

INTERNATIONAL STANDARD

ISO
14230-3

First edition
1999-03-15

Road vehicles — Diagnostic systems — Keyword Protocol 2000 —

Part 3: Application layer

*Véhicules routiers — Systèmes de diagnostic —
Protocole «Keyword 2000» —*

Partie 3: Couche application



Contents

1 Scope	1
2 Normative references	1
3 Definitions	2
4 Conventions	2
4.1 General.....	2
4.2 Service description convention	3
4.3 Functional unit table.....	6
4.4 Service Identifier value summary table	6
4.5 Response Code value summary table	6
4.6 Response handling.....	8
5 General implementation rules	9
5.1 Parameter definitions	9
5.2 Functional and physical addressed service requests	10
5.3 Message flow examples of physical/functional addressed services.....	10
6 Diagnostic Management functional unit.....	16
6.1 StartDiagnosticSession service	16
6.2 StopDiagnosticSession service	17
6.3 SecurityAccess service.....	19
6.4 TesterPresent service.....	22
6.5 ECUReset service	23
6.6 ReadECUIdentification service.....	25
7 Data Transmission functional unit.....	26
7.1 ReadDataByLocalIdentifier service.....	27

© ISO 1999

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Organization for Standardization
Case postale 56 • CH-1211 Genève 20 • Switzerland
Internet iso@iso.ch

Printed in Switzerland

7.2 ReadDataByCommonIdentifier service	29
7.3 ReadMemoryByAddress service.....	31
7.4 DynamicallyDefineLocalIdentifier service.....	32
7.5 WriteDataByLocalIdentifier service	37
7.6 WriteDataByCommonIdentifier service	38
7.7 WriteMemoryByAddress service.....	39
7.8 SetDataRates service	41
8 Stored Data Transmission functional unit	42
8.1 ReadDiagnosticTroubleCodes service	42
8.2 ReadDiagnosticTroubleCodesByStatus service	44
8.3 ReadStatusOfDiagnosticTroubleCodes service	45
8.4 ReadFreezeFrameData service.....	46
8.5 ClearDiagnosticInformation service	51
9 InputOutput Control functional unit	52
9.1 InputOutputControlByLocalIdentifier service.....	52
9.2 InputOutputControlByCommonIdentifier service	53
10 Remote Activation Of Routine functional unit.....	55
10.1 StartRoutineByLocalIdentifier service.....	55
10.2 StartRoutineByAddress service.....	56
10.3 StopRoutineByLocalIdentifier service.....	57
10.4 StopRoutineByAddress service	59
10.5 RequestRoutineResultsByLocalIdentifier service	60
10.6 RequestRoutineResultsByAddress service.....	62
11 Upload Download functional unit	63
11.1 RequestDownload service	63
11.2 RequestUpload service	65
11.3 TransferData service	66
11.4 RequestTransferExit service	68
12 Keyword Protocol 2000 extended service.....	70
12.1 EscapeCode service	70

13 Application examples 71

13.1 Description of on-vehicle ECUs 71

13.2 Functional initialization and functional addressed communication 73

13.3 Single and multiple response and termination of communication..... 73

13.4 SecurityAccess, data transfer and modification of timing parameters..... 74

13.5 ReadDataByLocalIdentifier service with dynamicallyDefineLocalIdentifier 77

Annex A (informative) Bibliography 81

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.

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.

International Standard ISO 14230-1 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, subcommittee SC 3, *Electrical and electronic equipment*.

ISO 14230 consists of the following parts, under the general title *Road vehicles — Diagnostic systems — Keyword Protocol 2000*:

- *Part 1: Physical layer*
- *Part 2: Data link layer*
- *Part 3: Application layer*
- *Part 4: Requirements for emissions-related systems*

Annex A of this part of ISO 14230 is for information only.

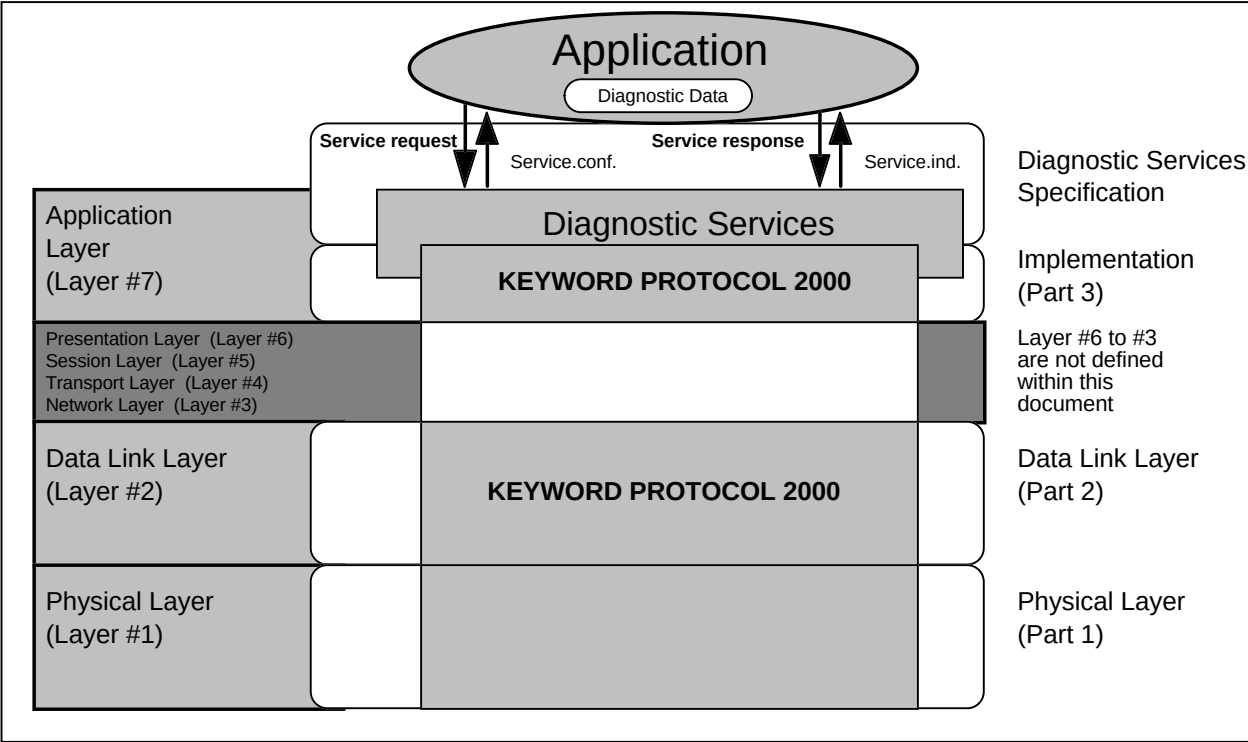
Introduction

ISO 14230 has been established in order to define common requirements for diagnostic systems implemented on a serial data link.

To achieve this, it is based on the Open Systems Interconnection (OSI) Basic Reference Model in accordance with ISO 7498 which structures communication systems into seven layers. When mapped on this model, the services used by a diagnostic tester and an Electronic Control Unit (ECU) are broken into

- diagnostic services (layer 7),
- communication services (layers 1 to 6).

See figure 1.



Example of serial data links: KWP2000, VAN, CAN, J1850, etc.

Figure 1 — Mapping of Diagnostic Services and Keyword Protocol 2000 on OSI Model

Road vehicles — Diagnostic systems — Keyword Protocol 2000 —

Part 3: Application layer

1 Scope

This part of ISO 14230 specifies the requirements for the Keyword Protocol 2000 data link on which one or several on-vehicle Electronic Control Units are connected to an off-board tester in order to perform diagnostic functions.

This part of ISO 14230 specifies the requirements of the implementation of the Diagnostic Services specified in ISO 14229, including

- byte-encoding and hexadecimal values for the service identifiers;
- byte-encoding for the parameters of the diagnostic service requests and responses;
- hexadecimal values for the standard parameters.

The vehicle environment to which this part of ISO 14230 applies may consist of a single tester that may be temporarily connected to the on-vehicle diagnostic data link and several on-vehicle Electronic Control Units connected directly or indirectly (see figure 2).

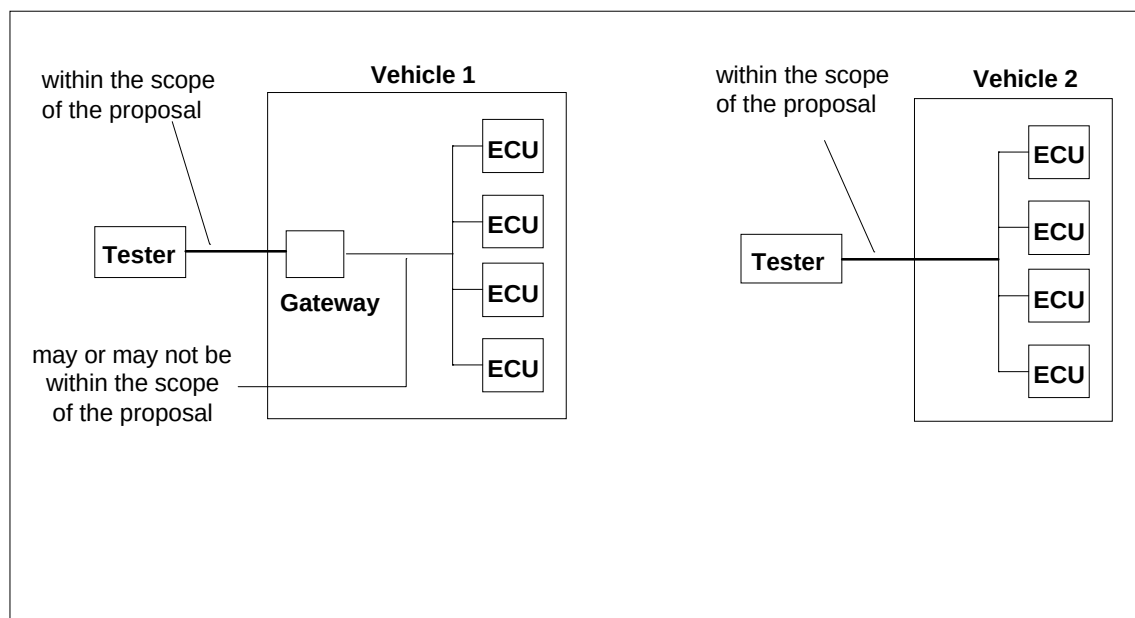


Figure 2 — Vehicle diagnostic architecture

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this document. All standards are subject to revision, and parties to agreement based on this document are encouraged

to investigate the possibility of applying the most recent editions of the standards listed below. Members of ISO maintain registers of currently valid International Standards.

ISO 14229:—¹⁾, *Road vehicles — Diagnostic systems — Diagnostic services specification*.

ISO 14230-2:—¹⁾, *Road vehicles — Diagnostic systems — Keyword Protocol 2000 — Part 2 : Data link layer*.

SAE J 1930: 1995, *Electrical/electronic systems diagnostic — Terms, definitions, abbreviations and acronyms*.

SAE J 1979: 1997, *E/E diagnostic test modes— Terms, definitions, abbreviations and acronyms*.

3 Definitions

For the purposes of this part of ISO 14230, the definitions given in ISO 14229 and SAE J 1930 apply.

4 Conventions

4.1 General

4.1.1 This part of ISO 14230 is guided by the OSI service conventions (CVT; see ISO 8509) to the extent that they are applicable to the diagnostic services. These conventions define the interactions between the service use and the service provider by the supplier through service primitives which themselves may convey parameters.

4.1.2 Table 1 indicates the different ranges of service identifier values, which are defined in SAE J 1979, ISO 14230 or by the vehicle manufacturer.

Table 1 — Service Identifier value convention table

Service Identifier Hex Value	Service type ¹⁾ (bit 6)	Where defined
00 - 0F	Request	SAE J 1979
10 - 1F 20 - 2F 30 - 3E	Request (bit 6 = 0)	ISO 14230-3
3F	Not applicable	reserved
40 - 4F	Response	SAE J1979
50 - 5F 60 - 6F 70 - 7E	Positive Response to Services (\$10 - \$3E) (bit 6 = 1)	ISO 14230-3
7F	Negative Response	
80	Request 'ESC' - Code	
81 - 8F	Request (bit 6 = 0)	ISO 14230-2
90 - 9F	Request (bit 6 = 0)	reserved for future exp. as needed
A0 - BF	Request (bit 6 = 0)	defined by vehicle manufacturer
C0	Positive Resp. 'ESC' - Code	ISO 14230-3
C1 - CF	Positive Response (bit 6 = 1)	ISO 14230-2
D0 - DF	Positive Response (bit 6 = 1)	reserved for future exp. as needed
E0 - FF	Positive Response (bit 6 = 1)	defined by vehicle manufacturer

¹⁾ There is a one-to-one correspondence between request messages and positive response messages, with "bit 6" of the service identifier hex value indicating the service type.

¹⁾ To be published.

4.1.3 The table content consists of the following:

- under the **<Service Name> Request Message** are listed the parameters specific to the service request/indication;
- under the **<Service Name> Positive Response Message** are listed the parameters specific to the service response/confirmation in case the requested service was successful;
- under the **<Service Name> Negative Response Message** are listed the parameters specific to the service response/confirmation in case the requested service has failed or could not be completed in time.

4.1.4 For a given primitive, the presence of each parameter is described by one of the following values:

- M: mandatory;
- U: user option; the parameter may or may not be supplied, depending on dynamic usage by the user;
- C: conditional; the presence of the parameter depends upon other parameters within the service;
- S: mandatory (unless specified otherwise) selection of a parameter from a parameter list.

4.2 Service description convention

This clause defines the layout used to describe the diagnostic services. It includes

- Parameter Definition;
- Message Data Bytes;
- Message Description;
- Message Flow Example.

4.2.1 Parameter definition

This section defines the use and the values of parameters used by the service.

4.2.2 Message data bytes

The definition of each message includes a table which lists the parameters of its primitives: request/indication ("Req/Ind"), response/confirmation ("Rsp/Cnf") for positive or negative result. All have the same structure. Table 2 describes the request message, table 3 the positive response message and table 4 the negative response message.

A positive response message shall be given by the server if it can carry out all the operations requested. It shall otherwise give a negative response.

The response messages are listed in separate tables because the list of parameters differs between positive and negative response messages.

Table 2 — Request message

Type	Parameter Name	CVT ¹⁾	Hex Value	Mnemonic
Header Bytes ²⁾	Format Byte Target Byte Source Byte Length Byte	M C ³⁾ C ³⁾ C ⁴⁾	xx xx xx xx	FMT TGT SRC LEN
<ServiceId>	<Service Name> Request Service Identifier	M	xx	SN
<Parameter Type> : <Parameter Type>	<List of parameters> = [<Parameter Name> : <Parameter Name>]	C ⁵⁾	xx=[xx : xx]	PN
CS ²⁾	Checksum Byte	M	xx	CS

1) See 4.1.4

2) Defined in ISO 14230-2.

3) The header bytes "Target" and "Source" depend on the content of the "Format Byte" which is specified in ISO 14230-2 (KWP 2000 Part 2: Data Link Layer) document. Both either exist or do not exist in the header of each message.

4) The header byte "Length" depends on the content of the "Format Byte" which is specified in ISO DIS 14230-2.

5) These parameters may be either mandatory (M) or user optional (U), depending on the individual message.

Table 3 — Positive response message

Type	Parameter Name	CVT ¹⁾	Hex Value	Mnemonic
Header Bytes ²⁾	Format Byte Target Byte Source Byte Length Byte	M C ³⁾ C ³⁾ C ⁴⁾	xx xx xx xx	FMT TGT SRC LEN
<ServiceId>	<Service Name> Positive Response Service Identifier	M	xx	SNPR
<Parameter Type> : <Parameter Type>	<List of parameters> = [<Parameter Name> : <Parameter Name>]	C ⁵⁾	xx=[xx : xx]	PN
CS ²⁾	Checksum Byte	M	xx	CS

1) See 4.1.4

2) Defined in ISO 14230-2.

3) The header bytes "Target" and "Source" depend on the content of the "Format Byte" which is specified in ISO 14230-2 (KWP 2000 Part 2: Data Link Layer) document. Both either exist or do not exist in the header of each message.

4) The header byte "Length" depends on the content of the "Format Byte" which is specified in ISO DIS 14230-2.

5) These parameters may be either mandatory (M) or user optional (U), depending on the individual message.

Table 4 — Negative response message

Type	Parameter Name	CVT ¹⁾	Hex Value	Mnemonic
Header Bytes ²⁾	Format Byte	M	xx	FMT
	Target Byte	C ³⁾	xx	TGT
	Source Byte	C ³⁾	xx	SRC
	Length Byte	C ⁴⁾	xx	LEN
<ServiceId>	NegativeResponse Service Identifier	M	xx	NACK
<ServiceId>	<Service Name> Request Service Identifier	M	xx	SN
<Parameter Type>	ResponseCode=[KWP2000ResponseCode, ManufacturerSpecific]	M	xx=[00-7F, 80-FF]	RC
CS ²⁾	Checksum Byte	M	xx	CS

1) See 4.1.4
2) Defined in ISO 14230-2.
3) The header bytes "Target" and "Source" depend on the content of the "Format Byte" which is specified in ISO 14230-2 (KWP 2000 Part 2: Data Link Layer) document. Both either exist or do not exist in the header of each message.
4) The header byte "Length" depends on the content of the "Format Byte" which is specified in ISO DIS 14230-2..

4.2.3 Message description

This section "Message description" provides a description of the actions performed by the client and the server which are specific to the KWP 2000 Protocol (see ISO 14230-2).

The response condition is service specific and defined separately for each service.

4.2.4 Message flow examples

This section provides message flow descriptions presented in a table format (see table 5). The table consists of three columns:

- column 1: includes the relevant inter-message timing which is specified in the document ISO/DIS 14230-2. The message shall be started within the relevant inter-message timing;
- column 2: includes all requests send by the client to the server;
- column 3: includes all responses send by the server to the client.

Sending of a message shall start during the period of time between appropriate messages.

Time relates to the table in a top to bottom sequence. The reading entry of the message flow table always starts with the first item in the time column "P3" (1st column) followed by a request message of the client (2nd column). The next entry is the timing parameter "P2" (1st column) for the server to send the positive or negative response message (3rd column).

For simplification all messages are described without any identifiers and/or data values. Details of messages are always specified in the section: Message data bytes.

The message flow example above is not documented for each service. Only services, which call for more detailed message flow description shall have their own message flow section.

Table 5 — Message flow example of physical addressed service

time	client (tester)	server (ECU)
P3	<Service Name> Request[...]	
P2		<Service Name> PositiveResponse[...]

4.3 Functional unit table

The intention of specifying functional unit tables is to group similar Keyword Protocol (KWP) 2000 services into a functional unit. The definition of each functional unit includes a table such as table 6 which lists its services.

Table 6 — Keyword Protocol 2000 functional units

Functional Unit	Description
Diagnostic Management	This functional unit includes Keyword Protocol 2000 services which are used to realise diagnostic management functions between the client (tester) and the server (ECU).
Data Transmission	This functional unit includes Keyword Protocol 2000 services which are used to realise data transmission functions between the client (tester) and the server (ECU).
Stored Data Transmission	This functional unit includes Keyword Protocol 2000 services which are used to realise stored data transmission functions between the client (tester) and the server (ECU).
Input / Output Control	This functional unit includes Keyword Protocol 2000 services which are used to realise input / output control functions between the client (tester) and the server (ECU).
Remote Activation of Routine	This functional unit includes Keyword Protocol 2000 services which are used to realise remote activation of routine functions between the client (tester) and the server (ECU).
Upload / Download	This functional unit includes Keyword Protocol 2000 services which are used to realise upload / download functions between the client (tester) and the server (ECU).

4.4 Service Identifier value summary table

The left column of table 7 lists all services of the Diagnostic Services Specification, the middle column assigns the KWP 2000 Implementation "Request Hex Value" and the right column assigns the KWP 2000 Implementation "Positive Response Hex Value". The positive response service identifier values are built from the request service identifier values by setting "bit 6 = 1".

4.5 Response Code value summary table

Table 8 lists and assigns hex values for all response codes used in KWP 2000. The definition of each response code is described in ISO 14229.

Table 7 — Service Identifier value summary table

Diagnostic Service Name	KWP 2000 Implementation	
	Request Hex Value	Response Hex Value
StartDiagnosticSession	10	50
ECUReset	11	51
ReadFreezeFrameData	12	52
ReadDiagnosticTroubleCodes	13	53
ClearDiagnosticInformation	14	54
ReadStatusOfDiagnosticTroubleCodes	17	57
ReadDiagnosticTroubleCodesByStatus	18	58
ReadECUIdentification	1A	5A
StopDiagnosticSession	20	60
ReadDataByLocalIdentifier	21	61
ReadDataByCommonIdentifier	22	62
ReadMemoryByAddress	23	63
SetDataRates	26	66
SecurityAccess	27	67
DynamicallyDefineLocalIdentifier	2C	6C
WriteDataByCommonIdentifier	2E	6E
InputOutputControlByCommonIdentifier	2F	6F
InputOutputControlByLocalIdentifier	30	70
StartRoutineByLocalIdentifier	31	71
StopRoutineByLocalIdentifier	32	72
RequestRoutineResultsByLocalIdentifier	33	73
RequestDownload	34	74
RequestUpload	35	75
TransferData	36	76
RequestTransferExit	37	77
StartRoutineByAddress	38	78
StopRoutineByAddress	39	79
RequestRoutineResultsByAddress	3A	7A
WriteDataByLocalIdentifier	3B	7B
WriteMemoryByAddress	3D	7D
TesterPresent	3E	7E
EscCode ¹⁾	80	C0

1) Does not form part of diagnostic services specification but only of KWP Protocol 2000.

Table 8 —Response Code value summary table

Hex Value	Response Code
10	GeneralReject
11	ServiceNotSupported
12	SubFunctionNotSupported-invalidFormat
21	Busy-RepeatRequest
22	ConditionsNotCorrect or requestSequenceError
23	RoutineNotComplete
31	RequestOutOfRange
33	SecurityAccessDenied
35	InvalidKey
36	ExceedNumberOfAttempts
37	RequiredTimeDelayNotExpired
40	DownloadNotAccepted
41	ImproperDownloadType
42	Can'tDownloadToSpecifiedAddress
43	Can'tDownloadNumberOfBytesRequested
50	UploadNotAccepted
51	ImproperUploadType
52	Can'tUploadFromSpecifiedAddress
53	Can'tUploadNumberOfBytesRequested
71	TransferSuspended
72	TransferAborted
74	IllegalAddressInBlockTransfer
75	IllegalByteCountInBlockTransfer
76	IllegalBlockTransferType
77	BlockTransferDataChecksumError
78	ReqCorrectlyRcvd-RspPending
79	IncorrectByteCountDuringBlockTransfer
80 - FF	ManufacturerSpecificCodes
1) RequestCorrectlyReceived-ResponsePending	

4.6 Response handling

Figure 3 specifies the server behaviour on a client request message. It shows the logic as specified in the description of the response codes and to be implemented in the server and client as appropriate.

The use of (a) negative response message(s) by the server shall be in case the server can not respond with a positive response message on a client (tester) request message. In such case the server shall send one of the response codes listed as specified in figure 3.

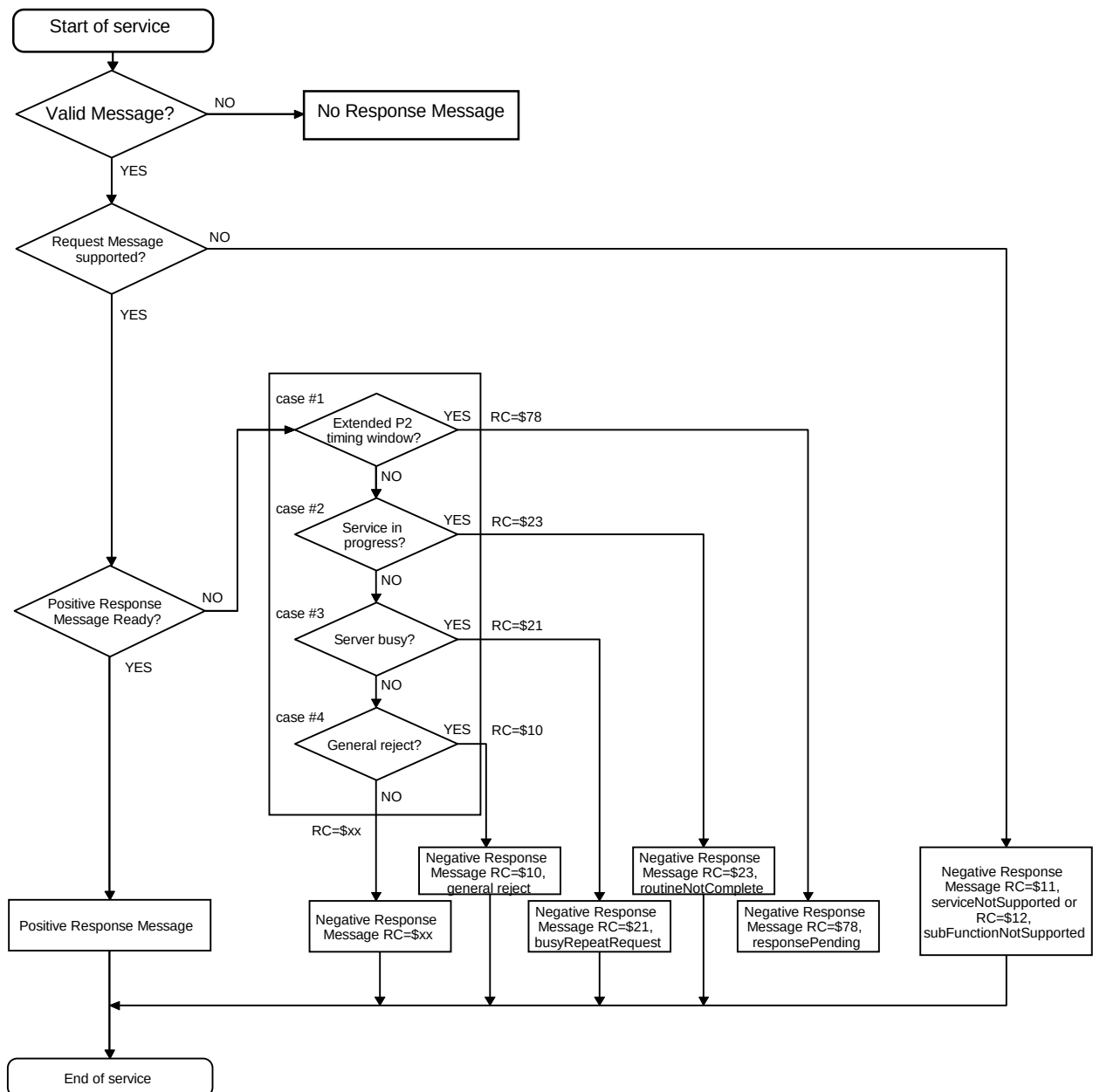


Figure 3 — Server positive and negative response message behaviour

5 General implementation rules

5.1 Parameter definitions

The following rules in regard to parameter definitions shall apply:

- clauses 6 to 12 define the services of each functional unit. In these clauses, the service structure makes reference to parameters, in order to describe the allowable values for such parameters. The parameters of general purpose are defined in ISO 14229. Parameters which are specific to a functional unit are described in the corresponding clause;
- this part of ISO 14230 lists and defines response codes and values. Negative response codes are specified in 4.4. Other response codes may be reserved either for future definition by this part of ISO 14230 or for the system designer's specific use;
- this part of ISO 14230 specifies the parameters which shall be used within each KWP 2000 Service;

- the sequence of parameters within a service shall not be changed during an implementation;
- this part of ISO 14230 specifies the parameter MemoryAddress based on a three (3) byte address (High Byte, Middle Byte and Low Byte). Additional bytes of specifying the MemoryAddress (e.g. memory type identifier, larger address range) may be implemented and is the responsibility of the vehicle manufacturer. This applies to all services which use the MemoryAddress parameter.

5.2 Functional and physical addressed service requests

Two different addressing methods are specified in KWP 2000 to send a service request to a server(s).

Functional addressing with a three byte header is used by the client if it does not know the physical address of the server that shall respond to a service request or if more than one server can respond to the request.

Functional addressing with a one byte header is not possible.

Physical addressing with a three byte header shall always be a dedicated message to one server. The header of the service request message indicates (target address) which server shall respond to the service request message.

Physical addressing with a one byte header is possible.

Only those server(s) which are initialized and in a diagnostic session which support the service request message shall send a response message.

Functional and Physical addressing methods are specified in detail in ISO 14230-2.

The data link shall always be initialized prior to sending any of the KWP 2000 services.

5.3 Message flow examples of physical/functional addressed services

5.3.1 Physical addressed services

5.3.1.1 Physical addressed service with positive/negative response message

Table 9 shows a typical service request followed by a positive response message and a service request followed by a negative response message.

Table 9 — Message flow example of physical addressed service

time	client (tester)	server (ECU)
P3 P2	<Service Name> Request[...]	<Service Name> PositiveResponse[...]
P3 P2 P3 P2	<Service Name> Request[...] <Service Name> Request[...]	<Service Name> NegativeResponse[RC] <Service Name> PositiveResponse[...]

5.3.1.2 Physical addressed service with periodic transmissions

Table 10 shows a message flow which describes a physical addressed service request with multiple positive response messages.

Table 10 — Message Flow example of physical addressed service with periodic transmissions

time	client (tester)	server (ECU)
P3	<ReadDataByLocalIdentifier>	
P2	Request[RLI,TXM]	<ReadDataByLocalIdentifier >
P2		PositiveResponse#1[RLI, ...]
P2		:
P3*	<ReadDataByLocalIdentifier>	<ReadDataByLocalIdentifier >
P2	<ReadDataByCommonIdentifier>	PositiveResponse#k[RLI, ...]
P2	Req.[RCI,TXM]	
P2		<ReadDataByCommonIdentifier> PosResp#1[RCI, ...]
P3 ¹⁹		:
P2	<ReadMemoryByAddress>	<ReadDataByCommonIdentifier> PosResp#k[RCI, ...]
P2	Request[MA,MS,TXM]	
P2		
P3*		<ReadMemoryByAddress> PosResp#1[RECVAl]
P2		:
P3	<Any Other Service Name> Request[...]	<ReadMemoryByAddress> PosResp#1[RECVAl]
P2		
	<ReadDataByLocalIdentifier> Request[RLI]	< Any Other Service Name > PositiveResponse[...]
		<ReadDataByLocalIdentifier> PositiveResponse[RLI,]

1) The values of the "P3" timing parameters shall be less than the value of "P2min" in order to allow the client (tester) to send a new request message.

The message flow in table 10 describes a physical addressed service request ReadDataByLocal/Common-Identifier or ReadMemoryByAddress with the periodic transmission mode enabled. The request message is followed by multiple positive response messages from the physically addressed server. The periodically transmitted response messages are terminated by the client with any request message not including the optional TransmissionMode parameter. This request message shall be send to the server within the "P3*" timing window.

5.3.1.3 Physical addressed service and negative response message(s) with "RoutineNotComplete" and "Busy-RepeatRequest"

Table 11 shows a message flow which describes a physical addressed service request followed by a negative response message with the response code set to "RoutineNotComplete".

Table 11 — Physical addressing and negative response with "RoutineNotComplete" and "Busy-RepeatRequest"

time	client (tester)	server (ECU)
P3	<Service Name> Request [...] ¹⁾	<Service Name> NegativeResponse[RoutineNotComplete]
P2		
P3	<Service Name> Request[...]	<Service Name> NegativeResponse[Busy-RepeatRequest]
P2		:
	:	
P3	<Service Name> Request[...]	<Service Name> NegativeResponse[Busy-RepeatRequest]
P2		<Service Name> PositiveResponse[...]
P3	<Service Name> Request [...] ²⁾	OR
P2		Service Name> NegativeResponse[RC != Busy-RepeatRequest]
P2		

1) server has started routine.

2) server has completed routine.

The message flow example in table 11 is based on a request message from the client which cause the server to respond with a negative response message including the negative response code "RoutineNotComplete". This response code indicates that the request message was properly received by the server and the routine/task/function (initiated by the request message) is in process, but not yet completed. If the client repeats or sends another request message the server shall "not reinitiate the task" (in case of the same request message) if the initial task has not been completed. The server shall send another negative response message with the response code "Busy-RepeatRequest". The negative response message shall be sent on each request message as long as the server has not yet completed the initial routine/task/function. If the server has finished the routine/task/function it shall respond with either a positive response message or a negative response message with a response code not equal to "Busy-RepeatRequest".

The communication timing is not affected.

Applications which may require this message flow are as follows:

- ClearDiagnosticInformation;
- execution of routines;
- SecurityAccess.

5.3.1.4 Implementation example of "server can not send a positive response within required timing"

5.3.1.4.1 Example of physical addressed service and negative response message with "ReqCorrectlyRcvd-RspPending within Normal or Extended Timing"

Table 12 shows a message flow which describes a physical addressed service request followed by a negative response message with the response code set to "RequestCorrectlyReceived-ResponsePending".

Table 12 — Example of physical addressing and negative response with "ReqCorrectlyRcvd-RspPending"

time	client (ester)	server (ECU)
P3	<Service Name> Request[...]	<Service Name> NegRsp#1[ReqCorrectlyRcvd-RspPending] : <Service Name> NegRsp#n[ReqCorrectlyRcvd-RspPending] <Service Name> PositiveResponse[...] <Service Name> PositiveResponse[...]
P2		
P2 ¹⁾		
P2 ¹⁾		
P3		
P2	<Service Name> Request[...]	
1) P2min to P3max (refer to ISO 14230-2).		

The message flow example in table 12 is based on a request message from the client which causes the server to respond with one or multiple negative response message(s) including the negative response code "RequestCorrectlyReceived-ResponsePending". This response code indicates that the request message was received correctly, and that any parameters in the request message were valid, but the action to be performed may not be completed yet. This response code may be used to indicate that the request message was properly received and does not need to be retransmitted, but the server is not yet ready to receive another request.

The negative response message may be sent once or multiple times within a service if required by the server.

During the period of (a) negative response message(s) the TesterPresent service shall be disabled in the client.

This response code shall only be used in a negative response message if the server will not be able to receive further request messages from the client within the P3 timing window. This may be the case if the server does data processing or executes a routine which does not allow any attention to serial communication.

NOTE — The communication timing method is specified in ISO 14230-2:—, clause 5.

Applications which may require above message flow are as follows:

- ClearDiagnosticInformation;
- execution of routines;
- TransferData request messages including up to 255 data bytes;
- Flash EEPROM or EEPROM memory programming.

5.3.1.4.2 Implementation of "Server can not send a positive response within Default Timing"

Table 13 shows a message flow which describes a physical addressed service request followed by a positive response message sent with previously modified timing.

Table 13 — Physical addressing and modified timing with AccessTimingParameter service

Time 1)	Client (tester)	Server (ECU)
P3	StartDiagnosticSession.Request[...]	
P2		StartDiagnosticSession.PosRsp[...]
P3	AccessTimingParameter.Request[ReadLimits]	
P2		AccessTimingParameter.PosRsp[ReadLimits, P2-P4]
P3	AccessTimingParameter.Request[setValues, P2-P4]	
P2	{ e.g. P2max = \$F2 (18850 ms) } { Modified Timing is active! }	AccessTimingParameter.PosRsp[setValues] ²⁾
P3	<Service Name> Request[...]	
P2		<Service Name> PositiveResponse[...]
P3	<StopDiagnosticSession> Request[]	
P2	OR	<StopDiagnosticSession> PositiveResponse[]
P3	<StartDiagnosticSession> Request[...]	OR
P2		<StartDiagnosticSession> PositiveResponse[...] ³⁾
	{ Default Timing is active! }	
P3	<Service Name> Request[...]	
P2		<Service Name> PositiveResponse[...]
1) New timing parameter values are active. 2) Modified Timing is active. 3) Default Timing is active.		

The message flow example in table 13 describes the method of modifying the timing parameter values in the server and the client by using the AccessTimingParameter service as specified in ISO 14230-2.

This method shall be used in case a server does not support a negative response message with the response code (\$78) "RequestCorrectlyReceived-ResponsePending".

5.3.2 Functional addressed services

5.3.2.1 Functional addressed service with positive/negative response message

Table 14 shows a message flow example which describes a functional addressed service request followed by response messages of multiple servers (ECU#1, ECU#2 ... ECU#n-1, ECU#n).

Table 14 — Message flow example of functional addressed service

Time	client (tester)	server (ECU)
P3	<Service Name> Request[...]	
P2		<Service Name> PositiveResponse[...] {ECU#1}
P2		<Service Name> NegativeResponse[...] {ECU#2}
:		:
P2		<Service Name> PositiveResponse[...] {ECU#n-1}
P2		<Service Name> PositiveResponse[...] {ECU#n}
P3	<Service Name> Other Request[...]	
P2		<Service Name> Other PositiveResponse[...] {ECU}

The message flow example in table 14 is based on a functional request message send by the client. An unknown number of servers has been initialized previously (e.g. fast initialization by wake up pattern) which send positive and negative response messages.

5.3.2.2 Functional addressed service and negative response messages with "RoutineNotComplete" and "Busy-RepeatRequest"

Table 15 shows a message flow which describes a functional addressed service request followed by a negative response message with the response code set to "RoutineNotComplete" from one server. All other servers send positive response messages.

Table 15 — Functional addressing and negative response with "Busy-RepeatRequest"

Time	client (tester)	server (ECU)
P3	<Service Name> Request[...] ^{1), 2)}	
P2		<Service Name> NegRsp[RoutineNotComplete] {ECU#1}
P2		<Service Name> PositiveResponse[...] {ECU#2}
P2		<Service Name> PositiveResponse[...] {ECU#3}
P3	<Service Name> Request[...] ¹⁾ { Functional }	
P2		<Service Name> NegRsp[Busy-RepeatRequest] {ECU#1}
P2		<Service Name> PositiveResponse[...] {ECU#2}
P2		<Service Name> PositiveResponse[...] {ECU#3}
P3	<Service Name> Request[...] ^{1), 3)} { Functional }	
P2		<Service Name> PositiveResponse[...] {ECU#1}
P2		<Service Name> PositiveResponse[...] {ECU#2}
P2		<Service Name> PositiveResponse[...] {ECU#3}
1) Functional. 2) Server has started routine. 3) Server has completed routine.		

The message flow example in table 15 is based on a functional request message from the client which cause one server (ECU#1) to respond with a negative response message including the negative response code "RoutineNotComplete" The other servers (ECU#2, ECU#3) send a positive response message.

The response code indicates that the request message was properly received by the server and the routine/task/function (initiated by the request message) is in process, but not yet completed. If the client repeats or sends another functional request message the server shall "not reinitiate the task" (in case of the same request message) if the initial task has not been completed. The server shall send another negative response message with the response code "Busy-RepeatRequest". The negative response message shall be sent on each functional

request message as long as the server has not yet completed the initial routine/task/function. If the server (ECU#1) has finished the routine/task/function it shall respond with either a positive response message or a negative response message with a response code not equal to "Busy-RepeatRequest".

The communication timing is not affected.

Applications which may require above message flow are as follows:

- ClearDiagnosticInformation;
- execution of routines;
- SecurityAccess.

5.3.2.3 Implementation example of functional addressed service and negative response message with "RequestCorrectlyReceived-ResponsePending"

Table 16 shows a message flow example which describes a functional addressed request message followed by a negative response message with the response code set to "RequestCorrectlyReceived-ResponsePending" from one server (ECU#1) and positive response messages from the other servers (ECU#2, ECU#3).

Table 16 — Example of functional addressing and negative response with "ReqCorrectlyRcvd-RspPending"

Time	Client (tester)	Server (ECU)
P3	<Service Name> Req[...] ¹⁾	
P2		<Service Name> NegRsp#1[ReqCorrectlyRcvd-RspPendg] {ECU#1}
P2 ²⁾		<Service Name> PositiveResponse[...] {ECU#2}
P2 ²⁾		<Service Name> PositiveResponse[...] {ECU#3}
P2 ²⁾		<Service Name> NegRsp#2[ReqCorrectlyRcvd-RspPendg] {ECU#1}
P2 ²⁾		:
P2 ²⁾	<Service Name> Req[...] { Functional }	<Service Name> NegRsp#n[ReqCorrectlyRcvd-RspPendg] {ECU#1}
P2 ²⁾		<Service Name> PositiveResponse[...] {ECU#1}
P3		<Service Name> PositiveResponse[...] {ECU#1}
P2		<Service Name> PositiveResponse[...] {ECU#2}
P2		<Service Name> PositiveResponse[...] {ECU#3}
P2		<Service Name> PositiveResponse[...] {ECU#3}

1) Functional.
2) P2min to P3max (refer to ISO 14230 KWP-2).

The message flow example in table 16 is based on a request message from the client which cause the server to respond with one or multiple negative response message including the negative response code "RequestCorrectlyReceived-ResponsePending". This response code indicates that the request message was received correctly, and that any parameters in the request message were valid, but the action to be performed may not be completed yet. This response code can be used to indicate that the request message was properly received and does not need to be retransmitted, but the server is not yet ready to receive another request.

The negative response message may be sent once or multiple times within a service if required by the server.

During the period of (a) negative response message(s) the TesterPresent service shall be disabled in the client.

This response code shall only be used in a negative response message if the server will not be able to receive further request messages from the client within the P3 timing window. This may be the case if the server does data processing or executes a routine which does not allow any attention to serial communication.

The communication timing method is specified in ISO 14230-2:—, clause 5.

Applications which may require above message flow are as follows:

- ClearDiagnosticInformation;
- execution of routines;
- TransferData request messages including up to 255 data bytes;
- Flash EEPROM or EEPROM memory programming.

6 Diagnostic Management functional unit

The services provided by this functional unit are described in table 17.

Table 17 — Diagnostic Management functional unit

Service name	Description
StartDiagnosticSession	The client requests to start a diagnostic session with a server(s).
StopDiagnosticSession	The client requests to stop the current diagnostic session.
SecurityAccess	The client requests to unlock a secured server.
TesterPresent	The client indicates to the server(s) that it is still present.
ECUReset	The client forces the server(s) to perform a reset.
ReadECUIdentification	The client requests identification data from the server(s).

6.1 StartDiagnosticSession service

6.1.1 Parameter definition

The parameter **DiagnosticMode** is used by the StartDiagnosticSession service to select the specific behaviour of the server(s). No values are defined in this part of ISO 14230, but table 18 describes the range of values.

Table 18 — Definition of DiagnosticMode Values

Hex	Description
00 - 7F	ReservedByDocument This range of values is reserved by this standard for future definition.
80 - FF	ManufacturerSpecific This range of values is reserved for vehicle-manufacturer-specific use.

6.1.2 Message data bytes

Tables 19 to 21 describe the three possible messages for StartDiagnosticSession service.

Table 19 — StartDiagnosticSession Request Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	StartDiagnosticSession Request Service Id	M	10	SDS
2	DiagnosticMode=[ReservedByDocument, ManufacturerSpecific]	M	xx=[00-7F, 80-FF]	DIAGMODE

Table 20 — StartDiagnosticSession Positive Response Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	StartDiagnosticSession Positive Response Service Id	S	50	SDSPR
2	DiagnosticMode=[ReservedByDocument, ManufacturerSpecific]	U	xx=[00-7F, 80-FF]	DIAGMODE

Table 21 — StartDiagnosticSession Negative Response Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	NegativeResponse Service Id	S	7F	NACK
2	StartDiagnosticSession Request Service Id	S	10	SDS
3	ResponseCode=[KWP2000ResponseCode, ManufacturerSpecific]	M	xx=[00-7F, 80-FF]	RC

6.1.3 Message description

The messages specified in 6.1.2 are used to enable different diagnostic modes in the server. Possible diagnostic modes are not defined in this part of ISO 14230.

A diagnostic session shall only be started if communication has been established between the client and the server. For more detail on how to start communication, refer to ISO 14230-2.

If no diagnostic session has been requested by the client after StartCommunication a default session shall be automatically enabled in the server. The default session shall support at least the following services:

- "StopCommunication service" (see ISO 14230-2);
- "TesterPresent service" (see ISO 14230-3).

The default timing parameters in the server shall be active while the default session is running.

If a diagnostic session has been requested by the client which is already running the server shall send a positive response message.

Whenever a new diagnostic session is requested by the client, the server shall first send a StartDiagnosticSession positive response message before the new session becomes active in the server. If the server sends a negative response message with the StartDiagnosticSession request service identifier the current session shall continue. There shall be only one session active at a time. A diagnostic session enables a specific set of KWP 2000 services which shall be defined by the vehicle manufacturer. A session may enable vehicle-manufacturer-specific services which are not part of this part of ISO 14230.

6.1.4 Message flow example

See message flow example of Physical Addressed Service in 5.3.1.1 and Functional Addressed Service in 5.3.2.1.

6.2 StopDiagnosticSession service

6.2.1 Parameter Definition

This service shall not use any parameter definition.

6.2.2 Message Data Bytes

Tables 22 to 24 describe the three possible messages for StopDiagnosticSession service.

Table 22 — StopDiagnosticSession Request Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	StopDiagnosticSession Request Service Id	M	20	SPDS

Table 23 — StopDiagnosticSession Positive Response Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	StopDiagnosticSession Positive Response Service Id	S	60	SPDSPR

Table 24 — StopDiagnosticSession Negative Response Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	NegativeResponse Service Id	S	7F	NACK
2	StopDiagnosticSession Request Service Id	S	20	SPDS
3	ResponseCode=[KWP2000ResponseCode, ManufacturerSpecific]	M	xx=[00-7F, 80-FF]	RC

6.2.3 Message description

The messages specified in 6.2.2 are used to disable the current diagnostic mode in the server.

The following implementation rules shall be followed:

- a diagnostic session shall only be stopped if communication has been established between the client and the server and a diagnostic session is running;
- if no diagnostic session is running the default session is active (see StartDiagnosticSession). The default session cannot be disabled by a StopDiagnosticSession service;
- if the server has sent a StopDiagnosticSession positive response message the default timing parameters in the server shall be valid while the default session is running;
- if the server has sent a StopDiagnosticSession positive response message it shall have stopped the current diagnostic session, that is, perform the necessary action to return to a state in which it is able to restore its normal operating conditions. Restoring the normal operating conditions of the server may include the reset of all the actuators controlled if they have been activated by the client during the diagnostic session being stopped and resuming all normal algorithms of the server;
- if the server has sent a StopDiagnosticSession positive response message it shall have re-locked the server if it was unlocked by the client during the diagnostic session;
- if a StopDiagnosticSession has been requested by the client and the default session is already running the server shall send a positive response message and immediately reset all timing parameters;
- the client shall send a StopDiagnosticSession request message before disabling communication via a StopCommunication service but only if a StartDiagnosticSession request message has been sent previously;
- if the server sends a negative response message with the StopDiagnosticSession request service identifier the active session shall be continued;

- a StopDiagnosticSession service shall also be used to disable vehicle-manufacturer-specific diagnostic sessions.

6.2.4 Message flow example

See message flow example of Physical Addressed Service in 5.3.1.1 and Functional Addressed Service in 5.3.2.1.

6.3 SecurityAccess service

6.3.1 Parameter definition

The parameter *AccessMode* is used in the SecurityAccess service. It indicates to the server the step in progress for this service, the level of security the client wants to access and the format of seed and key. Values are defined in table 25 for RequestSeed and SendKey.

Table 25 — Definition of AccessMode values

Hex	Description
01	RequestSeed RequestSeed with the level of security defined by the vehicle manufacturer.
02	SendKey SendKey with the level of security defined by the vehicle manufacturer.
03, 05, 07 - 7F	RequestSeed RequestSeed with different levels of security defined by the vehicle manufacturer.
04, 06, 08 - 80	SendKey SendKey with different levels of security defined by the vehicle manufacturer.
81 - FF	ManufacturerSpecific This range of values is reserved for vehicle-manufacturer-specific use.

The values of the parameter **Seed** are not defined in this standard except for the value "\$00 00" which indicates to the client that the server is not locked.

The values of the parameter **Key** are not defined in this part of ISO 14230.

The parameter **SecurityAccessStatus** may be used to receive the status of the server security system. The value is defined in table 26.

Table 26 — Definition of SecurityAccessStatus value

Hex	Description
34	SecurityAccessAllowed

6.3.2 Message data bytes

Tables 27 to 32 describe the different messages for SecurityAccess service.

Table 27 — SecurityAccess Request#1 Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	SecurityAccess Request#1 Service Id	M	27	SA#1
2	AccessMode=[RequestSeed: ManufacturerSpecific	M	xx=[¹⁾ 01,03, :7F,81-FF]	ACCMODE

1) Shall be an odd number greater than \$01 if additional levels of security are supported,(\$01 default),

Table 28 — SecurityAccess Positive Response#1 Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	SecurityAccess Positive Response#1 Service Id	S	67	SA#1PR
2	AccessMode=[RequestSeed: ManufacturerSpecific]	M	xx=[²⁾ 01,03, :7F, 81-FF]	ACCMODE
3 : n - 1	seed#1 : seed#m	C ¹⁾ : C ¹⁾	xx : xx	SEED
n	SecurityAccessStatus=[SecurityAccessAllowed]	U	34	SACCSTAT

1) The condition is AccessMode set to RequestSeed.
2) Shall be an odd number greater than \$01 if additional levels of security are supported,(\$01 default).

Table 29 — SecurityAccess Negative Response#1 Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	NegativeResponse#1 Service Id	S	7F	NACK
2	SecurityAccess Request Service Id	S	27	SA
3	ResponseCode=[KWP2000ResponseCode, ManufacturerSpecific]	M	xx=[²⁾ 00-7F, 80-FF]	RC

Table 30 — SecurityAccess Request#2 Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	SecurityAccess Request#2 Service Id	M	27	SA#2
2	AccessMode=[SendKey: shall be an even number one greater than AccessMode in request#1 if additional levels of security are supported, (\$02 default) ManufacturerSpecific]	M	xx=[02, 04, : 80, 82-FE]	ACCMODE
3 : n	key#1 : key#m	C ¹⁾ : C ¹⁾	xx : xx	KEY

1) The condition is AccessMode set to SendKey.
2) Shall be an even number greater than \$01 if additional levels of security are supported,(\$02 default).

Table 31 — SecurityAccess Positive Response#2 Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	SecurityAccess Positive Response#2 Service Id	S	67	SA#2PR
2	SendKey: shall be an even number one greater than AccessMode in request#1 if additional levels of security are supported, (\$02 default) ManufacturerSpecific]	M	xx=[¹⁾ 02, 04, : 80, 82-FE]	ACCMODE
3	SecurityAccessStatus=[SecurityAccessAllowed]	U	34	SACCSTAT

1) Shall be an even number greater than \$01 if additional levels of security are supported,(\$02 default).

Table 32 — SecurityAccess Negative Response#2 Message

Data Byte	Parameter Name	CVT	Hex Value	Mnemonic
1	NegativeResponse#2 Service Id	S	7F	NACK
2	SecurityAccess Request Service Id	S	27	SA
3	ResponseCode=[KWP2000ResponseCode, ManufacturerSpecific]	M	xx=[00-7F, 80-FF]	RC

6.3.3 Message description

This mode is intended to be used to implement the data link security measures defined in SAE J 2186.

The client shall request the server to "unlock" itself by sending the service SecurityAccess request#1. The server shall respond by sending a "seed" using the service SecurityAccess positive response#1. The client shall respond by returning a "key" number back to the server using the service SecurityAccess request#2. The server shall compare this "key" to one internally stored. If the two numbers match, then the server shall enable ("unlock") the client's access to specific KWP 2000 services and indicate that with the service SecurityAccess positive response#2. If upon two (2) attempts of a service SecurityAccess request#2 by the client, where the two keys do not match, then the server shall insert a 10 s time delay before allowing further attempts.

The 10 s time delay shall also be required before the server responds to a service SecurityAccess request#1 from the client after server power-on.

If a device supports security, but is already unlocked when a SecurityAccess request#1 is received, that server shall respond with a SecurityAccess positive response#1 service with a seed of "\$00 00". A client shall use this method to determine if a server is locked by checking for a non-zero seed.

The security system shall not prevent normal diagnostic or vehicle communications between the client and the servers. Proper "unlocking" of the server is a prerequisite to the client's ability to perform some of the more critical functions such as reading specific memory locations within the server, downloading information to specific locations, or downloading routines for execution by the controller. In other words, the only access to the server permitted while in a "locked" mode is through the server specific software. This permits the server specific software to protect itself from unauthorized intrusion.

Servers that provide security shall support reject messages if a secure mode is requested while the server is locked.

Some servers could support multiple levels of security, either for different functions controlled by the server, or to allow different capabilities to be exercised. These additional levels of security can be accessed by using the service SecurityAccess requests#1 and #2 with an AccessMode value greater than the default value. The second data byte of the "RequestSeed" shall always be an odd number, and the second data byte of the service to "SendKey" shall be the next even number.