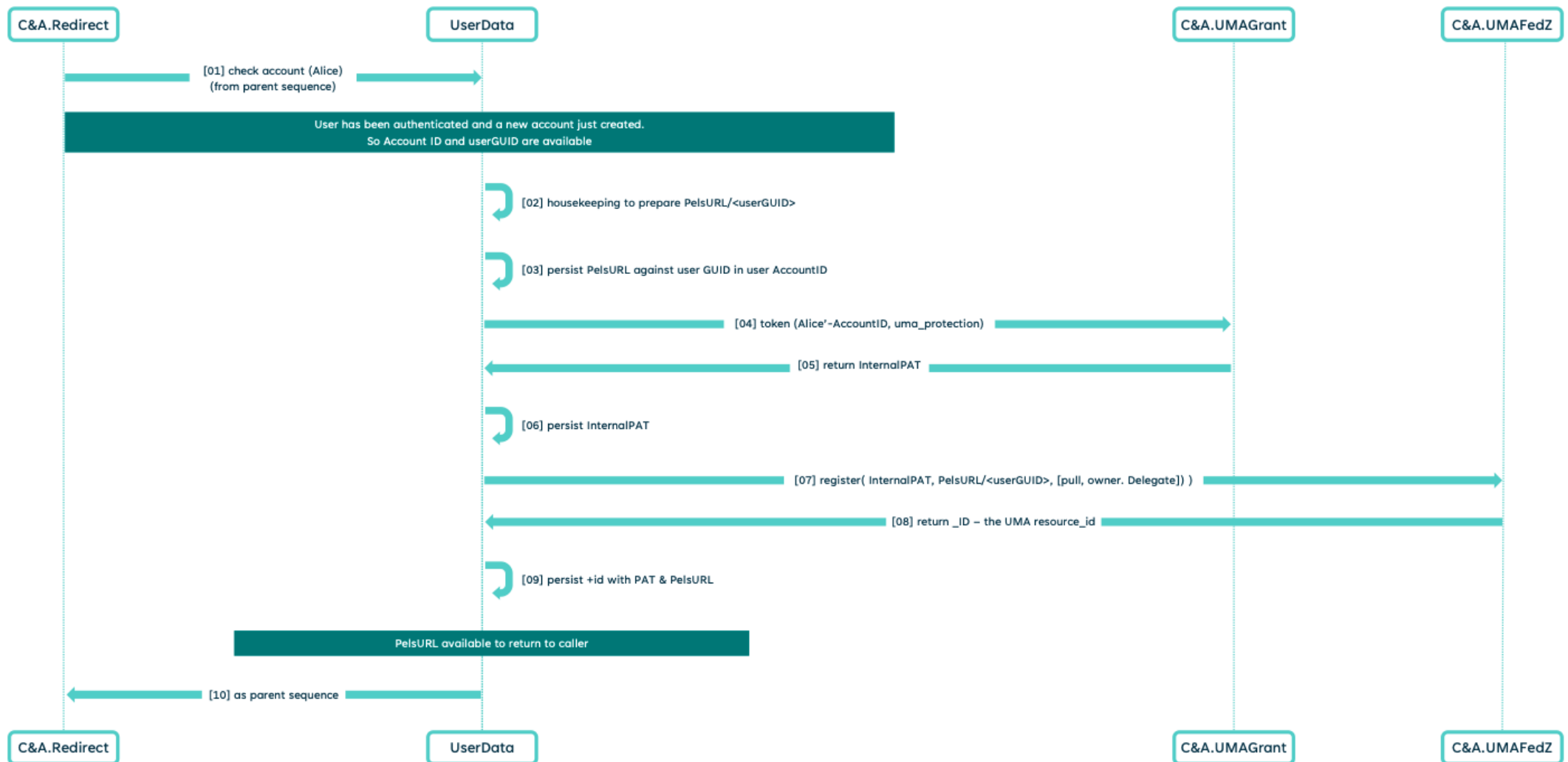
Initial find



Internal register obtain Pels-URL

31. This sequence is nested in the above flow. After creating a C&A account at step 07 this sequence registers the protected resource for the owner user's PullPels resource. In this case both the UMA authorization server and the UMA Resource Server which presents the PullPels resources are in the *same security domain*, ie are part of the C&A service. Accordingly, there is the possibility of significant optimisation (customisation) of the relationship between the AS and the PullPels resource server, yet to still offer the same external UMA2 grant based access to the protected resource by dashboards.
32. The following takes an 'UMA2 Federated authorization' approach.
33. In step 04 the parameter Alice'-AccountID is Alice's identity token, available from the parent sequence, along with the new derived AccountID, from which a token can be constructed in the trusted UserData RS. This token is used as an authorization grant for the authorization server to issue an internal PAT. In implementation, this could use the OAuth2 JWT-bearer grant. Since the AS and the RS (the UserData component) are coupled and in the same domain, the AS.

Obtain Pels URL

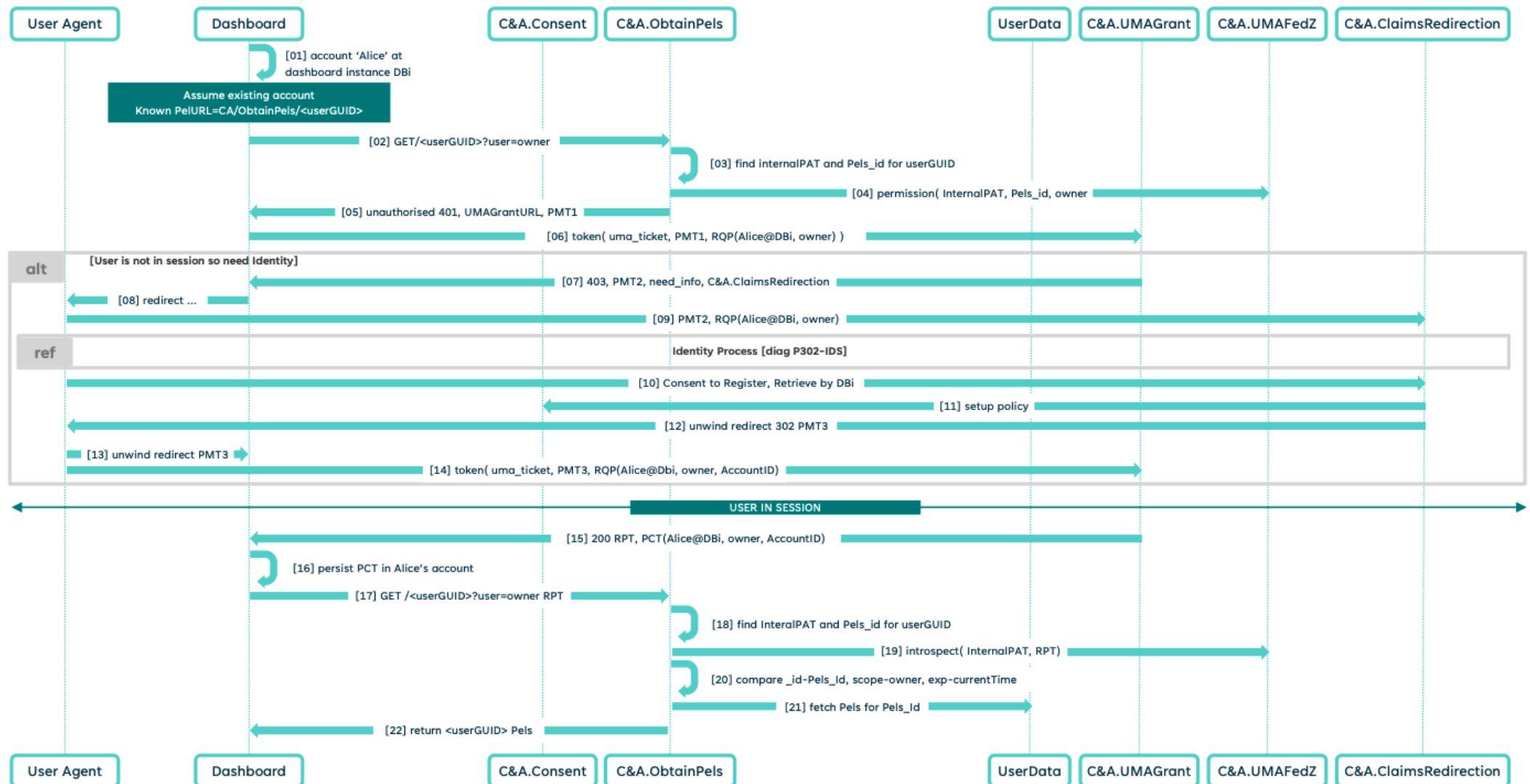


could check that Alice's account (AccountID) exists (not shown on the diagram) or simply trust the internal caller and signature, and issue the requested access token (the InternalPAT) at step 05.

Pull Pels

34. Once the dashboard knows the user's C&A account in the form of the relevant protected Pels resource, it can pull the Pels as follows. The dashboard attempts to GET the URL `https://CA.ObtainPels/<userGUID>` with a parameter of the user is 'owner'. Initially, this will fail because the dashboard does not quote an appropriate access token, but it will trigger the UMA authorization dance which obtains the RPT which then succeeds in the subsequent call.
35. This flow is not limited to being a direct successor to the initial find flow above, it also stands alone and can be performed whenever necessary:
 - immediately after find new
 - for up to the defined SLA after an initial find as the (slower) pension providers and schemes find pension records
 - after repeat finds (user-controlled timing, user directed finds)
 - when a delegate (financial adviser or MaPS guider) is initialising their records for the client in which they have the client-specific PelsURL in their communication from the C&A server
 - when an owner-user initialises a new dashboard by importing Pel data from a previous one, the UMA Pels resource would bootstrap the process of initialising the new dashboard
 - new pensions dashboard attempting to obtain the Pels would initiate the whole of this process, (including new identity assertion, account update, new consent and policy for the new dashboard)

Pulls Pels



Provider registers Pels

36. This flow is referenced from initial find flow above.
37. The PFS's only role is to orchestrate the calls, step 01 in the diagram below, the receipt of each 'find' call should be acknowledged using low level http 202 code; note this has nothing to do with whether the find actually results in the location of a pension.
38. When a pension is found it is registered at the C&A service by the pension provider or schemes using the UMA Protection API and an access token issued for the purpose, a PAT.
39. If a pension is located (and consents permit) the provider should register the corresponding Pel with the C&A server as per the flow (steps 02ff).

Register Pels

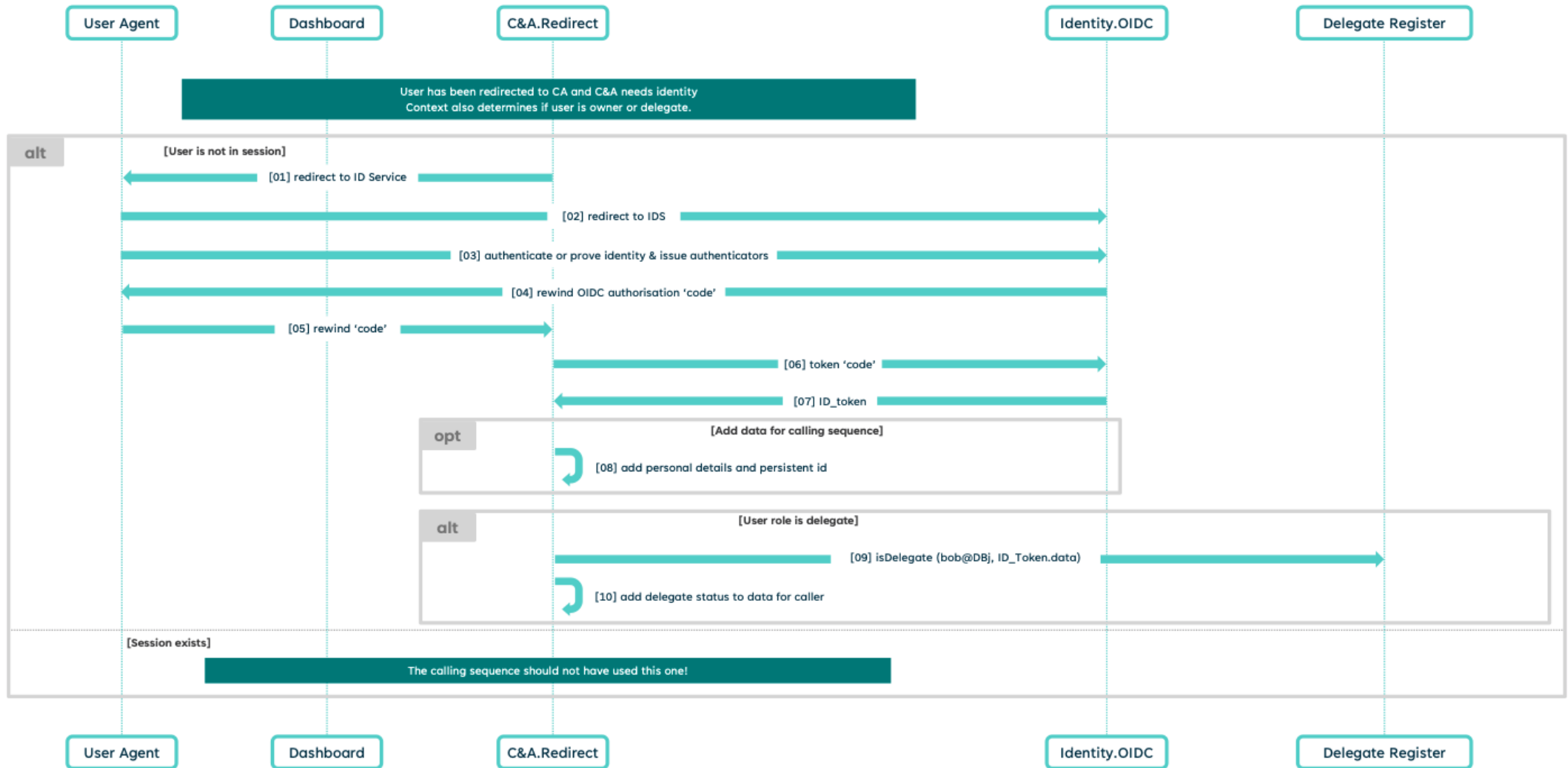


40. The case where the find failed has been mentioned above. However, in cases where the find succeeded (so steps 02-10 apply) a pension provider or scheme must arrange for **step 02 ‘get owner’s details’** to function correctly, for view and for find, whether or not the same user has executed a find operation in the past. The pensions dashboards ecosystem requires that these details are:
- any identifier the pensions provider or scheme wants to use for the owner’s account
 - any internal keys or similar which enable the pension provider or scheme to locate relevant internal records associated with the owner and their assets
 - the user’s authorization server (Of course in the PD ecosystem at least initially this must be the one C&A Server, but theoretically this might change over time if users appoint other ‘open finance’ components, so the actual C&A AS URL should be kept in the RS from the outset; in any case it is required to be returned to the dashboard in the UMA protocol.)
 - PAT when one exists
 - associated Pels and their descriptions, including a map of inbound URLs to the relevant Pel
 - UMA _id for each Pel
41. Pension providers and schemes must consider how to retrieve this data in two cases on the basis of the inbound View URL, this is a map of URL->Pel->_id and from that to the owner & PAT
42. **Step 03** shows the use of a temporary credential as an **OAuth2 grant** of type JWT-bearer, as described in section 3.5.
43. **Step 06 ‘generate or retrieve Pel’** also indicates a decision for providers. A Pel is a unique URI (not URL) identifying a ‘asset’ / ‘pot’ / ‘pension’ uniquely and persistently. It is *independent* of the owner of the asset. Thus it can be allocated *at any time* from the creation of the asset until its initial registration with the PD C&A service.

Identity process

44. Whenever the C&A service needs to determine a user's identity it uses the identity service, which is (assumed to be) a federation of external identity providers, interfaced via some form of hub. The purpose of this section is to present how this service integrates with the C&A.
45. There are two key use cases, dependent upon the 'role' of user. The user may be a pension *owner* or the user may be a *delegate* of one or more owners.

IDS



46. The implementation of the standard OIDC identity service will comprise of an authorization endpoint (handling redirection of the user agent to so the user can authenticate and prove their identity) and a token endpoint from which the identity assertion (ID Token) will be retrieved, using OIDC authorization code flow. OIDC Core also defines a 'user_info' endpoint to obtain additional claims, in addition to those which may be in the ID Token. In addition, it is likely that the OIDC interface must be brokered via a hub component (not shown in this document) so that a federation of identity providers can be supported.
47. The context of this sequence (eg initial find above) must determine details of the information required from the identity process. For example, the call must determine whether the user is a delegate on a delegate dashboard (eg bob@DBj) or an owner user. The sequence should only be called when the caller has checked there is no current session for the owner user. After this flow the caller may also create the C&A account for the owner user and establish the required session as appropriate, using the temporary data noted in the flow.
48. The dashboard may have already authenticated the user using the ecosystem identity service (see p301 6.1 for discussion). The fact that the user is authenticated by an IDP may be added to the redirect parameters (probably adding an element to the RQP so an RQP is comprised of: alice@DBi, owner, and idp=<IDP>). Given that the Pensions Dashboards Programme mandate that the IDPs in its federation must maintain an open session (for a controlled period) then the redirection to the identity service in the above flow can immediately return the OIDC grant code without requiring the user to reauthenticate at the open ID provider (IDP).

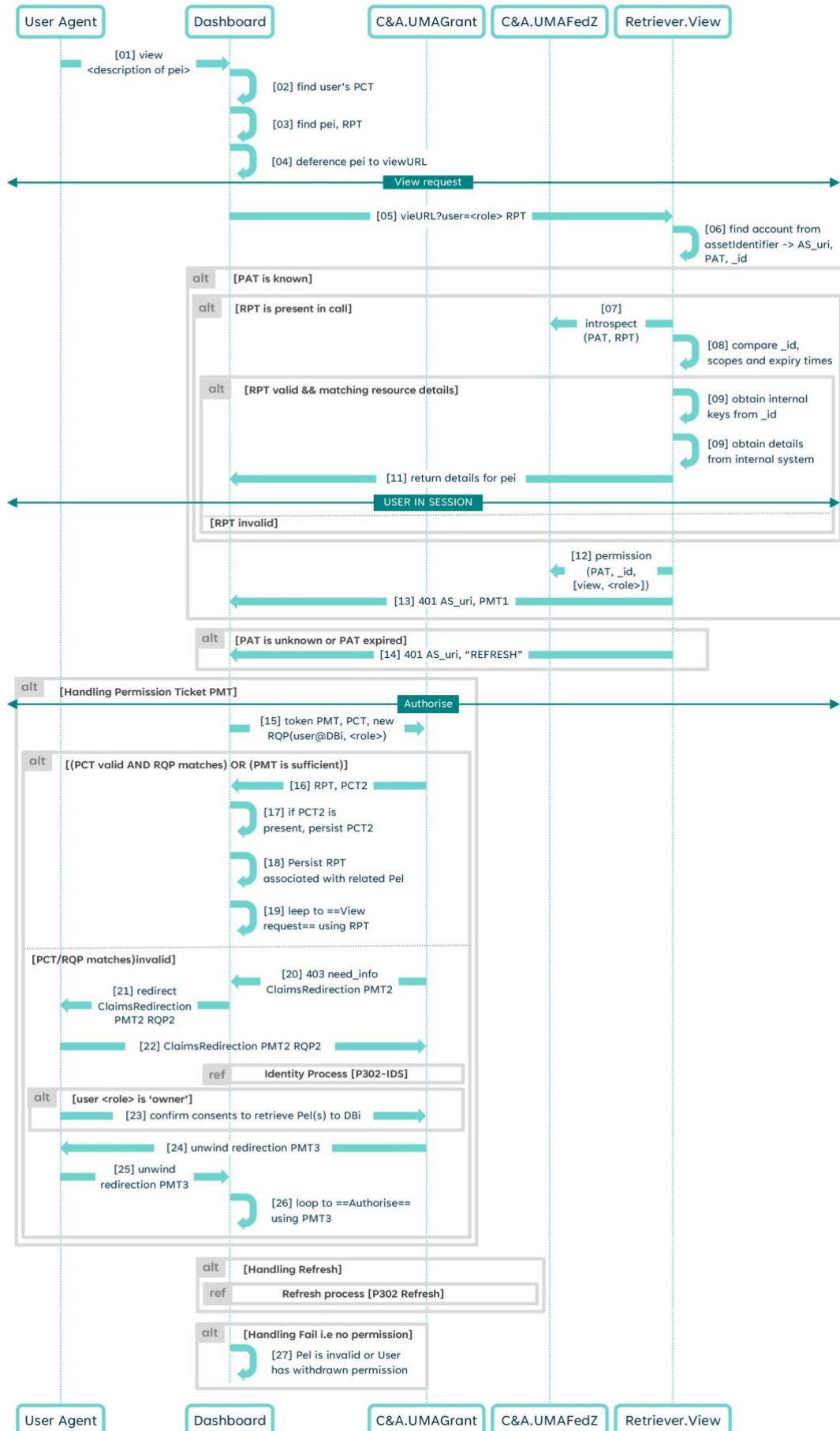
View

49. When a dashboard has obtained Pels for its user, it must usually seek to view the pension details, by GETting the *dereferenced* Pel. This action will initially fail, as there is no access token, initiating the UMA2 grant authorization dance.
50. **Dereferencing** a Pel means taking the Pel which is a URI (format 'pei':<holdername GUID>:<asset GUID> e.g. "pei: 85624cd0-de31-455c-8ce8-f26515a29577: 9112820e-7d9d-47e0-bbfe-129b8afbab0d") and breaking it into components, looking up the holdername, e.g. '85624cd0-de31-455c-8ce8-f26515a29577' in a configuration table to derive a host name, eg 'examplepensionco99.dashboard/pei', and composing a URL eg <https://examplepensionco99.dashboard/pei/85624cd0-de31-455c-8ce8-f26515a29577> . It is this URL which is an UMA protected resource; the URL of the pension details to be accessed by an http GET with appended role parameter, eg '?user=owner'.
51. The governance register API will map the <holdername> element of the Pel to the current URL of the pension provider or scheme's view endpoint (example above). A cache entry might be deleted if a resulting hostname is not resolved or periodically based on our standards.
52. **Dashboard preparation** a user may select one or more descriptions of their pensions to view (step 01). The dashboard finds related control information whether this be from a persistent store (for dashboards which have accounts & appropriate security controls) or from its current

session. For each user description of an asset there must be a PeI and may be an existing access token, RPT. If the user has been authenticated to the identity service recently from that dashboard there may be a PCT available from that user's account at the dashboard. The PeI is dereferenced as discussed above, deriving the protected resource for the asset.

53. **Authorised view access** the 'happy path' in which the pensions provider or scheme locates the UMA control information based on analysis of the URL (step06) and there is a valid access token (07) which matches the user's role (owner or delegate) and other information (08), results in the API returning the pension details associated with the PeI to the dashboard.
54. There are three types of **authorization failure**:
 - those for which the AS can provide an UMA permission token (PMT) to the RS which is returned to the dashboard
 - the case in which the PAT has expired, so the UMA AS cannot respond to the pension provider or scheme's RS request for a permission ticket. This is handled by a PDP-specific error message requesting 'REFRESH' processing (discussed in a later section)
 - final failure in which the RS or AS does not return a PMT nor a specific instruction, in this case the dashboard must assume that the PeI is invalid or that the user has withdrawn consent for that dashboard (despite having been directed to the C&A as part of the above automatic handling processes)

View

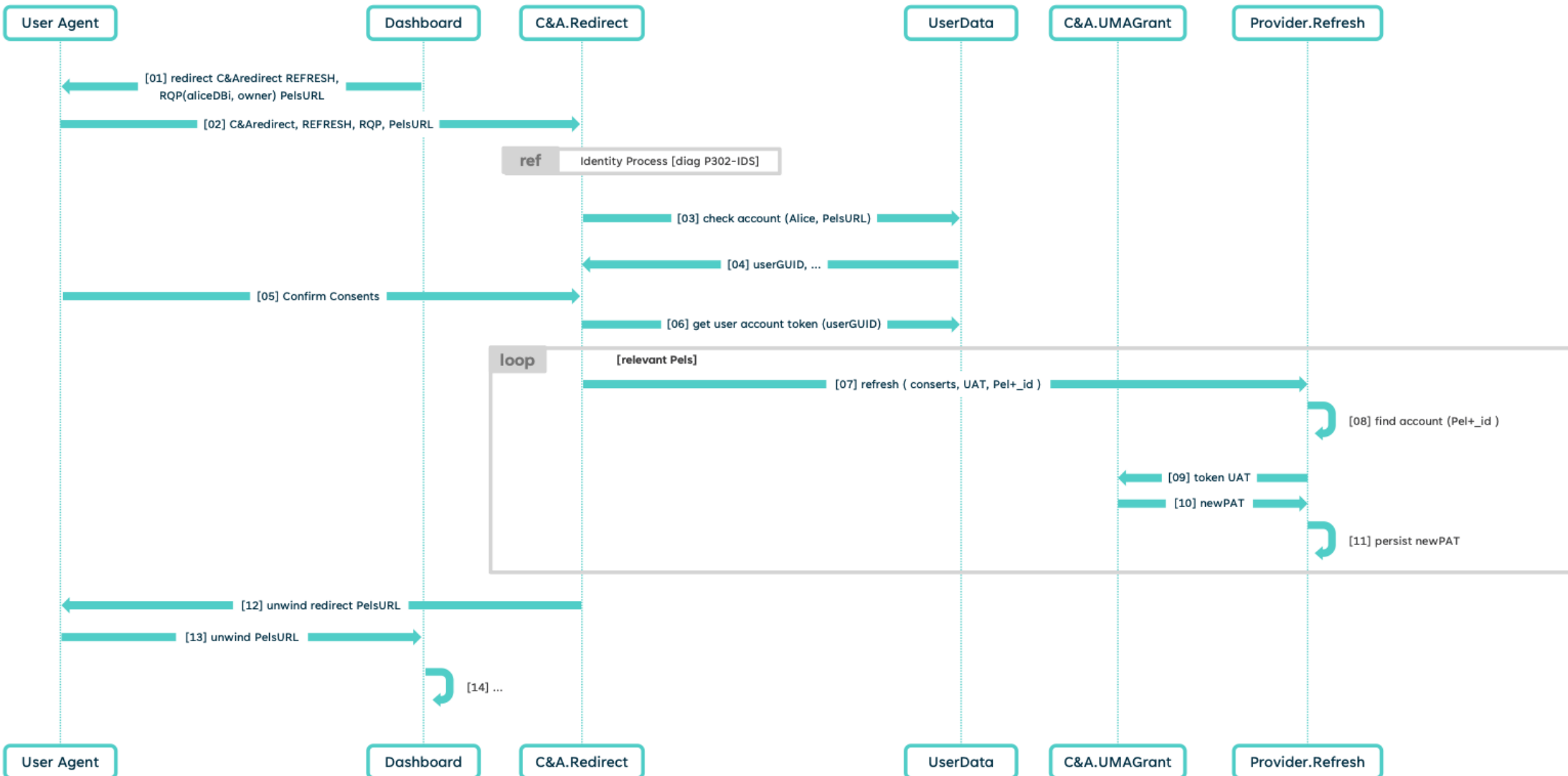


55. The **authorization process** (step 15) uses the current PMT (always present), and current PCT (if there is one) and a new RQP token, minted by the dashboard for every relevant call to the C&A service. The RQP (requesting party token) is an assertion by the dashboard that a specific user known to it (eg 'alice') is present at the interface of a specific dashboard instance (e.g. DBi), and that the user is acting in a specific role (i.e. owner or delegate).
56. The AS makes the authorization determination based on the state (contained in the PMT at step 15 and in the AS) and the other parameters. If the authorization request is successful, the AS returns an access token for the specific protected resource (RPT) and it may issue a new or replacement PCT. The PCT (persistent claims token) binds the user at the dashboard to the C&A's view of the identity (derived from the external identity service) and the user's validated role. When an authorization request is successful the dashboard should retry the view request.
57. If the AS determines that it cannot grant access, but that it can continue the authorization process, it issues another PMT and requests that the client (dashboard) redirects the user to its claims endpoint for an interactive session with the user. There the appropriate identity uplift occurs and if the user is an 'owner' the appropriate consents are refreshed. (Details of interaction between the claims redirection code and the consent and UserData components are omitted here (but see also the sequences Initial find and pull Pels above.) The AS also needs to consider whether the policy of the owner for a specific PeI and specific delegates matches the recently uplifted user. If identity uplift was successful and policy matches PeI and person, the AS issues another PMT and unwinds the redirection back to the dashboard. The dashboard uses the new PMT to re-attempt the authorization call.

Refresh process

58. Only owner-users can perform refresh (not delegates), the same is true of the other redirect purposes – find and consent.
59. Refresh processing may be entered as a result of failure of a view attempt in which a PAT has expired, or in *routine* use of the C&A service by the user (eg in adding dashboards, changing consents or stepping up identity after PCT expiry after 90 days eg user directed find or repeat find).
60. The owner-user is redirected to the C&A service so that consents can be confirmed or modified and, if necessary, the identity can be uplifted. If the user is in session at the C&A then there may be no reason to perform a reauthentication. (This is not shown on the diagram but simply skips the identity process.)
61. The flow shows the account details being checked, including the dashboard-provided user's PelsURL. There are error conditions related to dashboard and user which can be corrected as part of this process (not shown) eg the user's correct PelsURL is returned as described in the section 'Initial find' and the RQP can be used to check that the user has consent policy for that dashboard.

Refresh



62. Irrespective of the dashboard's stated purpose for the redirection (REFRESH, repeat FIND or CONSENT), or the purpose of the redirection to the claims redirection endpoint during authentication (to seek periodic step-up), the C&A has the *opportunity* to run the loop to refresh PATs at the relevant providers of the Pels of the user. Clearly if the redirection from a dashboard was for REFRESH, this implies that the dashboard had received an error from a provider during an attempted view, but even in this case the C&A needs to manage the process appropriately: for example if a user has withdrawn consent for any dashboard to view a Pel, or for registrations of Pels from that provider, there is no need to attempt the refresh.
63. Note that, unlike the initial find process in which the owner-user's personal details are sent to perform a search at the pension provider or scheme, there is no need for these biographic details in performing a refresh. All that is needed are the parameters shown at step 07. The 'relevant Pels' are those which a) are registered for the user's account, b) the RS hosting one or more of these Pels is not subject to a current (user directed, repeat) find² for the same user, c) the RS has not already been contacted³ in the same refresh loop. For each relevant Pel, which has the relevant resource _id, the C&A can determine the provider RS and hence which provider's refresh endpoint to call.
64. The above flow assumes that the user does approve refreshing the PAT and that the dashboard which initiated the refresh is still consented to receive the owner's Pels and other data. If not the return redirection (steps 13, 14) would not carry the current PelsURL and may have a specific error code so the dashboard knows the user has withdrawn consent.

² Because if a find is happening for that user for the same RS, the RS will have a separate opportunity to refresh the PAT.

³ Because only one PAT is needed for each user at an RS (not per Pel if the same user has more than one Pel).

API technical standards

Scope

65. The following areas is covered in the within API technical standards section:

- a summary of each API
- hosting
- format
- HTTP method
- authorization
- expected responses
- error handling

Transaction monitoring

66. All interfaces will carry a unique transaction identifier GUID (request_id) for logging, audit and monitoring purposes.
67. The transaction identifier is issued by the party which initiates the transaction. Both parties to the transaction must retain the same transaction identifier in their respective audit logs.
68. For transactions which PDP initiates, PDP will generate the identifier; for transactions which the pension provider, scheme or QPDS initiates, it must generate the identifier.
69. The transaction ID must be transmitted via the HTTP header: X-Request-ID.

Third party standards

70. These standards are built upon a number of third party standards:

Standard	Scope	Issuing authority	Contact details	Version
UMA grant 2.0		Kantara Initiative	https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html	2.0
UMA federated authorization 2.0		Kantara Initiative	https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html	2.0

Find API

Summary of the find API

71. The find API is exposed by pension providers and schemes and its purpose is to receive the find requests which are initiated by the pension finder service (PFS) containing the pension owner's PII data needed for a pension provider or scheme to use to determine a match within the internal records. The find input data will also carry the relevant consents given by the pension owner during the find process and the user account token needed for obtaining the PAT (protected API token) needed for the pension providers and schemes to use the UMA specific protection API to register Pels upon a successful find with the consent and authorization service.

Pension finder service (PFS)

72. The pension finder service is orchestration middleware, it distributes the find request across the pension provider or scheme endpoints by invoking their find APIs and manages the low-level interactions to achieve message delivery to the pension provider or scheme.

Hosting

73. Each pension provider or scheme connected to the ecosystem must host their find API within their domain.

Format

74. The find API must be a REST API using JSON encoded as UTF-8.

Authorization

75. This is a closed ecosystem with all endpoint connections secured using private PKI certificates issued by the governance register to suitably enrolled organisations. This enables connecting entities to establish a mutual TLS connection with the central infrastructure. For find requests only the PFS will be responsible for invoking the find API endpoints. Currently there are no additional API security requirements for the find API.

HTTP method

76. The PFS will be restricted to only make HTTP POST requests to each pension provider or scheme find endpoint with the body of the request containing the find parameters as a signed JWTs.
77. Summary of the find request data parameters sent to pension provider or scheme find endpoint:

user_token	<p>a combination of the following expressed as a JWT:</p> <ul style="list-style-type: none"> • verified identity details such as name, date of birth and postcode, from the verified details supplied by the identity service • identity details asserted by the user at the C&A's consent user interface eg National Insurance number
user_account_token	the user account token is an OAuth2 authorization grant, expressed as a JWT (JSON web token) which can be exchanged for the PAT
consents_token	a set of user consents for subsequent processing using the supplied identity details expressed as a JWT

78. For example, the PFS makes the following HTTP POST request using mTLS:

POST /find HTTP/1.1

Host: www.dashboard.examplepensionco.com

Content-Type: application/json;charset=UTF-8

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

```
{
  "user_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQsw5c",
  "user_account_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQsw5c",
  "consents_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQsw5c"
}
```

Response

79. If the find request sent from the PFS succeeds, the pension provider or scheme (UMA resource server) must respond with a *202 Accepted HTTP* status implying acknowledgment of the find request. Following this the PFS will not be involved in any request back from the pension provider or scheme.
80. HTTP/1.1 202 Accepted
81. If a match is found then the pension provider or scheme must issue a PeI in the set format alongside it's description and register this directly to the consent & authorization service via the UMA protection API. If no match is found then there is no further action required by the pension provider or scheme.

Error handling

82. If the find request sent from the PFS fails then the pension provider or scheme is expected to respond with the appropriate HTTP status code and error message. The body of the response for an error must contain an error code which correlates to a message so that the test harnesses and logs can be automated.
83. 4xx

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a POST request, the server must include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition.

Status	Message	Note
400	bad request	<p>potential bad requests examples:</p> <ul style="list-style-type: none"> • PFS is not sending a HTTP POST • parsing error by the pension provider or scheme • schema not configured correctly

84. 5xx

The 5xx (Server Error) class of status code indicates that an exception occurred during the elaboration of a request. An indication about the nature of the error must be included together with an indication if the error is temporary or permanent.

Status	Message	Note

500	internal server error	
503	service unavailable	this status is temporary used when the service is down for maintenance or is overloaded by requests

Obtain PAT API

Summary of the obtain PAT API

- 85. During find the PFS includes a valid user account token issued by the authorization server (part of the C&A) in the find request sent out to the pension provider or scheme find interface endpoints. The user account token is a OAuth2 authorization grant, expressed as a JWT (JSON web token) which can be exchanged for the PAT, which is an OAuth2 access token, as per the OAuth2 standard by the presentation of this token to the authorization server's token endpoint. Pension providers and schemes must be a client of the authorization server.
- 86. The obtain PAT API (i.e. the authorization server's OAuth2 token endpoint) must be exposed by the authorization server and must follow a standard implementation of <https://datatracker.ietf.org/doc/html/rfc6749#section-4.5> using urn:ietf:params:oauth:grant-type:jwt-bearer in accord with <https://datatracker.ietf.org/doc/html/rfc7523#section-2.1>

Hosting

- 87. The API must be hosted on the authorization server (AS) as the request is made is to the authorization server's token endpoint.

Format

- 88. The obtain PAT API must be a REST API with character encoding of UTF-8 in the HTTP request entity-body.

Authorization

- 89. The pension provider or scheme must be a client of the authorization server and must register it's software with it before a request can be made. This ensures the communication channel is encrypted and establishes dynamic trust between both parties.

HTTP method

- 90. The pension provider or scheme must use the HTTP "POST" method when making access token.requests to the authorization server's token endpoint, by sending the following parameters using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the HTTP request entity-body:

grant_type

REQUIRED. Value must be set to " urn:ietf:params:oauth:grant-

type:jwt-bearer".

assertion

REQUIRED. must contain a single JWT

scope

REQUIRED. Specifies the scope of the access request. must be uma_protection

91. For example, the pension provider or scheme makes the following HTTP POST request using mTLS:

POST /token HTTP/1.1

Host: as.pdp.com

Content-type: application/x-www-form-urlencoded;charset=UTF-8

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer&

assertion=PHNhbWxwOl...[omitted for brevity]...ZT4&scope= uma_protection

Response

92. If the request for an access token is valid, the authorization server generates an access token (PAT) as a structured JWT bound (in accordance with [Section 3 RFC8705](#)) to the resource server (pension provider or scheme certificate).
93. For example, a successful token response may look like the following with all the properties of the PAT token encoded in the payload:

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

{

“access_token”:“eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c”,

“token_type”:“pension_dashboard_pat”

```
}
```

Error handling

94. If the access token request is invalid, such as the redirect URL didn't match the one used during authorization, then the server must return an error response. The error handling must follow standard OAuth 2 failure codes as per [RFC6749](#).
95. Error responses are returned with an HTTP 400 status code (unless specified otherwise), with error and error_description parameters. The error parameter will always be one of the values listed below.
 - invalid_request – the request is missing a parameter so the server can't proceed with the request, this may also be returned if the request includes an unsupported parameter or repeats a parameter
 - invalid_grant – the authorization code (or user's password for the password grant type) is invalid or expired, this is also the error you would return if the redirect URL given in the authorization grant does not match the URL provided in this access token request
 - invalid_scope – for access token requests that include a scope (password or client_credentials grants), this error indicates an invalid scope value in the request
 - unauthorised_client – this client is not authorized to use the requested grant type, for example, if you restrict which applications can use the Implicit grant, you would return this error for the other apps
 - unsupported_grant_type – if a grant type is requested that the authorization server doesn't recognize, use this code, note that unknown grant types also use this specific error code rather than using the invalid_request above
96. The entire error response is returned as a JSON string, similar to the successful response. Below is an example of an error response.

HTTP/1.1 400 Bad Request

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

```
{
```

```
  "error": "invalid_request",
```

}

Register Pel API

Summary of the register Pel API

97. When the pension provider or scheme (UMA resource server) has determined a match within their internal records and obtained the PAT token to enable access to the UMA protection API hosted in the authorization server then the pension provider or scheme must be able to access the API to register Pels (resources) and place them under protection of an authorization control on behalf of the pension owner (resource owner) and manage them over time. Protection of a resource at the authorization server begins on successful registration and ends on successful deregistration.
98. The authorization server must support the following five registration options and must require a valid PAT for access to them; any other operations are undefined by this specification. Here, rreguri stands for the resource registration endpoint and _id stands for the authorization server-assigned identifier for the web resource, returned by the authorization server when the create resource operation was performed, corresponding to the resource at the time it was created, included within the URL returned in the Location header. Each operation is defined in its own section below.

Create resource description: POST rreguri/

Read resource description: GET rreguri/_id

Update resource description: PUT rreguri/_id

Delete resource description: DELETE rreguri/_id

List resource descriptions: GET rreguri/

99. The resource server must persist the following, for each resource owner following a create resource operation:
- Resource _id – index of the registered resource (Pel)
 - resource owner's PAT – access token to API
 - authorization Server 'AS URI' which issued the PAT (at which the resource _id is registered)
 - address of the authorization server token endpoint

The resource server must also persist these items in a manner which it can locate them using the inbound URL of the view request.

Hosting

100. The API will be hosted on the authorization server which is part of the C&A service. The resource server must use this API at the authorization server's resource registration endpoint to create, read, update, and delete resource descriptions, along with retrieving lists of such descriptions. The descriptions consist of JSON documents that are maintained as web resources at the authorization server. The authorization server should declare this endpoint in the discovery document so that the resource server knows the endpoint.

Format

101. The API must be a REST API using JSON encoded as UTF-8.

Authorization

102. The pension provider or scheme must use the relevant PAT specific to the pension owner to authorise its use of the API when making a request to the authorization server to create, read, update or delete a resource (PeI).

HTTP method

103. This will depend on the type of request ie create, read, update, delete, list. These is covered in the relevant sections below.

Resource description

104. A resource description is a JSON document that describes the characteristics of a resource sufficiently for an authorization server to protect it. A resource description must have the following parameters:

105. **Resource scopes** – required

These must be ["value", "owner", "delegate"]. All three scopes must be used for all registrations so that the resource owner can subsequently delegate access if required without further resource server activity.

106. **Name** – required

This is the URN of the resource, i.e. of the pension asset at the resource server. It must be used as the unique name against which the authorization server, must apply authorization protection and is the representation of the pension asset as is available to a pensions dashboard client. It is a URN of the form: 'urn:pei:'<holder-name GUID>':'<asset GUID>'

107. **Type** – required

This is a URI of the type of all 'name' parameters used in the pensions dashboards ecosystem. It is required for future extension, eg to support resources which have wider scope options than defined here, or for specialised RS-AS relationships in the future.

108. Description – required.

Data standards for the pensions dashboards ecosystem define 'type' and 'name' mandated here.

109. Registration reason – required

A description indicating various reasons for PeI registration. Valid values are:

Value	Description
match-yes	when a PeI is registered for a successful match for a pension asset or a possible match changes to a full match
match-possible	if a PeI is registered for a possible match (that the pension owner then needs to confirm outside the ecosystem)
match-no	if a match-possible has previously been registered but found not to be a match
match-timeout	if a possible match has previously been registered but has since been removed eg if the pension owner did not confirm a match within the required time period
match-withdrawn	if an erroneous match was yes but has now been withdrawn
asset-removed	if a previously registered PeI has been removed eg benefit crystallised or transfer out

110. Inbound request ID – required

The request_id of the inbound find or other trigger which eventually caused this registration event. This event could be generated by the CDA or another business event at the pension provider or scheme.

111. Inbound event type – required

The type of the inbound event (related to the request_id above and the timestamp below).

Value	Description
find	registration as a result of a global find
directed find	registration as a result of directed find
delete	deregistration

112. Inbound timestamp – required

The time stamp of the event which initiated this registration change. Usually this will be the time of the find request, but it may be a time at which an internal process was begun (as per inbound_request_id, which culminated in this registration event. ISO 8601 date and time format.

Create resource description

113. The resource server must use the HTTP "POST" method when registering the resource with the authorization server. The resource server must register each pension asset as a separate resource (to enable delegation and access at the most granular level). The request must contain the required parameters.
114. Example of a resource registration request message with a PAT in the header:

```
POST /rreg/ HTTP/1.1
```

```
Content-Type: application/json;charset=UTF-8
```

```
X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c
```

```
authorization: Bearer
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

```
...
```

```
{
```

```

"resource_scopes":["value", "owner", "delegate"],

"name": "pei:0e55140a-87d3-41cf-b6f7-bc822a4c3c3b:6e29eeb8-814c-44a6-a43f-
b4830f3f4590",

"type": "http://pdp.gov/uma/PEI",

"description" : "Example pensionco Pension" ,

"registration_reason": "match=yes",

"inbound_request_id": "fbe42e69-1307-4dc6-b236-9ad2e0e92275",

"inbound_event_type": "find",

"inbound_timestamp": "2022-09-15T16:16:43+0000"

}

```

115. If the request is successful, the resource is registered and the authorization server must respond with an HTTP 201 status message that includes a location header and an `resource_id` parameter. The `resource_id` parameter is issued by the authorization server for each registered resource, i.e. as a result of each UMA registration of a PEI, initiated by the resource server, authorised by the PAT. The `resource_id` is the common index between authorization server and resource server associated with each PEI.
116. Example resource registration response message:

```

HTTP/1.1 201 Created

Content-Type: application/json;charset=UTF-8

Location: /rreg/KX3A-39WE

...

{

    "resource_id":"KX3A-39WE"

}

```

Read resource description

117. The resource server must use HTTP GET method to read a previously registered resource description. If the request is successful, the authorization server must respond with an HTTP 200 status message that includes a body containing the referenced resource description, along with an `_id` parameter.
118. Example of a read request with a PAT in the header:

```
GET /rreg/KX3A-39WE HTTP/1.1
```

```
X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c
```

```
authorization: Bearer
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

```
...
```

119. Example of a successful response, containing all the parameters that were registered as part of the description:

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
```

```
...
```

```
{
```

```
  "resource_id": "KX3A-39WE",
```

```
  "resource_scopes": ["value", "owner", "delegate"],
```

```
  "name": "pei:0e55140a-87d3-41cf-b6f7-bc822a4c3c3b:6e29eeb8-814c-44a6-a43f-b4830f3f4590",
```

```
  "type": "http://pdp.gov/uma/PEI",
```

```
  "description": "Example pensionco Pension" ,
```

```

    "registration_reason": "Match-yes",

    "inbound_request_id": "fbe42e69-1307-4dc6-b236-9ad2e0e92275",

    "inbound_event_type": "find",

    "inbound_timestamp": "2022-09-15T16:16:43+0000"

}

```

Update resource description

120. The resource server must use the HTTP PUT method to update a previously registered resource description, by means of a complete replacement of the previous resource description. If the request is successful, the authorization server must respond with an HTTP 200 status message that includes an `resource_id` parameter.
121. Example of a update request adding a description parameter to a resource description that previously had none, with a PAT in the header:

```
PUT /rreg/9UQU-DUWW HTTP/1.1
```

```
Content-Type: application/json
```

```
X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c
```

```
authorization: Bearer
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

```
...
```

```
{
```

```
  "resource_scopes":["value", "owner", "delegate"],
```

```
  "name": "pei:0e55140a-87d3-41cf-b6f7-bc822a4c3c3b:6e29eeb8-814c-44a6-a43f-b4830f3f4590",
```

```
  "type": "http://pdp.gov/uma/PEI",
```

```

    "description" : "Example pensionco Pension",

    "registration_reason": "Match-yes",

    "inbound_request_id": "fbe42e69-1307-4dc6-b236-9ad2e0e92275",

    "inbound_event_type": "find",

    "inbound_timestamp": "2022-09-15T16:16:43+0000"

}

```

122. Form of a successful response, not containing the optional `user_access_policy_uri` parameter:

```

HTTP/1.1 200 OK

...

{

    "resource_id": "9UQU-DUWW"

}

```

Delete resource description

123. The resource server must use the HTTP delete method to delete a previously registered resource description. If the request is successful, the resource is deregistered and the authorization server must respond with an HTTP 200 or 204 status message.
124. Form of a delete request, with a PAT in the header:

```

DELETE /rreg/9UQU-DUWW

Content-Type: application/json

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

```

```
...

{

  "registration_reason": "match-withdrawn",

  "inbound_request_id": "fbe42e69-1307-4dc6-b236-9ad2e0e92275",

  "inbound_event_type": "find",

  "inbound_timestamp": "2022-09-15T16:16:43+0000"

}
```

125. Form of a successful response:

HTTP/1.1 204 No content

...

List resource description

- 126. The resource server must use the HTTP GET method to get a list of all previously registered resource identifiers for this resource owner. The authorization server must return the list in the form of a JSON array of resource_id string values.
- 127. The resource server can use this method as a first step in checking whether its understanding of protected resources is in full synchronization with the authorization server's understanding.
- 128. Form of a list request, with a PAT in the header:

GET /rreg/ HTTP/1.1

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

authorization: Bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

...

Form of a successful response:

HTTP/1.1 200 OK

...

["KX3A-39WE",

"9UQU-DUWW"

]

Error handling

- 129. If the request fails because the resource server does not have a valid access token (ie PAT is missing or has expired) then the authorization server responds with an HTTP 401 status code as the request cannot be authenticated.
- 130. If a request is successfully authenticated, but is invalid for another reason, the authorization server produces an error response by supplying a JSON-encoded object with the following members in the body of the HTTP response:
- 131. **error** – REQUIRED except as noted. A single error code. Values for this parameter are defined throughout this specification.
- 132. **error_description** – OPTIONAL. Human-readable text providing additional information.
- 133. **error_uri** – OPTIONAL. A URI identifying a human-readable web page with information about the error.

HTTP/1.1 400 Bad Request

Content-Type: application/json

Cache-Control: no-store

...

{

"error": "invalid_resource_id",

"error_description": "Permission request failed with bad resource ID.",

"error_uri": "errors to be defined later in design"

}

- 134. If the request to the resource registration endpoint is incorrect, then the authorization server instead responds as follows:
- 135. If the referenced resource cannot be found, the authorization server must respond with an HTTP 404 (Not Found) status code and MAY respond with a `not_found` error code.
- 136. If the resource server request used an unsupported HTTP method, the authorization server must respond with the HTTP 405 (Method Not Allowed) status code and MAY respond with an `unsupported_method_type` error code.
- 137. If the request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed, the authorization server must respond with the HTTP 400 (Bad Request) status code and MAY respond with an `invalid_request` error code.

View API

Summary of view API

- 138. Enables a dashboard (client) to retrieve pension details on behalf of a requesting party (i.e pension owner or delegate) by dereferencing the `Pel` which resolves to a URL and making a HTTP GET request to access the pension details. It is this URL which is an UMA protected resource and if the request is authorised via the UMA protocol then the pension provider or scheme must respond back to the dashboard with the pension details encoded within the data payload as a JWT.

Hosting

- 139. Each pension provider or scheme connected to the ecosystem must be required to host their view API within their domain.

Format

- 140. The view API must be a REST API using JSON encoded as UTF-8.

Authorization

- 141. The dashboard client must authenticate itself with the authorization server at run time as defined in [rfc8705 section 2](#). In addition to this, all endpoint connections are secured using mutual TLS.

HTTP method

142. The dashboard must make a HTTP GET request to the pension provider or scheme's view endpoint by dereferencing the PeI which resolves into a URL. The request must carry sufficient information to identify the resource owner at the pension provider or scheme and the type of access being attempted:
- an identifier for the 'pension resource' owned by the resource owner at the resource server which is being accessed
 - the type of requesting party (owner or delegate)
143. The assumed design of the 'unique dereferenceable identifier' for each individual pension resource is of the following form, and the access can carry a query parameter asserting the nature of the requesting party user:
- 'urn:pei:'<holder-name GUID>':'<asset GUID>'
 - ?user=owner (the default if absent) or ?user=delegate (for advisers or guidance staff)
144. The asset GUID can be used as a key within the pension provider or scheme to locate both the internal asset and the pension owner (ie resource owner's identifier and PAT) with which it is associated and persisted by the pension provider or scheme after the create resource operation. The pension provider or scheme must use the PAT associated with the pension owner to coordinate with the authorization server to introspect the RPT (Section 6) or to obtain a permissions ticket (section 7) if necessary.
145. Example of a view request carrying an RPT:

```
GET/pei/b1c832df301a431aab330c7ef88275de?user=owner HTTP/1.1
```

```
Host: www.dashboard.examplepensionco.com
```

```
X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c
```

```
authorization: Bearer
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.cThIloDvwduQB468K5xDc5633seEFoqwxjF_xSJyQQ.
```

146. In the above example the asset GUID is b1c832df-301a-431a-ab33-0c7ef88275de.


```

VUKxcljogXCJsYVJvcmlzXCIsXG4gICAgXCJhbm51YWxSZXBvcnRVUkxcljogXCJxdWlZIGNvbNlY3RldHVyXCIsXG4gICAgXCJzdGF0ZVBlnNp
b25VUkxcljogXCJzaXQaXJ1cmUgY29uc2VjdGV0dXlGXCIsXG4gICAgXCJzaXBVUkxcljogXCJudWxsYSBpcHN1bVwiXG4gICAgXCJzdGF0ZVBlnNp
aW1hdGVkUmV0aXJlbWVudEluY29tZURldGFpbHM4XCi6IHtcbiAgICBclmVyaUFTb3VudDhcljogMTIzMzMxXG4gICAgXCJlcmIXYXJuaW5nOF
wiOiBbXG4gICAgICBcl9USFwiXG4gICAgXSxcbiAgICBclmVyaUlsbHVzdHJhdGlvbRhdGU4XCi6IWFwMTk2Ni0wNC0wOVwiLFxulCAglFwiZXJ
pU2FmZWd1YXJkZWRCZW5lZml0czhcljogdHJ1ZSxcbiAgICBclmVyaU1vbNRobHlBbW91bnQ4XCi6IEYmZmZLFxulCAglFwiZXJpQmVudWZpd
FR5cGU4XCi6IWFwQVZDXCIsXG4gICAgXCJlcmIQb3Q4XCi6IEYmZmZLFxulCAglFwiZXJpRW5kRGF0ZThcljogXCiyMDE3LTA4LTlwXCIsXG4gIC
AgXCJlcmJlbnNyZWZfZThcljogdHJ1ZSxcbiAgICBclmVyaU1vbNRobHlBbW91bnQ4XCi6IEYmZmZLFxulCAglFwiZXJpQmVudWZpd
XCJDU0hMXCIsXG4gICAgXCJlcmITdXJ2aXZvckJlbnVmaXQ4XCi6IGZhbHNILFxulCAglFwiZXJpQmFzaXM4XCi6IWFwU01QSVwiLFxulCAglFwiZ
XJpUGF5YVJzURhdGU4XCi6IWFwMjAwMi0xMC0zMVwiXG4gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
ydWVktW9udGhseUFTb3VudDZcljogNDMzMzLFxulCAglFwiYWNjcnVIZEFubnVhbEFtb3VudDZcljogMTIzMzMxXG4gICAgXCJhY2NydwVWkSW5jc
mVhc2U2XCi6IGZhbHNILFxulCAglFwiYWNjcnVIZEVuZERhdGU2XCi6IWFwMTk5NS0xMS0yMFwiLFxulCAglFwiYWNjcnVIZENhbGN1bGF0aW9
uRGF0ZThcljogXClyMDIxLTAxLTA2XCIsXG4gICAgXCJhY2NydwVWkSW5jdml2b3JCZW5lZml0Ni0wNC0wOVwiLFxulCAglFwiZXJpQmVudWZpd
91bnRUeXBINlwiOiBclmNTSfwiLFxulCAglFwiYWNjcnVIZEFubnVhbEFtb3VudDZcljogMTIzMzMxXG4gICAgXCJhY2NydwVWkSW5jdml2b3JCZW5lZml0Ni0wNC0wOVwiLFxulCAglFwiYWNjcnVIZFBvdDZcljogIDEYmZmZLFxulCAglFwiY3VvaWRhdGF0XzRcljogMjM1XG4gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
QzMdYTYMYNy03MDA4MjY0ZTU3OWMiLjYpYXQlOjE2NtC3MjQ2MjAsIm5iZiI6MTY1NzcyNDYyMCMwZiZlZlJpOjU3ODExMDIwQmVudWZpd
S9Xba0PXLqY5zNTgJStafsc7gxuJoMPIFbxxiQOZkhXyO24IsYpbndw5N7O1oZFY0Ob4sExHFUqSqQ8kCuxi0ENEHERiHN0oX1vXXgveg2B3N1
d0e7DQd-ONKptAHW7lW0dMRSEmntj0-
U8E8lp1aKtmNuZA7Fnp5IJxX_56UGFFVrVfD0mm7LnesblJeuJrRXaOK4_0pVkWwLmCvUSEWYhl3JoZlXmNyej-
DwwFkLhJHosQpZFL54g9qflvs7yyV8QqGLXo-YsyLrNOYz9zoJfY4Uyau_V41wnmP9UJygVEHNmqt8Y3RME7g_kx1_K4vJS9FsE72A "

```

```

}

```

Error handling

149. If the view request is invalid the pension provider or scheme must respond back to the dashboard with a HTTP 401 code and an appropriate error message.
150. If the view request is missing an RPT or has an invalid RPT then the pension provider or scheme in its response must provide a WWW-Authenticate header with the authentication scheme UMA, with the issuer URI from the authorization server's discovery document in an as_uri parameter indicating the URL of the authorization server where the dashboard should reach for further interactions to get an access token and the permission ticket in a ticket parameter. This will enable the dashboard to be able to initiate the authorization protocol and obtain a valid RPT with the authorization server.
151. For example:

```
HTTP/1.1 401 Unauthorized
```

```
WWW-Authenticate: UMA realm="PensionDashboard",
```

```
as_uri="https://as.pdp.com",
```

```

ticket="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWl0IiwiMjM0NTY3ODkwIiwibmFt
ZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.cThlloDvwdueQB468K5xDc5633seE
FoqwxjF_xSJyQQ"

```

```
...
```

152. If the pension provider or scheme is unable to provide a permission ticket from the authorization server or introspect the RPT with the authorization server as it is unable to access the protection API because the PAT has expired, then it includes an error message telling the dashboard the PAT has expired and needs to be refreshed.

153. For example:

```
HTTP/1.1 401 Unauthorized
```

```
as_uri="https://as.pdp.com",
```

```
error: "PAT expired"
```

154. Dashboards which have complied with the redirection and UMA protocols must not repeatedly retry and simply inform their user of remedial action, as such a state is possible if the user has withdrawn consent for this dashboard, or their state at the C&A is indeterminate (they haven't proved their identity) or no PeIs are shareable.

Introspect API

Summary of introspect API

155. When a resource server receives a view request from a dashboard which is accompanied by the access token (RPT), the resource server will need to determine whether the access token is active and, if so, its associated permissions before any pension details can be retrieved and sent back to the dashboard. It does this by introspecting the RPT at the authorization server by using the introspect API. The response of the introspection can be cached for an appropriate amount time so that if the same view request is made during the validity of the cached response then the resource server does not need to make a repeat request to the authorization server to check whether the RPT is active, it can determine this by using a cached copy of the token introspection response. This will avoid excessive load on the authorization server.

Hosting

156. The API must be hosted on the authorization server which is part of the C&A service. The authorization server must declare this endpoint in the discovery document so that the resource server knows the endpoint.

Format

157. The API must be a REST API using JSON encoded as UTF-8

Authorization

158. The pension provider or scheme must use the relevant PAT specific to the pension owner to authorise its use of the API when making an introspection request to the authorization server.

HTTP method

159. The resource server must call the introspection API using HTTP POST method.
160. Example of the resource server's request to the authorization server for introspection of an RPT, with a PAT in the header:

```
POST /introspect HTTP/1.1
```

```
Host: as.pdp.com
```

```
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
```

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

authorization: Bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJhdWQiOiIxMjM5MDIyNDg5MCI6ImV4cCI6MTIzNDU2Nzh9.XliAWL97BMJx4V3WIIZESvEWhw7DGGUTJAOpIGYv_H0

...

token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJhdWQiOiIxMjM5MDIyNDg5MCI6ImV4cCI6MTIzNDU2Nzh9.XliAWL97BMJx4V3WIIZESvEWhw7DGGUTJAOpIGYv_H0

Response

161. The authorization server responds with a JSON object in "application/json" format with the following parameters in the payload:

active

REQUIRED. Boolean indicator of whether or not the presented token is currently active.

permissions

REQUIRED. Extension parameter named permissions that contains an array of objects, each one (representing a single permission) containing these parameters:

resource_id

REQUIRED. A string that uniquely identifies the protected resource, access to which has been granted to this client on behalf of this requesting party. The identifier must correspond to a resource that was previously registered as protected.

resource_scopes

REQUIRED. An array referencing zero or more strings representing scopes to which access was granted for this resource. Each string must correspond to a scope that was registered by this resource server for the referenced resource. Pension dashboard valid granted scopes (in a requesting party token, RPT) must be a list of two containing "value" and "owner" or "delegate".

exp

OPTIONAL. Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this permission will expire. If the token-level exp value pre-dates a permission-level exp value, the token-level value takes precedence.

token_type

OPTIONAL. Type of the token as defined by PDPs UMA Profile, must be pension_dashboard_rpt

exp

OPTIONAL. Integer timestamp, measured in the number of seconds since January 1 1970 UTC, indicating when this token will expire. If the token-level exp value pre-dates a permission-level exp value, the token-level value takes precedence.

iss

OPTIONAL. String representing the issuer of this token.

Example of a response containing the introspection object:

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

...

{

 "active": true,

 "permissions": [

 {

 "resource_id": "658b9e38-dd91-4e35-93ca-5154aba7321e0",

 "resource_scopes": [

 "value", "owner"

],

```

      "exp": 1651001341

    }

  ],

  "token_type": " pension_dashboard_rpt",

  "exp": 1651001341,

  "iss": https://claimsgathe.sandbox.k8s.dev.pensiondashboard.org/am/oauth2

}

```

162. Resource servers are responsible for access to the resource – the introspection response needs to be compared with what is stored internally (ie resource_id, scopes and expiry times) in order to determine whether the access attempt is valid and relates to the correct resource.

Error handling

163. If the request to the introspection endpoint is incorrect, then the authorization server instead responds as follows:

If the referenced resource cannot be found, the authorization server must respond with an HTTP 404 (Not Found) status code and MAY respond with a not_found error code.

If the resource server request used an unsupported HTTP method, the authorization server must respond with the HTTP 405 (Method Not Allowed) status code and MAY respond with an unsupported_method_type error code.

If the request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed, the authorization server must respond with the HTTP 400 (Bad Request) status code and MAY respond with an invalid_request error code.

Permission API

Summary of permission API

164. If the view request made by the dashboard is without an RPT or is accompanied by an invalid RPT then the resource server must coordinate with the authorization server to request one or more permissions (resource identifiers and corresponding scopes) on the dashboard's behalf and

receive a permissions ticket on return. The resource server must request scopes “value” and either “delegate” (if the inbound call explicitly requested this), or, “owner” (by default or as explicitly requested), but not both. The permissions ticket is used by the dashboard to initiate the UMA grant protocol with the authorization server to obtain a new RPT in order to authorise it’s view request.

Hosting

165. The API must be hosted on the authorization server which is part of the C&A service. The authorization server must declare this endpoint in the discovery document so that the resource server knows the endpoint.

Format

166. The API must be a REST API using JSON encoded as UTF-8.

Authorization

167. The pension provider or scheme must use the relevant PAT specific to the pension owner to authorise it’s use of the API when making permissions requests to the authorization server.

HTTP method

168. The resource server must call the permission API using HTTP POST method. The body of the HTTP request message contains a JSON object for requesting a permission for single resource identifier.
169. The object format is derived from the resource description format specified in Section 4.6; it has the following parameters:

resource_id

REQUIRED. The identifier for a resource to which the resource server is requesting a permission on behalf of the client. The identifier must correspond to a resource that was previously registered.

resource_scopes

REQUIRED. An array referencing zero or more identifiers of scopes to which the resource server is requesting access for this resource on behalf of the client. Each scope identifier must correspond to a scope that was previously registered by this resource server for the referenced resource.

Pension Dashboard valid requested scopes (in a permission ticket) must be a list of two containing “value” and “owner” or “delegate”.

170. Example of an HTTP request for a single permission at the authorization server's permission endpoint, with a PAT in the header:

```
POST /perm HTTP/1.1
```

Content-Type: application/json;charset=UTF-8

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

Host: as.pdp.com

authorization: Bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJhdWQiOiIxMjZNdG5McIsImV4cCI6MTIzNDU2Nzh9.XliAWL97BMJx4V3WIIZESvEWhw7DGGUTJAOpIGYv_H0

...

```
{
  "resource_id": " KX3A-39WE",
  "resource_scopes":["value", "owner"],
}
```

Response

171. If the authorization server is successful in creating a permission ticket in response to the resource server's request, it responds with an HTTP 201 (Created) status code and includes the ticket parameter in the JSON-formatted body as signed JWT. Regardless of whether the request contained one or multiple permissions, only a single permission ticket is returned.
172. For example:

HTTP/1.1 201 Created

Content-Type: application/json;charset=UTF-8

...

{

```

"ticket": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJhdWQiOiIxMjM5NDg5MCIslmV4cCI6MjIzNDU2Nzh9.XliAWL97BMJx4V3WIIZESvEWhw7DGGUTJAOpIGYv_H"
}

```

173. The PMT token is a JWT⁴ and contains the following claims.

iss

REQUIRED. Registered claim name. Profiled: unique identifier within dashboard ecosystem of the AS issuing the JWT.

sub

REQUIRED. Registered claim name. Profiled: unique identifier within scope of iss of the resource owner identifier at the AS (*derived from the PAT used in the initial permission ticket request*).

aud

REQUIRED. Registered claim name. Profiled: unique identifier within the scope of the dashboard ecosystem of the authorization server.

iat

REQUIRED. Registered claim name. Profiled: time of issue.

exp

REQUIRED. Registered claim name. Profiled: time of expiry.

jti

REQUIRED. Registered claim name. Profiled: using jti as the unique token identifier.

Public claim name. *none*

⁴ PDP's profile of UMA suggests the use of structured tokens however this is not mandatory and is up to the choice of the UMA implementors

rs

REQUIRED. Private claim name. The identifier of the resource server. *(Derived from the PAT used in the initial permission ticket request.)*

owner

OPTIONAL. Private claim name. The identifier of the RO at the resource server. *(Derived from the PAT used in the initial permission ticket request.) May be of use to the RS to minimise lookup time of resource _id to derive the owner of the resource.*

permissions

REQUIRED. Private claim name. UMA permission as defined in [UMAGrant] and [UMAFedAuthz] using scopes defined in this profile.

rqp

OPTIONAL. Private claim name. Structured representation (JSON object) of the *pension_dashboard_rqp* which was presented with the permission ticket (if any) in a previous call to the AS. Claim must be present when the permission ticket is presented for the second or subsequent time by a dashboard client. The AS must populate this claim with the contents of the *pension_dashboard_rqp* which was presented in the call for which it is reissuing a permission ticket, having checked that the content is in accord with the same requesting party presenter.

assuredID

OPTIONAL. Private claim name. Structured representation (JSON object) of the identity of the requesting party (as uniquely represented at the AS) and the asserting identity provider reference. Claim is present when the AS reissues it after assured identification and if necessary, confirmation of the assured professional status of a 'delegate' requesting party.

174. The token must be signed by the issuer. It must be encrypted for the AS.
175. The token may be persisted by the AS to enable correlation across presentations of such tokens.
176. As per [UMAGrant] 5.5 permission tickets are single use: the AS must issue a new token with a new jti for every iteration of the permission process. The AS must ensure that authorization process and any dependent tokens are revoked if a permission ticket is replayed.
177. The token must always be presented to the token or claims interaction endpoints at the AS by the dashboard client so it does not need to be bound further than application level checking by the AS which must ensure that the *pension_dashboard_rqp* details match across related calls.

Error handling

178. If the resource server's permission registration request is authenticated properly but fails due to other reasons, the authorization server responds with an HTTP 400 (Bad Request) status code and includes one of the following error codes:

“invalid_resource_id” – At least one of the provided resource identifiers was not found at the authorization server.

“invalid_scope” – At least one of the scopes included in the request was not registered previously by this resource server for the referenced resource.

179. If access token (i.e. PAT) used is not valid then authorization server must return a HTTP 401 (unauthorized) with the following error description ‘The access token provided is expired, revoked, malformed, or invalid for other reasons’.

PAT refresh API

Summary of PAT refresh API

180. When the PAT needs to be refreshed the authorization server must send across the required parameters to the pension provider or scheme in order for them to be able to obtain a new PAT. The authorization server must call this API which must be exposed by pension providers and schemes and they must receive the required parameters and trigger them to coordinate with the authorization server to exchange the temporary credential user account token which is an Oauth authorization grant for the PAT.

Hosting

181. Each pension provider or scheme connected to the ecosystem must host their PAT refresh API within their domain.

Format

182. The refresh API must be an Oauth REST API using JSON encoded as UTF-8.

Authorization

183. This is a closed ecosystem with all endpoint connections secured using mutual TLS with only the authorization server invoking the PAT refresh endpoints. There are no additional API security requirements for the refresh API.

HTTP method

184. The authorization server must be restricted to only make a HTTP POST requests to each pension provider or scheme PAT refresh endpoint containing the required parameters: the user account token which is a OAuth2 authorization grant expressed as a JWT which can be exchanged for the PAT, the resource owner's consent expressed as a JWT , the resource_id and Pei so that the pension provider or scheme can locate the resource owner and PAT which requires refreshing.
185. For example, the C&A makes the following HTTP POST request using mTLS:

POST /refresh HTTP/1.1

Host: www.dashboard.examplepensionco.com

Content-Type: application/json; charset=UTF-8

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

{

"user_account_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c",

"consents_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c",

"resource_id": "KX3A-39WE",

"pei": "pei:aed123aq:WGF45920EJH348ASEWQ0284"

}

Response

186. If the request sent by the C&A is a success then the pension provider or scheme must respond with a HTTP 202 *Accepted* status implying acknowledgment of the PAT refresh request.

HTTP/1.1 202 Accepted

187. Following that the resource server requests a new PAT by quoting the user account token in its request the authorization server token endpoint – see Section 3 Obtain PAT.

Error handling

188. If the PAT refresh request sent by the authorization server is fails then the pension provider or scheme is expected to respond with the appropriate HTTP status code and error message. The body of the response for an error must contain a short, focused message where it describes, accordingly with the error catching process, what went wrong and how to amend so a new, valid request can be issued.

189. 4xx

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a POST request, the server **SHOULD** include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition.

Status	Message	Note
400	Bad Request	<p>potential bad requests examples:</p> <ul style="list-style-type: none"> • authorization server is not sending a HTTP POST • parsing error by pension provider or scheme • schema not configured correctly

190. 5xx

The 5xx (server error) class of status code indicates that an exception occurred during the elaboration of a request. An indication about the nature of the error **SHOULD** be included together with an indication if the error is temporary or permanent.

Status	Message	Note
500	internal server error	

Authorise API

Summary of authorise API

191. The dashboard initiates the authorization 'dance' using this OAuth specific API to AS token endpoint to obtain a new access token (RPT). The dashboard must provide a set of claims needed for the AS to assess the current authorization context when interacting with the AS token endpoint. Once this process is successful the AS in its response must issue a new access token (RPT) for the dashboard to authorise it's view requests in the future.

Hosting

192. The API must be hosted on the authorization server which is part of the C&A service. The dashboard must have obtained the `as_uri` from the resource server following a failed view request.

Format

193. The API must be a REST API using JSON encoded as UTF-8.

Authorization

194. The dashboard client must authenticate itself with the authorization server at run time as defined in [rfc8705 section 2](#). In addition to this, all endpoint connections are secured using mutual TLS.

HTTP method

195. The dashboard must use the HTTP "POST" method when making access token.
 196. Requests to the authorization server's token endpoint. It sends the following parameters using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the HTTP request entity-body:

grant_type

REQUIRED. Must be the value `urn:ietf:params:oauth:grant-type:uma-ticket`.

ticket

REQUIRED. The most recent permission ticket received by the client (dashboard) as part of this authorization process. Permission ticket is a structured JWT.

claim_token

REQUIRED. The client must provide a claim of type `pension_dashboard_rqp`. The contents of this token must represent the current requesting party user at the dashboard client.

claim_token_format

REQUIRED. The claim (above) is of a specific type `'pension_dashboad_rqp'`. UMA requires this parameter to match the claim(s).

scope

REQUIRED. The dashboard client request must contain the requested pension dashboard scopes: value and owner/delegate.

pct

OPTIONAL. The client must provide its existing PCT for the requesting party (i.e. for the combination of the client and its user), for the resource the client is seeking to access for that requesting party, if it has one, even if it knows that the PCT has expired.

rpt

OPTIONAL. The client SHOULD provide its existing RPT for the resource it requested in the previous call to the same RS for the same requesting party, if it has one, even if it knows that this RPT is expired. *(Although the profile will not upgrade RPTs, this is included to enable flexibility for possible future extensions.)*

197. For example, the dashboard client makes the following HTTP POST request using mTLS:

POST /token HTTP/1.1

Host: as.pdp.com

Content-Type: application/x-www-form-urlencoded;charset=UTF-8

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

...

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Auma-ticket

&ticket=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

&claim_token= aGVsbG8gd29ybGQ

&claim_token_format= *this type name needs agreed URI format, based on the profile's domain*

&scope=value owner

&pct=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJleHAiOjM1Nzg5Nzg5NywiYWianRpljo1NDIzNjIzOTgwfwQ.hp1udPrSdTRkLf3QzRoHFff_T6BzRqISADIZoZ-c5sl

Response

198. If the permission request is successful the authorization server will issue the RPT and optionally a PCT if required.

HTTP/1.1 200 OK

Content-Type: application/json; charset=UTF-8

...

{

"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c ",

"token_type": "pension_dashboard_rpt",

"upgraded": false,

"pct": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJleHAiOjM1Nzg5Nzg5NywiYWianRpljo1NDIzNjIzOTgwfwQ.hp1udPrSdTRkLf3QzRoHFff_T6BzRqISADIZoZ-c5sl"

```
}

```

Error handling

199. If the permission request is unsuccessful the authorization server will respond with the appropriate error.
200. Example of a need_info response with a hint to redirect the requesting party to a claims interaction endpoint:

```
HTTP/1.1 403 Forbidden

```

```
Content-Type: application/json;charset=UTF-8

```

```
Cache-Control: no-store

```

```
...

```

```
{

```

```
  "error": "need_info",

```

```
  "ticket": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjMONTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyLCJleHAiOiM1Nzg5Nzg5NywiYWV0IjoiNDIzNjIzOTgw",
  "fQ.hp1udPrSdTRkLf3QzRoHFff_T6BzRqISADIZoZ-c5sI ",

```

```
  "redirect_user": "https://as.pdp.com/rqp_claims?id=2346576421"

```

```
}

```

Example when the client was not authorised to have the permissions:

```
HTTP/1.1 403 Forbidden

```

```
Content-Type: application/json;charset=UTF-8

```

```
Cache-Control: no-store

```

```
...

```

```
{

```

```
  "error": "request_denied"

```

}

Obtain Pels API

Summary of obtain Pels API

201. In order for the dashboard, whether used by the owner or their delegate, to obtain the owner's Pels following the completion of find, the dashboard must pull the Pels from the owner user's C&A account via an API hosted at the authorization server.

Hosting

202. The API (an UMA Resource Server) must be hosted on the authorization server which is part of the C&A service.

Format

203. The API must be a REST API using JSON encoded as UTF-8.

Authorization

204. The dashboard client must authenticate itself with the authorization server at run time as defined in [rfc8705 section 2](#). In addition to this, all endpoint connections are secured using mutual TLS.

HTTP method

205. The dashboard must make a HTTP GET request to the obtain Pel endpoint for that specific user's C&A account – ie the dashboard attempts to GET the URL `https://CA.ObtainPels/<userGUID>` with a parameter of the user is 'owner' or 'delegate'. The 'userGUID' is the unique identifier for the user's C&A account. As it is a protected resource the dashboard must need to quote an appropriate access in order for the request to be successful.
206. Example of a obtain Pels request carrying an RPT:

```
GET/a7f542a22c8647b3b62c2fb9a81c2495?user=owner HTTP/1.1
```

```
X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c
```

```
Host: www.CA.ObtainPels.com
```

```

authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.cThIloDvwdueQB468K5xDc5633seEFoqwxjF_xS
JyQQ.

```

Response

207. If the obtain PeI request has been determined to be authorised (as a result of introspecting the RPT or the cached result of previous introspection) then the C&A must respond back to the dashboard with the user's PeI(s).

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
```

```
...
```

```
{
```

```
  "peiList": [ { "pei": "9d0f04b2-8a47-4fb9-91d9-08828036b631:463aabf7-75c1-4d87-
    b5c8-2c7ffca2341f", "description": "Premium Scheme 27" },
```

```
    { "pei": "f315e508-0c84-41bf-afdf-e40db1cb10ec:9da49adb-bc27-4904-8303-
      ba367ead8592", "description": "Example pension Pension" },
```

```
    { "pei": "728f9722-88c1-42f3-965a-d2faab8967e8:26d93ebc-0dfd-43c0-bfee-
      2b8f8ad7a742", "description": "Money Money Pension" } ]
```

```
}
```

208. If the list is empty Send back an HTTP 200 success code with an empty list.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
```

```
...
```

```
{
```

```
  "peiList": []
```

```
}
```

Error handling

209. If the obtain Pel request is invalid the C&A must respond back to the dashboard with a HTTP 401 error code and an appropriate error message.
210. If the request is missing an RPT or has an invalid RPT then the C&A in its response must provide a WWW-Authenticate header with the authentication scheme UMA, with the issuer URI from the authorization server's discovery document in an as_uri parameter indicating the URL of the authorization server where the dashboard should reach for further interactions to get an access token and the permission ticket in a ticket parameter. This must enable the dashboard to be able to initiate the authorization protocol and obtain a valid RPT with the authorization server.
211. For example:

```
HTTP/1.1 401 Unauthorized
```

```
WWW-Authenticate: UMA realm="PensionDashboard",
```

```
as_uri="https://as.pdp.com",
```

```
ticket="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.cThIloDvwduQB468K5xDc5633seEFoqwxjF_xSJyQQ"
```

```
...
```

212. If, after attempting authorization as above a dashboard cannot obtain an access token (RPT) it should stop attempting to do so, perhaps informing its user of remedial action. This condition is possible if the user has withdrawn consent for that dashboard, or in some other way consent to share Pels is not current.

Obtain Pel configuration API

Summary of obtain Pel configuration API

213. Dashboards must be able to dereference the pension identifier which will be in the form of a URI to compose the URL to make the HTTP GET request on to view pension details. Dereferencing a Pel means taking the Pel which is a URI (format 'pei':<holder-name GUID>':<asset GUID> e.g. "pei:cd2a3015-090e-469e-839b-5e5435a29512:3d2b0cde-5831-4537-b4e4-d6c44bf1373a")

and breaking it into components, looking up the holdername, e.g. 'cd2a3015-090e-469e-839b-5e5435a29512' in a configuration table to derive a scheme name, eg 'examplepensionco.dashboard/pei', and composing a URL, e.g. <https://examplepensionco.dashboard/pei/3d2b0cde58314537b4e4d6c44bf1373a>. The dereferencing configuration table at the dashboard is based on master data maintained from the governance register. The master must be accessible via an API by registered dashboard providers.

Hosting

214. The API must be hosted on the governance register.

Format

215. The API must be a REST API using JSON encoded as UTF-8.

Authorization

216. The dashboard client must authenticate itself with the authorization server at run time as defined in [rfc8705 section 2](#). In addition to this, all end point connections are secured using mutual TLS.

HTTP method

217. The Dashboard must make a HTTP GET request to the obtain Pel configuration endpoint with the parameters containing the variable name (i.e. pei) and corresponding value (i.e. holdername).

GET HTTP/1.1

Host: www. GR.ObtainPelsConfig.com

X-Request-ID: 96b71ad6-6e16-43f9-9c3b-8d6bd871a79c

Response

218. The governance register must respond with the corresponding hostname for each of holdername.

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

...

```
{
  "26d93ebc-0dfd-43c0-bfee-2b8f8ad7a742": "examplepensionco.dashboard/pei",
  "728f9722-88c1-42f3-965a-d2faab8967e8": "exampleavcco.dashboard/pei"
}
```

Error handling

219. If the request sent by the dashboard fails then governance register is expected to respond with the appropriate HTTP status code and error message.

400 Bad Request

220. Potential bad requests examples:

- dashboard is not sending a HTTP GET
- parsing error by governance register
- schema not configured correctly

221. 503 Service unavailable

- service is down for maintenance or overloaded by requests

Technical standards

Dashboard redirection protocols

222. Dashboards expose no interfaces (other than their redirection endpoint (which is used to unwind a previous redirection to the relevant Consent and authorization redirection interface)).
223. Redirection from dashboard to the C&A is a vital constituent of these processes and for it is vital for QPDS to understand in connection with the strict 'APIs'.
224. QPDS must redirect their user agents to either of C&A's interfaces – UMAGrant.ClaimsRedirection or ConsentandControl.Redirect. The ClaimsRedirection interface is part of the UMA authorization flow. The Consent.Redirect interface handles requests of type:
- find (incl pullPeis)
 - refresh PAT (RS failure case)
 - consent
 - account deletion (GDPR)
225. For every interaction made by the dashboard to the C&A it has to mint a new RQP token.
226. Formal description of the token. The RqP token is a JWT as defined in [JWT] and must be profiled as follows.

REQUIRED iss. Registered claim name. Defined [JWT]. Profiled: unique identifier within dashboard ecosystem of the dashboard instance issuing the JWT. (This is Dbi.)

REQUIRED sub. Registered claim name. Defined [JWT]. Profiled: unique identifier within scope of iss, of the requesting party which is authenticated to iss at the time the JWT is issued. (This is user@dbi.)

REQUIRED aud. Registered claim name. Defined [JWT]. Profiled: unique identifier within the scope of the dashboard ecosystem of the authorization server.

REQUIRED iat. Registered claim name. Defined [JWT]. Profiled: time of issue.

REQUIRED exp. Registered claim name. Defined [JWT]. Profiled: time of expiry.

REQUIRED jti. Registered claim name. Defined [JWT]. Profiled: using jti as the unique token identifier.

Public claim name. none

REQUIRED role. Private claim name. The iss states the role in which the requesting party is acting. String value. One of “owner” or “delegate”.

- 227. The token must be signed by the issuer. The token may be encrypted for the AS.
- 228. The token must always be presented to the token or claims interaction endpoints at the AS by the dashboard client along with an AS issued token. It does not need to be bound.

JWT signing and verification

- 229. After successfully onboarding to the CDA the pension provider, scheme or QPDS will receive a crypto package which will contain a signing certificate and embedded within it the private key which must be used to generate the signature using RS256 as the signing algorithm to sign a JWT.
- 230. In order verify the signature of a JWT the CDA will expose a centralised JWKS endpoint which the pension provider, scheme or QPDS must be able to call to obtain the corresponding public key to verify the signature by using the “kid” parameter to match on which is found in the JWT header.
- 231. Pension providers, schemes and QPDS must ensure they pass their assigned “kid” parameter as 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 must be a GUID. It will be sent to pension providers, schemes and QPDS as part of the crypto package.
- 232. Example of JWT header containing “kid” parameter:

```
{

  "alg": "RS256",

  "typ": "JWT",

  "kid": "ec1abf89-225b-49c2-ab87-1d425ac70f8d"

}
```

Pension identifier format

- 233. The pension identifier (PeI) need to have a standardised format across all pension providers and schemes. It is expressed in the form of a URN (uniform resource name) that provides a location-independent, globally-unique, persistent identifier, with a defined namespace. Pels are issued by the pension provider or scheme and must be associated with a matched (full or possible) pension asset. Pels (more specifically the PeI.assetGUID) can be associated with a pension asset at any

time prior to registration of the asset with the C&A Service. Once associated with a pension asset a Pel must not be reused for a different pension asset.

234. It is a URN of the form: 'urn:pei:'<holder-name GUID>':'<asset GUID>.
235. An example of a Pel is:

<urn:pei:f1c72611-438b-4f72-a4b5-ec7e69000c31:8ff2063a-48bd-4ed7-bcf8-7c3b8f89626d>

236. Formal description of the Pel. It must be profiled as follows.

REQUIRED

Both the Holder-name GUID and the Asset GUID are globally unique identifiers

Holdername is a globally unique string which can be dereferenced to an endpoint (i.e. URL) which serves view requests, composed of the view endpoint and the asset GUID making the query URL. If properly authorised, the pension details associated with that asset ID are served by the full URL.

<urn:pei:f1c72611-438b-4f72-a4b5-ec7e69000c31:8ff2063a-48bd-4ed7-bcf8-7c3b8f89626d>

example View URL constructed by dereferencing the holdername to the base URL:

<https://testISP.co.uk/8ff2063a48bd4ed7bcf87c3b8f89626d>

GUID creation protocols

237. Formal description of GUIDs. They 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>).

Pension providers and schemes (UMA resource servers)

238. Pension providers and schemes (UMA resource servers) must operate the UMA FedAuthz protocols, which include introspection, permission and register API calls, as described in the above sections.
239. The resource server must persist the following for each resource owner following a create resource operation:
 - resource_id – index of the registered resource (Pel)
 - resource owner's PAT – access token to the Protection API

- authorization server 'AS URI' which issued the PAT (at which the resource_id is registered) - address of the authorization server token endpoint
240. The resource server should also persist these items in a manner which allows it to locate them using the inbound URL of the view request.
241. Resource servers are responsible for access to the resource – introspection response needs to be compared with what is stored internally for the resource in order to ensure access request is authorised.

Appendix

Glossary

Term	Definition
resource owner (i.e. pension owner)	the resource owner is a user or legal entity that is capable of granting access to a protected resource
client (ie dashboard)	the client is an application that is capable of making requests with the resource owner's authorization and on the requesting party's behalf
UMA resource server (ie pension provider or scheme)	the resource server hosts resources on a resource owner's behalf and is capable of accepting and responding to requests for protected resources
UMA authorization server	part of the C&A, the authorization server protects resources hosted on a resource server on behalf of resource owners, it manages and applies the resources owner's policy
PMT - permission token	gives permission for a dashboard to initiate the authorization protocol in order to authorise a retrieval request
RQP - requesting party	identifies the current user and their role in session at the dashboard when requesting access, can be either a pension owner or a delegate
RPT - requesting party access token	needed by the dashboard to authorise its view call to the pension provider or scheme endpoint in order to retrieve pensions details related to the Pel, each Pel must have a unique RPT (i.e. one to one association)

PCT – persistent claims token	this claim binds the asserted user at dashboard and role to that user's assured identity at the ecosystem and, where applicable, professional status, in doing so it acts as a medium-term authenticator correlator helping reduce user friction in subsequent sessions when attempting to retrieve pension details
PAT – protection API token	OAuth2 token scope UMA protection which represents the Resource Owner's authorization for the RS to manage federated authorization at the AS (permits APIs to register, obtain PMT, introspect RPTs)
user account token	OAuth2 authorization grant issued by the authorization server to the resource server so that it can be exchanged at the authorization server's token endpoint for a PAT
