

DPMSG



Digital Property Market Steering Group

Smart Property Data Trust Framework Sandbox



Open Property
Data Association

RAI O I A M

**Introduction to the Sandbox
Technical Guide**

Contents

Introduction	3
How to Obtain Help & Support:	4
Setting up your organisation as a Data Provider or Data Receiver in the Sandbox	5
The Data Provider role	6
If you are a Data Provider (Sharing Data with Others)	7
If you are a Data Receiver (Receiving Data from Others)	12
Appendix A.....	23
Glossary.....	25

Introduction

This guide builds on the [Introduction to the Sandbox: Setup Guide](#), which covered the key concepts behind the sandbox and walked through the initial organisation registration process. This document goes deeper, taking you through the technical steps needed to get fully set up and start sharing data.

This guide covers:

1. **Registering your APIs** (Application Programming Interfaces) - this allows *Data Providers* to make their data available to other authorised organisations in the sandbox.
2. **Creating an Application** - an Application is a definition of a piece of software that can access another organisation's data. *Data Providers* use it to confirm that the software connecting to them genuinely belongs to an authorised *Data Receiver*.
Generating certificates and key pairs - these establish secure, verified communication between your organisation and others in the ecosystem.
3. **Making API calls** - we walk through how to make API calls to a *Data provider* to retrieve data, demonstrating how data flows between participants in the sandbox.

A [glossary of key terms](#) is included at the end of this document and is a useful reference as you work through each step.

How to Obtain Help & Support:

This is a pilot and we support members of all levels of technical capability to participate – and we recognise you and/or your organisation may not feel technically ready. The sandbox is designed to uncover any issues in a safe, non-production environment, and highlight ways to improve the process.

We are here to help. If you are having problems with any step during this guide, please email the appropriate contact:

RAIDIAM Technical lead: alan.hughes@raidiam.com

RAIDIAM Programme Manager: anthony.jones@raidiam.com

OPDA Contact: paul@openpropdata.org.uk

Setting up your organisation as a Data Provider or Data Receiver in the Sandbox

Once your organisation is registered and your administrator account is set up, you will be assigned one or more roles – either as a *Data Provider*, a *Data Receiver*, or both.

- **Data Receiver** - your organisation consumes data from other organisations' APIs.
- **Data Provider** - your organisation shares its data with other authorised organisations via your APIs.

If you're unsure which role has been assigned to your organisation, or you'd like to check that you've met all the prerequisites outlined in the [Introduction to the Sandbox: Setup Guide](#), please get in touch with the project team.

The Data Provider role

As a *Data Provider*, you make your APIs available to *Data Receivers* within the sandbox. Any *Data Receiver* that has been authorised by OPDA can discover and call your APIs – there is no need for you to individually approve or manage each connection. Access is governed by the sandbox itself.

To participate as a *Data Provider*, you will need to register your Authorisation Server with the sandbox and register your APIs so that *Data Receivers* can discover them.

What this role can do:

- Register and manage your Authorisation Server within the sandbox.
- Register your APIs, making them discoverable to authorised *Data Receivers*.
- Share data securely with any *Data Receiver* that has been authorised within the sandbox.

Prerequisites and requirements:

Before registering as a *Data Provider*, you must have the following in place:

- **An Authorisation Server** to manage access to your APIs. You can either run your own OAuth 2.0 compliant authorisation server or use the Raidiam managed authorisation server if you prefer not to manage this yourself.
- **An mTLS gateway** to protect your APIs and ensure only verified participants can connect. See the section on mTLS gateway in the [Introduction to the Sandbox: Setup Guide](#), for guidance.

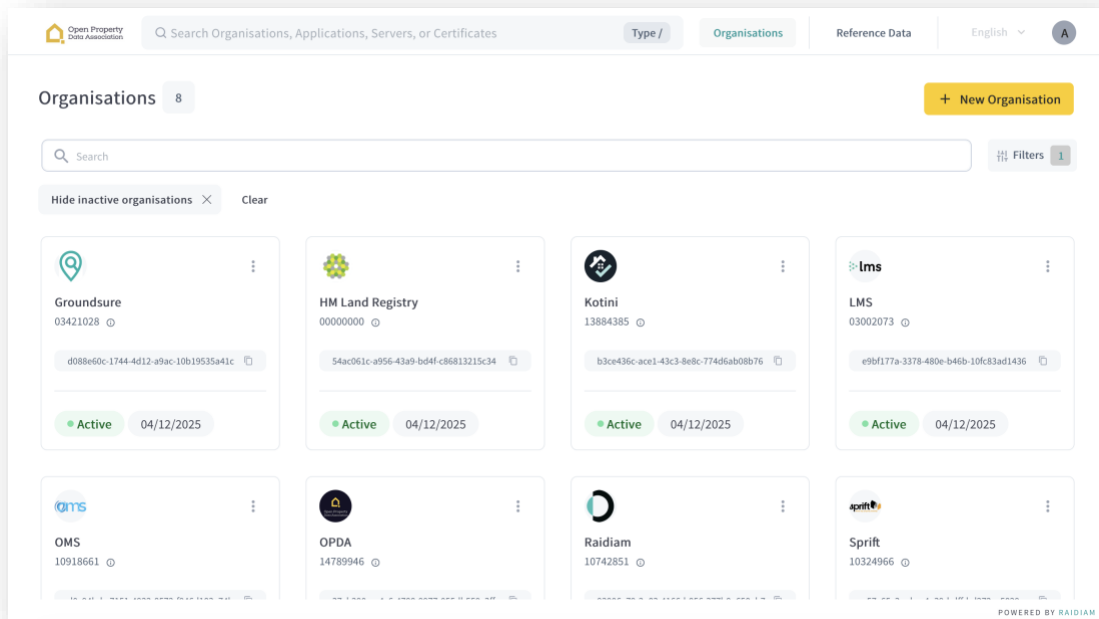
If you have any questions about whether your setup meets these requirements, please get in touch with the project team.

If you are a Data Provider (Sharing Data with Others)

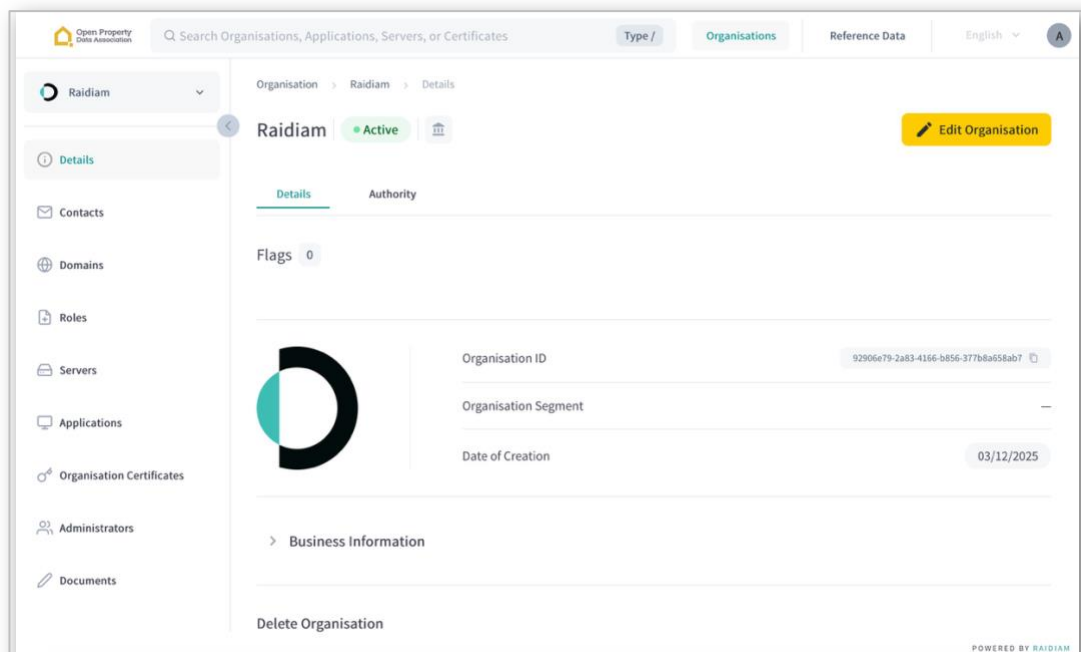
*Note: If you will **only** perform the role of a Data Receiver you can skip this section.*

To enable your APIs to be consumed by *Data Receivers* you will need to create a server within the sandbox and generate the necessary certificates. To do this:

1. Log in to the sandbox (<https://web.directory.pdf.t.raidiam.io>).

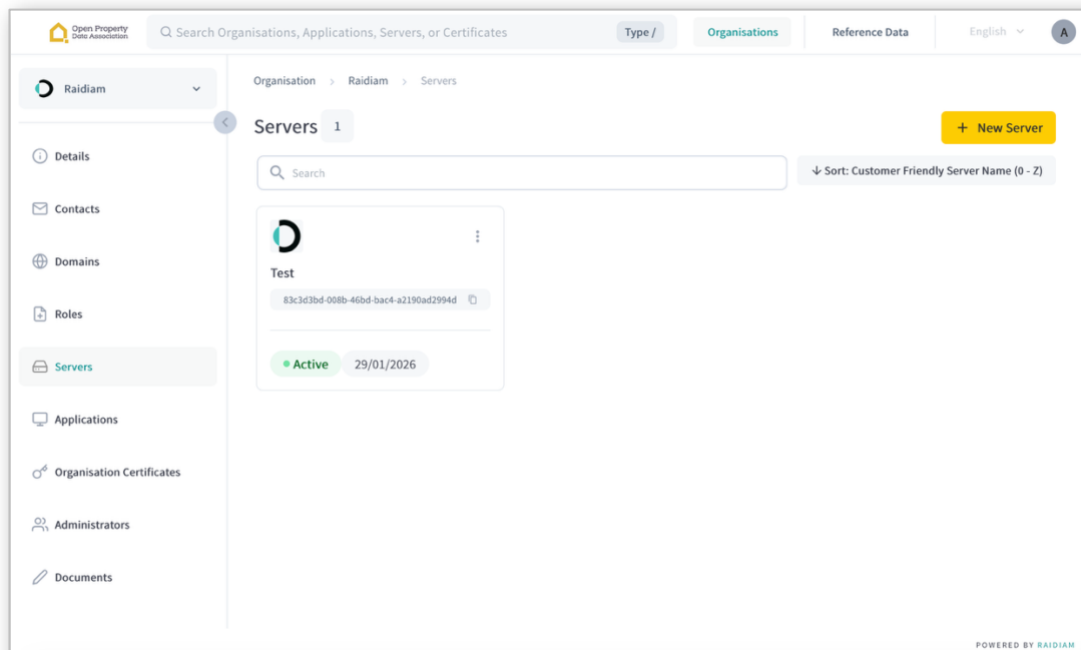


2. Click on your organisation's tile to select your organisation. [Organisations Overview](#) contains more information on the concept of an organisation, and *Data Receiver* and *Data Provider* roles in the sandbox.



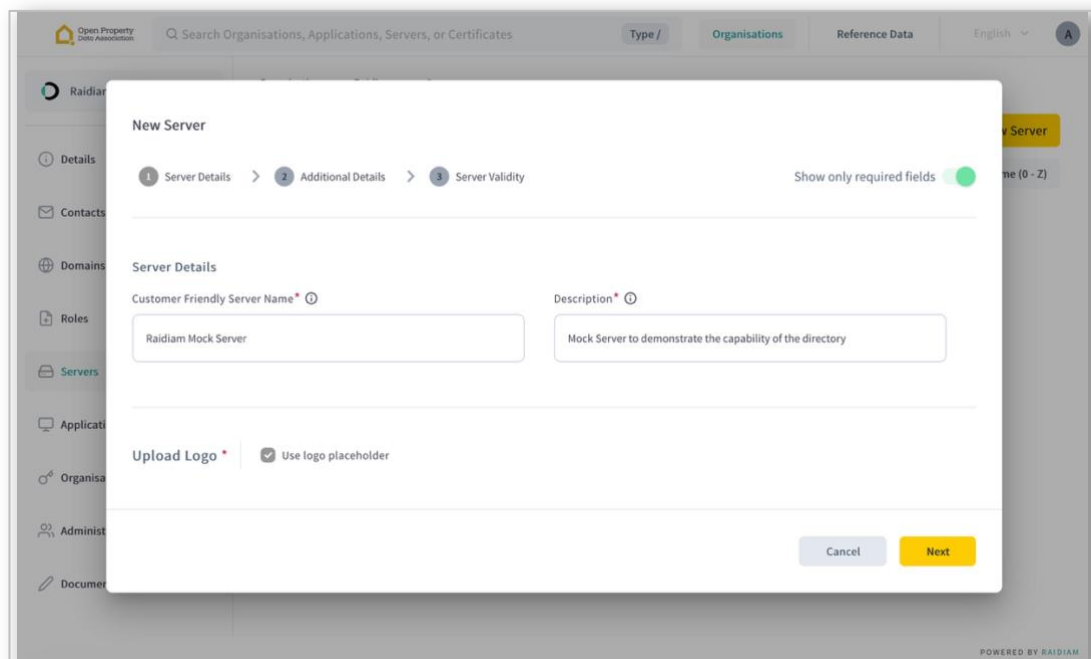
3. Select the 'Servers' option on the left-hand menu to navigate to where registered Authorisation Servers are displayed and managed.

[Authorisation Server Overview](#) contains more information on Auth Servers.



4. Click on the yellow **+ New Server** button at the top right of the screen.
5. Fill in the details for your new server, including its name, description, and logo. To keep things simple, select "Show only required fields" in the top right corner to display just the fields you need to complete. When you're done, click **Next** to continue.

[Add Server](#) contains more information on how to create a server.

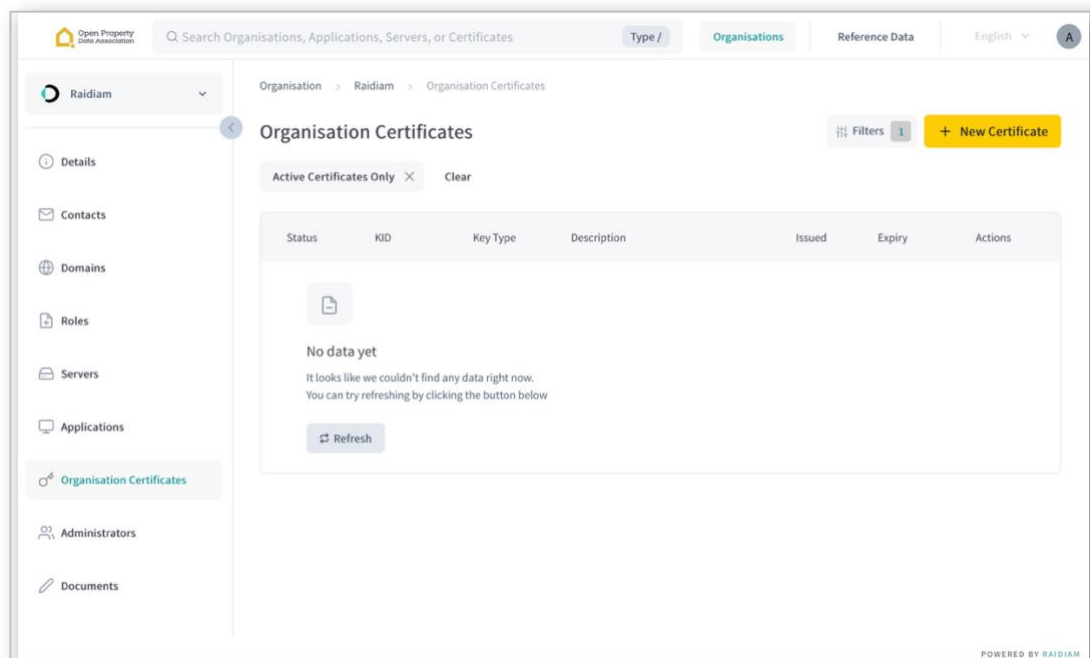


- Next, you'll need to generate certificates for your organisation. These enable secure, verified connections between your organisation and others in the sandbox, and allow you to sign the data you share so that recipients can verify its origin.

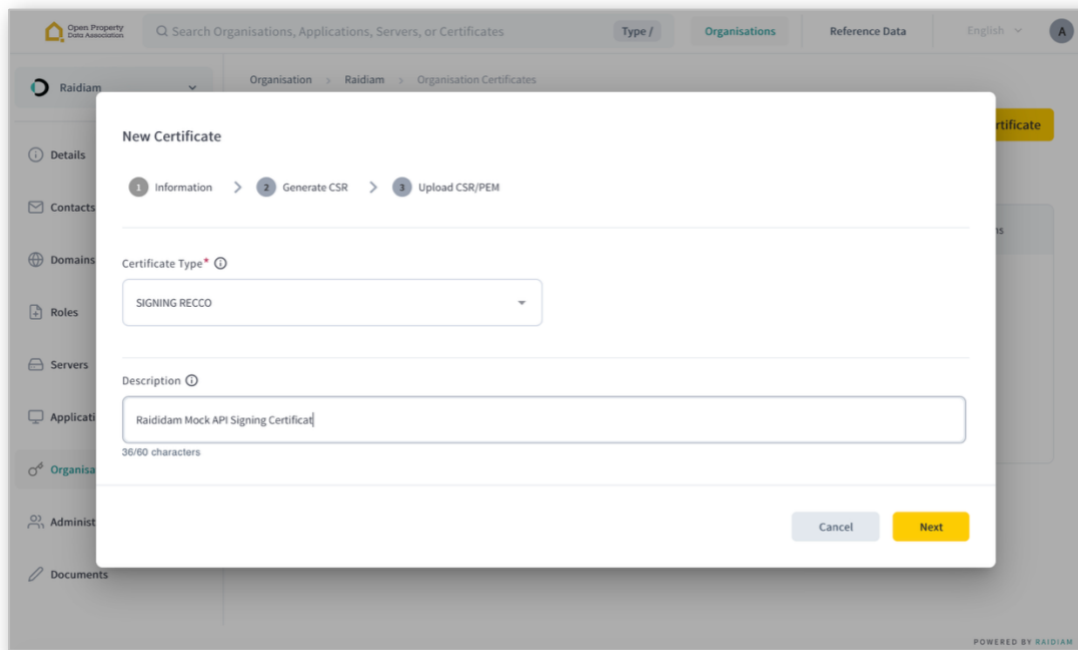
To get started, select Organisation Certificates from the left-hand menu and click **+ New Certificate** in the top right corner. You'll then be asked to choose the type of certificate you need:

- **Transport Certificate** – used for mTLS authentication to establish secure connections.
- **Signing Certificate** – used to sign data payloads, allowing recipients to verify that the data came from you and hasn't been tampered with.

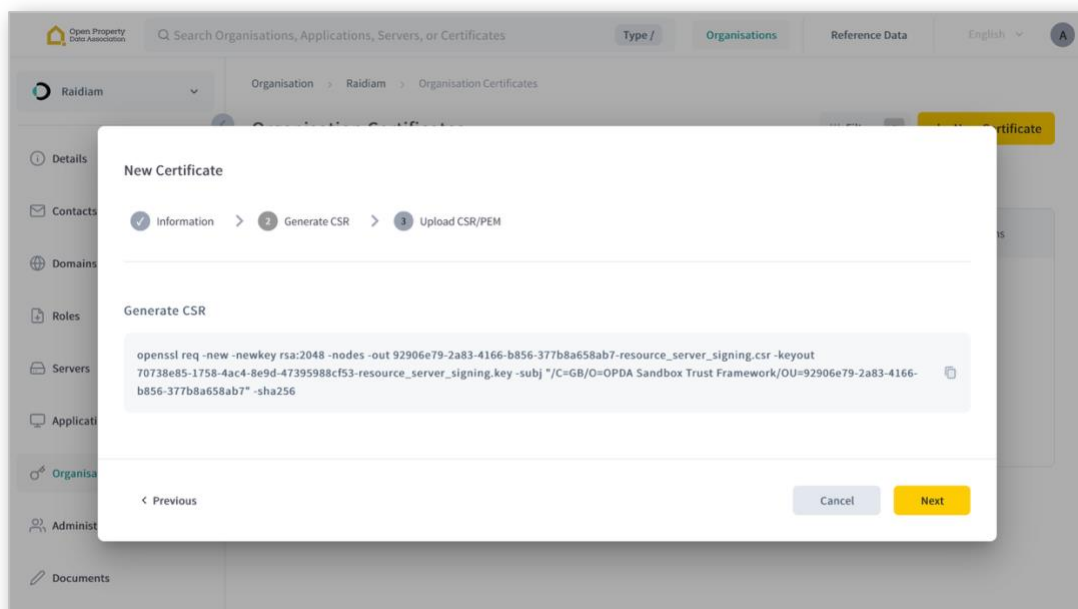
Instructions on how to generate certificates can be found on [Manage Organisation Certificates](#).



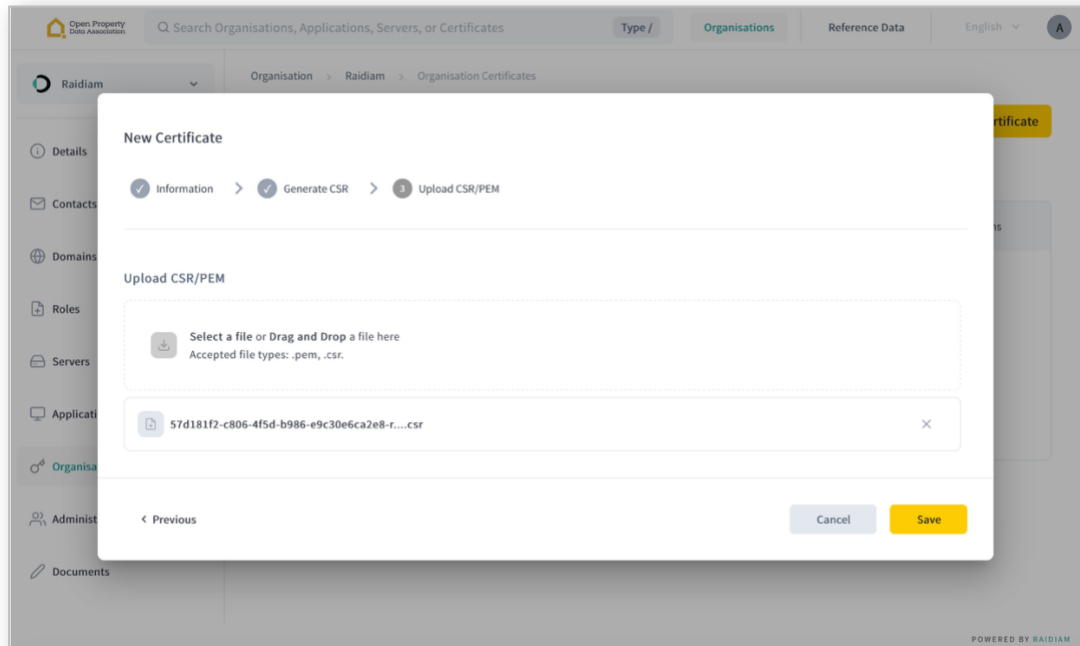
- In this example, we're creating a signing certificate – this allows *Data Receivers* to verify that the data they receive was created by you and has not been tampered with. Give the certificate a description and click **Next** to continue.



- You will be presented with an OpenSSL command to run on your local machine, which will generate your private key and certificate signing request (CSR). Copy the command, run it on your local machine, and then click the **Next** button to continue.



- Once you have run the OpenSSL command and generated your CSR file, upload it by dragging and dropping it into the Upload CSR screen. When you're ready, click **Save** to proceed.



Once validated, the sandbox issues your certificate and stores the associated public key in the sandbox keystore, making it discoverable by other participants for trust validation. You will be given the option to download your certificate at this point, but if you need to download it later, you can do so by clicking the three dots next to the certificate in the certificate list and selecting "Download".

Details of how the keystores operate can be found on: [JSON Web Key Sets - JWKS](#).

If you are a Data Receiver (Receiving Data from Others)

*Note: This section may be skipped if you are **only** acting as a Data Provider.*

As a Data Receiver you must register your applications in the Raidiam Directory.

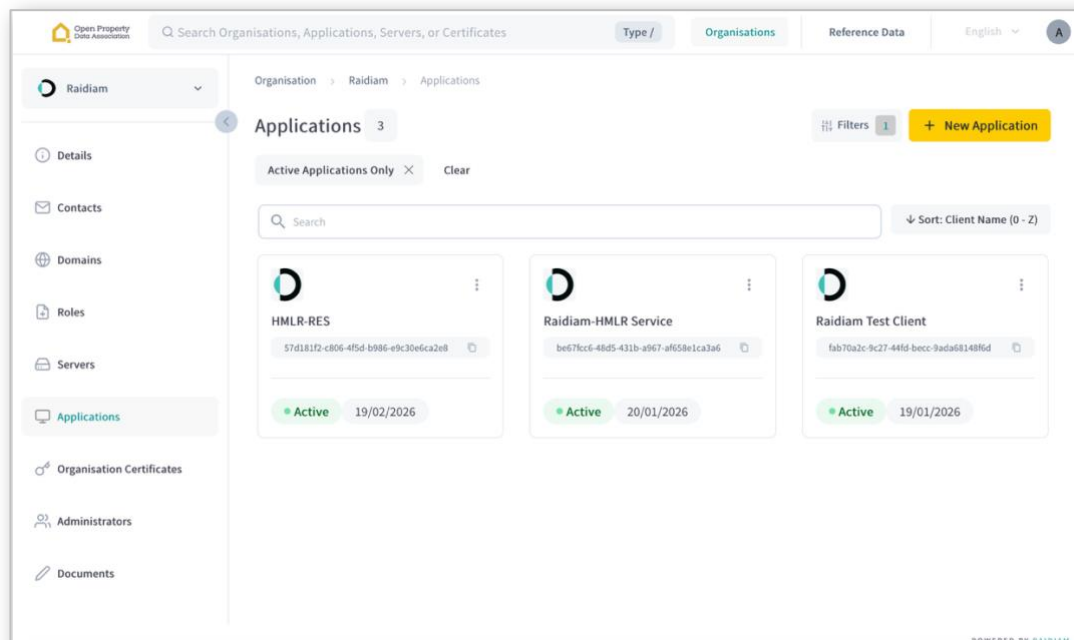
What is an Application?

An Application – also known as a software statement – is a way of registering a specific piece of software with the sandbox. It acts as a formal declaration of who you are and what your software is authorised to do, and is used by *Data Providers* to verify that the software connecting to them genuinely belongs to an authorised participant in the sandbox.

Each Application holds key information about your software, including its associated certificates and the roles it is permitted to perform. When your software makes a connection to a *Data Provider*, it presents this information so that the *Data Provider* can trust the connection before any data is exchanged.

Creating an Application

1. Select Applications from the left-hand menu and click **+ New Application** in the top right corner. An Application is a definition of your software that represents you when connecting to *Data Providers* – it tells the *Data Provider* who you are and what you are authorised to access.



- a. Select the roles that should be associated with this application and click **Next**. In this example, we are selecting the “PIC” (Property Information Consumer) role, which is the role assigned to organisations that consume property data within the sandbox.

The screenshot shows the 'New Client' form with the 'Roles' section active. The progress bar at the top indicates the current step is 'Roles'. Below the progress bar, there is a search bar labeled 'Search by Role'. A tag 'OPDA: PIC' is visible. Two role cards are shown: 'PrDP' (unchecked) and 'PIC' (checked). The 'PIC' card has a dropdown arrow and the text 'scope: land-registry (value) scope: openid (value)'. At the bottom right, there are 'Cancel' and 'Next' buttons.

- b. Next, complete the client details section by providing a name, your software version number and a logo.

The screenshot shows the 'New Client' form with the 'Client Details' section active. The progress bar at the top indicates the current step is 'Client Details'. Below the progress bar, there are two input fields: 'Client Name' with the value 'Raidiam Sample App' and 'Software Version' with the value '1.0'. Below these fields, there is an 'Upload Logo' section with a checked checkbox for 'Use logo placeholder'. At the bottom left, there is a '< Previous' button, and at the bottom right, there are 'Cancel' and 'Next' buttons.

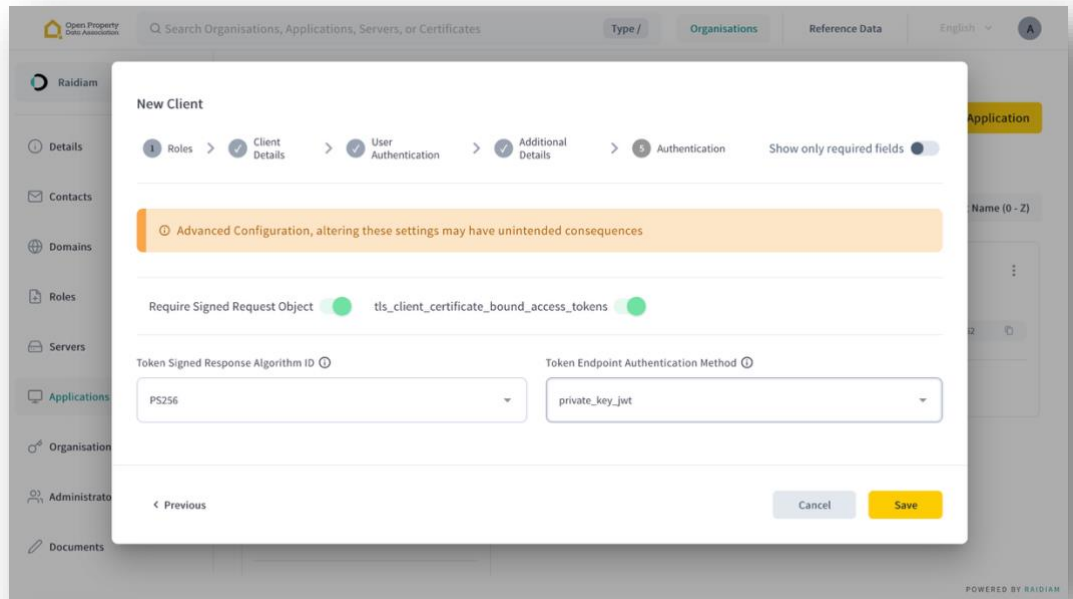
- c. Enter a redirect URI – this is the address the Authorisation Server will send users to after they have successfully authenticated. If your application does not require a user to authenticate (for example, if you are using an OAuth client credentials flow), you can enter a placeholder value here.

The screenshot shows the 'New Client' form in the Raidiam application. The form is divided into five steps: 1. Roles, 2. Client Details, 3. User Authentication, 4. Additional Details, and 5. Authentication. The 'User Authentication' step is currently active. A text input field labeled 'Redirect URI' contains the value 'https://example.com/cb'. Below the input field are 'Previous', 'Cancel', and 'Next' buttons. The 'Next' button is highlighted in yellow.

- d. Click **Next** to skip **Additional Details**.

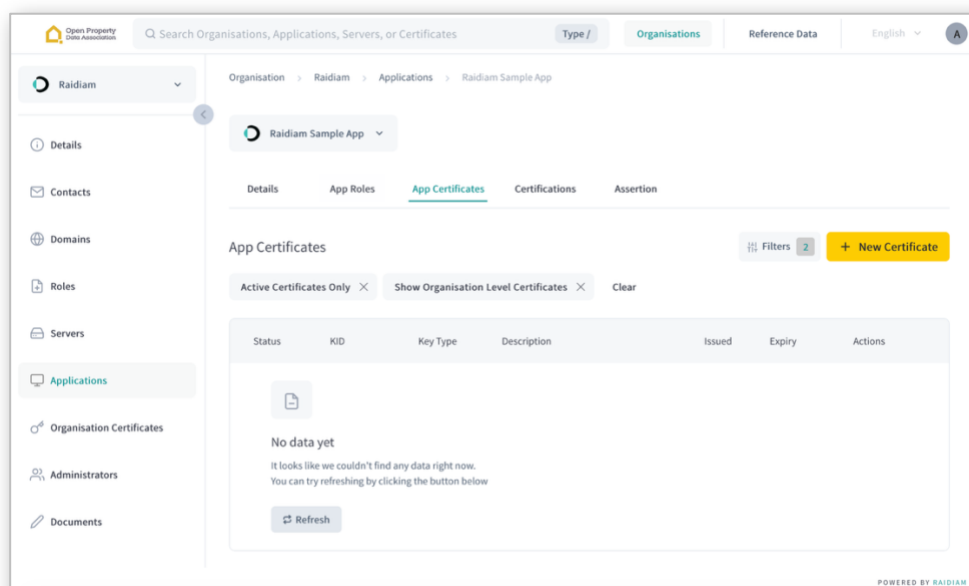
The screenshot shows the 'New Client' form in the Raidiam application. The form is divided into five steps: 1. Roles, 2. Client Details, 3. User Authentication, 4. Additional Details, and 5. Authentication. The 'Additional Details' step is currently active. A message box displays 'No required fields' and 'There are only optional fields in this section. Turn off the toggle to display it'. Below the message are 'Previous', 'Cancel', and 'Save' buttons. The 'Save' button is highlighted in yellow.

- e. On the Authentication section, select “private_key_jwt” from the **Token Endpoint Authentication Method** dropdown list and then choose **Save**.



2. Create a Transport Certificate – your transport certificate is used to establish secure, verified connections between your application and *Data Providers*.

- a. Within your application, click the **App Certificates** tab, then click **+ New Certificate** in the top right corner.



- b. Select “Transport” from the Certificate Type dropdown and give the certificate a description, then click **Next**.

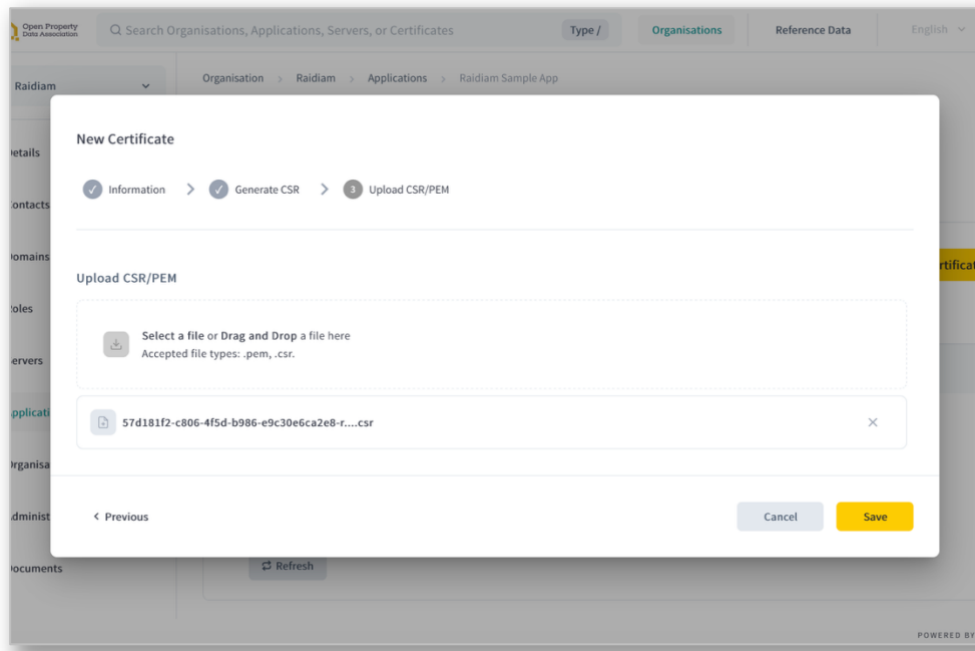
The screenshot shows a web interface for creating a new certificate. The main heading is "New Certificate". Below it, there are three numbered steps: "1 Information", "2 Generate CSR", and "3 Upload CSR/PEM". The "Certificate Type" dropdown menu is open, showing "TRANSPORT" selected. Below that, the "Description" field contains the text "Raidiam Sample App - Transport" and shows a character count of "30/60 characters". At the bottom right of the form, there are two buttons: "Cancel" and "Next".

- c. An OpenSSL command will be generated on screen. Copy this command and run it on your local machine — this will generate a CSR (Certificate Signing Request) and a private key file.

The screenshot shows the "New Certificate" form at the "Generate CSR" step. The "Information" step is marked as complete with a checkmark. The "Generate CSR" section contains a text box with the following OpenSSL command: `openssl req -new -newkey rsa:2048 -nodes -out 92906e79-2a83-4166-b856-377b8a658ab7-resource_server_signing.csr -keyout 70738e85-1758-4ac4-8e9d-47395988cf53-resource_server_signing.key -subj "/C=GB/O=OPDA Sandbox Trust Framework/OU=92906e79-2a83-4166-b856-377b8a658ab7" -sha256`. At the bottom left, there is a "< Previous" button, and at the bottom right, there are "Cancel" and "Next" buttons.

- d. Click **Next** to proceed.

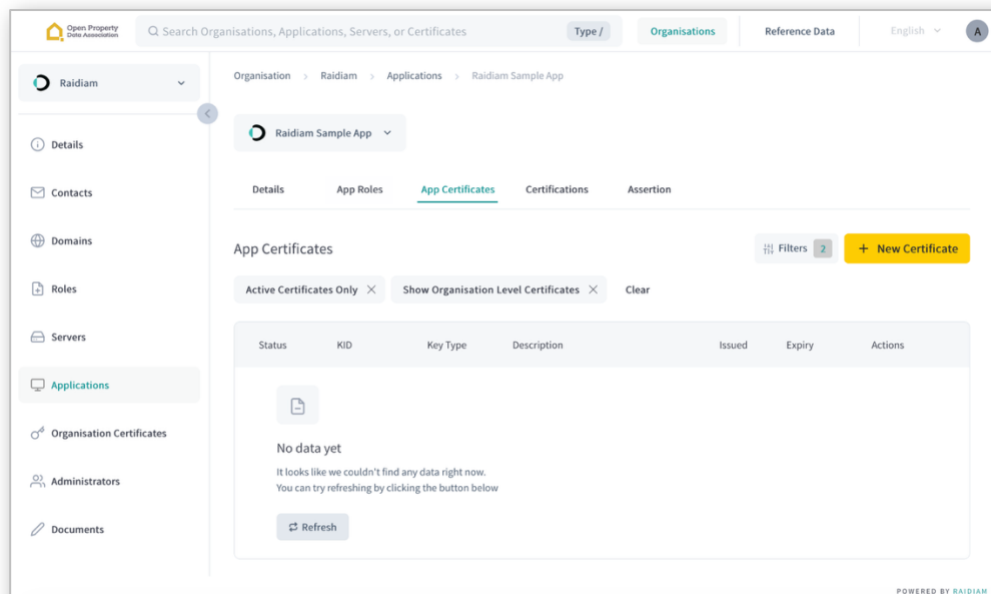
- e. Drag and drop your CSR file onto the Upload CSR screen.



- f. Click **Save**. The sandbox will validate the CSR and store your public certificate in the keystore, making it available to other participants in the sandbox.

3. Create a Signing Certificate

- a. Within your application, click the “App Certificates” tab, then click **+ New Certificate** in the top right corner

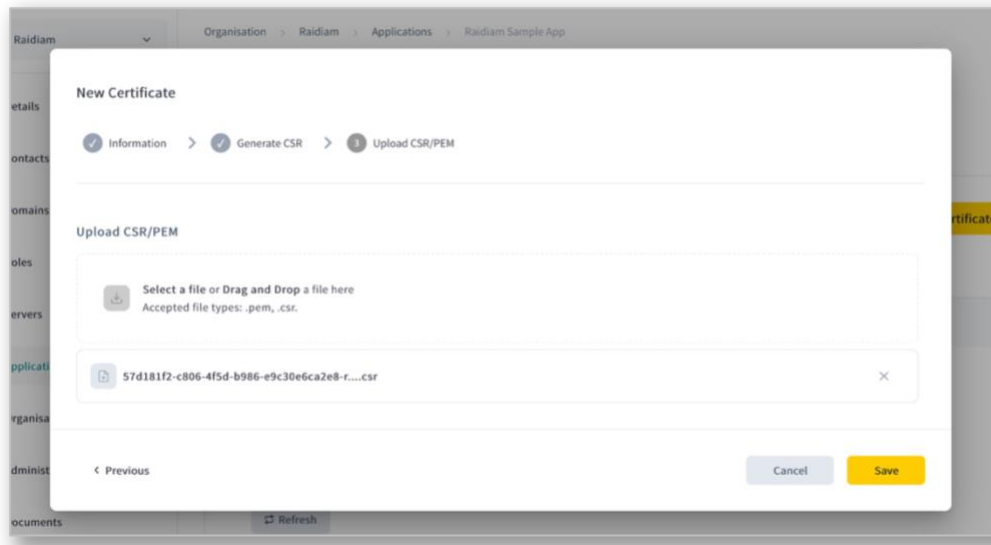


- b. Select “Signing” from the Certificate Type dropdown and give the certificate a description, then click **Next**.

- c. An OpenSSL command will be generated on screen. Copy this command and run it on your local machine — this will generate a CSR (Certificate Signing Request) and a private key file.

- d. Click **Next** to proceed.

- e. Drag and drop your CSR file onto the Upload CSR screen.



- f. Click **Save**. The sandbox will validate the CSR and store your public certificate in the keystore, making it available to other participants in the sandbox.

4. Making a Call to a *Data Provider* - you are now ready to use your Application's Client ID, transport certificate, and signing certificate to make a call to a *Data Provider*. To illustrate how this works, we provide a sample bash script that walks through the full process:

- a. Set up environment variables – the script begins by configuring the required variables: your Client ID, the token endpoint URL, the API URL, and the OAuth scopes required for the call.

```

1  #!/usr/bin/env bash
2
3  #####
4  # Configuration
5  #####
6
7  CLIENT_ID="https://rp.directory.pdf.raidiam.io/openid_relying_party/57d181f2-c806-4f5d-b986-e9c30e6ca2e8"
8  TOKEN_URL="https://matls-auth.directory.pdf.raidiam.io/token"
9  API_URL="https://matls-mockapi.directory.pdf.raidiam.io/opda/official-copies/v1/register-extract"
10 SCOPES="land-registry"
11
12 # Transport certificate and key for mTLS
13 TRANSPORT_KEY="certs/5c097807-5af6-4b9e-99ff-aadf70781d07-rtstransport.key"
14 TRANSPORT_CERT="certs/rlm7cl8FjL2M57mNFCYsaDFcVdAcCuk0DxqvUS14hVo-rtstransport.pem"
15
16 # Signing certificate and key
17 SIGNING_KEY="certs/15352e8c-ae9a-41f0-8378-babc4eb1f0bf-rtssigning.key"
18 SIGNING_CERT="certs/svAJ8cNapJ3TaF4-Po3wGXIr_wW7YR2d5GbLsz6klc-rtssigning.pem"
19

```

- b. Build a Private Key JWT – your Client ID and signing certificate and key are used to construct a Private Key JWT, which is sent to the Authorisation Server to prove your identity.

```

#####
# Step 1: Build Private Key JWT
#####
echo "🟢 Generating client assertion (JWT)"

HEADER='{"alg":"RS256","typ":"JWT"}'
PAYLOAD=$(jq -n \
  --arg iss "$CLIENT_ID" \
  --arg sub "$CLIENT_ID" \
  --arg aud "$TOKEN_URL" \
  --arg jti "$(uuidgen)" \
  --arg exp "$((${date +%s}+300))" \
  '{iss:$iss, sub:$sub, aud:$aud, jti:$jti, exp:(exp|tonumber)}')

# Base64url encode - helper function
b64u() { openssl base64 -e | tr '+/' '-_' | tr -d '=' | tr -d '\n'; }

HEADER_B64=$(echo -n "$HEADER" | b64u)
PAYLOAD_B64=$(echo -n "$PAYLOAD" | b64u)

SIGNING_INPUT="${HEADER_B64}.${PAYLOAD_B64}"

SIGNATURE=$(echo -n "$SIGNING_INPUT" \
  | openssl dgst -sha256 -sign "$SIGNING_KEY" \
  | b64u)

ASSERTION="${SIGNING_INPUT}.${SIGNATURE}"

echo "$ASSERTION"

```

- c. Request a bearer token – the script makes a token request to the Authorisation Server using the Private Key JWT. Your transport certificate and key are used to establish the secure connection.

```

52 #####
53 # Step 2: Request Access Token
54 #####
55 echo "🔵 Requesting OAuth2 token..."
56
57 TOKEN_RESPONSE=$(curl -k -s -X POST "$TOKEN_URL" \
58   --cert "$TRANSPORT_CERT" \
59   --key "$TRANSPORT_KEY" \
60   -H "Content-Type: application/x-www-form-urlencoded" \
61   -d "grant_type=client_credentials" \
62   -d "client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer" \
63   -d "client_assertion=$ASSERTION" \
64   -d "scope=$SCOPES"
65 )
66
67 echo "$TOKEN_RESPONSE"
68

```

- d. Extract the bearer token – the bearer token is extracted from the Authorisation Server's response.

```

69 ACCESS_TOKEN=$(echo "$TOKEN_RESPONSE" | jq -r '.access_token')
70
71 if [ "$ACCESS_TOKEN" == "null" || -z "$ACCESS_TOKEN" ]; then
72   echo "❌ Failed to retrieve access_token:"
73   echo "$TOKEN_RESPONSE"
74   exit 1
75 fi
76

```

- e. Call the *Data Provider's* API – the bearer token is used to make the API call. Again, your transport certificate and key are used to establish the secure connection.

```

#####
# Step 3: Call the API with mTLS
#####
echo "🔵 Calling HMLR Register Extract Service API with mTLS..."

POST_BODY='{
  "messageId": "170100",
  "externalReference": "My_Ext_Ref",
  "customerReference": "BgUser1",
  "titleNumber": "GR506405",
  "expectedPrice": 8.00,
  "titleKnownOfficialCopy": {
    "continueIfTitleIsClosedAndContinued": false,
    "notifyIfPendingFirstRegistration": false,
    "notifyIfPendingApplication": false,
    "sendBackDated": false,
    "continueIfActualFeeExceedsExpected": false,
    "IncludeTitlePlanIndicator": true
  }
}'

curl -k -X POST "$API_URL" \
  --cert "$TRANSPORT_CERT" \
  --key "$TRANSPORT_KEY" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $ACCESS_TOKEN" \
  -H "Accept: application/json" \
  -d "$POST_BODY"

echo -e "\n🟢 API call completed."

```

- f. Receive the response – the API returns its response, completing the data exchange.

```

"OCSummaryData": {
  "OfficialCopyDateTime": "2012-02-06T10:07:49",
  "EditionDate": "2012-01-17",
  "PricePaidEntry": {
    "EntryDetails": {
      "EntryNumber": "2",
      "EntryText": "The value stated as at 19th December 2011 was £149,000.",
      "RegistrationDate": "2011-12-19",
      "SubRegisterCode": "B",
      "Infills": {
        "Amount": "£149,000",
        "Date": "2011-12-19"
      }
    }
  },
  "PropertyAddress": {
    "AddressLine": {
      "Line": [
        "2 William Prance Road"
      ]
    }
  },
  "Title": {
    "TitleNumber": "GR506405",
    "ClassOfTitleCode": "60",
    "CommonholdIndicator": false,
    "TitleRegistrationDetails": {
      "DistrictName": "CITY OF PLYMOUTH",
      "AdministrativeArea": "CITY OF PLYMOUTH",
      "LandRegistryOfficeName": "Plymouth Office",
      "LatestEditionDate": "2012-01-17",
      "RegistrationDate": "2011-12-12"
    }
  },
  "RegisterEntryIndicators": {
    "AgreedNoticeIndicator": true,
    "BankruptcyIndicator": true,
    "CautionIndicator": true,
    "CCBIIndicator": false,
    "ChargeeIndicator": true,
    "ChargeIndicator": true,
    "ChargeRelatedRestrictionIndicator": true,
    "ChargeRestrictionIndicator": true,
    "CreditorsNoticeIndicator": false,
    "DeathOfProprietorIndicator": true,
    "DeedOfPostponementIndicator": true,
    "DiscountChargeIndicator": true,
    "EquitableChargeIndicator": true,
    "GreenOutEntryIndicator": false,
    "HomeRightsChangeOfAddressIndicator": false,
    "HomeRightsIndicator": true,
    "LeaseHoldTitleIndicator": true,
    "MultipleChargeIndicator": true,
    "NonChargeRestrictionIndicator": true,
    "NotedChargeIndicator": true,
    "PricePaidIndicator": true,
    "PropertyDescriptionNotesIndicator": true,
    "RentChargeIndicator": true,
    "RightOfPreEmptionIndicator": true,
    "ScheduleOfLeasesIndicator": true,
    "SubChargeIndicator": true,
    "UnidentifiedEntryIndicator": true,
    "UnilateralNoticeBeneficiaryIndicator": true,
    "UnilateralNoticeIndicator": true,
    "VendorsLienIndicator": true
  },
  "Proprietorship": {
    "CurrentProprietorshipDate": "2011-12-12",
    "RegisteredProprietorParty": [

```

The bash script described above is attached in Appendix A.

Appendix A

```
#!/usr/bin/env bash
#####
# Configuration
#####
CLIENT_ID="https://rp.directory.pdtf.raidiam.io/openid_relying_party/57d181f2-c806-4f5d-b986-
e9c30e6ca2e8"
TOKEN_URL="https://matls-auth.directory.pdtf.raidiam.io/token"
API_URL="https://matls-mockapi.directory.pdtf.raidiam.io/opda/official-copies/v1/register-extract"
SCOPES="land-registry"
# Transport certificate and key for mTLS
TRANSPORT_KEY="certs/5c097807-5af6-4b9e-99ff-aadf70781d07-rtstransport.key"
TRANSPORT_CERT="certs/r1m7c18FjL2M57mNFCYsaDFCvDacCuk0DXqvUS14hVo-rtstransport.pem"
# Signing certificate and key
SIGNING_KEY="certs/15352e8c-ae9a-41f0-8378-babc4eb1f0bf-rtssigning.key"
SIGNING_CERT="certs/svAJ8cNapJ3TaF4-Po3wGXIr_ww7YR2d5GbBLSz6klc-rtssigning.pem"
#####
# Step 1: Build Private Key JWT
#####
echo "🌀 Generating client assertion (JWT)"
HEADER='{ "alg": "RS256", "typ": "JWT" }'
PAYLOAD=$(jq -n \
  --arg iss "$CLIENT_ID" \
  --arg sub "$CLIENT_ID" \
  --arg aud "$TOKEN_URL" \
  --arg jti "$(uuidgen)" \
  --arg exp "$((($(date +%s)+300)))" \
  '{iss:$iss, sub:$sub, aud:$aud, jti:$jti, exp:(($exp|tonumber))}'
)
# Base64Url encode - helper function
b64u() { openssl base64 -e | tr '+/' '-_' | tr -d '=' | tr -d '\n'; }
HEADER_B64=$(echo -n "$HEADER" | b64u)
PAYLOAD_B64=$(echo -n "$PAYLOAD" | b64u)
SIGNING_INPUT="{ $HEADER_B64 }. $PAYLOAD_B64"
SIGNATURE=$(echo -n "$SIGNING_INPUT" \
  | openssl dgst -sha256 -sign "$SIGNING_KEY" \
  | b64u
)
ASSERTION="{ $SIGNING_INPUT }. $SIGNATURE"
echo "$ASSERTION"
#####
# Step 2: Request Access Token
#####
echo "🌀 Requesting OAuth2 token..."
TOKEN_RESPONSE=$(curl -k -s -X POST "$TOKEN_URL" \
  --cert "$TRANSPORT_CERT" \
  --key "$TRANSPORT_KEY" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=client_credentials" \
  -d "client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer" \
  -d "client_assertion=$ASSERTION" \
  -d "scope=$SCOPES"
)
echo "$TOKEN_RESPONSE"
ACCESS_TOKEN=$(echo "$TOKEN_RESPONSE" | jq -r '.access_token')
if [[ "$ACCESS_TOKEN" == "null" || -z "$ACCESS_TOKEN" ]]; then
  echo "❌ Failed to retrieve access_token:"
  echo "$TOKEN_RESPONSE"
  exit 1
fi
#####
# Step 3: Call the API with mTLS
#####
echo "🌀 Calling HMLR Register Extract Service API with mTLS..."
POST_BODY='{
  "messageId": "170100",
  "externalReference": "My_Ext_Ref",

```

```
"customerReference": "BgUser1",
"titleNumber": "GR506405",
"expectedPrice": 8.00,
"titleKnownOfficialCopy": {
  "continueIfTitleIsClosedAndContinued": false,
  "notifyIfPendingFirstRegistration": false,
  "notifyIfPendingApplication": false,
  "sendBackDated": false,
  "continueIfActualFeeExceedsExpected": false,
  "IncludeTitlePlanIndicator": true
}
}'
curl -k -X POST "$API_URL" \
  --cert "$TRANSPORT_CERT" \
  --key "$TRANSPORT_KEY" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $ACCESS_TOKEN" \
  -H "Accept: application/json" \
  -d "$POST_BODY"
echo -e "\n API call completed."
```

Glossary

Term	Definition
Aggregator	A Third Party Provider (TPP) that connects to multiple providers and then brokers access to other providers.
API	Application Programming Interface: a set of rules and protocols that allows different software applications to communicate and interact with each other.
API Provider	Any institution or entity that provides APIs for data sharing or service initiation.
Application	In the context of Raidiam Connect, an Application is a definition for a piece of software that needs to access another organisation's data. It confirms who owns the software, what it is allowed to do, and that it has been approved by the ecosystem owner. Data Providers use it to trust that the software connecting to them genuinely belongs to an authorised Data Receiver.
Authentication	The process of verifying that a user, device, or entity is who they claim to be before granting access to a system or resource.
Authorisation	Deciding whether a particular subject has privilege to access a particular resource based on the specified rights/privileges for accessing resources, in most cases through an access policy
Certificate	An electronic document that links a public key to the identity of a user, device, or organization. It contains the public key, information about its owner, and a digital signature from a trusted Certificate Authority (CA) that verifies the certificate's information. This allows others to confirm the identity of the certificate's owner and establish a secure, encrypted connection.
Certificate Authority	A certificate authority (CA) is a trusted third-party organization that issues digital certificates to verify the identity of entities such as websites, users, and organizations. The CA validates the identity of the certificate applicant and then digitally signs the certificate with its own private key, allowing others to verify the authenticity of the public key and the associated identity.
CRL	A Certificate Revocation List (CRL) is a digitally signed "blacklist" published by a Certificate Authority (CA) that lists the serial numbers of digital certificates that have been revoked before their scheduled expiration date. A CRL is a critical component of a Public Key Infrastructure (PKI) used to manage the trustworthiness of digital certificates.
Data Provider	A category of third party provider that provides data to other third party providers or intermediaries within the homebuying process.
Data Receiver	A category of third party provider that accesses data from a Data Provider/intermediary within the homebuying process.

Term	Definition
DCR	Dynamic Client Registration (DCR) is a standard mechanism within OAuth 2.0 and OpenID Connect frameworks that enables applications to register themselves with an authorization server at runtime.
Endpoint	An endpoint is the specific address for a resource (such as an API, webpage, server, or physical device) that can receive requests and send responses: the URL (Uniform Resource Locator) which provides the unique web address of a resource on the internet.
HTTP	Hypertext Transfer Protocol is the foundation of data communication on the World Wide Web, defining how messages are formatted and transmitted between web servers and browsers.
HTTPS	Hypertext Transfer Protocol Secure is the secure version of HTTP, which is the primary protocol used to send data between a web browser and a website. HTTPS is encrypted in order to increase security of data transfer.
JavaScript	JavaScript is a high-level, multi-paradigm programming language primarily known for enabling interactive and dynamic content on websites.
JSON	JavaScript Object Notation, is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is the de-facto standard protocol to define data objects in an API operation.
JWE	JSON Web Encryption is a standard that defines a way to represent encrypted content as a JSON-based data structure. It allows you to encrypt data, like a JWT or other sensitive information, to ensure its confidentiality.
JWT	JSON Web Token is a standard for securely transmitting information between parties as a JSON object. It's commonly used for authentication and authorisation in web applications and APIs.
MFA	Multi-Factor Authentication, is a security process that requires users to provide two or more verification factors to gain access to an account, application, or system. This method is more secure than a single password because it combines different types of evidence, such as something you know (like a password), something you have (like a phone or hardware key), or something you are (like a fingerprint or face scan).
mTLS	Mutual Transport Layer Security, is a method for two-way authentication that ensures both a client and a server are who they claim to be. Unlike standard Transport Layer Security (TLS) where only the server is verified, mTLS requires both parties to present and validate a digital certificate to establish a secure connection. This provides enhanced security by verifying the identity of both the client and the server before data is exchanged.
OAuth 2.0	OAuth 2.0, which stands for "Open authorisation", is a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user.
OIDC	OpenID Connect (OIDC) is an identity layer built on top of OAuth 2.0. It allows clients to verify the identity of users and obtain basic profile

Term	Definition
	information. OIDC standardises the process of authenticating users when they sign in to access digital services. It's commonly used for single sign on (SSO) allowing users to log in once and access multiple applications. Documentation: https://openid.net/developers/how-connect-works/
OpenID	OpenID is a global standard for identify developed by the OpenID Foundation (OIDF). It enables secure and decentralized single sign-on (SSO), allowing users to log into multiple applications and websites using a single set of credentials from an identity provider. Instead of creating a new username and password for each service, users can sign in with an existing account.
OSCP	Online Certificate Status Protocol (OCSP) is a component of Public Key Infrastructure (PKI) that provides real-time information on the revocation status of a digital certificate. It is an alternative to Certificate Revocation Lists (CRLs), allowing a client to ask an OCSP responder (a server operated by a certificate authority) if a specific certificate is valid, revoked, or unknown.
Participant	A catchall for any entity (Aggregator, API Provider, Data Provider, Data Receiver, Third Party Participant, PropTech, etc) that participates in the Open Property ecosystem.
PKI	Public Key Infrastructure is a system of hardware, software, people, policies, and procedures for creating, managing, distributing, and revoking digital certificates and public keys, enabling secure electronic transactions and communications, specifically for transport layer security, message signing and message encryption.
REST (API)	A REST API is an application programming interface (API) that conforms to the design principles of the representational state transfer (REST) architectural style, a style used to connect distributed hypermedia systems. REST APIs are sometimes referred to as RESTful APIs or RESTful web APIs.
Role	A role is a collection of permissions that determines what a user can do within a system.
Scheme Operator	The body responsible for governing the ecosystem (OPDA in the context of the sandbox), including setting rules and standards, determining the trust framework, granting licenses, etc.
Server	A computer or computer program that manages access to a centralized resource or service in a network.
Software Statement	A software statement is a single, executable instruction in a programming language, or a structured data format like a Software Statement Assertion (SSA). In programming, a statement is the smallest unit that performs an action.
TLS	Transport Layer Security (TLS) is a cryptographic protocol that encrypts data exchanged between two computers, such as a web browser and a

Term	Definition
	server, to ensure the confidentiality, integrity, and authentication of the communication. It prevents third parties from eavesdropping or tampering with messages by creating a secure, private tunnel for digital communications. TLS is the modern successor to SSL
TPP	Third Party Provider - an entity that, with customer consent, consumes the APIs provided by a Data Provider to access property data or initiate services. Can be a Data Recipient and/or a Service Initiator.
Trust Framework	The platform and rules which (at a minimum) acts as the single source of truth for the identity and roles of all Participants and provides central PKI for transport layer security, message signing and encryption.
UI	User interface
URI	A Uniform Resource Identifier is a string of characters that uniquely identifies a resource, which can be anything from a website to an email address to a device.
URL	A Uniform Resource Locator is the unique web address of a resource on the internet, such as an API, webpage, image, or file. It tells your web browser how and where to access that resource, functioning like a street address for the web.
X.509	X.509 is a standard for public key certificates, which are digital documents that securely link a public key to an identity like a website, organization, or individual. These certificates are essential for secure internet protocols like HTTPS (TLS/SSL), validating the identity of a server or user and enabling encrypted communication. An issuing certificate authority digitally signs an X.509 certificate to guarantee its authenticity, and it contains information such as the public key and the identity it belongs to, but not the private key.