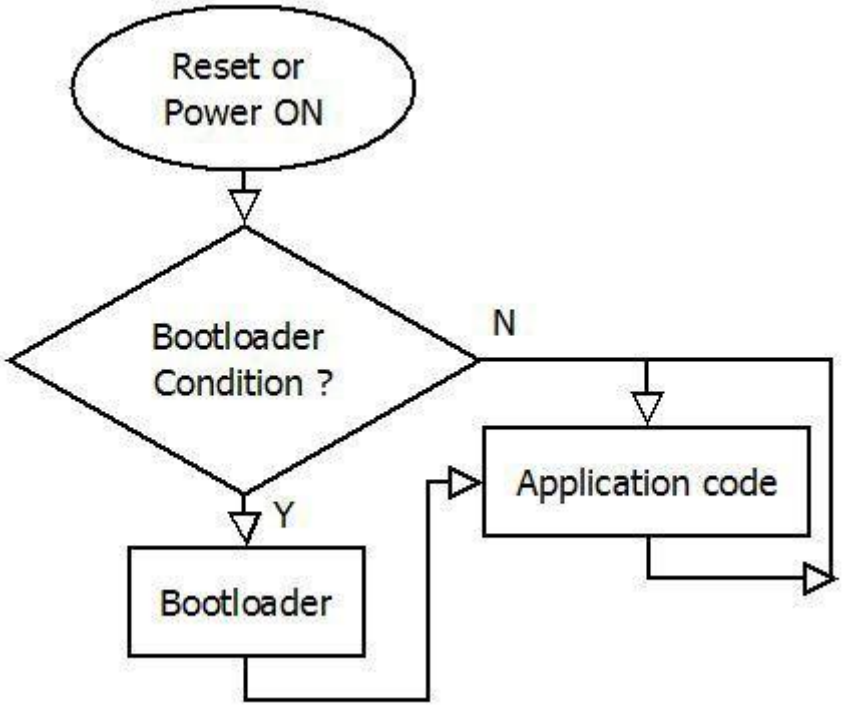
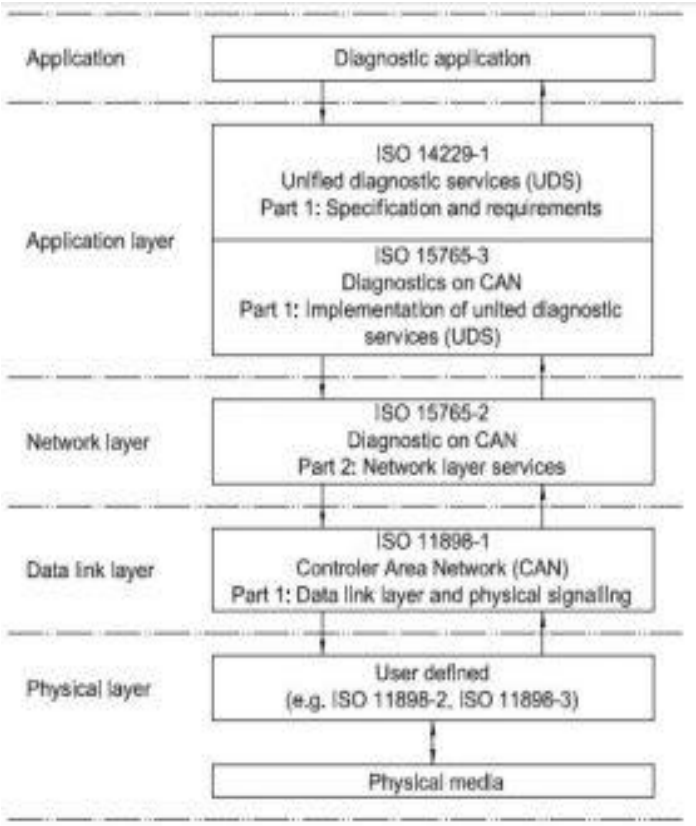
NANOCHIP SOLUTIONS

Unified Diagnostic Services (UDS)

SW Control Flow

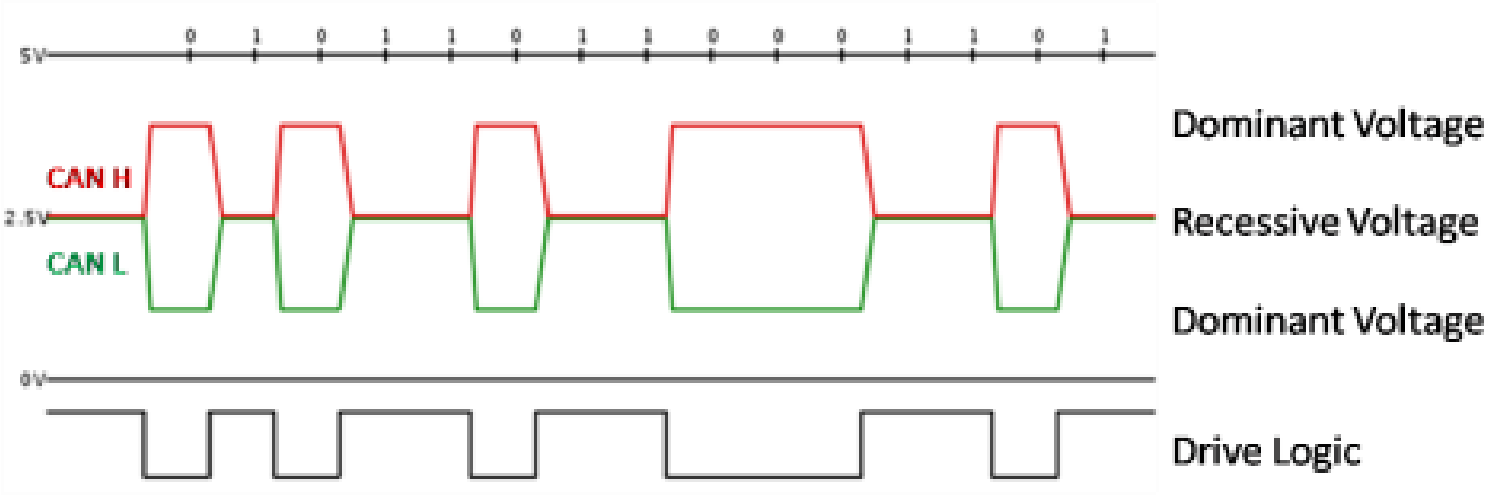


UDS Stack

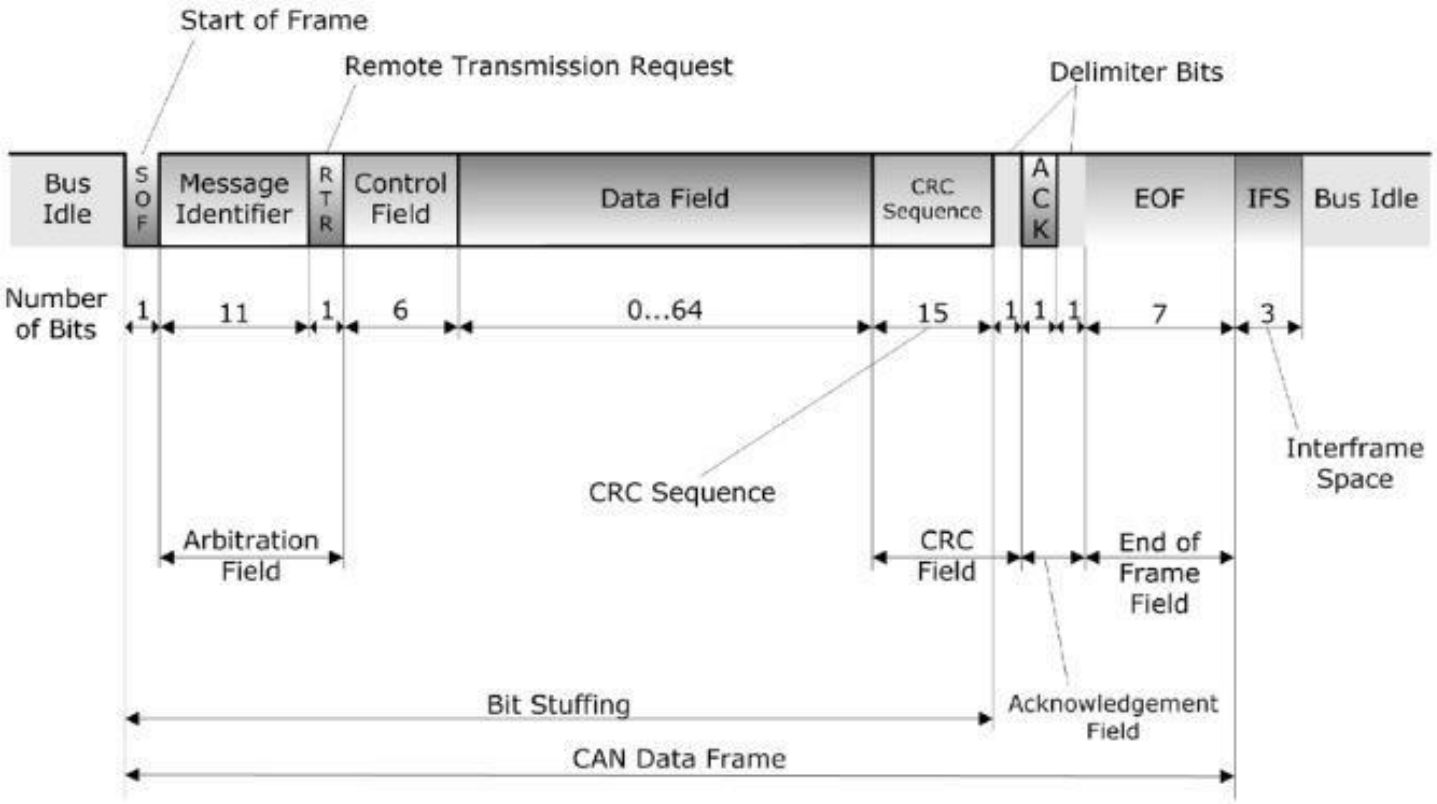


Layer	Description	Standards for UDS
„8“	Diagnostic Application	User
7	Application Layer	ISO 14229-1 ISO 15765-3
6	Presentation Layer	Not applicable
5	Session Layer	ISO 15765-3
4	Transport layer	ISO 15765-2
3	Network Layer	ISO 15765-2
2	Data Link Layer	ISO 11898-1
1	Physical Layer	ISO 11898-2* ISO 11898-3

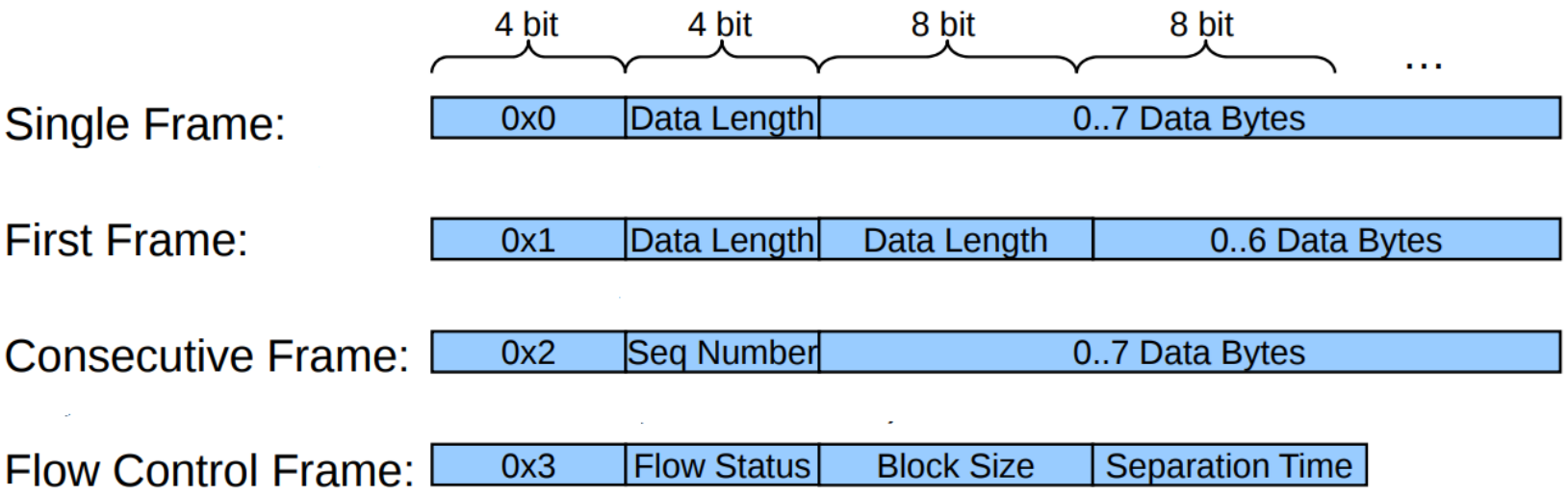
Physical Media



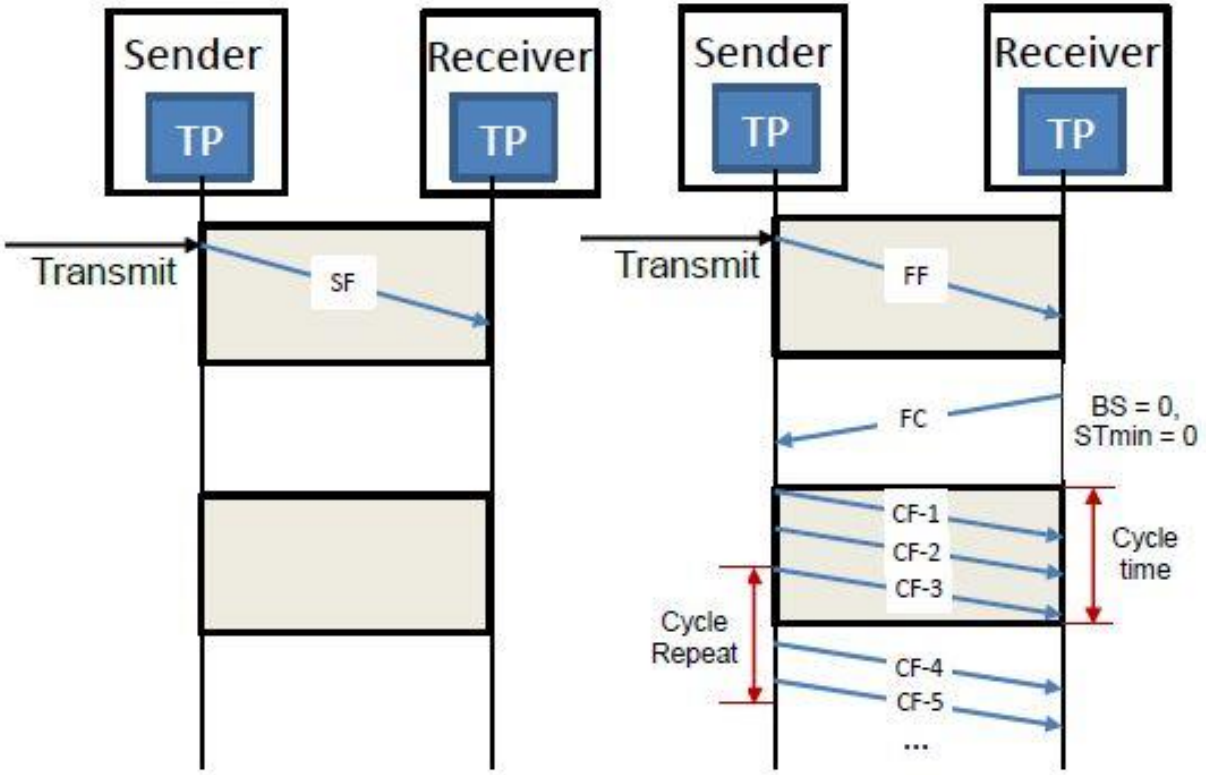
Physical Layer



Transport Layer



Transport Layer



Unsegmented Data Transfer

Segmented Data Transfer

Transport Layer

Flow Status

Hex value	Description
0	**ContinueToSend (CTS)** The FlowControl ContinueToSend parameter shall be encoded by setting the lower nibble of the N_PCI byte #1 to "0". It shall cause the sender to resume the sending of Consecutive frames. The meaning of this value is that the receiver is ready to receive a maximum of BS number of Consecutive frames.
1	**Wait (WT)** The FlowControl Wait parameter shall be encoded by setting the lower nibble of the N_PCI byte #1 to "1". It shall cause the sender to continue to wait for a new FlowControl N_PDU and to restart its N_BS timer.
2	**Overflow (OVFLW)** The FlowControl Overflow parameter shall be encoded by setting the lower nibble of the N_PCI byte #1 to "2". It shall cause the sender to abort the transmission of a segmented message and make an N_USData.confirm service call with the parameter <N_Result>=N_BUFFER_OVFLW. This N_PCI FlowStatus parameter value is only allowed to be transmitted in the FlowControl N_PDU that follows the FirstFrame N_PDU and shall only be used in case the message length FF_DL of the received FirstFrame N_PDU exceeds the buffer size of the receiving entity.
3 – F	**Reserved** This range of values is reserved by this part of ISO 15765.

Transport Layer

Block Size

Hex value	Description
00	**BlockSize (BS)** The BS parameter value zero (0) shall be used to indicate to the sender that no more FC frames shall be sent during the transmission of the segmented message. The sending network layer entity shall send all remaining consecutive frames without any stop for further FC frames from the receiving network layer entity.
01 – FF	**BlockSize (BS)** This range of BS parameter values shall be used to indicate to the sender the maximum number of consecutive frames that can be received without an intermediate FC frame from the receiving network entity.

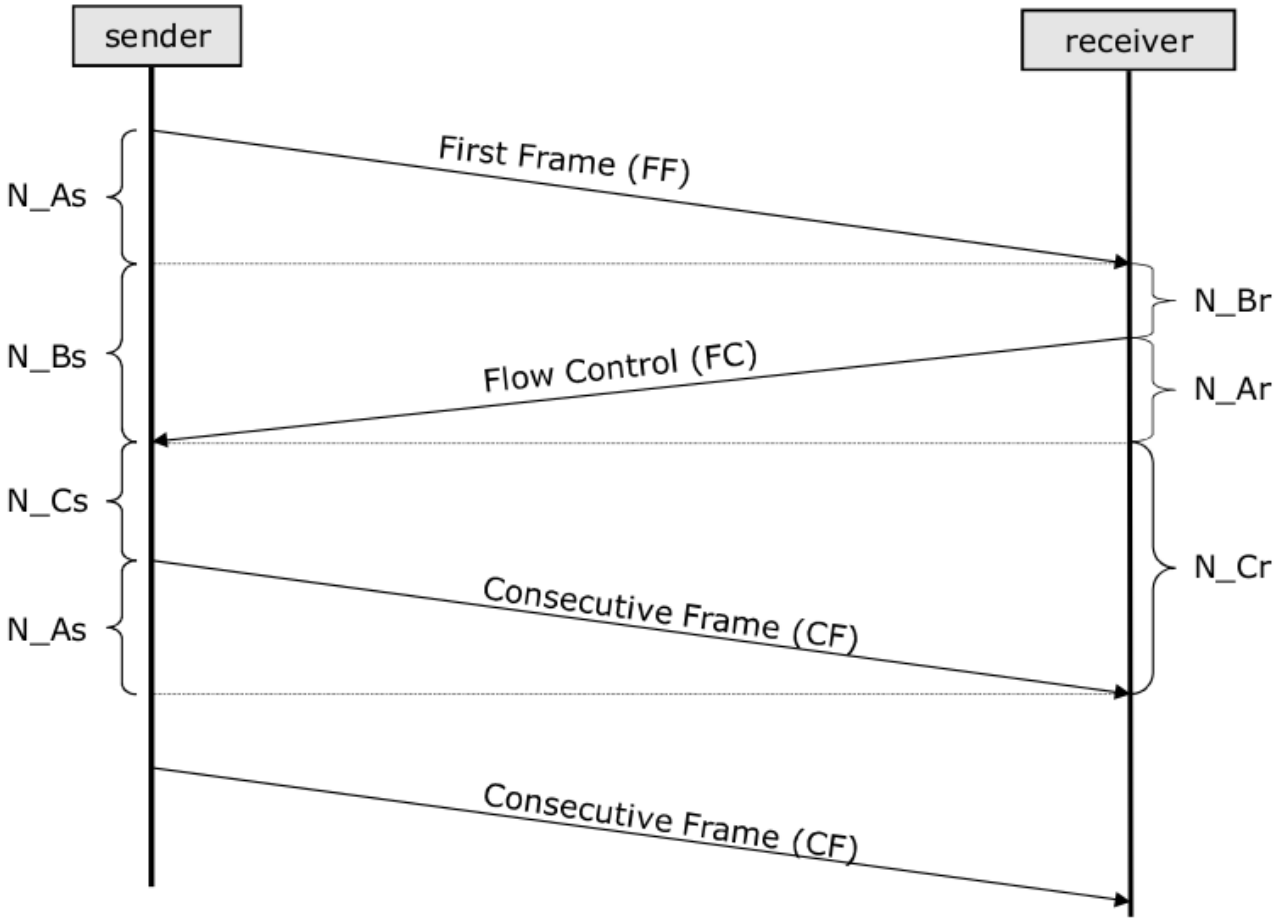
Separation Time

Hex value	Description
00 – 7F	**SeparationTime (STmin) range: 0 ms – 127 ms** The units of STmin in the range 00 hex – 7F hex are absolute milliseconds (ms).
80 – F0	**Reserved** This range of values is reserved by this part of ISO 15765.
F1 – F9	**SeparationTime (STmin) range: 100 µs – 900 µs** The units of STmin in the range F1 hex – F9 hex are even 100 microseconds (µs), where parameter value F1 hex represents 100 µs and parameter value F9 hex represents 900 µs.
FA – FF	**Reserved** This range of values is reserved by this part of ISO 15765.

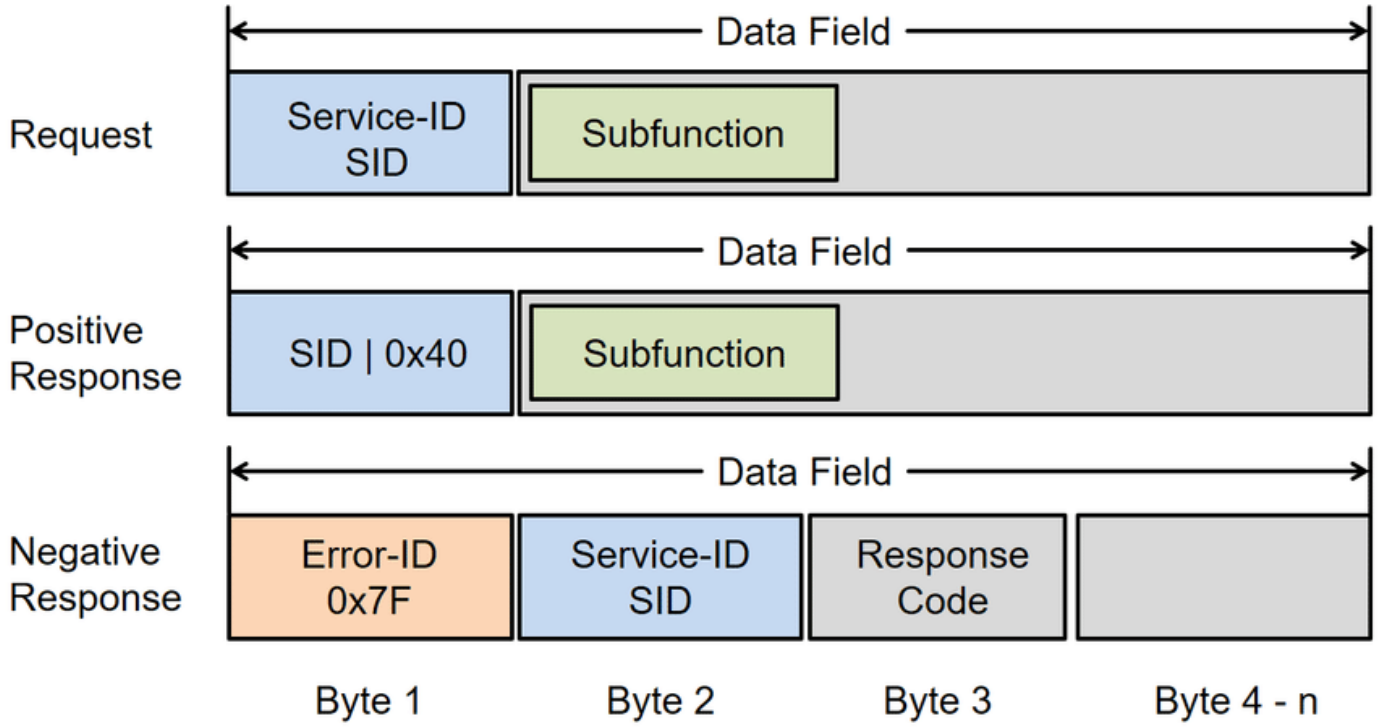
Transport Layer

- CAPL script to illustrate unsegmented data transfer
- CAPL script to illustrate segmented data transfer

Network Layer



Application Layer



Application Layer

Sl.No	UDS Services	Service Identifiers
1	DiagnosticSessionControl	10
2	ECUReset	11
3	SecurityAccess	27
4	TesterPresent	3E
5	ReadDataByIdentifier	22
6	WriteDataByIdentifier	2E
7	ReadMemoryByAddress	23
8	WriteMemoryByAddress	3D
9	CommunicationControl	28
10	ControlDTCSetting	85
11	RoutineControl	31
12	RequestDownload	34
13	TransferData	36
14	RequestTransferExit	37
15	RequestUpload	35

Application Layer

NRC Hex Value	Response Code
11	serviceNotSupported
12	subFunctionNotSupported
13	incorrectMessageLengthOrInvalidFormat
22	conditionsNotCorrect
24	requestSequenceError
31	requestOutOfRange
33	securityAccessDenied
35	invalidKey
36	exceedNumberOfAttempts
37	requiredTimeDelayNotExpired
70	uploadDownloadNotAccepted
71	transferDataSuspended
72	generalProgrammingFailure
73	wrongBlockSequenceCounter
78	requestCorrectlyReceived-ResponsePending
7E	subFunctionNotSupportedInActiveSession
7F	serviceNotSupportedInActiveSession

Diagnostic Session Control - $10

Service	Sub-Function	Description
Diagnostic Session Control (10h)	ECU Programming Session (02)	Causes the ECU to enter the boot loader to prepare for software download. PERFORM IMMEDIATE ECU RESET WITHOUT WRITING TO EEPROM
	ECU Extended Diagnostic Session (03)	Causes the ECU to enter a special diagnostic mode. This mode will automatically exit if a diagnostic message addressed to the ACC ECU is not received every 5 seconds.
	Default Session (01)	Returns to normal mode. WRITE KEEP- ALIVE VALUES TO EEPROM AND FORCE COP RESET

Diagnostic Session Control - $10

Message ID	Dir	DLC	DATA							
706	Tx	8	02	10	01	00	00	00	00	00
70E	Rx	8	02	50	01	00	00	00	00	00
706	Tx	8	02	10	02	00	00	00	00	00
70E	Rx	8	06	50	02	00	14	01	F4	00
706	Tx	8	02	10	03	00	00	00	00	00
70E	Rx	8	06	50	03	00	32	01	F4	00

Diagnostic Session Control - $10

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	DiagnosticSessionControl Response SID	M	0x50	DSCPR
#2	sub-function = [diagnosticSessionType]	M	0x00 – 0xFF	LEV_DS_
#3 : #6	sessionParameterRecord[]#1 = [data#1 : data#4]	M : M	0x00 – 0xFF : 0x00 – 0xFF	SPREC_ DATA_1 : DATA_m

Byte pos. in record	Description	Cvt	Byte Value	Mnemonic
	sessionParameterRecord[] = [			SPREC_
#1	$P2_{Server\_max}$ (high byte)	M	0x00 – 0xFF	P2SMH
#2	$P2_{Server\_max}$ (low byte)	M	0x00 – 0xFF	P2SML
#3	$P2^*_{Server\_max}$ (high byte)	M	0x00 – 0xFF	P2ESMH
#4	$P2^*_{Server\_max}$ (low byte)]	M	0x00 – 0xFF	P2ESML

Table 29 — sessionParameterRecord content definition

Parameter	Description	# of bytes	Resolution	minimum value	maximum value
$P2_{Server\_max}$	Default $P2_{Server\_max}$ timing supported by the server for the activated diagnostic session.	2	1 ms	0 ms	65 535 ms
$P2^*_{Server\_max}$	Enhanced (NRC 0x78) $P2_{Server\_max}$ supported by the server for the activated diagnostic session.	2	10 ms	0 ms	655 350 ms

NRC	Description	Mnemonic
0x12	**sub-functionNotSupported** This NRC shall be sent if the sub-function parameter is not supported.	SFNS
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be returned if the criteria for the request DiagnosticSessionControl are not met.	CNC

Ecu Reset - $11

Service	Sub-Function	Description
ECU Reset (11h)	Hard Reset (01h)	This value identifies a "hard reset" condition which simulates the power-on / start-up sequence typically performed after an ECU has been previously disconnected from its power supply (i.e. battery). Perform shutdown routines before reset occurs. Update the keep-alive variables and then force a COP reset (ECUReset)

Message ID	Dir	DLC	DATA							
706	Tx	8	02	11	01	00	00	00	00	00
70E	Rx	8	02	51	01	00	00	00	00	00

Ecu Reset - $11

— Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	ECUReset Response SID	M	0x51	ERPR
#2	sub-function = [resetType]	M	0x00 – 0x7F	LEV_RT_
#3	powerDownTime	C	0x00 – 0xFF	PDT
C: This parameter is present if the sub-function parameter is set to the enableRapidPowerShutDown value (0x04);				

Positive response message data-parameter definition

Table 36 defines the data-parameters of the response message.

Response message data-parameter definition

Definition
resetType This parameter is an echo of bits 6 - 0 of the sub-function parameter from the request message.
powerDownTime This parameter indicates to the client the minimum time of the stand-by-sequence the server will remain in the power down sequence. The resolution of this parameter is one (1) second per count. The follwing values are valid: — 0x00 – 0xFE: 0 – 254 seconds powerDownTime, — 0xFF: indicates a failure or time not available.

– Supported negative response codes

NRC	Description	Mnemonic
0x12	**sub-functionNotSupported** This NRC shall be sent if the sub-function parameter is not supported.	SFNS
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be returned if the criteria for the ECUReset request is not met.	CNC
0x33	**securityAccessDenied** This NRC shall be sent if the requested reset is secured and the server is not in an unlocked state.	SAD

Security Access - $27

Service	Sub-Function	Description
Security Access (27h)	requestSeed (01h)	Client requests for a randomly generated 3-byte seed at different levels.
	sendKey (02h)	Server unlocks if key sent by client matches internal key at different levels.

Message ID	Dir	DLC	DATA							
706	Tx	8	02	27	01	00	00	00	00	00
70E	Rx	8	05	67	01	Seed1	Seed2	Seed3	00	00
706	Tx	8	05	27	02	Key 1	Key 2	Key 3	00	00
70E	Rx	8	02	67	02	00	00	00	00	00

Security Access - $27

— Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	SecurityAccess Response SID	M	67	SAPR
#2	sub-function = [securityAccessType]	M	00-7F	LEV_SAT_SK
#3 : #n	securitySeed[] = [seed#1 (high byte) : seed#m (low byte)]	C : C	0x00 – 0xFF : 0x00 – 0xFF	SECSEED_ SEED1HB : SEEDmLB
C: The presence of this p[illegible] on the securityAccessType parameter. It is mandatory to be present if the securityAccessType parameter indicates that the client wants to retrieve the seed from the server.				

- Supported negative response codes

NRC	Description	Mnemonic
0x12	**sub-functionNotSupported** This NRC shall be sent if the sub-function parameter is not supported.	SFNS
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be returned if the criteria for the request SecurityAccess are not met.	CNC
0x24	**requestSequenceError** Send if the 'sendKey' sub-function is received without first receiving a 'requestSeed' request message.	RSE
0x31	**requestOutOfRange** This NRC shall be sent if the user optional securityAccessDataRecord contains invalid data.	ROOR
0x35	**invalidKey** Send if an expected 'sendKey' sub-function value is received and the value of the key does not match the server's internally stored/calculated key.	IK
0x36	**exceededNumberOfAttempts** Send if the delay timer is active due to exceeding the maximum number of allowed false access attempts.	ENOA
0x37	**requiredTimeDelayNotExpired** Send if the delay timer is active and a request is transmitted.	RTDNE

Tester Present - $3E

Service	Sub-Function	Description
Tester Present (3Eh)	zeroSubFunction (00h)	Message sent by tester to keep non-default diagnostic sessions active.

Message ID	Dir	DLC	DATA							
706	Tx	8	02	3E	00	00	00	00	00	00
70E	Rx	8	02	7E	00	00	00	00	00	00

Tester Present - $3E

– Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	TesterPresent Response SID	M	0x7E	TPPR
#2	sub-function = [zeroSubFunction]	M	0x00	LEV_ZSUBF

· Supported negative response codes

NRC	Description	Mnemonic
0x12	**sub-functionNotSupported** This NRC shall be sent if the sub-function parameter is not supported.	SFNS
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF

Read or Write Data By Identifier - $22 / $2E

Service	Description
Read Data By Identifier (22h)	The tester requests one or more bytes of data records identified by a 2-byte data record identifier. The format and definition of the data records shall be vehicle manufacturer or ECU specific, and may include analog input and output signals, digital input and output signals, internal data, ECU status, ECU and system configuration, and ECU identification data.
Write Data By Identifier (2Eh)	Similar to ReadDataByIdentifer, but allows writes to certain data records.

Message ID	Dir	DLC	DATA							
706	Tx	8	05	2E	A2	26	FF	38	00	00
70E	Rx	8	03	6E	A2	26	00	00	00	00
706	Tx	8	03	2E	A2	26	00	00	00	00
70E	Rx	8	05	6E	A2	26	FF	38	00	00

Read Data By Identifier - $22

– ReadDataByIdentifier positive response message example #3

Message direction	server → client		
Message Type	Response		
A_Data Byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	ReadDataByIdentifier Response SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0x01	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x04	DID_B2
#4	dataRecord [data#1]	0xXX	DREC_DATA1
#5	dataRecord [data#2]	0xXX	DREC_DATA2
#6	dataRecord [data#3]	0xXX	DREC_DATA3
#7	dataRecord [data#4]	0xXX	DREC_DATA4
#8	dataRecord [data#5]	0xXX	DREC_DATA5
#9	dataRecord [data#6]	0xXX	DREC_DATA6
#10	dataRecord [data#7]	0xXX	DREC_DATA7
#11	dataRecord [data#8]	0xXX	DREC_DATA8
#12	dataRecord [data#9]	0xXX	DREC_DATA9
#13	dataRecord [data#10]	0xXX	DREC_DATA10
#14	dataRecord [data#11] data content of byte#11: 0x14	0x14	DREC_DATA11
#15	dataRecord [data#12] data content of byte#12: 0x7B	0x7B	DREC_DATA12
:	:	:	:

Write Data By Identifier - $2E

Table 237 — WriteDataByIdentifier positive response message flow example #1

Message direction	**server → client**		
Message Type	**Response**		
A_Data Byte	**Description (all values are in hexadecimal)**	**Byte Value**	**Mnemonic**
#1	WriteDataByIdentifier Response SID	0x6E	WDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x90	DID_B2

ReadMemoryByAddress - $23

Table 151 — Request message definition

A_Data Byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	ReadMemoryByAddress Request SID	M	0x23	RMBA
#2	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#3 : #(m-1)+3	memoryAddress[] = [byte#1 (MSB) : byte#m]	M : C1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#n-(k-1) : #n	memorySize[] = [byte#1 (MSB) : byte#k]	M : C2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
C1: The presence of this parameter depends on address length information parameter of the addressAndLengthFormatIdentifier				
C2: The presence of this parameter depends on the memory size length information of the addressAndLengthFormatIdentifier.				

This service does not use a sub-function parameter.

ReadMemoryByAddress - $23

Table 153 — Positive response message definition

A_Data Byte	Parameter Name	Cvt
#1	ReadMemoryByAddress Response SID	M
#2 : #n	dataRecord[] = [data#1 : data#m]	M : U

Table 155 — Supported negative response codes

NRC	Description	Mnemonic
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be sent if the operating conditions of the server are not met to perform the required action.	CNC
0x31	**requestOutOfRange** This NRC shall be sent if: — Any memory address within the interval [0xMA, (0xMA + 0xMS -0x1)] is invalid; — Any memory address within the interval [0xMA, (0xMA + 0xMS -0x1)] is restricted; — The memorySize parameter value in the request message is not supported by the server; — The specified addressAndLengthFormatIdentifier is not valid; — The memorySize parameter value in the request message is zero;	ROOR
0x33	**SecurityAccessDenied** This NRC shall be sent if any memory address within the interval [0xMA, (0xMA + 0xMS - 0x1)] is secure and the server is locked.	SAD

WriteMemoryByAddress - $3D

Table 238 — Request message definition

A_Data Byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	WriteMemoryByAddress Request SID	M	0x3D	WMBA
#2	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#3 : #m+2	memoryAddress[] = [byte#1 (MSB) : byte#m]	M : C1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#n-r-2-(k-1) : #n-r-2	memorySize[] = [byte#1 (MSB) : byte#k]	M : C2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
#n-(r-1) : #n	dataRecord[] = [data#1 : data#r]	M : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_r
C1: The presence of this parameter depends on address length information parameter of the addressAndLengthFormatIdentifier				
C2: The presence of this parameter depends on the memory size length information of the addressAndLengthFormatIdentifier.				

WriteMemoryByAddress - $3D

Table 241 — Response message data-parameter definition

Definition
addressAndLengthFormatIdentifier This parameter is an echo of the addressAndLengthFormatIdentifier from the request message.
memoryAddress This parameter is an echo of the memoryAddress from the request message.
memorySize This parameter is an echo of the memorySize from the request message.

Table 242 — Supported negative response codes

NRC	Description	Mnemonic
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be sent if the operating conditions of the server are not met to perform the required action.	CNC
0x31	**requestOutOfRange** This NRC shall be sent if: — Any memory address within the interval [0xMA, (0xMA + 0xMS -0x1)] is invalid; — Any memory address within the interval [0xMA, (0xMA + 0xMS -0x1)] is restricted; — The memorySize parameter value in the request message is not supported by the server; — The specified addressAndLengthFormatIdentifier is not valid; — The memorySize parameter value in the request message is zero;	ROOR
0x33	**securityAccessDenied** This NRC shall be sent if any memory address within the interval [0xMA, (0xMA + 0xMS -0x1)] is secure and the server is locked.	SAD
0x72	**generalProgrammingFailure** This NRC shall be returned if the server detects an error when writing to a memory location.	GPF

WriteMemoryByAddress - $3D

Table 243 — WriteMemoryByAddress request message flow example #1

Message direction	client → server		
Message Type	Request		
A_Data Byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	WriteMemoryByAddress Request SID	0x3D	WMBA
#2	addressAndLengthFormatIdentifier	0x12	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2] (LSB)	0x48	MA_B2
#5	memorySize [byte#1]	0x02	MS_B1
#6	dataRecord [data#1]	0x00	DREC_DATA_1
#7	dataRecord [data#2]	0x8C	DREC_DATA_2

Table 244 — WriteMemoryByAddress positive response message flow example #1

Message direction	server → client		
Message Type	Response		
A_Data Byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	WriteMemoryByAddress Response SID	0x7D	WMBAPR
#2	addressAndLengthFormatIdentifier	0x12	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2] (LSB)	0x48	MA_B2
#5	memorySize [byte#1]	0x02	MS_B1

CommunicationControl - $28

Table 54 — Request message sub-function parameter definiti

Bits 6 – 0	Description	C
0x00	**enableRxAndTx** This value indicates that the reception and transmission of messages shall be enabled for the specified communicationType.	
0x01	**enableRxAndDisableTx** This value indicates that the reception of messages shall be enabled and the transmission shall be disabled for the specified communicationType.	
0x02	**disableRxAndEnableTx** This value indicates that the reception of messages shall be disabled and the transmission shall be enabled for the specified communicationType.	
0x03	**disableRxAndTx** This value indicates that the reception and transmission of messages shall be disabled for the specified communicationType.	
0x04	**enableRxAndDisableTxWithEnhancedAddressInformation** This value indicates that the addressed bus master shall switch the related sub-bus segment to the diagnostic-only scheduling mode.	
0x05	**enableRxAndTxWithEnhancedAddressInformation** This value indicates that the addressed bus master shall switch the related sub-bus segment to the application scheduling mode.	
0x06 – 0x3F	**ISOSAEReserved** This range of values is reserved by this document for future definition.	
0x40 – 0x5F	**vehicleManufacturerSpecific** This range of values is reserved for vehicle manufacturer specific use.	

CommunicationControl - $28

Table 56 — Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	CommunicationControl Response SID	M	0x68	CCPR
#2	sub-function = [controlType]	M	0x00 – 0x7F	LEV_CTRLTP

Table 58 — Supported negative response codes

NRC	Description	Mnemonic
0x12	**sub-functionNotSupported** This NRC shall be sent if the sub-function parameter is not supported.	SFNS
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** Used when the server is in a critical normal mode activity and therefore cannot disable/enable the requested communication type.	CNC
0x31	**requestOutOfRange** The server shall use this response code, if it detects an error in the communicationType or nodeIdentificationNumber parameter.	ROOR

DTC – Diagnostics Trouble Code

statusOfDTC: bit field name	Bit #
testFailed	0
testFailedThisOperationCycle	1
pendingDTC	2
confirmedDTC	3
testNotCompletedSinceLastClear	4
testFailedSinceLastClear	5
testNotCompletedThisOperationCycle	6
warningIndicatorRequested	7

Condition	DTC Status When Condition exists
Preconditions are not fulfilled and fault is not present	**0x50**
Preconditions are fulfilled and Fault is present	**0x2F/0xAF**
Preconditions are fulfilled and fault got latched	**0x2E/0xAE**
Hard reset after fault getting latched	**0x2C**
One more hard reset	**0x28**

ControlDTCSetting - $85

Table 87 — Request message sub-function parameter definiti

Bits 6 – 0	Description
0x00	**ISOSAEReserved** This value is reserved by this document.
0x01	**on** The server(s) shall resume the updating of diagnostic trouble code status bits according to normal operating conditions
0x02	**off** The server(s) shall stop the updating of diagnostic trouble code status bits.
0x03 – 0x3F	**ISOSAEReserved** This range of values is reserved by this document for future definition.
0x40 – 0x5F	**vehicleManufacturerSpecific** This range of values is reserved for vehicle manufacturer specific use.
0x60 – 0x7E	**systemSupplierSpecific** This range of values is reserved for system supplier specific use.
0x7F	**ISOSAEReserved** This value is reserved by this document for future definition.

ControlDTCSetting - $85

Table 89 — Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	ControlDTCSetting Response SID	M	0xC5	CDTCSPR
#2	DTCSettingType	M	00-7F	DTCSTP

Table 91 — Supported negative response codes

NRC	Description	Mnemonic
0x12	**sub-functionNotSupported** This NRC shall be sent if the sub-function parameter is not supported.	SFNS
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** Used when the server is in a critical normal mode activity and therefore cannot perform the requested DTC control functionality.	CNC

NRC	Description	Mnemonic
0x31	**requestOutOfRange** The server shall use this response code, if it detects an error in the DTCSettingControlOptionRecord.	ROOR

ControlDTCSetting - $85

Table 92 — ControlDTCSetting request message flow example #1

Message direction	client → server		
Message Type	Request		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	ControlDTCSetting Request SID	0x85	RDTCS
#2	DTCSettingType = off, suppressPosRspMsgIndicationBit = FALSE	0x02	DTCSTP_OFF

Table 93 — ControlDTCSetting positive response message flow example #1

Message direction	server → client		
Message Type	Response		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	ControlDTCSetting Response SID	0xC5	RDTCSPR
#2	DTCSettingType = off	0x02	DTCSTP_OFF

Control DTC settings - $85

Service	Sub-Function	Description
Control DTC Setting (85h)	On (01h)	Allow diagnostic self-tests to run.
	Off (02h)	Suspend diagnostic self-tests.

Message ID	Dir	DLC	DATA							
706	Tx	8	02	85	01	00	00	00	00	00
70E	Rx	8	02	C5	01	00	00	00	00	00
706	Tx	8	02	85	02	00	00	00	00	00
70E	Rx	8	02	C5	02	00	00	00	00	00

Application Layer

- CAPL script to illustrate security access clearance

Routine Control - $31

Service	Sub-Function	Description
Routine Control (31h)	startRoutine (01h)	Starts a diagnostic routine. ACC3 will support only two diagnostic routines one which will initiate manufacturing alignment, one which will run on-demand self-test (Global Control Routine).
	stopRoutine (02h)	Stops a diagnostic routine previously initiated with this service. For ACC3, the manufacturing alignment routine will run to completion, so this sub-function will always fail with a "ConditionsNotCorrect" fail code but the on-demand self-test routine can be aborted.
	Request Routine Results (03h)	Returns the status of the last diagnostic routine initiated with this service.

RESULT = "Completion Pass" (00) or "Completion Fail" (01) or "Routine Aborted" (02)

Routine Control - $31

Message ID	Dir	DLC	DATA							
706	Tx	8	04	31	01	02	02	00	00	00
70E	Rx	8	04	71	01	02	02	00	00	00
706	Tx	8	04	31	02	02	02	00	00	00
70E	Rx	8	04	71	02	02	02	00	00	00
706	Tx	8	04	31	03	02	02	00	00	00
70E	Rx	8	05	71	03	02	02	RESULT	00	00

Routine Control - $31

Table 385 — positive response message flow - example #1

Message direction	server → client		
Message Type	Response		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	RoutineControl Response SID	0x71	RCPR
#2	routineControlType = startRoutine	0x01	STR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2
#5	routineStatusRecord [routineStatus#1] = vehicle manufactuer specific	0x32	RRS_

Table 387 — RoutineControl positive response message flow - example #2

Message direction	server → client		
Message Type	Response		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	StopRoutine Response SID	0x71	RCPR
#2	routineControlType = stopRoutine	0x02	SPR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2
#5	routineStatusRecord [routineStatus#1] = vehicle manufactuer specific	0x30	RRS_

Message direction	server → client		
Message Type	Response		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	RoutineControl Response SID	0x71	RCPR
#2	routineControlType = requestRoutineResults	0x03	RRR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2
#5	routineStatusRecord [routineStatus#1] = Vehicle Manufactuer Specific	0x30	RRS_
#6	routineStatusRecord [routineStatus#2] = inputSignal#1	0x33	RRS_
:	:	:	:
#n	routineStatusRecord [routineStatus#m] = inputSignal#m	0x8F	RRS_

RequestDownload - $34

This service does not use a sub-function parameter.

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	RequestDownload Request SID	M	0x34	RD
#2	dataFormatIdentifier	M	0x00 – 0xFF	DFI_
#3	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#4 : #(m-1)+4	memoryAddress[] = [byte#1 (MSB) : byte#m]	M : C_1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#n-(k-1) : #n	memorySize[] = [byte#1 (MSB) : byte#k]	M : C_2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
C_1: The presence of this parameter depends on address length information parameter of the addressAndLengthFormatIdentifier C_2: The presence of this parameter depends on the memory size length information of the addressAndLengthFormatIdentifier.				

Request message definition

RequestDownload - $34

Request message data-parameter definition

Definition
dataFormatIdentifier This data-parameter is a one byte value with each nibble encoded separately. The high nibble specifies the "compressionMethod", and the low nibble specifies the "encryptingMethod". The value 0x00 specifies that neither compressionMethod nor encryptingMethod is used. Values other than 0x00 are vehicle manufacturer specific.
addressAndLengthFormatIdentifier This parameter is a one byte value with each nibble encoded separately (see H.1 for example values): — bit 7 - 4: Length (number of bytes) of the memorySize parameter — bit 3 - 0: Length (number of bytes) of the memoryAddress parameter
memoryAddress The parameter memoryAddress is the starting address of the server memory where the data is to be written to. The number of bytes used for this address is defined by the low nibble (bit 3 - 0) of the addressAndLengthFormatIdentifier. Byte#m in the memoryAddress parameter is always the least significant byte of the address being referenced in the server. The most significant byte(s) of the address can be used as a memory identifier. An example of the use of a memory identifier would be a dual processor server with 16 bit addressing and memory address overlap (when a given address is valid for either processor but yields a different physical memory device or internal and external flash is used). In this case, an otherwise unused byte within the memoryAddress parameter can be specified as a memory identifier used to select the desired memory device. Usage of this functionality shall be as defined by vehicle manufacturer / system supplier.
memorySize This parameter shall be used by the server to compare the memory size with the total amount of data transferred during the TransferData service. This increases the programming security. The number of bytes used for this size is defined by the high nibble (bit 7 - 4) of the addressAndLengthFormatIdentifier. If data compression is used, it is vehicle manufacturer specific whether or not the memory size represents the compressed or uncompressed size.

RequestDownload - $34

Table 395 — Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	RequestDownload Response SID	M	0x74	RDPR
#2	lengthFormatIdentifier	M	0x00 – 0xF0	LFID
#3 : #n	maxNumberOfBlockLength = [byte#1 (MSB) : byte#m]	 M : M	 0x00 – 0xFF : 0x00 – 0xFF	MNROB_ B1 : Bm

Table 397 — Supported negative response codes

NRC	Description	Mnemonic
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be returned if a server receives a request for this service while in the process of receiving a download of a software or calibration module. This could occur if there is a data size mismatch between the server and the client during the download of a module.	CNC
0x31	**requestOutOfRange** This NRC shall be returned if: — the specified dataFormatIdentifier is not valid. — the specified addressAndLengthFormatIdentifier is not valid. — the specified memoryAddress/memorySize is not valid.	ROOR
0x33	**securityAccessDenied** This NRC shall be returned if the server is secure (for server's that support the SecurityAccess service) when a request for this service has been received.	SAD
0x70	**uploadDownloadNotAccepted** This NRC indicates that an attempt to download to a server's memory cannot be accomplished due to some fault conditions.	UDNA

TransferData - $36

This service does not use a sub-function parameter.

Table 403 — Request message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	TransferData Request SID	M	0x36	TD
#2	blockSequenceCounter	M	0x00 – 0xFF	BSC
#3 : #n	transferRequestParameterRecord[] = [transferRequestParameter#1 : transferRequestParameter#m]	 C : U	 0x00 – 0xFF : 0x00 – 0xFF	TRPR_ TRTP_ : TRTP_
C = Conditional: this parameter is mandatory if a download is in progress.				

TransferData - $36

Table 404 — Request message data-parameter definition

Definition
blockSequenceCounter The blockSequenceCounter parameter value starts at 0x01 with the first TransferData request that follows the RequestDownload (0x34) or RequestUpload (0x35) service. Its value is incremented by 1 for each subsequent TransferData request. At the value of 0xFF the blockSequenceCounter rolls over and starts at 0x00 with the next TransferData request message. Example use cases: — If a TransferData request to download data is correctly received and processed in the server but the positive response message does not reach the client then the client would determine an application layer timeout and would repeat the same request (including the same blockSequenceCounter). The server would receive the repeated TransferData request and could determine based on the included blockSequenceCounter that this TransferData request is repeated. The server would send the positive response message immediately without writing the data once again into its memory. — If the TransferData request to download data is not received correctly in the server then the server would not send a positive response message. The client would determine an application layer timeout and would repeat the same request (including the same blockSequenceCounter). The server would receive the repeated TransferData request and could determine based on the included blockSequenceCounter that this is a new TransferData. The server would process the service and would send the positive response message. — If a TransferData request to upload data is correctly received and processed in the server but the positive response message does not reach the client then the client would determine an application layer timeout and would repeat the same request (including the same blockSequenceCounter). The server would receive the repeated TransferData request and could determine based on the included blockSequenceCounter that this TransferData request is repeated. The server would send the positive response message immediately accessing the previously provided data once again in its memory. — If the TransferData request to upload data is not received correctly in the server then the server would not send a positive response message. The client would determine an application layer timeout and would repeat the same request (including the same blockSequenceCounter). The server would receive the repeated TransferData request and could determine based on the included blockSequenceCounter that this is a new TransferData. The server would process the service and would send the positive response message.
transferRequestParameterRecord This parameter record contains parameter(s) which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific. EXAMPLE For a download, the transferRequestParameterRecord include the data to be transferred.

TransferData - $36

Table 405 — Positive response message definition

A_Data byte	Parameter Name	Cvt
#1	TransferData Response SID	M
#2	blockSequenceCounter	M
#3 : #n	transferResponseParameterRecord[] = [transferResponseParameter#1 : transferResponseParameter#m]	C : U
C = Conditional: this parameter is mandatory if an upload is in progress.		

Table 407 — Supported negative response codes

NRC	Description	Mnemonic
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.(e.g., message length does not meet requirements of maxNumberOfBlockLength parameter returned in the positive response to the requestDownload service).	IMLOIF
0x24	**requestSequenceError** The server shall use this response code: — If the RequestDownload or RequestUpload service is not active when a request for this service is received; — If the RequestDownload or RequestUpload service is active, but the server has already received all data as determined by the memorySize parameter in the active RequestDownlod or RequestUpload service; NOTE The repetition of a TransferData request message with a blockSequenceCounter equal to the one included in the previous TransferData request message shall be accepted by the server.	RSE
0x31	**requestOutOfRange** This NRC shall be returned if: — The transferRequestParameterRecord contains additional control parameters (e.g. additional address information) and this control information is invalid. — The transferRequestParameterRecord is not consistent with the requestDownload or requestUpload service parameter maxNumberOfBlockLength. — The transferRequestParameterRecord is not consistent with the server's memory alignment constraints.	ROOR
0x71	**transferDataSuspended** This NRC shall be returned if the download module length does not meet the requirements of the memorySize parameter sent in the request message of the requestDownload service.	TDS
0x72	**generalProgrammingFailure** This NRC shall be returned if the server detects an error when erasing or programming a memory location in the permanent memory device (e.g. Flash Memory) during the download of data.	GPF
0x73	**wrongBlockSequenceCounter** This NRC shall be returned if the server detects an error in the sequence of the blockSequenceCounter. NOTE The repetition of a TransferData request message with a blockSequenceCounter equal to the one included in the previous TransferData request message shall be accepted by the server.	WBSC
0x92 / 0x93	**voltageTooHigh / voltageTooLow** This return code shall be sent as applicable if the voltage measured at the primary power pin of the server is out of the acceptable range for downloading data into the server's permanent memory (e.g. Flash Memory).	VTH / VTL

RequestTransferExit - $37

This service does not use a sub-function parameter.

Table 408 — Request message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	RequestTransferExit Request SID	M	0x37	RTE
#2 : #n	transferRequestParameterRecord[] = [transferRequestParameter#1 : transferRequestParameter#m]	U : U	0x00 – 0xFF : 0x00 – 0xFF	TRPR_ TRTP_ : TRTP_

Table 409 — Request message data-parameter definition

Definition
transferRequestParameterRecord This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific.

Table 410 — Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	RequestTransferExit Response SID	M	0x77	RTEPR
#2 : #n	transferResponseParameterRecord[] = [transferResponseParameter#1 : transferResponseParameter#m]	U : U	0x00 – 0xFF : 0x00 – 0xFF	TREPR_ TREP_ : TREP

RequestTransferExit - $37

Table 412 — Supported negative response codes

NRC	Description	Mnemonic
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be returned if the length of the message is wrong.	IMLOIF
0x24	**requestSequenceError** This NRC shall be returned if: — The programming process is not completed when a request for this service is received; — The RequestDownload or RequestUpload service is not active;	RSE
0x31	**requestOutOfRange** This NRC shall be returned if the transferRequestParameterRecord contains invalid data.	ROOR
0x72	**generalProgrammingFailure** This NRC shall be returned if the server detects an error when finalizing the data transfer between the client and server (e.g., via an integrity check).	GPF

RequestDownload - message flow

Table 415 — RequestDownload request message flow example

Message direction	client → server		
Message Type	**Request**		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte Value**	**Mnemonic**
#1	RequestDownload Request SID	0x34	RD
#2	dataFormatIdentifier	0x11	DFI
#3	addressAndLengthFormatIdentifier	0x33	ALFID
#4	memoryAddress [byte#1] (MSB)	0x60	MA_B1
#5	memoryAddress [byte#2]	0x20	MA_B2
#6	memoryAddress [byte#3] (LSB)	0x00	MA_B3
#7	MemorySize [byte#1] (MSB)	0x00	UCMS_B1
#8	MemorySize [byte#2]	0xFF	UCMS_B2
#9	MemorySize [byte#3] (LSB)	0xFF	UCMS_B3

Table 416 — RequestDownload positive response message flow example

Message direction	server → client		
Message Type	**Response**		
A_Data byte	**Description (all values are in hexadecimal)**	**Byte Value**	**Mnemonic**
#1	RequestDownload Response SID	0x74	RDPR
#2	LengthFormatIdentifier	0x20	LFID
#3	maxNumberOfBlockLength [byte#1] (MSB)	0x00	MNROB_B1
#4	maxNumberOfBlockLength [byte#2] (LSB)	0x81	MNROB_B1

TransferData- message flow

Table 417 — TransferData request message flow example

Message direction	client → server		
Message Type	Request		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	TransferData Request SID	0x36	TD
#2	blockSequenceCounter	0x01	BSC
#3	transferRequestParameterRecord [transferRequestParameter#1] = dataByte#3	0xXX	TRTP_1
:	:	:	:
#129	transferRequestParameterRecord [transferRequestParameter#127] = dataByte#129	0xXX	TRTP_127

Table 418 — TransferData positive response message flow example

Message direction	server → client		
Message Type	Response		
A_Data byte	Description (all values are in hexadecimal)	Byte Value	Mnemonic
#1	TransferData Response SID	0x76	TDPR
#2	blockSequenceCounter	0x01	BSC

RequestUpload - $35

This service does not use a sub-function parameter.

Table 398 — Request message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	RequestUpload Request SID	M	0x35	RU
#2	dataFormatIdentifier	M	0x00 – 0xFF	DFI_
#3	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#4 : #(m-1)+4	memoryAddress[] = [byte#1 (MSB) : byte#m]	M : C_1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#n-(k-1) : #n	memorySize[] = [byte#1 (MSB) : byte#k]	M : C_2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
C_1: The presence of this parameter depends on address length information parameter of the addressAndLengthFormatIdentifier C_2: The presence of this parameter depends on the memory size length information of the addressAndLengthFormatIdentifier.				

RequestUpload - $35

Table 399 — Request message data-parameter definition

Definition
dataFormatIdentifier This data-parameter is a one byte value with each nibble encoded separately. The high nibble specifies the "compressionMethod", and the low nibble specifies the "encryptingMethod". The value 0x00 specifies that neither compressionMethod nor encryptingMethod is used. Values other than 0x00 are vehicle manufacturer specific.
addressAndLengthFormatIdentifier This parameter is a one byte value with each nibble encoded separately (see H.1 for example values): — bit 7 - 4: Length (number of bytes) of the memorySize parameter — bit 3 - 0: Length (number of bytes) of the memoryAddress parameter
memoryAddress The parameter memoryAddress is the starting address of server memory from which data is to be retrieved. The number of bytes used for this address is defined by the low nibble (bit 3 - 0) of the addressAndLengthFormatIdentifier. Byte#m in the memoryAddress parameter is always the least significant byte of the address being referenced in the server. The most significant byte(s) of the address can be used as a memory identifier. An example of the use of a memory identifier would be a dual processor server with 16 bit addressing and memory address overlap (when a given address is valid for either processor but yields a different physical memory device or internal and external flash is used). In this case, an otherwise unused byte within the memoryAddress parameter can be specified as a memory identifier used to select the desired memory device. Usage of this functionality shall be as defined by vehicle manufacturer / system supplier.
memorySize This parameter shall be used by the server to compare the memory size with the total amount of data transferred during the TransferData service. This increases the programming security. The number of bytes used for this size is defined by the high nibble (bit 4) of the addressAndLengthFormatIdentifier. If data compression is used, it is vehicle manufacturer specific whether or not the memory size represents the compressed or uncompressed size.

RequestUpload - $35

Table 400 — Positive response message definition

A_Data byte	Parameter Name	Cvt	Byte Value	Mnemonic
#1	RequestUpload Response SID	M	0x75	RUPR
#2	lengthFormatIdentifier	M	0x00 – 0xF0	LFID
#3 : #n	maxNumberOfBlockLength = [byte#1 (MSB) : byte#m]	M : M	0x00 – 0xFF : 0x00 – 0xFF	MNROB_ B1 : Bm

Table 402 — Supported negative response codes

NRC	Description	Mnemonic
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.	IMLOIF
0x22	**conditionsNotCorrect** This NRC shall be returned if the criteria for the requestUpload are not met. This could occur if a server receives a request for this service while a requestUpload is already active, but not yet completed.	CNC
0x31	**requestOutOfRange** This NRC shall be returned if: — The specified dataFormatIdentifier is not valid; — The specified addressAndLengthFormatIdentifier is not valid; — The specified memoryAddress/memorySize is not valid;	ROOR
0x33	**securityAccessDenied** This NRC shall be returned if the server is secure (for server's that support the SecurityAccess service) when a request for this service has been received.	SAD
0x70	**uploadDownloadNotAccepted** This NRC indicates that an attempt to upload to a server's memory cannot be accomplished due to some fault conditions.	UDNA

RequestUpload - $35

Table 398 — Request message

A_Data byte	Parameter Name
#1	RequestUpload Request SID
#2	dataFormatIdentifier
#3	addressAndLengthFormatIdentifier
#4 : #(m-1)+4	memoryAddress[] = [byte#1 (MSB) : byte#m]
#n-(k-1) : #n	memorySize[] = [byte#1 (MSB) : byte#k]

Table 399 — Request message data-parameter definition

Definition
dataFormatIdentifier This data-parameter is a one byte value with each nibble encoded separately. The high nibble specifies the "compressionMethod", and the low nibble specifies the "encryptingMethod". The value 0x00 specifies that neither compressionMethod nor encryptingMethod is used. Values other than 0x00 are vehicle manufacturer specific.
addressAndLengthFormatIdentifier This parameter is a one byte value with each nibble encoded separately (see H.1 for example values): — bit 7 - 4: Length (number of bytes) of the memorySize parameter — bit 3 - 0: Length (number of bytes) of the memoryAddress parameter
memoryAddress The parameter memoryAddress is the starting address of server memory from which data is to be retrieved. The number of bytes used for this address is defined by the low nibble (bit 3 - 0) of the addressAndLengthFormatIdentifier. Byte#m in the memoryAddress parameter is always the least significant byte of the address being referenced in the server. The most significant byte(s) of the address can be used as a memory identifier. An example of the use of a memory identifier would be a dual processor server with 16 bit addressing and memory address overlap (when a given address is valid for either processor but yields a different physical memory device or internal and external flash is used). In this case, an otherwise unused byte within the memoryAddress parameter can be specified as a memory identifier used to select the desired memory device. Usage of this functionality shall be as defined by vehicle manufacturer / system supplier.
memorySize This parameter shall be used by the server to compare the memory size with the total amount of data transferred during the TransferData service. This increases the programming security. The number of bytes used for this size is defined by the high nibble (bit 4) of the addressAndLengthFormatIdentifier. If data compression is used, it is vehicle manufacturer specific whether or not the memory size represents the compressed or uncompressed size.

RequestUpload - $35

Table 400 — Positive response message definition

A_Data byte	Parameter Name	Cvt
#1	RequestUpload Response SID	M
#2	lengthFormatIdentifier	M
#3 - #n	maxNumberOfBlockLength = [byte#1 (MSB) - byte#m]	M - M

Table 402 — Supported negative response codes

NRC	Description
0x13	**incorrectMessageLengthOrInvalidFormat** This NRC shall be sent if the length of the message is wrong.
0x22	**conditionsNotCorrect** This NRC shall be returned if the criteria for the requestUpload are not met. This could occur if a server receives a request for this service while a requestUpload is already active, but not yet completed.
0x31	**requestOutOfRange** This NRC shall be returned if: — The specified dataFormatIdentifier is not valid; — The specified addressAndLengthFormatIdentifier is not valid; — The specified memoryAddress/memorySize is not valid;
0x33	**securityAccessDenied** This NRC shall be returned if the server is secure (for server's that support the SecurityAccess service) when a request for this service has been received.
0x70	**uploadDownloadNotAccepted** This NRC indicates that an attempt to upload to a server's memory cannot be accomplished due to some fault conditions.

ECU Programming Flow