

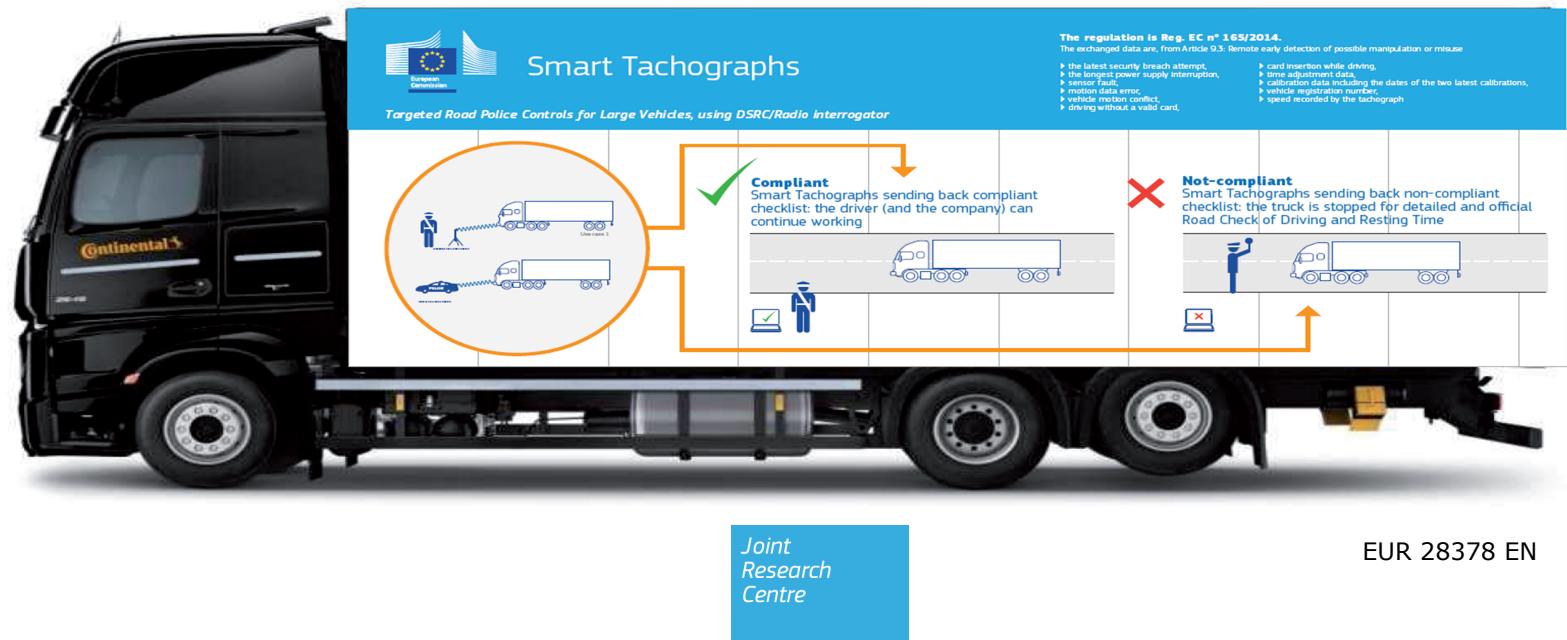
JRC TECHNICAL REPORTS

Proof of Concept (PoC) of the remote interrogation for the smart tachograph based on CEN-Dedicated Short Range Communications (DSRC)

Description of the CEN-DSRC prototype for remote interrogation

Gianmarco Baldini,
Raimondo Giuliani,
Eduardo Cano-Pons

2016



This publication is a Technical report by the Joint Research Centre (JRC), the European Commission's science and knowledge service. It aims to provide evidence-based scientific support to the European policymaking process. The scientific output expressed does not imply a policy position of the European Commission. Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use that might be made of this publication.

Contact Information

Name: Gianmarco Baldini

Address: European Commission, Joint Research Centre, Via Enrico Fermi 2749, TP 360, 21027 Ispra (VA), Italy

E-mail: gianmarco.baldini@jrc.ec.europa.eu

Tel.: +39 0332 78 6618

JRC Science Hub

<https://ec.europa.eu/jrc>

JRC105039

EUR 28378 EN

PDF ISBN 978-92-79-64805-2 ISSN 1831-9424 doi:10.2788/866989

Print ISBN 978-92-79-64804-5 ISSN 1018-5593 doi:10.2788/422399

Luxembourg: Publications Office of the European Union, 2016

© European Union, 2016

The reuse of the document is authorised, provided the source is acknowledged and the original meaning or message of the texts are not distorted. The European Commission shall not be held liable for any consequences stemming from the reuse.

How to cite: Gianmarco Baldini et al. Proof of concept for remote early interrogation. EUR 28378 EN. doi 10.2788/866989

All images © European Union 2016

PoC of the remote interrogation for the smart tachograph based on CEN-DSRC

This report provides an overview of Proof of Concept for remote interrogation for the smart tachograph application

PoC of the remote interrogation for the smart tachograph based on CEN- DSRC

Gianmarco Baldini,

Raimondo Giuliani,

Eduardo Cano-Pons

Contents

Abstract	1
1. Introduction	2
2. Remote Interrogation	3
3. Proof-of-Concept	6
4. Conclusions	8
5. Annex 1 - Manual of use of the remote interrogator for the smart tachograph	9
References	18
6. Annex 2 - Manual of use of the development kit.....	18
References	104
List of abbreviations and definitions	105
List of figures.....	106

Abstract

The aim of this technical report is to describe the proof-of-concept of the CEN-DSRC implementation of the remote interrogation function of the new version of the digital tachograph. The current digital tachograph (DT) system to monitor the driving time in commercial vehicles above 3.5 tons is governed by Council Regulation (EEC) No 3821/85 of 20 December 1985, which was modified at several occasions and more recently in 2006, when the digital tachograph was introduced, and in 2009, when it was updated to technical progress to avoid fraud and reduce the administrative burden. In July 2011 the Commission made a proposal (COM(2011) 451 final) to modify the tachograph regulation, which has been the object of discussions in Council and Parliament in the course of the ordinary legislative procedure. The final version of the approved regulation was published in February 2014 (Regulation 165/2014 (Commission, 2004)). The technical specifications of the smart tachograph were published as Regulation 799/2016 (Commission, 2011). One of the main functions is the remote interrogation of the Smart Tachograph (ST) installed in the commercial vehicle through the CEN-DSRC standard. The function supports law enforcers in the checking of potential frauds or malfunctions in the ST. To support the future deployment of the smart tachograph and to validate the technical specifications of the smart tachograph regarding the remote interrogation function, JRC issued a tender to the DSRC manufacturer Q-FREE to implement a prototype of the new remote interrogation systems. Q-FREE was chosen because it is only the leading producer of CEN-DSRC equipment and because it was not directly involved in the discussion of the technical specifications of the ST, so it did not have specific bias and it could provide a critical review of the specifications. The prototype was successfully implemented and tested. It was shown during the JRC Open day in May 2016 to thousands of visitors at the stand of the smart tachograph organized by unit DG.JRC.E3.

1. Introduction

The current digital tachograph (DT) system to monitor the driving time in commercial vehicles above 3.5 tons is governed by Council Regulation (EEC) No 3821/85 of 20 December 1985, which was modified at several occasions and more recently in 2006, when the digital tachograph was introduced, and in 2009, when it was updated to technical progress to avoid fraud and reduce the administrative burden. In July 2011 the Commission made a proposal (COM(2011) 451 final) to modify the tachograph regulation, which has been the object of discussions in Council and Parliament in the course of the ordinary legislative procedure. The final version of the approved regulation was published in February 2014 (Regulation 165/2014) (Commission, 2004). From the publication date of the new regulation, the technical specifications of the new digital tachograph must be defined within a time frame of 24 months (February 2016). According to the new regulation, the Tachograph shall be equipped with a remote communication functionality that shall allow law enforcers to read Tachograph information from passing vehicles at a road side control site.

One of the limitations of the current version of the Digital Tachograph is that the law enforcer must stop each commercial vehicle to perform an inspection. Obviously the number of commercial vehicles, which can be inspected in such a fashion is limited. In the drafting of the regulation (EU) N°165/2014 (Commission, 2004), it was proposed to provide means for targeted interrogation of the digital tachograph in a commercial vehicle to filter out potential infringements of the regulation. From an operational point of view, this means that a law enforcer will be able to interrogate directly the smart tachograph in a moving truck using an enforcement wireless communication equipment. Note that is not the intention of the legislator that the data transmitted thru the wireless communication will be used for direct fining, but it is only a selection tool to stop commercial vehicles and subsequently perform a complete manual check.

During the technical discussion with the stakeholders involved in the revision of the Digital Tachograph, CEN-Dedicated Short Range Communications (DSRC) has been selected as the main wireless communication technology to be used for the remote communication functionality. The exchanged data shall contain only those required for targeted road side checks, which is defined in the Article 9 of the published regulation 165/2014.

Even if the CEN-DSRC standard is widely deployed and various studies have already been performed for the application of electronic tolling, the use cases defined for the new DT can be quite different. The two use cases are a) roadside check with a CEN-DSRC reader operated by law enforcers on the side of the road and b) mobile reader installed in a law enforcer vehicle. In addition, the size of the data to be exchanged is also different from what is defined in electronic tolling.

To support the future deployment of the smart tachograph and to validate the technical specifications of the smart tachograph regarding the remote interrogation function, JRC issued a tender to the DSRC manufacturer Q-FREE to implement a prototype of the new remote interrogation systems. Q-FREE was chosen because it is only the leading producer of CEN-DSRC equipment and because it was not directly involved in the discussion of the technical specifications of the Smart Tachograph (ST), so it did not have specific bias and it could provide a critical review of the specifications. The prototype was successfully implemented and tested. It was shown during the JRC Open day in May 2016 to thousands of visitors at the stand of the smart tachograph organized by unit DG.JRC.E3.

This report provide a simple description of the remote interrogation function in the fugure smart tachograph and a description of the proof of concept and its presentation at the JRC Open Day in May 2016 in front of thousands of visitors.

2. Remote Interrogation

The goal of the remote communication as described in (Commission, 2004) and (Commission, 2011) determines that the tachograph shall be equipped with a remote communication functionality that shall enable agents of the competent control authorities to read tachograph information from passing commercial vehicles by using remote communication equipment. This equipment is called the Remote early detection communication reader. It is important to comprehend that this functionality is intended to serve only as a pre-filter in order to select vehicles for closer inspection, and it does not replace the formal inspection process as determined in the provisions of Regulation (EU) No. 165/2014 (recital 9 in the preamble of this regulation (Commission, 2004), stating that remote communication between the tachograph and control authorities for roadside control purposes facilitates targeted roadside checks).

The regulation requests a very precise set of data, which is described here:

1. the *latest security breach attempt*. This is the latest security breach attempt recorded by the system, which gives a clear indication that a malicious entity has tampered with the system.
2. the *longest power supply interruption*. This is the longest interruption of the power supply, which may give indication on the potential manipulation of the tachograph by a malicious entity.
3. *sensor fault*, which gives indication on a fault of the motion sensor or the Global Navigation Satellite System (GNSS) sensor.
4. *motion data error*, which provides an indication of potential errors in the processing of the data from the motion sensor.
5. *vehicle motion conflict*, which indicates a discrepancy between the position or speed recorded by the motion sensor, the GNSS sensor or any other sensor used by the manufacturers. This event is an important information, which could be used to detect malicious tampering of the tachograph.
6. *driving without a valid card*, which gives a direct information that there is non compliance to the regulation.
7. *card insertion while driving*,
8. *time adjustment data*. This information identifies the moment when the tachograph needed to readjust the time. This information is also useful to detect a malicious activity because tampering with the time of the tachograph could give an economic benefit to a criminal driver or company.
9. *calibration data including the dates of the two latest calibrations*. The calibration data is needed to ensure that calibration time is consistent with the operation of the vehicle and the tachograph.
10. *vehicle registration number*, which identifies the vehicle and the Vehicle Unit (VU).
11. *speed recorded by the tachograph*. This information cannot be directly used to detect infringements against speed limits, but it can be used to detect malfunctions (intentional or un-intentional) in the speed calculation and recording of the tachograph. We note that the speed can be calculated both through the GNSS receiver and the odometer.

These data has been defined in the regulatory process for filtering purpose and to highlight the possibility of tampering or malfunction (e.g., the events field). The evaluation for the conformation to the regulation is quite a complex process, which require access to most of the recorded data of the tachograph in the last 28 days of operation. This evaluation could not be implemented and executed on the road, but it requires the stop of the vehicle, the download of data and the analysis of the data with special software. On the other side, the current version of the digital tachograph has been subject to various attacks to undermine the integrity of the collected (i.e., from the motion sensor) data or the recorded data as described in the introduction section ??.

In the current version of the digital tachograph, a law enforcer must stop the vehicle to check the presence of events, which could indicate malfunction or tampering. In the new version of the Smart Tachograph, the remote communication through the CEN-DSRC at 5.8 GHz can provide the list of outstanding events and other useful information, which can alert the law enforcer for potential misuse of the smart tachograph application.

The typical scenarios where the CEN-DSRC can be used are shown in figure 1, where the smart tachograph present in the commercial vehicle can be interrogated through CEN-DSRC either from a mobile vehicle or a roadside equipment system. This proof of concept has been mainly designed for the roadside use case, even it can be easily adapted to the mobile vehicle as well.

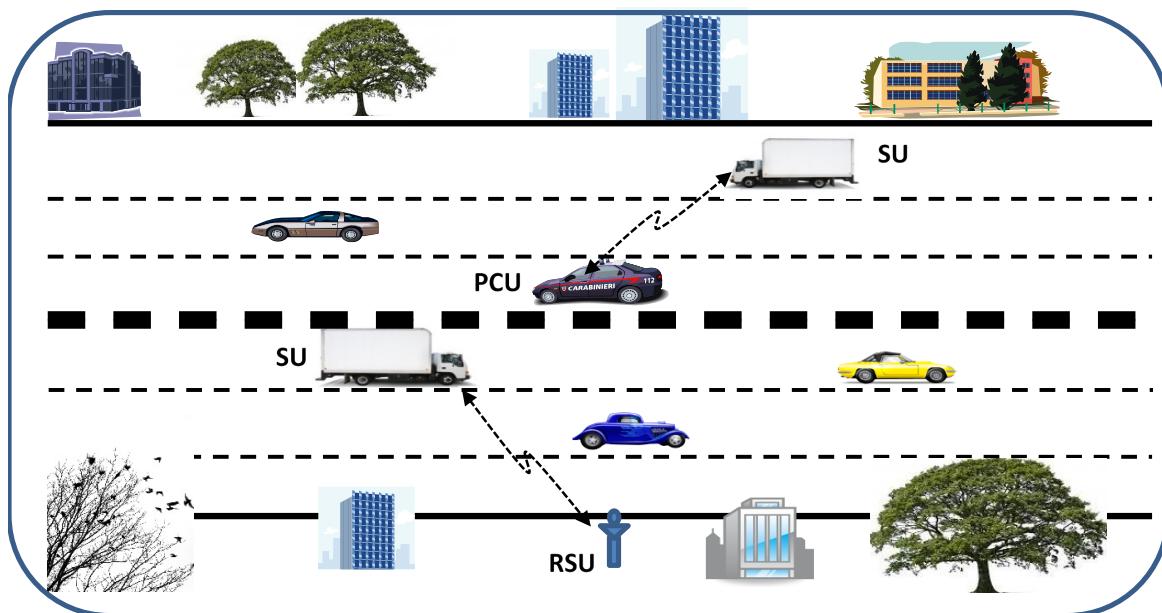


Figure 1: Typical scenarios for the application of CEN-DSRC to the smart tachograph

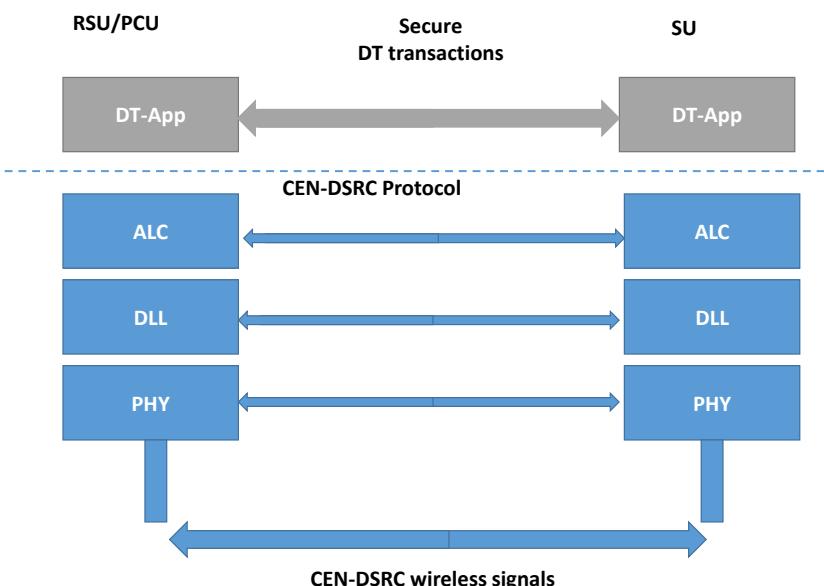


Figure 2: remote communication architecture

In the rest of this section, we describe in detail how the remote communication function

is implemented in the smart tachograph. The overall architecture of the smart tachograph for the specific aspects of remote communication are described in figure 3.

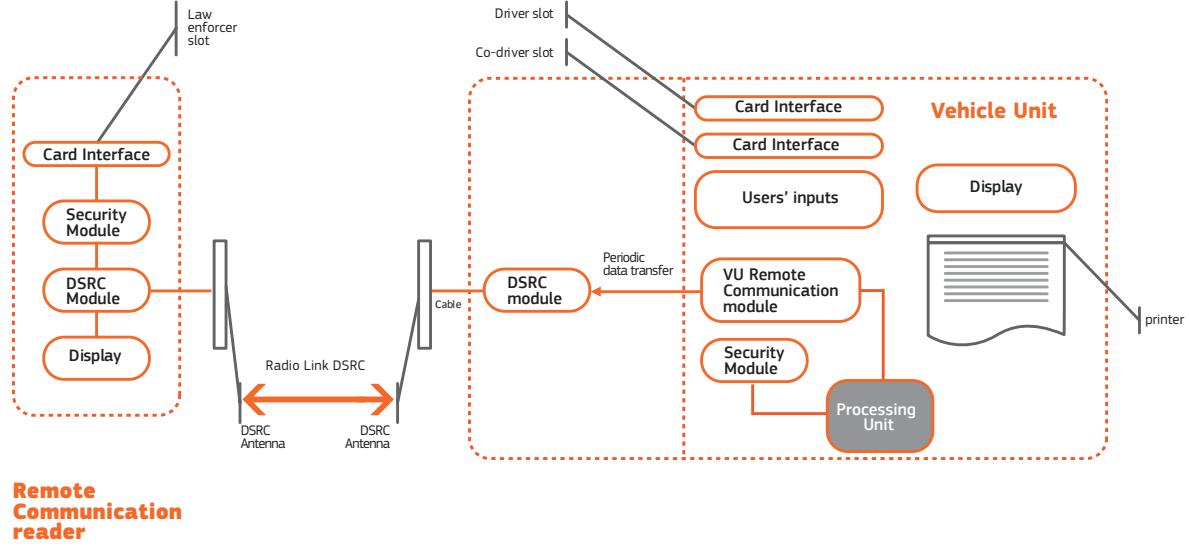


Figure 3: remote communication architecture

The system on the left of figure 3 represent the reader system used by the law enforcer on the road or from a mobile police vehicle. The law enforcer activates the system by inserting his/her smartcard. This operation is needed to authenticate the law enforcer. After authentication, a law enforcer can interrogate a commercial vehicle by requesting the remote communication reader to issue a wireless CEN-DSRC challenge to the DSRC On Board Unit (OBU) in the commercial vehicle by using the CEN-DSRC standard at 5.8 GHz. The challenge specifies the applications for which the law enforcer requests the data. Each application has its own id. At the moment, only one application is defined: the Smart Tachograph, but other applications can also be implemented in the DSRC OBU. In fact, Appendix 14 of the technical specifications of the Smart Tachograph specifies the support for another regulated application: the Weighing and Dimensions directive, which also uses the CEN-DSRC 5.8 GHz. For this application, the use of CEN-DSRC is explicitly requested. In the future the same system could support other applications. Each application is identified by the related application id. The wireless challenge contains the list of application id to which the DSRC OBU must provide data if it is available.

Upon the reception of the wireless challenge, the DSRC OBU checks the application id and verifies if the related data is present in the system. The process of storing the data in the DSRC OBU is described in the following paragraph. The VU of the Smart Tachograph periodically (every 60 seconds) verifies the content of its memory to verify the presence of new events, the update of status information, the current values of specific parameters and so on. The VU generates the data identified before in this section from its current memory. Before the information is stored in the DSRC-VU module, a message authentication code is appended to ensure its integrity and authenticity. Symmetric keys are used to secure the data to be sent in response to the wireless challenge request. This process does also implement authentication of the commercial vehicles, because it embeds the vehicle registration number of the commercial vehicle and the serial number of the VU. The authentication is needed to ensure that the VU has not been replaced by another VU, which could be used to provide false information.

In this way, end-to-end security is implemented, where the two ends are respectively the remote reader used by the law enforcer and the vehicle unit with its cryptomodule. It was a design decision to implement a new end-to-end security mechanism rather than leaning on the proprietary security solutions already defined in the electronic tolling standards ((?)). There were two reasons for this: the first one is for future upgradeability of the system for future wireless communication systems. In the long foreseen lifetime of the smart tachograph (15 years or more), new communication technologies could be developed. With end to end authentication, the CEN-DSRC at 5.8 GHz could be easily be replaced without an significant impact on the rest of the smart tachograph system. The second reason is that an harmonized security standard at European level must be defined, which would take time

to develop, while the technical specifications needed to be finalized in a specific timeframe.

A specific workflow for the wireless CEN-DSRC has been defined (details are in (Commission, 2011)). This workflow is derived from existing applications like the electronic tolling, but it has been improved and made more efficient for the specific needs of the smart tachograph (e.g., the type and format of data to be transmitted and the absence of fields, which are specific for electronic tolling). An unique workflow was implemented also because there is not an unique electronic tolling workflow across Europe. Countries like Italy has a different electronic tolling implementation in comparison to country like Germany or France. While the application layer workflow was specifically designed for the smart tachograph, the definition of the physical layer was strictly based on the standard EN 13372.

This was done for various reasons:

1. The need to use mass market electronic components, with a wide market deployment.
2. to be conformant to the radio frequency spectrum regulations already valid for the EN 12253.
3. to reuse the existing testing standards already defined for EN 12253.

In this way, the hardware implementations of the CEN-DSRC available in the market for electronic tolling, could also used for the smart tachograph with a new version of the software, implementing the challenge-response interaction between the reader and the OBU in the commercial vehicle. This provides the advantage of decreased costs for the smart tachograph.

A simplified schema of the overall workflow and layered stack of the CEN-DSRC communication is shown in figure fig:flowdsrc. The higher layer is the application layer where the information is exchanged. The application layer core (ALC) directly interacts with the DT Application for communications. The lower layers of the network are the the physical layer (PHY) and the data link layer (DLL) which implements the medium access control (MAC).

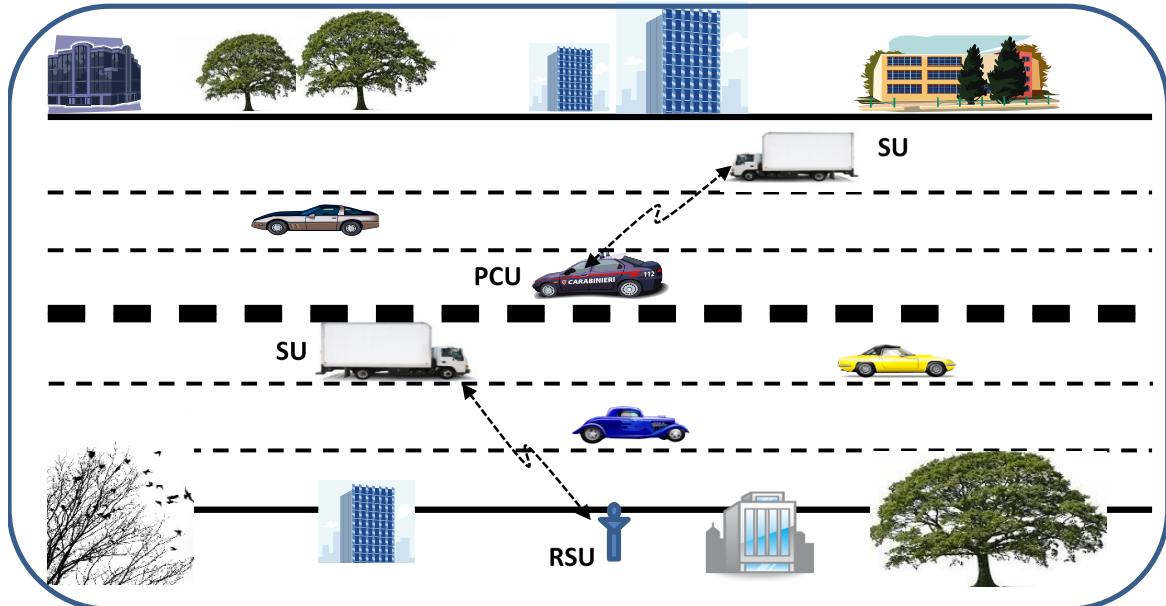


Figure 4: Layered protocol for CEN-DSRC in the new version of the Digital Tachograph

As described previously, the application layer and parts of the ALC have been implemented in this prototype. All the other sub-layers are implemented with the same hardware components of the european electronic tolling.

3. Proof-of-Concept

In 2016, a contract was given to Q-FREE to implement the new workflow of the remote interrogation of the smart tachograph on the basis of the technical specifications defined in (Commission, 2011).

The prototype was composed by a remote reader and four CEN-DSRC tags to be installed in the commercial vehicles.

Q-Free implemented the proof-of-concept with the following hardware and software components:

- the hardware systems already developed for electronic tolling: RSE650 for the remote reader equipment (one unit) and OBU615 for the OBU (four units). The RSE650 was provided together with auxiliary components and power supply as for the list of items provided in figure 5.
- the software was implemented on the basis of the specifications in (Commission, 2011). This was the main part of the work.

Model	Description	Part Number	Count
Q-Free ® RSE650	CEN DSRC Transceiver	A24A0MR8	1
Q-Free ® ACC650	RSE650 Connection Box	A1400002	1
Q-Free ® ACC652	RSE650 Generic Bracket (round tube, 2 angles, w/ ACC650 holder)	B11BGMR8	1
Planet POE-161	Planet Single POE Injector	A3CE0006	1
	CAT5 Cable (5m)		1
	Mini Inline Coupler		1
Q-Free ® RSE622	Handheld DSRC Transceiver	A24A0MRB	1

Figure 5: List of hardware components for the reader system

The overall system was used during the Open Day in May 2016. Figure 6 shows the stand where the CEN-DSRC prototype reader was positioned during the Open Day. Behind the reader, the representatives of the Italian road enforcement (who are going to use the reader) can be seen. The participation of the Italian police force was quite useful to receive a feedback, which was quite positive on the new tool.

Figure 7 shows the images of the CEN-DSRC prototype while it was used to interrogate the CEN-DSRC OBU installed in a commercial vehicle provided by Continental for the JRC Open Day 2016. The prototype was able to interrogate and process the data many times a second. A spectrum analyzer was also used during the open day to evaluate the levels of transmitted Radio Frequency (RF) power by the CEN-DSRC system, which was compliant to the RF spectrum regulations.

The manual of use of the CEN-DSRC system is provided in the Annex 1. The manual of the software components of the CEN-DSRC system, their structure and how the software can be used is described in Annex 2.



Figure 6: Image of the CEN-DSRC prototype reader during the JRC Open Day 2016

4. Conclusions

This report describes the prototype, which implements the remote interrogation function for the smart tachograph based on the CEN-DSRC standard. The prototype has been evaluated and tested by the JRC and it satisfies the operational requirements of the smart tachograph. It has been used and shown at the JRC Open Day in May 2016 in collaboration with the Italian law enforcement (i.e., Italian Polizia Stradale), which will be one of users of this tool together with their other European colleagues. The remote interrogation function will allow a more efficient filtering of the commercial vehicles on the road for the smart tachograph regulation.



Figure 7: Image of the CEN-DSRC prototype reader interrogating the CEN-DSRC OBU installed in a Continental truck

5. Annex 1 - Manual of use of the remote interrogator for the smart tachograph



Tachograph Getting Started Guide

DocumentID:	2016-642-Tachograph-001
Revision number:	1
Edited by:	Truls Gulbrandsen
Checked by:	Not checked
Approved by:	Not approved

Copyright © 2016 Q-Free ASA

All rights reserved. No part of this publication may be reproduced or distributed, without the prior consent of Q-Free ASA.

Revision history:

NUMBER	DATE	DESCRIPTION	NAME
v1.0	2016-03-01	First revision of document	TMAG

Contents

1	Introduction	3
2	Kit	3
2.1	Hardware	3
2.2	Software	3
2.3	Tags	3
3	QF Packages Installation	4
4	RSE650 Configuration	4
4.1	Change IP Address	4
4.2	Advanced Test Configuration	4
5	Transaction Model Installation	4
6	RSE650 Link Testing	4
7	RSE650 Transaction Logging	5

References

USER DOCUMENTATION

- [1] 2009-645-RSE650-001 RSE650 installation manual
- [2] 2011-642-RSE650-mdmrapp-001 User Manual

Glossary**DSRC**

Dedicated Short-Range Communications

OBU

On-Board Unit

RTM

Remote Tachograph Monitoring

1 Introduction

This guide tells you how to quickly get started using the tachograph kit.

2 Kit

2.1 Hardware

Table 1 shows the hardware included in the kit.

Table 1: Hardware

Model	Description	Part Number	Count
Q-Free ® RSE650	CEN DSRC Transceiver	A24A0MR8	1
Q-Free ® ACC650	RSE650 Connection Box	A1400002	1
Q-Free ® ACC652	RSE650 Generic Bracket (round tube, 2 angles, w/ ACC650 holder)	B11BGMR8	1
Planet POE-161	Planet Single POE Injector	A3CE0006	1
	CAT5 Cable (5m)		1
	Mini Inline Coupler		1
Q-Free ® RSE622	Handheld DSRC Transceiver	A24A0MRB	1

2.2 Software

Table 2 shows the software used included in the kit.

Table 2: Software

Name	Description	Part Number	Count
MDMRAPP SW RSE650	Multipurpose application for RSE650	A9E65003	1

2.3 Tags

Table 3 shows the DSRC tags included in the kit.

Table 3: Tags

Part Number	Description	Customer Product Specification	Count
A2Rxxxx1	OBUS615 w/ static RTM data	a2rxxxx1_tachograph.xml	1
A2Rxxxx2	OBUS615 w/ static RTM data	a2rxxxx2_tachograph.xml	1

Table 3: (continued)

Part Number	Description	Customer Product Specification	Count
A2Rxxxx3	OBU615 w/ static RTM data	a2rxxxx3_tachograph.xml	1
A2R7AC01	OBU611 w/ dynamic RTM data	a2r7ac01_tachograph.xml	1

3 QF Packages Installation

Table 4 shows the additional packages that have been installed on your RSE650.

Table 4: QF Packages

Name	Version	Revision
qfree-rse650-libqa1c	1.0.0-04	80269
qfree-rse650-libdsrcl2	1.1.0-04	80271
qfree-rse650-app-mdmrapp	7.01-09	80714

See [1] section 5.1.4 for how to check that the correct packages have been installed on your RSE650.

4 RSE650 Configuration

Your RSE650 has been pre-configured as a single-gantry reader.

4.1 Change IP Address

The default IP address of the RSE650 is *192.168.127.81*. See [1] section 5.1.1 for how to change the IP address.



Note

You must be on the same subnet (e.g. *192.168.127.xyz*) as the reader to be able to reach the reader.

4.2 Advanced Test Configuration

See tip box on the bottom of page 5 and section 10 in [2] for tips on how to decrease time between reads of the same OBU in a test scenario.

5 Transaction Model Installation

6 RSE650 Link Testing

See [1] section 5.1.3 for how to perform link testing.

7 RSE650 Transaction Logging

See [2] section 2.1 for how to access the transaction log. Clicking on the *xml* link under the *Layer7XmlLog* column allows you to see the content of the OBU transaction.

See [2] sections 2.2, 3 and 4.1-4.3 for how to implement logging to an external HTTP server.

6. Annex 2 - Manual of use of the development kit

Q-Free Tachograph Software Development Kit

Version 1.0.0

Contents

1 Q-Free Tachograph Software Development Kit	2
1.1 Platform Support	2
1.2 Tachograph Library Dependencies	2
1.3 Build Instructions	2
1.3.1 Example Tachograph Application	2
1.3.2 Tachograph Library	3
2 Module Index	4
2.1 Modules	4
3 Namespace Index	5
3.1 Namespace List	5
4 Class Index	6
4.1 Class List	6
5 File Index	7
5.1 File List	7
6 Module Documentation	8
6.1 Tachograph	8
6.1.1 Detailed Description	8
6.2 QFTOP	9
6.2.1 Detailed Description	10
6.2.2 Macro Definition Documentation	10
6.2.2.1 MAXIMUMQFTOPFRAMESIZE	10
6.2.3 Enumeration Type Documentation	10
6.2.3.1 qftop_cmd_type_t	10
6.2.3.2 qftop_Types	11
6.2.4 Function Documentation	11
6.2.4.1 operator<<	11
6.2.4.2 operator>>	11
6.2.4.3 operator<<	11

6.2.4.4	operator<<	11
6.2.4.5	operator<<	12
6.2.4.6	operator<<	12
6.2.4.7	print_message	12
6.2.4.8	qftop_ack	12
6.2.4.9	qftop_addParameter	12
6.2.4.10	qftop_addToCRC	13
6.2.4.11	qftop_application	13
6.2.4.12	qftop_buildMessage	14
6.2.4.13	qftop_clear	15
6.2.4.14	qftop_extractMessage	16
6.2.4.15	qftop_parse	16
6.3	Serial Port	18
6.4	Utility	19
6.4.1	Detailed Description	19
6.5	Example	20
6.5.1	Detailed Description	20
6.5.2	Function Documentation	20
6.5.2.1	main	20
7	Namespace Documentation	21
7.1	qftop Namespace Reference	21
7.2	string_extra Namespace Reference	22
7.2.1	Function Documentation	22
7.2.1.1	has_prefix	22
7.2.1.2	hex_to_bytes	22
7.2.1.3	split_on_whitespace	23
7.3	tachograph Namespace Reference	23
8	Class Documentation	24
8.1	tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH > Class Template Reference	24
8.1.1	Detailed Description	24
8.1.2	Constructor & Destructor Documentation	24
8.1.2.1	application	25
8.1.3	Member Function Documentation	25
8.1.3.1	read_rtm_data	25
8.1.3.2	write_rtm_data	25
8.2	qftop::application Class Reference	25
8.2.1	Detailed Description	26
8.2.2	Constructor & Destructor Documentation	26

8.2.2.1	application	26
8.2.3	Member Function Documentation	26
8.2.3.1	push_message	26
8.2.3.2	send_read_without_cred	26
8.2.3.3	send_write_without_cred	27
8.2.3.4	start_polling	27
8.2.3.5	stop_polling	27
8.3	qftop::attribute_list Struct Reference	27
8.3.1	Detailed Description	28
8.3.2	Member Function Documentation	28
8.3.2.1	to_bytes	28
8.3.3	Friends And Related Function Documentation	28
8.3.3.1	operator<<	28
8.3.4	Member Data Documentation	28
8.3.4.1	attribute_id	28
8.3.4.2	attribute_value	29
8.3.4.3	container_type	29
8.4	qfTopMSG::crc Union Reference	29
8.4.1	Detailed Description	29
8.4.2	Member Data Documentation	29
8.4.2.1	bytes	29
8.4.2.2	word	29
8.5	qftop::get_control Struct Reference	29
8.5.1	Detailed Description	30
8.5.2	Friends And Related Function Documentation	30
8.5.2.1	operator<<	30
8.5.3	Member Data Documentation	30
8.5.3.1	"@5	30
8.5.3.2	action	30
8.5.3.3	byte	30
8.5.3.4	has_attribute_list	30
8.5.3.5	has_credentials	30
8.5.3.6	has_iid	30
8.5.3.7	mode	30
8.6	qfTopMSG::header Union Reference	30
8.6.1	Detailed Description	31
8.6.2	Member Data Documentation	31
8.6.2.1	"@16	31
8.6.2.2	bytes	31
8.6.2.3	frameType	31

8.6.2.4	length	31
8.6.2.5	preamble	31
8.6.2.6	sequence	31
8.6.2.7	status	31
8.6.2.8	syn	31
8.7	qfTopMSG::pdu Union Reference	32
8.7.1	Detailed Description	32
8.7.2	Member Data Documentation	32
8.7.2.1	"@18	32
8.7.2.2	bytes	32
8.7.2.3	messageType	32
8.7.2.4	PARAMETERS	32
8.8	qfTopMessage Struct Reference	32
8.8.1	Detailed Description	33
8.8.2	Member Data Documentation	33
8.8.2.1	head_read	33
8.8.2.2	head_start	33
8.8.2.3	message	33
8.8.2.4	message_length	33
8.8.2.5	msg	33
8.8.2.6	pre_escape	33
8.9	qfTopMSG Struct Reference	33
8.9.1	Detailed Description	34
8.9.2	Member Data Documentation	34
8.9.2.1	CRC	34
8.9.2.2	CRC_REC	34
8.9.2.3	HEADER	34
8.9.2.4	PDU	34
8.10	qftop::read_response_callback Class Reference	34
8.10.1	Detailed Description	34
8.10.2	Member Function Documentation	34
8.10.2.1	on_error	34
8.10.2.2	on_success	34
8.11	qftop::read_without_cred Struct Reference	35
8.11.1	Detailed Description	35
8.11.2	Member Function Documentation	35
8.11.2.1	to_bytes	35
8.11.3	Friends And Related Function Documentation	36
8.11.3.1	operator<<	36
8.11.4	Member Data Documentation	36

8.11.4.1 "@9	36
8.11.4.2 attribute_count	36
8.11.4.3 attribute_list	36
8.11.4.4 bytes	36
8.11.4.5 control	36
8.11.4.6 element_id	36
8.11.4.7 fragment_header	36
8.11.4.8 length	36
8.12 qftop::read_without_cred_response Struct Reference	36
8.12.1 Detailed Description	37
8.12.2 Friends And Related Function Documentation	37
8.12.2.1 operator<<	37
8.12.3 Member Data Documentation	37
8.12.3.1 attribute_count	37
8.12.3.2 attribute_id	37
8.12.3.3 attribute_value	37
8.12.3.4 bytes	37
8.12.3.5 container_id	37
8.12.3.6 control	37
8.12.3.7 element_id	38
8.12.3.8 fragment_header	38
8.12.3.9 header	38
8.12.3.10 length	38
8.13 qftop::set_control Struct Reference	38
8.13.1 Detailed Description	38
8.13.2 Friends And Related Function Documentation	38
8.13.2.1 operator<<	38
8.13.3 Member Data Documentation	39
8.13.3.1 "@1	39
8.13.3.2 action	39
8.13.3.3 byte	39
8.13.3.4 fill	39
8.13.3.5 has_credentials	39
8.13.3.6 has_iid	39
8.13.3.7 mode	39
8.14 qftop::tty Class Reference	39
8.14.1 Detailed Description	39
8.14.2 Constructor & Destructor Documentation	40
8.14.2.1 tty	40
8.14.3 Member Function Documentation	41

8.14.3.1	send_message	41
8.14.3.2	start_reading	41
8.14.3.3	stop_reading	42
8.15	qftop::write_response_callback Class Reference	42
8.15.1	Detailed Description	42
8.15.2	Member Function Documentation	42
8.15.2.1	on_error	42
8.15.2.2	on_success	42
8.16	qftop::write_without_cred Struct Reference	43
8.16.1	Detailed Description	43
8.16.2	Member Function Documentation	43
8.16.2.1	to_bytes	43
8.16.3	Friends And Related Function Documentation	44
8.16.3.1	operator<<	44
8.16.4	Member Data Documentation	44
8.16.4.1	attribute_count	44
8.16.4.2	attribute_list	44
8.16.4.3	control	44
8.16.4.4	element_id	44
8.16.4.5	fragment_header	44
8.16.4.6	length	44
9	File Documentation	45
9.1	dox/mainpage.dox File Reference	45
9.2	tachographApp/src/main/cpp/main.cpp File Reference	45
9.3	main.cpp	45
9.4	tachographApp/src/main/cpp/string_extra.cpp File Reference	47
9.5	string_extra.cpp	48
9.6	tachographApp/src/main.hpp/string_extra.hpp File Reference	49
9.7	string_extra.hpp	50
9.8	tachographLib/src/main/c/qftop_client.c File Reference	50
9.8.1	Function Documentation	51
9.8.1.1	qftop_copyIn	51
9.8.1.2	qftop_copyOut	52
9.9	qftop_client.c	52
9.10	tachographLib/src/main/cpp/qftop_application.cpp File Reference	55
9.11	qftop_application.cpp	55
9.12	tachographLib/src/main/cpp/qftop_tty.cpp File Reference	59
9.13	qftop_tty.cpp	60
9.14	tachographLib/src/main/h/qftop_client.h File Reference	61

9.15 qftop_client.h	63
9.16 tachographLib/src/main.hpp/qftop_application.hpp File Reference	64
9.17 qftop_application.hpp	66
9.18 tachographLib/src/main.hpp/qftop_tty.hpp File Reference	69
9.19 qftop_tty.hpp	70
9.20 tachographLib/src/main.hpp/tachograph.hpp File Reference	70
9.21 tachograph.hpp	71
9.22 tachographLib/src/main/ipp/tachograph.ipp File Reference	72
9.23 tachograph.ipp	73
Index	75

Chapter 1

Q-Free Tachograph Software Development Kit

This software development kit (SDK) consists of a library and an example application.

The library provides an interface for reading and writing the RTM data element in an OBU611 tag.

The example application demonstrates the following:

- Communication with OBU611 (QFTOP over UART)
 1. Open serial port
 2. Write QFTOP message(s) to serial port
 3. Read QFTOP messages(s) from serial port
- Tachograph Client
 1. Read RTM data
 2. Write RTM data

1.1 Platform Support

Both x86 and x64 is supported.

1.2 Tachograph Library Dependencies

Compiler:

- g++-4.9
- clang-3.7

Compile Dependencies:

- libboost1.55-all-dev
- g++-4.9-multilib (if compiling x64 binaries from i386)

1.3 Build Instructions

1.3.1 Example Tachograph Application

g++:

```
./gradlew clean tachographAppX64ReleaseExecutable
```

clang:

```
./gradlew clean tachographAppX64ReleaseExecutable -PuseClang
```

1.3.2 Tachograph Library

g++:

```
./gradlew clean tachographLibX64ReleaseExecutable
```

clang:

```
./gradlew clean tachographLibX64ReleaseExecutable -PuseClang
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Tachograph	8
QFTOP	9
Serial Port	18
Utility	19
Example	20

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

qftop	21
string_extra	22
tachograph	23

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH >	24
Tachograph Client	
qftop::application	25
QFTOP Client	
qftop::attribute_list	27
Attribute List	
qfTopMSG::crc	29
qftop::get_control	29
Get Control Data	
qfTopMSG::header	30
qfTopMSG::pdu	32
qfTopMessage	32
State holder of QFTop Message plus state info when parsing	
qfTopMSG	33
Structure of QFTop Message	
qftop::read_response_callback	34
Read Response Callback Interface	
qftop::read_without_cred	35
Read Without Credentials Request	
qftop::read_without_cred_response	36
Read Without Credentials Response	
qftop::set_control	38
Set Control Data	
qftop::tty	39
qftop::write_response_callback	42
Write Response Callback Interface	
qftop::write_without_cred	43
Write Without Credentials Request	

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

tachographApp/src/main/cpp/ main.cpp	45
tachographApp/src/main/cpp/ string_extra.cpp	48
tachographApp/src/main.hpp/ string_extra.hpp	50
tachographLib/src/main/c/ qftop_client.c	52
tachographLib/src/main/cpp/ qftop_application.cpp	55
tachographLib/src/main/cpp/ qftop_tty.cpp	60
tachographLib/src/main/h/ qftop_client.h	63
tachographLib/src/main.hpp/ qftop_application.hpp	66
tachographLib/src/main.hpp/ qftop_tty.hpp	70
tachographLib/src/main.hpp/ tachograph.hpp	71
tachographLib/src/main/ipp/ tachograph.ipp	73

Chapter 6

Module Documentation

6.1 Tachograph

Classes

- class `tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH >`
Tachograph Client.

6.1.1 Detailed Description

Elements related to the tachograph client.

6.2 QFTOP

Classes

- struct `qftop::set_control`
Set Control Data.
- struct `qftop::attribute_list`
Attribute List.
- struct `qftop::write_without_cred`
Write Without Credentials Request.
- struct `qftop::get_control`
Get Control Data.
- struct `qftop::read_without_cred`
Read Without Credentials Request.
- struct `qftop::read_without_cred_response`
Read Without Credentials Response.
- class `qftop::write_response_callback`
Write Response Callback Interface.
- class `qftop::read_response_callback`
Read Response Callback Interface.
- class `qftop::application`
QFTOP Client.
- struct `qfTopMSG`
Structure of QFTop Message.
- struct `qfTopMessage`
State holder of QFTop Message plus state info when parsing.

Macros

- #define `MAXIMUMQFTOPFRAMESIZE` 200

Enumerations

- enum `qftop_cmd_type_t` {
 `QFTOP_ECHO_REQ` = 0x00, `QFTOP_ECHO_RESP` = 0x80, `QFTOP_ACK` = 0x01, `QFTOP_NACK` = 0x02,
 `QFTOP_MMI_REQ` = 0x30, `QFTOP_INIT_NOTIFICATION` = 0x31, `QFTOP_TRANSPIRESP` = 0x33, `QFTOP_TRANSPIREQ` = 0x34,
 `QFTOP_REGISTER_APP_REQ` = 0x36, `QFTOP_REGISTER_APP` = 0x37, `QFTOP_TEST_REQ` = 0x38, `QFTOP_TEST_RESP` = 0x39,
 `QFTOP_PERS_REQ` = 0x3A, `QFTOP_PERS_RESP` = 0x3B, `QFTOP_DSRC_L7_REQ` = 0x3C, `QFTOP_DSRC_L7_RESP` = 0x3D,
 `QFTOP_TRACE_LOG_REQ` = 0xF0, `QFTOP_TRACE_LOG_RESP` = 0xF1 }

Types of QFTOP messages.
- enum `qftop_Types` {
 `Application` = 0, `ACK` = 1, `NACK` = 2, `dsrc_l7_req` = 0x3C,
 `dsrc_l7_resp` = 0x3D, `crc_init` = 0x6363, `qftop_preamble` = 0xB5, `maximumQFTOPFrameSize` = `MAXIMUMQFTOPFRAMESIZE` }

Types of QFTOP messages.

Functions

- void `qftop::print_message` (std::ostream &out, const `qfTopMessage` *rhs)

Print QFTOP message to stream.
- std::ostream & `qftop::operator<<` (std::ostream &out, const set_control &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const attribute_list &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const write_without_cred &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const get_control &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const read_without_cred &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const read_without_cred_response &rhs)
- int `qftop_parse` (struct `qfTopMessage` *msg, uint8_t cr)

Function to parse a new byte into a message being received.
- uint16_t `qftop_extractMessage` (struct `qfTopMessage` *msg_out, struct `qfTopMessage` *msg_in)

Function to build an internal message based on bytes in another message.
- void `qftop_addToCRC` (struct `qfTopMessage` *msg, uint8_t ch)

Modify crc calculation for a new byte.
- void `qftop_addParameter` (struct `qfTopMessage` *msg, uint8_t p)

Function for adding a single parameter to a QFTop message.
- void `qftop_clear` (struct `qfTopMessage` *msg)

Convenience function to zero / reset a message.
- void `qftop_application` (struct `qfTopMessage` *msg)

Convenience function to initialise an application message (dsrc req)
- void `qftop_ack` (struct `qfTopMessage` *msg)

Convenience function to build an ACK message.
- unsigned int `qftop_buildMessage` (struct `qfTopMessage` *msg)

Function to build a byte stream ready for transmission based on a message.

6.2.1 Detailed Description

Elements related to QFTOP

6.2.2 Macro Definition Documentation

6.2.2.1 #define MAXIMUMQFTOPFRAMESIZE 200

Definition at line 12 of file `qftop_client.h`.

6.2.3 Enumeration Type Documentation

6.2.3.1 enum qftop_cmd_type_t

Types of QFTOP messages.

Enumerator

```

QFTOP_ECHO_REQ
QFTOP_ECHO_RESP
QFTOP_ACK
QFTOP_NACK
QFTOP_MMI_REQ
QFTOP_INIT_NOTIFICATION

```

```
QFTOP_TRANS_P_RESP
QFTOP_TRANS_P_REQ
QFTOP_REGISTER_APP_REQ
QFTOP_REGISTER_APP
QFTOP_TEST_REQ
QFTOP_TEST_RESP
QFTOP_PERS_REQ
QFTOP_PERS_RESP
QFTOP_DSRC_L7_REQ
QFTOP_DSRC_L7_RESP
QFTOP_TRACE_LOG_REQ
QFTOP_TRACE_LOG_RESP
```

Definition at line 19 of file [qftop_client.h](#).

6.2.3.2 enum qftop_Types

Types of QFTOP messages.

Enumerator

```
Application
ACK
NACK
dsrc_l7_req
dsrc_l7_resp
crc_init
qftop_preamble
maximumQFTOPFrameSize
```

Definition at line 42 of file [qftop_client.h](#).

6.2.4 Function Documentation

6.2.4.1 std::ostream & qftop::operator<< (std::ostream & out, const set_control & rhs)

Definition at line 22 of file [qftop_application.cpp](#).

6.2.4.2 std::ostream & qftop::operator<< (std::ostream & out, const attribute_list & rhs)

Definition at line 30 of file [qftop_application.cpp](#).

6.2.4.3 std::ostream & qftop::operator<< (std::ostream & out, const write_without_cred & rhs)

Definition at line 41 of file [qftop_application.cpp](#).

6.2.4.4 std::ostream & qftop::operator<< (std::ostream & out, const get_control & rhs)

Definition at line 50 of file [qftop_application.cpp](#).

6.2.4.5 `std::ostream & qftop::operator<< (std::ostream & out, const read_without_cred & rhs)`

Definition at line 58 of file [qftop_application.cpp](#).

6.2.4.6 `std::ostream & qftop::operator<< (std::ostream & out, const read_without_cred_response & rhs)`

Definition at line 67 of file [qftop_application.cpp](#).

6.2.4.7 `void qftop::print_message (std::ostream & out, const qfTopMessage * rhs)`

Print QFTOP message to stream.

Parameters

in	<i>out</i>	output stream
in	<i>rhs</i>	QFTOP message to print

Definition at line 6 of file [qftop_application.cpp](#).

6.2.4.8 `void qftop_ack (struct qfTopMessage * msg)`

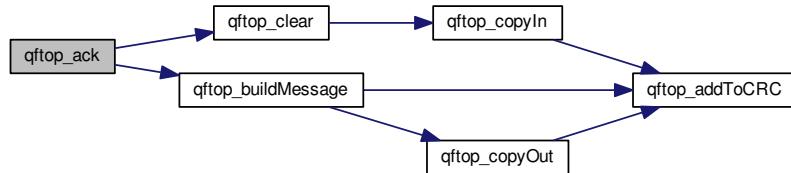
Convenience function to build an ACK message.

Parameters

in	<i>msg</i>	pointer to qfTopMessage
----	------------	---

Definition at line 109 of file [qftop_client.c](#).

Here is the call graph for this function:



6.2.4.9 `void qftop_addParameter (struct qfTopMessage * msg, uint8_t p)`

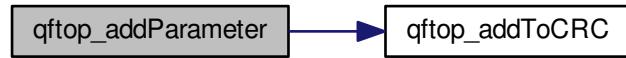
Function for adding a single parameter to a QFTop message.

Parameters

in	<i>msg</i>	pointer to qfTopMessage
in	<i>p</i>	Parameter byte being added

Definition at line 133 of file [qftop_client.c](#).

Here is the call graph for this function:



6.2.4.10 void qftop_addToCRC (struct qfTopMessage * msg, uint8_t ch)

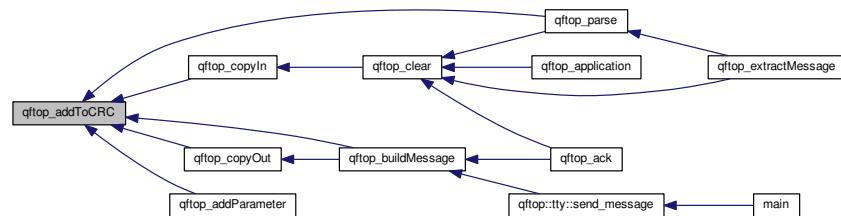
Modify crc calculation for a new byte.

Parameters

in	<i>msg</i>	pointer to qfTopMessage
in	<i>ch</i>	byte being added

Definition at line 8 of file [qftop_client.c](#).

Here is the caller graph for this function:



6.2.4.11 void qftop_application (struct qfTopMessage * msg)

Convenience function to initialise an application message (dsrc req)

Parameters

in	<i>msg</i>	pointer to qfTopMessage
----	------------	---

Definition at line 121 of file [qftop_client.c](#).

Here is the call graph for this function:



6.2.4.12 unsigned int qftop_buildMessage (struct qfTopMessage * msg)

Function to build a byte stream ready for transmission based on a message.

Parameters

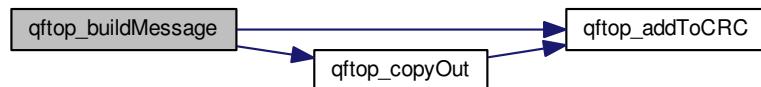
in	<i>msg</i>	pointer to qfTopMessage
----	------------	---

Returns

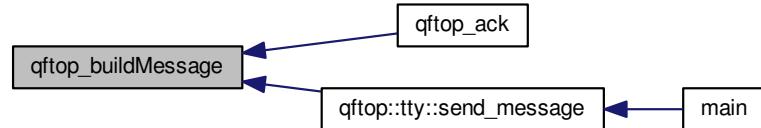
len final length of message

Definition at line 81 of file [qftop_client.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**6.2.4.13 void qftop_clear (struct qfTopMessage * msg)**

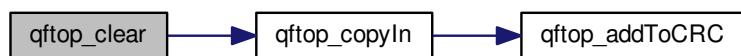
Convenience function to zero / reset a message.

Parameters

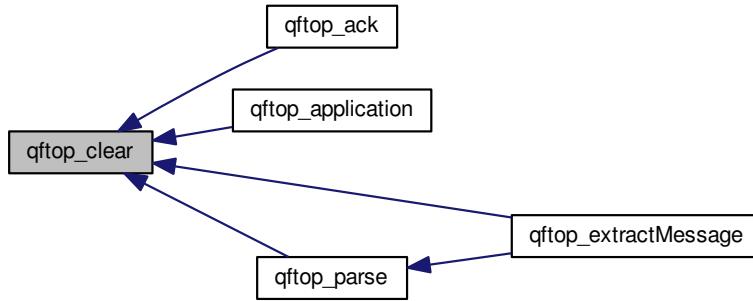
in	<i>msg</i>	pointer to qfTopMessage
----	------------	---

Definition at line 68 of file [qftop_client.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.4.14 uint16_t qftop_extractMessage (struct qfTopMessage * msg_out, struct qfTopMessage * msg_in)

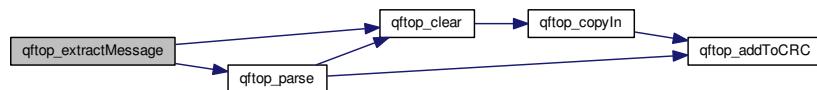
Function to build an internal message based on bytes in another message.

Parameters

in	<i>msg_in</i>	pointer to qfTopMessage
out	<i>msg_out</i>	pointer to qfTopMessage

Definition at line 138 of file [qftop_client.c](#).

Here is the call graph for this function:



6.2.4.15 int qftop_parse (struct qfTopMessage * msg, uint8_t cr)

Function to parse a new byte into a message being received.

This function is pretty ruthless. It will continue passing bytes till a message is complete or buffer overflow. If there's a protocol error the message will be restarted.

Parameters

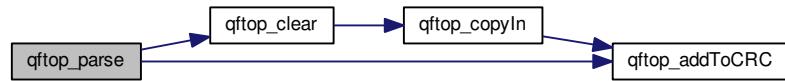
in	<i>msg</i>	pointer to qfTopMessage
in	<i>cr</i>	new byte

Returns

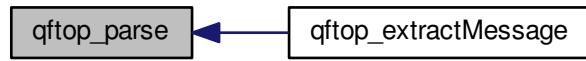
- > 0 once a complete message is received.
- = 0 if message is not yet complete
- < 0 on buffer overflow

Definition at line 150 of file [qftop_client.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.3 Serial Port

QFTOP over Serial Port.

QFTOP over Serial Port. Elements related to serial port communication.

6.4 Utility

String Helper Functions.

Namespaces

- [`string_extra`](#)

6.4.1 Detailed Description

String Helper Functions. Convenience methods.

6.5 Example

Example Tachograph Application.

Functions

- int `main ()`

6.5.1 Detailed Description

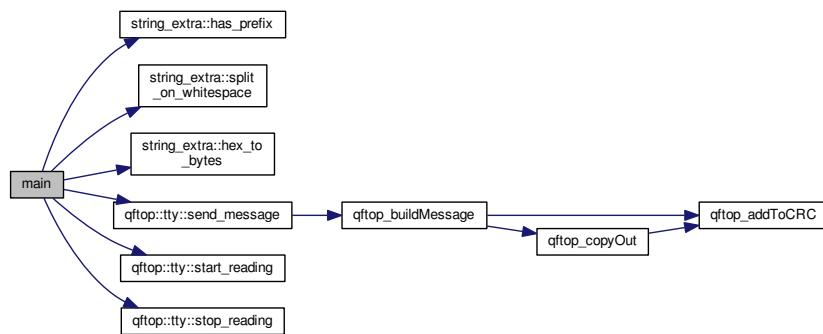
Example Tachograph Application. Example Tachograph Application

6.5.2 Function Documentation

6.5.2.1 int main ()

Definition at line 22 of file `main.cpp`.

Here is the call graph for this function:



Chapter 7

Namespace Documentation

7.1 qftop Namespace Reference

Classes

- struct `set_control`
Set Control Data.
- struct `attribute_list`
Attribute List.
- struct `write_without_cred`
Write Without Credentials Request.
- struct `get_control`
Get Control Data.
- struct `read_without_cred`
Read Without Credentials Request.
- struct `read_without_cred_response`
Read Without Credentials Response.
- class `write_response_callback`
Write Response Callback Interface.
- class `read_response_callback`
Read Response Callback Interface.
- class `application`
QFTOP Client.
- class `tty`

Functions

- void `print_message` (std::ostream &out, const `qfTopMessage` *rhs)
Print QFTOP message to stream.
- std::ostream & `operator<<` (std::ostream &out, const `set_control` &rhs)
- std::ostream & `operator<<` (std::ostream &out, const `attribute_list` &rhs)
- std::ostream & `operator<<` (std::ostream &out, const `write_without_cred` &rhs)
- std::ostream & `operator<<` (std::ostream &out, const `get_control` &rhs)
- std::ostream & `operator<<` (std::ostream &out, const `read_without_cred` &rhs)
- std::ostream & `operator<<` (std::ostream &out, const `read_without_cred_response` &rhs)

7.2 string_extra Namespace Reference

Functions

- bool [has_prefix](#) (std::string *string_to_check*, std::string *prefix*)
Check if string has prefix.
- std::vector< std::string > [split_on whitespace](#) (std::string *string_to_split*)
Split string on whitespace.
- std::vector< unsigned char > [hex_to_bytes](#) (std::string *hex_string*)
Convert hex string to bytes.

7.2.1 Function Documentation

7.2.1.1 bool string_extra::has_prefix (std::string *string_to_check*, std::string *prefix*)

Check if string has prefix.

Parameters

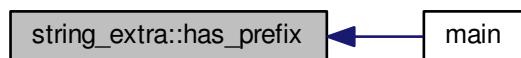
in	<i>string_to_check</i>	string that is checked for prefix
in	<i>prefix</i>	the prefix that is checked for

Returns

true if string has prefix, otherwise false

Definition at line 6 of file [string_extra.cpp](#).

Here is the caller graph for this function:



7.2.1.2 std::vector< unsigned char > string_extra::hex_to_bytes (std::string *hex_string*)

Convert hex string to bytes.

Parameters

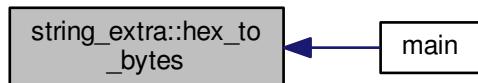
in	<i>hex_string</i>	hex string
----	-------------------	------------

Returns

bytes

Definition at line 20 of file [string_extra.cpp](#).

Here is the caller graph for this function:



7.2.1.3 std::vector< std::string > string_extra::split_on whitespace (std::string string_to_split)

Split string on whitespace.

Parameters

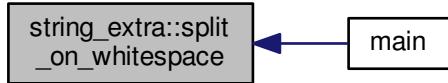
in	<i>string_to_split</i>	string to be split
----	------------------------	--------------------

Returns

the split string

Definition at line 10 of file [string_extra.cpp](#).

Here is the caller graph for this function:



7.3 tachograph Namespace Reference

Classes

- class [application](#)

Tachograph Client.

Chapter 8

Class Documentation

8.1 tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH > Class Template Reference

Tachograph Client.

```
#include <tachograph.hpp>
```

Public Member Functions

- `application` (std::shared_ptr< qftop::application > qftop_application_ptr, std::shared_ptr< std::ostream > output_stream_ptr)
Constructor.
- void `read_rtm_data` (std::function< void(std::array< unsigned char, TACHOGRAPH_PAYLOAD_LENGTH > tachograph_payload, std::array< unsigned char, DSRC_SECURITY_DATA_LENGTH > dsr_security_data)> callback)
read RTM data without credentials
- void `write_rtm_data` (std::array< unsigned char, TACHOGRAPH_PAYLOAD_LENGTH > tachograph_payload, std::array< unsigned char, DSRC_SECURITY_DATA_LENGTH > dsr_security_data)
write RTM data without credentials

8.1.1 Detailed Description

```
template<std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>class tachograph-  
::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH >
```

Tachograph Client.

Template Parameters

<code>TACHOGRAPH_PAYLOAD_LENGTH</code>	length of tachograph payload
<code>DSRC_SECURITY_DATA_LENGTH</code>	length of DSRC security data

Definition at line 26 of file [tachograph.hpp](#).

8.1.2 Constructor & Destructor Documentation

```
8.1.2.1 template<std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH
>::application ( std::shared_ptr< qftop::application< TACHOGRAPH_PAYLOAD_LENGTH,
DSRC_SECURITY_DATA_LENGTH > > qftop_application_ptr, std::shared_ptr< std::ostream > output_stream_ptr )
```

Constructor.

Definition at line 8 of file [tachograph.hpp](#).

8.1.3 Member Function Documentation

```
8.1.3.1 template<std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
void tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH
>::read_rtm_data ( std::function< void(std::array< unsigned char, TACHOGRAPH_PAYLOAD_LENGTH >
tachograph_payload, std::array< unsigned char, DSRC_SECURITY_DATA_LENGTH > dsrc_security_data)> callback )
```

read RTM data without credentials

Parameters

in	<i>callback</i>	callback called when RTM data message is received
----	-----------------	---

Definition at line 14 of file [tachograph.hpp](#).

```
8.1.3.2 template<std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
void tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH
>::write_rtm_data ( std::array< unsigned char, TACHOGRAPH_PAYLOAD_LENGTH > tachograph_payload,
std::array< unsigned char, DSRC_SECURITY_DATA_LENGTH > dsrc_security_data )
```

write RTM data without credentials

Parameters

in	<i>tachograph_payload</i>	content of tachograph payload to be written
in	<i>dsrc_security_data</i>	content of DSRC security data to be written

Definition at line 52 of file [tachograph.hpp](#).

8.2 qftop::application Class Reference

QFTOP Client.

```
#include <qftop_application.hpp>
```

Public Member Functions

- **application** (std::function< void(std::unique_ptr< [qftopMessage](#) >)> *on_message_write_callback*, std::shared_ptr< std::ostream > *output_stream*)
Constructor.
- void **push_message** (std::unique_ptr< [qftopMessage](#) > *message*)
Add message to back of input buffer.
- void **start_polling** ()
Start polling for messages.
- void **stop_polling** ()
Stop polling for messages.

- void `send_write_without_cred` (unsigned char `element_id`, unsigned char `attribute_id`, std::vector< unsigned char > `attribute_value`, std::shared_ptr< `write_response_callback` > `on_write_response_callback`)
write a single attribute without credentials
- void `send_read_without_cred` (unsigned char `element_id`, unsigned char `attribute_id`, std::shared_ptr< `read_response_callback` > `on_read_response_callback`)
read a single attribute without credentials

8.2.1 Detailed Description

QFTOP Client.

Definition at line 185 of file [qftop_application.hpp](#).

8.2.2 Constructor & Destructor Documentation

- 8.2.2.1 `qftop::application::application` (std::function< void(std::unique_ptr< `qfTopMessage`)>
`on_message_write_callback`, std::shared_ptr< std::ostream > `output_stream`)

Constructor.

Definition at line 83 of file [qftop_application.cpp](#).

8.2.3 Member Function Documentation

- 8.2.3.1 `void qftop::application::push_message` (std::unique_ptr< `qfTopMessage` > `message`)

Add message to back of input buffer.

Parameters

<code>in</code>	<code>message</code>	message to add to back of input buffer
-----------------	----------------------	--

Definition at line 92 of file [qftop_application.cpp](#).

- 8.2.3.2 `void qftop::application::send_read_without_cred` (unsigned char `element_id`, unsigned char `attribute_id`, std::shared_ptr< `read_response_callback` > `on_read_response_callback`)

read a single attribute without credentials

Parameters

<code>in</code>	<code>element_id</code>	ID of the element containing the attribute to be read
<code>in</code>	<code>attribute_id</code>	ID of the attribute to be read
<code>in</code>	<code>on_read_response_callback</code>	callback called when the read response is received

Definition at line 184 of file [qftop_application.cpp](#).

Here is the call graph for this function:



8.2.3.3 void qftop::application::send_write_without_cred (unsigned char element_id, unsigned char attribute_id, std::vector<unsigned char > attribute_value, std::shared_ptr< write_response_callback > on_write_response_callback)

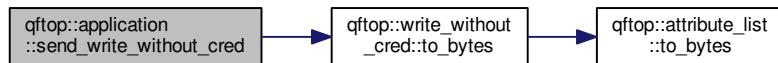
write a single attribute without credentials

Parameters

in	<i>element_id</i>	ID of element containing the attribute to be written
in	<i>attribute_id</i>	ID of attribute to be written to
in	<i>attribute_value</i>	attribute value to be written
in	<i>on_write_response_callback</i>	callback called when the write response is received

Definition at line 151 of file [qftop_application.cpp](#).

Here is the call graph for this function:



8.2.3.4 void qftop::application::start_polling ()

Start polling for messages.

Definition at line 132 of file [qftop_application.cpp](#).

8.2.3.5 void qftop::application::stop_polling ()

Stop polling for messages.

Definition at line 142 of file [qftop_application.cpp](#).

8.3 qftop::attribute_list Struct Reference

Attribute List.

```
#include <qftop_application.hpp>
```

Public Member Functions

- std::vector< unsigned char > [to_bytes \(\)](#)

Public Attributes

- unsigned char [attribute_id](#)
- unsigned char [container_type](#)
- std::vector< unsigned char > [attribute_value](#)

Friends

- std::ostream & [operator<< \(std::ostream &, const attribute_list &\)](#)

8.3.1 Detailed Description

Attribute List.

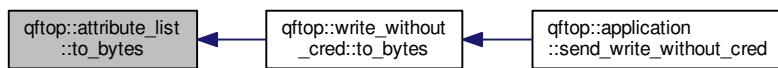
Definition at line 53 of file [qftop_application.hpp](#).

8.3.2 Member Function Documentation

8.3.2.1 std::vector<unsigned char> qftop::attribute_list::to_bytes ()

Definition at line 57 of file [qftop_application.hpp](#).

Here is the caller graph for this function:



8.3.3 Friends And Related Function Documentation

8.3.3.1 std::ostream& operator<< (std::ostream & , const attribute_list &) [friend]

Definition at line 30 of file [qftop_application.cpp](#).

8.3.4 Member Data Documentation

8.3.4.1 unsigned char qftop::attribute_list::attribute_id

Definition at line 54 of file [qftop_application.hpp](#).

8.3.4.2 std::vector<unsigned char> qftop::attribute_list::attribute_value

Definition at line 56 of file [qftop_application.hpp](#).

8.3.4.3 unsigned char qftop::attribute_list::container_type

Definition at line 55 of file [qftop_application.hpp](#).

8.4 qfTopMSG::crc Union Reference

```
#include <qftop_client.h>
```

Public Attributes

- uint8_t [bytes](#) [2]
- uint16_t [word](#)

8.4.1 Detailed Description

Definition at line 76 of file [qftop_client.h](#).

8.4.2 Member Data Documentation

8.4.2.1 uint8_t qfTopMSG::crc::bytes[2]

< check sums

Definition at line 77 of file [qftop_client.h](#).

8.4.2.2 uint16_t qfTopMSG::crc::word

Definition at line 78 of file [qftop_client.h](#).

8.5 qftop::get_control Struct Reference

Get Control Data.

```
#include <qftop_application.hpp>
```

Public Attributes

- union {
 struct {
 unsigned char [mode](#): 1
 unsigned char [has_attribute_list](#): 1
 unsigned char [has_id](#): 1
 unsigned char [has_credentials](#): 1
 unsigned char [action](#): 4
 }
 unsigned char [byte](#)
};

Friends

- std::ostream & [operator<<](#) (std::ostream &, const [get_control](#) &)

8.5.1 Detailed Description

Get Control Data.

Definition at line [98](#) of file [qftop_application.hpp](#).

8.5.2 Friends And Related Function Documentation

8.5.2.1 std::ostream& [operator<<](#) (std::ostream & , const [get_control](#) &) [friend]

Definition at line [50](#) of file [qftop_application.cpp](#).

8.5.3 Member Data Documentation

8.5.3.1 union { ... }

8.5.3.2 unsigned char [qftop::get_control::action](#)

Definition at line [105](#) of file [qftop_application.hpp](#).

8.5.3.3 unsigned char [qftop::get_control::byte](#)

Definition at line [107](#) of file [qftop_application.hpp](#).

8.5.3.4 unsigned char [qftop::get_control::has_attribute_list](#)

Definition at line [102](#) of file [qftop_application.hpp](#).

8.5.3.5 unsigned char [qftop::get_control::has_credentials](#)

Definition at line [104](#) of file [qftop_application.hpp](#).

8.5.3.6 unsigned char [qftop::get_control::has_iid](#)

Definition at line [103](#) of file [qftop_application.hpp](#).

8.5.3.7 unsigned char [qftop::get_control::mode](#)

Definition at line [101](#) of file [qftop_application.hpp](#).

8.6 qfTopMSG::header Union Reference

```
#include <qftop_client.h>
```

Public Attributes

- struct {
 unsigned char **preamble**: 8
 unsigned char **sequence**: 4
 unsigned char **frameType**: 2
 unsigned char **status**: 1
 unsigned char **syn**: 1
 unsigned char **length**: 8
};
- uint8_t **bytes** [3]

8.6.1 Detailed Description

Definition at line 58 of file [qftop_client.h](#).

8.6.2 Member Data Documentation

8.6.2.1 struct { ... }

8.6.2.2 uint8_t qfTopMSG::header::bytes[3]

Definition at line 67 of file [qftop_client.h](#).

8.6.2.3 unsigned char qfTopMSG::header::frameType

Definition at line 62 of file [qftop_client.h](#).

8.6.2.4 unsigned char qfTopMSG::header::length

Definition at line 65 of file [qftop_client.h](#).

8.6.2.5 unsigned char qfTopMSG::header::preamble

< QFTop header

Definition at line 60 of file [qftop_client.h](#).

8.6.2.6 unsigned char qfTopMSG::header::sequence

Definition at line 61 of file [qftop_client.h](#).

8.6.2.7 unsigned char qfTopMSG::header::status

Definition at line 63 of file [qftop_client.h](#).

8.6.2.8 unsigned char qfTopMSG::header::syn

Definition at line 64 of file [qftop_client.h](#).

8.7 qfTopMSG::pdu Union Reference

```
#include <qftop_client.h>
```

Public Attributes

- struct {
 uint8_t messageType
 uint8_t PARAMETERS [maximumQFTOPFrameSize]
};
- uint8_t bytes [maximumQFTOPFrameSize+1]

8.7.1 Detailed Description

Definition at line 69 of file [qftop_client.h](#).

8.7.2 Member Data Documentation

8.7.2.1 struct { ... }

< The ASN.1 or other data payload

8.7.2.2 uint8_t qfTopMSG::pdu::bytes[maximumQFTOPFrameSize+1]

Definition at line 74 of file [qftop_client.h](#).

8.7.2.3 uint8_t qfTopMSG::pdu::messageType

Definition at line 71 of file [qftop_client.h](#).

8.7.2.4 uint8_t qfTopMSG::pdu::PARAMETERS[maximumQFTOPFrameSize]

Definition at line 72 of file [qftop_client.h](#).

8.8 qfTopMessage Struct Reference

State holder of QFTop Message plus state info when parsing.

```
#include <qftop_client.h>
```

Public Attributes

- struct qfTopMSG msg
- bool pre_escape
- bool head_start
- bool head_read
- uint8_t message [maximumQFTOPFrameSize+10]
- unsigned int message_length

8.8.1 Detailed Description

State holder of QFTop Message plus state info when parsing.

Definition at line 85 of file [qftop_client.h](#).

8.8.2 Member Data Documentation

8.8.2.1 bool qfTopMessage::head_read

state: header read

Definition at line 89 of file [qftop_client.h](#).

8.8.2.2 bool qfTopMessage::head_start

state: reading header

Definition at line 88 of file [qftop_client.h](#).

8.8.2.3 uint8_t qfTopMessage::message[maximumQFTOPFrameSize+10]

serialised message

Definition at line 90 of file [qftop_client.h](#).

8.8.2.4 unsigned int qfTopMessage::message_length

length of serialised message

Definition at line 91 of file [qftop_client.h](#).

8.8.2.5 struct qfTopMSG qfTopMessage::msg

The message being sent/received

Definition at line 86 of file [qftop_client.h](#).

8.8.2.6 bool qfTopMessage::pre_escape

state: preamble must be escaped

Definition at line 87 of file [qftop_client.h](#).

8.9 qfTopMSG Struct Reference

Structure of QFTop Message.

```
#include <qftop_client.h>
```

Classes

- union [crc](#)
- union [header](#)
- union [pdu](#)

Public Attributes

- union `qfTopMSG::header HEADER`
- union `qfTopMSG::pdu PDU`
- union `qfTopMSG::crc CRC`
- union `qfTopMSG::crc CRC_REC`

8.9.1 Detailed Description

Structure of QFTop Message.

see the specification doc for details.

Definition at line 57 of file [qftop_client.h](#).

8.9.2 Member Data Documentation

8.9.2.1 union qfTopMSG::crc qfTopMSG::CRC

generated from passing

8.9.2.2 union qfTopMSG::crc qfTopMSG::CRC_REC

received

8.9.2.3 union qfTopMSG::header qfTopMSG::HEADER

8.9.2.4 union qfTopMSG::pdu qfTopMSG::PDU

8.10 qftop::read_response_callback Class Reference

Read Response Callback Interface.

```
#include <qftop_application.hpp>
```

Public Member Functions

- virtual void `on_success` (`std::vector< unsigned char > attribute_value`)=0
- virtual void `on_error` ()=0

8.10.1 Detailed Description

Read Response Callback Interface.

Definition at line 176 of file [qftop_application.hpp](#).

8.10.2 Member Function Documentation

8.10.2.1 virtual void qftop::read_response_callback::on_error() [pure virtual]

8.10.2.2 virtual void qftop::read_response_callback::on_success(`std::vector< unsigned char > attribute_value`) [pure virtual]

8.11 qftop::read_without_cred Struct Reference

Read Without Credentials Request.

```
#include <qftop_application.hpp>
```

Public Member Functions

- std::vector< unsigned char > [to_bytes \(\)](#)

Public Attributes

- union {
 struct {
 unsigned char [length](#)
 unsigned char [fragment_header](#)
 struct [get_control](#) [control](#)
 unsigned char [element_id](#)
 unsigned char [attribute_count](#)
 unsigned char [attribute_list](#) [1]
 }
 unsigned char [bytes](#) [6]
 };

Friends

- std::ostream & [operator<< \(std::ostream &, const read_without_cred &\)](#)

8.11.1 Detailed Description

Read Without Credentials Request.

Definition at line 117 of file [qftop_application.hpp](#).

8.11.2 Member Function Documentation

8.11.2.1 std::vector<unsigned char> qftop::read_without_cred::to_bytes ()

Definition at line 129 of file [qftop_application.hpp](#).

Here is the caller graph for this function:



8.11.3 Friends And Related Function Documentation

8.11.3.1 `std::ostream& operator<<(std::ostream &, const read_without_cred &)` [friend]

Definition at line 58 of file [qftop_application.cpp](#).

8.11.4 Member Data Documentation

8.11.4.1 `union { ... }`

8.11.4.2 `unsigned char qftop::read_without_cred::attribute_count`

Definition at line 124 of file [qftop_application.hpp](#).

8.11.4.3 `unsigned char qftop::read_without_cred::attribute_list[1]`

Definition at line 125 of file [qftop_application.hpp](#).

8.11.4.4 `unsigned char qftop::read_without_cred::bytes[6]`

Definition at line 127 of file [qftop_application.hpp](#).

8.11.4.5 `struct get_control qftop::read_without_cred::control`

Definition at line 122 of file [qftop_application.hpp](#).

8.11.4.6 `unsigned char qftop::read_without_cred::element_id`

Definition at line 123 of file [qftop_application.hpp](#).

8.11.4.7 `unsigned char qftop::read_without_cred::fragment_header`

Definition at line 121 of file [qftop_application.hpp](#).

8.11.4.8 `unsigned char qftop::read_without_cred::length`

Definition at line 120 of file [qftop_application.hpp](#).

8.12 qftop::read_without_cred_response Struct Reference

Read Without Credentials Response.

```
#include <qftop_application.hpp>
```

Public Attributes

- `union {`
- `struct {`
- `unsigned char length`
- `unsigned char fragment_header`

```

struct get_control control
unsigned char element_id
unsigned char attribute_count
unsigned char attribute_id
unsigned char container_id
}
unsigned char bytes [sizeof(unsigned char)*6+sizeof(get_control)]
} header
• std::vector< unsigned char > attribute_value

```

Friends

- std::ostream & [operator<<](#) (std::ostream &, const [read_without_cred_response](#) &)

8.12.1 Detailed Description

Read Without Credentials Response.

Definition at line 145 of file [qftop_application.hpp](#).

8.12.2 Friends And Related Function Documentation

8.12.2.1 std::ostream& operator<< (std::ostream & , const [read_without_cred_response](#) &) [friend]

Definition at line 67 of file [qftop_application.cpp](#).

8.12.3 Member Data Documentation

8.12.3.1 unsigned char [qftop::read_without_cred_response::attribute_count](#)

Definition at line 152 of file [qftop_application.hpp](#).

8.12.3.2 unsigned char [qftop::read_without_cred_response::attribute_id](#)

Definition at line 153 of file [qftop_application.hpp](#).

8.12.3.3 std::vector<unsigned char> [qftop::read_without_cred_response::attribute_value](#)

Definition at line 158 of file [qftop_application.hpp](#).

8.12.3.4 unsigned char [qftop::read_without_cred_response::bytes](#)[sizeof(unsigned char)*6+sizeof(get_control)]

Definition at line 156 of file [qftop_application.hpp](#).

8.12.3.5 unsigned char [qftop::read_without_cred_response::container_id](#)

Definition at line 154 of file [qftop_application.hpp](#).

8.12.3.6 struct [get_control](#) [qftop::read_without_cred_response::control](#)

Definition at line 150 of file [qftop_application.hpp](#).

8.12.3.7 `unsigned char qftop::read_without_cred_response::element_id`

Definition at line 151 of file [qftop_application.hpp](#).

8.12.3.8 `unsigned char qftop::read_without_cred_response::fragment_header`

Definition at line 149 of file [qftop_application.hpp](#).

8.12.3.9 `union { ... } qftop::read_without_cred_response::header`

8.12.3.10 `unsigned char qftop::read_without_cred_response::length`

Definition at line 148 of file [qftop_application.hpp](#).

8.13 qftop::set_control Struct Reference

Set Control Data.

```
#include <qftop_application.hpp>
```

Public Attributes

- `union {
 struct {
 unsigned char mode: 1;
 unsigned char fill: 1;
 unsigned char has_iid: 1;
 unsigned char has_credentials: 1;
 unsigned char action: 4;
 };
 unsigned char byte
};`

Friends

- `std::ostream & operator<< (std::ostream &, const set_control &)`

8.13.1 Detailed Description

Set Control Data.

Definition at line 34 of file [qftop_application.hpp](#).

8.13.2 Friends And Related Function Documentation

8.13.2.1 `std::ostream& operator<< (std::ostream & , const set_control &) [friend]`

Definition at line 22 of file [qftop_application.cpp](#).

8.13.3 Member Data Documentation

8.13.3.1 union { ... }

8.13.3.2 unsigned char qftop::set_control::action

Definition at line 41 of file [qftop_application.hpp](#).

8.13.3.3 unsigned char qftop::set_control::byte

Definition at line 43 of file [qftop_application.hpp](#).

8.13.3.4 unsigned char qftop::set_control::fill

Definition at line 38 of file [qftop_application.hpp](#).

8.13.3.5 unsigned char qftop::set_control::has_credentials

Definition at line 40 of file [qftop_application.hpp](#).

8.13.3.6 unsigned char qftop::set_control::has_iid

Definition at line 39 of file [qftop_application.hpp](#).

8.13.3.7 unsigned char qftop::set_control::mode

Definition at line 37 of file [qftop_application.hpp](#).

8.14 qftop::tty Class Reference

```
#include <qftop_tty.hpp>
```

Public Member Functions

- **tty** (const std::string &device_name, std::function< void(std::unique_ptr< [qfTopMessage](#) >) > on_new_message_callback)
- void **send_message** (std::unique_ptr< [qfTopMessage](#) > message)
Send a QFTOP message.
- void **start_reading** ()
Start reading data from tty.
- void **stop_reading** ()
Stop reading data from tty.

8.14.1 Detailed Description

Definition at line 27 of file [qftop_tty.hpp](#).

8.14.2 Constructor & Destructor Documentation

8.14.2.1 `qftop::tty (const std::string & device_name, std::function< void(std::unique_ptr< qfTopMessage >) > on_new_message_callback)`

Constructor

Parameters

in	<i>device_name</i>	name of tty device
in	<i>on_new_message_callback</i>	callback called when a QFTOP message has been received

Definition at line 10 of file [qftop_tty.cpp](#).

8.14.3 Member Function Documentation**8.14.3.1 void qftop::tty::send_message (std::unique_ptr< qfTopMessage > message)**

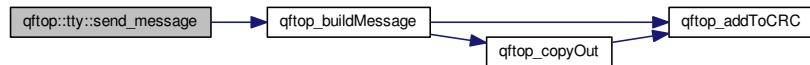
Send a QFTOP message.

Parameters

in	<i>message</i>	QFTOP message to be sent
----	----------------	--------------------------

Definition at line 26 of file [qftop_tty.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.14.3.2 void qftop::tty::start_reading ()**

Start reading data from tty.

Definition at line 49 of file [qftop_tty.cpp](#).

Here is the caller graph for this function:



8.14.3.3 void qftop::tty::stop_reading()

Stop reading data from tty.

Definition at line 59 of file [qftop_tty.cpp](#).

Here is the caller graph for this function:



8.15 qftop::write_response_callback Class Reference

Write Response Callback Interface.

```
#include <qftop_application.hpp>
```

Public Member Functions

- virtual void [on_success](#) ()=0
- virtual void [on_error](#) ()=0

8.15.1 Detailed Description

Write Response Callback Interface.

Definition at line 167 of file [qftop_application.hpp](#).

8.15.2 Member Function Documentation

8.15.2.1 virtual void qftop::write_response_callback::[on_error](#)() [pure virtual]

8.15.2.2 virtual void qftop::write_response_callback::[on_success](#)() [pure virtual]

8.16 qftop::write_without_cred Struct Reference

Write Without Credentials Request.

```
#include <qftop_application.hpp>
```

Public Member Functions

- std::vector< unsigned char > [to_bytes \(\)](#)

Public Attributes

- unsigned char [length](#)
- unsigned char [fragment_header](#)
- struct [set_control](#) [control](#)
- unsigned char [element_id](#)
- unsigned char [attribute_count](#)
- struct [attribute_list](#) [attribute_list](#)

Friends

- std::ostream & [operator<< \(std::ostream &, const write_without_cred &\)](#)

8.16.1 Detailed Description

Write Without Credentials Request.

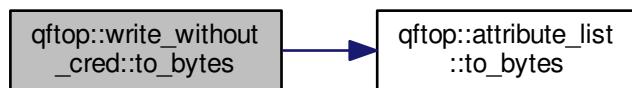
Definition at line [72](#) of file [qftop_application.hpp](#).

8.16.2 Member Function Documentation

8.16.2.1 std::vector<unsigned char> qftop::write_without_cred::to_bytes ()

Definition at line [79](#) of file [qftop_application.hpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.3 Friends And Related Function Documentation

8.16.3.1 `std::ostream& operator<< (std::ostream &, const write_without_cred &) [friend]`

Definition at line 41 of file [qftop_application.cpp](#).

8.16.4 Member Data Documentation

8.16.4.1 `unsigned char qftop::write_without_cred::attribute_count`

Definition at line 77 of file [qftop_application.hpp](#).

8.16.4.2 `struct attribute_list qftop::write_without_cred::attribute_list`

Definition at line 78 of file [qftop_application.hpp](#).

8.16.4.3 `struct set_control qftop::write_without_cred::control`

Definition at line 75 of file [qftop_application.hpp](#).

8.16.4.4 `unsigned char qftop::write_without_cred::element_id`

Definition at line 76 of file [qftop_application.hpp](#).

8.16.4.5 `unsigned char qftop::write_without_cred::fragment_header`

Definition at line 74 of file [qftop_application.hpp](#).

8.16.4.6 `unsigned char qftop::write_without_cred::length`

Definition at line 73 of file [qftop_application.hpp](#).

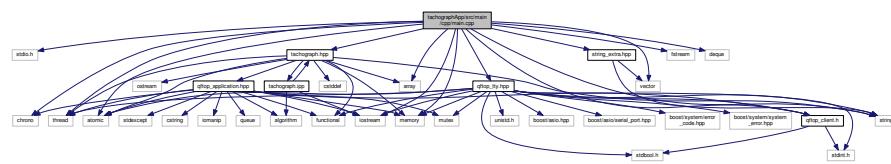
Chapter 9

File Documentation

9.1 dox/mainpage.dox File Reference

9.2 tachographApp/src/main/cpp/main.cpp File Reference

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <thread>
#include <memory>
#include <atomic>
#include <string>
#include <chrono>
#include <vector>
#include <deque>
#include <array>
#include "qftop_tty.hpp"
#include "qftop_client.h"
#include "tachograph.hpp"
#include "string_extra.hpp"
```



Functions

- int main ()

9.3 main.cpp

```
00001  
00006 #include <stdio.h>  
00007 #include <iostream>  
00008 #include <fstream>
```

```

00009 #include <thread>
00010 #include <memory>
00011 #include <atomic>
00012 #include <string>
00013 #include <chrono>
00014 #include <vector>
00015 #include <deque>
00016 #include <array>
00017 #include "qftop_tty.hpp"
00018 #include "qftop_client.h"
00019 #include "tachograph.hpp"
00020 #include "string_extra.hpp"
00021
00022 int main() {
00023     using string_extra::has_prefix;
00024     using string_extra::split_on whitespace;
00025     using string_extra::hex_to_bytes;
00026
00027     std::cout << "Tachograph" << std::endl;
00028     std::shared_ptr<std::ofstream> output_stream_ptr = std::make_shared<std::ofstream>();
00029     output_stream_ptr->open("log.txt");
00030
00031     std::unique_ptr<qftop::tty> tty_ptr;
00032     const auto qftop_application_ptr = std::make_shared<qftop::application>(
00033         [&](auto message_ptr) { tty_ptr->send_message(std::move(message_ptr)); },
00034         output_stream_ptr);
00035
00036     std::string user_input;
00037     const std::string default_tty_device = "/dev/ttyUSB0";
00038
00039     while (true) {
00040         std::cout << "Input tty device (or <> to use " << default_tty_device << ")" << std::endl;
00041         std::getline(std::cin, user_input);
00042         std::string tty_device;
00043         if (user_input.empty()) {
00044             tty_device = default_tty_device;
00045         } else {
00046             tty_device = user_input;
00047         }
00048         try {
00049             tty_ptr = std::make_unique<qftop::tty>(tty_device, [&](std::unique_ptr<qfTopMessage> msg_ptr) {
00050                 qftop_application_ptr->push_message(std::move(msg_ptr));
00051             });
00052         } catch (const std::exception &e) {
00053             std::cout << "Failed to initialize tty" << std::endl;
00054             continue;
00055         }
00056         std::cout << "Initialized tty" << std::endl;
00057         break;
00058     }
00059
00060     std::cout << "Input 'i' to write data with counter and increment counter" << std::endl;
00061     std::cout << "Input 'r' to read data " << std::endl;
00062     std::cout << "Input 'w <tachograph_payload> [<dsrc_security_data>]" to write RTM data" << std::endl;
00063     std::cout << "Input 'q' to finish" << std::endl;
00064
00065     int counter = 0;
00066
00067     const std::size_t dsrc_security_data_length = 16;
00068     const std::size_t tachograph_payload_length = 67;
00069     const auto tachograph_application_ptr =
00070         std::make_unique<tachograph::application<tachograph_payload_length, dsrc_security_data_length>>(
00071             qftop_application_ptr, output_stream_ptr);
00072     qftop_application_ptr->start_polling();
00073     tty_ptr->start_reading();
00074
00075     while (true) {
00076         std::getline(std::cin, user_input);
00077         if (user_input == "q") {
00078             break;
00079         } else if (user_input == "r") {
00080             tachograph_application_ptr->read_rtm_data(
00081                 [](std::array<unsigned char, tachograph_payload_length> &tachograph_payload,
00082                  std::array<unsigned char, dsrc_security_data_length> &dsrc_security_data) {
00083                     std::cout << "Tachograph Payload: ";
00084                     for (const auto &character : tachograph_payload) {
00085                         std::cout << std::setfill('0') << std::setw(2) << std::hex << (int)character;
00086                     }
00087                     std::cout << std::endl << "Dsrc Security Data: ";
00088                     for (const auto &character : dsrc_security_data) {
00089                         std::cout << std::setfill('0') << std::setw(2) << std::hex << (int)character;
00090                     }
00091                     std::cout << std::endl;
00092                 });
00093         } else if (has_prefix(user_input, "w")) {
00094             auto tokens = split_on whitespace(user_input);
00095
00096         }
00097     }
00098 }
```

```

00095     std::array<unsigned char, dsrc_security_data_length> dsrc_security_data;
00096     int tokens_count = tokens.size();
00097     if (tokens_count == 2) {
00098         dsrc_security_data.fill(0);
00099     } else if (tokens_count == 3) {
00100         auto dsrc_security_data_string = tokens[2];
00101         if (dsrc_security_data_string.size() == (dsrc_security_data_length * 2)) {
00102             auto bytes = hex_to_bytes(dsrc_security_data_string);
00103             for (int i = 0; i < (int)dsrc_security_data_length; i += 1) {
00104                 dsrc_security_data[i] = bytes[i];
00105             }
00106         } else {
00107             std::cout << "<dsrc_security_data> length incorrect. Should be 32 hex chars. Was "
00108             << dsrc_security_data_string.size() << std::endl;
00109             continue;
00110         }
00111     } else {
00112         std::cout << "Incorrect input" << std::endl;
00113         continue;
00114     }

00115     std::array<unsigned char, tachograph_payload_length> tachograph_payload;
00116     auto tachograph_payload_string = tokens[1];
00117     if (tachograph_payload_string.size() == (tachograph_payload_length * 2)) {
00118         auto bytes = hex_to_bytes(tachograph_payload_string);
00119         for (int i = 0; i < (int)tachograph_payload_length; i += 1) {
00120             tachograph_payload[i] = bytes[i];
00121         }
00122     } else {
00123         std::cout << "<tachograph_payload> length incorrect. Should be 134 hex chars. Was "
00124         << tachograph_payload_string.size() << std::endl;
00125         continue;
00126     }

00127     tachograph_application_ptr->write_rtm_data(tachograph_payload, dsrc_security_data);
00128 } else if (user_input == "i") {
00129     counter += 1;
00130     std::array<unsigned char, tachograph_payload_length> tachograph_payload;
00131     for (int i = 0; i < (int)tachograph_payload_length; i += 4) {
00132         tachograph_payload[i] = 0x7a;
00133         tachograph_payload[i + 1] = 0xc0;
00134         tachograph_payload[i + 2] = 0xbe;
00135         tachograph_payload[i + 3] = counter;
00136     }
00137     tachograph_payload[64] = 0xff;
00138     tachograph_payload[65] = 0xff;
00139     tachograph_payload[66] = 0xff;

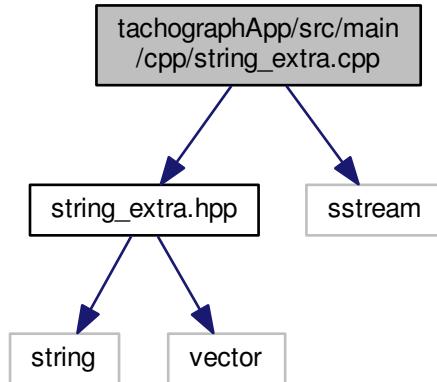
00140     std::array<unsigned char, dsrc_security_data_length> dsrc_security_data;
00141
00142     for (int i = 0; i < (int)dsrc_security_data_length; i += 4) {
00143         dsrc_security_data[i] = 0x01;
00144         dsrc_security_data[i + 1] = 0x23;
00145         dsrc_security_data[i + 2] = 0x45;
00146         dsrc_security_data[i + 3] = 0x67;
00147     }

00148     tachograph_application_ptr->write_rtm_data(tachograph_payload, dsrc_security_data);
00149 } else {
00150     tty_ptr->stop_reading();
00151     qftp_application_ptr->stop_polling();
00152     output_stream_ptr->close();
00153 }
00154
00155     return EXIT_SUCCESS;
00156 }
00157
00158
00159
00160
00161
00162 }
```

9.4 tachographApp/src/main/cpp/string_extra.cpp File Reference

```
#include "string_extra.hpp"
#include <iostream>
```

Include dependency graph for string_extra.cpp:



Namespaces

- `string_extra`

Functions

- `bool string_extra::has_prefix (std::string string_to_check, std::string prefix)`
Check if string has prefix.
- `std::vector< std::string > string_extra::split_on whitespace (std::string string_to_split)`
Split string on whitespace.
- `std::vector< unsigned char > string_extra::hex_to_bytes (std::string hex_string)`
Convert hex string to bytes.

9.5 string_extra.cpp

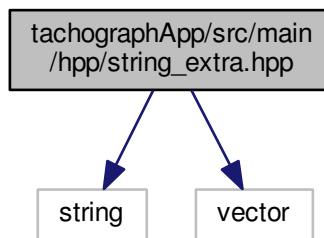
```

00001 #include "string_extra.hpp"
00002 #include <iostream>
00003
00004 namespace string_extra {
00005
00006 bool has_prefix(std::string string_to_check, std::string prefix) {
00007     return std::equal(prefix.begin(), prefix.end(), string_to_check.begin());
00008 }
00009
00010 std::vector<std::string> split_on_whitespace(std::string string_to_split) {
00011     std::string buffer;
00012     std::stringstream stream(string_to_split);
00013     std::vector<std::string> tokens;
00014     while (stream >> buffer) {
00015         tokens.push_back(buffer);
00016     }
00017     return tokens;
00018 }
00019
00020 std::vector<unsigned char> hex_to_bytes(std::string hex_string) {
00021     std::stringstream stream(hex_string);
00022     std::vector<unsigned char> bytes;
00023     unsigned int buffer;
00024     unsigned int offset = 0;
00025     while (offset < hex_string.length()) {
  
```

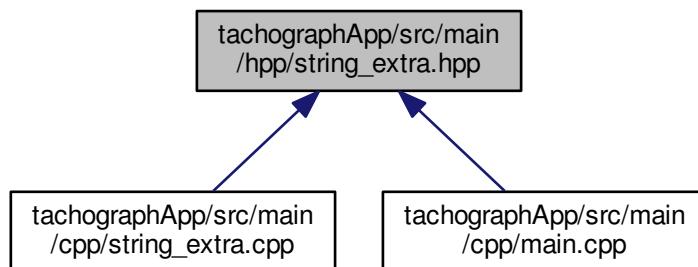
```
00026     stream.str(std::string());
00027     stream.clear();
00028     stream << std::hex << hex_string.substr(offset, 2);
00029     stream >> std::hex >> buffer;
00030     bytes.push_back(static_cast<unsigned char>(buffer));
00031     offset += 2;
00032 }
00033 return bytes;
00034 }
00035 }
```

9.6 tachographApp/src/main.hpp/string_extra.hpp File Reference

```
#include <string>
#include <vector>
Include dependency graph for string_extra.hpp:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- `string_extra`

Functions

- bool `string_extra::has_prefix` (std::string `string_to_check`, std::string `prefix`)
Check if string has prefix.
- std::vector< std::string > `string_extra::split_on_whitespace` (std::string `string_to_split`)
Split string on whitespace.
- std::vector< unsigned char > `string_extra::hex_to_bytes` (std::string `hex_string`)
Convert hex string to bytes.

9.7 string_extra.hpp

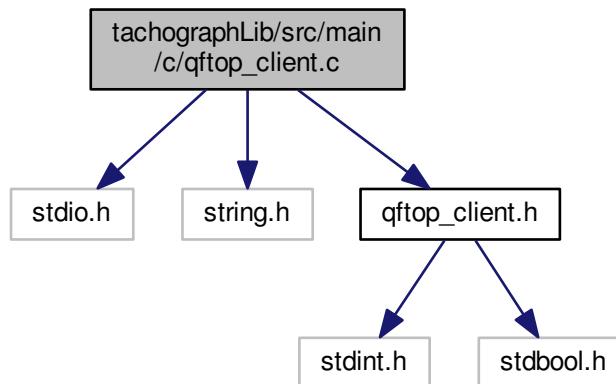
```

00001
00006 #ifndef STRING_EXTRA_HPP
00007 #define STRING_EXTRA_HPP
00008
00009 #include <string>
00010 #include <vector>
00011
00012 namespace string_extra {
00020     bool has_prefix(std::string string_to_check, std::string prefix);
00021
00028     std::vector<std::string> split_on_whitespace(std::string string_to_split);
00029
00036     std::vector<unsigned char> hex_to_bytes(std::string hex_string);
00037 } // namespace string_extra
00038 #endif // STRING_EXTRA_HPP
00039

```

9.8 tachographLib/src/main/c/qftop_client.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "qftop_client.h"
Include dependency graph for qftop_client.c:
```



Functions

- uint8_t * `qftop_copyIn` (struct `qfTopMessage` *msg, uint8_t *in, uint8_t *out, int len)

- `uint8_t * qftop_copyOut (struct qfTopMessage *msg, uint8_t *out, uint8_t *in, int len)`

helper to copy a string of bytes from another vector based on the state of a messages
- `void qftop_addToCRC (struct qfTopMessage *msg, uint8_t ch)`

Modify crc calculation for a new byte.
- `void qftop_clear (struct qfTopMessage *msg)`

Convenience function to zero / reset a message.
- `unsigned int qftop_buildMessage (struct qfTopMessage *msg)`

Function to build a byte stream ready for transmission based on a message.
- `void qftop_ack (struct qfTopMessage *msg)`

Convenience function to build an ACK message.
- `void qftop_application (struct qfTopMessage *msg)`

Convenience function to initialise an application message (dsrc req)
- `void qftop_addParameter (struct qfTopMessage *msg, uint8_t p)`

Function for adding a single parameter to a QFTop message.
- `uint16_t qftop_extractMessage (struct qfTopMessage *msg_out, struct qfTopMessage *msg_in)`

Function to build an internal message based on bytes in another message.
- `int qftop_parse (struct qfTopMessage *msg, uint8_t cr)`

Function to parse a new byte into a message being received.

9.8.1 Function Documentation

9.8.1.1 `uint8_t * qftop_copyIn (struct qfTopMessage * msg, uint8_t * in, uint8_t * out, int len)`

helper to copy a string of bytes into another vector based on the state of a messages

Parameters

<code>in</code>	<code>msg</code>	pointer to <code>qfTopMessage</code>
<code>in</code>	<code>in</code>	input bytes
<code>in</code>	<code>out</code>	output bytes
<code>in</code>	<code>len</code>	no of bytes to process

Returns

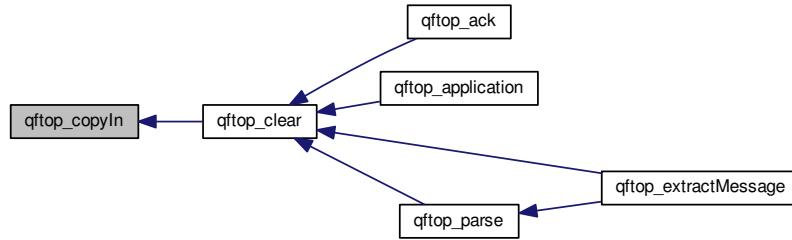
`len` final pointer

Definition at line 23 of file `qftop_client.c`.

Here is the call graph for this function:



Here is the caller graph for this function:



9.8.1.2 uint8_t * qftop_copyOut (struct qfTopMessage * msg, uint8_t * out, uint8_t * in, int len)

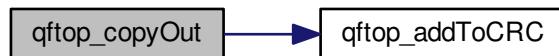
helper to copy a string of bytes from another vector based on the state of a messages

Parameters

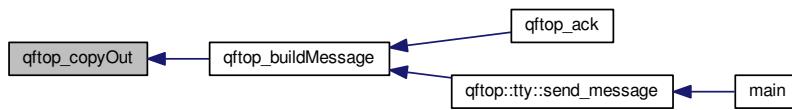
in	<i>msg</i>	pointer to qfTopMessage
in	<i>in</i>	input bytes
in	<i>out</i>	output bytes
in	<i>len</i>	No of bytes to copy

Definition at line 53 of file [qftop_client.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



9.9 qftop_client.c

```

00001 #include <stdio.h>
00002 #include <string.h>
  
```

```

00003 #include "qftop_client.h"
00004
00005 uint8_t *qftop_copyIn(struct qfTopMessage *msg, uint8_t *in, uint8_t *out, int len)
00006 ;
00007 uint8_t *qftop_copyOut(struct qfTopMessage *msg, uint8_t *out, uint8_t *in, int len);
00008 void qftop_addToCRC(struct qfTopMessage *msg, uint8_t ch) {
00009     ch = (ch ^ (uint8_t)(msg->msg.CRC.word & 0xFF));
00010     ch = (ch ^ (uint8_t)(ch << 4));
00011     msg->msg.CRC.word = (msg->msg.CRC.word >> 8) ^ ((uint16_t)ch << 8) ^ ((uint16_t)ch
00012     << 3) ^ ((uint16_t)ch >> 4);
00013 }
00014
00015 uint8_t *qftop_copyIn(struct qfTopMessage *msg, uint8_t *in, uint8_t *out, int len)
00016 {
00017     uint8_t *in_p = in;
00018     uint8_t *out_p = out;
00019     uint8_t v;
00020     for (int i = 0; i < len;) {
00021         v = *in_p++;
00022         if (v == qftop_preamble) {
00023             if (msg->pre_escape) {
00024                 msg->pre_escape = false;
00025                 continue;
00026             }
00027             msg->pre_escape = true;
00028         } else {
00029             msg->pre_escape = false;
00030         }
00031         *out_p++ = v;
00032         qftop_addToCRC(msg, v);
00033         i += 1;
00034     }
00035     return in_p;
00036 }
00037
00038 uint8_t *qftop_copyOut(struct qfTopMessage *msg, uint8_t *out, uint8_t *in, int len) {
00039     uint8_t *in_p = in;
00040     uint8_t *out_p = out;
00041     uint8_t v;
00042     for (int i = 0; i < len; i += 1) {
00043         v = *in_p++;
00044         *out_p++ = v;
00045         qftop_addToCRC(msg, v);
00046         if (v == qftop_preamble) {
00047             *out_p++ = qftop_preamble;
00048         }
00049     }
00050     return out_p;
00051 }
00052
00053 void qftop_clear(struct qfTopMessage *msg) {
00054     uint8_t zero[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
00055     qftop_copyIn(msg, zero, msg->msg.HEADER.bytes, (int)3);
00056     qftop_copyIn(msg, zero, msg->msg.PDU.PARAMETERS, (int)10);
00057     msg->msg.PDU.messageType = 0;
00058     msg->msg.CRC.word = crc_init;
00059     msg->msg.CRC_REC.word = 0xFFFF;
00060     msg->message_length = 0;
00061     msg->pre_escape = false;
00062     msg->head_read = false;
00063     msg->head_start = false;
00064 }
00065
00066 unsigned int qftop_buildMessage(struct qfTopMessage *msg) {
00067     uint8_t *m = msg->message;
00068     uint8_t v;
00069     msg->msg.CRC.word = crc_init;
00070     v = msg->msg.HEADER.bytes[0];
00071     *m++ = v;
00072     if (v == qftop_preamble) {
00073         *m++ = qftop_preamble;
00074     }
00075     v = msg->msg.HEADER.bytes[1];
00076     *m++ = v;
00077     qftop_addToCRC(msg, v);
00078     if (v == qftop_preamble) {
00079         *m++ = qftop_preamble;
00080     }
00081     v = msg->msg.HEADER.bytes[2];
00082     *m++ = v;
00083     qftop_addToCRC(msg, v);
00084     if (v == qftop_preamble) {
00085         *m++ = qftop_preamble;
00086     }

```

```

00102     m = qftop_copyOut(msg, m, msg->msg.PDU.bytes, msg->
00103     msg.HEADER.length);
00104     *m++ = msg->msg.CRC_REC.bytes[0] = msg->msg.CRC.bytes[0];
00105     *m++ = msg->msg.CRC_REC.bytes[1] = msg->msg.CRC.bytes[1];
00106     msg->message_length = m - msg->message;
00107 }
00108
00109 void qftop_ack(struct qfTopMessage *msg) {
00110     qftop_clear(msg);
00111     msg->pre_escape = false, msg->head_read = false, msg->
00112     head_start = false;
00113     msg->message_length = 0;
00114     msg->msg.HEADER.preamble = qftop_preamble;
00115     msg->msg.HEADER.frameType = 1; // ACK;
00116     msg->msg.PDU.messageType = 0;
00117     msg->msg.HEADER.length = (uint8_t)0;
00118     msg->msg.CRC.word = crc_init;
00119     qftop_buildMessage(msg);
00120
00121 void qftop_application(struct qfTopMessage *msg) {
00122     qftop_clear(msg);
00123     msg->pre_escape = false;
00124     msg->head_read = false;
00125     msg->message_length = 0;
00126     msg->msg.HEADER.preamble = qftop_preamble;
00127     msg->msg.HEADER.frameType = Application;
00128     msg->msg.PDU.messageType = dsrc_17_req;
00129     msg->msg.HEADER.length = (uint8_t)1;
00130     msg->msg.CRC.word = crc_init;
00131 }
00132
00133 void qftop_addParameter(struct qfTopMessage *msg, uint8_t p) {
00134     msg->msg.PDU.PARAMETERS[msg->msg.HEADER.length++ - 1] = 0xFF & p;
00135     qftop_addToCRC(msg, p);
00136 }
00137
00138 uint16_t qftop_extractMessage(struct qfTopMessage *msg_out, struct
00139     qfTopMessage *msg_in) {
00140     uint16_t rc = 0;
00141     qftop_clear(msg_out);
00142
00143     for (int i = 0; i < msg_in->message_length; i += 1) {
00144         rc = qftop_parse(msg_out, msg_in->message[i]);
00145         msg_out->message[i] = msg_in->message[i];
00146     }
00147     msg_out->message_length = msg_in->message_length;
00148     return rc;
00149 }
00150 int qftop_parse(struct qfTopMessage *msg, uint8_t cr) {
00151     if (cr == qftop_preamble) {
00152         if (msg->pre_escape) {
00153             msg->pre_escape = false;
00154         } else {
00155             msg->pre_escape = true;
00156             return 0;
00157         }
00158     } else if (msg->pre_escape) {
00159         qftop_clear(msg);
00160         msg->message[msg->message_length++] =
00161             qftop_preamble;
00162         msg->message[msg->message_length++] = cr;
00163         msg->head_start = true;
00164         msg->head_read = false;
00165         msg->pre_escape = false;
00166         return 0;
00167     }
00168     if (!msg->head_start) {
00169         return 0;
00170     }
00171
00172     msg->message[msg->message_length++] = cr;
00173     if (msg->message_length > maximumQFTOPFrameSize) {
00174         msg->head_start = false;
00175         msg->message_length = 0;
00176         return -1;
00177     }
00178
00179     if (!msg->head_read) {
00180         msg->msg.HEADER.bytes[0] = msg->message[0];
00181         msg->msg.HEADER.bytes[1] = msg->message[1];
00182         msg->msg.HEADER.bytes[2] = msg->message[2];
00183         qftop_addToCRC(msg, msg->msg.HEADER.bytes[1]);
00184         qftop_addToCRC(msg, msg->msg.HEADER.bytes[2]);

```

```

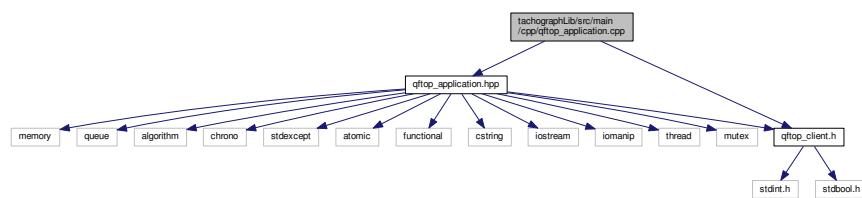
00185     msg->head_read = true;
00186     return 0;
00187 } else if (msg->message_length == msg->msg.HEADER.
00188 length + 5) {
00189     for (int i = 0; i < msg->msg.HEADER.length; i += 1) {
00190         msg->msg.PDU.bytes[i] = msg->message[i + 3];
00191         qftop_addToCRC(msg, msg->msg.PDU.bytes[i]);
00192     }
00193     msg->msg.CRC_REC.bytes[1] = msg->message[msg->
00194 message_length - 1];
00195     msg->msg.CRC_REC.bytes[0] = msg->message[msg->
00196 message_length - 2];
00197     return 1;
00198 }
00199 return 0;
00200 }
```

9.10 tachographLib/src/main/cpp/qftop_application.cpp File Reference

#include "qftop_application.hpp"

#include "qftop_client.h"

Include dependency graph for qftop_application.cpp:



Namespaces

- `qftop`

Functions

- void `qftop::print_message` (std::ostream &out, const `qfTopMessage` *rhs)
Print QFTOP message to stream.
 - std::ostream & `qftop::operator<<` (std::ostream &out, const `set_control` &rhs)
 - std::ostream & `qftop::operator<<` (std::ostream &out, const `attribute_list` &rhs)
 - std::ostream & `qftop::operator<<` (std::ostream &out, const `write_without_cred` &rhs)
 - std::ostream & `qftop::operator<<` (std::ostream &out, const `get_control` &rhs)
 - std::ostream & `qftop::operator<<` (std::ostream &out, const `read_without_cred` &rhs)
 - std::ostream & `qftop::operator<<` (std::ostream &out, const `read_without_cred_response` &rhs)

9.11 qftop_application.cpp

```

00001 #include "qftop_application.hpp"
00002 #include "qftop_client.h"
00003
00004 namespace qftop {
00005
00006 void print_message(std::ostream &out, const qfTopMessage *rhs) {
00007     out << "Preamble: " << std::setw(2) << std::hex << (int)rhs->msg.HEADER.
00008 preamble << std::endl
00009     << "Sequence number: " << std::setw(2) << std::hex << (int)rhs->msg.
00010 HEADER.sequence << std::endl
00011     << "Frame type: " << (int)rhs->msg.HEADER.frameType << std::endl
00012 }
```

```

00010      << "Status: " << (int)rhs->msg.HEADER.status << std::endl
00011      << "SYN: " << (int)rhs->msg.HEADER.syn << std::endl
00012      << "Length: " << (int)rhs->msg.HEADER.length << std::endl
00013      << "Message type: " << (int)rhs->msg.PDU.messageType << std::endl
00014      << "Parameters: ";
00015      for (int i = 0; i < (int)(rhs->msg.HEADER.length - 1); i += 1) {
00016          out << std::setfill('0') << std::setw(2) << std::hex << (int)rhs->msg.
00017          PDU.PARAMETERS[i];
00018      }
00019      out << std::endl;
00020      out << "CRC: " << std::setfill('0') << std::setw(2) << std::hex << rhs->msg.
00021      CRC.word << std::endl;
00022 }
00023
00024 std::ostream &operator<<(std::ostream &out, const set_control &rhs) {
00025     return out << "Mode: " << (int)rhs.mode << std::endl
00026         << "Fill: " << (int)rhs.fill << std::endl
00027         << "Has IID: " << (int)rhs.has_id << std::endl
00028         << "Has Credentials: " << (int)rhs.has_credentials << std::endl
00029         << "Action: " << (int)rhs.action << std::endl;
00030 }
00031
00032 std::ostream &operator<<(std::ostream &out, const attribute_list &rhs) {
00033     out << "Attribute ID: " << (int)rhs.attribute_id << std::endl
00034         << "Container type: " << std::setfill('0') << std::setw(2) << std::hex << (int)rhs.
00035         container_type << std::endl
00036         << "Attribute Value: ";
00037         for (auto &&character : rhs.attribute_value) {
00038             out << std::setfill('0') << std::setw(2) << std::hex << (int)character;
00039         }
00040     out << std::endl;
00041     return out;
00042 }
00043
00044 std::ostream &operator<<(std::ostream &out, const write_without_cred &rhs) {
00045     return out << "Length: " << (int)rhs.length << std::endl
00046         << "Fragment header: " << std::setfill('0') << std::setw(2) << std::hex << (int)rhs.
00047         fragment_header
00048         << std::endl
00049         << rhs.control << "Element ID: " << (int)rhs.element_id << std::endl
00050         << "Attribute count: " << (int)rhs.attribute_count << std::endl
00051         << rhs.attribute_list;
00052 }
00053
00054 std::ostream &operator<<(std::ostream &out, const get_control &rhs) {
00055     return out << "Mode: " << (int)rhs.mode << std::endl
00056         << "Has attribute list: " << (int)rhs.has_attribute_list << std::endl
00057         << "Has IID: " << (int)rhs.has_id << std::endl
00058         << "Has Credentials: " << (int)rhs.has_credentials << std::endl
00059         << "Action: " << (int)rhs.action << std::endl;
00060 }
00061
00062 std::ostream &operator<<(std::ostream &out, const read_without_cred &rhs) {
00063     return out << "Length: " << (int)rhs.length << std::endl
00064         << "Fragment header: " << std::setfill('0') << std::setw(2) << std::hex << (int)rhs.
00065         fragment_header
00066         << std::endl
00067         << rhs.control << "Element ID: " << (int)rhs.element_id << std::endl
00068         << "Attribute count: " << (int)rhs.attribute_count << std::endl
00069         << "Attribute list: " << (int)rhs.attribute_list[0] << std::endl;
00070 }
00071
00072 std::ostream &operator<<(std::ostream &out, const
00073     read_without_cred_response &rhs) {
00074     out << "Length: " << (int)rhs.header.length << std::endl
00075         << "Fragment header: " << std::setfill('0') << std::setw(2) << std::hex << (int)rhs.
00076         header.fragment_header
00077         << std::endl
00078         << rhs.header.control << "Element ID: " << (int)rhs.header.
00079         element_id << std::endl
00080         << "Attribute count: " << (int)rhs.header.attribute_count << std::endl
00081         << "Attribute ID: " << (int)rhs.header.attribute_id << std::endl
00082         << "Container ID: " << (int)rhs.header.container_id << std::endl
00083         << "Attribute value: " << std::endl;
00084         for (auto character : rhs.attribute_value) {
00085             out << std::setfill('0') << std::setw(2) << std::hex << (int)character;
00086         }
00087     out << std::endl;
00088     return out;
00089 }
00090
00091 application::application(std::function<void(std::unique_ptr<qfTopMessage>)>
00092     on_message_write_callback,
00093     std::shared_ptr<std::ostream> output_stream)
00094     : on_message_write_callback(on_message_write_callback), output_stream(std::move(output_stream)) {
00095     auto x = this->create_empty_queue();
00096     this->queue = std::move(x);
00097 }
```

```

00088     this->run = std::make_shared<std::atomic<bool>>(&false);
00089     this->sequence_counter.store((unsigned char)0);
00090 }
00091
00092 void application::push_message(std::unique_ptr<qfTopMessage> message) {
00093     this->queue_lock.lock();
00094     this->queue->push(std::move(message));
00095     this->queue_lock.unlock();
00096 }
00097
00098 bool application::has_messages() {
00099     this->queue_lock.lock();
00100     bool queue_not_empty = !this->queue->empty();
00101     this->queue_lock.unlock();
00102     return queue_not_empty;
00103 }
00104
00105 std::unique_ptr<qfTopMessage> application::pop_message() {
00106     this->queue_lock.lock();
00107     auto message = std::move(this->queue->front());
00108     this->queue->pop();
00109     this->queue_lock.unlock();
00110     return std::move(message);
00111 }
00112
00113 void application::send_message(std::unique_ptr<qfTopMessage> message_ptr) {
00114     print_message(*this->output_stream, &(*message_ptr));
00115     this->on_message_write_callback(std::move(message_ptr));
00116 }
00117
00118 void application::clear_messages() {
00119     this->queue_lock.lock();
00120     auto empty_queue = this->create_empty_queue();
00121     std::swap(this->queue, empty_queue);
00122     this->queue_lock.unlock();
00123 }
00124
00125 std::unique_ptr<qfTopMessage> application::create_message() {
00126     auto message_ptr = std::make_unique<qfTopMessage>();
00127     auto message_raw_ptr = message_ptr.get();
00128     qftop_application(message_raw_ptr);
00129     return std::move(message_ptr);
00130 }
00131
00132 void application::start_polling() {
00133     this->input_thread_lock.lock();
00134     bool was_running = this->run->exchange(true);
00135     if (!was_running) {
00136         this->input_thread =
00137             std::make_unique<std::thread>(std::bind(&application::process_input, this, this->run, this->
00138                                         output_stream));
00139     }
00140     this->input_thread_lock.unlock();
00141 }
00142 void application::stop_polling() {
00143     this->input_thread_lock.lock();
00144     bool was_running = this->run->exchange(false);
00145     if (was_running) {
00146         this->input_thread->join();
00147     }
00148     this->input_thread_lock.unlock();
00149 }
00150
00151 void application::send_write_without_cred(unsigned char element_id,
00152                                             unsigned char attribute_id,
00153                                             std::vector<unsigned char> attribute_value,
00154                                             std::shared_ptr<write_response_callback>
00155                                             on_write_response_callback) {
00156     unsigned char tapdu_set_request = 4;
00157
00158     struct write_without_cred request;
00159     request.length = sizeof(request);
00160     request.fragment_header = 0x91;
00161     request.control.action = tapdu_set_request;
00162     request.control.has_credentials = 0;
00163     request.control.fill = 0;
00164     request.control.mode = 1;
00165     request.element_id = element_id;
00166     request.attribute_count = 1;
00167     request.attribute_list.attribute_id = attribute_id;
00168     request.attribute_list.container_type = 10;
00169     request.attribute_list.attribute_value = std::move(attribute_value);
00170
00171     *(this->output_stream) << "----Sending SET.request----" << std::endl << request << "-----"
00172     << std::endl;

```

```

00171     auto request_bytes = request.to_bytes();
00172
00173     const unsigned char sequence_number = this->sequence_counter.fetch_add(1, std::memory_order_seq_cst);
00174     this->callbacks_lock.lock();
00175     this->write_response_callbacks.push(std::move(on_write_response_callback));
00176     this->callbacks_lock.unlock();
00177
00178     const unsigned char message_type_in = QFTOP_DSRC_L7_REQ;
00179     this->send_message(request_bytes, message_type_in, sequence_number);
00180     delay();
00181
00182 }
00183
00184 void application::send_read_without_cred(unsigned char element_id,
00185                                         unsigned char attribute_id,
00186                                         std::shared_ptr<read_response_callback> on_read_response_callback)
00187 {
00188     unsigned char a[] = {attribute_id};
00189     unsigned char tapdu_get_request = 6;
00190     struct read_without_cred get_request;
00191     get_request.length = 5;
00192     get_request.fragment_header = 0x91;
00193     get_request.control.action = tapdu_get_request;
00194     get_request.control.has_credentials = 0;
00195     get_request.control.has_iid = 0;
00196     get_request.control.has_attribute_list = 1;
00197     get_request.control.mode = 0;
00198     get_request.element_id = element_id;
00199     get_request.attribute_count = 1;
00200     get_request.attribute_list[0] = a[0];
00201
00202     * (this->output_stream) << "----Sending GET.request----" << std::endl
00203     << get_request << "-----" << std::endl;
00204
00205     const unsigned char sequence_number = this->sequence_counter.fetch_add(1, std::memory_order_seq_cst);
00206
00207     this->callbacks_lock.lock();
00208     this->read_response_callbacks.push(std::move(on_read_response_callback));
00209     this->callbacks_lock.unlock();
00210
00211     const unsigned char command_type_in = QFTOP_DSRC_L7_REQ;
00212     this->send_message(get_request.to_bytes(), command_type_in, sequence_number);
00213
00214     delay();
00215 }
00216
00217 void application::send_message(std::vector<unsigned char> raw_message, uint8_t message_type_in,
00218                                 unsigned char sequence_number) {
00219     auto message_request_ptr = this->create_message();
00220
00221     message_request_ptr->msg.PDU.messageType = message_type_in;
00222     message_request_ptr->msg.HEADER.syn = 1;
00223     message_request_ptr->msg.HEADER.sequence = sequence_number;
00224
00225     auto message_request_raw_ptr = message_request_ptr.get();
00226     for (auto &&character : raw_message) {
00227         qftop_addParameter(message_request_raw_ptr, character);
00228     }
00229     this->send_message(std::move(message_request_ptr));
00230
00231     std::unique_ptr<std::queue<std::unique_ptr<qfTopMessage>>> application::create_empty_queue() {
00232         return std::move(std::make_unique<std::queue<std::unique_ptr<qfTopMessage>>>());
00233     }
00234
00235     void application::delay() {
00236         const int sleep_time_ms = 100;
00237         std::this_thread::sleep_for(std::chrono::milliseconds(sleep_time_ms));
00238     }
00239
00240     void application::process_input(std::shared_ptr<std::atomic<bool>> run, std::shared_ptr<std::ostream>
00241                                     output_stream) {
00242         while (run->load()) {
00243             this->delay();
00244             while (this->has_messages()) {
00245                 * (output_stream) << "Has message(s)" << std::endl;
00246                 auto message_ptr = this->pop_message();
00247
00248                 if (message_ptr->msg.PDU.messageType ==
00249                     QFTOP_DSRC_L7_RESP) {
00250                     * (output_stream) << "Message is L7 RESP" << std::endl;
00251                     unsigned char sequence_number = message_ptr->msg.HEADER.
00252                     sequence;
00253                     * (output_stream) << "seq num = " << (int)sequence_number << std::endl;
00254                     print_message(* (output_stream), &(*message_ptr));
00255                     // TODO: sanity check parameters length
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876

```

```

00253         const unsigned char action = (message_ptr->msg.PDU.
00254             PARAMETERS[2] >> 4);
00255         const unsigned char tapdu_get_response = 7;
00256         const unsigned char tapdu_set_response = 5;
00257         if (action == tapdu_get_response) {
00258             this->callbacks_lock.lock();
00259             if (!this->read_response_callbacks.empty()) {
00260                 const auto callback = this->read_response_callbacks.front();
00261                 *(output_stream) << "Found read callback" << std::endl;
00262                 read_without_cred_response response;
00263                 std::memcpy(&response.header, message_ptr->msg.PDU.
00264                     PARAMETERS, sizeof(response.header));
00265                 const int header_length = sizeof(response.header.bytes);
00266                 const int attribute_value_start = header_length;
00267                 const int attribute_value_end = attribute_value_start + response.header.length;
00268
00269                 std::vector<unsigned char> attribute_value;
00270                 for (int i = attribute_value_start; i != attribute_value_end; i += 1) {
00271                     attribute_value.push_back(message_ptr->msg.PDU.
00272                         PARAMETERS[i]);
00273                 }
00274                 *(output_stream) << response << std::endl;
00275
00276                 // TODO: Error checking
00277                 callback->on_success(std::move(attribute_value));
00278
00279                 this->read_response_callbacks.pop();
00280             } else {
00281                 *(output_stream) << "Could not find read callback" << std::endl;
00282             }
00283             this->callbacks_lock.unlock();
00284         } else if (action == tapdu_set_response) {
00285             this->callbacks_lock.lock();
00286             if (!this->write_response_callbacks.empty()) {
00287                 const auto callback = this->write_response_callbacks.front();
00288                 *(output_stream) << "Found write callback" << std::endl;
00289
00290                 // write_without_cred_response response;
00291                 // TODO: Error checking
00292                 callback->on_success();
00293                 this->write_response_callbacks.pop();
00294             } else {
00295                 *(output_stream) << "Could not find write callback" << std::endl;
00296             }
00297             this->callbacks_lock.unlock();
00298         } else {
00299             *(output_stream) << "Unknown action: " << std::setfill('0') << std::setw(2) << std::hex
00300                 << (int)action << std::endl;
00301         }
00302     }
00303 } else {
00304     *(output_stream) << "Message was ignored" << std::endl;
00305 }
00306 }
00307 }
00308 }
00309 } // namespace qftop

```

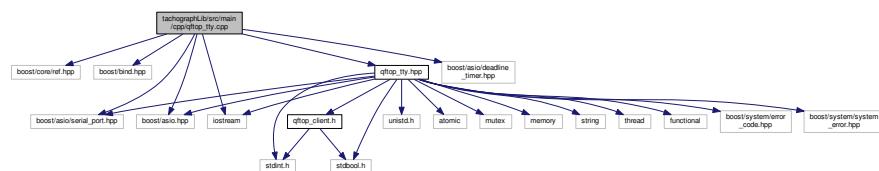
9.12 tachographLib/src/main/cpp/qftop_tty.cpp File Reference

```

#include <boost/core/ref.hpp>
#include <boost/bind.hpp>
#include <boost/asio/serial_port.hpp>
#include <boost/asio.hpp>
#include <boost/asio/deadline_timer.hpp>
#include "qftop_tty.hpp"
#include <iostream>

```

Include dependency graph for qftop_tty.cpp:



Namespaces

- qftop

9.13 qftop_tty.cpp

```
00001 #include <boost/core/ref.hpp>
00002 #include <boost/bind.hpp>
00003 #include <boost/asio/serial_port.hpp>
00004 #include <boost/asio.hpp>
00005 #include <boost/asio/deadline_timer.hpp>
00006 #include "qftop_tty.hpp"
00007 #include <iostream>
00008
00009 namespace qftop {
00010 tty::tty(const std::string &device_name, std::function<void(std::unique_ptr<qfTopMessage>)>
    on_new_message_callback)
00011     : io_service()
00012     , port(io_service, device_name)
00013     , on_new_message_callback(on_new_message_callback)
00014     , run(std::make_shared<std::atomic<bool>>(false)) {
00015 {
00016     using boost::asio::serial_port_base;
00017     boost::system::error_code error_code;
00018     this->port.set_option(serial_port_base::baud_rate(115200));
00019     this->port.set_option(serial_port_base::parity(serial_port_base::parity::type::none));
00020     this->port.set_option(serial_port_base::character_size(8));
00021     this->port.set_option(serial_port_base::stop_bits(serial_port_base::stop_bits::type::one));
00022     this->port.set_option(serial_port_base::flow_control(serial_port_base::flow_control::type::none));
00023 }
00024 }
00025
00026 void tty::send_message(std::unique_ptr<qfTopMessage> message_ptr) {
00027     qftop_buildMessage(&(*message_ptr));
00028
00029     using boost::system::error_code;
00030     using boost::asio::buffer;
00031
00032     const char wake_up_sequence[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
00033     error_code wake_up_sequence_error_code;
00034     const int wakeup_bytes_written =
00035         this->port.write_some(buffer(wake_up_sequence, sizeof(wake_up_sequence)),
00036         wake_up_sequence_error_code);
00037     if (wake_up_sequence_error_code || (wakeup_bytes_written != sizeof(wake_up_sequence))) {
00038         std::cerr << "Error writing wake up bytes" << std::endl;
00039         return;
00040     }
00041     error_code message_error_code;
00042     const int message_bytes_written =
00043         this->port.write_some(buffer(message_ptr->message, message_ptr->
00044         message_length), message_error_code);
00045     if (message_error_code || (message_bytes_written != message_ptr->
00046         message_length)) {
00047         std::cerr << "Error writing message" << std::endl;
00048     }
00049 void tty::start_reading() {
00050     this->input_thread.lock.lock();
00051     bool was_running = this->run->exchange(true);
00052     if (was_running) {
00053         return;
00054     }
00055     this->input_thread = std::make_unique<std::thread>(std::bind(&tty::read_messages, this));
```

```

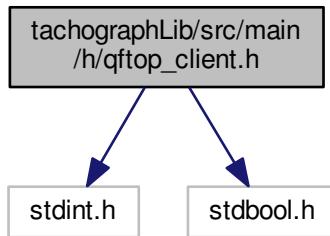
00056     this->input_thread_lock.unlock();
00057 }
00058
00059 void tty::stop_reading() {
00060     this->input_thread_lock.lock();
00061     bool was_running = this->run->exchange(false);
00062     if (was_running) {
00063         this->input_thread->join();
00064     }
00065     this->input_thread_lock.unlock();
00066 }
00067
00068 void tty::read_messages() {
00069     std::vector<char> input_buffer(1024);
00070     std::atomic<bool> data_available(false);
00071     const unsigned int timeout = 100;
00072     boost::asio::deadline_timer timer(this->io_service);
00073     std::unique_ptr<qfTopMessage> current_message_ptr = std::make_unique<qfTopMessage>();
00074
00075     while (this->run->load()) {
00076         this->port.async_read_some(boost::asio::buffer(input_buffer),
00077                                     boost::bind(&tty::read_callback, this, boost::ref(data_available),
00078                                     boost::ref(timer),
00079                                     boost::asio::placeholders::error,
00080                                     boost::asio::placeholders::bytes_transferred));
00081         timer.expires_from_now(boost::posix_time::milliseconds(timeout));
00082         timer.async_wait(
00083             boost::bind(&tty::timeout_callback, this, boost::ref(this->port),
00084             boost::asio::placeholders::error));
00085         this->io_service.run(); // Blocks until all async callbacks are finished
00086         this->io_service.reset();
00087         if (!(data_available.load())) {
00088             continue;
00089         }
00090         for (auto &&character : input_buffer) {
00091             uint8_t return_code = qftop_parse(&(*current_message_ptr), character);
00092             if (return_code > 0) {
00093                 this->on_new_message_callback(std::move(current_message_ptr));
00094                 current_message_ptr = std::make_unique<qfTopMessage>();
00095             }
00096         }
00097     }
00098
00099 void tty::read_callback(std::atomic<bool> &data_available, boost::asio::deadline_timer &timeout,
00100                         const boost::system::error_code &error_code, std::size_t bytes_transferred) {
00101     if (error_code || !bytes_transferred) {
00102         data_available.store(false);
00103         return;
00104     }
00105     timeout.cancel(); // Will cause wait_callback to fire with an error
00106     data_available.store(true);
00107 }
00108
00109 void tty::timeout_callback(boost::asio::serial_port &serial_port, const boost::system::error_code &
00110                           error_code) {
00111     if (error_code) {
00112         // Data was read and this timeout was canceled
00113         return;
00114     }
00115     serial_port.cancel(); // Will cause read_callback to fire with an error
00116 } // namespace qftop

```

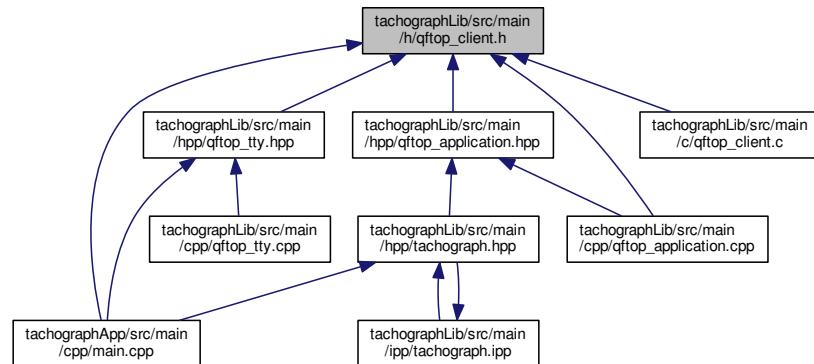
9.14 tachographLib/src/main/h/qftop_client.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for qftop_client.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct `qfTopMSG`
Structure of QFTop Message.
- union `qfTopMSG::header`
- union `qfTopMSG::pdu`
- union `qfTopMSG::crc`
- struct `qfTopMessage`
State holder of QFTop Message plus state info when parsing.

Macros

- #define `MAXIMUMQFTOPFRAMESIZE` 200

Enumerations

- enum `qftop_cmd_type_t` {
 `QFTOP_ECHO_REQ` = 0x00, `QFTOP_ECHO_RESP` = 0x80, `QFTOP_ACK` = 0x01, `QFTOP_NACK` = 0x02,
`QFTOP_MMI_REQ` = 0x30, `QFTOP_INIT_NOTIFICATION` = 0x31, `QFTOP_TRANSPIRESP` = 0x33, `QFT-`

```

OP_TRANS_P_REQ = 0x34,
QFTOP_REGISTER_APP_REQ = 0x36, QFTOP_REGISTER_APP = 0x37, QFTOP_TEST_REQ = 0x38, Q-
FTOP_TEST_RESP = 0x39,
QFTOP_PERS_REQ = 0x3A, QFTOP_PERS_RESP = 0x3B, QFTOP_DSRC_L7_REQ = 0x3C, QFTOP_D-
SRC_L7_RESP = 0x3D,
QFTOP_TRACE_LOG_REQ = 0xF0, QFTOP_TRACE_LOG_RESP = 0xF1 }

```

Types of QFTOP messages.

- enum `qftop_Types` {
 Application = 0, ACK = 1, NACK = 2, dsrc_l7_req = 0x3C,
 dsrc_l7_resp = 0x3D, crc_init = 0x6363, qftop_preamble = 0xB5, maximumQFTOPFrameSize = MAXIMUM-
 QFTOPFRAMESIZE }

Types of QFTOP messages.

Functions

- int `qftop_parse` (struct `qfTopMessage` *msg, uint8_t cr)

Function to parse a new byte into a message being received.
- uint16_t `qftop_extractMessage` (struct `qfTopMessage` *msg_out, struct `qfTopMessage` *msg_in)

Function to build an internal message based on bytes in another message.
- void `qftop_addToCRC` (struct `qfTopMessage` *msg, uint8_t ch)

Modify crc calculation for a new byte.
- void `qftop_addParameter` (struct `qfTopMessage` *msg, uint8_t p)

Function for adding a single parameter to a QFTop message.
- void `qftop_clear` (struct `qfTopMessage` *msg)

Convenience function to zero / reset a message.
- void `qftop_application` (struct `qfTopMessage` *msg)

Convenience function to initialise an application message (dsrc req)
- void `qftop_ack` (struct `qfTopMessage` *msg)

Convenience function to build an ACK message.
- unsigned int `qftop_buildMessage` (struct `qfTopMessage` *msg)

Function to build a byte stream ready for transmission based on a message.

9.15 qftop_client.h

```

00001
00005 #ifndef QFTOP_CLIENT_H
00006 #define QFTOP_CLIENT_H
00007
00008 #ifdef __cplusplus
00009 extern "C" {
00010 #endif
00011
00012 #define MAXIMUMQFTOPFRAMESIZE 200
00013
00014 #include <stdint.h>
00015 #include <stdbool.h>
00016
00019 typedef enum {
00020     QFTOP_ECHO_REQ = 0x00,
00021     QFTOP_ECHO_RESP = 0x80,
00022     QFTOP_ACK = 0x01,
00023     QFTOP_NACK = 0x02,
00024     QFTOP_MMI_REQ = 0x30,
00025     QFTOP_INIT_NOTIFICATION = 0x31,
00026     QFTOP_TRANS_P_REQ = 0x33,
00027     QFTOP_TRANS_P_RESP = 0x34,
00028     QFTOP_REGISTER_APP_REQ = 0x36,
00029     QFTOP_REGISTER_APP = 0x37,
00030     QFTOP_TEST_REQ = 0x38,
00031     QFTOP_TEST_RESP = 0x39,
00032     QFTOP_PERS_REQ = 0x3A,
00033     QFTOP_PERS_RESP = 0x3B,

```

```

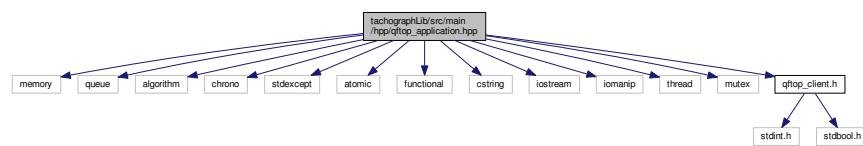
00034     QFTOP_DSRC_L7_REQ = 0x3C,
00035     QFTOP_DSRC_L7_RESP = 0x3D,
00036     QFTOP_TRACE_LOG_REQ = 0xF0,
00037     QFTOP_TRACE_LOG_RESP = 0xF1
00038 } qftop_cmd_type_t;
00039
00042 enum qftop_Types {
00043     Application = 0,
00044     ACK = 1,
00045     NACK = 2,
00046     dsrcl7_req = 0x3C,
00047     dsrcl7_resp = 0x3D,
00048     crc_init = 0x6363,
00049     qftop_preamble = 0xB5,
00050     maximumQFTOPFrameSize = MAXIMUMQFTOPFRAMESIZE
00051 };
00057 struct qfTopMSG {
00058     union header {
00059         struct {
00060             unsigned char preamble : 8;
00061             unsigned char sequence : 4;
00062             unsigned char frameType : 2;
00063             unsigned char status : 1;
00064             unsigned char syn : 1;
00065             unsigned char length : 8;
00066         };
00067         uint8_t bytes[3];
00068     } HEADER;
00069     union pdu {
00070         struct {
00071             uint8_t messageType;
00072             uint8_t PARAMETERS[maximumQFTOPFrameSize];
00073         };
00074         uint8_t bytes[maximumQFTOPFrameSize + 1];
00075     } PDU;
00076     union crc {
00077         uint8_t bytes[2];
00078         uint16_t word;
00079     } CRC;
00080     CRC_REC;
00081 };
00085 struct qfTopMessage {
00086     struct qfTopMSG msg;
00087     bool pre_escape;
00088     bool head_start;
00089     bool head_read;
00090     uint8_t message[maximumQFTOPFrameSize + 10];
00091     unsigned int message_length;
00092 };
00093
00107 int qftop_parse(struct qfTopMessage *msg, uint8_t cr);
00108
00115 uint16_t qftop_extractMessage(struct qfTopMessage *msg_out, struct
00116     qfTopMessage *msg_in);
00116
00123 void qftop_addToCRC(struct qfTopMessage *msg, uint8_t ch);
00124
00130 void qftop_addParameter(struct qfTopMessage *msg, uint8_t p);
00131
00136 void qftop_clear(struct qfTopMessage *msg);
00137
00142 void qftop_application(struct qfTopMessage *msg);
00143
00148 void qftop_ack(struct qfTopMessage *msg);
00149
00156 unsigned int qftop_buildMessage(struct qfTopMessage *msg);
00157
00158 #ifdef __cplusplus
00159 }
00160#endif
00161
00162#endif // QFTOP_CLIENT_H
00163 /* @} */

```

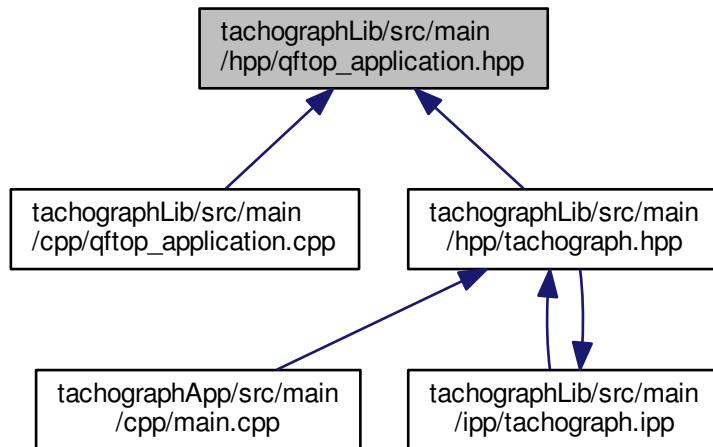
9.16 tachographLib/src/main.hpp/qftop_application.hpp File Reference

```
#include <memory>
```

```
#include <queue>
#include <algorithm>
#include <chrono>
#include <stdexcept>
#include <atomic>
#include <functional>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <thread>
#include <mutex>
#include "qftop_client.h"
Include dependency graph for qftop_application.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct `qftop::set_control`
Set Control Data.
- struct `qftop::attribute_list`
Attribute List.
- struct `qftop::write_without_cred`
Write Without Credentials Request.
- struct `qftop::get_control`

- struct `qftop::read_without_cred`
Read Without Credentials Request.
- struct `qftop::read_without_cred_response`
Read Without Credentials Response.
- class `qftop::write_response_callback`
Write Response Callback Interface.
- class `qftop::read_response_callback`
Read Response Callback Interface.
- class `qftop::application`
QFTOP Client.

Namespaces

- `qftop`

Functions

- void `qftop::print_message` (std::ostream &out, const `qfTopMessage` *rhs)
Print QFTOP message to stream.
- std::ostream & `qftop::operator<<` (std::ostream &out, const `set_control` &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const `attribute_list` &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const `write_without_cred` &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const `get_control` &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const `read_without_cred` &rhs)
- std::ostream & `qftop::operator<<` (std::ostream &out, const `read_without_cred_response` &rhs)

9.17 qftop_application.hpp

```

00001 #ifndef QFTOP_APPLICATION_HPP
00002 #define QFTOP_APPLICATION_HPP
00003
00004 #include <memory>
00005 #include <queue>
00006 #include <algorithm>
00007 #include <chrono>
00008 #include <stdexcept>
00009 #include <atomic>
00010 #include <functional>
00011 #include <cstring>
00012 #include <iostream>
00013 #include <iomanip>
00014 #include <thread>
00015 #include <mutex>
00016 #include "qftop_client.h"
00017
00018 namespace qftop {
00029 void print_message(std::ostream &out, const qfTopMessage *rhs);
00030
00034 struct set_control {
00035     union {
00036         struct {
00037             unsigned char mode : 1;
00038             unsigned char fill : 1;
00039             unsigned char has_iid : 1;
00040             unsigned char has_credentials : 1;
00041             unsigned char action : 4;
00042         };
00043         unsigned char byte;
00044     };
00045     friend std::ostream &operator<<(std::ostream &, const set_control &);

00046 };
00047
00048 std::ostream &operator<<(std::ostream &out, const set_control &rhs);
00049

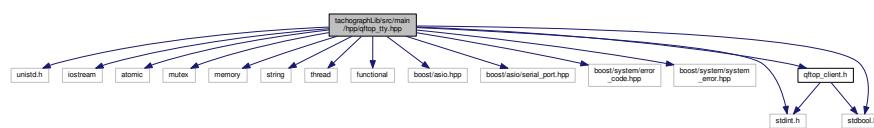
```

```
00053 struct attribute_list {
00054     unsigned char attribute_id;
00055     unsigned char container_type;
00056     std::vector<unsigned char> attribute_value;
00057     std::vector<unsigned char> to_bytes() {
00058         std::vector<unsigned char> bytes;
00059         bytes.push_back(attribute_id);
00060         bytes.push_back(container_type);
00061         bytes.insert(bytes.end(), attribute_value.begin(),
00062                     attribute_value.end());
00062         return std::move(bytes);
00063     }
00064     friend std::ostream &operator<<(std::ostream &, const
00065     attribute_list &);
00065 };
00066
00067 std::ostream &operator<<(std::ostream &out, const attribute_list &rhs);
00068
00069 struct write_without_cred {
00070     unsigned char length;
00071     unsigned char fragment_header;
00072     struct set_control control;
00073     unsigned char element_id;
00074     unsigned char attribute_count;
00075     struct attribute_list attribute_list;
00076     std::vector<unsigned char> to_bytes() {
00077         std::vector<unsigned char> bytes;
00078         bytes.push_back(length);
00079         bytes.push_back(fragment_header);
00080         bytes.push_back(control.byte);
00081         bytes.push_back(element_id);
00082         bytes.push_back(attribute_count);
00083         auto attribute_list_bytes = attribute_list.to_bytes();
00084         bytes.insert(bytes.end(), attribute_list_bytes.begin(),
00085                     attribute_list_bytes.end());
00086         return std::move(bytes);
00087     }
00088     friend std::ostream &operator<<(std::ostream &, const
00089     write_without_cred &);
00090 };
00091
00092
00093 std::ostream &operator<<(std::ostream &out, const write_without_cred &rhs);
00094
00095 struct get_control {
00096     union {
00097         struct {
00098             unsigned char mode : 1;
00099             unsigned char has_attribute_list : 1;
00100             unsigned char has_iid : 1;
00101             unsigned char has_credentials : 1;
00102             unsigned char action : 4;
00103         };
00104         unsigned char byte;
00105     };
00106     friend std::ostream &operator<<(std::ostream &, const get_control &);
00107 };
00108
00109 std::ostream &operator<<(std::ostream &out, const get_control &rhs);
00110
00111
00112 std::ostream &operator<<(std::ostream &out, const get_control &rhs);
00113
00114 struct read_without_cred {
00115     union {
00116         struct {
00117             unsigned char length;
00118             unsigned char fragment_header;
00119             struct get_control control;
00120             unsigned char element_id;
00121             unsigned char attribute_count;
00122             unsigned char attribute_list[1];
00123         };
00124         unsigned char bytes[6];
00125     };
00126     std::vector<unsigned char> to_bytes() {
00127         std::vector<unsigned char> bytes;
00128         for (auto &&character : this->bytes) {
00129             bytes.push_back(character);
00130         }
00131         return std::move(bytes);
00132     }
00133     friend std::ostream &operator<<(std::ostream &, const
00134     read_without_cred &);
00135 };
00136
00137 std::ostream &operator<<(std::ostream &out, const read_without_cred &rhs);
00138
00139
00140 std::ostream &operator<<(std::ostream &out, const read_without_cred &rhs);
00141
00142 struct read_without_cred_response {
00143     union {
00144         struct {
```

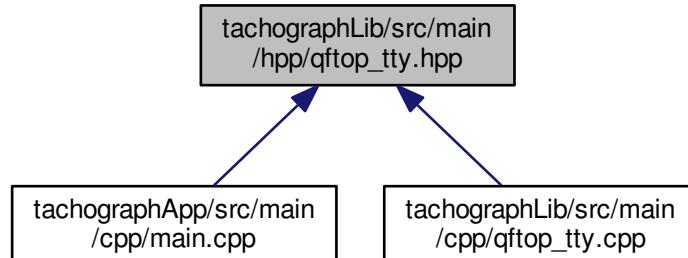
```
00148         unsigned char length;
00149         unsigned char fragment_header;
00150         struct get_control control;
00151         unsigned char element_id;
00152         unsigned char attribute_count;
00153         unsigned char attribute_id;
00154         unsigned char container_id;
00155     };
00156     unsigned char bytes[sizeof(unsigned char) * 6 + sizeof(get_control)];
00157 } header;
00158 std::vector<unsigned char> attribute_value;
00159 friend std::ostream &operator<<(std::ostream &, const
    read_without_cred_response &);
00160 };
00161
00162 std::ostream &operator<<(std::ostream &out, const
    read_without_cred_response &rhs);
00163
00167 class write_response_callback {
00168 public:
00169     virtual void on_success() = 0;
00170     virtual void on_error() = 0;
00171 };
00172
00176 class read_response_callback {
00177 public:
00178     virtual void on_success(std::vector<unsigned char> attribute_value) = 0;
00179     virtual void on_error() = 0;
00180 };
00181
00185 class application {
00186 public:
00187     application(std::function<void(std::unique_ptr<qfTopMessage>)> on_message_write_callback,
00188                 std::shared_ptr<std::ostream> output_stream);
00189
00190     void push_message(std::unique_ptr<qfTopMessage> message);
00191
00192     void start_polling();
00193
00194     void stop_polling();
00195
00196     void send_write_without_cred(unsigned char element_id, unsigned char
        attribute_id,
00197                                     std::vector<unsigned char> attribute_value,
00198                                     std::shared_ptr<write_response_callback> on_write_response_callback);
00199
00200     void send_read_without_cred(unsigned char element_id, unsigned char attribute_id,
00201                                     std::shared_ptr<read_response_callback> on_read_response_callback);
00202
00231 private:
00232     std::queue<std::shared_ptr<write_response_callback>> write_response_callbacks;
00233     std::queue<std::shared_ptr<read_response_callback>> read_response_callbacks;
00234     std::function<void(std::unique_ptr<qfTopMessage>)> on_message_write_callback;
00235     std::shared_ptr<std::ostream> output_stream;
00236     std::shared_ptr<std::atomic<bool>> run;
00237     std::unique_ptr<std::thread> input_thread;
00238     std::mutex queue_lock;
00239     std::mutex callbacks_lock;
00240     std::unique_ptr<std::queue<std::unique_ptr<qfTopMessage>>> queue;
00241     std::mutex input_thread_lock;
00242     std::atomic<unsigned char> sequence_counter;
00243
00244     static std::unique_ptr<std::queue<std::unique_ptr<qfTopMessage>>> create_empty_queue();
00245
00246     static void delay();
00247
00248     void process_input(std::shared_ptr<std::atomic<bool>> run, std::shared_ptr<std::ostream> output_stream)
    ;
00249
00250     void send_message(std::vector<unsigned char> raw_message, uint8_t message_type_in, unsigned char
        sequence_number);
00251
00252     bool has_messages();
00253
00254     std::unique_ptr<qfTopMessage> pop_message();
00255
00256     void send_message(std::unique_ptr<qfTopMessage> message_ptr);
00257
00258     void clear_messages();
00259
00260     static std::unique_ptr<qfTopMessage> create_message();
00261
00262 };
00263 // namespace qftop
00264 #endif // QFTOP_APPLICATION_HPP
```

9.18 tachographLib/src/main.hpp/qftop_tty.hpp File Reference

```
#include <unistd.h>
#include <iostream>
#include <atomic>
#include <mutex>
#include <memory>
#include <string>
#include <thread>
#include <functional>
#include <boost/asio.hpp>
#include <boost/asio/serial_port.hpp>
#include <boost/system/error_code.hpp>
#include <boost/system/system_error.hpp>
#include <stdint.h>
#include <stdbool.h>
#include "qftop_client.h"
Include dependency graph for qftop_tty.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `qftop::tty`

Namespaces

- `qftop`

9.19 qftop_tty.hpp

```

00001 #ifndef QFTOP_TTY_HPP
00002 #define QFTOP_TTY_HPP
00003
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <atomic>
00007 #include <mutex>
00008 #include <memory>
00009 #include <string>
00010 #include <thread>
00011 #include <functional>
00012 #include <boost/asio.hpp>
00013 #include <boost/asio/serial_port.hpp>
00014 #include <boost/system/error_code.hpp>
00015 #include <boost/system/system_error.hpp>
00016 #include <stdint.h>
00017 #include <stdbool.h>
00018
00019 #include "qftop_client.h"
00020
00021 namespace qftop {
00022
00027 class tty {
00028 public:
00029     tty(const std::string &device_name, std::function<void(std::unique_ptr<qfTopMessage>)>
00030         on_new_message_callback);
00031
00032     void send_message(std::unique_ptr<qfTopMessage> message);
00033
00034     void start_reading();
00035
00036     void stop_reading();
00037
00038 private:
00039     std::function<void(std::unique_ptr<qfTopMessage>)> on_new_message_callback;
00040     boost::asio::io_service io_service;
00041     boost::asio::serial_port port;
00042     std::unique_ptr<std::thread> input_thread;
00043     std::shared_ptr<std::atomic<bool>> run;
00044     std::mutex input_thread_lock;
00045     void read_messages();
00046     void read_callback(std::atomic<bool> &data_available, boost::asio::deadline_timer &timeout,
00047                         const boost::system::error_code &error, std::size_t bytes_transferred);
00048     void timeout_callback(boost::asio::serial_port &serial_port, const boost::system::error_code &
00049                           error_code);
00050 };
00051 } // namespace qftop
00052
00053 #endif // QFTOP_TTY_HPP

```

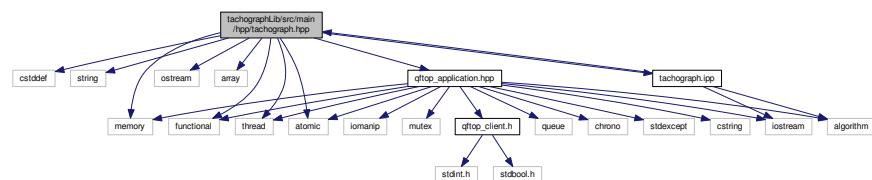
9.20 tachographLib/src/main.hpp/tachograph.hpp File Reference

```

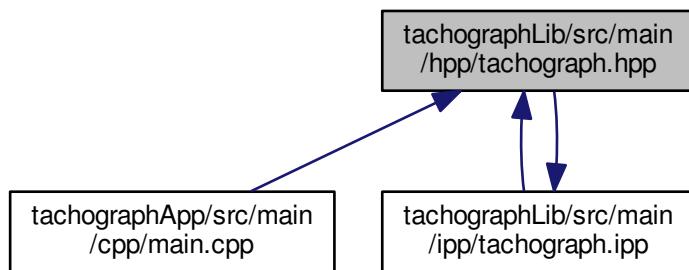
#include <cstddef>
#include <string>
#include <memory>
#include <iostream>
#include <array>
#include <functional>
#include <thread>
#include <atomic>
#include "qftop_application.hpp"
#include "tachograph.hpp"

```

Include dependency graph for tachograph.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `tachograph::application< TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH >`
Tachograph Client.

Namespaces

- tachograph

9.21 tachograph.hpp

```
00001 #ifndef TACHOGRAPH_H
00002 #define TACHOGRAPH_H
00003
00004 #include <cstddef>
00005 #include <string>
00006 #include <memory>
00007 #include <iostream>
00008 #include <array>
00009 #include <functional>
00010 #include <thread>
00011 #include <atomic>
00012 #include "qftop_application.hpp"
00013
00014 namespace tachograph {
00025 template <std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
00026 class application {
00027 public:
00031     application(std::shared_ptr<qftop::application> qftop_application_ptr,
```

```

00032             std::shared_ptr<std::ostream> output_stream_ptr);
00033
00039     void read_rtm_data(
00040         std::function<void(std::array<unsigned char, TACHOGRAPH_PAYLOAD_LENGTH> tachograph_payload,
00041                           std::array<unsigned char, DSRC_SECURITY_DATA_LENGTH> dsr_security_data)>
00042         callback);
00042
00049     void write_rtm_data(std::array<unsigned char, TACHOGRAPH_PAYLOAD_LENGTH>
00050         tachograph_payload,
00051                         std::array<unsigned char, DSRC_SECURITY_DATA_LENGTH> dsr_security_data);
00051
00052 private:
00053     std::shared_ptr<qftop::application> application_ptr;
00054     std::unique_ptr<std::thread> input_thread_ptr;
00055     std::shared_ptr<std::ostream> output_stream_ptr;
00056     std::shared_ptr<std::atomic<bool>> run_ptr;
00057
00058     static void process_input(std::shared_ptr<std::atomic<bool>> run_ptr,
00059                               std::shared_ptr<qftop::application> application_ptr,
00060                               std::shared_ptr<std::ostream> output_stream_ptr);
00061 };
00065 }
00066 #include "tachograph.hpp"
00067
00068 #endif // TACHOGRAPH_H

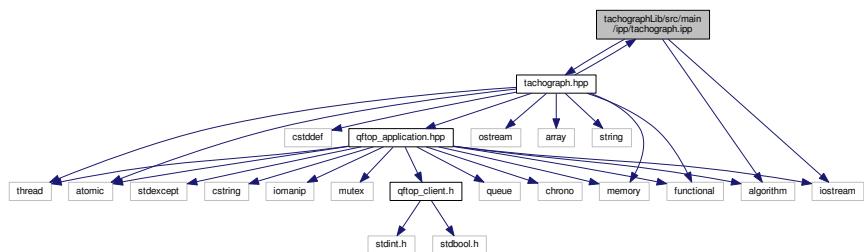
```

9.22 tachographLib/src/main/ipp/tachograph.ipp File Reference

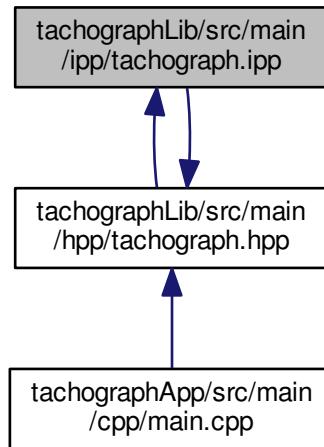
```

#include "tachograph.hpp"
#include <algorithm>
#include <iostream>
Include dependency graph for tachograph.ipp:

```



This graph shows which files directly or indirectly include this file:



Namespaces

- [tachograph](#)

9.23 tachograph.hpp

```

00001 #include "tachograph.hpp"
00002 #include <algorithm>
00003 #include <iostream>
00004
00005 namespace tachograph {
00006
00007 template <std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
00008 application<TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH>::application
00009     (
00010         std::shared_ptr<qftop::application> qftop_application_ptr, std::shared_ptr<std::ostream>
00011         output_stream_ptr)
00012     : application_ptr(qftop_application_ptr), output_stream_ptr(output_stream_ptr) {
00013 }
00014
00015 template <std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
00016 void
00017     application<TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH>::read_rtm_data
00018     (
00019         std::function<void(std::array<unsigned char, TACHOGRAPH_PAYLOAD_LENGTH> tachograph_payload,
00020                         std::array<unsigned char, DSRC_SECURITY_DATA_LENGTH> dsr_security_data)> callback)
00021     {
00022         uint8_t element_id_rtm = 1;
00023         uint8_t attribute_id_rtm_data = 1;
00024
00025         class internal_callback : public qftop::read_response_callback {
00026             public:
00027                 internal_callback(
00028                     std::function<void(std::array<unsigned char, TACHOGRAPH_PAYLOAD_LENGTH> tachograph_payload,
00029                         std::array<unsigned char, DSRC_SECURITY_DATA_LENGTH> dsr_security_data)>
00030                     callback)
00031                     : callback(std::move(callback)) {
00032             }
00033             void on_success(std::vector<unsigned char> attribute_value) override {
00034                 std::cout << "Successful read" << std::endl;
00035                 std::array<unsigned char, (int)TACHOGRAPH_PAYLOAD_LENGTH> tachograph_payload;
00036                 std::copy_n(attribute_value.begin(), (int)TACHOGRAPH_PAYLOAD_LENGTH, tachograph_payload.begin());
00037             };
00038             std::array<unsigned char, (int)DSRC_SECURITY_DATA_LENGTH> dsr_security_data;
  
```

```
00032     std::copy_n(attribute_value.begin() + 1 + (int)TACHOGRAPH_PAYLOAD_LENGTH, (int)
00033     DSRC_SECURITY_DATA_LENGTH,
00034         dsrc_security_data.begin());
00035     this->callback(std::move(tachograph_payload), std::move(dsrc_security_data));
00036 }
00037 void on_error() override {
00038     std::cout << "Erronous read" << std::endl;
00039 }
00040 private:
00041     std::function<void(std::array<unsigned char, TACHOGRAPH_PAYLOAD_LENGTH> tachograph_payload,
00042                         std::array<unsigned char, DSRC_SECURITY_DATA_LENGTH> dsrc_security_data)>
00043     callback;
00044 };
00045 auto internal_callback_ptr = std::make_shared<internal_callback>(std::move(callback));
00046
00047 this->application_ptr->send_read_without_cred(element_id_rtm, attribute_id_rtm_data,
00048                                                 std::move(internal_callback_ptr));
00049 }
00050
00051 template <std::size_t TACHOGRAPH_PAYLOAD_LENGTH, std::size_t DSRC_SECURITY_DATA_LENGTH>
00052 void
00053     application<TACHOGRAPH_PAYLOAD_LENGTH, DSRC_SECURITY_DATA_LENGTH>::write_rtm_data
00054     (
00055         std::array<unsigned char, TACHOGRAPH_PAYLOAD_LENGTH> tachograph_payload,
00056         std::array<unsigned char, DSRC_SECURITY_DATA_LENGTH> dsrc_security_data) {
00057     const std::size_t DSRC_SECURITY_DATA_SIZE_LENGTH = 1;
00058     const std::size_t RTM_DATA_LENGTH =
00059         TACHOGRAPH_PAYLOAD_LENGTH + DSRC_SECURITY_DATA_SIZE_LENGTH + DSRC_SECURITY_DATA_LENGTH;
00060     uint8_t element_id_rtm = 1;
00061     uint8_t attribute_id_rtm_data = 1;
00062
00063     class internal_callback : public qftop::write_response_callback {
00064     public:
00065         void on_success() override {
00066             std::cout << "Successful write" << std::endl;
00067         }
00068         void on_error() override {
00069             std::cout << "Erronous write" << std::endl;
00070         }
00071     };
00072     std::vector<unsigned char> rtm_data;
00073     for (auto &&character : tachograph_payload) {
00074         rtm_data.push_back(character);
00075     }
00076     rtm_data.push_back((unsigned char)DSRC_SECURITY_DATA_SIZE_LENGTH);
00077     for (auto &&character : dsrc_security_data) {
00078         rtm_data.push_back(character);
00079     }
00080     auto internal_callback_ptr = std::make_shared<internal_callback>();
00081
00082     this->application_ptr->send_write_without_cred(element_id_rtm, attribute_id_rtm_data, std::move(
00083         rtm_data),
00084                                                 std::move(internal_callback_ptr));
00085 }
```

Index

ACK
 QFTOP, 11

action
 qftop::get_control, 30
 qftop::set_control, 39

Application
 QFTOP, 11

application
 qftop::application, 26
 tachograph::application, 24

attribute_count
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 37
 qftop::write_without_cred, 44

attribute_id
 qftop::attribute_list, 28
 qftop::read_without_cred_response, 37

attribute_list
 qftop::read_without_cred, 36
 qftop::write_without_cred, 44

attribute_value
 qftop::attribute_list, 28
 qftop::read_without_cred_response, 37

byte
 qftop::get_control, 30
 qftop::set_control, 39

bytes
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 37
 qfTopMSG::crc, 29
 qfTopMSG::header, 31
 qfTopMSG::pdu, 32

CRC
 qfTopMSG, 34

CRC_REC
 qfTopMSG, 34

container_id
 qftop::read_without_cred_response, 37

container_type
 qftop::attribute_list, 29

control
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 37
 qftop::write_without_cred, 44

crc_init
 QFTOP, 11

dox/mainpage.dox, 45

dsrc_I7_req
 QFTOP, 11

dsrc_I7_resp
 QFTOP, 11

element_id
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 37
 qftop::write_without_cred, 44

Example, 20
 main, 20

fill
 qftop::set_control, 39

fragment_header
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 38
 qftop::write_without_cred, 44

frameType
 qfTopMSG::header, 31

HEADER
 qfTopMSG, 34

has_attribute_list
 qftop::get_control, 30

has_credentials
 qftop::get_control, 30
 qftop::set_control, 39

has_id
 qftop::get_control, 30
 qftop::set_control, 39

has_prefix
 string_extra, 22

head_read
 qfTopMessage, 33

head_start
 qfTopMessage, 33

header
 qftop::read_without_cred_response, 38

hex_to_bytes
 string_extra, 22

length
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 38
 qftop::write_without_cred, 44
 qfTopMSG::header, 31

main
 Example, 20

maximumQFTOPFrameSize

QFTOP, 11
message
 qfTopMessage, 33
message_length
 qfTopMessage, 33
messageType
 qfTopMSG::pdu, 32
mode
 qftop::get_control, 30
 qftop::set_control, 39
msg
 qfTopMessage, 33

NACK
 QFTOP, 11

on_error
 qftop::read_response_callback, 34
 qftop::write_response_callback, 42
on_success
 qftop::read_response_callback, 34
 qftop::write_response_callback, 42
operator<<
 QFTOP, 11, 12
 qftop::attribute_list, 28
 qftop::get_control, 30
 qftop::read_without_cred, 36
 qftop::read_without_cred_response, 37
 qftop::set_control, 38
 qftop::write_without_cred, 44

PARAMETERS
 qfTopMSG::pdu, 32
PDU
 qfTopMSG, 34
pre_escape
 qfTopMessage, 33
preamble
 qfTopMSG::header, 31
print_message
 QFTOP, 12
push_message
 qftop::application, 26

QFTOP
 ACK, 11
 Application, 11
 crc_init, 11
 dsrc_l7_req, 11
 dsrc_l7_resp, 11
 maximumQFTOPFrameSize, 11
 NACK, 11
 QFTOP_ACK, 10
 QFTOP_DSRC_L7_REQ, 11
 QFTOP_DSRC_L7_RESP, 11
 QFTOP_ECHO_REQ, 10
 QFTOP_ECHO_RESP, 10
 QFTOP_INIT_NOTIFICATION, 10
 QFTOP_MMI_REQ, 10
 QFTOP_NACK, 10
 QFTOP_PERS_REQ, 11
 QFTOP_PERS_RESP, 11
 QFTOP_REGISTER_APP, 11
 QFTOP_REGISTER_APP_REQ, 11
 QFTOP_TEST_REQ, 11
 QFTOP_TEST_RESP, 11
 QFTOP_TRACE_LOG_REQ, 11
 QFTOP_TRACE_LOG_RESP, 11
 QFTOP_TRANSPI_REQ, 11
 QFTOP_TRANSPI_RESP, 10
 qftop_preamble, 11
 QFTOP_ACK
 QFTOP, 10
 QFTOP_DSRC_L7_REQ
 QFTOP, 11
 QFTOP_DSRC_L7_RESP
 QFTOP, 11
 QFTOP_ECHO_REQ
 QFTOP, 10
 QFTOP_ECHO_RESP
 QFTOP, 10
 QFTOP_INIT_NOTIFICATION
 QFTOP, 10
 QFTOP_MMI_REQ
 QFTOP, 10
 QFTOP_NACK
 QFTOP, 10
 QFTOP_PERS_REQ
 QFTOP, 11
 QFTOP_PERS_RESP
 QFTOP, 11
 QFTOP_REGISTER_APP
 QFTOP, 11
 QFTOP_REGISTER_APP_REQ
 QFTOP, 11
 QFTOP_TEST_REQ
 QFTOP, 11
 QFTOP_TEST_RESP
 QFTOP, 11
 QFTOP_TRACE_LOG_REQ
 QFTOP, 11
 QFTOP_TRACE_LOG_RESP
 QFTOP, 11
 QFTOP_TRANSPI_REQ
 QFTOP, 11
 QFTOP_TRANSPI_RESP
 QFTOP, 10
 QFTOP, 9
 operator<<, 11, 12
 print_message, 12
 qftop_Types, 11
 qftop_ack, 12
 qftop_addParameter, 12
 qftop_addToCRC, 13
 qftop_application, 13
 qftop_buildMessage, 13
 qftop_clear, 15

qftop_cmd_type_t, 10
qftop_extractMessage, 16
qftop_parse, 16
qfTopMSG, 33
CRC, 34
CRC_REC, 34
HEADER, 34
PDU, 34
qfTopMSG::crc, 29
bytes, 29
word, 29
qfTopMSG::header, 30
bytes, 31
frameType, 31
length, 31
preamble, 31
sequence, 31
status, 31
syn, 31
qfTopMSG::pdu, 32
bytes, 32
messageType, 32
PARAMETERS, 32
qfTopMessage, 32
head_read, 33
head_start, 33
message, 33
message_length, 33
msg, 33
pre_escape, 33
qftop, 21
qftop_preamble
 QFTOP, 11
qftop::application, 25
 application, 26
 push_message, 26
 send_read_without_cred, 26
 send_write_without_cred, 27
 start_polling, 27
 stop_polling, 27
qftop::attribute_list, 27
 attribute_id, 28
 attribute_value, 28
 container_type, 29
 operator<<, 28
 to_bytes, 28
qftop::get_control, 29
 action, 30
 byte, 30
 has_attribute_list, 30
 has_credentials, 30
 has_iid, 30
 mode, 30
 operator<<, 30
qftop::read_response_callback, 34
 on_error, 34
 on_success, 34
qftop::read_without_cred, 35
attribute_count, 36
attribute_list, 36
bytes, 36
control, 36
element_id, 36
fragment_header, 36
length, 36
operator<<, 36
to_bytes, 35
qftop::read_without_cred_response, 36
attribute_count, 37
attribute_id, 37
attribute_value, 37
bytes, 37
container_id, 37
control, 37
element_id, 37
fragment_header, 38
header, 38
length, 38
operator<<, 37
qftop::set_control, 38
action, 39
byte, 39
fill, 39
has_credentials, 39
has_iid, 39
mode, 39
operator<<, 38
qftop::tty, 39
send_message, 41
start_reading, 41
stop_reading, 42
tty, 40
qftop::write_response_callback, 42
 on_error, 42
 on_success, 42
qftop::write_without_cred, 43
attribute_count, 44
attribute_list, 44
control, 44
element_id, 44
fragment_header, 44
length, 44
operator<<, 44
to_bytes, 43
qftop_Types
 QFTOP, 11
qftop_ack
 QFTOP, 12
qftop_addParameter
 QFTOP, 12
qftop_addToCRC
 QFTOP, 13
qftop_application
 QFTOP, 13
qftop_buildMessage
 QFTOP, 13

qftop_clear
 QFTOP, 15

qftop_client.c
 qftop_copyIn, 51
 qftop_copyOut, 52

qftop_cmd_type_t
 QFTOP, 10

qftop_copyIn
 qftop_client.c, 51

qftop_copyOut
 qftop_client.c, 52

qftop_extractMessage
 QFTOP, 16

qftop_parse
 QFTOP, 16

read_rtm_data
 tachograph::application, 25

send_message
 qftop::tty, 41

send_read_without_cred
 qftop::application, 26

send_write_without_cred
 qftop::application, 27

sequence
 qfTopMSG::header, 31

Serial Port, 18

split_on_whitespace
 string_extra, 23

start_polling
 qftop::application, 27

start_reading
 qftop::tty, 41

status
 qfTopMSG::header, 31

stop_polling
 qftop::application, 27

stop_reading
 qftop::tty, 42

string_extra, 22
 has_prefix, 22
 hex_to_bytes, 22
 split_on_whitespace, 23

syn
 qfTopMSG::header, 31

Tachograph, 8

tachograph, 23

tachograph::application
 application, 24
 read_rtm_data, 25
 write_rtm_data, 25

tachographApp/src/main/cpp/main.cpp, 45

tachographApp/src/main/cpp/string_extra.cpp, 47, 48

tachographApp/src/main.hpp/string_extra.hpp, 49, 50

tachographLib/src/main/c/qftop_client.c, 50, 52

tachographLib/src/main/cpp/qftop_application.cpp, 55

tachographLib/src/main/cpp/qftop_tty.cpp, 59, 60

tachographLib/src/main/h/qftop_client.h, 61, 63

tachographLib/src/main/hpp/qftop_application.hpp, 64, 66

tachographLib/src/main/hpp/qftop_tty.hpp, 69, 70

tachographLib/src/main/hpp/tachograph.hpp, 70, 71

tachographLib/src/main/ipp/tachograph.ipp, 72, 73

to_bytes
 qftop::attribute_list, 28
 qftop::read_without_cred, 35
 qftop::write_without_cred, 43

tty
 qftop::tty, 40

Utility, 19

word
 qfTopMSG::crc, 29

write_rtm_data
 tachograph::application, 25

References

Commission, E., 'Regulation (eu) no 165/2014 of the european parliament and of the council', 2004.

Commission, E., 'Commission implementing regulation (eu) 2016/799 of 18 march 2016 implementing regulation (eu) no 165/2014 of the european parliament and of the council laying down the requirements for the construction, testing, installation, operation and repair of tachographs and their components'. 2011.

List of abbreviations and definitions

ADEV Allan Deviation

AKOS Agency for Communication Networks and Services of the Republic of Slovenia

C-ITS Cooperative Intelligent Transport Systems

CNIT Consorzio Nazionale Interuniversitario per le Telecomunicazioni

COTS Commercial Off-The-Shelf

CS Commercial Service

DSRC Dedicated Short Range Communications

EKF Extended Kalman Filtering

GNSS Global Navigation Satellite System

GSA European GNSS Agency

IMU Inertial Mounted Unit

ITS Intelligent Transportation System

IV Intelligent Vehicle

NMA Navigation Message Authentication

OBU On Board Unit

OS Open Service

PF Particle Filtering

PoC Proof of Concept

PRS Public Regulated Service

PVT Position Velocity and Time

RF Radio Frequency

ST Smart Tachograph

SDR Software Defined Radio

TESLA Timed Efficient Stream Loss-tolerant Authentication

VU Vehicle Unit

List of figures

Figure 1. Typical scenarios for the application of CEN-DSRC to the smart tachograph	4
Figure 2. remote communication architecture	4
Figure 3. remote communication architecture	5
Figure 4. Layered protocol for CEN-DSRC in the new version of the Digital Tachograph	6
Figure 5. List of hardware components for the reader system	7
Figure 6. Image of the CEN-DSRC prototype reader during the JRC Open Day 2016	8
Figure 7. Image of the CEN-DSRC prototype reader interrogating the CEN-DSRC OBU installed in a Continental truck	9

***Europe Direct is a service to help you find answers
to your questions about the European Union.***

Freephone number (*):

00 800 6 7 8 9 10 11

(*) The information given is free, as are most calls (though some operators, phone boxes or hotels may charge you).

More information on the European Union is available on the internet (<http://europa.eu>).

HOW TO OBTAIN EU PUBLICATIONS

Free publications:

- one copy:
via EU Bookshop (<http://bookshop.europa.eu>);
- more than one copy or posters/maps:
from the European Union's representations (http://ec.europa.eu/represent_en.htm);
from the delegations in non-EU countries (http://eeas.europa.eu/delegations/index_en.htm);
by contacting the Europe Direct service (http://europa.eu/europedirect/index_en.htm) or
calling 00 800 6 7 8 9 10 11 (freephone number from anywhere in the EU) (*).

(*) The information given is free, as are most calls (though some operators, phone boxes or hotels may charge you).

Priced publications:

- via EU Bookshop (<http://bookshop.europa.eu>).

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.



EU Science Hub

ec.europa.eu/jrc



@EU_ScienceHub



EU Science Hub - Joint Research Centre



Joint Research Centre



EU Science Hub



Publications Office

doi:10.2788/866989

ISBN 978-92-79-64805-2