INTERNATIONAL STANDARD

ISO 13185-4

First edition
2020-05

Intelligent transport systems — Vehicle interface for provisioning and support of ITS Services —

Part 4: Unified vehicle interface protocol (UVIP) conformance test specification

Systèmes de transport intelligents - Interface véhicule pour la fourniture et le support de services —

Partie 4: Spécification d'essai de conformité du protocole d'interface véhicule unifié (PIVU)

Reference number
ISO 13185-4:2020(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

A list of all parts in the ISO 13185 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document has been established to define the UVIP client and server conformance tests of a common protocol interface to a vehicle UVIP server to easily exchange vehicle information data amongst nomadic and/or mobile devices, cloud servers, vehicle servers and the vehicle's Electronic Control Units (ECUs).

NOTE The abbreviation "UVIP" (Unified Vehicle Interface Protocol) derives from the original abbreviation "UGP" (Unified Gateway Protocol) see ISO 13185-2. The name was changed to avoid confusion in regard to the need of a Gateway implementation in the vehicle.

To achieve this, it is based on the Open Systems Interconnection (OSI) Basic Reference Model specified in ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers.

This document can be used by vehicle manufacturers for future vehicle design to support the design of a UVIP server to interface with NDs.

The ND applications need vehicle information data through an in-vehicle interface access method.

This document supports ITS applications which are based on ND in vehicles to operate on a common software interface to a V-ITS-S to easily exchange vehicle information data among ND, vehicle V-ITS-S and ECUs.

Figure 1 shows an overview of the UVIP Client and Server API. A UVIP Client Application on a P-ITS-S communicates with a UVIP Server Application on a V-ITS-S. The UVIP Client Application implements the UVIP Client API using ISO 13185-2. The UVIP Server Application implements the UVIP Server API using ISO 13185-2.

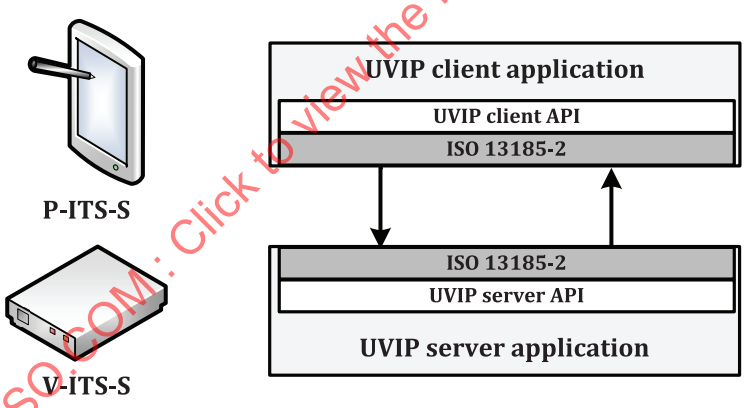

Figure 1 — UVIP client on clouds and vehicle server

Intelligent transport systems — Vehicle interface for provisioning and support of ITS Services —

Part 4:
Unified vehicle interface protocol (UVIP) conformance test specification

1 Scope

This document specifies a conformance test for a UVIP server and client system developer assessment of self-conformance of the supplier's UVIP server and client system. The conformance test cases follow the use cases definition of ISO 13185-1 and the requirements stated in ISO 13185-2 and ISO 13185-3.

The purpose of this document is to provide information to the UVIP server and client system provider to build and test the UVIP server and client system against the conformance test cases.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 13185-1, *Intelligent transport systems (ITS) — Vehicle interface for provisioning and support of ITS services — Part 1: General information and use case definition*

ISO 13185-2, *Intelligent transport systems — Vehicle interface for provisioning and support of ITS services — Part 2: Unified gateway protocol (UGP) requirements and specification for vehicle ITS station gateway (V-ITS-SG) interface*

ISO 13185-3, *Intelligent transport systems — Vehicle interface for provisioning and support of ITS Services — Part 3: Unified vehicle interface protocol (UVIP) server and client API specification*

ISO 21217, *Intelligent transport systems — Communications access for land mobiles (CALM) — Architecture*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 13185-1, ISO 13185-2, ISO 13185-3, and ISO 21217 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.2 Abbreviated terms

For the purposes of this document, the abbreviations given in ISO 13185-1 and the following apply.

CTC	conformance test case
CTP	conformance test plan
HVAC	heating, ventilation and air conditioning
ND	nomadic device
OBD	on-board diagnostics
UC	use case

4 Conventions

This document is based on the conventions discussed in the OSI Service Conventions in ISO/IEC 10731:1994[2] as they apply for communication services. The vehicle data transfer protocol is applicable to OSI layers 5, 6 and 7.

5 Conformance test plan (CTP) basic principles and clustering

5.1 Basic principles for CTC definition

Basic principles have been established as a guideline to define the UVIP implementation conformance test cases:

— BP1: The primary objective of the CTC is to support a company which has developed UVIP server and client systems in the assessment of self-conformance of the UVIP server and client system.

— BP2: The CTC addresses the UVIP implementation.

— BP3: The CTC is a positive test in order to test the proper functioning of the UVIP implementation i.e., correct input data provides correct output data.

— BP4: The person performing the CTC shall verify that the purpose of the use case is achieved following the descriptions of the UVIP regarding the implementation of the use case and the steps to enter the input and to obtain the output according to ISO 13185-1.

— BP5: The name of the CTC should be the same as the name of the use case (see ISO 13185-1) or requirement (see ISO 13185-2).

— BP6: Each CTC should have a preamble (setup state).

— BP7: Classification for each CTC is included in order to support the classification criteria specified for use cases and requirements.

— BP8: A CTC is only applicable if the use case or requirement is supported by the UVIP client.

— BP9: Some CTCs may require payment or a valid subscription before processing the next step.

CAUTION — The person performing the conformance test is responsible for entering valid data and correctly executing necessary actions in order to maintain the integrity of the implementation of the UVIP implementation between the UVIP client and the UVIP server (V-ITS-SG).

5.2 CTC structure

5.2.1 CTC – General structure

5.2.1.1 CTC reference number and title [CTC_...] [title]

Each CTC is structured by six (6) subclauses:

— Overview,

— Test purpose,

— Configuration,

— Preamble (setup state),

— Test execution,

— Postamble.

In the following, details and examples for each of these titles (ordered list) are given.

A reference to the corresponding CTC requirement is specified via a unique abbreviation, number and title as follows:

— [CTC_UC x.y] of ISO 13185-1;

where:

— 'x', 'y' are numeric numbers as assigned in ISO 13185-1.

5.2.1.2 Overview

This is a conformance test for checking the UC x.y followed by the subject description as specified in ISO 13185-2.

5.2.1.3 Test purpose

The test purpose gives a short description of the relevant CTCs and a reference to the corresponding requirement specified in ISO 13185-1 and ISO 13185-2.

NOTE The CTC approach depends on the definition of the referenced requirement in ISO 13185-1 and ISO 13185-2.

5.2.1.4 Configuration

The configuration describes the CTC scenario pre-requisites.

5.2.1.5 Preamble (setup state)

The "Preamble" defines preconditions which are used for preparation and initialisation of the UVIP with a view to performing the specific conformance test. For example, a precondition could be the successful establishment of a connection between the UVIP client and the UVIP server (V-ITS-SG).

5.2.1.6 Test execution

Test execution of a single CTC is organized in steps. These steps are described in the example in Table 1.

Table 1 — CTC execution example

Step #	Description
1	Description of 1st test step.
2	Description of 2nd test step.
N	Description of Nth test step.

5.2.1.7 Postamble

The "Postamble" defines post conditions which are used to return the UVIP protocol implementation back to a definite state.

5.2.2 Result criteria

The CTC result criteria are composed of three different results as listed in Table 2.

Table 2 — CTC result criteria

Result	Definition
Pass	The CTC purpose was achieved as expected.
Deficiency	The CTC purpose was achieved with opportunities for improvement identified and documented in detail.
Fail	The CTC purpose was not achieved. Reason(s) shall be documented in detail.

5.3 CTC system setup

Figure 2 shows the conformance test system setup.

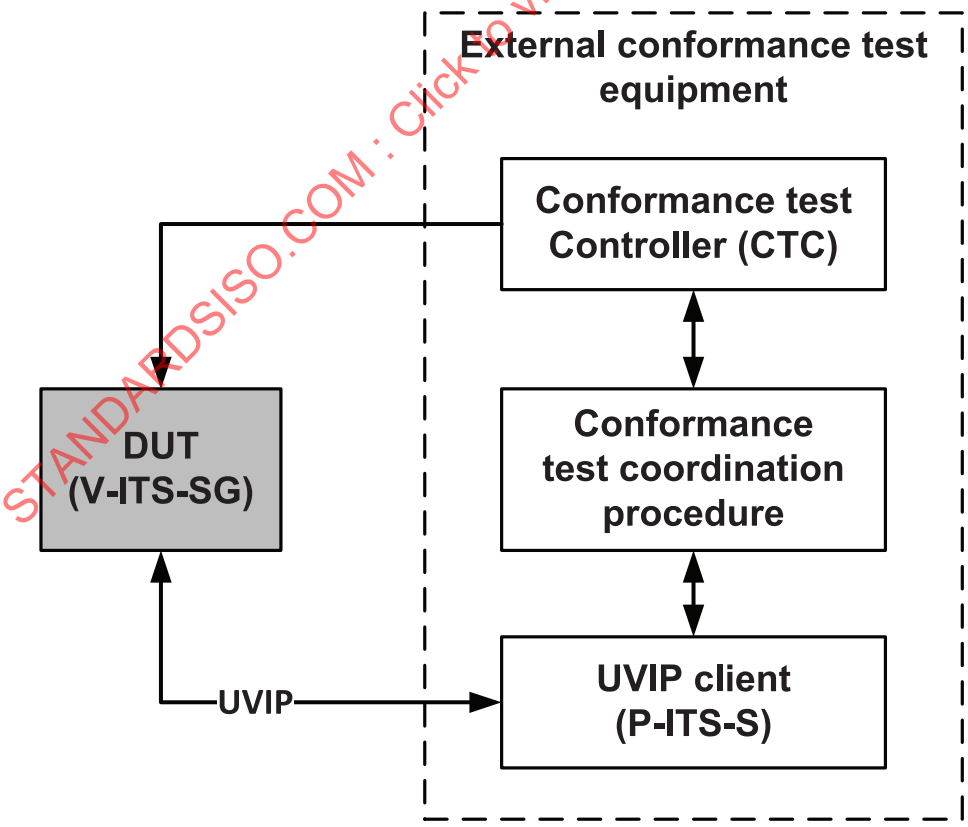

Figure 2 — Conformance test system setup

5.4 CTC clustering

5.4.1 General

Subclause 5.4.2 provides an overview of all conformance test clusters and the associated test cases for mandatory and optional use cases and requirements. Each test case is assigned to one CTC cluster. The clusters cover technical areas, where the assigned CTC(s) apply.

5.4.2 Main CTC clusters

Table 3 defines the main CTC clusters.

Table 3 — Main CTC clusters

# - Name of CTC cluster	Brief description	CTC reference
1 – Requesting V-ITS-SG and vehicle identification related information	This cluster describes the CTCs that check generic information about the V-ITS-SG itself and the vehicle's electronic systems. It also checks how information about the available ECUs and the available data is requested from the V-ITS-SG.	CTC_UC 1.1 – Browsing available V-ITS-SGs CTC_UC 1.2 – Browsing supported ECUs CTC_UC 1.3 – Browsing supported data Ids
2 – Requesting vehicle and ECU data values	This cluster describes the CTCs that check the request of data IDs and values.	CTC_UC 2.1 – Requesting data ID values for single usage CTC_UC 2.2 – Requesting data ID values for repeated usage CTC_UC 2.3 – Requesting data ID text and data type information CTC_UC 2.4 – Requesting data type definitions CTC_UC 2.5 – Requesting all available text and data type information
3 – Requesting and clearing DTCs and related data	This cluster describes the CTCs that check the request of DTCs, additional DTC data, clearing of DTCs, and the sending of DTC and status based on SendOnChange.	CTC_UC 3.1 – Requesting DTCs CTC_UC 3.2 – Requesting additional DTC data CTC_UC 3.3 – Clearing DTCs CTC_UC 3.4 – SendOnChange – Provide DTC and status
4 – Unsolicited V-ITS-SG messages	This cluster describes the CTCs that check SendOnEvent information.	CTC_UC 4.1 – SendOnEvent — Emergency situation CTC_UC 4.2 – SendOnEvent — Critical driving situation CTC_UC 4.3 – SendOnEvent — Safety situation CTC_UC 4.4 – SendOnEvent — Warning situation CTC_UC 4.5 – SendOnEvent — Data ID value matches threshold
5 – Real-time data transmission	This cluster describes the CTCs that check the sending of real-time data and DTCs.	CTC_UC 5.1 – Real-time data ID value measurement CTC_UC 5.2 – Real-time DTC reporting
6 – Controlling /adjusting various equipment of the vehicle	This cluster describes the CTCs that check the lean settings of customer profile, the control of a convenience system, and the control of EV charging.	CTC_UC 6.1 – Learn settings of customer profile CTC_UC 6.2 – Control convenience system CTC_UC 6.3 – Control charging for EV

Table 3 *(continued)*

# - Name of CTC cluster	Brief description	CTC reference
7 – Writing short- and long-term data to V-ITS-SG	This cluster describes the CTCs that check writing of data in short- and long-term V-ITS-SG memory.	CTC_UC 7.1 – Write data to V-ITS-SG's memory CTC_UC 7.2 – Write vehicle profile to V-ITS-SG's memory CTC_UC 7.3 – Enable/disable functional system related data IDs CTC_UC 7.4 – Write data ID thresholds to V-ITS-SG's memory
8 – V-ITS-SG accessibility restrictions and firewall protection cluster	This cluster describes the CTCs that check the secure access and firewall functionality.	CTC_UC 8.1 – Secure access to V-ITS-SG CTC_UC 8.2 – Request V-ITS-SG firewall status CTC_UC 8.3 – Confguration of V-ITS-SG frewall
9 – V-ITS-SG special features	This cluster describes the CTCs that check the general data exchange, the V-ITS-SG activation mode, and the upload of EventLogFile from the V-ITS-SG.	CTC_UC 9.1 – General data exchange CTC_UC 9.2 – V-ITS-SG activation mode CTC_UC 9.3 – Upload EventLogFile from V-ITS-SG
10 – Vehicle diagnostics	This cluster describes the CTCs that check the functional OBD group, the perform enhanced OBD function, and the upload of VSOCLogFile from the V-ITS-SG.	CTC_UC 10.1 – Perform functional group OBD CTC_UC 10.2 – Perform enhanced OBD CTC_UC 10.3 – Upload VSOCLogFile from V-ITS-SG
11 – Electric vehicle system status	This cluster describes the CTCs that check the monitor battery charge station, the monitoring of the connection between the charging station and the EV, and the battery charge start/stop notification.	CTC_UC 11.1 – Monitor battery charge status CTC_UC 11.2 – Monitor connection between charging station and EV CTC_UC 11.3 – Battery charge start/stop notification
12 – V-ITS-SG maintenance	This cluster describes the CTCs that check the update of the core software of the V-ITS-SG, the function to perform a key Off/On reset in the V-ITS-SG, the function to perform a reset in the V-ITS-SG, the upload of the V-ITS-SG configuration file(s), and the download of the V-ITS-SG configuration file(s).	CTC_UC 12.1 – Update core software of V-ITS-SG CTC_UC 12.2 – Perform a key Off/On reset in V-ITS-SG CTC_UC 12.3 – Perform a reset in V-ITS-SG CTC_UC 12.4 – Upload V-ITS-SG configuration file CTC_UC 12.5 – Download V-ITS-SG configuration file

6 CT cluster 1 – Requesting V-ITS-SG and vehicle identification related information

6.1 CTC_UC 1.1 – Browsing available V-ITS-SGs

6.1.1 Overview

This is a conformance test for checking the UC 1.1 when identifying available V-ITS-SGs as specified in ISO 13185-1.

6.1.2 Test purpose

The purpose of the conformance test is to verify that after an IP discovery and sending a `UVIPPackage`, created from an `authenticationReq`, by the Client, the Client method `authenticationConf` is executed.

6.1.3 Configuration

Setup two V-ITS-Ss with configurations.

6.1.4 Preamble (setup state)

Setup conditions: both V-ITS-Ss have IP addresses in the same subnet (255.255.255.0).

6.1.5 Test execution

Table 4 defines the test execution.

Table 4 — CTC_UC 1.1 – Browsing available V-ITS-SGs test execution

Step #	Description
1	Connect to a subnet with mask 255.255.255.0.
2	Discover all available IP devices in this subnet.
3	For all available IP devices: Establish an IP connection. If available, do steps 6 to X, otherwise go on with next device.

4	`//Create an authentication request`
	`String authenticationKey = ...;`
	`UVIPPackage reqPackage = uvipClient.authenticationReq(authenticationKey);`

5	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the authentication confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void authenticationConf(AuthorizationBits authorization)`
	`  throws NumberedException {`
	`  //request vehicle information parameter only`
	`  tcpClient.sendCall(uvipClient.getSupportedDataReq(`
	`  SupportedDataFilter.vehicle_info_only, null, null, null));`
	`  );`
	`  }`
	`  public void getSupportedDataConf(Vector<EcuDataParam> params)`
	`  throws NumberedException {`
	`  //display own ip address – IP address of a V-ITS-S`
	`  //display the names of all vehicle information data parameters`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

6	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
7	Check if the V-ITS-Ss IP addresses are displayed with their vehicle information.

6.1.6 Postamble (setup state)

Setup conditions: None.

6.2 CTC_UC 1.2 – Browsing supported ECUs

6.2.1 Overview

This is a conformance test for checking the UC 1.2 when identifying supported ECUs as specified in ISO 13185-1.

6.2.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getSupportedDataReq`, by the Client, the Client method `getSupportedDataConf` is executed, providing the supported ECUs.

6.2.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing some parameters assigned to several ECUs.

6.2.4 Preamble (setup state)

Setup conditions: None.

6.2.5 Test execution

Table 5 defines the test execution.

Table 5 — CTC_UC 1.2 – Browsing supported ECUs test execution

Step #	Description

1	`//Create supported data request with ecu data, without any filter`
	`UVIPPackage reqPackage = uvipClient.getSupportedDataReq(`
	`  SupportedDataFilter.with_ecu_data, null, null, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the supported data confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getSupportedDataConf(Vector<EcuDataParam> params)`
	`  throws NumberedException {`
	`  // display the names of the ECUs the supported data parameters are assigned to`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if the configured supported ECUs are displayed.

6.2.6 Postamble (setup state)

Setup conditions: None.

6.3 CTC_UC 1.3 – Browsing supported data Ids

6.3.1 Overview

This is a conformance test for checking the UC 1.3 when identifying supported data identifiers as specified in ISO 13185-1.

6.3.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getSupportedDataReq`, by the Client, the Client method `getSupportedDataConf` is executed, providing the supported data parameters.

6.3.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing some parameters.

6.3.4 Preamble (setup state)

Setup conditions: None.

6.3.5 Test execution

Table 6 defines the test execution.

Table 6 — CTC_UC 1.3 – Browsing supported data Ids test execution

Step #	Description

1	`//Create supported data request without ecu data, without any filter`
	`UVIPPackage reqPackage = uvipClient.getSupportedDataReq(`
	`  SupportedDataFilter.without_ecu_data, null, null, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the supported data confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getSupportedDataConf(Vector<EcuDataParam> params)`
	`  throws NumberedException {`
	`  //display the ids, names of the supported data parameters`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if the configured supported data parameters are displayed.

6.3.6 Postamble (setup state)

Setup conditions: None.

7 CT cluster 2 – Requesting vehicle and ECU data values

7.1 CTC_UC 2.1 – Requesting data ID values for single usage

7.1.1 Overview

This is a conformance test for checking the UC 2.1 when requesting data IDs in a way they can be directly displayed on the ND as specified in ISO 13185-1.

7.1.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getValueReq`, by the Client, the Client method `getValueConf` is executed, providing the values of the requested data parameters.

7.1.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing at least 3 parameters P1, P2 and P3.

7.1.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids only return P1 and P2.

7.1.5 Test execution

Table 7 defines the test execution.

Table 7 — CTC_UC 2.1 – Requesting data ID values for single usage test execution

Step #	Description

1	`//Create get value request`
	`Integer rvIdP1 = ...;`
	`UVIPPackage reqPackage = uvipClient.getValueReq(0, rvIdP1, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {`
	`  ...  //see steps 4 to 7`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	`//Inside the getValueConf:` `//Assert the params.size() to be 1, because the request contains one parameter`
5	`//Assert the first parameter (param.get(0)) to be of type EcuDataParam (not null)` `  for this first time`
6	`//Ensure the first parameter has a value of the correct type`

7	`//Create get value request with parameter P3 which is not supported`
	`Integer rvIdP3 = ...;`
	`UVIPPackage reqPackage = uvipClient.getValueReq(0, rvIdP3, null);`

8	`//Send request message for second parameter` `tcpClient.sendCall(reqPackage, threadUVIP);`
10	`//Inside the getValueConf:` `//Assert the params.size() to be 1, because the request contains one parameter`
11	`//Assert the first parameter (param.get(0)) to be null, because parameter P2` `//is not supported`

7.1.6 Postamble (setup state)

Setup conditions: None.

7.2 CTC_UC 2.2 – Requesting data ID values for repeated usage

7.2.1 Overview

This is a conformance test for checking the UC 2.2 when requesting data IDs in a way that the response message is optimised for repeated usage (less size, faster transmission) as specified in ISO 13185-1.

7.2.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getValueReq`, by the Client, the Client method `getValueConf` is executed multiple times, providing the values of the requested data parameters again and again.

7.2.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing at least 3 parameters P1, P2 and P3.

7.2.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids only return P1 and P2, the vehicle supports these 3 parameters.

7.2.5 Test execution

Table 8 defines the test execution.

Table 8 — CTC_UC 2.2 – Requesting data ID values for repeated usage test execution

Step #	Description

1	`//Create a value request with 2 parameters every second (1000 milliseconds)`
	`Integer rvIdP1 = ...;`
	`Integer rvIdP2 = ...;`
	`UVIPPackage reqPackage =`
	`  uvipClient.getValueReq(1000, new Integer[ { rvIdP1, rvIdP2}, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`public int counter = 0;`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {`
	`  counter++;`
	`  //display the parameters with their values labelled with counter`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if the 2 parameters are displayed every second and the counter is incremented.

7.2.6 Postamble (setup state)

Setup conditions: None.

7.3 CTC_UC 2.3 – Requesting data ID text and data type information

7.3.1 Overview

This is a conformance test for checking the UC 2.3 when gathering text and data type information, after a data ID for repeated usage has been requested as specified in ISO 13185-1.

All data parameter texts and data type information are already included in the EcuDataParam object returned by the methods defined in CTC_UC 1.3 – Browsing supported data Ids, CTC_UC 2.1 – Requesting data ID values for single usage, CTC_UC 2.2 – Requesting data ID values for repeated usage. Grab the wanted information from these objects.

7.4 CTC_UC 2.4 – Requesting data type definitions

7.4.1 Overview

This is a conformance test for checking the UC 2.4 when gathering information about all available labels and data types from the V-ITS-SG as specified in ISO 13185-1.

The data type definitions are already included in the EcuDataParam object returned by the methods defined in CTC_UC 1.3 – Browsing supported data Ids, CTC_UC 2.1 – Requesting data ID values for single usage, CTC_UC 2.2 – Requesting data ID values for repeated usage. Grab the wanted information from these objects.

7.5 CTC_UC 2.5 – Requesting all available text and data type information

7.5.1 Overview

This is a conformance test for checking the UC 2.5 when getting information about DTCs and associated status in the vehicle's ECU(s) as specified in ISO 13185-1.

All available texts and data type information are already included in the EcuDataParam object returned by the methods defined in CTC_UC 1.3 – Browsing supported data Ids, CTC_UC 2.1 – Requesting data ID values for single usage, CTC_UC 2.2 – Requesting data ID values for repeated usage. Grab the wanted information from these objects.

8 CT cluster 3 – Requesting and clearing DTCs and related data

8.1 CTC_UC 3.1 – Requesting DTCs

8.1.1 Overview

This is a conformance test for checking the UC 3.1 when obtaining information about current active DTCs from the vehicle’s V-ITS-SG as specified in ISO 13185-1.

8.1.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getDtcInfoReq`, by the Client, the Client method `getDtcInfoConf` is executed possibly multiple times, providing the requested DTCs.

8.1.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle defining some DTCs.

8.1.4 Preamble (setup state)

Setup conditions: The vehicle supports some of the configured DTCs and has at least one active DTC.

8.1.5 Test execution

Table 9 defines the test execution.

Table 9 — CTC_UC 3.1 – Requesting DTCs test execution

Step #	Description

1	`//Create a get DTC info request without ECU filter, environment data and`
	`//condition, checking the DTCs every ten seconds`
	`UVIPPackage reqPackage = uvipClient.getDtcInfoReq(10000, null, false, null);`

2	`//Define Thread implementing the UVIP configuration interface`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getDtcInfoConf(Vector<DtcInfo> dtcInfos) throws NumberedException {`
	`  //display the DTC info (DTC Base id, DTC Symptom id, status, severity and`
	`  //DTC class)`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if at least one known DTC is displayed.

8.1.6 Postamble (setup state)

Setup conditions: None.

8.2 CTC_UC 3.2 – Requesting additional DTC data

8.2.1 Overview

This is a conformance test for checking the UC 3.2 when obtaining information about additional stored data for a specific DTC as specified in ISO 13185-1.

8.2.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getDtcInfoReq`, by the Client, the Client method `getDtcInfoConf` is executed possibly multiple times, providing the requested DTCs including environment data.

8.2.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle defining DTC Base definitions including `dataParamMappings` for the environment data.

8.2.4 Preamble (setup state)

Setup conditions: The vehicle supports some of the configured DTCs including environment data and has at least one active DTC.

8.2.5 Test execution

Table 10 defines the test execution.

Table 10 — CTC_UC 3.2 – Requesting additional DTC data test execution

Step #	Description

1	`//Create a get DTC info request without ECU filter and condition but with`
	`//environment data enabled, checking the DTCs every ten seconds`
	`UVIPPackage reqPackage = uvipClient.getDtcInfoReq(10000, null, true, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getDtcInfoConf(Vector<DtcInfo> dtcInfos) throws NumberedException {`
	`  //display the DTC info (DTC Base id, DTC symptom id, status, severity,`
	`  //DTC class) with environment data (list of data parameter with their values)`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if at least one known DTC is displayed including the environment data.

8.2.6 Postamble (setup state)

Setup conditions: None.

8.3 CTC_UC 3.3 – Clearing DTCs

8.3.1 Overview

This is a conformance test for checking the UC 3.3 when clearing DTCs and appropriate data from the vehicle's ECUs as specified in ISO 13185-1.

8.3.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `clearDtcInfoReq`, by the Client, the Client method `positiveConf` is executed returning the `callSequenceNumber` or the `clearDtcInfoReq`.

8.3.3 Configuration

Prepare a `VIDFConfiguration` as defined in CTC_UC 3.1 – Requesting DTCs or CTC_UC 3.2 – Requesting additional DTC data.

8.3.4 Preamble (setup state)

Setup conditions: The vehicle supports some of the configured DTCs and has at least one active DTC. CTC_UC 3.1 – Requesting DTCs or CTC_UC 3.2 – Requesting additional DTC data have been executed successfully containing at least one DTC info.

8.3.5 Test execution

Table 11 defines the test execution.

Table 11 — CTC_UC 3.3 – Clearing DTCs test execution

Step #	Description
1	`//Create a clear DTC info request without ECU filter` `public UVIPPackage reqPackage = uvipClient.clearDtcInfoReq(null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void positiveConf(int callChoice, int callSequenceNumber)`
	`  throws NumberedException {`
	`  //if DTCs are cleared, re-read the DTC infos`
	`  if (callSequenceNumber == reqPackage.getCallSequenceNumber()) {`
	`  tcpClient.sendCall(uvipClient.getDtcInfoReq(0, null, false, null),`
	`  threadUVIP);`
	`  }`
	`  }`
	`  public void getDtcInfoConf(Vector<DtcInfo> dtcInfos) throws NumberedException {`
	`  //display the list of DTC infos`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check that no more DTCs are displayed.

8.3.6 Postamble (setup state)

Setup conditions: None.

8.4 CTC_UC 3.4 – SendOnChange – Provide DTC and status

8.4.1 Overview

This is a conformance test for checking the UC 3.4 when the ND requests DTCs by “SendOnChange” as specified in ISO 13185-1.

The Server implementation always sends all DTCs, not only the changed one. So no DTCs get lost. Verify by executing CTC_UC 3.1 – Requesting DTCs or CTC_UC 3.2 – Requesting additional DTC data.

9 CT cluster 4 – Unsolicited V-ITS-SG messages

9.1 CTC_UC 4.1 – SendOnEvent — Emergency situation

9.1.1 Overview

This is a conformance test for checking the UC 4.1 when the V-ITS-SG shall send an unsolicitated message to the ND with pre-configured emergency related data items (i.e. location information) immediately after receiving the emergency trigger from the vehicle system(s) as specified in ISO 13185-1.

This use case is not supported by ISO 13185-2 or ISO 13185-3.

9.2 CTC_UC 4.2 – SendOnEvent — Critical driving situation

9.2.1 Overview

This is a conformance test for checking the UC 4.2 when the V-ITS-SG shall send an unsolicited message to the ND with pre-configured critical driving situation related data items immediately after receiving the critical driving situation trigger from the vehicle system(s) as specified in ISO 13185-1.

This use case is not supported by ISO 13185-2 or ISO 13185-3.

9.3 CTC_UC 4.3 – SendOnEvent — Safety situation

9.3.1 Overview

This is a conformance test for checking the UC 4.3 when the V-ITS-SG shall send an unsolicited message to the ND with pre-configured safety related data items immediately after receiving the safety trigger from the vehicle system(s) as specified in ISO 13185-1.

This use case is not supported by ISO 13185-2 or ISO 13185-3.

9.4 CTC_UC 4.4 – SendOnEvent — Warning situation

9.4.1 Overview

This is a conformance test for checking the UC 4.4 when the V-ITS-SG shall send an unsolicited message to the ND with pre-configured warning related data items immediately after receiving the warning trigger identified by the V-ITS-SG warning related data analysis as specified in ISO 13185-1.

This use case is not supported by ISO 13185-2 or ISO 13185-3.

9.5 CTC_UC 4.5 – SendOnEvent — Data ID value matches threshold

9.5.1 Overview

This is a conformance test for checking the UC 4.5 when the ND and the V-ITS-SG shall be capable of providing data items based on a “SendOnEvent” trigger for ND selected data ID values (if supported by the vehicle system and reported to be supported by the V-ITS-SG) as specified in ISO 13185-1.

This use case is not supported by ISO 13185-2 or ISO 13185-3.

10 CT cluster 5 – Real-time data transmission

10.1 CTC_UC 5.1 – Real-time data ID value measurement

10.1.1 Overview

This is a conformance test for checking the UC 5.1 when the V-ITS-SG shall be capable of sending real-time optimized data ID value measurements to the ND as specified in ISO 13185-1. The data item value refresh rate shall be configurable by the ND to avoid burst message situations between the V-ITS-SG and ND.

Use CTC_UC 2.2 – Requesting data ID values for repeated usage. Vary the `testInterval` (first parameter of the `getValueReq` method) to check if no burst message situation occurs.

10.2 CTC_UC 5.2 – Real-time DTC reporting

10.2.1 Overview

This is a conformance test for checking the UC 5.2 when the V-ITS-SG shall be capable of reporting real-time optimized DTC values to the ND as specified in ISO 13185-1. The DTC reporting refresh rate shall be configurable by the ND to avoid burst message situations between the V-ITS-SG and ND.

Use CTC_UC 3.1 – Requesting DTCs. Vary the `testInterval` (first parameter of the `getDtcInfoReq` method) to check if no burst message situation occurs.

11 CT cluster 6 – Controlling /adjusting various equipment of the vehicle

11.1 CTC_UC 6.1 – Learn settings of customer profile

11.1.1 Overview

This is a conformance test for checking the UC 6.1 when the V-ITS-SG is setup with the use of the ND to learn the vehicle system settings of a customer profile, e.g. store learned settings of electrically controlled seat, sun roof, window position, preset radio stations as specified in ISO 13185-1.

There is no function to store and reset values at the V-ITS-SG (server side). But it is possible on the client side:

The customer moves the electronically controlled systems in the desired position. When done, the customer calls the 'store customer profile' function in the ND app (see test execution steps 1 to 3) and chooses a name for this profile. The next time the customer enters the vehicle and selects 'load customer profile' by selecting the profile name (see test execution steps 5 to 7) in the ND app, the ND sends UVIP request to set the stored customer profile settings.

11.1.2 Test purpose

The purpose of the conformance test is to verify that after storing the customer profile settings, changing all vehicle systems and loading the customer profile settings, all settings are equal to the settings at start-time.

11.1.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing all provided customer profile settings data parameters, e.g. electrically controlled seats, sun roof, window position, mirror position, pre-set radio stations. Additionally, define a data parameter group 'customer profile' containing these customer profile settings.

11.1.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids support all of the defined customer profile settings.

11.1.5 Test execution

Table 12 defines the test execution.

Table 12 — CTC_UC 6.1 – Learn settings of customer profile test execution

Step #	Description

1	`//Create get value request to read all customer profile settings`
	`Integer rvIdCustomerProfile = ...;`
	`UVIPPackage reqPackage = uvipClient.getValueReq(0,rvIdCustomerProfile, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {`
	`  //display all parameters and their values`
	`  //store all values in a file with the customer profile with at least`
	`  //ecuId, rvId, value`
	`  }`
	`  public void positiveConf() throws NumberedException {`
	`  //display end of setting the customer profile`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Change some/all of the customer profile settings in the vehicle.
5	`//Load all values from the customer profile settings and store it to` `Vector<DataParamValueMapping> vCustomerProfile = ...;`
6	`//Send set value request to set the customer profile` `tcpClient.sendCall(uvipClient.setValueReq(vCustomerProfile), threadUVIP);`

Table 12 *(continued)*

Step #	Description
7	After end of customer profile settings is displayed, verify if the settings are the same as before executing the test.

11.1.6 Postamble (setup state)

Setup conditions: None.

11.2 CTC_UC 6.2 – Control convenience system

11.2.1 Overview

This is a conformance test for checking the UC 6.2 when controlling the convenience systems in the vehicle, e.g. HVAC heating, ventilation and air conditioning, vehicle speed sensitive door locking and unlocking as specified in ISO 13185-1.

11.2.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `setValueReq`, by the Client, the Client method `positiveConf` is executed and the settings are done in the vehicle.

11.2.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing the convenience systems settings, e.g. HVAC temperature of driver side. Additionally, define a data parameter group 'convenience systems' containing all of these parameters.

11.2.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids support all of the defined convenience systems settings.

11.2.5 Test execution

Table 13 defines the test execution.

Table 13 — CTC_UC 6.2 – Control convenience system test execution

Step #	Description

1	`//Create get value request to read all convenience systems settings`
	`Integer rvIdConvenienceSystems = ...;`
	`UVIPPackage reqPackage = uvipClient.getValueReq(0, rvIdConvenienceSystems, null);`

Table 13 *(continued)*

Step #	Description

2	//Define Thread implementing the UVIP configuration interface,
	//handling the value confirmation of the defined request
	Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {
	...
	public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {
	//display all parameters and their values on a page and let the user
	//make changes to the values
	}
	public void positiveConf() throws NumberedException {
	//display end of setting the convenience systems
	}
	}
	ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);

3	//Send request message
	tcpClient.sendCall(reqPackage, threadUVIP);

4	After displaying all convenience systems settings in an editor, the user can make changes to the values.
5	//Take all changed values from the editor and store it to Vector<DataParamValueMapping> vConvenienceSystems = ...;

6	//Send set value request to set the convenience systems
	tcpClient.sendCall(uvipClient.setValueReq(vConvenienceSystems), threadUVIP);

7	After end of convenience systems settings is displayed, verify if the settings are as defined in the editor.

11.2.6 Postamble (setup state)

Setup conditions: None.

11.3 CTC_UC 6.3 – Control charging for EV

11.3.1 Overview

This is a conformance test for checking the UC 6.3 when control EV to start/stop for charging procedure (based on the ISO 15118 series) as specified in ISO 13185-1.

11.3.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `controlValueReq`, by the Client, the Client method `controlValueConf` is executed.

11.3.3 Configuration

Prepare a `VIDFConfiguration` containing the charging parameters.

11.3.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids support the defined charging parameters.

11.3.5 Test execution

Table 14 defines the test execution.

Table 14 — CTC_UC 6.3 – Control charging for EV test execution

Step #	Description

1	`//Create control value request to control the charging using charging values`
	`Integer[ rvIdsCharging = ...;`
	`Vector<DataParamValue> vChargingValues = ...;`
	`UVIPPackage reqPackage = uvipClient.controlValueReq(0, rvIdsCharging,`
	`  vChargingValues, ExecutionType.start);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void controlValueConf(ExecutionStatus status,`
	`  Vector<EcuDataParam> params) throws NumberedException {`
	`  //display the execution status (in progress, pass or fail)`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	When displaying execution status 'in progress', check if the vehicle is charging.
5	When displaying execution status 'pass', check if the vehicle is completely charged.

11.3.6 Postamble (setup state)

Setup conditions: None.

12 CT cluster 7 – Writing short- and long-term data to V-ITS-SG

12.1 CTC_UC 7.1 – Write data to V-ITS-SG's memory

12.1.1 Overview

This is a conformance test for checking the UC 7.1 when ND writes vehicle's licence plate number, VIN, etc. to V-ITS-SG's long-term memory as specified in ISO 13185-1.

12.1.2 Test purpose

For all OBD vehicles the VIN is available and should not be saved in any additional long-term memory, but other static information, which is normally not readable from the vehicle, i.e. license plate number, vehicle type or vehicle model, can be saved permanently. The saving of this data cannot be part of any private app, but instead, these parameters should be defined by the configuration.

The purpose of the conformance test is now to verify that after sending a `UVIPPackage`, created from a `getValueReq`, by the Client, the Client method `getValueConf` returns the configured fixed values for the vehicle information data parameters.

12.1.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing vehicle information, e.g. license plate number, vehicle type or vehicle model. Define fixed values for the vehicle information parameters.

12.1.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids support the defined vehicle information parameters.

12.1.5 Test execution

Table 15 defines the test execution.

Table 15 — CTC_UC 7.1 – Write data to V-ITS-SG's memory test execution

Step #	Description

1	`//Create a value request with an array of vehicle information parameters`
	`Integer[ rvIdsVehicleInfo = ...;`
	`UVIPPackage reqPackage =`
	`  uvipClient.getValueReq(0, rvIdsVehicleInfo, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {`
	`  //display the parameters with their values`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if the vehicle information parameters are displayed.

12.1.6 Postamble (setup state)

Setup conditions: None.

12.2 CTC_UC 7.2 – Write vehicle profile to V-ITS-SG's memory

12.2.1 Overview

This is a conformance test for checking the UC 7.2 when ND writes vehicle profile (vehicle manufacturer published vehicle data) to V-ITS-SG 's long term memory, e.g. average fuel consumption, CO^2/km as specified in ISO 13185-1.

All static information, which is normally not readable from the vehicle, can be configured as fixed value in a `VIDFConfiguration`. See CTC_UC 7.1 – Write data to V-ITS-SG's memory.

12.3 CTC_UC 7.3 – Enable/disable functional system related data IDs

12.3.1 Overview

This is a conformance test for checking the UC 7.3 when ND enables/disables functional system (e.g. navigation system) related data IDs to be sent periodically for transmission to ITS infrastructure as specified in ISO 13185-1.

Within a `VIDFConfiguration` it is possible to define data parameter groups for every functional system. It is possible to call the get value request with one of these data parameter groups to get all data parameter values. See CTC_UC 2.1 – Requesting data ID values for single usage or CTC_UC 2.2 – Requesting data ID values for repeated usage.

12.4 CTC_UC 7.4 – Write data ID thresholds to V-ITS-SG's memory

12.4.1 Overview

This is a conformance test for checking the UC 7.4 when the ND writes specific data ID thresholds to V-ITS-SG's short/long term memory to trigger event messages (SendOnEvent) as specified in ISO 13185-1.

In the current version it is not possible to define thresholds for parameters. In an update of 13185-2, data parameter limits can be defined for every data parameter inside the `VIDFConfiguration`.

13 CT cluster 8 – V-ITS-SG accessibility restrictions and firewall protection cluster

13.1 CTC_UC 8.1 – Secure access to V-ITS-SG

13.1.1 Overview

This is a conformance test for checking the UC 8.1 when SecurityAccess provides a means to access data and services which have restricted access for security, safety or other reasons as specified in ISO 13185-1.

13.1.2 Test purpose

The purpose of the conformance test is to verify that after connecting to a V-ITS-SG it is necessary to send an `authenticationReq VIPPackage` by the Client. This shall result into an `authenticationConf`.

13.1.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle.

13.1.4 Preamble (setup state)

Setup conditions: None.

13.1.5 Test execution

Table 16 defines the test execution.

Table 16 — CTC_UC 8.1 – Secure access to V-ITS-SG test execution

Step #	Description
1	//Create an authentication request String authenticationKey = ...; UVIPPackage reqPackage = uvipClient.authenticationReq(authenticationKey);

2	//Define Thread implementing the UVIP configuration interface,
	//handling the authentication confirmation of the defined request
	Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {
	...
	public void authenticationConf(AuthorizationBits authorization)
	throws NumberedException {
	//display authorizaton rights
	}
	}
	ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);

3	//Send request message
	tcpClient.sendCall(reqPackage, threadUVIP);

4	Verify the authorization rights with the displayed ones.

13.1.6 Postamble (setup state)

Setup conditions: None.

13.2 CTC_UC 8.2 – Request V-ITS-SG firewall status

13.2.1 Overview

This is a conformance test for checking the UC 8.2 when requesting the status of the firewall (active/inactive) as specified in ISO 13185-1.

This service is not supported.

13.3 CTC_UC 8.3 – Configuration of V-ITS-SG firewall

13.3.1 Overview

This is a conformance test for checking the UC 8.3 when ConfigFirewall is used to send firewall configuration data from the ND to the V-ITS-SG to enable/disable specific services/features as specified in ISO 13185-1.

13.3.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from an `enablePassThruReq`, by the Client, the Client method `positiveConf` is executed with the `callSequenceNumber` of the request.

13.3.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle.

13.3.4 Preamble (setup state)

Setup conditions: None.

13.3.5 Test execution

Table 17 defines the test execution.

Table 17 — CTC_UC 8.3 – Configuration of V-ITS-SG firewall test execution

Step #	Description

1	`//Create an enable pass thru request`
	`String passThruSeed = ...;`
	`String passThruKey = ...;`
	`public UVIPPackage reqPackage =`
	`  uvipClient.enablePassThruReq(passThruSeed, passThruKey);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void positiveConf(int callChoice, int callSequenceNumber) {`
	`  if (callSequenceNumber == reqPackage.getCallSequenceNumber()) {`
	`  //display success of enablePassThu`
	`  }`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if enablePassThru succeeds.
5	Change passThruKey to null. Repeat steps 1 to 4 to disable the firewall.

13.3.6 Postamble (setup state)

Setup conditions: None.

14 CT cluster 9 – V-ITS-SG special features

14.1 CTC_UC 9.1 – General data exchange

14.1.1 Overview

This is a conformance test for checking the UC 9.1 when the Exchange function shall provide the facilities for general data exchange between ND and V-ITS-SG as specified in ISO 13185-1.

With the `VIDFConfiguration` it is possible to define different access types for parameters (read, read and write, ...) so there is nothing to do but to setup a configuration.

14.2 CTC_UC 9.2 – V-ITS-SG activation mode

14.2.1 Overview

This is a conformance test for checking the UC 9.2 when the ActivationMode is used to activate/deactivate the V-ITS-SG without a connection to the ND. If the ActivationMode in the V-ITS-SG is enabled, the V-ITS-SG shall perform all enabled functionality and store any event which occurs in the EventLogFile as specified in ISO 13185-1.

The V-ITS-SG captures all events in the EventLogFile if the ActivationMode parameter is enabled. See 14.3, CTC_UC 9.3 - Upload EventLogFile from V-ITS-SG.

14.3 CTC_UC 9.3 – Upload EventLogFile from V-ITS-SG

14.3.1 Overview

This is a conformance test for checking the UC 9.3 when UploadEventLogFile is used to read (upload) the `EventLogFile` from the V-ITS-SG with the ND as specified in ISO 13185-1.

14.3.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `manageFileDownloadReq`, by the Client, the Client method `manageFileConf` is executed with the name of the downloaded file.

14.3.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle.

14.3.4 Preamble (setup state)

Setup conditions: None.

14.3.5 Test execution

Table 18 defines the test execution.

Table 18 — CTC_UC 9.3 – Upload EventLogFile from V-ITS-SG test execution

Step #	Description

1	`//Create a manage file download request`
	`String sFilename = "Event2018-01-01.log"; //The name of the file to download`
	`File dirDownload = ...; //The directory where to save the file`
	`public UVIPPackage reqPackage =`
	`  uvipClient.manageFileDownloadReq(FileType.log, sFilename, dirDownload);`

Table 18 *(continued)*

Step #	Description

2	//Define Thread implementing the UVIP configuration interface,
	//handling the value confirmation of the defined request
	Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {
	...
	public void manageFileConf(String filename) {
	//display the downloaded file name
	}
	}
	ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);

3	//Send request message
	tcpClient.sendCall(reqPackage, threadUVIP);

4	Check if downloaded file name is displayed and the downloaded file exists.

14.3.6 Postamble (setup state)

Setup conditions: None.

15 CT cluster 10 – Vehicle diagnostics

15.1 CTC_UC 10.1 – Perform functional group OBD

15.1.1 Overview

This is a conformance test for checking the UC 10.1 when providing functional group-oriented vehicle diagnostics to the ND. The implementation of functional group-oriented vehicle diagnostics in the V-ITS-SG requires the collection of data items which are part of the, e.g. emissions-related system, safety-related system, as specified in ISO 13185-1.

15.1.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getValueReq`, by the Client, the Client method `getValueConf` is executed, providing the values of the requested data parameters.

15.1.3 Configuration

Prepare a `VIPFConfiguration` for the vehicle containing OBD relevant data, e.g. emissions-related system, safety-related system. Put these data into data parameter groups.

15.1.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids contain the OBD parameters.

15.1.5 Test execution

Table 19 defines the test execution.

Table 19 — CTC_UC 10.1 – Perform functional group OBD test execution

Step #	Description

1	//Create a value request with an array of vehicle information parameters
	Integer rvIdEmissionsRelatedSystem = ...;
	Integer rvIdSafetyRelatedSystem = ...;
	UVIPPackage reqPackage = uvipClient.getValueReq(0, new Integer[{
	rvIdEmissionsRelatedSystem, rvIdSafetyRelatedSystem }, null);

2	//Define Thread implementing the UVIP configuration interface,
	//handling the value confirmation of the defined request
	Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {
	...
	public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {
	//display the parameters with their values
	}
	}
	ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);

3	//Send request message
	tcpClient.sendCall(reqPackage, threadUVIP);

4	Check if the vehicle information parameters are displayed.

15.1.6 Postamble (setup state)

Setup conditions: None.

15.2 CTC_UC 10.2 – Perform enhanced OBD

15.2.1 Overview

This is a conformance test for checking the UC 10.2 when the V-ITS-SG shall provide the capability to support any type of enhanced diagnostics in order to enable the ND to utilize the diagnostic functionality built into the V-ITS-SG without knowing any detail of the vehicle manufacturer's specific data and test sequences as specified in ISO 13185-1.

15.2.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `getValueReq`, by the Client, the Client method `getValueConf` is executed, providing the values of the requested data parameters.

15.2.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle containing enhanced OBD relevant data. For every system, create a data parameter group.

15.2.4 Preamble (setup state)

Setup conditions: 6.3 CTC_UC 1.3 – Browsing supported data Ids contain the OBD parameters.

15.2.5 Test execution

Table 21 defines the test execution.

Table 20 — CTC_UC 10.2 – Perform enhanced OBD test execution

Step #	Description

1	`//Create a value request with a system`
	`Integer rvId = ...;`
	`UVIPPackage reqPackage = uvipClient.getValueReq(0, rvId, null);`

2	`//Define Thread implementing the UVIP configuration interface,`
	`//handling the value confirmation of the defined request`
	`Class ThreadUVIPConfirmation extends Thread implements IUVIPConfirmation {`
	`  ...`
	`  public void getValueConf(Vector<EcuDataParam> params) throws NumberedException {`
	`  //display the parameters with their values`
	`  }`
	`}`
	`ThreadUVIPConfirmation threadUVIP = new ThreadUVIPConfirmation (...);`

3	`//Send request message` `tcpClient.sendCall(reqPackage, threadUVIP);`
4	Check if the vehicle information parameters are displayed.

15.2.6 Postamble (setup state)

Setup conditions: None.

15.3 CTC_UC 10.3 – Upload VSOCLogFile from V-ITS-SG

15.3.1 Overview

This is a conformance test for checking the UC 10.3 when the `UploadVSOCLogFile` is used to read (upload) the VSOC (Vehicle State Of Capabilities) data from the V-ITS-SG. The VSOC data contain the detected DTCs, associated data and potential test results of V-ITS-SG supported vehicle system/function monitoring applications as specified in ISO 13185-1.

15.3.2 Test purpose

The purpose of the conformance test is to verify that after sending a `UVIPPackage`, created from a `manageFileDownloadReq`, by the Client, the Client method `manageFileConf` is executed with the name of the downloaded file.

15.3.3 Configuration

Prepare a `VIDFConfiguration` for the vehicle.

15.3.4 Preamble (setup state)

Setup conditions: None.

15.3.5 Test execution

Table 21 defines the test execution.