OADA API Overview and Example Flows

Andrew Balmos

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)
    - Lightweight, text-based, language-independent data interchange format.

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)
  - Lightweight, text-based, language-independent data interchange format.
  - Designed to closely resemble JavaScript objects.

```
{
    "key1": {
        "key2": "val1"
    },
    "key2": "val2"
}
```

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)
  - Lightweight, text-based, language-independent data interchange format.
  - Designed to closely resemble JavaScript objects.

```
{
    "key1": {
        "key2": "val1"
    },
    "key2": "val2"
}
```

- **Endpoint** or **URI** - Uniform Resource Identitifer (RFC 3986)

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)
  - Lightweight, text-based, language-independent data interchange format.
  - Designed to closely resemble JavaScript objects.

```
{
    "key1": {
        "key2": "val1"
    },
    "key2": "val2"
}
```

- **Endpoint** or **URI** - Uniform Resource Identitifer (RFC 3986)
  - A path to a resource, for example: */example/data*

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)
  - Lightweight, text-based, language-independent data interchange format.
  - Designed to closely resemble JavaScript objects.

```
{
    "key1": {
        "key2": "val1"
    },
    "key2": "val2"
}
```

- **Endpoint** or **URI** - Uniform Resource Identitifer (RFC 3986)
  - A path to a resource, for example: */example/data*
  - The content may be dependent on the context, e.g., the user, time, etc.

Terms - Just in case

- **JSON** - JavaScript Object Notation (RFC 7159)
  - Lightweight, text-based, language-independent data interchange format.
  - Designed to closely resemble JavaScript objects.

```
{
    "key1": {
        "key2": "val1"
    },
    "key2": "val2"
}
```

- **Endpoint** or **URI** - Uniform Resource Identitifer (RFC 3986)
  - A path to a resource, for example: */example/data*
  - The content may be dependent on the context, e.g., the user, time, etc.
  - To be precise, OADA actually defines URN's

API Endpoint Overview

```
/resources
/configs
/about
/users
/groups
/authorizations
/.well-known
```

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.
- Storing user defined metadata about the resource.

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.
- Storing user defined metadata about the resource.
- Transform and represent data in multiple formats.

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.
- Storing user defined metadata about the resource.
- Transform and represent data in multiple formats.
- Organize the data in a parent/child structure (think Google Drive).

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.
- Storing user defined metadata about the resource.
- Transform and represent data in multiple formats.
- Organize the data in a parent/child structure (think Google Drive).
- Share data with other users.

/resources/{resourceId}

OADA's /resources are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.
- Storing user defined metadata about the resource.
- Transform and represent data in multiple formats.
- Organize the data in a parent/child structure (think Google Drive).
- Share data with other users.

In order to accomplish these tasks with more ease, the following endpoints beneath /resources/{resourceId} are defined:

- /data
- /meta
- /formats
- /parents
- /children
- /permissions

/resources/{resourceId}

OADA's `/resources` are the meat of the OADA API and are also the most complex. It's responsibilities include:

- Storing all data: binary files, JSON documents, etc.
- Storing user defined metadata about the resource.
- Transform and represent data in multiple formats.
- Organize the data in a parent/child structure (think Google Drive).
- Share data with other users.

In order to accomplish these tasks with more ease, the following endpoints beneath `/resources/{resourceId}` are defined:

- `/data`
- `/meta`
- `/formats`
- `/parents`
- `/children`
- `/permissions`

All of the above endpoints return a JSON document except for `/data`, which may be any media type.

Example `/resource/{resourceId}` document

```
{
    "href": "https://api.agcloud.com/resources/ixm24ws",
    "etag": "lajscfa938f23fuj8x",
    "title": "Frank's Yield",
    "mimeType": "application/vnd.oada.yield+json",
    "created": "1985-04-12T23:20:50.52Z",
    "createdBy": {
        "href": "https://api.agcloud.com/users/kdufe3f",
        "account": "frank@agidentity.com",
        "name": "Frank Fellow",
        "picture": {
            "href": "http://www.gravatar.com/avatar/c7e1ee573f"
        },
        "email": "frank@agcloud.com"
    },
    "modified": "1985-04-12T23:20:50.52Z",
    "modifiedBy": {
        "href": "https://api.agcloud.com/users/kdufe3f",
        "account": "frank@agidentity.com",
        "name": "Frank Fellow",
        "picture": {
            "href": "http://www.gravatar.com/avatar/c7e1ee573f"
        },
        "email": "frank@agcloud.com"
    },
...
```

Example `/resource/{resourceId}` document, cont.

```
    "data": {
        "href": "https://api.agcloud.com/resources/ixm24ws/data",
        "etag": "aj3ja8ecuidshfaifx",
        "totalYield": {
            "value": 5.6,
            "unit": "bushel"
        },
        "type": "FeatureCollection",
        "bbox": [40.42426718029455, 40.42429718029455, -86.841822197086, -86.841852197086],
        "features": [{
                ....
        }
    },
    "meta": {
        "href": "https://api.agcloud.com/resources/ixm24ws/meta",
        "etag": "ewiudfaw82y3udhcxz",
        "totalYield": {
            "value": 5.6,
            "unit": "bushel"
        }
    },
...
```

Example `/resource/{resourceId}` document, cont.

```
    "format": {
        "href": "https://api.agcloud.com/resources/ixm24ws/format",
        "etag": "iafueic9jcklhvcyfa",
        "transforms": {
            "application/vnd.oada.yield+json": {
                "lossy": false,
                "name": "OADA GeoJSON Yield Open Format",
                "openFormat": true
            },
            "application/json": {
                "lossy": false,
                "name": "JavaScript Object Notation",
                "openFormat": false
            },
            "application/netcdf": {
                "lossy": false,
                "name": "Network Common Data Form",
                "openFormat": true
            },
            "application/shape": {
                "lossy": false,
                "name": "Esri Shapefile",
                "openFromat": false
            }
        }
    },
...
```

Example `/resource/{resourceId}` document, cont.

```
    "parents": {
        "href": "https://api.agcloud.com/resources/ixm24ws/parents",
        "etag": "oui9032uc8y78a9chx",
        "items": [{
            "href": "https://api.agcloud.com/resources/ixm24ws/parents/me30fzp",
            "resource": {
                "href": "https://api.agcloud.com/resources/me30fzp"
            }
        },
        {
            "href": "https://api.agcloud.com/resources/ixm24ws/parents/me30fzp",
            "resource": {
                "href": "https://api.agcloud.com/resources/me30fzp"
            }
        }]
    },
...
```

Example `/resource/{resourceId}` document, cont.

```
    "children": {
        "href": "https://api.agcloud.com/resources/ixm24ws/children",
        "etag": "asjc93uciasduf3jxj",
        "items": [{
            "href": "https://api.agcloud.com/resources/ixm24ws/children/kl3j93s",
            "resource": {
                "href": "https://api.agcloud.com/resources/kl3j93s"
            }
        },
        {
            "href": "https://api.agcloud.com/resources/ixm24ws/parents/op302xa",
            "resource": {
                "href": "https://api.agcloud.com/resources/op302xa"
            }
        }]
    },
...
```

Example `/resource/{resourceId}` document, cont.

```
    "permissions": {
        "href": "https://api.agcloud.com/resources/ixm24ws/permissions",
        "etag": "cvxfj23fjr9ifu932f",
        "items": [{
            "href": "https://api.agcloud.com/resources/ixm24ws/permissions/kdufe3f",
            "user": {
                "href": "https://api.agcloud.com/users/idnmz83",
                "account": "frank@agidentity.com",
                "name": "Frank Fellow",
                "picture": {
                    "href": "http://www.gravatar.com/avatar/c7e1ee573f"
                },
                "email": "frank@agcloud.com"
            },
            "type": "user",
            "level": "owner"
        }]
    }
}
```

/configs/{key 1}/.../{key N}

- OADA's `/configs` allow API consumers to automatically discover interesting resources without having to search them all manually.

`/configs/{key 1}/.../{key N}`

- OADA's `/configs` allow API consumers to automatically discover interesting resources without having to search them all manually.
- Configurations can only store `href`'s to resources and other lower level configurations.

`/configs/{key 1}/.../{key N}`

- OADA's `/configs` allow API consumers to automatically discover interesting resources without having to search them all manually.
- Configurations can only store `href`'s to resources and other lower level configurations.
- The number of levels of keys is abritary.

/configs/{key 1}/.../{key N}

- OADA's `/configs` allow API consumers to automatically discover interesting resources without having to search them all manually.
- Configurations can only store `href`'s to resources and other lower level configurations.
- The number of levels of keys is abritary.
- To improve interoperability between clouds, applications, and devices, OADA will define a standard set of configuration keys. For example,
  - /configs/fields
  - /configs/seeds
  - /configs/prescriptions/planting
  - /configs/prescriptions/fertilizing
  - etc.

Example `/configs` document

```
{
    "href": "https://api.agcloud.com/configs",
    "etag": "jd983954u7tu90t7dx",
    "items": [{
        "href": "https://api.agcloud.com/configs/fields"
    },
    {
        "href": "https://api.agcloud.com/configs/prescriptions"
    }],
    "resource": {
        "href": "https://api.agcloud.com/resources/fd8as8c"
    }
}
```

`/users/{userId}`

- OADA's `/users` allow API consumers to discover details of any *known* user's real personal identity, such as:
    - Real name
    - Email
    - Avatar

`/users/{userId}`

- OADA's `/users` allow API consumers to discover details of any *known* user's real personal identity, such as:
    - Real name
    - Email
    - Avatar
- Knowledge of personal identity makes sharing a lot nicer.
    - A user can see a picture and real name of another *before* sharing data.

/users/{userId}

- OADA's `/users` allow API consumers to discover details of any *known* user's real personal identity, such as:
    - Real name
    - Email
    - Avatar
- Knowledge of personal identity makes sharing a lot nicer.
    - A user can see a picture and real name of another *before* sharing data.
- A user becomes *known* to you when:
    - It is local to the cloud and its profile is "public".
    - It has previously shared files with you.
- A cloud can not return real identity information for a particular federated identity until it logs into the cloud.
    - Later versions of OADA may consider user discovery across the federation.

Example `/users/{userId}` document

```
{
    "href": "https://api.agcloud.com/users/kdufe3f",
    "etag": "a98345kjgfvjvcvkrc",
    "account": "frank@agidentity.com",
    "name": "Frank Fellow",
    "picture": {
        "url": "http://www.gravatar.com/avatar/c7e1ee573fc6b0956a4455560d5839d9"
    },
    "email": "frank@agcloud.com"
}
```

/about

- OADA's `/about` allows API consumers to discover the currently logged in user and any information needed to "bootstrap" the application or device.

/about

- OADA's `/about` allows API consumers to discover the currently logged in user and any information needed to "bootstrap" the application or device.
    - "Bootstrapping" is the application or device discovering the necessary information to show the user a reasonable first screen.

/about

- OADA's `/about` allows API consumers to discover the currently logged in user and any information needed to "bootstrap" the application or device.
    - "Bootstrapping" is the application or device discovering the necessary information to show the user a reasonable first screen.
        - For example, locating the resource which is the top most parent of all other resources.

/about

- OADA's `/about` allows API consumers to discover the currently logged in user and any information needed to "bootstrap" the application or device.
    - "Bootstrapping" is the application or device discovering the necessary information to show the user a reasonable first screen.
        - For example, locating the resource which is the top most parent of all other resources.

Example `/about` document

```
{
    "href": "https://api.agcloud.com/about",
    "etag": "ajs938r8c87au4t3jm",
    "rootResource": {
        "href": "https://api.agcloud.com/resources/jx9j3x8"
    },
    "currentUser": {
        "href": "https://api.agcloud.com/users/kdufe3f"
    }
}
```

/groups/{groupId}

- OADA's `/groups` allows an API consumer to create and manage groups of users.

/groups/{groupId}

- OADA's `/groups` allows an API consumer to create and manage groups of users.
- Groups can be used to allocate resource permissions more flexibly.
    - For example, users can be added to a group at any time and all previously shared files are automatically accessible.

/groups/{groupId}

- OADA's `/groups` allows an API consumer to create and manage groups of users.
- Groups can be used to allocate resource permissions more flexibly.
    - For example, users can be added to a group at any time and all previously shared files are automatically accessible.

Example `/groups/{groupId}` document

```
{
    "href": "https://api.agcloud.com/groups/jf72jsd",
    "etag": "kjasfd9c7ua3c772rx",
    "title": "Employees",
    "members": [{
        "href": "https://api.agcloud.com/users/kdufe3f"
    },
    {
        "href": "https://api.openagi.io/users/3jkxi82"
    }]
}
```

`/authorizations/{authorizationId}`

- OADA's `/authorizations` allows an API consumer to create and manage authorizations for a third party.

`/authorizations/{authorizationId}`

- OADA's `/authorizations` allows an API consumer to create and manage authorizations for a third party.
- Currently it is meant to manage OAuth 2.0 tokens, but could hypothetically manage any type of authorization.

/authorizations/{authorizationId}

- OADA's `/authorizations` allows an API consumer to create and manage authorizations for a third party.
- Currently it is meant to manage OAuth 2.0 tokens, but could hypothetically manage any type of authorization.

Example `/authorizations/{authorizationId}` document

```
{
    "href": "https://api.agcloud.com/authorizations/8ackam3",
    "etag": "fkjasdc9772893r7ex",
    "user": {
        "href": "https://api.agcloud.com/users/fjf23cd"
    },
    "scope": "resources groups",
    "created": "1985-04-12T23:20:50.52Z",
    "modified": "1985-04-12T23:20:50.52Z",
    "expires": "1985-05-12T23:20:50.52Z"
}
```

`/.well-known`

- OADA's `/.well-known` allows an API consumer to find the base URI for a particular domain's OADA API as well as discover the needed details to authorize users.

`/.well-known`

- OADA's `/.well-known` allows an API consumer to find the base URI for a particular domain's OADA API as well as discover the needed details to authorize users.
- This endpoint follows RFC 5785

`/.well-known`

- OADA's `/.well-known` allows an API consumer to find the base URI for a particular domain's OADA API as well as discover the needed details to authorize users.
- This endpoint follows RFC 5785
- Currently two required documents
  - `/.well-known/oada-configuration` - discover OADA base and authorization URI's
  - `/.well-known/openid-configuration` - discover OpenId Connect endpoints to initiate a federated idenity assertion

/.well-known

- OADA's `/.well-known` allows an API consumer to find the base URI for a particular domain's OADA API as well as discover the needed details to authorize users.
- This endpoint follows RFC 5785
- Currently two required documents
  - `/.well-known/oada-configuration` - discover OADA base and authorization URI's
  - `/.well-known/openid-configuration` - discover OpenId Connect endpoints to initiate a federated idenity assertion

Example `/.well-known/oada-configuration` document

```
{
    "authorizationEndpoint": "http://id.openag.io/connect/authorize",
    "tokenEndpoint": "http://api.agcloud.com/connect/token",
    "OADABaseUri": "https://api.agcloud.com"
}
```

Example `/.well-known/openid-configuration` document


```
{
    "issuer": "https://api.agcloud.com",
    "authorization_endpoint": "https://api.agcloud.com/connect/authorize",
    "token_endpoint": "https://api.agcloud.com/connect/token",
    "token_endpoint_auth_methods_supported": ["client_secret_basic", "private_key_jwt"],
    "token_endpoint_auth_signing_alg_values_supported": ["RS256", "ES256"],
    "userinfo_endpoint": "https://api.agcloud.com/connect/userinfo",
    "check_session_iframe": "https://api.agcloud.com/connect/check_session",
    "end_session_endpoint": "https://api.agcloud.com/connect/end_session",
    "jwks_uri": "https://api.agcloud.com/jwks.json",
    "registration_endpoint": "https://api.agcloud.com/connect/register",
    "scopes_supported": ["openid", "resources", "groups", "config"],
    "response_types_supported": ["code", "code id_token", "id_token", "token id_token"],
    "acr_values_supported": ["urn:mace:incommon:iap:silver","urn:mace:incommon:iap:bronze"],
    "subject_types_supported": ["public", "pairwise"],
    "userinfo_signing_alg_values_supported": ["RS256", "ES256", "HS256"],
    "userinfo_encryption_alg_values_supported": ["RSA1_5", "A128KW"],
    "userinfo_encryption_enc_values_supported": ["A128CBC-HS256", "A128GCM"],
    "id_token_signing_alg_values_supported": ["RS256", "ES256", "HS256"],
    "id_token_encryption_alg_values_supported": ["RSA1_5", "A128KW"],
    "id_token_encryption_enc_values_supported": ["A128CBC-HS256", "A128GCM"],
    "request_object_signing_alg_values_supported": ["none", "RS256", "ES256"],
    "display_values_supported": ["page", "popup"],
    "claim_types_supported": ["normal", "distributed"],
    "claims_supported": ["sub", "iss", "auth_time", "name", "picture", "email", "account"],
    "claims_parameter_supported": true,
    "service_documentation": "http://api.agcloud.com/connect/service_documentation.html",
    "ui_locales_supported": ["en-US"]
}
```

Example Use Cases

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.
- **POST**
  - An HTTP method to upload a resource when the final URI is not known by the client.

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.
- **POST**
  - An HTTP method to upload a resource when the final URI is not known by the client.
    - For example, you POST to /resources to create a new resource. The server generates a new resourceId and serves the uploaded data from a new URI `/resources/resourceId`.

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.
- **POST**
  - An HTTP method to upload a resource when the final URI is not known by the client.
    - For example, you POST to /resources to create a new resource. The server generates a new resourceId and serves the uploaded data from a new URI `/resources/resourceId`.
- **PUT**
  - An HTTP method to upload a resource when the final URI is already known by the client.

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.
- **POST**
  - An HTTP method to upload a resource when the final URI is not known by the client.
    - For example, you POST to /resources to create a new resource. The server generates a new resourceId and serves the uploaded data from a new URI `/resources/resourceId`.
- **PUT**
  - An HTTP method to upload a resource when the final URI is already known by the client.
    - For example, you PUT to a resource's permission document, `/resources/{resourceId}/permission` to modified permissions beacuse the full URI is known.

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.
- **POST**
  - An HTTP method to upload a resource when the final URI is not known by the client.
    - For example, you POST to /resources to create a new resource. The server generates a new resourceId and serves the uploaded data from a new URI `/resources/resourceId`.
- **PUT**
  - An HTTP method to upload a resource when the final URI is already known by the client.
    - For example, you PUT to a resource's permission document, `/resources/{resourceId}/permission` to modified permissions beacuse the full URI is known.
- **PATCH**
  - An HTTP method to upload a partial change to a document at a URI that is already known by the client.

More Terms - Just in case

- **GET**
  - An HTTP method to download a resource at a given URI.
- **POST**
  - An HTTP method to upload a resource when the final URI is not known by the client.
    - For example, you POST to /resources to create a new resource. The server generates a new resourceId and serves the uploaded data from a new URI `/resources/resourceId`.
- **PUT**
  - An HTTP method to upload a resource when the final URI is already known by the client.
    - For example, you PUT to a resource's permission document, `/resources/{resourceId}/permission` to modified permissions beacuse the full URI is known.
- **PATCH**
  - An HTTP method to upload a partial change to a document at a URI that is already known by the client.
- **DELETE**
  - An HTTP method to delete a URI from existence.

Use Case Characters

- **Frank**
  - Is a farmer.

Use Case Characters

- **Frank**
  - Is a farmer.
  - Stores his data in agcloud.com.

Use Case Characters

- **Frank**
  - Is a farmer.
  - Stores his data in agcloud.com.
  - Uses a federated identity from agidentity.com.

Use Case Characters

- **Frank**
  - Is a farmer.
  - Stores his data in agcloud.com.
  - Uses a federated identity from agidentity.com.
  - Has an OADA compliant telematics device.

Use Case Characters

- **Frank**
  - Is a farmer.
  - Stores his data in agcloud.com.
  - Uses a federated identity from agidentity.com.
  - Has an OADA compliant telematics device.
  - Has OADA compliant apps on his Android tablet.

Use Case Characters

- **Frank**
  - Is a farmer.
  - Stores his data in agcloud.com.
  - Uses a federated identity from agidentity.com.
  - Has an OADA compliant telematics device.
  - Has OADA compliant apps on his Android tablet.

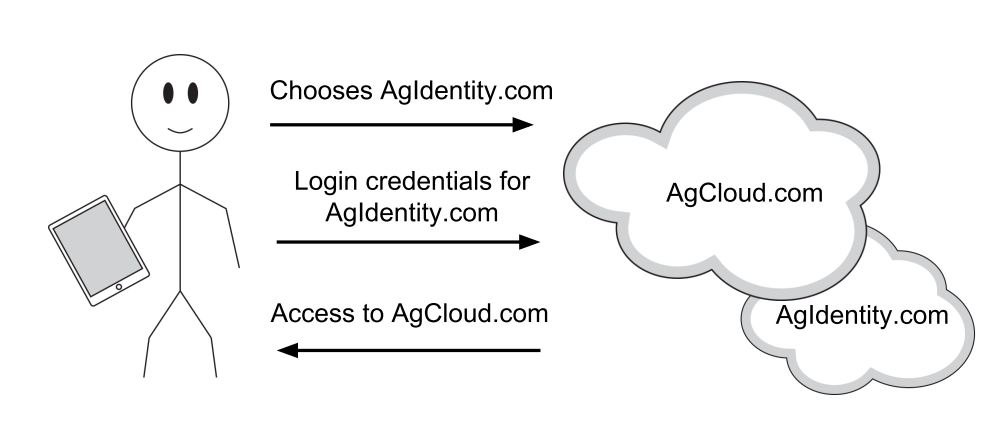
- **Andy**
  - Is a agronomist.

Use Case Characters

- **Frank**
  - Is a farmer.
  - Stores his data in agcloud.com.
  - Uses a federated identity from agidentity.com.
  - Has an OADA compliant telematics device.
  - Has OADA compliant apps on his Android tablet.

- **Andy**
  - Is a agronomist.
  - Wants to access Frank's data at agcloud.com.

Federated Login Use Case



Frank logs into his agcloud.com OADA account with an OADA compliant Android app using his agidentity.com federated identity.

To begin the process Frank's app discovers the authorization endpoints and agcloud.com's OADA base URI by querying the well-known oada-configuration URI.

To begin the process Frank's app discovers the authorization endpoints and agcloud.com's OADA base URI by querying the well-known oada-configuration URI.

Request

```
GET /.well-known/oada-configuration HTTP/1.1
Host: api.agcloud.com
Accept: application/json
```

To begin the process Frank's app discovers the authorization endpoints and agcloud.com's OADA base URI by querying the well-known oada-configuration URI.

Request

```
GET /.well-known/oada-configuration HTTP/1.1
Host: api.agcloud.com
Accept: application/json
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "authorizationEndpoint": "http://api.agcloud.com/authorize",
    "tokenEndpoint": "http://api.agcloud.com/token",
    "OADABaseUri": "https://api.agcloud.com"
}
```

The second step is to start the OAuth 2.0 procedure by making an implicit flow request to the specified `authorizationEndpoint`. Implicit flow is used because it makes most sense for an Android app. However, other OAuth 2.0 flows could be used.

The second step is to start the OAuth 2.0 procedure by making an implicit flow request to the specified `authorizationEndpoint`. Implicit flow is used because it makes most sense for an Android app. However, other OAuth 2.0 flows could be used.

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
&redirect_uri=https%3A%2F%2Flocalhost HTTP/1.1
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

The second step is to start the OAuth 2.0 procedure by making an implicit flow request to the specified `authorizationEndpoint`. Implicit flow is used because it makes most sense for an Android app. However, other OAuth 2.0 flows could be used.

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
&redirect_uri=https%3A%2F%2Flocalhost HTTP/1.1
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html>
...
</html>
```

The second step is to start the OAuth 2.0 procedure by making an implicit flow request to the specified `authorizationEndpoint`. Implicit flow is used because it makes most sense for an Android app. However, other OAuth 2.0 flows could be used.

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
&redirect_uri=https%3A%2F%2Flocalhost HTTP/1.1
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html>
...
</html>
```

Agcloud.com's response is an HTML web page that challenges Frank to login with local user credentials or with an OADA federated account.

The second step is to start the OAuth 2.0 procedure by making an implicit flow request to the specified `authorizationEndpoint`. Implicit flow is used because it makes most sense for an Android app. However, other OAuth 2.0 flows could be used.

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
&redirect_uri=https%3A%2F%2Flocalhost HTTP/1.1
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html>
...
</html>
```

Agcloud.com's response is an HTML web page that challenges Frank to login with local user credentials or with an OADA federated account.

Frank elects to login with the OADA federated identity `frank@agidentity.com`. If agidentity.com's OpenId Connect endpoint is unknown to agcloud.com then it queries `agidentity.com/.well-known/openid-configuration` to discover the correct URL.

Once the correct URL is known, agcloud.com generates a redirect response to the OpenID Connect endpoint. This begins the OpenID Connect flow.

Once the correct URL is known, agcloud.com generates a redirect response to the OpenID Connect endpoint. This begins the OpenID Connect flow.

Response *Line breaks in URI are for formating purposes only*

```
HTTP/1.1 302 Found
Location: https://agidentity.com/authorize?response_type=id_token%20token
&client_id=s6BhdRkqt3&redirect_uri=https%3A%2F%2Fapi.agcloud.com%2Fcb&scope=openid%20profile
&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
```

Once the correct URL is known, agcloud.com generates a redirect response to the OpenID Connect endpoint. This begins the OpenID Connect flow.

Response *Line breaks in URI are for formating purposes only*

```
HTTP/1.1 302 Found
Location: https://agidentity.com/authorize?response_type=id_token%20token
&client_id=s6BhdRkqt3&redirect_uri=https%3A%2F%2Fapi.agcloud.com%2Fcb&scope=openid%20profile
&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
```

Therefore, Frank's user-agent makes the redirect request to agidentity.com

Once the correct URL is known, agcloud.com generates a redirect response to the OpenID Connect endpoint. This begins the OpenID Connect flow.

Response *Line breaks in URI are for formating purposes only*

```
HTTP/1.1 302 Found
Location: https://agidentity.com/authorize?response_type=id_token%20token
&client_id=s6BhdRkqt3&redirect_uri=https%3A%2F%2Fapi.agcloud.com%2Fcb&scope=openid%20profile
&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
```

Therefore, Frank's user-agent makes the redirect request to agidentity.com

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=code&client_id=s6bhdrkqt3
&redirect_uri=https%3a%2f%2fapi.agcloud.com%2fcb&scope=openid%20profile http/1.1
Host: agidentity.com
Accept: text/html,application/xhtml+xml,application/xml
```

Once the correct URL is known, agcloud.com generates a redirect response to the OpenID Connect endpoint. This begins the OpenID Connect flow.

Response *Line breaks in URI are for formating purposes only*

```
HTTP/1.1 302 Found
Location: https://agidentity.com/authorize?response_type=id_token%20token
&client_id=s6BhdRkqt3&redirect_uri=https%3A%2F%2Fapi.agcloud.com%2Fcb&scope=openid%20profile
&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
```

Therefore, Frank's user-agent makes the redirect request to agidentity.com

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=code&client_id=s6bhdrkqt3
&redirect_uri=https%3a%2f%2fapi.agcloud.com%2fcb&scope=openid%20profile http/1.1
Host: agidentity.com
Accept: text/html,application/xhtml+xml,application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html>
...
</html>
```

Once the correct URL is known, agcloud.com generates a redirect response to the OpenID Connect endpoint. This begins the OpenID Connect flow.

Response *Line breaks in URI are for formating purposes only*

```
HTTP/1.1 302 Found
Location: https://agidentity.com/authorize?response_type=id_token%20token
&client_id=s6BhdRkqt3&redirect_uri=https%3A%2F%2Fapi.agcloud.com%2Fcb&scope=openid%20profile
&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
```

Therefore, Frank's user-agent makes the redirect request to agidentity.com

Request *Line breaks in URI are for formating purposes only*

```
GET /authorize?response_type=code&client_id=s6bhdrkqt3
&redirect_uri=https%3a%2f%2fapi.agcloud.com%2fcb&scope=openid%20profile http/1.1
Host: agidentity.com
Accept: text/html,application/xhtml+xml,application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html>
...
</html>
```

agidentity.com's response is also an HTML web page that challenges Frank to login with his local user credentials (the federated identity).

When Frank successfully logs in and confirms that agcloud.com may access his profile information, e.g., real name, email, etc. agidentity.com issues a redirect response back to agcloud.com.

When Frank successfully logs in and confirms that agcloud.com may access his profile information, e.g., real name, email, etc. agidentity.com issues a redirect response back to agcloud.com.

Response

```
HTTP/1.1 302 Found
Location: https://agcloud.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
```

When Frank successfully logs in and confirms that agcloud.com may access his profile information, e.g., real name, email, etc. agidentity.com issues a redirect response back to agcloud.com.

Response

```
HTTP/1.1 302 Found
Location: https://agcloud.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
```

Therefore, Frank's user-agent makes the associated request and so ends the OpenId Connect flow and resumes the original OAuth 2.0 from the users perspective.

When Frank successfully logs in and confirms that agcloud.com may access his profile information, e.g., real name, email, etc. agidentity.com issues a redirect response back to agcloud.com.

Response

```
HTTP/1.1 302 Found
Location: https://agcloud.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
```

Therefore, Frank's user-agent makes the associated request and so ends the OpenId Connect flow and resumes the original OAuth 2.0 from the users perspective.

Request

```
GET /cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

When Frank successfully logs in and confirms that agcloud.com may access his profile information, e.g., real name, email, etc. agidentity.com issues a redirect response back to agcloud.com.

Response

```
HTTP/1.1 302 Found
Location: https://agcloud.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
```

Therefore, Frank's user-agent makes the associated request and so ends the OpenId Connect flow and resumes the original OAuth 2.0 from the users perspective.

Request

```
GET /cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

Now agcloud.com communicates with agidentity.com using standard OpenID Connect protocol to receive an `id_token` that asserts Frank's identity and a document that contains his profile information.

When Frank successfully logs in and confirms that agcloud.com may access his profile information, e.g., real name, email, etc. agidentity.com issues a redirect response back to agcloud.com.

Response

```
HTTP/1.1 302 Found
Location: https://agcloud.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
```

Therefore, Frank's user-agent makes the associated request and so ends the OpenId Connect flow and resumes the original OAuth 2.0 from the users perspective.

Request

```
GET /cb?code=SplxlOBeZQQYbYS6WxSbIA&state=af0ifjsldkj
Host: api.agcloud.com
Accept: text/html,application/xhtml+xml,application/xml
```

Now agcloud.com communicates with agidentity.com using standard OpenID Connect protocol to receive an `id_token` that asserts Frank's identity and a document that contains his profile information.

If the `id_token` is valid then agcloud.com considers the authorization challenge successfully completed for the identity `frank@agidentity.com` and generates an OAuth 2.0 token.

As a result Agcloud.com responds with redirect that includes the token generated token.

As a result Agcloud.com responds with redirect that includes the token generated token.

Response

```
HTTP/1.1 302 Found
Location: https://localhost#access_token=SlAV32hkKG&token_type=bearer&expires_in=3600
&state=af0ifjsldkj
```

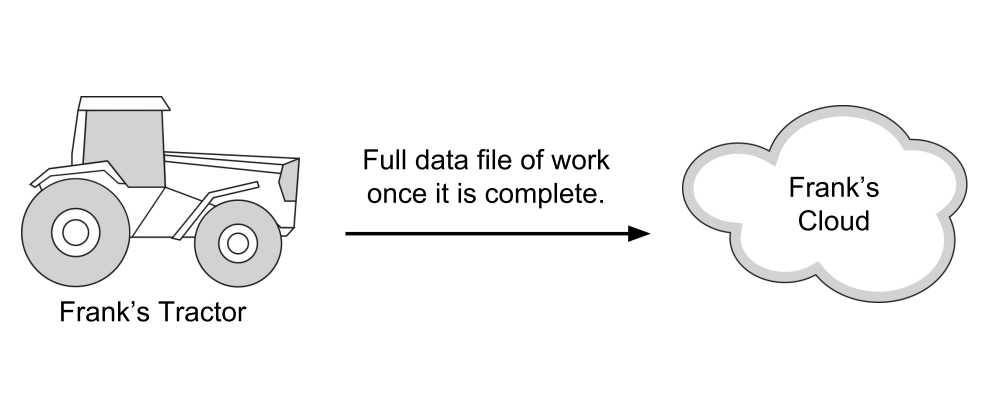
As a result Agcloud.com responds with redirect that includes the token generated token.

Response

```
HTTP/1.1 302 Found
Location: https://localhost#access_token=SlAV32hkKG&token_type=bearer&expires_in=3600
&state=af0ifjsldkj
```

Finally, the Android app parses the `access_token` from the Location RUI fragment and stores it for later use.

Resource Upload Use Case



Frank's telematics device records yield measurements through the entire day into a GeoJSON file. While Frank finishes his work for the day he touches the "sync to OADA cloud" button. As a result, Frank's telematics device uploads the GeoJSON file as a new resource to Frank's agcloud.com.

Assumptions

- The telematics device already has authorization and a valid token.

To both create a new resource and upload the associated data simultaneously a POST request with `Content-Type` equal to `multipart/form-data` is made. The JSON resource document is sent with the form-data name `resource` and the data with form-data name `data`.

To both create a new resource and upload the associated data simultaneously a POST request with `Content-Type` equal to `multipart/form-data` is made. The JSON resource document is sent with the form-data name `resource` and the data with form-data name `data`.

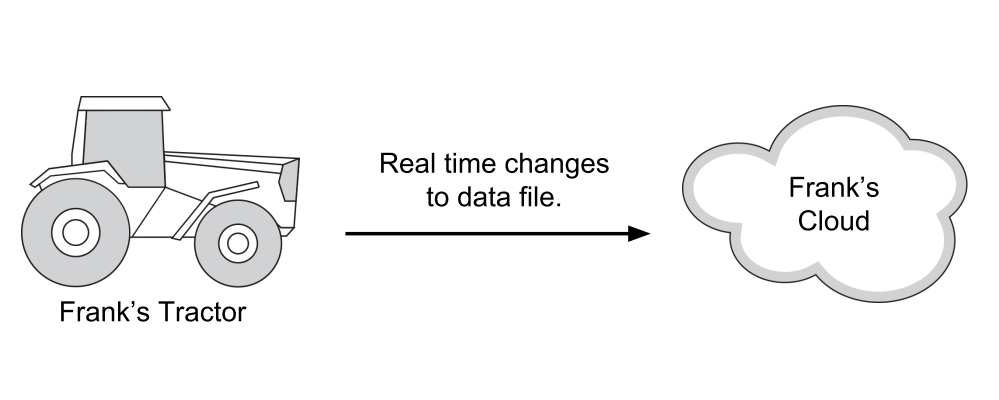
Request

```
POST /resources HTTP/1.1
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Content-Type: multipart/form-data; boundary=AaB03x
Content-Length: 496

--AaB03x
Content-Disposition: form-data; name="resource"
{
    "title": "Frank's Yeild"
}

--AaB03x
Content-Disposition: form-data; name="data"
Content-Type: applcation/vnd.oada.yield+json
{
    "totalYield": {
        "value": 5.6,
        "unit": "bushel"
    },
    "type": "FeatureCollection",
    "bbox": [40.42426718029455, 40.42429718029455, -86.841822197086, -86.841852197086],
    "features": [{
            ....
    }
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: applcation/json
Location: /resources/ixm24ws
Etag: "9083423jkadfu9382x"

{
    "href": "https://api.agcloud.com/resources/ixm24ws",
    "etag": "alsjfadksja9388x7d",
    "title": "Frank's Yield",
    "mimeType": "application/vnd.oada.yield+json",
    "created": "1985-04-12T23:20:50.52Z",
    "createdBy": {
        "href": "https://api.agcloud.com/users/kdufe3f",
    },
    "modified": "1985-04-12T23:20:50.52Z",
    "modifiedBy": {
        "href": "https://api.agcloud.com/users/kdufe3f",
    },
    "data": {
        "href": "https://api.agcloud.com/resources/ixm24ws/data",
    },
    "meta": {
        "href": "https://api.agcloud.com/resources/ixm24ws/meta",
    },
    "format": {
        "href": "https://api.agcloud.com/resources/ixm24ws/format",
    },
    "parents": {
        "href": "https://api.agcloud.com/resources/ixm24ws/parents",
    },
    "children": {
        "href": "https://api.agcloud.com/resources/ixm24ws/children",
    },
    "permissions": {
        "href": "https://api.agcloud.com/resources/ixm24ws/permissions",
    }
```

Resource Update Use Case



Whenever Frank's tractor is on, his telematics device records the number of hours it has been running. Periodically, the telematics device updates a tractor status resource in Frank's agcloud.com with the new total runtime.

Assumptions

- The tractor status resource is already known.
- The telematics device already has authorization and a valid token.

For this use case assume the following resource already exists:

Request

```
GET /resources/kdj83mx/data
Host: api.agcloud.com
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Content-Length: 133
Etag: "686897696a7c876b7e"

{
    "hours": 1523,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -4,
        "100_hour": 46
    }
}
```

For this use case assume the following resource already exists:

Request

```
GET /resources/kdj83mx/data
Host: api.agcloud.com
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Content-Length: 133
Etag: "686897696a7c876b7e"

{
    "hours": 1523,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -4,
        "100_hour": 46
    }
}
```

Then if the telematics device needs to update the `hours` field to `1524`, then it should also update the `service_intervals` to `-5` and `46` for `50_hour` and `100_hour` respectively.

This can be accomplished several ways.

This can be accomplished several ways.

With PUT:

Request

```
PUT /resources/kdj83mx/data
Host: api.agcloud.com
Content-Type: applcation/json
Content-Length: 133
If-Match: "686897696a7c876b7e"

{
    "hours": 1524,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -5,
        "100_hour": 45
    }
}
```

Notice the `If-Match` header provides some concurrency protection.

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Content-Length: 133
Etag: "893rjdklia9w383984"

{
    "hours": 1524,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -5,
        "100_hour": 45
    }
}
```

With two separate puts to update each section of the document:

With two separate puts to update each section of the document:

Request

```
PUT /resources/kdj83mx/data/hours
Host: api.agcloud.com
Content-Type: plain/text
Content-Length: 4
If-Match: "686897696a7c876b7e"

1524
```

With two separate puts to update each section of the document:

Request

```
PUT /resources/kdj83mx/data/hours
Host: api.agcloud.com
Content-Type: plain/text
Content-Length: 4
If-Match: "686897696a7c876b7e"

1524
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Content-Length: 133
Etag: "asf9cka3a08345rjj4"

{
    "hours": 1524,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -4,
        "100_hour": 46
    }
}
```

Request

```
PUT /resources/kdj83mx/data/service_intervals
Host: api.agcloud.com
Content-Type: application/json
Content-Length: 78
If-Match: "asf9cka3a08345rjj4"

{
    "50_hour": -5,
    "100_hour": 45
}
```

Request

```
PUT /resources/kdj83mx/data/service_intervals
Host: api.agcloud.com
Content-Type: application/json
Content-Length: 78
If-Match: "asf9cka3a08345rjj4"

{
    "50_hour": -5,
    "100_hour": 45
}
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Content-Length: 133
Etag: "893rjdklia9w383984"

{
    "hours": 1524,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -5,
        "100_hour": 45
    }
}
```

With PATCH:

With PATCH:

Request

```
PATCH /resources/kdj83mx/data
Host: api.agcloud.com
Content-Type: applcation/json
Content-Length: 133
If-Match: "686897696a7c876b7e"

{
    "hours": 1524,
    "service_intervals": {
        "50_hour": -5,
        "100_hour": 45
    }
}
```

With PATCH:

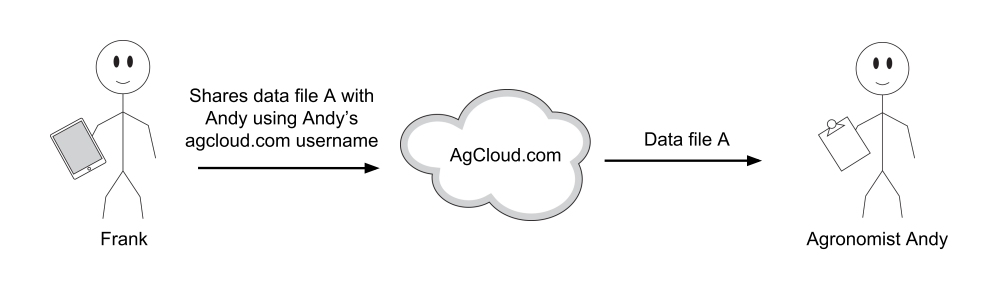
Request

```
PATCH /resources/kdj83mx/data
Host: api.agcloud.com
Content-Type: applcation/json
Content-Length: 133
If-Match: "686897696a7c876b7e"

{
    "hours": 1524,
    "service_intervals": {
        "50_hour": -5,
        "100_hour": 45
    }
}
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Content-Length: 133
Etag: "893rjdklia9w383984"

{
    "hours": 1524,
    "fuel_level": "80%",
    "service_intervals": {
        "50_hour": -5,
        "100_hour": 45
    }
}
```

Resource Sharing Use Case



Frank instructs his OADA complaint Android app to share a resource with Andy. Now Andy can access it directly with his own account.

Assumptions

- Frank's Android app already has authorization and a valid token for Frank's user agcloud.com.
- Andy's OADA application already has authorization and a valid token for Andy's user at agcloud.com.

To share `/resources/ixm24ws`, the GeoJSON yield resource we made earlier, with Andy as an owner, `userId = jdx83jx` we need to add a new entry to the resource permission document.

To share `/resources/ixm24ws`, the GeoJSON yield resource we made earlier, with Andy as an owner, `userId = jdx83jx` we need to add a new entry to the resource permission document.

Request

```
POST /resources/ixm24ws/permissions HTTP/1.1
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Content-Type: applcation/json
Content-Length: 496

{
    "user": {
        "href": "https://api.agcloud.com/users/jdx83jx"
    },
    "type": "user",
    "level": "owner"
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: applcation/json
Location: /resources/ixm24ws
Etag: "9083423jkadfu9382x"

{
    "href": "https://api.agcloud.com/resources/ixm24ws/permissions",
    "etag": "9238fasjakdfaf39f7",
    "items": [{
    "href":
        "https://api.agcloud.com/resources/ixm24ws/permissions/jfi30x9"
    }]
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: applcation/json
Location: /resources/ixm24ws
Etag: "9083423jkadfu9382x"

{
    "href": "https://api.agcloud.com/resources/ixm24ws/permissions",
    "etag": "9238fasjakdfaf39f7",
    "items": [{
    "href":
        "https://api.agcloud.com/resources/ixm24ws/permissions/jfi30x9"
    }]
}
```

Now Andy can access the resource with his identity.

Response

```
HTTP/1.1 201 Created
Content-Type: applcation/json
Location: /resources/ixm24ws
Etag: "9083423jkadfu9382x"

{
    "href": "https://api.agcloud.com/resources/ixm24ws/permissions",
    "etag": "9238fasjakdfaf39f7",
    "items": [{
    "href":
        "https://api.agcloud.com/resources/ixm24ws/permissions/jfi30x9"
    }]
}
```

Now Andy can access the resource with his identity.

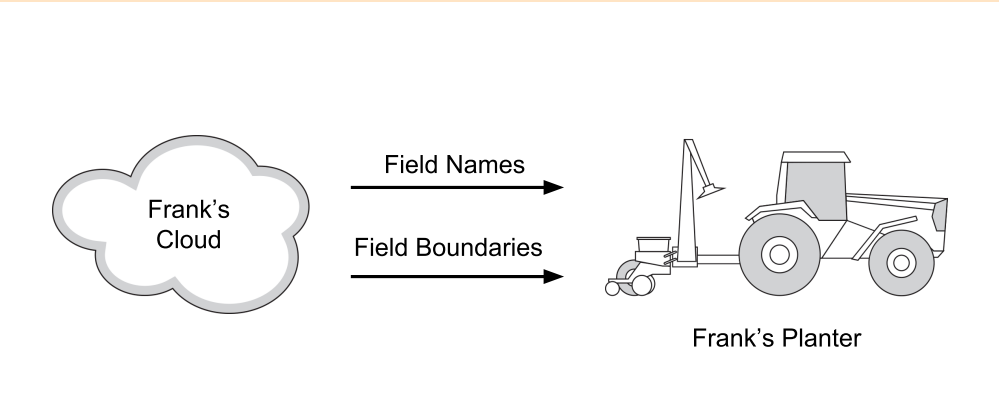
Request

```
GET /resources/ixm24ws
Host: api.agcloud.com
Authentication: Bearer kaJH38da3x
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "aodskjfoa3j9af7883"

{
    "href": "https://api.agcloud.com/resources/ixm24ws",
    "etag": "alsjfadksja9388x7d",
    "title": "Frank's Yield",
    "mimeType": "application/vnd.oada.yield+json",
    "created": "1985-04-12T23:20:50.52Z",
    "createdBy": {
        "href": "https://api.agcloud.com/users/kdufe3f",
    },
    "modified": "1985-04-12T23:20:50.52Z",
    "modifiedBy": {
        "href": "https://api.agcloud.com/users/kdufe3f",
    },
    "data": {
        "href": "https://api.agcloud.com/resources/ixm24ws/data",
    },
    "meta": {
        "href": "https://api.agcloud.com/resources/ixm24ws/meta",
    },
    "format": {
        "href": "https://api.agcloud.com/resources/ixm24ws/format",
    },
    "parents": {
        "href": "https://api.agcloud.com/resources/ixm24ws/parents",
    },
    "children": {
        "href": "https://api.agcloud.com/resources/ixm24ws/children",
    },
    "permissions": {
        "href": "https://api.agcloud.com/resources/ixm24ws/permissions",
    }
}
```

Field Discovery Use Case



Frank drives his tractor to a field and starts planting. Instead of asking Frank what field he is in, the monitor automatically discovers the current set of fields using the fields configuration on Frank's agcloud.com storage. However, the resource is in the Shape format but the monitor only understands GeoJSON. Therefore, the monitor requests a Shape to GeoJSON transformation when downloading the resource.

Assumptions

- Frank's monitor device already has authorization and a valid token.

The fields resource is located via the field configuration.

The fields resource is located via the field configuration.

Request

```
GET /config/fields
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/json
```

The fields resource is located via the field configuration.

Request

```
GET /config/fields
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "aodskjfoa3j9af7883"

{
    "href": "https://api.agcloud.com/config/prescriptions/planting",
    "etag": "jkx6yc3c7cja89434inc8ascfjdkasfjc8i7a37x",
    "items": [],
    "resource": {
        "href": "https://api.agcloud.com/resources/fd8as8c"
    }
}
```

The fields resource is located via the field configuration.

Request

```
GET /config/fields
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "aodskjfoa3j9af7883"

{
    "href": "https://api.agcloud.com/config/prescriptions/planting",
    "etag": "jkx6yc3c7cja89434inc8ascfjdkasfjc8i7a37x",
    "items": [],
    "resource": {
        "href": "https://api.agcloud.com/resources/fd8as8c"
    }
}
```

Now that the resource is known the formats document needs to be consulted to determine if the fields can be returned in an acceptable format, in this case `application/vns.oada.fields+json`.

Request

```
GET /resources/fd8as8c/formats
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/json
```

Request

```
GET /resources/fd8as8c/formats
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "qewriuquicjdkcj832"

{
    "href": "https://api.agcloud.com/resources/fd8as8c/format",
    "etag": "mnewahfau3&83djcx2",
    "transforms": {
        "application/vnd.oada.field+json": {
            "lossy": false,
            "name": "OADA GeoJSON Field Open Format",
            "openFormat": true
        },
        "application/shape": {
            "lossy": false,
            "name": "Esri Shapefile",
            "openFromat": false
        }
    }
}
```

Request

```
GET /resources/fd8as8c/formats
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "qewriuquicjdkcj832"

{
    "href": "https://api.agcloud.com/resources/fd8as8c/format",
    "etag": "mnewahfau3&83djcx2",
    "transforms": {
        "application/vnd.oada.field+json": {
            "lossy": false,
            "name": "OADA GeoJSON Field Open Format",
            "openFormat": true
        },
        "application/shape": {
            "lossy": false,
            "name": "Esri Shapefile",
            "openFromat": false
        }
    }
}
```

This resource can be transformed into the desired format, so the resource is downloaded.

Request

```
GET /resources/fd8as8c/data
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/vnd.oada.field+json
```

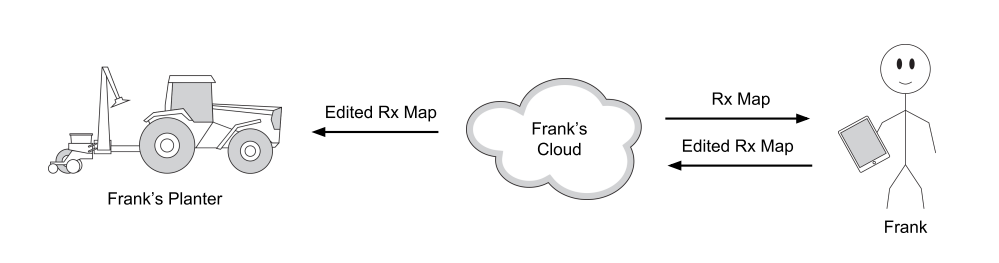
Request

```
GET /resources/fd8as8c/data
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/vnd.oada.field+json
```

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "qewriuquicjdkcj832"

{
    "type": "GeometeryCollection",
    ...
    "features": [{
            ....
    }
    ...
}
```

Resource Syncing Use Case



Frank uses his OADA compliant Android app to discover his planting prescription resource via the prescription configuration. He proceeds to edit the resource and sync it back to his agcloud.com storage. This same prescription resource was discovered and downloaded by Frank's monitor. However, the monitor periodically checks the adcloud.com storage for changes in the resource and automatically re-downloads it.

Assumptions

- Frank's monitor device already has authorization and a valid token.

In the interest of time it is assumed that both the app and the monitor have successfully discovered and downloaded the prescription planting resource with the help of `/configs/prescriptions/planting` using the techniques previously described.

In the interest of time it is assumed that both the app and the monitor have successfully discovered and downloaded the prescription planting resource with the help of `/configs/prescriptions/planting` using the techniques previously described.

Then, we also assume that the discovered resource is `/resources/ajd82mx` and the original resource Etag is `k23odjuasidfjasdkf`.

In the interest of time it is assumed that both the app and the monitor have successfully discovered and downloaded the prescription planting resource with the help of `/configs/prescriptions/planting` using the techniques previously described.

Then, we also assume that the discovered resource is `/resources/ajd82mx` and the original resource Etag is `k23odjuasidfjasdkf`.

To poll for an update a request with the `If-None-Match` is made periodically.

In the interest of time it is assumed that both the app and the monitor have successfully discovered and downloaded the prescription planting resource with the help of `/configs/prescriptions/planting` using the techniques previously described.

Then, we also assume that the discovered resource is `/resources/ajd82mx` and the original resource Etag is `k23odjuasidfjasdkf`.

To poll for an update a request with the `If-None-Match` is made periodically.

Request

```
GET /resources/fd8as8c/data
Host: api.agcloud.com
Authentication: Bearer SlAV32hkKG
Accept: application/vnd.oada.prescription.planting+json
If-None-Match: "k23odjuasidfjasdkf"
```

Typically the response will be one of following two

Typically the response will be one of following two

No changes

Response

```
HTTP/1.1 304 Not Modified
Content-Type: applcation/json
Etag: "k23odjuasidfjasdkf"
```

Typically the response will be one of following two

No changes

Response

```
HTTP/1.1 304 Not Modified
Content-Type: applcation/json
Etag: "k23odjuasidfjasdkf"
```

Avaiable changes

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "ajja97823jfaksdhfx"

{
    ...
}
```

Typically the response will be one of following two

No changes

Response

```
HTTP/1.1 304 Not Modified
Content-Type: applcation/json
Etag: "k23odjuasidfjasdkf"
```

Avaiable changes

Response

```
HTTP/1.1 200 Ok
Content-Type: applcation/json
Etag: "ajja97823jfaksdhfx"

{
    ...
}
```

Future version of OADA may consider adding push notifications so that devices do not have to continuously poll for changes to stay up-to-date.