iVRI Overnamepunt (TLEX) specificatie v1.1

Table of Contents

1 Wijzigingshistory

Versie	Datum	Status	Wijziging	Door
1.0	14-12-2016	Release	Versie nummer en status aangepast vanwege initiële release	L. Rijneveld
1.1	13-03-2017	Revision 1	Wijzigingshistory aangepast (concept wijzigingen verwijderd) Aanpassingen in technische specificatie: • Sessie beëindiging bij ontbrekende tijd synchronisatie antwoorden toegevoegd • Back-pressure thresholds CAM en SPAT verhoogd van 200ms naar 1000ms • Term "Api-key" aangepast naar "authorization token" • API autorisatie aangepast/aangescherpt t.b.v. "DOMAIN_ADMIN" rol • API autorisatie aangepast/verruimt t.b.v. "TLC_SYSTEM" en "BROKER_SYSTEM" rollen • API autorisatie aangepast/verruimt t.b.v. "TLC_ANALYST" en "BROKER_ANALYST" rollen • API end-point beschrijvingen aangevuld • API end-point "sessionmetrics" verwijderd • API end-point "tlcs" PUT methode verwijderd • API end-point "sessions" gewijzigd: "session type" opgesplitst in "session type" en "session protocol" • "Session events" aangepast: "expired" en "disconnect" verwijderd en "end" toegevoegd • Dubbelzinnige term "datagram" aangepast naar "frame" bij uitleg "framing"	L. Rijneveld

2 Referentiedocumenten

Nr.	Beschrijving	Versienr./Datum	Auteur
[1]	RFP Talking Traffic 1.1 Beter Benutten 2016.07.01	1.1 / 2016.07.01	Min. I&E
[2]	EN_30263702v010302 (CAM berichten)	010302	ETSI
[3]	EN_30263703v010202 (DENM berichten)	010202	ETSI
[4]	J2735_201603 (SPAT en MAP berichten)	201603	SAE
[5]	V-log protocol en definities v3 0 0 dd 20161013	V3.0.0 / 20161013	Vialis
[6]	Deliverable F – iTLC Architecture v1.2	V1.2 / 27-01-2016	BBV

3 Algemeen

Dit document beschrijft de communicatie van en naar het iVRI overnamepunt zoals beschreven in document [1].

Het iVRI overnamepunt is nodig om de real time data van en naar elke afzonderlijke iVRI in cluster1 en cluster2 goed te laten functioneren, door deze data ‘op te vangen’ en te routeren tussen beide clusters.

Van cluster1 naar cluster2 gaat het om de SPAT, MAP, DENM en SSM data van elke afzonderlijke iVRI door te geven.

Van cluster2 naar cluster1 gaat het om de CAM en SRM data (prioriteitsaanvragen, herkomst/bestemmingsdata, snelheidsinformatie op wegvakniveau) van weggebruikers aan de door cluster2 aangemerkte iVRI door te gegeven ten behoeve van de verkeersregeling.

Hoofdstuk 4 beschrijft de totstandkoming van een verbinding tussen het iVRI Overnamepunt en een iVRI.

Hoofdstuk 5 beschrijft de totstandkoming van een verbinding tussen het iVRI Overnamepunt en een cluster2-systeem.

Hoofdstuk 6 beschrijft hoe de data worden overgedragen tussen een iVRI en een cluster2-systeem via het iVRI Overnamepunt.

Hoofdstuk 7 beschrijft de faciliteiten die beschikbaar zijn voor het ontwikkelen en testen van cluster1-systemen en cluster2-systemen.

Hoofdstuk 8 beschrijft authenticatie en autorisatie aangaande de toegang tot het iVRI Overnamepunt.

Hoofdstuk 9 bevat de technische specificatie van het iVRI Overnamepunt. Vanwege de technische aard is deze bijlage in het engels geschreven.

4 Verbinding met iVRI

4.1 Initiatief

Het initiatief tot het opbouwen van de verbinding ligt bij de afzonderlijke iVRI.

Het initiatief tot het afbreken van de verbinding kan zowel bij de afzonderlijke iVRI, als bij het iVRI Overnamepunt liggen. Vanuit de iVRI of het iVRI Overnamepunt wordt de verbinding in ieder geval verbroken indien er gedurende 5 seconden geen berichten meer worden ontvangen.

Het initiatief voor het heropbouwen van de verbinding ligt bij de afzonderlijke iVRI.

4.2 Verbinden

Toelichting aangaande de totstandkoming van de verbinding is terug te vinden in hoofdstuk 9.

4.3 Actieve verbinding

Zodra de verbinding tot stand is gekomen, wordt vanuit de iVRI SPAT, MAP, DENM en SSM berichten naar het iVRI Overnamepunt gestuurd.

- SPAT berichten worden verstuurd, zodra er een verandering is in de inhoud. Er worden maximaal 10 SPAT berichten per seconde verstuurd.
- MAP berichten worden verstuurd zodra de verbinding tot stand komt en zodra er een verandering is in de inhoud. Alhoewel de ETSI standaard partieele MAP berichten ondersteund moeten MAP berichten naar iVRI overnamepunt altijd compleet zijn. Er wordt maximaal 1 bericht per uur verstuurd.
- DENM berichten worden event gebaseerd verstuurd. Er wordt maximaal 1 DENM bericht per minuut verstuurd.
- SSM berichten worden verstuurd als reactie op een ontvangen SRM bericht.

Andersom worden - vanuit cluster2 ontvangen – CAM en SRM berichten, door het iVRI Overnamepunt naar de betreffende iVRI gestuurd.

- CAM berichten worden verstuurd zodra er een CAM bericht wordt ontvangen van één van de cluster2-systemen en deze geadresseerd is aan de betreffende iVRI.
- SRM berichten worden verstuurd zodra er een SRM bericht wordt ontvangen van één van de cluster2-systemen en deze geadresseerd is aan de betreffende iVRI.

4.4 Security

De verbinding tussen iVRI en iVRI Overnamepunt is een beveiligde verbinding. In analogie met de keuzes in de iVRI architectuur [6] is er gekozen om ook hier voor TLS beveiliging te kiezen. Er zal geen gebruik gemaakt worden van TLS client authenticatie. Er wordt geen specifieke PKI ingericht.

Indien de beveiliging van de verbinding op een lager niveau kan worden gegarandeerd is tevens mogelijk om te verbinden zonder TLS.

5 Verbinding met Cluster2-systeem

5.1 Initiatief

Het initiatief tot het opbouwen van de verbinding ligt bij het cluster2-systeem.

Het initiatief tot het afbreken van de verbinding kan zowel bij het cluster2-systeem, als bij het iVRI Overnamepunt liggen. Vanuit het cluster2-systeem of het iVRI Overnamepunt wordt de verbinding in ieder geval verbroken indien er gedurende 5 seconden geen berichten meer worden ontvangen.

Het initiatief voor het heropbouwen van de verbinding ligt bij het cluster2-systeem.

5.2 Verbinden

Het cluster2-systeem zal bij het verbinden aangeven wat de “VRI scope” van de verbinding is. Op deze manier is het mogelijk om met behulp van meerdere complementerende verbindingen, “load balancing” toe te passen.

Toelichting aangaande de totstandkoming van de verbinding is terug te vinden in hoofdstuk 9.

5.3 Actieve verbinding

Zodra de verbinding tot stand is gekomen, wordt vanuit het iVRI Overnamepunt SPAT, MAP, DENM en SSM berichten naar het cluster2-systeem gestuurd.

- SPAT berichten worden verstuurd zodra er een SPAT bericht wordt ontvangen van één van de VRI’s.
- MAP berichten worden verstuurd zodra er een MAP bericht wordt ontvangen van één van de VRI’s. Daarnaast wordt zodra de verbinding tot stand is gekomen het laatst door het iVRI Overnamepunt ontvangen MAP bericht per iVRI verstuurd.
- DENM berichten worden verstuurd zodra er een DENM bericht wordt ontvangen van één van de iVRI’s.
- SSM berichten worden verstuurd zodra er een SSM bericht wordt ontvangen van één van de iVRI’s.

De berichten worden in het iVRI Overnamepunt voorzien van een uniek iVRI identificatie. Deze iVRI identificatie wordt meegestuurd, zodat een cluster2-systeem weet van welke iVRI de data afkomstig is.

Andersom worden - vanuit cluster2 ontvangen – CAM berichten, door het iVRI Overnamepunt naar de betreffende iVRI gestuurd, op basis van de iVRI identificatie, die door het cluster2-systeem wordt meegestuurd.

- CAM berichten worden ontvangen uit cluster2 per voertuig, niet vaker dan 1 x per seconde met een maximum van 400 berichten per seconde per iVRI. Daarnaast zorgt cluster2 voor het selecteren van de juiste iVRI, die het betreffende bericht moet ontvangen. Deze selectie vindt plaats op basis van “geo-fencing” waarbij een CAM bericht alleen wordt verstuurd indien deze een voertuig beschrijft welke zich binnen de, nader te bepalen, radius van een aangesloten iVRI bevindt. De iVRI is verantwoordelijk voor het eventueel de-dupliceren van CAM berichten indien er meerdere CAM berichten ontstaan voor hetzelfde voertuig.
- SRM berichten worden ontvangen uit cluster2 waar hierbij ook cluster2 zorgt voor het selecteren van de juiste iVRI, die het betreffende bericht moet ontvangen.

Het aantal maximale CAM berichten per iVRI van 400 voertuigen is gebaseerd op een technisch theoretisch maximum en zal na gezamenlijke evaluatie wellicht moeten worden bijgesteld naar een realistisch maximum.

5.4 Security

De verbinding tussen een cluster2-systeem en het iVRI Overnamepunt is een beveiligde verbinding. In analogie met de keuzes in de iVRI architectuur [6] is er gekozen om ook hier voor TLS beveiliging te kiezen. Er zal geen gebruik gemaakt worden van TLS client authenticatie. Er wordt geen specifieke PKI ingericht.

Indien de beveiliging van de verbinding op een lager niveau kan worden gegarandeerd is tevens mogelijk om te verbinden zonder TLS.

6 Data overdracht

6.1 Uitgangspunten

Het uitgangspunt voor de dataoverdracht tussen een iVRI en een cluster2-systeem via het iVRI Overnamepunt is om zo dicht mogelijk bij de ETSI berichten te blijven. Het iVRI Overnamepunt dient hierbij zoveel mogelijk agnostisch te blijven van het daadwerkelijke formaat van de data. De dataoverdracht is van het type “streaming”. Er is geen sprake van request/reply. Berichten worden zodra er verbinding is, gestreamed naar de beide kanten.

Verder technische toelichting aangaande de dataoverdracht via het iVRI Overnamepunt is terug te vinden in hoofdstuk 9.

6.2 Berichten van cluster1 naar cluster2

De SPAT, MAP, SSM en DENM berichten die via het iVRI Overnamepunt van de iVRI naar de cluster2-systemen worden gestreamed, zijn beschreven middels een ASN.1 notatie, die terug te vinden zijn in ETSI documenten.

Voor SPAT berichten is dat terug te vinden in document [4] par. 5.13 en onderliggend.

Voor MAP berichten is dat beschreven in document [4] par. 5.6 en onderliggend.

Voor DENM berichten is dat beschreven in document [3] annex A.

Voor SSM berichten is dat beschreven in document [4] par. 5.15 en onderliggend.

De berichten zullen middels de ASN.1 UPER codering binair worden verzonden.

6.3 Berichten van cluster2 naar cluster1

De CAM en SRM berichten die via het iVRI Overnamepunt van de cluster2-systemen naar de iVRI worden gestuurd worden eveneens gestreamed, en zijn beschreven middels een ASN.1 notatie, die terug te vinden zijn in de ETSI documenten.

Voor CAM berichten is dat terug te vinden in document [2] annex A.

Voor SRM berichten is dat terug te vinden in document [4] par 5.14 en onderliggend.

De berichten zullen middels de ASN.1 UPER codering binair worden verzonden.

7 Testen

Ter ondersteuning van de ontwikkeling en het testen van cluster1-systemen en cluster2-systemen krijgt elke partij een eigen test domein toegewezen binnen het iVRI Overnamepunt.

Toegang tot het iVRI Overnamepunt is altijd in de context van een specifiek domein. Partijen hebben in de context van hun eigen test domein toegang tot zowel de cluster1 als cluster2 functionaliteiten en zijn daarmee, onder andere, in staat beide soorten verbindingen op te bouwen.

Cluster1 partijen kunnen op deze manier een cluster2-systeem “stub” ontwikkelen om, eventueel geautomatiseerde, tests uit te voeren rondom de implementatie van hun cluster1-systeem.

Cluster2 partijen kunnen op deze manier een cluster1-systeem “stub” ontwikkelen om, eventueel geautomatiseerde, tests uit te voeren rondom de implementatie van hun cluster2-systeem.

Verdere details aangaande de domein scheiding en de manier waarop partijen in staat zullen zijn om zelf hun test domein te beheren via de API van het iVRI Overnamepunt zijn terug te vinden in hoofdstuk 9.

8 Authenticatie en autorisatie

Partijen en systemen krijgen toegang tot het iVRI Overnamepunt d.m.v. een “API key”. Het autorisatiemodel stelt partijen in staat om zelf “API keys” te beheren (aanmaken en verwijderen) voor de systemen die onder hun verantwoordelijkheid vallen.

Voor de eerder beschreven test-domeinen heeft een desbetreffende partij hiermee de vrijheid om zelf de “API keys” te genereren die nodig zijn voor test werkzaamheden.

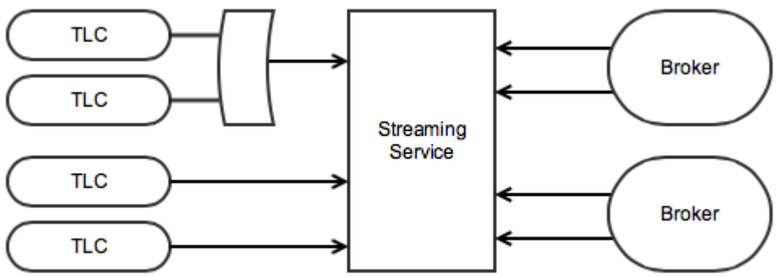
Voor het produktie-domein zullen de partijen die operationeel verantwoordelijk zijn voor de VRI’s (wegbeheerders) in staat zijn om zelf de VRI’s te registreren en voor deze VRI’s de “API keys” te beheren.

Meer details omtrent het autorisatiemodel zijn terug te vinden in hoofdstuk 9.

9 Technische specificatie

9.1 Overview

The Streaming Service is designed to route TLC related data between multiple TLC's and multiple TLC data brokers. Data from a TLC is always sent to all connected brokers. Data from a broker is always sent to one specific TLC. The TLCs and the Brokers initiate the connection with the Streaming Service. When, for whatever reason, the session is terminated, the TLCs and/or Brokers have to reinitiate the connection.



The Streaming Service is build up from two components:

1. the REST API, primarily for creating streaming sessions;
2. the Streaming Service (nodes) for handling continuous streaming of payloads over TCP. One streaming session can stream several types of payload.

In order to establish a connection for a streaming session a Client has to create the session using the Streaming Service API. The Streaming Service API will allocate a free Streaming Service Node for the streaming session and return the created session details, including the connection details, in the request's response.

In order to support testing and debugging the streaming sessions can be created for specific domain. The streaming sessions created in different domains are isolated from each other, making it possible for multiple parties to test and debug the streaming functionality simultaneously without the risk of interfering with each other.

Besides creating new streaming sessions the Streaming Service API can also used to request the session details of a previously created session and to request all created sessions.

9.2 Payload types

A streaming session can stream more then one type of payload. Although the Streaming Service itself does not interpret the payload while streaming between sessions, the different types of payload are pre-defined in order to:

1. inform the receiving end-point of the payload type so that the payload data can be interpreted correctly;
2. register streaming metrics per payload type;
3. calculate the "Session stale payloads" metric based on payload type specific TTL definitions;
4. enforce payload type specific back pressure policies.

The described TTL (Time-To-Live) of the payloads is mostly relevant for the parties receiving the payload. The Streaming Service uses this TTL only for determining the session metric "stale payload received".

Payload type	**Stream direction**	**Payload format**	**TTL**	**TTL rationale**
MAP	TLC -> Brokers	ASN.1 UPER encoded MAP	300 seconds	MAP is a semi-static message, it is important that it is delivered.
SPaT	TLC -> Brokers	ASN.1 UPER encoded SPaT	3 seconds	SPaT carries the current state of the traffic light, therefore it must be delivered on time.
DENM	TLC -> Brokers	ASN.1 UPER encoded DENM	60 seconds	A DENM has a timestamp and validity, it is acceptable if it is delivered later, and ignored if not valid anymore.
SSM	TLC -> Brokers	ASN.1 UPER encoded SSM	10 seconds	An SSM that is delivered too late will not be informative for the user.
CAM	Brokers -> TLC	ASN.1 UPER encoded CAM	5 seconds	A bit older positions can still be used, the CAM itself contains a timestamp which rolls over after 64 seconds.
Secure CAM	Brokers -> TLC	ASN.1 UPER encoded CAM (with security envelop)		
SRM	Brokers -> TLC	ASN.1 UPER encoded SRM	10 seconds	An SRM that is delivered too late will not be able to provide priority in time.
Secure SRM	Brokers -> TLC	ASN.1 UPER encoded SRM (with security envelop)		

9.2.1 Payload type related standards

Subject	Standard	Version	Comment
CAM	EN 302 637-2	v1.3.2	Cooperative Awareness Message
DENM	EN 302 637-3	v1.2.2	Decentralized Environmental Notification
CDD	TS 102 894-2	v1.2.1	Common Data Dictionary for all messages
Geonetworking	EN 302 636-4-1	v1.2.1	For 802.11p networks
BTP	TS 103 248	v1.0.1	For 802.11p networks
Security	TS 103 097	v1.2.5	Security envelope and certificate format
MAP, SPaT, SRM, SSM	TS 103 301	V1.1.1	ETSI header including protocol version for TS 19091
MAP, SPaT, SRM, SSM	TS 19091	v0910	Based on SAE J2735

9.3 Time synchronisation

It is important that all connected systems have their times properly synced to a reliable clock source. A big clock difference between the systems would be problematic for determining the age of the payload. The Streaming Service will continuously measure the clock difference during a streaming session by sending a protocol specific time enquiry message every 15 seconds and will terminate sessions if the average clock difference (average over 1 minute period) exceeds the "clock difference threshold". This threshold will be set to 3 seconds. The Streaming Service requires at least 2 time synchronisation responses each minute; if less then 2 responses are received the session will also be disconnected.

9.4 Load

9.4.1 Rate and size indication

The described payload sizes are all based on ASN.1 Unaligned PER encoding.

The "broker" rate limits and throughput calculations are based on a maximum of 1300 connected TLCs.

⚠ Streaming protocol overhead has not been included

Theoretical maximums

⚠ The rate metrics used are based on theoretical maximums and will need to be evaluated against the actual rates expected in real life.

Name	Maximum rate	Average payload size	Maximum payload size	Size explanation	Maximum TLC rate	Maximum average TLC throughput	Maximum broker rate	Maximum average broker throughput
MAP	1/hour/TLC	4 KB	20 KB	Average based on 10 signal groups Maximum based on 50 signal groups	1/hour	4 KB/hour	1300/hour	5.08 MB/hour
SPaT	10/second /TLC	1 KB	5 KB	Average based on 10 signal groups Maximum based on 50 signal groups	10/second	10 KB/second	13000/second	12.7 MB/second
DENM	1/minute/TLC	200 B	300 B	Maximum based on 50% increase	1/minute	200 B/minute	1300/minute	253.91 KB/minute
SSM	60/hour/TLC	100 B	150 B	Based on CAM size	60/hour	5.86 KB/hour	78000/hour	7.44 MB/hour
CAM	1/second /vehicle 400/second /TLC	100 B	150 B	Maximum based on 50% increase	400/second	39.06 KB/second	520000/second	49.59 MB/second
SRM	60/hour/TLC	100 B	150 B	Based on CAM size	60/hour	5.86 KB/hour	78000/hour	7.44 MB/hour

Expected

Name	Average TLC rate	Average payload size	Average TLC throughput	Average broker rate	Average broker throughput
MAP	1/day/TLC	4 KB	4 KB/day	1300/day	5.08 MB/day
SPaT	1/second/TLC	1 KB	1 KB/second	1300/second	1.27 MB/second
DENM	1/hour/TLC	200 B	200 B/hour	1300/hour	253.91 KB/hour
SSM	1/minute/TLC	100 B	100 B/minute	1300/minute	126,95 KB/minute
CAM	100/second/TLC	100 B	9.77 KB/second	130000/second	12.40 MB/second
SRM	1/minute/TLC	100 B	100 B/minute	1300/minute	126,95 KB/minute

9.4.2 Session rate and throughput limits

In order to safe guard that sessions do not generate more load than anticipated the streaming sessions are rate and throughput limited regarding the that traffic is allowed to be transmitted. If one of the limits is exceeded the session will be terminated. The rate and throughput limits depend on the session mode (TLC or Broker) and TLC count (in case of a multiplex session).

Mode	Description	Limit/TLC	Rationale
TLC	Maximum payloads per second	12 payload/s	Mainly SPaT based, including some overhead for MAP, DENM, SSM and general headroom Theoretical maximum is applied
	Maximum throughput per second	60 KB/s	
Broker	Maximum payloads per second	120 payload/s	Mainly CAM based including some overhead for CAM, SRM and general headroom Expected average is applied
	Maximum throughput per second	12 KB/s	

When a session is created using the Streaming Service API the rate limits which apply for the created session will also be available in the response.

9.4.3 Back pressure policies

If payloads are not processed quickly enough by the receiving end, queues and buffers start to fill up within the Streaming Service. This phenomenom, commonly called "back pressure", occurs when congestion occurs somewhere in the processing chain. One of the indicators of "back pressure" is an increased (increasing) streaming latency (time between reception and transmission of a payload within the Streaming Service). In case of back pressure, the Streaming Service will lower the load on the receiving streaming session by applying a payload type specific "back pressure latency threshold". When a payload's streaming latency exceeds this threshold, the payload will be dropped. By temporary dropping a specific portion of the load before transmission the receiver has the opportunity to "catch up".

Payload type	Latency threshold
MAP	<none>
SPaT	1000 ms
DENM	<none>
SSM	<none>
CAM	1000 ms
Secure CAM	<none>
SRM	<none>
Secure SRM	<none>

9.4.4 Load balancing

It is possible to balance the streaming load over more then one connection. Since it is required to explicitly specify the TLCs while create a new streaming session, parties can setup multiple connections over which the total amount of TLCs is distributed. It is also possible to change the "TLC scope" of an already connected streaming session in cases where TLCs should be added or removed from a connected session. There a few things to consider regarding the management of session TLC scopes:

- Adding TLCs to an active session could be refused due to lack of resources on the session's Streaming Service Node. If the TLC cannot be added to an other session a new session needs to be created, or the current session needs to be reestablished;
- Brokers can poll the API for new TLC registrations and add TLCs at any given time. In case of reestablishing sessions it is best to do this during daily designated maintenance windows (midnight);
- TLCs can never be in scope of more then one broker session of the same account and can never be in scope of more then one TLC session. When redistributing TLCs between connections the TLCs should first be removed from the original session before adding it to the new session. Redistribution of TLCs is best done during daily designated maintenance window since it will always cause a brief intermission of the moved TLC's payload stream.

9.5 Streaming Service API

In order to establish a TCP connection with the Streaming Service, the Streaming Service REST API needs to request a new "Streaming Service Session". The Streaming Service API will validate whether the requested session is allowed and prepare a "Streaming Service Node" in the Streaming Service platform for the requested session.

9.5.1 Security

Encryption

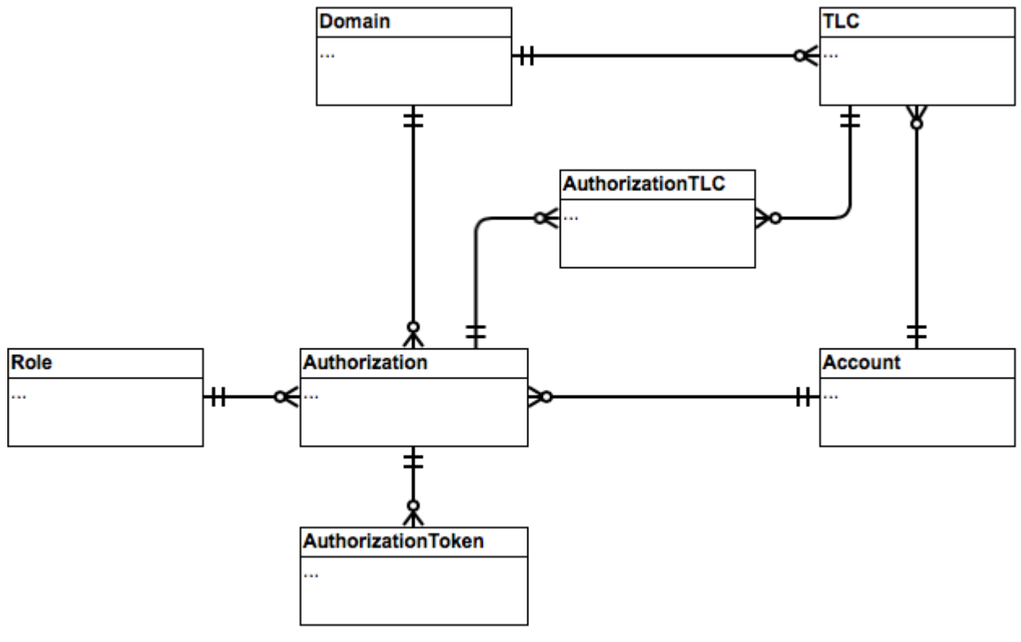
The Streaming Service API will be exposed using HTTPS only.

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.

Authorization model

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



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.

Roles

The following roles have been defined:

Name	Access	Access scope	Intended for
Platform administrator	Can manage accounts Can manage domains Can manage all TLC registrations Can manage all sessions Can manage authorizations Can manage authorizations tokens Can view all session logs and metrics	Platform wide	Platform setup and administration
Domain administrator	Can manage own TLC registrations within the Domain scope Can manage all sessions within the domain scope Can manage own authorizations within the domain scope Can manage own authorization tokens within the domain scope Can view all session logs and metrics within the domain scope	Domain, Account	Testing (TLC and broker organisations)
TLC administrator	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	Can create TLC sessions within the domain and authorization TLC scope	Domain, Account, Set of TLCs	TLC systems
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
Broker administrator	Can manage own "Broker system" authorizations within the domain scope Can manage own "Broker system" authorization tokens within the domain scope	Domain, Account	Broker organisations

Name	Access	Access scope	Intended for
	Can manage all own broker sessions within the domain scope Can view all own broker session logs and metrics within the domain scope Can view all TLC registrations within the domain scope		
Broker system	Can create broker sessions within the domain scope Can view all TLC registrations within the domain scope	Domain, Account	Broker system
Broker analyst	Can view broker session logs and metrics within the domain scope Can view all TLC registrations within the domain scope	Domain, Account	Analysts/engineers responsible for the broker system

9.5.2 API endpoints

⚠ Endpoint URI and JSON request/response bodies are subject to change upon implementation.

API endpoint			**Authorization role access (★ = platform scope, ★ = domain scope, ★ = own scope)**							
Method	**URI**	**Description**	**Platform admin**	**Domain admin**	**TLC admin**	**TLC system**	**TLC analyst**	**Broker admin**	**Broker system**	**Broker analyst**
POST	/sessions	Creates a new streaming session	★	★	★	★		★	★	
GET	/sessions	Retrieves all active streaming sessions	★	★	★	★		★	★	
GET	/sessions/<token>	Retrieves one active streaming session	★	★	★	★		★	★	
PUT	/sessions/<token>	Updates one active streaming session	★	★	★	★		★	★	
DELETE	/sessions/<token>	Ends one active streaming version	★	★	★			★		
GET	/sessionlogs	Retrieve all session logs	★	★	★		★	★		★
GET	/sessionlogs/<token>	Retrieve one specific session's log	★	★	★		★	★		★
POST	/tlcs	Create a new TLC registration	★	★	★					
GET	/tlcs	Retreive all TLC registrations	★	★	★		★	★	★	★
GET	/tlcs/<uuid>	Retrieve one specific TLC registration	★	★	★		★	★	★	★
DELETE	/tlcs/<uuid>	Removes one specific TLC registration	★	★	★					
POST	/domains	Create a new domain	★							
GET	/domains	Retrieve all domains	★							
GET	/domains/<name>	Retrieves one specific domain	★							

API endpoint			Authorization role access (★ = platform scope, ★ = domain scope, ★ = own scope)							
DELETE	/domains/<name>	Removes on specific domain	★							
POST	/accounts	Create a new account	★							
GET	/accounts	Retrieve all accounts	★							
GET	/accounts/<uuid>	Retrieves one specific account	★							
PUT	/accounts/<uuid>	Updates one specific account	★							
DELETE	/accounts/<uuid>	Removes on specific account	★							
POST	/authorizations	Create a new authorization	★	★	★			★		
GET	/authorizations	Retrieve all authorizations	★	★	★			★		
GET	/authorizations/<uuid>	Retrieves one specific authorization	★	★	★			★		
PUT	/authorizations/<uuid>	Updates one specific authorization	★	★	★			★		
DELETE	/authorizations/<uuid>	Removes on specific authorization	★	★	★			★		
POST	/authorizationtokens	Create a new authorization token	★	★	★			★		
GET	/authorizationtokens	Retrieve all authorization tokens	★	★	★			★		
GET	/authorizationtokens /<uuid>	Retrieves one specific authorization token	★	★	★			★		
PUT	/authorizationtokens /<uuid>	Updates one specific authorization token	★	★	★			★		
DELETE	/authorizationtokens /<uuid>	Removes on specific authorization token	★	★	★			★		

Sessions

POST /sessions

Creates a new streaming session.

Request

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 one of the predefined types: 1. TLC 2. Broker
protocol	The session protocol; must be one of the predefined types: 1. TCPStreaming_Singleplex 2. TCPStreaming_Multiplex

Name	Description
details	Session protocol specific details for creating the session

Response

Response body (JSON)

```
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

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.

Request details

Request protocol details (JSON)

```
{
    "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)

Response details

Response protocol details (JSON)

```
{
    "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

Example

request

```
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"
    }
}
```

response

```
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"
    }
}
```

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.

Request details

Request protocol details (JSON)

```
{
    "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)

Response details

Response protocol details (JSON)

```
{
    "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.

Example

request

```
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"]
    }
}
```

response

```
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"
    }
}
```

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.

⚠ The request and response details elements are identical to the ones described for "TLC" with protocol "TCPStreaming_Multiplex" sessions and are therefor not described again.

Example

request

```
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"]
    }
}
```

response

```
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"
    }
}
```

GET /sessions

Retrieves all active streaming sessions.

Intended for monitoring and debug purposes.

Request

Request

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

Response

Response body (JSON)

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

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

Example

request

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

response

```
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"
    }
 },
 {
    "token": "dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k",
    "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"
    }
 }
]
```

GET /sessions/<token>

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

Intended for monitoring and debug purposes.

Request

Request

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

Response

Response body (JSON)

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

<session specific details>
```

Example

request

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

response

```
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"
    }
}
```

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.

Request

Request

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

Response

Response body (JSON)

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

<protocol details>
```

Example

request

```
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"]
}
```

response

```
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"
    }
}
```

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.

Request

Request

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

Response

Response body (JSON)

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

Example

request

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

response

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

Session logs

GET /sessionlogs

Retrieves all streaming session logs.

Must be filtered by time range.

Intended for monitoring and debug purposes.

Request

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
```

Response

Response body (JSON)

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

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

Example

request

```
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
```

response

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


```
[
  {
    "token": "y_gUPT6kpTILfs-hHY3kW6P7It_kvV2NI0VKgyMkaA0",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "type": "TLC",
    "protocol": "TCPStreaming_Singleplex",
    "created": "2017-03-09T20:44:28Z",
    "connected": "2017-03-09T20:44:28Z",
    "remoteAddress": "/172.17.210.254:55624",
    "ended": null,
    "endReason": null,
    "tlcScopeHistory": [
      {
        "timestamp": "2017-03-09T20:44:28Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0001"
      }
    ]
  },
  {
    "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"
      },
      {
        "timestamp": "2017-03-10T11:23:12Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0005"
      }
    ]
  }
]
```

GET /sessionlogs/<token>

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

Intended for monitoring and debug purposes.

Request

Request

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

Response

Response body (JSON)

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

<session log>
```

Example

request

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

response

```
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"
      },
      {
        "timestamp": "2017-03-10T11:23:12Z",
        "scope": "ADDED",
        "tlcIdentifier": "tlc_0005"
      }
    ]
}
```

TLCs

POST /tlcs

Creates a TLC registration.

Request

Request

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

{
  "identifier": "<TLC identifier>"
}
```

Name	Description
identifier	The TLC identifier of the TLC Must be exactly 8 characters Must be unique within the domain Is case insensitive

Response

Response body (JSON)

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

{
  "uuid": "<TLC uuid>",
  "identifier": "<TLC identifier>",
  "domain": "<domain>",
  "account": "<account>"
}
```

Name	Description
uuid	The unique id for the created TLC
identifier	See request
domain	The domain in which the TLC is registered
account	Unique id of the account that "owns" the TLC registration

Example

request

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

{
  "identifier": "tlc_abcd"
}
```

response

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

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

GET /tlcs

Retreives all TLC registrations.

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

Request

Request

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

Response

Response body (JSON)

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

[
    <TLC>, <TLC>, ...
]
```

Example

request

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

response

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

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

GET /tlcs/<uuid>

Retreives the TLC registration with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

<TLC>
```

Example

request

```
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
```

response

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

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

DELETE /tlcs/<uuid>

Removes the TLC registration with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

Example

request

```
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
```

response

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

Domains

POST /domains

Creates a domain.

Request

Request

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

{
  "name": "<domain name>"
}
```

Name	Description
name	The unique name of the domain Must be at least 1 character Cannot be larger then 50 characters Is case insensitive (will be lower-cased upon creation)

Response

Response body (JSON)

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

{
  "name": "<domain name>"
}
```

Name	Description
name	See request

Example

request

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

{
  "name": "test"
}
```

response

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

{
  "name": "test"
}
```

GET /domains

Retreives all domains.

Request

Request

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

Response

Response body (JSON)

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

[
    <domain>, <domain>
]
```

Example

request

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

response

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

[
  {
    "name": "test"
  },
  {
    "name": "test2"
  }
]
```

GET /domains/<name>

Retreives the domain with the given "name".

Request

Request

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

Response

Response body (JSON)

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

<domain>
```

Example

request

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

response

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

{
    "name": "test"
}
```

DELETE /domains/<name>

Removes the domain with the given "name".

Request

Request

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

Response

Response body (JSON)

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

Example

request

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

response

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

Accounts

POST /accounts

Creates an account.

Request

Request

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

{
  "name": "<account name>"
}
```

Name	Description
name	The name of the account Must be at least 1 character Cannot be larger then 50 characters

Response

Response body (JSON)

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

{
  "uuid": "<account uuid>",
  "name": "<account name>"
}
```

Name	Description
uuid	The unique id of the created account
name	See request

Example

request

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

{
  "name": "test"
}
```

response

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

{
  "uuid": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
  "name": "test"
}
```

GET /accounts

Retreives all accounts.

Request

Request

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

Response

Response body (JSON)

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

[
    <account>, <account>
]
```

Example

request

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

response

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

[
  {
    "uuid": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "name": "test"
  },
  {
    "uuid": "026e59d4-68b3-4dfa-bb81-48f4ab2553ff",
    "name": "test2"
  }
]
```

GET /accounts/<uuid>

Retreives the account with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

<account>
```

Example

request

```
GET api/v1/accounts/80b142ab-88e8-4600-9a86-8807c19b1b2a HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

response

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

{
    "uuid": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "name": "test"
}
```

DELETE /accounts/<uuid>

Removes the account with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

Example

request

```
DELETE api/v1/accounts/80b142ab-88e8-4600-9a86-8807c19b1b2a HTTP/1.1
Host: api.tlex.eu
X-Authorization: dtNB_vhvJ0wgTGf1N0DxN38_AmTL_4yiPRZdqZSuK3k
Content-Type: application/json
```

response

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

Authorizations

POST /authorizations

Creates an authorization.

Request

Request

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

{
  "domain": "<domain name>",
  "account":"<account uuid>",
  "role": "<role>",
  "tlcIdentifiers": [<TLC identifier>, <TLC identifier>]
}
```

Name	Description
domain	The domain for which the authorization will be created ⚠ Optional, only relevant when called by a PLATFORM_ADMIN user
account	The account for which the authorization will be created ⚠ Optional, only relevant when called by a PLATFORM_ADMIN user
role	The role granted to the authorization, must be one of the predefined types: 1. PLATFORM_ADMIN 2. DOMAIN_ADMIN 3. BROKER_ADMIN 4. BROKER_SYSTEM 5. BROKER_ANALYST 6. TLC_ADMIN

Name	Description
	7. TLC_SYSTEM 8. TLC_ANALYST
tlcIdentifiers	Optional list of TLC identifiers for which the authorization is valid ⚠ Only applicable for authorizations with role TLC_SYSTEM and TLC_ANALYST ⚠ Optional, when not specified the authorization is valid for all TLCs owned by the authorization's account

Response

Response body (JSON)

```
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	See request
account	See request
role	See request
tlcIdentifiers	See request

Example

request

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

{
    "role": "TLC_SYSTEM"
}
```

response

```
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"
}
```

GET /authorizations

Retreives all authorizations.

Request

Request

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

Response

Response body (JSON)

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

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

Example

request

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

response

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

[
  {
    "uuid": "e72d150f-bd78-4395-b3ea-9101924b7bc1",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "TLC_ADMIN"
  },
  {
    "uuid": "c5ac28e6-9239-4c26-aa95-328a34446802",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "TLC_ANALYST"
  },
  {
    "uuid": "98a6890d-589d-43d4-bdff-3165425736d8",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_ADMIN"
  },
  {
    "uuid": "cd3e0ac6-4718-4f0e-8195-82e0c92e8cb6",
    "domain": "test",
    "account": "80b142ab-88e8-4600-9a86-8807c19b1b2a",
    "role": "BROKER_ANALYST"
  }
]
```

GET /authorizations/<uuid>

Retrieves the authorization with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

<authorization>
```

Example

request

```
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
```

response

```
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"
}
```

PUT /authorizations/<uuid>

Updates the authorization with the given "uuid".

Request

Request

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

<authorization>
```

Response

Response body (JSON)

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

<authorization>
```

Example

request

```
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"]
}
```

response

```
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"]
}
```

DELETE /authorizations/<uuid>

Removes the authorization with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

Example

request

```
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
```

response

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

Authorization tokens

POST /authorizationtokens

Creates an authorization token.

Request

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

Response

Response body (JSON)

```
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

Example

request

```
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"
}
```

response

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

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

GET /authorizationtokens

Retreives all authorization tokens

Request

Request

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

Response

Response body (JSON)

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

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

Example

request

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

response

```
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"
  }
]
```

GET /authorizationtokens/<uuid>

Retrieves the authorization token with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

<authorization token>
```

Example

request

```
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
```

response

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

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

PUT /authorizationtokens/<uuid>

Updates the authorization with the given "uuid".

Request

Request

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

<authorization token>
```

Response

Response body (JSON)

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

<authorization token>
```

Example

request

```
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"
}
```

response

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

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

DELETE /authorizationtokens/<uuid>

Removes the authorization token with the given "uuid".

Request

Request

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

Response

Response body (JSON)

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

Example

request

```
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
```

response

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

9.6 Session logging and metrics

9.6.1 Events

The following session related events will be captured in the session's log

Name	Description	Attributes
Create	The creation of the session (via the API)	timestamp, authorization token, session token, account, domain, session type, session protocol, tlc scope
Connect	TCP connection connect	timestamp, client ip
Update	The update of the session details (change of TLC scope)	timestamp, new tlc scope
End	TCP connection disconnect	timestamp, reason

9.6.2 Session metrics (time series)

Name	Description	Sample window	Measurements	Dimensions
Session roundtrip measurements	The amount of roundtrip delays measured on a session's connection using a received "Timestamps response" datagram	1 minute	datagram count	Session
Session roundtrip delay	The roundtrip delay in milliseconds that was measured on a session's connection using a received "Timestamps response" datagram	1 minute	avg, min, max delay in milliseconds	Session
Session clock measurements	The amount of clock differences measured on a session's connection using a received "Timestamps response" datagram	1 minute	datagram count	Session
Session clock difference	The clock difference in milliseconds that was measured on a session's connection using a received "Timestamps response" datagram	1 minute	avg, min, max difference in milliseconds	Session
Session clock difference utilisation	The utilisation percentage of the session's clock difference limit	1 minute	avg, min, max percentage	Session
Session rate utilisation	The utilisation percentage of the session's rate limit	1 second	percentage	Session

Name	Description	Sample window	Measurements	Dimensions
Session throughput utilisation	The utilisation percentage of the session's throughput limit	1 second	percentage	Session
Session payloads	The amount of payloads received from a session's connection (before any drops)	1 second	payload count	Session, TLC, Payload type
Session payload age	The payload age in seconds, calculated from the payload's "Source timestamp". The age has been offset by the clock difference	1 second	avg, min, max age in seconds	Session, TLC, Payload type
Session stale payloads	The amount of payloads received from a session's connection having their age exceed their type specific TTL	1 second	payload count	Session, TLC, Payload type
Session drops	The amount of payloads that were dropped on reception due to invalid TLC identifier or payload type	1 second	payload count	Session, TLC, Payload type
Stream drops	The amount of payload that were dropped before transmission due to invalid TLC identifier or payload type	1 second	payload count	Session pair, TLC, Payload type
Stream overflow drops	The amount of payloads that were dropped before transmission due to buffer overflows	1 second	payload count	Session pair, TLC, Payload type
Stream back pressure drops	The amount of payloads that were dropped before transmission due to back pressure policies	1 second	payload count	Session pair, TLC, Payload type
Stream latency	The processing latency (time between reception and transmission of a payload) in milliseconds between two sessions	1 second	avg, min, max latency in milliseconds	Session pair, TLC, Payload type
Stream payloads	The amount of payloads transmitted (excluding any drops)	1 second	payload count	Session pair, TLC, Payload type
Stream payloads with excessive latency	The amount of payload transmitted with excessive latency (excluding any drops)	1 second	payload count	Session pair, TLC, Payload type

9.7 TCP streaming protocol

The streaming protocol facilitates a continuous asynchronous bi-directional stream of datagrams.

9.7.1 Transport Layer Security

The Streaming Service optionally uses Transport Layer Security version 1.2 (TLSv1.2) to secure the TCP communication used for TCP streaming protocol sessions. The only supported cipher suite is "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256". Only server side authentication will be used.

9.7.2 Protocol version establishment

To support future protocol enhancement or replacement a one-byte version identifier is sent before the datagram streaming starts. When the peer does not support the protocol version the connection should be closed by the peer.

Protocol version (1 byte)
0x01

9.7.3 Protocol version 0x01

In protocol version 0x01 datagrams are streamed by using frames with a 4 byte header which is exists out of a 2 byte fixed prefix and a 2 byte frame data size. The header is followed by the frame data containing the actual datagram.

Byte Order

The byte order used by the protocol is "big-endian" also known as "network byte order": the byte containing the most significant bit (MSB) will be transmitted first.

Datagram framing

The datagrams are framed by having a 2 byte data size indication in the header. The header prefix, the fixed bytes 0xAA and 0xBB, has been added for "out of sync" detection and also to have 32 bit "aligned" header. By having the data size in the header no stream decoding is needed to read subsequent frames. Since the header data size is 2 bytes, one frame can hold a maximum of 63KB of data. The size of the frames should always by > 0; empty frames are not supported.

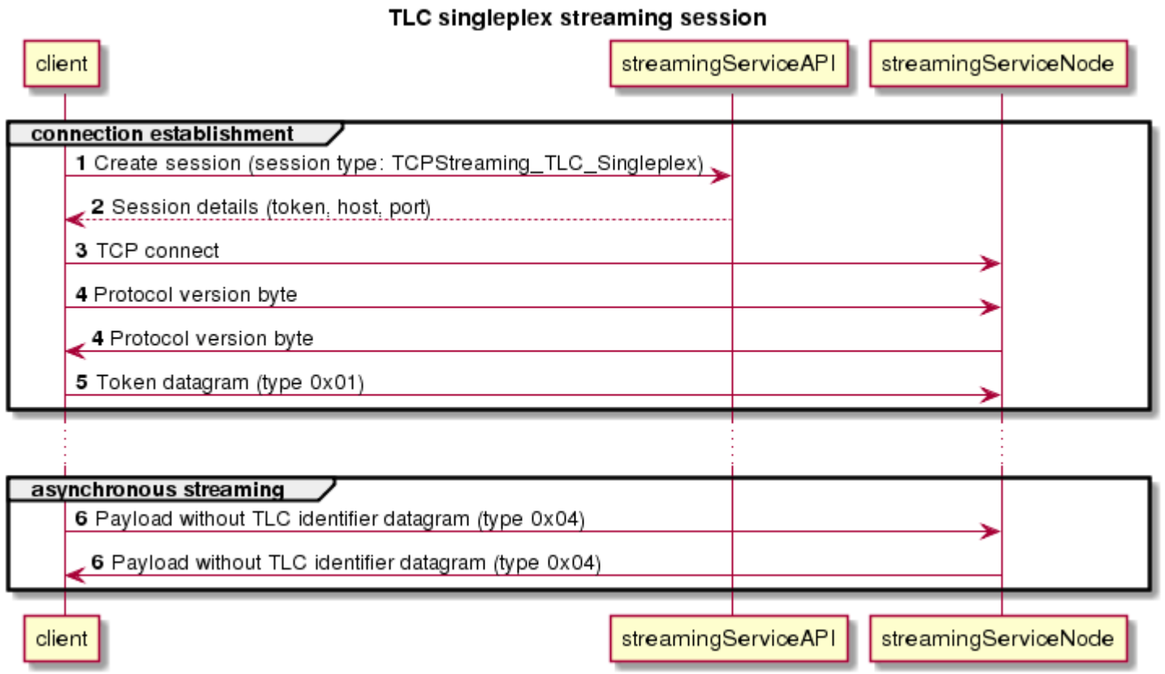
Version preamble (1 byte)	**Frame 1**			**Frame 2**			**Frame n**		
	Header		**Data**	**Header**		**Data**	**Header**		**Data**
	Prefix	**Data size**		**Prefix**	**Data size**		**Prefix**	**Data size**	
0x01	0xAA BB	0x00 01	0x00	0xAA BB	0x00 04	0x04 01 03 23	0xAA BB	...	...

⚠️ The fixed header prefix bytes should always be checked (asserted) since a mismatch would indicate a framing issue resulting in data corruption. When such a framing issue is detected the connection should be terminated as soon as possible.

Communication modes

TLC singleplex mode

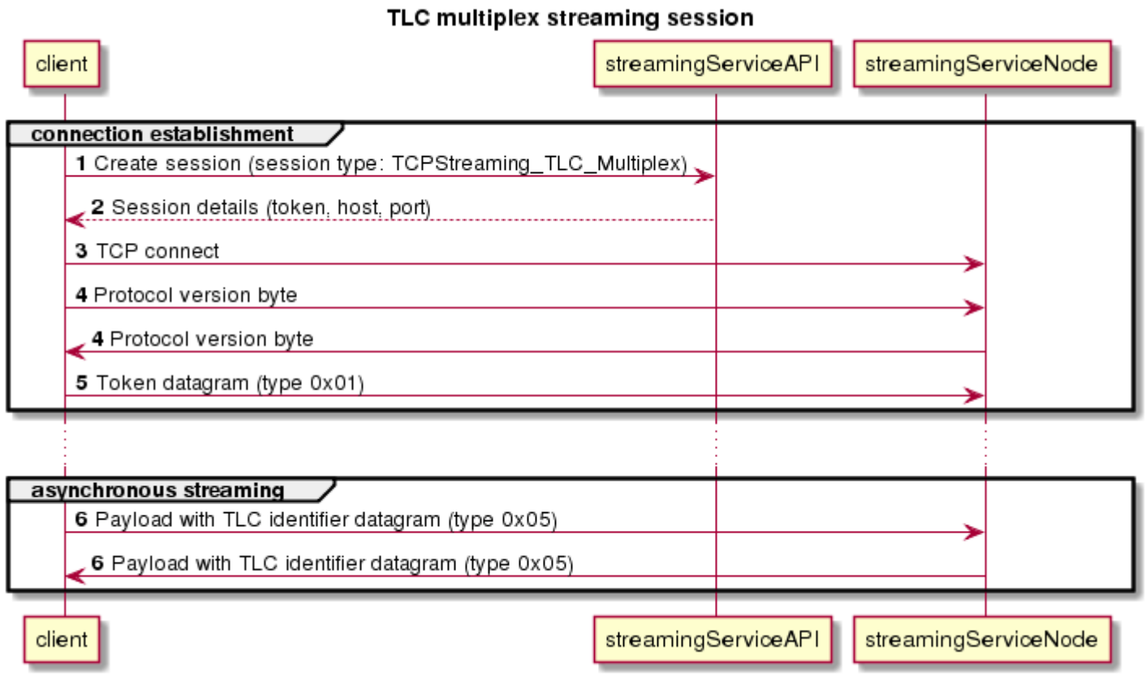
This mode can be used by "Cluster 1" organisations for direct TLC connections. Streaming sessions of this type will receive all payloads from "Broker" sessions when the "TLC identifier" matches the "TLC identifier" specified in the "sessions" API call used for creating the streaming session. Only one session per "TLC identifier" will be allowed.



TLC multiplex mode

This mode can be used by the "Cluster 1" organisations. Streaming session in this mode will receive all payloads from "Broker" sessions when the "TLC identifier" matches one of the "TLC identifiers" specified in the "sessions" API call used for creating the streaming session.

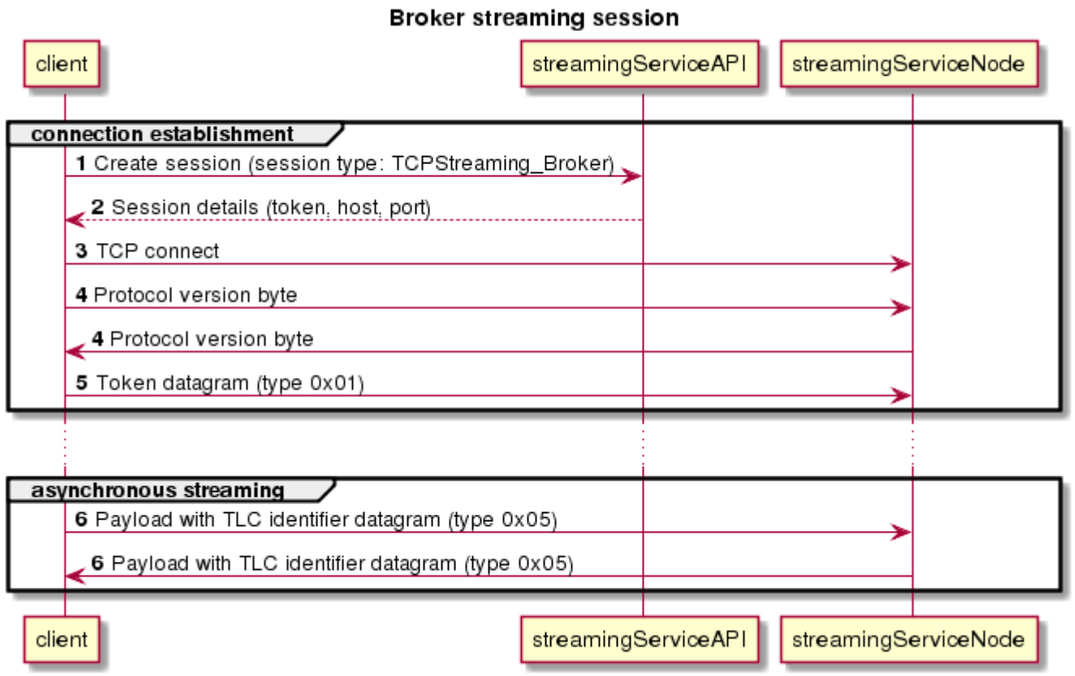
⚠ If one of the "TLC identifiers" in the "create session" call is already being used in an other active session, the API will refuse the "create session" call.



Broker mode

This mode will be used by the "Cluster 2" organisations. Streaming session in this mode will receive all payloads from "TLC singleplex" or "TLC multiplex" sessions when the "TLC identifier" matches one of the "TLC identifiers" specified in the "sessions" API call used for creating the streaming session.

⚠ If one of the "TLC identifiers" in the "create session" call is already being used in an other active session of the "Cluster 2" organisation, the API will refuse the "create session" call.



Datagrams

The frame data contains the actual datagram which is identified by a one byte "datagram type" identifier. The following datagrams are defined:

KeepAlive (0x00)

This datagram can be used to prevent disconnects when no data is available to sent within the keep alive timeout.

Datagram type (1 byte)
0x00

Token (0x01)

The first datagram sent by the Client should be "Token" in order the properly authenticate the Client.

Datagram type (1 byte)	Token (ASCII encoded)
0x01	<Token>

Bye (0x02)

This datagram can be used by either party to indicate that the connection will be closed. The reason (ASCII Encoded) is optional. It is the last datagram sent.

Datagram type (1 byte)	Disconnect reason (optional, ASCII encoded)
0x02	<Reason>

Reconnect (0x03)

This datagram can be used by the Streaming Service to instruct the Client to reconnect as soon as possible.

⚠️ A new session needs to be created using the API in order to reconnect.

Datagram type (1 byte)
0x03

Payload datagrams (0x04, 0x05)

The payload datagrams "carry" the actual payload which is being streamed by the Streaming Service. Both payload datagrams have an "Origin timestamp" field that should be filled with transmission time timestamp. This timestamp must be obtain from the same clock source as used to handle the "timestamps request" message. The payload type is identified by a single payload type byte.

Payload type byte	Payload type
0x00	MAP
0x01	SPAT
0x02	DENM
0x03	SSM
0x10	CAM
0x11	Secure CAM
0x12	SRM
0x13	Secure SRM

Payload without TLC identifier (0x04)

This datagram is in "TLC_Singleplex" communication mode to send and receive data and can only be used for singleplex sessions (session type "TCPStreaming_TLC_Singleplex").

Datagram type (1 byte)	Payload type byte (1 byte)	Origin timestamp (8 bytes)	Payload (n bytes)
0x04	<Payload type byte>	<UTC milliseconds>	<Payload>

⚠ If this datagram is used for a multiplex session, the session will be terminated by the Streaming Service.

Payload with TLC identifier (0x05)

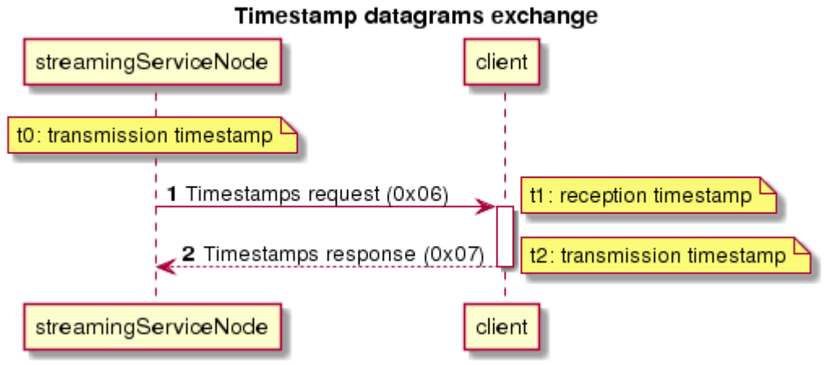
This datagram is used in "TLC_Multiplex" and "Broker" communication modes to send and receive data in a multiplex session (session types "TCPStreaming_TLC_Multiplex" and "TCPStreaming_Broker").

Datagram type (1 byte)	TLC identifier (8 bytes, ASCII encoded)	Payload type byte (1 byte)	Origin timestamp (8 bytes)	Payload (n bytes)
0x05	<TLC identifier>	<Payload type byte>	<UTC milliseconds>	<Payload>

⚠ In communication modes "TLC_Multiplex" and "Broker" only datagrams with a "TLC identifier" requested for the streaming session are allowed. If a non-requested "TLC identifier" is used, the payload will be dropped by the Streaming Service.

Timestamp datagrams (0x06, 0x07)

The timestamp datagrams are used to determine connection latency and clock difference.



Timestamps request (0x06)

This datagram contains the transmission timestamp which will be "echo'd back" by the recipient using the "Timestamps response" datagram. The Streaming Service will send this datagram at a regular interval (15 seconds) to measure the connection latency and to calculate the clock difference between the Streaming Service and the connected party. The connection latency is used to evaluate the accuracy of the calculated clock difference.

Datagram type (1 byte)	t0: transmission timestamp (8 bytes)
0x06	<UTC milliseconds>

Timestamps response (0x07)

This datagram must be used as response to a received "Timestamps request" datagram and must contain the original "t0: transmission timestamp" of this datagram. The response should by prioritised to any other traffic; in other words: it should be sent as soon as possible after receiving the "Timestamps request" message. The "t1: request reception timestamp" field must be filled with the reception time timestamp and should be determined as soon as possible upon reception of the "Timestamps request" message. The "t2: response transmission timestamp" field must be filled with the transmission time timestamp and should be determined as late as possible in the transmission process.

Datagram type (1 byte)	t0: transmission timestamp from request (8 bytes)	t1: request reception timestamp (8 bytes)	t2: response transmission timestamp (8 bytes)
0x07	<UTC milliseconds>	<UTC milliseconds>	<UTC milliseconds>