TLEX-TLC-ADMIN Interface v1.2.1

Table of contents

1 Versioning

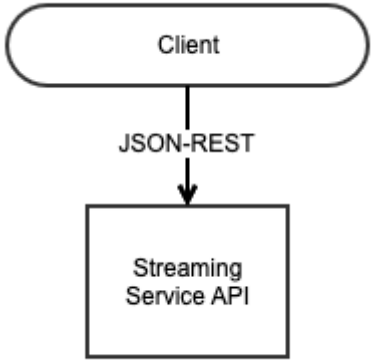
This document is using a versioning scheme that indicates the version of this TLEX interface and tracks the revisions of this document. This version scheme is <interface version major>.<interface version minor>.<document revision>. The first two version numbers (major and minor) indicate the version 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 Added API end-point "sessions" POST use case: Session type "TLC" with protocol "VLOG" Added TLC type "TCPStreaming" and "VLOG" to API end-point "tlcs" Added PUT / update operation for API end-point "tlcs"
1.2.0	14 Mar 2018	L. Rijneveld	Updated for TLEX release 1.3 Updated /tlcs end point (added VLOG specific detail fields)
1.2.1	23 Mar 2020	L. Rijneveld	Improved layout and formatting

2 Overview

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

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



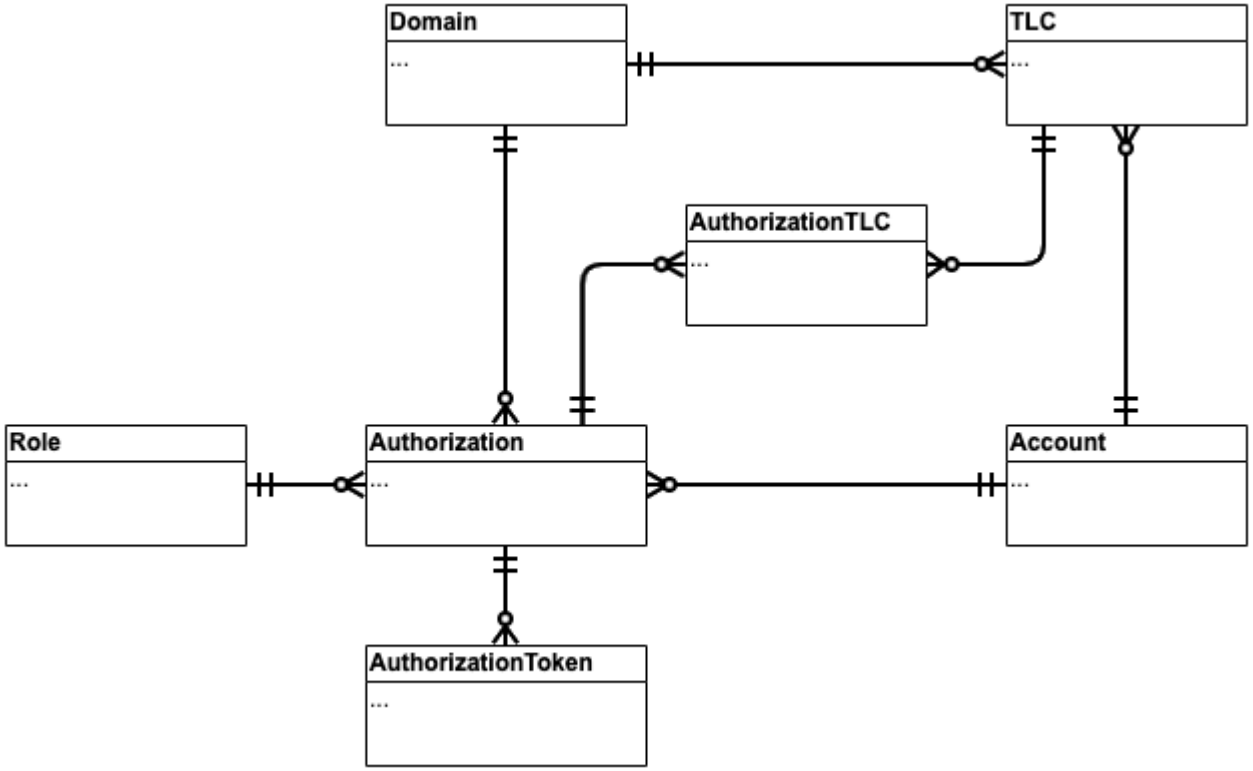
3 API

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 "TLC_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: 1. TLC registrations 2. authorizations
Authorization	The authorization to use the API, which is a combination of: 1. Account 2. Domain 3. Role 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.
AuthorizationTLC	The authorization's TLC scope. Only applicable for "TLC system" and "TLC analyst" role authorizations, see roles.
AuthorizationToken	The authorization token. One authorization can have multiple authorization tokens.

3.2.2 Roles

The following roles have been defined for TLC related authorizations:

Name	Access	Access scope	Intended for
TLC administrator (TLC_ADMIN)	Can manage own TLC registrations within the domain scope Can manage own "TLC system" authorizations within the domain scope Can manage own "TLC system" authorization tokens within the domain scope Can manage all own TLC sessions within the domain scope Can view all own TLC session logs and metrics within the domain scope	Domain, Account	Organisations responsable for the production TLCs
TLC system (TLC_SYSTEM)	Can create TLC sessions within the domain and authorization TLC scope	Domain, Account, Set of TLCs	TLC systems
TLC analyst (TLC_ANALYST)	Can view TLC session logs and metrics within the domain and authorization TLC scope Can view TLC registrations within the domain and authorization TLC scope	Domain, Account, Set of TLCs	Analysts/engineers responsible for one or more production TLCs

3.3 API endpoints

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

API endpoint			Authorization role access scope (account/authorization/none)		
Method	**URI**	**Description**	**TLC_ADMIN**	**TLC_SYSTEM**	**TLC_ANALYST**
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
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 Only sessions created for the same domain will be able to stream data to each other
type	The session type; must be “ TLC”
protocol	The session protocol; must be one of the predefined types: 1. TCPStreaming_Singleplex 2. TCPStreaming_Multiplex 3. VLOG
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 ⚠ 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 "TLC" with protocol "TCPStreaming_Singleplex"

TCP based singleplex streaming session for one specific TLC.

Payloads sent by the Client will be received by Broker" session Clients if the "TLC identifier" of this specific TLC is within their scope.

Payloads sent by "Broker" session clients having a payload "TLC identifier" that matches this specific TLC's identifier will be received.

3.3.1.1.3.1 Request details

```
{
    "securityMode": "<security mode>",
    "tlcIdentifier": "<TLC identifier>"
}
```

Name	Description
securityMode	Security mode of the streaming session Must be one of the predefined values: 1. NONE 2. TLSv1.2
tlcIdentifier	The TLC identifier for the session Since the session is for one specific TLC, payload data will be streamed without TLC identifier (see protocol datagram 0x04)

3.3.1.1.3.2 Response details

```
{
    "securityMode": "<security mode>",
    "tlcIdentifier": "<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
tlcIdentifier	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 If the TCP connection has not been established before this time the listener will expire and the streaming session will no longer be available 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 If no TCP data is received during the specified duration, the TCP connection will be terminated by the Streaming Service 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 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 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 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" : "TLC",
    "protocol" : "TCPStreaming_Singleplex",
    "details": {
        "securityMode": "TLSv1.2",
        "tlcIdentifier": "NLZH0023"
    }
}
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "cXXrqTkrehOvLbuuYKKQQGAU1MTGGGBC1N1izwYaqu8",
    "domain": "test",
    "type": "TLC",
    "protocol": "TCPStreaming_Singleplex",
    "details": {
        "securityMode": "TLSv1.2",
        "tlcIdentifier": "NLZH0023",
        "listener": {
            "host": "n24.tlex.eu",
            "port": 58142,
            "expiration": "2016-11-17T16:01:45Z"
        },
        "keepAliveTimeout": "PT5S",
        "clockDiffLimit": "PT3S",
        "clockDiffLimitDuration": "PT60S",
        "payloadRateLimit": 15,
        "payloadRateLimitDuration": "PT5S",
        "payloadThroughputLimit": 15,
        "payloadThroughputLimitDuration": "PT5S"
    }
}
```

3.3.1.1.4 Session type "TLC" with protocol "TCPStreaming_Multiplex"

TCP based multiplex streaming session for one or more TLC's.

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

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

3.3.1.1.4.1 Request details

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

⚠ The only difference in the request details is the tlcIdentifiers element which replace the tlcIdentifier element used in the "TCPStreaming_TLC_Singleplex" session type request details.

Name	Description
tlcIdentifiers	The TLC identifiers for the session Since the session is for multiple TLC's, payload data will be streamed with TLC identifier (see protocol datagram 0x05)

3.3.1.1.4.2 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>"
}
```

⚠ The only difference in the response details is the tlcIdentifiers element which replace the tlcIdentifier element used in the "TCPStreaming_TLC_Singleplex" session type response details.

3.3.1.1.4.3 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": "TLC",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "TLSv1.2",
        "tlcIdentifiers": ["NLZH0023", "NLZH0024", "NLZH0025"]
    }

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

{
    "token": "dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k",
    "domain": "test",
    "type": "TLC",
    "protocol": "TCPStreaming_Multiplex",
    "details": {
        "securityMode": "TLSv1.2",
        "tlcIdentifiers": ["NLZH0023", "NLZH0024", "NLZH0025"],
        "listener": {
            "host": "n25.tlex.eu",
            "port": 51253,
            "expiration": "2016-11-17T16:04:23Z"
        },
        "keepAliveTimeout": "PT5S",
        "clockDiffLimit": "PT3S",
        "clockDiffLimitDuration": "PT60S",
        "payloadRateLimit": 45,
        "payloadRateLimitDuration": "PT5S",
        "payloadThroughputLimit": 45,
        "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": "TLC",
    "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": "TLC",
    "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": "TLC",
    "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": "TLC",
    "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": "TLC",
    "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 POST /tlcs

Creates a TLC registration.

3.3.3.1.1 Request

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

{
  "identifier": "<TLC identifier>",
  "type": "<TLC type>",
  "details": {
    "<Type specific details, only applicable for TLC type 'VLOG'>"
  }
}
```

Name	Description
identifier	The TLC identifier of the TLC Must be exactly 8 characters Must be unique within the domain Is case insensitive
type	Optional (will default to 'TCPStreaming' when not specified) Type of TLC; must be one of the predefined types: 1. TCPStreaming 2. VLOG
details	See TLC type specific details

3.3.3.1.2 Response

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

{
  "uuid": "<TLC uuid>",
  "identifier": "<TLC identifier>",
  "type": "<TLC type>",
  "details": {
    "<Type specific details, only applicable for TLC type 'VLOG'>"
  },
  "domain": "<domain>",
  "account": "<account>"
}
```

Name	Description
uuid	The unique id for the created TLC
identifier	See request
type	See request
details	See request
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

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

{
  "identifier": "tlc_abcd",
  "type": "TCPStreaming"
}
```

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

{
  "uuid": "a54deb2b-419b-4e55-9efc-69797b469669",
  "identifier": "tlc_abcd",
  "type": "TCPStreaming",
  "domain": "test",
  "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a"
}
```

3.3.3.1.4 Request details TLC type VLOG

```
{
    "host": "<host address>",
    "port": <port number>,
    "itf": "<Base64 encoded XML document>",
    "authorization": "<authorization uuid>",
    "autoConnect": <true or false>,
    "itfSynchronization": <true or false>,
    "vlogVersion": "<version string>",
    "itfVersion": "<version string>"
}
```

Name	Description
host	The host address on which the VLOG ASCII device will accept the TCP connection
port	The TCP port on which the VLOG ASCII device will accept the TCP connection
itf	A base64 encoded ITF XML document. The ITF must meet the following criteria: • Valid XML, based on ITF XSD; • Must contain at least one SignalGroup definition in the ControlData section; • MapData must be MAP compliant. Optional if autoConnect is set to 'false' or when itfSynchronization is set to 'true'
authorization	Authorization UUID which will be used by the AutoConnect Session Manager to obtain an authorizationToken for creating sessions for this TLC Must be an existing authorization that has sufficient access rights for creating sessions for this TLC The authorization must have at least one authorizationToken Can be acquired by creating an authorization using the /authorizations end-point Optional if autoConnect is set to 'false'
autoConnect	When set to 'true' the AutoConnection Session Manager will automatically create a session for this TLC when no session exists
itfSynchronization	When set to 'true' the ITF Synchronisation Service will automatically retrieve the latest ITF document from the pre-defined Topology Exchange (Topolex)

Name	Description
vlogVersion	Optional user definable string to indicate the vlog version used This field has no functional impact
itfVersion	Optional user definable string to indicate the version of the ITF document, as stored in the "itf" field ⚠ When itfSynchronization is set to 'true' this field will contain the version of the current ITF document as versioned by the Topology Exchange (Topolex)

3.3.3.1.5 Example TLC type VLOG

⚠ For the sake of readability, the "itf" field's value has been clipped.

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

{
  "identifier": "tlc_abcd",
  "type": "VLOG"
  "details" {
        "host": "vri.foo.bar",
        "port": 7070,
        "itf": "77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlb ... YT4NCjwvdG9wb2xvZ3k+",
        "authorization": "60b142ab-88e8-4600-9a86-8807c19b1b2c",
        "autoConnect": true,
        "itfSynchronization": false,
        "vlogVersion": "3.0.0",
        "itfVersion": "4.9"
  }
}
```

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

{
  "uuid": "a54deb2b-419b-4e55-9efc-69797b469669",
  "identifier": "tlc_abcd",
  "type": "VLOG",
  "details" {
        "host": "vri.foo.bar",
        "port": 7070,
        "itf": "77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlb ... YT4NCjwvdG9wb2xvZ3k+",
        "authorization": "60b142ab-88e8-4600-9a86-8807c19b1b2c",
        "autoConnect": true,
        "itfSynchronization": false,
        "vlogVersion": "3.0.0",
        "itfVersion": "4.9"
  },
  "domain": "test",
  "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a"
}
```

3.3.3.2 GET /tlcs

Retreives all TLC registrations

3.3.3.2.1 Request

```
GET <API base URL>/tlcs 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>, <TLC>, ...
]
```

3.3.3.2.3 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"
  }
]
```

3.3.3.3 GET /tlcs/<uuid>

Retreives the TLC registration with the given "uuid".

3.3.3.3.1 Request

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

3.3.3.3.2 Response

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

<TLC>
```

3.3.3.3.3 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"
}
```

3.3.3.4 PUT /tlcs/<uuid>

Updates a TLC registration.

Can only be used to update the TLC's type and, if applicable, details.

3.3.3.4.1 Request

See POST /tlcs regarding the request since the body is exactly the same.

3.3.3.4.2 Response

See POST /tlcs regarding the request since the body is exactly the same.

3.3.3.5 DELETE /tlcs/<uuid>

3.3.3.5.1 Request

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

3.3.3.5.2 Response

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

3.3.3.5.3 Example

```
DELETE 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/1.1 204 No Content
```

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>",
  "tlcIdentifiers": [<TLC identifier>, <TLC identifier>]
}
```

Name	Description
role	The role granted to the authorization, must be one of the predefined types: 1. TLC_SYSTEM 2. TLC_ANALYST
tlcIdentifiers	Optional list of TLC identifiers for which the authorization is valid ⚠ When not specified the authorization is valid for all TLCs owned by the authorization's accoun

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>",
  "tlcIdentifiers": [<TLC identifier>, <TLC identifier>]
}
```

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
tlcIdentifiers	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": "TLC_SYSTEM",
    "tlcIdentifiers": ["tlc00001"]
}
```

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

{
    "uuid": "c6fb449f-0bea-49d3-8d39-9a4689902d99",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "TLC_SYSTEM",
    "tlcIdentifiers": ["tlc00001"]
}
```

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": "TLC_SYSTEM",
    "tlcIdentifiers": ["tlc00001"]
  },
  {
    "uuid": "cd3e0ac6-4718-4f0e-8195-82e0c92e8cb6",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "TLC_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": "TLC_SYSTEM",
    "tlcIdentifiers": ["tlc00001"]
}
```

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": "TLC_SYSTEM",
    "tlcIdentifiers": ["tlc_0001", "tlc_0002"]
}
```

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

{
    "uuid": "c6fb449f-0bea-49d3-8d39-9a4689902d99",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "TLC_SYSTEM",
    "tlcIdentifiers": ["tlc_0001", "tlc_0002"]
}
```

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
```