FINANCIAL™ DATA EXCHANGE

Recipient Registration with Delegation to an Ecosystem Registry Guidelines

Version 1.0
May 2022

Legal Notice

Revision History

Document Version	Notes	Date
1.0	Initial document release created as a result of RFC 0206 Recipient Registration with Delegation to Ecosystem Registry	**May 2022**

Contents

1 Introduction

1.1 Purpose

Recipient Registration with Delegation to an Ecosystem Registry (RRDER) defines a mechanism by which Data Providers delegate recipient identity to “Ecosystem Registries”, which allows Data Providers to retrieve metadata on Data Recipients for Consent and/or Authorization and/or Consent Management purposes. Under RRDER, Data Providers do not register Data Recipients, instead they delegate registration to an Ecosystem Registry, which maintains and verifies up-to-date metadata on all Data Recipients receiving data from the Data Provider via that Registry.

Recipient Registries are trusted parties who take on the responsibility of gathering recipient metadata on behalf of the Data Provider. Recipient Registries may be provided by Data Access Platforms, which hold agreements with both Data Recipients for onboarding purposes and Data Providers for connectivity, transparency, and data minimization.

This proposal does not specify how Recipient Registries register recipients, but instead defines the interactions between Recipient Registries and Data Providers.

1.2 Problem Statement

Dynamic Client Registration, while an improvement on manual processes, requires continuous onboarding and updating both from initial migrations and on a go-forward basis as new use cases and participants arise. This introduces challenges where multiple recipient metadata data sources must stay in sync in order to maintain current recipient details and serve consumer data requests. Recipient registration is a focal point where Data Providers can delegate the responsibilities for maintaining up-to-date recipient metadata to Ecosystem Registries - including those provided by Data Access Platforms, which already maintain up-to-date client information via their commercial agreements.

RRDER provides Data Providers all the necessary metadata about all relevant Data Recipients via dedicated Ecosystem Registry API endpoints, which Data Providers can call both for initial migration and onboarding, for batch updates when needed, and during the authorization flow for transparency purposes.

1.3 Definitions

See FDX Taxonomy document for a full list of Permissioned Data Sharing terms. These are several of the common terms used in this document to define the Dynamic Client Registration Process.

Recipient Registry: trusted parties who take on the responsibility of gathering recipient metadata on behalf of the Data Provider. Recipient Registries are the typically Data Access Platforms, which already collect Data Recipient information while onboarding their customers. Recipient Registries do not necessarily register Data Recipients with Data Providers, but instead store Data Recipient metadata for Data Provider retrieval.

End Users: include consumers, individuals acting in a business capacity, and entities, such as a business or other legal entity, who are giving permission to share their data.

Data Providers: the entities who hold End Users' Financial Account Information, including and without limitation to banks, credit unions, brokerages, bank and investment service providers, and direct lenders.

Data Recipients: service companies, applications (financial apps), financial institutions, products and services where End Users (on their own or through their End User Delegates) manage or act on their finances, whether actively managing their finances (such as moving money or applying for credit) or passively doing so (such as garnering recommendations or insights).

Data Access Platforms: intermediaries that facilitate financial data access, transit, storage and/or permissioning on behalf of Data Recipients or End Users, also commonly referred to as "Data Aggregators". In some cases, Data Access Platforms do not have a direct relationship with the End User.

Intermediaries: Data Access Platforms, Service Providers, or any other entity in the data sharing chain between a Data Provider to a Data Recipient (**Note**: Term Not defined in the Taxonomy document)

2 Recipient Registration with Delegation to an Ecosystem Registry Standard

2.1 Recipient Metadata

The Data Recipient will register themselves with the Recipient Registry. Data Access Platforms or Direct Data Recipients will inform the Data Provider of the Recipient Registry and Data Recipient identifier at the Recipient Registry. Data Providers will request Data Recipient Metadata from the Recipient Registry.

Joint responsibilities:

- Data Access Platforms or Direct Data Recipients, and Data Providers **MUST** agree on a Recipient Registry and Recipient Registry identifier (i.e., "registry" field passed with OAuth authorization request information).

Data Providers responsibilities:

- Data Providers **MUST** have an established relationship with the Recipient Registry that allows them to request Data Recipient information from the Recipient Registry.

Data Access Platform responsibilities (if involved):

- Data Access Platforms **MUST** be able to identify Data Recipients at agreed upon Recipient Registries for Data Recipients who rely on their services.
- Data Access Platforms **MUST** collect intermediary information.

Direct Data Recipients responsibilities (if involved):

- Direct Data Recipients **MUST** be able to identify themselves at the agreed upon Recipient Registry.

Recipient Registry responsibilities:

- Recipient Registry **MUST** collect the Data Recipient metadata.
- Recipient Registry **MUST** ensure accuracy for all recipient fields. If there is a question about how a recipient is displayed (Name, Logo), the Recipient Registry MUST work with the Data Recipient to obtain the correct information.
- Data Recipients **SHOULD** be easy to identify based on clear names, logos and URIs that can be looked up on the internet.

2.1.1 Recipient Registration

This mechanism by which Data Recipients register themselves with Recipient Registries is out of the scope of this proposal. Recipient Registries and the Recipients they register **MUST** define and agree to this themselves.

2.1.2 Recipient Registry APIs

Recipient Registries **MUST** host two endpoints for bulk recipient metadata retrieval ("/recipients") and individual recipient metadata retrieval ("/recipient/<recipient_id>").

- RRDER GET "/recipients" will return the metadata for Data Recipients registered at the Recipient Registry (i.e., a list of "Recipient Metadata" objects under the "recipients" key). The endpoint is paginated as described in https://fdx.atlassian.net/wiki/spaces/RFC/pages/8585219/REST+Best+Practices+RFC#Pagination.
- RRDER GET "/recipient/<recipient_id>" will take an recipient_id parameter that identifies a specific Data Recipient registered at the Recipient Registry and returns the Data Recipient metadata (i.e., a single "Recipient Metadata" object), or returns a HTTP 404 error if the Recipient Registry cannot identify the Data Recipient using the provided recipient_id. This recipient_id corresponds to the client_id field passed to Data Providers indirectly by Data Access Platforms or directly by Direct Data Recipients in the OAuth authorization request.

2.1.3 Recipient Registry Identification

The exact identifiers used by Data Access Platforms or Direct Data Recipients, and Data Providers to identify Recipient Registries is out of scope of this document. Data Access Platforms or Direct Data Recipients, and Data Providers **MUST** define and agree to the identifier themselves (solving for any identifier conflicts).

2.1.4 Recipient Registry Authentication

The exact mechanism that Recipient Registries use to onboard and authenticate clients (e.g., Data Providers) is out of scope for this proposal. Recipient Registries and their clients **MUST** define and agree to this themselves, and may include sharing client identifiers and secrets.

2.1.5 Recipient Metadata

The following fields are available in Data Recipient Metadata objects in RRDER GET calls to "/recipients" and "/recipient/<recipient_id>":

Name	Required / Optional	Description	Available in UX	Definition Reference
recipient_id	Required	The recipient identifier at the Recipient Registry.	No	
client_name	Required	The recipient name displayed by Data Provider during the consent Flow as well as in the Consent Dashboard. The client_name **MUST** be a user recognizable name.	Yes	RFC7591

Name	Required / Optional	Description	Available in UX	Definition Reference
description	Optional	A short description of the Data Recipient recipient, that **MAY** be presented by the Data Provider to the End-User during the consent flow or in the consent dashboard. The description **MAY** also be included in the Initiate and Disclose steps in the Consent journey.	Yes	RFC7591
redirect_uris[]	Optional	An array of eligible Redirect URI targets to which the Authorization code is sent at the completion of the consent process. Inclusion of redirect_uris is possible in this version only in instances in which DAP is the registry. Future versions will determine how non-DAP ecosystem registry can manage redirect_uris.	No	RFC0153
logo_uri	Optional	Data Recipient Logo URL location. The Recipient Logo **MAY** be included along with the client_name in the Consent Flow and Consent Dashboard.	Yes	RFC7591
client_uri	Optional	The URI which provides additional information about the Data Recipient, and possibly the specific recipient that is being registered.	Yes	RFC7591
contacts[]	Optional	Array of strings representing ways to contact individuals responsible for the Data Recipient application. The contacts array **SHOULD** accurately represent the email aliases of the support groups for the Data Provider in the event of an issue. "contacts":["customer_support@company.com", "abuse@company.com", "site_reliability@company.com"], Contacts **SHOULD** be included if offered as a Data Provider option. For Data Access Platform managed Data Recipients, the support contacts **MAY** include the Data Access Platform email aliases and/or the Data Recipient aliases.	No	RFC7591

Name	Required / Optional	Description	Available in UX	Definition Reference
scope	Optional	String form field with a list of data clusters v1.1 Data Clusters that are in scope for Data Recipient access. The string should list the scope Data Clusters with a space separating each entry. "scope": "ACCOUNT_DETAILED TRANSACTIONS" Data Clusters include: • CUSTOMER_CONTACT • CUSTOMER_PERSONAL • ACCOUNT_BASIC • ACCOUNT_DETAILED • ACCOUNT_PAYMENTS • INVESTMENTS • TRANSACTIONS • STATEMENTS	Yes	RFC7591
duration_type[]	Optional	The duration of consent for the Data Recipient consumers. Options include: • persistent • time_bound • one_time This is an array for Providers that offer duration optionality during consumer authorization and consent.	Yes	RFC0153
duration_period	Optional	Duration period is required when duration_type=time_bound. The duration_period is the maximum consent duration that would be requested for a Recipient consumer. The Data Provider may reject the Recipient DCR request if duration_period requested exceeds the maximum allowed duration offered by the Provider.	Yes	RFC0153
lookback_period	Optional	The maximum number of days allowed for Data Recipient consumers to obtain in transaction history, effective from the current date.	Yes	RFC0153

Name	Required / Optional	Description	Available in UX	Definition Reference
registry_references[]	Optional	An array of external registries containing registered_entity_name, registered_entity_id and registry fields for the registries where the data recipient is registered. If one or more registry entries is included, then all three fields are required to create a valid array object.	No	RFC0206
registered_entity_name	Optional	The official company name for the recipient. This may be the same as the client_name. In many cases this could be a parent company name or the recipient name with Corporation are LLC	No	RFC0153
registered_entity_id	Optional	An ID representing the company that can be looked up from a legal identity registry source.	No	RFC0153
registry	Optional	The Registry source. ENUMS include [PRIVATE, FDX, GLEIF, ICANN]	No	RFC0153

2.1.6 “/recipients” Endpoint Example

2.1.6.1 Request

```
GET /recipients HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: server.example.com
Authorization: Bearer 8IGFGXKXZBV5LL38Y3X1
```

2.1.6.2 Response

```
HTTP/1.1 200
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "page": {
    "nextOffset": "nextoffset",
    "prevOffset": "prevoffset",
    "total": 5000
  },
  "links": {
```

```
    "next": {"href": "/recipients?offset=nextoffset"},
    "prev": {"href": "/recipients?offset=prevoffset"}
  },
  "recipients": [
  {
    "recipient_id": "12345",
    "recipient_name": "My Example Client",
    "description": "Recipient Application for specified financial use case",
    "logo_uri": "https://client.example1.org/logo.png",
    "client_uri": "https://example1.net/",
    "contacts": ["support@example1.net"],
    "scope": "ACCOUNT_DETAILED TRANSACTIONS INVESTMENTS",
    "duration_type": ["time_bound"],
    "duration_period": 365,
    "lookback_period": 365,
    "registry_references": [
    {
      "registered_entity_name": "Data Recipient company legal name",
      "registered_entity_id": "4HCHXIURY78NNH6JH",
      "registry": "GLIEF"
    }]
  },
  {
    "recipient_id": "23456",
    "recipient_name": "Another Example Client",
    "description": "Recipient Application servicing financial use case",
    "logo_uri": "https://client.example2.org/logo.png",
    "client_uri": "https://example2.net/",
    "contacts": ["support@example2.net"],
    "scope": "ACCOUNT_DETAILED INVESTMENTS",
    "duration_type": ["time_bound"],
    "duration_period": 365,
    "lookback_period": 365,
    "registry_references": [
    {
      "registered_entity_name": "Data Recipient company legal name ",
      "registered_entity_id": "8XKSJGEU2465KSOGI",
      "registry": "GLIEF"
    }]
  }]
}
```


2.1.7 “/recipient/<recipient_id>” Endpoint Example

2.1.7.1 Request

```
GET /recipient/12345 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: server.example.com
Authorization: Bearer 8IGFGXKXZBV5LL38Y3X1
```

2.1.7.2 Response

```
HTTP/1.1 200
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "recipient_id": "12345",
  "client_name": "My Example Client",
  "description": "Recipient Application for specified financial use case",
  "logo_uri": "https://client.example.org/logo.png",
  "client_uri": "https://example.net/",
  "contacts": ["support@example.net"],
  "scope": "ACCOUNT_DETAILED TRANSACTIONS INVESTMENTS",
  "duration_type": ["time_bound"],
  "duration_period": 365,
  "lookback_period": 365,
  "registry_references": [
  {
    "registered_entity_name": "Data Recipient company legal name",
    "registered_entity_id": "4HCHXIURY78NNH6JH",
    "registry": "GLIEF"
  }]
}
```

2.2 OAuth Authorization Endpoint Extensions

2.2.1 New Fields

The Data Provider **MUST** extend its OAuth authorization endpoint to support two new authorization request fields, “registry” and “intermediaries”. Data Access Platform **MUST** indirectly provide the “registry” and **MUST** provide the “intermediaries” fields in OAuth authorization requests, if intermediaries are present. If a Direct Data Recipient is involved, the Direct Data Recipient **MUST** provide the “registry” and **MUST NOT** provide the “intermediaries” fields in OAuth authorization requests. These fields are defined below.

1. “registry” specifies the Recipient Registry where the Data Recipient has been registered and can be identified with the client_id passed as the recipient in the “/recipient/<recipient_id>” endpoint.
2. “intermediaries” which is a base64 encoded list of “Intermediary Metadata” objects (fields described below) that the Data Access Platform has collected in order for the Data Recipient to use its service.

2.2.1.1 Intermediary Metadata

Name	Required/ Optional	Description	Available in UX	Definition Reference
name	Optional	A Provider recognizable name for the intermediary. The Data Access Provider Intermediary name **MAY** may be presented during the Initiate, Disclose and Select Data Provider portions of the Consumer Permissioned Access User Experience.	Yes	RFC0117
description	Optional	A short description of the intermediary that **MAY** be also presented to the End-User along with the intermediary name in the consent flow.	Yes	RFC0117
uri	Optional	The URI which provides additional information about the Intermediaries in the chain. The Data Access Platform URI **MAY** be included along with the name and description during the consent flow.	Yes	RFC0117
logo_uri	Optional	Intermediary logo URI location. Similar to the other Intermediary data elements, the Data Access Platform Logo **MAY** be included in the consent journey.	Yes	RFC0117
contacts[]	Optional	Array of strings representing ways to contact individuals responsible for the Intermediary services, typically email addresses. The contacts array **SHOULD** accurately represent the email aliases for the support groups which the Provider should contact in the event of an issue. "contacts": ["customer_support@company.com", "abuse@company.com" "SiteReliabilityEngineering@company.com"], For Data Access Platform managed Data Recipients, the primary support contacts for the Provider **SHOULD** be provided by the Data Access access Platform as part of the for listed Intermediary. The Data Access Platform is the direct OAuth client and will be the direct support contact for the Data Provider.	Yes	RFC0117

Name	Required/ Optional	Description	Available in UX	Definition Reference
registry_references[]	Optional	An array of external registries containing registered_entity_name, registered_entity_id and registry fields for the registries where the data recipient is registered. If one or more registry entries is included, then all three fields are required to create a valid array object.	No	RFC0206
registered_entity_name	Optional	The legal company name for the intermediary.	No	RFC0153
registered_entity_id	Optional	An ID representing the intermediary that can be looked up from a legal identity registry source	No	RFC0153
registry	Optional	The Registry source. ENUMS include [PRIVATE, FDX, GLEIF, ICANN]	No	RFC0153

2.2.2 RFC 9126: OAuth 2.0 Pushed Authorization Requests

Due to the introduction of new fields in the OAuth authorization request that can be tampered with by user agents, Data Access Platforms or Direct Data Recipients, and Data Providers **MUST** agree to use RFC 9126: OAuth 2.0 Pushed Authorization Requests (PAR).

Direct Data Recipients and Data Access Platforms **MUST** pre-share their client credentials with the Data Provider and the Data Provider **MUST** authenticate the client in the incoming PAR requests.

Since these PAR requests are authenticated with client credentials, there is not a risk of open redirection and therefore recipient “redirect_uris” do not need to be pre-registered with the Recipient Registry.

2.3 Registration and UX Guidelines Alignment

This section is identical to corresponding section in the Recipient Registration Guidelines.

2.4 Financial Data Flows

2.4.1 Direct Data Recipient

Data Recipient (Seedling) is also the OAuth client. Data Recipient directly services Application End Consumers redirect and token Authorization with Provider (New Alliance Bank).

Example PAR POST request:

```
POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

response_type=code
&client_id=MZJU23KISOLM89LIF&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fseedling.auth.com%2Fcallback
&code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bww-uCHaoeK1t8U
&code_challenge_method=S256
&registry=FDX
&client_assertion_type=
 urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJraWQiOiI0MiIsImFsZyI6IkVTMjU2In0.eyJpc3MiOiJDTE
 lFTlQxMjM0Iiwic3ViIjoiQ0xJRU5UMTIzNCIsImF1ZCI6Imh0dHBzOi8vc2VydmVyL
 mV4YW1wbGUuY29tIiwiZXhwIjoxNjI1ODY4ODc4fQ.Igw8QrpAWRNPDGoWGRmJumLBM
 wbLjeIYwqWUu-ywgvvufl_0sQJftNs3bzjIrP0BV9rRG-3eI1Ksh0kQ1CwvzA
```

Example PAR POST response:

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json
{
  "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2",
  "expires_in": 90
}
```

Example authorization request:

```
GET /authorize?client_id=MZJU23KISOLM89LIF
&request_uri=urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
Host: as.example.com
```

After receiving and validating this authorization request, the Data Provider **MAY** send a GET "/recipient/MZJU23KISOLM89LIF" to FDX, the Recipient Registry, in this example.

2.4.2 Data Access Platform

Data Recipient (Seedling) registers themselves with the Data Access Platform (ezConnect) which is also acting as an Recipient Registry. The means by which the Data Recipient registers themselves is out of scope for this proposal. Data Access Platform directly services Recipient End Consumers redirect and token Authorization with Provider (New Alliance Bank).

Example PAR POST request:

```
POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
```

```
response_type=code
&client_id=MZJU23KISOLM89LIF&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fezconnect.auth.com%2Fcallback
&code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bww-uCHaoeK1t8U
&code_challenge_method=S256
&registry=FDX
&intermediaries=WwogIHsKICAgICJuYW1lIjogImV6Q29ubmVjdCIsCiAgICAiZGVz
 Y3JpcHRpb24iOiAiQ3JlYXRlIGJlc3QgaW4gY2xhc3MgZmluYW5jaWFsIGV4cGVyaWV
 uY2VzIGZvciB5b3UgY3VzdG9tZXJzIiwKICAgICJ1cmkiOiAiaHR0cHM6Ly9lemNvbm
 5lY3QuY29tLyIsCiAgICAibG9nb191cmkiOiAiaHR0cHM6Ly9lemNvbm5lY3QuY29tL
 2xvZ28ucG5nIiwKICAgICJjb250YWN0cyI6IFsic3VwcG9ydEBlemNvbm5lY3QuY29t
 Il0KICB9Cl0=
&client_assertion_type=
 urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJraWQiOiI0MiIsImFsZyI6IkVTMjU2In0.eyJpc3MiOiJDTE
 lFTlQxMjM0Iiwic3ViIjoiQ0xJRU5UMTIzNCIsImF1ZCI6Imh0dHBzOi8vc2VydmVyL
 mV4YW1wbGUuY29tIiwiZXhwIjoxNjI1ODY4ODc4fQ.Igw8QrpAWRNPDGoWGRmJumLBM
 wbLjeIYwqWUu-ywgvvufl_0sQJftNs3bzjIrP0BV9rRG-3eI1Ksh0kQ1CwvzA
```

Intermediaries parameter after base64 decoding:

```
[
  {
    "name": "ezConnect",
    "description": "Best in class financial experiences for your customers",
    "uri": "https://ezconnect.com/",
   "logo_uri": "https://ezconnect.com/logo.png",
   "contacts": ["support@ezconnect.com"]
  }
]
```

Example PAR POST response:

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json
{
  "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2",
  "expires_in": 90
}
```

Example authorization request:

```
GET /authorize?client_id=MZJU23KISOLM89LIF
&request_uri=urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
Host: as.example.com
```

After receiving and validating this authorization request, the Data Provider **MAY** send a GET "/recipient/MZJU23KISOLM89LIF" to FDX, the Recipient Registry, in this example.

2.4.2 Data Access Platform with Recipient Chain

Data Recipient's (Seedling) is operated by an Online Banking Service Provider (BankUX.com) and is registered with the Data Access Platform (ezConnect) which is also acting as a Recipient Registry. The means by which the Data Recipient registers themselves is out of scope for this proposal. Data Access Platform directly services Recipient End Consumers redirect and token Authorization with Provider (New Alliance Bank).

Example PAR POST request:

```
POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

response_type=code
&client_id=MZJU23KISOLM89LIF&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fezconnect.auth.com%2Fcallback
&code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bww-uCHaoeK1t8U
&code_challenge_method=S256
&registry=FDX
&intermediaries=WwogIHsKICAgICJuYW1lIjogImV6Q29ubmVjdCIsCiAgICAiZGVz
 Y3JpcHRpb24iOiAiQ3JlYXRlIGJlc3QgaW4gY2xhc3MgZmluYW5jaWFsIGV4cGVyaWV
 uY2VzIGZvciB5b3UgY3VzdG9tZXJzIiwKICAgICJ1cmkiOiAiaHR0cHM6Ly9lemNvbm
 5lY3QuY29tLyIsCiAgICAibG9nb191cmkiOiAiaHR0cHM6Ly9lemNvbm5lY3QuY29tL
 2xvZ28ucG5nIiwKICAgICJjb250YWN0cyI6IFsic3VwcG9ydEBlemNvbm5lY3QuY29t
 Il0KICB9LAogIHsKICAgICJuYW1lIjrigJxCYW5rVVguY29tIiwKICAgICJkZXNjcml
 wdGlvbiI6IOKAnEZ1bGwgU2VydmljZSBPbmxpbmUgYW5kIE1vYmlsZSBCYW5raW5nIG
 1hZGUgZWFzeSIsCiAgICAidXJpIjoiaHR0cHM6Ly9iYW5rdXguY29tLyIsCiAgICAib
 G9nb191cmkiOiJodHRwczovL2Jhbmt1eC5jb20vbG9nby5wbmciLAogICAg4oCcY29u
 dGFjdHMiOiBbInN1cHBvcnRAYmFua3V4LmNvbSJdIAogIH0KXQ==
&client_assertion_type=
 urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJraWQiOiI0MiIsImFsZyI6IkVTMjU2In0.eyJpc3MiOiJDTE
 lFTlQxMjM0Iiwic3ViIjoiQ0xJRU5UMTIzNCIsImF1ZCI6Imh0dHBzOi8vc2VydmVyL
 mV4YW1wbGUuY29tIiwiZXhwIjoxNjI1ODY4ODc4fQ.Igw8QrpAWRNPDGoWGRmJumLBM
 wbLjeIYwqWUu-ywgvvufl_0sQJftNs3bzjIrP0BV9rRG-3eI1Ksh0kQ1CwvzA
```

Intermediaries parameter after base64 decoding:

```
[
  {
    "name": "ezConnect",
    "description": "Best in class financial experiences for you customers",
    "uri": "https://ezconnect.com/",
    "logo_uri": "https://ezconnect.com/logo.png",
    "contacts": ["support@ezconnect.com"]
  },
  {
    "name":"BankUX.com",
    "description": "Full Service Online and Mobile Banking made easy",
    "uri":"https://bankux.com/",
    "logo_uri":"https://bankux.com/logo.png",
    "contacts": ["support@bankux.com"]
  }
]
```

Example PAR POST response:

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json
{
  "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2",
  "expires_in": 90
}
```

Example authorization request:

```
GET /authorize?client_id=MZJU23KISOLM89LIF
&request_uri=urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
Host: as.example.com
```

After receiving and validating this authorization request, the Data Provider **MAY** send a GET "/recipient/MZJU23KISOLM89LIF" to FDX, the Recipient Registry, in this example.

3 Recipient Registration with Delegation to an Ecosystem Registry Use Cases

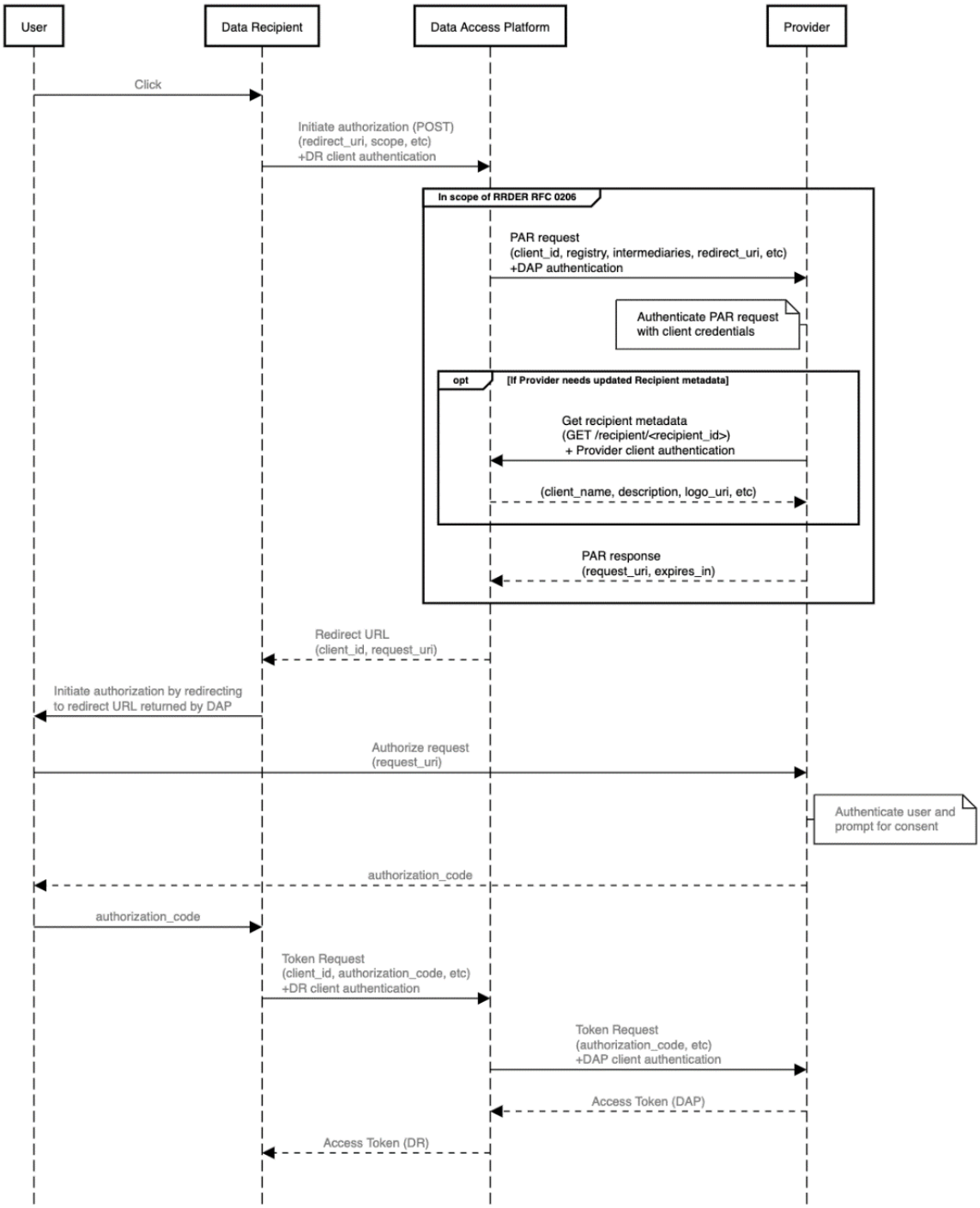
3.1 Data Recipient Metadata Retrieval on Authorization

Data Recipient Metadata Retrieval on Authorization is the simplest use case to implement. Whenever an authorization request is made, the Data Provider requests Data Recipient Metadata from the Recipient Registry using the "/recipient/<recipient_id>" endpoint.

However, since this involves introducing an extra request into the authorization flow, this would add extra latency to this flow which might not be deemed acceptable by either the Data Recipient, Data Access Platform, or Data Provider. The Data Provider can address this by caching Data Recipient Metadata for a certain amount of time and only sending "/recipient/<recipient_id>" requests if the Data Recipient Metadata is not in the cache. The Data Provider **SHOULD NOT** cache Data Recipient Metadata requests for more than 24 hours to avoid serving End Users with stale Data Recipient Metadata.

1. End User makes an authorization request to the Data Provider.
2. Data Provider checks to see if Data Recipient Metadata is in cache.
    1. If yes, the Data Provider renders cached Data Recipient Metadata in authorization flow.
    2. If not, the Data Provider requests fresh Data Recipient Metadata from the Recipient Registry using the "/recipient/<recipient_id>" endpoint and renders it in the authorization flow.

Recipient Registration with Delegation to an Ecosystem Registry: User Authorization with Recipient Metadata Retrieval



Note: This proposal is agnostic to messages/actions in gray, and those are only included in the diagram above for readability reasons. Only messages/actions in black were introduced or changes as part of this proposal.

3.2 Data Recipient Metadata Retrieval on Data Recipient Registration Update Notifications

Recipient Registries (e.g., Data Access Platforms) **MAY** send Data Recipient Registration Update Notifications to Data Providers via webhooks whenever a Data Recipients Metadata changes (e.g., name or logo). If the Data Provider has chosen to cache Data Recipient metadata, they **MAY** choose to implement Data Recipient Metadata Retrieval on Data Recipient Registration Update Notification, in order to reduce the amount of time that stale Data Recipient Metadata is served. This involves retrieving the latest Data Recipient Metadata from the Recipient Registry using the "/recipient/<recipient_id>" endpoint and updating the cache.

If the Recipient Registry agrees to always send Data Recipient Registration Update Notifications to the Data Provider, the Data Provider **MAY** increase the cache TTL to reduce the number of unnecessary Recipient Registry "/recipient/<recipient_id>" calls. The Data Provider **SHOULD** still implement a reasonable cache TTL (e.g., 1 month) so that stale data isn't maintained indefinitely if Data Recipient Registration Update Notifications fail to be delivered or processed.

1. Data Provider receives a Data Recipient Registration Update Notification from a Recipient Registry.
2. Data Provider requests fresh Data Recipient Metadata from the Recipient Registry using the "/recipient/<recipient_id>" endpoint and stores it in the cache, possibly overriding an existing entry.

3.3 Data Recipient Metadata Retrieval on Data Recipient Registration Revoke Notifications

Recipient Registries (e.g., Data Access Platforms) **MAY** send Data Recipient Registration Revoke Notifications to Data Providers, Data Access Platforms, and Direct Data Recipients via webhooks.

Upon receiving a Data Recipient Registration Revoke Notification,

- Data Providers **SHOULD** immediately revoke all active consumer tokens associated with the revoked Data Recipient's registry and recipient_id (i.e., client_id).
- Data Providers **SHOULD** remove the Data Recipient Metadata from the cache, if applicable, thereby preventing new consumer tokens from being issued for the Data Recipient's registry and recipient_id (i.e., client_id).
- Data Access Platforms **SHOULD** stop all PAR, token, and FDX calls to the Provider for the given Data Recipient's registry and recipient_id (i.e., client_id), until a Data Recipient has been registered with the Recipient Registry with the same recipient_id.
- Direct Data Recipients **SHOULD** stop all PAR, token, and FDX calls to the Data Provider for the given Data Recipient's registry and recipient_id (i.e., client_id), until the Data Recipient has been reregistered with the Recipient Registry with the same recipient_id.

3.4 Bulk Data Recipient Metadata Retrieval

Data Providers **MAY** call an Recipient Registry's "/recipients" endpoint on a one-off or recurring basis to avoid having to implement Data Recipient Metadata Retrieval on Authorization or Data Recipient Metadata Retrieval on Data Recipient Registration Update Notifications. If Data Recipient Metadata Retrieval on Data Recipient Registration Update Notifications is not implemented, Data Providers **SHOULD** bulk refresh Data Recipient Metadata every 24 hours to avoid serving End Users stale Data Recipient Metadata. If Data Recipient Metadata Retrieval on Data Recipient Registration Update Notifications is implemented, Data Providers **SHOULD** bulk refresh Data Recipient Metadata every 1 week to avoid serving End Users stale Data Recipient Metadata if Data Recipient Registration Update Notifications fail to be delivered or processed.

If Data Recipient Metadata Retrieval on Data Recipient Registration Update Notifications is not implemented:

1. Data Provider retrieves all Data Recipient Metadata for all registered Data Recipients with an Recipient Registry.
2. Data Provider waits 24 hours and repeats steps 1-2.

If Data Recipient Metadata Retrieval on Data Recipient Registration Update Notifications is implemented:

1. Data Provider retrieves all Data Recipient Metadata for all registered Data Recipients with an Recipient Registry.
2. Data Provider waits 1 week and repeats steps 1-2.