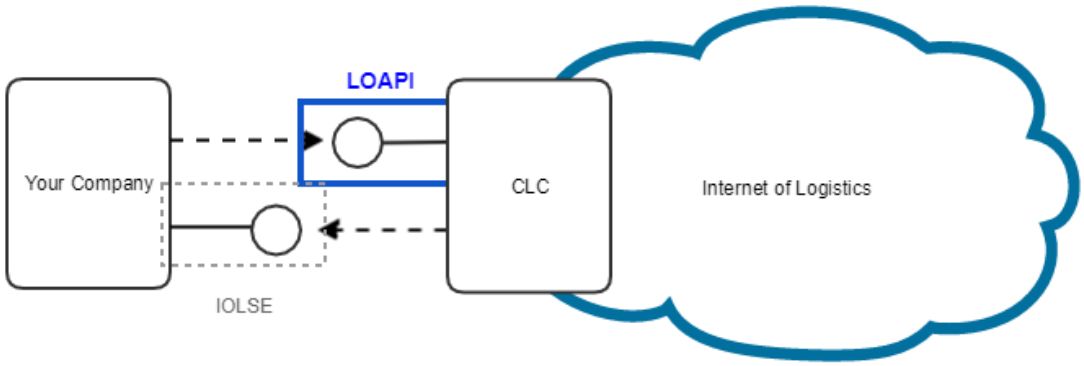
Logistics Object API

Contents

Overview

The Logistics Object API (LOAPI) is the key integration point to the Internet of Logistics. The LOAPI is a REST based API that supports the following operations:

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



Logistics Object Unique ID

A Logistics Object can be created, patched or read by using the LO Unique ID (URI). A LO Unique ID (URI) can be any URI that is guaranteed to be unique. An example of a LO Unique ID could be for example:

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

where

{CLC Domain} - The domain name associated with the CLC e.g. clc.ericsson.net

{license plate} - The preregistered Identifier for the company on CLC. e.g. lelme

{unique ID to identify LO} - A unique identifier within the company for the logistics object.

e.g https://clc.ericsson.net/lelmeeab/shippersinstruction-12345679

> The unique ID to identify the LO should be URL friendly. Avoid unsafe characters that include the blank/empty space and " < > # % { } | \ ^ ~ [ ] `.

Create Logistics Object

Publishes a logistics object resource to CLC.

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 LOAPI URI.

Request

HTTP Request type: **POST**

HTTP Headers

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

Authorization	A valid Bearer Token
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: • **application/x-turtle** or `text/turtle (recommended)` • **`application/ld+json`** `(recommended)` • `application/rdf+xml` • `application/n-triples`
Content-Type	The content type that is contained with the HTTP body. Valid content types include: • **application/x-turtle** or `text/turtle (recommended)` • **`application/ld+json`** `(recommended)` • `application/rdf+xml` • `application/n-triples`

HTTP Body

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

- http://tcfassociation.com/schema/ShippersInstruction
- http://tcfassociation.com/schema/ShippersInstructionConfirmation
- http://tcfassociation.com/schema/HAWB
- http://tcfassociation.com/schema/AirwayBill
- http://tcfassociation.com/schema/FlightManifest
- http://tcfassociation.com/schema/CMR
- http://tcfassocation.com/schema/BillOfLading
- http://tcfassocation.com/schema/LogisticsVehicle

In the future this would be logistics objects as defined by DCF

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 in the Internet of Logistics or expired token	Error model
403	Not authorized to publish the Logistics Object to the Internet of Logistics	Error model
415	Unsupported Content Type	Error model

Read Logistics Object

Retrieves a Logistics Object resource from the Internet of Logistics.

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

Request

HTTP Request type: **GET**

HTTP Headers

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

Authorization	A valid Bearer Token that is provided by the CLC Authentication and Authorization Service
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: • **application/x-turtle** or `text/turtle (recommended)` • **`application/ld+json`** `(recommended)` • `application/rdf+xml` • `application/n-triples`

Response

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

Code	Description	Response Body
200	The request to retrieve the logistics object has been successful	DCF Logistics Object Model
401	Not authenticated in the Internet of Logistics or expired token	Error model

403	Not authorized to retrieve the Logistics Object	Error model
406	Unsupported Accept Type	Error model

Patch Processing Updates to Logistics Object

The PATCH request should be used to provide:

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

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

Request

HTTP Request type: **PATCH**

HTTP Headers

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

Authorization	A valid Bearer Token that is provided by the CLC Authentication and Authorization Service
Accept	The content type that you want the HTTP response to be formatted in. Valid content types include: • **application/x-turtle** or `text/turtle (recommended)` • **`application/ld+json`** `(recommended)` • `application/rdf+xml` • `application/n-triples`
Content-Type	The content type that is contained with the HTTP body. Valid content types include: • **application/x-turtle** or `text/turtle (recommended)` • **`application/ld+json`** `(recommended)` • `application/rdf+xml` • `application/n-triples`

HTTP Body

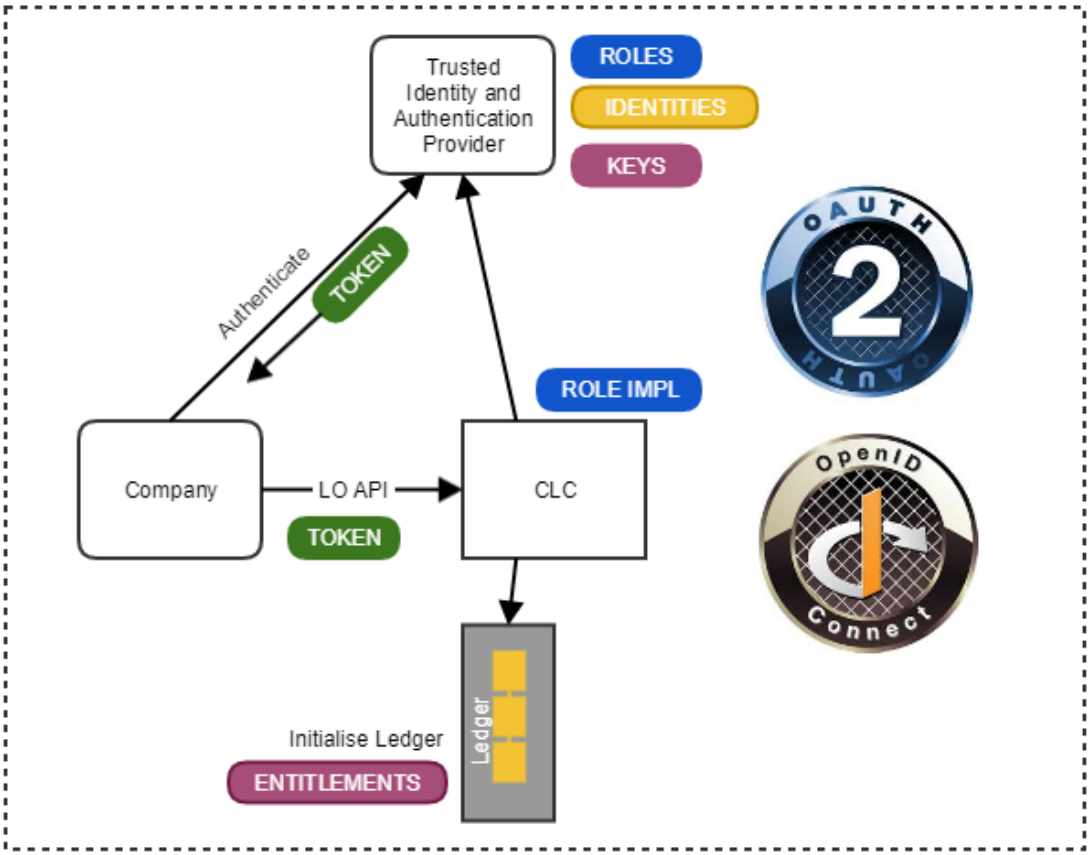
The HTTP body MUST be a valid DCF Status Update or Partner Access in the format as specified by the Content-Type in the header. The following is the list of supported PATCH request bodies:

- http://tcfassociation.com/schema/StatusUpdate
- http://tcfassociation.com/schema/PartnerAccess

Response

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

Security



Trusted Identity and Authentication Providers (IAP)

Within the Internet of Logistics (IoL) there will be a need for trusted Identity and Authentication Providers (IAP). An example of an IAP could be DCF or IATA.

Each IAP would control which other IAPs that they trust. E.g IATA could trust DCF and choose not to trust the "Dodgy Logistics Association".

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

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

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


{"keys":
       [
         {
          "kty":"RSA",
          "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMstn64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbISD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqbw0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
          "e":"AQAB",
          "alg":"RS256",
          "use": "sig",
          "key_ops": "verify",
          "kid":"dcf1"
         },
         {
          "kty":"RSA",
          "n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc_BJECPebWKRXjBZCiFV4n3oknjhMstn64tZ_2W-5JsGY4Hc5n9yBXArwl93lqt7_RN5w6Cf0h4QyQ5v-65YGjQR0_FDW2QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbISD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G_xBniIqbw0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",
          "e":"AQAB",
          "alg":"RS256",
          "use": "sig",
          "key_ops": "verify",
          "kid":"iata1"
         }
       ]
}
```

Authentication

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

IoL participants MUST authenticate against an IAP.

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

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

JWT Access Tokens from an IAP

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

JOSE Header

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

Payload

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

Signature

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

Gap

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

Token Verification

When an IoL consumer or provider receives a request from another IoL participant with a JWT they would need to first verify the JWT before accepting the request. The verification of the JWT would include:

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

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

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

Authorization

The two aspects to authorization in the IoL are:

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

Authorization to a Logistics Object

Authorization to a logistics object can be given in two ways:

1. **Explicitly in the Logistics Object** - A company can be given access to a LO by specifying the companies IoL identifier in the LO.
2. **Using the LO API to PATCH the LO** with additional partner access. Any company that has access to the LO can cascade the trust to other companies within the IoL.

Field level authorization within a Logistics Object

Field level authorization within a LO is possible using standardized roles in the IoL. Based on the users role CLC could make a decision to only provide certain field elements within the LO to the requestor.