monotch

SMART MOBILITY PLATFORMS

# TLEX-BROKER-ADMIN Interface v1.1.1

# Table of contents

# 1 Versioning

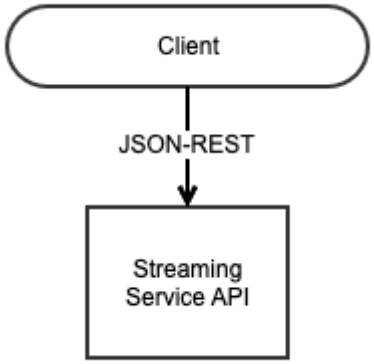
This document is using a versioning scheme that indicates the version of this TLEX interface and tracks the revisions of this document. This version scheme is <interface version major>.<interface version minor>.<document revision>. The first two version numbers (major and minor) indicate the version of the interface and only change when there is technical change in the described interface. Major version is only bumped when there is compatibility breaking change. Minor version is bumped on trivial, non breaking changes of the interface. The last version number indicates the revision of this document.

| Version | Date | Author | Changes |
|---|---|---|---|
| 1.0.0 | 13 Mar 2017 | L. Rijneveld | Initial specification for TLEX release v1.0 |
| 1.1.0 | 25 Jul 2017 | L. Rijneveld | Update for TLEX release v1.1<br>TLC type field in response of /tlcs end point |
| 1.1.1 | 23 Mar 2020 | L. Rijneveld | Improved layout and formatting |

# 2 Overview

The administrative interface for Brokers with TLEX is based on a JSON-REST API and is used for:

1. Managing authorizations;
2. Managing authorizationtokens;
3. Requesting active sessions;
4. Requesting session logs.



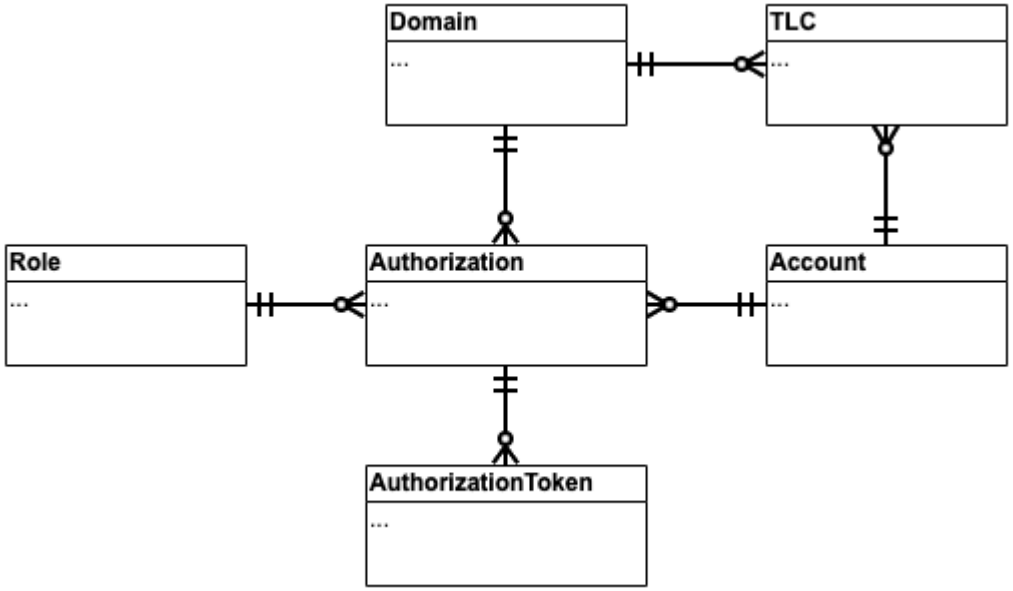
# 3 API

## 3.1 Authentication and authorization

The authentication of the Client will be based on a "authorization token". This "authorization token" needs to be passed as the "X-Authorization" request header value. The authorization token needs to belong to an "BROKER_ADMIN".

## 3.2 Authorization model

The API authorization model is illustrated in the following entity relation diagram:



### 3.2.1 Entities

| Entity | Description |
| --- | --- |
| Account | The identity of the authorization holder. An account can own:<br>1. TLC registrations<br>2. authorizations |
| Authorization | The authorization to use the API, which is a combination of:<br>1. Account<br>2. Domain<br>3. Role<br>It is possible for an account to have more then one authorization per domain. Since the API request are performed in context of one specific key, the API requests are always in context of one specific authorization. |
| Role | The role of the authorization holder (in context of the authorization). |
| Domain | The domain to which an authorization applies to. |
| TLC | The registration of a TLC within a certain domain. |
| AuthorizationToken | The authorization token. One authorization can have multiple authorization tokens. |

### 3.2.2 Roles

The following roles have been defined for Brokers:

| Name | Access | Access scope | Intended for |
|---|---|---|---|
| Broker administrator (BROKER_ADMIN) | Can manage own "Broker system" authorizations within the domain scope<br>Can manage own "Broker system" authorization tokens within the domain scope<br>Can manage all own broker sessions within the domain scope<br>Can view all own broker session logs and metrics within the domain scope<br>Can view all TLC registrations within the domain scope | Domain, Account | Broker organisations |
| Broker system (BROKER_SYSTEM) | Can create broker sessions within the domain scope<br>Can view all TLC registrations within the domain scope | Domain, Account | Broker systems |
| Broker analyst (BROKER_ANALYST) | Can view broker session logs and metrics within the domain scope<br>Can view all TLC registrations within the domain scope | Domain, Account | Analysts/engineers responsible for Broker systems |

## 3.3 API endpoints

| API endpoint | | | Authorization role access scope (domain/account/none) | | |
|---|---|---|---|---|---|
| **Method** | **URI** | **Description** | **BROKER_ADMIN** | **BROKER_SYSTEM** | **BROKER_ANALYST** |
| POST | /sessions | Creates a new streaming session | ACCOUNT | ACCOUNT | NONE |
| GET | /sessions | Retrieves all active streaming sessions | ACCOUNT | ACCOUNT | NONE |
| GET | /sessions/<token> | Retrieves one active streaming session | ACCOUNT | ACCOUNT | NONE |
| PUT | /sessions/<token> | Updates one active streaming session | ACCOUNT | ACCOUNT | NONE |
| DELETE | /sessions/<token> | Ends one active streaming version | ACCOUNT | NONE | NONE |
| GET | /sessionlogs | Retrieve all session logs | ACCOUNT | NONE | ACCOUNT |
| GET | /sessionlogs/<token> | Retrieve one specific session's log | ACCOUNT | NONE | ACCOUNT |
| GET | /tlcs | Gets all TLC registrations | DOMAIN | DOMAIN | DOMAIN |
| GET | /tlcs/<uuid> | Retrieve one specific TLC registration | DOMAIN | DOMAIN | DOMAIN |
| POST | /authorizations | Create a new authorization | ACCOUNT | NONE | NONE |
| GET | /authorizations | Retrieve all authorizations | ACCOUNT | NONE | NONE |
| GET | /authorizations/<uuid> | Retrieves one specific authorization | ACCOUNT | NONE | NONE |
| PUT | /authorizations/<uuid> | Updates one specific authorization | ACCOUNT | NONE | NONE |
| DELETE | /authorizations/<uuid> | Removes on specific authorization | ACCOUNT | NONE | NONE |

| API endpoint | | | Authorization role access scope (domain/account/none) | | |
|---|---|---|---|---|---|
| **Method** | **URI** | **Description** | **BROKER_ADMIN** | **BROKER_SYSTEM** | **BROKER_ANALYST** |
| POST | /authorizationtokens | Create a new authorization token | ACCOUNT | NONE | NONE |
| GET | /authorizationtokens | Retrieve all authorization tokens | ACCOUNT | NONE | NONE |
| GET | /authorizationtokens/<uuid> | Retrieves one specific authorization token | ACCOUNT | NONE | NONE |
| PUT | /authorizationtokens/<uuid> | Updates one specific authorization token | ACCOUNT | NONE | NONE |
| DELETE | /authorizationtokens/<uuid> | Removes on specific authorization token | ACCOUNT | NONE | NONE |

## 3.3.1 Sessions

### 3.3.1.1 POST /sessions

Creates a new streaming session.

#### 3.3.1.1.1 Request

```
POST <API base URL>/sessions HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json

{
    "domain": "<domain>",
    "type": "<type>",
    "protocol": "<protocol>",
    "details": {
        <protocol details>
    }
}
```

| Name | Description |
|---|---|
| domain | Sessions are created within a specific domain, identified by a single string<br><br>Only sessions created for the same domain will be able to stream data to each other |
| type | The session type; must be “Broker” |
| protocol | The session protocol; must be “TCPStreaming_Multiplex” |
| details | Session protocol specific details for creating the session |

#### 3.3.1.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "<token>",
    "domain": "<domain>",
    "type": "<type>",
    "protocol": "<protocol>",
    "details": {
        <protocol details>
    }
}
```

| Name | Description |
| --- | --- |
| token | The token for the created session<br>⚠ This token is unique and can only be used once for establishing a TCP connection; if the session expires or ends (TCP disconnect) a new session needs be created to obtain a new token |
| domain | See request |
| type | See request |
| protocol | See request |
| details | Session protocol specific details of the created session |

#### 3.3.1.1.3 Session type "Broker" with protocol "TCPStreaming_Multiplex"

TCP based multiplex streaming session for a payload broker.

Payloads sent by the client will be received by "TLC" session clients if the payload "TLC identifier" is within their scope.

Payloads sent by "TLC" session clients having a payload "TLC identifier" that is within the scope of the session will be received.

##### 3.3.1.1.3.1 Request details

```
{
    "securityMode": "<security mode>",
    "tlcIdentifiers": ["<TLC identifier>", "<TLC identifier>", ...]
}
```

| Name | Description |
|---|---|
| securityMode | Security mode of the streaming session<br>Must be one of the predefined values:<br>1. NONE<br>2. TLSv1.2 |
| tlcIdentifiers | The TLC identifiers for the session<br>Since the session is for multiple TLC's, payload data will be streamed with TLC identifier (see protocol datagram 0x05) |

#### 3.3.1.1.3.2 Response details

```
{
    "securityMode": "<security mode>",
    "tlcIdentifiers": ["<TLC identifier>", "<TLC identifier>", ...]
    "listener": {
        "host": "<host address>",
        "port": <port number>,
        "expiration": "<ISO 8601 date time>"
    },
    "keepAliveTimeout": "<ISO 8601 duration>",
    "clockDiffLimit": "<ISO 8601 duration>",
    "clockDiffLimitDuration": "<ISO 8601 duration>",
    "payloadRateLimit": <payload/second limit>,
    "payloadRateLimitDuration": "<ISO 8601 duration>",
    "payloadThroughputLimit": <KB/second limit>,
    "payloadThroughputLimitDuration": "<ISO 8601 duration>"
}
```

| Name | Description |
|---|---|
| securityMode | See request details |
| tlcIdentifiers | See request details |
| listener | The Streaming Service Node listener details for establishing the TCP connection |
| listener.host | The host address of the Streaming Service Node |
| listener.port | The TCP port of the Streaming Service Node's session listener |
| listener.expiration | The expiration date time of the listener in ISO 8601 date time format<br><br>If the TCP connection has not been established before this time the listener will expire and the streaming session will no longer be available<br><br>The default listener expiration will be 5 seconds after creation |
| keepAliveTimeout | The keep alive timeout duration of the TCP connection in ISO 8601 duration format<br><br>If no TCP data is received during the specified duration, the TCP connection will be terminated by the Streaming Service<br><br>If a Client receives no TCP data during the specified duration, it must terminate the TCP connection |

| Name | Description |
| --- | --- |
| clockDiffLimit | The maximum clock difference, in ISO 8601 duration format, allowed for the streaming session<br><br>If the average clock difference during the duration (see clockDiffLimitDuration) exceeds the limit the Streaming Service will terminate the TCP connection |
| clockDiffLimitDuration | The period, in ISO 8601 duration format, during which the average clock difference should not exceed the clockDiffLimit |
| payloadRateLimit | The maximum amount of payloads per second allowed for the streaming session<br><br>If the average amount of received payloads per second during the duration (see payloadRateLimitDuration) exceeds the limit the Streaming Service will terminate the TCP connection |
| payloadRateLimitDuration | The period, in ISO 8601 duration format, during which the average amount of received payloads per second should not exceed the payloadRateLimit |
| payloadThroughputLimit | The maximum amount of payload kilobytes (KB) per second allowed for the streaming session<br><br>If the average amount of received payload kilobytes (KB) per second during the duration (see payloadThroughputLimitDuration) exceeds the limit the Streaming Service will terminate the TCP connection |
| payloadThroughputLimitDuration | The period, in ISO 8601 duration format, during which the average amount of received payload kilobytes (KB) per second should not exceed the payloadThroughputLimit |

#### 3.3.1.1.3.3 Example

```
POST api/v1/sessions HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json

{
    "domain": "test",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "NONE",
        "tlcIdentifiers": ["NLZH0023", "NLZH0024", "NLZH0025"]
    }
}
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8",
    "domain": "test",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "NONE",
        "tlcIdentifiers": ["NLZH0023", "NLZH0024", "NLZH0025"],
        "listener": {
            "host": "n11.tlex.eu",
            "port": 40344,
            "expiration": "2016-11-17T16:07:56Z"
        },
        "keepAliveTimeout": "PT5S",
        "clockDiffLimit": "PT3S",
        "clockDiffLimitDuration": "PT60S",
        "payloadRateLimit": 1200,
        "payloadRateLimitDuration": "PT5S",
        "payloadThroughputLimit": 120,
        "payloadThroughputLimitDuration": "PT5S"
    }
}
```

#### 3.3.1.2 GET /sessions

Retrieves all active streaming sessions.

Intended for monitoring and debug purposes.

##### 3.3.1.2.1 Request

```
GET <API base URL>/sessions HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.1.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    <session>, <session>
]
```

#### 3.3.1.2.3 Example

```
GET api/v1/sessions HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
HTTP/1.1 200 OK
Content-Type: application/json

[
 {
    "token": "cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8",
    "domain": "test",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "NONE",
        "tlcIdentifiers": ["NLZH0023", "NLZH0024", "NLZH0025"],
        "listener": {
            "host": "n44.tlex.eu",
            "port": 40344,
            "expiration": "2016-11-17T16:07:56Z"
        },
        "keepAliveTimeout": "PT5S",
        "clockDiffLimit": "PT3S",
        "clockDiffLimitDuration": "PT60S",
        "payloadRateLimit": 1200,
        "payloadRateLimitDuration": "PT5S",
        "payloadThroughputLimit": 120,
        "payloadThroughputLimitDuration": "PT5S"
    }
 }
]
```

#### 3.3.1.3 GET /sessions/<token>

Retrieves the active streaming session with the given "token".

Intended for monitoring and debug purposes.

##### 3.3.1.3.1 Request

```
GET <API base URL>/sessions/<session token> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.1.3.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<session>
```

#### 3.3.1.3.3 Example

```
GET api/v1/sessions/cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8",
    "domain": "test",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "NONE",
        "tlcIdentifiers": ["NLZH0023", "NLZH0024", "NLZH0025"],
        "listener": {
            "host": "n44.tlex.eu",
            "port": 40344,
            "expiration": "2016-11-17T16:07:56Z"
        },
        "keepAliveTimeout": "PT5S",
        "clockDiffLimit": "PT3S",
        "clockDiffLimitDuration": "PT60S",
        "payloadRateLimit": 1200,
        "payloadRateLimitDuration": "PT5S",
        "payloadThroughputLimit": 120,
        "payloadThroughputLimitDuration": "PT5S"
    }
}
```

#### 3.3.1.4 PUT /sessions/<token>

Updates the protocol details of the active streaming session with the given "token".
Intended to support the addition and removal of TLC identifiers for multiplex sessions.

##### 3.3.1.4.1 Request

```
PUT <API base URL>/sessions/<session token> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.1.4.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<protocol details>
```

#### 3.3.1.4.3 Example

```
PUT api/v1/sessions/cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json

{
    "securityMode": "NONE",
    "tlcIdentifiers": ["NLZH0023", "NLZH0026"]
}
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8",
    "domain": "test",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "NONE",
        "tlcIdentifiers": ["NLZH0023", "NLZH0026"],
        "listener": {
            "host": "n44.tlex.eu",
            "port": 40344,
            "expiration": "2016-11-17T16:07:56Z"
        },
        "keepAliveTimeout": "PT5S",
        "clockDiffLimit": "PT3S",
        "clockDiffLimitDuration": "PT60S",
        "payloadRateLimit": 1200,
        "payloadRateLimitDuration": "PT5S",
        "payloadThroughputLimit": 120,
        "payloadThroughputLimitDuration": "PT5S"
    }
}
```

#### 3.3.1.5 DELETE /sessions/<token>

Removes (ends, disconnects) the active streaming session with the given "token".

⚠ Not needed for normal operation. Intended for administrative purposes and testing purposes.

##### 3.3.1.5.1 Request

```
DELETE <API base URL>/sessions/<session token> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.1.5.2 Response

```
HTTP/1.1 204 No Content
```

#### 3.3.1.5.3 Example

```
DELETE api/v1/sessions/cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 204 No Content
```

## 3.3.2 Session logs

### 3.3.2.1 GET /sessionlogs

Retrieves all streaming session logs.

Must be filtered by time range.

Intended for monitoring and debug purposes.

#### 3.3.2.1.1 Request

```
GET <API base URL>/sessionlogs?from=<ISO 8601 date time>&until=<ISO 8601 date time> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

#### 3.3.2.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    <session log>, <session log>, ...
]
```

#### 3.3.2.1.3 Example

```
GET api/v1/sessionlogs?from=2017-03-09T20:00:00Z&until=2017-03-09T21:00:00Z HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
 {
    "token": "1AhfqqkcBtOvUdoPrFTHg1x3PMHzbHRLJc848mY016U",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "created": "2017-03-09T20:44:28Z",
    "connected": "2017-03-09T20:44:29Z",
    "remoteAddress": "/172.17.210.254:50036",
    "ended": "2017-03-10T11:23:18Z",
    "endReason": "Average payload rate in the last 5 seconds has exceeded the limit by 1753.600000 payload/s",
    "tlcScopeHistory": [
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0001"
      },
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0002"
      },
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0003"
      },
      {
        "timestamp": "2017-03-10T11:23:12Z",
        "scope": "REMOVED",
        "tlcIdentifier": "tlc_0003"
      }
    ]
  }
]
```

#### 3.3.2.2 GET /sessionlogs/<token>

Retrieves the streaming session's log with the given "token".

Intended for monitoring and debug purposes.

##### 3.3.2.2.1 Request

```
GET <API base URL>/sessionlogs/<session token> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.2.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<session log>
```

#### 3.3.2.2.3 Example

```
GET api/v1/sessionlogs/1AhfqqkcBtOvUdoPrFTHg1x3PMHzbHRLJc848mY016U HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "1AhfqqkcBtOvUdoPrFTHg1x3PMHzbHRLJc848mY016U",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "type": "Broker",
    "protocol": "TCPStreaming_Multiplex",
    "created": "2017-03-09T20:44:28Z",
    "connected": "2017-03-09T20:44:29Z",
    "remoteAddress": "/172.17.210.254:50036",
    "ended": "2017-03-10T11:23:18Z",
    "endReason": "Average payload rate in the last 5 seconds has exceeded the limit by 1753.600000 payload/s",
    "tlcScopeHistory": [
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0001"
      },
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0002"
      },
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0003"
      },
      {
        "timestamp": "2017-03-10T11:23:12Z",
        "scope": "REMOVED",
        "tlcIdentifier": "tlc_0003"
      }
    ]
}
```

## 3.3.3 TLCs

### 3.3.3.1 GET /tlcs

Retreives all TLC registrations.

Intended for supporting dynamic setup of multiple load balancing "TCPStreaming_Multiplex" sessions without having to maintain a static "TLC identifier" list.

#### 3.3.3.1.1 Request

```
GET <API base URL>/tlcs HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

#### 3.3.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    <TLC>, <TLC>, ...
]
{
  "uuid": "<TLC uuid>",
  "identifier": "<TLC identifier>",
  "type": "<TLC type>",
  "domain": "<domain>",
  "account": "<account>"
}
```

| Name | Description |
|---|---|
| uuid | The unique id for the created TLC |
| identifier | The TLC identifier of the TLC |
| type | Type of TLC; must be one of the predefined types:<br>1. TCPStreaming<br>2. VLOG |

| Name | Description |
| --- | --- |
| domain | The domain in which the TLC is registered |
| account | Unique id of the account that "owns" the TLC registration |

#### 3.3.3.1.3 Example

```
GET api/v1/tlcs HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "uuid": "4aa1ace8-32b0-42b6-925a-7d7a33e97859",
    "identifier": "tlc_0001",
    "type": "TCPStreaming",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a"
  },
  {
    "uuid": "d1c9ca3d-23e1-4191-bfa3-b8364c52a4ce",
    "identifier": "tlc_0002",
    "type": "TCPStreaming",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a"
  }
]
```

#### 3.3.3.2 GET /tlcs/<uuid>

Retreives the TLC registration with the given "uuid".

##### 3.3.3.2.1 Request

```
GET <API base URL>/tlcs/<TLC uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.3.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<TLC>
```

#### 3.3.3.2.3 Example

```
GET api/v1/tlcs/4aa1ace8-32b0-42b6-925a-7d7a33e97859 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "4aa1ace8-32b0-42b6-925a-7d7a33e97859",
    "identifier": "tlc_0001",
    "type": "TCPStreaming",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a"
}
```

## 3.3.4 Authorizations

### 3.3.4.1 POST /authorizations

Creates an authorization.

#### 3.3.4.1.1 Request

```
POST <API base URL>/authorizations HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json

{
  "role": "<role>"
}
```

| Name | Description |
| --- | --- |
| role | The role granted to the authorization, must be one of the predefined types:<br>1. BROKER_SYSTEM<br>2. BROKER_ANALYST |

#### 3.3.4.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "uuid": "<authorization uuid>",
  "domain": "<domain name>",
  "account":"<account uuid>",
  "role": "<role>"
}
```

| Name | Description |
|---|---|
| uuid | The unique id of the created authorization |
| domain | The domain for which the authorization is created |
| account | The account for which the authorization is created |
| role | See request |

#### 3.3.4.1.3 Example

```
POST api/v1/authorizations HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json

{
    "role": "BROKER_SYSTEM"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "c6fb449f-0bea-49d3-8d39-9a4689902d99",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_SYSTEM"
}
```

#### 3.3.4.2 GET /authorizations

##### 3.3.4.2.1 Request

```
GET <API base URL>/authorizations HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.4.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    <authorization>, <authorization>, ...
]
```

#### 3.3.4.2.3 Example

```
GET api/v1/authorizations HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "uuid": "98a6890d-589d-43d4-bdff-3165425736d8",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_SYSTEM"
  },
  {
    "uuid": "cd3e0ac6-4718-4f0e-8195-82e0c92e8cb6",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_ANALYST"
  }
]
```

#### 3.3.4.3 GET /authorizations/<uuid>

Retrieves the authorization with the given "uuid".

##### 3.3.4.3.1 Request

```
GET <API base URL>/authorizations/<authorization uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.4.3.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<authorization>
```

#### 3.3.4.3.3 Example

```
GET api/v1/authorizations/c6fb449f-0bea-49d3-8d39-9a4689902d99 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "c6fb449f-0bea-49d3-8d39-9a4689902d99",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_SYSTEM"
}
```

#### 3.3.4.4 PUT /authorizations/<uuid>

Updates the authorization with the given "uuid".

##### 3.3.4.4.1 Request

```
PUT <API base URL>/authorizations/<authorization uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json

<authorization>
```

##### 3.3.4.4.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<authorization>
```

#### 3.3.4.4.3 Example

```
PUT api/v1/authorizations/c6fb449f-0bea-49d3-8d39-9a4689902d99 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json

{
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_SYSTEM"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "c6fb449f-0bea-49d3-8d39-9a4689902d99",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_SYSTEM"
}
```

#### 3.3.4.5 DELETE /authorizations/<uuid>

Removes the authorization with the given "uuid".

##### 3.3.4.5.1 Request

```
DELETE <API base URL>/authorizations/<authorization uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.4.5.2 Response

```
HTTP/1.1 204 No Content
```

#### 3.3.4.5.3 Example

```
DELETE api/v1/authorizations/c6fb449f-0bea-49d3-8d39-9a4689902d99 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 204 No Content
```

## 3.3.5 Authorizationtokens

### 3.3.5.1 POST /authorizationtokens

Creates an authorization token.

#### 3.3.5.1.1 Request

```
POST <API base URL>/authorizationtokens HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json

{
  "authorization": "<authorization uuid>"
}
```

| Name | Description |
|---|---|
| authorization | The unique id of the authorization for which the authorization token will be created |

#### 3.3.5.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "uuid": "<authorization token uuid>",
  "token": "<token>",
  "authorization":"<authorization uuid>"
}
```

| Name | Description |
|---|---|
| uuid | The unique id of the created authorization token |
| token | The token that can be used to perform API calls |
| authorization | The unique id of the authorization to which the token belongs |

#### 3.3.5.1.3 Example

```
POST api/v1/authorizationtokens HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json

{
    "authorization": "c6fb449f-0bea-49d3-8d39-9a4689902d99"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "1040b7e5-6a72-4370-8b70-cbe08cc8fee3",
    "token": "cNjf5zQgV51YWG9Wf1vYF1aWdDBOEhwEzkfCtk8SBkw",
    "authorization": "c6fb449f-0bea-49d3-8d39-9a4689902d99"
}
```

#### 3.3.5.2 GET /authorizationtokens

Retreives all authorization tokens

##### 3.3.5.2.1 Request

```
GET <API base URL>/authorizationtokens HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.5.2.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    <authorization token>, <authorization token>, ...
]
```

#### 3.3.5.2.3 Example

```
GET api/v1/authorizationtokens HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "uuid": "aebd94b2-8eb7-4ba4-8414-d9c6c623cc63",
    "token": "oESyc4mCjhHB7p98_vAuggu-w8c6FtLJia1ewZsK2BE",
    "authorization": "c6fb449f-0bea-49d3-8d39-9a4689902d99"
  },
  {
    "uuid": "7ced02c2-9384-4d17-9032-9dbaa3f16805",
    "token": "_lZteZcPTSkaHqtgrqPqp7yFlo3SMxlFO_eJT5-c6cY",
    "authorization": "98a6890d-589d-43d4-bdff-3165425736d8"
  },
  {
    "uuid": "1040b7e5-6a72-4370-8b70-cbe08cc8fee3",
    "token": "cNjf5zQgV51YWG9Wf1vYF1aWdDBOEhwEzkfCtk8SBkw",
    "authorization": "c6fb449f-0bea-49d3-8d39-9a4689902d99"
  }
]
```

#### 3.3.5.3 GET /authorizationtokens/<uuid>

Retrieves the authorization token with the given "uuid".

##### 3.3.5.3.1 Request

```
GET <API base URL>/authorizationtokens/<authorization token uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.5.3.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<authorization token>
```

#### 3.3.5.3.3 Example

```
GET api/v1/authorizationtokens/1040b7e5-6a72-4370-8b70-cbe08cc8fee3 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "1040b7e5-6a72-4370-8b70-cbe08cc8fee3",
    "token": "cNjf5zQgV51YWG9Wf1vYF1aWdDBOEhwEzkfCtk8SBkw",
    "authorization": "c6fb449f-0bea-49d3-8d39-9a4689902d99"
}
```

#### 3.3.5.4 PUT /authorizationtokens/<uuid>

Updates the authorization with the given "uuid".

##### 3.3.5.4.1 Request

```
PUT <API base URL>/authorizationtokens/<authorization token uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json

<authorization token>
```

##### 3.3.5.4.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/json

<authorization token>
```

#### 3.3.5.4.3 Example

```
PUT api/v1/authorizationtokens/1040b7e5-6a72-4370-8b70-cbe08cc8fee3 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json

{
    "authorization": "5cb0a102-cff6-4ee1-a4ae-d8300f32e785"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uuid": "1040b7e5-6a72-4370-8b70-cbe08cc8fee3",
    "token": "cNjf5zQgV51YWG9Wf1vYF1aWdDBOEhwEzkfCtk8SBkw",
    "authorization": "5cb0a102-cff6-4ee1-a4ae-d8300f32e785"
}
```

#### 3.3.5.5 DELETE /authorizationtokens/<uuid>

Removes the authorization token with the given "uuid".

##### 3.3.5.5.1 Request

```
DELETE <API base URL>/authorizationtokens/<authorization token uuid> HTTP/1.1
Host: <hostname>
X-Authorization: <authorization token>
Content-Type: application/json
```

##### 3.3.5.5.2 Response

```
HTTP/1.1 204 No Content
```

#### 3.3.5.5.3 Example

```
DELETE api/v1/authorizationtokens/1040b7e5-6a72-4370-8b70-cbe08cc8fee3 HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

```
HTTP/1.1 204 No Content
```