Unified Diagnostic Services (UDS) Protocol

July 2020

L&T Technology Services

LTTS GLOBAL ENGINEERING ACADEMY

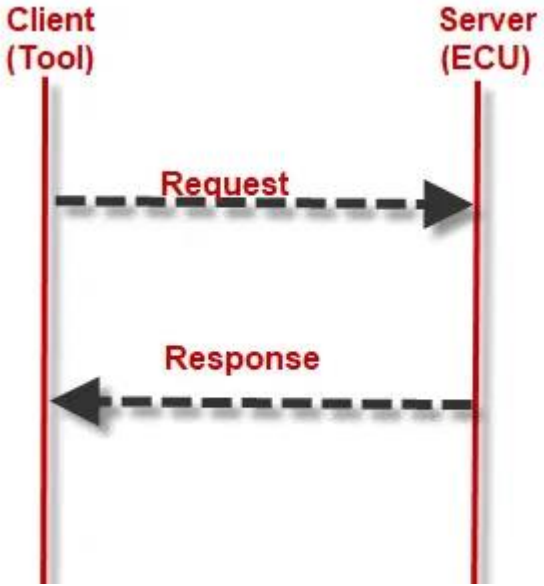
Unified Diagnostic Services

UDS Protocol

- UDS stands for **Unified Diagnostic Protocol**
- Diagnostic is basically a technique to identify any kind of illness and here this illness(fault) belongs to Vehicle.
- UDS is an standard by ISO which offers some unified(Uniform for All ECU Suppliers) services through which a UDS enabled diagnostic tool can perform off-board(When Vehicle in Garage) diagnosis for faulty ECU.
- **U**nified **D**iagnostic Protocol offers some ISO standard diagnostic services and its implementation specified into ISO-14229(1)

UDS Protocol

- ISO offers some services under UDS with fixed functionality.
- As per requirement UDS enabled external diagnostic tool sends request to the UDS enabled server ECU and server ECU respond back to the client with desired output.



UDS Protocol

- The Complete **U**nified **D**iagnostic **P**rotocol development was documented in three segments by ISO technical team-
  - Diagnostic requirement and specifications – **iso–14229(1)**
  - Session Layer Services – **iso– 14229(2)**
  - Diagnostic Implementation – **Depends on Data Link Protocol**
    - UDS On CAN – **iso-14229(3)**
    - UDS On Flex Ray – **iso-14229(4)**
    - UDS On Ethernet – **iso-14229(5)**
    - UDS On K-Line- **iso-14229(6)**
    - UDS On LIN – **iso-14229(7)**

    - **Note:** I will target UDS over CAN.

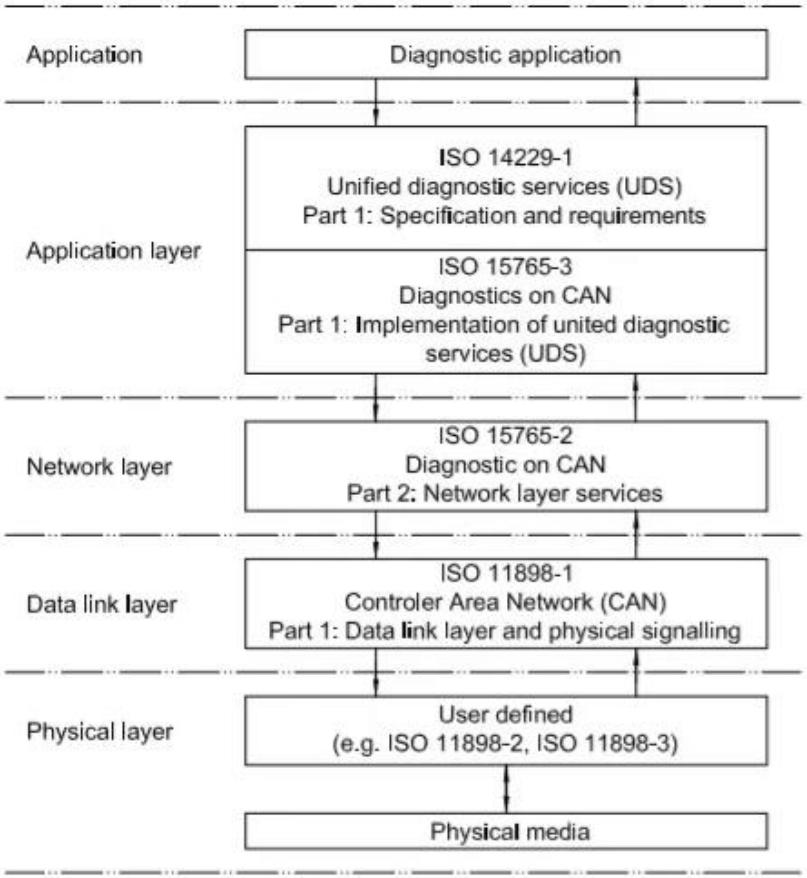
Need Of UDS

- In the Course of time Vehicle's features is continuously growing to attract and provide more and more safe luxury to its customers which causes many more ECUs to be installed in the vehicle system.
- Before UDS, there were many diagnostic protocols like KWP 2000, ISO 15765, and diagnostics over K-Line.
- OEMs and Suppliers have to face compatibility issue between these protocols
- To get rid of this compatibility issue OEM and Suppliers agreed to rely on a standard protocol which is named as **U**nified **D**iagnostic **P**rotocol.

Implementation of UDS on CAN in OSI model

- Although UDS specification works on application layer(14229-1) and session layer(14229-2) but ultimately Server(ECU) and Diagnostic Tool(Client) are connected over CAN physical medium.
- UDS request and response should follow the Bosch CAN specification to send data over CAN Bus

Implementation of UDS on CAN in OSI model



UDS Message Structure

- There are basically three types of frames in UDS protocol.
  - Request Frame with Sub-Function ID
  - Request Frame without Sub-function ID
  - Positive Response Frame
  - Negative Response Frame
- **Service id –** It is basically 1 byte ID belongs to the service defined in 14229-1. Server see this Identifier and perform that particular function related to this service.
- **Sub-function –** It is an optional field and sub-part of service ID.

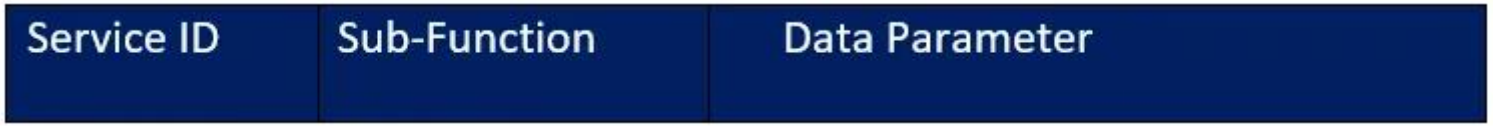
UDS Message Structure

Example:

- Function : session control
- Service ID : 0x10
- Sub-Function – default session, programming session ,extended session
- so under the function session control(0x10) we have 3 sub-functions.

Request Message With Sub-Function

- Request frame is used to send any request to server from tool. It has generally following mandatory bytes-



UDS Message Structure

Request Message Without Sub-Function

Positive Response Message Structure

- when tool sends some request then the response is also expected but it can be suppressed if response not required. let's see how?

UDS Message Structure

SUB-Function Byte role in response

- The server decides about the response through the Bit-7 of subfunction byte of request frame. You need to do bit-7(Sub-Function Byte) modification in the request frame.
- If bit-7(D1)=0——————> Response required
- If bit-7(D1)=1——————> Response not required

Negative response Frame Format

0x7F	Service ID	Data Parameter

UDS Application Layer

- **U**nified **D**iagnostic **P**rotocol application layer implementation is categories in two segment by ISO
- UDS specification and requirement(14229(1)) **– Listed UDS services**
- UDS implementation over CAN(14229(3)) –**Implementation of UDS by 14229(1) on CAN**
- There are many services and each service may have many sub-function to perform various kind of functions in ECUs.
  - Diagnostic and communication management.
  - Data Transmission
  - Stored Data Transmission
  - Input/output Control
  - Remote activation of routine
  - Upload/Download

Diagnostic and communication management.

- Diagnostic and Communication Management function group of UDS services contains following services-

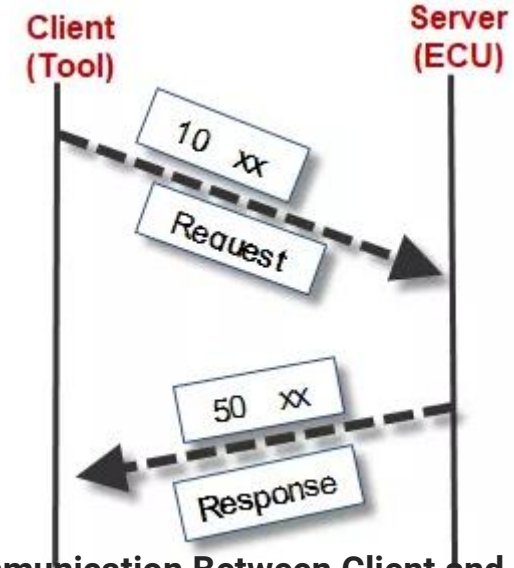
Diagnostic Services	Service Id	Sub-Function
Diagnostic Session Control	0x10	Default Session(0xXX) Programming Session(0xYY) Extended Session()xZZ)
ECU Reset	0x11	Hard Reset(0xXX) Soft Reset(0xYY) Key-Off Reset(0xZZ)
Security Access	0x27	Seed Request(0xXX) Key Response(0xYY)
Communication Control	0x28	Enable Tx and Rx for ECU Disable Tx and Rx for ECU Enable Tx and Disable Rx for ECU Disable Tx and Enable Rx for ECU
Tester Present	0x3E	No subfunction
Access Timing Parameter	0x83	Yes(Vehicle Manufacturer Specific)
Secure Data Transmission	0x84	Yes(Vehicle Manufacturer Specific)
Control DTC setting	0x85	Enable DTC Detection(0xXX) Disable DTC Detection(0xYY)
Response To Event	0x86	Yes(Vehicle Manufacturer Specific)
Link Control	0x87	Yes(Vehicle Manufacturer Specific)

Note: Sub-function IDs are usually decided by a Vehicle manufacturer. Here I have mentioned some generic sub-functions.

Diagnostic Session Control

- There are many services but not all are accessible until ECU is not running in its compatible session like some services are accessible in default session and some required ECU extended session and some for programming. It is decided by the vehicle manufacturer.
- On ECU start, ECU runs in default session and that time Client can access only that services of the server those are allowed by the Manufacturer UDS specification.
- So prior to doing any diagnostic request, Client must be sure that this service is accessible or not in ECU current session. if not, then first send a request for session change after that desirable request shall be performed.

Diagnostic Session Control- 0x10



UDS Communication Between Client and Server

- **Note:**
- Application Specific Data Bytes are reserved values that can be defined for vehicle manufacturers and suppliers use.
- UDS frame data length has no limit but we know data length allowed at the CAN BUS is up to 8 byte. So data length greater then the allowed frame length must be segmented into multi-Frame as per CAN transport layer protocol(15765-2)before sending it at CAN-Bus.

ECU Reset - 0x11

- This service request is done by the client to restart the Server(Electronic Control Unit). Mainly three types of reset can perform by the Client over the server
  - Hard reset
  - Soft Reset
  - Key Off on Reset
- **Hard Reset:**
  - Server Power shut down(battery removed) followed by the power-up sequence is known as “Hard Reset”.
  - There is no standard action performed during hard-reset command it is completely application Specific.
  - A hard reset might end with the re-initialization of volatile and non-volatile memory.

ECU Reset - 0x11

Key Off-On Reset:

- As the name suggests, this command is like off the ignition and then again on.
- Like other resets there is nothing to do standard for this value, all is implementation specific.
- Volatile memory re-initialized but Non-volatile data will remain protected.

Soft Reset:

- A server will not completely shut down when soft reset will receive. It will just restart the application.
- Like the Hard reset, No standard action performed in Soft reset. It is implementation specific.
- Soft reset might end with the volatile data clear but Non -Volatile data will be preserved.

Security Access- 0x27

- To protect the unauthorized access of the server's secured area, the client must have to go through some kind of security.
- This UDS service helps the client to access the server's secured area if it is authorized. Although there are many methods to perform security.
- **UDS FRAME for Security Verification**

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Seed –Request (Tool→ECU)	27	xx(Seed_Sunfunc)	Application specific Data	
Seed –Response (Tool←ECU)	67	xx(Seed_Subfunc)	Seed_Value[n]	
Key-Response (Tool→ECU)	27	zz (key_Subfunc)	Key_Value[n]	
Response (If Key Verified) (Tool←ECU)	67	zz (key_Subfunc)	Application specific Data	

*Method to find seed-key value is specific to the vehicle manufacturer

Communication Control - 0x28

- This service is used to control the communication of server **i.e** enable or disable transmission and reception of messages from the server. With this service, we can perform the following sub-functions(xx) –
  - Enable Tx and Enable Rx
  - Enable Tx and Disable Rx
  - Disable Tx and Enable Rx
  - Disable Tx and Disable Rx

Tester Present Service - 0x3E

- This service is sent by the Client to keep the server alive in current diagnostics session otherwise communication link will disconnect after some time between Server and Client.

DATA TRANSMISSION

This functional unit has the following services used for data transmission between server and client –

Read data By Identifier	0x22
Read Memory By Address	0x23
ReadScalingDataByIdentifier	0x24
ReadDataByPeriodicIdentifier	0x2A
DynamicallyDefineDataIdentifier	0x2C
WriteDataByIdentifier	0x2E
WriteDataByMemoryAddress	0x3D

Read Data By Identifier – 0x22

- **Read By** – Client
- **Read From** – server's memory
- **Read Through** – Data identifier(DID)
  - i.e. Data record read by the client from server using data identifier(DID) through this service id.
- **Request Format for Single DID data record:**
- **Assume:**
- Data Identifier = 0xAA55

0x22 (SID)	0xAA (DID_MSB)	0x55 (DID_LSB)	0x00	0x00	0x00	0x00	0x00

- **RESPONSE Format for Single DID data record:**

0x62 (Response SID)	0xAA (DID1_MSB)	0x55 (DID1_LSB)	D1 (Data Record 1)	----------D(n-1)	Dn

Read Data By Identifier – 0x22

- **REQUEST Format for Multiple DID data record:**
- Assume:
- DID1= 0xAA55
- DID2= 0xF001

0x22 (Response SID)	0xAA (DID1_MSB)	0x55 (DID1_LSB)	0xF0 (DID2_MSB)	0x01 (DID2_LSB)	0x00	0x00	0x00

- **RESPONSE Format for Multiple DID data record:**

0x62 (Response SID)	0xAA (DID1_MSB)	0x55 (DID1_LSB)	D1 (Data Record 1)	----------D(n-1)	Dn

0xF0 (DID2_MSB)	0x01 (DID2_LSB)	D1 (Data Record 1)	D2 (Data Record 2)	----------D(n-1)	Dn

Read Memory By Address - 0x23

- "Read memory by address" i.e. client will access the server's data through the address location.
- **Read By** – Client
- **Read From** – server
- **Read Through** – Server's memory address
- **Input** – Memory address, byteCount to be read
- **Output** – **D**ata Bytes of size (memory Address+byteCount)

Read Memory By Address - 0x23

Request Format (16-bit server platform):

Data Bytes	Parameter	Value
1d	Service Id	**0x23**
2d	Address and length Identifier (SUB-FUNCTION) **Bit3...0** → Tells about the length of memory address **Bit 7...4**(n)→Tells about the maximum data length record(2^n).	0x12 (for example) 0x01= 1 byte data size i.e. total data count will be 2^8=256 0x02= 2 byte address (assume 16 bit memory address)

Response Format:

Data Bytes	Parameter	Value
1d	Service Id	**0x63**
2d . . . : 2d+Datasize(0xFA)	Data record	Data record1(MSB) . . . : Data record250(LSB)

Write Data By Identifier

- This Service is used by a client to write the data on the server's memory.
- The client has to send the data record along with the Data Identifier to the server.
- If the DID is successfully validated by the server then the data record is to write under that DID at server.if not then respective NRC will be responded by the server
  - **Request format for sending a request to a server is:**

Data Bytes	Parameter	Value
1d	Service Id	0x2E
2d 3d	Data Identifier	0x00 – 0xFF(MSB) 0x00 – 0xFF(LSB)
4d . . .(md+3)	Data Record	m bytes

Write Data By Identifier

- If Success then Response will be:

Data Bytes	Parameter	Value
1d	Service Id	**0x6E**
2d 3d	Data Identifier	0x00 – 0xFF(MSB) 0x00 – 0xFF(LSB)

- In case of error, response frame format will be:

Data Bytes	Parameter	Value
1d	Negative Response Id	**0x7F**
2d	Service Id	**0x2E**
2d 3d	Data Identifier	0x00 – 0xFF(MSB) 0x00 – 0xFF(LSB)
4d	Negative Response Code(NRC)	Error Specific

For NRC follow the iso spec 14229(1).

Stored Data Transmission

- Functional Unit *Stored Data Transmission* of UDS protocol contains following services-

Service	Request ServiceID	Response ServiceID	Sub-Function
ClearDTC Service	0x14	0x54	No
ReadDTC Service	0x19	0x59	Yes

Clear Diagnostic Information

- This Diagnostic service used by the diagnostic tool to clear the DTC(Diagnostic Trouble Code) or Diagnostic Information from the server's memory.
- DTC(Diagnostic Trouble Code) is a fault code stored in memory for faulty control units. There are Number of DTCs for each control unit(Power Train, chassis, Break etc.)
- DTCs Can be clear in one shot or individual from server's memory for any control unit through the Request Frame.
- **Clear Diagnostic UDS Frame Format**

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Request Frame	14	DTC bytes or Application specific Data		
Positive Response Frame	54	Application specific Data		
Negative Response Frame	7F	14	yy	Application specific Data

Read DTC Information (SID=0x19)

- This service is used to read DTC from the server's memory. Sub-function defined by manufacturers decided the data about to read
- **Clear Diagnostic UDS Frame Format**

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Request Frame	19	Sub-Function ID	Application specific Data	
Positive Response Frame	59	Sub-Function ID	DTC Bytes	
Negative Response Frame	7F	19	yy	Application specific Data

DTC(Diagnostic Trouble Code)

- In the ’80s, It’s an impossible thing to get the information about malfunctioned unit if a problem is detected in your car and because of this, it was very tough to diagnose the malfunctioning unit in the car.
- As automobile engineering grows by time, they(ISO, Manufacturer) define the range of generic and manufacturer specific trouble codes for each automobile unit(Power Train, Chassis, body and network unit) and these trouble codes are known as *diagnostic trouble codes(DTC)*.
- Every Vehicle Manufacture provides an OBD Connector which is a digital communication medium through which anyone whether he is car owner or repair technician, can read the real-time data of his car and get the information(DTCs) about the malfunctioned unit and repair it accordingly.

Diagnostic Trouble Code Format

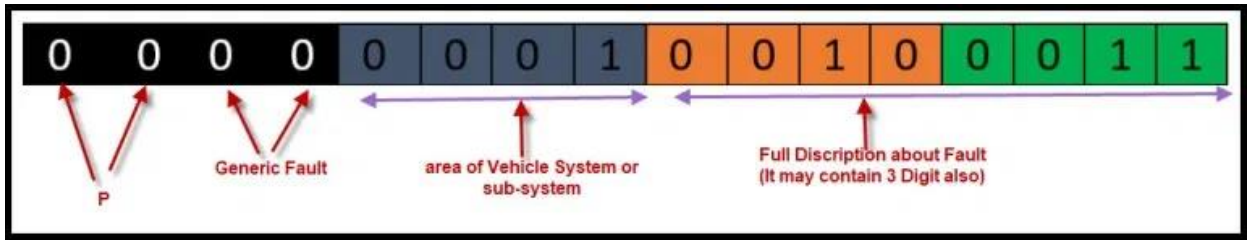
- DTC code may contain 2 or 3 bytes. Basically, DTC code has DTC high bytes and DTC middle bytes while the last byte is optional

DTC High Byte								DTC Middle Byte								DTC Low Byte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
00 : P 01 : C 10 : B 11 : U		1st Char. of Code		2nd Character of Code				3rd Character of Code				4th Character of Code				DTC Failure Category				DTC Failure Sub Type			

DTC Bytes	Bits	DTC code
High Byte	7-6	Family of DTC 00- P(Powertrain) 01- C(Chassis) 10 -B(Body) 11-U(Network)
	5-4	Code is generic/Manufacturer specific 00- Generic(ISO) 01- Manufacturer controlled
	3-0	Code for Vehicle SUB-System
Middle Byte	7-0	Detail Description about the Fault
Low Bytes (optional)	7-0	Detail Description about the Fault

Diagnostic Trouble Code Format

- **Example:**
- Let's investigate how will look A DTC code 0x0123 to a repair technician?



- So it will look **P0123** to Technician through which he can understand fault code is generic and related to powertrain family.

UDS-Input/Output Control

- UDS Input-output control unit has only one service id i.e 0x2F. This service ID is used by the tester to control the input signals /internal functionalities or output of UDS enabled Server ECU.

Service	Request ServiceID	Response ServiceID	Sub-Function
Input/Output Control Service	0x2F	0x6F	No (there are Application specific Data identifier to perform Specific task)

Input/Output Control Service Message

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Request Frame	2F	ECU Identifier + Input/output Control Parameter		
Positive Response Frame	6F	ECU Identifier + Input/output Control Parameter		
Negative Response Frame	7F	2F	yy	Application specific Data

- ECU Identifier: Is like a sub-function. As in the above frame format, ECU Identifier represents some specific set of input/output signals and those can be only accessible if tester sends that particular identifier. Its may have single or multi-byte value.

Input / Output Control Parameter(optional):

- A tester also can request to server through optional bytes –
- **Return Control To ECU** – Indicates To server that the client has no longer access and the mentioned signal gets its control.
- **Reset to default**– Indicates To server that set mentioned input and output signal, and internal parameters to its default value.
- **Freeze Current State** – Indicates To server that freeze mentioned input, output signal, and internal parameters to its current value
- **Short-Term Adjustment**– Indicates To server that set its input, output signal and internal parameters to the provided value.

UDS-Remote Activation Of Routine

This function unit has only one service Id 0x31 *"*Remote Activation Of Routine*".*

This service is used by the client to perform various project specific functions through OEM-specific routine

Identifier.

Service	Request ServiceID	Response ServiceID	Sub-Function
Routine Control Service	0x31	0x71	Yes

UDS-Remote Activation Of Routine

- There are Basically three SUB-function under this service –
- **Start routine –** Initiate any service and used to indicate start/completion of any service
- **Stop routine** – Through this sub-function Client can interrupt running service at any time inside server.
- **Requestroutineresult**– It is basically a request message by client to get the result of service from server.
- **So Basically this service is used for –**
- To notify the start/stop execution of any task.
- To erase memory or perform a checksum of integrity after download etc.
-

UDS-Remote Activation Of Routine

- Example:
- Suppose you want to erase data of flash memory prior to download.let as per OEM specification –
- Routine Identifier for Erase memory Function = 0xAABB then
- Request(Client—>Server) = 0x31 01 0xAA 0xBB ….. ……. ……
- if memory erase successfully then
- Pos. Response(client<——-Server)= 0x71 01 0xAA 0xBB … …. …. …
- Negative Response(client<——-Server)= 0x7F 0x31 01 0xAA 0xBB … …. …. …
- Like this Client can perform the various function inside UDS enabled server through different identifier and these data Identifier are depends on Manufacture diagnostic specification.

Upload/Download Functional Unit

- This functional unit has services through which a client can write or read new firmware from server's memory. See the table for related services

Service	Request ServiceID	Response ServiceID	Sub-Function
Request Download	0x34	0x74	No
Request Upload	0x35	0x75	No
Transfer Data	0x36	0x76	No
Transfer exit	0x37	0x77	No

Request Download

- Request download service is used to send the request for new software or other data downloads in ECU. The client may send the base address and firmware size with this request and server will acknowledge with allowed transfer packet size.
- **Request Download Message**

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Request Frame	34	Download Address(optional) + Download Size(optional)		
Positive Response Frame	74	Transfer Packet size + Application specific Data		
Negative Response Frame	7F	34	yy	Application specific Data

Request Upload

- Request upload service is the same as the Request download and it is used to send the request by the tester to read the downloaded firmware from ECU. The client must specify the address and firmware size with this request but this time the client must tell the server how long would be transferred packet size.
- Frame format would be same as request download service except for serviceID.

Transfer Data service

- Transfer Data service is used to transfer packets to server in allowed packet size(In request download). if the packet is transferred correctly then the server should send a positive acknowledgment, if not then a negative acknowledgment and this would continue until whole firmware size does not transfer correctly to the server.
- **Transfer Data Message**

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Request Frame	36	Transfer Packet size + Download Address+ Firmware Data		
Positive Response Frame	76	Application specific Data		
Negative Response Frame	7F	36	yy	Application specific Data

Transfer exit service

- At this service client request to the server to perform transferred software integrity or to send a negative acknowledgment to stop data transfer request.
- **Transfer Exit Message**

UDS Frame	D0	D1	D2	D3.....Dn(Optional)
Request Frame	37	Application specific Data		
Positive Response Frame	77	Application specific Data		
Negative Response Frame	7F	37	yy	Application specific Data

- so that's all a basic overview of this functional unit service ids. It will be more clear once you get a chance to implement these services as per OEM given specification because application data varies OEM to OEM and once you do practical then it would be more clear how things going on.

Thank You !

L&T Technology Services