INTERNATIONAL STANDARD

ISO 14229

First edition
1998-07-15

Road vehicles — Diagnostic systems — Diagnostic services specification

Véhicules routiers — Systèmes de diagnostic — Spécification des services de diagnostic

Reference number
ISO 14229:1998(E)

Contents

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, govermental and non-govermental, 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 committee 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 14229 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

Annex A of this International Standard is for information only.

Introduction

This International Standard has been established in order to define common requirements for diagnostic systems, whatever the serial data link is.

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), in accordance with figure 1.

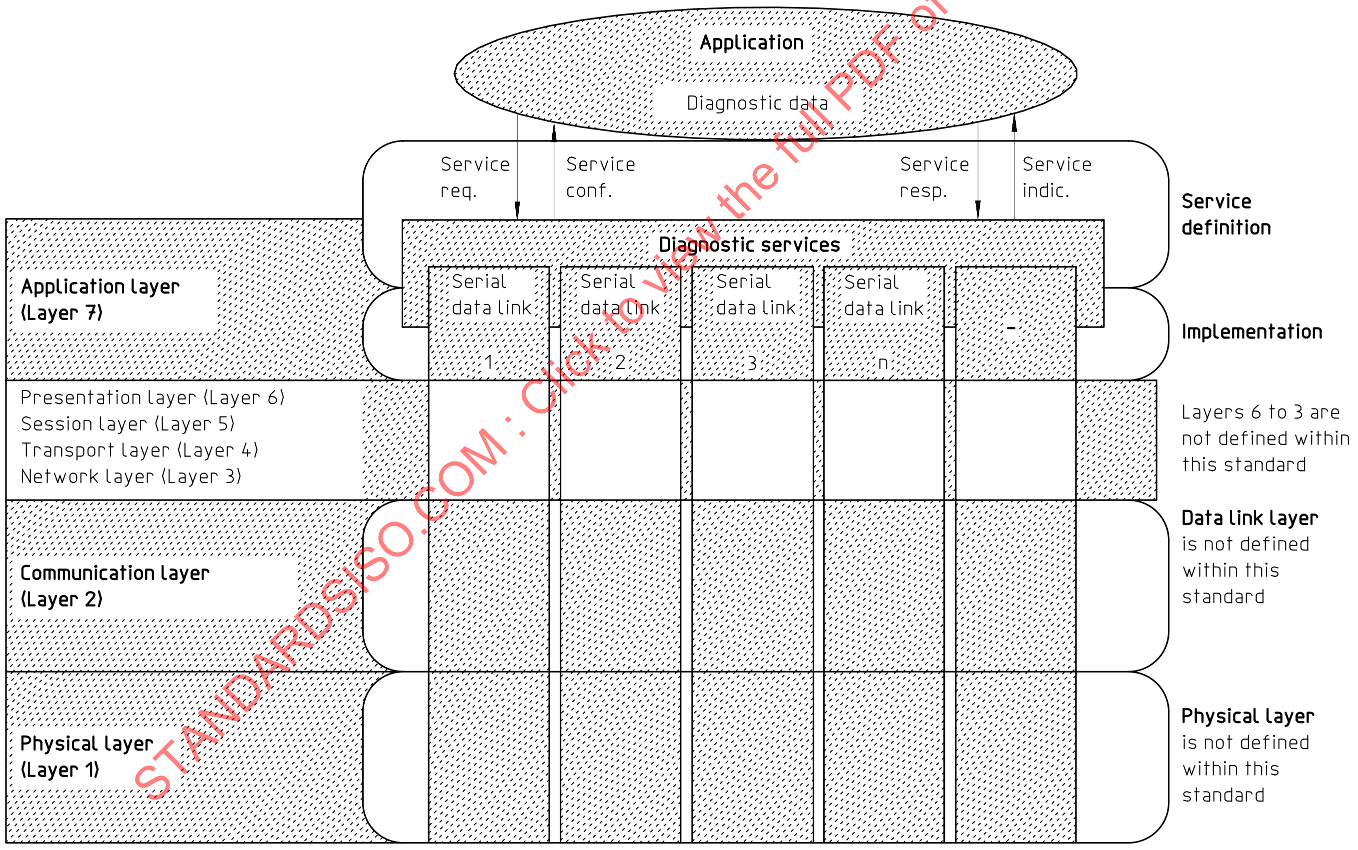


Figure 1 — Mapping of the diagnostic services on the OSI Model

This International Standard contains references to SAE publications, which are regularly amended/updated without any visible change (neither in the numbering, nor any additive letter, etc.). To ensure precisely to which particular edition this International Standard refers, annex A gives the precise dates of the SAE publications used.

Road vehicles — Diagnostic systems — Diagnostic services specification

1 Scope

This International Standard specifies common requirements of diagnostic services which allow a diagnostic tester to control diagnostic functions in an on-vehicle electronic control unit (e.g. electronic fuel injection, automatic gearbox, anti-lock braking system, etc.) connected on a serial data link embedded in a road vehicle.

It specifies generic services which allow the diagnostic tester to stop or to resume non-diagnostic message transmission on the data link.

This International Standard does not apply to non-diagnostic message transmission, nor to use of the communication data link between two electronic control units.

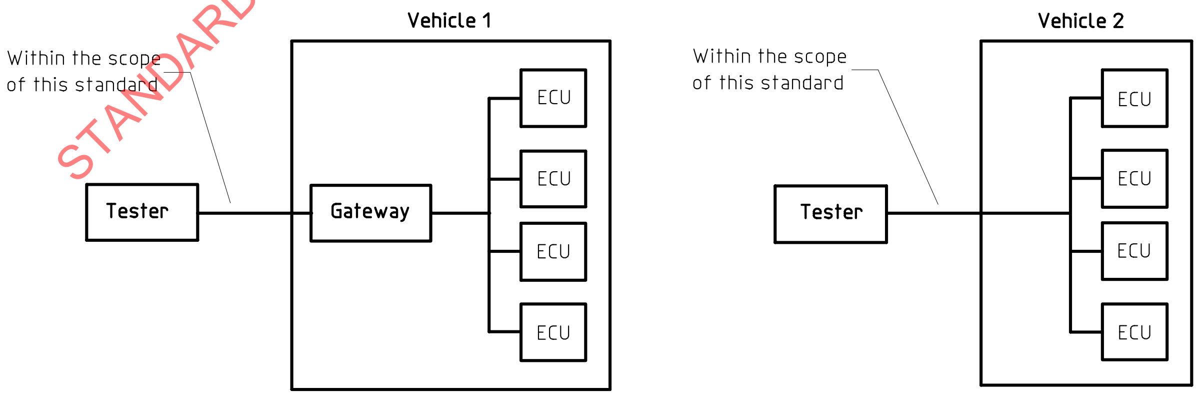
This International Standard neither specifies implementation requirements, numerical values of services and parameters, nor requirements for the communication services.

The vehicle diagnostic architecture of this International Standard applies to

— a single tester that may be temporarily or permanently connected to the on-vehicle diagnostic data link, and

— several on-vehicle electronic control units connected directly or indirectly.

See figure 2.


Vehicle 1
Within the scope of this standard
Tester
Gateway
ECU
ECU
ECU
ECU
Vehicle 2
Within the scope of this standard
Tester
ECU
ECU
ECU
ECU

In vehicle 1, the ECUs are connected over an internal data link and indirectly connected to the diagnostic data link through a gateway. This standard applies to the diagnostic communications over the diagnostic data link; the diagnostic communications over the internal data link may conform to this standard or to another protocol.

In vehicle 2, the ECUs are directly connected to the diagnostic data link.

Figure 2 — Vehicle diagnostic architecture

2 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreement based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 10731:1994, *Information technology — Open Systems Interconnection — Basic Reference Modal — Conventions for the definition of OSI services*.

3 Definitions and abbreviations

3.1 Terms defined in other standards

3.1.1
type
named set of values

3.1.2
bitstring type
type whose values are strings (sequences) of bits

NOTE — The bitstring type is used if no range is defined except the number of bits used (e.g. diagnostic trouble codes could be represented by a sequence of bits).

3.1.3
integer type
a simple type with distinguished values which are the positive and the negative whole numbers, including zero

NOTE — The range of type integer is not specified.

3.1.4
diagnostic trouble code
numerical common identifier for a fault condition identified by the on-board diagnostic system [SAE J 1930]

3.2 Additional definitions

3.2.1
diagnostic service
information exchange initiated by a client in order to require diagnostic information from a server or/and to modify its behaviour for diagnostic purpose

3.2.2
client
function that is part of the tester and that makes use of the diagnostic services

NOTE — A tester normally makes use of other functions such as data base management, specific interpretation, man-machine interface.

3.2.3
server
function that is part of an ECU and that provides the diagnostic services

NOTE — This International Standard differentiates between the server (i.e. the function) and the ECU so that it remains independent of the implementation.

3.2.4
tester
system that controls functions such as test, inspection, monitoring, or diagnosis of an on-vehicle ECU

NOTE — Testers may be dedicated to a specific type of operators (e.g. a scan tool dedicated to garage mechanics or a test tool dedicated to assembly plant agents). They may also be dedicated to several or to all types of operators.

3.2.5
diagnostic data
data that is located in the memory of an ECU which may be inspected and/or possibly modified by the tester (diagnostic data includes analogue inputs and outputs, digital inputs and outputs, intermediate values and various status information)

EXAMPLE — Examples of diagnostic data: vehicle speed, throttle angle, mirror position, system status, etc.

Two types of values are defined for diagnostic data:

— the current value: the value currently used by (or resulting from) the normal operation of the ECU;

— a stored value: an internal copy of the current value made at specific moments (e.g. when a malfunction occurs or periodically); this copy is made under the control of the ECU. The ECU is not obliged to keep internal copies of its data for diagnostic purposes, in which case the tester may only request the current value.

3.2.6
diagnostic session
level of diagnostic functionality provided by the server

EXAMPLE — Defining a repair shop or development testing session selects different ECU functionality (e.g. access to all memory locations may only be allowed in the development testing session).

3.2.7
diagnostic routine
routine that is embedded in an ECU and that may be started by a Server upon a request from the Client

EXAMPLE — It could either run instead of a normal operating program, or could be enabled in this mode and executed with the normal operating program. In the first case, normal operation for the Server is not possible. In the second case, multiple diagnostic routines may be enabled that run while all other parts of the ECU are functioning normally.

3.2.8
record
one or more diagnostic data elements that are referred to together by a single means of identification

EXAMPLE — A snapshot including various input/output data and trouble codes is an example of a record.

3.2.9
freeze frame
record of datas stored at the occurrence of an event

NOTE — Services defined allow the tester to request data stored in a freeze frame.

EXAMPLE — A freeze frame may be used to store the context of the ECU's operation either at periodic moments or at the occurrence of a malfunction. The Server may maintain several freeze frames (e.g. the same context may be stored several times as a trace of the N latest malfunctions, or each freeze frame may consist of a different context).

3.2.10
functional unit
set of functionally close or complementary diagnostic services

The following functional units have been identified:

- diagnostic management: this functional unit allows the initialization and the termination of a sequence of diagnostic interchanges between a client and a server;

- data transmission: this functional unit allows a client to request the server for the current values of a record. This functional unit may be used to request identification information;

- stored data transmission: this functional unit allows a client to request the server for the stored values of a record (for instance, the diagnostic trouble codes). A server may store record values (either periodically or at the occurrence of a malfunction) in a freeze frame. This freeze frame may be cleared upon request from the client. If the server supports multiple freeze frames then the client may trace the server's operation;

- input/output control: this functional unit allows the client to control the input and output peripherals of the ECU where the server is present;

- remote activation of routine: this functional unit allows the client to control specific diagnostic routines on the ECU where the server is present;

- upload download : this functional unit allows the client to control the transfer of large blocks of data to or from the server.

3.3 Abbreviations

OSI	Open Systems Interconnection.
ECU	Electronic Control Unit.

EXAMPLE — Systems considered as electronic control units: anti-lock braking system (ABS), engine management system, etc.

PID	Parameter IDentifier.

4 Convention

4.1 Interactions

This International Standard is guided by the conventions adopted in the OSI service conventions (ISO/IEC 10731) as they apply to the diagnostic services. These conventions specify the interactions between the service user and the service provider. Information is passed between the service user and the service provider by service primitives, which may convey parameters.

The distinction between service and protocol is shown in figure 3.

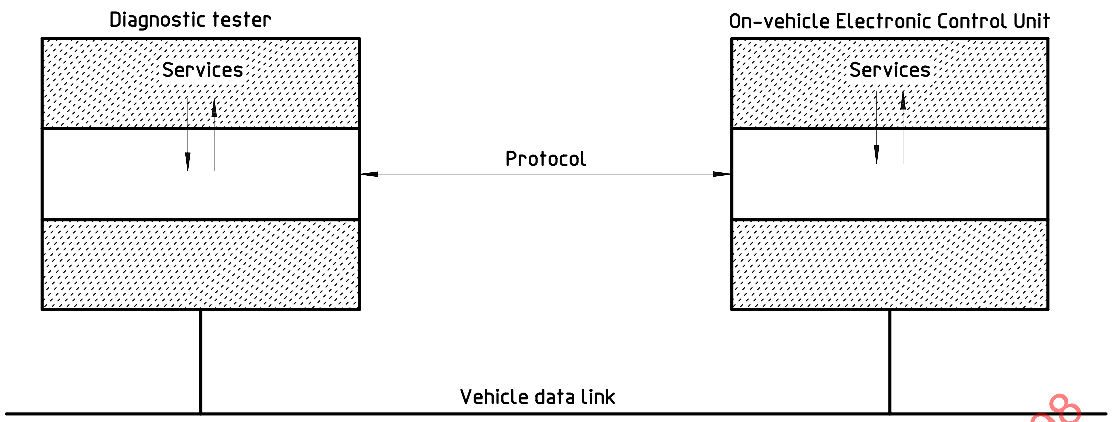


Figure 3 — Services and protocol

4.2 OSI service

This International Standard makes use of the following conventions defined in ISO/IEC 10731 as they apply to the diagnostic services:

- Service primitive: a service is transferred to or from the diagnostic layer by passing a service primitive across the layer interface.

- Service user: the service user is the diagnostic application which requests the service of the diagnostic layer. For diagnostic services described in this International Standard, it is always the client.

- Service provider: the service provider is the diagnostic application which responds to the service requested by the service user. For diagnostic services described in this International Standard, it is always the server.

- Request primitive: the request transfers the service from the service user to the diagnostic layer.

- Indication primitive: the indication transfers the service from the diagnostic layer to the service provider.

- Response primitive: the response transfers an answer to an indication primitive from the service provider to the diagnostic layer.

- Confirmation primitive: the confirmation transfers an answer to a request primitive from the diagnostic layer to the service user.

These conventions are shown in figure 4.

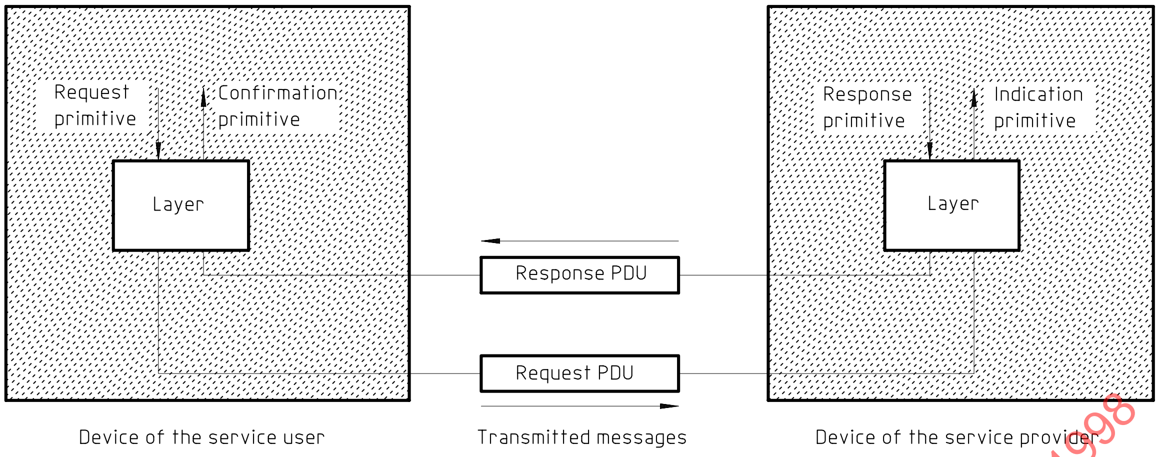


Figure 4 — Description of OSI service convention

4.3 Service description

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

- service purpose;
- service table;
- service procedure.

4.3.1 Service purpose

This subclause gives a brief outline of the functionality of the service.

4.3.2 Service table

The description of each service includes a table which lists the parameters of its primitives: request/indication, response/confirmation for positive or negative result. All have the same structure. For instance, table 1 shows the ReadDataByCommonIdentifier service table.

Table 1 — Example of service table

ReadDataByCommonIdentifier request	M
Record common identifier	M
Transmission mode	U
ReadDataByCommonIdentifier positive response	S
Record common identifier	M
Record value	M
ReadDataByCommonIdentifier negative response	S
Response code	M
Record common identifier[1)]	U
Transmission mode[1)]	U/C
1) Parameters of the request shall be repeated together. C = condition: parameter may be present only if it was already in the request.	

In all requests and responses, Client and Server Identifiers are mandatory, unless the service was performed on a point-to-point communication. They are not shown in the table 1.

NOTE — The parameter Server Identifier may be included in the initialization phase for a point-to-point communication (it will depend on the implementation).

Under the **"Service name" request** are listed the parameters specific to the service request/indication.

Under the **"Service name" positive response** are listed the parameters specific to the service response/confirmation in case the requested service has succeeded.

Under the **"Service name" negative response** are listed the parameters specific to the service response/confirmation in case the requested service has failed.

Parameters are indented under the service name.

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

M	mandatory;
U	user option: the parameter shall or shall not be supplied, depending on dynamic usage by the manufacturer;
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.

4.3.3 Service procedure

This subclause provides a description of the actions performed by the client and the server.

4.4 Functional unit table

Similar services are grouped into a functional unit. The description of each functional unit includes a table which lists its services.

5 Parameter specification

5.1 General purpose parameters

The structure of a service makes reference to parameters exchanged between server and client when the service is in progress.

The parameters of general purpose are specified in this clause. Parameters specific to a functional unit are specified in the corresponding clause.

This International Standard gives a list for some parameters (e.g. the response code parameter). Outside this list, some other parameter may be reserved either for future definition in the revision of this standard or for the system designer's specific use.

5.2 Server identifier

This parameter identifies a server. This clause neither specifies the list of possible server identifiers, nor defines a specific means for the client to get this list.

EXAMPLE 1 — Example of server identifiers are : ABS, engine management system.

EXAMPLE 2 — A server (e.g. the engine control) may be implemented in one ECU or distributed in several ECUs. An ECU may include only one server or several servers. At present, in most cases, a server is implemented in one ECU, and there is only one server in each ECU.

5.3 Client identifier

This parameter identifies a client. This clause neither specifies the list of possible client identifiers, nor defines a specific means for the server to get this list.

EXAMPLE — Example of client identifier: Scan tool.

5.4 Local identifier and common identifier

This subclause differentiates between local identifiers and common identifiers for the identification of a record, input/output or routine.

A common identifier is understood in the same way by all servers supporting this identifier.

The interpretation of a local identifier is server-specific.

EXAMPLE — The parameter identifier (PID) used by SAE J2190 and SAE J1979 are a sub-set of the record common identifier. For instance, in the SAE J1979 Recommended Practice, PID $0B identifies the intake manifold absolute pressure for every system. This data could be assigned a specific local identifier on some systems.

5.5 Memory address and routine address

This parameter identifies a specific server memory or routine location. It contains all the necessary information to address any memory or routine location including memory type (e.g. RAM, EEPROM, etc.) or memory paging.

5.6 Response code

The response code parameter is used within the negative response to indicate that the server could not complete the request.

The standard list for this parameter is described in the subclauses below.

5.6.1 General reject

The service is rejected but the server does not specify the reason of the rejection.

5.6.2 Service not supported

It indicates that the requested action will not be taken because the server does not support the requested service.

EXAMPLE — The client requests a service of an advanced functional unit (e.g. data transfer) which is not supported.

5.6.3 Subfunction not supported or invalid format

It indicates that the server does not support the arguments of the requested service or the format of the argument bytes do not match the prescribed format for the specified service.

EXAMPLE — The client requests a ReadDataByLocalIdentifier service, but the specified record local identifier value is not supported by this server.

5.6.4 Busy - RepeatRequest

The server has understood the service request, but it cannot execute at this time and will not start. In order to get a positive response the client should repeat the request later.

EXAMPLE — This response code indicates that the server is temporarily too busy to perform the requested operation. In this circumstance repetition of the request will eventually result in an affirmative response. This code may be returned while a server is in the process of clearing stored codes or fetching information.

5.6.5 Conditions not correct or request sequence error

It indicates that the requested action will not be taken because the server prerequisite conditions are not met. This request may elicit an affirmative response at another time. This code may occur when sequence-sensitive requests are issued in the wrong order.

5.6.6 Routine Not Complete or Service In Progress

This response code is used to indicate that the message was properly received and the service is in process, but not yet completed.

EXAMPLE — Whenever completion of the requested operation will exceed the maximum response time limit, it is appropriate to use the routine Not Complete or Service In Progress response code to lengthen the acceptable response period.

5.6.7 Request out of range

It indicates that the requested action will not be taken because the server detects the request message contains a data byte which attempts to substitute a value beyond its range of authority.

EXAMPLE — Attempting to modify a data byte of 111 when the data is only defined to 100.

5.6.8 Security access denied

The service is rejected because the server's security strategy has not been satisfied by the client.

5.6.9 Invalid key

It indicates that security access has not been given because the server's security key was not matched by the client (this counts as an attempt to gain security access).

5.6.10 Exceed number of attempts

It indicates that the requested action will not be taken because the client has unsuccessfully attempted to gain security access more times than the server's security strategy will allow.

5.6.11 Required time delay not expired

It indicates that the requested action will not be taken because the client’s latest attempt to gain security access was initiated before the server’s required time-out period had elapsed.

5.6.12 Download not accepted

This response code indicates that an attempt to download to a server's memory cannot be accomplished due to some fault condition.

5.6.13 Improper download type

This response code indicates that an attempt to download to a server's memory cannot be accomplished because the server does not support the type of download being attempted.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.14 Can't Download to specified address

This response code indicates that an attempt to download to a server's memory cannot be accomplished because the server does not recognise the target address for the download as being available.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload / download protocol.

5.6.15 Can't download number of bytes requested

This response code indicates that an attempt to download to a server's memory cannot be accomplished because the server does not recognise the number of bytes requested for the download as being available.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.16 Upload not accepted

This response code indicates that an attempt to upload from a server's memory cannot be accomplished due to some fault condition.

5.6.17 Improper upload type

This response code indicates that an attempt to upload from a server's memory cannot be accomplished because the server does not support the type of upload being attempted.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.18 Can't upload from specified address

This response code indicates that an attempt to upload from a server's memory cannot be accomplished because the server does not recognise the target address for the upload as being available.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.19 Can't upload number of bytes requested

This response code indicates that an attempt to upload from an server's memory cannot be accomplished because the server does not recognise the number of bytes requested for the upload as being available.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.20 Transfer suspended

This response code indicates that a block transfer operation was halted due to some fault, but will be completed later.

5.6.21 Transfer aborted

This response code indicates that a block transfer operation was halted due to some fault, and will not be completed later.

5.6.22 Illegal address in block transfer

This response code indicates that the starting address included in the block transfer request message is either out of range, protected, the wrong type of memory for receiving data, or cannot be written to for some reason.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.23 Illegal byte count in block transfer

This response code indicates that the number of data bytes included in the block transfer request message is either more than the block transfer can accommodate, requires more memory than is available at the requested starting address, or cannot be handled by the block transfer software.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.24 Illegal block transfer type

This response code is used to indicate that the block transfer type included in the request for block transfer is not valid for this application.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.25 Block transfer data checksum error

This response code is used to indicate that the block transfer data checksum calculated for the block transfer message does not agree with the expected value.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.26 Incorrect byte count during block transfer

This response code indicates that the number of bytes that was expected to be sent was not the same as the number of bytes received.

This code corresponds to parameters of the upload/download functional unit which are not defined in this International Standard but which are likely to be used by manufacturers. Its definition shall therefore be completed by the manufacturer depending on its specific upload/download protocol.

5.6.27 Request correctly received - Response pending

This response code is used to indicate that the message was received correctly, and that any parameters included in the message were valid, but the action to be performed may not be completed yet. This response code can be used to indicate that the message was properly received and shall not be re-transmitted, but the receiving device is not yet ready to receive another request message.

5.6.28 Manufacturer specific codes

These response codes are reserved to be specified by the vehicle manufacturer.

5.7 Response code handling

Figure 5 specifies the server behaviour on a client request message. This figure shows the logic as specified in the description of the response codes and to be implemented in the server and client as appropriate.

6 Diagnostic management functional unit

6.1 General

6.1.1 Table 2 describes the diagnostic management functional unit.

Table 2 — Diagnostic management functional unit

Service name	Description
StartDiagnosticSession	The client requests to start a diagnostic session with a specific server
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 that it is still present
ECUReset	The client forces the server to perform a reset.
ReadECUIdentification	The client requests identification data from the server
DisableNormalMessageTransmission	The client requests the server to stop transmitting non-diagnostic messages
EnableNormalMessageTransmission	The client requests the server to resume transmitting non-diagnostic messages
NOTE — It is recalled that: - the client is the function of the tester that makes use of the diagnostic services. - the server is the function of the on-vehicle ECU that provides the diagnostic services.	

6.1.2 The services of this functional unit make use of the following parameters:

Diagnostic mode
This parameter is used by the StartDiagnosticSession service to select the diagnostic session (specific behaviour of the server). This International Standard neither specifies the list for this parameter, nor defines a specific means for the server to get this list.

EXAMPLES — Repair shop, on-line, manufacturer specific, regulation, etc.

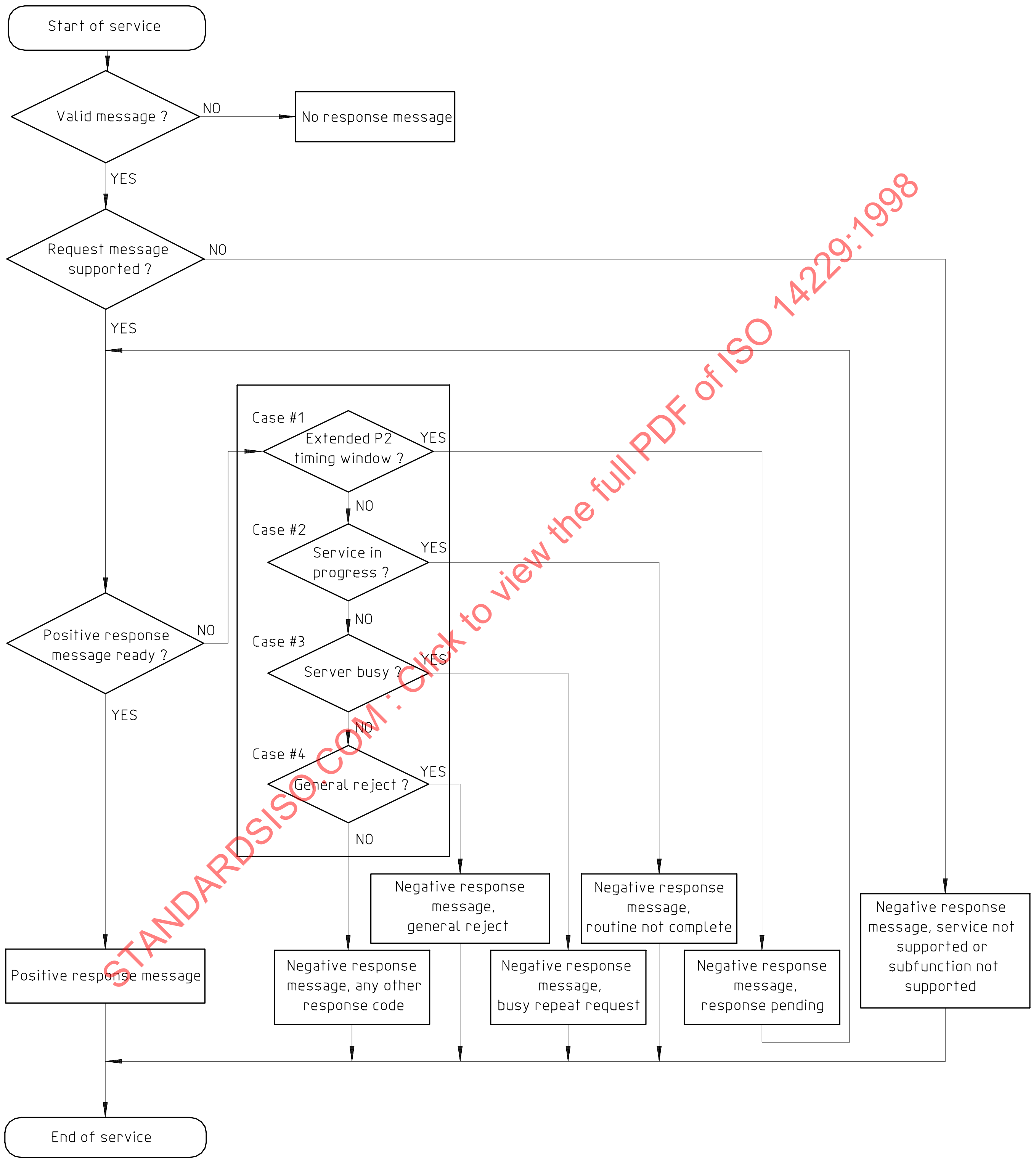

Start of service
Valid message ?
NO
No response message
YES
Request message supported ?
NO
YES
Case #1
Extended P2 timing window ?
YES
NO
Case #2
Service in progress ?
YES
NO
Case #3
Server busy ?
YES
NO
Case #4
General reject ?
YES
NO
Positive response message ready ?
NO
YES
Positive response message
Negative response message, any other response code
Negative response message, general reject
Negative response message, busy repeat request
Negative response message, routine not complete
Negative response message, response pending
Negative response message, service not supported or subfunction not supported
End of service

Figure 5 — Server positive and negative response message behaviour

Access mode
This parameter 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. Two values are defined for the step in progress: Request seed and Send key.

Key
This parameter is used by the SecurityAccess service to unlock a server.

The key is computed by the client from the seed and sent to the server which compares it to its own calculation result.

Seed
This parameter is used by the SecurityAccess service to unlock a server.

The seed is supplied by the server in response to a request for security access by the client.

Security Access status
This parameter is used by the SecurityAccess service to indicate positive response to a request. It indicates that the server's security strategy has been satisfied by the client.
One value is defined for this parameter: Security access allowed along with a range of manufacturer-specific-values.

Response required
This parameter is used in the TesterPresent service.
Two values are defined: Yes, No.

Reset mode
This parameter is used by the ECUReset service to describe how the server has to perform the reset. The possible values are:

Power on: the state of the server after the performance of the reset shall be the same as if the power supply is switched off and on again.

Power on while maintaining the communication: the state of the server after the performance of the reset shall be the same as if the power supply is switched off and on again; furthermore the communication link between the client and the server is maintained.

Manufacturer-specific: this reset mode value is manufacturer defined.

Reset status
This parameter is used by the ECUReset service to give to the client information about the status of the requested reset.

Identification option
This parameter is used by the ReadECUIdentification service to describe the kind of identification data requested by the client.

Transmission level
This parameter is used by the DisableNormalMessageTransmission service. The structure of this parameter is manufacturer specific.

6.2 StartDiagnosticSession service

6.2.1 Service purpose

The purpose of this service is to switch the server from the previous operating mode (normal operation or previous diagnostic mode) to a new diagnostic mode (e.g. repair shop, on-line, manufacturer, etc.).

6.2.2 Service table

Table 3 defines the StartDiagnosticSession service

Table 3 — StartDiagnosticSession service

StartDiagnosticSession request Diagnostic mode	M M
StartDiagnosticSession positive response Diagnostic mode	S U
StartDiagnosticSession negative response Response code Diagnostic mode	S M U

6.2.3 Service procedure

Upon receiving a StartDiagnosticSession service indication primitive, the server shall check that the conditions for starting the diagnostic session are valid. These conditions are not specified.

If valid, it will issue a StartDiagnosticSession response primitive with the positive response parameters.

If not, the server will issue a StartDiagnosticSession response primitive with the negative response parameters.

The following is valid:

— System designers have the flexibility to use a StartDiagnosticSession service to stop a previous session and start a new one. If the diagnostic mode requested by the client is the same as the current one, the server shall continue the active diagnostic session.

— The on-board devices may need to alter their normal operation in order for the diagnostic procedures to be effective. One example is that the system may need to reduce the amount of data being sent on the data link to make more time available for the diagnostic messages. Another possibility is that the system may need to degrade its normal operation in order to be able to process the diagnostic requests. Systems that rely on data from modules being diagnosed may not receive data from those modules as frequently as normal.

— This mechanism may be used to prevent a diagnostic session from timing-out.

6.3 StopDiagnosticSession service

6.3.1 Service purpose

The purpose of this service is to allow the server to terminate correctly the current diagnostic session (e.g. to perform some preventive actions) and to switch to the normal operation.

6.3.2 Service table

Table 4 defines the StopDiagnosticSession service.

Table 4 — StopDiagnosticSession service

StopDiagnosticSession request	M
StopDiagnosticSession positive response	S
StopDiagnosticSession negative response Response code	S M

6.3.3 Service procedure

Upon receiving a StopDiagnosticSession service indication primitive, the server shall check if a diagnostic session is currently running. In this case, it shall stop the current diagnostic session, that is, perform the necessary action to return to a state in which it may restore its normal operating conditions. Then it shall issue a StopDiagnosticSession response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a StopDiagnosticSession response primitive with the negative response parameters selected.

EXAMPLE — Restoring the normal operating conditions of the server may include:

— resuming normal message transmission if they have been stopped by the client during the diagnostic session being stopped; in this case the procedure is equivalent to that of the EnableNormalMessageTransmissions service;

— the reset of all the actuators controlled it they have been activated by the client during the diagnostic session being stopped;

— resuming all normal algorithms of the server;

— re-locking the server if it had been unlocked by the client during the diagnostic session being stopped.

6.4 SecurityAccess service

6.4.1 Service purpose

The purpose of this service is to allow access to restricted information from the server.

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 server.

6.4.2 Service table

Table 5 defines the SecurityAccess service.

Table 5 — SecurityAccess service

SecurityAccess request	M
Access mode	M
Key	C1
SecurityAccess positive response	S
Access mode	M
Seed	C2
Security access status	U
SecurityAccess negative response	S
Response code	M
Access mode	M
Key	U/C3
C1 = condition: Key shall be present if access mode = Send key. C2 = condition: Seed shall be present if access mode = Request seed. C3 = condition: parameter may be present only if it was already in the request.	

6.4.3 Service procedure

The client requests the server to unlock itself by sending a first request with the parameter access mode set to Request seed. The server shall issue a SecurityAccess positive response with the parameter seed. The client calculates a key and then sends a second request with the parameter Access mode set to Send key and the parameter key calculated. The server compares this key to one internally stored. If the two numbers match, then the server shall issue a SecurityAccess positive response without the parameter seed and unlock itself.

If the server cannot execute the service, it shall issue a SecurityAccess response primitive with the negative response parameters.

The system designer has the flexibility to specify what the server may actually do if errors occur during the service. SAE J2186 includes a method for security access procedure.

6.5 TesterPresent service

6.5.1 Service purpose

The purpose of this service is to prevent a server from automatically returning to normal operation.

6.5.2 Service table

Table 6 defines the TesterPresent service.

Table 6 — TesterPresent service

TesterPresent request	M
Response required	U
TesterPresent positive response	S / C
TesterPresent negative response	S / C
Response code	M
C = condition: Response required = YES.	

6.5.3 Service procedure

Upon receiving a TesterPresent indication primitive, the server shall do whatever is necessary to continue its operation under the control of the client.

If parameter response required is not present, the default value shall be YES.

The positive or negative response is conditional on the response required parameter of the request.

EXAMPLE — This service may be used when it is possible that the current diagnostic session terminates automatically (instead of using the service StopDiagnosticSession). This default termination procedure may be realised with a time-out which may be re-started at the reception of a TesterPresent request.

This service may thus be used with the ReadDataBy LocalIdentifier / CommonIdentifier services, so that the server is sure never to terminate automatically the diagnostic session if the client does not request any other service while the server transmits record values in multiple response mode: the client may transmit the TesterPresent request each time it receives a value from the server.

6.6 ECUReset service

6.6.1 Service purpose

The purpose of this service is to allow the client to force the server to perform a reset, e.g. a power-on reset or to allow a new default parameter set to become active. The type of reset which is performed is up to the manufacturer.

6.6.2 Service table

Table 7 defines the ECUReset service.

Table 7 — ECUReset service

ECUReset request	M
Reset Mode	U
ECUReset positive response	S
Reset Status	U
ECUReset negative response	S
Response code	M
Reset mode	U/C
C = condition: parameter may be present only if it was already in the request.	

6.6.3 Service procedure

Upon receiving an ECUReset indication primitive, the server shall first check if the requested reset may be performed under the present conditions. The type of reset is specified by the user-optional parameter reset mode. If the parameter reset mode is not present, the server shall perform a default reset.

If it is possible to perform the requested reset, the server shall issue an ECUReset response primitive with the positive response parameters selected.

The server has to take care that it will be able to send a positive response after the reset otherwise response shall be provided before.

EXAMPLE — If the service is used on a data link which requires initialization.

If the server is not able to execute the requested reset, the server shall issue an ECUReset response primitive with the negative response parameters selected.

6.7 ReadECUIdentification service

6.7.1 Service purpose

The purpose of this service is to allow the client to read ECU identification data.

6.7.2 Service table

Table 8 defines the ReadECUIdentification service.

Table 8 — ReadECUIdentification service

ReadECUIdentification request	M
Identification Option	U
ReadECUIdentification positive response	S
Identification Record Value	M
ReadECUIdentification negative response	S
Response code	M
Identification option	U/C
C = condition: parameter may be present only if it was already in the request.	

6.7.3 Service procedure

After receiving a ReadECUIdentification indication primitive, the server shall check if the requested information may be provided. The type of information which is requested may be specified by the optional parameter identification option. If not present, the server shall deliver a default set of information.

If the requested information may be provided, the server shall deliver a ReadECUIdentification response primitive with the positive response parameters selected.

If the requested information cannot be provided for any reason, the server shall deliver a ReadECUIdentification response primitive with the negative response parameters selected.

NOTE — This service may be used by the client to select the proper application software.

Examples of identification data are a part number, or a date of manufacturing. Other examples may include the list of available services or PIDs.

6.8 DisableNormalMessageTransmission service

6.8.1 Service purpose

The purpose of this service is to reduce server transmitted data on the link while still performing other functions normally. The server will continue to operate in whatever mode it was operating in prior to the service request.

6.8.2 Service table

Table 9 defines the DisableNormalMessageTransmission service.

Table 9 — DisableNormalMessageTransmission service

DisableNormalMessageTransmission request Transmission Level	M U
DisableNormalMessageTransmission positive response Transmission Level	S U
DisableNormalMessageTransmission negative response Response code Transmission level	S M U/C
C = condition: parameter may be present only if it was already in the request.	

6.8.3 Service procedure

The optional parameter transmission level specifies which messages to disable.

EXAMPLE 1 If an unsafe or undesirable vehicle operating condition would result from the lack of normal messages, then this service could cause all non-essential messages to be inhibited. The optional level parameter may be used to indicate which normal mode messages to disable. Defining which messages to disable is determined by the system designer to ensure safe vehicle operation. When using this service, some provision shall be made to allow for other servers that rely on information from the silenced server so that they do not set diagnostic data due to the lack of required information.

Upon receiving a DisableNormalMessageTransmission indication primitive, the server shall stop transmitting the non-diagnostic messages specified by the transmission level parameter if it is present, or all non-diagnostic messages if it is not present. Then it shall issue a DisableNormalMessageTransmission response primitive with the positive response parameters selected.

EXAMPLE 2 One use for this service is to reduce message traffic on the data link. This would make more time available for diagnostic messages.
Another use for this service is to allow the Client to emulate a server for diagnostic purposes. In this scenario, the client would send a DisableNormalMessageTransmission service to the server to be emulated. The client would then respond to all normal communication polls directed to that server, most likely with data intended to cause a known response by other servers that use the information in the response. The client may then observe the actions of those servers.

If it is not possible to execute the service, the server shall issue a DisableNormalMessageTransmission response primitive with the negative response parameters selected.

6.9 EnableNormalMessageTransmission service

6.9.1 Service purpose

The purpose of this service is to cause a server to resume normal communications after previously disabling these messages by a DisableNormalMessageTransmission service. The server will continue to operate in whatever conditions were set by previous messages, such as executing on-board routines.

6.9.2 Service table

Table 10 defines the EnableNormalMessageTransmission service.

Table 10 — EnableNormalMessageTransmission service

EnableNormalMessageTransmission request	M
EnableNormalMessageTransmission positive response	S
EnableNormalMessageTransmission negative response Response code	S M

6.9.3 Service procedure

Upon receiving a EnableNormalMessageTransmission indication primitive, the server shall resume transmitting non-diagnostic messages, and then issue an EnableNormalMessageTransmission response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a EnableNormalMessageTransmission response primitive with the negative response parameters selected.

7 Data Transmission functional unit

7.1 Services of functional unit

Table 11 gives the data transmission functional unit.

The services of this functional unit make use of the following parameters:

Transmission mode
This parameter indicates how the server is requested to transmit the record in case of multiple responses. The possible values are:

— Stop: the server stops transmitting the record. This applies only if the record is being transmitted periodically;

— Single: the server transmits the record value only once;

— Slow: the server transmits the record periodically at the slow rate. The value of the slow rate is always predefined on the server and may be set by the client with the SetDataRates service;

— Medium: the server transmits the record periodically at the medium rate. The value of the medium rate is always predefined on the server and may be set by the client with the SetDataRates service;

— Fast: the server transmits the record periodically at the fast rate. The value of the fast rate is always predefined on the server and may be set by the client with the SetDataRates service;

— Conditional: the server transmits the record value when a predefined software condition becomes true.

NOTE — The conditional option is useful for transmitting data at more specific moments than periodic transmission: only when a specific condition occurs. It saves the bus load from possible non-significant transmissions. Examples of software conditions are "at each beginning of the main loop of the program" or "when this variable becomes greater than a threshold value". The system designer has the flexibility to define the realization of the software condition (definition, detection and signalling). At present, only one predefined condition may be specified, but it is possible to reserve several values of the transmission mode parameter to allow the specification of several conditions.

Table 11 — Data transmission functional unit

Service name	Description
ReadDataByLocalIdentifier	The client requests the transmission of the current value of a record with access by record local identifier
ReadDataByCommonIdentfier	The client requests the transmission of the current value of a record with access by record common identifier
ReadMemoryByAddress	The client requests the transmission of a memory area
DynamicallyDefineLocalIdentifier	The client requests to dynamically define local identifiers that may subsequently be accessed by record local identifier
WriteDataByLocalIdentifier	The client requests to write a record accessed by record local identifier
WriteDataByCommonIdentifier	The client requests to write a record accessed by record common identifier
WriteMemoryByAddress	The client requests to overwrite a memory area
SetDataRates	The client changes the data rates for the multiple transmissions
StopRepeatedDataTransmission	The client requests to stop all transmissions
NOTE — It is recalled that: - the client is the function of the tester that makes use of the diagnostic services; - the server is the function of the on-vehicle ECU that provides the diagnostic services.	

Maximum number of responses to send
This parameter of type integer specifies the maximum number of responses the server shall send in case of multiple responses.

Record value
Value of a record which may be identified by a record local identifier, record common identifier or a memory address.

Memory size
This parameter specifies the length of a memory area.

Dynamically defined local identifier
This parameter is used in the DynamicallyDefineLocalIdentifier service. It specifies to the server the local identifier that the client requests to define.

Definition mode
This parameter is used in the DynamicallyDefineLocalIdentifier service. It allows the client to specify to the server the parameter which is to be used for defining the dynamically defined local identifier. The possible values are:

define by local identifier,
define by common identifier,
define by memory address, or
clear dynamically defined local identifier.

Position in dynamically defined local identifier
This parameter is used in the DynamicallyDefineLocalIdentifier service. It allows the client to specify to the server the concerned data field in the dynamically defined local identifier.

Position in record local identifier and position in record common identifier
These parameters are used in the DynamicallyDefineLocalIdentifier service. They allow the client to specify to the server which data field of the local or common identifier shall be included in the dynamically defined local identifier.

Slow rate, medium rate or fast rate
These parameters are used by the SetDataRate service. They specify the value of the period at which the server transmits the record value upon the next ReadDataBy LocalIdentifier / CommonIdentifier or ReadMemoryByAddress service request with the transmission mode parameter set to slow, medium or fast. Two values are reserved for these parameters:

— No change: the rate is not changed and keeps its current value;

— As fast as possible rate: the record value shall be transmitted as fast as possible.

7.2 ReadDataByLocalIdentifier service

7.2.1 Service purpose

The purpose of this service is to request a current record value by local identifier. The length and format of the data record is predefined on the server.

7.2.2 Service table

Table 12 defines the ReadDataByLocalIdentifier service.

7.2.3 Service procedure

Upon receiving a ReadDataByLocalIdentifier indication primitive, the server shall access the data elements of the record specified by the record local identifier parameter and transmit its value in a ReadDataByLocalIdentifier positive response primitive with the positive response parameters selected.

The server shall transmit the record value in accordance with the value of the transmission mode parameter, as defined in 7.1.

If it not possible for any reason, the server shall issue a ReadDataByLocalIdentifier negative response primitive with the negative response parameters selected.

NOTE — Record value may include analogue input and output, digital input and output, and system status information.

Table 12 — ReadDataByLocalIdentifier service

ReadDataByLocalIdentifier request	M
Record local identifier	M
Transmission mode	U
Maximum number of responses to send	U/C1
ReadDataByLocalIdentifier positive response	S
Record local identifier	M
Record value	M
ReadDataByLocalIdentifier negative response	S
Response code	M
Record local identifier 1)	U
Transmission mode 1)	U/C2
Maximum number of responses to send 1)	U/C2
1) Parameters of the request shall be repeated together. C1 = condition: Transmission mode parameter present. C2 = condition: Parameter may be present only if it was already in the request.	

7.3 ReadDataByCommonIdentifier service

7.3.1 Service purpose

The purpose of this service is to request a current record value by record common identifier. The length and format of the data record is predefined on the server.

7.3.2 Service table

Table 13 defines the ReadDataByCommonIdentifier service.

Table 13 — ReadDataByCommonIdentifier service

ReadDataByCommonIdentifier request	M
Record common identifier	M
Transmission mode	U
Maximum number of responses to send	U/C1
ReadDataByCommonIdentifier positive response	S
Record common identifier	M
Record value	M
ReadDataByCommonIdentifier negative response	S
Response code	M
Record common identifier 1)	U
Transmission mode 1)	U/C2
Maximum number of responses to send 1)	U/C2
1) Parameters of the request shall be repeated together. C1 = condition: Transmission mode parameter present C2 = condition: Parameter may be present only if it was already in the request.	

7.3.3 Service procedure

Upon receiving a ReadDataByCommonIdentifier indication primitive, the server shall access the data elements of the record specified by the record common identifier parameter and transmit its value in a ReadDataByCommonIdentifier positive response primitive with the positive response parameters selected.

The server shall transmit the record value in accordance with the value of the transmission mode parameter.

If it not possible for any reason, the server shall issue a ReadDataByCommonIdentifier negative response primitive with the negative response parameters selected.

NOTE — Record value may include analogue input and output, digital input and output, and system status information.

7.4 ReadMemoryByAddress service

7.4.1 Service purpose

The purpose of this service is to request, in a single response, the content of a memory area. The length of the memory area is either predefined on the server or defined with the memory size parameter.

7.4.2 Service table

Table 14 defines the ReadMemoryByAddress service.

Table 14 — ReadMemoryByAddress service

ReadMemoryByAddress request	M
Memory address	M
Memory size	M
Transmission mode	U
Maximum number of responses to send	U/C1
ReadMemoryByAddress positive response	S
Record value	M
Memory address	U
ReadMemoryByAddress negative response	S
Response code	M
Memory address [1)]	U
Memory size [1)]	U
Transmission mode [1)]	U/C2
Maximum number of responses to send [1)]	U/C2
1) Parameters of the request shall be repeated together C1 = condition: Transmission mode parameter present C2 = condition: Parameter may be present only if it was already in the request.	

7.4.3 Service procedure

After receiving a ReadMemoryByAddress indication primitive, the server shall read the requested data from the memory locations specified by memory address and memory size.

If the read access to the specified memory locations was successful, the server shall deliver a ReadMemoryByAddress response primitive with the positive response parameters selected.

If the access to the specified memory locations is impossible for any reason, the server shall deliver a ReadMemoryByAddress response primitive with the negative response parameters selected.

7.5 DynamicallyDefineLocalIdentifier service

7.5.1 Service purpose

This service shall be used to dynamically define local identifiers that may subsequently be requested by ReadDataByLocalIdentifier service. In a single request, several or all data fields of the dynamically defined local identifier may be defined, each one by using memory locations or a selected data field of already existing local or common identifiers.

7.5.2 Service table

Table 15 defines the DynamicallyDefineLocalIdentifier service.

Table 15 — DynamicallyDefineLocalIdentifier service

DynamicallyDefineLocalIdentifier request	M
Dynamically defined local identifier	M
1. Definition mode	M
Position in dynamically defined local identifier	C1 or C2 or C3
Memory size	C1 or C2 or C3
Record local identifier	C1
Position in record local identifier	U/C1
Record common identifier	C2
Position in record common identifier	U/C2
Memory address	C3
...n. Definition mode	M
Position In dynamically defined local identifier	C1 or C2 or C3
Memory size	C1 or C2 or C3
Record local identifier	C1
Position in record local identifier	U/C1
Record common Identifier	C2
Position in record common identifier	U/C2
Memory address	C3
DynamicallyDefineLocalIdentifier positive response	S
Dynamically Defined local identifier	M
DynamicallyDefineLocalIdentifier negative response	S
Response code	M
Dynamically defined local identifier [1)]	U
1. definition mode [1)]	U
Position In dynamically defined local identifier [1)]	U
Memory size [1)]	U
Record local identifier [1)]	U/C4
Position in record local identifier [1)]	U/C4
Record common identifier [1)]	U/C4
Position in record common identifier [1)]	U/C4
Memory address [1)]	U/C4
...n. Definition mode [1)]	U
Position In dynamically defined local identifier [1)]	U
Memory size [1)]	U
Record local identifier [1)]	U
Position in record local identifier [1)]	U/C4
Record common identifier [1)]	U/C4
Position in record common identifier [1)]	U/C4
Memory address [1)]	U/C4

[1)] Parameters of the request shall be repeated together.
C1 = condition: Definition mode = Define by local identifier.
C2 = condition: = Define by common identifier.
C3 = condition: = Define by memory address.
C4 = condition: parameter may be present only if it was already in the request.

7.5.3 Service procedure

Upon receiving a DynamicallyDefineLocalIdentifier indication primitive, the server shall issue a DynamicallyDefineLocalIdentifier response primitive with the positive response parameters selected.

If the parameter position in record local/common identifier is not present, the corresponding record local/common identifier is entirely included in the dynamically defined Local identifier.

If it is not possible to execute the service, the server shall issue a DynamicallyDefineLocalIdentifier response primitive with the negative response parameters selected.

EXAMPLE — Following data are already defined in a server:

- Local identifier	$32	**Vehicle speed**	**2 bytes**
- Common identifier	$0104	Engine Speed **Water temperature**	2 bytes **1 byte**
- Memory address	$248170	**Throttle angle**	**2 bytes**

Definition by a single request of local identifier $80 as water temperature, vehicle speed and throttle angle:

DynamicallyDefineLocalIdentifier request	Service identifier value
Dynamically defined local identifier	$80
Definition mode	DEFINE BY COMMON IDENTIFIER
Position In dynamically defined local identifier	1
Memory size	1
Position in record common identifier	2
Record common identifier	$0104
Definition mode	DEFINE BY LOCAL IDENTIFIER
Position In dynamically defined local identifier	2
Memory size	2
Position in record local identifier	1
Record common identifier	$32
Definition mode	DEFINE BY MEMORY ADDRESS
Position In dynamically defined local identifier	3
Memory size	2
Memory address	$248170

Definition by 3 requests of local identifier $80 as water temperature,vehicle speed and throttle angle:

DynamicallyDefineLocalIdentifier request #1	Service Identifier value
Dynamically defined local identifier	$80
Definition mode	DEFINE BY COMMON IDENTIFIER
Position In dynamically defined local identifier	1
Memory size	1
Position in record common identifier	2
Record common identifier	$0104

DynamicallyDefineLocalIdentifier request #2	Service Identifier value
Dynamically defined local identifier	$80
Definition mode	DEFINE BY LOCAL IDENTIFIER
Position In dynamically defined local identifier	2
Memory size	2
Position in record local identifier	1
Record common identifier	$32

DynamicallyDefineLocalIdentifier request #3	Service Identifier value
Dynamically defined local identifier	$80
Definition mode	DEFINE BY MEMORY ADDRESS
Position In dynamically defined local identifier	3
Memory size	2
Memory address	$248170

7.6 WriteDataByLocalIdentifier service

7.6.1 Service purpose

The purpose of this service is to provide a means for the client to change the contents of record local identifier memory locations of the server. The record local identifier and associated memory locations need to be known by the server. This service does not allow the client to change any locations other than those predefined in the server.

The number of data bytes included in each record local identifier depends on the system designer and intent of the usage for that record.

7.6.2 Service table

Table 16 defines the WriteDataByLocalIdentifier service.

Table 16 — WriteDataByLocalIdentifier service

WriteDataByLocalIdentifier request	M
Record local identifier	M
Record value	U
WriteDataByLocalIdentifier positive response	S
Record local identifier	M
WriteDataByLocalIdentifier negative response	S
Response code	M
Record value [1)]	U
Record local identifier [1)]	U/C
[1)] Parameters of the request shall be repeated together C = condition : parameter may be present only if it was already in the request.	

7.6.3 Service procedure

Upon receiving a WriteDataByLocalIdentifier indication primitive, the server shall execute the specific service procedure that the system designer has the flexibility to define.

EXAMPLE — Possible uses for this service are:

- Clear non-volatile memory,
- Reset learned values in a single table,
- Set option content,
- Set Vehicle Identification Number (VIN),
- Change calibration values.

This service may be used to clear or reset tables stored in non-volatile memory. A record local identifier could be defined as memory locations that need to be either cleared or resetted to predefined initial values, such as learned values in a table. Sending the record local identifier with no data bytes could cause the server to clear/reset all learned values to known nominal values.

Another use is to allow the VIN to be input. A record local identifier could be defined to expect a portion of the VIN. Multiple messages would be required to send the entire seventeen character VIN. Whatever data values were included in the poll would be stored in the memory locations known by the server for VIN.

In some server, this service could be used for programming calibrations or vehicle-specific information in the assembly plant or aftersales service without using secured methods for reprogramming. If desired that those values not be changed after production, the capability to modify those blocks could be disabled after the initial change.

At last the server shall issue a WriteDataByLocalIdentifier positive response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a WriteDataByLocalIdentifier negative response primitive with the negative response parameters selected.

7.7 WriteDataByCommonIdentifier service

7.7.1 Service purpose

The purpose of this service is to provide a means for the client to change the contents of a record common identifier memory locations of the server. The record common identifier and associated memory locations need to be known by the server. This service does not allow the client to change any locations other than those predefined in the server.

The number of data bytes included in each record common identifier depends on the system designer and intent of the usage for that record.

7.7.2 Service table

Table 17 defines the WriteDataByCommonIdentifier service.

Table 17 — WriteDataByCommonIdentifier service

WriteDataByCommonIdentifier request	M
Record common identifier	M
Record value	U
WriteDataByCommonIdentifier positive response	S
Record common identifier	M
WriteDataByCommonIdentifier negative response	S
Response code	M
Record common identifier [1)]	U
Record value [1)]	U/C
[1)] Parameters of the request shall be repeated together C = condition : parameter may be present only if it was already in the request	

7.7.3 Service procedure

Upon receiving a WriteDataByCommonIdentifier indication primitive, the server shall execute the specific service procedure that the system designer has the flexibility to specify.

At last the server shall issue a WriteDataByCommonIdentifier positive response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a WriteDataByCommonIdentifier response primitive with the negative response parameters selected.

7.8 WriteMemoryByAddress service

7.8.1 Service purpose

The purpose of this service is to overwrite or to set to a default value the content of a memory area. The length of the memory area is defined with the memory size parameter.

7.8.2 Service table

Table 18 defines the WriteMemoryByAddress service.

Table 18 — WriteMemoryByAddress service

WriteMemoryByAddress request	M
Memory address	M
Memory size	M
Record value	U
WriteMemoryByAddress positive response	S
Memory address	U
WriteMemoryByAddress negative response	S
Response code	M
Memory address [1)]	U
Memory size [1)]	U
Record value [1)]	U/C
1) Parameters of the request shall be repeated together C = condition : parameter may be present only if it was already in the request.	

7.8.3 Service procedure

After receiving a WriteMemoryByAddress indication primitive, the server shall write the received data starting at the memory address as many times as necessary to fit with the memory size. If no record value is specified in the request, a default value shall be taken instead.

If the write access to the specified memory locations was successful, the server shall deliver a WriteMemoryByAddress response primitive with the positive response parameters selected.

If the access to the specified memory locations is impossible by any reason, the server shall deliver a WriteMemoryByAddress response primitive with the negative response parameters selected.

7.9 SetDataRates service

7.9.1 Service purpose

The purpose of this service is to specify the setting of data rates for data to be transmitted by services ReadDataBy LocalIdentifier / CommonIdentifier or ReadMemoryByAddress. Servers shall have default slow, medium or fast rates.

7.9.2 Service table

Table 19 defines the SetDataRates service.

Table 19 — SetDataRates service

SetDataRates request	M
Slow rate	M
Medium rate	M
Fast rate	M
SetDataRates positive response	S
SetDataRates **n**egative response	S
Response code	M
Slow rate [1)]	U
Medium rate [1)]	U
Fast rate [1)]	U
[1)] Parameters of the request shall be repeated together	

7.9.3 Service procedure

Upon receiving a SetDataRates indication primitive, the server shall set its internally stored data rates to the values specified by the slow rate, medium rate and fast rate parameters and then issue a SetDataRates response primitive with the Positive response parameters selected.

The internally stored data rates shall be reset to the default values:

- at the reception of a StopDiagnosticSession service request,

- at the next power-off/power-on sequence.

If it is not possible to execute this service, the server shall issue a SetDataRates response primitive with the negative response parameters selected.

NOTE — The default values of the data rates are ECU-dependent: the three values may be the same or different, depending of the capability of the ECU.

EXAMPLE — The slow rate may be about 1 or 2 samples per second, the medium rate may be about 4 to 10 samples per second, and the fast rate 20 or more samples per second. The default rates shall be based on the frequency of change typical for the value reported, uses for that data, and the capability of the processor of the ECU.

7.10 StopRepeatedDataTransmission service

7.10.1 Service purpose

The purpose of this service is to stop all data transmission that were started by services ReadDataBy LocalIdentifier / CommonIdentifier or ReadMemoryByAddress.

If only an individual data request is to be stopped, then that value may be stopped by calling the appropriate service request with the transmission mode parameter set to Stop.

7.10.2 Service table

Table 20 defines the StopRepeatedDataTransmission service.

Table 20 — StopRepeatedDataTransmission service

StopRepeatedDataTransmission request	M
StopRepeatedDataTransmission positive Response	S
StopRepeatedDataTransmission negative response Response code	S M

7.10.3 Service procedure

Upon receiving a StopTransmission indication primitive, the server shall stop all the currently running multiple-response transmissions.

The server shall issue a StopTransmission positive response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a StopTransmission negative response primitive with the negative response parameters selected.

EXAMPLE — Possible actions are:

- the reset of the timer and/or counters assigned to each transmission;
- the inhibition of the detection of the software condition.

8 Stored data transmission functional unit

8.1 General

Table 21 gives the stored data transmission functional unit.

Table 21 — Stored data transmission functional unit

Service name	Description
ReadDiagnosticTroubleCodes	The client requests to the server the transmission of both number and values of the diagnostic trouble codes
ReadDiagnosticTroubleCodesByStatus	The client requests to the server the transmission of both number and values of the diagnostic trouble codes depending on their status.
ReadStatusOfDiagnosticTroubleCodes	The client requests to the server the transmission of the number, values and status of the diagnostic trouble codes.
ReadFreezeFrameData	The client requests to the server the transmission of the value of a record stored in a freeze frame
ClearDiagnosticInformation	The client requests to the server to clear all or a group of the diagnostic information stored.
NOTE — It is recalled that: - the client is the function of the tester that makes use of the diagnostic services; - the server is the function of the on-vehicle ECU that provides the diagnostic services.	

The services of this functional unit make use of the following parameters:

Group of diagnostic trouble codes (DTC)
This parameter is used by the ReadDTCs services to select to which functional group of DTC or to which specific DTC access is requested.

EXAMPLE — Format and standard group of DTCs are defined in SAE J2012 / SAE J1587 or may be defined in a manufacturer-specific document.

Number of DTC
This parameter is used by the ReadDTCs services to indicate to the client how many trouble codes have been stored by the server.

List of DTC
This parameter is used by the ReadDTC positive response to indicate which DTCs of the group specified in the request have been stored by the client. A specific standard value is NO DTCs STORED.

EXAMPLE — Format and standard list of DTCs are defined in SAE J2012 / SAE J1587 or may be defined in a manufacturer-specific document.

Status of DTC
This parameter is used by the ReadDTCByStatus to indicate the status of the DTCs which shall be read.

EXAMPLE — Possible status are:

- warning lamp illuminated;
- warning lamp was previously illuminated for this code, malfunction not currently detected, code not yet erased;
- current code present at time of request;
- stored trouble code;
- manufacturer-specific.

List of DTC and status
This parameter is used by the ReadStatusOfDTC positive response to indicate which DTCs and their associated status have been stored by the client. A specific standard value is NO DTCs STORED.

Freeze frame number
This parameter, of type integer, identifies the freeze frame in which the requested values are stored.

Record access method identifier
This parameter is used by the ReadFreezeFrameData service and allows to select the type of the record identification to access a specific data within a freeze frame.

EXAMPLE — Record identifications may be local identifier, common identifier, memory address, etc., or any other manufacturer-specific parameters.

Record identification
This parameter is used by the ReadFreezeFrameData service to provide the identification data to the client.

Group of diagnostic information
This parameter is used by the ClearDiagnosticInformation service to select which functional group of diagnostic information or which specific diagnostic information shall be cleared.

8.2 ReadDiagnosticTroubleCodes service

8.2.1 Service purpose

The purpose of this service is to read the number of stored diagnostic trouble codes and the codes themselves.

8.2.2 Service table

Table 22 defines the ReadDiagnosticTroubleCodes service.

Table 22 — ReadDiagnosticTroubleCodes service

ReadDiagnosticTroubleCodes request	M
Group of diagnostic trouble codes	U
ReadDiagnosticTroubleCodes positive response	S
Number of diagnostic trouble codes	M
List of diagnostic trouble codes	C1
ReadDiagnosticTroubleCodes negative response	S
Response code	M
Group of diagnostic trouble codes	U/C2
C1 = condition : parameter maybe present only if number of DTC is greater than 00. C2 = condition : parameter may be present only if it was already in the request.	

8.2.3 Service procedure

Upon receiving a ReadDiagnosticTroubleCodes service indication primitive, the server shall access the number and values of all the diagnostic trouble codes or the number and values within a group of diagnostic trouble codes if the optional parameter is present and then issue a ReadDiagnosticTroubleCodes service response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a ReadDiagnosticTroubleCodes service response primitive with the negative response parameters selected.

8.3 ReadDiagnosticTroubleCodesByStatus service

8.3.1 Service purpose

The purpose of this service is to read the number and the stored diagnostic trouble codes based on the status by which they have been stored.

8.3.2 Service table

Table 23 defines the ReadDiagnosticTroubleCodesByStatus service.

Table 23 — ReadDiagnosticTroubleCodesByStatus service

ReadDiagnosticTroubleCodesByStatus request	M
Status of diagnostic trouble codes	M
Group of diagnostic trouble codes	U
ReadDiagnosticTroubleCodesByStatus positive response	S
Number of diagnostic trouble codes	M
List of diagnostic trouble codes and Status	C1
ReadDiagnosticTroubleCodesByStatus negative response	S
Response code	M
Status of diagnostic trouble codes [1)]	U
Group of diagnostic trouble codes [1)]	U/C2
1) Parameters of the request shall be repeated together. C1 = condition: Parameter may be present only if number of DTC is greater than 00. C2 = condition: Parameter may be present only if it was already in the request.	

8.3.3 Service procedure

Upon receiving a ReadDiagnosticTroubleCodesByStatus service indication primitive, the server shall access to the number of diagnostic trouble codes and values which have the same status and then issue a ReadDiagnosticTroubleCodesByStatus service response primitive with the positive response parameters selected. It may be all or a group of diagnostic trouble codes if the optional parameter is present.

If it is not possible to execute the service, the server shall issue a ReadDiagnosticTroubleCodesByStatus service response primitive with the negative response parameters selected.

8.4 ReadStatusOfDiagnosticTroubleCodes service

8.4.1 Service purpose

The purpose of this service is to read the number, values and status of stored diagnostic trouble codes.

8.4.2 Service table

Table 24 defines the ReadStatusOfDiagnosticTroubleCodes service.

Table 24 — ReadStatusOfDiagnosticTroubleCodes service

ReadStatusOfDiagnosticTroubleCodes request	M
Group of diagnostic trouble codes	U
ReadStatusOfDiagnosticTroubleCodes positive response	S
Number Of diagnostic trouble codes	M
List of diagnostic trouble codes and status	M
ReadStatusOfDiagnosticTroubleCodes negative response	S
Response code	M
Group of diagnostic trouble codes	U/C
C = condition : parameter may be present only if it was already in the request.	

8.4.3 Service procedure

Upon receiving a ReadStatusOfDiagnosticTroubleCodes service indication primitive, the server shall access to the number of diagnostic trouble codes, their values and their status and then issue a ReadStatusOfDiagnosticTroubleCodes service response primitive with the positive response parameters selected. It may be all or a group of diagnostic trouble codes if the optional parameter is present.

If it is not possible to execute the service, the server shall issue a ReadStatusOfDiagnosticTroubleCodes service response primitive with the negative response parameters selected.

8.5 ReadFreezeFrameData service

8.5.1 Service purpose

This service is used by the client to access data values which were stored during freeze frame conditions specified by the vehicle manufacturer.

8.5.2 Service table

Table 25 defines the ReadFreezeFrameData service.

Table 25 — ReadFreezeFrameData service

ReadFreezeFrameData request Freeze frame number Record access method identifier [1)] Record identification [1)]	M M U U
ReadFreezeFrameData positive response Freeze frame number Freeze frame data Record access method identifier [2)] Record identification [2)]	S M M U/C U/C
ReadFreezeFrameData negative response Response code Freeze frame number [2)] Record access method identifier [2)] Record identification [2)]	S M U U/C U/C
[1)] Parameters of the request shall only be present together. [2)] Parameters of the request shall only be repeated together. C = condition : parameter may be present only if it was already in the request.	

8.5.3 Service procedure

After receiving a ReadFreezeFrameData indication primitive, the server shall read the requested freeze frame or the specified record within the freeze frame if the optional parameters are present.

If access to the requested data was possible, the server shall deliver a ReadFreezeFrameData response primitive with the Positive response parameters selected.

If access to the requested freeze frame or record within the freeze frame is impossible for any reason, the server shall deliver a ReadFreezeFrameData response primitive with the negative response parameters selected.

NOTE — Data content, data format, and method of retrieval are specified by the vehicle manufacturer. Typical uses for this mode are to report data stored upon detection of a system malfunction. Multiple frames of data may be stored before and/or after the malfunction is detected.

8.6 ClearDiagnosticInformation service

8.6.1 Service purpose

The purpose of this service is to provide a means for the client to command a server to clear stored diagnostic information.

8.6.2 Service table

Table 26 defines the ClearDiagnosticInformation service.

Table 26 — ClearDiagnosticInformation service

ClearDiagnosticInformation request Group of diagnostic Information	M U
ClearDiagnosticInformation positive response Group of diagnostic information	S U/C
ClearDiagnosticInformation negative response Response code Group of diagnostic information	S M U/C
C = condition : parameter may be present only if it was already in the request.	

8.6.3 Service procedure

Upon receiving a ClearDiagnosticInformation indication primitive, the server shall clear all diagnostic information, a group of diagnostic information or a specific diagnostic information if the optional parameter is present. Then the server shall issue a ClearDiagnosticInformation response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a ClearDiagnosticInformation response primitive with the negative response parameters selected.

NOTE — Diagnostic information includes primarily diagnostic trouble codes, but should also include freeze frame data or other on-vehicle test results that may be stored as a result of a diagnostic trouble code being set.

Diagnostic information may be cleared for a single trouble code by including the trouble code to clear as optional parameter in the request. The information stored with an individual trouble code would need to be clearly identified within the module and cleared at the same time.

Exercise caution when clearing by single trouble code: this capability should not be provided and used unless consideration is given to the possible consequences of this option. If a single fault caused multiple trouble codes to be set, and a service technician found a defective component, repaired it, and cleared a single trouble code associated with that fault, then the device would still contain trouble codes that were set based on that fault. The next technician to check codes would read those codes, and if he did not know the vehicle history, he might try to repair the vehicle based only on the remaining trouble codes, which could waste time and result in replacement of good components.

9 InputOutput Control functional unit

9.1 General

Table 27 gives the InputOutput control functional unit.

Table 27 — InputOutput Control functional unit

Service name	Description
InputOutputControlByLocalIdentifier	The client requests the control of an input/output specific to the Server
InputOutputControlByCommonIdentifier	The client requests the control of a common input/output
NOTE — It is recalled that: - the client is the function of the tester that makes use of the diagnostic services; - the server is the function of the on-vehicle ECU that provides the diagnostic services.	

The services of this functional unit make use of the following parameters:

Control option
This parameter is used by the InputOutputControlBy LocalIdentifier / CommonIdentifier services to describe how the server has to control its inputs or outputs.

Control status
This parameter is used by the InputOutputControlBy LocalIdentifier / CommonIdentifier services to give to the client information about the status of the requested control.

9.2 InputOutputControlByLocalIdentifier service

9.2.1 Service purpose

The purpose of this service is to request the control of an input or output peripheral to the server. The request for the peripheral is by local identifier.

9.2.2 Service table

Table 28 defines the InputOutputControlByLocalIdentifier service.

Table 28 — InputOutputControlByLocalIdentifier service

InputOutputControlByLocalIdentifier request	M
InputOutput local identifier	M
Control option	U
InputOutputControlByLocalIdentifier positive response	S
InputOutput local identifier	M
Control status	U
InputOutputControlByLocalIdentifier negative response	S
Response code	M
InputOutput local identifier [1)]	U
Control option [1)]	U/C
[1)] Parameters of the request shall be repeated together. C = condition : parameter may be present only if it was already in the request.	

9.2.3 Service procedure

Upon receiving an InputOutputControlByLocalIdentifier indication primitive, the server shall perform the requested control. Then the server shall issue an InputOutputControlByLocalIdentifier response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue a InputOutputControlByLocalIdentifier response primitive with the negative response parameters selected.

EXAMPLE — The control option parameter may be implemented as a single ON/OFF parameter, or as a more complex sequence of control parameters including a number of cycles, a duration, etc. It may also be a substitute value for an input.

9.3 InputOutputControlByCommonIdentifier service

9.3.1 Service purpose

The purpose of this service is to request the control of an input or output peripheral to the server. The request for the peripheral is by common identifier.

9.3.2 Service table

Table 29 defines the InputOutputControlByCommonIdentifier service.

Table 29 — InputOutputControlByCommonIdentifier service

InputOutputControlByCommonIdentifier request InputOutput common identifier Control option	M M U
InputOutputControlByCommonIdentifier positive response InputOutput common identifier Control status	S M U
InputOutputControlByCommonIdentifier negative response Response code InputOutput common identifier [1)] Control option [1)]	S M U U/C
[1)] Parameters of the request shall be repeated together C = condition : parameter may be present only if it was already in the request.	

9.3.3 Service procedure

Upon receiving an InputOutputControlByCommonIdentifier indication primitive, the server shall perform the requested control. Then the server shall issue an InputOutputControlByCommonIdentifier response primitive with the positive response parameters selected.

If it is not possible to execute the service, the server shall issue an InputOutputControlByCommonIdentifier response primitive with the negative response parameters selected.

EXAMPLE - The Control Option parameter may be implemented as a single ON/OFF parameter, or as a more complex sequence of control parameters including a number of cycles, a duration, etc. It may also be a substitute value for an input.

10 Remote activation of routine functional unit

10.1 General

Table 30 gives the remote activation of routine functional unit.

Table 30 — Remote activation of routine functional unit

Service name	Description
StartRoutineByLocalIdentifier	The client requests to start a routine in the ECU of the server
StartRoutineByAddress	The client requests to start a routine in the ECU of the server
StopRoutineByLocalIdentifier	The client requests to stop a routine in the ECU of the server
StopRoutineByAddress	The client requests to stop a routine in the ECU of the server
RequestRoutineResultsByLocalIdentifier	The client requests the results of a routine by the routine local identifier
RequestRoutineResultsByAddress	The client requests the results of a routine by the routine address
NOTE — It is recalled that: - the client is the function of the tester that makes use of the diagnostic services; - the server is the function of the on-vehicle ECU that provides the diagnostic services.	

EXAMPLES — The client uses a StartRoutineByLocalIdentifier service to start a routine. The server reports that the routine has started with the response primitive and runs the routine until a StopRoutineByLocalIdentifier service is issued to stop the routine. The server informs the client that the routine has stopped with the response primitive. The client then requests routine results with a RequestRoutineResultsByLocalIdentifier service, and the server reports the results with the response primitive.

The services of this functional unit make use of the following parameters:

Routine Entry Option
This parameter is used by the StartRoutineBy LocalIdentifier / Address request primitives to specify the start conditions of the routine.

Routine Entry Status
This parameter is used by the StartRoutineBy LocalIdentifier / Address positive response primitives to give to the client additional information about the status of the server following the start of the routine.

Routine Exit Option
This parameter is used by the StopRoutineBy LocalIdentifier / Address request primitives to specify the stop conditions of the routine.

Routine Exit Status
This parameter is used by the StopRoutineBy LocalIdentifier / Address positive response primitives to give to the client additional information about the status of the server following the stop of the routine. The possible values are:

— Normal exit with results available: this positive response code is used to indicate that the requested exit action will be taken, or is already complete, and that additional information is available in the form of data, stored codes, or PIDs;

— Normal exit without results available: this positive response code is used to indicate that the requested exit action will be taken, or is already complete, but no additional information is available in the form of data, stored codes, or PIDs;

— Manufacturer-specific: this positive response code is manufacturer-defined;

— Abnormal exit with results: this response code is used to indicate that the requested exit action will be taken, or is already complete. It also indicates that the requested execution did not occur in the expected manner and that additional information is available in the form of data, stored codes, or PIDs;

— Abnormal exit without results: this response code is used to indicate that the requested exit action will be taken, or is already complete. It also indicates that the requested execution did not occur in the expected manner, but additional information is not available.

Routine Results
This parameter is used by the RequestRoutineResultsBy LocalIdentifier/Address positive response primitives to give to the Client the results of the routine.

EXAMPLE — Reported results may consist of any set of return information, such as measured test values or fault codes.

10.2 StartRoutineByLocalIdentifier service

10.2.1 Service purpose

The purpose of this service is to request the start of the execution of a routine which is resident in a server. The routines are known by the client and identified by a routine local identifier.