

SUPERVISOR: Mr. ANTONIOLI Bruno D.T. – F.V. – S.I.E.E. – S.S.E. tel. 6834766

MANAGER: Mr. COLOMBERO Flavio D.T. – F.V. – S.I.E.E. – S.S.E. tel. 6835423

PURPOSE

This document describes Fiat requirements for the implementation of Keyword Protocol 2000 (KWP2000) services on CAN, defining the general rules.

Change	Date	Description
=	Oct. '98	Edition 1 – New
=	July '99	Edition 2 – Changed the following paragraphs: <ul style="list-style-type: none"> • par.4.3.7. time schedule specifications added • par. 4.3.7.1. summary table Transport Protocol characteristics added • par. 5.1. “slow” and “fast” corrected in “lento” and “veloce” respectively • par. 5.2.2. clarification sentence on responseCode 78h added • par. 5.2.2. summary table responseCodes recommended/used added • par. 6.1.2.1. new diagnostic modules added • par. 6.3. summary table services requiring securityAccess added • par. 6.3.2.2. (key length in case of download) removed • par. 6.6.2.1. table corrected (italian terms translated in english, TBD deleted) • par. 7.0. ortographic error corrected • par. 8.2.2.4. table added for max DTC number insertion and max blocks to be memorised by ECU • par.8.3.1. positive answer description for possible contradiction elimination reviewed • par. 10.1.2.1. RELI summary table and download reserved RELI description changed • par. 6.1.1. negative answer description tables changed • par. A.2.1. “receiving” changed in “transmission” • par. A.2.2. changed action on event S7

(continued on the next page)

ANY PRINTED COPY IS TO BE DEEMED AS UNCHECKED; THEREFORE THE UPDATED COPY MUST BE CHECKED IN THE APPROPRIATE WEB SITE



= 05/11/00

Edition 3 – Changed the following paragraphs:

- par. 4.3.6.6. FCS parameter management specified
- par. 4.3.7.1. clarification on ECU behaviour added
- par. 4.7. conditions for startCommunication/stopCommunication changed
- par. 5.2.2. table responseCodes changed for 11h and 10h
- par. 6.0. ECU behaviour outside diagnostic sessions specified
- par. 6.1.2.1. diagnostic session for starting components 89h introduced
- par. 6.1.2.2. “reject” changed into “ignore”
- par. 7.1.2.1. RLI 80h-9Fh management changed
- par A.2.1. “Receiver” changed into “Sender”

Contents

1	PURPOSE	6
2	REFERENCES	7
3	DEFINITIONS AND ABBREVIATIONS	8
3.1	Service Identifier	8
4	COMMUNICATION	9
4.1	Physical Layer	9
4.2	Physical Topology	9
4.3	Transport Protocol	10
4.3.1	Overview	
4.3.2	Message segmentation	12
4.3.3	Flow control	12
4.3.4	Mapping of KWP2000 messages	13
4.3.5	Transport Protocol Data Unit	13
4.3.5.1	TPDU format	14
4.3.5.2	Target Address (TA)	14
4.3.5.3	Transport Protocol Control Information (TPCI)	14
4.3.5.4	DATA[] (Data field)	14
4.3.5.5	Data Length (DL)	14
4.3.5.6	Block Size (BS)	15
4.3.5.7	Separation Time (ST)	15
4.3.6	Transport Protocol Control Information format	15
4.3.6.1	Operation Code (OC)	15
4.3.6.2	Data Length (DL)	16
4.3.6.3	Reserved	16
4.3.6.4	Extended Data Length (XDL)	16
4.3.6.5	Segment Number (SN)	16
4.3.6.6	Flow Control Status (FCS)	16
4.3.7	Configuration and Usage	17
4.3.7.1	ECU Configuration	18
4.3.7.2	Tester Configuration	19
4.3.8	Message flow examples	20
4.3.8.1	Single frame - physical address	20
4.3.8.2	Segmented transfer - physical address	21
4.4	ECU and Tester physical addresses	22
4.5	Timing	23
4.6	Communication Services	25
4.6.1	StartCommunication service	25
4.6.2	StopCommunication service	25
4.6.3	AccessTimingParameters service	25
4.7	Protocol Structure	26
5	IMPLEMENTATION	27
5.0	Introduction	27
5.1	Service Identifiers	27
5.2	negative Response	28
5.2.1	Message data bytes	28
5.2.2	Parameters definition	28
6	DESCRIPTION OF DIAGNOSTIC MANAGEMENT FUNCTIONAL UNIT	32
6.0	Introduction	32
6.1	startDiagnosticSession service	33
6.1.1	Message data bytes	33
6.1.2	Parameters definition	33
6.1.2.1	diagnosticMode	33
6.1.2.2	baudrateIdentifier	33
6.1.2.3	diagnosticModes and diagnosticServices availability table	34

6.2	stopDiagnosticSession service.....	35
6.2.1	Message data bytes	35
6.3	securityAccess Service	36
6.3.1	Message data bytes	38
6.3.2	Parameters definition.....	39
6.3.2.1	accessMode	39
6.4	TesterPresent service.....	40
6.4.1	Message data bytes	40
6.5	ecuReset Service	40
6.6	readEcuIdentification Service	41
6.6.1	Message data bytes	41
6.6.2	Parameters definition.....	42
6.6.2.1	identificationOption	42
6.6.2.2	ISO Code value summary table (identificationOption = 97h)	43
7	DESCRIPTION OF DATA TRANSMISSION FUNCTIONAL UNIT	44
7.0	Introduction	44
7.1	readDataByLocalIdentifier Service	44
7.1.1	Message data bytes	44
7.1.2	Parameters definition.....	45
7.1.2.1	recordLocalIdentifier	45
7.2	readDataByCommonIdentifier Service	46
7.3	readMemoryByAddress Service	46
7.4	dinamicallyDefineLocalIdentifier Service	46
7.5	writeDataByLocalIdentifier Service	47
7.5.1	Message data bytes	47
7.6	writeDataByCommonIdentifier Service.....	47
7.7	writeMemoryByAddress Service	47
8	DESCRIPTION OF STORED DATA TRANSMISSION FUNCTIONAL UNIT	48
8.0	Introduction	48
8.1	readDiagnosticTroubleCodes Service.....	48
8.2	readDiagnosticTroubleCodesByStatus Service	49
8.2.1	Message data bytes	49
8.2.2	Parameters definition.....	50
8.2.2.1	groupOfDTC	50
8.2.2.2	numberOfDTC	50
8.2.2.3	Error Memory structure and management	50
8.2.2.4	DTC and environmentalConditions tables.....	51
8.2.2.5	statusOfDTC.....	53
8.3	readStatusOfDiagnosticTroubleCodes Service.....	54
8.3.1	Message data bytes	54
8.4	readFreezeFrameData Service	55
8.4.1	Message data bytes	55
8.5	clearDiagnosticInformation Service.....	56
8.5.1	Message data bytes	56
9	DESCRIPTION OF INPUT/OUTPUT CONTROL FUNCTIONAL UNIT	57
9.0	Introduction	57
9.1	inputOutputControlByLocalIdentifier Service	57
9.1.1	Message data bytes	57
9.1.2	Parameters definitions.....	58
9.1.2.1	inputOutputControlParameter	58
9.1.2.2	inputOutputControlState value summary table	58
9.1.2.3	inputOutputLocalIdentifier	59
9.1.2.4	General rules for the activation/deactivation of a component	59
9.2	inputOutputControlByCommonIdentifier Service	59
10	DESCRIPTION OF ROUTINE REMOTE ACTIVATION OF FUNCTIONAL UNIT	60
10.0	Introduction	60
10.1	startRoutineByLocalIdentifier Service	60
10.1.1	Message data bytes	60
10.1.2	Parameters definition.....	60
10.1.2.1	routineLocalIdentifier	60

10.1.2.2.	routineEntryOption.....	61
10.2.	startRoutineByAddress Service.....	61
10.3.	stopRoutineByLocalIdentifier Service.....	62
10.3.1.	Message data bytes	62
10.4.	stopRoutineByAddress Service.....	62
10.5.	requestRoutineResultsByLocalIdentifier Service	63
10.5.1.	Message data bytes	63
10.5.2.	Parameters definition.....	64
10.6.	requestRoutineResultsByAddress Service.....	64
11.	DESCRIPTION OF UPLOAD DOWNLOAD FUNCTIONAL UNIT	65
11.0.	Introduction	65
11.1.	requestDownload Service	65
11.1.1.	Message data bytes	65
11.1.2.	Parameters definition.....	65
11.1.2.1.	dataFormatIdentifier	65
11.2.	requestUpload Service	66
11.3.	transferData Service.....	67
11.3.1.	Message data bytes	67
11.4.	requestTransferExit Service	68
11.4.1.	Message data bytes	68
APPENDIX A	69
A.	Transport Protocol implementation specifications.....	69
A.1.	Transport Protocol Receiver Entity.....	69
A.1.1.	Service primitives	69
A.1.2.	State Transition Table	70
A.1.3.	Actions description	71
A.2.	Transport Protocol Sender Entity	72
A.2.1.	Service primitives	72
A.2.2.	State Transition Table	73
A.2.3.	Actions description	74
A.2.4.	Events description	74
A.3.	Timing	75
A.3.1.	Receiver TPE timing.....	75
A.3.2.	Sender TPE timing	75
A.4.	Constants	75

1

PURPOSE

This document describes FIAT requirements for the implementation of Keyword Protocol 2000 (KWP2000) services on CAN, defining the general rules. It applies to all the ECUs that use the CAN bus as a diagnostic link to a Tester directly or indirectly through a gateway.

This document does not contain the definition of the protocol for the SCAN TOOL (ISO 14230-4).

For security reasons, some implementation parts will not be described in any public document; they rather will be specified and discussed directly by the people in charge of the relevant aspects.

All the parameter values which are strictly ECU-dependent will not be assigned in this document and they will be referred to as TBD.

The "FIAT Standard Diagnostic Protocol on CAN" document shall be used as a template to complete the definition of all the diagnostic services. Using this document as a reference, a dedicated document named "Specifica Finalizzata di Diagnosi" shall be written for any ECU using this protocol.

2

REFERENCES

The following list contains the documents used as reference:

ISO standards

- /1/ ISO 14230 - Part 2 Keyword Protocol 2000 - Communication
- /2/ ISO 14230 - Part 3 Keyword Protocol 2000 - Implementation
- /3/ ISO 11519 Road vehicles - Low-speed serial data communication - Part 2: Low-speed controller area network
- /4/ ISO 11898 - Road vehicles - Interchange of digital information - Controller area network for high speed communication
- /5/ Proposal for Transport Layer Protocol for Diagnostics on CAN - document N61, April 22 1997 ISO/TC22/SC3/WG1/TF2

Other documents

- /6/ Keyword Protocol 2000, Implementation of Diagnostic Services, Recommended Practice - version 1.4, September 16, 1997.
- /7/ SAE J 2012 Diagnostic Trouble Codes Definitions

3

DEFINITIONS AND ABBREVIATIONS

3.1

Service Identifier

English	Italian
startCommunication	Inizio comunicazione
stopCommunication	Termine comunicazione
accessTimingParameter	Accesso ai parametri di comunicazione
startDiagnosticSession	Inizio sessione diagnostica
stopDiagnosticSession	Termine sessione diagnostica
securityAccess	Accesso alle procedure di sicurezza
testerPresent	Blocco di mantenimento comunicazione
readECUIdentification	Lettura codice identificativo ECU
readDataByLocalIdentifier	Lettura parametri
writeDataByLocalIdentifier	Scrittura parametri
readDiagnosticTroubleCodesByStatus	Lettura codici errori presenti in centralina raggruppati tramite il loro stato
readStatusOfDiagnosticTroubleCodes	Lettura parametri ambientali collegati ad un singolo errore
readFreezeFrameData	Lettura blocco informazioni catturate durante il rilevamento di anomalie rilevanti per le emissioni
clearDiagnosticInformation	Cancellazione memoria errori
inputOutputControlByLocalIdentifier	Operazioni effettuate su attuatori (attivazioni , tarature,etc.)
startRoutineByLocalIdentifier	Attivazione programma residente
stopRoutineByLocalIdentifier	Interruzione programma residente attivato in precedenza
requestRoutineResulByLocalIdentifier	Informazioni sullo stato programma (in esecuzione) attivato in precedenza
requestDownload	Richiesta di programmazione
transferData	Comando utilizzato per trasferire i dati da programmare
requestTransferExit	Richiesta chiusura procedura di programmazione

4**COMMUNICATION****4.1****Physical Layer**

The physical layer is not within the scope of this document.

This document applies to the ECUs that use the CAN bus as diagnostic link to the Tester directly or indirectly through a gateway. The physical layer of the protocol shall be compliant to the CAN standards */4/* and */3/*.

The CAN bus characteristics will not be described in this document; since they obviously are project dependent. For any ECU the project specifications shall be used as the right reference.

4.2**Physical Topology**

The physical topology is not within the scope of this document, please refer to */4/* , */3/*.and the project specifications.

4.3

Transport protocol

The FIAT Standard Diagnostic Protocol on CAN adopts the services and the message structure of the KWP2000 standard as defined in */1/* and */2/*. The KWP2000 messages however can be longer than what can be comprised in a single CAN frame (max. 8 data bytes). Moreover, some fields defined by KWP2000 are redundant when using the CAN Data Link Layer (e.g.: the source address can be encoded in the CAN identifier field; the KWP2000 checksum byte can be replaced by the CRC CAN field).

A transport layer shall then be implemented either in the Tester and in the ECU to allow the correct mapping and segmentation of the diagnostic messages into valid CAN frames.

The following chapters contain the description of the Transport Protocol used in FIAT KWP2000 on CAN. In *Appendix A* the reader will find the Transport Protocol implementation specifications.

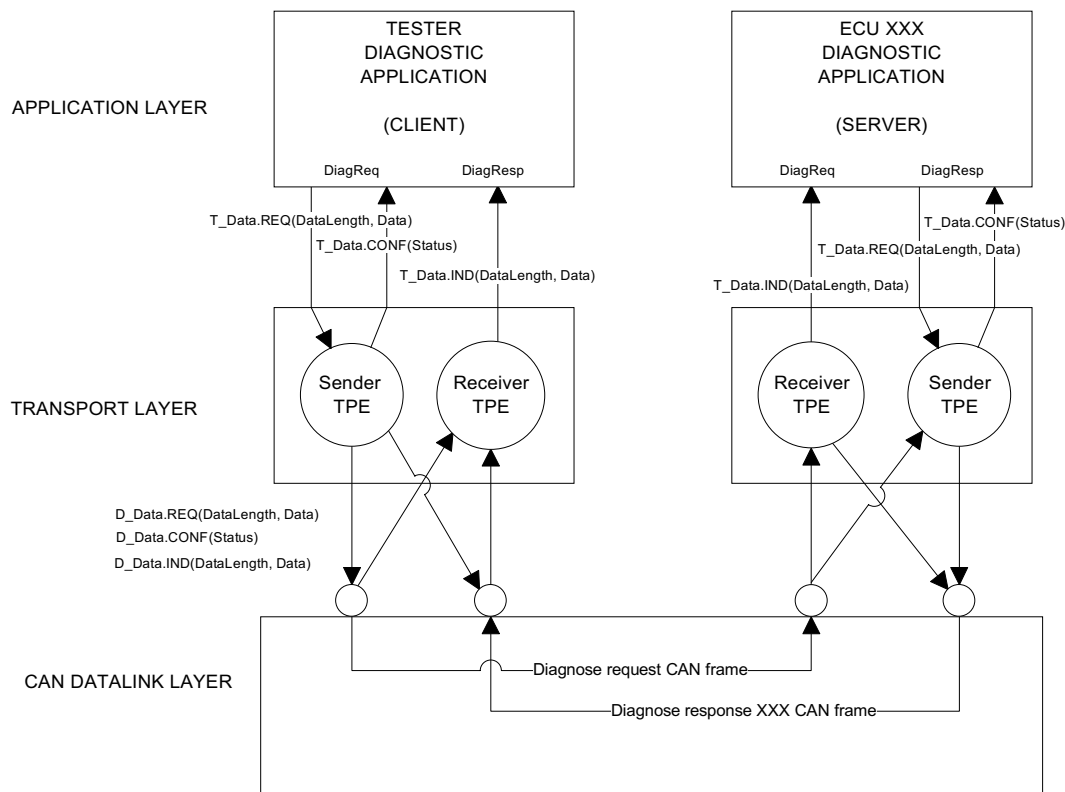
4.3.1

Overview

The Transport Protocol specified in this document is based on the current proposal from the ISO/TC22/SC3/WG1/TF2 group for Diagnostics on CAN */5/*. However, only a subset of the functionalities defined in the ISO proposal has been specified in this document, i.e. functional addressing capability and transfer of messages longer than 255 byte are not supported since this will not be required for the diagnosis of the ECUs currently defined.

The diagnostic messages are segmented by the sending Transport Protocol Entity into several CAN frames, each containing up to six bytes of diagnostic data, (except for the first frame that contain five bytes of data only) which are transmitted consecutively on the CAN bus. The receiving Transport Protocol Entity is then reassembling the segments into the original diagnostic message before it is presented to the receiving Diagnostic Application to be processed.

The figure below visualises the basic relations between Diagnostic Applications, the Transport Layer with the Transport Protocol Entities, and the CAN Data Link Layer.



The Diagnostic Application in the Tester prepares a DiagReq message and requests the Sender TPE to transfer the message using the T_Data.REQ service primitive.

The Sender TPE in the Tester segments the messages into smaller data-packets (if necessary) and sends the message segment by segment to the Receiver TPE in the ECU that shall be diagnosed. The DIAGNOSE REQUEST CAN frame (defined according to the project specifications) is used for this transfer.

If the message is segmented, the Receiver TPE in the ECU has a possibility to control the message flow during the transfer by sending back Flow Control messages to the sender TPE, using the DIAGNOSE RESPONSE CAN Frame.

When the complete message has been received by the Receiver TPE in the ECU, the DiagReq message is presented to the ECU Diagnostic Application by the T_Data.IND service primitive.

The ECU Diagnostic Application performs the requested service and prepares the DiagResp message to be sent back to the Tester.

4.3.2**Message segmentation**

The message requested for transmission by the diagnostic application is, if required, segmented by the Sender Transport Protocol Entity into data packages that can be transmitted in one CAN frame.

The CAN frame has a fixed length of 8 bytes. This means that not all bytes are used in every CAN frame. Not used bytes shall be padded with zeroes.

In each CAN frame the first byte (TA, Target Address) is reserved for address information and the second byte (TPCI, Transport Protocol Control Information) is reserved for protocol information.

In the first CAN frame in a segmented transfer, one byte is also reserved for data length information.

4.3.3**Flow control**

In some networks it may be required for a receiver of segmented diagnostic messages to limit the flow of CAN frames (e.g. due to limitations in buffer size or different transmission speeds on different networks).

The transport protocol provides two ways for the receiving node (Tester/ECU) to control the message flow; minimum time between segments (ST, Separation Time) and maximum number of segments (BS, Block Size) according to the following definitions:

Max. # segment: $1 \leq BS \leq 254$; maximum number of segments transmitted before a new Flow Control message shall be transmitted from the receiver.
 BS = 255; no limit of number of segments transmitted.

Min time: ST = minimum time [ms] between two consecutive CAN frames within a segmented transmission (ST = 0; no limit).

A Tester/ECU may use either one, both or none of the two techniques.

FIAT requirement is that any ECU shall set BS = 255 and ST = 0 (see 4.3.7.1). This configuration can be changed if required by ECU Supplier only after agreement with FIAT.

Each time a segmented transmission is started the transmitting Tester/ECU may only send the first segment after which it shall wait for a "confirmation to send" message from the receiving node. The confirmation to send message shall contain information about the minimum time between consecutive segments and the maximum number of segments before a new confirmation to send message shall be sent.

When the transmitting Tester/ECU has received the confirmation to send message it may send as many consecutive frames as have been indicated by the receiver after which it shall wait for a new confirmation to send message.

The values of BS and ST, transferred in the first "confirmation to send" message, is used throughout the remaining part of the segmented transmission, i.e. the BS and ST parameters in the following confirmation to send messages (within the same segmented transmission) are treated as "don't care" values. The values of BS and ST can only be changed when a new segmented transmission is started.

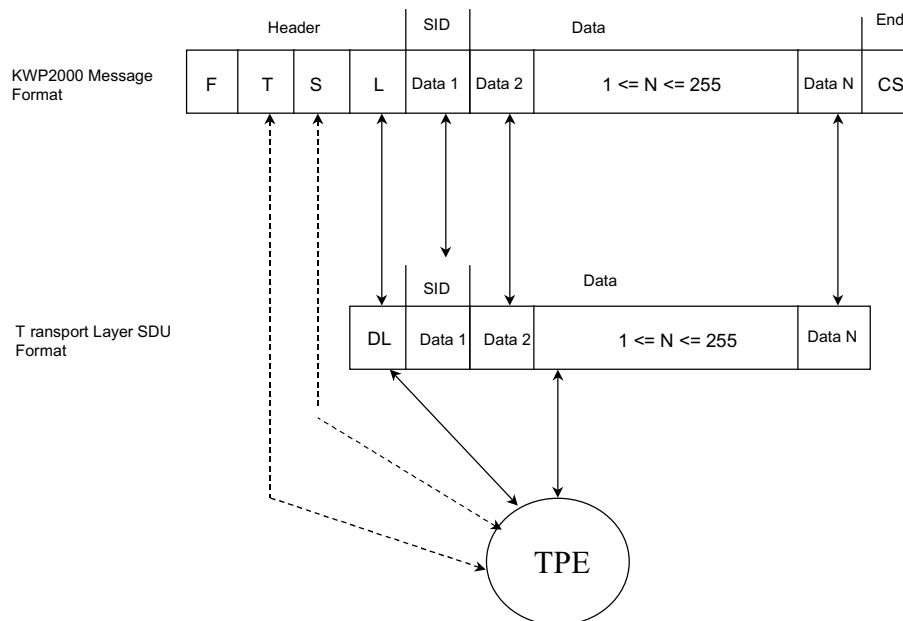
The "confirmation to send" message is always sent after the first segment in a segmented transfer even if the receiving Tester/ECU does not require any flow control. In this case the receiving Tester/ECU shall set BS = 255 and ST = 0 ms.

The receiving Tester/ECU shall send a new confirmation to send message only when a number of segments equal to the number indicated by BS have been received and after the Tester/ECU are ready to receive more segment frames, or no confirmation message is sent at the end of transmission.

4.3.4

Mapping of KWP2000 messages

The mapping of the diagnostic messages defined in KWP2000 shall be according to the figure below.



The Format Byte (F) and the Checksum Byte (CS) in the KWP2000 message shall not be transferred at all in the new message structure.

The Target Address (T) and the Source Address (S) in the KWP2000 message are handled by the Transport Protocol. The Source address is encoded within the CAN Identifier.

The Data Length information (L) in the KWP2000 message shall be transferred by the DL parameter.

The Data field (DATA[]) starting with the diagnostic Service Identifier (SID) is byte by byte mapped into the Data field (DATA[]) of the new message structure.

The used CAN frames have a fixed length of 8 bytes, not used bytes shall be padded with zeroes.

4.3.5

Transport Protocol Data Unit

The Transport Protocol Data Unit (TPDU) consists of Protocol Control Information, Target Address Information and Data.

The Protocol Control Information is encoded in the TPCI-, DL-, BS- and ST-bytes and serves the following purpose:

- coding of different TPDU types for representation of transfer services
- providing flags and counters for controlling the data flow

4.3.5.1**TPDU format**

The different TPDU's are coded as follows:

		TPDU							
TPDU type	Byte	#1	#2	#3	#4	#5	#6	#7	#8
Single Frame (SF)		TA	TPCI	DATA[0..5]					
First Frame (FF)		TA	TPCI	DL	DATA[0..4]				
Consecutive Frame (CF)		TA	TPCI	DATA[0..5]					
Flow Control (FC)		TA	TPCI	BS	ST	-	-	-	-

The format of the different TPDU's contains the following parameters:

4.3.5.2**Target Address (TA)**

The Tester and the ECUs are all given a unique physical diagnostic address. The *Target Address* byte indicates the intended receiver of the diagnostic message.

For client (Tester) request messages the *Target Address* is always the physical address of the server (ECU). For server (ECU) response messages the *Target Address* is always the physical address of the client (Tester).

The Target Address is referred to as TPDU.TA in the protocol specification.

4.3.5.3**Transport Protocol Control Information (TPCI)**

See § 4.3.6.

4.3.5.4**DATA[] (Data field)**

Data bytes used for diagnostic information.

The data field DATA[] is referred to as TPDU.DATA[] in the protocol specification.

4.3.5.5**Data Length (DL)**

The "*Data Length*" field indicates the total number of bytes in the complete message. Depending on the TPDU type the Data Length parameter can be a 3 or an 8 bit number.

The 8 bit "*Data Length*" parameter is referred to as TPDU.DL in the protocol specification. For information about the 3 bit Data Length see § 4.3.6.

4.3.5.6

Block Size (BS)

The *Block Size*, with value [1..255], is used in FC TPDUs to indicate, to the transmitting entity, the number of data segments (CF TPDUs) to send before expecting an FC TPDU from the receiving entity.

The value 255 indicates that no further FC TPDUs will be used by the receiver for flow control.

The Block Size parameter is referred to as TPDU.BS in the protocol specifications.

4.3.5.7

Separation Time (ST)

The *Separation Time*, with value [0..255], is used in FC TPDUs to indicate the receivers requirement on minimum time, in milliseconds, between consecutive CF TPDUs sent from the transmitting entity.

The Separation Time parameter is referred to as TPDU.ST in the protocol specifications.

4.3.6

Transport Protocol Control Information format

The Transport Protocol Control Information (TPCI) byte shall be coded as follows:

		TPCI byte							
TPDU type	Bit	#7	#6	#5	#4	#3	#2	#1	#0
Single Frame (SF)		OC = 0000b				R	DL		
First Frame (FF)		OC = 0001b				XDL			
Consecutive Frame (CF)		OC = 0010b				SN			
Flow Control (FC)		OC = 0011b				FCS			

The coding of the TPCI byte contains the following parameters:

4.3.6.1

Operation Code (OC)

The *Operation Code* is used to define the type of the TPDU. Four different TPDUs are defined:

SF TPDU: OC = 0000b
FF TPDU: OC = 0001b
CF TPDU: OC = 0010b
FC TPDU: OC = 0011b

The Operation Code parameter is referred to as TPDU.TPCI.OC in the protocol specification.

4.3.6.2**Data Length (DL)**

The 3 bit *Data Length* information is used in SF TPDU's to indicate how many bytes in the SF TPDU are used for diagnostic information.

The 3 bit Data Length parameter is referred to as TPDU.TPCI.DL in the protocol specification.

Note: All TPDU's have a fixed length of 8 bytes.

4.3.6.3**Reserved**

This bit is currently not used and is reserved for future use. It shall be set to 0 by the sending protocol entity and be checked to be 0 by the receiving protocol entity.

The Reserved bit parameter is referred to as TPDU.TPCI.R in the protocol specification.

4.3.6.4**Extended Data Length (XDL)**

For segmented messages a 12-bit Data Length value is encoded as follows:

Bit 0..7 in a separate data length byte following the TA byte of the first frame (TPDU.DL).
Bit 8..11 encoded inside the TPCI byte of the First Frame using the XDL field.

The 4 bit Extended Data Length parameter is referred to as TPDU.TPCI.XDL in the protocol specification. This parameter shall be set to 0 in the FIAT actual implementation.

4.3.6.5**Segment Number (SN)**

The *Segment Number*, with value [0..15], is used in CF TPDU's to provide error detection capabilities (lost or duplicated segments) at the receiving entity. Lost or duplicated segments are recognised by evaluating the SN field. The first CF TPDU in a segmented message shall have SN = 0. After receiving a CF TPDU with SN = n the receiving entity shall expect a new CF TPDU with SN = (n+1) MOD 16 (unless it was the last CF TPDU in the message).

The Segment Number parameter is referred to by TPDU.TPCI.SN in the protocol specification.

4.3.6.6**Flow Control Status (FCS)**

The FCS field is currently not used by the transport protocol, and is reserved for future use. It shall be set to 0 by the sending protocol entity. No check by the Receiver Protocol Entity is envisaged.

The Flow Control Status parameter is referred to as TPDU.TPCI.FCS in the protocol specification.

4.3.7**Configuration and Usage**

Different reaction and timeout times of each single Transport Protocol Entity are specified below:

First Frame reaction time

Reaction time from receiving of a FF TPDU to transmission request of the following FC TPDU from receiving TPE must be within 0 ms and 30 ms. This reaction time is called FF_FC_ Reaction Time in the protocol specifications.

Flow Control reaction time

Reaction time from receiving of a FC TPDU to transmission request of the following CF TPDU from receiving TPE must be within 0 ms and 30 ms. This reaction time is called FC_CF_ Reaction Time in the protocol specifications.

Separation Time

Reaction time from from Data Link Level confirmation of CF TPDU transmission request occurrence from transmission TPE to the TPE itself request of transmission of the following CF TPDU must be within St_Timeout and ST_Timeout + 30 ms where ST_Timeout shows the separation time value received with the FC TPDU.

Timeout on receiving of 1st Flow Control

Timeout between transmission request of a FF TPDU from transmission TPE and receiving of the following FC TPDU must be within 200 ms and 250 ms. This Timeout is called cCN_Timeout in the protocol specifications.

Timeout on receiving of the following Flow Control

Timeout between transmission request of the last CF TPDU in a block by transmission TPE and receiving of the following FC TPDU must be within 200 ms and 250 ms. This Timeout is called cFC_Timeout in the protocol specifications.

4.3.7.1

ECU Configuration

The ECU shall implement one Sender Transport Protocol Entity to send DiagResp messages to the Tester and one Receiver Transport Protocol Entity to receive DiagReq messages from the Tester.

Block Size

The ECU shall set the BS to 255 in the first FC TPDU indicating that no further FC TPDUs will be sent by the receiver during the reception of the rest of the message.

This parameter can be changed if required by any ECU supplier only after agreement with FIAT.

Separation Time

The ECU shall set the ST to 0 in the first FC TPDU indicating that no limit in minimum time between two consecutive received segments is required.

This parameter can be changed if required by any ECU supplier only after agreement with FIAT.

CAN Identifier

The Identifiers of the CAN frames to be used for the sending and the receiving of the TPDUs are defined in the project specifications.

Receive Buffer size

Each ECU must have buffering capacity to receive a complete DiagReq message from the Tester. The required buffer size varies for the different ECUs and is determined by the longest message that each ECU shall be able to receive. If the buffer size is less than the maximum length that can be handled by the Transport Protocol (255 bytes), the implementation must take precautions i.e. the ECU shall reject a message that is longer than what has been defined as the maximum length for that specific ECU. Therefore, the ECU may not send the FC TPDU, or it may send it and at the end of the data transfer it may either not send any answer to the Tester (which in either case will ascertain a communication timeout) or answer negatively to the request exceeding maximum length allowed.

In case of transfer implemented by the ECU at the end of the line (PROXI) the receiver buffer dimension must be 255 bytes.

The following table must be used to describe the Transport Protocol characteristics of each ECU.

Parameter	Value
Block Size (TPDU.BS)	255
Separation Time (TPDU.ST)	0
CAN identificative for request from Tester	TBD
CAN identificative for answer from ECU	TBD
Receiver buffer dimension	TBD

4.3.7.2**Tester Configuration**

The Tester shall implement one Sender Transport Protocol Entity to send DiagReq messages to the ECU and one Receiver Transport Protocol Entity to receive DiagResp messages from the ECU.

Block Size

The Tester shall set the BS to 255 in the first FC TPDU indicating that no further FC TPDUs will be sent by the receiver during the reception of the rest of the message.

Separation Time

The Tester shall set the ST to 0 in the first FC TPDU indicating that no limit in minimum time between two consecutive received segments is required.

CAN Identifier

The Identifiers of the CAN frames to be used for the sending TPDUs and for the receiving of the TPDUs from the different ECUs are defined in the project specifications.

4.3.8

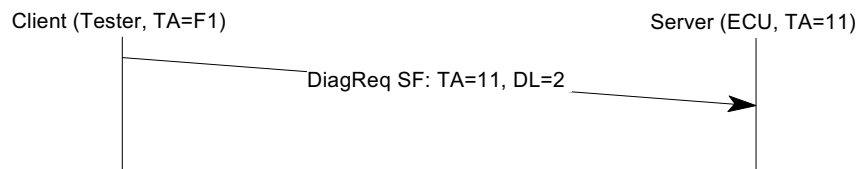
Message flow examples

Two message flow graphs follow that show the message sequences either in case of single frame and segmented data transfer. For more information about the timing parameters please refer to Appendix A.

4.3.8.1

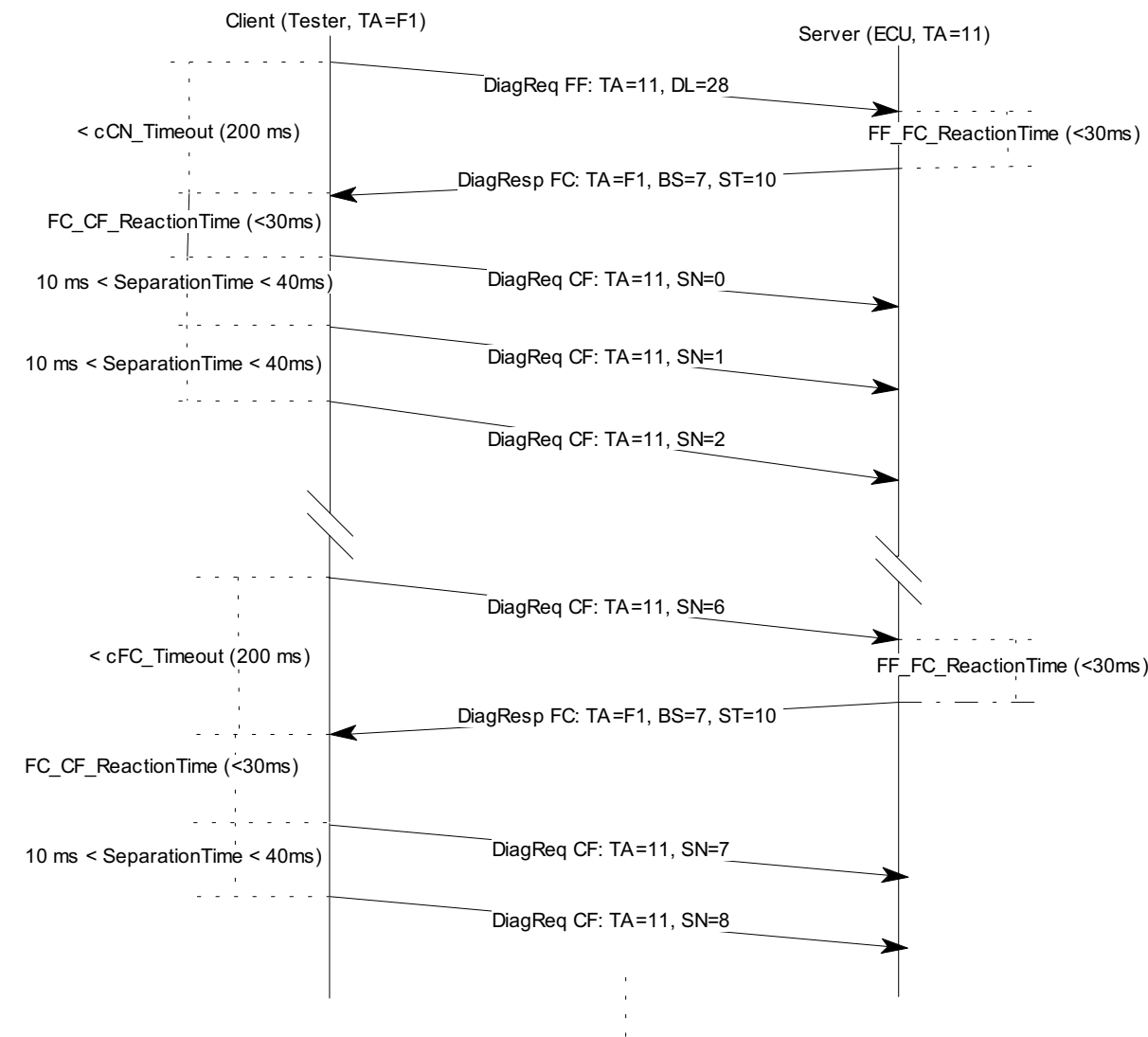
Single frame - physical address

The message flow example below shows a Tester transmitting a physically addressed single frame request:



4.3.8.2
Segmented Transfer - physical address

The message flow example below shows a Tester transmitting a physically addressed segmented request:



4.4

ECU and Tester physical addresses

For Tester request messages the Target Address is always the physical address of the ECU. For ECU response messages the Target Address is always the physical address of the Tester.

The addresses shall be defined by FIAT.

ECU Physical Address (Hex)	Tester Physical Address (Hex)
TBD	F1

4.5

Timing

The timing parameters defined according to KWP2000 standard /1/ are no applicable when using the Transport protocol for diagnostics on CAN. Application time-outs need to be defined, instead.

	Description	Parameter name
TESTER	Time delay introduced by the Tester from the time it recognises that it has completely received a Diagnostic response message (on a previous request), until the Tester itself starts the transmission of the next Diagnostic request message to the ECU.	T_dly_tx
	Time-out set by the Tester starting from the time it recognises that it has completely transmitted a Diagnostic request message, until the Tester itself completes the reception of the corresponding Diagnostic response message coming from the ECU.	T_tmo_rx
ECU	Time-out set by the ECU starting from the time it recognises that it has completely transmitted a Diagnostic response message (on a previous request), until the ECU itself completes the reception of the next Diagnostic request message coming from the Tester.	E_tmo_rx
	Time delay introduced by the ECU from the time it recognises that it has completely received a Diagnostic request message, until the ECU itself starts the transmission of the corresponding Diagnostic response message to the Tester.	E_dly_tx

T_dly_tx and T_tmo_rx parameters are relevant to the Tester while E_tmo_rx and E_dly_tx refer to the ECU.

T_dly_tx and E_dly_tx parameters define transmission time delays of either the Tester and the ECU, while T_tmo_rx and E_tmo_rx are to be considered as reception time-outs.

The time-out values shall be long enough to allow the ECU to complete all its services within the given time-frame. They must also include a margin for the transfer-time of a message on the CAN bus.

However, the diagnostic services can be separated into fast services and slow services.

The fast services are all services where the response messages are shorter or equal than 6 bytes (single segment data transfer), and where the ECU can process and complete the requested service very quickly.

The slow services are all the services where the response messages are longer than 6 bytes (multiple segment data transfer) or when it takes time for the ECU to process and complete the requested service.

For any ECU the grouping of the applicable diagnostic services on either one or the other of the two categories defined above shall be approved by FIAT and it shall be written in the "Specifica Finalizzata di Diagnosi" document relevant to such ECU.

The startDiagnosticSession and testerPresent are examples of Fast services.

The following table shows the suggested settings for all the timing parameters defined above. ECU suppliers shall declare the E_tmo_rx and E_dly_tx values used in their own systems. These values shall be approved by FIAT and they shall be written in the corresponding "Specifica Finalizzata di Diagnosi" document relevant to ECU.

Parameter name	Service type		Tester side settings (s)	ECU side settings (s)
	Slow	Fast		
T_dly_tx	✓	✓	≤5.0	
T_tmo_rx		✓	1.0	
	✓		10.0	
E_tmo_rx	✓	✓		10.0
E_dly_tx		✓		≤0.1
	✓			≤5.0

4.6**Communication Services**

The following chapters contain a detailed description of the Data field for all the communication services; all other fields are defined by the Transport Protocol specifications (in this section) and the CAN standards /4/ and /3/.

A single diagnostic service message can be segmented by the Transport Protocol into several data packages so that each of them can be transmitted in one CAN frame. The CAN frames have a fixed length of 8 bytes; not used bytes shall be padded with zeroes. In both single-frame and segmented data transfer, not used bytes shall be confined in the last transmitted portion of the data field.

4.6.1**startCommunication service**

startCommunication service shall not be implemented using a CAN bus for the diagnostic communication between the Tester and the ECU.

4.6.2**stopCommunication service**

stopCommunication service shall not be implemented using a CAN bus for the diagnostic communication between the Tester and the ECU.

4.6.3**accessTimingParameters service**

accessTimingParameters service shall not be implemented using a CAN bus for the diagnostic communication between the Tester and the ECU.

4.7

Protocol Structure

The communication can only start under the following conditions:

- +15 ON (from discrete wire or CAN signal)

The communication shall be maintained active by exchanging request/response frames or by using the testerPresent service.

After a time-out, the ECU shall return to its normal mode of operation. The communication shall be re-established by the Tester with a new startDiagnosticSession request and in the meanwhile, any pending service shall be aborted as well.

If a new startDiagnosticSession request is sent by the Tester while a diagnostic session is already active (e.g. to change diagnosticMode), the new session shall start after the ECU has aborted any pending service in the current diagnostic session.

The diagnostic transmission shall end when at least one of the following conditions is satisfied:

- +15 OFF (except for particular cases, e.g.: Body Computer)
- there is a time-out in the communication.
- the Tester has sent a stopDiagnosticSession request to stop the default session that was enabled at communication startup.

If the diagnostic session or the diagnostic communication is stopped during the activation of an ECU component (inputOutputControlByLocalIdentifier) the ECU must automatically take complete control of the external component to avoid any possible damage.

5 IMPLEMENTATION

5.0

Introduction

The following chapters contain a detailed description of the Data field for all the applicable services; all other fields are defined by the Transport Protocol (see 4. in this document) and the CAN standards /4/ and /3/.

A single diagnostic service message can be segmented by the Transport Protocol into several data packages so that each of them can be transmitted in one CAN frame. The CAN frames have a fixed length of 8 bytes; not used bytes shall be padded with zeroes. In both single-frame and segmented data transfer, not used bytes shall be confined in the last transmitted portion of the data field.

5.1

Service Identifiers

The following table lists all the diagnostic services available in FIAT implementation of KWP2000 on CAN.

Diagnostic Service Name	Request Hex Value	Response Hex Value	Service Type
startDiagnosticSession	10	50	fast
stopDiagnosticSession	20	60	fast
securityAccess	27	67	fast
testerPresent	3E	7E	fast
readECUIdentification	1A	5A	(¹)
readDataByLocalIdentifier	21	61	(¹)
writeDataByLocalIdentifier	3B	7B	(²)
readDiagnosticTroubleCodesByStatus	18	58	(¹)
readStatusOfDiagnosticTroubleCodes	17	57	(¹)
readFreezeFrameData	12	52	(¹)
clearDiagnosticInformation	14	54	(²)
inputOutputControlByLocalIdentifier	30	70	(¹)
startRoutineByLocalIdentifier	31	71	fast
stopRoutineByLocalIdentifier	32	72	fast
requestRoutineResultByLocalIdentifier	33	73	(¹)
requestDownload	34	74	fast
transferData	36	76	fast
requestTransferExit	37	77	fast

- (¹) The service type (fast/slow) depends on the positive response message length, if the positive response message exceeds the 6 byte length limit the service shall be "lento", in all other cases it shall be "veloce".
- (²) The service type (fast/slow) depends only on the time the ECU needs to perform the requested operation

5.2

negativeResponse

5.2.1

Message data bytes

negativeResponse Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	Request Service Id	M	xx
#3	responseCode	M	xx

5.2.2


Parameters definition

The responseCode values are specified in /6/.

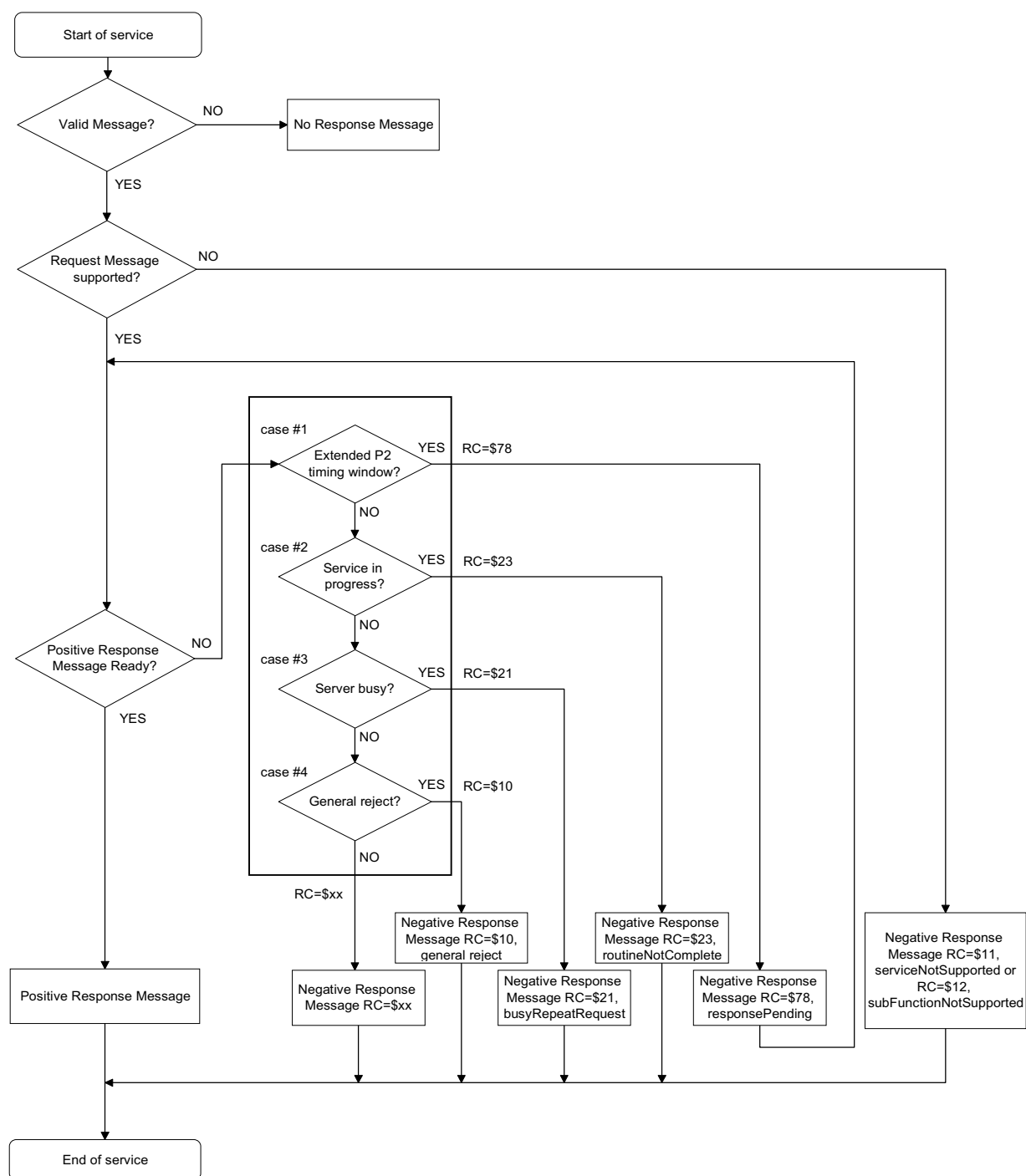
The following table summarises the responseCodes suggested by FIAT for each of the services implemented in protocol KWP2000 on K line. Supplier should use this table to list the responseCodes effectively implemented for each one of the services foreseen by its ECU.

Additional response Codes shall be used by any ECU Supplier only after agreement with FIAT. FIAT reserves the right to ask for specific responseCode values in the vehicleManufacturerSpecific range (90h-F9h) in case of need.

<div> <div>serviceIdentifiers</div> <div>responseCodes</div> </div>	(82h) stopCommunication	(83h) accessTimingParameter	(10h) startDiagnosticSession	(20h) stopDiagnosticSession	(27h) securityAccess	(3Eh) testerPresent	(1Ah) readECUIdentification	(21h) readDataByLocalIdentifier	(3Bh) writeDataByLocalIdentifier	(18h) readDiagnosticTroubleCodesByStatus	(17h) readStatusOfDiagnosticTroubleCodes	(12h) readFreezeFrameData	(14h) clearDiagnosticInformation	(30h) inputOutputControlByLocalIdentifier	(31h) startRoutineByLocalIdentifier	(32h) stopRoutineByLocalIdentifier	(33h) requestRoutineResultByLocalIdentifier	(34h) requestDownload	(36h) transferData	(37h) requestTransferExit
(10h) generalReject	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$
(11h) serviceNotSupported	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$	\$
(12h) subFunctionNotSupported- invalidFormat		#	#		#		#	#	#	#	#	#	#	#	#	#	#			
(21h) busy-repeatRequest									#				#				#			
(22h) conditionsNotCorrectOr RequestSequenceError			#		#				#				#	#	#	#	#	#		#
(23h) routineNotComplete													#				#			
(31h) requestOutOfRange		#	#						#						#		#			
(33h) securityAccessDenied- securityAccessRequested					#			#	#						#			#		
(35h) invalidKey					#															
(36h) exceedNumberOfAttempts					#															
(37h) requiredTimeDelayNotExpired					#															
(40h) downloadNotAccepted																		#		
(41h) improperDownloadType																		#		
(42h) canNotDownloadTo SpecifiedAddress																		#		
(43h) canNotDownloadNumberOf BytesRequested																		#		
(50h) uploadNotAccepted																				
(51h) improperUploadType																				
(52h) canNotUploadFrom SpecifiedAddress																				
(53h) canNotUploadNumberOf BytesRequested																				
(71h) transferSuspended																				
(72h) transferAborted																			#	#
(74h) illegalAddressInBlockTransfer																				
(75h) illegalByteCountInBlockTransfer																			#	
(76h) illegalBlockTransferType																			#	
(77h) blockTransferDataChecksumError																				
(78h) requestCorrectlyReceived- ResponsePending									#				#					#	#	
(79h) incorrectlyByteCountDuring BlockTransfer																				
(80h) serviceNotSupportedInActive DiagnosticMode																				

Key:  responseCode must not be used for this serviceIdentifier
 # responseCode suggested for this serviceIdentifier
 ✓ responseCode used for this serviceIdentifier
 \$ mandatory responseCode in case of serviceIdentifierNotSupported

Following flow-chart contains the general rules for the assignment of the correct negativeResponse codes:



Response codes other than 10h, 11h, 12h, 23h and 78h shall be assigned according to /6/.

The following tables contain some examples of response codes 23h ("requestCorrectlyReceived-ResponsePending"), 21h ("busy-repeatRequest") or 78h ("reqCorrectlyRcvd-RspPending").

client (Tester)	server (ECU)
<Service Name> Request[...] <Service Name> Request[...] : <Service Name> Request[...] <Service Name> Request[...]	{ server has started routine! } <Service Name> NegativeResponse[routineNotComplete (\$23)] <Service Name> NegativeResponse[busy-RepeatRequest (\$21)] : <Service Name> NegativeResponse[busy-RepeatRequest (\$21)] { server has stopped routine! } <Service Name> PositiveResponse[...] OR <Service Name> NegativeResponse[RC ≠ busy-RepeatRequest (\$21)]

client (Tester)	server (ECU)
<Service Name> Request[...] <Service Name> Other Request[...]	<Service Name> NegRsp#1[reqCorrectlyRcvd-RspPending (\$78)] : <Service Name> NegRsp#n[reqCorrectlyRcvd-RspPending (\$78)] <Service Name> PositiveResponse[...] <Service Name> Other PositiveResponse[...]

The negative answer message with responseCode = 78h ("reqCorrectlyRcvd-RspPending") can be sent several times during the same service, at each expiring of time E_dly_tx until the ECU is not able to answer positively to the received request. During the period in which such negative answer is being transmitted the Tester should disable the TesterPresent service.

6

DESCRIPTION OF DIAGNOSTIC MANAGEMENT FUNCTIONAL UNIT

6.0

Introduction

The services specified below shall be used to enable/disable and maintain different diagnostic modes active in the ECU. There shall be only one session active at a time. A diagnostic session enables a specific set of KWP2000 services which shall be defined by FIAT and can be locked by a security access mechanism.

Using a CAN data link between the Tester and the ECU the diagnostic communication itself is established by means of a startDiagnosticSession request message sent by the Tester. In this implementation the Tester is not allowed to change the baudrate; the ECU shall discard any baudrateIdentifier parameter received with the startDiagnosticSession request message.

In case the ECU receives a different request from startDiagnosticSession outside diagnostic sessions, it shall ignore the request without sending any negative response.

If a diagnostic session has been requested by the Tester which is already running, the ECU shall send a positive response message.

The defaultMode-StandardDiagnosticMode-OBDIIMode (refer to 6.1.2.1) is mandatory for any ECU, some ECUs may implement other diagnostic modes in addition to that, depending on the diagnostic requirements. If this is the case, the ECU shall comply to the following rules:

- The very first startDiagnosticSession request received by the ECU shall be a request to enter the defaultMode-StandardDiagnosticMode-OBDIIMode. The ECU shall use a negative response message with responseCode = 22h ("conditionNotCorrectOrRequestSequenceError") to respond to a request to enter a diagnostic mode other than the default one at communication startup.
- If a new diagnostic session is requested by the Tester which is not already running (e.g. to change diagnosticMode) the ECU shall start the new session after it has aborted any pending service in the current diagnostic session.
- At maximum two diagnostic sessions can be nested: when the ECU receives the stopDiagnosticSession request it shall stop (if the ECU conditions allow that) the active session and return to the defaultMode-StandardDiagnosticMode-OBDIIMode.
- If the ECU receives a stopDiagnosticSession request being in the defaultMode-StandardDiagnosticMode-OBDIIMode it shall (if the ECU conditions allow that) send a positive response message and stop the communication.

Whenever a new diagnostic session is requested by the Tester, the ECU shall first send a startDiagnosticSession positive response message before the new session becomes active. If the ECU sends a negative response message with the startDiagnosticSession request service identifier the current session shall continue.

6.1

startDiagnosticSession service

This service shall be used to startup the diagnostic link between the Tester and the ECU and to enable different diagnostic modes in the ECU.

6.1.1

Message data bytes

startDiagnosticSession Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	startDiagnosticSession Request Service Id	M	10
#2	diagnosticMode=[refer to table 6.1.2.1]	M	xx
#3	baudrateIdentifier	U	xx

startDiagnosticSession Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	startDiagnosticSession Positive Response Service Id	S	50
#2	diagnosticMode = [refer to table 6.1.2.1]	M	xx

StartDiagnosticSession Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	startDiagnosticSession Request Service Id	M	10
#3	responseCode = [refer to table 5.2.2.]	M	xx

6.1.2

Parameters definition

6.1.2.1

diagnosticMode

The following table lists the diagnosticMode values supported by FIAT implementation of KWP2000 on CAN.

Hex Value	diagnosticMode	Used in this application
81	defaultMode-StandardDiagnosticMode-OBDDIIMode	✓
83	endOfLineVehicleManufacturerMode (EOL Fiat)	T.B.D.
85	ECUProgrammingMode	T.B.D.
89	Session devoted to starting components	T.B.D.
8A-F9	Specific for FIAT	T.B.D.

The ECU Supplier shall not use the defined diagnosticModes to implement their own services. For this purpose they shall use either endOfLineSystemSupplierMode (84h) or a systemSupplierSpecific (FAh-FEh) mode.

The diagnosticMode 89h shall be used by all those ECUs (e.g.: NCL) in which starting components during a default diagnostic session (81h) is either technically impossible or leads to a state which is different from that worked out by the client.

6.1.2.2

baudrateIdentifier

This parameter is not used in FIAT implementation of KWP2000 on CAN. It shall not be transmitted by the Tester; the ECU shall ignore any baudrateIdentifier parameter received with the startDiagnosticSession request message.

6.1.2.3

diagnosticModes and diagnosticServices availability table

The following table summarises the different diagnosticService sub-sets enabled in the two diagnosticModes supported by FIAT implementation of KWP2000 on CAN.

The ECU Supplier shall not use the defined diagnosticModes to implement their own services. For this purpose they shall use either endOfLineSystemSupplierMode (84h) or a systemSupplierSpecific (FAh-FEh) mode.

diagnosticService	diagnosticMode	
	defaultMode- StandardDiagnosticMode- OBDII Mode	ECUProgrammingMode
startDiagnosticSession	✓	✓
stopDiagnosticSession	✓	✓
securityAccess	✓	✓
testerPresent	✓	✓
readECUIdentification	✓	✓
readDataByLocalIdentifier	✓	✓
writeDataByLocalIdentifier	✓	✓
readDiagnosticTroubleCodesByStatus	✓	
readStatusOfDiagnosticTroubleCodes	✓	
readFreezeFrameData	✓	
clearDiagnosticInformation	✓	
inputOutputControlByLocalIdentifier	✓	
startRoutineByLocalIdentifier	✓	✓
stopRoutineByLocalIdentifier	✓	✓
requestRoutineResultByLocalIdentifier	✓	✓
requestDownload		✓
transferData		✓
requestTransferExit		✓

If FIAT and the ECU Supplier agree on the necessity of using additional diagnostic modes with respect to the those defined in this document (e.g.: endOfLineVehicleManufacturerMode), the ECU Supplier shall declare which services are enabled/disabled in these new diagnostic modes updating the table above. The new table shall be approved by FIAT and shall be inserted in the Specifica Finalizzata di Diagnosi document for that ECU.

6.2**stopDiagnosticSession service**

This service shall be used to break down the diagnostic link between the Tester and the ECU and to disable active diagnostic modes in the ECU.

If the ECU receives the stopDiagnosticSession request being in the defaultMode-Standard-Diagnostic Mode-OBDD Mode it shall (if the actual conditions allow that) send a positive response message and stop the communication.

6.2.1**Message data bytes****stopDiagnosticSession Request Message**

Data byte #	Parameter Name	Cvt	Hex Value
#1	stopDiagnosticSession Request Service Id	M	20

stopDiagnosticSession Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	stopDiagnosticSession Positive Response Service Id	S	60

stopDiagnosticSession Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	stopDiagnosticSession Request Service Id	M	20
#3	responseCode = [refer to table 5.2.2.]	M	xx]

6.3

securityAccess Service

This service shall be used to implement data link security measures on data exchange to protect some reserved or critical functions in the ECU.

Following are the general rules for the security access:

- The Tester shall request the ECU to "unlock" itself by sending the service securityAccess request#1. The ECU shall respond by sending a "seed" using the service securityAccess positive response#1. The Tester shall continue by returning a "key" number back to the ECU using the service securityAccess request#2. The ECU shall compare this "key" to one internally stored. If the two numbers match, then the ECU shall enable the Tester's access to specific KWP2000 services and indicate that with the service securityAccess positive response#2. If upon 2 attempts of a service securityAccess request#2 by the Tester, the two keys do not match, then the ECU shall insert a 10 second time delay before allowing further attempts.
- No additional time delay is required before the ECU responds to a securityAccess request#1 from the Tester after ECU power-on.
- If a device supports security, but is already unlocked when a securityAccess request#1 is received, that ECU shall respond with a securityAccess positive response#1 service with a seed of "00 00h". The Tester shall use this method to determine if the ECU is locked by checking for a non-zero seed.
- The security system shall not prevent normal diagnostic or vehicle communications between the Tester and the ECUs. Proper "unlocking" of the ECU is a prerequisite to the Tester's ability to perform some of the more critical functions such as reading and writing specific memory locations within the ECU, downloading and activating some routines in the ECU. In other words, the only access to the ECU permitted while in a "locked" mode is through the ECU specific software. This permits the ECU specific software to protect itself from unauthorized intrusion.
- ECUs which provide security shall support reject messages if a secure mode is requested while the ECU is locked.
- The accessMode parameter and the security algorithm depend on the kind of operation to be performed.

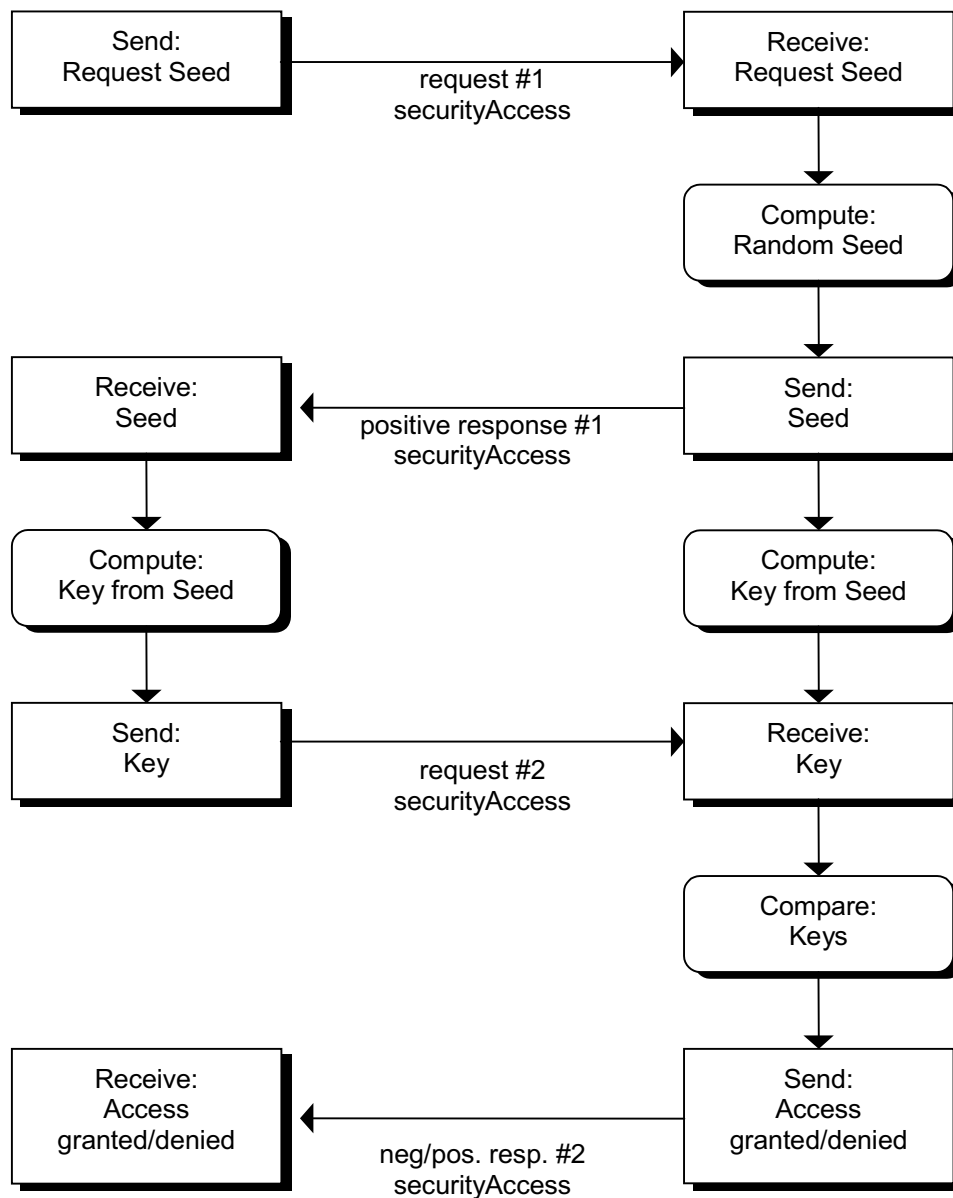
The security algorithm is not described in this document for security reasons.

The following table must be used to list, for each ECU, the services protected by securityAccess. In the column "conditions" any condition should be listed (example: parameter's particular values, access in writing...) which limits the requirement to make the service preceeded by securityAccess.

diagnosticService	Conditions for request of SecurityAccess	diagnosticMode
TBD	TBD	TBD

TESTER

ECU



6.3.1

Message data bytes

securityAccess #1 Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	securityAccess #1 Request Service Id	M	27
#2	accessMode = [requestSeed]	M	xx
#3	accessParameter	U	xx

securityAccess #1 Positive Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	securityAccess Positive Response #1 Service Id	S	67
#2	accessMode = [requestSeed]	M	xx
#3	seed#1 (High Byte)	C	xx
:	:	:	:
#n	seed#m (Low Byte)	C	xx

C = condition: accessMode must be set to "requestSeed"

securityAccess Negative Response #1 Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negative Response Service Id	S	7F
#2	securityAccess Request Service Id	M	27
#3	responseCode = [refer to table 5.2.2.]	M	xx

securityAccess #2 Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	securityAccess #2 Request Service Id	M	27
#2	accessMode = [sendKey]	M	xx
#3	key#1 (High Byte)	C	xx
:	:	:	:
#n	key#m (Low Byte)	C	xx

C = condition: accessMode must be set to "sendKey"

securityAccess Positive Response #2 Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	securityAccess Positive Response #2 Service Id	S	67
#2	accessMode = [sendKey]	M	xx
#3	securityAccessStatus = [securityAccessAllowed]	M	34

securityAccess Negative Response #2 Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negative Response Service Id	S	7F
#2	securityAccess Request Service Id	S	27
#3	responseCode = [refer to table 5.2.2.]	M	xx

6.3.2
Parameters definition

6.3.2.1
accessMode

The following table lists the accessMode values supported by FIAT implementation of KWP2000 on CAN.

Hex Value	accessMode
01,03,05-7F	requestSeed with different levels of security defined by FIAT
02,04,06-7E	sendKey with different levels of security defined by FIAT

6.4**testerPresent service**

This service shall be used to indicate to the ECU that the Tester is present to prevent the ECU from automatically returning to normal operation and stop the communication.

6.4.1**Message data bytes****testerPresent Request Message**

Data byte #	Parameter Name	Cvt	Hex Value
#1	testerPresent Request Service Id	M	3E

testerPresent Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	testerPresent Positive Response Service Id	S	7E

testerPresent Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	testerPresent Request Service Id	M	3E
#3	responseCode = [refer to table 5.2.2.]	M	xx

6.5**ecuReset Service**

The service ecuReset is not used in FIAT implementation of KWP2000 on CAN.

6.6

readEculdentification Service

This service shall be used to read out identification data from the ECU.

6.6.1

Message data bytes

readEculdentification Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readEculdentification Request Service Id	M	1A
#2	identificationOption = [refer to table 6.6.2.1]	M	xx

readEculdentification Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readEculdentification Positive Response Service Id	S	5A
#2	identificationRecordValue = [identificationOption=[ref. to table 6.6.2.1]	M	xx=[xx,
#3	ECUIdentificationParameter#1	C	xx,
:	:	:	:
#n	ECUIdentificationParameter#m]	C	xx]

C = condition: parameter depends on the identificationOption value

readEculdentification Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negative Response Service Id	S	7F
#2	readEculdentification Request Service Id	M	1A
#3	responseCode = [refer to table 5.2.2.]	M	xx

6.6.2**Parameters definition****6.6.2.1****identificationOption**

The identificationOption parameter value identify which identification code field is requested from the Tester.

Hex Value	Identification Option	ECU Identification Data Format	# of bytes	Data Format	Used in this application
80	Identification Code	All the following data	61	---	✓
91	FIAT drawing number	'123456789'	11	ASCII	✓
92	ECU Hardware Number	TBD	11	ASCII	✓
93	ECU Hardware Version	TBD	1	UNSGN	TBD
94	ECU Software Number	TBD	11	ASCII	✓
95	ECU Software Version	TBD	2	UNSGN	TBD
96	Omologation Number	TBD	6	ASCII	TBD
97	ISO Code	xx xx xx xx xx [refer to 6.6.2.2]	5	UNSGN	✓
98	Tester Code	TBD	10	ASCII	TBD
99	Date of download/production	19 97 03 31 [Y-Y-M-D]	4	BCD	✓

Identification option 91h, 92h, 94h, 96h and 98h shall be aligned to the left; bytes which are not used in these fields shall be padded with blanks (20h). If the identification options 93h and 95h are not used they shall be filled with zeroes (00h).

Id. option 98h shall be filled with blanks (20h) by the Supplier before the ECU is shipped to FIAT, it shall be re-written by the Tester whenever the ECU is completely re-programmed (downloaded). The ECUs that do not implement the download functionality shall anyway have this identification option filled by the Supplier at production time.

Id. option 99h shall be filled with the ECU date of production when the ECU is sent to FIAT. This field shall be re-written by the Tester whenever the ECU is completely re-programmed (downloaded). The ECUs that do not have download capability shall anyway have this identification option filled by the Supplier at production time.

The usage of identification options in case of End Of Line partial programming (EOL) is not specified in this document; it will be defined in a dedicated document.

6.6.2.2**ISO Code value summary table (identificationOption = 97h)**

FIAT is responsible for assigning a unique ISO Code which describes the hardware and software functionalities of the ECU with respect to diagnosys functions. The ISO Code shall be written in the ECU memory under the Supplier's responsibility. The first 2 Bytes of this field are assigned according to ISO 9141 while Bytes #3 to #5 are FIAT specific.

The ECU supplier shall ask FIAT for a new ISO Code whenever a change relevant to diagnostics is decided for the ECU:

- New ECU/Hardware differences
- New ECU pinout
- New diagnostic protocol
- New sensor(s)
- Addition/elimination of one or more sensor(s)
- Changes in scaling formula(s)
- Addition of pieces of information exchanged through the diagnostic link.

Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Vehicle
XX	XX	XX	XX	XX	TBD

7

DESCRIPTION OF DATA TRANSMISSION FUNCTIONAL UNIT

7.0

Introduction

The services specified in this functional unit shall be used by the Tester to read/write records into the ECU's memory, access must happen through local identifiers.

7.1

readDataByLocalIdentifier Service

This service shall be used by the Tester to access some ECU internal parameters and read their current values. Reserved parameters shall need a successful "security access" sequence in order to be accessed.

7.1.1

Message data bytes

readDataByLocalIdentifier Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readDataByLocalIdentifier Request Service Id	M	21
#2	recordLocalIdentifier = [refer to table 7.1.2.1]	M	xx

readDataByLocalIdentifier Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readDataByLocalIdentifier Positive Response Service Id	S	61
#2	recordLocalIdentifier = [refer to table 7.1.2.1]	M	xx
#3	recordValue #1	M	xx
:	:	:	:
#n	recordValue #m	M	xx

readDataByLocalIdentifier Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	readDataByLocalIdentifier Request Service Id	M	21
#3	responseCode = [refer to table 5.2.2.]	M	xx

responseCode = 33h in the readDataByLocalIdentifier negative response message shall be used whenever the requested recordLocalIdentifier is protected by a security access and the Tester has not yet request to unlock the ECU (with a "security access" procedure).

7.1.2

Parameters definition

7.1.2.1

recordLocalIdentifier

The recordLocalIdentifier applicable to the ECU shall be assigned by FIAT and by ECU Suppliers according to the following rules.

RLI (Hex)	Description	Read	Write	Num. of Bytes	Conversion formula
01-1F	Reserved by this document				
20-2F	EEPROM data	✓	TBD	2(*)	TBD
30-7F	RAM data	✓	TBD	2(*)	TBD
80-97	Reserved for identificationOption	NO	NO		
98	Tester code	NO	✓	refer to 6.6.2.1	refer to 6.6.2.1
99	Date of download/production	NO	✓	refer to 6.6.2.1	refer to 6.6.2.1
99-9F	Reserved for identificationOption	NO	NO		
A0-BF	snapshots	✓	NO	TBD	TBD
C0-FE	not used				

(*): All the ECU parameters that can be accessed via the readDataByLocalIdentifier service should preferably be two bytes long with the exception of:

- the variables which indicate a status (e.g.: digital inputs or outputs, ECU states...); these data shall be assigned to a 1-bit field each and shall be grouped into status bytes.
- data tables that shall be read/written altogether in a single service message (e.g.: EOL data tables).

All the available pieces of information shall be accessible by individual parameter access, if the ECU has enough resources they can also be inserted into snapshots.

The ECU shall always use the same data length and conversion formula independently of which method the Tester is using to read any specific parameter.

7.2**readDataByCommonIdentifier Service**

The service readDataByCommonIdentifier is not used in FIAT implementation of KWP2000 on CAN.

7.3**readMemoryByAddress Service**

The readMemoryByAddress service is not used in FIAT implementation of KWP2000 on CAN.

7.4**dinamicallyDefineLocalIdentifier Service**

The dinamicallyDefineLocalIdentifier service is not used in FIAT implementation of KWP2000 on CAN.

7.5

writeDataByLocalIdentifier Service

This service shall be used by the Tester to access some ECU parameters and update their current values. Reserved parameters shall need a successful security access sequence in order to be accessed.

7.5.1

Message data bytes

writeDataByLocalIdentifier Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	writeDataByLocalIdentifier Request Service Id	M	3B
#2	recordLocalIdentifier = [refer to table 7.1.2.1]	M	xx
#3	recordValue#1	M	xx
:	:	:	:
#n	recordValue#m	M	xx

writeDataByLocalIdentifier Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	writeDataByLocalIdentifier Positive Response Service Id	S	7B
#2	recordLocalIdentifier = [refer to table 7.1.2.1]	M	xx

writeDataByLocalIdentifier Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	writeDataByLocalIdentifier Request Service Id	M	3B
#3	responseCode = [refer to table 5.2.2.]	M	xx

7.6

writeDataByCommonIdentifier Service

The writeDataByCommonIdentifier service is not used in FIAT implementation of KWP2000 on CAN.

7.7

writeMemoryByAddress Service

The writeMemoryByAddress service is not used in FIAT implementation of KWP2000 on CAN.

8**DESCRIPTION OF STORED DATA TRANSMISSION FUNCTIONAL UNIT****8.0****Introduction**

The services provided by this functional unit shall be used by the Tester to read the actual content of the ECU's error memory. The ECU shall guarantee that the data sent to the Tester are up-to-dated at the time of the request. The ECU shall also provide against losing the error memory contents in case of key-off or power-off.

8.1**readDiagnosticTroubleCodes Service**

The readDiagnosticTroubleCodes service is not used in FIAT implementation of KWP2000 on CAN.

8.2

readDiagnosticTroubleCodesByStatus Service

This service shall be used by the Tester to read the complete list of the diagnosticTroubleCodes currently stored in the ECU's error memory at the time of the request independently from their current status.

8.2.1

Message data bytes

readDiagnosticTroubleCodesByStatus Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readDiagnosticTroubleCodesByStatus Request Service Id	M	18
#2	statusOfDTC = requestIdentifiedDTCAndStatus	M	00
#3	groupOfDTC (High Byte)	M	FF
#4	groupOfDTC (Low Byte) = [all groups]	M	00

readDiagnosticTroubleCodesByStatus Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readDiagnosticTroubleCodesByStatus Pos.Resp. Serv. Id	S	58
#2	numberOfDTC	M	xx
#3	listOfDTCAndStatus = [DTC #1 (High Byte)	C	xx
:	DTC #1 (Low Byte)		xx
:	statusOfDTC #1		xx
:	:		:
:	DTC #m (High Byte)		xx
:	DTC #m (Low Byte)		xx
#n	statusOfDTC #m]		xx

C = condition: listOfDTCAndStatus is only present if numberOfDTC > 0

readDiagnosticTroubleCodesByStatus Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	readDiagnosticTroubleCodesByStatus Request Service Id	M	18
#3	responseCode = [refer to table 5.2.2.]	M	xx

If the ECU does not have any DTC with status information stored it shall send a positive response message with numberOfDTC set to 0h, not including the parameters in listOfDTCAndStatus.

8.2.2

Parameters definition

8.2.2.1

groupOfDTC

The Tester is not allowed to read a single DTC or a specific group of DTCs using this service. Following is the only valid groupOfDTC parameter value.

High Byte (Hex)	Low Byte (Hex)	Description
FF	00	All groups

8.2.2.2

numberOfDTC

The numberOfDTC parameter value shall be the number of errors present in the memory at the time of the request. Valid range is 0 to the maximum number of errors that the ECU can store.

numberOfDTC = 00h shall indicate that no error is currently stored.

8.2.2.3

Error Memory structure and management

The error memory is "seen" by the Tester as it is divided in 10 byte blocks each. Each block shall contain all the data necessary to describe one diagnostic trouble code which has been stored by the system and it shall include the following fields:

Block	Byte	Description	Conversion	Units
#1	#1	DTC (High Byte)	refer to table 8.2.2.4	refer to table 8.2.2.4
	#2	DTC (Low Byte)		
	#3	statusOfDTC	refer to table 8.2.2.5	refer to table 8.2.2.5
	#4	environmentalCondition #1	TBD	TBD
	#5	environmentalCondition #2	TBD	TBD
	#6	environmentalCondition #3	TBD	TBD
	#7	environmentalCondition #4	TBD	TBD
	#8	environmentalCondition #5	TBD	TBD
	#9	environmentalCondition #6	TBD	TBD
	#10	eventCounter	1 bit = 1 event	[events]
#2	#1	DTC (High Byte)	refer to table 8.2.2.4	refer to table 8.2.2.4
	#2	DTC (Low Byte)		
	:	:	:	:
	#10	eventCounter	1 bit = 1 event	[events]
...	:	:	:	:
#n	#1	DTC (High Byte)	refer to table 8.2.2.4	refer to table 8.2.2.4
	#2	DTC (Low Byte)		
	:	:	:	:
	#10	eventCounter	1 bit = 1 event	[events]

The minimum number of blocks that can be stored at a time shall be greater than or equal to 5.

The maximum number of blocks which can be stored at the same time shall be agreed between FIAT and the ECU Supplier. If all the cells are occupied and a further fault occurs, the block with the lowest eventCounter value shall be overwritten. If two or more blocks have the same eventCounter value the block with the lowest priority DTC shall be overwritten first. FIAT and the ECU Supplier shall agree on the priority of the DTCs in each ECU.

The number of error memory cells and the total number of DTCs assigned to an ECU shall be related in a way that the probability of failure information loss is minimized even in case of serious malfunctions causing a great number of DTC to be detected.

For any sub-component or sub-function of the ECU only one DTC shall be assigned in order to prevent the filling of the error memory cells with different DTCs that are related to a single faulty element.

Whenever the ECU detects a fault which DTC is already present in memory, the behavior shall be the following:

- the field DTCStorageState in statusOfDTC shall be updated while all other fields in this byte shall be kept at their current values.
- the six bytes of environmental conditions shall not be updated.
- the counter shall be kept at its current value.

The **DTC Code** identifies the faulty element (see 8.2.2.4).

The **statusOfDTC** gives information about the status of memorised error (see table 8.2.2.5).

The **environmentalCondition** bytes are the values of maximum six relevant parameters stored at the moment a fault was detected the first time on the specified element. The parameters shall be the same for all the DTCs defined for a single ECU.

The **eventCounter** shall be positioned at value 64 decimal at the moment that the fault is stored the first time. The counter shall be decreased by 1 unit at the next key ON when a complete test cycle has been executed by the ECU and the ECU itself has verified the absence of the relative fault. The test cycle shall comprehend a list of operations (self-tests, actuations, monitoring...) that guarantee most of the ECU functionalities are being checked. FIAT and ECU Supplier shall agree upon the definition of the test cycle applicable to each different ECU.

When the counter reaches the value 0 the data contained in the error block shall be erased.

The conditions that allow the ECU to verify the presence/absence of a fault shall be specified by the ECU Supplier for each DTC and agreed by FIAT.

8.2.2.4

DTC and environmentalConditions tables

The following rules and tables shall be used for the specification of the DTCs and the associated environmentalCondition parameter.

DTCs and DTC groups

The DTCs are divided into 5 groups POWERTRAIN, CHASSIS, BODY, NETWORK COMMUNICATION, ALL (ALL = all vehicle systems) according to /7/. The DTC are of BCD type (binary coded decimal). The table below specifies the possible values for this parameter.

High Byte (Hex)	Low Byte (Hex)	Description
FF	00	All groups (A)
00	00	Powertrain group (P) - Powertrain DTCs range is 0001h to 3999h
40	00	Chassis group (C) - Chassis DTCs range is 4001h to 7999h
80	00	Body group (B) - Body DTCs range is 8001h to B999h
C0	00	Network group (U) - Network DTCs range is C001h to F999h

diagnosticTroubleCode assignment rules

The DTC parameter is used by the ECU to report system failures by a two byte BCD number. The format of a DTC is specified in /7/. The decoding is summarized in the following table.

Bit 15,14: P,C,B,U/A	Bit 13,12: sub groups 0-3	DTC number (Binary Coded Decimal) range: 001 - 999		
0 0 (P)owertrain	0	0	0	1
:	:	:	:	:
0 0 (P)owertrain	3	9	9	9
0 1 (C)hassis	0	0	0	1
:	:	:	:	:
0 1 (C)hassis	3	9	9	9
1 0 (B)ody	0	0	0	1
:	:	:	:	:
1 0 (B)ody	3	9	9	9
1 1 (U)Network/All	0	0	0	1
:	:	:	:	:
1 1 (U)Network/All	3	9	9	9

Sub group 0 shall be used for ISO/SAE controlled DTCs which are those codes where industry uniformity has been achieved. Sub group 3 is reserved: DTCs in that group shall not be used by the ECU.

FIAT DTC coding follows the rules specified below:

- DTCs shall belong to sub groups 1 or 2 ("manufacturer controlled") unless otherwise specified.
- a DTC shall identify a sub component of the ECU that shall be replaced as a whole during servicing. E.g.: there is no need to distinguish an ECU having a faulty memory bank from an ECU having a problem with its I/O drivers; since in both cases service people must substitute the ECU. In this case the same DTC shall be applied with different DTCFaultSymptoms thus reducing the total number of DTCs..

The following tables shall be used as a reference to specify all the DTCs defined for any specific ECU.

Total No. of DTC	Max number of stored blocks
TBD	TBD

DTC Code	Description	DTC Fault Symptom
TBD	TBD	TBD

The following table shall be used as a reference to specify the environmentalConditions defined for any specific ECU.

Position	Description	Conversion formula
1	TBD	TBD
2	TBD	TBD
3	TBD	TBD
4	TBD	TBD
5	TBD	TBD
6	TBD	TBD

8.2.2.5 statusOfDTC

The statusOfDTC parameter shall give information about the status of a particular DTC. It shall be updated according to the rules in paragraph 8.2.2.3.

The following table contains the description of the statusOfDTC bit fields, any change in the coding or in the meaning of statusOfDTC bits shall be agreed between FIAT and ECU Supplier.

SODTC-RE	Description of StatusOfDTC-Response
Bit: 7654 3210b	
WBAT SSSSb	DTCFaultSymptom { 'S' bit: 3 - 0 } (*)
{ \$x0 } 0000	no fault symptom available for this DTC
{ \$x0 } 0001	above maximum threshold
{ \$x0 } 0010	below minimum threshold
{ \$x0 } 0100	no signal
{ \$x0 } 1000	invalid signal
WBAT SSSSb	DTCReadinessFlag { 'T' bit: 4 }
0	test complete for this DTC or not applicable
1	test not complete for this DTC
WBAT SSSSb	DTCStorageState { B - bit 6, A = bit 5 }
0 0	noDTCDetected at time of request (no DTC stored in non volatile memory)
0 1	DTCNotPresent at time of request (DTC was present, DTC stored in non volatile memory)
1 0	DTCMaturing-Intermittent at time of request (insufficient data to consider as a malfunction to store in non volatile memory)
1 1	DTCPresent at time of request (DTC stored in non volatile memory)
WBAT SSSSb	DTCWarningLampCalibrationStatus { 'W' bit: 7 }
0	disabled (Warning Lamp is not illuminated for this DTC)
1	enabled (Warning Lamp illuminated for this DTC)

(*) DTCFaultSymptom values 03h, 05h, 06h, 07h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh and 0Fh can be used only with previous agreement between FIAT and ECU Supplier.

8.3**readStatusOfDiagnosticTroubleCodes Service**

This service shall be used by the Tester to read details related to a diagnosticTroubleCode currently stored into ECU's error memory at a time. The Tester is not allowed to use this service to read two or more diagnosticTroubleCode data together in the same message.

8.3.1**Message data bytes**

readStatusOfDiagnosticTroubleCodes Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readStatusOfDiagnosticTroubleCodes Request Service Id	M	17
#2	DTC (High Byte)	M	xx
#3	DTC (Low Byte) = [refer to table 8.2.2.4]	M	xx

readStatusOfDiagnosticTroubleCodes Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readStatusOfDiagnosticTroubleCodes Positive Response Service Id	S	57
#2	numberOfDTC	M	xx
#3	listOfDTCAndStatus = [DTC (High Byte) DTC (Low Byte) statusOfDTC environmentalCondition#1 environmentalCondition#2 environmentalCondition#3 environmentalCondition#4 environmentalCondition#5 environmentalCondition#6 eventCounter#7]	C	xx
#4			xx
#5			xx
#6			xx
#7			xx
#8			xx
#9			xx
#10			xx
#11			xx
#12			xx

C = condition: listOfDTCAndStatus is only present if numberOfDTC > 0

readStatusOfDiagnosticTroubleCodes Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	readStatusOfDiagnosticTroubleCodes Request Service Id	M	17
#3	responseCode = [refer to table 5.2.2.]	M	xx

If the required DTC is not stored in the ECU's memory at the time of the request, the ECU shall send a positive response message with numberOfDTC = 00h, it shall as well not include the listOfDTCAndStatus parameter.

The ECU shall use the negative response with responseCode set to 12h whenever an unknown DTC is requested by the Tester.

In the event of DTC recognised and stored by ECU:

- numberOfDTC shall be set at value 01h
- The length of the readStatusOfDiagnosticTroubleCodes positive response shall be fixed and equal to 12 bytes: if the ECU does not support 6 environmentalCondition parameters for its DTCs it shall pad the listOfDTCAndStatus field with zeroes (00h).

8.4

readFreezeFrameData Service

This service shall be used to access the “Freeze Frame” data on ECUs that shall comply with EOBD specifications.

8.4.1

Message data bytes

readFreezeFrameData Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readFreezeFrameData Request Service Id	M	12
#2	freezeFrameNumber	M	01
#3	recordAccessMethodIdentifier = requestAllData	M	00

readFreezeFrameData Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	readFreezeFrameData Positive Response Service Id	S	52
#2	freezeFrameNumber	M	01
#3	freezeFrameData#1	M	xx
:	:	:	:
#n-1	freezeFrameData#m	M	xx
#n	recordAccessMethodIdentifier = requestAllData	M	00

readFreezeFrameData Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	readFreezeFrameData Request Service Id	M	12
#3	responseCode = [refer to table 5.2.2.]	M	xx

8.5

clearDiagnosticInformation Service

The messages described below shall be used by the Tester to clear all diagnostic information related to any diagnosticTroubleCode currently stored into ECU's error memory. The Tester is not allowed to use this service to partially reset the contents of the error memory.

8.5.1

Message data bytes

clearDiagnosticInformation Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	clearDiagnosticInformation Request Service Id	M	14
#2	groupOfDiagnosticInformation (High Byte)	M	FF
#3	groupOfDiagnosticInformation (Low Byte) = [all groups]	M	00

clearDiagnosticInformation Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	clearDiagnosticInformation Positive Response Service Id	S	54
#2	groupOfDiagnosticInformation (High Byte)	M	FF
#3	groupOfDiagnosticInformation (Low Byte) = [all groups]	M	00

clearDiagnosticInformation Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	clearDiagnosticInformation Request Service Id	M	14
#3	responseCode = [refer to table 5.2.2.]	M	xx

9

DESCRIPTION OF INPUT/OUTPUT CONTROL FUNCTIONAL UNIT

9.0

Introduction

This functional unit provides the services for the Tester to control an input/output specific to an ECU.

9.1

inputOutputControlByLocalIdentifier Service

This service shall be used to substitute a value for an input signal, internal ECU function and/or control an output (actuator) of an ECU referenced by a local identifier.

9.1.1

Message data bytes

inputOutputControlByLocalIdentifier Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	inputOutputControlByLocalIdentifier Request Service Id	M	30
#2	inputOutputLocalIdentifier = [refer to table 9.1.2.3]	M	xx
#3	controlOption = [inputOutputControlParameter = [refer to table 9.1.2.1]	M	xx
#4	controlState#1 = [refer to table 9.1.2.2]	C	xx
:	:	:	:
#n	controlState#m = [refer to table 9.1.2.2]	C	xx

C = condition: parameter is present only if inputOutputControlParameter=07h or 08h

inputOutputControlByLocalIdentifier Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	inputOutputControlByLocalIdentifier Pos.Resp. Serv. Id	S	70
#2	inputOutputLocalIdentifier = [refer to table 9.1.2.3]	M	xx
#3	controlStatus = [inputOutputControlParameter = [refer to table 9.1.2.1]	M	xx
#4	controlState#1 = [refer to table 9.1.2.2]	C	xx
:	:	:	:
#n	controlState#m = [refer to table 9.1.2.2]	C	xx

C = condition: parameter is present only if inputOutputControlParameter=01h.

inputOutputControlByLocalIdentifier Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	inputOutputControlByLocalIdentifier Request Service Id	M	30
#3	responseCode = [refer to table 5.2.2.]	M	xx

9.1.2

Parameters definition

9.1.2.1

inputOutputControlParameter

Hex Value	Parameter Name and Description
00	returnControlToECU This value shall indicate to the ECU that the Tester does no longer have control about the input signal, internal parameter or output signal referenced by the InputOutputLocalIdentifier.
01	reportCurrentState This value shall indicate to the ECU that is requested to report the current state of the input signal, internal parameter or output signal referenced by the InputOutputLocalIdentifier.
04	resetToDefault This value shall indicate to the ECU that is requested to reset the input signal, internal parameter or output signal referenced by the inputOutputLocalIdentifier to its default value.
07	shortTermAdjustment This value shall indicate to the ECU that is requested to adjust the input signal, internal parameter or output signal referenced by the inputOutputLocalIdentifier in RAM to the value included in the ControlOptionParameter. (E.g. set Idle Control actuator to a specific position, set pulse width of valve to a specific value).
08	longTermAdjustment This value shall indicate to the ECU that is requested to adjust the input signal, internal parameter or output signal referenced by the inputOutputLocalIdentifier in EEPROM/FLASH EEPROM to the value included in the ControlOptionParameter. (E.g. set Engine Idle Speed, set CO Adjustment parameter to a specific value).

9.1.2.2

inputOutputControlState summary table

inputOutputControlState values can have different meanings depending on the type of inputOutputLocalIdentifier to which they refer as described in the following table.

Hex Value	Description
00	The component is set to OFF state (ON/OFF components)
FF	The component is set to ON state (ON/OFF components)
00 (000%) FF (100%)	The component is set to a definite setting (%) (CONTROL components)

For components that only allow a default activation (e.g.: ON/OFF cycle with a certain duty and timings) the inputOutputControlParameter = 07h (shortTermAdjustment) shall be used without controlState parameter.

9.1.2.3**inputOutputLocalIdentifier**

The following table shall be used to summarise the inputOutputLocalIdentifier values and the referenced components. For any component the valid inputOutputControlState values shall also be indicated.

inputOutputLocalIdentifier		Applicable inputOutputControlParameter Values				
Hex Value	Component Description	00h	01h	04h	07h	08h
TBD	TBD	TBD	TBD	TBD	TBD	TBD

9.1.2.4**General rules for the activation/deactivation of a component**

FIAT and ECU Supplier shall agree upon the environmental conditions that enable the activation of a component by means of a inputOutputControlByLocalIdentifier.

The activation of a component by means of a inputOutputControlByLocalIdentifier service shall be interrupted by the ECU if at least one of the following conditions are fulfilled:

- the ECU has not received any inputOutputControlByLocalIdentifier request since 30s.
- the ECU recognises a condition of "vehicle running" (e.g.: vehicle speed > 0 Km/h) and/or a condition of "engine out of idle condition" (e.g.: engine speed >= 1500 rpm).
- the ECU receives a stopDiagnosticSession
- there is a time-out in the diagnostic communication.

If any of the above mentioned conditions are fulfilled the component shall return immediately under the complete control of the ECU.

The ECU Supplier shall agree with FIAT upon any additional condition that shall disable the activation of a component.

9.2**inputOutputControlByCommonIdentifier Service**

The inputOutputControlByCommonIdentifier service is not used in FIAT implementation of KWP2000 on CAN.

10

DESCRIPTION OF REMOTE ROUTINE ACTIVATION FUNCTIONAL UNIT

10.0

Introduction

This functional unit specifies the services of remote activation of routines as they shall be implemented by ECUs and Tester. There are many methods of implementation possible, FIAT KWP2000 on CAN method is based on the assumption that after a routine has been started by the Tester in the ECU's memory, the ECU itself is responsible to stop the routine.

- The ECU routine shall be started in a time between the completion of the startRoutineByLocalIdentifier request message and the completion of the first response message.
- The ECU routine shall be stopped any time as programmed or previously initialised in the ECU's memory. The Tester shall use the stopRoutineByLocalIdentifier request message to abort the routine execution.
- The Tester shall use the requestRoutineResultsByLocalIdentifier service to wait for the completion of the routine and to get the its exit status information.
- During routine execution the ECU shall use the requestRoutineResultsByLocalIdentifier negative response with response code 23h ("routineNotComplete") and 21h ("busy-repeatRequest") to indicate to the Tester that the routine is already in progress but not yet completed (refer to paragraph 5.2.2 in this document).

10.1

startRoutineByLocalIdentifier Service

This service shall be used by the Tester to start the execution of a routine in the ECU's memory. The routine is referenced by a Local Identifier.

10.1.1

Message data bytes

startRoutineByLocalIdentifier Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	startRoutineByLocalIdentifier Request Service Id	M	31
#2	routineLocalIdentifier = [refer to table 10.1.2.1]	M	xx
#3	routineEntryOption#1	U	xx
:	:	:	:
#n	routineEntryOption#m	U	xx

startRoutineByLocalIdentifier Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	startRoutineByLocalIdentifier Pos.Resp. Serv. Id	S	71
#2	routineLocalIdentifier = [refer to table 10.1.2.1]	M	xx

startRoutineByLocalIdentifier Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	startRoutineByLocalIdentifier Request Service Id	M	31
#3	responseCode = [refer to table 5.2.2.]	M	xx

10.1.2**Parameters definition****10.1.2.1****routineLocalIdentifier**

The routineLocalIdentifiers applicable to ECU shall be assigned by FIAT and by ECU Supplier in accordance with the following rules.

RELI (Hex)	Routine description	#	routineEntryOptions description
01-02	Excluded from this document	(*)	(*)
03-FF	TBD	TBD	TBD

(*) These routines are used during download procedures, The ECU which do not implement download functionality should not implement these routines, they therefore should not use the routineLocalIdentifiers which are then excluded from this document. The description of such routines is beyond the purpose of this document.

10.1.2.2**routineEntryOption**

The definition of this parameter depends on the type of routine indicated by the routineLocalIdentifier.

10.2**startRoutineByAddress Service**

The startRoutineByAddress service is not used in FIAT implementation of KWP2000 on CAN.

10.3**stopRoutineByLocalIdentifier Service**

This service shall be used to stop the execution of a routine in the ECU's memory. The routine is referenced by a Local Identifier.

10.3.1**Message data bytes****stopRoutineByLocalIdentifier Request Message**

Data byte #	Parameter Name	Cvt	Hex Value
#1	stopRoutineByLocalIdentifier Request Service Id	M	32
#2	routineLocalIdentifier = [refer to table 10.1.2.1]	M	xx

stopRoutineByLocalIdentifier Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	startRoutineByLocalIdentifier Pos.Resp. Serv. Id	S	72
#2	routineLocalIdentifier = [refer to table 10.1.2.1]	M	xx

stopRoutineByLocalIdentifier Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	stopRoutineByLocalIdentifier Request Service Id	M	32
#3	responseCode = [refer to table 5.2.2.]	M	xx

10.4**stopRoutineByAddress Service**

The stopRoutineByAddress service is not used in FIAT implementation of KWP2000 on CAN.

10.5**requestRoutineResultsByLocalIdentifier Service**

This service shall be used by the Tester to request the results of a routine. The routine is referenced by a Local Identifier. A negative response message with responseCodes 23h ("routineNotComplete") and 21h ("busy-repeatRequest") shall be used by the ECU to indicate that the routine has not been completed (refer to paragraph 5.2.2 in this document).

10.5.1**Message data bytes****requestRoutineResultsByLocalIdentifier Request Message**

Data byte #	Parameter Name	Cvt	Hex Value
#1	requestRoutineResultsByLocalIdentifier Request Service Id	M	33
#2	routineLocalIdentifier = [refer to table 10.1.2.1]	M	xx

requestRoutineResultsByLocalIdentifier Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	requestRoutineResultsByLocalIdentifier Positive Response Service Id	S	73
#2	routineLocalIdentifier = [refer to table 10.1.2.1]	M	xx
#3	routineResults#1	U	xx
:	:	:	:
#n	routineResults#m	U	xx

requestRoutineResultsByLocalIdentifier Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	requestRoutineResultsByLocalIdentifier Request Service Id	M	33
#3	responseCode = [refer to table 5.2.2.]	M	xx

10.5.2

Parameters definition

The following tables shall be used to describe the supported ECU routines and the relative entry options and results. A single paragraph shall be written for any supported routine.

Routine TBD

Description	RELI (Hex)
TBD	TBD

Following are valid entry option values for routine TBD:

TBD routineEntryOption Description

Data Byte #	Parameter Name	Cvt	Hex Value
#1	TBD	T	TBD
:	:	B	:
#n	TBD	D	TBD

Following are valid routineResults values for routine TBD:

TBD routineResults Description

Data Byte #	Parameter Name	Cvt	Hex Value
#1	TBD	T	TBD
:	:	B	:
#n	TBD	D	TBD

10.6

requestRoutineResultsByAddress Service

The requestRoutineResultsByAddress service is not used in FIAT implementation of KWP2000 on CAN.

11.

DESCRIPTION OF UPLOAD DOWNLOAD FUNCTIONAL UNIT

11.0

Introduction

This functional unit specifies the services of negotiation for data transfers as they shall be implemented by ECUs and Tester. FIAT KWP2000 on CAN does not implement data transfer from the ECU to the Tester (upload).

11.1

requestDownload Service

This service shall be used by the Tester to initiate a data transfer from the Tester to the ECU (download). After the ECU has received the requestDownload request message it shall take all necessary actions to receive data before it sends a positive response message.

11.1.1

Message data bytes

requestDownload Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	requestDownload Request Service Id	M	34
#2	transferRequestParameter = [memoryAddress (High Byte) memoryAddress (Middle Byte) memoryAddress (Low Byte) dataFormatIdentifier unCompressedMemorySize (High Byte) unCompressedMemorySize (Middle Byte) unCompressedMemorySize (Low Byte)]	M	xx
#3		M	xx
#4		M	xx
#5		M	xx
#6		M	xx
#7		M	xx
#8		M	xx

requestDownload Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	requestDownload Pos.Resp. Serv. Id	M	74
#2	transferResponseParameter = [maxNumberOfBlockLength]	M	xx

requestDownload Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	requestDownload Request Service Id	M	34
#3	responseCode = [refer to table 5.2.2.]	M	xx

11.1.2

Parameters definition

11.1.2.1

dataFormatIdentifier

The table below lists the valid dataFormatIdentifier values.

Hex Value	Description	Cvt
0x	unCompressed	M
1x	Bosch Compression Method	U
2x	Hitachi Compression Method	U
3x	Marelli Compression Method	U
4x-Fx	TBD	U
x0	unEncrypted	M
x1	Bosch Encrypting Method	U
x2	Hitachi Encrypting Method	U
x3	Marelli Encrypting Method	U
x4-xF	TBD	U

11.2

requestUpload Service

The requestUpload service is not used in FIAT implementation of KWP2000 on CAN.

11.3

transferData Service

This service shall be used by the Tester to transfer data to the ECU. FIAT KWP2000 on CAN does not implement the data transfer in the reverse direction: the transferData positive response message shall not contain any parameter.

11.3.1

Message data bytes

transferData Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	transferData Request Service Id	M	36
#2	transferRequestParameter#1	U	xx
:	:	:	:
#n	transferRequestParameter#m	U	xx

transferData Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	transferData Pos.Resp. Serv. Id	M	76

transferData Negative Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	transferData Request Service Id	M	36
#3	responseCode = [refer to table 5.2.2.]	M	xx

11.4

requestTransferExit Service

This service shall be used by the Tester to terminate a data transfer between the Tester and the ECU.

11.4.1

Message data bytes

requestTransferExit Request Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	requestTransferExit Request Service Id	M	37

requestTransferExit Positive Response Message

Data byte #	Parameter Name	Cvt	Hex Value
#1	requestTransferExit Pos.Resp. Serv. Id	M	77

requestTransferExit Negative Response Message

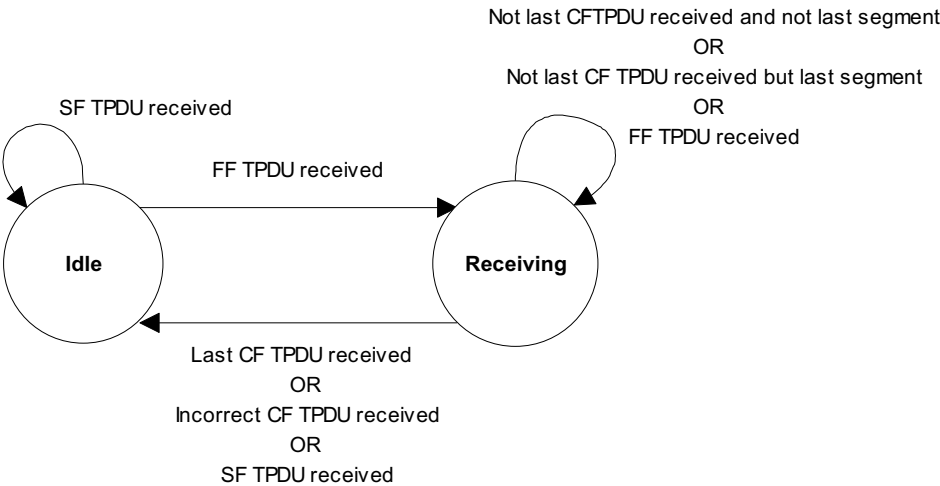
Data byte #	Parameter Name	Cvt	Hex Value
#1	negativeResponse Service Id	S	7F
#2	requestTransferExit Request Service Id	M	37
#3	responseCode = [refer to table 5.2.2.]	M	xx

Appendix A

A. TRANSPORT PROTOCOL IMPLEMENTATION SPECIFICATIONS

A.1
Receiver Transport Protocol Entity

A graphical overview of the Receiver TPE is shown by the State Transition Diagram that follows.



A.1.1
Service primitives

T_Data.Ind(DataLength, Data[])

Indication to the service user (the diagnostic application) from the Receiving Transport Protocol Entity that a message has been received.

DataLength: Number of data bytes in the data field Data[] that has been correctly received. The valid range of the DataLength parameter is 1 to 255.

Data[]: The data of the received message.

A.1.2

State Transition Table

The Receiver Transport Protocol Entity can reside in either of two states, IDLE or RECEIVING.

	Current State	Event	Action(s)	Next State
R1	IDLE	SF TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0000b) && (TPDU.TPCI.R == 0) && (TPDU.TPCI.DL <= 6) /* Single Frame data transfer */	DataLength = TPDU.TPCI.DL; Data[0..DataLength-1] = TPDU.DATA[0..DataLength-1]; T_Data.Ind(DataLength, Data[]);	IDLE
R2	IDLE	FF TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0001b) && (TPDU.TPCI.XDL == 0) /* Segmented data transfer */	DataLength = TPDU.DL; Data[0..4] = TPDU.DATA[0..4]; RcvdNrOfData = 5; NextSN = 0; SegmCnt = 1; TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0011b; TPDU.TPCI.FCS = 0; TPDU.BS = cThisNodeBS; TPDU.ST = cThisNodeST; D_Data.Req(TPDU); /* Send Connect FC TPDU */	RECEIVING
R3	RECEIVING	Not Last CF TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0010b) && (TPDU.TPCI.SN == NextSN) && ((DataLength - RcvdNrOfData) > 6) && (SegmCnt < cThisNodeBS)	Data[RcvdNrOfData..RcvdNrOfData+5] = TPDU.DATA[0..5]; RcvdNrOfData = RcvdNrOfData + 6; NextSN = (NextSN + 1) MOD 16; SegmCnt = SegmCnt + 1;	RECEIVING
R4	RECEIVING	Not Last CF TPDU received but last segment (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0010b) && (TPDU.TPCI.SN == NextSN) && ((DataLength - RcvdNrOfData) > 6) && (SegmCnt == cThisNodeBS)	Data[RcvdNrOfData..RcvdNrOfData+5] = TPDU.DATA[0..5]; RcvdNrOfData = RcvdNrOfData + 6; NextSN = (NextSN + 1) MOD 16; SegmCnt = 1; TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0011b; TPDU.TPCI.FCS = 0; TPDU.BS = cThisNodeBS; TPDU.ST = cThisNodeST; D_Data.Req(TPDU); /* Send FC TPDU */	RECEIVING
R5	RECEIVING	Last CF TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0010b) && (TPDU.TPCI.SN == NextSN) && ((DataLength - RcvdNrOfData) <= 6) /* Last Segment received in a segmented data transfer */	Data[RcvdNrOfData..DataLength-1] = TPDU.DATA[0..DataLength-RcvdNrOfData-1]; T_Data.Ind(DataLength, Data[]); /* Complete message received */	IDLE
R6	RECEIVING	Incorrect CF TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0010b) && (TPDU.TPCI.SN != NextSN) /* Incorrect Segment Number in CF TPDU */		IDLE
R7	RECEIVING	SF TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0000b) && (TPDU.TPCI.R == 0) && (TPDU.TPCI.DL <= 6) /* New request for Single Frame data transfer that aborts the current transfer */	DataLength = TPDU.TPCI.DL; Data[0..DataLength-1] = TPDU.DATA[0..DataLength-1]; T_Data.IND(DataLength, Data[]); /* Complete message received */	IDLE
R8	RECEIVING	FF TPDU received	DataLength = TPDU.DL;	RECEIVING

	IVING	(TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0001b) && (TPDU.TPCI.XDL == 0) /* New request for Segmented data transfer that aborts the current transfer */	Data[0..4] = TPDU.DATA[0..4]; RcvdNrOfData = 5; NextSN = 0; SegmCnt = 1; TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0011b; TPDU.TPCI.FCS = 0; TPDU.BS = cThisNodeBS; TPDU.ST = cThisNodeST; D_Data.Req(TPDU); /* Send Connect FC TPDU */	IVING
--	-------	--	---	-------

A.1.3

Actions description

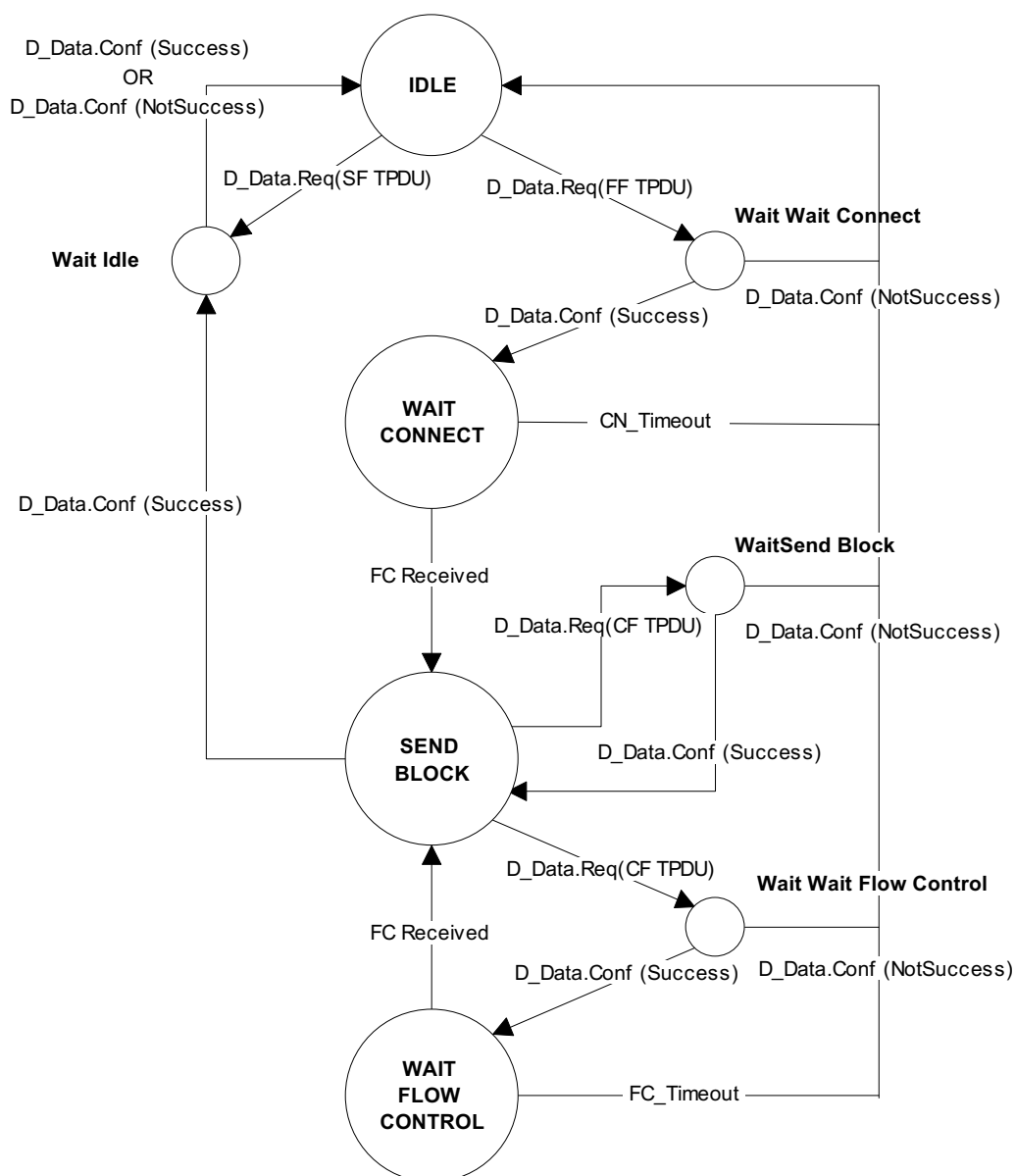
D_Data.Req(TPDU)

The Receiver TPE use this service to request the CAN data link layer for sending the FC TPDU message on the CAN bus.

The D_Data.Req() service shall use the CAN ID according to the project specifications when building the CAN frame. The data length in the CAN frame shall always be set to byte size 8. The TPDU shall be mapped on the CAN frame so that the first data byte in the CAN frame corresponds to the first byte in the TPDU and the second data byte in the CAN frame is the second TPDU byte etc.

Sender Transport Protocol Entity

A graphical overview of the Sender TPE is shown by the State Transition Diagram that follows.



Service primitives

T Data.Reg(DataLength, Data[])

The service user (the diagnostic application) uses this primitive to request a transfer of a message to the Sender Transport Protocol Entity.

DataLength: Number of data bytes in the data field Data[] that the service user requests the Transport Protocol Entity to send. The valid range of the DataLength parameter is 1 to 255.

Data[]: The data to be transferred.

T_Data.Conf(Status)

Confirmation sent from the Sender Transport Protocol Entity to the service user (the Diagnostic application). The Status parameter can have two values, Success or NotSuccess.

Status == Success Means that the requested message transfer is ready and that the complete message has been successfully sent.

Status == NotSuccess Means that the requested message transfer is ready but the message was not successfully sent.

A.2.2**State Transition Table**

The Transport Protocol Sender Entity can reside in either of four main states, *IDLE*, *WAIT CONNECT*, *SEND BLOCK* or *WAIT FLOW CONTROL*. The Transport Protocol Sender Entity has also four sub states where it waits for confirmation from the Data Link Layer each time a CAN frame has been requested for transmission. These sub states are: *Wait Idle*, *Wait Wait Connect*, *Wait Send block* and *Wait Wait Flow Control*.

	Current State	Event	Action(s)	Next State
S1	IDLE	T_Data.Req(DataLength, Data[]) received (DataLength <= 6) /* Single Frame data transfer */	TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0x0000b; TPDU.TPCI.R = 0; TPDU.TPCI.DL = DataLength; TPDU.DATA[0..DataLength-1] = Data[0..DataLength-1]; D_Data.Req(TPDU);	Wait Idle
S2	Wait Idle	D_Data.Conf(Success)	T_Data.Conf(Success);	IDLE
S3	Wait Idle	D_Data.Conf(NotSuccess)	T_Data.Conf(NotSuccess);	IDLE
S4	IDLE	T_Data.Req(DataLength, Data[]) received (DataLength > 6) /* Segmented data transfer */	TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0x0001b; TPDU.TPCI.XDL = 0; TPDU.DL = DataLength; TPDU.DATA[0..4] = Data[0..4]; SndNrOfData = 5; D_Data.REQ(TPDU);	Wait Wait Connect
S5	Wait Wait Connect	D_Data.Conf(Success)	SetTimer(CN_Timer, cCN_Timeout);	WAIT CONNECT
S6	Wait Wait Connect	D_Data.Conf(NotSuccess)	T_Data.Conf(NotSuccess);	IDLE
S7	WAIT CONNECT	FC TPDU received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0011b) /* Connect FC TPDU received */	BlockSize = TPDU.BS; ST_Timeout = TPDU.ST; SetTimer(ST_Timer, ST_Timeout); SegmCnt = 1; NextSN = 0;	SEND BLOCK
S8	WAIT CONNECT	Timeout CN_Timer /* No Connect FC TPDU received in time */	T_Data.Conf(NotSuccess);	IDLE
S9	SEND BLOCK	Timeout ST_Timer ((DataLength - SndNrOfData) > 6) && (SegmCnt < BlockSize) /* Not last segment to send in message, and not last segment to send in block */	TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0010b; TPDU.TPCI.SN = NextSN; TPDU.DATA[0..5] = Data[SndNrOfData..SndNrOfData+5]; SndNrOfData = SndNrOfData + 6; D_Data.Req(TPDU);	Wait Send block
S10	Wait Send block	D_Data.Conf(Success)	SetTimer(ST_Timer, ST_Timeout); SegmCnt = SegmCnt + 1; NextSN = (NextSN + 1) MOD 16	SEND BLOCK
S11	Wait Send block	D_Data.Conf(NotSuccess)	T_Data.Conf(NotSuccess);	IDLE
S12	SEND	Timeout ST_Timer	TPDU.TA = cDiagTargetAddr;	Wait

	BLOCK	((DataLength - SndNrOfData) > 6) && (SegmCnt == BlockSize) /* Not last segment to send in message, but last segment to send in block */	TPDU.TPCI.OC = 0010b; TPDU.TPCI.SN = NextSN; TPDU.DATA[0..5] = Data[SndNrOfData..SndNrOfData+5]; SndNrOfData = SndNrOfData + 6; D_Data.Req(TPDU);	Wait Flow Control
S13	Wait Wait Flow Control	D_Data.Conf(Success)	SetTimer(FC_Timer, cFC_Timeout); NextSN = (NextSN + 1) MOD 16	WAIT FLOW CON- TROL
S14	Wait Wait Flow Control	D_Data.Conf(NotSuccess)	T_Data.Conf(NotSuccess);	IDLE
S15	SEND BLOCK	Timeout ST_Timer ((DataLength - SndNrOfData) <= 6) /* Last segment to send in message*/	TPDU.TA = cDiagTargetAddr; TPDU.TPCI.OC = 0010b; TPDU.TPCI.SN = NextSN; TPDU.DATA[0..DataLength-SndNrOfData-1] = Data[SndNrOfData..DataLength-1]; D_Data.Req(TPDU);	Wait Idle
S16	WAIT FLOW CONTROL	FC TPDU Received (TPDU.TA == cThisNodeDiagAddr) && (TPDU.TPCI.OC == 0011b) /* Flow Control received */	SetTimer(ST_Timer, 0); SegmCnt = 1; /* ST_Timer set to immediate timeout! */	SEND BLOCK
S17	WAIT FLOW CONTROL	Timeout FC_Timer /*No Flow Control received in time */	T_Data.Conf(NotSuccess);	IDLE

A.2.3

Actions description

SetTimer(TimerID, Timeout):

The Sender TPE use this service to start a timer (TimerID) that shall generate an event after a specified time (TimeoutValue) has elapsed.

TimerID: A unique timer identification for the timer event.

TimeoutValue: Time out time in milliseconds.

D_Data.Req(TPDU):

The Sender TPE use the service D_Data.Req() to request the CAN Data Link Layer to transmit the TPDU on the CAN bus. When the Data Link Layer receives a D_Data.Req() from the sender TPE, the CAN Data Link Layer shall set a DL_Timer to DL_Timeout. If the DL_Timer times out before the CAN frame has been successfully transmitted, the Data Link Layer shall abort the requested transmission and send a confirmation with status = NotSuccess to the TPE. If the transmission was successful ended before DL_Timeout a confirmation with status = Success shall be sent to the TPE. See description of D_Data.Conf() below.

The D_Data.Req() shall use CAN ID according to project specifications when building the CAN frame. The data length in the CAN frame shall always be set to byte size 8. The TPDU shall be mapped on the CAN frame so that the first data byte in the CAN frame corresponds to the first data byte of the TPDU and the second data byte in the CAN frame is the second TPDU byte, etc.

A.2.4

Events description

D_Data.Conf(Status):

The Sender TPE shall use the service D_Data.Conf() to get confirmation from the CAN data link layer, that the service D_Data.Req() has been successful or not.

Status == Success: The CAN data link layer has successfully transmitted the TPDU CAN frame, within the DL_Timeout time.

Status == NotSuccess: The CAN data link layer has failed to transmit the TPDU CAN frame, within the DL_Timeout time and the transmission of the TPDU CAN frame has been aborted.

A.3**Timing**

Below is each Transport Protocol Entities different reaction time specified.

A.3.1**Receiver TPE timing**

FF_FC_ReactionTime: The reaction time from FF TPDU received to next FC TPDU requested for transmission by TPE
min. = 0ms, max. = 30ms.

A.3.2**Sender TPE timing**

FC_CF_ReactionTime: The reaction time from FC TPDU received to next CF TPDU requested for transmission by TPE.
min. = 0ms, max. = 30ms.

Separation Time: The time from confirmation from data link layer of CF TPDU requested for transmission by TPE to next CF TPDU requested for transmission by TPE.
min. = ST_Timeout, max. = ST_Timeout + 30ms.

A.4**Constants**

Constants used by the Transport Protocol.

cCN_Timeout: The time out between FF TPDU requested for transmission by TPE to next FC TPDU received time out
min = 200ms, max. = 250ms.

cFC_Timeout: Time out between last CF TPDU in block requested for transmission by TPE to next FC TPDU received time out. min = 200ms, max. = 250ms.

cThisNodeBS: The block size for a specific node.

cThisNodeST: The separation time for a specific node.

cDiagTargetAddr: See section 4.4 in this document.

cThisNodeDiagAddr: See section 4.4 in this document.