# ONE Record API

## Reference Specification – draft 1.0

April 5, 2019

**Note on this draft**

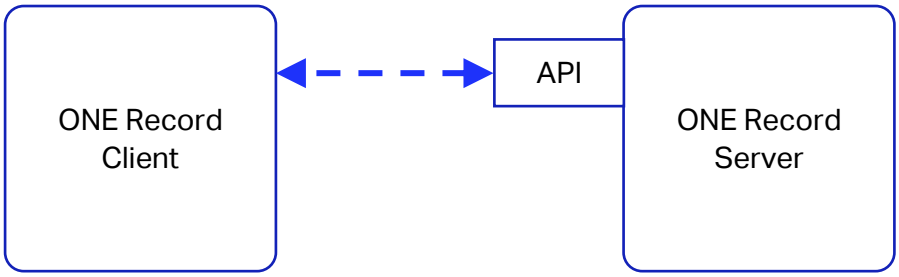
This reference specification is a draft. This means that it is subject to edits. Discussion on this specification is highly encouraged and please contact mulderh@iata.org with any comments or suggested improvements.

# Contents

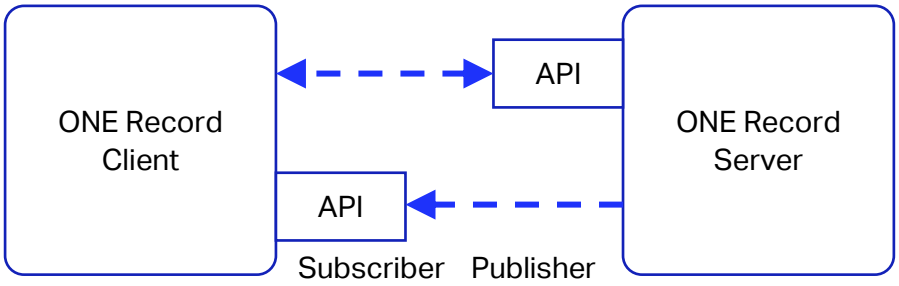
# Overview

ONE Record specifies the API and security model for data exchange over the Internet of Logistics. In fact, ONE Record is essentially the *specification* of the *Internet of Logistics*. This Internet of Logistics or IoL is the collection of all ONE Record Clients and Servers.

In this Internet of Logistics companies can exchange data as needed. They can host and publish Logistics Objects on ONE Record Servers and their partners can access these Logistics Objects using ONE Record Clients. Logistics Objects are also created or updated using this same ONE Record Server API.




In order to optimize the data flows in this Internet of Logistics, ONE Record provides for a Publish & Subscribe model. This requires that the ONE Record Clients implement a subscription API to which the ONE Record Server can publish new and updated Logistics Objects

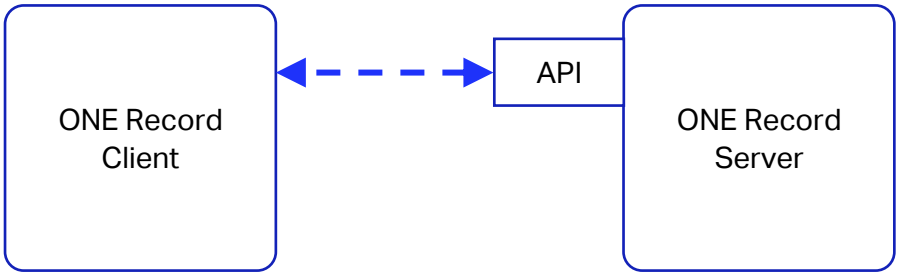



This document is in three parts. The first part specifies the ONE Record Server API, the second part concerns the security model and the third part covers the Publish-Subscribe model which includes the ONE Record Client API.

# ONE Record Server API

The ONE Record Server API is a REST based API that supports the following operations:

- Create Logistics Objects
- Read Logistics Objects
- Patch Processing Updates to Logistics Objects




## Logistics Object ID

A Logistics Object can be identified by a Logistics Objects ID. A LO ID can be any URL which by definition is unique. An example of a LO ID could be for example:

`https:// {ORS Domain} / {license plate} / {unique id to identify LO}`

where

`{ORS Domain}` - The domain name associated with the ONE Record Server e.g. www.onerecordcargo.org

`{license plate}` - The company identifier for this ONE Record Server, e.g. `my_airline`

`{unique ID to identify LO}` - An identifier for the Logistics Object that is unique at least for this company.

An example of a LO ID is: `https://www.onerecordcargo.org/my_airline/airwaybill_123-12345678`

The LO ID should be URL friendly, i.e. avoid unsafe characters that include the blank/empty spaces and " < > # % { } | \ ^ ~ [ ] `.

## Create Logistics Object

Publishes a Logistics Object resource to a ONE Record Server.

The user creating a Logistic Object must have authorization to create Logistics Objects and must belong to the company that is identified by the license plate in the LO ID.

### Request

HTTP Request type: **POST**

## HTTP Headers

The following HTTP header parameters MUST be present in the POST request:

| | |
|---|---|
| **Authentication** | A valid Bearer Token that is provided by the Identity and Authentication Provider. Refer *ONE Record security* |
| **Accept** | The content type that you want the HTTP response to be formatted in. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |
| **Content-Type** | The content type that is contained with the HTTP body. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |

## HTTP Body

The HTTP body MUST be a valid supported Logistics Object in the format as specified by the Content-Type in the header. The following is a list of some of the types of Logistics Objects that are currently supported by the ORS-API:

- AirWaybill
- Booking
- HousManifest
- HouseWaybill

The full specification of these Logistics Objects is here: https://github.com/IATA-Cargo/ONE-Record/

There is a very useful ontology manager tool for browsing this ontology here: https://tcfplayground.org/pouch/

This ontology is extended on an ongoing basis and more freight data structures will added regularly.

## Response

| Code | Description | Response Body |
|---|---|---|
| 201 | Logistics Object has been published to the Internet of Logistics. | No body required |
| 400 | Invalid Logistics Object | Error model[1] |
| 401 | Not authenticated or expired token | Error model |
| 403 | Not authorized to publish the Logistics Object to the Internet of Logistics | Error model |
| 415 | Unsupported Content Type | Error model |

[1] Error model to defined

# Read Logistics Object

Retrieves a Logistics Object resource from a ONE Record Server.

The user performing the GET request must belong to a company that has been given access to the Logistics Object.

## Request

HTTP Request type: **GET**

## HTTP Headers

The following HTTP header parameters MUST be present in the POST request:

| | |
|---|---|
| **Authentication** | A valid Bearer Token that is provided by the Identity and Authentication Provider. Refer *ONE Record security* |
| **Accept** | The content type that you want the HTTP response to be formatted in. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |

## Response

A positive HTTP 200 response is expected to a GET request. The body of the response is expected to be the Logistics Object in the format that has been requested in the Accept header of the request.

| Code | Description | Response Body |
|---|---|---|
| 200 | The request to retrieve the Logistics Object has been successful | Logistics Object |
| 401 | Not authenticated or expired token | Error model |
| 403 | Not authorized to retrieve the Logistics Object | Error model |
| 406 | Unsupported Accept Type | Error model |

# Updates to Logistics Object

The PATCH request should be used to provide:

- Status Updates - a status update on the Logistics Object.
- Partner Access - should be used when providing access to a Logistics Object to another trusted partner in the logistics chain. Refer *Authorization*

The user performing the PATCH must belong to a company that is authorized to access to the Logistics Object.

### Request

HTTP Request type: **PATCH**

### HTTP Headers

The following HTTP header parameters MUST be present in the PATCH LO request:

| | |
|---|---|
| **Authentication** | A valid Bearer Token that is provided by the Identity and Authentication Provider. Refer *ONE Record security* |
| **Accept** | The content type that you want the HTTP response to be formatted in. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |
| **Content-Type** | The content type that is contained with the HTTP body. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |

### HTTP Body

The HTTP body MUST be a valid Logistics Object Status Update or Partner Access in the format as specified by the Content-Type in the header. The following is the list of supported PATCH request bodies:

- http://tcfassociation.com/schema/StatusUpdate[2]
- http://tcfassociation.com/schema/PartnerAccess[3]

## Response

| Code | Description | Response Body |
|---|---|---|
| 201 | The update has been successful | No body required |
| 400 | The update is invalid | Error model |
| 401 | Not authenticated or expired token | Error model |
| 403 | Not authorized to update the Logistics Object | Error model |
| 415 | Unsupported Content Type | Error model |

[2] To be specified
[3] To be specified

# ONE Record security

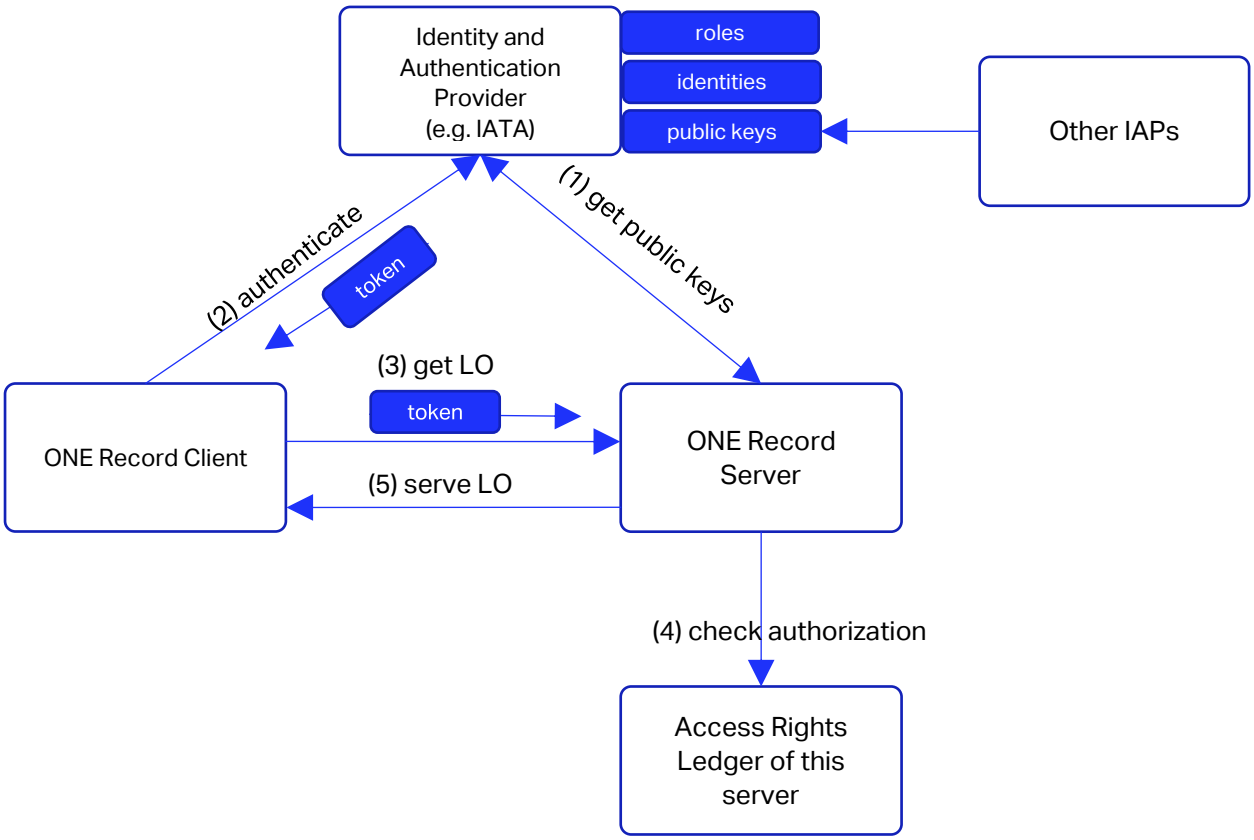
ONE Record as a basis for data sharing in the Internet of Logistics needs to ensure that data sharing is secure, i.e. that the participants sharing data are known, identified, authenticated and authorized for data access.

Since the Internet of logistics is potentially vast and may cover many different stakeholder groups, there is a need for a network of Identity and Authentication Providers (IAP) that can ensure the validity of the Internet of Logistics participants for their respective stakeholder groups. Each IAP will also hold an inventory of public keys of other IAP's that they trust. Therefore Internet of Logistics participants only need to interact with their own IAP and still be able to verify the validity of other participants even though they may be registered with another IAP.

The IAP's will use Public Key Cryptography to guarantee the authenticity of the Internet of Logistics participant whose identity and roles are in the payload of a signed Json Web Token. Once, validated, this is then used as a bearer token to access the Logistics Object.

## Identity and Authentication Providers (IAP)

Within the Internet of Logistics (IoL) there will be a need for trusted Identity and Authentication Providers (IAP).




An example of an IAP could be IATA or other industry entities that represent a group of stakeholders in logistics such as shippers and forwarders but also geographical groupings.

Each IAP would control which other IAPs that they trust. E.g IATA could trust FIATA or the IRU as an IAP and choose not to trust the state of Nutopia or even Ladonia.

Each IAP would have their own public/private key pair that they would use for signing tokens that would be provided to their members during authentication.

Each IAP would maintain the list of public keys of other IAPs that they trust. e.g IATA would have a record of FIATA's public key and any other IAP's that they trust. This list of IAP public keys would be available to its members via an API.

```
//An IAP would provide an API that would return a list of public keys that they
trust. The API to retrieve the list of public keys would return a JSON Web Key set
as per https://tools.ietf.org/html/rfc7517 e.g:


{"keys":
       [
         {
          "kty":"RSA",
          "n":
"0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAtVT86zwu1RK7aPFFxuhDR1L
6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMstn64tZ_2W-
5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-
65YGjQR0_FDW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbISD08q
NLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqbw0Ls1jF44-csFCur-
kEgU8awapJzKnqDKgw",
          "e":"AQAB",
          "alg":"RS256",
          "use": "sig",
          "key_ops": "verify",
          "kid":"dcf1"
         },
         {
          "kty":"RSA",
          "n":
"0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAtVT86zwu1RK7aPFFxuhDR1L
6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMstn64tZ_2W-
5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-
65YGjQR0_FDW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbISD08q
NLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqbw0Ls1jF44-csFCur-
kEgU8awapJzKnqDKgw",
          "e":"AQAB",
          "alg":"RS256",
          "use": "sig",
          "key_ops": "verify",
          "kid":"iata1"
         }
       ]
}
```

# Authentication

OAuth 2.0 using JWT provides the foundation for authentication within the Internet of Logistics.

IoL participants MUST authenticate against an IAP.

IoL participants would receive a JWT when authenticating against an IAP.

The JWT would be signed using the IAP's private key and would include the id (in the kid property) of the public key in the JWT header (JOSE header).

## JWT Access Tokens from an IAP

The JWT Access Tokens have a header, payload and signature.

### JOSE Header

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "dcf1"
}
```

### Payload

```
{
    "iss": "http://onerecord.iata.org", // the IAP
    "sub": "widgetco", // the user
    "exp": 1541859828, //The expiration time of the JWT
    "iat": 1516239022, // The time at which the JWT was issued. e.g.
    "jti": "dce6023b-375f-4a35-9b4a-41128bf616f" // the id of the JWT
    "https://github./schema/iolIdentifier": "https://widgetco.net/lelmeeab", //
The IOL identifier of the company the subject belongs to
    "https://github.com.org/IATA-Cargo/ONE-Record/schema/role": "SHP"
}
```

### Signature

The final part of the JWT is the signature using the private key of the IAP that is providing the token.

### Token leakage → to be addressed

In general a OAUTH 2.0 solution must avoid the leakage of tokens as much as possible as the tokens give access to resources. However in the Internet of Logistics tokens will be leaked as part of normal processing. Therefore it will be important that the JWT access tokens are bound to the sender to avoid the possibility of impersonation. A possible solution for this is described in the IETF OAUTH 2.0 Best practices - specifically the OAUTH 2.0 Token Binding proposal.

## Token Verification

When a ONE Record Server receives a request from another IoL participant with a JWT they would need to first verify the JWT before accepting the request. The verification of the JWT would include:

**Pre-Step -** As a pre-requisite the IoL participant would download and cache the list of public keys of trusted IAP's from their IAP. E.g an airline would download the list of public keys of trusted IAP's from IATA. IATA would also maintain the list of public keys of other accredited IAPs that it trusts. The cache would be refreshed on a periodic basis and could also refresh on a trigger such as receiving a signature with an id that is not in the cache.

**Verification Step -** When an IoL participant receives a request they would verify the JWT to ensure:

- That it is valid (not expired using exp property)
- It is signed by an IAP that is trusted by their provider (using the kid property to identify which public key to use to verify the signature).

# Authorization

The two aspects to authorization for a ONE Record Server are:

- Does the authenticated requestor have access to the LO? See Authorization to a Logistics Object below.
- If yes then what data does the requestor have access to within the LO? See Field level authorization within a Logistics Object.

## Authorization to a Logistics Object

Authorization to a Logistics Object can be given in two ways:

- **Explicitly in the Logistics Object** - A company can be given access to a LO by specifying the Company Identifier in the Logistics Object.
- **Using the ONE Record Server to PATCH the Logistics Object** with additional partner access. Any company that has access to the LO can cascade the trust to other companies within the IoL.

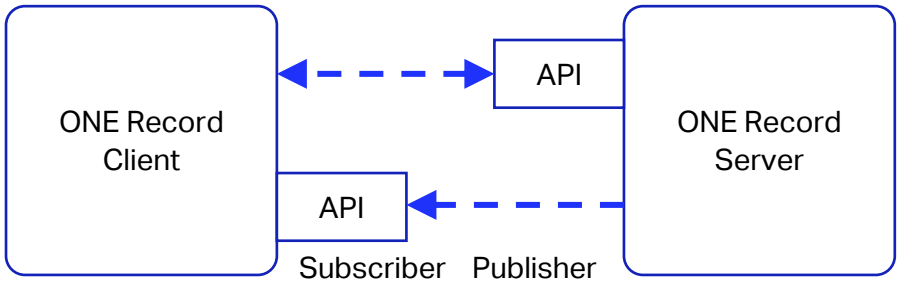
## Field level authorization within a Logistics Object

Field level authorization within a Logistics Object is possible using standardized roles. Based on the user role the ONE Record Server could make a decision to only provide certain field elements within the Logistics Object to the requestor.

# Publish & Subscribe with ONE Record

It is important that companies can receive data into their back-end systems in a near real time manner and ONE Record proposes a Publish & Subscribe pattern to allow for a distributed network of ONE Record compliant platforms.

This document describes the Publish & Subscribe model, it's implementation and the requirements of a Client Subscription API which a company must implement in order to receive Logistics Objects from ONE Record Servers through subscriptions.




## Publish & Subscribe model

The following steps describe how publish and subscribe is proposed to be implemented in the Internet of Logistics:

**Step 1 - Publish a Logistics Object**

The publish action occurs when a Logistics Object is created one a ONE Record Server. At this stage the Logistics Object is accessible via the Server API to authorized companies.

**Step 2 - Retrieve Subscription information from companies that you want to give access to**

The second step is retrieve the subscription information from the companies you want to give access to this Logistics Object. To achieve this, is the company publishing the Logistics Object must check with each of the companies it wants to give access to, whether they subscribe to these type of Logistics Objects. If they do, they provide the details of the endpoint where the Logistics Objects should be pushed to.

The prerequisite to this is that the companies must know each other through a previous exchanged Company Identifier so that the machines can ask this question during operation. These Company Identifiers may also be retrieved form common or local directories.


**Step 3 - Push to the company's ONE Record Clients**

Once the subscription information is received the publisher would push the Logistics Object to the intended ONE Record Client using the details provided. If Client Subscription API (server) was not available at the time then the publisher would need to queue and retry to publish the Logistics Object over a certain time period.

## The ONE Record Company Identifier

The ONE Record Company Identifier is a unique identifier for your company in the internet of logistics. It must be a URL that is unique to your company. An example of a Company Identifier is given below:

```
https:// {Server Domain} / {license plate}
```

However it is more than just an identifier, it is also behaves as an address for the company that can be used to retrieve company and subscription information and should be stored in publisher's backend system.

The characteristics that make a URL a Company Identifier are the following:

- **Company Information** - If you perform an authorized GET request on the URLit will return basic company information;

- **Subscription Information** - If you perform a GET request on the URL with a topic query parameter it will return subscription information if the company subscribes to the topic (i.e. type of Logistics Object) from your company.

The Company Identifier is used in the following use cases:

- In a **Bearer Token** - It is included in the Bearer Token so that the company the user belongs to can identified;
- In a **Logistics Objects** – It is included in Logistics Objects to identify companies related to shipment.
- For **Authorization** - Included when giving companies access to Logistics Objects.

Note: Authorization to a Logistics Object can be implicitly given by including the Company Identifier in the Logistics Object or it can be explicitly given by using the PATCH request on the ORS-API.

# Get Company Information

Retrieves basic company information

## Request

HTTP Request type: **GET**

```
GET /CompanyB
Host: myonerecordserver.net
Authorization: mybearertokenbase64encoded
Accept: application/ld+json
```

## HTTP Headers

The following HTTP header parameters MUST be present in the GET company information request:

| | |
|---|---|
| **Authorization** | A valid Bearer Token |
| **Accept** | The content type that you want the HTTP response to be formatted in. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |

## Response

| Code | Description | Response Body |
|---|---|---|
| 200 | Request for company information is successful | Company (Logistics Object Model) |
| 401 | Not authenticated or expired token | Error model |
| 403 | Not authorized to retrieve company information | Error model |
| 406 | Unsupported Accept Type | Error model |

# Get Subscription Information

Retrieves subscription information if the company subscribes to your topic

## Request

HTTP Request type: **GET**

```
GET /CompanyB?topic=AirwayBill
Host: myonerecordserver.net
Authorization: mybearertokenbase64encoded
Accept: application/ld+json
```

## HTTP Headers

The following HTTP header parameters MUST be present in the GET subscription information request:

| | |
|---|---|
| **Authorization** | A valid Bearer Token |
| **Accept** | The content type that you want the HTTP response to be formatted in. Valid content types include:<br>• `application/x-turtle or text/turtle`<br>• `application/ld+json` |

## Response

| Code | Description | Response Body |
|---|---|---|
| 200 | Request for subscription information is successful | Subscription (Logistics Object Model) |
| 204 | Request has been successful but the company does not subscribe | No response body |
| 401 | Not authenticated or expired token | Error model |
| 403 | Not authorized to retrieve company information | Error model |
| 406 | Unsupported Accept Type | Error model |

# Subscriptions

In order to receive Logistics Objects automatically you must setup a subscription in the relevant ONE Record Server. The subscription information includes:

1. The Company Identifier of the company you want to subscribe to
2. The Logistics Object type that you want to subscribe to
3. The endpoint address of the Client Subscription API where you want to receive Logistics Objects. (Refer requirements of your endpoint below.)
4. The content type you want to receive.
5. Either a secret or API Key that ensures that only companies with this subscription information can post to your endpoint.
6. You must also specify if you want to receive updates on that Logistics Object.

**Subscription Model**

```
{
    "@context": {
        "iata": {
            "@id": " https://github.com/IATA-Cargo/ONE-Record/schema /"
        }
    },
    "@id": "https://myORS.net/mySubscriptionToYourAirwayBill",
    "@type": "https://github.com/IATA-Cargo/ONE-Record/schema/Subscription",
    "iata:subscribedTo": "https://youronerecordserver.net/yourCompany",
    "iata:subscriptionSecret": "C89583BA9B1FEEAB25F715A3BA2F3",
    "iata:topic": " https://github.com/IATA-Cargo/ONE-Record/schema/ShippersInstruction",
    "iata:topicContentType": "application/ld+json",
    "iata:topicEndPoint": "https://myORCfI",
    "iata:statusSub": {
        "@id": "https:// myonerecordclient.net/AirwayBillStatusUpdates",
        "@type": "https://github.com/IATA-Cargo/ONE-Record//schema/Subscription",
        "iata:subscribeToStatusUpdates": "true",
        "iata:subscriptionSecret": "C89583BA9B1FEEAB25F715A3BA2F3",
        "iata:topicContentType": "application/ld+json",
        "iata:topicEndPoint": "https:// myonerecordclient.net/mysubsendpoint"
    }
}
```

## ONE Record Client Subscription API

The Client Subscription API is required to receive data from ONE Record Servers via a Subscription. Unlike the Server API, which can be accessed by any Internet of Logistics participant with adequate rights, the Client Subscription API is only exposed to ONE Record Servers with whom the company has set up a Subscription to agreed Logistics Objects.

The Client Subscription API:

- MUST support HTTP 1.1
- MUST support TLS 1.2
- MUST support the POST request on the end point.
- MUST expect an Logistics Object in the POST request. Note only the content types that you specify in the subscription request need to be supported.
- MUST respond with a 2XX response when it receives the Logistics Object.
- MUST verify either the HMAC signature or API key to ensure only authorized requests are processed.
    - The HMAC (https://tools.ietf.org/html/rfc6151) signature (in the header property X-Hub-Signature) with a shared subscription secret can be used to authorise the request. If the signature does not match, subscribers must locally ignore the message as invalid. Subscribers may still acknowledge this request with a 2xx response code in order to be able to process the message asynchronously and/or prevent brute-force attempts of the signature.
    - The API key (x-api-key) can also be used to authorize requests to the subscription endpoint.

The Client Subscription API can also expect to receive the following HTTP Headers:

| Header | Description | Expected |
|---|---|---|
| URI-resource | The top level URI of the Logistics Object of the resource being sent to the Client Subscriber.<br>Note: As RDF can come in any order of triples it is not always trivial to determine the top level resource ID. This header will inform the ORC of the top level resource URI | Yes, always |
| Orig-Request-Method | The HTTP request method that was used that generated this message to the ORC. Values here are the typical HTTP/REST verbs e.g. POST, PATCH | Yes, always |
| X-Hub-Signature | If a secret has been provided in the subscription, a HMAC signature would be present in this header | Optional, dependant on Subscription |
| x-api-key | If an API key is provided in the subscription, the API key would be provided in this header | Optional, dependant on subscription |
| Authorization | The ONE Record Access token of the ONE Record Server - to be discussed. This would require sender binding to make sense. There is also a discussion in that the owner of this API is not owning the security which may be an issue. The API Key or HMAC signature in contrast would be based on security information provided by the owner of the endpoint. | To be discussed. |

# Glossary

| Term | Description |
|---|---|
| Authentication | A process that validates the identity of IoL participant |
| Authorization | A process that determines whether a IoL participant is allowed to access a specific Logistics Object |
| Identity & Authentication Provider | A service that allows Internet of Logistics participants register and obtain an Public Key encrypted token identify themselves with ONE Record Servers and get access to Logistics Objects |
| Internet of Logistics | A network of ONE Record Clients and Servers that can share Logistics Objects over the internet using the ONE Record standard data model, APIs and security |
| Json Web Token (JWT) | Json specification for a token format that includes a user defined payload and the option for encryption. |
| Logistics Object | A data object that represents a meaningful entity in the logistics business. These may represent documents like air waybills but may also be more granular such as company details or a transport segment description. Logistics Objects are specified in a common data model by IATA and transport and logistics partners. |
| OAUTH2 | A protocol for delegation of authentication in a network of secure systems |
| ONE Record Client | A system that can access Logistics Objects on a ONE Record Server. This system may also have a ONE Record Subscriber API. |
| ONE Record Server | The platform that hosts Logistics Objects on a web server on behalf of one or more companies |
| ONE Record Subscriber API | A ONE Record Client API that has dedicated endpoint(s) for receiving Logistics Objects via a subscription |
| Participant | Company that access or shares data via the Internet of Logistics and that has registered with an Accredited Identity Provider and has possession of a valid certificate to prove this |
| Public Key Cryptography | An encryption technology that uses public and private keys to guarantee the authenticity of encrypted data without the need for both parties to share a common secret |
| Publisher | The Party that makes their Logistics Objects available through a ONE Record Server |
| Subscriber | The Party that subscribes to Logistics Objects in order to receive updates automatically |
| URI | In the web context, this is a URL that uniquely identifies a Logistics Object and a Host |