INTERNATIONAL STANDARD

ISO 20080

First edition
2019-03

Road vehicles — Information for remote diagnostic support — General requirements, definitions and use cases

Véhicules routiers — Information pour support de diagnostic à distance — Exigences générales, définitions et cas d'utilisation

Reference number
ISO 20080:2019(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 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

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

Remote diagnostic support is used for diagnosing vehicles at a distance prior to repair work, thereby minimizing the time spent in workshops, reducing disturbances on the road network, reducing inconvenience for the vehicle users and reducing cost for vehicle owners.

This document defines remote diagnostic support and the constraints that need to be respected. This document may also serve as a reference for other standards that relate to remote diagnostic support.

This document will facilitate exchange of information for remote diagnostic support between the different stakeholders (including vehicle manufacturers and independent operators) of the vehicle repair industry.

Road vehicles — Information for remote diagnostic support — General requirements, definitions and use cases

1 Scope

This document specifies general requirements and constraints applicable to a remote diagnostic process, the use cases and scenarios to support the implementation of a remote diagnostic process using a standardized interface of the ExVe.

It concerns:

— the road vehicles with four or more wheels designed and constructed primarily for the carriage of persons that are defined as Category 1 vehicles in the United Nations Special Resolution No.1 in TRANS/WP.29/1045, as last amended on 19 June 2012, and

— the road vehicle with four or more wheels designed and constructed primarily for the carriage of goods that are defined as Category 2 vehicles in the United Nations Special Resolution No.1 in TRANS/WP.29/1045, as last amended on 19 June 2012,

where these road vehicles are still in accordance with the specifications of the vehicle manufacturer.

This document does not define the interfaces provided by the ExVe nor the internal implementation inside the ExVe.

Processes like repair, prognostics, monitoring, configuration, re-programming and variant coding are not part of this document.

The prerequisites (e.g. authentication and authorization) for all use cases are not covered within this document. A possible specification of the required content for the implementation of a remote diagnostic application using the web interface of the ExVe according to ISO 20078 is given in Annex A.

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 20077-1:2017, *Road Vehicles — Extended vehicle (ExVe) methodology — Part 1: General information*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

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

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

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

3.1 Actors

3.1.1
local diagnostic facilitator
person at the vehicle that is in communication with the *remote diagnostician* (3.1.2) and facilitates the *diagnostics* (3.2) by that person's capacity to act as requested by the remote diagnostician and to answer the remote diagnostician's questions

Note 1 to entry: A local diagnostic facilitator is normally able to understand or describe the indications of the on-board instruments and tell-tales.

Note 2 to entry: The local diagnostics facilitator will normally have the necessary consent to operate the concerned device, system, or vehicle.

3.1.2
remote diagnostician
physical person, who may be assisted by technology, to perform a remote *diagnostic process* (3.2)

3.2
diagnostics
diagnostic process
process including the detection process of possible *malfunctions* (3.8), the identification of the likely root cause of these malfunctions and the appraisal of its relevance for the operation of the vehicle

[SOURCE: ISO 20077-1:2017]

3.3
diagnostics step 1
detection process of possible vehicle *malfunctions* (3.8)

Note 1 to entry: The detection process of possible vehicle malfunctions (diagnostics step 1) may lead to the conclusion of an absence of malfunction.

[SOURCE: ISO 20077-1:2017]

3.4
diagnostics step 2
identification of the likely root cause of *malfunctions* (3.8)

Note 1 to entry: Root cause analysis (diagnostics step 2) is only performed in presence of a malfunction. Root cause analysis is performed if diagnostics step 1 has been performed.

[SOURCE: ISO 20077-1:2017]

3.5
extended vehicle
entity, still in accordance with the specifications of the *vehicle manufacturer* (3.7), that extends beyond the physical boundaries of the road vehicle and consists of the road vehicle, off-board systems, external interfaces and the data communication between the road vehicle and the off-board systems

Note 1 to entry: Road vehicles without off-board systems and road vehicles equipped with telematics units are extended vehicles.

[SOURCE: ISO 20077-1:2017]

3.6
ExVe manufacturer
vehicle manufacturer (3.7) responsible for the extended vehicle

[SOURCE: ISO 20077-1:2017]

3.7
vehicle manufacturer
person or body who is responsible to the approval authority for all aspects of the type approval or authorization process and for ensuring conformity of production of a vehicle

Note 1 to entry: It is not essential that the person or body be directly involved in all stages of the construction of the vehicle, system, component or separate technical unit which is the subject of the approval process.

3.8
malfunction
state of a system or component that deviates from the specifications of the *vehicle manufacturer* (3.7)

Note 1 to entry: A malfunction may be the object of an alert on board the vehicle and possibly lead to a DTC, but a malfunction does not necessarily preclude a DTC.

Note 2 to entry: A slight deterioration of a system, such as the normal wear of that system, is not a malfunction as long as it does not impair the performance of that system against the design specifications of the vehicle manufacturer.

[SOURCE: ISO 20077-1:2017]

3.9
remote, adjective
performed on a vehicle from a distance where the operator responsible for the concerned operation is not co-located with the vehicle and where the vehicle is connected via an external network

EXAMPLE Remote diagnostics, remote access

Note 1 to entry: The operator responsible for the concerned operation is a specific actor in terms of use case.

[SOURCE: ISO 20077-1:2017]

3.10
remote diagnostic process
process including the detection of possible vehicle *malfunctions* (3.8), the identification of the likely root-cause of these malfunctions, and the assessment of its relevance for the operation of the vehicle, which is performed on a vehicle from a distance where the remote diagnostician responsible for the concerned operation is not co-located with the vehicle and where the vehicle is connected via an external network

3.11
remote diagnostic support
information provided to a *remote diagnostician* (3.1.2) to assist in the performance of the *remote diagnostic process* (3.10) of a vehicle

Note 1 to entry: Typical examples of remote diagnostics support are:

— information for performing a remote diagnostic process on a vehicle (for example instructions, training material, etc.),

— information specified for remote diagnostics use-cases, and

— information used by the after-sales remote diagnostics tool equipment systems.

Note 2 to entry: Diagnostic support that is provided for performing conventional diagnostics is the foundation for remote diagnostic support (the access to that information in the case of conventional diagnostics is standardised in ISO 18541-1).

[SOURCE: ISO 20077-1:2017]

3.12 Vehicle states

3.12.1
vehicle state
condition of the vehicle at a point in time with regard to motion and operation

EXAMPLE Stationary, moving, normally operating, non-operating vehicle.

3.12.2
moving vehicle
vehicle which is not stationary

3.12.3
stationary vehicle
vehicle which had a speed of 0 km/h for at least 1 minute

Note 1 to entry: The engagement of any measure such as a parking brake, a trailer-brake, or a hand-brake is not necessary for being stationary.

3.12.4
non-operating vehicle
stationary vehicle with propulsion system OFF and power-take-off systems off

Note 1 to entry: The engagement of any measure such as a parking brake, a trailer-brake, or a handbrake may be necessary for being non-operating vehicle.

3.12.5
normally operating vehicle
vehicle operating within its design criteria under normal duty operation

Note 1 to entry: A vehicle with a power take off engaged shall be considered as normally operating even if it is a stationary vehicle.

Note 2 to entry: A vehicle in a repair, maintenance, or inspection process shall not be considered as a normally operating vehicle.

3.13
prognostics
prognostic process
<automotive> process of forecasting the possible occurrence of vehicle *malfunctions* (3.8) and appraising the likely remaining operation time of the vehicle until these malfunctions occur

Note 1 to entry: A prognostic process cannot be performed without having performed the detection process for possible malfunctions of the same functionally related system [*diagnostics step 1* (3.3)].

Note 2 to entry: A *diagnostic process* (3.2) may be performed without performing a prognostic process for example, in the case of the presence of a malfunction, when *diagnostics step 2* (3.4) is performed.

[SOURCE: ISO 20077-1:2017]

3.14
snapshot data
specific data records associated with a DTC which are generated and stored at the time the DTC was detected

4 Symbols and abbreviated terms

API Application program interface

ECU Electronic control unit

ExVe Extended vehicle

DTC Diagnostic trouble code

ID Identifier

VIN Vehicle identification number

VM Vehicle manufacturer

5 Overview and general requirements for the remote diagnostic process and support

5.1 Remote diagnostic application and ExVe interaction

This document focuses on the interaction between a remote diagnostic client application and the ExVe as defined in ISO 20077-1 (see Figure 1).

For the use cases detailed, this document specify the communication established between the remote diagnostic application or server of the service provider and the ExVe ("machine-to-machine"). The output of the use-cases presented in this document is information that constitutes a major part of remote diagnostic support. This information may be used as well by the remote diagnostician when remotely diagnosing a vehicle or by the designer of a remote diagnostic application.

NOTE Such an application can be used at a later stage by the remote diagnostician.

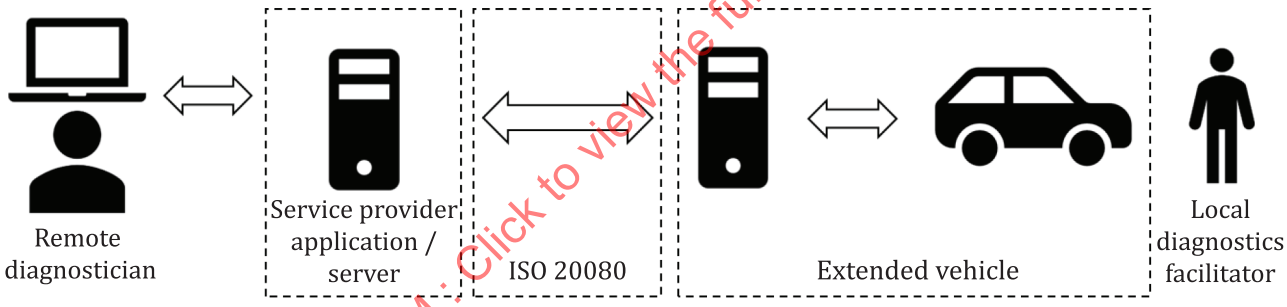

Figure 1 — Remote diagnostic client application and ExVe server to server communication

5.2 Content and use case scenarios

5.2.1 Capability to perform use cases

The capability to perform the different use cases can depend on the authorization, on the diagnosis tasks (e.g. remote activation of actuators), technical parameters (e.g. stability of connection, battery capability, vehicle status) and e.g. the brand and model of the vehicle.

5.2.2 Use case scenarios

There are two major scenarios: stationary vehicle and moving vehicle. In both situations, a remote diagnostician is responsible for the diagnostic procedure and a local diagnostic facilitator is in place and responsible for the vehicle.

5.3 Basic principles

Basic principle 1:

— All use cases are based on the condition that the diagnostician in charge for the remote diagnostic process is remote from the vehicle.

Basic principle 2:

— Some use cases require that a local diagnostic facilitator shall be present for necessary interactions within the remote diagnostic process. The use case description describes when this is required.

Basic principle 3:

— In the case of concurrent requests from service providers to the ExVe, the ExVe shall react to concurrent requests in a safe way. This reaction may for example be to respond that the requested functionality is currently not available, or to put the latter request in the queue.

Basic principle 4:

— The ExVe manufacturer is responsible for ensuring that the designed ExVe functionality respects that the correlation between the vehicle owner and the performed functions is not monitored for competition purposes.

Basic principle 5:

— The ExVe manufacturer is responsible for ensuring that the designed ExVe functionality respects that the correlation between the after-sales service provider and the performed functions is not monitored for competition purposes.

5.4 Access to information for remote diagnostic support

There are several alternatives for the vehicle manufacturer to provide remote diagnostic support:

— According to Annex A

  — The vehicle manufacturer shall supply supplementary vehicle manufacturer specific information.

— Vehicle manufacturer specific solution

  — Vehicle manufacturer shall supply specifications describing the solution to the service provider.

— Other standardised solution

Constraints such as vehicle state, compatibility with manufacturer specific functions, accessibility of data, freight and transport security and safety, etc. shall be considered.

5.5 Error conditions and handling

5.5.1 General

This clause describes the different error conditions that the ExVe interface may return to the service provider during a remote diagnostic process. Even if the service provider has access to a remote diagnostic functionality an error might be returned when the function is accessed. The error might be due to different reasons, e.g. wrong vehicle state, no connectivity or vehicle specification not supporting the requested remote diagnostic function.

Internal errors inside the ExVe itself (i.e.: errors between the ExVe backend server and the physical vehicle) are not relevant here.

5.5.2 Communication interface specific errors

— ExVe interface not responding

  — No communication between the remote diagnostic support application and the ExVe.

— Vehicle identifier not recognised

  — Either the vehicle identifier has been entered incorrectly or the system does not recognize the entered vehicle identifier as a vehicle with remote diagnostic capability.

— Request currently not possible to perform by the ExVe.

5.5.3 General errors for use cases

— Incompatible vehicle state.

5.5.4 Specific errors for use cases

— These are handled in the individual use cases:

  — Actuator identification not valid;

  — Actuation parameters not valid;

  — ECU ID not valid;

  — DTC status not valid;

  — DTC ID not valid;

  — Number of parameters exceeded;

  — Parameter ID not valid;

  — Routine ID not valid;

  — Parameters not valid;

  — Setting ID not valid;

  — Value not valid.

5.6 Conditions for conformance with this document

Compliance with this document means that the following conditions are met:

— The vehicle manufacturer implements the use case “use case discovery” specified in 6.2, in a web accessible manner, and the service provider will apply the “use case discovery”.

— The vehicle manufacturer, when acting as a service provider, and the service provider implement the use cases they support according to the relevant clauses as specified in this document.

6 Use cases for remote diagnostic support

6.1 General

A precondition is that the service provider has access to VM specific information about identifiers (including vehicle identifier), parameters, units, conversions and the definition of the DTC statuses (e.g. ACTIVE, PENDING and PREVIOUSLY_ACTIVE) used in the use cases.

For all supported use-cases, the ExVe shall provide an answer.

The actor in each use case is the remote diagnostic application.

The timestamp included in the use cases describes the point in time the data was received by the ExVe back-end server from the vehicle.

The vehicle identifier can be the VIN or similar unique identification of the vehicle. If VIN is not used, this unique vehicle identifier shall also be used consistently by the vehicle manufacturer for remote diagnostic support.

6.2 Use Case 01 — Use case discovery

See Table 1 for Use Case 01 - Use case discovery.

Table 1 — Use Case 01 — Use case discovery

Goal	Provide information about which remote diagnostic functionality that is available for the current vehicle through the ExVe interface.
Input	Command for supported diagnostic functionality. Interface input: — Vehicle identifier
Description	The command is received by the ExVe interface. If the discovery command is executed, the ExVe checks that this command can be carried out with respect to applicable constraints. The ExVe compiles the accessible diagnostic functionality in a format depending on the implementation.
Output	The supported diagnostic use cases, with respect to applicable constraints, are returned. Interface output: — Accessible use cases (remote diagnostic functions)
Example	Interface input: — Vehicle identifier: 12345678909876543 Interface output: — Use case 02, — Use case 03, — Use case 05, — Use case 06, — Use case 07
Error conditions	No use case specific errors

6.3 Use Case 02 — Identify ECUs installed in the vehicle

See Table 2 for Use Case 02 - Identify ECUs installed in the vehicle.

Table 2 — Use Case 02 — Identify ECUs installed in the vehicle

Goal	Provide all VM ECU IDs and corresponding vehicle manufacturer software and hardware part numbers which are installed in the chosen vehicle through the ExVe interface.
Input	Command to identify VM ECU ID, software and hardware for all ECU fitted to the vehicle. Interface input: — Vehicle identifier
Description	The command is received by the ExVe interface. If the command is executed in the vehicle, the ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe reads the current VM ECU ID, software and hardware for all ECUs fitted to the vehicle.
Output	The current VM ECU ID, software and hardware for all ECUs on the vehicle is returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — List of ECUs — For each ECU — ECU ID — HW part number(s) — SW part number(s)
Example	Interface input: — Vehicle identifier: 12345678909876543 Interface output: — Timestamp: 2016-02-24 09:23, — ECU ID: ABC, — HW part number: 1234567, — SW part number: 9876543; — ECU ID: DEF, — HW part number: 2345678, — SW part number: 8976543; — ECU ID: GHI, — HW part number: 3456789, — SW part number: 7896543, SW part number: 7896555
Error conditions	No use case specific errors

6.4 Use Case 03 — Read Diagnostic Trouble Codes (DTCs)

See Table 3 for Use Case 03 - Read Diagnostic Trouble Codes (DTCs).

Table 3 — Use Case 03 — Read Diagnostic Trouble Codes (DTCs)

Goal	Provide diagnostic trouble codes, depending on status through the ExVe interface.
Input	Command to read DTCs Interface input: — Vehicle identifier — ECU ID (if empty: read from all ECUs) — Status
Description	The command is received by the ExVe interface. If the command is executed in the vehicle, the ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe reads all requested DTCs.
Output	All requested DTCs are returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — List of ECUs — For each ECU — A list of DTCs — For each DTC — DTC ID — Status — Occurrence counter(s) (optional) — DTC Timestamp(s) (optional) — Additional parameter(s) (optional)
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC, — Status: ACTIVE

Table 3 *(continued)*

	Interface output: — Timestamp: 2016-02-24 09:23, — ECU ID: ABC, — DTC ID: 123456, — Status: ACTIVE, — Occurrence counter: 3; — DTC timestamp: 2016-01-20 08:23, — DTC ID: 345678, — Status: ACTIVE, — Occurrence counter: 1, — DTC Timestamp: 2016-01-07 14:56
Error conditions	ECU ID not valid DTC status not valid

6.5 Use Case 04 — Read readiness codes

See Table 4 for Use Case 04 - Read readiness codes.

Table 4 — Use Case 04 — Read readiness codes

Goal	Provide readiness codes through the ExVe interface.
Input	Command to read readiness codes Interface input: — Vehicle identifier
Description	The command is received by the ExVe interface. If the command is executed in the vehicle, the ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe reads the readiness codes of the vehicle.
Output	The readiness codes are returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — List of systems — For each system — system ID — ready/not ready
Example	Interface input: — Vehicle identifier: 12345678909876543 Interface output: — Timestamp: 2016-02-24 09:23, — System ID: EGR, — Readiness: ready; — System ID: SCR, — Readiness: not ready
Error conditions	No use case specific errors

6.6 Use Case 05 — Read DTC snapshot data

See Table 5 for Use Case 05 - Read RTC snapshot data.

Table 5 — Use Case 05 — Read RTC snapshot data

Goal	Provide snapshot data through the ExVe interface.
Input	Command to snapshot data Interface input: — Vehicle identifier — ECU ID — DTC ID
Description	The command is received by the ExVe interface. If the command is executed in the vehicle, the ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe reads the snapshot data of the vehicle.
Output	The snapshot data are returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — List of parameters — For each parameter — Parameter ID — Value
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC, — DTC ID: 123456 Interface output: — Timestamp: 2016-02-24 09:23, — Parameter ID: 1234, — Value: 83; — Parameter ID: 5678, — Value: 2; — Parameter ID: 9012, — Value: 27
Error conditions	ECU ID not valid DTC ID not valid

6.7 Use Case 06 — Read selected diagnostic parametric dynamic data

See Table 6 for Use Case 06 - Read selected diagnostic parametric dynamic data.

Table 6 — Use Case 06 — Read selected diagnostic parametric dynamic data

Goal	Provide selected diagnostic parametric dynamic data through the ExVe interface .
Input	Command to read selected diagnostic parametric dynamic data. Interface input: — Vehicle identifier — ECU ID — List of Parameter IDs — Parameter ID
Description	The command is received by the ExVe interface. If the command is executed in the vehicle, the ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe reads the selected diagnostic parametric dynamic data of the vehicle.
Output	The selected diagnostic parametric dynamic data are returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — List of parameters — For each parameter — Parameter ID — Value
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC, — Parameter ID: 1234, 5678 Interface output: — Timestamp: 2016-02-24 09:23, — Parameter ID: 1234, — Value: 648; — Parameter ID: 5678, — Value: 1000
Error conditions	ECU ID not valid Number of parameters exceeded Parameter ID not valid

6.8 Use Case 07 — Read malfunction indicator status

See Table 7 for Use Case 07 - Read malfunction indicator status

Table 7 — Use Case 07 — Read malfunction indicator status

Goal	Provide malfunction indicator status through the ExVe interface.
Input	Command to read the malfunction indicator status. Interface input: — Vehicle identifier
Description	The command is received by the ExVe interface. If the command is executed in the vehicle, the ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe reads the malfunction indicator status of the vehicle.
Output	The malfunction indicator status is returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — List of indicator IDs — For each indicator ID — Indicator ID — Status Interface input: — Vehicle identifier: 12345678909876543,
Example	Interface output: — Timestamp: 2016-02-24 09:23, — Indicator ID: A, — Status: active, — Indicator ID: B, — Status: yellow
Error conditions	No use case specific errors

6.9 Use Case 08 — Clear DTCs

See Table 8 for Use Case 08 - Clear DTCs.

Table 8 — Use Case 08 — Clear DTCs

Goal	Clear all DTCs.
Input	Command to clear DTCs. Interface input: — Vehicle identifier — ECU ID (if empty: clear all DTCs in all ECUs)
Description	The command is received by the ExVe interface. The ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. The ExVe clears either all DTCs in defined ECU or all DTCs in all ECUs.
Output	All DTCs are cleared and a confirmation is returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — Confirmation
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC Interface output: — Timestamp: 2016-02-24 09:23, — Confirmation: ok
Error conditions	ECU ID not valid

6.10 Use Case 09 — Adjust the setting of a selected system

See Table 9 for Use Case 09 — Adjust the setting of a selected system.

Table 9 — Use Case 09 — Adjust the setting of a selected system

Goal	Provide temporarily adjustment of a setting of selected systems for diagnostic purposes through the ExVe interface.
Input	Command to adjust the setting of a selected system. Interface input: — Vehicle identifier — ECU ID (optional) — Setting ID — Value
Description	The command is received by the ExVe interface. The ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. In some cases, the local diagnostic facilitator has to confirm and give feedback or control the command. For use cases that have safety implications the local diagnostic facilitator has to confirm to the remote diagnostician that the boundary conditions allow the operation. The ExVe adjusts the settings of the selected system in the vehicle.
Output	The selected system setting is adjusted and a confirmation is returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — Confirmation
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC, — Setting ID: 57812, — Value: 51 Interface output: — Timestamp: 2016-02-24 09:23, — Confirmation: ok
Error conditions	Setting ID not valid Value not valid ECU ID not valid (if used)

6.11 Use Case 10 — Activation of actuator(s)

See Table 10 for Use Case 10 — Activation of actuator(s).

Table 10 — Use Case 10 — Activation of actuator(s)

Goal	Provide control of specific actuator(s) for diagnostic purposes through the ExVe interface.
Input	Command to control a specific or specific set of actuators. Interface input: — Vehicle identifier — ECU ID (optional) — Actuator identification — Actuation parameters (actuator dependent) (optional)
Description	The command is received by the ExVe interface. The ExVe checks that this actuation can be carried out with respect to applicable constraints, such as vehicle state. In some cases the local diagnostic facilitator has to confirm and give feedback or control the command. For use cases that have safety implications the local diagnostic facilitator has to confirm to the remote diagnostician that the boundary conditions allow the actuation. The ExVe activates the specified actuator in the vehicle.
Output	Specific or specific set of actuators activated according to the command or the command is denied e.g. due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — Result
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC — Actuator ID: 1234 (e.g Open windows), — Value: 70 % Interface output: — Timestamp: 2016-02-24 09:23, — Result: 70 %
Error conditions	Actuator identification not valid Actuation parameters not valid ECU ID not valid (if used)

6.12 Use Case 11 — Activate a self-test routine

See Table 11 for Use Case 11 — Activate a self-test routine.

Table 11 — Use Case 11 — Activate a self-test routine

Goal	Provide control of a self-test routine for diagnostic purposes through the ExVe interface.
Input	Command to control a self-test routine. Interface input: — Vehicle identifier — ECU ID (optional) — Routine ID — Parameters (optional)
Description	The command is received by the ExVe interface. The ExVe checks that this command can be carried out with respect to applicable constraints, such as vehicle state. In some cases, the local diagnostic facilitator has to confirm and give feedback or control the command. For use cases that have safety implications the local diagnostic facilitator has to confirm to the remote diagnostician that the boundary conditions allow the operation. The ExVe activates a self-test routine in the vehicle.
Output	The self-test routine is controlled and a result is returned according to the command or the command is denied due to safety or security restrictions applied to the vehicle state. Interface output: — Timestamp (received at ExVe back-end server) — Result
Example	Interface input: — Vehicle identifier: 12345678909876543, — ECU ID: ABC Routine ID: 1234, — Parameters: 70;54;30 Interface output: — Timestamp:2016-02-24 09:23, — Result: 4073
Error conditions	Routine ID not valid Parameters not valid ECU ID not valid (if used)

Annex A
(informative)

Implementation based on ISO 20078 — Road Vehicles — Extended Vehicle (ExVe) — Web Services

A.1 Introduction

A.1.1 General

This annex contains a web service specification of the use cases listed in this document. The web service specification is based on the ISO 20078 (all parts)[1)] [1]. All web services are defined as REST APIs, using JSON for the transfer of content.

ISO 20078 (all parts) [1] is used to implement web services according to Annex A.

A.1.2 Security

All REST APIs are using OAuth2 compatible framework for access control, OpenID Connect compatible framework for identification purposes and https for securing the transfer, see ISO 20078-3[2)] for details. The exact details of how to obtain access is described by each offering party.

A.1.3 Error codes

A.1.3.1 ISO 20078

The HTTP status codes (error codes) listed for each REST API are described in ISO 20078-2[3)][1].

A.1.3.2 General error conditions

General error conditions are valid for all use cases. Table A.1 lists the mapping of error conditions in 5.5 to REST API errors.

1) Under preparation.

2) Under preparation.

3) Under preparation.

Table A.1 — General error conditions

Error condition	HTTP status code	ExveErrorId	Example

Request currently not possible to perform by the ExVe	503	20080-1000	<pre>{
			"dtcReadout": {
			"id": "abcde-12345-ghjke-67474",
			"messageTimestamp": "2016-02-24T09:23:46Z",
			"exveErrorId": "20080-1000",
			"exveErrorMsg": "Request currently not possible to perform by the ExVe",
			"vehicleId": "12345678909876543"
			}
			}</pre>

A.1.3.3 Use case specific error conditions

Use case specific errors are mapped to HTTP status codes in each REST API.

A.2 Resources

A web service is exposing access to one or more resources. To be able to access a resource through a web service, access needs to be granted. This can be done either directly to the resource or through a container.

Table A.2 maps the ISO 20080 use cases to REST APIs and resources. In some cases, mapping of a use case to a REST API provides little standardization benefit, as it is highly offering party specific.

Table A.2 — Mapping of use cases to REST APIs

UC	Use case name	REST API	Resource(s)	Comment
01	Use case discovery	resourceReadouts	Not applicable	
02	Identify ECUs installed in the vehicle	ecuReadouts	ECU Readout	
03	Read Diagnostic Trouble Codes (DTCs)	dtcReadouts	DTC Readout	
04	Read readiness codes	readinessCodeReadouts	Readiness Code Readout	
05	Read DTC snapshot data	dtcSnapshotReadouts	DTC Snapshot Readout	
06	Read selected diagnostic parametric dynamic data	parameterReadouts	Parameter Readout	
07	Read malfunction indicator status	malfunctionIndicatorReadouts	Malfunction Indicator Readout	
08	Clear DTCs	clearDtcJobs	Clear DTC Job	
09	Adjust the settings of a selected system	Not applicable / No standardized API due to differences between offering parties.	Not applicable	System setting input and result are offering party specific.
10	Activation of actuators	Not applicable / No standardized API due to differences between offering parties.	Not applicable	Actuator input and result are offering party specific.
11	Activate a self-test routine	Not applicable / No standardized API due to differences between offering parties.	Not applicable	Self-test input and result are offering party specific.

A.3 REST API information model

The rest API information model in Figure A.1 is compiled from the use cases in this document and used as a base for designing the REST APIs.

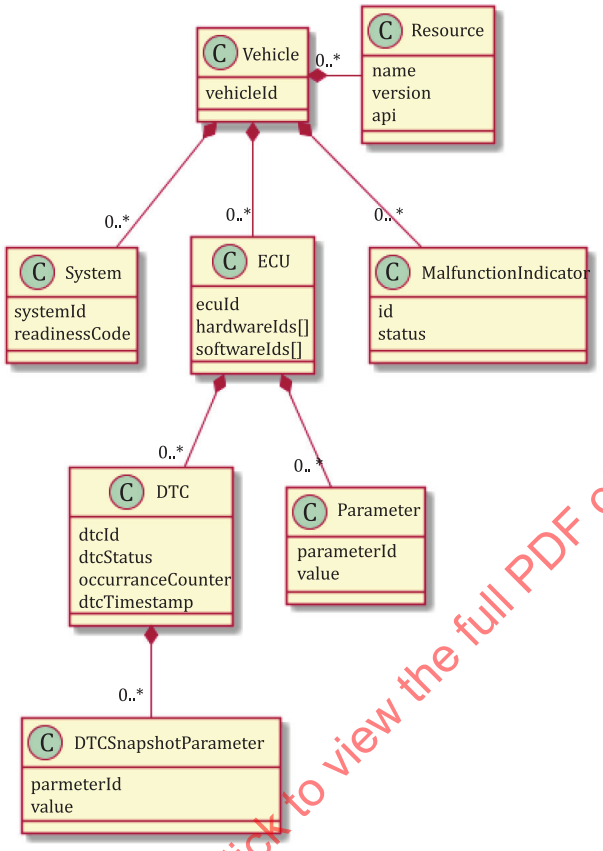

Figure A.1 — REST API information model

A.4 REST APIs

A.4.1 Use case 01 – resourceReadouts

The resourceReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.2). In some cases, it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a resource readout request (see Table A.3). If the result is available immediately, the result is returned directly (see Table A.4). If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed. When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

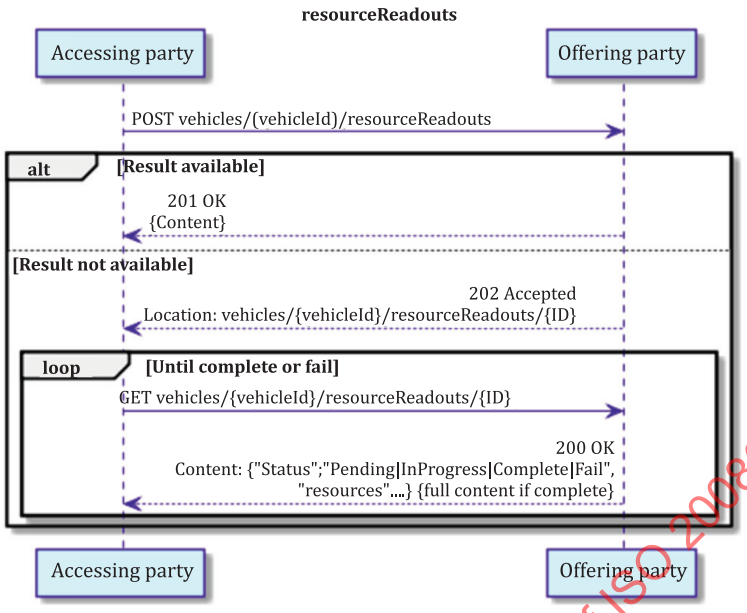

Figure A.2 — resourceReadouts sequence diagram

Table A.3 — POST resourceReadout

<table>
<tr><td colspan="2">POST vehicles/{vehicleId}/resourceReadouts</td></tr>
<tr><td>Description</td><td>This API creates a readout of available resources to the accessing party for one vehicle. If the result is available immediately, the result is returned. If the result is not available, a location to the resource readout is returned. This location shall be polled until the result is available.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.resourcereadout.v1+json;charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Location</td><td>Absolute URI of the /resourceReadouts endpoint</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>201</td><td>Example (result available immediately):
<pre>
{
"resourceReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "resources": [
    {
     "name": "DTC Readout",
     "version": "1",
     "api":
"https://example.org/vehicles/12345678909876543/dtcReadouts"
    },
    {
     "name": "ECU Readout",
     "version": "1",
     "api":
"https://example.org/vehicles/12345678909876543/ecuReadouts"
    }
    ]
  }
}
</pre>
</td></tr>
<tr><td>202</td><td>Result not available immediately. Location of resourceReadout will be returned, e.g. vehicles/{vehicleId}/resourceReadouts/{id}, see Location header.</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>Any resource</td><td>Access to any resource will give access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

Table A.4 — GET resourceReadout

GET vehicles/{vehicleId}/resourceReadouts/{id}

Description

This API returns a readout of available resources to the accessing party for one vehicle. The id of the readout is returned when posting the request. If the resource readout is not completed, the readout status is returned. If the resource readout is completed, the result is returned.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
id	string	required	Id of the DTC readout

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org.resourcereadout.v1+json;charset=utf-8

Response Headers

Content-Type	application/x.exve.org. resourcereadout.v1+json;charset=utf-8

Response (Success)

200

Example (result available):

```
{
"resourceReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "resources": [
    {
     "name": "DTC Readout",
     "version": "1",
     "api":
"https://example.org/vehicles/12345678909876543/dtcReadouts"
    },
    {
     "name": "ECU Readout",
     "version": "1",
     "api":
"https://example.org/vehicles/12345678909876543/ecuReadouts"
    }
    ]
  }
}
```

Example (result not available):

```
{
"resourceReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "InProgress",
  "asyncWait": 10000,
  "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543"
  }
}
```

Error codes

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
406	Not Acceptable
500	Internal Server Error
501	Not Implemented
503	Service Unavailable
505	Version not supported

Access

Any resource	Access to any resource will give access to this API

JSON Schema

See A.5

A.4.2 Use case 02 – ecuReadouts

The ecuReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.3). In some cases, it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting an ECU readout request (see Table A.5). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A.6). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

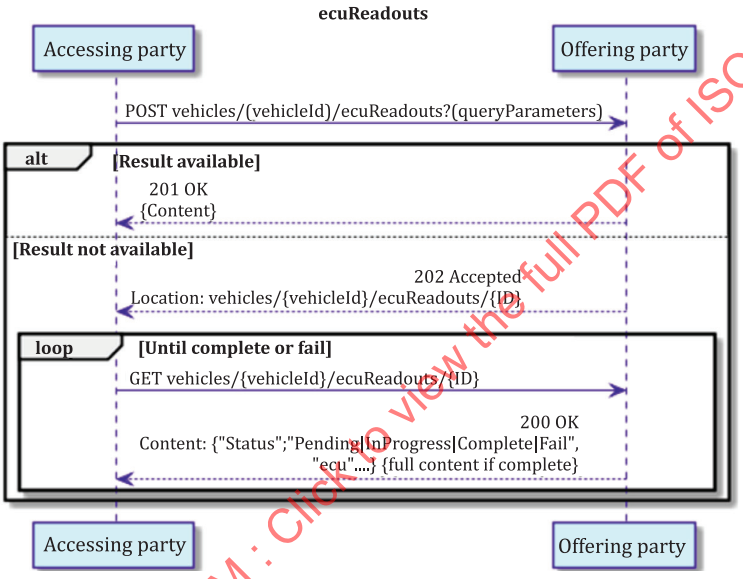

Figure A.3 — ecuReadouts sequence diagram

Table A.5 — POST ecuReadout

<table>
<tr><td colspan="2">POST vehicles/{vehicleId}/ecuReadouts?ecuId={ecuId}</td></tr>
<tr><td>Description</td><td>This API creates a readout of ECUs for one vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the ECU readout is returned. This location shall be polled until the result is available.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
<tr><td>ecuId</td><td>string</td><td>optional</td><td>Return this ECU id only.
Default: Return all ECUs.</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.ecureadout.v1+json; charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Location</td><td>Absolute URI of the /ecuReadouts endpoint</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>201</td><td>Example (result available immediately):
<pre>{
"ecuReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "ecus": [
      {
        "ecuId": "ABC",
        "hardwareIds": ["1234567"],
        "softwareIds": ["9876543"]
      }, {
        "ecuId": "DEF",
        "hardwareIds": ["2345678"],
        "softwareIds": ["8976543"]
      }, {
        "ecuId": "GHI",
        "hardwareIds": ["3456789"],
        "softwareIds": ["7896543","7896555"]
      }
    ]
  }
}</pre></td></tr>
<tr><td>202</td><td>Result not available immediately. Location of ecuReadout will be returned, e.g. vehicles/{vehicleId}/ecuReadouts/{id}, see Location header.</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>ECU Readout</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

Table A.6 — GET ecuReadout

GET vehicles/{vehicleId}/ecuReadouts/{id}	
Description	This API returns the ECU readout for one vehicle. The id of the readout is returned when posting the request. If the ECU readout is not completed, the readout status is returned. If the ECU readout is completed, the result is returned.
Parameters	vehicleId | string | required | The vehicle identifier of the vehicle to read from Id | string | required | Id of the ECU readout
Request Headers	Host | required | According to HTTP/1.1 RFC 2616 Authorization | required | Bearer {token} Accept | required | application/x.exve.org.ecureadout.v1+json; charset=utf-8
Response Headers	Content-Type | application/x.exve.org.ecureadout.v1+json;charset=utf-8

Response (Success)	200 | Example (result available):
	{
	"ecuReadout": {
	"id": "abcde-12345-ghjke-67474",
	"asyncStatus": "Complete",
	"messageTimestamp": "2016-02-24T09:23:46Z",
	"vehicleId": "12345678909876543",
	"receivedTimestamp": "2016-02-24T09:23:46Z",
	"ecus": [
	{
	"ecuId": "ABC",
	"hardwareIds": ["1234567"],
	"softwareIds": ["9876543"]
	}, {
	"ecuId": "DEF",
	"hardwareIds": ["2345678"],
	"softwareIds": ["8976543"]
	}, {
	"ecuId": "GHI",
	"hardwareIds": ["3456789"],
	"softwareIds": ["7896543","7896555"]
	}
	]
	}
	}
	Example (result not available):
	{
	"ecuReadout": {
	"id": "abcde-12345-ghjke-67474",
	"asyncStatus": "InProgress",
	"asyncWait": 10000,
	"asyncEstimatedComplete": "2016-02-24T09:24:00Z",
	"messageTimestamp": "2016-02-24T09:23:46Z",
	"vehicleId": "12345678909876543"
	}
	}

Error codes	400 | Bad Request 401 | Unauthorized 403 | Forbidden 404 | Not Found 406 | Not Acceptable 500 | Internal Server Error 501 | Not Implemented 503 | Service Unavailable 505 | Version not supported
Access	ECU Readout | Full access to this API
JSON Schema	See A.5

A.4.3 Use case 03 – dtcReadouts

The dtcReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.4). In some cases it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a DTC readout request (see Table A.7). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A.8). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

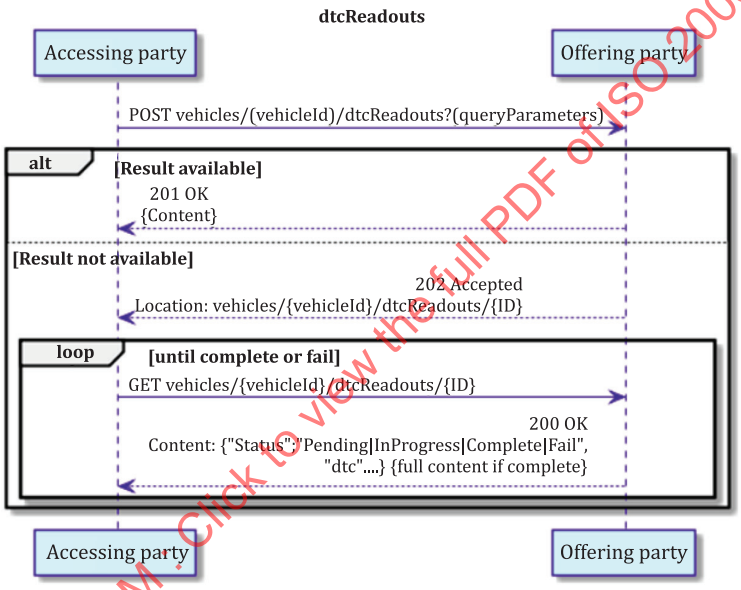

Figure A.4 — dtcReadouts sequence diagram

Table A.7 — POST dtcReadout

POST vehicles/{vehicleId}/dtcReadouts?ecuId={ecuId}&dtcStatus={dtcStatus}

Description

This API creates a readout of DTCs for one vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the DTC readout is returned. This location shall be polled until the result is available.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
ecuId	string	optional	Return DTCs from this ECU id only. Default: Return DTCs from all ECUs.
dtcStatus	string	required	Returns DTCs with this status only.

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org.dtcreadout.v1+json; charset=utf-8

Response Headers

Location	Absolute URI of the /dtcReadouts endpoint

Response (Success)

201 Example (result available immediately):

```
{
"dtcReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "dtcs": [
      {
       "dtcId": "123456",
       "status": "ACTIVE",
       "occurrenceCounter": 3,
       "ecuId": "ABC",
       "dtcTimestamp": "2016-01-20T08:23:46Z"
      },
      {
       "dtcId": "345678",
       "status": "ACTIVE",
       "occurrenceCounter": 1,
       "ecuId": "ABC",
       "dtcTimestamp": "2016-01-07T14:56:10Z"
      }
  ]
}
}
```

202 Result not available immediately. Location of dtcReadout will be returned, e.g. vehicles/{vehicleId}/dtcReadouts/{id}, see Location header.

Error codes

400	Bad Request	DTC status not valid
401	Unauthorized	
403	Forbidden	
404	Not Found	ECU ID not valid
406	Not Acceptable	
500	Internal Server Error	
501	Not Implemented	
503	Service Unavailable	
505	Version not supported	

Access

DTC Readout	Full access to this API

JSON Schema

See A.5

Table A.8 — GET dtcReadout

<table>
<tr><td colspan="2">GET vehicles/{vehicleId}/dtcReadouts/{id}</td></tr>
<tr><td>Description</td><td>This API returns the DTC readout for one vehicle. The id of the readout is returned when posting the request. If the DTC readout is not completed, the readout status is returned. If the DTC readout is completed, the result is returned.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
<tr><td>id</td><td>string</td><td>required</td><td>Id of the DTC readout</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.dtcreadout.v1+json; charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Content-Type</td><td>application/x.exve.org.dtcreadout.v1+json;charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>200</td><td>Example (result available):
<pre>{
"dtcReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "dtcs": [
      {
       "dtcId": "123456",
       "status": "ACTIVE",
       "occurrenceCounter": 3,
       "ecuId": "ABC",
       "dtcTimestamp": "2016-01-20T08:23:46Z"
      },
      {
       "dtcId": "345678",
       "status": "ACTIVE",
       "occurrenceCounter": 1,
       "ecuId": "ABC",
       "dtcTimestamp": "2016-01-07T14:56:10Z"
      }
  ]
 }
}</pre>
Example (result not available):
<pre>{
"dtcReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "InProgress",
  "asyncWait": 10000,
  "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543"
 }
}</pre>
</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>DTC Readout</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

A.4.4 Use case 04 – readinessCodeReadouts

The readinessCodeReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.5). In some cases it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a readiness code readout request (see Table A.9). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A 10). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

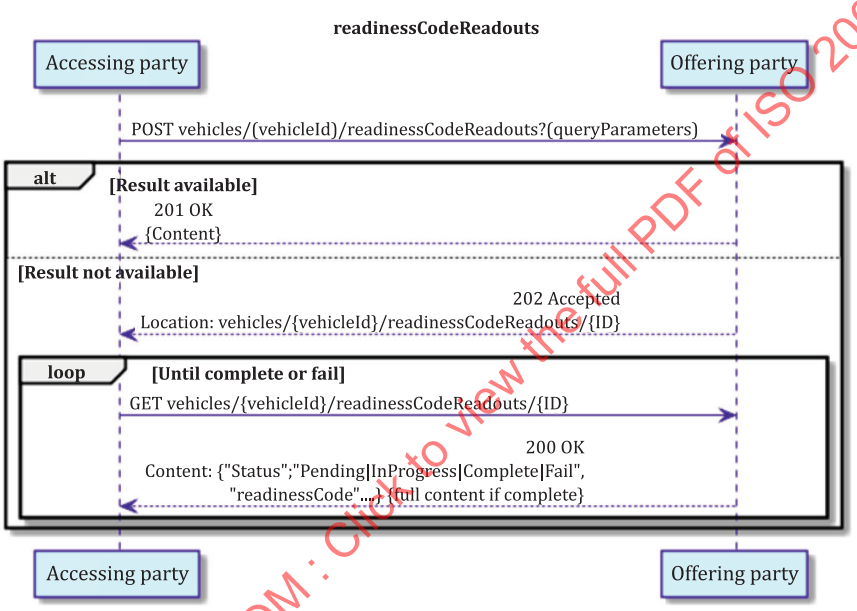

Figure A.5 — readinessCodeReadouts sequence diagram

Table A.9 — POST readinessCodeReadout

<table>
<tr><td colspan="2">POST vehicles/{vehicleId}/readinessCodeReadouts</td></tr>
<tr><td>Description</td><td>This API creates a readout of readiness codes for one vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the readiness code readout is returned. This location shall be polled until the result is available.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.readinesscodereadout.v1+json; charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Location</td><td>Absolute URI of the /readinessCodeReadouts endpoint</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>201</td><td>Example (result available immediately):
<pre>
{
"readinessCodeReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "readinessCodes": [
      {
       "systemId": "EGR",
       "isReady": true
      }, {
       "systemId": "SCR",
       "isReady": false
      }
  ]
 }
}
</pre>
</td></tr>
<tr><td>202</td><td>Result not available immediately. Location of readinessCodeReadout will be returned, e.g. vehicles/{vehicleId}/readinessCodeReadouts/{id}, see Location header.</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>Readiness Code Readout</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

Table A.10 — GET readinessCodeReadout

<table>
<tr><td colspan="2">GET vehicles/{vehicleId}/readinessCodeReadouts/{id}</td></tr>
<tr><td>Description</td><td>This API returns the readiness code readout for one vehicle. The id of the readout is returned when posting the request. If the readiness code readout is not completed, the readout status is returned. If the readiness code readout is completed, the result is returned.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
<tr><td>Id</td><td>string</td><td>required</td><td>Id of the readiness code readout</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.readinesscodereadout.v1+json; charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Content-Type</td><td>application/x.exve.org.readinesscodereadout.v1+json;charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>200</td><td>Example (result available):
<pre>{
"readinessCodeReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "readinessCodes": [
      {
       "systemId": "EGR",
       "isReady": true
      }, {
       "systemId": "SCR",
       "isReady": false
      }
  ]
 }
}</pre>
Example (result not available):
<pre>{
"readinessCodeReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "InProgress",
  "asyncWait": 10000,
  "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543"
 }
}</pre>
</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>Readiness Code Readout</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

A.4.5 Use case 05 – dtcSnapshotReadouts

The dtcSnapshotReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.6). In some cases it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a DTC snapshot readout request (see Table A.11). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A.12). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

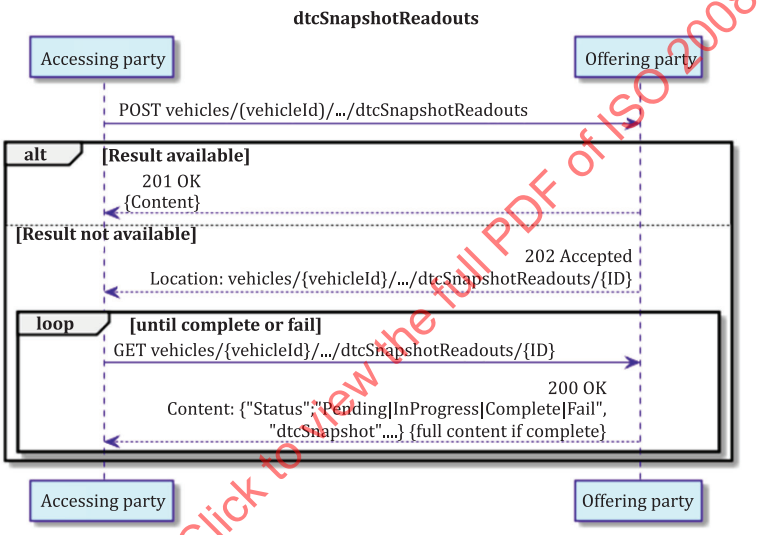

Figure A.6 — dtcSnapshotReadouts sequence diagram

Table A.11 — POST dtcSnapshotReadout

POST vehicles/{vehicleId}/ecuId/{ecuId}/dtcId/{dtcId}/dtcSnapshotReadouts

Description

This API creates a readout of a DTC snapshot from one vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the DTC snapshot readout is returned. This location shall be polled until the result is available.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
ecuId	string	required	The id of the ECU to read from
dtcId	string	required	The id of the DTC associated with the snapshot

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org.dtcsnapshotreadout.v1+json;charset=utf-8

Response Headers

Location	Absolute URI of the /dtcSnapshotReadouts endpoint

Response (Success)

201 — Example (result available immediately):

```
{
"dtcSnapshotReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "ecuId": "ABC",
  "dtcId": "123456",
  "dtcSnapshotParameters": [
       {
        "name": "1234",
        "value": "83"
       },
       {
        "name": "5678",
        "value": "2"
       },
       {
        "name": "9012",
        "value": "27"
       }
    ]
  }
}
```

202 — Result not available immediately. Location of dtcSnapshotReadout will be returned, e.g. vehicles/{vehicleId}/ecuId/{ecuId}/dtcId/{dtcId}/dtcSnapshotReadouts/{id}, see Location header.

Error codes

400	Bad Request	
401	Unauthorized	
403	Forbidden	
404	Not Found	ECU ID not valid DTC ID not valid
406	Not Acceptable	
500	Internal Server Error	
501	Not Implemented	
503	Service Unavailable	
505	Version not supported	

Access

DTC Snapshot Readout	Full access to this API

JSON Schema

See A.5

Table A.12 — GET dtcSnapshotReadout

GET vehicles/{vehicleId}/ecuId/{ecuId}/dtcId/{dtcId}/dtcSnapshotReadouts/{id}

Description

This API returns the DTC snapshot from one vehicle. The id of the readout is returned when posting the request. If the DTC snapshot readout is not completed, the readout status is returned. If the DTC snapshot readout is completed, the result is returned.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
ecuId	string	required	The id of the ECU to read from
dtcId	string	required	The id of the DTC associated with the snapshot
id	string	required	Id of the DTC snapshot readout

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org.dtcsnapshotreadout.v1+json;charset=utf-8

Response Headers

Content-Type	application/x.exve.org.dtcsnapshotreadout.v1+json;charset=utf-8

Response (Success)

200 Example (result available):

```
{
"dtcSnapshotReadout": {
   "id": "abcde-12345-ghjke-67474",
   "asyncStatus": "Complete",
   "messageTimestamp": "2016-02-24T09:23:46Z",
   "vehicleId": "12345678909876543",
   "receivedTimestamp": "2016-02-24T09:23:46Z",
   "ecuId": "ABC",
   "dtcId": "123456",
   "dtcSnapshotParameters": [
         {
          "name": "1234",
          "value": "83"
         },
         {
          "name": "5678",
          "value": "2"
         },
         {
          "name": "9012",
          "value": "27"
         }
      ]
   }
}
```

Example (result not available):

```
{
"dtcSnapshotReadout": {
   "id": "abcde-12345-ghjke-67474",
   "asyncStatus": "InProgress",
   "asyncWait": 10000,
   "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
   "messageTimestamp": "2016-02-24T09:23:46Z",
   "vehicleId": "12345678909876543"
   }
}
```

Error codes

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
406	Not Acceptable
500	Internal Server Error
501	Not Implemented
503	Service Unavailable
505	Version not supported

Access

DTC Snapshot Readout	Full access to this API

JSON Schema

See A.5

A.4.6 Use case 06 – parameterReadouts

The parameterReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.7). In some cases it is

possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a parameter readout request (see Table A.13). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A.14). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

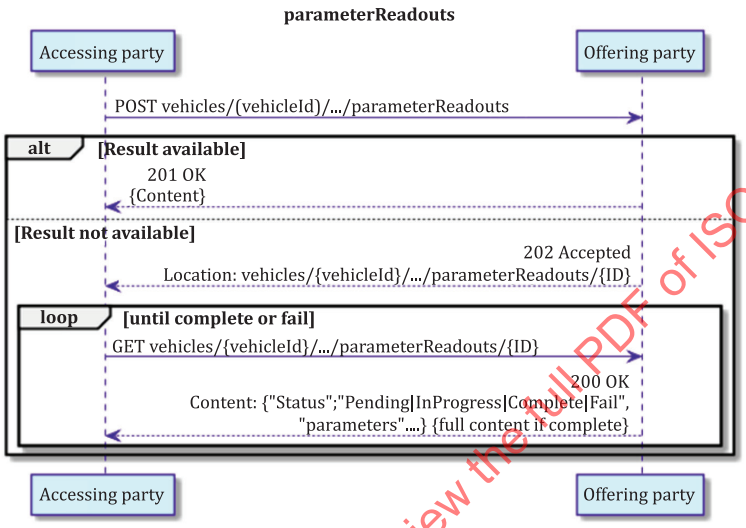

Figure A.7 — parameterReadouts sequence diagram

Table A.13 — POST parameterReadout

POST vehicles/{vehicleId}/ecuId/{ecuId}/parameterReadouts?paramId={paramId}*

Description

This API creates a readout of parameters from one ECU in a vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the parameter readout is returned. This location shall be polled until the result is available.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
ecuId	string	required	The id of the ECU to read from
paramId	string	required	The parameter id(s) to read. *) The paramId can occur many times.

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org.parameterreadout.v1+json;charset=utf-8

Response Headers

Location	Absolute URI of the /parameterReadouts endpoint

Response (Success)

201 — Example (result available immediately):

```
{
"parameterReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "ecuId": "ABC",
  "parameters": [
      {
       "id": "1234",
       "value": "648"
      },
      {
       "id": "5678",
       "value": "1000"
      }
    ]
  }
}
```

202 — Result not available immediately. Location of parameterReadout will be returned, e.g. vehicles/{vehicleId}/ecuId/{ecuId}/parameterReadouts/{id}, see Location header.

Error codes

400	Bad Request	Number of parameters exceeded
401	Unauthorized	
403	Forbidden	
404	Not Found	ECU ID not valid Parameter ID not valid
406	Not Acceptable	
500	Internal Server Error	
501	Not Implemented	
503	Service Unavailable	
505	Version not supported	

Access

Parameter Readout	Full access to this API

JSON Schema

See A.5

Table A.14 — GET parameterReadout

GET vehicles/{vehicleId}/ecuId/{ecuId}/parameterReadouts/{id}

Description

This API returns the parameter readout from one ECU in a vehicle. The id of the readout is returned when posting the request. If the parameter readout is not completed, the readout status is returned. If the parameter readout is completed, the result is returned.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
ecuId	string	required	The id of the ECU to read from
Id	string	required	Id of the parameter readout

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org. parameterreadout.v1+json;charset=utf-8

Response Headers

Content-Type	application/x.exve.org. parameterreadout.v1+json;charset=utf-8

Response (Success)

200

Example (result available):

```
{
"parameterReadout": {
   "id": "abcde-12345-ghjke-67474",
   "asyncStatus": "Complete",
   "messageTimestamp": "2016-02-24T09:23:46Z",
   "vehicleId": "12345678909876543",
   "receivedTimestamp": "2016-02-24T09:23:46Z",
   "ecuId": "ABC",
   "parameters": [
        {
         "id": "1234",
         "value": "648"
        },
        {
         "id": "5678",
         "value": "1000"
        }
     ]
  }
}
```

Example (result not available):

```
{
"parameterReadout": {
   "id": "abcde-12345-ghjke-67474",
   "asyncStatus": "InProgress",
   "asyncWait": 10000,
   "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
   "messageTimestamp": "2016-02-24T09:23:46Z",
   "vehicleId": "12345678909876543",
   "ecuId": "ABC"
  }
}
```

Error codes

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
406	Not Acceptable
500	Internal Server Error
501	Not Implemented
503	Service Unavailable
505	Version not supported

Access

Parameter Readout	Full access to this API

JSON Schema

See A.5

A.4.7 Use case 07 – malfunctionIndicatorReadouts

The malfunctionIndicatorReadout API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.8). In some cases it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a malfunction indicator readout request (see Table A.15). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A.16). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

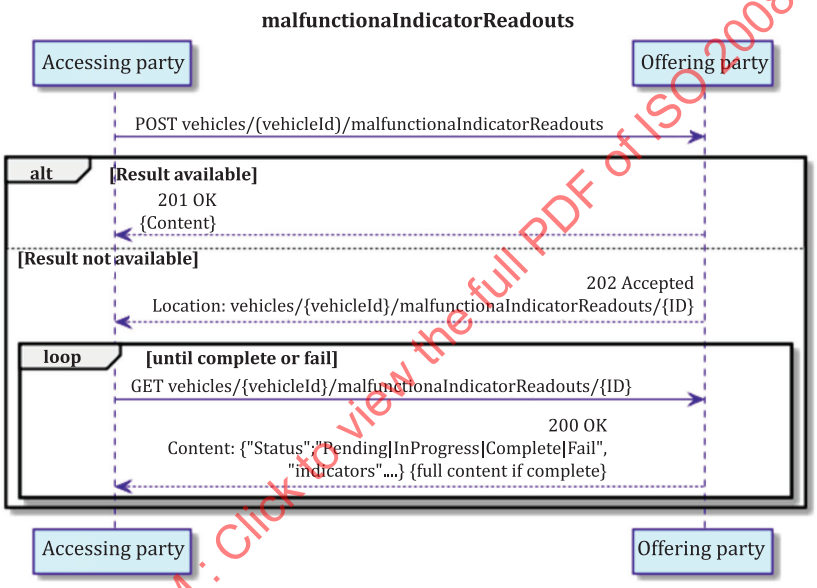

Figure A.8 — malfunctionIndicatorReadouts sequence diagram

Table A.15 — POST malfunctionIndicatorReadout

<table>
<tr><td colspan="2">POST vehicles/{vehicleId}/malfunctionIndicatorReadouts</td></tr>
<tr><td>Description</td><td>This API creates a readout of malfunction indicators from one vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the malfunction indicator readout is returned. This location shall be polled until the result is available.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.malfunctionindicatorreadout.v1+json;charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Location</td><td>Absolute URI of the /malfunctionIndicatorReadouts endpoint</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>201</td><td>Example (result available immediately):
<pre>
{
"malfunctionIndicatorReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "indicators": [
      {
       "id": "A",
       "status": "active"
      },
      {
       "id": "B",
       "status": "yellow"
      }
    ]
  }
}
</pre>
</td></tr>
<tr><td>202</td><td>Result not available immediately. Location of malfunctionIndicatorReadout will be returned, e.g. vehicles/{vehicleId}/malfunctionIndicatorReadouts/{id}, see Location header.</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>Malfunction Indicator Readout</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

Table A.16 — GET malfunctionIndicatorReadout

GET vehicles/{vehicleId}/malfunctionIndicatorReadouts/{id}

Description	This API returns the malfunction indicator readout from one vehicle. The id of the readout is returned when posting the request. If the malfunction indicator readout is not completed, the readout status is returned. If the parameter readout is completed, the result is returned.

Parameters

vehicleId	string	required	The vehicle identifier of the vehicle to read from
Id	string	required	Id of the parameter readout

Request Headers

Host	required	According to HTTP/1.1 RFC 2616
Authorization	required	Bearer {token}
Accept	required	application/x.exve.org. malfunctionindicatorreadout.v1+json;charset=utf-8

Response Headers

Content-Type	application/x.exve.org. malfunctionindicatorreadout.v1+json;charset=utf-8

Response (Success)

200 Example (result available):

```
{
"malfunctionIndicatorReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "indicators": [
       {
        "id": "A",
        "status": "active"
       },
       {
        "id": "B",
        "status": "yellow"
       }
    ]
  }
}
```

Example (result not available):

```
{
"malfunctionIndicatorReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "InProgress",
  "asyncWait": 10000,
  "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543"
  }
}
```

Error codes

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
406	Not Acceptable
500	Internal Server Error
501	Not Implemented
503	Service Unavailable
505	Version not supported

Access

Malfunction Indicator Readout	Full access to this API

JSON Schema	See A.5

A.4.8 Use case 08 – clearDtcJobs

The clearDtcJobs API follows the asynchronous interaction pattern, as the processing time of the request can vary dependent on the offering party implementation (see Figure A.9). In some cases it is possible to return the result immediately, whereas sometimes the accessing party needs to poll the API until the result is ready.

The accessing party starts by posting a clear DTC job request (see Table A.17). If the result is available immediately, the result is returned directly. If the result is not available immediately, a status is returned instead of the readout. The accessing party is supposed to poll the request status until the processing is completed (see Table A.18). When the processing is completed, the accessing party will receive the result.

The readout status and the completed readout will be available for a limited time after being created. This time is specified by the offering party.

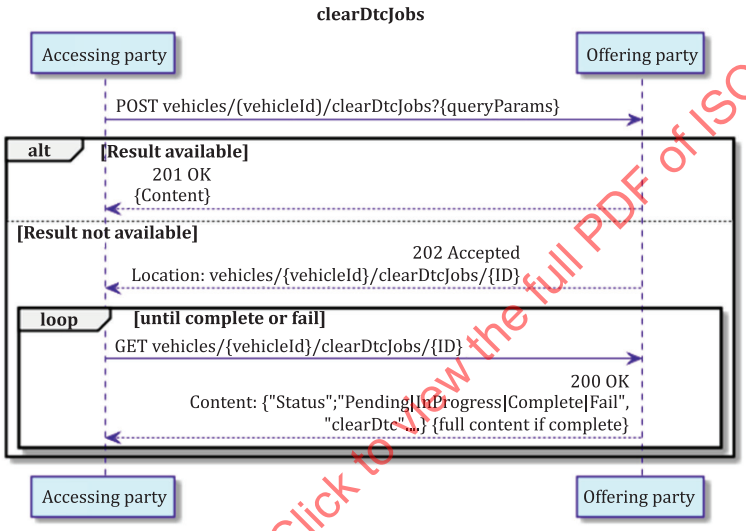

Figure A.9 — clearDtcJobs sequence diagram

Table A.17 — POST clearDtcJobs

<table>
<tr><td colspan="2">POST vehicles/{vehicleId}/clearDtcJobs?ecuId={ecuId}</td></tr>
<tr><td>Description</td><td>This API creates a clear DTC job for one vehicle. If the result is available immediately, the result is returned. If the result isn't available, a location to the clear DTC job is returned. This location shall be polled until the result is available.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
<tr><td>ecuId</td><td>string</td><td>optional</td><td>Clears DTCs on this ECU id only.
Default: Clears DTCs on all ECUs</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.cleardtcjob.v1+json; charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Location</td><td>Absolute URI of the /clearDtcJobs endpoint</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>201</td><td>Example (result available immediately):
<pre>{
"clearDtcJob": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "clearDtc": {
    "confirmation": "OK"
  }
}
}</pre></td></tr>
<tr><td>202</td><td>Result not available immediately. Location of clearDtcJob will be returned, e.g. vehicles/{vehicleId}/clearDtcJobs/{id}, see Location header.</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td><td></td></tr>
<tr><td>401</td><td>Unauthorized</td><td></td></tr>
<tr><td>403</td><td>Forbidden</td><td></td></tr>
<tr><td>404</td><td>Not Found</td><td>ECU ID not valid</td></tr>
<tr><td>406</td><td>Not Acceptable</td><td></td></tr>
<tr><td>500</td><td>Internal Server Error</td><td></td></tr>
<tr><td>501</td><td>Not Implemented</td><td></td></tr>
<tr><td>503</td><td>Service Unavailable</td><td></td></tr>
<tr><td>505</td><td>Version not supported</td><td></td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>Clear DTC Job</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

Table A.18 — GET clearDtcJobs

<table>
<tr><td colspan="2">GET vehicles/{vehicleId}/clearDtcJobs/{id}</td></tr>
<tr><td>Description</td><td>This API returns the clear DTC job result for one vehicle. The id of the job is returned when posting the request. If the job is not completed, the job status is returned. If the job is completed, the result is returned.</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><td>vehicleId</td><td>string</td><td>required</td><td>The vehicle identifier of the vehicle to read from</td></tr>
<tr><td>Id</td><td>string</td><td>required</td><td>Id of the clear DTC job</td></tr>
</table>
</td></tr>
<tr><td>Request Headers</td><td>
<table>
<tr><td>Host</td><td>required</td><td>According to HTTP/1.1 RFC 2616</td></tr>
<tr><td>Authorization</td><td>required</td><td>Bearer {token}</td></tr>
<tr><td>Accept</td><td>required</td><td>application/x.exve.org.cleardtcjob.v1+json; charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response Headers</td><td>
<table>
<tr><td>Content-Type</td><td>application/x.exve.org.cleardtcjob.v1+json;charset=utf-8</td></tr>
</table>
</td></tr>
<tr><td>Response (Success)</td><td>
<table>
<tr><td>200</td><td>Example (result available):
<pre>{
"clearDtcJob": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543",
  "receivedTimestamp": "2016-02-24T09:23:46Z",
  "clearDtc": {
    "confirmation": "OK"
  }
 }
}</pre>
Example (result not available):
<pre>{
"clearDtcJob": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "InProgress",
  "asyncWait": 10000,
  "asyncEstimatedComplete": "2016-02-24T09:24:00Z",
  "messageTimestamp": "2016-02-24T09:23:46Z",
  "vehicleId": "12345678909876543"
 }
}</pre>
</td></tr>
</table>
</td></tr>
<tr><td>Error codes</td><td>
<table>
<tr><td>400</td><td>Bad Request</td></tr>
<tr><td>401</td><td>Unauthorized</td></tr>
<tr><td>403</td><td>Forbidden</td></tr>
<tr><td>404</td><td>Not Found</td></tr>
<tr><td>406</td><td>Not Acceptable</td></tr>
<tr><td>500</td><td>Internal Server Error</td></tr>
<tr><td>501</td><td>Not Implemented</td></tr>
<tr><td>503</td><td>Service Unavailable</td></tr>
<tr><td>505</td><td>Version not supported</td></tr>
</table>
</td></tr>
<tr><td>Access</td><td>
<table>
<tr><td>Clear DTC Job</td><td>Full access to this API</td></tr>
</table>
</td></tr>
<tr><td>JSON Schema</td><td>See A.5</td></tr>
</table>

A.5 JSON schemas

A.5.1 General

A.5.1.1 Introduction

This sub-clause contains descriptions and graphical representations of the JSON schemas[2] referenced from the REST APIs. The actual schemas can be downloaded from http://standards.iso.org/iso/20080/ed-1/en. All JSON schemas are using the draft 4 version, as it has the best tooling support at the time of writing.

Legend for properties tables:

M/O – Mandatory/Optional

A.5.1.2 Extensible

All JSON schemas are extensible ("additionalProperties": true) to allow for future versions and/or additional information from the offering party.

A.5.1.3 Asynchronous Interaction Pattern

All JSON schemas supporting asynchronous REST APIs have some parameters in common (see Figure A.10) and Table A.19). Those are described here and not repeated in each section.

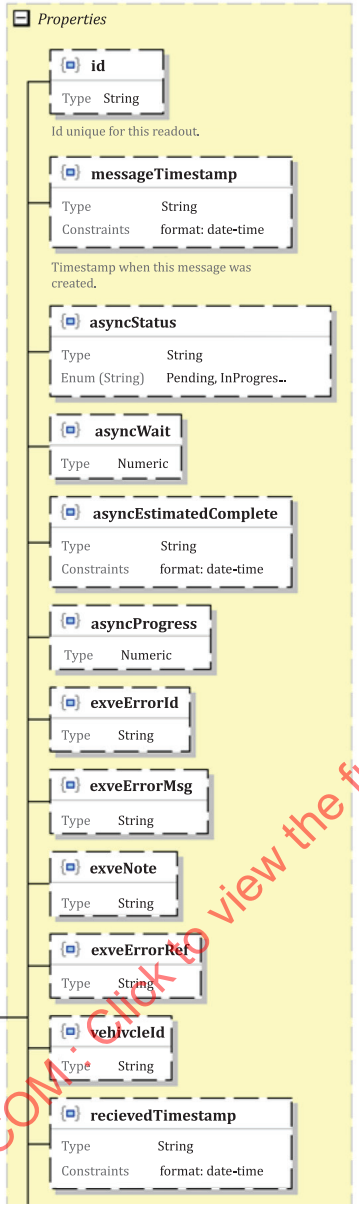

Figure A.10 — Common parameters supporting asynchronous REST APIs

Table A.19 — Common parameters supporting asynchronous REST APIs

Property	M/O	Description
id	M	The id of the readout, allocated when the readout is created.
messageTimestamp	M	Timestamp when the JSON document was created, e.g. 2017-04-12T23:20:50.52Z
asyncStatus	M	Reflects the status of the readout. Possible values are: **Pending**: The requested operation has not yet started processing. **InProgress**: The requested operation is processing but is not completed. **Complete**: The requested operation is complete and has succeeded. **Fail**: The requested operation is completed and has failed or timed out.
asyncWait	O	Recommended waiting time in milliseconds until next request for status
asyncEstimatedComplete	O	Estimated completion time of the readout
asyncProgress	O	Progress of the readout in percent