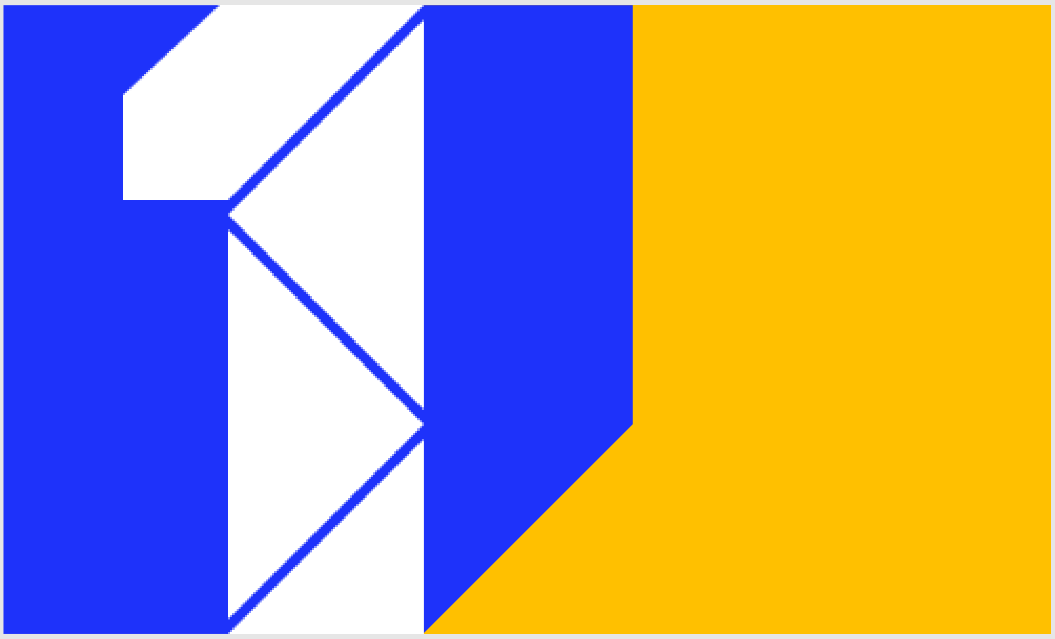
ONE Record API & Security

Reference Specification



Contents

Discussion on this specification is highly encouraged and please contact onerecord@iata.org with any comments or suggested improvements.

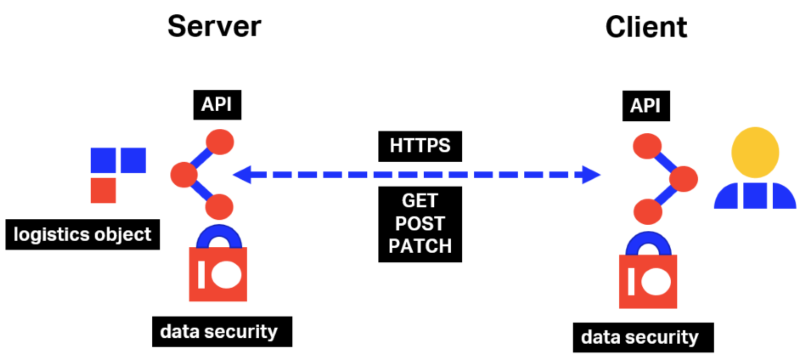
Change log

Change No.	Description	Modified by	Date Submitted	Comments
1	Add scenario 2 Add scenario 2 for Publisher/Subscriber (process initiated by the subscriber)	A. Blaj	2019/10/16	
2	Add delegation model	A. Blaj	2019/10/16	
3	Rename serverInformation to companyInformation. Add companyIdentifier	A. Blaj	2019/11/07	
4	Add notification model for pub/sub	A. Blaj	2019/12/19	
5	Add logisticsObjectRef field to audit trail model	A. Blaj	2020/01/22	
6	Add Events (status updates) specifications	A. Blaj	2020/03/09	
7	Change document layout according to IATA official template	A. Blaj	2020/04/06	
8	Add specifications for Access Control	A. Blaj	2020/04/07	
9	Add Publish & Subscribe in multi-link/multiparty scenario specifications proposal	A. Blaj	2020/04/09	
10	Add second use case for delegation	A. Blaj	2020/04/09	
11	Add specifications for versioning	A. Blaj	2020/05/19	
12	Update JSON-LD example models	A. Blaj	2020/09/07	
13	Add ONE Record Data Model & API Versioning specifications	A. Blaj	2020/10/15	
14	Update ONE Record Security specifications according to WISeKey Proof of Concept	A. Blaj	2020/11/19	

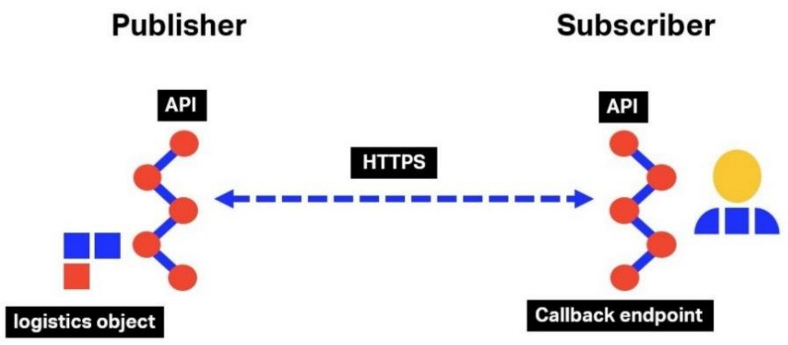
Overview

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. The **Internet of Logistics** or **IoL** is the collection of all ONE Record Clients and Servers.

In the 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 several parts: the first part specifies the ONE Record Server API, the second one includes the publisher/subscriber model, the third part presents the delegation in ONE Record, the fourth one the events creation/update, the fifth one the Access Control and the last part concerns the ONE Record security specifications.

ONE Record Data Model & API Versioning

Overview

Versioning is needed in order to manage complexity and breaking changes within the Data Model and API specifications, and to iterate faster when breaking changes are identified.

Some possible changes in the Data Model (ontology):

- Addition/Removal of Classes/Properties
- Updates of cardinality/data types
- Updates of comments/labels

Some possible changes in the API:

- Request/Response bodies changes due to the constant review of the ONE Record API specifications
- ONE Record ontology changes, which reflect on the request/response bodies
- Addition of new API endpoint
- Removal of an existing API endpoint

The solution should:

- Express a standard, clear and concise way to the client which API they are calling
- Reference the correct documentation
- Be preserved over time
- Have the ONE Record API Version in line with the ONE Record Data Model Version

ONE Record Data Model Versioning

The ONE Record ontology is developed incrementally. The ontology has several releases, which are stored in the GitHub repository.

Data Model Versioning:

- Each release should have a name that is assigned upon publication and endorsement by the COTB (e.g. 1.0)
- The content of each release is based on the discussions with the domain experts and the discussions hosted on the GitHub site
- No need for versioning at class/property level
- Use the **owl:versionInfo** annotation at the ontology level for expressing the version
- Make sure that the release notes of every version of the ontology contain enough information about what is had changed between the last version and the current one
- First version of the standard (Aug 2020) - Endorsed version in Mar2020 = 1.0
- Use standard approach for minor/major versions
- At the moment, Deprecated Property/Classes annotations are not used. To be used at a later stage if needed.

ONE Record API Versioning

The versioning for the ONE Record API is done through **route versioning**:

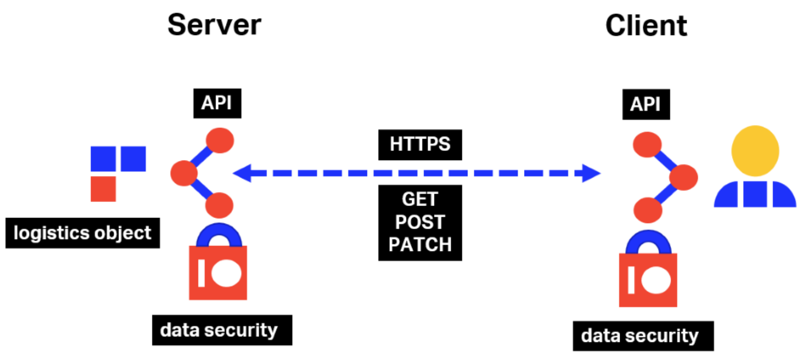
- The version is incremented when API specifications are endorsed by the COTB

- First version of the standard (Aug 2020) - Endorsed version in Mar2020 = 1.0
- If no version is specified, the latest should be returned and bind to a specific version of the API
- When a version is obsolete or not supported anymore, the client should be redirected to the latest API version, through the **Location** header and 302 HTTP status.

ONE Record 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

Every Logistics Object is identified by a **Logistics Objects ID**. A Logistics Object ID (LOID) can be any URL which is unique. An example of a LOID could be for example:

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

Field	Description
ORS Domain	The domain name associated with the ONE Record Server e.g. `onerecordcargo.org`
License plate	The company identifier for this ONE Record Server, e.g. `myAirline`
Unique ID to identify LO	An identifier for the Logistics Object that is unique at least for this server. An example of a LOID is: `https://onerecordcargo.org/myAirline/Waybill_123-12345678` The LOID should be URL friendly, i.e. avoid unsafe characters that include the blank/empty spaces and " < > # % { } | \ ^ ~ [ ] `.

Create Logistics Object (POST)

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 LOID.

Request

HTTP Request type: **POST**

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json
Content-Type	The content type that is contained with the HTTP body. Valid content types include: ▪ application/x-turtle or text/turtle ▪ 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 full specification of the Logistics Objects can be found in the ONE Record ontology: https://github.com/IATA-Cargo/ONE-Record/

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
401	Not authenticated	Error model
403	Not authorized to publish the Logistics Object to the server	Error model
415	Unsupported Content Type	Error model

Read Logistics Object (GET)

Retrieves a Logistics Object resource from a ONE Record Server.

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

Request

HTTP Request type: **GET**

The request URL should contain the Logistics Objects ID to be retrieved.

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ 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.

`GET` request could also return a `Link` Header with the location of the Access Control List (see Access Control List Resources). If the Logistics Object does not have an individual ACL (and therefore relies on an implicit ACL from a parent), this link header will still be present, but will return a 404. Returning this header is not mandatory.

```
Link: <http://myServer/myAirline/logisticsObject/acl>; rel="acl"
```

`GET` request could also return a `Link` Header with the location of the TimeMap for the Logistics Object (see Versioning. Returning this header is not mandatory.

```
Link: <http://myServer/myAirline/logisticsObject/timemap>; rel="timemap"
```

Code	Description	Response body
200	The request to retrieve the Logistics Object has been successful	Logistics Object
401	Not authenticated	Error model
403	Not authorized to retrieve the Logistics Object	Error model

404	Logistics Object not found	Error model

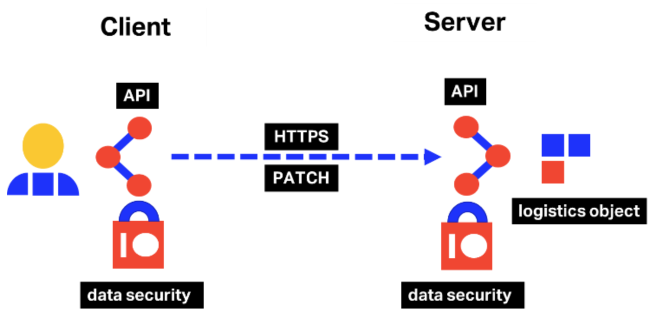
Update Logistics Object (PATCH)

Why use HTTP PATCH Method?

PATCH is a concept derived from the common understanding of the word "patch". A patch is basically a description of differences between two states of a resource.

The PATCH method is a request method supported by the HTTP protocol for making partial changes to an existing resource.

In ONE Record, an authorized client can only request partial updates of a Logistics Object and not request to replace the entire data, hence the choice of using HTTP PATCH instead of HTTP PUT. When updating a single field of a Logistics Object, sending the complete object via HTTP PUT request might also be heavy and utilizes a lot of unnecessary bandwidth. Therefore, in the ONE Record scenarios, the semantics of HTTP PATCH make a lot more sense.



The PATCH request should be used to provide updates on the Logistics Object. The user performing the PATCH must belong to a company that is authorized to access to the Logistics Object.

Linked Data PATCH Format

There are few available examples of implementations showcasing how to develop HTTP PATCH from an RDF or JSON-LD perspective and that take in account RDF particularities such as blank nodes and `rdf:List` manipulations.

RDF is very simple in its structure and does not require more than add and delete. Also, RDF triples are much simpler than either JSON or XML, so PATCH should be easier to implement as there is less functionality required. However, JSON-LD is more complex, as it is not plain-old JSON.

W3C has developed the Linked Data PATCH Format, a format for describing changes to apply to Linked Data. It defines a list of operations to be performed against a Linked Data resource, namely the addition or removal of RDF triples in a graph representing the resource.

The ONE Record API & Security group of experts has based its PATCH proposal on the W3C Linked Data PATCH Format and on the JSON-LD PATCH specifications.

PATCH in ONE Record

Update Logistics Objects with `PATCH` in ONE Record are inspired by the specifications from: https://www.w3.org/TR/ldpatch/ and https://github.com/digibib/ls.ext/wiki/JSON-LD-PATCH.

Some considerations related to updating Logistics Objects:

- Only publisher can change the Logistics Object, where publisher is the party that creates the Logistics Object on the ONE Record server. The publisher can be considered also the **owner** of the Logistics Object.
- A business partner can request a change on the Logistics Object.
- The publisher will make the Logistics Object changes based on the defined business rules.
- An **audit trail** of all the changes will be stored and can be retrieved at any moment from a dedicated endpoint on the ONE Record Server (e.g. `https://onerecordcargo.org/myAirline/waybill_123-12345678/auditTrail`).
- Logistics Objects should have a **revision number**, which is an integer to be incremented after every applied change.
- When retrieving a Logistics Object, the latest version should be returned always.
- Evaluation of a `PATCH` document occurs as a single event. Operations are sorted and processed as groups of delete and then add operations until the operations are applied, or the entire `PATCH` fails. Meaning that if a field update fails, the whole `PATCH` request is unsuccessful. No partial updates should be accepted.
- As a best practice, a `GET` Logistics object should be performed before requesting a `PATCH` in order to make sure that the change is made towards the latest version of the object.
- If a change request is rejected, the revision number of the Logistics Object is not incremented.
- Rejected update requests be kept in the audit trail of the Logistics Object.

Request

HTTP Request type: **PATCH**

The request URL should contain the Logistics Objects ID to be updated.

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: - application/x-turtle or text/turtle - application/ld+json
Content-Type	The content type that is contained with the HTTP body. Valid content types include: - application/x-turtle or text/turtle - application/ld+json

HTTP Body

The HTTP body MUST be a valid array of objects; each object represents a single operation to be applied to a target linked data resource. The format should be as specified by the `Content-Type` in the header. The following is an example of supported `PATCH` request body:

```
{
  "@type": [
    "https://onerecord.iata.org/PatchRequest"
  ],
  "https://onerecord.iata.org/PatchRequest#description": "Optional reason for the
change",
  "https://onerecord.iata.org/PatchRequest#logisticsObjectRef": "
logisticsObjectId",
  "https://onerecord.iata.org/PatchRequest#operations": [
    {
      "@type": [
        "https://onerecord.iata.org/Operation"
      ],
      "https://onerecord.iata.org/Operation#o": [
        {
          "@type": [
            "https://onerecord.iata.org/OperationObject"
          ],
          "https://onerecord.iata.org/OperationObject#datatype":
"https://www.w3.org/2001/XMLSchema#decimal",
          "https://onerecord.iata.org/OperationObject#value": "10"
        }
      ],
      "https://onerecord.iata.org/Operation#op": "del",
      "https://onerecord.iata.org/Operation#p":
"https://onerecord.iata.org/Waybill#totalPieceAndULDCount"
    },
         {
      "@type": [
        "https://onerecord.iata.org/Operation"
      ],
      "https://onerecord.iata.org/Operation#o": [
        {
          "@type": [
            "https://onerecord.iata.org/OperationObject"
          ],
          "https://onerecord.iata.org/OperationObject#datatype":
"https://www.w3.org/2001/XMLSchema#decimal",
          "https://onerecord.iata.org/OperationObject#value": "11"
        }
      ],
      "https://onerecord.iata.org/Operation#op": "add",
      "https://onerecord.iata.org/Operation#p":
"https://onerecord.iata.org/Waybill#totalPieceAndULDCount"
    },
    {
      "@type": [
        "https://onerecord.iata.org/Operation"
      ],
      "https://onerecord.iata.org/Operation#o": [
        {
          "@type": [
            "https://onerecord.iata.org/OperationObject"
          ],
          "https://onerecord.iata.org/OperationObject#datatype": "
http://www.w3.org/2001/XMLSchema#date",
          "https://onerecord.iata.org/OperationObject#value": "12019-08-18"
        }
      ],
      "https://onerecord.iata.org/Operation#op": "add",
      "https://onerecord.iata.org/Operation#p":
"https://onerecord.iata.org/Waybill#date
    }
  ],
  "https://onerecord.iata.org/PatchRequest#requestorCompanyIdentifier": "string",
  "https://onerecord.iata.org/PatchRequest#revision": "string"
}
```

- Operation objects must have exactly one "`op`" (operation) member; this value indicates which operation is to be performed. The value must be one of "`add`" or "`del`"; all other values result in an error.
- Operations objects must have exactly one "`p`", predicate member. The value of this member must be an IRI.
- Operations objects must have exactly one "`o`", object member. The value of this member must an object. This object must contain two members, a "`value`" and a "`datatype`".
- Future addition: Add provenance information? (Suggestion of ontology to use: https://www.w3.org/TR/prov-o/)

Operations:

- `add`: Add has a very simple function, it always adds new sets of statements. If a pre-existing statement exists with similar or the same characteristics, it must not be overwritten. To overwrite, a delete and an add operation must be performed.
- `del`: Del also has a very simple function, it always removes sets of statements.

Response

Code	Description	Response body
204	The update has been successful	No body required
400	The update request body is invalid	Error model
401	Not authenticated	Error model
403	Not authorized to update the Logistics Object	Error model
404	Logistics Object not found	Error model
415	Unsupported Content Type, response when the client sends a `PATCH` document format that the server does not support for the resource identified by the `Request-URI`.	Error model
422	Unprocessable request, when the server understands the `PATCH` document and the syntax of the `PATCH` document appears to be valid, but the server is incapable of processing the request.	Error model

Audit trail of Logistics Object

Request

A Logistics Object audit trail can be retrieved by performing a `GET` request to:

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

In order to retrieve the history of a Logistics Object after a given date, a query parameter should be added to the request URL as follows:

```
https:// {ORS Domain} / {license plate} / {unique id to identify LO}
/auditTrail?updatedFrom=YYYYMMDDThhmmssZ&updatedTo=YYYYMMDDThhmmssZ
```

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json

Response

```
{
  "@type": [
    "https://onerecord.iata.org/AuditTrail"
  ],
  "https://onerecord.iata.org/AuditTrail#changeRequests": [
    {
      "@type": [
        "https://onerecord.iata.org/ChangeRequest"
      ],
      "https://onerecord.iata.org/ChangeRequest#changeRequest": {
  "@type": [
    "https://onerecord.iata.org/PatchRequest"
  ],
  "https://onerecord.iata.org/PatchRequest#description": "Optional reason for the
change",
  "https://onerecord.iata.org/PatchRequest#logisticsObjectRef": "
logisticsObjectId",
  "https://onerecord.iata.org/PatchRequest#operations": [
    {
      "@type": [
        "https://onerecord.iata.org/Operation"
      ],
      "https://onerecord.iata.org/Operation#o": [
        {
          "@type": [
            "https://onerecord.iata.org/OperationObject"
          ],
          "https://onerecord.iata.org/OperationObject#datatype":
"https://www.w3.org/2001/XMLSchema#decimal",
          "https://onerecord.iata.org/OperationObject#value": "10"
        }
      ],
      "https://onerecord.iata.org/Operation#op": "del",
      "https://onerecord.iata.org/Operation#p":
"https://onerecord.iata.org/Waybill#totalPieceAndULDCount"
    },
         {
      "@type": [
        "https://onerecord.iata.org/Operation"
      ],
      "https://onerecord.iata.org/Operation#o": [
        {
          "@type": [
            "https://onerecord.iata.org/OperationObject"
          ],
          "https://onerecord.iata.org/OperationObject#datatype":
"https://www.w3.org/2001/XMLSchema#decimal",
          "https://onerecord.iata.org/OperationObject#value": "11"
        }
      ],
      "https://onerecord.iata.org/Operation#op": "add",
      "https://onerecord.iata.org/Operation#p":
"https://onerecord.iata.org/Waybill#totalPieceAndULDCount"
    },
    {
      "@type": [
        "https://onerecord.iata.org/Operation"
      ],
      "https://onerecord.iata.org/Operation#o": [
        {
          "@type": [
            "https://onerecord.iata.org/OperationObject"
          ],
          "https://onerecord.iata.org/OperationObject#datatype": "
http://www.w3.org/2001/XMLSchema#date",
          "https://onerecord.iata.org/OperationObject#value": "12019-08-18"
        }
      ],
      "https://onerecord.iata.org/Operation#op": "add",
      "https://onerecord.iata.org/Operation#p":
"https://onerecord.iata.org/Waybill#date
    }
  ],
  "https://onerecord.iata.org/PatchRequest#requestorCompanyIdentifier": "string",
  "https://onerecord.iata.org/PatchRequest#revision": "string"
},
      "https://onerecord.iata.org/ChangeRequest#companyId": "string",
      "https://onerecord.iata.org/ChangeRequest#status": "ACCEPTED",
      "https://onerecord.iata.org/ChangeRequest#timestamp": "2020-09-
07T11:55:45.768Z"
    }
  ],
  "https://onerecord.iata.org/AuditTrail#create": {
    "@type": [
      "https://onerecord.iata.org/Create"
    ],
    "https://onerecord.iata.org/Create#lo": {}
  },
  "https://onerecord.iata.org/AuditTrail#error": [
    {
      "@type": [
        "https://onerecord.iata.org/Error"
      ],
      "https://onerecord.iata.org/Error#details": [
        {
          "@type": [
            "https://onerecord.iata.org/Details"
          ],
          "https://onerecord.iata.org/Details#attribute": "string",
          "https://onerecord.iata.org/Details#code": "string",
          "https://onerecord.iata.org/Details#message": "string",
          "https://onerecord.iata.org/Details#resource": "string"
        }
      ],
      "https://onerecord.iata.org/Error#title": "string"
    }
  ],
  "https://onerecord.iata.org/AuditTrail#logisticsObjectRef": "string"
}
```

Error model

This section describes the datatype definitions used within the ONE Record API for error handling.

Error HTTP Status Codes

The following table contains a non-exhaustive list of HTTP error statuses that require an error model response:

Code	Description
400	Bad request
401	Unauthorized or expired token
403	Forbidden to perform action
404	Not Found
405	Method not allowed
415	Unsupported content type
500	Internal Server Error

The error response should always contain the `HTTP Status` and the `Content-Type` header:

```
HTTP/1.1 400 Bad Request
Content-Type: application/ld+json
```

Error Payload

The error response should contain the following fields:

```
{
  "@type": [
    "https://onerecord.iata.org/Error"
  ],
  "https://onerecord.iata.org/Error#details": [
    {
      "@type": [
        "https://onerecord.iata.org/Details"
      ],
      "https://onerecord.iata.org/Details#attribute": ".WayBillNumber ",
      "https://onerecord.iata.org/Details#code": "1234",
      "https://onerecord.iata.org/Details#message": "Waybill number could not be
dereferenced, an error occurred",
      "https://onerecord.iata.org/Details#resource":
"https://onerecord.iata.org/Waybill"
    }
  ],
  "https://onerecord.iata.org/Error#title": "Request contains invalid field "
}
```

Field	Description	Required
@id	A unique identifier of the error on the ONE Record Server.	YES
@type	Error model from ONE Record ontology.	YES

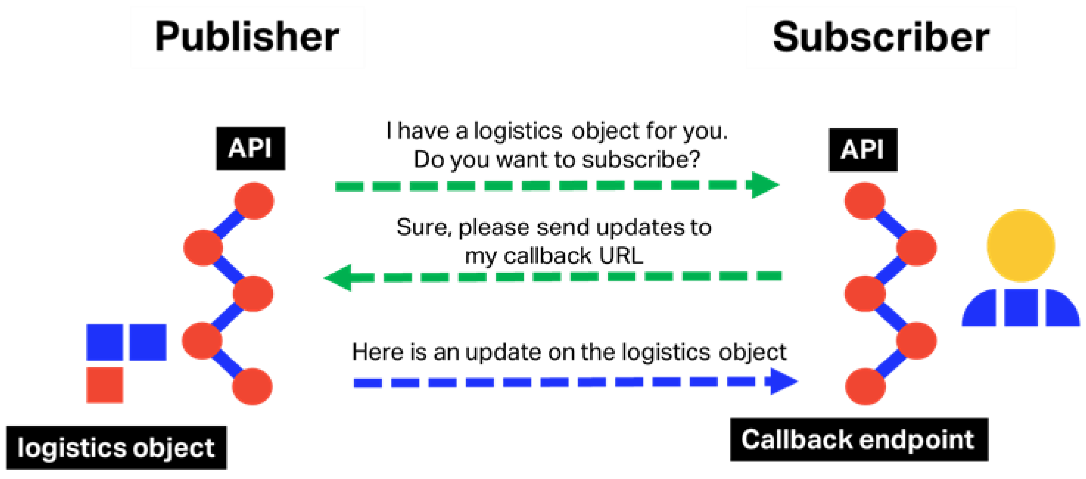
title	A short, human-readable summary of the problem that SHOULD NOT change from occurrence to occurrence of the problem, except for purposes of localization.	YES
code	A ONE Record application-specific error code expressed as a string value.	NO
attribute	Field which was not validated correctly / generated the error.	NO
resource	Schema/Class that contains the non-validated element.	NO
message	A human-readable explanation specific to this occurrence of the problem. Like *title*, this field's value can be localized.	NO

There is a list of non-exhaustive JSON-LD syntax error types (relative to 400 Bad Request error family) on the official JSON-LD API specifications website.

Publish & Subscribe with ONE Record

ONE Record proposes a Publish & Subscribe pattern to allow for a distributed network of ONE Record compliant platforms.

This chapter describes the Publish & Subscribe model, its implementation and the requirements of a Client Subscription API which a company must implement to receive Logistics Objects from ONE Record Servers through subscriptions.


Publisher
Subscriber
API
I have a logistics object for you. Do you want to subscribe?
API
Sure, please send updates to my callback URL
Here is an update on the logistics object
logistics object
Callback endpoint

Publish & Subscribe model

Publish/subscribe topics and guaranteed delivery queue

Data is exchanged between applications using a notion of topics and delivery queues. While in transit, data is kept in message queues that ensure integrity and availability of the system. Should a subscribing application go down, messages are safely retained until the recipient is ready to read them again.

For each subscriber and each topic, a message queue is maintained automatically by the publisher to keep data in until the subscriber confirms it has received a particular object.

Two scenarios were identified for initiating the publish/subscribe process. For simplicity reasons, the security part was not detailed in the following diagrams.

Scenario 1 – Publisher/Subscriber initiated by the Publisher

In the first and most usual use case, the subscription process is initiated by the publisher.

The following steps describe how publish and subscribe is proposed to be implemented in the ONE Record Internet of Logistics (**webhook model with dynamic subscription**):

Step 1 - Publish a Logistics Object

The publish action occurs when a Logistics Object is created on 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

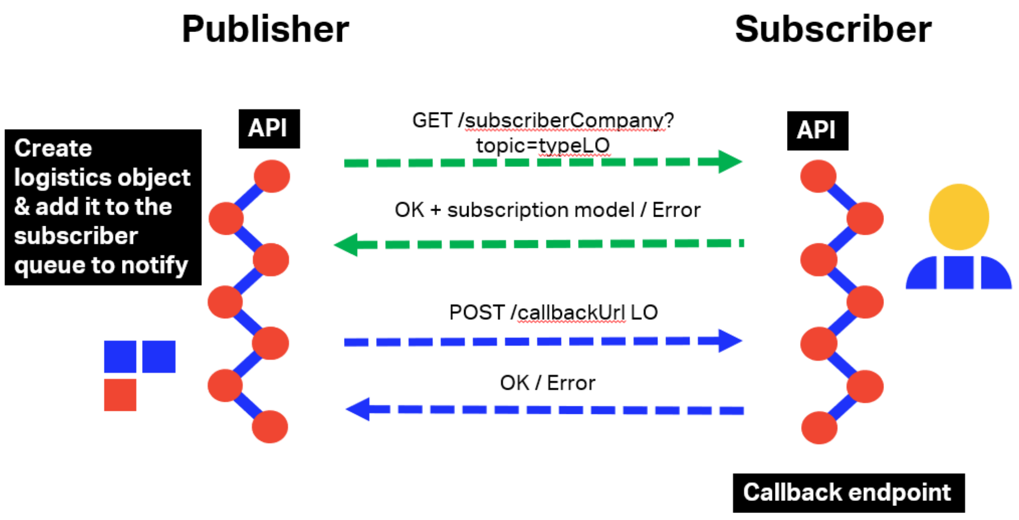
The second step is retrieving the subscription information from the companies you want to give access to this Logistics Object. To achieve this, the company publishing the Logistics Object must check with each of the companies it wants to give access to, whether they subscribe to these types of Logistics 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 from 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.

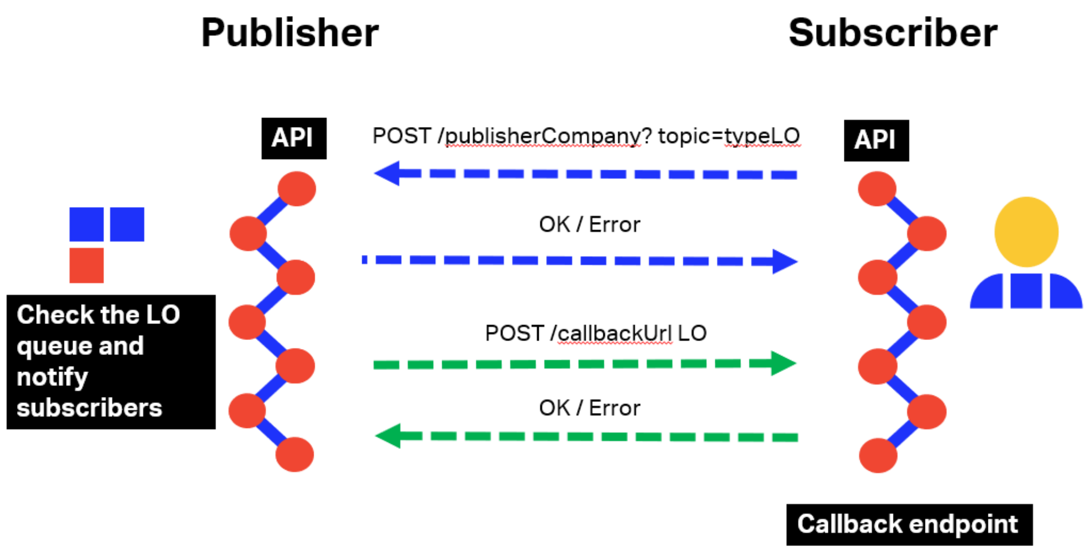
Note: In Publish & Subscribe, publishing parties need to save a list of all the parties subscribed to their Logistics Objects in their backend systems. One of the possibilities would be that the list of subscribers is embedded in the body of the Logistics Object.


Publisher
Subscriber
API
API
Create logistics object & add it to the subscriber queue to notify
GET /subscriberCompany?topic=typeLO
OK + subscription model / Error
POST /callbackUrl LO
OK / Error
Callback endpoint

Publish & Subscribe Sequence Diagram – Scenario 1

Scenario 2 – Publisher/Subscriber initiated by the Subscriber

In the second scenario, the subscriber initiates the subscription process by pulling the publisher in order to verify if there are any logistics objects/updates of a given topic to which it can subscribe to.



Publish & Subscribe Sequence Diagram – Scenario 2

The ONE Record Company Identifier

The ONE Record Company Identifier is a unique identifier for a company in the Internet of Logistics. It must be a URL that is unique to a specific 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 behaving as an address for the server 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 URL it 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 server subscribes to the topic (i.e. type of Logistics Object) from your server.

The Company Identifier is used in the following use cases:

- **In a Logistics Objects** – It is included in Logistics Objects to identify companies related to shipment.
- **For Authorization/Access Control** - 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 <u>or</u> it can be explicitly given by using the PATCH request on the ORS-API.

GET Company Information from the IoL

Retrieves basic company information.

Request

HTTP Request type: **GET**

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

HTTP Headers

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

Header	Value
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json

```
{
  "@id": "https://myonerecordserver.org/myLicensePlate",
  "@type": [
    "https://onerecord.iata.org/CompanyInformation"
  ],
  "https://onerecord.iata.org/CompanyInformation#company": {
    "@type": [
      "https://onerecord.iata.org/Company"
    ],
    "https://onerecord.iata.org/Company#airlineCode": "string",
    "https://onerecord.iata.org/Company#airlinePrefix": 0,
    "https://onerecord.iata.org/Company#branch": {
      "@type": [
        "https://onerecord.iata.org/Branch"
      ],
      "branchName": "string",
      "https://onerecord.iata.org/Branch#contactPerson": [
        {
          "@type": [
            "https://onerecord.iata.org/Person"
          ],
          "https://onerecord.iata.org/Person#contact": [
            {
              "@type": [
                "https://onerecord.iata.org/Contact"
              ],
              "https://onerecord.iata.org/Contact#emailAddress": "string",
              "https://onerecord.iata.org/Contact#other": {
                "@type": [
                  "https://onerecord.iata.org/ContactOther"
                ],
                "https://onerecord.iata.org/ContactOther#detail": "string",
                "https://onerecord.iata.org/ContactOther#otherType": "string"
              },
              "https://onerecord.iata.org/Contact#phoneNumber": "string"
            }
          ],
          "https://onerecord.iata.org/Person#contactType": "string",
          "https://onerecord.iata.org/Person#department": "string",
          "https://onerecord.iata.org/Person#employeeId": "string",
          "https://onerecord.iata.org/Person#firstName": "string",
          "https://onerecord.iata.org/Person#jobTitle": "string",
          "https://onerecord.iata.org/Person#lastName": "string",
          "https://onerecord.iata.org/Person#middleName": "string",
          "https://onerecord.iata.org/Person#salutation": "string"
        }
      ],
      "https://onerecord.iata.org/Branch#iataCargoAgentLocationIdentifier": 0,
      "https://onerecord.iata.org/Branch#location": {
        "@type": [
          "https://onerecord.iata.org/Location"
        ],
        "https://onerecord.iata.org/Location#address": {
          "@type": [
            "https://onerecord.iata.org/Address"
          ],
          "https://onerecord.iata.org/Address#addressCode": [
            "string"
          ],
          "https://onerecord.iata.org/Address#addressCodeType": "string",
          "https://onerecord.iata.org/Address#cityCode": "string",
          "https://onerecord.iata.org/Address#cityName": "string",
          "https://onerecord.iata.org/Address#country": {
            "@type": [
              "https://onerecord.iata.org/Country"
            ],
            "https://onerecord.iata.org/Country#countryCode": "string",
            "https://onerecord.iata.org/Country#countryName": "string"
          },
          "https://onerecord.iata.org/Address#poBox": "string",
          "https://onerecord.iata.org/Address#postalCode": "string",
          "https://onerecord.iata.org/Address#regionCode": "string",
          "https://onerecord.iata.org/Address#regionName": "string",
          "https://onerecord.iata.org/Address#street": [
            "string"
          ]
        },
        "https://onerecord.iata.org/Location#code": "string",
        "https://onerecord.iata.org/Location#geolocation": {
          "@type": [
            "https://onerecord.iata.org/Geolocation"
          ],
          "https://onerecord.iata.org/Geolocation#elevation": {
            "@type": [
              "https://onerecord.iata.org/Value"
            ],
            "https://onerecord.iata.org/Value#unit": "string",
            "https://onerecord.iata.org/Value#value": 0
          },
          "https://onerecord.iata.org/Geolocation#latitude": {
            "@type": [
              "https://onerecord.iata.org/Value"
            ],
            "https://onerecord.iata.org/Value#unit": "string",
            "https://onerecord.iata.org/Value#value": 0
          },
          "https://onerecord.iata.org/Geolocation#longitude": {
            "@type": [
              "https://onerecord.iata.org/Value"
            ],
            "https://onerecord.iata.org/Value#unit": "string",
            "https://onerecord.iata.org/Value#value": 0
          }
        },
        "https://onerecord.iata.org/Location#locationName": "string",
        "https://onerecord.iata.org/Location#locationType": "string"
      },
      "https://onerecord.iata.org/Branch#otherIdentifier": [
        {
          "@type": [
            "https://onerecord.iata.org/OtherIdentifier"
          ],
          "https://onerecord.iata.org/OtherIdentifier#identifier": "string",
          "https://onerecord.iata.org/OtherIdentifier#otherIdentifierType":
"string"
        }
      ]
    },
    "https://onerecord.iata.org/Company#companyName": "string",
    "https://onerecord.iata.org/Company#iataCargoAgentCode": 0
  },
  "https://onerecord.iata.org/CompanyInformation#companyId": "
https://myonerecordserver.org/myLicensePlate",
  "https://onerecord.iata.org/CompanyInformation#error": [
    {
      "@type": [
        "https://onerecord.iata.org/Error"
      ],
      "https://onerecord.iata.org/Error#details": [
        {
          "@type": [
            "https://onerecord.iata.org/Details"
          ],
          "https://onerecord.iata.org/Details#attribute": "string",
          "https://onerecord.iata.org/Details#code": "string",
          "https://onerecord.iata.org/Details#message": "string",
          "https://onerecord.iata.org/Details#resource": "string"
        }
      ],
      "https://onerecord.iata.org/Error#title": "string"
    }
  ],
  "https://onerecord.iata.org/CompanyInformation#serverEndpoint": "
https://myonerecordserver.org",
  "https://onerecord.iata.org/CompanyInformation#supportedContentTypes": [
    "application/ld+json"
  ],
  "https://onerecord.iata.org/CompanyInformation#supportedLogisticsObjects": [
    "https://onerecord.iata.org/WayBill"
  ]
}
```

Response

Code	Description	Response body

200	The request to retrieve the Company Information has been successful	Company Identifier
401	Not authenticated	Error model
403	Not authorized to retrieve the Company Information	Error model
404	Company Information not found	Error model

GET Subscription Information

Retrieves subscription information if the server subscribes to a topic:

Request

HTTP Request type: **GET**

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

HTTP Headers

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

Header	Value
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json

Response

```
{
  "@id": "https://subscriberonerecordserver?topic=ShippersInstructions",
  "@type": [
    "https://onerecord.iata.org/Subscription"
  ],
  "https://onerecord.iata.org/Subscription#contentType": [
    " application/ld+json "
  ],
  "https://onerecord.iata.org/Subscription#cacheFor": "86400",
  "https://onerecord.iata.org/Subscription#callbackUrl":
"https://subscriberonerecordserver/callback",
  "https://onerecord.iata.org/Subscription#error": [
    {
      "@type": [
        "https://onerecord.iata.org/Error"
      ],
      "https://onerecord.iata.org/Error#details": [
        {
          "@type": [
            "https://onerecord.iata.org/Details"
          ],
          "https://onerecord.iata.org/Details#attribute": "string",
          "https://onerecord.iata.org/Details#code": "string",
          "https://onerecord.iata.org/Details#message": "string",
          "https://onerecord.iata.org/Details#resource": "string"
        }
      ],
      "https://onerecord.iata.org/Error#title": "string"
    }
  ],
  "https://onerecord.iata.org/Subscription#myCompanyIdentifier": "
https://subscriberonerecordserver",
  "https://onerecord.iata.org/Subscription#secret": "
C89583BA9B1FEEAB25F715A3BA2F3",
  "https://onerecord.iata.org/Subscription#sendLogisticsObjectBody": true,
  "https://onerecord.iata.org/Subscription#subscribeToStatusUpdates": true,
  "https://onerecord.iata.org/Subscription#subscribedTo": "
https://publisheronerecordserver.net/yourCompany",
  "https://onerecord.iata.org/Subscription#topic":
"https://onerecord.iata.org/WayBill"
}
```

Where:

Field	Description	Required
subscribedTo	The Company Identifier of the company you want to subscribe to (delegation scenario)	YES
topic	The Logistics Object type that you want to subscribe to	YES

callbackUrl	The callback URL where you want to receive notifications on Logistics Objects. (Refer requirements of your endpoint below)	YES
contentType	The content type you want to receive.	YES
secret	Either a secret or API Key that ensures that only companies with this subscription information can `POST` to your `callbackUrl` endpoint	NO
subscribeToStatusUpdates	You must also specify if you want to receive updates on that Logistics Object	NO
sendLogisticsObjectBody	Flag specifying if the publisher should send the whole logistics object or not in the notification object	YES
cacheFor	Duration of the period to cache the subscription information in seconds	NO

Further development: A subscriber could also send specific field filter to which it wants to subscribe to. (e.g. destination countries).

Code	Description	Response body
200	The request to retrieve the Subscription Information has been successful	Subscription
204	Request has been successful, but the server does not subscribe	No response body
401	Not authenticated	Error model
403	Not authorized to retrieve the Subscription Information	Error model
404	Subscription Information not found	Error model

Subscription API Requirements

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 endpoint.
- MUST expect a 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	Required
URI-resource	The top-level URI of the Logistics Object of the resource being sent to the Client Subscriber. 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
Resource-Type	Class of the logistics object sent (e.g. WayBill, Booking)	YES
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
X-Hub-Signature	If a secret has been provided in the subscription, a HMAC signature would be present in this header	NO
x-api-key	If an API key is provided in the subscription, the API key would be provided in this header	NO
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 DISCUSS

POST Notification Request

The publisher sends a notification request to the subscriber when a logistics object is created or updated. If the subscriber chose to receive the entire logistics object body via `sendLogisticsObjectBody=true` field, then the whole object is sent.

Request

HTTP Request type: **POST**

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json
Content-Type	The content type that is contained with the HTTP body. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json

Request body for notification without the content of the object:

```
{
  "@type": [
    "https://onerecord.iata.org/Notification"
  ],
  "https://onerecord.iata.org/Notification#eventType": "OBJECT_CREATED",
  "https://onerecord.iata.org/Notification#logisticsObjectRef":
"https://server/licence_plate/some_id",
  "https://onerecord.iata.org/Notification#topic": "
https://onerecord.iata.org/WayBill"
}
```

Where:

Field	Description	Required
eventType	`OBJECT_CREATED` or `OBJECT_UPDATED`	YES
topic	Type of Logistics Object	YES
logisticsObjectRef	Logistics Object for which the notification is sent	YES

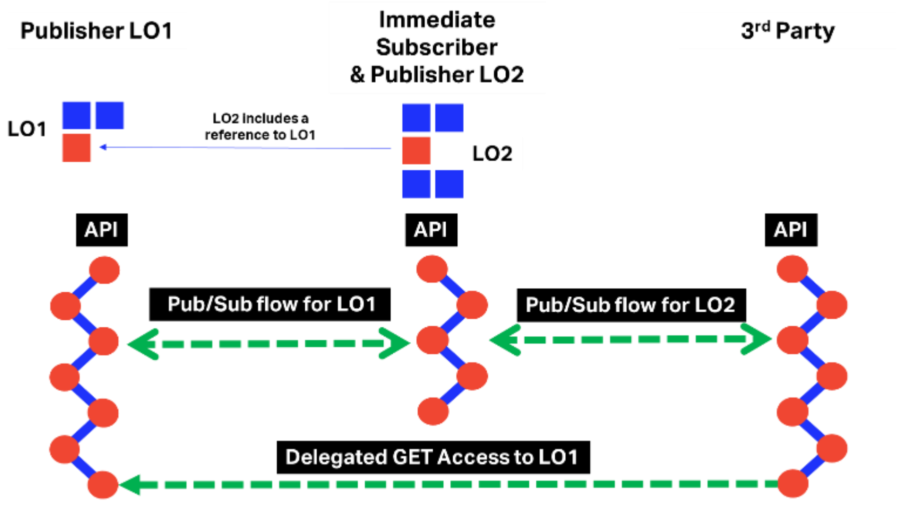
Response

Code	Description	Response body
204	Notification received successfully	No response body
400	Notification format is invalid	Error model
401	Not authenticated	Error model

403	Not authorized to send a notification request	Error model

Publish & Subscribe in multi-link/multi-party scenario

By definition, Linked Data is interlinked with other data, and the ownership of the different levels of data can be distributed throughout the Internet of Logistics. One of the challenges encountered while building a publisher/subscriber solution for ONE Record is to make sure that when there is an update on a logistics object, all the partners which are subscribed to other data that are linked to this object receive the update.


Publisher LO1
Immediate Subscriber & Publisher LO2
3rd Party
LO1
LO2 includes a reference to LO1
LO2
API
API
API
Pub/Sub flow for LO1
Pub/Sub flow for LO2
Delegated GET Access to LO1

In order to better illustrate the publisher/subscriber interactions in a multi-party context, we consider a scenario in which a **FirstParty**, a **SecondParty** and a **ThirdParty** are exchanging data in the Internet of Logistics.

Step 1 – FirstParty and SecondParty agree on Data Policy

In ONE Record, every data sharing of any type needs to be supported by data policy. Data policy determines, among others, the rights of the recipients to share data and with whom they do so. This policy determines also what data protection a recipient of data needs to apply. Data policy can have different levels of granularity, depending on the needs of each party.

The process is contractual, and it is effectuated at the beginning of the data exchange partnership between two parties of the Internet of Logistics. The policy can have an expiration date.

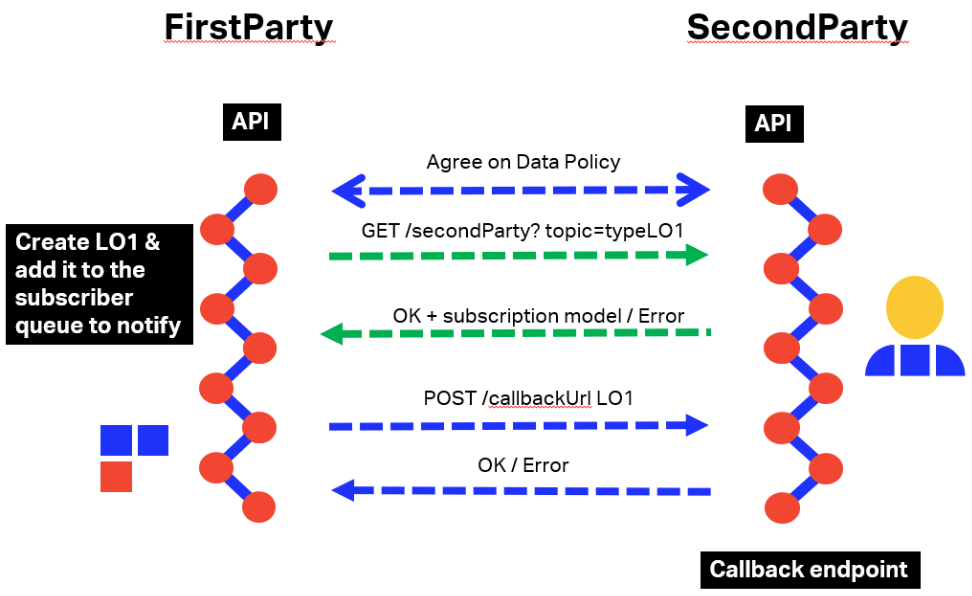
In this first step, FirstParty and SecondParty agree, among others, on which type of data the SecondParty can share to third parties.

Step 2 – FirstParty creates a Logistics Object (LO1) on its ONE Record Server

FirstParty creates a Logistics Object (called LO1 in this document) on its own ONE Record Server, using `POST` specifications of the ONE Record standard.

Step 3 – Publish & Subscribe flow between FirstParty and SecondParty

1. FirstParty asks SecondParty if it wants to subscribe to LO1.
2. SecondParty replies with `OK` and sends its subscription information back to FirstParty. The content of the subscription can be found here.
3. FirstParty sends notification containing the URI of the LO1 on his ONE Record Server to the callback endpoint on the SecondParty ONE Record Server.
4. SecondParty effectuates a `GET` LO1 on the FirstParty ONE Record Server in order to retrieve the full content of the LO1.

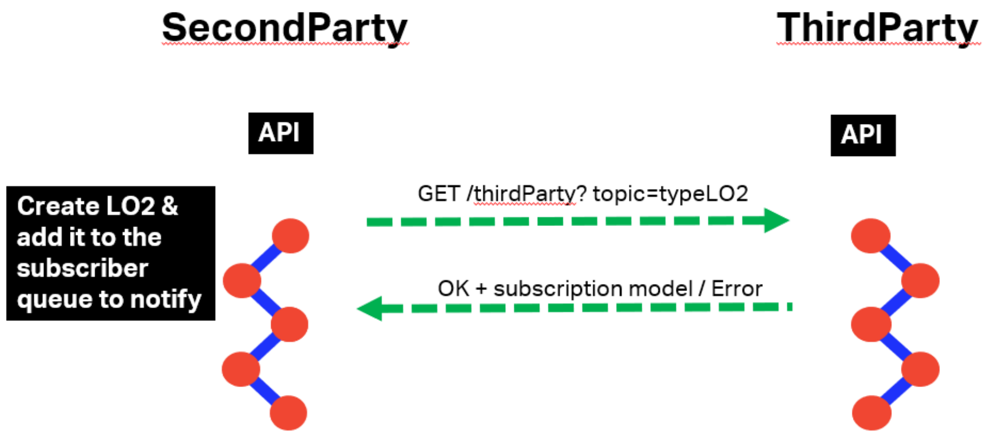

FirstParty
SecondParty
API
API
Agree on Data Policy
Create LO1 & add it to the subscriber queue to notify
GET /secondParty? topic=typeLO1
OK + subscription model / Error
POST /callbackUrl LO1
OK / Error
Callback endpoint

Step 4 – SecondParty creates a Logistics Object (LO2) on its own ONE Record Server

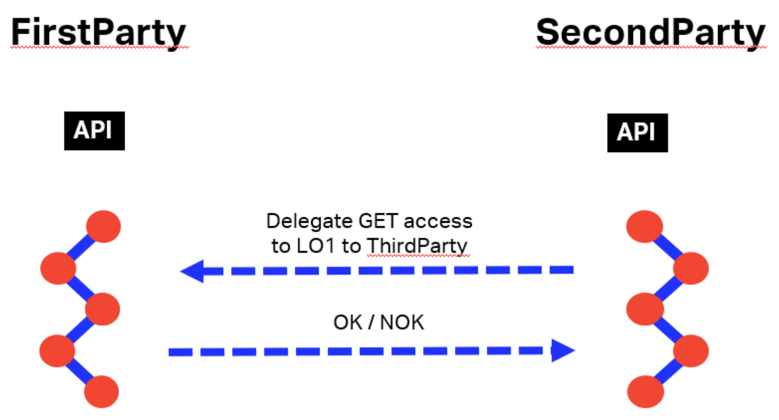
SecondParty creates a Logistics Object (called LO2 in this document) on its own ONE Record Server, using `POST` specifications of the ONE Record standard. **LO2 contains a link to LO1**.

Step 5 – Publish & Subscribe flow between SecondParty and ThirdParty with Delegation of access to LO1 to the ThirdParty

1. SecondParty asks ThirdParty if it wants to subscribe to LO2.
2. ThirdParty replies with `OK` and sends its subscription information back to SecondParty. The content of the subscription can be found here.

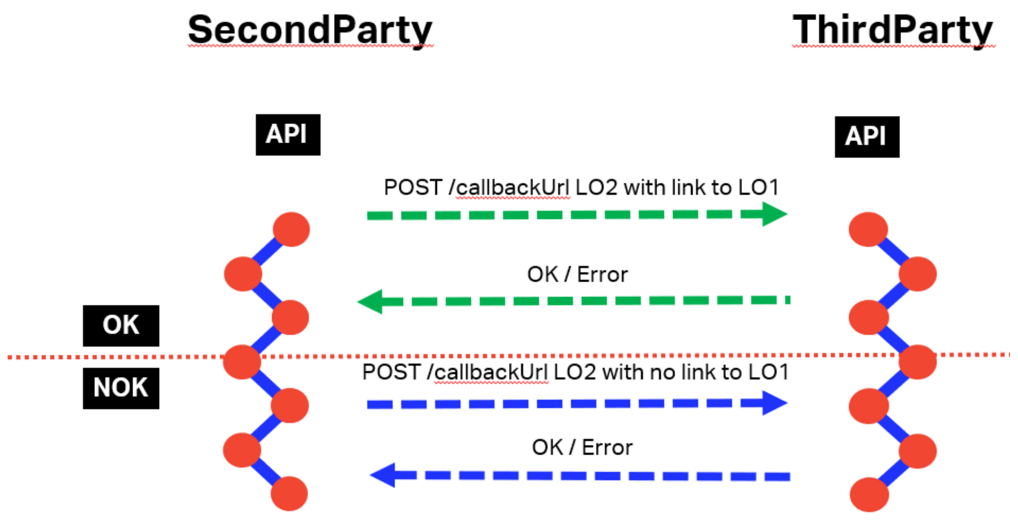

SecondParty
ThirdParty
API
API
Create LO2 & add it to the subscriber queue to notify
GET /thirdParty? topic=typeLO2
OK + subscription model / Error

3. Delegation flow between SecondParty and FirstParty in order to give `GET` access to LO1 to ThirdParty. More information about the delegation can be found here.



Important: If FirstParty does not want to give access of LO1 to ThirdParty, SecondParty should not share the link to LO1 in LO2.

4. SecondParty sends notification containing the URI of the LO2 (containing link to LO1) on his ONE Record Server to the callback endpoint on the ThirdParty ONE Record Server.
5. ThirdParty effectuates a `GET` LO2 on the SecondParty ONE Record Server in order to retrieve the full content of the LO2.
6. ThirdParty effectuates a `GET` LO1 on the FirstParty ONE Record Server in order to retrieve the full content of the LO1.


SecondParty
ThirdParty
API
API
POST /callbackUrl LO2 with link to LO1
OK / Error
OK
NOK
POST /callbackUrl LO2 with no link to LO1
OK / Error

Step 6 – FirstParty updates LO1

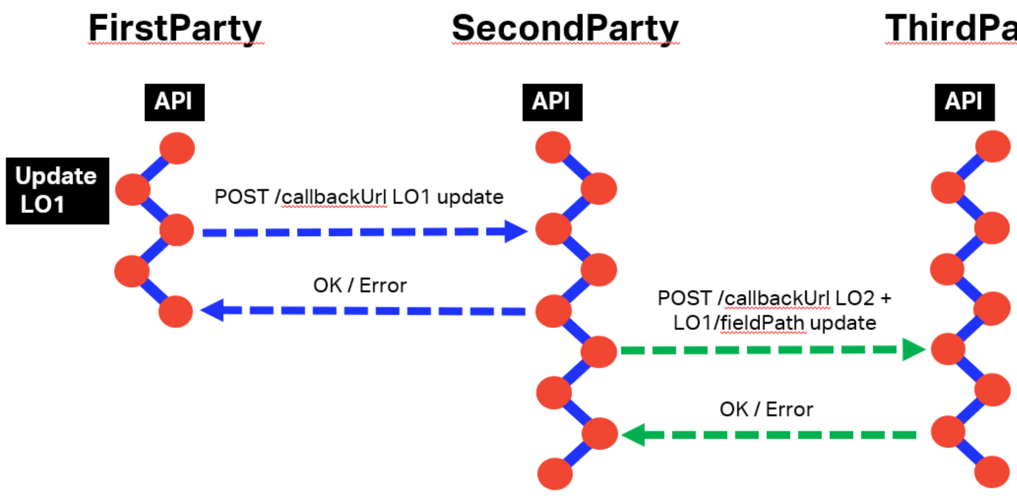
FirstParty updates LO1 on its own ONE Record Server, using `PATCH` specifications of the ONE Record standard.

Step 7 – FirstParty sends notification to SecondParty that the LO1 was updated

FirstParty sends `OBJECT_UPDATED` notification containing the URI of the LO1 on his ONE Record Server to the callback endpoint on the SecondParty ONE Record Server. The notification contains also the path of the field that has changed.

Step 8 – SecondParty may send notification to ThirdParty that the LO2 (which contains a link to LO1) was updated

SecondParty sends `OBJECT_UPDATED` notification containing the URI of the LO2 on his ONE Record Server to the callback endpoint on the ThirdParty ONE Record Server. The notification contains also the path of the field that has changed (in this case `LO1/fieldPath`).


FirstParty
SecondParty
ThirdParty
API
API
API
Update LO1
POST /callbackUrl LO1 update
OK / Error
POST /callbackUrl LO2 + LO1/fieldPath update
OK / Error

There are use cases in which SecondParty doesn't notify ThirdParty when LO1 is changed and the ThirdParty doesn't get the updated information.

SecondParty decides by its own internal processes (for example an internal rule engine) when it is necessary to inform ThirdParty that LO1 was updated. Usually, this would happen when the LO1 update triggers an action, e.g. object ready for pickup. If the LO1 update does not trigger an action for the ThirdParty, there is no need for notification between SecondParty and ThirdParty.

What if a Logistics Object contains multiple embedded links to other Logistics Objects?

The same mechanism applies in case of multiple embedded Logistics Objects.

In delegation scenario, what happens if FirstParty doesn't want to grant POST access to ThirdParty?

In delegation scenario, in case FirstParty does not want ThirdParty to `POST` updates, then the message from ThirdParty to SecondParty that something is wrong should be passed through traditional ways (e.g. email, phone, etc).

Data sovereignty is one of the base principles of ONE Record and the owner has the right to deny access to its data.

Conclusion

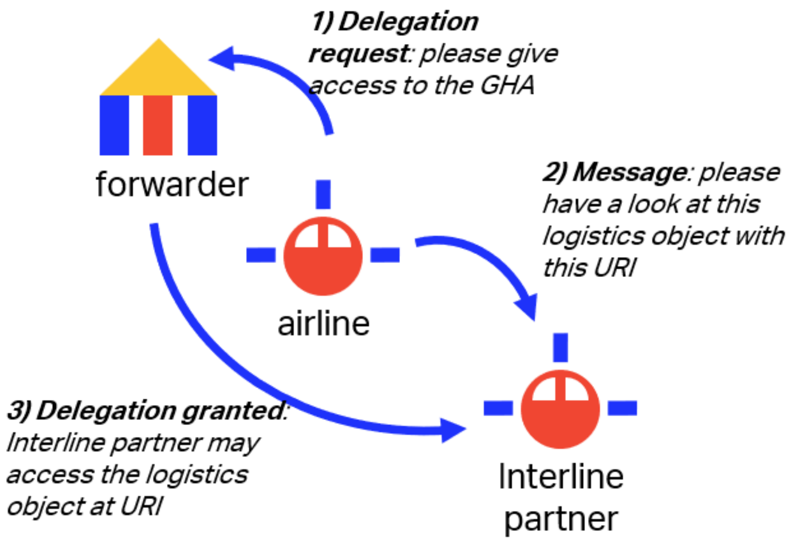
Publish & Subscribe only exists between the sharer and the immediate subscribers (in our case FirstParty - SecondParty) and third, fourth, n-th parties get delegated `GET` access, and not via Publish & Subcribe. Each party in between has the possibility of updating through their existing subscribers list the next party in the chain.

Delegation

In ONE Record parties are enabled to grant other parties access to (parts of) their data. The standard allows parties to modify or withdraw these access rights to their data, whenever they wish.

Before a company can access a logistics object of another company, it needs to be authorized to do so and the server that hosts the logistics objects will determine whether to grant access. Typically, when a company creates a logistics object on a server, it will share the URI of that logistics object with another company and grant them access by default. For example, a forwarder creates a logistics object for a booking request and then sends the URI to the airline. When the airline then accesses the logistics object, the forwarder will usually grant access to the airline but only to that airline and no one else.

The party granting access is referred to as the `delegator` and the party receiving the access is the `delegate`.


1) Delegation request: please give access to the GHA
forwarder
2) Message: please have a look at this logistics object with this URI
airline
3) Delegation granted: Interline partner may access the logistics object at URI
Interline partner

Request

HTTP Request type: **POST**

```
POST /delegation
Host: myonerecordserver.net
Accept: application/ld+json
```

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json
Content-Type	The content type that is contained with the HTTP body. Valid content types include:

- application/x-turtle or text/turtle
- application/ld+json

```
{
  "@type": [
    "https://onerecord.iata.org/DelegationRequest"
  ],
  "https://onerecord.iata.org/DelegationRequest#action": "DELEGATE",
  "https://onerecord.iata.org/DelegationRequest#operations": [
    "GET"
  ],
  "https://onerecord.iata.org/DelegationRequest#targetCompany": [
    "https:/onerecordserver.org/delegated_company_identifier"
  ],
  "https://onerecord.iata.org/DelegationRequest#targetLogisticsObject": [
    "https://1recordserver.org/my_airline/waybill_123"
  ]
}
```

Where:

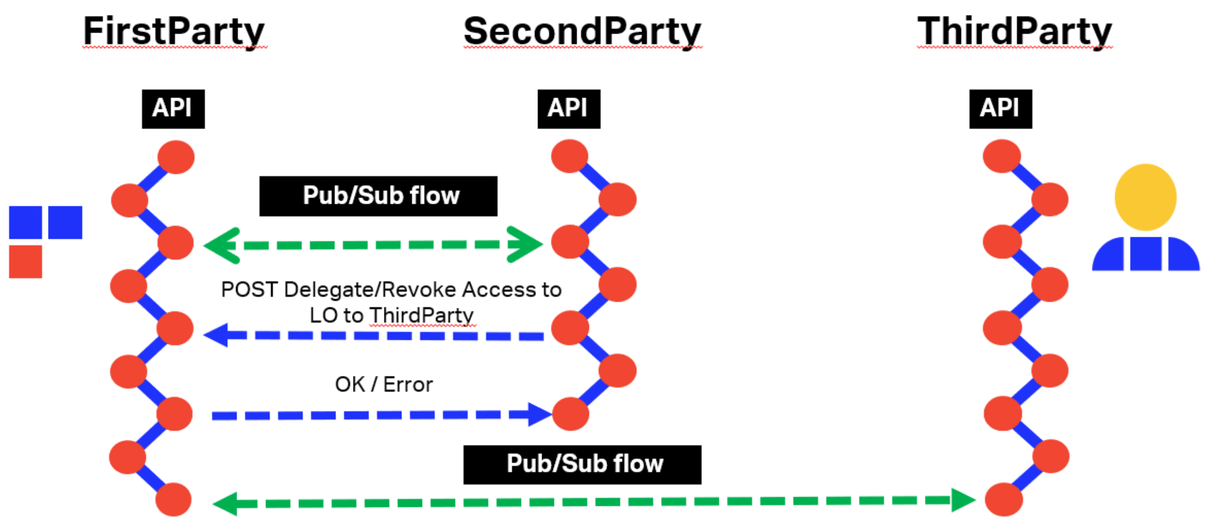
Field	Description	Required
@type	DelegationRequest	YES
targetLogisticsObject	The identifier of the logistics object to which the access is requested	YES
targetCompany	The party that receives the delegated rights	YES
action	The action to perform: `REVOKE` or `DELEGATE`	YES
operations	The API operations to which the access is requested: `GET`, `PATCH`, or both.	YES

Response

Code	Description	Response body
204	Request for delegation was successful	No response body
401	Not authenticated	Error model
403	Not authorized to send a Delegation request	Error model

In the following use cases, we consider three parties: **FirstParty**, **SecondParty** and **ThirdParty**.

Use Case 1



Delegation Scenario 1

Step 1 – Publish & Subscribe flow between FirstParty and SecondParty

First step contains a Publish & Subscribe flow between FirstParty and SecondParty for a Logistics Object LO with is created on the FirstParty ONE Record Server.

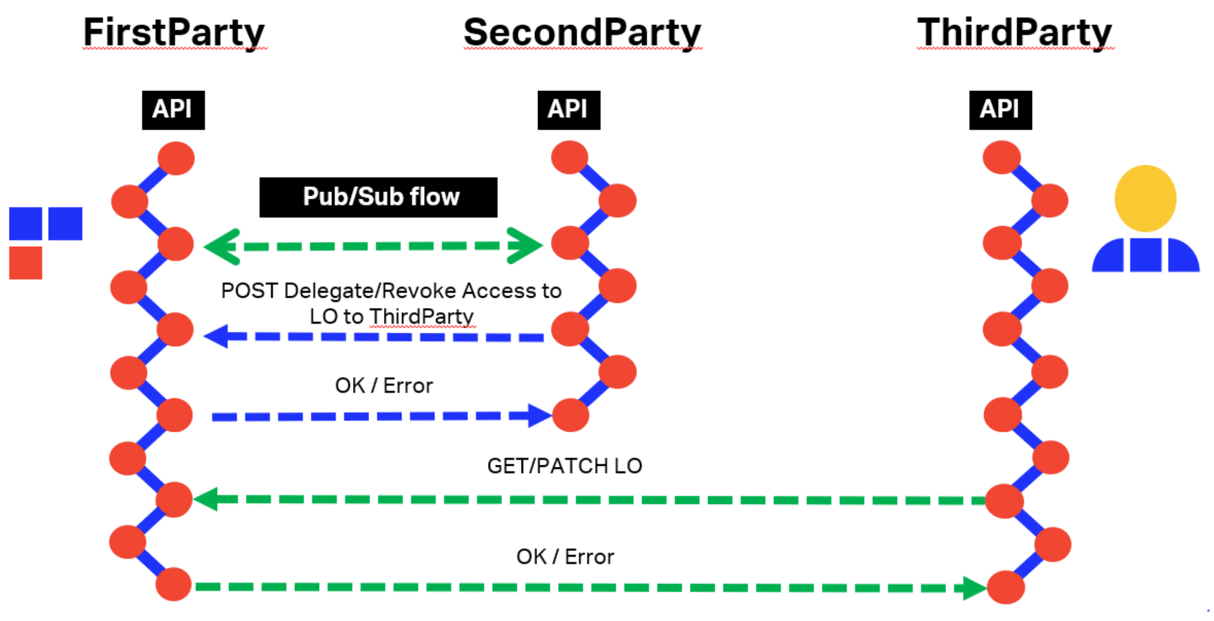
Step 2 – SecondParty requests delegation access to LO for ThirdParty

The SecondParty sends a delegation request to the FirstParty in order to grant ThirdParty access to LO (`GET`, `PATCH` or both).

Step 3 – Publish & subscribe flow between FirstParty and ThirdParty

If FirstParty decides to grant ThirdParty access to LO, then it initiates a Publish & Subscribe flow that would allow ThirdParty get notifications related to LO.

Use Case 2



Delegation Scenario 2

Step 1 – Publish & Subscribe flow between FirstParty and SecondParty

First step contains a Publish & Subscribe flow between FirstParty and SecondParty for a Logistics Object LO with is created on the FirstParty ONE Record Server.

Step 2 – SecondParty requests delegation access to LO for ThirdParty

The SecondParty sends a delegation request to the FirstParty in order to grant ThirdParty access to LO (`GET`, `PATCH` or both).

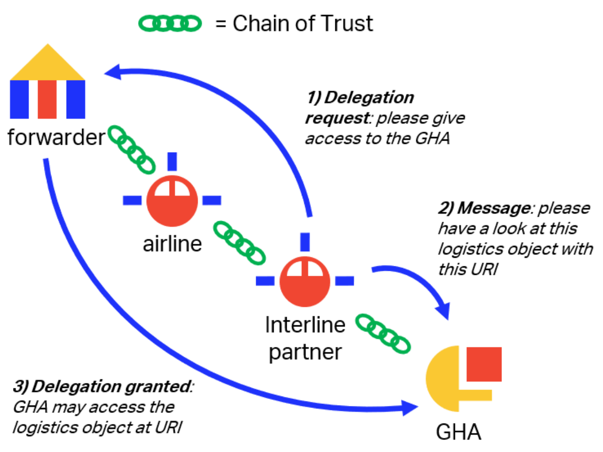
Same Request/Response as in Step 2 – SecondParty requests delegation access to LO for ThirdParty.

Step 3 – ThirdParty effectuates a GET or PATCH request on LO to FirstParty

If FirstParty decides to grant ThirdParty access to LO, then ThirdParty can perform a `GET` or `PATCH` request on the LO to FirstParty.

Trust Chains

The concept of companies requesting a delegation of access to their partners can also be used by these partners themselves, who are now third parties. In the example above, the interline partner can request that the forwarder gives access to their ground handler. The forwarder will grant the access on the basis that they trust the airline who has trusted their interline partner who trusts their ground handler.



= Chain of Trust
forwarder
airline
Interline partner
GHA
1) Delegation request: please give access to the GHA
2) Message: please have a look at this logistics object with this URI
3) Delegation granted: GHA may access the logistics object at URI

These chains of trust are based on business partnerships and trust in the transport chain. It ensures that the company who has shared a logistics object on a server, always knows who may access this and at any time, it can revoke all or part of the chain of trust.

Events (status update)

Status updates in ONE Record can pe added to Logistics Objects through Events. By definition, each Logistics Object can be assigned events.

Create event (POST)

Request

HTTP Request type: **POST**

```
POST logisticsObjectURI/events
Host: myonerecordserver.net
Accept: application/ld+json
```

Example:

POST https://www.onerecordcargo.org/my_airline/shipment_123456/events

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json
Content-Type	The content type that is contained with the HTTP body. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json

```
{
  "@type": [
    "https://onerecord.iata.org/Event"
  ],
  "https://onerecord.iata.org/Event#dateTime": "2020-08-25T13:44:49.399Z",
  "https://onerecord.iata.org/Event#eventCode": "string",
  "https://onerecord.iata.org/Event#eventName": "string",
  "https://onerecord.iata.org/Event#eventTypeIndicator": "string",
  "https://onerecord.iata.org/Event#location": {
    "@type": [
      "https://onerecord.iata.org/Location"
    ],
    "https://onerecord.iata.org/Location#address": {
      "@type": [
        "https://onerecord.iata.org/Address"
      ],
      "https://onerecord.iata.org/Address#addressCode": [
        "string"
      ],
      "https://onerecord.iata.org/Address#addressCodeType": "string",
      "https://onerecord.iata.org/Address#cityCode": "string",
      "https://onerecord.iata.org/Address#cityName": "string",
      "https://onerecord.iata.org/Address#country": {
        "@type": [
          "https://onerecord.iata.org/Country"
        ],
        "https://onerecord.iata.org/Country#countryCode": "string",
        "https://onerecord.iata.org/Country#countryName": "string",
        "id": "string"
      },
      "https://onerecord.iata.org/Address#poBox": "string",
      "https://onerecord.iata.org/Address#postalCode": "string",
      "https://onerecord.iata.org/Address#regionCode": "string",
      "https://onerecord.iata.org/Address#regionName": "string",
      "https://onerecord.iata.org/Address#street": [
        "string"
      ],
      "id": "string"
    },
    "https://onerecord.iata.org/Location#code": "string",
    "https://onerecord.iata.org/Location#geolocation": {
      "@type": [
        "https://onerecord.iata.org/Geolocation"
      ],
      "https://onerecord.iata.org/Geolocation#elevation": {
        "@type": [
          "https://onerecord.iata.org/Value"
        ],
        "https://onerecord.iata.org/Value#unit": "string",
        "https://onerecord.iata.org/Value#value": 0,
        "id": "string"
      },
      "https://onerecord.iata.org/Geolocation#latitude": {
        "@type": [
          "https://onerecord.iata.org/Value"
        ],
        "https://onerecord.iata.org/Value#unit": "string",
        "https://onerecord.iata.org/Value#value": 0,
        "id": "string"
      },
      "https://onerecord.iata.org/Geolocation#longitude": {
        "@type": [
          "https://onerecord.iata.org/Value"
        ],
        "https://onerecord.iata.org/Value#unit": "string",
        "https://onerecord.iata.org/Value#value": 0,
        "id": "string"
      },
      "id": "string"
    },
    "https://onerecord.iata.org/Location#locationName": "string",
    "https://onerecord.iata.org/Location#locationType": "string",
    "id": "string"
  },
  "https://onerecord.iata.org/Event#logisticsObjectRef": "string",
  "https://onerecord.iata.org/Event#performedBy": {
    "@type": [
      "https://onerecord.iata.org/Company"
    ],
    "https://onerecord.iata.org/Company#airlineCode": "string",
    "https://onerecord.iata.org/Company#airlinePrefix": 0,
    "https://onerecord.iata.org/Company#branch": {
      "@type": [
        "https://onerecord.iata.org/Branch"
      ],
      "branchName": "string",
      "https://onerecord.iata.org/Branch#contactPerson": [
        {
          "@type": [
            "https://onerecord.iata.org/Person"
          ],
          "https://onerecord.iata.org/Person#contact": [
            {
              "@type": [
                "https://onerecord.iata.org/Contact"
              ],
              "https://onerecord.iata.org/Contact#emailAddress": "string",
              "https://onerecord.iata.org/Contact#other": {
                "@type": [
                  "https://onerecord.iata.org/ContactOther"
                ],
                "https://onerecord.iata.org/ContactOther#detail": "string",
                "https://onerecord.iata.org/ContactOther#otherType": "string",
                "id": "string"
              },
              "https://onerecord.iata.org/Contact#phoneNumber": "string",
              "id": "string"
            }
          ],
          "https://onerecord.iata.org/Person#contactType": "string",
          "https://onerecord.iata.org/Person#department": "string",
          "https://onerecord.iata.org/Person#employeeId": "string",
          "https://onerecord.iata.org/Person#firstName": "string",
          "https://onerecord.iata.org/Person#jobTitle": "string",
          "https://onerecord.iata.org/Person#lastName": "string",
          "https://onerecord.iata.org/Person#middleName": "string",
          "https://onerecord.iata.org/Person#salutation": "string",
          "id": "string"
        }
      ],
      "https://onerecord.iata.org/Branch#iataCargoAgentLocationIdentifier": 0,
      "https://onerecord.iata.org/Branch#location": {
        "@type": [
          "https://onerecord.iata.org/Location"
        ],
        "https://onerecord.iata.org/Location#address": {
          "@type": [
            "https://onerecord.iata.org/Address"
          ],
          "https://onerecord.iata.org/Address#addressCode": [
            "string"
          ],
          "https://onerecord.iata.org/Address#addressCodeType": "string",
          "https://onerecord.iata.org/Address#cityCode": "string",
          "https://onerecord.iata.org/Address#cityName": "string",
          "https://onerecord.iata.org/Address#country": {
            "@type": [
              "https://onerecord.iata.org/Country"
            ],
            "https://onerecord.iata.org/Country#countryCode": "string",
            "https://onerecord.iata.org/Country#countryName": "string",
            "id": "string"
          },
          "https://onerecord.iata.org/Address#poBox": "string",
          "https://onerecord.iata.org/Address#postalCode": "string",
          "https://onerecord.iata.org/Address#regionCode": "string",
          "https://onerecord.iata.org/Address#regionName": "string",
          "https://onerecord.iata.org/Address#street": [
            "string"
          ],
          "id": "string"
        },
        "https://onerecord.iata.org/Location#code": "string",
        "https://onerecord.iata.org/Location#geolocation": {
          "@type": [
            "https://onerecord.iata.org/Geolocation"
          ],
          "https://onerecord.iata.org/Geolocation#elevation": {
            "@type": [
              "https://onerecord.iata.org/Value"
            ],
            "https://onerecord.iata.org/Value#unit": "string",
            "https://onerecord.iata.org/Value#value": 0,
            "id": "string"
          },
          "https://onerecord.iata.org/Geolocation#latitude": {
            "@type": [
              "https://onerecord.iata.org/Value"
            ],
            "https://onerecord.iata.org/Value#unit": "string",
            "https://onerecord.iata.org/Value#value": 0,
            "id": "string"
          },
          "https://onerecord.iata.org/Geolocation#longitude": {
            "@type": [
              "https://onerecord.iata.org/Value"
            ],
            "https://onerecord.iata.org/Value#unit": "string",
            "https://onerecord.iata.org/Value#value": 0,
            "id": "string"
          },
          "id": "string"
        },
        "https://onerecord.iata.org/Location#locationName": "string",
        "https://onerecord.iata.org/Location#locationType": "string",
        "id": "string"
      },
      "https://onerecord.iata.org/Branch#otherIdentifier": [
        {
          "@type": [
            "https://onerecord.iata.org/OtherIdentifier"
          ],
          "https://onerecord.iata.org/OtherIdentifier#identifier": "string",
          "https://onerecord.iata.org/OtherIdentifier#otherIdentifierType":
"string",
          "id": "string"
        }
      ],
      "id": "string"
    },
    "https://onerecord.iata.org/Company#companyName": "string",
    "https://onerecord.iata.org/Company#iataCargoAgentCode": 0,
    "id": "string"
  },
  "id": "string"
}
```

Where:

Field	Description	Required
@type	Event	YES
logisticsObjectRef	Logistics object the event is valid for	YES
performedBy	Company IoL identifier of the entity from which the event comes from	YES
eventCode	Valid event code	YES
eventName	Valid event name	YES
eventTypeIndicator		YES
dateTime	Date and time when the event occurred	YES
eventApplicableTo	List of IRI, used only when the event is only applicable to certain linked logistics objects in the primary logistics object (logisticsObjectRef) – To discuss with the group if still needed in future releases	NO
location	Location of where the event occurred (of type ONE Record - Location)	NO

Response

Code	Description	Response body
201	Event has been published to the Internet of Logistics.	No response body
400	Invalid Event	Error model
401	Not authenticated	Error model
403	Not authorized to publish an event update to the Internet of Logistics	Error model
415	Unsupported Content Type	Error model

Retrieve events (GET)

Request

HTTP Request type: **GET**

```
GET logisticsObjectURI/events
Host: myonerecordserver.net
Accept: application/ld+json
```

Example:

GET https://www.onerecordcargo.org/my_airline/shipment_123456/events

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ 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 events list in the format that has been requested in the Accept header of the request.

Code	Description	Response body
200	Events retrieved successfully	Event

401	Not authenticated	Error model
403	Not authorized to retrieve Events	Error model
404	Logistics Object Not Found	Error model

Access Control

In ONE Record, access to resources could be handled by using Access Control Lists (ACLs) stored in the backend systems of the ONE Record Servers and defined using the Web Access Control standard from W3C.

Web Access Control

According to W3C specifications, Web Access Control it is a standard that enforces access control based on the Access Control List (ACL) RDF resource associated with the requested resource. It's concerned with giving access to agents (users, groups and more) to perform various kinds of operations (read, write, append, etc) on resources. In Web Access Control, an ACL consists of a set of **Authorizations**. Each Authorization is a single rule for access, such as "`entities one and two may write to resource someresource`", described with a set of RDF properties.

Access Control List Resources

In a system that uses Web Access Control, each web resource has a set of **Authorization** statements describing:

1. Who has access to that resource (that is, who the authorized **agents** are).
2. What types (or **modes**) of access they have.

These Authorizations are either explicitly set for an individual resource, or (more often) inherited from that resource's parent folder or container. In either case, the Authorization statements are placed into separate WAC documents called `Access Control List Resources` (or simply `ACLs`).

An Authorization is written as a series of RDF triples, with the Authorization resource as the subject for each of these triples. The Authorization resource URI is a hash URI, because of the requirement for potentially multiple, distinct Authorizations to be included in a single ACL resource and it has an `rdf: type` of `http://www.w3.org/ns/auth/acl#Authorization`.

The complete WAC ontology can be found here.

The location of the `acl` for a given resource may be discovered via a `Link` header with relation `rel="acl"`. Given a URL for an individual resource or logistics object, a client can discover the location of its corresponding ACL by performing a `GET` request and parsing the `rel="acl"` link relation.

Example:

GET `http://myServer/myAirline/logisticsObject` would return:

```
HTTP/1.1 200 OK
Link: <http://myServer/myAirline/logisticsObject/acl>; rel="acl"
```

If a resource does not have an individual ACL (and therefore relies on an implicit ACL from a parent), this link header will still be present, but will return a 404.

Clients MUST NOT assume that the location of an ACL resource can be deterministically derived from a document's URL.

Example of Authorizations

Single Authorization

Below is an example ACL resource that specifies that Party1 (as identified by its ONE Record Company Identifier `https://party1.server.com/company`) has full access (Read, Write and Control) to one of its web resources, located at `https://party1.server.com/company/logisticsObject`.

```
# Contents of https://party1.server.com/company/logisticsObject/acl
@prefix acl:  <http://www.w3.org/ns/auth/acl#>.

<#authorization1>
    a              acl:Authorization;
    acl:agent      <https://party1.server.com/company>;  # Company Identifier in the IoL
    acl:accessTo   <https://party1.server.com/company/logisticsObject>;
    acl:mode       acl:Read,
                   acl:Write,
                   acl:Control.
```

Agent = Company Identifier in the Internet of Logistics

In this case, Party1 is the Owner of the Logistics Object. Owners are agents that have Read, Write and Control permissions.

Group Authorization

```
# Group authorization, giving Read/Write access to two groups, which are
# specified in the 'work-groups' document.
<#authorization2>
    a               acl:Authorization;
    acl:accessTo    <https://party1.server.com/company/logisticsObject2>;
    acl:mode        acl:Read,
                    acl:Write;
    acl:agentGroup  <https://party1.server.com/company/groups#Accounting>;
    acl:agentGroup  <https://party1.server.com/company/groups#Management>.
```

Corresponding `work-groups` Group Listing document:

```
# Contents of https://party1.server.com/company/groups
@prefix    acl:  <http://www.w3.org/ns/auth/acl#>.
@prefix     dc:  <http://purl.org/dc/elements/1.1/>.
@prefix  vcard:  <http://www.w3.org/2006/vcard/ns#>.
@prefix    xsd:  <http://www.w3.org/2001/XMLSchema#>.

<#Accounting>
    a                vcard:Group;
    vcard:hasUID     <urn:uuid:8831CBAD-1111-2222-8563-F0F4787E5398:ABGroup>;
    dc:created       "2018-09-11T07:18:19+0000"^^xsd:dateTime;
    dc:modified      "2019-08-08T14:45:15+0000"^^xsd:dateTime;

    # Accounting group members:
    vcard:hasMember  <https://party2.server.com/company5>;
    vcard:hasMember  <https://party3.server.com/company7>.

<#Management>
    a                vcard:Group;
    vcard:hasUID     <urn:uuid:8831CBAD-3333-4444-8563-F0F4787E5398:ABGroup>;
    # Management group members:
    vcard:hasMember  <https://party4.server.com/company1>.
```

Authenticated Agents (Default)

Authenticated access is a bit like public access, but it is not anonymous. Access is only given to clients who have logged on and provided a specific Company Identifier. This allows the server to track the entities who have used the resource.

```
@prefix   acl:  <http://www.w3.org/ns/auth/acl#>.

<#authorization2>
    a               acl:Authorization;
    acl:agentClass  acl:AuthenticatedAgent;
    acl:mode        acl:Read;
    acl:accessTo    <https://party1.server.com/company/logisticsObject>.
```

An application of this feature is to throw a resource open to all authenticated parties for a specific amount of time, accumulate the list of those who case as a group, and then later restrict access to that group, to prevent spam.

Modes of Access

The `acl:mode` predicate denotes a class of operations that the agents can perform on a resource.

`acl:Read`
This includes access to HTTP verb `GET`.

`acl:Write`
This includes `POST`, and `PATCH`. (`PUT` and `DELETE` are out of scope of ONE Record).

`acl:Control`
All methods, if the request URI is an ACL.

Out of scope of ONE Record

`acl:Append`
gives a more limited ability to write to a resource - Append-Only.

Inheritance

Use `acl:default.`

If you use `acl:accessTo` to protect a container, and add an `acl:default` predicate, that authorization rule by default will also apply to any of that container's children, unless that child has its own ACL.

The second is to use the `acl:accessToClass` property to state that the authorization rule applies to any resource with the named RDF type.

Delegation

When delegating access to a resource to a third party, a new Authorization element should be added to the ACL.

Access Control in ONE Record

In ONE Record, access to resources can be specified by using Access Control Lists (ACLs) associated to specific Logistics Objects (LOs). Each LO resource possesses a related ACL containing a set of **Authorization** statements that describe:

- **who** has access to that resource;
- **what types** (or **modes**) **of access** they have.

Each Authorization is a single rule for access, such as "`entities one and two may write to LO logisticObjectRef`", described with a set of RDF properties.

ONE Record recommends the use of the ACL ontology in order to express the Authorizations. As the ACL is specific to each ONE Record Server and it is not a mandatory requirement to make it available to external entities, any other kind of data model/ontology can be used instead.

Given an URI for an individual LO, a ONE Record Client can discover the location of its corresponding ACL by performing a `GET` request and parsing the `rel="acl"` **Link** header.

GET `http://myServer/myAirline/logisticsObject` would return the headers:

```
HTTP/1.1 200 OK
Link:
<http://myServer/myAirline/logisticsObject/acl>; rel="acl"
```

Example of GET LO returning an ACL

__Note__: ONE Record Clients must not assume that the location of an ACL resource can be derived from an LO's URI.

ONE Record recommends the definition of three types of Authorization:

1. **Single Authorization –** when a single company identifier from the Internet of Logistics has access to the LO;
2. **Group Authorization –** when a group of company identifiers has access to the LO. The ONE Record Server can define internally groups of access such as Airlines, Ground Handlers, Customs, etc.
3. **"Public" Authorization –** when every authenticated company identifier accessing the LO URI can retrieve the data.

ONE Record specifies three modes of access on LOs:

READ / **GET**
Read the contents (including querying it)

WRITE / **POST** and **PATCH**
Write contents or modify part of it

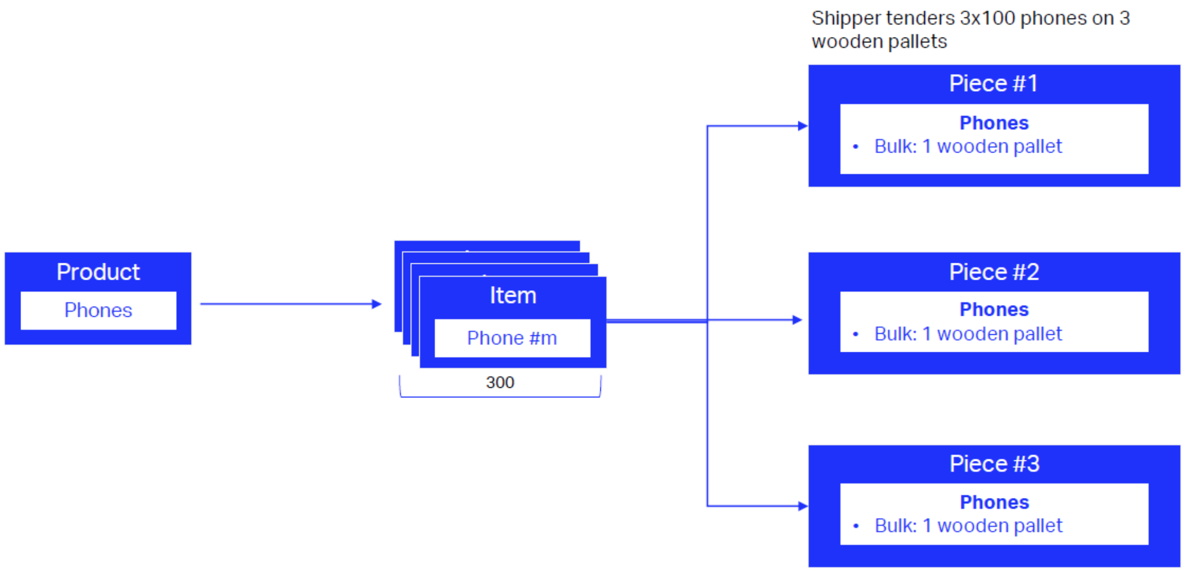
CONTROL / **GET**, **POST** and **PATCH**
Read and Write

Note: In a delegation scenario, when delegating access to a resource to a third party, a new Authorization element should be added to the ACL.

Use case

In this illustrative scenario, the entities would be **Shipper** (which has its own ONE Record Server) and **Forwarder**.

- Shipper creates a Shipment logistics object. Shipper is the owner of the phones, which constitute the Product. Each Product with a series number is an Item.
- Shipper packages the phones on wooden pallets, creates Pieces and handles them to Forwarder.



- Forwarder is responsible to create/update Transport Segment data for each Piece.

```
# Contents of https://shipperserver.com/company/shipment.acl
@prefix acl:  <http://www.w3.org/ns/auth/acl#>.

<#authorization1>
    a              acl:Authorization;
    acl:agent      <https://forwarderserver.com/company>;  # Forwarder Identifier in the IoL
    acl:accessTo   <https://shipperserver.com/company/shipment/transportSegment>;
    acl:mode       acl:Read,
                   acl:Write,
                   acl:Control.
```

- Each Piece contains other information such as Goods Description which is only created/updated by Shipper.

```
# Contents of https://shipperserver.com/company/shipment.acl
@prefix acl:  <http://www.w3.org/ns/auth/acl#>.
```

```
<#authorization1>
    a             acl:Authorization;
    acl:agent     <https://shipperserver.com/company>;  # Shipper Identifier in the IoL
    acl:accessTo  <https://shipperserver.com/company/shipment/goodDescription>;
    acl:mode      acl:Read,
                  acl:Write,
                  acl:Control.
```

- UPID information inside Piece could either be created by Shipper, in which case Shipper would be the owner of this data element. However, Shipper could handle the UPID creation/update to Forwarder, in which case Forwarder would be the owner of this data, meaning that only Forwarder could update this information.

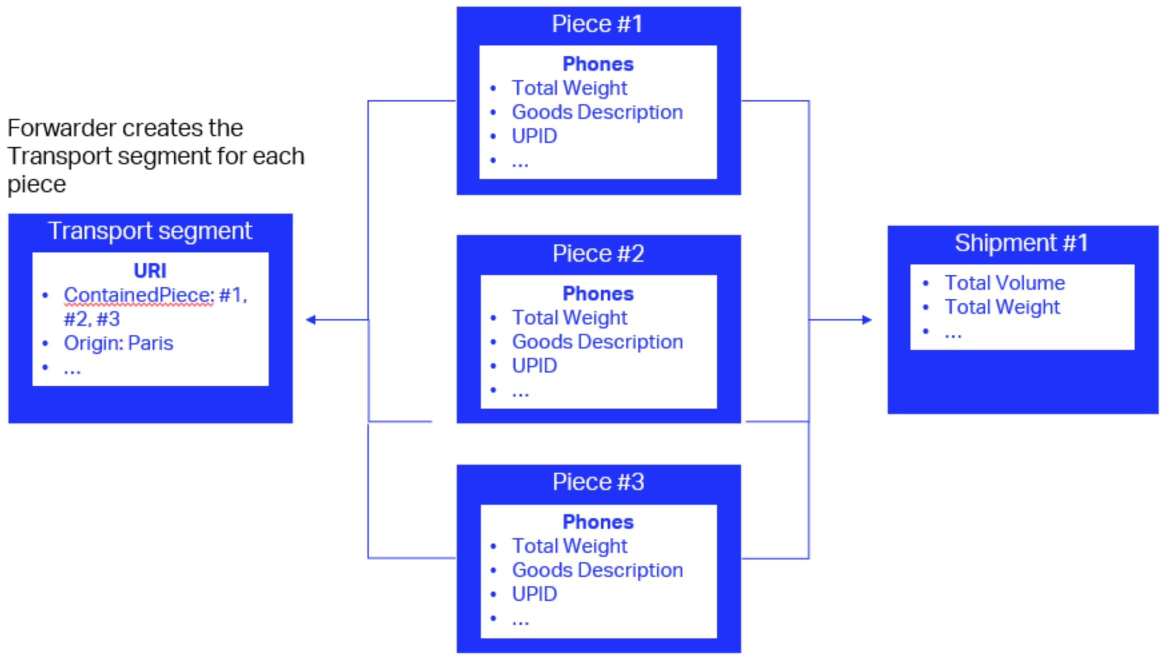
```
# Contents of https://shipperserver.com/company/shipment.acl
@prefix acl:  <http://www.w3.org/ns/auth/acl#>.

<#authorization1>
    a             acl:Authorization;
    acl:agent     <https://forwarder.com/company>;  # Forwarder Identifier in the IoL
    acl:accessTo  <https://shipperserver.com/company/shipment/UPID>;
    acl:mode      acl:Read,
                  acl:Write,
                  acl:Control.
```

- Same principle as UPID could apply for Total Weight element inside Piece.

```
# Contents of https://shipperserver.com/company/shipment.acl
@prefix acl:  <http://www.w3.org/ns/auth/acl#>.

<#authorization1>
    a             acl:Authorization;
    acl:agent     <https://forwarder.com/company>;  # Forwarder Identifier in the IoL
    acl:accessTo  <https://shipperserver.com/company/shipment/totalWeight>;
    acl:mode      acl:Read,
                  acl:Write,
                  acl:Control.
```



Create ACL (POST)

Request

HTTP Request type: **POST**

```
POST logisticsObjectURI/acl
Host: myonerecordserver.net
Accept: application/ld+json
```

Example:

POST https://www.onerecordcargo.org/my_airline/shipment_123456/acl

ONE Record does not define a specific model for ACL, but suggests the utilization of Access Control Ontology defined by W3C.

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json
Content-Type	The content type that is contained with the HTTP body. Valid content types include: ▪ application/x-turtle or text/turtle

- application/ld+json

Response

Code	Description	Response body
201	ACL has been published for a Logistics Object	No response body
400	Invalid ACL	Error model
401	Not authenticated	Error model
403	Not authorized to publish an ACL	Error model
415	Unsupported Content Type	Error model

Retrieve ACL (GET)

Request

HTTP Request type: **GET**

```
GET logisticsObjectURI/acl
Host: myonerecordserver.net
Accept: application/ld+json
```

Example:

GET https://www.onerecordcargo.org/my_airline/shipment_123456/acl

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ 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 ACL list in the format that has been requested in the Accept header of the request.

Code	Description	Response body

200	ACL returned successfully	ACL
401	Not authenticated	Error model
403	Not authorized to retrieve ACL	Error model
404	Logistics Object/ACL Not Found	Error model

Bibliography

Access Control Ontology: https://www.w3.org/ns/auth/acl

Web Access Control Specification from W3: https://www.w3.org/wiki/WebAccessControl

Web Access Control specifications by Solid project: https://github.com/solid/web-access-control-spec

Fedora project: https://wiki.lyrasis.org/display/FEDORA51/WebAC+Authorizations

Access Control in Linked Data Using WebID: https://arxiv.org/pdf/1611.03019.pdf

VCard ontology: https://www.w3.org/2006/vcard/ns#%3E

Context-Aware Access Control and Presentation of Linked Data: https://tel.archives-ouvertes.fr/tel-00934617/document

Versioning

In ONE Record, data is updated in real time. There is a need to snapshot a version of a document, for example MAWB, and we need to know which version of data was used for that snapshot.

Every time a transaction/update is committed successfully, a new version entry is created by the versioning service.

Note: Revert to a previous memento (version) of a Logistics Object with PATCH and Deleting a previous memento (version) of a Logistics Object with DELETE are not supported as out of scope of ONE Record.

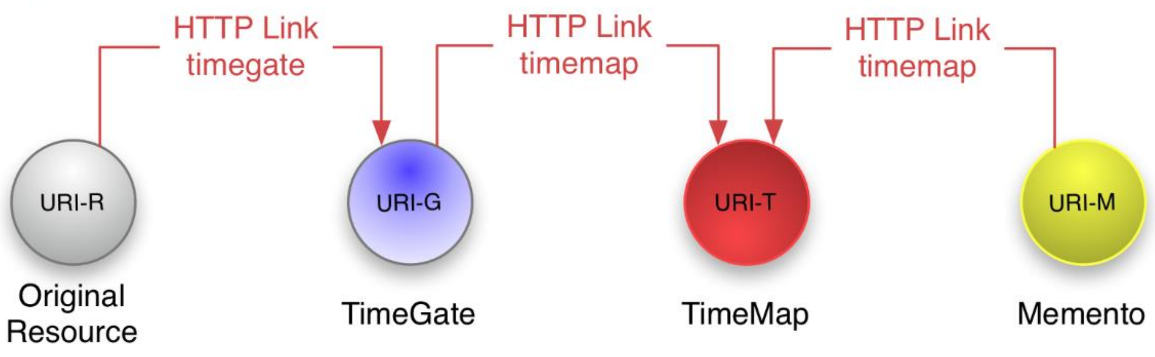
Memento Protocol

The Memento Protocol defines four concepts:

1. **Original Resource**: A Web resource that exists or used to exist on the live Web for which we want to find a prior version. By prior version is meant a Web resource that encapsulates what the Original Resource was like at some time in past.
2. **Memento**: A Web resource that is a prior version of the Original Resource, i.e. that encapsulates what the Original Resource was like at some time in the past.
3. **TimeGate**: A Web resource that "decides" on the basis of a given datetime, which Memento best matches what the Original Resource was like around that given datetime.
4. **TimeMap**: A TimeMap for an Original Resource is a resource from which a list of URIs of Mementos of the Original Resource is available.

Memento decides between **resources** (URI-R), **timemap** (URI-T), **timegates** (URI-G) and **mementos** (URI-Mx).

This picture provides an architectural overview of how the Memento framework supports batch discovery of Mementos:



Retrieve available versions of a Logistics Object – TimeMap (GET)

A **TimeMap** (URI-T) for an Original Resource (URI-R) is a machine-readable document that lists the Original Resource itself, its TimeGate, and its Mementos as well as associated metadata such as archival datetime for Mementos. TimeMaps are exposed by systems that host prior versions of Original Resources and allow for batch discovery of Mementos.

As explained before, the existing versions (**TimeMap**) endpoint URL can be returned in the HTTP `Link` header when performing a `GET` request on a Logistics Object.

Example:

```
GET https://www.onerecordcargo.org/my_airline/shipment_123456/
```

Would return as one of the headers:

```
Link: <https://www.onerecordcargo.org/my_airline/shipment_123456/timemap>;
rel="timemap"
```

It would be then possible to get the version history of an object by appending "/`timemap`" to its base URL. Each memento will be listed, with the memento label as the title.

Request

HTTP Request type: **GET**

```
GET logisticsObjectURI/timemap
Host: myonerecordserver.net
Accept: application/ld+json
```

Example:

```
GET https://www.onerecordcargo.org/my_airline/shipment_123456/timemap
```

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ 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 TimeMap in the format that has been requested in the Accept header of the request.

```
{
  "@id": "https://www.onerecordcargo.org/my_airline/shipment_123456/timemap",
  "@type": [
    "https://onerecord.iata.org/Timemap"
  ],
  "https://onerecord.iata.org/Timemap#mementos": {
    "@type": [
      "https://onerecord.iata.org/Mementos"
    ],
    "https://onerecord.iata.org/Mementos#firstMemento":
"https://www.onerecordcargo.org/my_airline/shipment_123456/mementos/677b715e-3b46-
492b-93df-bf9cc059a111",
    "https://onerecord.iata.org/Mementos#lastMemento":
"https://www.onerecordcargo.org/my_airline/shipment_123456/mementos/677b715e-3b46-
492b-93df-bf9cc059a111",
    "https://onerecord.iata.org/Mementos#list": {
      "@type": [
        "https://onerecord.iata.org/MementoList"
      ],
      "https://onerecord.iata.org/MementoList#mementoEntry": [
        {
          "@type": [
            "https://onerecord.iata.org/MementoEntry"
          ],
          "https://onerecord.iata.org/MementoEntry#datetime": "2020-09-
07T12:00:00.183Z",
          "https://onerecord.iata.org/MementoEntry#label": "v1",
          "https://onerecord.iata.org/MementoEntry#memento":
"https://www.onerecordcargo.org/my_airline/shipment_123456/mementos/677b715e-3b46-
492b-93df-bf9cc059a111"
        }
      ]
    }
  },
  "https://onerecord.iata.org/Timemap#original": "
https://www.onerecordcargo.org/my_airline/shipment_123456",
  "https://onerecord.iata.org/Timemap#timegate":
"https://www.onerecordcargo.org/my_airline/shipment_123456/timegate"
}
```

Code	Description	Response body
200	TimeMap returned successfully	TimeMap model
401	Not authenticated	Error model
403	Not authorized to retrieve TimeMap	Error model
404	Logistics Object/TimeMap Not Found	Error model

Retrieve a version of a Logistics Object at a certain moment in time – TimeGate (GET)

A **TimeGate** supports content negotiation in the datetime dimension. When negotiating with the TimeGate, the HTTP client uses an `Accept-Datetime` header to express the desired datetime of a prior/archived version of

URI-R (original resource). The TimeGate responds with the location of a matching version, named a Memento (URI-M1 or URI-M2), allowing the HTTP client to access it. Using the Memento-Datetime header, Mementos express their version/archival datetime.

`GET` TimeGate request should contain an `Accept-Datetime` header and the response will contain a `Link` header to a previous version of the original resource (Memento) closest to the date sent in the header**.**

Request

HTTP Request type: **GET**

```
GET logisticsObjectURI/timegate
Host: myonerecordserver.net
Accept: application/ld+json
Accept-Datetime: 2020-04-21T10:36:42+00:00
```

Example:

```
GET https://www.onerecordcargo.org/my_airline/shipment_123456/timegate
```

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ application/ld+json
Accept-Datetime	A date for which we would want to know the closest Memento

Response

A positive `HTTP 204` response is expected to a `GET` Timegate request. No response body is expected.

`Link` and `Memento-Datetime` headers should be returned in the response:

```
Link: <https://www.onerecordcargo.org/my_airline/shipment_123456/mementos/addd87ed-f208-4e8a-8b69-d7cf698c8d4e>; rel="memento"

Memento-Datetime: 2020-04-21T11:00:42+00:00
```

Code	Description	Response body
204	Timegate Link returned correctly	No content
401	Not authenticated	Error model
403	Not authorized to retrieve Timegate	Error model

404	Logistics Object/Timegate Not Found	Error model

Retrieve a Memento of a Logistics Object (GET)

Request

HTTP Request type: **GET**

```
GET logisticsObjectURI/mementos/mementoId
Host: myonerecordserver.net
Accept: application/ld+json
```

Example:

```
GET https://www.onerecordcargo.org/my_airline/shipment_123456/mementos/addd87ed-
f208-4e8a-8b69-d7cf698c8d4e
```

HTTP Headers

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

Header	Description
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: ▪ application/x-turtle or text/turtle ▪ 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 Memento in the format that has been requested in the Accept header of the request.

```
{
   "@id":
"https://www.onerecordcargo.org/my_airline/shipment_123456/mementos/addd87ed-f208-
4e8a-8b69-d7cf698c8d4e",
  "@type": [
    "https://onerecord.iata.org/Memento"
  ],
  "https://onerecord.iata.org/Memento#created": "2020-09-07T11:57:42.597Z",
  "https://onerecord.iata.org/Memento#createdBy": "admin",
  "https://onerecord.iata.org/Memento#label": "v1",
  "https://onerecord.iata.org/Memento#original": "
https://www.onerecordcargo.org/my_airline/shipment_123456"
}
```

Code	Description	Response body
200	Memento returned successfully	Memento model
401	Not authenticated	Error model

403	Not authorized to retrieve Memento	Error model
404	Logistics Object/Memento Not Found	Error model

Create a Memento

The creation of a Memento for a Logistics Object should be an internal process of the ONE Record Server. Making available a POST endpoint for the external parties to use is not mandatory.

Bibliography

Memento Protocol: http://mementoweb.org/

Apache Marmotta: http://marmotta.apache.org/platform/versioning-module.html

HTTP Framework for Time-Based Access to Resource States – Memento: https://tools.ietf.org/html/rfc7089

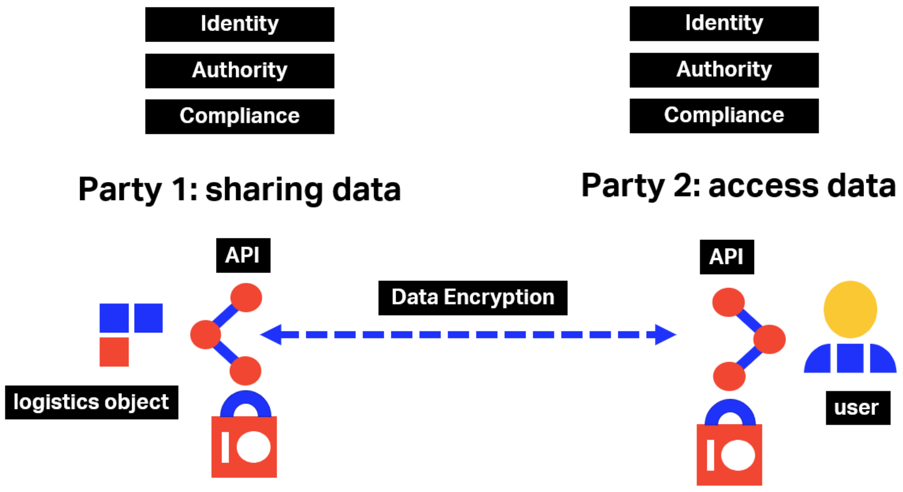
https://www.slideshare.net/hvdsomp/an-httpbased-versioning-mechanism-for-linked-data

Security in ONE Record

This section discusses the security options for the ONE Record API that governs the connectivity between ONE Record clients and servers on the Internet of Logistics.

Background

When exchanging data, each party needs to know with certainty the true identity of the other party and that they have the authority to receive or share the data. They also need to be certain that the data being shared is private, secured and confidential and cannot be intercepted or changed by any unauthorized third party. The ONE Record security framework works globally and for all stakeholders in the full logistics and transport supply chain, and in compliance with corporate and local data security requirements.



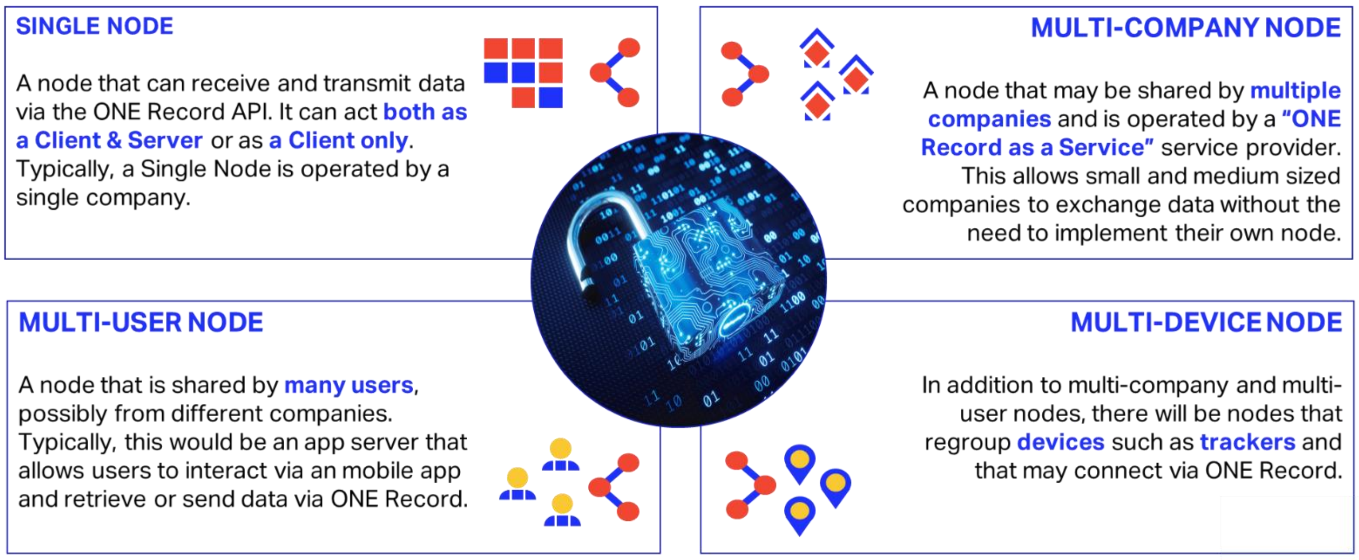
IT and business experts from the industry from ONE Record Task Force have explored and discussed the different connectivity configurations within the Internet of Logistics and two models – possibly complementary – were retained. This section presents these models and showcases the implications and benefits that they bring to the air cargo industry.

IoL Nodes

The connectivity between clients and servers on the Internet of Logistics (IoL) is governed by the ONE Record API and Security specifications. Since the IoL is proposed to include end-to-end transport and logistics chain connectivity, there are many configurations for interaction between stakeholders and their systems, both large and small, that may include all types such as shippers, forwarders, airlines, ground handlers, airports, customers as well as entities from other modes. The following non-exhaustive list of IoL Nodes types have been identified:

5. Single Node: A node that can receive and transmit data via the ONE Record API. It can act **both as a Client & Server** or as **a Client only**. Typically, a Single Node is operated by a single company.
6. Multi-Company Node: A node that may be shared by **multiple companies** and is operated by a **"ONE Record as a Service"** service provider. This allows small and medium sized companies to exchange data without the need to implement their own node.
7. Multi-User Node: A node that is shared by **many users**, possibly from different companies. Typically, this would be an app server that allows users to interact via an mobile app and retrieve or send data via ONE Record.
8. Multi-Device Node: In addition to multi-company and multi-user nodes, there will be nodes that regroup **devices** such as **trackers** and that may connect via ONE Record.

Together these are referred to as **<u>IoL Nodes</u>**.



SINGLE NODE

A node that can receive and transmit data via the ONE Record API. It can act **both as a Client & Server** or as **a Client only**. Typically, a Single Node is operated by a single company.

MULTI-COMPANY NODE

A node that may be shared by **multiple companies** and is operated by a **"ONE Record as a Service"** service provider. This allows small and medium sized companies to exchange data without the need to implement their own node.

MULTI-USER NODE

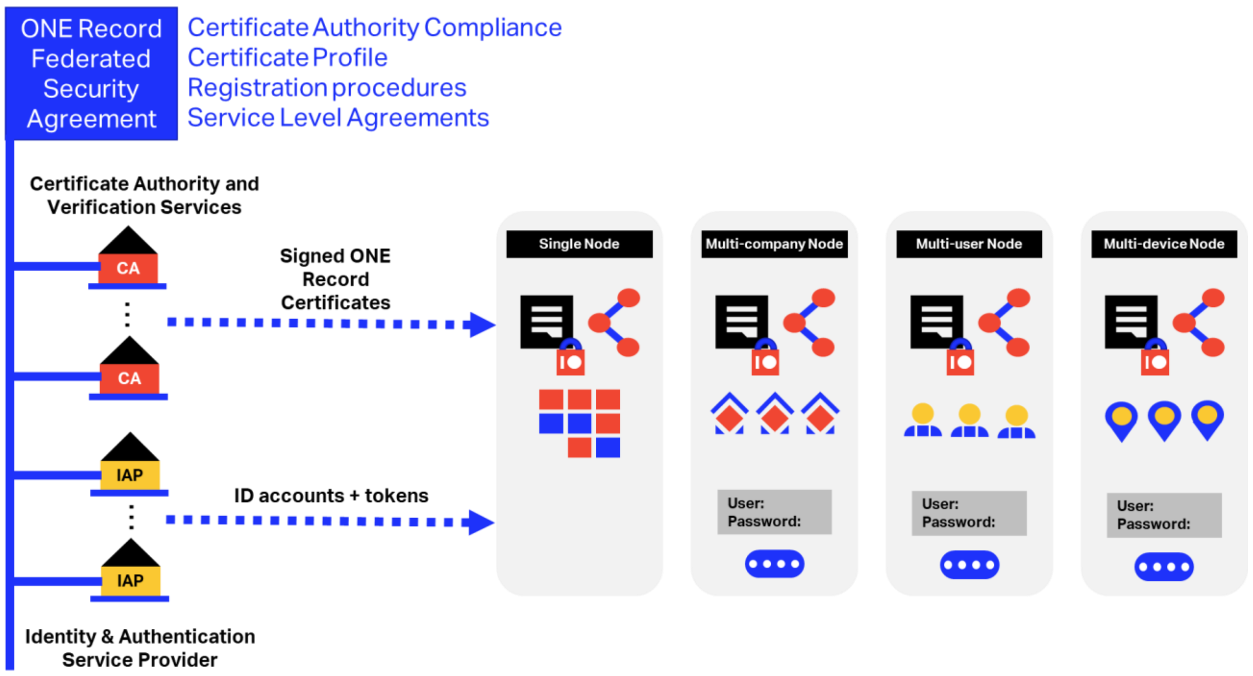
A node that is shared by **many users**, possibly from different companies. Typically, this would be an app server that allows users to interact via an mobile app and retrieve or send data via ONE Record.

MULTI-DEVICE NODE

In addition to multi-company and multi-user nodes, there will be nodes that regroup **devices** such as **trackers** and that may connect via ONE Record.

IoL Nodes

Below is an example of a small IoL network where the different types of IoL Nodes interact using the ONE Record API and Security specifications.



From an IoL perspective, these are all clients and services but, in the Multi-User, Multi-Company and Multi-Device cases, the actual IoL API is shared between multiple companies and/or users which impacts the identification, authentication and authorization models for ONE Record.

Definitions and Acronyms

Here are some definitions and acronyms used in the following sections:

1R-ID URI that refers to a 1R identity of the node, with the form https://<FQDN>/<COMPANYID>	**ENDPOINT** Either a 1R-CLIENT or a 1R-SERVER application
1R-SERVER Application that accepts ONE Record API requests from a 1R-CLIENT	**END USER** Entity that uses a 1R-CLIENT application
1R-CLIENT Application that sends ONE Record API requests to a 1R-SERVER	**TOKEN** Authorization token, refers to the OAuth2 standard
TRUSTED CA Certificate issuer that is approved by IATA	**IAP** Identity & Authentication Provider. Authentication service that implements OAuth2

Practical implementation of a first ONE Record Security working concept

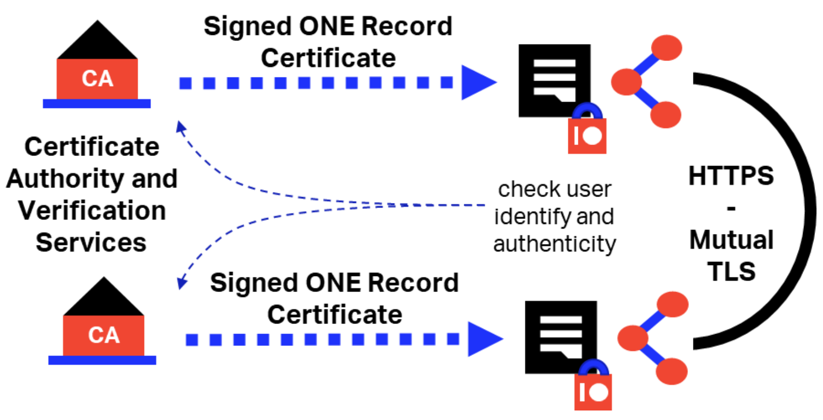
ONE Record Security specifications are built around two concepts: **mutual TLS** (short for Transport Layer Security) and **OAuth2** (an authorization protocol). Mutual TLS secures all the Node-to-Node channels whereas OAuth2 adds an extra security layer for identification and authentication.
IATA has conceptualized a first practical implementation, which is promoted as the official security model for ONE Record. This implementation is split in two modules:

- **TLS authentication support**, including the definition of certificate profiles and practical use of digital certificates for client and server certificates;
- **Token-based authentication support**, including a practical implementation using OAuth2 as authorization protocol, based on the concept of a "Trusted Identity Provider".

The next sections provide more detailed descriptions on the chosen security models and common use cases.

TLS Authentication Model

The first ONE Record Security layer is based on Mutual TLS (with TLS1.2), where the Nodes identify themselves via digital certificates (X509) issued by Certificate Authorities (CA) that are recognized by the ONE Record community. Therefore, there is a prerequisite that the ONE Record Community operates and manages one or more CAs that meets the registration and issuance requirements of the Internet of Logistics.


Signed ONE Record Certificate
Certificate Authority and Verification Services
CA
check user identify and authenticity
HTTPS - Mutual TLS
Signed ONE Record Certificate
CA

TLS in ONE Record

Digital Certificates and PKI

Digital Certificates enable to:

- **Identify the entities** (persons, applications or objects) connecting to an application;
- **Encrypt** the data, to ensure the **confidentiality**, either during the communication, or permanently;
- **Digitally sign** the data, to ensure both:
  - the **authenticity**, and
  - the **integrity** of the data.

TLS Overview

TLS defines a handshake procedure to authenticate the parties and encrypt the communication channel.

Key points to consider:

- TLS Authentication can be used with certificates for server side only, or for both client and server-side authentication;
- A certificate can allow an application to act as client or as a server;
- The same certificate can be used for other purposes, like digitally signing the messages;
- TLS is standardized and supports common server applications and development frameworks.

TLS in ONE Record

IATA defines two digital certificate profiles:

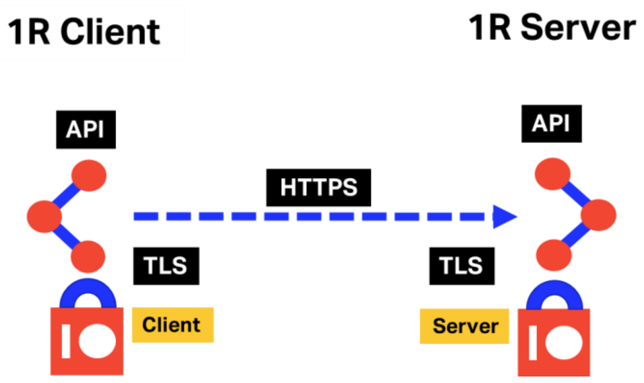
- **Client Certificate.** This certificate can authenticate a 1R-Client, and it includes one or more “ONE Record IDs” in the form of URI, that represent the endpoints that are authorized for the 1R-Client, to receive ONE Record responses. These ONE Record IDs are included in the certificate as SAN URI extensions. IATA recommends that these certificates are issued by a publicly trusted CA that is operating according to the PKI industry regulations and that is conforming to a WebTrust or equivalent independent audit criteria.

- **Server Certificate.** This is valid for server authentication and it is similar to common TLS certificates used to protect web sites. The main particularity and requirement are that, to cover the needs of Multi-Company Nodes, the Server Certificate must contain all the internet domains that can be extracted from the list of authorized 1R-IDs.

Please see the section below for more details about the client and server certificates requirements.

Use-Case: TLS Client/Server Authentication

The 1R-Client needs to establish a secure connection to the 1R-Server, and both parties need to authenticate the other, so both will be required to present a valid digital certificate to establish the connection.



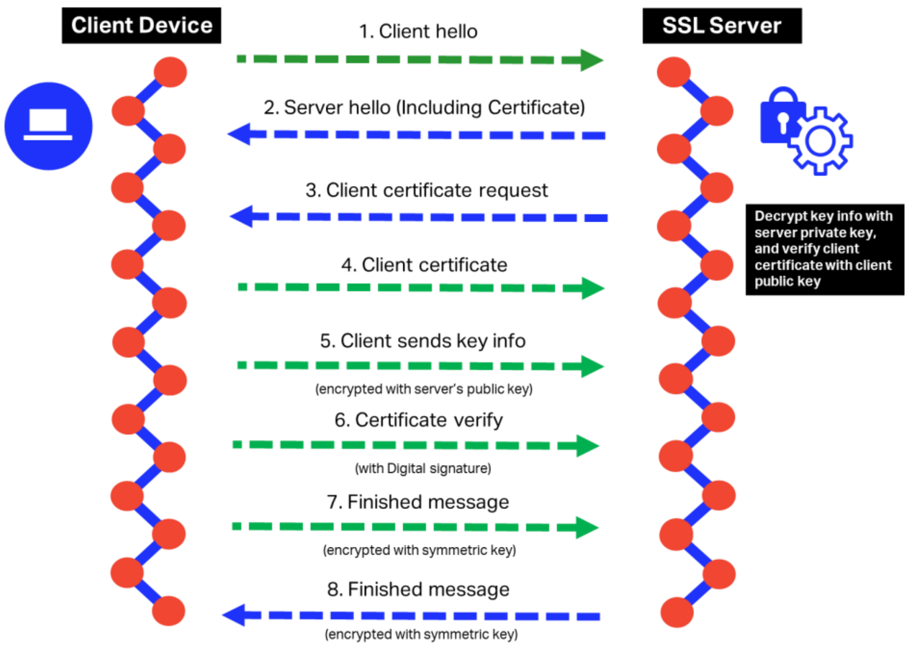
It is important to note that ONE Record uses a Publish/Subscribe approach, therefore:

- The 1R-Client sends a request to the 1R-Server and includes a callback URL;
- The 1R-Server will process asynchronously the request and will send a notification to the 1R-Client using the callback URL.

This means that when the response from the 1R-Server is ready, it will be sent to the 1R-Client as a new TLS flow, initiated by the 1R-Server, but it is now acting as a TLS client as it pushes data back to the 1R-Client. This means that both entities will operate alternatively as both client and server.

Traffic flow:

1. The 1R-Client Initiates the request using HTTPS, i.e to an SSL server.
2. The 1R-Server presents its certificate, and requests a Client certificate.
3. The TLS connection is established if the 1R-Server and 1R-Client are properly configured to require TLS authentication with Client certificate, and the CA issuing the certificate is included in the CA list of the 1R-Server.
4. The 1R-Client Accepts the connection if:
   a) the Server certificate comes from a Trusted CA and
   b) it is not expired or revoked
5. The 1R-Server Accepts the connection if:
   a) the Client certificate comes from a Trusted Source and
   b) it is not expired or revoked, and
   c) the Client certificate contains a 1R-ID that is allowed to make 1R requests.


Client Device
SSL Server
1. Client hello
2. Server hello (Including Certificate)
3. Client certificate request
4. Client certificate
5. Client sends key info
(encrypted with server's public key)
6. Certificate verify
(with Digital signature)
7. Finished message
(encrypted with symmetric key)
8. Finished message
(encrypted with symmetric key)
Decrypt key info with server private key, and verify client certificate with client public key

TLS Flow

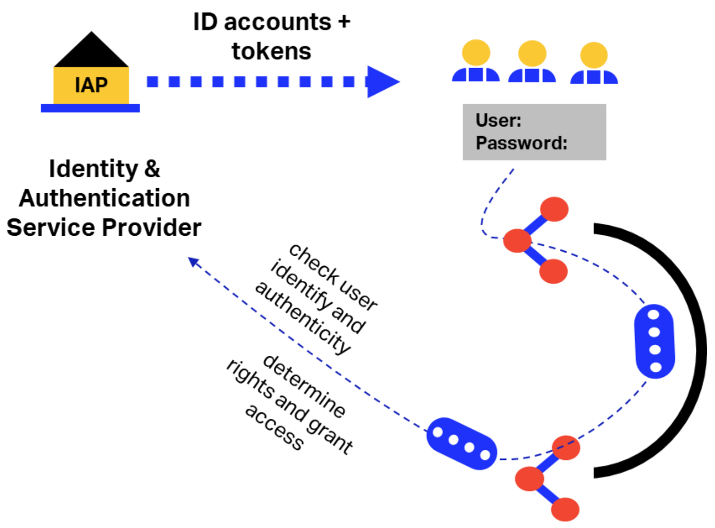
OAuth2 Authorization model

Overview

The mutual TLS model proposed for Node-to-Node data exchange is not enough to ensure identification for the Multi-Company/User/Device configurations.

Mutual TLS can only identify the owner of the digital certificate. When that server is shared by multiple companies and/or multiple users, this is insufficient for guaranteed identification of such companies and/or users for a 3rd party IoL node to apply its authorization and access policies.

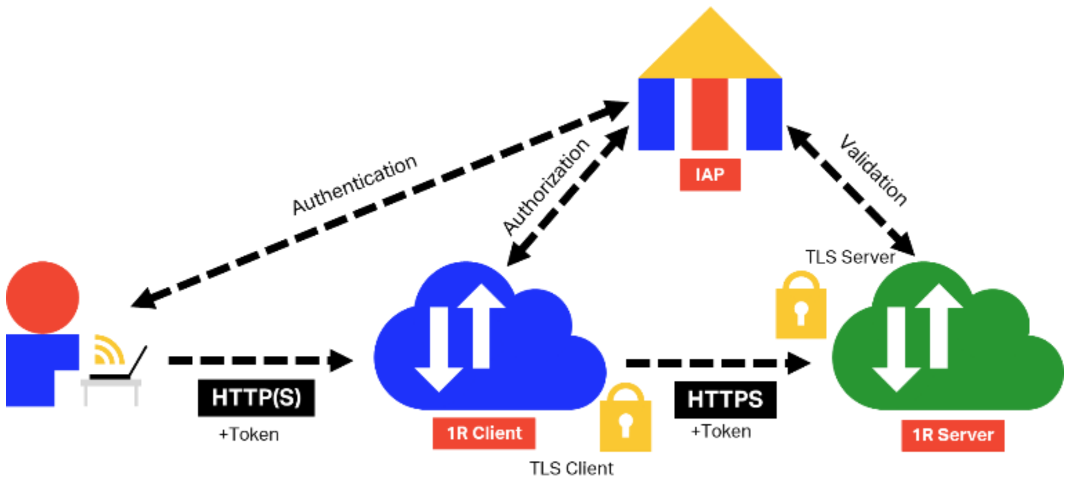
Although Multi-Company/User/Device Nodes could obtain a digital certificate for each of its platform users (i.e. companies) and ensure their identity as such using the mutual TLS model, the ONE Record Task Force proposes an complementary identification and authentication approach.


ID accounts + tokens
IAP
Identity & Authentication Service Provider
User:
Password:
check user identify and authenticity
determine rights and grant access

It is proposed to use the services of one or more Identity & Authentication Provider/s (IAP) that can register companies, users and other entities such as devices based on policies agreed by the ONE Record community. These IAPs provide authentication and validation as requested.

This second security model intends to cover the use cases where a multi-company/user/device client application is connecting to a ONE Record service, and the server needs to take an authorization decision based on the identity of the end client making the request. This authorization will be based on the use of claims-based technology, in particular:

- The end client is authenticated by a trusted identity provider who has a digital certificate issued by a ONE Record CA;
- The client can attach to the request an authorization token issued by the trusted identity provider;
- The server application can extract from the token the identity of the end client and can also ensure that his identity is endorsed by a trusted identity provider.



Using OAuth2 Tokens

Calling ONE Record API over TLS and OAuth2

- 1R API requires TLS mutual authentication;
- 1R API requires OAuth2 authentication using OpenID Connect Code Flow;
- 1R API validates TLS client authentication;
- 1R API validates ID token to check if the token is signed by a trusted IAP.

OAuth2 Auth - OpenID Connect Code Flow

6. Client prepares an authentication request containing the desired request parameters.
7. Client sends the request to the authorization server.
8. Authorization server authenticates the end-user.
9. Authorization server obtains end-user consent/authorization.
10. Authorization server sends the end-user back to the client with an authorization code.
11. Client requests a response using the authorization code at the token endpoint.
12. Client receives a response that contains an ID token and access token in the response body.
13. Client validates the ID token and retrieves the end-user's subject identifier.

Token Validation Process

The security model enforces the concept of "trusted identity provider", this implies that the server must ensure that the token is issued by a trusted party. This validation implies these steps:

- Parsing;
- Loading IAP's public key;
- Validate signature;
- TRUST IAP validation through by doing validate if the public key linked with a trusted certificate.

Federated trust centres

PKI Compliance

Mutual TLS requires the use of digital certificates that were issued by trusted Certificate Authorities (CAs). It is proposed that trusted organizations like IATA and/or other industry bodies provide such CA services to their IoL stakeholders.

It is essential that all these IoL CAs operate on the same basis, i.e. that they implement the same registration and operational policies. It is therefore proposed that these CA's federate under a common agreement that they jointly develop, sign and maintain.

Certificate profile 1 – Client

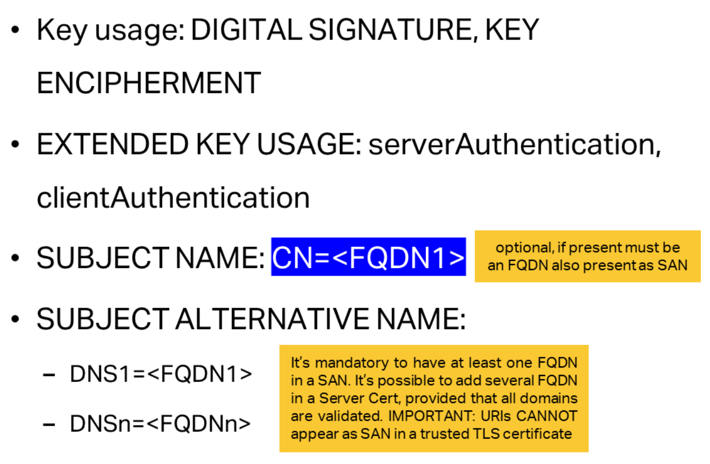
It is recommended to use client certificates with only the clientAuthentication EKU, in order to simplify the issuance process and reduce audit complexity.

- Key usage: DIGITAL SIGNATURE
- EXTENDED KEY USAGE: clientAuthentication
- SUBJECT NAME: CN=<COMPANY_ID>
- SUBJECT ALTERNATIVE NAME:
  - URI1=<1R_ID1>
  - URIn=<1R_IDn>

ONE Record Client Certificate

Certificate profile 2 – Server

It is recommended to use TLS server certificates of type "Domain Validated", that don't contain company name or other identity attributes, in order to simplify the issuance process and reduce audit complexity.



- Key usage: DIGITAL SIGNATURE, KEY ENCIPHERMENT
- EXTENDED KEY USAGE: serverAuthentication, clientAuthentication
- SUBJECT NAME: CN=<FQDN1>

optional, if present must be an FQDN also present as SAN

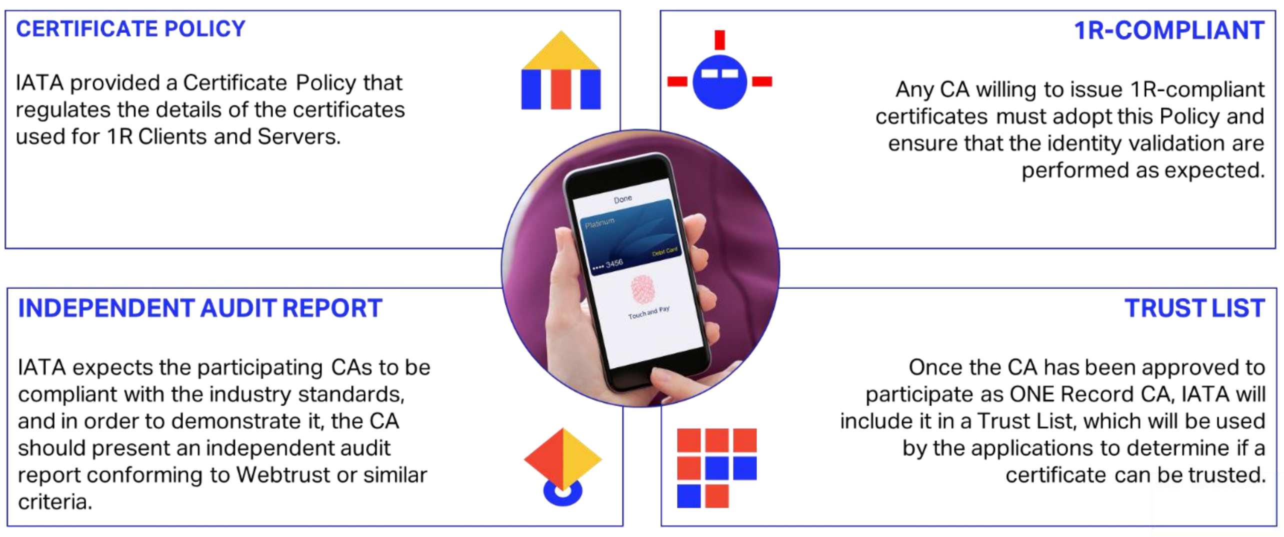
- SUBJECT ALTERNATIVE NAME:
  - DNS1=<FQDN1>
  - DNSn=<FQDNn>

It's mandatory to have at least one FQDN in a SAN. It's possible to add several FQDN in a Server Cert, provided that all domains are validated. IMPORTANT: URIs CANNOT appear as SAN in a trusted TLS certificate

ONE Record Server Certificate

Requirements for Certificate Authorities (CAs)

- Issue and authenticate valid ONE Record certificates
- Must be internationally accredited to issue public certificates
- Meets ONE Record requirements for registration and service levels
- Is federated with other certificate authorities and identity & authentication services



CERTIFICATE POLICY

IATA provided a Certificate Policy that regulates the details of the certificates used for 1R Clients and Servers.

1R-COMPLIANT

Any CA willing to issue 1R-compliant certificates must adopt this Policy and ensure that the identity validation are performed as expected.

INDEPENDENT AUDIT REPORT

IATA expects the participating CAs to be compliant with the industry standards, and in order to demonstrate it, the CA should present an independent audit report conforming to Webtrust or similar criteria.

TRUST LIST

Once the CA has been approved to participate as ONE Record CA, IATA will include it in a Trust List, which will be used by the applications to determine if a certificate can be trusted.

CA Requirements

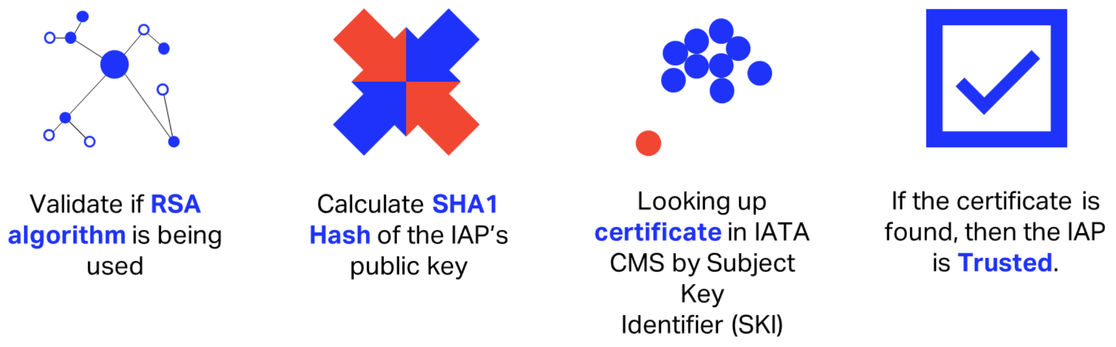
Trusted IAP

Similarly, the OAuth2 security model requires that Identity and Authentication Providers provide a service to their stakeholders that must also be operated on the same basis, i.e. using the same registration and operational policies.

It is therefore proposed that both CAs and IAPs operate under the same ONE Record and IoL data security agreement that allows CAs and IAPs to offer a federated trust environment to the transport and logistics industry.

Although ONE Record and IoL are open initiatives and aim to create a global network for data sharing and connectivity, these CA's and IAPs' will incur cost that they may pass on to the registrants. This should be considered the cost of security. It is very well possible that as the IoL grows, that the CA's and IAP's find business models that would allow them to provide these security services for free to their stakeholders.

Trusted IAP validation


Validate if **RSA algorithm** is being used
Calculate **SHA1 Hash** of the IAP's public key
Looking up **certificate** in IATA CMS by Subject Key Identifier (SKI)
If the certificate is found, then the IAP is **Trusted**.

IAP Validation

Resources available for interested parties

With the purposes of implementing this security concept in pilot or real projects, IATA has made available to all interested parties a platform that provides the necessary services for:

- Issuing Client and Server Digital Certificates, for TLS Authentication. This will facilitate to obtain certificates conforming to the profiles defined by IATA and issued out of a publicly trusted certification authority;

- Obtaining digital identities compatible with OpenID Connect and OAuth2, for token-based authorization purposes.

Additionally, IATA can deliver guidelines and sample code that illustrate the processes for client and server authentication, using TLS and/or OAuth2 methods.

WISeKey Proof of Concept

These specifications and the ONE Record Security Proof of Concept are the result of the cooperation with WISeKey company, which provided IATA with documentation and testing systems for certificates generations via a Certificate Authority and IAP integration.

Glossary

Term	Description
ACL	Access Control List
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 (IoL)	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-LD	JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases such as CouchDB and MongoDB.
Json Web Token (JWT)	JSON specification for a token format that includes a user defined payload and the option for encryption.
Linked Data	Linked Data empowers people that publish and use information on the Web. It is a way to create a network of standards-based, machine-readable data across Web sites. It allows an application to start at one piece of Linked Data and follow embedded links to other pieces of Linked Data that are hosted on different sites across the Web.
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	Server 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
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
WAC	Web Access Control