TOYOTA MOTOR CORPORATION

Common Specification for the Communication Interface between In-Car and Out-Car

Version 1.2

Internal distribution point/TMC	Copy
Electronics Development Dept. No.11 Connected System Development Div.	1
InfoTech Connected Advanced Development Div.	1
Smart Center Development Dept. Connected System Development Div.	1
Electronics Development Dept. No.13 Connected System Development Div.	1
Information Security Management Dept.	1
System network & architecture development Dept. E/E Architecture Development Div.	1
Software Defined Architecture Vehicle Development Dept. E/E Architecture Development Div.	1
BR Electronics Control Renovation Dept. E/E Architecture Development Div.	1
Digital Cockpit Software Development Dept. Connected Advanced Development Div.	1
Electronics Development Dept. No.13 Connected System Development Div.	1
OTA Dept.	1
Value Chain PF Development Div.	1
Advanced Project Promotion Div.	1
Software First	1
Advanced Electronics Development Dept.	1
Vehicle Dynamics Electronics Control System Development Dept. Electronics Control System Development Div.	1
Electronic Performance Development & Engineering Dept. Electronics Control System Development Div.	1
Testing Section No.2 Electronics Control System Development Div.	1
System Development Dept. No.2 Automated Driving & Advanced Safety System Development Div.	1

··· CONFIDENTIAL 秘 Communication Specification	System	Common Specification for the Communication Interface between In-Car and Out-Car	2
	No.	1.2	

External distribution point	Copy
Woven Planet	1
Woven Core	1
Woven Alpha	1
Suzuki Motor Corporation	1
SUBARU CORPORATION	1
DAIHATSU MOTOR CO.,LTD.	1
Mazda Motor Corporation	1

Created by

	Electronics Development Dept. No.11 Connected System Development Div.	Smart Center Development Dept. Connected System Development Div.	InfoTech Connected Advanced Development Div.
Created by	Katayama	S. Suzuki	Shibasaki
Approved by	Maeda	Toya	Maeda

Table of Contents

List of Tables

List of Figures

1. Objective and positioning

1.1. Objective

This document defines specification that should be commonly supported by applications that perform data communication between the Center for the next-generation vehicle in 2024 and the target ECU.
As the number of applications that communicate with the center implemented in in-Car units is expected to increase, the objective is to specify a generic protocol specification that can be used by applications and a common authentication specification between in-Car units and the center.
This document distributes to OEM which develops technical specifications for next-generation vehicle communications devices. The purpose is to make the specifications of each OEM connected to TSC common.

1.2. Positioning

The positioning of this document in relation to other specifications is shown in Figure 1-1 Position of this document

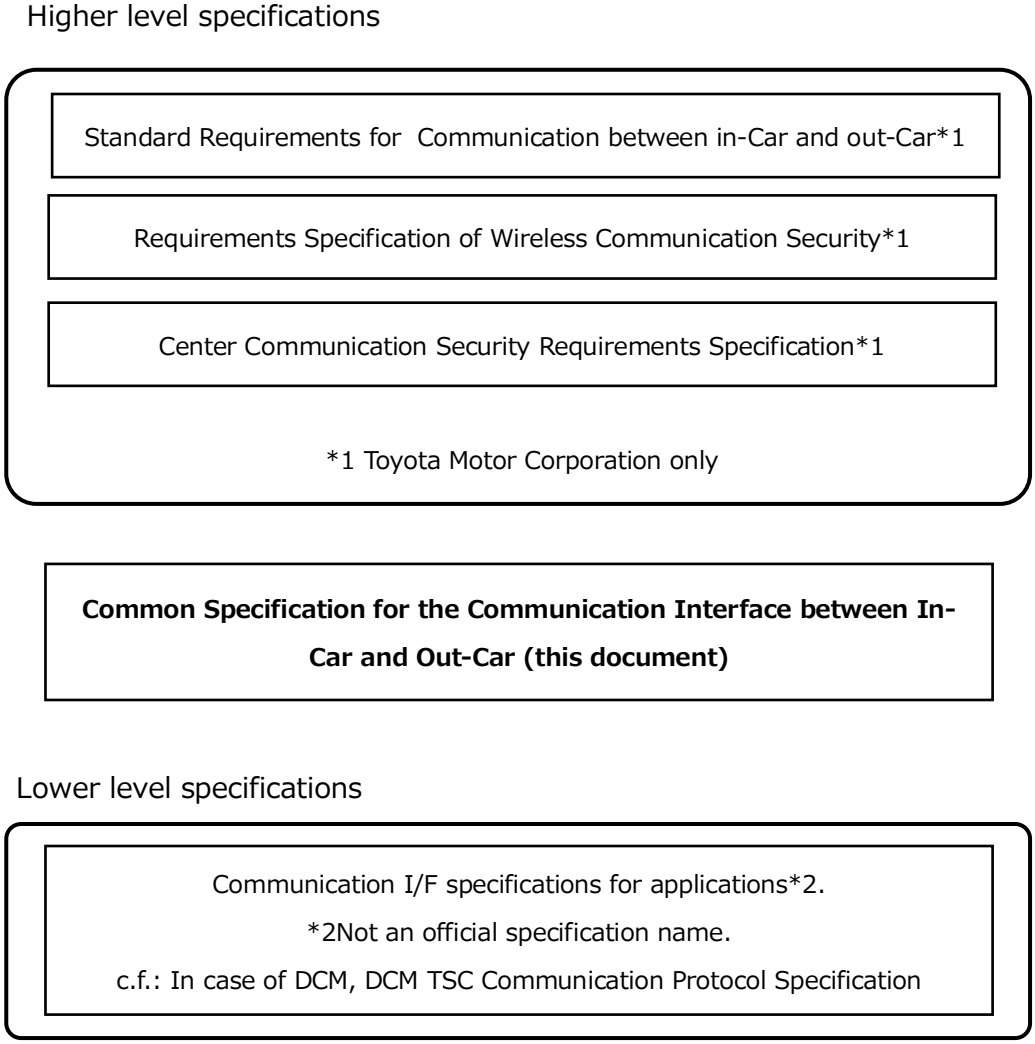


Figure 1-1 Position of this document

2. Definition of a Terms

The terms used in this specification are defined in Table 2-1.

Table 2-1 Definition of Terms

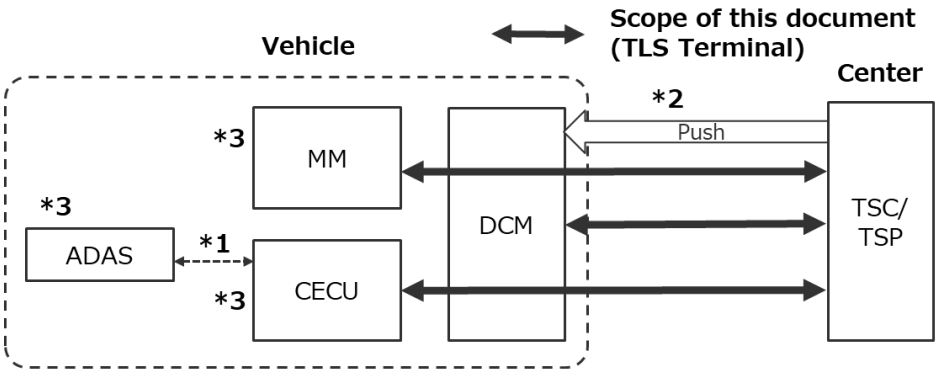
Terminology	Description
ADAS	Advanced Driver-Assistance System
ADF	Air Data Feed
ALB	Application Load Balancer
ALPN	Application Layer Protocol Negotiation
API	Application Programming Interface
CDP	CRL Distribution Point
CRL	Certificate Revocation List
DCM	Data Communication Module
DHE	Diffie-Hellman ephemeral
ECDHE	Elliptic curve Diffie-Hellman ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electronic Control Unit
HTTP	Hypertext Transfer Protocol
IDL	Interface Description Language
IMEI	International Mobile Equipment Identity
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
mTLS	Mutual TLS
NEV	New Energy Vehicle
NLB	Network Load Balancer
OAuth	Authorization framework (RFC6749、6750)
OCSP	Online Certificate Status Protocol
QoS	Quality of Service
RPC	Remote Procedure Call
RSASSA PKCS#1	RSASSA Public Key Cryptography Standards
RSASSA-PSS	RSASSA Probabilistic Signature Scheme
SNI	Server Name Indication
TBDC	TOYOTA Big Data Center
TSC	TOYOTA Smart Center
TSP	Telematics Service Providers
UTC	Universal Time Coordinated
V2X	Vehicle to X
VIN	Vehicle Identification Number
Electronic PF	Electronic platform (Platform)

3. Scope of Application

3.1. List of Applicable ECUs

Applicable ECUs are the ones equipped with applications that communicate with the TSC for the next-generation vehicle in 2024.
Specifically, DCM, Multimedia (MM), Central ECU (CECU), and ADAS ECU(ADAS).

The applicable ECUs and the scope of this document are shown in Figure 3-1 and Figure 3-2
The scope of this document applies to protocols higher than TLS. (See 6.1.3 Protocol Stack)

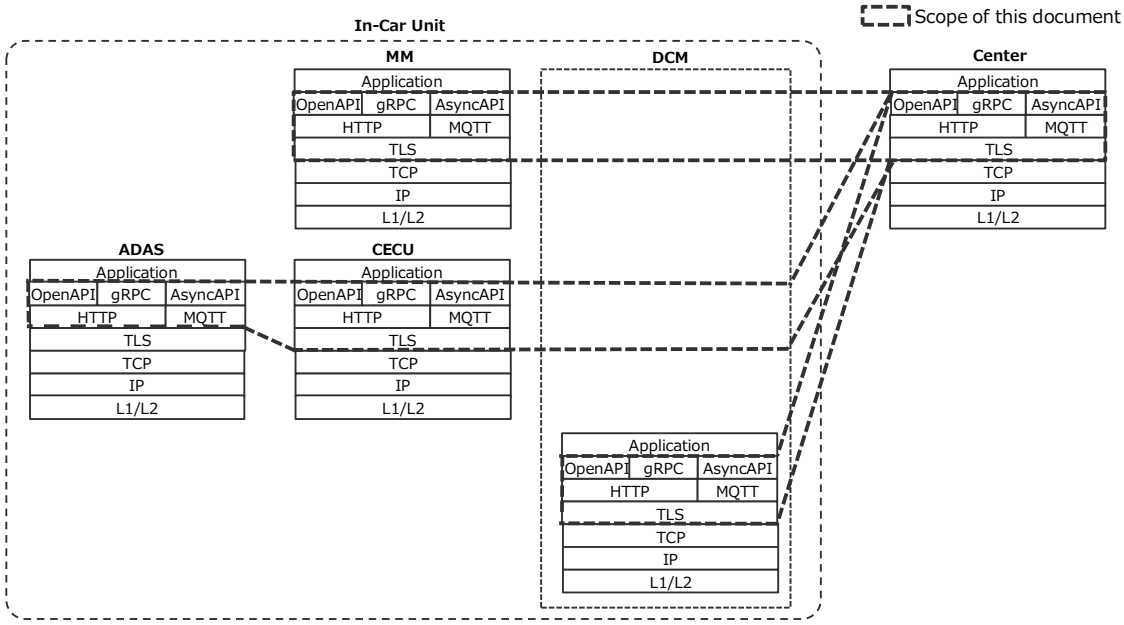


* 1 It is assumed that ADAS satisfies communication specification described in this document via CECU.
Function allocation and communication specification between ADAS and CECU are outside the scope of this document.

* 2 It is assumed that DCM receives push messages from the center to in-Car units and the DCM forwards the messages to other ECUs.
Therefore this document describes communication specification only between DCM and the center.
Communication specification from DCM to other ECU is outside the scope of this document.

*3 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS.
Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

Figure 3-1 Applicable ECUs and the scope of this document



※This figure shows applicable scope and protocols (HTTP, gRPC, MQTT etc.) in the figure might be different from actual ones used in each ECU.
Refer to specifications of each ECU for the actual protocols used in the ECU.

Figure 3-2 Applicable ECUs and Application Scope (Protocol Stack)

3.2. Applicable Electronic PF

This chapter applies only to Electronic PF for Toyota Motor Corporation.
Table 3-1 shows the list of applicable electronic PFs.

Table 3-1 Applicable Electronic PF

Electronic PF	DCM	C-ECU	MM	ADAS
19v1	N/A	N/A	N/A	N/A
19v2	*Note 1	*Note 1	*Note 1	*Note 1
P19ePF	24DCM	C-ECU	p21MM	ADAS ECU

Note 1: There is a possibility that p19 generation ECUs will be installed in 19v2 electronic platform. In that case, this specification is possibly applied.
On the other hand, there is a possibility that the 19v2 generation software (which does not conform to this specification) will be used for the sake of uniformity in each electronic platform.
Therefore, this specification also applies to the case where a p19 generation ECU is installed in 19v2 electronic platform, but the scope of compliance (partial or full compliance) is not specified.

3.3. List of Applicable Servers

Applicable servers are the TSC and TSPs which is under control of local affiliates, that is for the next-generation vehicle in 2024.
The latter board of TSC/TSP which is the end of Center is outside the scope.

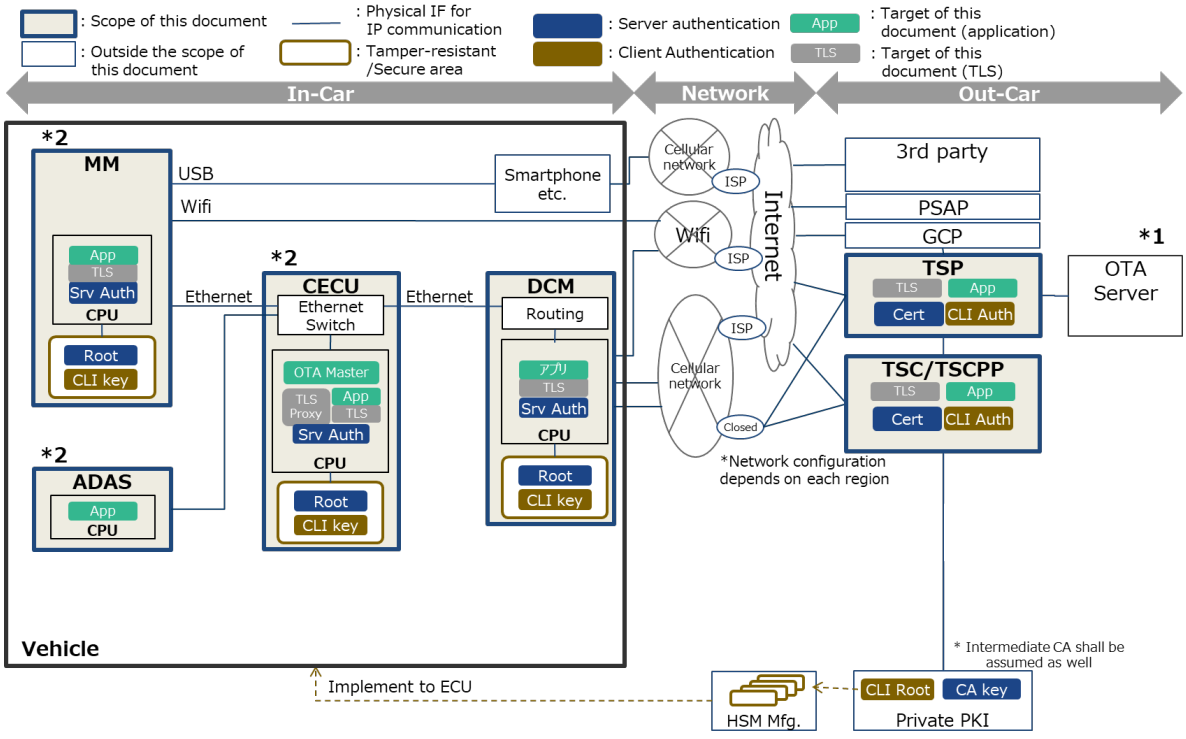
3.4. List of Applicable Destinations

If destination is not specified, it should be common for all destinations.

4. System Architecture

The system architecture covered in this document is shown in Figure 4-1.

In accordance with the reference specification 2 "Requirements Specification of Wireless Communication Security", ECUs (DCM, MM and CECU) in the first layer directly communicate with TSC and TSP, while ADAS in the second layer communicates via TLS Proxy. (*TSPs which communicate with ECUs via TSC are out of scope of this document and only TSPs which directly communicate with ECUs are in the scope of this document.) Also, Proxy side (CECU) shall relay communications only from ECUs which are allowed by authentication and so on in advance, so that unspecified ECUs are not allowed to communicate via TLS Proxy.



*1 Depending on the Center configuration, the TSP and OTA Server connection routes may differ. Refer to the communication-specifications of each ECU for details.

*2 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS. Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

Figure 4-1 System Architecture

5. Use Cases

Typical use cases covered in this document are defined here.
The use cases are explained by service-based scenarios and communication characteristics.

Example scenarios for the use cases covered in this document are shown below.

- Uploading CAN data
- Uploading video data
- Uploading diagnostics data
- API call from vehicle to the center
- Push from the center (unicast and multicast)
- Obtain an access token (initial authentication)
- Data download
- OTA*.
  *This document applies only to authentication, and doesn't cover application protocols higher than TLS.

Following use case is outside the scope of this document.

- Voice call

The communication characteristics are classified as follows by classifying the communication in the use cases according to the type of communication data, request source, periodicity, etc.

- Communication type
  Indicates the purpose of the communication and classified as follows
  - Command
  - Data communication
- Request source
  Indicates whether the communication is initiated by ECU or by the center.
- Periodicity
  Indicates the cycle (timing) at which communication takes place and is classified as follows
  Cycle (regular communication)
  Event (communications that occur irregularly)
- Communication Frequency

Indicates the frequency with which communication occurs.

- Responsiveness
  Indicates the time required for the request source to receive the first response after sending the request.
- Data size
  Indicates the data size of the communication per event or per cycle.

Table 5-1 shows the use cases.

Table 5-1 Use cases

Scenario	Description.	Communication Characteristics					
		Comm. Type	Request Source	Periodicity	Comm. Frequency	Responsiveness	Data Size
Uploading CAN data	Periodic uploads (every minute) - Size of a few tens of KB - Simultaneous connection with a large number of ECUs	Data comm.	ECU	Cycle	High (per sec ~ per min)	Middle (a few sec)	Small~Middle (several tens KB ~ several MB)
Uploading video data	Non-stationary (on-demand) data - data size of several MB - continuous data transmission	Data comm	ECU	Event	Middle (several times/ hour)	Low (tens of sec)	Big (several MB ~ several tens MB)
Uploading diagrnostics data	Uploading only when an event occurs, such as an accident, etc. Data size per event is several to several dozen MB .	Data comm	ECU	Event	Low (several times/month ~ several times/day)	Middle (a few sec)	Small~Middle (several KB ~ several tens KB)
API call from vehicle to the center	- Event notification etc. for remote services - High responsiveness (a few seconds) - Data size is a few KB - Confirmation of arrival	Command	ECU	Event	Middle (several times/ hour)	Middle (a few sec)	Small~Middle (several KB ~ several hundreds KB)
Push from the center (unicast and multicast)	Push during application communication from the center to the vehicle for remote services Data size is a few KB.	Command	Center	Event	Middle (several times/ hour)	Middle (a few sec)	Small (several KB)
Obtain an access token (initial authentication)	- Only for the first access during IG-ON. - Access token is issued - Data size is a few KB - Confirmation of arrival	Data comm	ECU	Event	Low (several times/ day)	Middle (a few sec)	Small (several KB)
Data download	- Non-stationary (on-demand) data - Data size of several MB - Continuous data reception	Data comm	ECU	Event	Middle (several times/ hour)	Low (tens of sec)	Big (several MB ~ several tens MB)

6. Communication Requirements

6.1. Communication Protocol

6.1.1. Selection of communication protocol

In the use case shown in Section5, it is necessary to select an appropriate communication protocol by considering the following

- The method should be cost effective when a large amount and high frequency of diverse data is uploaded.
- The function which performs constant center communication is based on always-connected communication with the center.
- If an HTTP session is established with the center for each application, the center will need to maintain a large number of TCP connections, which will increase costs. Design in which each in-Car unit reuse one session is required.

Under consideration of the above, the selected communication protocols and the concept of using them for different use cases are shown below.

- gRPC
  - Recommended for many use cases.
- HTTP/2
  - Recommended for many use cases.
- MQTT
  - Apply to use cases where communication by Publish/Subscribe is useful.
- HTTP/1.1
  - Apply only when it is necessary to follow existing (legacy) communication specifications. (Not recommended)

For the description of the API between in-Car unit and TSC, it is recommended to use OpenAPI, Protocol Buffers, and AsyncAPI for the following reasons.

[Reason]

In implementing APIs, there are advantages such as the ability to generate client libraries, create server stubs, and generate documents (specifications).

In addition, HTTP/3 is to be considered in the future and is not subject to

application to this applicable e-PF.

Table 6-1 Comparison of Communication Protocols

Classification	Item	Communication Protocol			
		HTTP/1.1	HTTP/2	gRPC	MQTT
Communication model	Connection	Whenever (Not reusable)	Single (reusable)	Single (reusable)	Single (reusable)
	Session	Stateless	Stateful	Stateful	Stateful
	Communication direction	Client → Server	Bidirectional	Bidirectional	Bidirectional
Communication quality	Arrival guarantee	Depends on TCP	Depends on TCP	Depends on TCP	Possible (depending on QoS settings)
	Order guarantee	Impossible	Depends on TCP	Depends on TCP	Possible (depending on QoS settings)
	Responsiveness	Bad (with delay due to connection establishment)	Good	Good	Bad (Large delay due to asynchronous communication)
	Priority control	Impossible	Possible	Possible	Possible
Security	Secret	Depends on TLS	Depends on TLS	Depends on TLS	Depends on TLS
	Authentication	Basic authentication Digest authentication Dependent on other libraries such as TLS, OAuth, etc.	Dependent on other libraries such as TLS, OAuth, etc.	Dependent on other libraries such as TLS, OAuth, etc.	Basic authentication Depends on TLS

6.1.2. Communication Protocols and Use Cases

For each communication protocol, the suitability for the typical use cases shown in the section5 use cases is shown in Table 6-2.

Table 6-2 Communication protocols and their suitability for use cases

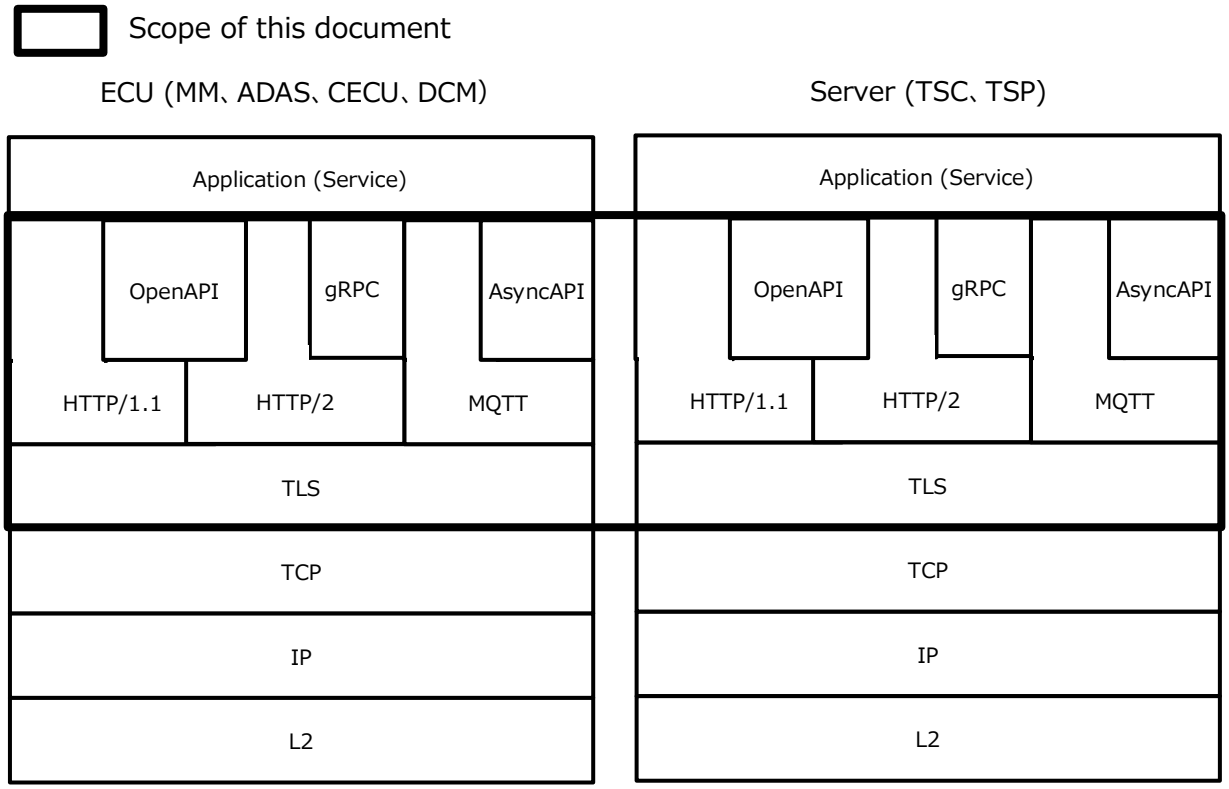
Scenario	Communication characteristics						Protocol			
	Comm. Type	Request Source	Periodcity	Comm. Frequency	Responsiveness	Data Size	HTTP/1.1	HTTP/2	gRPC	MQTT
Uploading CAN data	Data comm.	ECU	Cycle	High (per sec ~ per min)	Middle (a few sec)	Small~Middle (several tens KB ~ several MB)	X Cyclic communication is not suitable.	O	O	O
Uploading video data	Data comm	ECU	Event	Middle (several times/ hour)	Low (tens of sec)	Big (several MB ~ several tens MB)	O	O*1	Δ Large data needs to be split.	O
Uploading diagrnostics data	Data comm	ECU	Event	Low (several times/month ~ several times/day)	Middle (a few sec)	Small~Middle (several KB ~ several tens KB)	O	O	O	O
API call from vehicle to the center	Command	ECU	Event	Middle (several times/ hour)	Middle (a few sec)	Small~Middle (several KB ~ several hundreds KB)	O*1	O*1	O	X Poor asynchronous responsiveness
Push from the center (unicast and multicast)	Command	Center	Event	Middle (several times/ hour)	Middle (a few sec)	Small (several KB)	X Not suitable for push communication	O	O	O
Obtain an access token (initial authentication)	Data comm	ECU	Event	Low (several times/ day)	Middle (a few sec)	Small (several KB)	O*1	O*1	Δ Redundant for one-time communication only	X Not suitable for one-time communication
Data download	Data comm	ECU	Event	Middle (several times/ hour)	Low (tens of sec)	Big (several MB ~ several tens MB)	O*1	O*1	Δ Large data needs to be split.	O

O: Suitable, Δ: Conditionally applicable, X: Not suitable

*1 When session maintenance is not required, HTTP/1.1 is suitable. HTTP/1.1 is suitable when session maintenance is not required. Multiple HTTP/1.1 communications using HTTP/2 streams can be used for efficient communications.

6.1.3. Protocol Stack

Protocol stack and scope of this document are shown in Figure 6-1.
It is recommended to use OpenAPI, gRPC and AsyncAPI to describe APIs between in-Car unit and the TSC.
(Because of the advantages of generating client libraries, creating server stubs, and generating documentation (specifications).)



* It is assumed that each ECU can resolve the address by DNS.

Figure 6-1 Protocol Stack

6.2. Security

6.2.1. Transport Layer Security

Communication between in-Car unit and the center shall be performed using TLS. This will provide encrypted communication and server authentication using server certificates.

6.2.2. Client Authentication

When communicating with the TSC and TSP, mutual authentication shall be performed using a client certificate issued by Toyota.

6.2.3. Certificate

When communicating with TSC and TSP, shall only trust certificates be chaining up to root certificates issued by Toyota Motor Corporation shall be used.

6.2.4. Application Layer Security

It is recommended that DNS server supports DNSSEC, it should be compliant with RFC4033.

- DNS over HTTPS should be compliant with RFC8484.
- DNS over TLS should be compliant with RFC7858.

6.3. Session Management

Sessions for communication between in-Car unit and TSC/TSP should be maintained as much as possible.

6.4. Vehicle Identification

The TSC should be able to identify which vehicle the communication from in-Car unit (ECU) is from.

Authentication shall be performed for in-Car unit (ECU) while telematics service shall be provided for vehicles. Therefore, it is not possible to judge the needs of telematics service only by authentication, and the center side shall perform a validity check between in-Car unit (ECU) and vehicle (vehicle identifier) for preventing impersonation of genuine products.
Server shall perform authorization in addition to authentication to make sure the In-Car unit is authorized to access specific services. This shall be compliant with RFC8705 to bind the authorization token to the client certificate.

7. Communication Specifications

7.1. Transport Layer Security

7.1.1. TLS

7.1.1.1. Basic Specifications

The transport protocol shall conform to the following TLS1.3 specification.

- RFC8446 The Transport Layer Security (TLS) Protocol Version 1.3

The application protocol must use TLS.

Support SNI.

If HTTP/2 upgrade is performed by ALPN, it should be compliant with RFC7301.

7.1.1.2. Encryption Algorithm

The cipher suites, key exchange, and signature schemes are shown below.

7.1.1.2.1. Cipher Suite

Table 7-1 Cipher suites

Priority	Cipher Suite	Encryption (Enc) AEAD used as Message authentication code (Mac)
1	TLS_AES_256_GCM_SHA384	Enc=AESGCM(256) Mac=AEAD
	TLS_CHACHA20_POLY1305_ SHA256	Enc=CHCHA20/POLY1305(256) Mac=AEAD

7.1.1.2.2. Key Exchange Method

- ECDHE 384 bits or more
- DHE 4096 bits or more

7.1.1.2.3. Signature Method

The following 2 algorithms shall be kept and utilized to be able to correspond when a vulnerability is found in the encryption algorithm in the future.

- ECDSA256
- RSA3072
  - RSASSA PKCS#1 3072
  - RSASSA-PSS 3072

7.1.1.3. Client Authentication

For mutual authentication (mTLS) between in-Car unit and the server, the server shall perform client authentication based on TLS1.3 (RFC8446).

Server shall validate the client certificate chain is issued by Toyota Motor Corporation.

7.1.1.4. Session Management

The session resumption method shall be applied to TLS sessions, except for new connections, in order to reduce the processing load on both in-Car unit and the center.
TLS session resumption shall be performed using the stateless resumption PSK+(EC)DHE 1-RTT method* in consideration of ensuring forward secrecy.
* 0-RTT method should not be applied because it requires the application to take countermeasures against replay attacks.

TLS session timeout shall be set as 12 hours.

Take care not to generate SIGPIPE (abnormal termination of the process caused by using closed socket) on reestablishment of connection etc.

7.1.1.5. Certificate Management

- In-Car unit shall check the revocation of all certificates in the server certificate chain, and the server shall check the revocation of the client certificates in the client certificate chain.

[Specifications common to in-Car unit and server]

- The following shall be addressed for checking certificate revocation.
  - CRL(RFC5280)
  - OCSP (RFC5019, RFC6066, RFC6960, RFC7633, RFC8446)
- CRL/OCSP Response shall be obtained from the CRL distribution point/OCSP responder listed in each certificate obtained in TLS connection.
- If any certificate in the certificate chain is revoked, no communication shall be performed.
- If it cannot be confirmed by a valid CRL/OCSP Response, no communication shall be performed.
- The latest CRL/OCSP Response shall be obtained in any of the following cases,
  (1) The validity period of the CRL/OCSP Responce has expired.
  (2) Unable to determine the validity period of CRL/OCSP Response
  (3) No internal CRL/OCSP Response is maintained.

- The CRL/OCSP Response retrieved from the CRL distribution point/OCSP responder shall be cached as content. For cache control, the following shall be supported. The basic flow is shown in Figure 7-1.
  - Content Cache Control (RFC7234)
  - E-Tag and Last-Modified (RFC7232)
- If the CRL/OCSP Response cannot be retrieved at the time of cache update, the cache shall not be cleared, and the revocation check shall be performed using the retained CRL/OCSP Response, as long as it is not expired.

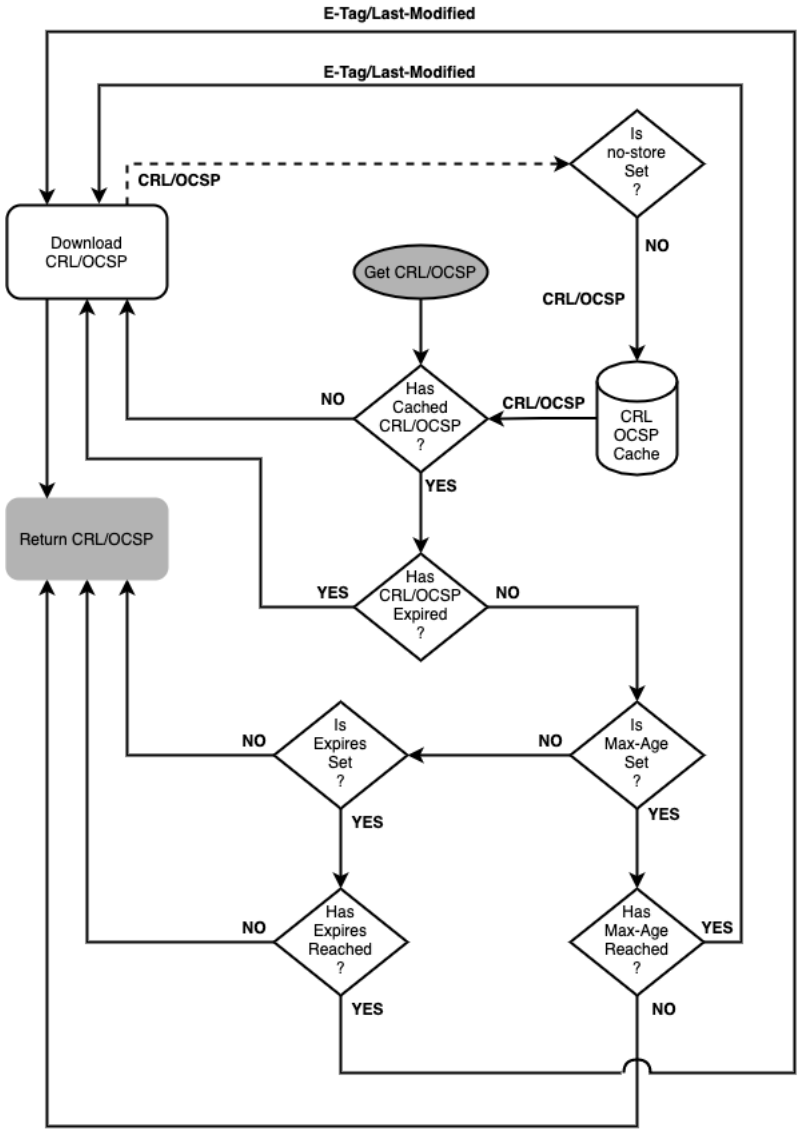


Figure 7-1 CRL and OCSP caching flow diagram

[Specifications only for in-Car unit]

- When in-Car unit uses CRL/OCSP Response for certificate revocation check, retain the acquired CRL/OCSP Response until the expiration date indicated in the CRL/OCSP Response regardless of the IG status, it should be compliant with RFC7234.

  However if CRL/OCSP Response for the same certificate already exists in newly acquired CRL/OCSP Response, the existing CRL/OCSP Response shall be discarded and updated with the newly acquired CRL/OCSP Response.

  In addition even before the expiration date, CRL/OCSP Response shall be

discarded only when +B is turned OFF.

- When In-Car unit uses OCSP for certificate revocation check, In-Car unit shall send *status_request* TLS extension as described in RFC6066.
- When OCSP is stapled in the server certificate, the OCSP shall be checked for revocation based on the stapled OCSP response.
- When must-staple extension is present in any of the server certificate in the chain, in-Car unit shall reject connection if OCSP is not stapled for any certificate with the extension, it should be compliant with RFC7633.
- CRL and OCSP shall be supported for revocation checking of server certificate issued by Toyota as shown Figure 7-2.
- CRL shall be used for revocation check of server certificate issued by Toyota.

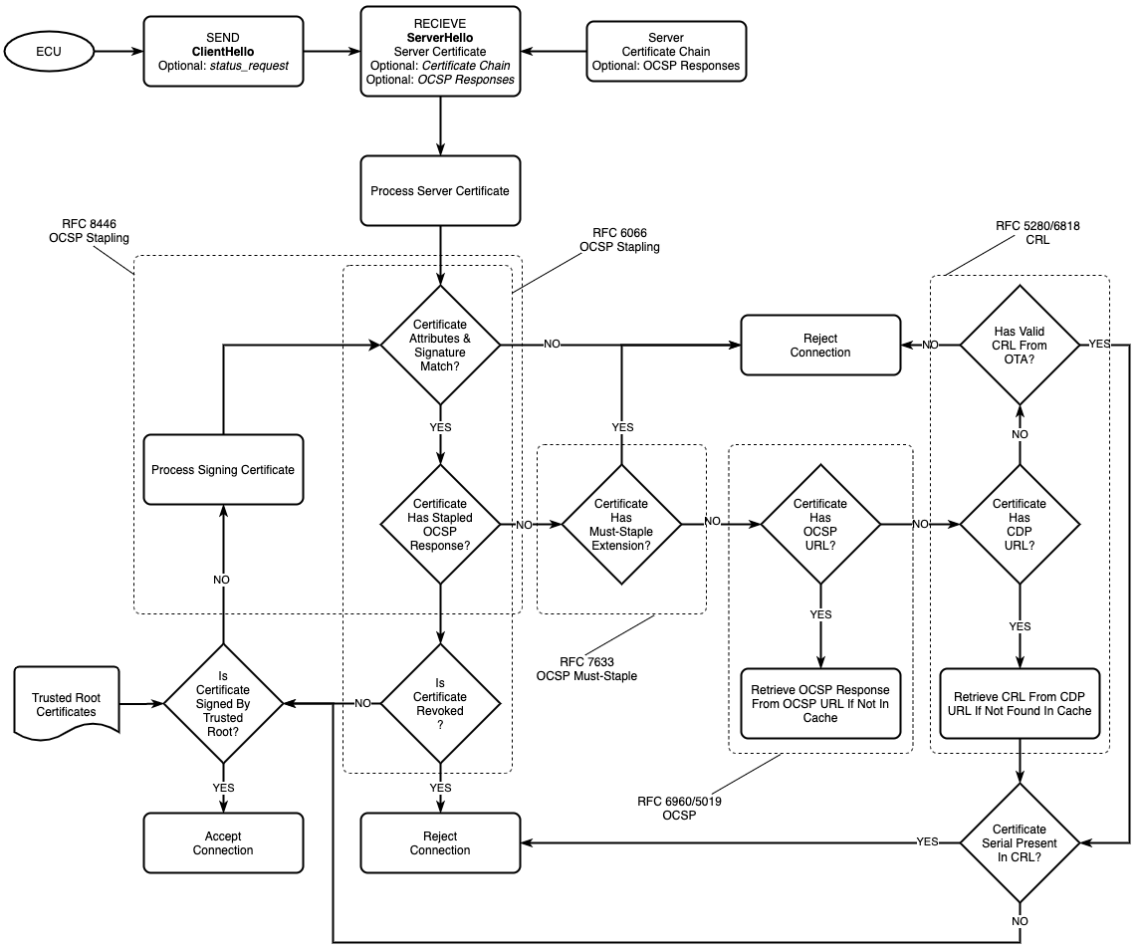


Figure 7-2 Revocation checking flow diagram

[Specifications only for the server]

- In case that OCSP is used for revocation check for client certificate, the server shall not support OCSP stapling.

7.1.1.6. Basic Sequence

As the basic sequence when using TLS1.3, the sequence and messages during the full handshake are shown in Figure 7-3 and
Table 7-2 and the sequence and messages during session resumption are shown in Figure 7-4 and Table 7-3.

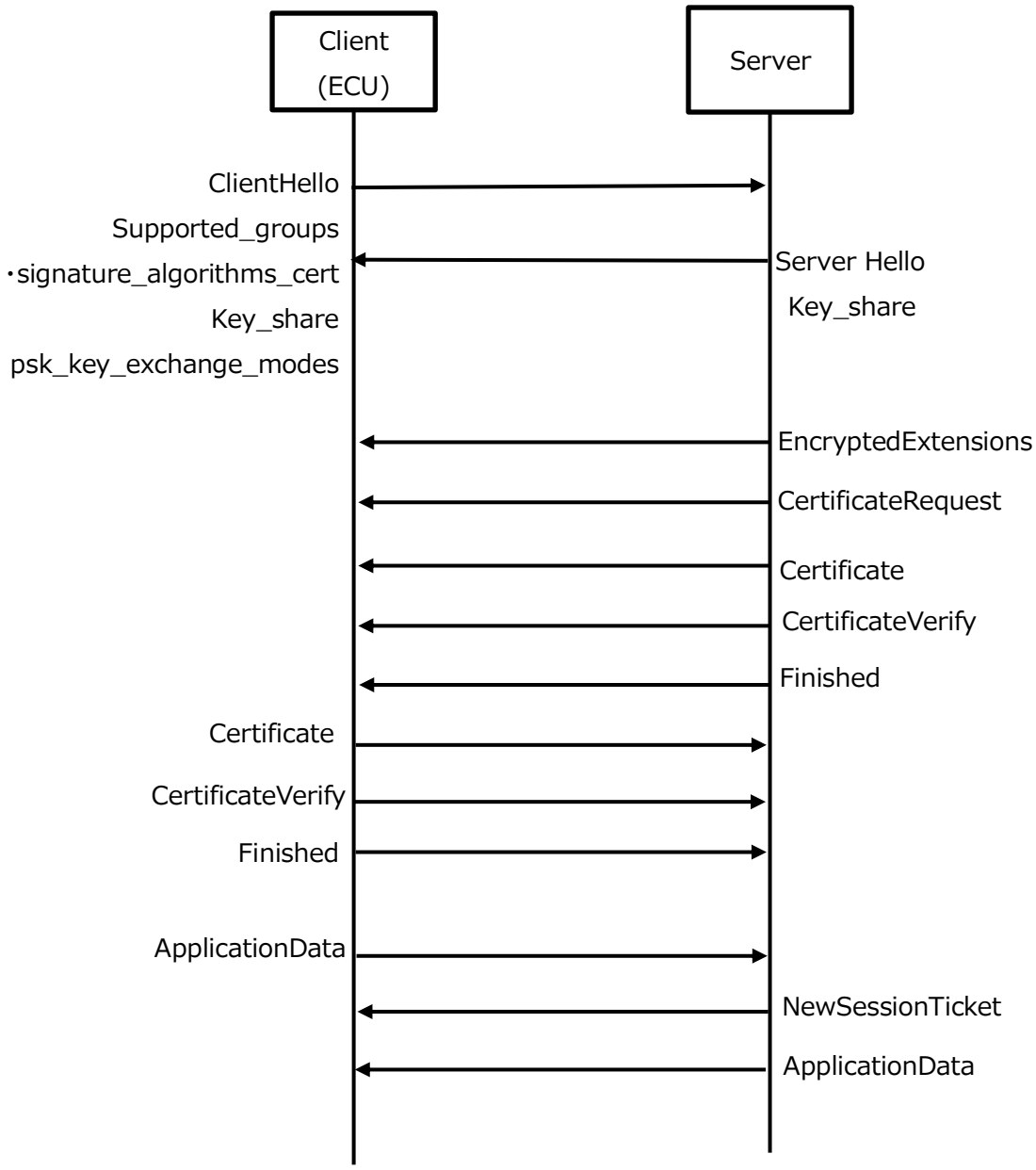


Figure 7-3 Sequence during full handshake for TLS 1.3

Table 7-2 Messages during TLS 1.3 full handshake

Sender	Message	Contents
Client	Client Hello	・Version ・Random ・Session ID ・Cipher Suites ・Compression Methods ・Extensions supported_versions: Set TLS1.3 application_layer_protocol_negotiation:Set HTTP1.1 or / and HTTP2
Server	Server Hello	・Random ・legacy_session_id_echo ・Cipher Suite ・Compression Method ・Extensions supported_versions: Set TLS1.3
	key_share	Information required for the key exchange algorithm. Extension
Client	Certificate	Client certificate to be sent to the server
	CertificateVerify	A signature and signature algorithm for proving that the private key is hold for a submitted certificate.
	Finished	HMAC value calculated from communication history
	ApplicationData	Communication data by the protocol specified in application_layer_protocol_negotiation
Server	NewSessionTicket	PSK sharing for session resumption
	ApplicationData	Communication data by the protocol specified in application_layer_protocol_negotiation

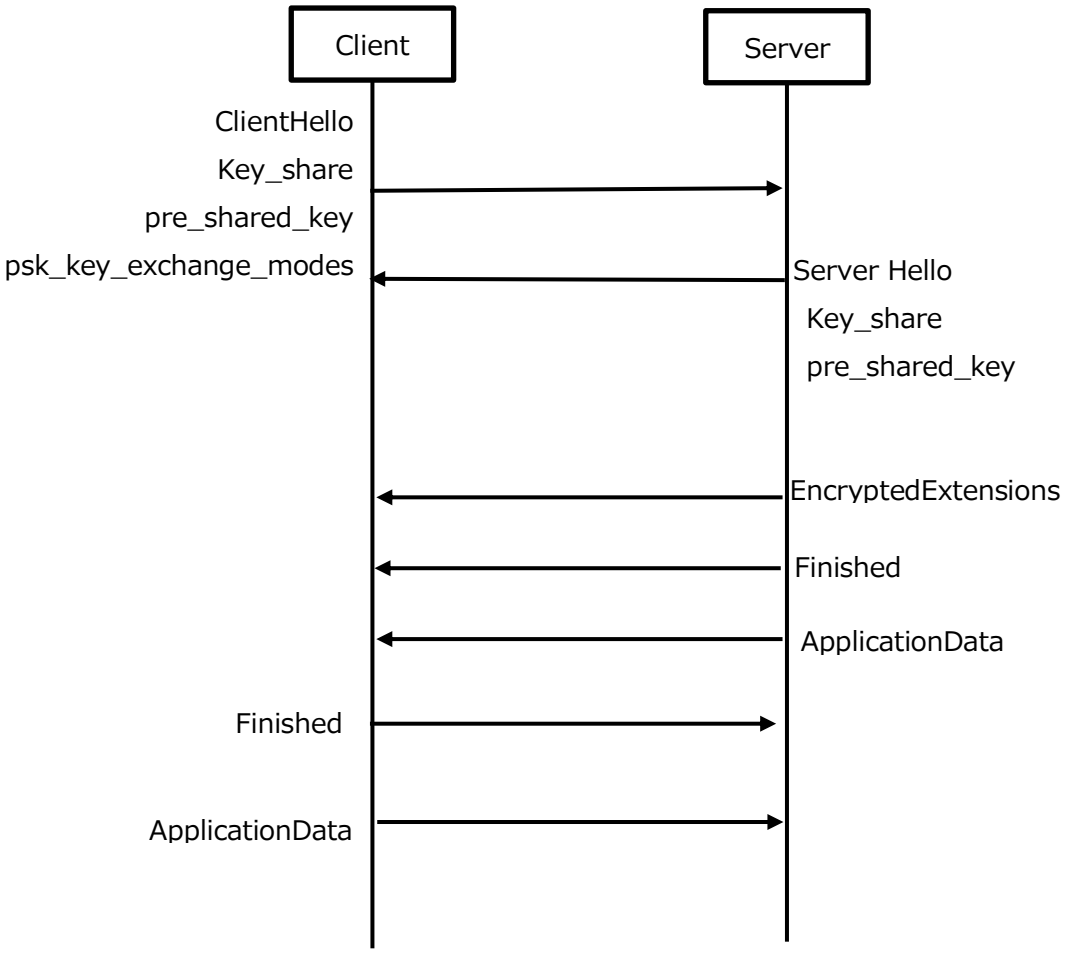


Figure 7-4 TLS1.3 Session Resumption (1-RTT) Sequence

Table 7-3 Messages for TLS1.3 Session Resumption (1-RTT)

Sender	Message	contents
Client	Client Hello	・Version ・Random ・Session ID ・Cipher Suites ・Compression Methods ・Extensions supported_versions: Set TLS1.3 application_layer_protocol_negotiation:Set HTTP1.1 or / and HTTP2
Server	Server Hello	・Random ・legacy_session_id_echo ・Cipher Suite ・Compression Method ・Extensions supported_versions: Set TLS1.3
Client	ApplicationData	Communication data by the protocol specified in application_layer_protocol_negotiation

7.1.1.7. Abnormal System Scenario

The scenarios of abnormal systems in the authentication phase of TLS and their responses are shown below.

Regarding the abnormal system scenario for server authentication at in-Car unit(ECU) side, the sequence and correspondence based on revocation check by CRL for server certificate are shown in

Figure **7-5** and Table 7-4 .

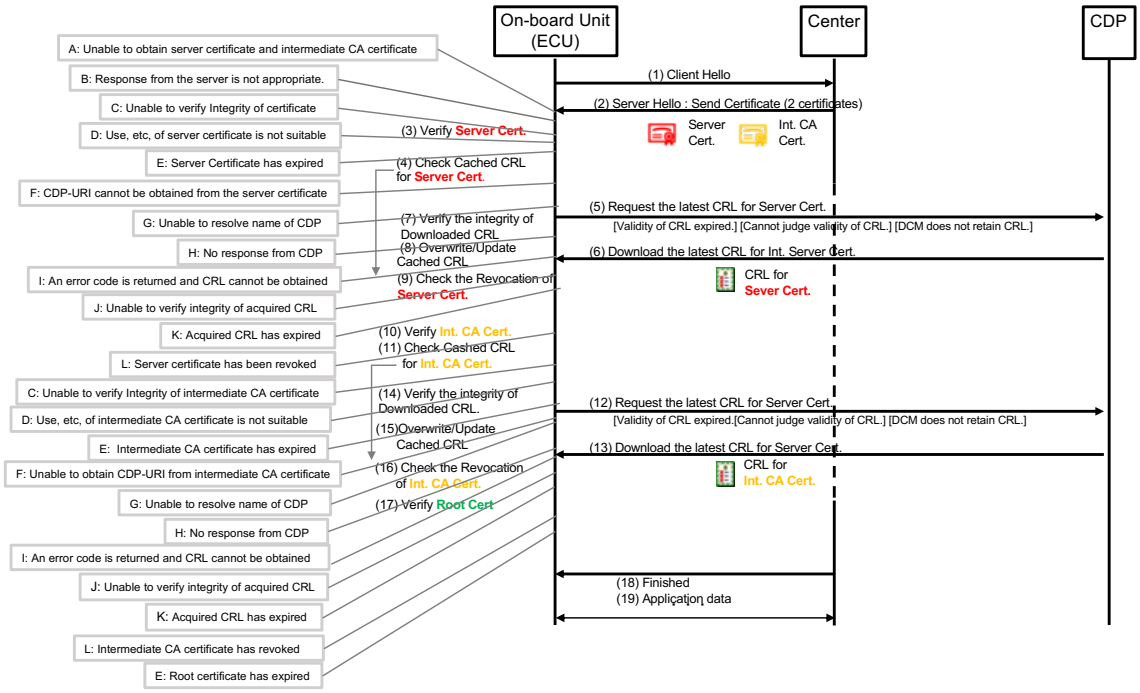


Figure 7-5 Sequence of abnormal system scenario in server authentication

Table 7-4 Correspondence to abnormal scenario in TLS authentication phase

Abnormal Scenario	Expected behavior	
A: Unable to obtain the certificate	Retry	Attempt to obtain the certificate by 1 sec interval.
	Termination condition (unable to communicate)	One-time retry failure
B: Inappropriate response from the server	Retry	None
	Termination condition (unable to communicate)	1 sec timeout in case of no response. Judged as a failure in case that the response content is incorrect. Discard acquired certificate immediately.
C: Unable to verify integrity of the certificate	Retry	Attempt to obtain the certificate by 1 sec interval. (Suspect certificate is corrupted.)
	Termination condition (unable to communicate)	When same error occurs after retry, it shall be judged as a failure. Discard acquired certificate immediately.
D: The use, etc. of certificate is not suitable	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired certificate immediately.
E: The certificate has expired	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired certificate immediately.
F: Unable to obtain CDP-URI or OCSP-URI from the certificate	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired certificate immediately.
G: Unable to resolve name of CDP or OCSP responder	Retry	Wait for DNS response by 1 sec timeout. If there are multiple DNS answers, sequentially connect to them.
	Termination condition (unable to communicate)	Terminate authentication process with one-time retry failure. (fail)

H: No response from CDP or OCSP responder	Retry	Attempt to connect by 5 sec timeout.
	Termination condition (unable to communicate)	Terminate authentication process with one-time retry failure. (fail)
I: An error code is returned and CRL/OCSP cannot be obtained	Retry	Attempt to obtain CRL by 1 sec interval.
	Termination condition (unable to communicate)	Terminate authentication process when retry failed X times. (fail)
J: Unable to verify digital signature in acquired CRL/OCSP	Retry	Attempt to obtain the certificate by 1 sec interval. (Suspect CRL is corrupted.)
	Termination condition (unable to communicate)	When same error occurs after retry, terminate authentication process. (fail) Discard acquired certificate immediately.
K: Acquired CRL/OCSP has expired	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired CRL immediately.
L: The certificate has been revoked	Retry	None
	Termination condition (unable to communicate)	The center communication shall be unacceptable by normal judgement.

Regarding the abnormal system scenario in client authentication at server side, the sequence and correspondence based on revocation confirmation of client certificate by OCSP are shown in Figure 7-6 and Table 7-5 .

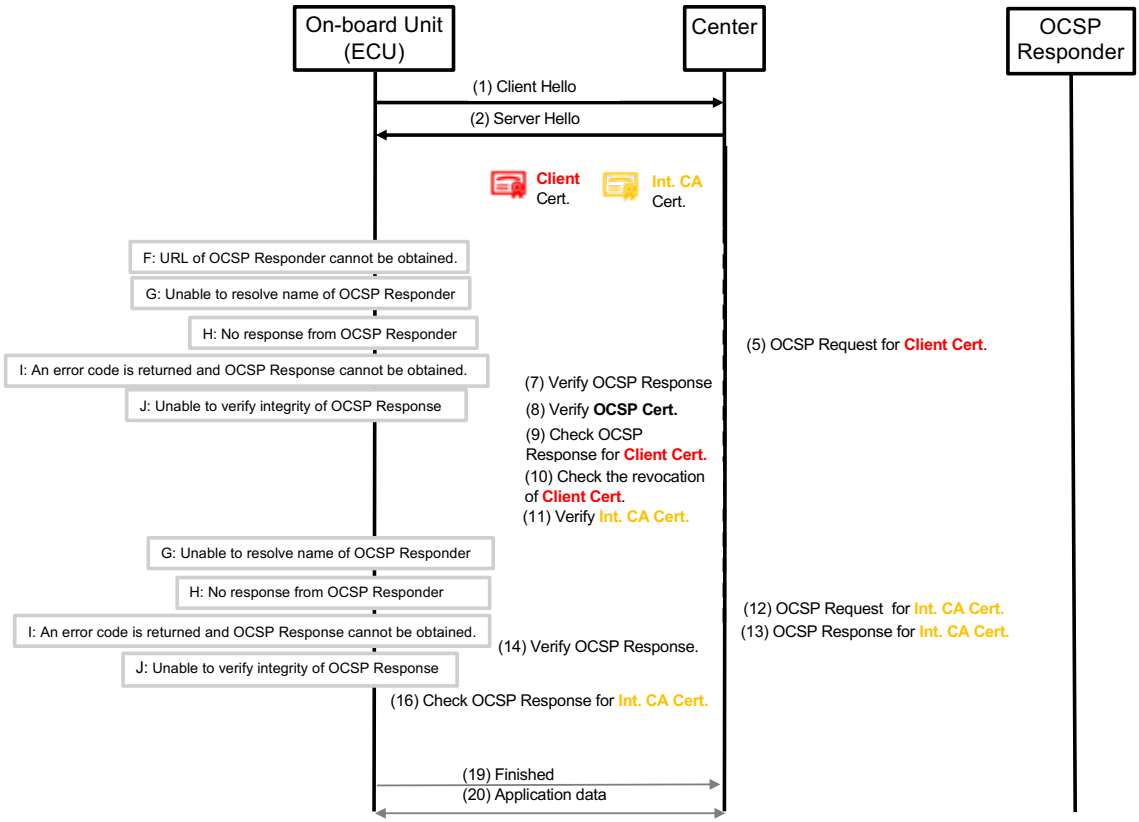


Figure 7-6 Sequence of abnormal system scenario in Client authentication

Table 7-5 Correspondence to abnormal scenario in Client authentication

Abnormal Scenario	Expected behavior	
A: Unable to obtain client certificate and intermediate CA certificate	Retry	Attempt to obtain the certificate by 1 sec interval.
	Termination condition (unable to communicate)	One-time retry failure
B: Response from the client is not suitable	Retry	None
	Termination condition (unable to communicate)	1 sec timeout in case of no response. Judged as a failure in case that the response content is incorrect. Discard acquired certificate immediately.
C: Unable to verify integrity of the client certificate	Retry	Attempt to obtain the certificate by 1 sec interval. (Suspect certificate is corrupted.)
	Termination condition (unable to communicate)	When same error occurs after retry, it shall be judged as a failure. Discard acquired certificate immediately.
D: The use, etc. of certificate is not suitable	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired certificate immediately.
E: The client certificate has expired	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired certificate immediately.
F: Unable to obtain URL of OCSP Responder	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired certificate immediately.
G: Unable to resolve name of OCSP Responder	Retry	Wait for DNS response by 1 sec timeout. If there are multiple DNS answers, sequentially connect to them.

	Termination condition (unable to communicate)	Terminate authentication process with one-time retry failure. (fail)
H: No response from OCSP Responder	Retry	Attempt to connect by 5 sec timeout.
	Termination condition (unable to communicate)	Terminate authentication process with one-time retry failure. (fail) Discard acquired certificate immediately.
I: An error code is returned and OCSP Response cannot be obtained	Retry	Attempt to obtain OCSP Response by 1 sec interval.
	Termination condition (unable to communicate)	Terminate authentication process with one-time retry failure. (fail) Discard acquired certificate immediately.
J: Unable to verify integrity of OCSP Response	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired CRL immediately.
K: OSCP Response cannot be trusted	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired CRL immediately.
L: Contents of OCSP Response is different from requirements, or incorrect	Retry	None
	Termination condition (unable to communicate)	Terminate authentication process immediately. (fail) Discard acquired CRL immediately.
M: The client certificate has been invalid or revoked	Retry	None
	Termination condition (unable to communicate)	The center communication shall be unacceptable by normal judgement.

7.2. Application Protocol

7.2.1. Application Data

For application data, the data allocation, purpose and use are shown in Table 7-6 and Table 7-7 for each protocol.

Table 7-6 HTTP data and its purpose and use

data		Purpose and use
URL	Path parameter	Indicates a unique resource on the center side.
	Query Parameters	Used for access analysis that does not affect the content provided, and for providing dynamic results that change the content provided. This parameter can be omitted.
Message	Header	Indicates application-independent information. Example. Content-Type (media type) User-Agent (product identifier) Custom header (vehicle identifier)
	Body	Used for application-specific data.

Table 7-7 MQTT data and purpose/use

Data		Use
Topics.		Indicates a unique resource on the center side. - API Name - Vehicle identifier
Message	Header	Indicates application-independent information. Example. Content-Type (media type)
	Body (payload)	Used for application-specific data.

7.2.2. HTTP/1.1

7.2.2.1. Basic Specifications

HTTP/1.1 protocol shall be supported and the communication shall conform to the following specifications/RFCs.

- RFC7230 Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing
- RFC7231 Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content

APIs should be described in accordance with the following specifications of OpenAPI v3.0 or later.
http://spec.openapis.org/oas/v3.1.0

7.2.2.2. Message Format

The message format in HTTP/1.1 communication and the scope of this document, i.e., path parameters, query parameters, request/response and common parts of headers (parts commonly used by applications (services)) are shown in Figure 7-7.

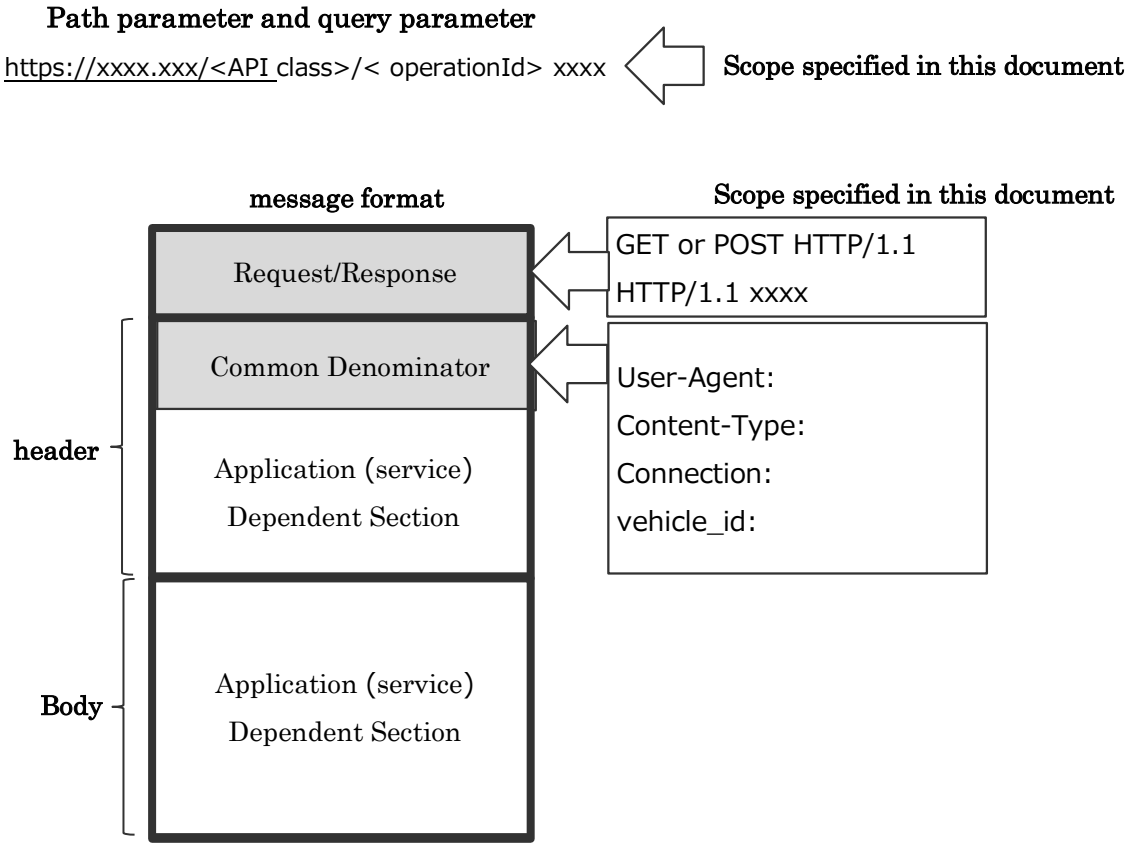


Figure 7-7 Message format in HTTP/1.1

7.2.2.3. Header Definition

The following media types are supported. Media types must be case-insensitive as case-insensitive strings according to RFC6838, and must be specified in "Content-Type" as one of the header fields of an HTTP request.

- application/octet-stream (binary format)
- application/x-protobuf (Protocol Buffers format)
- application/json (JSON format)
- text/plain (text format)

User-Agent shall be used to assign product identifiers in accordance with RFC7231, and shall not be used for application-specific information.
The product identifier shall consist of the identification information of the product (ECU) (information management key), the version of the product, and sub-product information (software information and version), and shall always include the information management key.

Other conditions are shown below.

- IMEI, GUID, TCON, etc. shall not be embeded in User-Agent header fields.
- The vehicle identifier shall be assigned as a custom header.
- Do not use proprietary header fields such as the following
  - Fields for Hmac authentication such as X-D-Authenticate, X-Y-Authentication, etc.

7.2.2.4. Response Status Code

For each HTTP response status code, the necessity of retry is specified below.
Retries due to errors in client-side specification shall not be performed.
For client-side errors (Code:400, 404, 408, 409) that depend on server-side status, the specification is to roll back (or discard) the status when the error code is issued on the server side, and no retry shall be performed on the client side.
Besides, the reason-phrase is a textual description of the status-code, and the client should not use the reason-phrase to determine the result.

Table 7-8 HTTP/1.1 response status codes

Class	Code	Reason-Phrase	Retry
Informational	100	Continue	No
	101	Switching Protocols	
Successful	200	OK.	
	201	Created	
	202	Accepted	
	203	Non-Authoritative Information	
	204	No Content	
	205	Reset Content	
	206	Partial Content	
Redirection	300	Multiple Choices	
	301	Moved Permanently	
	302	Found	
	303	See Other	
	304	Not Modified	
	305	Use Proxy	
	307	Temporary Redirect	
Client Error	401	Unauthorized	
	402	Payment Required	
	403	Forbidden	
	405	Method Not Allowed	
	406	Not Acceptable	
	407	Proxy Authentication Required	
	410	Gone	
	411	Length Required	
	412	Precondition Failed	
	413	Payload Too Large	
	414	URI Too Long	
	415	Unsupported Media Type	
	416	Range Not Satisfiable	
	417	Expectation Failed	
	426	Upgrade Required	
	400	Bad Request	

Class	Code	Reason-Phrase	Retry
	404	Not Found	
	408	Request Timeout	
	409	Conflict	
Server Error	500	Internal Server Error	Yes
	501	Not Implemented	
	502	Bad Gateway	
	503	Service Unavailable	
	504	Gateway Timeout	
	505	HTTP Version Not Supported	
	567	No communication until next IGON	
	568	No communication for 10 minutes	

7.2.2.5. Timeout Specification

In principle, the timeout period for interruption when there is no center response after sending a request is 60 seconds in case of out of coverage situation etc.
However, uploading or downloading of large-volume data is out of this scope.
The communication timeout period shall not include the line connection time.
In order to prevent the load concentration on the center, the retransmission process after timeout shall be under control to extend the retransmission interval.

7.2.2.6. Retry Specification

The pattern when retries are required is as follows.
Note that requirements by regulations such as NEV regulation are outside the scope.

Use case	Real-time response	Number of retries	Retry interval
Data upload (in-Car unit → Center)	Required	2	3sec
	Not required	2	1min
Data download (Center → in-Car unit)	Required	N/A	N/A

7.2.2.7. Session Management

・ Session establishment

Establish an http / 1.1 connection when all of the following conditions are applied.

- WAN side IP address shall be assigned to in-Car unit communication termination in the possible status for the data communication.
- Obtaining VIN
- ADF upload has been completed.

・ Session maintenance

In case of using HTTP/1.1, basically session shall not be maintained. (Cookie shall not be used.)

7.2.2.8. Basic Sequence

The basic sequence of GET and POST in HTTP/1.1 is shown in Figure 7-8 and Figure 7-9.

7.2.2.8.2. GET

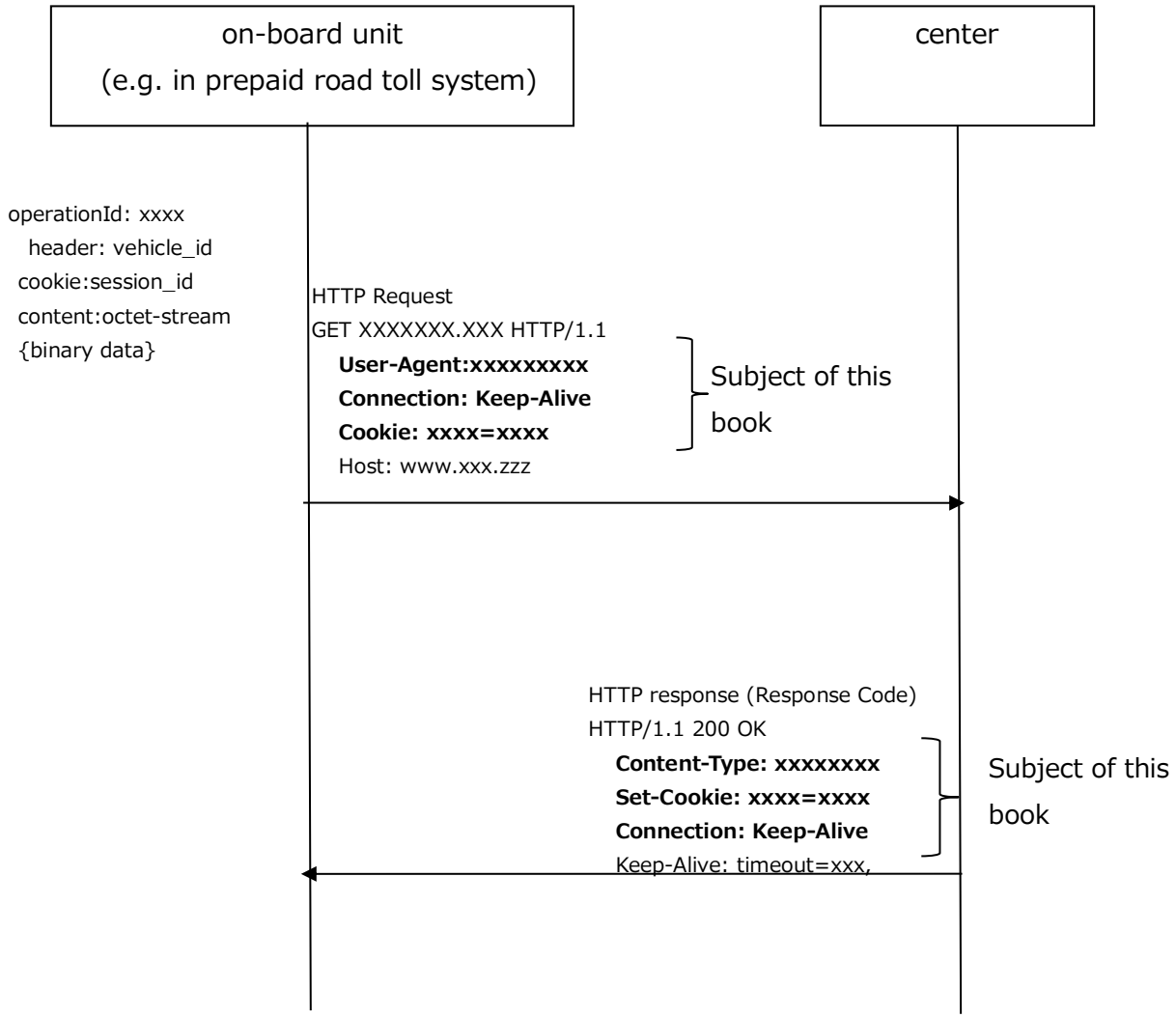


Figure 7-8 Basic Sequence of HTTP/1.1 (GET)

7.2.2.8.3. POST

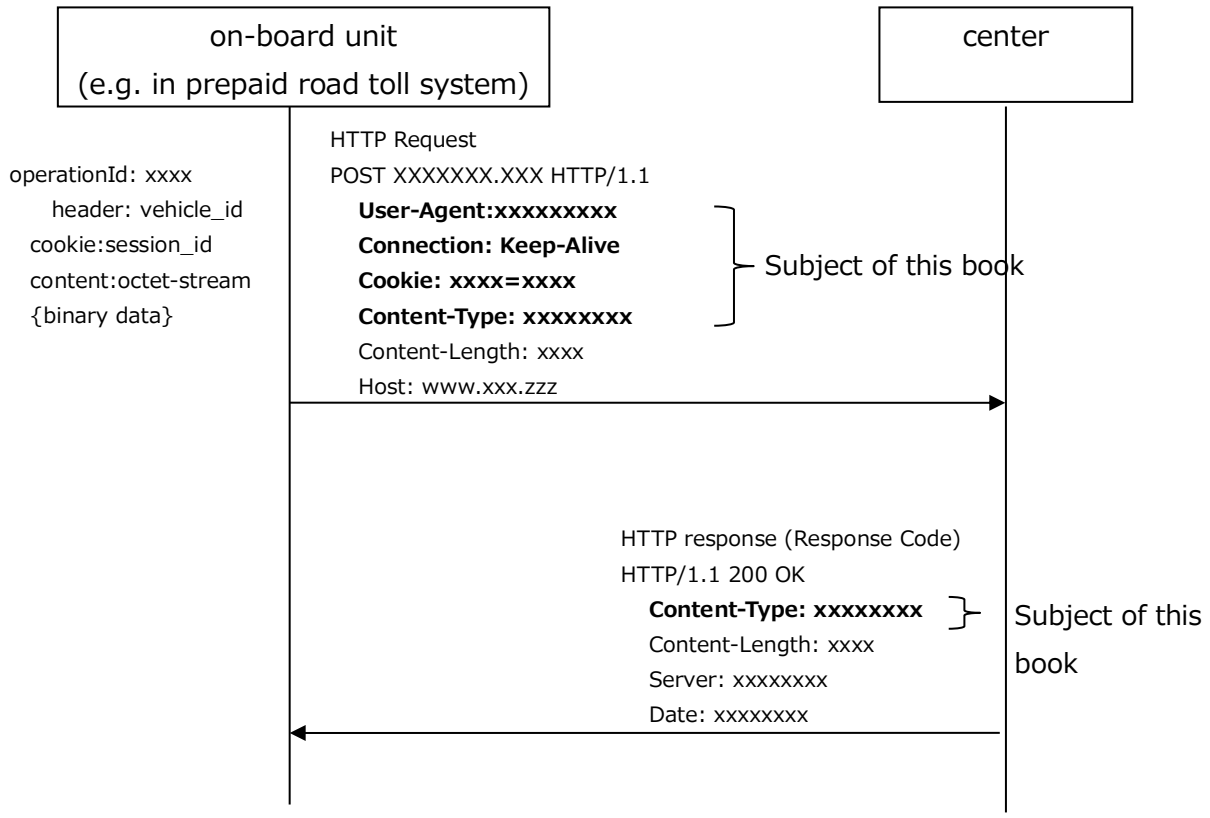


Figure 7-9 Basic Sequence of HTTP/1.1 (POST)

7.2.2.9. API Specification

Define API specifications for each use case.
The yaml file based on the specification to be defined separately shall be the correct specification, and this document shall be referred to as a reference. Do not modify or reimplement the yaml files mentioned in this document.

7.2.2.9.1. Data Uploading

The API specification for data uploading is shown in Table 7-9.

Table 7-9 API Specification for Data Uploading (OpenAPI)

Object/Field				Description
openapi	openapi			OpenAPI Version
info	title			API Title
	version			API version
	description			API Description
servers	url			URL of the server
paths	/{ApplicationId}			API path template (ApplicationId:Name of application)
	post	operationId		API Name
		parameters		Parameters
		in: header	name: vehicle_id	Vehicle identifier (see 7.3)
		in: cookie	name: session_id	Set session ID.
		requestBody		Data (Body part)
		content		Media Type (See 7.2.2.3)
			schema	schema *Content of the schema depends on application (service).
		responses		

Object/Field						Description
			200			response code Error codes other than 200 are application (service) dependent.
				content		Media Type (See 7.2.2.3)
					schema	schema *Content of the schema depends on application (service).

< . yaml file: data upload >

```
openapi: 3.0.3
info:
  title: api-doc-can-data
  description: API Documentation for CAN data upload
  version: 1.0.2
servers:
- url: http://example.com/
paths:
  /files/{filename}:
    post:
      operationId: uploadCANDataFile
      parameters:
        - in: header
          name: vehicle_id
          schema:
            type: string
            minLength: 15
            maxLength: 15
        - in: path
          name: filename
          description: Can data file name
          required: true
          schema:
            type: string
        - in: cookie
          name: session_id
          schema:
            type: string
      requestBody:
        content:
          application/octet-stream:
            schema:
              type: string
              format: binary
      responses:
        '200':
          description: Successfully file uploaded 200 response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Result'
        '400':
          description: Invalid request 400 response
          content: {}
        '500':
          description: Internal error 500 response
          content: {}


components:
  schemas:
    Result:
      type: object
      properties:
        status:
          type: string
          enum:
            - SUCCESS
            - ERROR
        messages:
          type: array
          items:
            type: string
            example: Request success
```

7.2.2.9.2. Push from the Center

REST based on HTTP/1.1 is not suitable for pushing from the center. (See 6.1.2) Therefore, push from the center is out of scope of REST application.

7.2.2.9.3. API Call from Vehicle to Center

The API specifications for API calls from vehicle to the center are shown in Table 7-10.

Table 7-10 API specification for API call from vehicle to center (OpenAPI)

Object/Field				Description
openapi	openapi			OpenAPI Version
info	title			API Title
	version			API version
	description			API Description
servers	url			URL of the server
paths	/{ApplicationId}/{number}			API Path Template (ApplicationId: Application name)
	post	operationId		API Name
		parameters		Parameters
		in: header	name: vehicle_id	
		in: cookie	name: session_id	Set the session ID.
		in: path	name: number	API Number
		requestBody		Data (Body part)
		content		Media Type (See 7.2.2.3)
			schema	schema *Content of the schema depends on application (service).
		responses		

Object/Field						Description
			200			response code *Error codes other than 200 are application (service) dependent.
				content		Media Type (See 7.2.2.3)
					schema	schema *Content of the schema depends on application (service).

<. yaml file: API call from vehicle to center>

```
openapi: 3.0.3
info:
  title: api-doc-call-api
  description: API Documentation for API Call
  version: 1.0.2
servers:
- url: http://example.com/
paths:
  /api/{number}:
    post:
      operationId: callApi
      parameters:
        - in: header
          name: vehicle_id
          schema:
            type: string
            minLength: 15
            maxLength: 15
        - in: path
          name: number
          description: Number of API
          required: true
          schema:
            type: string
        - in: cookie
          name: session_id
          schema:
            type: string
      requestBody:
        content:
          application/octet-stream:
            schema:
              type: string
              format: binary
      responses:
        '200':
          description: Successfully file uploaded 200 response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Result'
        '400':
          description: Invalid request 400 response
          content: {}
        '500':
          description: Internal error 500 response
          content: {}


components:
  schemas:
    Result:
      type: object
      properties:
        status:
          type: string
          enum:
            - SUCCESS
            - ERROR
        messages:
          type: array
          items:
            type: string
            example: Request success
```

7.2.2.9.4. Data Download

The API specification for data download is shown in Table 7-11.

Table 7-11 API specification for data download (OpenAPI)

Object	Field/Type						Description
openapi	openapi						OpenAPI Version
info	title						API Title
	version						API version
	description						API Description
servers	url						URL of the server
paths	/{ApplicationId}						API Path Template (ApplicationId: Application name)
	get	operationId					API Name
		parameters					Parameters
			in: header	name: vehicle_id			Vehicle identifier (see 7.3)
			in: cookie	name: session_id			Set session ID.
		responses					
			200				response code *Error codes other than 200 are application (service) dependent.
				content			Media Type (See 7.2.2.3)
					schema		schema *Content of the schema depends on application (service).

<. yaml file: data download>

```
openapi: 3.0.3
info:
  title: api-doc-download
  description: API Documentation for Data Download
  version: 1.0.2
servers:
- url: http://example.com/
paths:
  /download/{filename}:
    get:
      operationId: download
      parameters:
        - in: header
          name: vehicle_id
          schema:
           type: string
           minLength: 15
           maxLength: 15
        - in: path
          name: filename
          description: Download file name
          required: true
          schema:
            type: string
        - in: cookie
          name: session_id
          schema:
            type: string
      responses:
        '200':
          description: Successfully file uploaded 200 response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Result'
        '400':
          description: Invalid request 400 response
          content: {}
        '500':
          description: Internal error 500 response
          content: {}


components:
  schemas:
    Result:
      type: object
      properties:
        status:
          type: string
          enum:
            - SUCCESS
            - ERROR
        messages:
          type: array
          items:
            type: string
            example: Request success
```

7.2.3. HTTP/2

7.2.3.1. Basic Specifications

HTTP/2 protocol shall be supported and the communication shall conform to the following specifications/RFCs.

- RFC7540 Hypertext Transfer Protocol Version 2 (HTTP/2)
- RFC7541 HPACK: Header Compression for HTTP/2

APIs should be described in accordance with the following specifications of OpenAPI v3.0 or later.
http://spec.openapis.org/oas/v3.1.0

7.2.3.2. Message Format

Figure 7-10 shows the message format in HTTP/2 and the common parts of the request/response and header (parts commonly used by applications (services)) that are the scope of this document.

Path parameters and query parameters should be used based on the following concept.

- Path parameter
  Indicates a unique resource on the center side.
- Query Parameters
  Used for access analysis that does not affect the content provided, and for providing dynamic results that change the content provided. May be omitted.

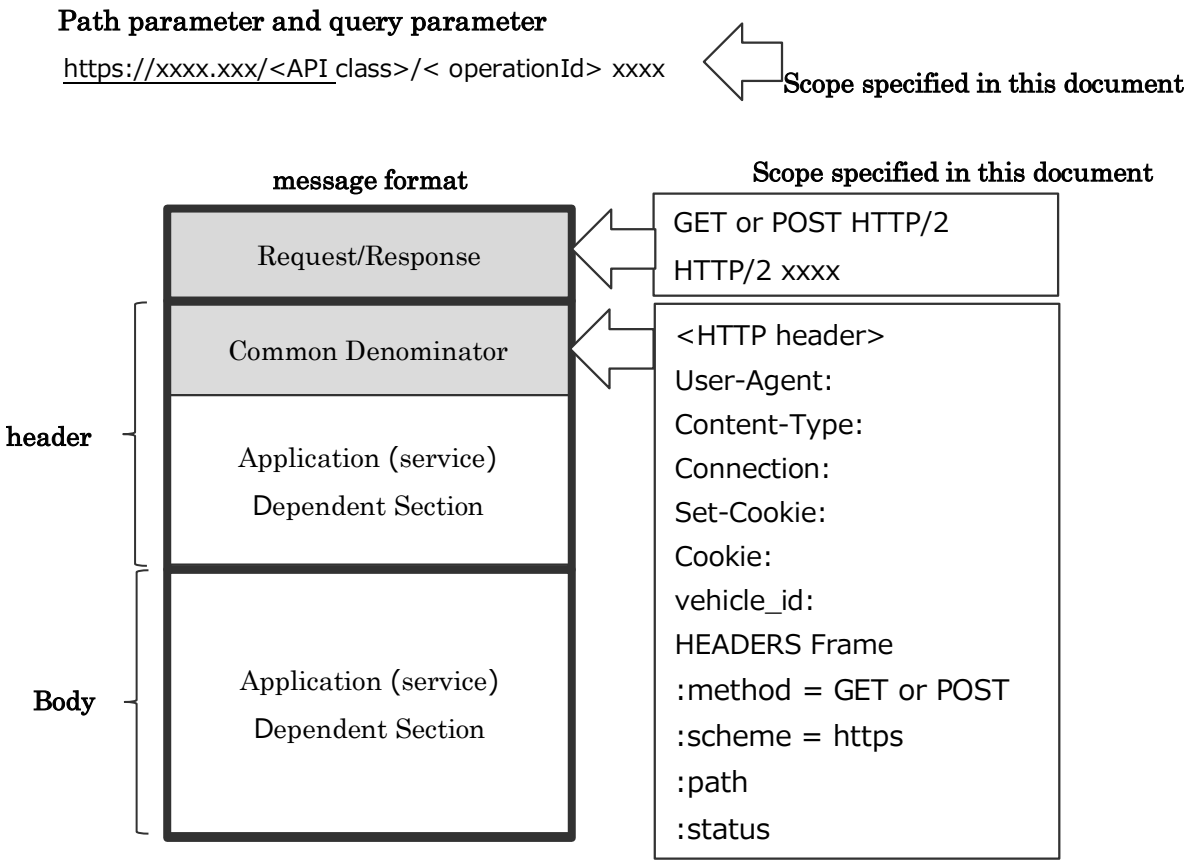


Figure 7-10 Message format in HTTP/2

7.2.3.3. Header Definition

Request header and response header of HTTP/2 shall conform to the specifications (RFC) shown in 7.2.2.3
Perform header compression based on RFC7541.
In case that stateful connection is required based on session management by the center, use Set-Cookie and Cookie header.
Other headers shall conform to HTTP/1.1 specification (7.2.2.3).

7.2.3.4. Response Status Code

Conform to HTTP/1.1 specification (7.2.2.4).

7.2.3.5. Timeout Specification

Conform to HTTP/1.1 specification (7.2.2.5)

7.2.3.6. Retry Specification

Conform to HTTP/1.1 specification (7.2.2.6)

7.2.3.7. Session Management

・ Session establishment

Establish an http/2 connection when all of the following conditions are applied.

- WAN side IP address shall be assigned to in-Car unit communication termination in the possible status for the data communication.
- Obtaining VIN
- ADF upload has been completed.

・ Session maintenance

In case that stateful connection is required based on session management by the center with ALB configured environment, use sticky sessions (Cookie).

7.2.3.8. Basic Sequence

The basic sequence of GET and POST of HTTP/2 is shown in Figure 7-11 and Figure 7-12.

7.2.3.8.1. GET

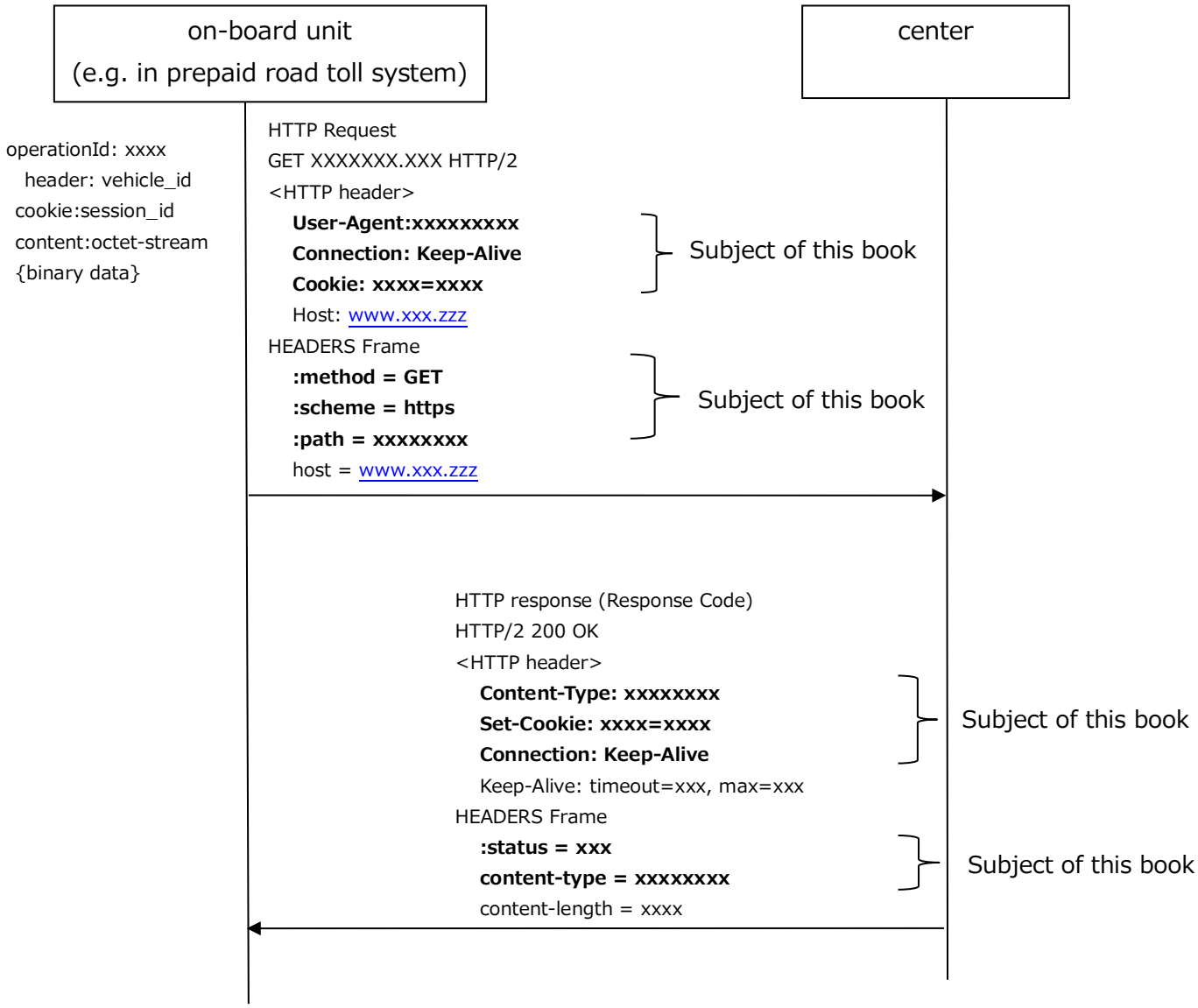


Figure 7-11 Basic Sequence of HTTP/2 (GET)

7.2.3.8.2. POST

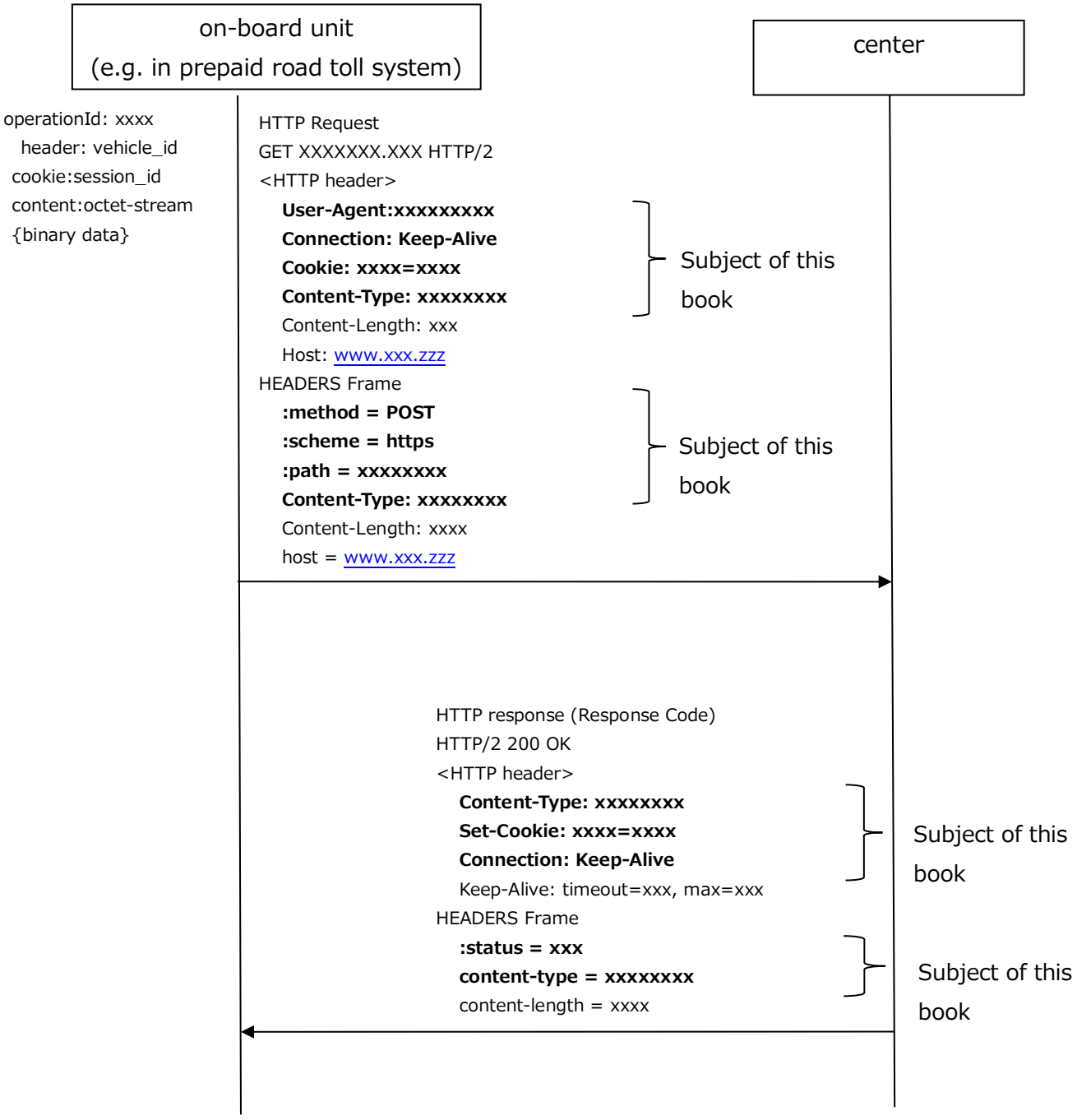


Figure 7-12 Basic Sequence of HTTP/2 (POST)

7.2.3.9. API Specification

Define API specifications for each use case.
The yaml file based on the specification to be defined separately shall be the correct specification, and this document shall be referred to as a reference. Do not modify or reimplement the yaml files mentioned in this document.

7.2.3.9.1. Data Uploading

Conform to HTTP/1.1 specification (7.2.2.9.1)

7.2.3.9.2. Push from the Center

The API specification for push from the center is shown in Table 7-12.

Table 7-12 API specification for push from the center (OpenAPI)

Object/Field				Description
openapi	openapi			OpenAPI Version
info	title			API Title
	version			API version
	description			API Description
servers	url			URL of the server
paths	/{ApplicationId}/{app_id}/{sub_id}			API Path Template (ApplicationId: Application name)
	post	operationId		API Name
		parameters		Parameters
		in: header	name: vehicle_id	Vehicle identifier (see 7.3)
		in: cookie	name: session_id	Set session ID.
		in: path	name: app_id	Application ID of the push destination
		in: path	name: sub_id	Application sub-ID of the push destination
		requestBody		Data (Body part)

Object/Field	Description
content	Media Type (See 7.2.2.3
schema	schema *Content of the schema depends on application (service).
responses	
200	response code *Error codes other than 200 are application (service) dependent.
content	Media Type (See 7.2.2.3
schema	schema *Content of the schema depends on application (service).

<. yaml file: push from center >

```
openapi: 3.0.3
info:
  title: api-doc-call-api
  description: API Documentation for API Call
  version: 1.0.2
servers:
- url: http://example.com/
paths:
  /api/{number}:
    post:
      operationId: callApi
      parameters:
        - in: header
          name: vehicle_id
          schema:
           type: string
           minLength: 15
           maxLength: 15
        - in: path
          name: number
          description: Number of API
          required: true
          schema:
            type: string
        - in: cookie
          name: session_id
          schema:
            type: string
      requestBody:
        content:
          application/octet-stream:
            schema:
              type: string
              format: binary
      responses:
        '200':
          description: Successfully file uploaded 200 response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Result'
        '400':
          description: Invalid request 400 response
          content: {}
        '500':
          description: Internal error 500 response
          content: {}


components:
  schemas:
    Result:
      type: object
      properties:
        status:
          type: string
          enum:
            - SUCCESS
            - ERROR
        messages:
          type: array
          items:
            type: string
            example: Request success
```

7.2.3.9.3. API Call from Vehicle to Center

Conform to HTTP/1.1 specification (7.2.2.9.2)

7.2.3.9.4. Data Download

Conform to HTTP/1.1 specification (7.2.2.9.4)

7.2.4. gRPC

7.2.4.1. Basic Specifications

The gRPC version to be supported is v1.20 or later.

- Follow the wire protocol specification for gRPC over HTTP2. (https://github.com/grpc/grpc/blob/master/doc/PROTOCOL-HTTP2.md)
- Assume the use of Protocol Buffers 3 as IDL

The communication shall conform to the following specifications/RFCs.

- RFC7540 Hypertext Transfer Protocol Version 2 (HTTP/2)
- RFC7541 HPACK: Header Compression for HTTP/2

As for the data size such as video by gRPC, when sending binary files exceeding MB, split the suitable size by Client Streaming at in-Car unit side, or take response including setting change, etc. at the server side.
Also, in the first case (taking response at in-Car unit side), the preferred split size should be about 1MB.

Care about data size and such for data uploading with multiple stream because the bandwidth will possibly become insufficient if another communication occurs while bulk data communication such as video is performed.

7.2.4.2. Message Format

The message format in gRPC and the scope of this document, request/response, common parts of the header, and common parts of the body, are shown in Figure

7-13.

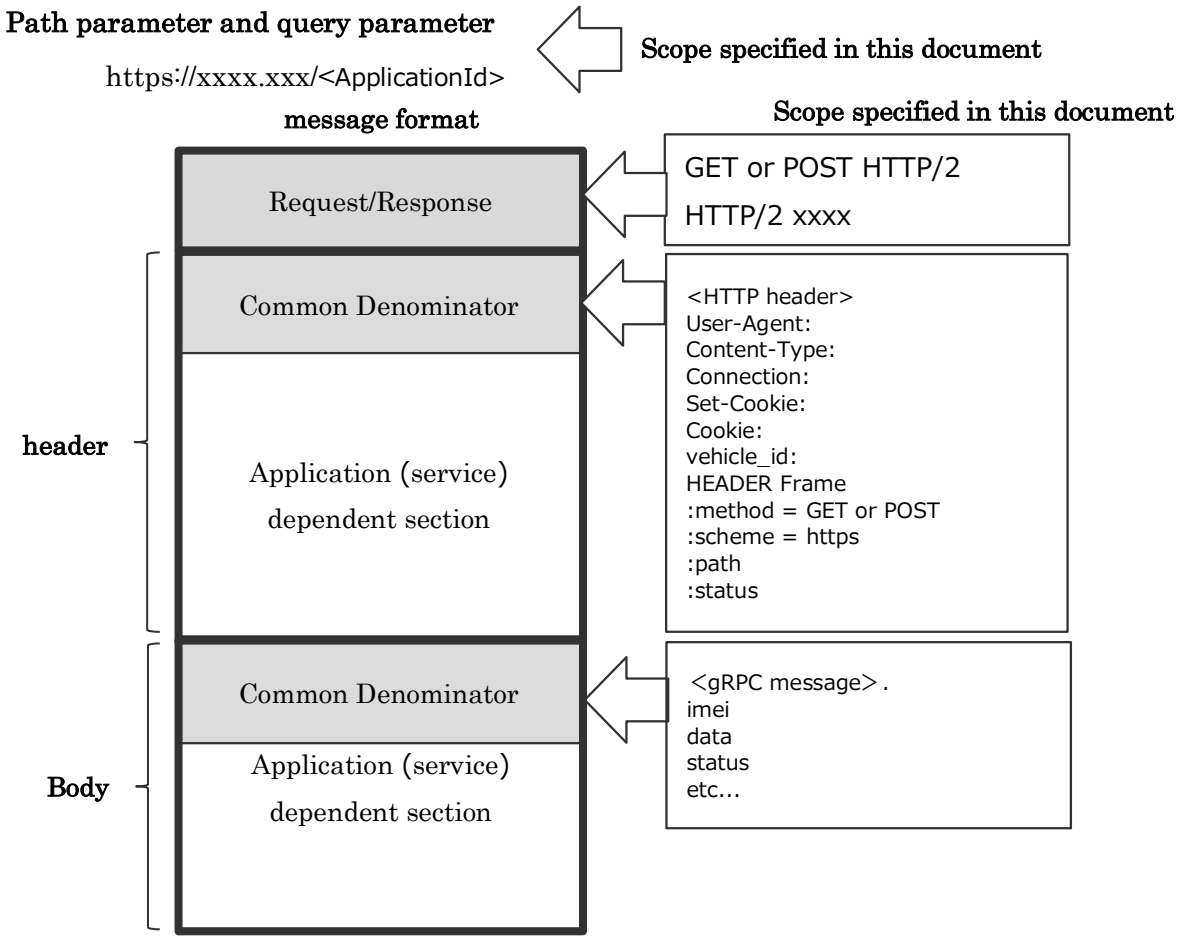


Figure 7-13 Message format in gRPC

7.2.4.3. Header Definition

For content-type, set the following when the media type needs to be specified.

- application/grpc+proto

Complying with RFC6838, upper case and lower case shall not be distinguished as case-insensitive character strings.

Product identifier and gPRC recommended library name shall be set to user-agent in space-separated format.

e.g.) user-agent: <product identifier> grpc-go/1.32.0-dev

7.2.4.4. API Specification

Define API specifications for each use case.
Note that the proto file based on the specification to be defined separately is the correct specification, and this document should be referred to as a reference.
Also, do not modify or reimplement the proto files mentioned in this document.

Vehicle identifier (vehicle_id) shall not be set in proto file and shall be set as Meta-Data by in-Car unit application or server side application

7.2.4.4.1. Data Uploading

The API specifications for data upload are shown in Table 7-13 and Table 7-14.

Table 7-13 API specification for data upload (gRPC Request)

<Request>

Type	Field Name	Index	Description
Required string	vehicle_id	1	Vehicle identifier (see 7.3)
Required bytes	data	2	Data

Table 7-14 API Specification for Data Uploading (gRPC Response)

<Response>

Type	Field Name	Index	Description
Required message	Status	-	Indicates the response from the server for the data uploading.
required enum	statuscode	-	OK (0): Normal RETRYAFTER (1): Used when the server wants the application to wait for a certain amount of time. See the description of retry_after_sec for details. BUSY (2): Used when the server is unable to respond immediately except for RETRYAFTER.
required statuscode	status	1	Indicates the above statuscode.
Required fixed32	retry_after_sec	2	Valid only when "status" is "RETRYAFTER". (seconds) After receiving this Response, the application will immediately close the connection, wait the number of seconds indicated by retry_after_sec, and then Request the server again. This is used when the server wants to stop providing the service temporarily for some reason.

<.proto file: data upload>

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "grpc.can";
option java_outer_classname = "CanProto";
option objc_class_prefix = "CAN";

service CanService {
  rpc SendCans(stream CanRequest) returns (CanResponse);
}

message CanRequest {
  string vehicle_id    = 1;
  bytes  data   = 2;
}

message CanResponse {
  // message for server status response
  message Status {
    enum statuscodes {
      OK = 0;
      RETRYAFTER = 1;
      BUSY = 2;
    }
    statuscodes status = 1;
    fixed32 retry_after_sec = 2;
  }

  Status status = 1;
}
```

7.2.4.4.2. Push from the center

The API specification for push from the center is shown in Table 7-15.

<Request
None

Table 7-15 API Specification for Push from Center (gRPC Response)

<Response

Type	Field Name	Index	Description
required message	PushServerStatus	-	Indicates the status of the server.
required enum	statuscode	-	OK (0): Normal RETRYAFTER (1): Used when the server wants the application to wait for a certain amount of time. See the description of retry_after_sec for details. BUSY (2): Used when the server is unable to respond immediately except for RETRYAFTER.
required statuscode	srv_response_code	1	Indicates the above statuscode.
required fixed32	retry_after_sec	2	Valid only when "status" is "RETRYAFTER". (seconds) After receiving this Response, the application will immediately close the connection, wait the number of seconds indicated by retry_after_sec, and then Request the server again. This is used when the server wants to stop providing the service temporarily for some reason.
required message	Notification	-	A message indicating the contents of the push notification.
required fixed32	app_id	1	ID indicating the application to be pushed.

<Response

Type	Field Name	Index	Description
required fixed32	sub_id	2	Sub-ID to be used by the push destination application.
required string	expires	3	Expiration date of the notified push. The handling of the expiration date depends on the application.
required bytes	app_data	4	Arbitrary data for each application. It shall be less than 2048 bytes.

< Proto file >

```
syntax = "proto3";
package service.push;
service PushService {
    rpc GetPushNotifications(Request) returns (stream Response);
}
message Request {}
message Response {
    // message for server status response
    message PushServerStatus {
        enum statuscodes {
            OK = 0;
            RETRYAFTER = 1;
            BUSY = 2;
        }
        statuscodes srv_response_code = 1;
        fixed32 retry_after_sec = 2;
    }
    // message for push notification
    message PushNotification {
        fixed32 app_id = 1;
        fixed32 sub_id = 2;
        string  expires = 3;
        bytes app_data = 4;
    }
    PushServerStatus msg_status = 1;
    PushNotification msg_notification = 2;
}
```

7.2.4.4.3. API Call from Vehicle to Center

The API specifications for API calls from the vehicle to the center are shown in Table 7-16 and Table 7-17

Table 7-16 API specification of API call from vehicle to center (gRPC Request)

<Request>

Type	Field Name	Index	Description
required string	vehicle_id	1	Vehicle identifier (see 7.3)
required fixed32	api_id	2	API ID
required bytes	data	3	Data (data at the time of API call)

Table 7-17 API Specification of API Call from Vehicle to Center (gRPC Response)

<Response>

Type	Field Name	Index	Description
required message	Status	-	Indicates the response from the server to the request.
required enum	statuscode	-	OK (0): Normal RETRYAFTER (1): Used when the server wants the application to wait for a certain amount of time. Follow the description of retry_after_sec in details. BUSY (2): Used when the server is unable to respond immediately except for RETRYAFTER.
required statuscode	status	1	Indicates the above statuscode.
required fixed32	retry_after_sec	2	Valid only when "status" is "RETRYAFTER". (seconds) After receiving this response, the application will immediately close the connection, wait the number of seconds indicated by retry_after_sec, and then Request the server again. This is used when the server wants to stop providing the service temporarily for some reason.

<Response>

required bytes	data	3	Any data in the API response. It shall be less than 2048 bytes.

< Proto file>

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "grpc.api";
option java_outer_classname = "ApiProto";
option objc_class_prefix = "API";

service ApiService {
  rpc CallApi(ApiRequest) returns (ApiResponse);
}

message ApiRequest {
  string vehicle_id = 1;
  fixed32  api_id = 2;
  bytes   data   = 3;

}

message ApiResponse {
  // message for server status response
  message Status {
    enum statuscodes {
      OK = 0;
      RETRYAFTER = 1;
      BUSY = 2;
    }
    statuscodes status = 1;
    fixed32 retry_after_sec = 2;
    bytes data= 3;
  }

  Status status = 1;
```

7.2.4.4.4. Data Download

The API specifications for data download are shown in Table 7-18 and Table 7-19

Table 7-18 API Specification for Data Download (gRPC Request)

<Request>

Type	Field Name	Index	Description
required string	vehicle_id	1	Vehicle identifier (see 7.3)
required bytes	data	2	Data (data required for download)

Table 7-19 API Specification for Data Download (gRPC Response)

<Response>

Type	Field Name	Index	Description
Required message	Status	-	Indicates the response from the server to the request.
required enum	statuscode	-	OK (0): Normal RETRYAFTER (1): Used when the server wants the application to wait for a certain amount of time. See the description of retry_after_sec for details. BUSY (2): Used when the server is unable to respond immediately except for RETRYAFTER.
required statuscode	status	1	Indicates the above statuscode.
Required fixed32	retry_after_sec	2	Valid only when "status" is "RETRYAFTER". (seconds) After receiving this Response, the application will immediately close the connection, wait the number of seconds indicated by retry_after_sec, and then Request the server again. This is used when the server wants to stop providing the service temporarily for some reason.
required	data	3	Download Data

<Response>

bytes			

< Proto file >

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "grpc.dl";
option java_outer_classname = "DlProto";
option objc_class_prefix = “DL";

service DlService {
  rpc Download(DlRequest) returns (stream DlResponse);
}

message DlRequest {
  string vehicle_id = 1;
  bytes  data   = 2;
}

message DlResponse {
  // message for server status response
  message Status {
    enum statuscodes {
      OK = 0;
      RETRYAFTER = 1;
      BUSY = 2;
    }
    statuscodes status = 1;
    fixed32 retry_after_sec = 2;
    bytes data= 3;
  }

  Status status = 1;
}
```

7.2.4.5. Response status code

The response status code of gRPC and the necessity of retry are shown in Table 7-20.
If the error is caused by a client-side specification, no retry is performed.

Table 7-20 Response Status Code and Retry Required

Code	Mean	HTTP Mapping	Retry
0	OK	200 OK	No
1	CANCELLED	499 Client Closed Request	No
2	UNKNOWN	500 Internal Server Error	Yes
3	INVALID_ARGUMENT	400 Bad Request	No
4	DEADLINE_EXCEEDE	504 Gateway Timeout	Yes
5	NOT_FOUND	404 Not Found	No
6	ALREADY_EXISTS	409 Conflict	No
7	PERMISSION_DENIED	403 Forbidden	No
16	UNAUTHENTICATED	401 Unauthorized	No
8	RESOURCE_EXHAUSTED	429 Too Many Requests	No
9	FAILED_PRECONDITION	400 Bad Request	No
10	ABORTED	409 Conflict	No
11	OUT_OF_RANGE	400 Bad Request	No

Code	Mean	HTTP Mapping	Retry
12	UNIMPLEMENTED	501 Not Implemented	Yes
13	INTERNAL	500 Internal Server Error	Yes
14	UNAVAILABLE	503 Service Unavailable	Yes
15	DATA_LOSS	500 Internal Server Error	Yes

7.2.4.6. Session Management

・ Session establishment

Establish gRPC connection when all of the following conditions are applied.

- WAN side IP address shall be assigned to in-Car unit communication termination in the possible status for the data communication.
- Obtaining VIN
- ADF upload has been completed.

In gRPC, the same session (channel) can be used by multiple applications, and a failure in one application will not affect other applications in the same channel.

Therefore, for the purpose of efficient communication, the application shall communicate in stream units for multiple communications in one connection.

However, the sharing of channels between applications shall be done in consideration of the amount of communication data, communication bandwidth, and latency risk.

・ Session maintenance

In case that stateful connection is required based on session management by the center with ALB configured environment, use sticky sessions (Cookie).

7.2.5. MQTT

7.2.5.1. Basic Specifications

- The MQTT version 5 protocol is supported. The communication shall conform to the following specifications.
  - MQTT Version 5.0 OASIS Standard (https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html)
- Use the topic alias function to reduce communication volume.
- Since MQTT security (confidentiality and authentication) is achieved by TLS, the extended authentication function should not be used.
- MQTT broker shall use TLS client authentication to authenticate publishers and subscribers.
- Authorization shall be performed in addition to authentication.The authentication token shall be bound to the client certificate in compliance with RFC8705.
- MQTT broker shall user only allow authorized publishers and subscribers to publish to or subscribe from a topic.
- MQTT publishers and subscribers shall implement end to end confidentiality protection to sensitive messages that are sent over MQTT. Mutual TLS is terminated on the broker or the authentication GW before the broker and messages are stored on the broker.

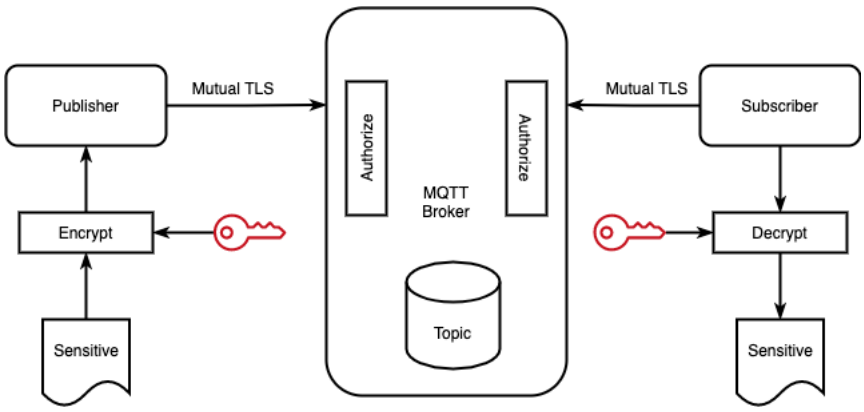


Figure 7-14 MQTT Authentication, Authorization and Confidentiality

The API must be written in accordance with the following specifications of AsyncAPI

v2.0 or later.
https://www.asyncapi.com/docs/specifications/2.0.0

7.2.5.2. Message Format

The message format in MQTT and the common parts of fixed and variable headers (parts commonly used by applications (services)), which are the scope of this document, are shown in Figure 7-15.
Vehicle identifier shall be assigned as the topic name.

- Basic topic name

  Topic= <Vehicle identifier>/<Application Name>

- Topic name in PUBLISH

  Topic= <Vehicle identifier >/<Application Name>/<Direction*>

  *Direction: C2V (Center→in-Car unit), V2C (in-Car unit→Center)

- Topic name used for Ack response when PUSH notification is received

  Applications that received PUSH notification shall perform response PUBLISH to the Response Topic included in the PUBLISH of the PUSH notification.

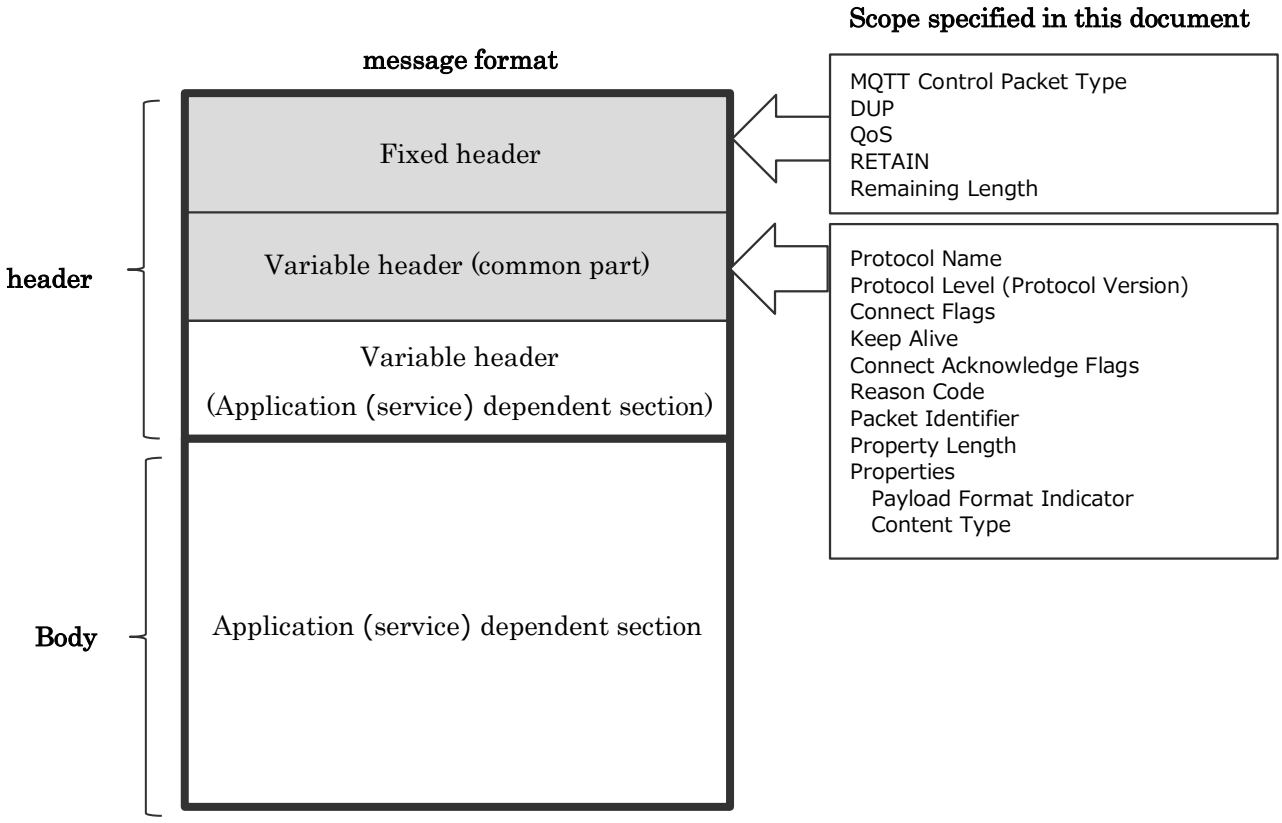


Figure 7-15 Message format in MQTT

7.2.5.3. Header Definition

The QoS of the fixed header shall be set to "1 (At least once delivery)".
For CONNECT, the variable header "keep alive" shall be used.

The setting value of keep alive is shown in 7.2.5.7 Session Management

For PUBLISH, use the following variable header Property.

- Payload Format Indicator
  - 0 (0x00) Byte
  - 1 (0x01) UTF-8 (RFC 3629)

- Content Type
  Based on RFC6838, one of the following must be set. Note that case-insensitive strings are not case-sensitive.
  - application/octet-stream (binary format)
  - application/x-protobuf (Protocol Buffers format)
  - application/json (JSON format)
  - text/plain (text format)

7.2.5.4. Response Status Code

Table 7-21,Table 7-22,Table 7-23 and Table 7-24 specify whether or not a retry is required for each Reason Code of each command.
Do not retry if the error is caused by a client-side specification.

- Connect (CONNACK)

Table 7-21 Reason Code and Retry Required (CONNACK)

Reason Code	Hex	Name	Retry
0	0x00	Success	No
128	0x80	Unspecified error	No
129	0x81	Malformed Packet	No
130	0x82	Protocol Error	Yes
131	0x83	Implementation specific error	Yes
132	0x84	Unsupported Protocol Version	No
133	0x85	Client Identifier not valid	No
134	0x86	Bad User Name or Password	No
135	0x87	Not authorized	No
136	0x88	Server unavailable	Yes
137	0x89	Server busy	Yes
138	0x8A	Banned	Yes
140	0x8C	Bad authentication method	No
144	0x90	Topic Name invalid	No
149	0x95	Packet too large	No
151	0x97	Quota exceeded	Yes
152	0x98	Administrative action	No
153	0x99	Payload format invalid	No
154	0x9A	Retain not supported	No
155	0x9B	QoS not supported	No
156	0x9C	Use another server	No
157	0x9D	Server moved	No
159	0x9F	Connection rate exceeded	No

- Publish (PUBACK)

Table 7-22 Reason Code and Retry Required (PUBACK)

Reason Code	Hex	Reason Code name	Retry
0	0x00	Success	No
16	0x10	No matching subscribers	No
128	0x80	Unspecified error	No
131	0x83	Implementation specific error	Yes
135	0x87	Not authorized	No
144	0x90	Topic Name invalid	No
145	0x91	Packet identifier in use	No
151	0x97	Quota exceeded	Yes
153	0x99	Payload format invalid	No

- SUBSCRIBE (SUBACK)

Table 7-23 Reason Code and Retry Required (SUBACK)

Reason Code	Hex	Name	Retry
0	0x00	Granted QoS 0	No
1	0x01	Granted QoS 1	No
2	0x02	Granted QoS 2	No
128	0x80	Unspecified error	No
131	0x83	Implementation specific error	Yes
135	0x87	Not authorized	No
143	0x8F	Topic Filter invalid	No
145	0x91	Packet Identifier in use	No
151	0x97	Quota exceeded	Yes
158	0x9E	Shared Subscription not supported	No
161	0xA1	Subscription Identifiers not supported	No
162	0xA2	Wildcard Subscription not supported	No

・ Unsubscribe (UNSUBACK)

Table 7-24 Reason Code and Retry Required or Not (UNSUBACK)

Reason Code	Hex	Name	Retry
0	0x00	Success	No
17	0x11	No subscription existed	No
128	0x80	Unspecified error	No
131	0x83	Implementation specific error	Yes
135	0x87	Not authorized	No
143	0x8F	Topic Filter invalid	No
145	0x91	Packet Identifier in use	No

7.2.5.5. Timeout specification

The waiting time for receiving ACK for each command transmission shall be as follows

・ CONNACK wait time: 30 seconds
・ PUBACK waiting time: 30 seconds
・ SUBACK waiting time: 30 seconds

Note that cases where uploading or downloading requires a certain amount of time are out of this scope.

In order to prevent the load concentration on the center, the retransmission process after timeout shall be under control to extend the retransmission interval.

7.2.5.6. Retry Specification

The use cases that require retries, the number of retries, and the retry interval are shown in Table 7-25.

Table 7-25 Use cases requiring retry, number of retries, and retry interval

Use case	Real-time nature	Number of retries	Retry interval
Data upload (in-Car unit → Center)	Required	2	3sec
	Not required	2	1min

Data download (Center → in-Car unit)	Required	N/A	N/A

7.2.5.7. Session Management

・ Session establishment

Establish an MQTT connection constantly when all of the following conditions are applied.

- WAN side IP address shall be assigned to in-Car unit communication termination in the possible status for the data communication.
- Obtaining VIN
- ADF upload has been completed.

If the validity check between in-Car unit (ECU) and vehicle (vehicle identifier) cannot be realized by the authentication process due to the implementation SW limitation at the center side, obtain Onetime Password (OTP) using HTTP GET from TSC by Onetime Password acquisition function before connecting to MQTT.
Authenticate with username and password of MQTT CONNECT.
Set VIN for Username and acquired OTP for password.
Set VIN for the Client ID of MQTT CONNECT.

・ Session maintenance

Keep alive between in-Car unit and the center shall be performed from client side. (See 7.2.5.3)
The value (interval) of keep alive setting shall be set by considering power consumption, disconnecting time by the network side and so on.

・ Session discard

When disconnecting the external communication automatically by in-Car unit, MQTT DISCONNECT shall be completed.

7.2.5.8. Basic Sequence

The basic sequence of CONNECT, SUBSCRIBE, and PUBLISH for MQTT is shown in Figure 7-16,Figure 7-17 and Figure 7-18

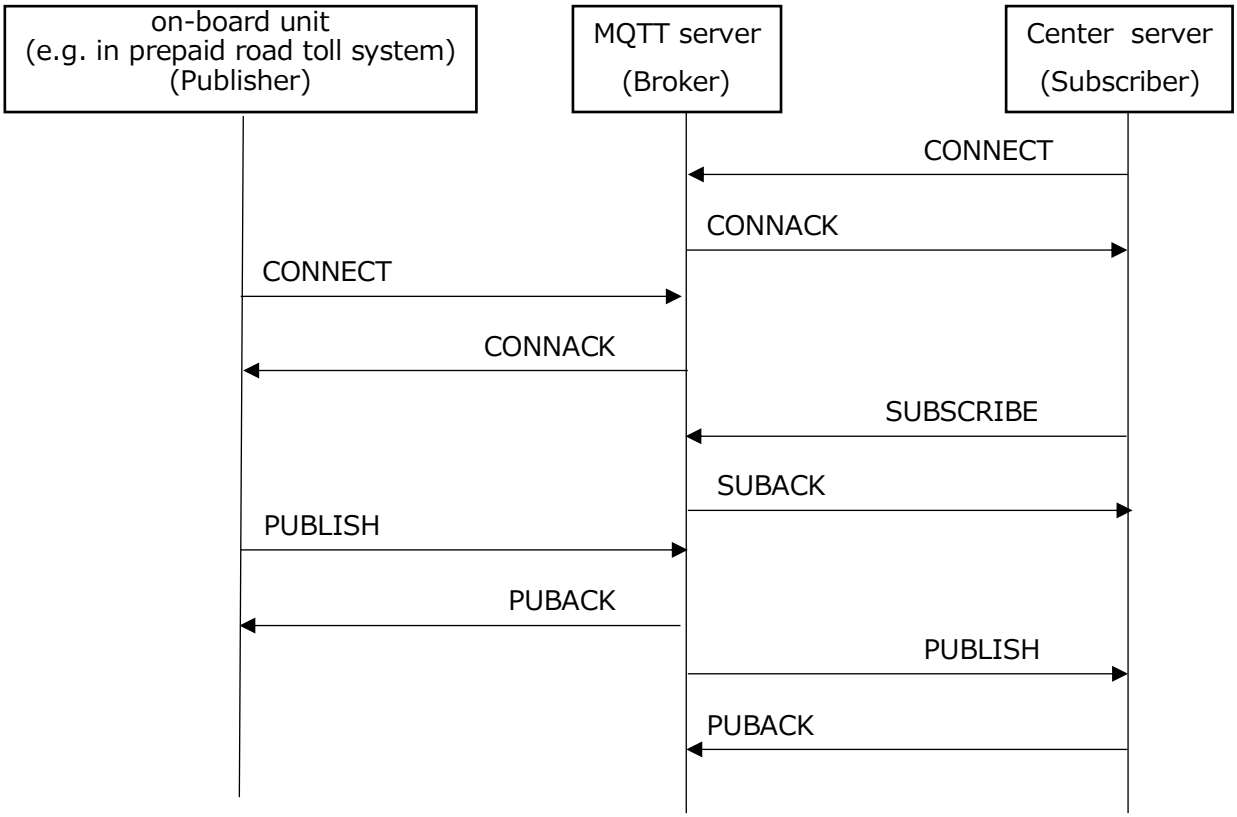


Figure 7-16 Basic sequence of MQTT (data upload)

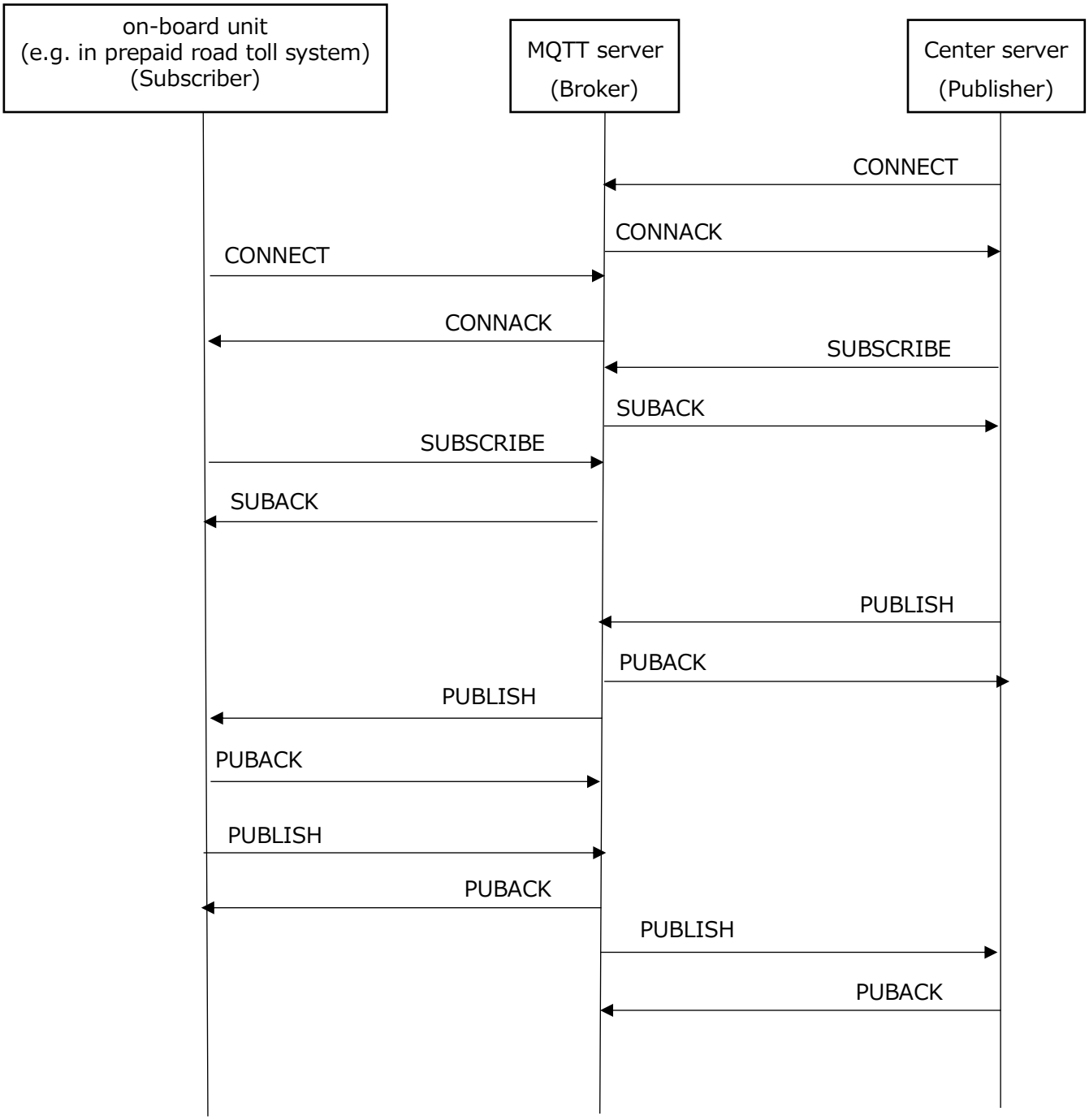


Figure 7-17 Basic MQTT sequence (push from center)

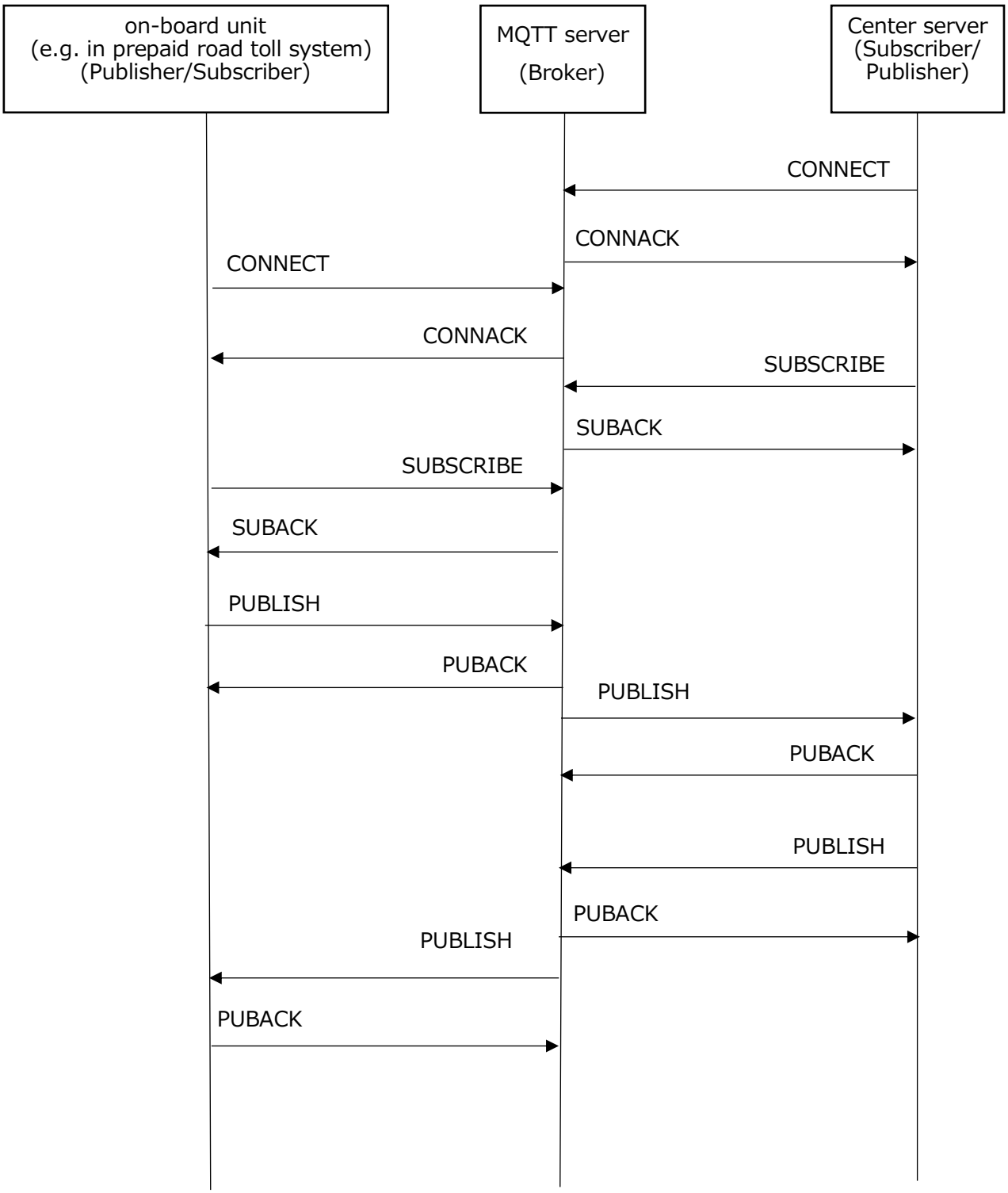


Figure 7-18 Basic sequence of MQTT (data download)

7.2.5.9. API Specification

AsyncAPI is schema-compatible with OpenAPI.
Therefore, in consideration of compatibility with the API specification of OpenAPI (REST) (7.2.2.9), the API specification for each use case is shown below.

7.2.5.9.1. Data Uploading

The API specification for data upload is shown in Table 7-26.

Table 7-26 API specification for data upload (AsyncAPI)

Object/Field				Description
asyncapi	asyncapi			Async API Version
info	title			API Title
	version			API version
	description			API Description
servers	url			URL of the server
	protocol: mqtt			Specify mqtt as the protocol
channels	{vehicle_id}/{ApplicationId}			API channel name ・vehicle_id: Vehicle identifier (see 7.3) ・ApplicationId: Application name
	parameters			Parameters
		vehicle_id		Vehicle identifier (see 7.3)
			schema	schema
			type: string	character string
	publish	operationId		API Name
		message		Message * Content of the message depends on the application (service).
			payload	payload
			type: string	character string

Object/Field					Description
	subscribe	operationId			API Name
		message			Message * Content of the message depends on the application (service).
			payload		payload
				type: string	character string

<. yaml file: data upload >>

```
asyncapi: 2.0.0

info:
  title: api-doc-can-data
  version: 1.0.0
  description:  API Documentation for CAN data upload

servers:
  production:
    url: example.com
    protocol: mqtt

channels:
  /upload/{vehicle_id}:
    subscribe:
      operationId: uploadCANDataFileReceive
      message:
        payload:
          type: string
    publish:
      operationId: uploadCANDataFileSend
      message:
        payload:
          type: string
    parameters:
      vehicle_id:
        schema:
          type: string
```

7.2.5.9.2. Push from the Center

The API specification for push from the center is shown in Table 7-27.

Table 7-27 API specification for push from the center (AsyncAPI)

Object/Field					Description
asyncapi	asyncapi				Async API Version
info	title				API Title
	version				API version
	description				API Description
servers	url				URL of the server
	protocol: mqtt				Specify mqtt as the protocol
channels	{vehicle_id}/{ApplicationId}/{app_id}/{sub_id}				API channel name vehicle_id: Vehicle identifier(see 7.3) ApplicationId: Application name app_id: Application ID of the push destination sub_id: Application sub-ID of push destination
	parameters				Parameters
		vehicle_id			Vehicle identifier (see 7.3)
			schema		schema
				type: string	character string
		app_id			Application ID of the push destination
			schema		schema
				type: integer	integer
				format: int32	32-bit integer
		sub_id			Application sub-ID of the push destination
			schema		schema
				type: integer	integer
				format: int32	32-bit integer

Object/Field					Description
	publish	operationId			API Name
		message			Message * Content of the message depends on the application (service).
			payload		payload
				type: string	character string
	subscribe	operationId			API Name
		message			Message * Content of the message depends on the application (service).
			payload		payload
				type: string	character string

<. yaml file: data upload >

```
asyncapi: 2.0.0

info:
  title: api-doc-push
  version: 1.0.0
  description:  API Documentation for Server Push

servers:
  production:
    url: example.com
    protocol: mqtt

channels:
  /push/{vehicle_id}/{app_id}/{sub_id}:
    subscribe:
      operationId: pushDataReceive
      message:
        payload:
          type: string
    publish:
      operationId: pushDataSend
      message:
        payload:
          type: string
    parameters:
      vehicle_id:
        schema:
          type: string
      app_id:
        schema:
          type: integer
          format: int32
      sub_id:
        schema:
          type: integer
          format: int32
```

7.2.5.9.3. API call from vehicle to center

MQTT is not suitable for API calls from a vehicle to a center. (See 6.1.2)
Therefore, API calls from the vehicle to the center are not subject to MQTT application.

7.2.5.9.4. Data Download

The API specification for data download is shown in Table 7-28.

Table 7-28 API specification for data download (AsyncAPI)

Object/Field				Description
asyncapi	asyncapi			Async API Version
info	title			API Title
	version			API version
	description			API Description
servers	url			URL of the server
	protocol: mqtt			Specify mqtt as the protocol
channels	{vehicle_id}/{ApplicationId}			API channel name ・vehicle_id: Vehicle identifier (see 7.3) ・ApplicationId: Application name
	parameters			Parameters
		vehicle_id		Vehicle identifier (see 7.3)
			schema	schema
			type: string	character string
	publish	operationId		API Name
		message		Message * Content of the message depends on the application (service).
			payload	payload
			type: string	character string

Object/Field					Description
	subscribe	operationId			API Name
		message			Message * Content of the message depends on the application (service).
			payload		payload
				type: string	character string
channels	<ApplicationId >/{vehicle_id}/{filename}				API channel name ・ApplicationId : Application name ・vehicle_id: Vehicle identifier (see 7.3) ・filename: File name
	parameters				Parameters
		vehicle_id			Vehicle identifier(see 7.3)
			schema		schema
				type: string	character string
		filename			File name
			schema		schema
				type: string	character string
	publish	operationId			API Name
		message			Message * Content of the message depends on the application (service).
			payload		payload
				type: string	character string
	subscribe	operationId			API Name
		message			Message * Content of the message depends on the application (service).
			payload		payload
				type: string	character string

<. yaml file: data upload >

```
asyncapi: 2.0.0

info:
  title: api-doc-download
  version: 1.0.0
  description:   API Documentation for Download Data Request and Download

servers:
  production:
    url: example.com
    protocol: mqtt

channels:
  /downloadRequest/{vehicle_id}:
    subscribe:
      operationId: downloadDataRequestReceive
      message:
        payload:
          type: string
    publish:
      operationId: downloadDataRequestSend
      message:
        payload:
          type: string
    parameters:
      vehicle_id:
        schema:
          type: string

  /download/{vehicle_id}/{filename}:
    subscribe:
      operationId: downloadDataReceive
      message:
        payload:
          type: string
    publish:
      operationId: downloadDataSend
      message:
        payload:
          type: string
    parameters:
      vehicle_id:
        schema:
          type: string
      filename:
        schema:
          type: string
```

7.3. Vehicle Identification

For the purpose of in-Car unit (ECU) identification and management at TSC, values (VIN) that can be used to identify vehicles shall be assigned for all communications to TSC.

Appendix 1

This Appendix shows valuable information in consideration of lower level specifications.

1. Data Format

The following formats are supported for sending and receiving data.

- binary data
- text data
  - JSON
  - plain text

When using HTTP or gRPC, if the data size is 2KB* or larger (fixed size), it shall be compressed using gzip.
Since MQTT is designed to handle lightweight data, the need for compression should be determined by individual requirements and if compression is required, gzip shall be used to have common method in each protocol.
*** The size should be determined by verifying the trade-off between compression load and communication size.**

In gRPC, Protocol Buffers are used as the message format.
When Protocol Buffers are used in IDL, their serialization specification shall be followed.

Other definitions are shown below.

- -endian
  - In the case of binary format, be sure to enter the byte order for each item.
  - Big endian shall be adopted

- valid value
  - Describe the format, and if it is a number, whether it is a natural number or an integer.
  - Describe the maximum number of digits, minimum value, maximum value, and possible patterns of values.

- indefinite value
  - Indefinite values must be set to 0 for the number of digits.
  - For items where 0 has significant meaning, specify F for the number of digits.

- padding
  - When the value of an item is variable and does not satisfy the number of digits, the padding format should be described.

2. File name

YYYYMMDD_HHMMSS_<generation>_<function>_<counter value>_<request ID>_<trigger identifier>.<extension>

- <generation> shall be composed of 1-digit alphanumeric and 2-digit decimal number so that the generation of the in-Car unit can be identified. (Expressed by same two digits as the generation of electronic PF.)
  e.g.: CXX (XXth of CECU)
- <function> shall be composed of 3-digit decimal number which is assigned for each function. (Expressed by three digits in consideration of increase of functions in the future.)
- <counter value> shall be set by counter value prepared by each in-Car unit. Expressed by 2-digit decimal number and assigned for each application. (Since date and time is on the second timescale, expressed by two digits assuming a few tens of triggers in a second at the most.)
- <request ID> shall be set by the request ID which is set in the requested command.
  If there's no request ID, invalid value ("FFF…" for binary and "000…" for text and json) shall be set.
- <trigger identifier> shall be set to identify triggers that launched each process.
  ※ This makes it possible to identify the cause for the same URL.
  e.g.)
  - ✧ n: IG ON trigger
  - ✧ f: IG OFF trigger
  - ✧ m: Triggered by SMS sent from the center
  - ✧ s: Triggered by scheduler

✧ c: CAN/Ethernet communication trigger

- For functions that require information to be added to the file name other than the counter value, request ID and trigger identifier, add a character string after the trigger identifier.
- The extension should be in lower case and should be set individually for each function.

- The date and time shall be as follows based on UTC (RFC3339).

	data length	Settings
YYYY	4 digits	Use 4-digit display with western calendar.
MM	2 digits	Two-digit display of 01 to 12
DD	2 digits	Two-digit display of 01 to 31
HH	2 digits	Two-digit display of 00 to 23
MM	2 digits	Two-digit display of 00 to 59
SS	2 digits	Two-digit display of 00 to 59

If the time is unknown due to lack of GPS reception or such, the time shall be set as 00000000_000000.

3. Character Code

Character code to be used shall be UTF-8 (RFC3629).

4. communication sequence

4.1. data communication flow

4.1.1. Data Uploading

4.1.1.1. HTTP/1.1

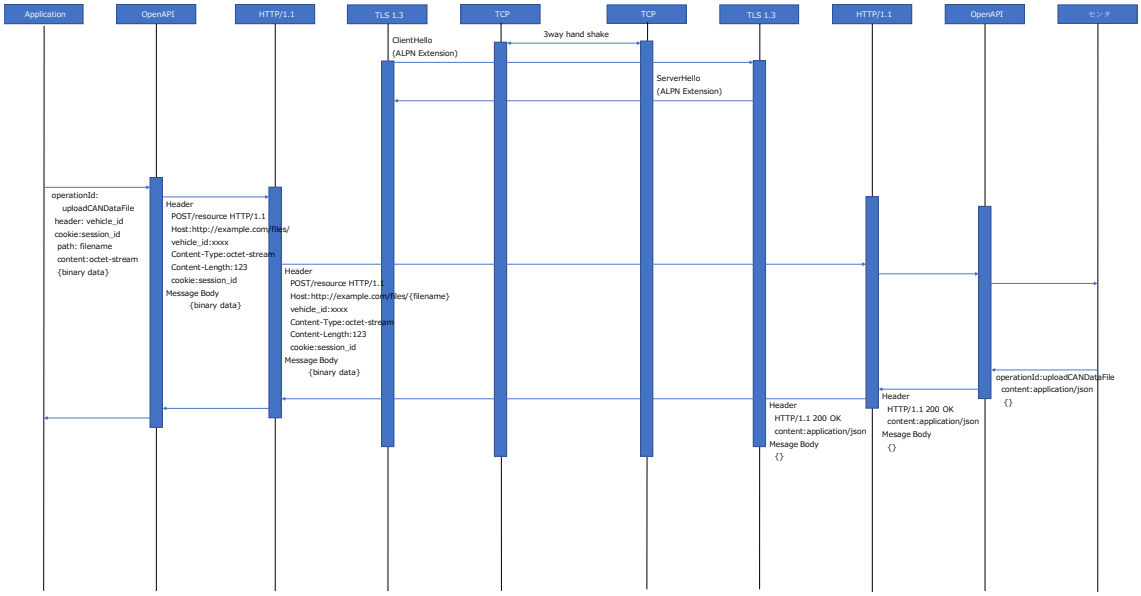


Figure A-1 Data upload sequence (HTTP/1.1)

4.1.1.2. HTTP/2

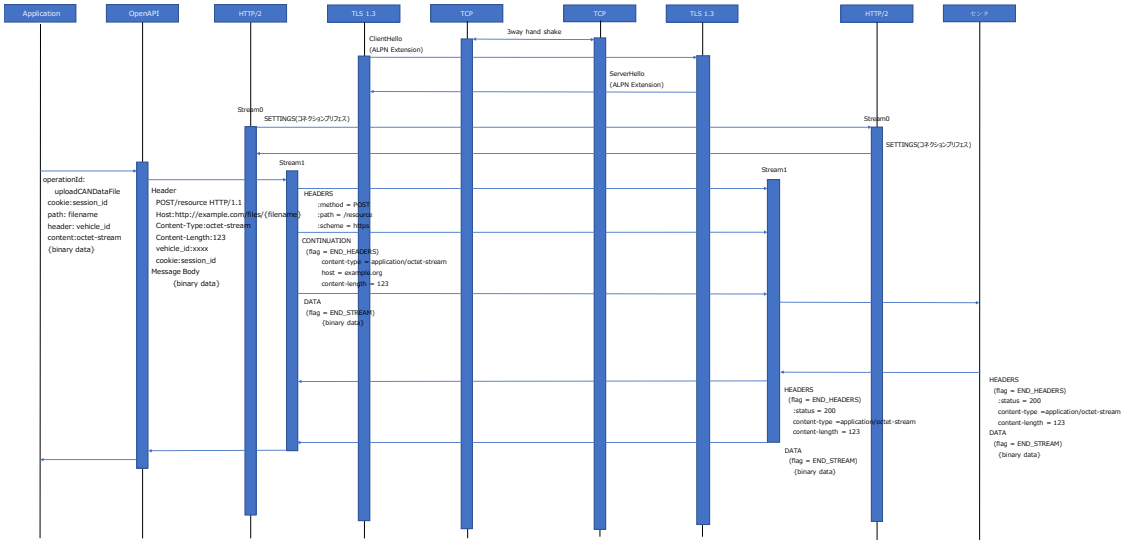


Figure A-2 Data upload sequence (HTTP/2)

4.1.1.3. gRPC

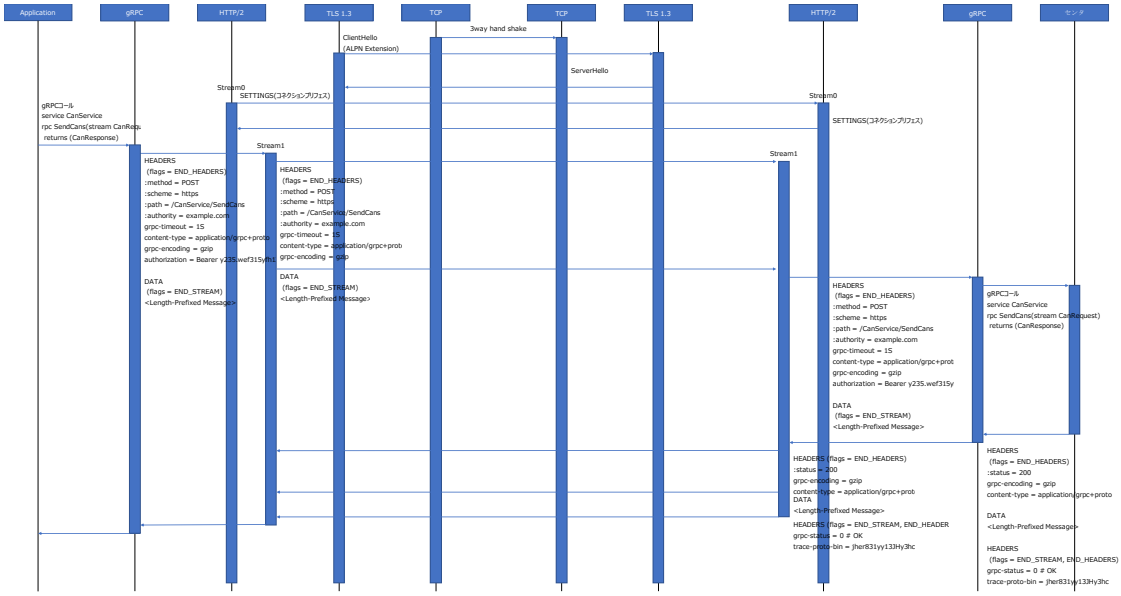


Figure A-3 Data upload sequence (gRPC)

4.1.1.4. MQTT

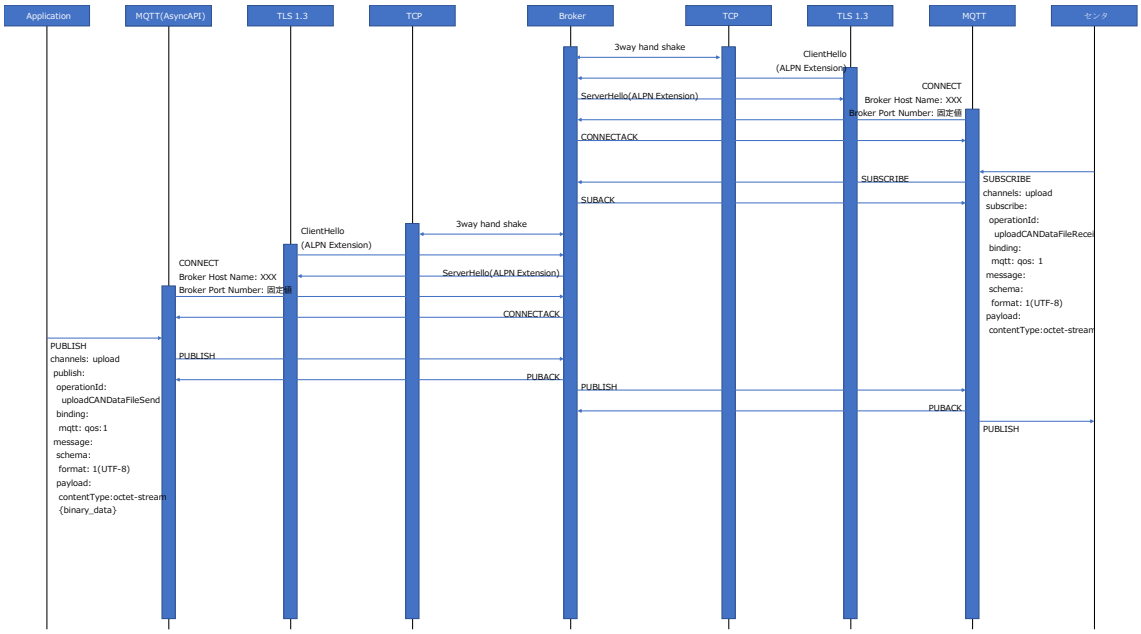


Figure A-4 Data upload sequence (MQTT)

4.1.2. Push from the center

4.1.2.1. HTTP/2

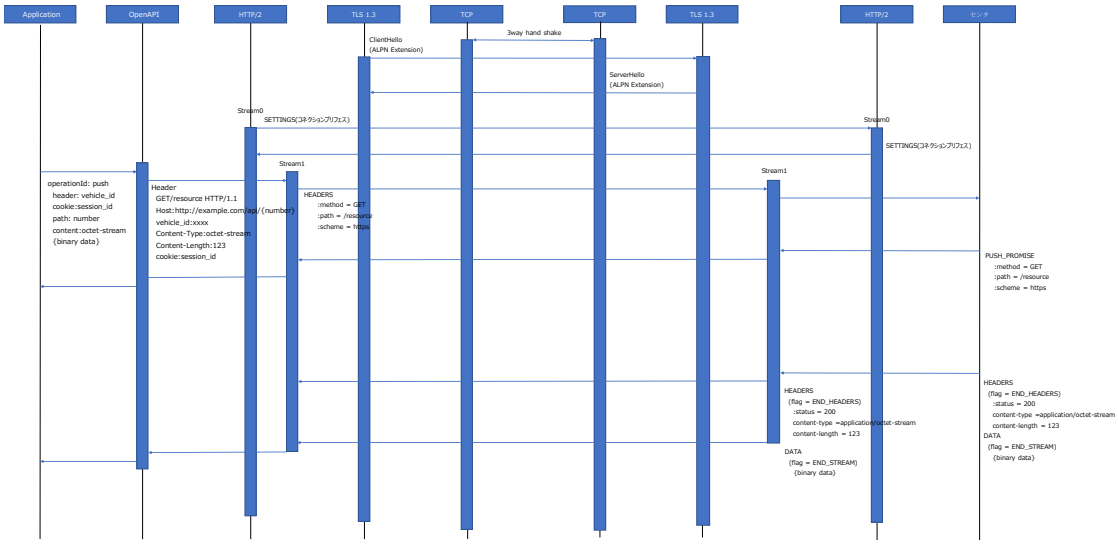


Figure A-5 Push sequence from the center (HTTP/2)

4.1.2.2. gRPC

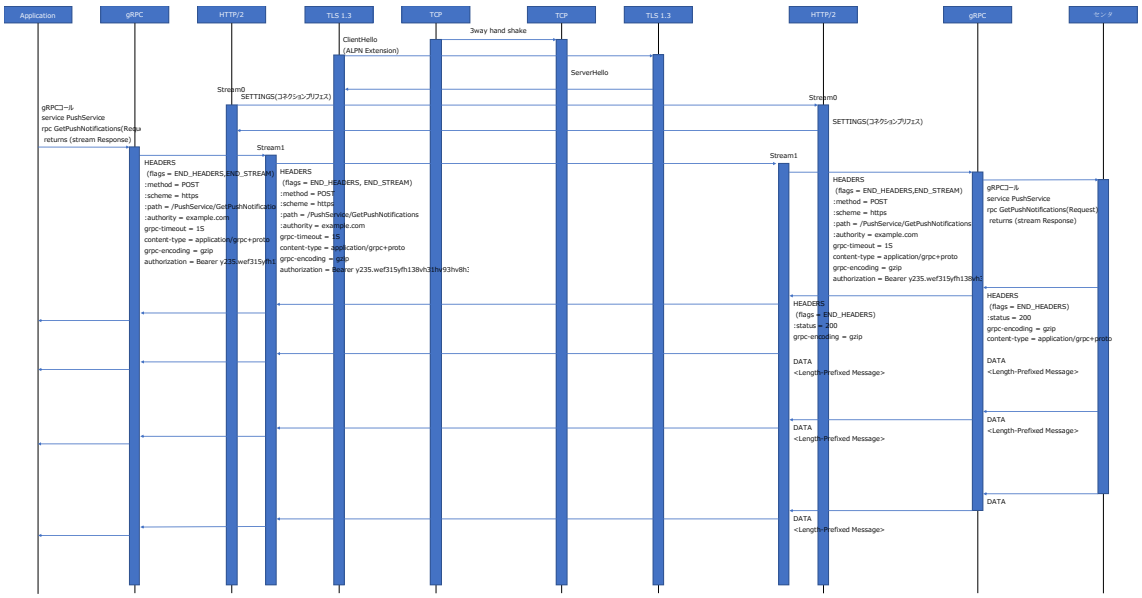


Figure A-6 Push sequence from the center (gRPC)

4.1.2.3. MQTT

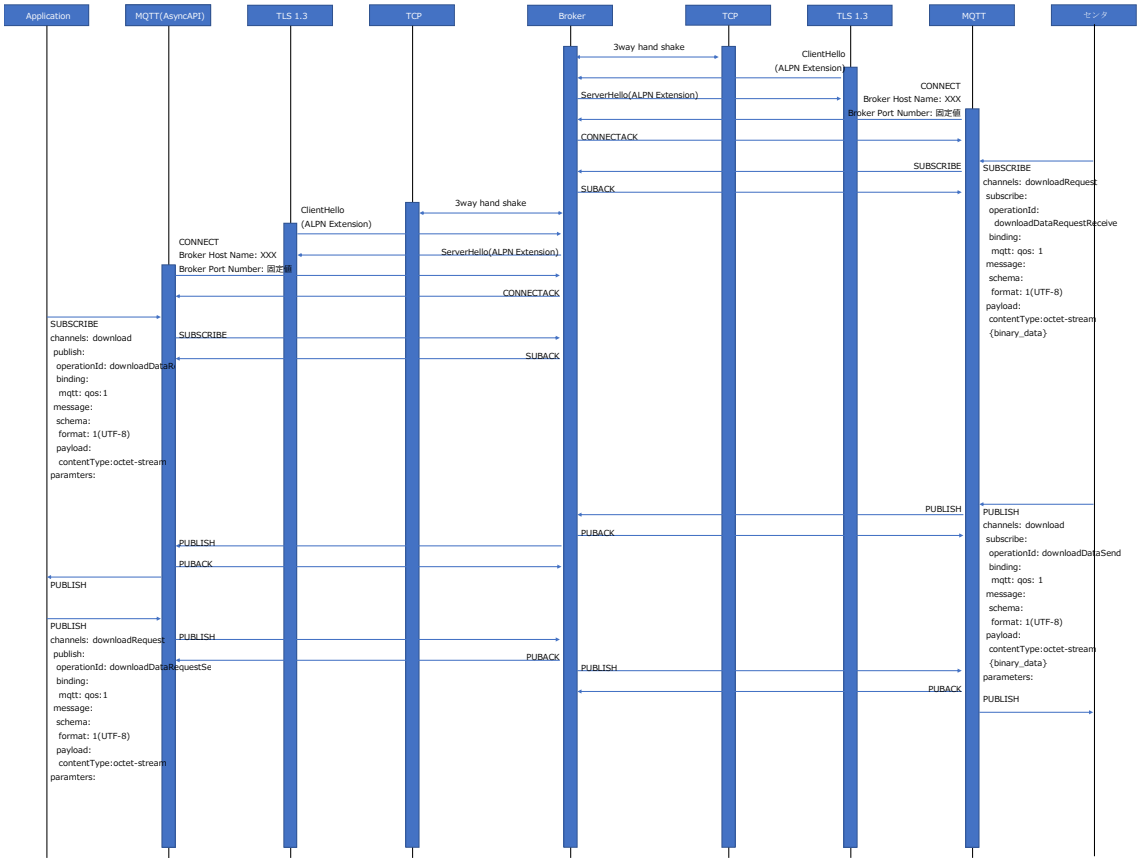


Figure A-7 Push sequence from the center (MQTT)

4.1.3. API call from vehicle to center

4.1.3.1. HTTP/1.1

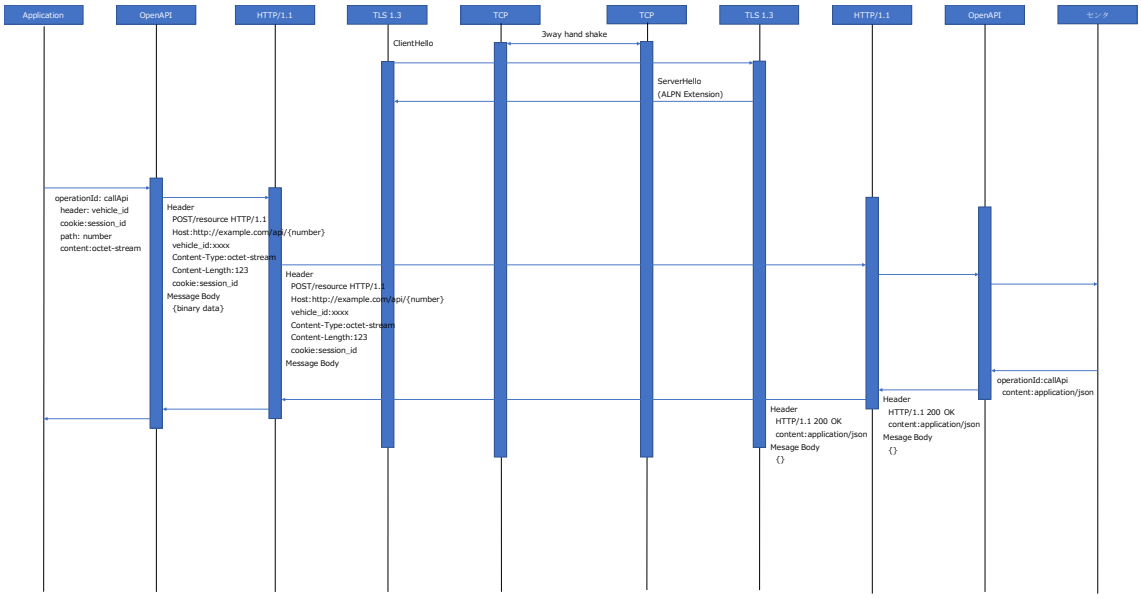


Figure A-8 API call sequence from vehicle to center (HTTP/1.1)

4.1.3.2. HTTP/2

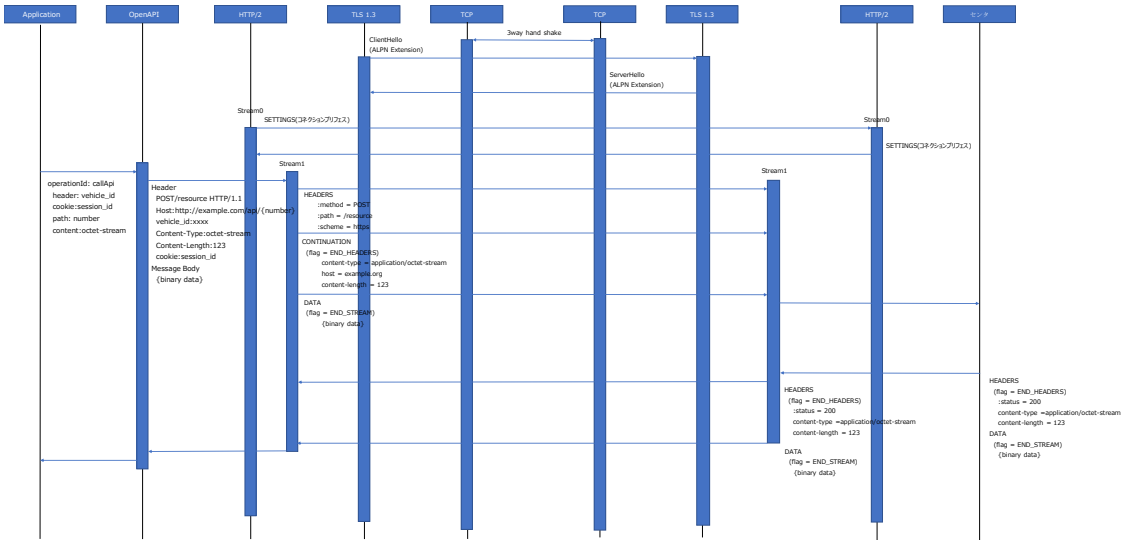


Figure A-9 API call sequence from vehicle to center (HTTP/2)

4.1.3.3. gRPC

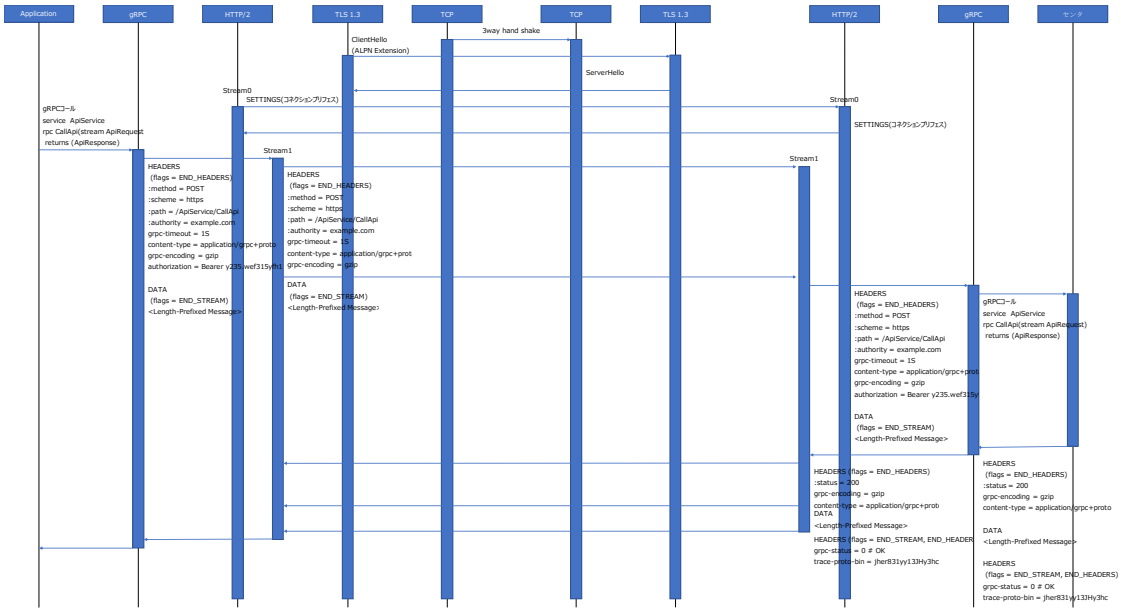


Figure A-10 API call sequence from vehicle to center (gRPC)

4.1.4. Data Download

4.1.4.1. HTTP/1.1

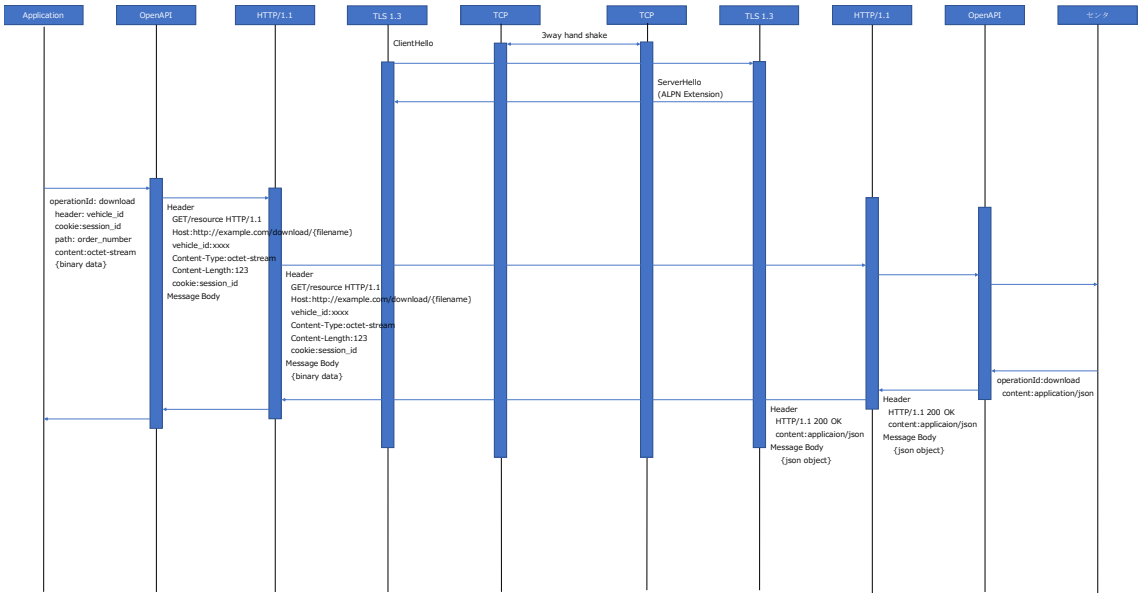


Figure A-11 Data download sequence (HTTP/1.1)

4.1.4.2. HTTP/2

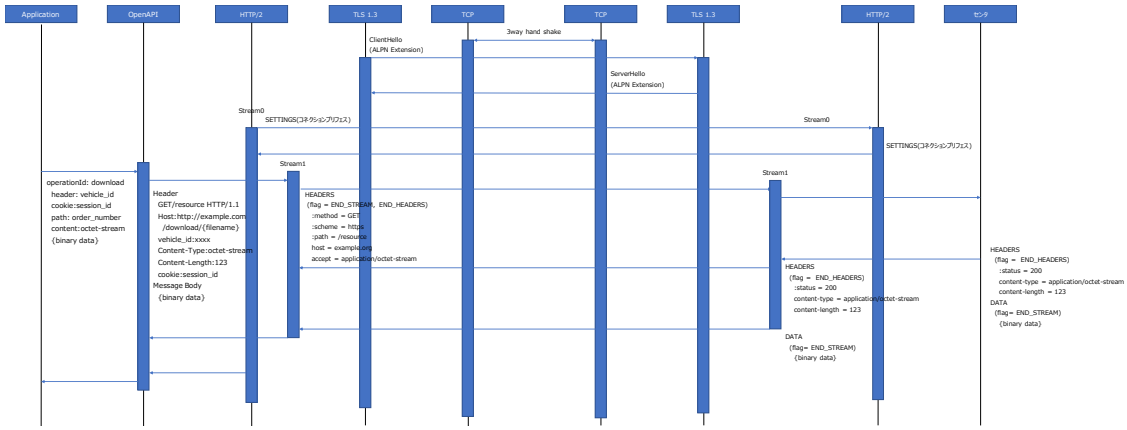


Figure A-12 Data download sequence (HTTP/2)

4.1.4.3. gRPC

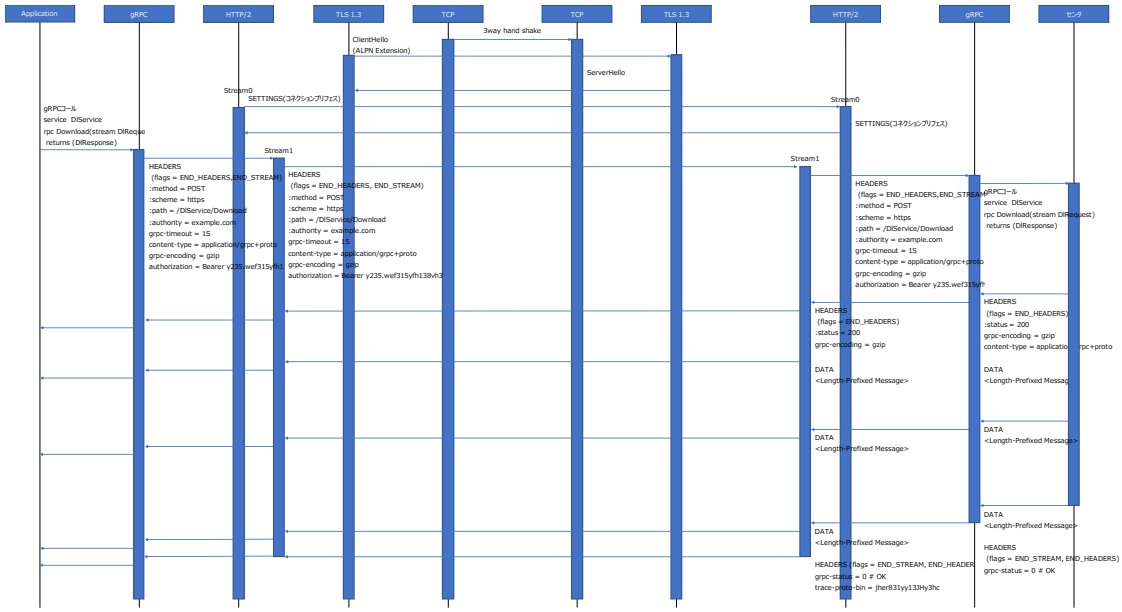


Figure A-13 Data download sequence (gRPC)

4.1.4.4. MQTT

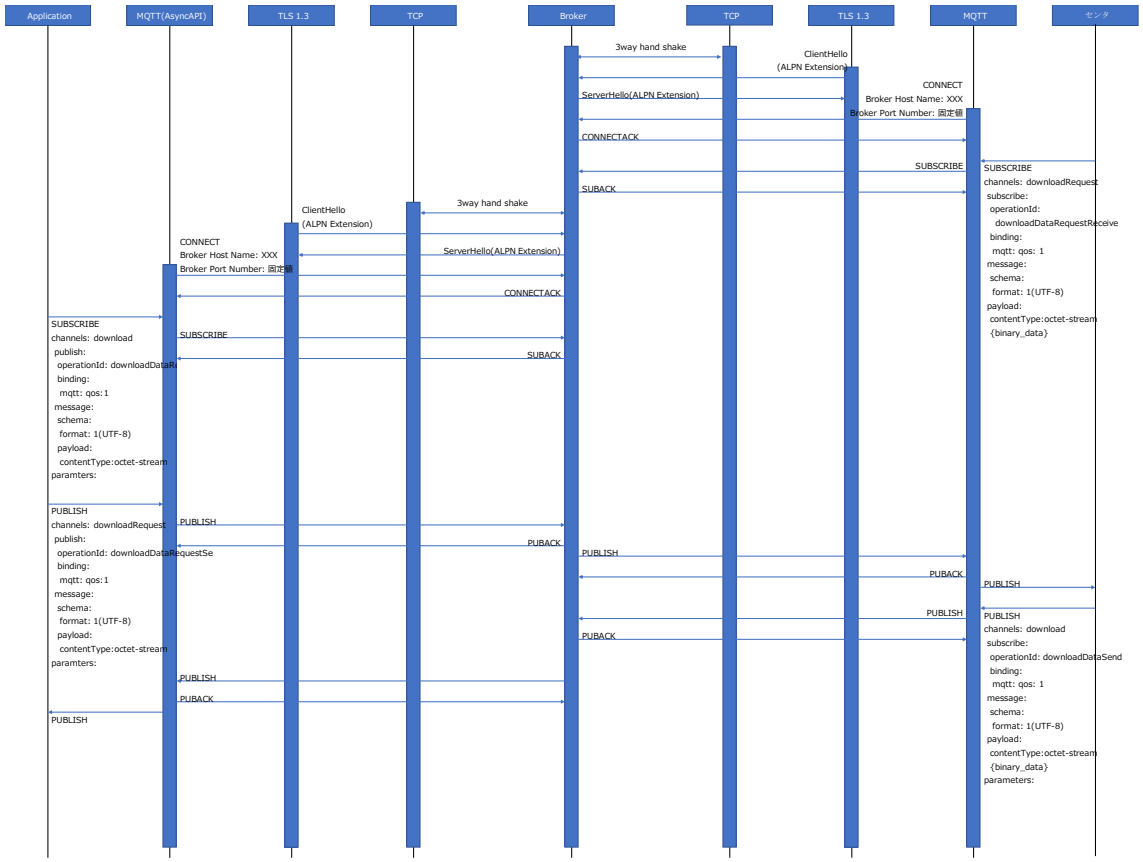


Figure A-14 Data download sequence (MQTT)

4.1.5. error sequence

4.1.5.1. HTTP/1.1

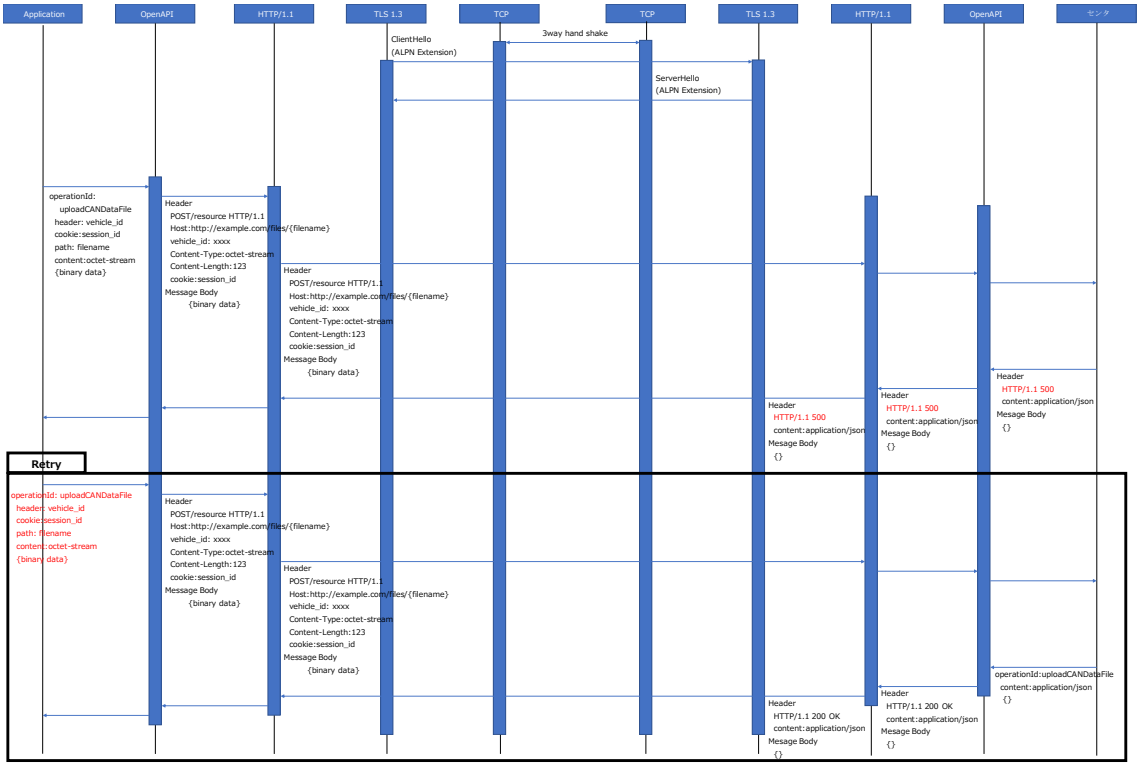


Figure A-15 [Error sequence] Data upload (HTTP/1.1)

4.1.5.2. HTTP/2

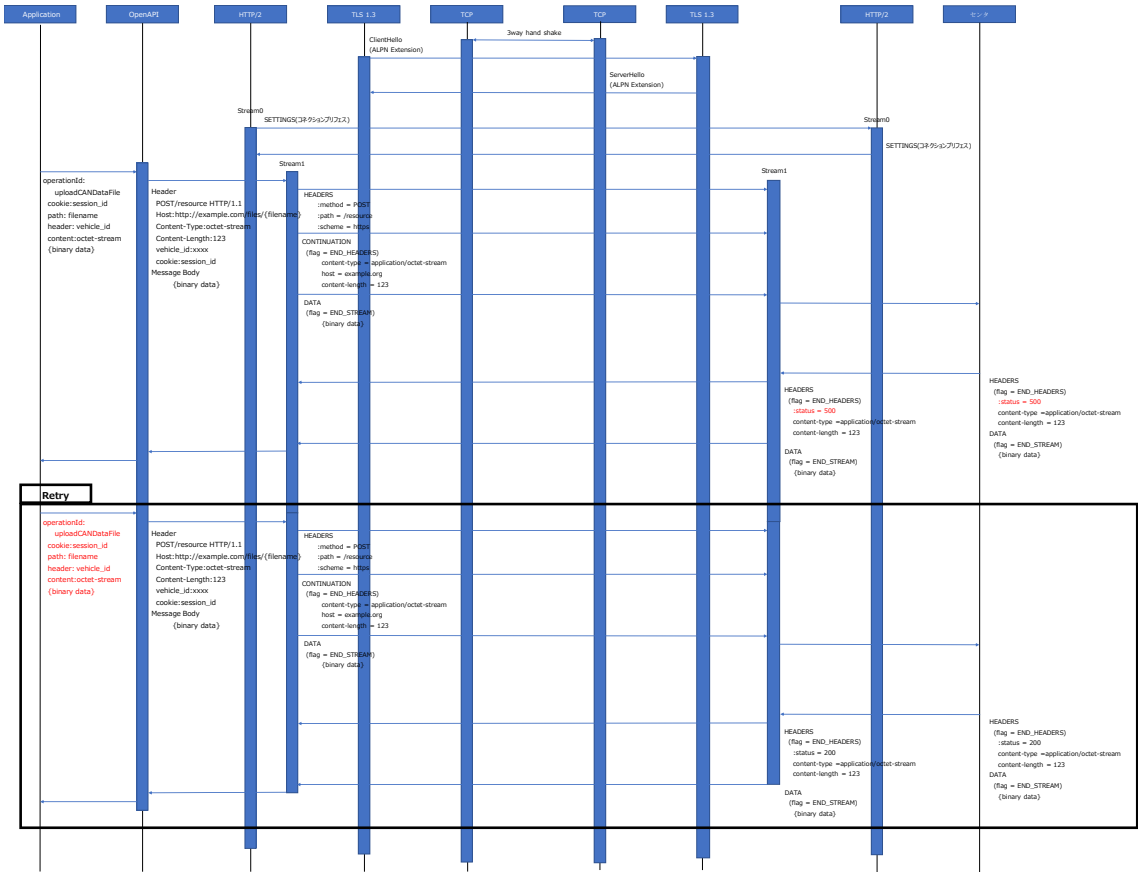


Figure A-16 [Error sequence] Data upload (HTTP/2)

4.1.5.3. gRPC

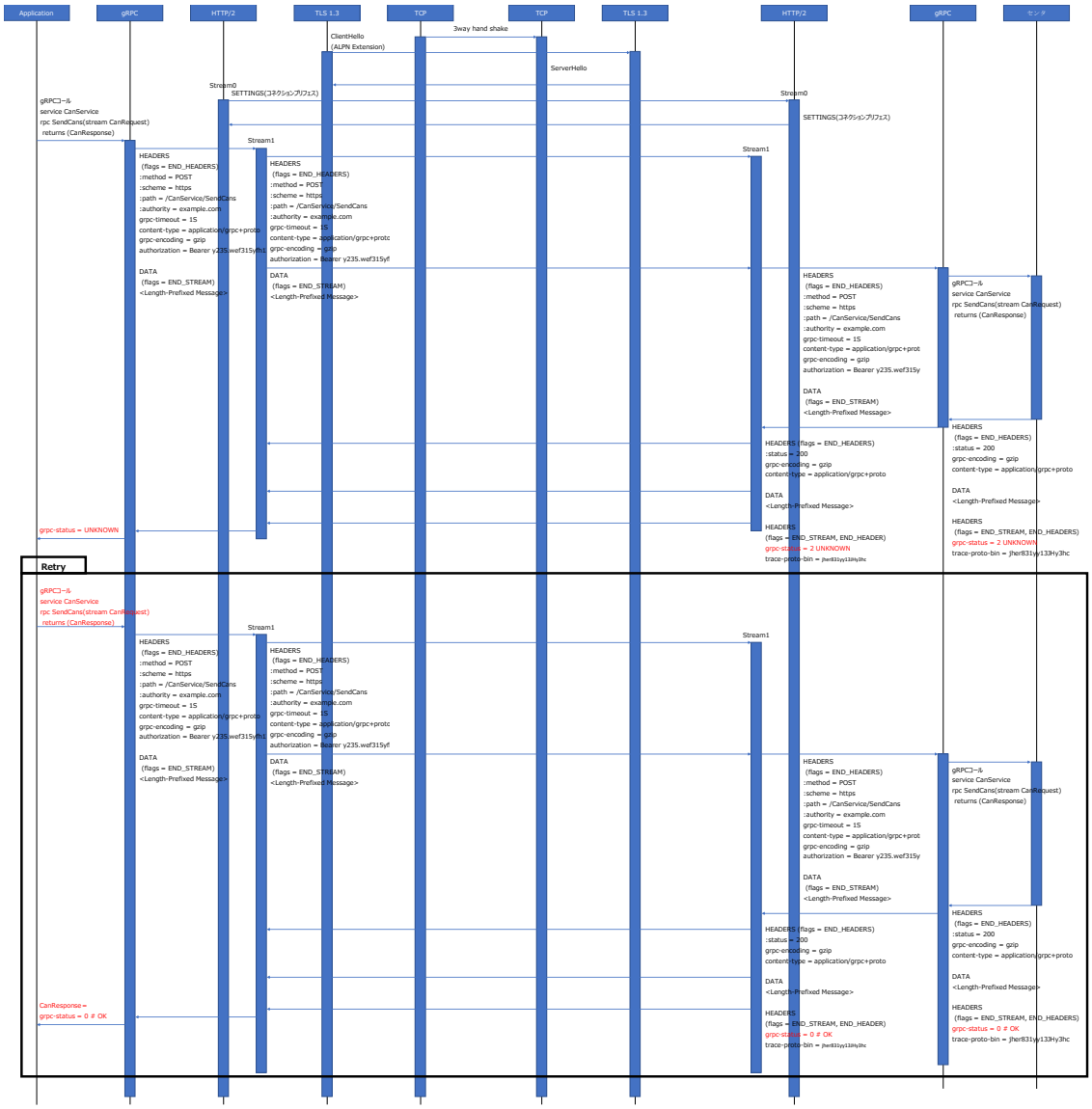


Figure A-17 [Error sequence] Data upload (gRPC)

4.1.5.4. MQTT

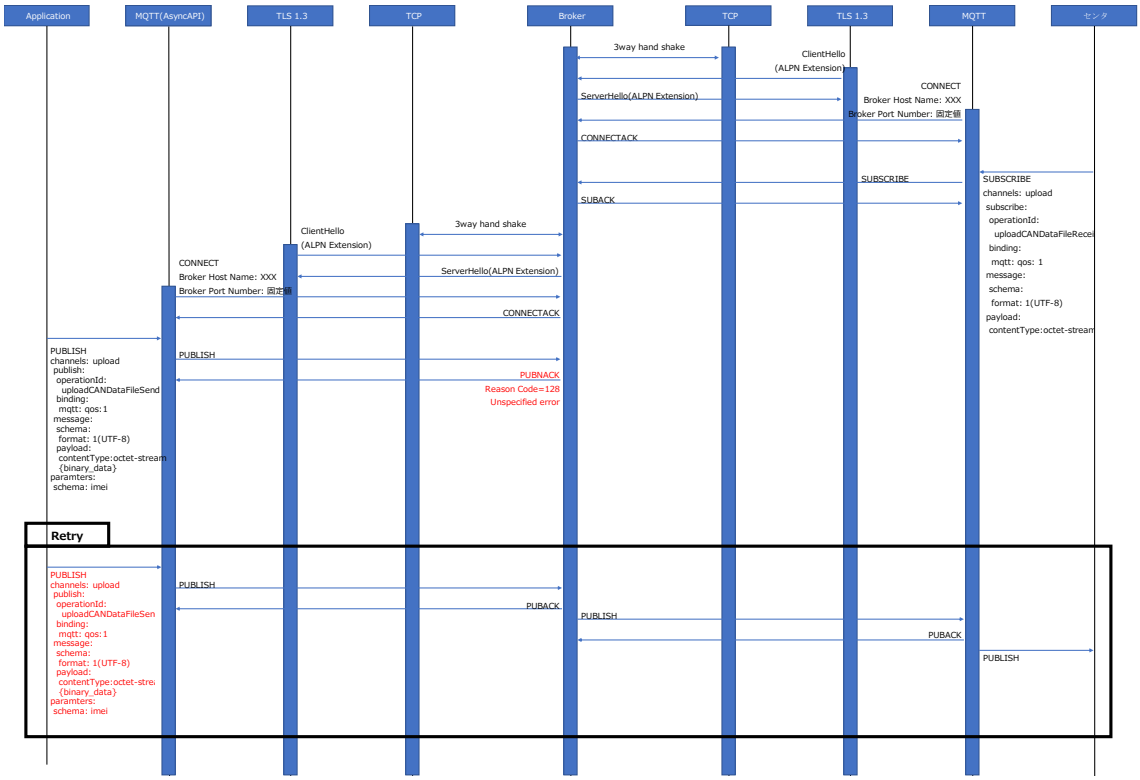


Figure A-18 [Error sequence] Data upload (MQTT)

Appendix 2

In case of the occurrence of redesign and reorganizing on Electronic PF/TSC configuration in the future, this Appendix shows the contents need to be changed accordingly.

1. Configuration

1.1. HTTP/1.1

1.1.1. Base URL

T.B.D（Method to receive URLs dynamically is under study.）

1.1.2. Port Number

Port number specified in the HTTP request shall be TCP:443 (HTTP TLS).

1.2. HTTP/2

1.2.1. Base URL

T.B.D（Method to receive URLs dynamically is under study.）

1.2.2. Port Number

T.B.D

1.3. gRPC

1.3.1. Base URL

T.B.D（Method to receive URLs dynamically is under study.）

1.3.2. Port Number

Port number specified in HTTP request shall be TCP:9080.

1.4. MQTT

1.4.1. Base URL

T.B.D（Method to receive URLs dynamically is under study.）

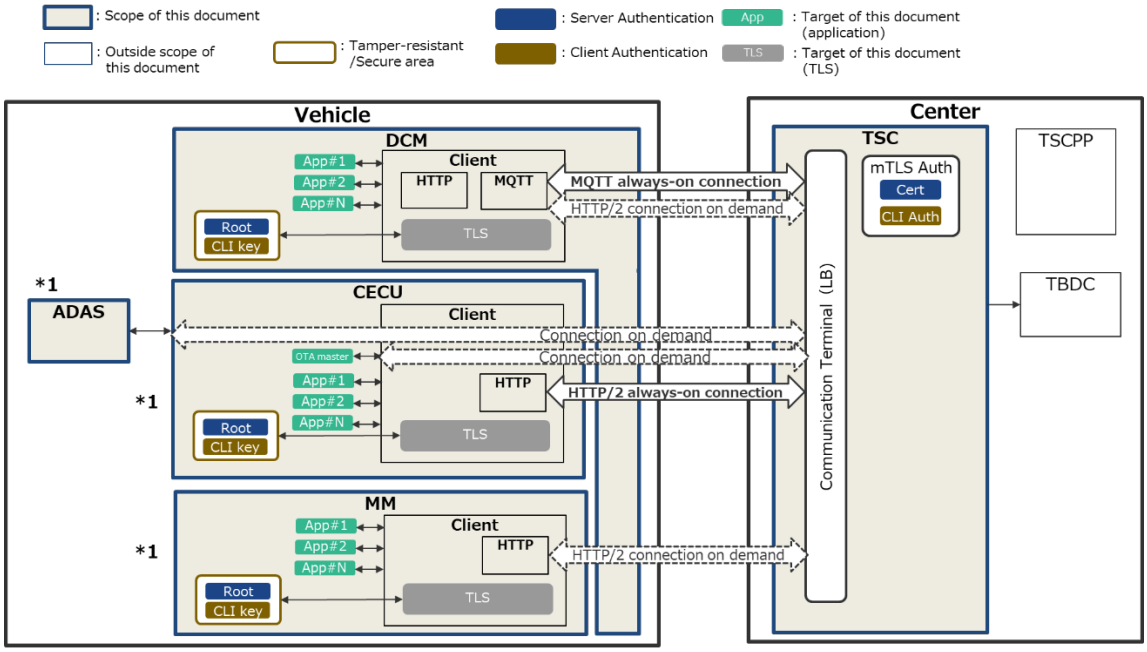
1.4.2. Port Number

Port number specified in MQTT CONNECT shall be TCP:8883 (MQTT TLS).

2. Authentication and Connection Configuration

The authentication between in-Car unit and the center will be mutual authentication including client authentication using TLS.
The number of connections between in-Car unit and the center will be reduced by reusing sessions to reduce the load in the center.
When it is required to establish an always-on communication connection between vehicle and the center, the connection protocol by in-Car unit shall not be adopted as much as possible so as not to have multiple center functions.
The push function from the center, which requires a constant connection between vehicle and the center, is basically established per each vehicle, not in-Car unit.
（See Figure A2-2）If the requirements are met by in-Car and center unit level,

multiple protocols shall not be selected. (See



*1 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS.
Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

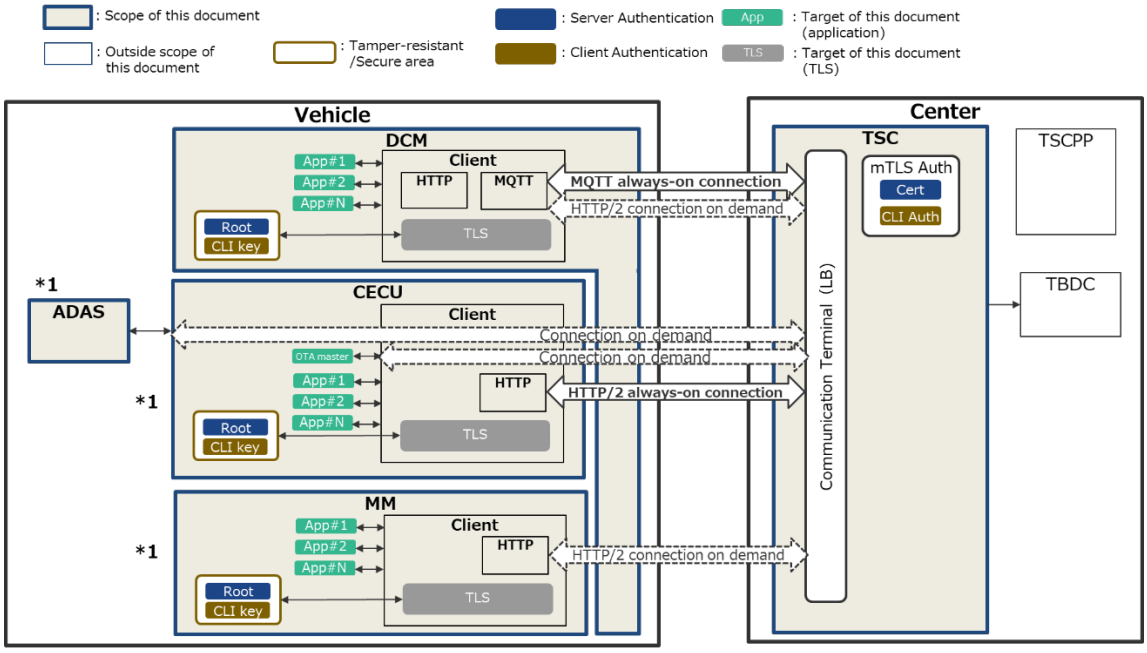
Figure A2-1)

Typical use cases and connection types are shown in Table A2-1.

Table A2-1 Typical use cases and connection types

Use case	Connection configuration
Upload CAN data	Always-on connection
Upload video data	On-demand connection (per event)
Upload diagnostics data	On-demand connection (per event)
API access from vehicle to center	On-demand connection (per event)
Push from the center	Always-on connection (between DCM and TSC)
Obtain access token (initial authentication)	On-demand connection (per event)
Download data	On-demand connection (per event)
OTA	On-demand connection (per event)

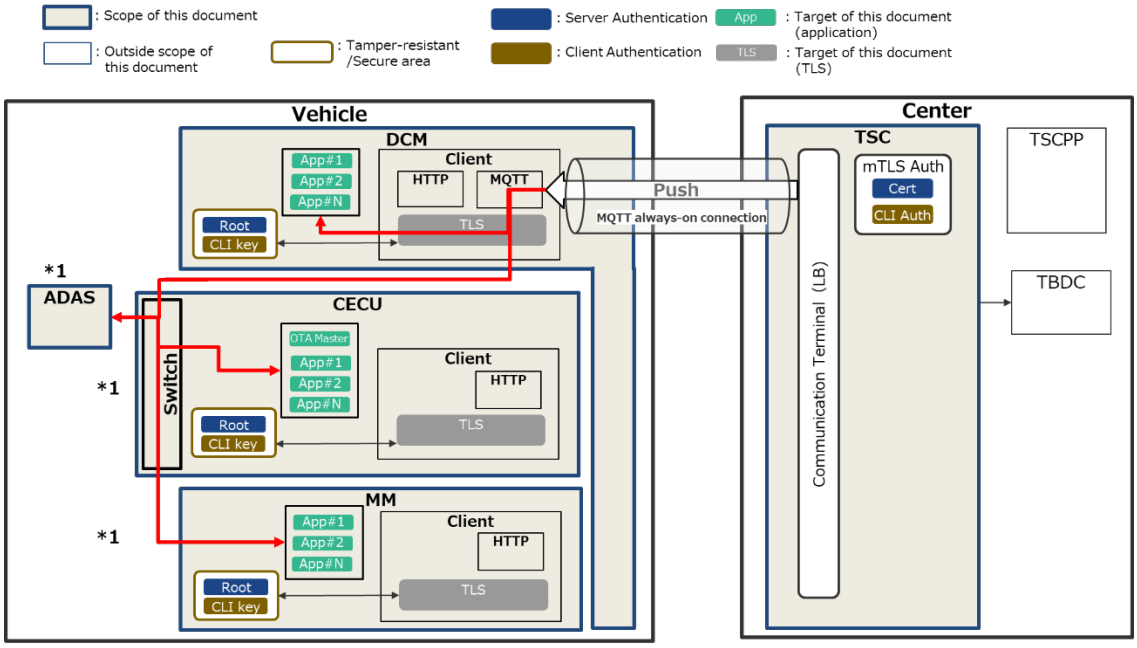
Authentication between in-Car unit and the center and its overall structure are shown in



*1 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS.
Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

Figure A2-1.

Also connection structure dedicated to push from the center is shown in

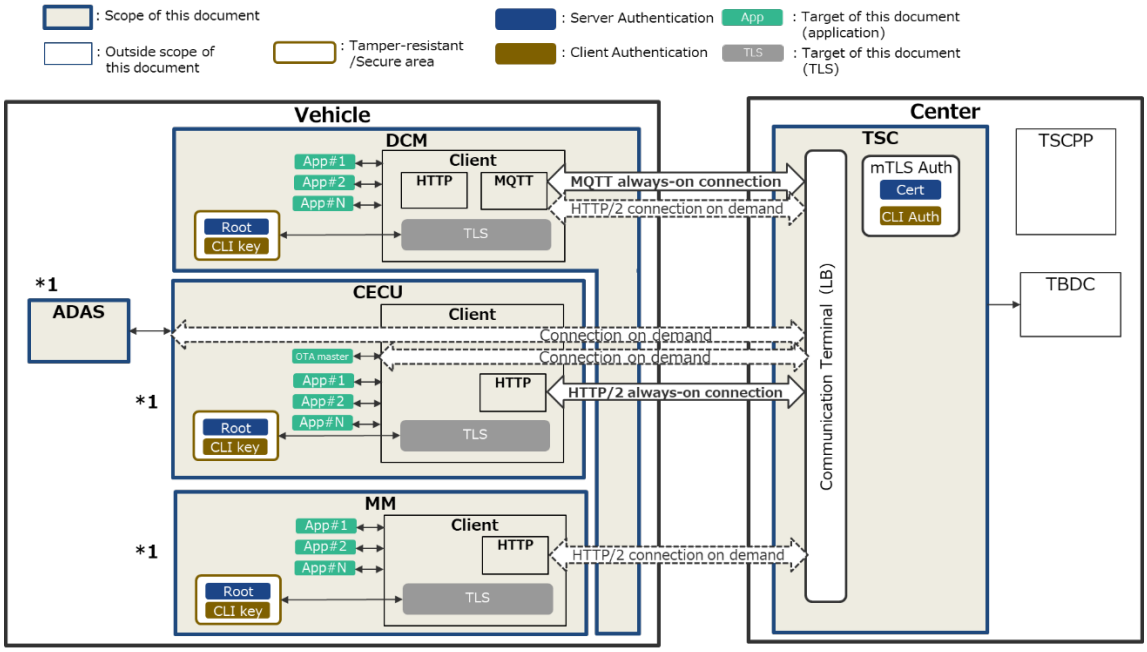


*1 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS.
Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

Figure A2-2.

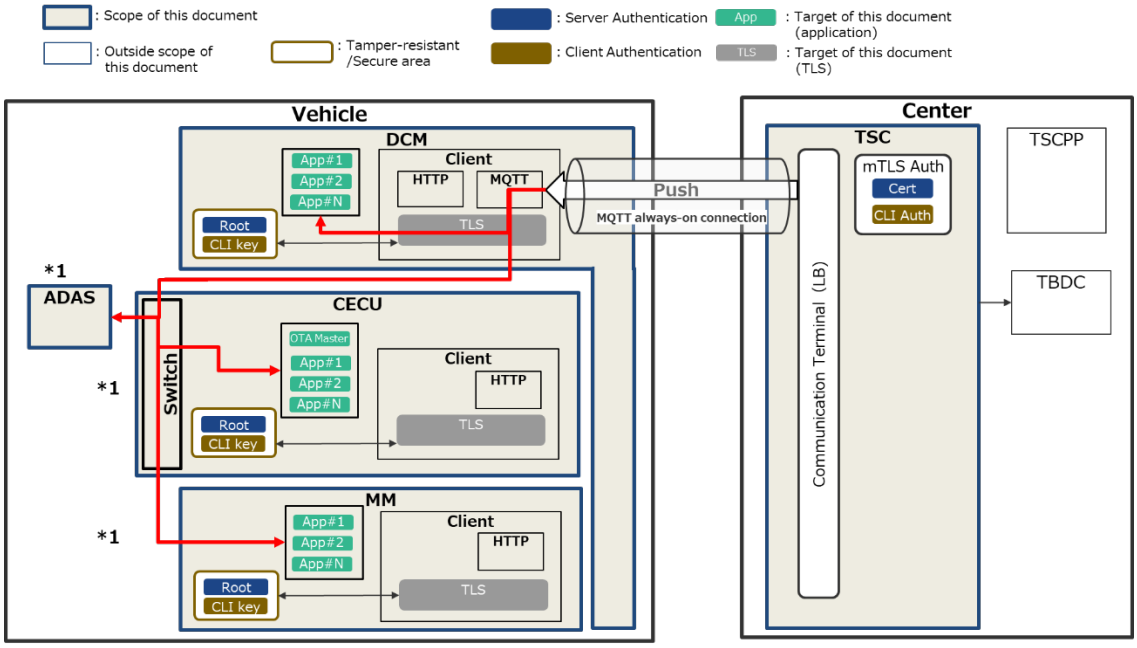
On linking in-Car unit and vehicle, in-Car unit uses information, which is secret only with the center, as token based on specific duration or specific conditions, and registers vehicle identifier or ECU ID, and request the telematics service.

Token based authorization shall be compliant with RFC8705 to bind the refresh and access tokens to the client certificate.



*1 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS.
Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

Figure A2-1 Overall structure of authentication between Vehicle and the center



*1 Depending on the system configuration of vehicle, there are vehicle without MM or CECU or ADAS.
Refer to the specifications of each ECU for the presence or absence of the corresponding ECU.

Figure A2-2 Connection structure of push notification from the center

Change History (No English translation available)

Version	Date	Revision	Author
1.2	2021.12.24	OEM 協業の観点で以下修正 ・表紙 社外配布先を明記。社外配布先として OEM を追記。 ・1.1.目的 ・1.2.位置づけ ・3.1.適用 ECU 一覧 ・図 3-1 適用 ECU と本書の適用範囲 ・3.2.適用電子 PF ・3.3.適用サーバ一覧 ・図 4-1 システム構成 ・Appendix 2 ・図 A2-1 車両とセンタ間の認証の全体構成 ・図 A2-2 センタからのプッシュの接続構成 次世代の車載通信機の技術仕様を共同で開発する OEM 向けにトヨタ固有となる表現を修正 セキュリティ観点で以下修正 ・6.2.4 アプリケーションレイヤセキュリティ セキュリティ要件からDNSSECをサポート推奨とし、文言追加 ・7.2.5.1. 基本仕様（MQTT） セキュリティ要件の実装方法として拡張認証機能を必要とする可能性があるために削除 MQTT 認証、認可プロセスを追加するために全面的に改訂	
1.1	2021.11.26	・6.2.3. 証明書 「チェーンする信頼された証明書のみ」の文言を追加 ・6.4 車両識別 認証に加えて認可を行う点を追記 認証トークンをクライアント証明書にバインドするために、RFC8705 に準拠すること。 ・7.1.1.1 基本仕様 DNSSEC をサポートする方針とし、文言追加 ・7.1.1.2.1. 暗号化スイート 強度の観点を踏まえ、優先 1 のみとし優先 2 の記	

Version	Date	Revision	Author
		載を削除 ・7.1.1.2.2. 鍵交換方式 強度の観点を踏まえ、ECDHE を 384 ビット以上と修正 ・7.1.1.2.3. 署名方式 RSA3072 系統のものをインデントにて区分（方式の追加変更はありません） ・7.1.1.3 クライアント認証 証明書チェーンと正確な文言に修正 ・7.1.1.5. 証明書管理 サーバ認証に OCSP を利用することの追加、CRL/OCSP のキャッシュコントロールの追加を踏まえ、全面的に改訂 図 7-16、図 7-17 を追加 ・7.1.1.7. 異常系シナリオ OCSP 対応につき、表 7-4 の記載事項を追加 ・7.2.5.1. 基本仕様 MQTT 認証、認可プロセスを追加するために全面的に改訂 ・Appendix2 トークンによる認可について一文追加	
1.04	2021.10.01	・3.1 適用 ECU 一覧 ADAS ドメコンを ADASECU 表記に変更 ・図 7-8 HTTP/2 におけるメッセージフォーマット ・図 7-9 HTTP/2 の基本シーケンス（GET） ・図 7-10 HTTP/2 の基本シーケンス（POST） ・図 7-11 gRPC におけるメッセージフォーマット http2 表記に変更 6.1.2 通信プロトコルとユースケース ダイアグデータのアップロードを見直し 7.1.1.2.3 署名方式 将来セキュリティ脅威を踏まえたアルゴ保持を明記 ・7.2.5.5 タイムアウト仕様 タイムアウト時の再送間隔について追記 ・7.2.5.7 セッション管理 セッション確立、維持、破棄の仕様について追記	
1.03	2021.08.26	・1.1. 目的 本書のセンタ側対象範囲について、TSC 以外も含	

Version	Date	Revision	Author
		むよう、表現を修正 ・7.1.1.5. 証明書管理 トヨタ発行の証明書の失効確認方法について追記 有効期間満了前でも、+B=OFF 時は保持した CRL を破棄する旨記載を変更 ・7.1.1.7. 異常系シナリオ(TLS) サーバ認証のシーケンスと対応(図 7-3、表 7-4)修正 クライアント認証シーケンスと対応(図 7-4、表 7-5)追加 タイムアウト時間、リトライ間隔、リトライ回数追記(表 7-4、表 7-5) ・7.2.4.1 基本仕様(gRPC) v.0.9 版で誤って削除した gRPC によるデータ送信時の分割について記載 ・7.2.5.7. セッション管理(MQTT) センタからのプッシュの場合は、keep alive をクライアント側から行い、keep alive の時間間隔は消費電力、ネットワーク側からの切断時間などを考慮して設定する旨追記 ・Appendix1 下位仕様を検討するにあたって、有益となる情報を記載する旨追記 ・Appendix2 p19ePF または TSC の構成に今後見直し・改編が発生した場合、それに応じ変更が必要となる内容を記載する旨追記	
1.02	2021.08.05	・7.1.1.2.1 暗号化スイートの記載を優先度順に変更 ・7.1.1.2.2 DHE の鍵長を 4096 ビット以上とする ・7.1.1.2.3 RSASSA に鍵長を指定 ・7.1.1.5 サーバから OCSP レスポンス受信時の車載器側の対応を追記。サーバ側はクライアント証明の失効確認であることを明記 ・誤記修正および校正	
1.00	2021.07.29	・1.2. 位置づけ 図 1 1 本書の位置づけ の参照仕様書を削除 ・7.1.1.7. 異常系シナリオ	

Version	Date	Revision	Author
		図 7-3、表 7-4 は、車載器(ECU)側におけるサーバ認証の異常系のシナリオについて、CRL によるサーバ証明書の失効確認を前提としたシーケンスおよび対応内容である旨追記	
0.92	2021.07.28	・6.1.3. プロトコルスタック 誤記修正 ・7.2.4.3. ヘッダ定義 gRPC の user-agent の説明を修正(製品識別子および gRPC 推奨のライブラリ名をスペース区切りで設定する旨明記) ・図 3-1 適用 ECU と本書の適用範囲 図 4-1 システム構成 図 A2-1 車載器とセンタ間の認証の全体構成 図 A2-2 センタからのプッシュの接続構成 V2X を削除	
0.90	2021.07.21	・ヘッダ、フッタを変更 ・7.1.1.2. 暗号化アルゴリズム 車載器、センタ間双方の暗号化スイートと鍵交換、署名方式であることを記載 ・7.1.1.4. セッション管理 コネクションの再接続等において、SIGPIPE（一度切断されたソケットを用いた送信におけるプロセスの異常終了）が発生しないよう考慮する旨追記 ・7.1.1.5. 証明書管理 参照 RFC の間違い修正(8954→6960) ・7.2.2.3 ヘッダ定義 Cookie に関する記載を削除 センタでのセッション管理を前提としたステートフルな接続が必要な場合は Set-Cookie、Cookie ヘッダを使用するよう記載内容を変更 ・7.2.3.3 ヘッダ定義 Cookie に関する記載を追加 ・7.2.3.7. セッション管理 ・7.2.4.6. セッション管理 センタでのセッション管理を前提としたステートフルな接続が必要な場合はスティッキーセッション（Cookie）を使用するよう記載内容を変更 ・7.2.5.5. タイムアウト仕様	

Version	Date	Revision	Author
		アップロードまたはダウンロードに時間を要する場合は対象外とする旨追記 ・7.2.5.8. 基本シーケンス（MQTT） Appenx1 4.1.2.3. センタからのプッシュ 図 3-17 MQTT の基本シーケンス（センタからのプッシュ） 図 A 7 センタからのプッシュシーケンス（MQTT） MQTT のプッシュを双方向のシーケンスに変更 ・7.3 車両識別 車両を識別できる値として VIN、IMEI 双方への対応が検討である内容に変更 ・Appendix を Appendix1、Appendix2 に分離 ・Appendix1 1.1.1. Base URL 1.2.1. Base URL 1.3.1. Base URL 1.3.1. Base URL T.B.D.であるが、動的に URL を取得する仕組みを検討中である旨追記 ・Appendix1 2.ファイル名 世代は、電子 PF の世代表現に合わせて 2 桁とする旨追記 機能は、将来の機能の増加を鑑み 3 桁の表現とする旨追記 カウンタ値は、日時が秒単位であり、1 秒以内に発生するトリガが多くても数十で収まる前提において、2 桁の表現とする旨追記 ・Appendix2 2 認証と接続構成 VIN マッピングについて追記	
0.55	2021.07.20	4.1. システム構成 ECU と TSC 経由で通信する TSP は本書の対象外とし、ECU と直接通信する TSP のみ本書の対象とする旨追記 ・5. ユースケース 「イベント通知」を「車両からセンタへの API 呼出し」に統合	

Version	Date	Revision	Author
		→説明に、"リモートサービスのイベント通知など"を追記 「リモートコントロール」を「センタからのプッシュ」に統合 →説明に、"リモートサービスにおけるリモートコントロールなど"を追記 ・7.1.1.2.3. 署名方式 ECDSA256、RSA3072 を追記 ・7.1.1.4. セッション管理 セッションの有効期間を 12 時間とする旨追記 ・7.1.1.5. 証明書管理 OCSP、CRL の両方対応とする記載とし、OCSP stapling は非対応であることを記載 ・7.2.3.3. ヘッダ定義 HTTP/2 のヘッダは RFC7541 に基づき圧縮を行う旨追記 ・7.2.4.1. 基本仕様 複数ストリームのデータアップロードに関しては、動画などの容量大のデータ通信を行っている最中に他の通信を行うと帯域が不十分となる可能性があるため、データサイズ等に留意する旨追記 ・7.2.2.7. セッション管理 HTTP/1.1 利用時において、基本的にセッションの維持は行わない。（Cookie を使用しない）旨記載 ・7.2.3.7. セッション管理 ・7.2.4.6. セッション管理 車載器とセンタとの常時接続が必要な場合は、スティッキーセッション（Cookie）を使用する内容に変更 ・7.2.5.2. メッセージフォーマット（MQTT） PUSH 通知を受信したアプリケーションは、PUSH 通知の PUBLISH に含まれる Response Topic 宛に、応答の PUBLISH を行う旨追記 ・7.3 車両識別 車両を識別できる値は、VIN または IMEI の何れかを検討中である旨追記 ・Appendix 2 ファイル名	

Version	Date	Revision	Author
		世代は 2 桁の 10 進表現とする旨追記 ファイル名に機能を追加し、機能毎に 3 桁の 10 進表現とし、機能毎に付与する旨追記 カウンタ値は 2 桁の 10 進表現とし、アプリケーション毎に付与する旨追記 トリガ識別子の具体的を記載 ・Appendix 章構成 今後の変更の可能性が有るもの、無いものに章立てを分離 変更なし：1～4 章(データフォーマット、ファイル名、文字コード、通信シーケンス) 変更の可能性あり：5、6 章(コンフィギュレーション、認証と接続構成)	
0.54	2021.07.19	・2. 用語定義 NEV、VIN を追加 ・4.1 システム構成 TLS Proxy 経由の通信において、不特定の ECU からの接続を許可しない旨追記 ・5. ユースケース ダイアグデータのユースケース変更 イベント通知、リモートコントロールのユースケース追加 ・7.1.1.7. 異常系シナリオ 表 3 8 をリトライ、通信不可(終了)条件と、アクションに分けて記載 ・7.1.1.4. セッション管理 ・7.2.2.7. セッション管理 ALB 利用時において、セッションの維持が必要な場合、スティッキーセッション（Cookie）を使用する向け説明を変更 ・7.1.1.5. 証明書管理 サーバ証明書は CRL、クライアント証明書は OCSP により失効確認を行う内容で修正 ・7.2.2.2. メッセージフォーマット 7.2.2.9. API 仕様 7.2.3.2. メッセージフォーマット 7.2.3.9. API 仕様 7.2.4.2. メッセージフォーマット	

Version	Date	Revision	Author
		7.2.4.4. API 仕様 7.2.5.2. メッセージフォーマット 7.2.5.9. API 仕様 パス名(HTTP)、トピック名(MQTT)にアプリケーション名を追加 ・7.2.2.4. レスポンスステータスコード HTTP レスポンスのステータスコード 567 を修正（次回 IG-ON まで通信しない） ・7.2.2.5. タイムアウト仕様 タイムアウトに対する条件を追記 ・7.2.2.6 リトライ仕様 法規要件は対象外とする旨追記 ・7.2.4.1.基本仕様 gRPC のストリーム利用時のサイズへの留意について追記 ・7.2.5.7. セッション管理 KeepAlive はクライアント側から行う旨追記 ・7.3 車両識別 車両識別子として VIN または IMEI を使う旨追記 ・Appendix2 ファイル名 世代管理はファイル名の重複がないようにするためであることを例と共に追記 トリガ識別子はアプリケーション固有の識別子は重複しない英数字一文字とする旨追記 リクエスト ID がなかった場合、バイナリの場合"FFF…"、text や json の場合は"000…“を設定する旨追記 カウンタ値はアプリケーション毎に持つ旨追記 UTC による時刻表記は RFC3339 に基づく旨追記 ・Appendix 6 図 A 20 センタからのプッシュの接続構成 CECU に対する HTTP/2 によるプッシュを削除	
0.53	2021.07.13	・7.2.4.1 基本仕様(gRPC) 送信するデータサイズが MB を超える場合の分割サイズを追記 ・7.2.4.4. API 仕様(gRPC)	

Version	Date	Revision	Author
		車両識別子(vehicle_id)は、proto ファイルではなく、Meta-Data としてアプリケーションによる設定を行う旨追記 ・7.2.5.2. メッセージフォーマット(MQTT) トピック名の付与に関するルールを追記 ・Appendix 6 図 A-19 を修正(MQTT は DCM のみ、HTTP は HTTP/2 とする) 図 A-20 センタからのプッシュの接続構成を追加	
0.52	2021.07.08	・2 用語定義 UTC を追加 ・Appendix 2.ファイル名 -GPS 未受信等で時刻が不明な場合の時刻表記を修正(00000000_000000 とする) -日時は UTC による旨追記	
0.51	2021.07.05	・2 用語定義 mTLS を追加 ・図 3 1 適用 ECU と本書の適用範囲 図から冗長や矢印を削除 ・7.1.1.3. クライアント認証 RFC8446 に基づきクライアント認証を行うよう記載 ・7.1.1.5 証明書管理 車載器とサーバの双方要件である(サーバ証明書とクライアント証明書それぞれ失効確認が必要である)旨追記・修正 ・7.2.1 アプリケーションデータ User-Agent の内容を製品識別子として記載 7.2.2.2. メッセージフォーマット パス・クエリパラメータの冗長な説明を削除 ・7.2.2.3 ヘッダ定義 User-Agent、製品識別子の説明を修正 ・7.2.2.7 セッション管理 ・7.2.3.7 セッション管理 ・7.2.4.6 セッション管理 Cookie はセッション維持が必要な場合のみとする旨追記 ・7.2.2.9.1 データアップロード ・7.2.2.9.3 車両からの API 呼出し	

Version	Date	Revision	Author
		・7.2.2.9.4 データダウンロード ・7.2.3.9.2 センタからのプッシュ yaml ファイルの不備を修正 ・7.2.4.3. ヘッダ定義 情報管理キーを製品識別子に修正 ・Appendix6 認証と接続構成 車載器とセンタ間の認証と、代表的なユースケース毎の接続形態を追記	
0.5	2021.06.25	初版	