INTERNATIONAL STANDARD

ISO 26021-2

First edition
2008-07-15

Road vehicles — End-of-life activation of on-board pyrotechnic devices —

Part 2: Communication requirements

Véhicules routiers — Activation de fin de vie des dispositifs pyrotechniques embarqués —

Partie 2: Exigences de communication

Reference number
ISO 26021-2:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 26021-2 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 26021 consists of the following parts, under the general title *Road vehicles — End-of-life activation of on-board pyrotechnic devices*:

— *Part 1: General information and use case definitions*

— *Part 2: Communication requirements*

— *Part 3: Tool requirements*

— *Part 4: Additional communication line with bidirectional communication*

— *Part 5: Additional communication line with pulse width modulated signal*

NOTE Additional parts will be introduced as necessary to take into account requirements not yet covered by the standard.

Introduction

ISO 26021 describes a method for the in-vehicle deployment of pyrotechnically activated components (also referred to as pyrotechnic components or pyrotechnic devices) in cars.

Worldwide, nearly all new vehicles are equipped with one or more safety systems. Advanced protection systems using pyrotechnic actuators are becoming more common. All components which contain pyrotechnic substances should be handled in the same way.

Recycling of these vehicles requires a new process which ensures that the deactivation of airbags will be safe and cost-efficient. Based on the harmonization of the on-board diagnostics (OBD) interface, there is an opportunity to use this interface for on-board deployment, utilizing the same tools and processes.

The representatives of the global automobile industry have decided the following:

— automobile manufacturers do not support reuse as an appropriate treatment method for pyrotechnic devices;

— automobile manufacturers believe treatment of pyrotechnic devices is required before shredding;

— automobile manufacturers support in-vehicle deployment as the preferred method.

Based on this decision, the four major automobile manufacturer associations (ACEA, Alliance, JAMA and KAMA) started to develop a method for the in-vehicle deployment of pyrotechnic components in cars with the pyrotechnic device deployment tool (PDT). The vision is that, one day, a dismantler will need only one tool without any accessories in order to deploy all the pyrotechnic devices inside an end-of-life vehicle (ELV). The target is to use an existing interface to the car.

This part of ISO 26021 is applicable to the in-vehicle deployment of pyrotechnic devices in vehicles. It defines communication methods to be implemented by a pyrotechnic control unit (PCU) to allow the PDT to successfully establish and maintain communication with the PCUs in the vehicle to deploy all of the pyrotechnic devices sequentially.

Road vehicles — End-of-life activation of on-board pyrotechnic devices —

Part 2: Communication requirements

1 Scope

This part of ISO 26021 defines the deployment process, the system architecture, CAN-based communication methods and system preconditions which have to be implemented to fulfil the use cases defined in ISO 26021-1. Additionally, the relationship to and use with other existing standards are defined.

This part of ISO 26021 also describes the technical details of the on-board deployment method. The way in which the pyrotechnic devices contained in the vehicle function in conjunction with the PDT is the primary focus of this document. Under the provisions of this document, the design of the PDT or PCU can be implemented in accordance with specific functionality and hardware requirements.

This part of ISO 26021 specifies the access to the PCU. This includes communication as well as the logic sequences which are involved during the activation process.

2 Normative references

The following referenced documents are indispensable for the application 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/IEC 10731, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

ISO 11898-1, *Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling*

ISO 14229-1, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements*

ISO 15031-3, *Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics — Part 3: Diagnostic connector and related electrical circuits, specification and use*

ISO 15765-2:2004, *Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part 2: Network layer services*

ISO 15765-3:2004, *Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part 3: Implementation of unified diagnostic services (UDS on CAN)*

ISO 15765-4, *Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part 4: Requirements for emissions-related systems*

ISO 26021-1, *Road vehicles — End-of-life activation of on-board pyrotechnic devices — Part 1: General information and use case definitions*

ISO 26021-4, *Road vehicles — End-of-life activation of on-board pyrotechnic devices — Part 4: Additional communication line with bidirectional communication*

ISO 26021-5, *Road vehicles — End-of-life activation of on-board pyrotechnic devices — Part 5: Additional communication line with pulse width modulated signal*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 14229-1 and the following apply.

3.1
key
data value sent from the external test equipment to the on-board controller in response to the seed in order to gain access to the locked services

3.2
pyrotechnic device deployment tool
tool designed to be plugged into the OBD interface in order to communicate via the internal computer network in an end-of-life vehicle with all control units which are able to activate pyrotechnic devices

NOTE This tool will comprise e.g. a computer, a connection between the computer and the diagnostic connector, and some software.

3.3
pyrotechnic control unit
PCU
electronic control unit in the vehicle network which controls the activation of pyrotechnic devices

3.4
safing
mechanism whose primary purpose is to prevent an unintended functioning of the PCU processor prior to detection of a crash situation

3.5
safing unit
part of the PCU (e.g. an electromechanically operated switch or a separate processor) that allows the pyrotechnic component deployment microprocessor (µP) to deploy the pyrotechnic devices via the driver stage

3.6
scrapping program module
module responsible for firing the selected pyrotechnic device loops one by one

3.7
scrapping program module loader
module responsible for converting the scrapping program module to an executable format

3.8
seed
pseudo-random data value sent from the on-board controller to the external test equipment, which is processed by the security algorithm to produce the key

4 Symbols and abbreviated terms

ACL additional communication line

CAN controller area network

ELV end-of-life vehicle

OBD on-board diagnostics

PCU	pyrotechnic control unit
PDT	pyrotechnic device deployment tool
RAM	random access memory
SPL	scrapping program module loader
SPM	scrapping program module
SRS	supplementary restraint system
µC	microcontroller

5 Conventions

ISO 26021 is based on the conventions for the definition of OSI services (see ISO/IEC 10731) as they apply to diagnostic services.

6 Pyrotechnic device deployment via on-board diagnostic architecture

6.1 Vehicle system description

ISO 26021 is based on a vision of the diagnostic network architecture in combination with the PCU deployment architecture that is described below.

The PCU is connected to the vehicle diagnostic connector in accordance with ISO 15031-3. The PDT communicates with the PCU on CAN_H and CAN_L, which are the mandatory vehicle interfaces.

Depending upon the specific architecture of the vehicle, the mandatory link of the PCU may be connected via a gateway to the diagnostic connector; thus a CAN interface in the PCU for the mandatory link may not be required.

During the deployment procedure, the vehicle PCUs shall be powered by the vehicle battery or, if the battery is flat, by connecting an external power source to the battery terminals.

This requires an undamaged electrical architecture for the devices involved.

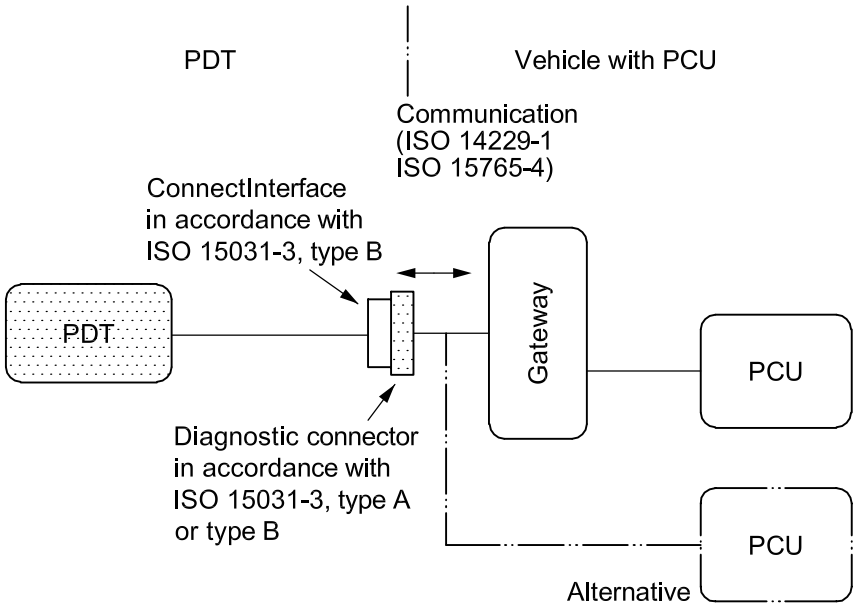


Figure 1 — Access to the vehicle via diagnostic connector

To interface the PCU with the vehicle, a PDT to diagnostic connector interface shall be used. The purpose of this interface is to

— provide the CAN bus interface and optional ACL interface with the vehicle;

— provide widely known standard wire interfaces like UART (RS232) or USB, or wireless interfaces like BLUETOOTH and WLAN.

The PDT could be based on a PC architecture running the operation system and application software from a bootable compact disc (CD) to avoid independence from software of any operating system, or the PDT could consist of a separate operating console.

6.2 Example of in-vehicle hardware and software required

To execute the on-board deployment via the communication link, the software inside the PCU shall have full access to the output driver stage, which controls the deployment loops.

In the solution without an ACL, a mechanical acceleration switch in the deployment loop circuit would not be able to carry out the redundant safing function.

To achieve a reasonable functionality without the classical safing design, an independent electronic safing unit is recommended due to the different safing philosophies of the various vehicle manufacturers. This unit could receive the required safing acceleration data from a second acceleration sensor inside the PCU or from an external frontal sensor.

Thus the deployment loop output stage is controlled by two independent branches. Depending on the safing philosophy of the vehicle manufacturer, the safing path could be controlled via the optional ACL or the main deployment processor.

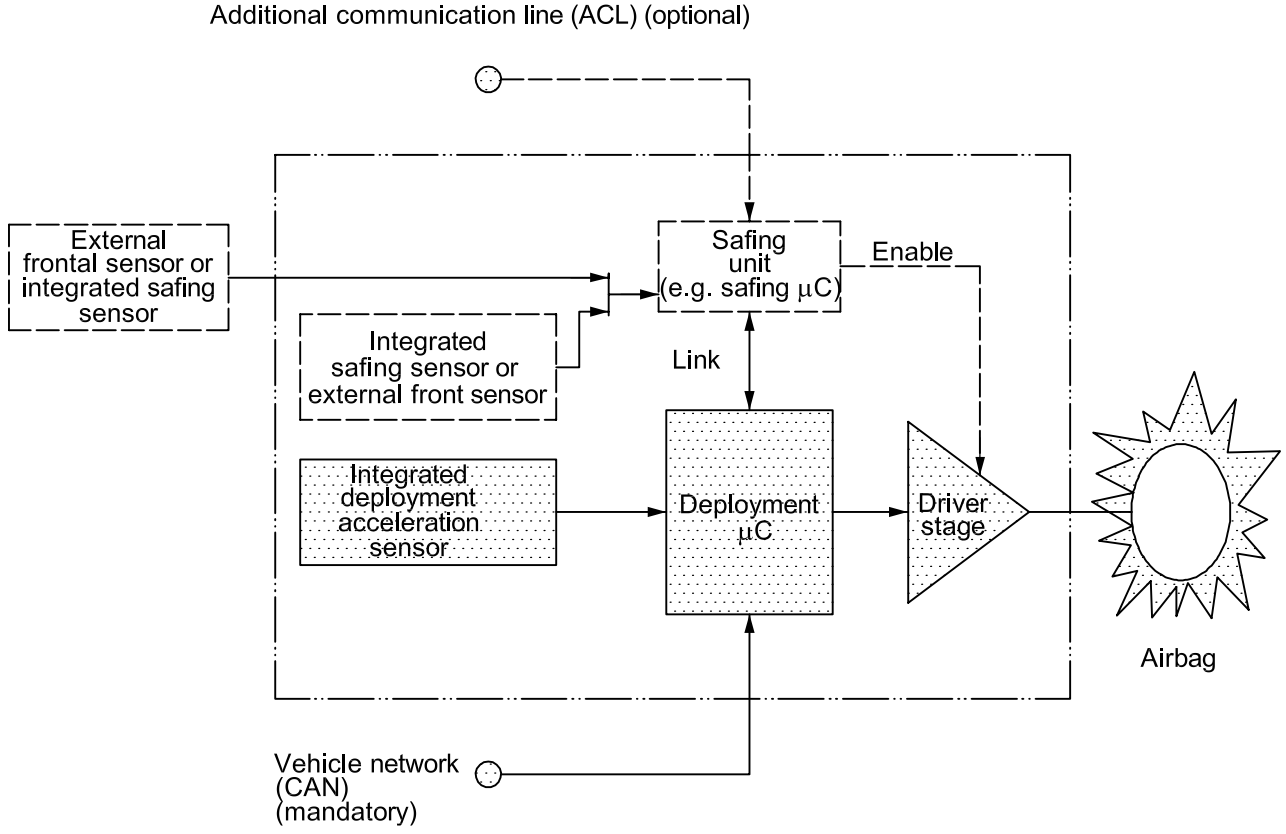


Figure 2 — Overview of hardware and software required

6.3 Additional communication line (optional)

For special safety requirements, an additional signal can be applied. General requirements for the interface between the deployment sequence and the ACL sequence are as shown in Figure 3.

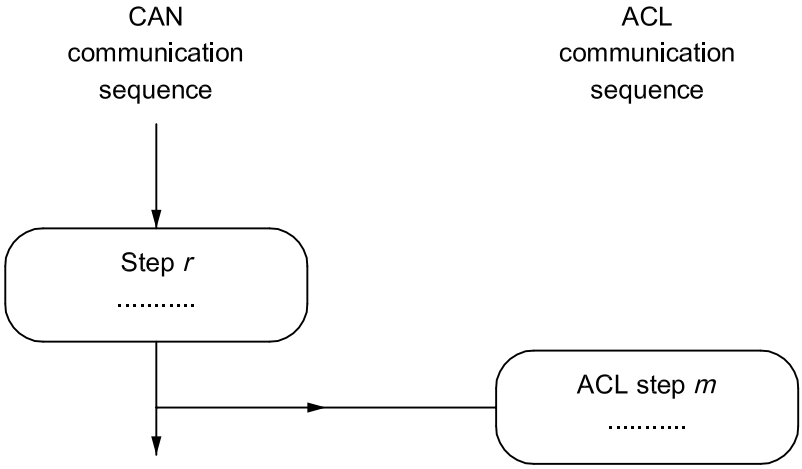


Figure 3 — Integration of ACL communication into deployment process

The standardized steps specify the diagnostic sequence. This type of step is mandatory. The PDT and the PCU shall behave as specified. The optional steps are necessary if the original equipment manufacturer (OEM) chooses the additional communication line. Optional ACL steps will depend on the use of the ACL option. Only while communicating with a PCU having an ACL is the PDT allowed to connect with the ACL line. These steps are mandatory only for the optional ACL. These optional steps are detailed in ISO 26021-4 and ISO 26021-5.

6.4 Requirements for the PDT

6.4.1 Power supply

This subclause defines the requirements for the PDT to be able to establish communication with the PCUs and successfully activate the pyrotechnic devices installed in the vehicle.

The PCUs shall be powered internally so that they are independent of the power supply of the vehicle even if pins 4 and 16 on the diagnostic connector provide a permanent power supply when the vehicle is powered.

This is to achieve robustness against a damaged power supply in the diagnostic connector. However, the tool may provide the capability to recharge its internal batteries using pins 4 and 16 of the diagnostic connector if the power supply on these pins has not been damaged.

6.4.2 Initial condition of vehicle

The operation can only start if full access to the vehicle is granted via a suitable identification method (e.g. ignition key or keyless entry unit).

It is necessary to ensure that the vehicle's electronic system is active for communication via the diagnostic connector.

6.4.3 Safety requirements

To execute the on-board deployment function via the diagnostic connection, a software module inside the PCU is required which performs the necessary steps to control the output stages, overrides the safing unit (alternative use of ACL) and carries out the communication to the PDT. To avoid any inadvertent deployment

caused by the deployment software module in the PCU, it shall be stored in a non-executable format in the program memory of the PCU and shall only be activated by an SPL which is an executable program code.

A key code from the PDT is required to enable the SPL to load the SPM into the RAM and convert the SPM to an executable format.

After a further communication sequence of the PDT with the PCU, the SPM will communicate to the independent electronic safing unit (if no ACL is available), activate the output stages and record this event individually for each deployment loop.

When an ACL is present, the unlock signal on this line has to be present during the deployment event and evaluated by the independent safing unit to release the output.

After the deployment sequence of this particular PCU is completed, the PDT may request a reset of the PCU and the PCU exits the scrapping mode.

6.4.4 Suitability of vehicle

Vehicles that can be scrapped via the diagnostic connector will respond to the PDT. All others will not.

7 Relationship to existing standards

7.1 General

All clauses of ISO 11898-1, ISO 14229-1 and the relevant part of ISO 15765 are applicable for the PCU deployment process, with the restrictions/additions defined below.

7.2 Application layer

This part of ISO 26021 uses the application layer services and protocol as defined in ISO 14229-1 for client-server-based systems to perform functions such as initialization, monitoring or start of functions of on-board vehicle servers like the PCU.

7.3 Session layer

For security reasons, the scrapping sequence shall take place in the deployment session.

7.4 Application layer and diagnostic session management timing

This part of ISO 26021 uses the application layer and session layer timing parameters as defined in ISO 14229-1 and ISO 15765-3. The detailed timing parameter descriptions for physical communication and functional communication shall be in accordance with ISO 15765-3. Although functional communication is not necessary for this part of ISO 26021, ΔP2CAN shall be between 0 ms and 500 ms. The PDT needs to detect P2CAN_Client timeout and this part of ISO 26021 specifies the ΔP2CAN value for gateway design.

In the case of a communication error, it is assumed that the client and the server implement the application and session layer timing as defined in ISO 15765-3. The client (i.e. the PDT) shall repeat the last request a maximum of two (2) times, which means that the greatest number of service request transmissions is three (3).

In the case of the worst-case communication error, the PDT shall stop the execution of the deployment process.

7.5 Network layer

The network layer of the external equipment (i.e. the PDT) and the vehicle, which may have one or more PCUs, shall be compliant with the standard diagnostic specification in ISO 15765-4, with the restrictions defined below:

— the PDT shall not transmit the FC.WAIT frame in the segmented response message;

— the PDT shall not transmit the FC frame with the non-zero block size (BS) parameter in the segmented response message;

— the PDT shall not transmit the FC frame with the non-zero separation time (STmin) parameter in the segmented response message;

— the maximum number of FC.Wait frame transmission (N_WFTmax) parameters shall be zero.

For the parameters used above, see ISO 15765-2:2004, Subclause 6.5.5, “FlowControl N_PCI parameter definition”.

From the external test equipment (PDT) point of view, each PCU in a compliant vehicle must have an addressed CAN identifier.

For the initialization sequence only, the normal addressing format and normal fixed addressing format as defined in ISO 15765-2 shall be used.

— 11 bit CAN identifiers: normal addressing;

— 29 bit CAN identifiers: normal fixed addressing.

After the initialization sequence, the OEM-specific combinations can be used as defined in Table 3 to Table 5.

7.6 Data link layer

7.6.1 11 bit CAN identifiers

Table 1 specifies the 11 bit CAN identifiers for safety-relevant initialization aspects, based on the defined mapping of the diagnostic addresses.

Table 1 — 11 bit safety-relevant identifiers

CAN identifier (hex)	Description
7F1	Physical request CAN identifier from the external test equipment to PCU #1
7F9	Physical response CAN identifier from PCU #1 to the external test equipment PDT

7.6.2 29 bit CAN identifiers

Table 2 specifies the 29 bit CAN identifiers for safety-relevant initialization aspects, based on the defined mapping of the diagnostic addresses.

Table 2 — 29 bit PCU deployment CAN identifiers

CAN identifier (hex)	Description
18 DA 53 F1	Physical request CAN identifier from the external test equipment to PCU 0x53 (hex)
18 DA F1 53	Physical response CAN identifier from PCU 0x53 (hex) to the external test equipment

The maximum number of safety-relevant ECUs in a PCU deployment compliant vehicle is not limited. The physical PCU diagnostic address of the fixed-address PCU is “0x53” hex. This address, embedded in the physical CAN identifiers, shall be unique to any one vehicle.

7.6.3 Mapping of network layer protocol data units to PCU address information of in-vehicle PCUs

The PCUAddressFormat byte defines the mapping of the request and response addresses into a 32 bit field as defined in Tables 3 to 5.

Table 3 — Reserved PCU Address Format

	8 bit field PCUAddress Format	32 bit field request/response address									
		Byte 1 (MSB)		Byte 2				Byte 3		Byte 4 (LSB)	
		31	24	23	19	18	16	15	8	7	0
Reserved	0x00	0x00, 0x00, 0x00, 0x00									
Reserved	0x07 … 0xFF	ISO/SAE reserved for future use									

Table 4 — Mapping of 11 bit N_PDU parameters into the 32 bit request/response address

11 bit mapping (e.g. CAN)	8 bit field PCUAddress Format	32 bit field request/response address See ISO 15765-2:2004, Subclause 7.3, Mapping of the N_PDU fields.									
		Byte 1 (MSB)		Byte 2				Byte 3		Byte 4 (LSB)	
		31	24	23	19	18	16	15	8	7	0
11 bit — normal addressing	0x01	0x00		0b0000 0		N_AI				0xFF (default)	
11 bit — extended addressing	0x02	0x00		0b0000 0		N_AI, except N_TA				N_TA	
11 bit — mixed addressing	0x03	0x00		0b0000 0		N_AI				N_AE	

Table 5 — Mapping of 29 bit N_PDU parameters into the 32 bit request/response address

29 bit mapping (e.g. CAN)	8 bit field PCUAddress Format	32 bit field request/response address See ISO 15765-2:2004, Subclause 7.3, Mapping of the N_PDU fields, and ISO 15765-3:2004, Subclause 8.3, Enhanced diagnostics 29 bit CAN identifiers.					
		Byte 1 (MSB)	**Byte 2**		**Byte 3**		**Byte 4 (LSB)**
29 bit — normal fixed addressing	0x04	0x00	N_TA		N_SA		0xFF
29 bit — mixed addressing	0x05	0x00	N_TA		N_SA		N_AE
ISO 15765-3 mapping		31	22	21	11	10	0
Unique addressing	0x06	0b00 0110 1111		Unique source address		Unique destination address	

7.7 Data link layer

There is no addition or restriction to the data link layer.

7.8 Physical layer

The PDT shall support all legislated OBD baud rates. If this is not possible, the PDT shall support at least the baud rates of 250 kbit/s and 500 kbit/s.

8 Deployment process

8.1 General information

This clause defines the general steps in the deployment process. It starts with the interface requirements for the external test equipment. This is followed by a description of the deployment sequence. See also Figure 4.

8.2 System preconditions

8.2.1 General

If the minimum requirements (see 6.4.2) are not fulfilled, the deployment session cannot be started. All enable conditions shall be checked before the session is initiated.

If the PCU cannot determine whether the conditions are met (e.g. due to a system failure or because of the design), it shall be assumed the conditions are fulfilled.

8.2.2 Notification enable condition

After adequate notification of the user, the system should awaken and the power to the PCUs should be switched on.

If a signal “ignition key on” or other such information is available, these shall be checked.

8.2.3 Not in motion enable condition

If the information “vehicle is not in motion” is available, it shall be checked.

8.3 Initiation of the communication between PDT and PCU

The purpose of the initiation sequence is to automatically detect if the vehicle supports standardized communication on the CAN bus, with the physical layer defined in ISO 15765-4. All related messages using this protocol shall use either a 250 kbit/s or a 500 kbit/s baud rate.

The PDT shall transmit a service ID 0x22 (Read Data By Identifier) with Data Identifier 0xFA00 “Number of PCUs in Vehicle”, using the ISO 15765-2 transport protocol with “normal fixed addressing” on the predefined Request-CAN-ID assigned to the fixed-address PCU. This service is a single-frame service.

The PDT cycles through the available protocols in the following sequence:

— ISO 15765-4 — 11 bit identifier with physical normal addressing;

— ISO 15765-4 — 29 bit identifier with physical normal fixed addressing.

Contrary to the CAN-identifiers defined in ISO 15765-4, the PDT shall use the following CAN-identifiers to establish communication with the fixed-address PCU installed in the vehicle using the CAN-connection:

11 bit CAN-ID:

— Request: 0x7F1 (fixed-address PCU)

— Response: 0x7F9 (fixed-address PCU)

— Data length code (DLC): 8

29 bit CAN-ID:

— Request: 0x18DA53F1

— Response: 0x18DAF153

— Data length code (DLC): 8

NOTE There is no support of former OBD protocols (e.g. SAE J1850 41,6 kbit/s PWM, SAE J1850 10,4 kbit/s VPW, ISO 9141-2, ISO 14230-4) or OEM-specific hooks.

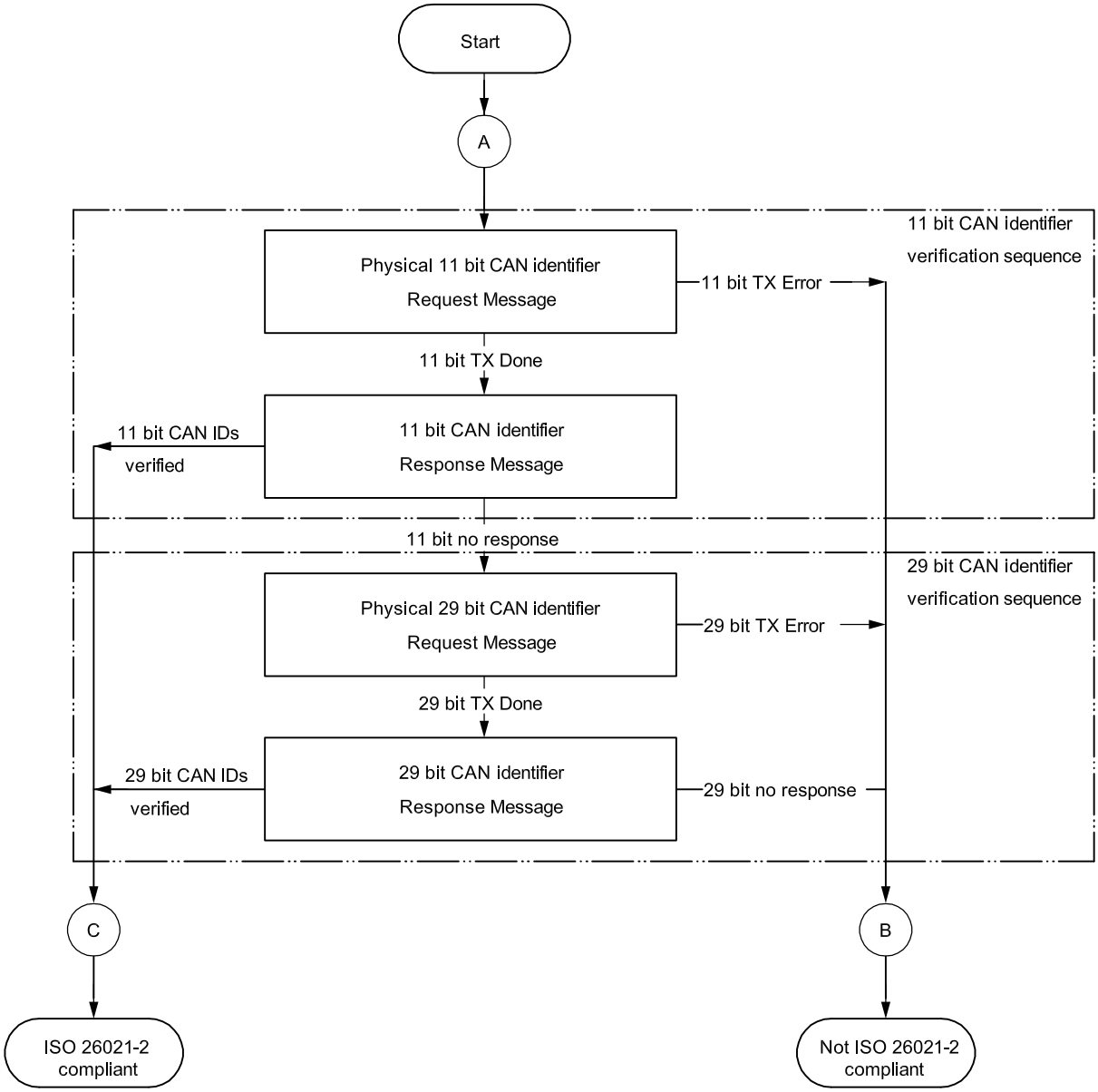


Figure 4 — CAN interface initialization sequence — Overview

8.4 Deployment process description

8.4.1 Deployment process — Overview

After the PDT has detected the PCU (connector C), the PDT shall carry out the following steps to perform the deployment process.

If the optional ACL is used, there are three possible connections to the ACL process sequence. The PDT shall manage these options at the defined steps. See Figure 5 for detailed information. Subclauses 8.4.2 to 8.4.6 contain a detailed description of the CAN communication sequence.

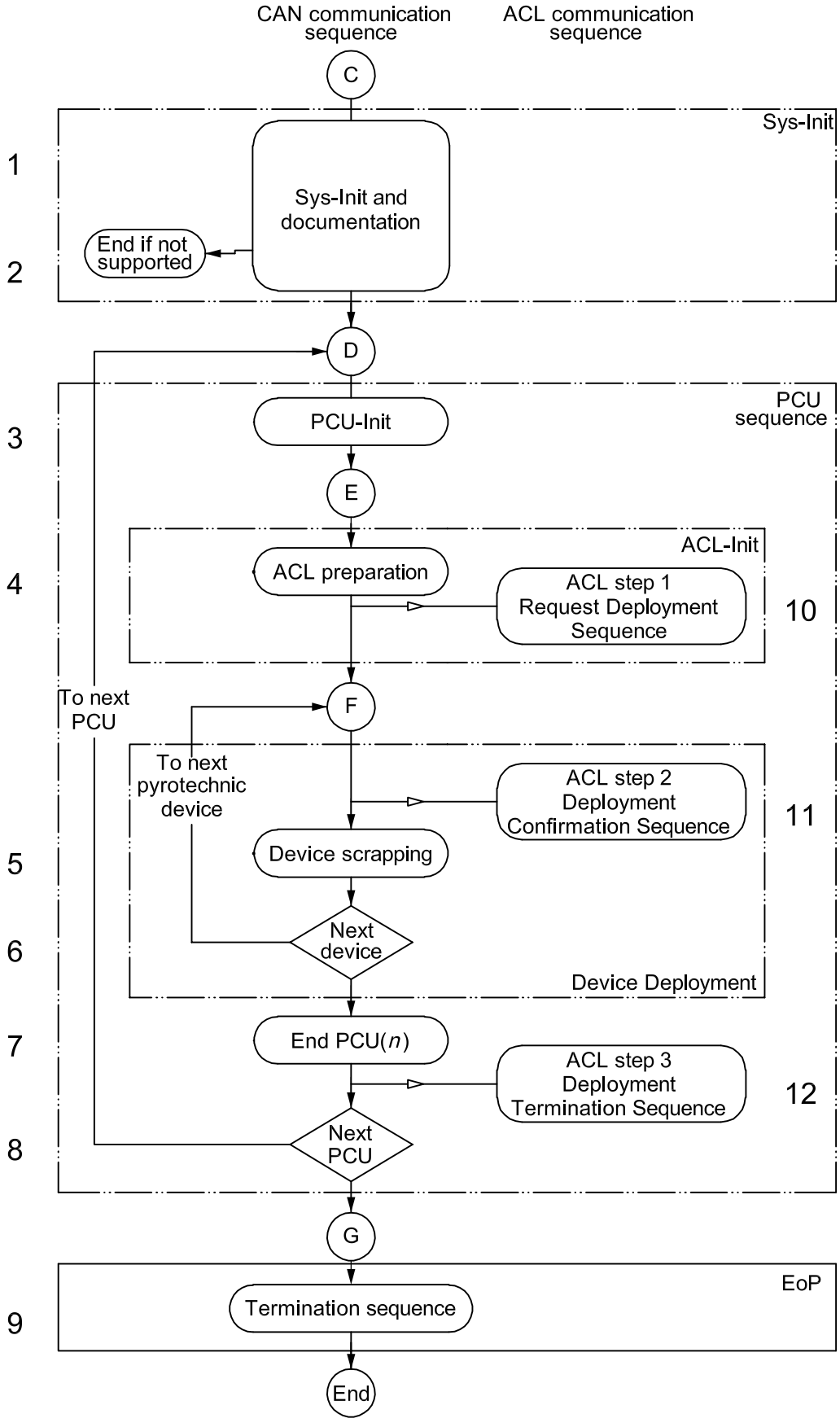


Figure 5 (*continued on next page*)

Key

1 The implementation of the PCU can differ from vehicle to vehicle. The Sys-Init part does the in-vehicle configuration. First, the PDT ascertains the identification of the vehicle. Then, the PDT starts the documentation and stores the dismantler information in the vehicle.

2 The PDT shall check whether the version of the method of deployment used by the first PCU is supported. If the version is supported, the PDT shall proceed with connector D. If the PCU version required is higher than the PDT version supported, the PDT shall end the process and update its software. Currently only version 0x01 is defined. However, the PCU deployment method version allows for extending and adapting the standard for PCU deployment to meet future demands.

3 The PDT selects the PCUs in the sequence based on the address information of the PCUs and carries out the security handling. The PDT checks the conditions, authorizes the test and selects one PCU which will be referred to as PCU(n). The PDT proceeds to connector E.

4 Currently, no additional ACL preparation in the CAN communication is necessary. The PDT shall proceed to connector F if no ACL is selected. If the ACL option is selected, the PDT shall proceed to 10.

5 The PDT reads out from PCU(n) the status of the pyrotechnic devices connected to it. Then the PDT starts the device scrapping session for this PCU(n).

6 If PCU(n) supports more than one pyrotechnic device, the PDT shall select the next device and proceed to connector F. If the last pyrotechnic device has been selected, the PDT shall end the device loop and proceed to 7.

7 The PDT ends the session with PCU(n) and proceeds to 8.

8 If there is more than one PCU, the PDT shall select the next PCU and proceed to connector E. If the last PCU has been selected, the PDT shall proceed to connector G.

9 The PDT writes the documentation to the PCU 1. The connector “End” terminates the sequence.

10 If the ACL is selected, the PDT can prepare ACL step 1 — request deployment sequence (see ISO 26021-4 or ISO 26021-5 for a detailed description).

11 If the ACL is selected, the PDT can prepare ACL step 2 — deployment confirmation sequence. If necessary, the PDT shall hold the ACL deployment process currently in progress (see ISO 26021-4 or ISO 26021-5 for a detailed description).

12 If the ACL is selected, the PDT can prepare ACL step 3 — deployment termination sequence (see ISO 26021-4 or ISO 26021-5 for a detailed description).

Figure 5 — Deployment process — Overview

8.4.2 Deployment process — Sys-Init

Most of today's vehicle designs have only one PCU. However, future designs could include more than one PCU and the PCUs could be responsible for different numbers of pyrotechnic devices. For example, today the fixed-address PCU is responsible for at least two PCU devices. The power module could be responsible for the pyrotechnic device and the battery clamp. Nevertheless, the PDT has to distinguish between a simple design with one PCU and a decentralized system with more than one PCU. The purpose of this subclause is to detect automatically the in-vehicle configuration. See Figure 6 for a detailed description.

In the Sys-Init deployment process, the vehicle is requested to provide data on the specific configuration. The PDT stores all of these data for use later in the sequence. The PCUs will be handled in exactly the sequence in which they are stored in the fixed-address PCU.

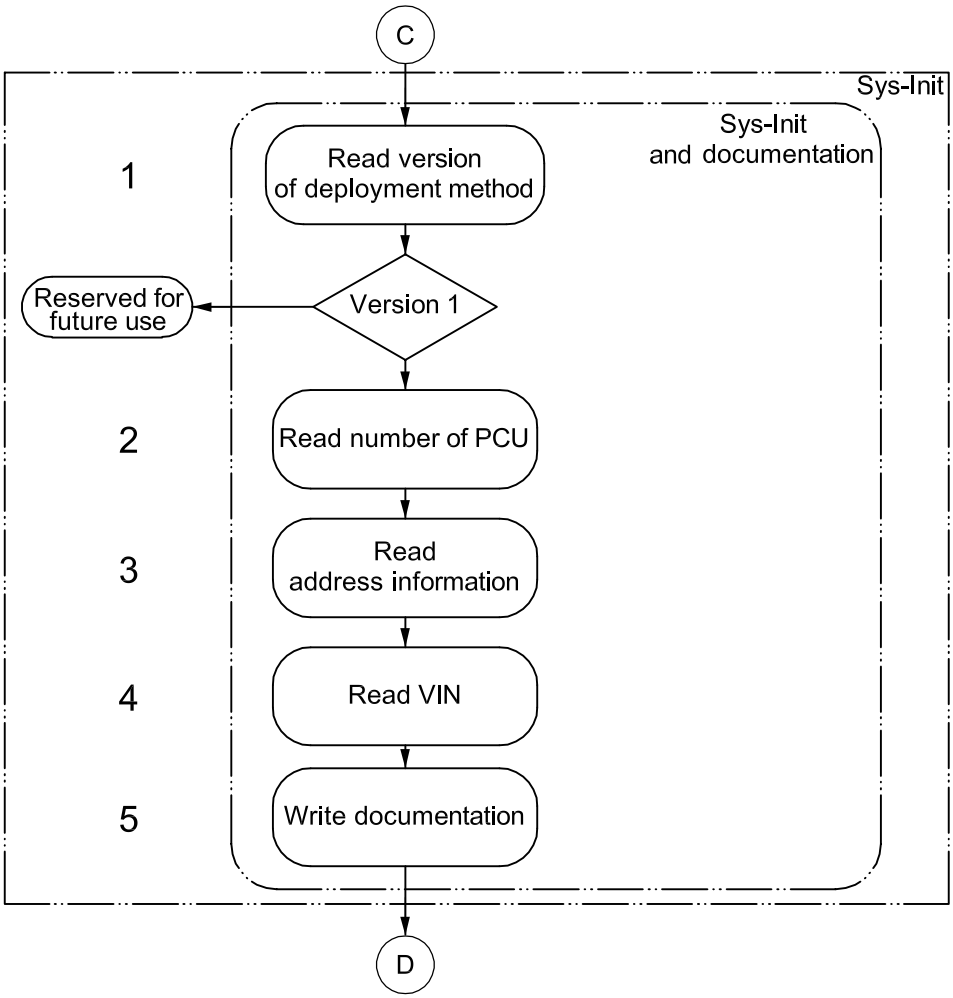
To be compatible with special requirements and future changes, the PDT requests the parameter “PCU deployment method” from the fixed-address PCU (see 9.4.4). The PDT determines from the result, “PCU deployment method version”, the protocol implemented by all the PCUs in the vehicle concerned. Currently, only version 0x01 is defined. However, the PCU deployment method version allows for extending and adapting the standard to future demands.

The parameter “ACL type” of the data identifier “deployment method version” determines the hardware design version implemented in the vehicle. If the parameter “ACL type” is set to “additional communication line is used”, the optional ACL steps are selected. If no ACL is supported, the ACL line shall be assumed to be high impedance for safety reasons. See ISO 26021-4 and the following for detailed specifications.

The PCU/PDT shall use the additional communication line type definitions specified in Table A.6 of this part of ISO 26021 (Data Identifier 0xFA06; Data Byte 1) in accordance with the descriptions in Table 6.

Table 6 — PCU/PDT description

Scaling	ACL type definition	Description
0x01	CAN only	Additional communication line is not used.
0x02	ACL_CommMode_12V	Additional communication line with bidirectional communication in accordance with ISO 26021-4 is used; vehicle battery voltage is 12 V.
0x03	ACL_PWM_FixedLevel_8V	Additional communication line with pulse width modulated signal in accordance with ISO 26021-5 is used (high level of signal is derived from a fixed voltage of 8 V which has to be provided by the DTT/PDT).
0x04	ACL_CommMode_24V	Additional communication line with bidirectional communication in accordance with ISO 26021-4 is used; vehicle battery voltage is 24 V.
0x05	ACL_PWM_UbattLevel_12V	Additional communication line with pulse width modulated signal is used. Signal timing is compliant with ISO 26021-5. Signal voltage level is compliant with ISO 26021-4 (high level of signal is derived from vehicle battery voltage which is provided via pin 16 of the diagnostic connector). Vehicle battery voltage is 12 V.
0x06	ACL_PWM_UBattLevel_24V	Additional communication line with pulse width modulated signal is used. Signal timing is compliant with ISO 26021-5. Signal voltage level is compliant with ISO 26021-4 (high level of signal is derived from vehicle battery voltage which is provided via pin 16 of the diagnostic connector). Vehicle battery voltage is 24 V.



Key

1 The PDT request data identifier 0xFA01 (PCU hardware deployment method) (see 9.4.4) from the fixed-address PCU. This data identifier contains the ACL type parameter and is used to switch the ACL interface circuit of the PDT.

2 After a successful test of the supported version, the PDT requests data identifier 0xFA00 (NumberOfPCU) (see 9.4.3) from the fixed-address PCU. This number, which defines the outer loop of the scrapping sequence, is stored for later use in the sequence of the PDT.

3 The PDT requests data identifier 0xFA02 (address information of PCUs in vehicle) from the fixed-address PCU to retrieve the CAN IDs to be used for communication with the PCUs (see 9.4.5). The address of the fixed-address PCU is defined in this part of ISO 26021. OEM-specific information is read from the fixed-address PCU to retrieve the CAN IDs to be used for communication with the PCUs.

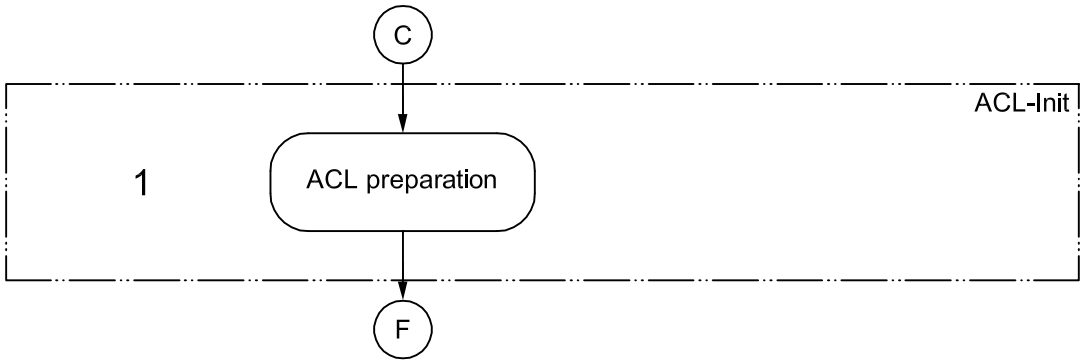
4 The dismantler's documentation contains the VIN (vehicle identification number). If the fixed-address PCU supports this DID (data identifier), it could be read automatically (see 9.4.2). The support of this DID is optional.

5 To document in the vehicle the dismantler information, the PDU writes the dismantler information into the fixed-address PCU (see 9.5.2).

Figure 6 — Deployment process — Sys-Init

8.4.3 Deployment process — ACL-Init

See Figure 7. This is for future use.



Key

1 No ACL preparation is actually necessary in the CAN communication sequence. The PDT shall proceed to connector F.

Figure 7 — Deployment process — ACL-Init

8.4.4 Deployment process — PCU sequence

This subclause describes the PCU loop. If there is more than one PCU in a vehicle, the PDT selects PCU(n).

The PDT shall monitor the diagnostic communication from connector F to the end of this security level (see Figure 8, step 5). With no diagnostic communication for more than $S3_{Server\ max}$ (5 s), the PCU in the vehicle shall end the safetySystemDiagnosticSession. To continue to function at this security level, TesterPresent has to be carried out (every 2 s) if $S3_{Client}$ timeout takes place without diagnostic communication. This service request is physically addressed to selected PCU(n)s.

Steps 1 to 3 in Figure 8 are optional. See ISO 26021-4 or ISO 26021-5 for a detailed description.

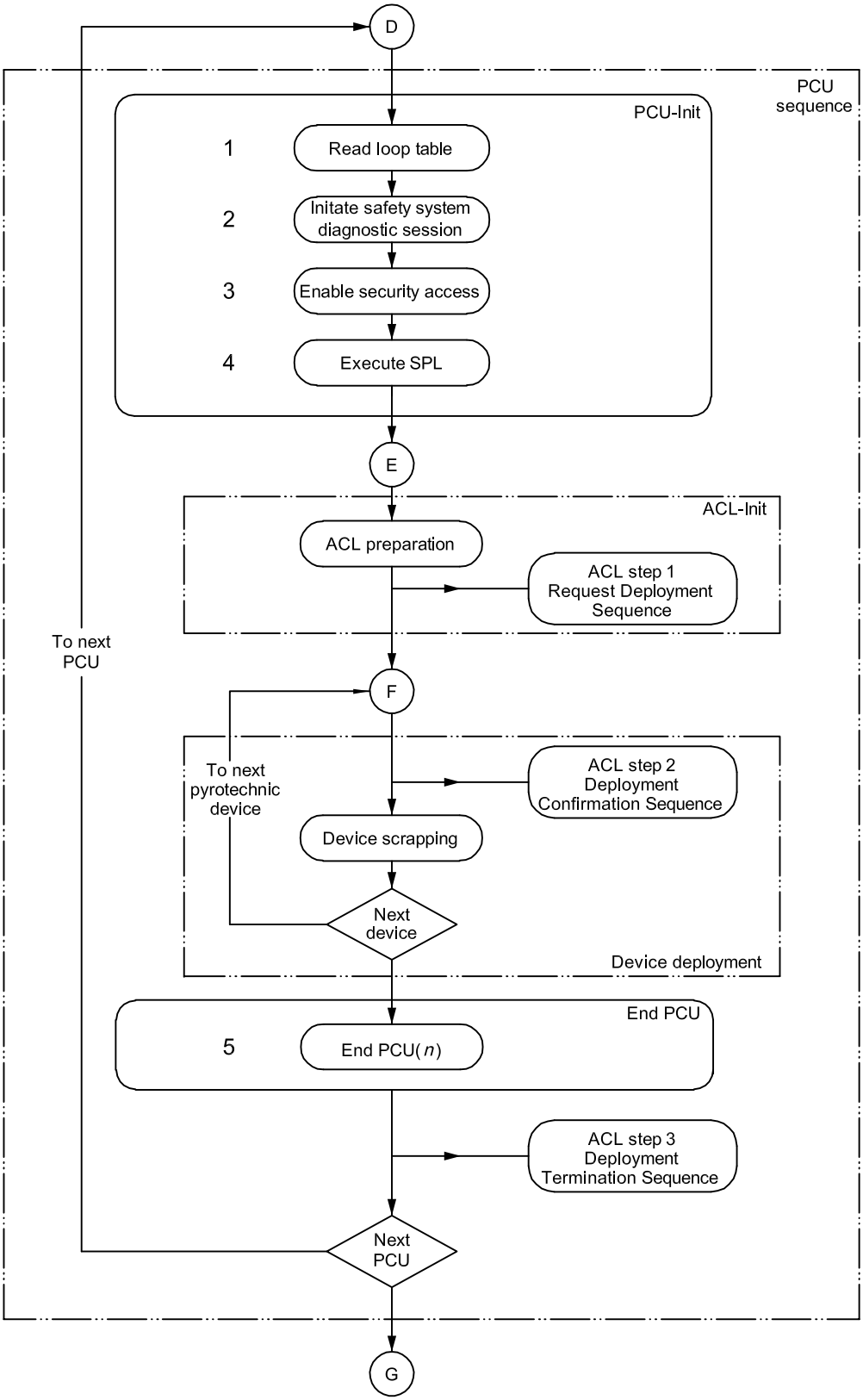


Figure 8 (*continued on next page*)

Key

1 This service (see 9.4.6) is mandatory for every PCU with one or more pyrotechnic devices. The result of this service is to determine automatically the number of pyrotechnic devices supported and their status.

2 In order to enable an additional security level, it is necessary to switch to a non-default session. In this sequence, the physically addressed PCU(n) with a pyrotechnic device is switched to the deployment session (see 9.2). Before switching to the deployment session, the PCU shall check whether the preconditions, as defined in 8.2, are met. If one of the preconditions is not met, the PCU shall respond negatively and shall not enter the deployment session.

3 When this security level is achieved, the PCU(n) is unlocked (see 9.6). Only with this security level achieved in the deployment session is the scrapping deployment loop allowed. The security level shall be valid for the entire scrapping cycle of the selected PCU(n).

4 The SPL is responsible for converting the SPM format to an executable format (see 8.5.1).

5 The PDT ends the communication with the PCU(n). The PCU(n) will not receive any diagnostic service or TesterPresent service. Therefore a timeout error will occur and the PCU(n) will end the safetySystemDiagnosticSession. This part of ISO 26021 does not specify the various implementation methods concerning how to restart valid diagnostic communication. When necessary, the PCU(n) may be reset for redeployment in case the scrapping of some of the pyrotechnic devices is interrupted.

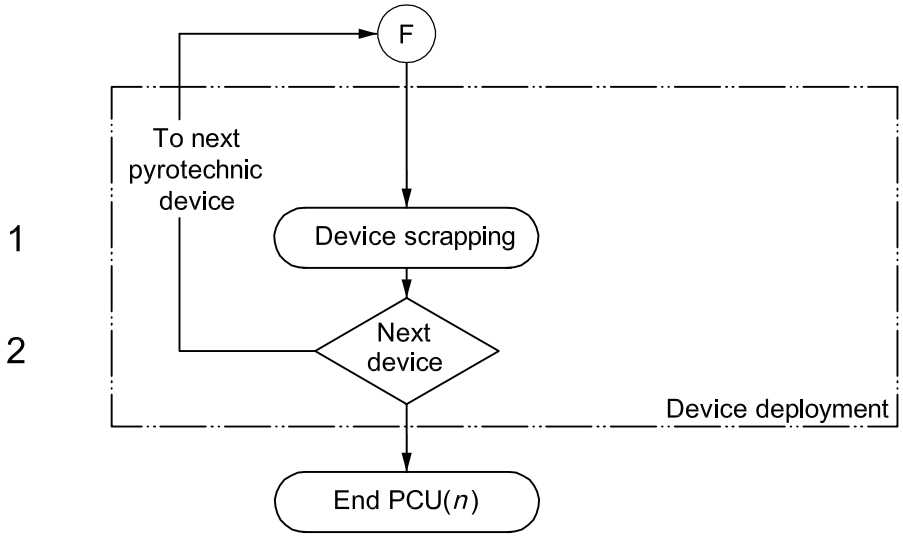
Figure 8 — Deployment process — PCU sequence

8.4.5 Deployment process — Device deployment

This subclause describes the scrapping sequence for the selected pyrotechnic device and the updating of the status bits. To avoid any inadvertent deployment caused by this program module, it is stored in a non-executable format in the program memory of the PCU and is activated by an SPL which is an executable program code.

After the scrapping of the selected pyrotechnic device of this PCU(n) has been carried out, the positive response delivers the updated status of the device. This means the PCU sends the “DeploymentLoopStatus” bit “deployedByPDT”.

NOTE The scrapping of the battery clamp has to be done as the last step of the scrapping sequence for the last PCU to be handled. If not, the PCUs would lose their power supply and communication would stop before they could send a positive response.



Key

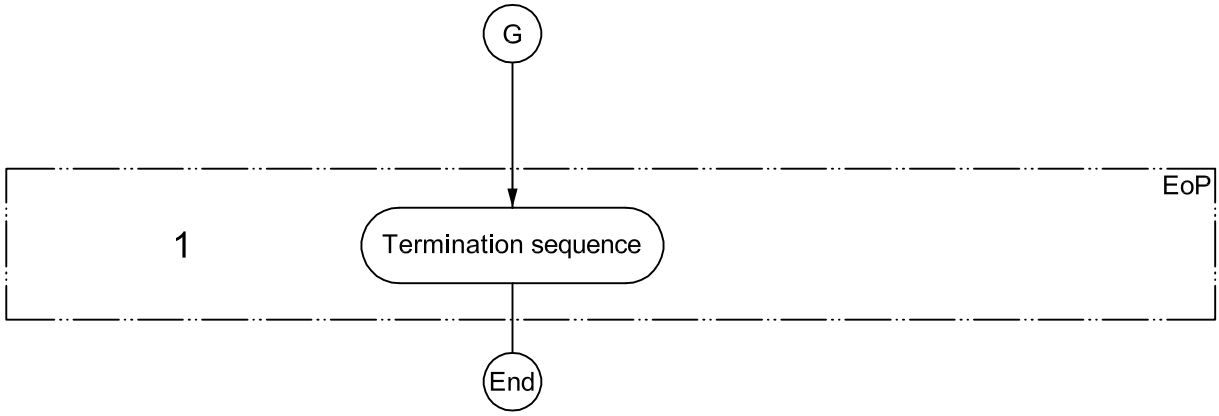
1 If all preconditions are met and the PCU is unlocked, then the PDT uses the routine control service to carry out the scrapping of one pyrotechnic device (see 9.7). After the scrapping of the first pyrotechnic device has been carried out, the normal operation mode of the PCU ends and a warning lamp, if available, shall be switched on. The status bits of the device loop are updated. If the PCU is locked, the PCU responds with a negative response code 0x33, “SecurityAccessDenied”.

2 If the PCU supports more than one pyrotechnic device, the PDT selects the next pyrotechnic device controlled by the selected PCU(n). After all the devices controlled by the PCU(n) have been scrapped, the PDT proceeds to connection “end PCU(n)” in the PCU loop.

Figure 9 — Deployment process — Device-deployment

8.4.6 Deployment process — EoP

After the deployment sequence is completed, the PDT ends the scrapping sequence with the part EndOfProgram. All of the PCUs are now in a safe status.



Key

1 The dismantler shall update his documentation, for example with the number of PCUs scrapped and the status of their pyrotechnic devices.

Figure 10 — Deployment process — EoP

8.5 Software provisions

8.5.1 Scrapping program module

For safety reasons, the SPM shall be stored in a non-executable format. The SPL is responsible for converting the SPM into an executable format.

The implementation is the responsibility of the vehicle manufacturer.

EXAMPLE

The SPM is usually stored in a ROM/FLASH memory. To reduce the RAM space required, only the security-related part of the program which enables the output stages may be executed from RAM.

— The PDT requests RoutineControl with RID = 0xE200 to the PCU.

  — The SPL copies the SPM into a free RAM space. Note that this RAM space will have to be initialized after reset.

  — The SPL converts the SPM into an executable format, e.g. the SPL overwrites some values of port-IO or function addresses with the correct values.

— The PDT requests RoutineControl with RID = 0xE201 and parameter loop ID to the PCU (for each pyrotechnic device connected to this PCU).

  — The SPM fires the pyrotechnic device.

  — The PDT receives the result.

  — The SPM updates the LoopStatus.

— The PDT requests PCU-Reset which clears the RAM and deletes the executable SPM.

8.5.2 Loop identification

To support the work of the dismantler, it is necessary to identify the number and area of installed PCU components. Every PCU can support a defined number, 1 to 255, of pyrotechnic devices. There is no standardized allocation between the firing channel and the type of PCU generator.

For simple identification of the mapped PCU channels, it is essential to have channel identification. For this reason, each PCU has an internal “loop identification table” which describes the link between the PCU and the system-specific allocation of the firing loops.

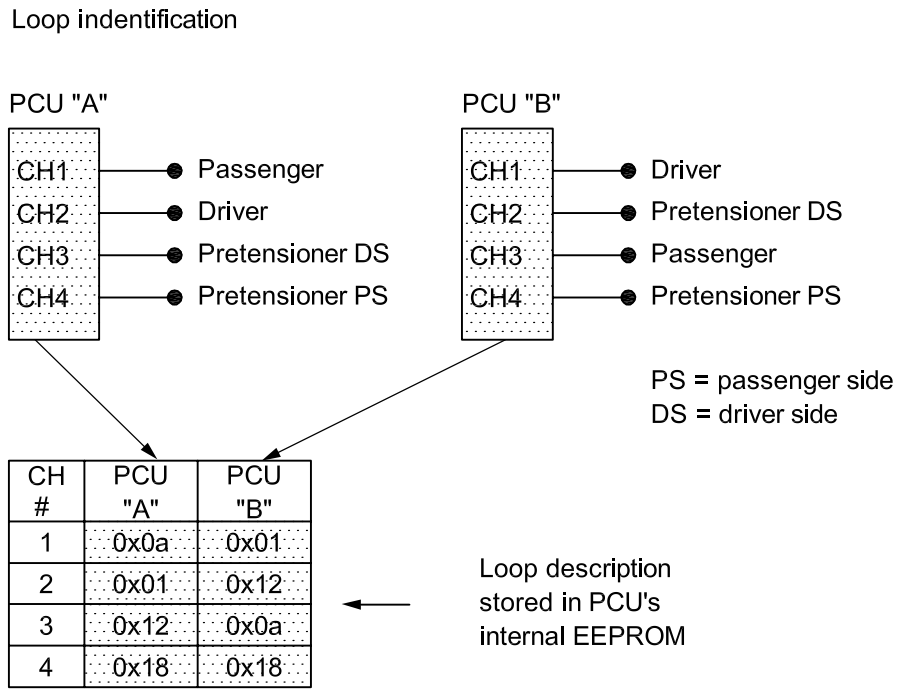


Figure 11 — Loop identification table

The “loop identification parameter” (see Annex B, Clause B.1) gives a clear explanation so as to enable every recycler to find the relevant pyrotechnic device.

EXAMPLE 1

A PCU of type “A” supports channel 1 with the content “loop ID parameter = 0x0a”.

The loop ID parameter 0x0a defines “airbag passenger side frontal 1st stage” (see Clause B.1).

EXAMPLE 2

A PCU of type “B” supports channel 1 with the content “loop ID parameter = 0x01”.

The loop ID parameter 0x01 defines “airbag driver side frontal 1st stage” (see Clause B.1).

8.6 Error handling and reaction

Any error in the sequence shall end the process with documentation of the status of the vehicle. The remaining pyrotechnic devices will have to be deployed in a conventional way (by physical removal) or by using another tool.

9 Communication with diagnostic services

9.1 Unified diagnostic services overview

This clause defines how the diagnostic services as defined in ISO 14229-1 apply to PCU deployment via on-board diagnostics. For each applicable service, the applicable sub-function and data parameters are defined (see Table 7).

Table 7 — Overview of used services

Diagnostic service name (ISO 14229-1)	Service ID value (hex)	Subclause number	Example
DiagnosticSessionControl	10	9.2	Enable safetySystemDiagnosticSession
ECUReset	11	9.3	End of scrapping
ReadDataByIdentifier	22	9.4	Read ACL version
WriteDataByIdentifier	2E	9.5	Write dismantler information
SecurityAccess	27	9.6	Enable scrapping
RoutineControl	31	9.7	Carry out scrapping
TesterPresent	3E	9.8	Scrapping in progress

9.2 Diagnostic session control (10 hex) service

The DiagnosticSessionControl service is used to enable different diagnostic sessions in the server(s) as indicated in Tables 8, 9 and 10. This message flow shows how to enable the diagnostic session "safetySystemDiagnosticSession" in a server. The client requests a response message by setting the suppressPosRspMsgIndicationBit (bit 7 of the sub-function parameter) to "FALSE" ("0"). For the PCU, it is assumed that the sessionParameterRecord is supported for the data link layer for which the service is implemented.

Table 8 — Safety system diagnostic session request message flow

Message direction:	Client → server (PDT → PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	DiagnosticSessionControl request SID	10	DSC
#2	diagnosticSessionType = safetySystemDiagnosticSession, suppressPosRspMsgIndicationBit = FALSE	04	DS_ECUSSDS

Table 9 — Safety system diagnostic session positive response message flow

Message direction:	Server → client (PCU → PDT)			
Message type:	Response			
A_Data byte	**Description (all values are in hexadecimal)**	**Cvt**	**Byte value (hex)**	**Mnemonic**
#1	DiagnosticSessionControl response SID	S	50	DSCPR
#2	diagnosticSessionType = safetySystemDiagnosticSession	M	04	DS_ECUSSDS
#3 : #*n*	sessionParameterRecord[ ]#1 = [data#1 : data#*m*]	C[a] : C	00-FF : 00-FF	SPREC_ DATA_1 : DATA_m

[a] C indicates that the presence, structure and content of the sessionParameterRecord is data link layer dependent.

Table 10 — Safety system diagnostic session negative response codes

Hex	Description	Cvt	Mnemonic
13	incorrectMessageLengthOrInvalidFormat The length of the message is wrong.	M	IMLOIF
22	conditionsNotCorrect This code shall be returned if the criteria (e.g. speed = 0, engine is off) for the request DiagnosticSessionControl are not met.	M	CNC

9.3 EcuReset (11 hex) service

The EcuReset service is used by the client to request a server reset (see Table 11). This message flow shows the preferred way of ending the diagnostic session "safetySystemDiagnosticSession" in a PCU. The client requests not to receive a positive response message by setting the suppressPosRspMsgIndicationBit (bit 7 of the sub-function parameter) to "True" ("1").

Table 11 — EcuReset request

Message direction:	Client → server (PDT → fixed-address PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	EcuReset request SID	11	ER
#2	ResetType = hardReset, suppressPosRspMsgIndicationBit = True	81	RT_HR

9.4 Read data by identifier (22 hex) service

9.4.1 Description

The ReadDataByIdentifier (RDBI) service allows the client to request data record values from the server identified by one or more dataIdentifiers. For the end-of-life activation of on-board pyrotechnic devices, the client shall only request one dataIdentifier in a request message.

In the message flow description indicated in Tables 12 and 13, the behaviour for deployment is described.

9.4.2 Read vehicle identification number (VIN)

The service RDBI with DID = F190 hex is optionally supported by the fixed-address PCU. The PDT reads the VIN with the DID specified in this subclause.

This service is physically addressed to the fixed-address PCU.

Table 12 — Read data by identifier F190 hex — VIN number

Message direction:	Client → server (PDT → fixed-address PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte Value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier request SID	22	RDBI
#2	dataIdentifier [byte#1] (MSB)	F1	DID_B1
#3	dataIdentifier [byte#2]	90	DID_B2

Table 13 — Read data by identifier F190 hex positive response

Message direction:	Server → client (fixed-address PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte Value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	F1	DID_B1
#3	dataIdentifier [byte#2]	90	DID_B2
#4	dataRecord [data#1] = VIN digit 1 = “W”	57	DREC_DATA1
#5	dataRecord [data#2] = VIN digit 2 = “0”	30	DREC_DATA2
#6	dataRecord [data#3] = VIN digit 3 = “L”	4C	DREC_DATA3
#7	dataRecord [data#4] = VIN digit 4 = “0”	30	DREC_DATA4
#8	dataRecord [data#5] = VIN digit 5 = “0”	30	DREC_DATA5
#9	dataRecord [data#6] = VIN digit 6 = “0”	30	DREC_DATA6
#10	dataRecord [data#7] = VIN digit 7 = “0”	30	DREC_DATA7
#11	dataRecord [data#8] = VIN digit 8 = “4”	34	DREC_DATA8
#12	dataRecord [data#9] = VIN digit 9 = “3”	33	DREC_DATA9
#13	dataRecord [data#10] = VIN digit 10 = “M”	4D	DREC_DATA10
#14	dataRecord [data#11] = VIN digit 11 = “B”	42	DREC_DATA11
#15	dataRecord [data#12] = VIN digit 12 = “5”	35	DREC_DATA12
#16	dataRecord [data#13] = VIN digit 13 = “4”	34	DREC_DATA13
#17	dataRecord [data#14] = VIN digit 14 = “1”	31	DREC_DATA14
#18	dataRecord [data#15] = VIN digit 15 = “3”	33	DREC_DATA15
#19	dataRecord [data#16] = VIN digit 16 = “2”	32	DREC_DATA16
#20	dataRecord [data#17] = VIN digit 17 = “6”	36	DREC_DATA17

Missing: negative response “not supported” in the case of the option.

9.4.3 Read number of PCUs in vehicle

This service is mandatory for the fixed-address PCU. It is processed in the initiation sequence with physical addressing. See Tables 14 and 15.

Table 14 — Read data by identifier FA00 hex — Number of PCUs

Message direction:	Client → server (PDT → fixed-address PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier request SID	22	RDBI
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	00	DID_B2

Table 15 — Read data by identifier FA00 hex — Number of PCUs

Message direction:	Server → client (fixed-address PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	00	DID_B2
#4	dataRecord [data#1] = number of PCUs	02	DREC_DATA1

9.4.4 Read version of deployment method

This service is mandatory for the fixed-address PCU. It is physically addressed to the fixed-address PCU. See Tables 16 and 17.

Table 16 — Read data by identifier FA01 hex — Deployment method version

Message direction:	Client → server (PDT → fixed-address PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier request SID	22	RDBI
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	01	DID_B2

Table 17 — Deployment method version positive response

Message direction:	Server → client (fixed-address PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	01	DID_B2
#4	dataRecord [data#1] = method version	01	DREC_DATA1
#5	dataRecord [data#2] = (default)	00	DREC_DATA2
#6	dataRecord [data#3] = (default)	00	DREC_DATA3
#7	dataRecord [data#4] = (default)	00	DREC_DATA4
#8	dataRecord [data#5] = (default)	00	DREC_DATA5
#9	dataRecord [data#6] = (default)	00	DREC_DATA6
#10	dataRecord [data#7] = (default)	00	DREC_DATA7
#11	dataRecord [data#8] = (default)	00	DREC_DATA8
#12	dataRecord [data#9] = (default)	00	DREC_DATA9
#13	dataRecord [data#10] = (default)	00	DREC_DATA10

9.4.5 Read address information of PCU

This service is mandatory for a fixed-address PCU.

The aim of this service with the DID is to support the already used OEM-specific addressing methods (see Tables 18, 19 and 20). The fixed-address PCU is addressed using the normal addressing format. The PDT has to support the OEM-specific address type. This service is physically addressed. The supported addresses for request and response are OEM-specifically stored in the vehicle and could be obtained automatically by the service “Read AddressInformation”.

This service is physically addressed to the fixed-address PCU.

Table 18 — Read data by identifier FA02 hex — Address information of PCU

Message direction:	Client → server (PDT → fixed-address PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier request SID	22	RDBI
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	02	DID_B2

Table 19 — Read data by identifier positive response — One PCU in vehicle

Message direction:	Server → client (fixed-address PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	02	DID_B2
#4	dataRecord [data#1] = [PCU #1 address format] = 0x01 normal addressing	01	DREC_DATA1
#5	dataRecord [data#2-5] = [PCU #1 request address]	00 00 07 F1	DREC_DATA2 DREC_DATA3 DREC_DATA4 DREC_DATA5
#9	dataRecord [data#6-9] = [PCU #1 response address]	00 00 07 F9	DREC_DATA6 DREC_DATA7 DREC_DATA8 DREC_DATA9

Table 20 — Read data by identifier positive response — Two PCUs in vehicle

Message direction:	Server → client (fixed-address PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	02	DID_B2
#4	dataRecord [data#1] = [PCU #1 address format] = 0x01 normal addressing	01	DREC_DATA1
#5	dataRecord [data#2-5] = [PCU #1 request address]	00 00 07 F1	DREC_DATA2 DREC_DATA3 DREC_DATA4 DREC_DATA5
#9	dataRecord [data#6-9] = [PCU #1 response address]	00 00 07 F9	DREC_DATA6 DREC_DATA7 DREC_DATA8 DREC_DATA9
	dataRecord [data#1] = [PCU #2 address format] = 0x02 extended addressing	02	DREC_DATA10
	dataRecord [data#2-5] = [PCU #2 request address]	00 06 F1 77	DREC_DATA11 DREC_DATA12 DREC_DATA13 DREC_DATA14
	dataRecord [data#6-9] = [PCU #2 response address]	00 06 77 F1	DREC_DATA15 DREC_DATA16 DREC_DATA17 DREC_DATA18

9.4.6 Read deployment loop table

In future designs, more than one PCU could be responsible for different numbers of deployment loops. The purpose of this service is to obtain automatically the number of loops supported. See Tables 21, 22 and 23.

This service is mandatory for every PCU with one or more pyrotechnic devices.

This service is physically addressed to one PCU.

Table 21 — Read data by identifier FA06 hex — Deployment loop table for PCU

Message direction:	Client → server (PDT → physically addressed PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier request SID	22	RDBI
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	06	DID_B2

Table 22 — Read data by identifier positive response — ACL used (CommMode12V) — One pyrotechnic device

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	06	DID_B2
#4	dataRecord [data#1] = additional communication line type definition	02	DREC_DATA1
#5	dataRecord [data#2] = additional communication line method version	01	DREC_DATA2
#5	dataRecord [data#1] = number of pyrotechnic devices	01	DREC_DATA3
#6	dataRecord [data#2] = [loopType #1]	05	DREC_DATA4
#7	dataRecord [data#3] = [loopStatus #1]	40	DREC_DATA5

In this example, the client answers with: “airbag passenger side frontal 1st stage is already deployed”.

Table 23 — Read data by identifier positive response — ACL not used (CAN only) — n pyrotechnic devices

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	06	DID_B2
#4	dataRecord [data#1] = additional communication line type definition	01	DREC_DATA1
#5	dataRecord [data#2] = additional communication line method version	01	DREC_DATA2
#6	dataRecord [data#3] = number of pyrotechnic devices	#n	DREC_DATA3
#7	dataRecord [data#4] = [loopID #1] = deployment loop ID of loop to be deployed at position #1	xx	DREC_DATA4
#8	dataRecord [data#5] = [loopStatus #1] = deployment status of loop #1	xx	DREC_DATA5
#...	……		….
#...	dataRecord [data#((n*2)+2)] = [loopID #n] = Deployment loop ID of loop to be deployed at position #n	xx	DREC_DATA (n*2+2)
#...	dataRecord [data#((n*2)+3)] = [loopStatus #n] = deployment status of loop #n	xx	DREC_DATA (n*2+3)

NOTE The order of the loop IDs in the table determines the order in which the PDT has to deploy the pyrotechnic devices.

9.4.7 Read dismantler info

This service is mandatory for every PCU.

The parameter “dismantlerIdentificationNumber” and “PCUDeploymentDeviceIdentification” are reserved for future use and shall respond with the default values. The scrapping date shall be either equal to the original production value 00 00 00 00 or equal to the actual deployment date. See Tables 24 and 25.

This service is physically addressed to one PCU.

Table 24 — Read data by identifier FA07 hex — Deployment loop table of PCU

Message direction:	Client → server (PDT → physically addressed PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier request SID	22	RDBI
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	07	DID_B2

Table 25 — Read data by identifier positive response — Dismantler information

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	ReadDataByIdentifier response SID	62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	07	DID_B2
#4	dataRecord [DismantlerNumber #1] = 0	00	DREC_DATA1
#5	dataRecord [DismantlerNumber #2] = 0	00	DREC_DATA2
#6	dataRecord [DismantlerNumber #3] = 0	00	DREC_DATA3
#7	dataRecord [DismantlerNumber #4] = 0	00	DREC_DATA4
#8	dataRecord [DismantlerNumber #5] = 0	00	DREC_DATA5
#9	dataRecord [DismantlerNumber #6] = 0	00	DREC_DATA6
#10	dataRecord [DismantlerNumber #7] = 0	00	DREC_DATA7
#11	dataRecord [DismantlerNumber #8] = 0	00	DREC_DATA8
#12	dataRecord [PDTDeviceIdentification #1] = 0	00	DREC_DATA9
#13	dataRecord [PDTDeviceIdentification #2] = 0	00	DREC_DATA10
#14	dataRecord [PDTDeviceIdentification #3] = 0	00	DREC_DATA11
#15	dataRecord [PDTDeviceIdentification #4] = 0	00	DREC_DATA12
#16	dataRecord [YearofDeploymentDate HighByte]	07	DREC_DATA13
#17	dataRecord [YearofDeploymentDate LowByte]	D6	DREC_DATA14
#18	dataRecord [MonthofDeploymentDate] = May	05	DREC_DATA15
#19	dataRecord [DayofDeploymentDate] = 01	01	DREC_DATA16

9.5 Write data by identifier (2E hex) service

9.5.1 Description

The WriteDataByIdentifier service allows the client to write information into the server at an internal location specified by the data identifier provided.

The PDT request message contains a dataRecord, the dismantler information, to write to the PCU. The data is identified by a dataIdentifier (DismantlerInformation) (see Annex A, Clause A.2) and may or may not be secured.

9.5.2 Write dismantler information

This subclause specifies the conditions to be fulfilled in the example given in Tables 26 and 27 to perform "write DismantlerInformation". The following shall be noted:

— The dismantlerIdentificationNumber is reserved for future use. The default value is 00000000.

— The PCUDeploymentDeviceIdentification is reserved for future use. The default value is 0000.

— The scrappingDate value shall be used to reference the date when the PCU was deployed. Record data content and format shall be unsigned numeric and shall be given in the order year, month, day, e.g. 2006-May-01 = 07D6 05 01.

Table 26 — Write data by identifier 0xFA07 — Dismantler information

Message direction:	Client → server		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	WriteDataByIdentifier request SID	2E	WDBI
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2]	07	DID_B2
#4	dataRecord [DismantlerNumber #1] = 0	00	DREC_DATA1
#5	dataRecord [DismantlerNumber #2] = 0	00	DREC_DATA2
#6	dataRecord [DismantlerNumber #3] = 0	00	DREC_DATA3
#7	dataRecord [DismantlerNumber #4] = 0	00	DREC_DATA4
#8	dataRecord [DismantlerNumber #5] = 0	00	DREC_DATA5
#9	dataRecord [DismantlerNumber #6] = 0	00	DREC_DATA6
#10	dataRecord [DismantlerNumber #7] = 0	00	DREC_DATA7
#11	dataRecord [DismantlerNumber #8] = 0	00	DREC_DATA8
#14	dataRecord [PDTDeviceIdentification #1] = 0	00	DREC_DATA11
#15	dataRecord [PDTDeviceIdentification #2] = 0	00	DREC_DATA12
#16	dataRecord [PDTDeviceIdentification #3] = 0	00	DREC_DATA13
#17	dataRecord [PDTDeviceIdentification #4] = 0	00	DREC_DATA14
#18	dataRecord [YearofDeploymentDate HighByte]	07	DREC_DATA15
#19	dataRecord [YearofDeploymentDate LowByte]	D6	DREC_DATA16
#20	dataRecord [MonthofDeploymentDate] = May	05	DREC_DATA17
#21	dataRecord [DayofDeploymentDate] = 01	01	DREC_DATA18

Table 27 — Write data by identifier positive response message flow example #1

Message direction:	Server → client		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	WriteDataByIdentifier response SID	6E	WDBIPR
#2	dataIdentifier [byte#1] (MSB)	FA	DID_B1
#3	dataIdentifier [byte#2] (LSB)	07	DID_B2

9.6 Security access (27 hex) service

The purpose of this service is to provide a means to access diagnostic services which have restricted access. This method is part of a set of provisions designed to fulfil safety and security requirements.

The use of diagnostic services for scrapping pyrotechnic devices is a situation where security access is required. To avoid the improper use of the scrapping mode without correct PDT implementation, the security concept uses a seed and key relationship while a safetySystemDiagnosticSession is active. It is required that the PDT first initiate the safetySystemDiagnosticSession. This session is a timed session which requires a diagnostic request service to be sent by the PDT for the session to stay active. After successful initiation of the safetySystemDiagnosticSession, the PDT sends a securityAccess request service with the SecurityAccessType set to RequestDeploymentSeed. If it supports the SecurityAccessType value, the PCU shall respond with a securityAccess positive response message. The positive response message shall also include the securitySeed value. The PDT then sends a SecurityAccess request message with the SecurityAccessType set to the SendDeploymentKey value and the securityKey value required to successfully unlock the PCU. If the securityKey value received by the PCU is incorrect, this shall be regarded as an erroneous attempt and shall be answered with a negative response message including the appropriate negative response code. The negative response codes shall be applied as specified in ISO 14229-1. There is no time period which needs to be inserted between access attempts.

The PCU shall support the 0x5f and 0x60 sub-functions only while in the safetySystemDiagnosticSession session.

For the message flow examples given in Tables 28 to 31, the following conditions have to be fulfilled to successfully unlock the server if it is in the “locked” state:

— Sub-function to request the seed: 5F hex (RequestDeploymentSeed).

— Seed of the PCU (2 bytes):

  — MSB byte PCU deployment method version = 0x01;

  — LSB byte shall be any value between 0x00 and 0xFF (e.g. random value or part of serial number or fixed value) for all PCUs.

— Sub-function to send the key: 60 hex (sendKey).

— Key of the PDT (2 bytes): the PDT compares the version number and builds the complement of the seed value:

  — MSB byte: 0xFE;

  — LSB byte: e.g. 0xAA.

— The PCU builds the complement of the seed value and compares the result with the PDT key. The PCU will switch to the deployment security level and shall send a positive response message.

The client requests a response message by setting the suppressPosRspMsgIndicationBit (bit 7 of the sub-function parameter) to “FALSE” (“0”).

Table 28 — SecurityAccess request DeploymentSeed

Message direction:	Client → server (PDT → physically addressed PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	SecurityAccess request SID	27	SA
#2	SecurityAccessType = requestSeed suppressPosRspMsgIndicationBit = FALSE	5F	LEV_SAT_RSD

Table 29 — SecurityAccess positive response message SendDeploymentSeed

Message direction:	Server → client (physical addressed PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	SecurityAccess response SID	67	SAPR
#2	securityAccessType = requestSeed	5F	SAT_RSD
#3	securitySeed [byte#1] = seed #1 (high byte)	01	SECHB
#4	securitySeed [byte#2] = seed #2 (low byte)	55	SECLB

Table 30 — SecurityAccess request SendDeploymentKey

Message direction:	Client → server		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	SecurityAccess request SID	27	SA
#2	securityAccessType = sendKey, suppressPosRspMsgIndicationBit = FALSE	60	SAT_SK
#3	securityKey [byte#1] = key #1 (high byte)	FE	SECKEY_HB
#4	securityKey [byte#2] = key #2 (low byte)	AA	SECKEY_LB

Table 31 — SecurityAccess positive response DeploymentKey

Message direction:	Server → client		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	SecurityAccess response SID	67	SAPR
#2	securityAccessType = sendKey	60	SAT_SK

9.7 RoutineControl (31 hex) service

9.7.1 Description

The RoutineControl service is used by the PDT to address the PCU in order to

— start the SPL or the SPM;

— request routine results.

The PCU shall allow execution of the two routines ExecuteSPL and DeployloopRoutine only while unlocked by the SecurityAccess service with sub-function 0x60.

9.7.2 ExecuteSPL

The ExecuteSPL is referenced by a 2 byte routineIdentifier = 0xE200 (see Annex C, Clause C.1).

Tables 32 to 35 specify start routine and request routine results. The "# of ExecuteSPL" will be selected by the routineControlOption:

— for testing use cases: 0x00 — ExecuteSPL into RAM without converting;

— used in deployment process: 0x01 — ExecuteSPL into RAM with conversion.

The client requests a response message by setting the suppressPosRspMsgIndicationBit (bit 7 of the sub-function parameter) to "FALSE" ("0").

Table 32 — RoutineControl request message — startRoutine

Message direction:	Client → server (physically addressed) (PDT → PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl request SID	31	RC
#2	Sub-function = startRoutine suppressPosRspMsgIndicationBit = FALSE	01	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	00	RI_B2
#5	routineControlOptionRecord [routineControlOption #1] = # of ExecuteSPL	01	RI_B3

Table 33 — RoutineControl positive response message — startRoutine

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl response SID	71	RCPR
#2	routineControlType = startRoutine	01	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	00	RI_B2
#5	routineStatusRecord [routineStatus#1] = OEM-specific usage	00	RI_B3
#6	routineStatusRecord [routineStatus#2] = # of ExecuteSPL	01	RI_B4

The PDT transmits a RoutineControl (31 hex) service request message to the server. The server accepts the request and starts the SPL (see 8.5.1). The PCU does not send a positive response until the routine is stopped and the scrapping module is in an executable form.

In order to keep the communication with the client active, the server may need to transmit negative response messages including response code 78 hex (requestCorrectlyReceived-ResponsePending). When the routine is completely executed, then the server transmits the RoutineControl positive response message.

In testing use cases, the PDT requests routine results. The PDT transmits a requestRoutineResult (31 hex) service request message to the server. The server accepts the request and responds with the routineControlOption "# SPL number". The result will be either 0x00 (SPL without converting) or 0x01 (SPL with conversion).

Table 34 — RoutineControl request message — requestRoutineResult

Message direction:	Client → server (physically addressed) (PDT → PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl request SID	31	RC
#2	Sub-function = requestRoutineResult suppressPosRspMsgIndicationBit = FALSE	03	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	00	RI_B2

Table 35 — RoutineControl positive response message — requestRoutineResult

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl response SID	71	RCPR
#2	routineControlType = startRoutine	03	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	00	RI_B2
#5	routineStatusRecord [routineStatus#1] = OEM-specific usage	00	RI_B3
#6	routineStatusRecord [routineStatus#2] = # of ExecuteSPL	00	RI_B4

9.7.3 DeployLoopRoutineID

The DeployLoopRoutineID is referenced by a 2 byte routineIdentifier = 0xE201 (see Annex C, Clause C.2).

Tables 36 to 39 specify start routine and request routine results. The "# of deploymentLoopID" will be selected by the routineControlOption. The definition of the deploymentLoopID parameter is specified in Annex C.

The client requests a response message by setting the suppressPosRspMsgIndicationBit (bit 7 of the sub-function parameter) to "FALSE" ("0").

Table 36 — RoutineControl request message — startRoutine

Message direction:	Client → server (physically addressed) (PDT → PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl request SID	31	RC
#2	Sub-function = startRoutine suppressPosRspMsgIndicationBit = FALSE	01	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	01	RI_B2
#5	routineControlOptionRecord [routineControlOption #1] = deploymentLoopID	*nn*	RI_B3

Table 37 — RoutineControl positive response message — startRoutine

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl response SID	71	RCPR
#2	routineControlType = startRoutine	01	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	01	RI_B2
#5	routineStatusRecord [routineStatus#1] = OEM-specific usage	00	RI_B3
#6	routineStatusRecord [routineStatus#2 = deploymentLoop]	*nn*	RI_B4
#7	routineStatusRecord [routineStatus #3] = Deployment loop status	xx	RI_B5

The PDT transmits a RoutineControl (31 hex) service request message to the server. The server accepts the request and starts the routine. For safety reasons, the scrapping program is executed out of e.g. RAM memory (see 8.5.1). The PCU does not send a positive response until the routine is stopped and the loop status is updated.

In order to keep the communication with the client active, the server may need to transmit negative response messages including response code 78 hex (requestCorrectlyReceived-ResponsePending). When the routine is completely executed, the server transmits the RoutineControl positive response message that includes the results of the check performed on the programming dependencies.

Table 38 — RoutineControl request message — requestRoutineResult

Message direction:	Client → server (physically addressed) (PDT → PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl request SID	31	RC
#2	sub-function = startRoutine suppressPosRspMsgIndicationBit = FALSE	03	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	01	RI_B2
#5	routineControlOptionRecord [routineControlOption #1] = deploymentLoopID	*nn*	RI_B3

Table 39 — RoutineControl positive response message — requestRoutineResult

Message direction:	Server → client (PCU → PDT)		
Message type:	Response		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	RoutineControl response SID	71	RCPR
#2	routineControlType = startRoutine	03	STR
#3	routineIdentifier [byte#1] (MSB)	E2	RI_B1
#4	routineIdentifier [byte#2]	01	RI_B2
#5	routineStatusRecord [routineStatus#1] = OEM-specific usage	00	RI_B3
#6	routineStatusRecord [routineStatus #2] = deploymentLoopID	*nn*	RI_B4
#7	routineStatusRecord [routineStatus #3] = Deployment loop status	xx	RI_B5

9.8 TesterPresent (3E hex) service

This service is used to indicate to all PCUs that a PDT is still connected to the vehicle and that certain diagnostic services and/or communications that have been previously activated are to remain active.

This service is used to keep the selected PCUs in a diagnostic session, especially in the session “safetySystemDiagnosticSession”. This can be done either by transmitting the TesterPresent request message periodically or, in the case of the absence of other diagnostic services, by preventing the server(s) from automatically returning to the defaultSession.

Table 40 — TesterPresent request

Message direction:	Client → server (physically addressed) (PDT → PCU)		
Message type:	Request		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte value (hex)**	**Mnemonic**
#1	TesterPresent Request Service ID	3E	TP
#2	zeroSubFunction, suppressPosRspMsgIndicationBit = TRUE	80	ZSUBF

The client requests not to receive a positive response message by setting the suppressPosRspMsgIndicationBit (bit 7 of the sub-function parameter) to “True” (“1”).

Annex A
(normative)

Specification of the data identifier used

A.1 Positive response message definition of data identifier

Table A.1 — Structure of data identifier

DID (hex)	Description	A_Data byte	Scaling
Value	Short description of data identifier General description of data identifier		
	Description of data #*n*	*n*	Format (optional formula) Value (optional data range example)

A.2 Definition of value of data identifier used

Table A.2 — Data identifier 0xFA00

DID (hex)	Description	A_Data byte	Scaling
0xFA00	Number of PCUs in vehicle (including mandatory fixed-address PCU)		
	This data identifier contains the overall number of PCUs installed in the vehicle, including the fixed-address PCU (default 1). For example, if there is only the fixed-address PCU in the vehicle, a value of one (1) shall be reported. If there are two (2) additional PCUs in the vehicle, the value reported shall be three (3). This data identifier shall only be supported by the fixed-address PCU in the vehicle. This data identifier shall be read-only and shall not be changed to a multiple-frame definition.	1	8 bit unsigned character 0x00: not valid 0x01 to 0xF0: number 0xF0 to 0xFF: reserved

Table A.3 — Data identifier 0xFA01

DID (hex)	Description	Data byte	Scaling
0xFA01	PCU HW-deployment method (mandatory fixed-address PCU)		
	This data identifier contains the version of the PCU deployment method implemented by the PCU and an identification string for the PCU. This data identifier shall be read-only.		
	PCU deployment method version (ADMV) The PCU deployment method version identifies the version of the diagnostic protocol services and the sequence used for the PCU deployment. It shall be incremented every time a protocol service or a data identifier is changed in ISO 26021 and is no longer backward-compatible.	1	8 bit unsigned character 0x00: reserved 0x01: ISO 26021 0x02 to 0xFF reserved 0x00: reserved
	PCU identification string This contains a PCU identification string that is assigned to the PCU by the OEM.	2…10	8 bit unsigned character 0x00: default value
	Additional data for future use		

Table A.4 — Data identifier 0xFA02

DID (hex)	Description	Data byte	Scaling
0xFA02	Address information on PCUs in vehicle (mandatory fixed-address PCU)		
	This data identifier contains the address type (11 bit, 29 bit) and address numbers (request and response) to be used to communicate with the PCUs in the vehicle. This information is dependent upon the physical link (see ISO 15765-4). Only “normal addressing” and “normal fixed addressing” as defined in ISO 15765-2 shall be supported on CAN. This data identifier shall only be supported by the fixed-address PCU in the vehicle. The structure defined below shall be repeated in the response message for each PCU (including the fixed-address PCU) in the vehicle. This data identifier shall be read-only.		
	PCU address format #1 This format identifier defines the format of the 1st PCU's address information. The 1st PCU's address information type contains the address format of the PCU to be deployed first. Possible address information types are defined in PCUAddressFormat in 7.6.3.	1	8 bit unsigned character 0x01: 11 bit normal addressing 0x02: 11 bit extended addressing 0x03: 11 bit mixed addressing 0x04: 29 bit normal fixed addressing 0x05: 29 bit mixed addressing 0x06: 29 bit unique addressing

Table A.4 (*continued*)

DID (hex)	Description	Data byte	Scaling
	PCU request address #1 This parameter contains the diagnostic request address to which the PDT must transmit the diagnostic requests to communicate with a PCU. Depending on the address information format, this is either an 11 bit or a 29 bit CAN-identifier. The 1st PCU request address shall be that of the PCU to be fired first. The unused most significant bits shall be padded with zeros (0).	2…5	32 bit unsigned integer
	PCU response address #1 This parameter contains the diagnostic response address to which the PCU will respond to the requests of the PDT. Dependent upon the address information format this is either an 11 bit or 29 bit CAN-identifier or an 8 bit K-Line address. The 1st PCU request address shall be that of the PCU to be fired first. The unused most significant bits shall be padded with zeros (0).	6…9	32 bit unsigned integer
	PCU address format #n This format identifier defines the format of the nth PCU's address information.	$(n - 1) * 9 + 1$	8 bit unsigned character (see PCU address format #1)
	PCU request address #n This is the diagnostic request address of the nth PCU in the vehicle.	$(n - 1) * 9 + 2$ … $(n - 1) * 9 + 5$	32 bit unsigned integer
	PCU response address #n This is the diagnostic response address of the nth PCU in the vehicle.	$(n - 1) * 9 + 6$ … $(n - 1) * 9 + 9$	32 bit unsigned integer

Table A.5 — Data identifiers 0xFA03 to 0xFA05

DID (hex)	Description	A_Data byte	Scaling
0xFA03	Reserved for future use		
	To be defined, if used	1	8 bit unsigned character
0xFA04	Reserved for future use		
	To be defined, if used	1	8 bit unsigned character
0xFA05	Reserved for future use		
	To be defined, if used	1	8 bit unsigned character

Table A.6 — Data identifier 0xFA06

DID (hex)	Description	Data byte	Scaling
0xFA06	Number of deployment loop table records and status		
	This data identifier contains the number of loop table records in this PCU. Every loop record is made up of the type and the associated status of the deployment loops supported by the PCU. This data identifier shall be read-only.		
	Additional communication line type definition The PCU deployment identifies the type of ACL required by the diagnostic protocol services and the sequence used for deployment of the pyrotechnic device. It shall be incremented every time a protocol service or a data identifier is changed and is no longer backward-compatible.	1	8 bit unsigned character 0x01: CAN only 0x02: ACL_CommMode_12V 0x03: ACL_PWM_FixedLevel_8V 0x04: ACL_CommMode_24V 0x05: ACL_PWM_UbattLevel_12V 0x06: ACL_PWM_UBattLevel_24V 0x07 to 0xFF: reserved
	Additional communication line method version The ACL method version identifies the version of the diagnostic protocol services and sequence used for PCU deployment. It shall be incremented every time a protocol service or a data identifier is changed in the relevant parts of ISO 26021 and is no longer backward-compatible.	2	8 bit unsigned character 0x00: reserved 0x01: ISO 26021 0x02 to 0xFF: reserved for new communication line methods
	Number of loop table records	3	8 bit unsigned character 0x00: not valid 0x01 to 0xF0: number 0xF0 to 0xFF: reserved
	Loop identification #1 This parameter contains the identification of the 1st loop in the PCU providing the function this loop is assigned to.	4	8 bit unsigned character Refer to B.1 for the definition of the available loop IDs.
	Loop status #1 This parameter contains the current status of the loop identified in the parameter “loop identification”.	5	8 bit unsigned character Refer to B.2 for the definition of the loop status information.
	Loop identification #n This parameter contains the identification of the nth loop in the PCU providing the function this loop is assigned to.	(n * 2) + 2	8 bit unsigned character Refer to B.1 for the definition of the available loop IDs.
	Loop status #n This parameter contains the current status of the loop identified in the parameter “loop identification”.	(n * 2) + 3	8 bit unsigned character Refer to B.2 for the definition of the loop status information.

Table A.7 — Data identifier 0xFA07

DID (hex)	Description	Data byte	Scaling
0xFA07	Dismantler information		
	This data identifier contains the dismantler identification number. It shall be written to the PCU prior to the execution of any loop ignition procedure. This data identifier shall be readable and writeable. This data identifier shall be locked and be made read-only after the first successful write access.		
	Dismantler identification number This parameter identifies the dismantler who has executed the PCU deployment sequence.	1…8	8 bit unsigned character reserved for future use default: 0x00
	PCU deployment device identification This parameter identifies the PDT deployment tool that was involved in the PCU deployment sequence.	9…12	8 bit unsigned character reserved for future use default: 0x00
	Year of deployment This parameter contains the year in which the PCU deployment sequence was executed. (0x7D5 = “2005”) If this parameter has never been successfully written before, the default value (0x0000) shall be reported when this parameter is read.	13 to 14	16 bit unsigned integer 0x0000: default 0x0001 to 0x07D4: reserved 0x07D5 to 0xFFFE: year 0xFFFF: default
	Month of deployment This parameter contains the month in which the PCU deployment sequence was executed. If this parameter has never been successfully written before, the default value (0x00) shall be reported when this parameter is read.	15	8 bit unsigned character 0x00: default 0x01 to 0x0C: month 0x0D to 0xFF: reserved
	Day of deployment This parameter contains the day of the month on which the PCU deployment sequence was executed. If this parameter has never been successfully written before, the default value (0x00) shall be reported when this parameter is read.	16	8 bit unsigned character 0x00: default 0x01 to 0x1F: day 0x20 to 0xFF: reserved

Table A.8 — Data identifier 0xFA08 to 0xFA0F

DID (hex)	Description	A_Data byte	Scaling
0xFA08 to 0xFA0F	Reserved for future use		
	To be defined, if used	1	8 bit unsigned character

Annex B
(normative)

Deployment loop parameter definitions

B.1 Deployment loop ID parameter definitions

Deployment loop IDs shall be defined as in Table B.1.

Table B.1 — Deployment loop ID definitions

Hex	Description
0x00	ISOSAEReserved This value is reserved by this document for a future definition.
0x01	airbag driver side frontal 1st stage
0x02	airbag left side frontal 1st stage
0x03	airbag right side frontal 1st stage
0x04	airbag driver side frontal 2nd stage
0x05	airbag left side frontal 2nd stage
0x06	airbag right side frontal 2nd stage
0x07	airbag driver side frontal 3rd stage
0x08	airbag left side frontal 3rd stage
0x09	airbag right side frontal 3rd stage
0x0a	airbag passenger side frontal 1st stage
0x0b	airbag passenger side frontal 2nd stage
0x0c	airbag passenger side frontal 3rd stage
0x0d	airbag left side frontal 3rd stage
0x0e	airbag right side frontal 3rd stage
0x0f	airbag passenger frontal 1st stage — centre
0x10	airbag passenger frontal 2nd stage — centre
0x11	airbag passenger frontal 3rd stage — centre
0x12	1st pretensioner driver side
0x13	1st pretensioner left side
0x14	1st pretensioner right side
0x15	2nd pretensioner driver side
0x16	2nd pretensioner left side
0x17	2nd pretensioner right side
0x18	1st pretensioner passenger side
0x19	2nd pretensioner passenger side
0x1a	1st pretensioner passenger — centre

Table B.1 (*continued*)

Hex	Description
0x1b	2nd pretensioner passenger — centre
0x1c	1st pretensioner (2nd row) left
0x1d	2nd pretensioner (2nd row) left
0x1e	1st pretensioner (2nd row) right
0x1f	2nd pretensioner (2nd row) right
0x20	1st pretensioner (2nd row) centre
0x21	2nd pretensioner (2nd row) centre
0x22	1st pretensioner (3rd row) left
0x23	2nd pretensioner (3rd row) left
0x24	1st pretensioner (3rd row) right
0x25	2nd pretensioner (3rd row) right
0x26	1st pretensioner (3rd row) centre
0x27	2nd pretensioner (3rd row) centre
0x28	belt-force limiter driver side
0x29	belt-force limiter left side
0x2a	belt-force limiter right side
0x2b	belt-force limiter passenger side
0x2c	belt-force limiter passenger — centre
0x2d	belt-force limiter 2nd row — left
0x2e	belt-force limiter 2nd row — right
0x2f	belt-force limiter 2nd row — centre
0x30	belt-force limiter 3rd row — left
0x31	belt-force limiter 3rd row — right
0x32	belt-force limiter 3rd row — centre
0x33	headbag — driver side roof mounted
0x34	headbag — passenger side roof mounted
0x35	headbag — right side roof mounted
0x36	headbag — left side roof mounted
0x37	headbag — 2nd row — left roof mounted
0x38	headbag — 2nd row — right roof monted
0x39	headbag — 3rd row — left roof mounted
0x3a	headbag — 3rd row — right roof mounted
0x3b	sidebag (curtain) — driver side

Table B.1 (*continued*)

Hex	Description
0x3c	sidebag (curtain) — passenger side
0x3d	sidebag (curtain) — left side
0x3e	sidebag (curtain) — right side
0x3f	sidebag (curtain) — 2nd row — left
0x40	sidebag (curtain) — 2nd row — right
0x41	sidebag (curtain) — 3rd row — left
0x42	sidebag (curtain) — 3rd row — right
0x43	sidebag — driver side door mounted
0x44	sidebag — passenger side door mounted
0x45	sidebag — left side door mounted
0x46	sidebag — right side door mounted
0x47	sidebag — 2nd row — left door mounted
0x48	sidebag — 2nd row — right door mounted
0x49	sidebag — 3rd row — left door mounted
0x4a	sidebag — 3rd row — right door mounted
0x4b	seatbag (cushion) — driver side seat mounted
0x4c	seatbag (cushion) — passenger side seat mounted
0x4d	seatbag (cushion) — left side seat mounted
0x4e	seatbag (cushion) — right side seat mounted
0x4f	seatbag (cushion) — 2nd row — left seat mounted
0x50	seatbag (cushion) — 2nd row — right seat mounted
0x51	seatbag (cushion) — 3rd row — left seat mounted
0x52	seatbag (cushion) — 3rd row — right seat mounted
0x53	kneebag — driver side
0x54	kneebag — passenger side
0x55	kneebag — left side
0x56	kneebag — right side
0x57	kneebag — passenger side — centre

Table B.1 (*continued*)

Hex	Description
0x58	footbag — driver side
0x59	footbag — passenger side
0x5a	footbag — left side
0x5b	footbag — right side
0x5c	footbag — passenger side — centre
0x5e	active headrest — driver side
0x5f	active headrest — passenger side
0x60	active headrest — left side
0x61	active headrest — right side
0x62	active headrest — passenger side — centre
0x63	active headrest — 2nd row — left
0x64	active headrest — 2nd row — right
0x65	active headrest — 2nd row — centre
0x66	active headrest — 3rd row — left
0x67	active headrest — 3rd row — right
0x68	active headrest — 3rd row — centre
0x69	battery clamp main battery
0x6a	battery clamp 2nd battery
0x6b	battery clamp 3rd battery
0x6c	battery clamp 4th battery
0x6d	roof-airbag front
0x6e	roof-airbag front
0x6f	bag in belt driver side
0x70	bag in belt passenger side
0x71	bag in belt left side
0x72	bag in belt right side
0x73	bag in belt passenger side — centre
0x74	bag in belt 2nd row — left
0x75	bag in belt 2nd row — right
0x76	bag in belt 2nd row — centre
0x77	bag in belt 3rd row — left
0x78	bag in belt 3rd row — right
0x79	bag in belt 3rd row — centre
0x7a	rollover bar #1
0x7b	rollover bar #2
0x7c	rollover bar #3
0x7d	rollover bar #4
0x7e	active anti-submarining driver seat

Table B.1 (*continued*)

Hex	Description
0x7f	active anti-submarining passenger seat
0x80	active anti-submarining left seat
0x81	active anti-submarining right seat
0x82	active anti-submarining passenger seat — centre
0x83	active anti-submarining seat 2nd row — left
0x84	active anti-submarining seat 2nd row — right
0x85	active anti-submarining seat 2nd row — centre
0x86	active anti-submarining seat 3rd row — left
0x87	active anti-submarining seat 3rd row — right
0x88	active anti-submarining seat 3rd row — centre
0x89	pedestrian protection front left
0x8a	hood lifter
0x8b	pedestrian protection front right
0x8c	hood lifter
0x8d	pedestrian protection rear left
0x8e	hood lifter
0x8f	pedestrian protection rear right
0x90	hood lifter
0x91	pedestrian protection a-pillar left
0x92	pedestrian protection a-pillar right
0x93	pedestrian protection wind screen
0x94	pedestrian protection bumper left
0x95	pedestrian protection bumper centre
0x96	pedestrian protection bumper right
0x97	active steering column
0x98	front screen — emergency release
0x99	rear window — emergency release
0x9a to 0xBF	reserved for future use
0xC0 to 0xEF	vehicle-manufacturer-specific
0xF0 to 0xFF	ISOSAEReserved These values are reserved by this part of ISO 26021 for an extension of the DeploymentLoopID from an 8 bit to a 16 bit range.

B.2 Deployment loop status parameter definitions

The “deployment loop status” shall be defined as in Table B.2 if the corresponding bit is set to “1”.

Table B.2 — Deployment loop status definitions

Encoding of bit	Value	Description	Cvt	Mnemonic
0-1	000b	**ISOSAEReserved** These bits are reserved by this document for future definitions. Default value = “0”	M	
2	1	**Operating status** If set to “1”, the PCU is out of the operating mode. The deployment will not be successful. CAUTION — External deployment will be required. Default value = “0”	U	
3	1	**Deactivation status** If set to “1”, the pyrotechnic device is deactivated by software or a switch and disconnected. CAUTION — External deployment will be required. Default value = “0”	U	
4	1	**Inhibit status** If set to “1”, the pyrotechnic device is inhibited, e.g. by a removable seat. No additional work is required. Default value = “0”	U	
5	1	**Deployed status** If set to “1”, the pyrotechnic device is deployed by a PDT. No additional work is required. The warning lamp shall be switched on if this flag is set to “1”.	M	
6	1	**Normal deployed status** This bit is reserved by this document for a future definition. Default value = “0”	U	
7	1	**Failure status** An electrical fault in the firing loop, e.g. interrupted, short to ground, has deactivated this loop. CAUTION — External deployment will be required. Default value = “0”	U	

Annex C
(normative)

Routine control parameter definitions

C.1 RoutineIdentifier data parameter definitions

RoutineIdentifier data parameters shall be defined as in Table C.1.

Table C.1 — RoutineIdentifier data parameters

Hex	Description	Cvt	Mnemonic
E200	ExecuteSPL This value shall be used to convert a program module to an executable form.	M	DLRI_
E201	DeployLoopRoutineID This value shall be used to initiate the deployment of the previously selected ignition loop.	M	
E202 to E2FF	SafetySystemRoutineIDs This range of values shall be reserved by this document for future definition of routines implemented by safety-related systems.	M	SASRI_

C.2 RoutineControlOption data parameter definitions

RoutineControlOption data parameters shall be defined as in Tables C.2 and C.3.

Table C.2 — RoutineControlOption data parameters

RID (hex)	Description	Data byte	Scaling
0xE200	ExecuteSPL This RoutineControlOption executes the scrapping module loader program. It is started in the PCU prior to scrapping. This RoutineControlOption shall be readable and writeable.		
	SPL number This parameter contains the identification of the SPL conversion method which is to be executed.	1	8 bit unsigned character 0x00: load SPM to RAM without converting 0x01: load SPM and convert

Table C.3 — Routine identifier 0xE201 DeployLoopRoutineID

RID (hex)	Description	Data byte	Scaling
0xE201	Number of loop to be deployed This RoutineControlOption contains the “Loop Number” to be deployed. It is written to the PCU prior to the execution of the deployment routine and selects the loop that shall be ignited. When read and no valid loop ID had been written to this RoutineControlOption before the current PCU Deployment Session, the RoutineControlOption shall report the default value 0x00 (no loop). This RoutineControlOption shall be readable and writeable.		
	Loop identification #1 This parameter contains the identification of the loop that shall be deployed next by the PCU.	1	8 bit unsigned character 0x00: no loop 0x01 to 0xFF: number of loop

Bibliography

[1] SAE J1850, *Class B Data Communications Network Interface*

[2] ISO 9141-2, *Road vehicles — Diagnostic systems — Part 2: CARB requirements for interchange of digital information*

[3] ISO 14230-4, *Road vehicles — Diagnostic systems — Keyword Protocol 2000 — Part 4: Requirements for emission-related systems*

ICS 43.040.80

Price based on 49 pages