Certificated access to the vehicle and its data
Report on ISO work (G. Feiter)

Gangolf Feiter - Concepts & Services Consulting
Ostpromenade 107 (City Center)
52525 Heinsberg, Germany
Ph.: +49 (2452) 96 484 0
info@csc-online.eu
www.concepts-services-consulting.de

Meeting @EGEA Office in Bruessels
Sept. 05, 2017

Current Aftermarket "Best Practices"

- Reverse Engineering of raw diagnostic messages using custom "spy" tooling connected to the e.g., CAN bus whilst VM tester talks to the vehicle ECUs
- Raw diagnostic messages will be analyzed to interprete:
  - Data Identifier
  - Number of Bytes
  - Data Type
  - Data conversion
  - Text String(s)/Unit is copied from VM tester Display
  - Etc.

Future Aftermarket "Best Practices"

- Order a certificate from VM for a vehicle type/model (all ECUs)
- Use VM tester with certificate to communicate to the vehicle ECUs
- Identify the vehicle's system content and functionality
- Reverse Engineering of raw diagnostic messages using custom "spy" tooling connected to the e.g., CAN bus whilst VM tester talks to the vehicle ECUs will ONLY BE POSSIBLE if messages are NOT ENCRYPTED
- Above solution does NOT work for vehicles with encrypted diagnostic messages, e.g. UDS service 0x84 SecuredDataTransmission

What are the Alternatives/Options for the Aftermarket for encrypted IVNs? – Alternative 1

- Buy certificate from VM
- Buy the enhanced diagnostic protocol and raw diagnostic data documentation from the VM
- Implement new authentication service and message encryption/decryption software
- Implement messages and data according to VM documentation
- Test against vehicle

What are the Alternatives/Options for the Aftermarket for encrypted IVNs? – Alternative 2

- Modify your VCI to become a SVI (Secured Vehicle Interface)
  - Vehicle-specific IVN configuration to convert raw data into standardized format (ISO 13185-2 UGP: VIDF = Vehicle Interface Data Format)
  - Implement ISO 13185-2 UGP (Unified Gateway Protocol) into your SVI and use any IP-connected test equipment incl. Smartphone, Tablet, … which communicates with your SVI via UGP
  - Your SVI requires a vehicle-specific VIDF configuration to convert the standardized data format (integer) into readable data parameters

Objective

Today's·aftermarket·diagnostics·business·models·can·no·longer·be·applied·in·an·efficient·effective·way.· Future· vehicle-to-infrastructure· (V2I)· and· infrastructure-to-vehicle· (I2V)· communication· will· be· implemented·according·to·ITS·standards.¶

New·business·demands·will·drive·requirements·to·¶

—→ Collect·vehicle·data·for·continuous·product·improvement,¶

—→ send·data·to·the·ExVe·(Extended·Vehicle)·OEM·cloud·server,·¶

—→ send·data·to·the·insurance·company's·cloud·server·to·calculate·driver·profiles,·¶

—→ send·data·to·the·roadside·assistance·server·to·support·the·customer's·broken·down·vehicle,·¶

—→ send·data·to·an·anonymous·location·server·to·support·ITS·traffic·management,¶

—→ provide·data·to·third-party·Apps·installed·by·customer·owned·mobile·device(s),¶

—→ other·use·cases.¶

"One fits all" technical solution

2.1 Business models and use cases

Ideally, one technical solution should cover all business models and use cases of current (used) and future vehicles which are compliant to security requirements inside and outside of the vehicle. The technical solution should also minimize the amount of changes (hardware and software) required to be designed into the future vehicles by the vehicle manufacturers.

2.2 Basic principles

One of the basic principles which such technical solution must fullfil is that all secured vehicles provide an abstract interface. The abstract interface must be defined that all involved communication devices (access to IVN data and ECUs, external applications, cloud servers, ...) shall speak the same language (protocol) and use the same syntax (data format) but different semantics (vocabulary).

2.3 Standardization

The protocol and data format are candidates for standardization. The vocabulary, which represents the IVN and diagnostic data of a specific vehicle brand and model cannot be standardized across multi-brand vehicles. The very best example are the SAE J1979DA (Digital Annex of emissions system-related OBD data) and SAE J2012DA (Digital Annex of diagnostic trouble codes), which are continuously under standardization caused by new vehicle technology since more than 20 years. The standardized SAE data are a very small subset of what is required to perform successful diagnostic and repair, to satisfy future ITS and future smart grid use cases.

2.4 Standardized data types

All data must be converted into a data format using standardized data types. One of the objectives is that no software compilation and software updates are required if new/additional data are to be supported during the life cycle of the vehicle.

2.5 Separation of IVN and application domain technology

Future business models should strictly follow the principles to be established for secured multi-brand vehicles with a separation of IVN and application domain technology. The vehicle manufacturer is responsible for designing secured and non vulnerable vehicle systems. Therefore, the vehicle manufacturer should not be obliged to provide any information about the raw data streams and diagnostic messages inside the vehicle's IVNs. Such communication messages should be encrypted in a future securely designed vehicle system. Such provision will reduce the vulnerability of the IVN systems.

2.6 Multi-brand vehicle life cycle use cases

In order to support necessary multi-brand vehicle life cycle use cases (diagnostic and repair, ITS, smart grid, I/M, ePTI, roadside assistance, ...) the vehicle manufacturer must publish a Digital Annex of all vehicle model systems in a standardized data type format. This technical approach can be understood in a way that all multi-brand vehicles are socalled "MVM (Mobile Virtual Machines)". External applications access MVM resources independently of their hardware and software implementation.

Security strategy on std. access to IVN data

2.7 Standardized data access to secured vehicle IVNs

Figure 1 shows the security strategy on standardized access to in-vehicle data.

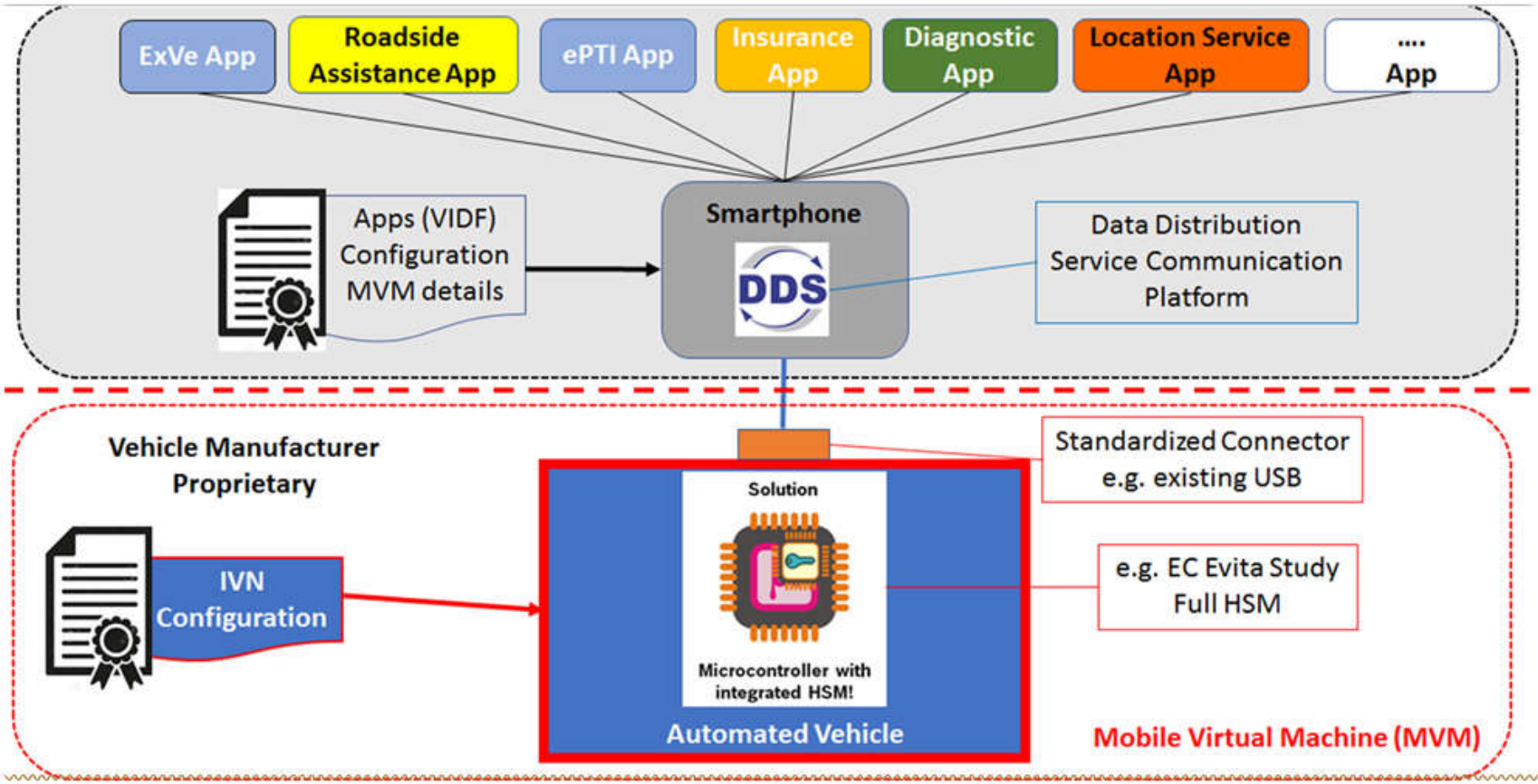


Figure 1 — Security strategy on standardized access to in-vehicle data

Description of Figure 1

The upper part of Figure 1 shows the customer's mobile device e.g. Smartphone with installed vehicle manufacturer and third party Apps. The mobile device uses a Data Distribution Service (DDS™) communication platform software installation. The DDS™ is a middleware software layer that abstracts the applications from the details of the operating system, network transport, and low-level data formats. The same concepts and APIs are provided in different programming languages allowing applications to exchange information across operating systems, languages, and processor architectures. Low-level details like data wire format, discovery, connections, reliability, protocols, QoS (Quality of Service) management, etc. are managed by the middleware.

The DDS™ communication platform supports the standardized Vehicle Interface Data Format (VIDF) which is used as a configuration defining all data parameters supported by a vehicle model including all option systems. The Apps installed on the mobile device access the DDS™ communication platform via standardized API functions to retrieve the subset of data parameters supported by the vehicle systems via the standardized e.g. USB port of the vehicle.

New vehicles w/ Apply Car Play or Android Auto

Figure 2 shows a vehicle with Apple CarPlay or Android Auto.

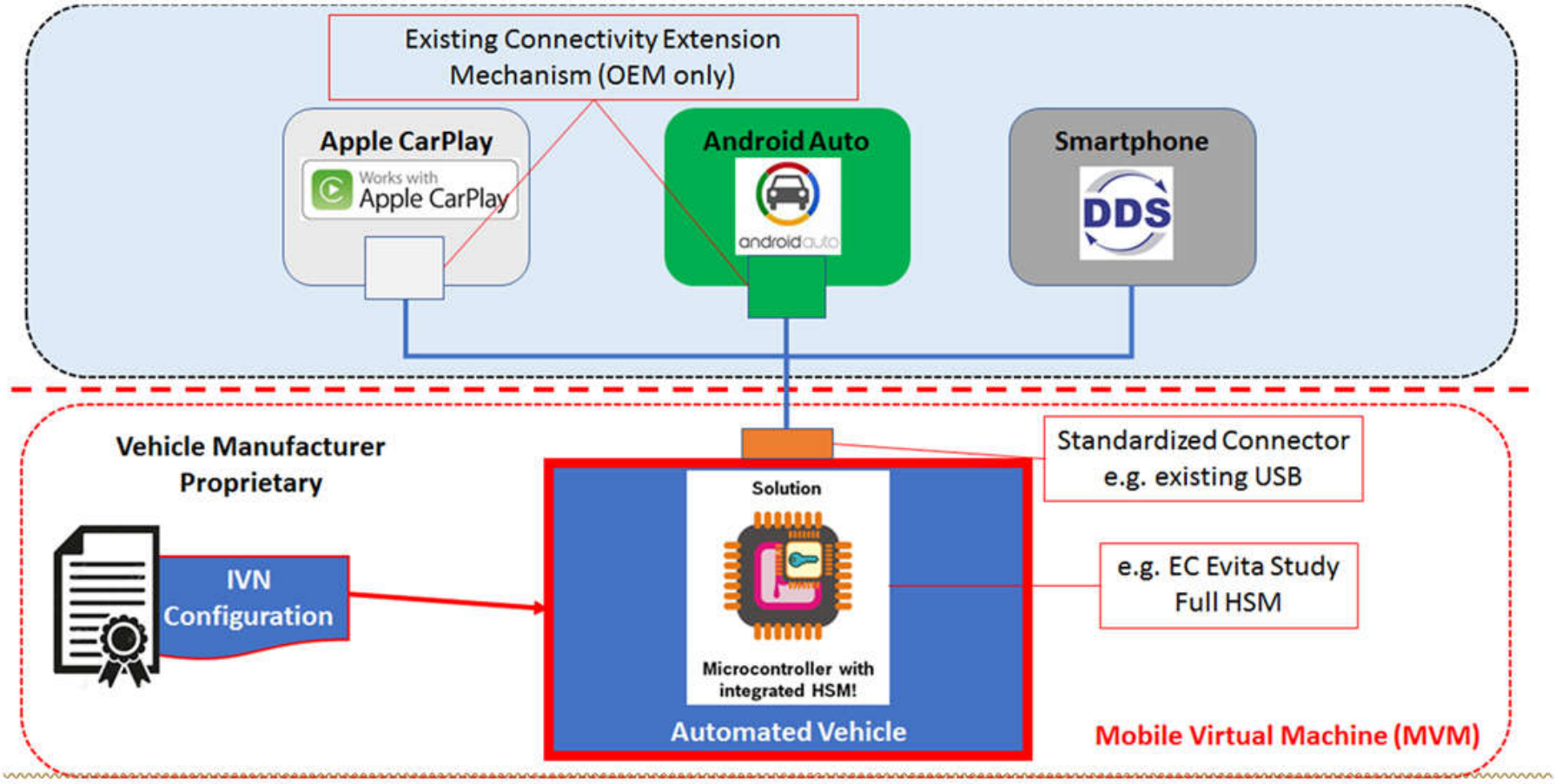


Figure 2 — Vehicles with Apple CarPlay or Android Auto

4 Five reasons why chosing a mobile device

4.1 Best price for connectivity

The mobile device e.g. smartphone provides the best price and value compared to computing power and flexible connectivity. It is a fast growing technology used by billions of customers. There is no other comparable communication platform available on the market. Wide Area Network (WAN) connectivity bandwidth is continuously improving from 3G, 4G to 5G. The amount of data to be communicated between a vehicle and the customer's subscribed services will continuously grow in the future. The amount of real-time data exchange an automated vehicle requires to communicate with the infra-structure with high priority will also grow with the level of automation the vehicle has been designed to.

4.2 Great computing platform

The mobile device e.g. smartphone is a great computing platform with lots of resources.

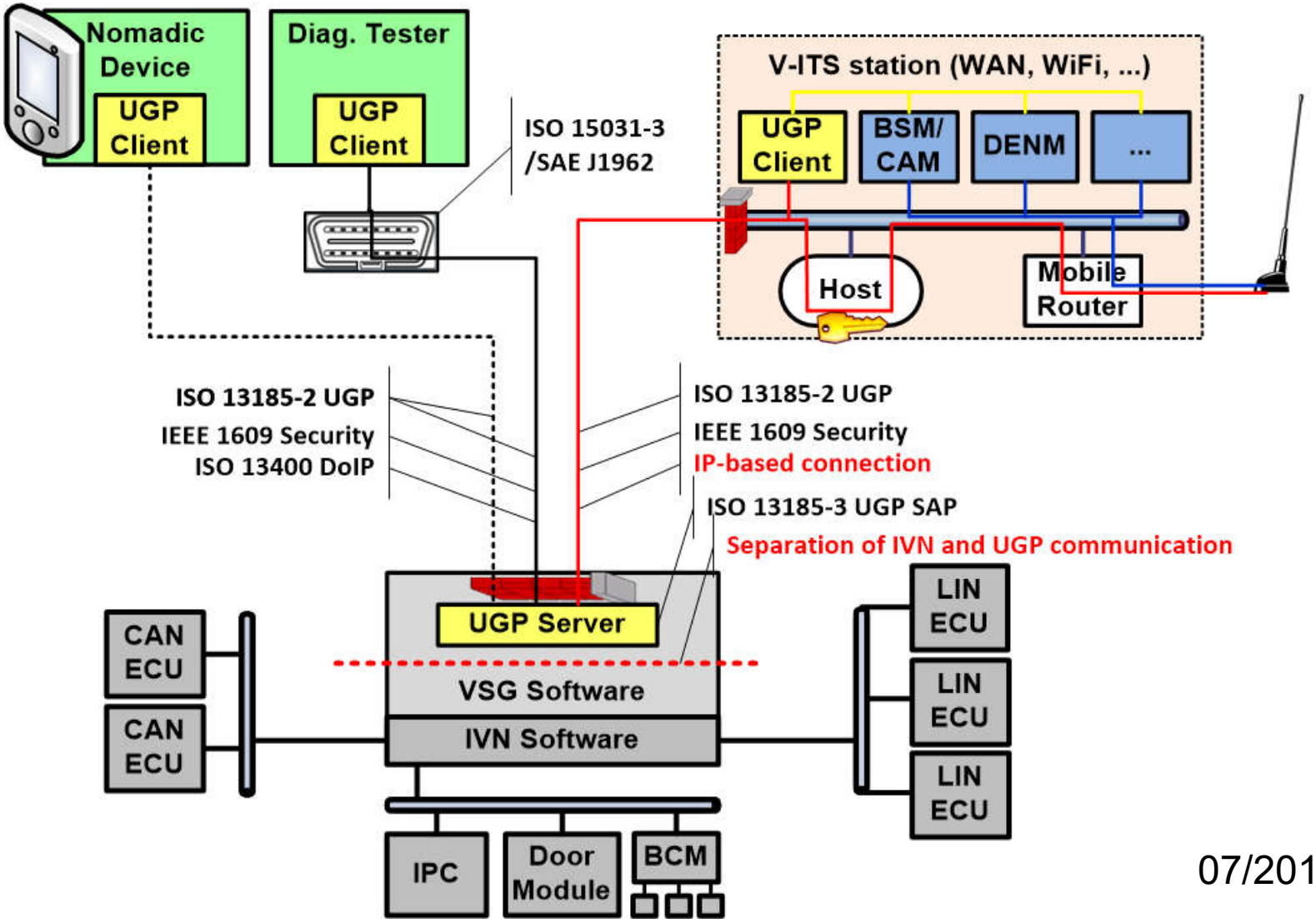
4.3 Liability

Another important reason is that the vehicle manufacturer does not need to design the hardware and software functionality required to support third party Apps into a vehicle designed Electronic Control Unit (ECU). The vehicle manufacturer is not liable for hardware, software and installed Apps on the mobile device.

4.4 Life cycle of vehicle vs. mobile devices

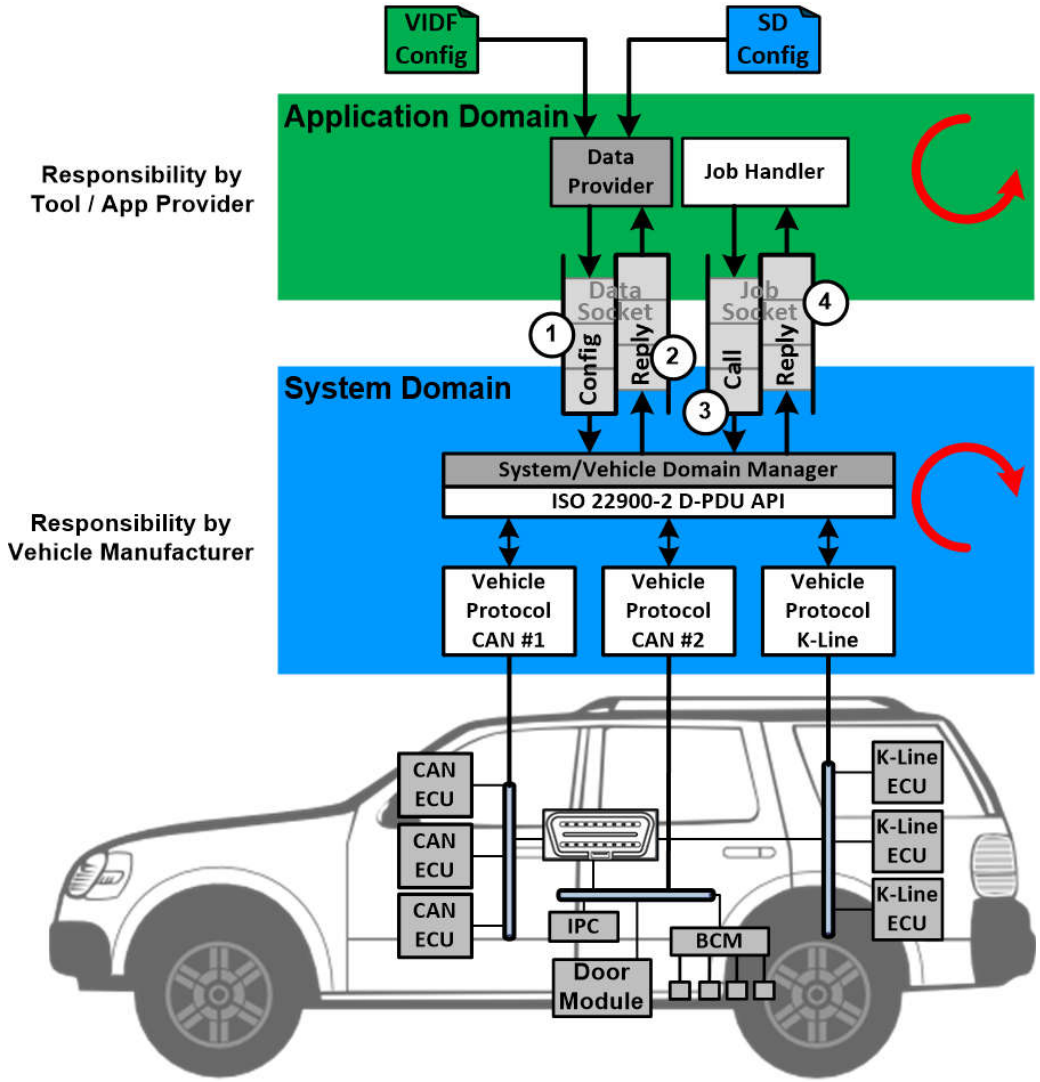
The life cycle of a vehicle is 5 to 10 times longer than any mobile device. many customers worldwide replace their smartphone after 2 to 3 years. The new smartphone has more computing power, resources and increased connectivity bandwidth.

VSG=SVI w/ Direct Access and V-ITS Station



07/2014

Application and System Domain



1	Data Socket - Send configuration (VIDF, SD)
2	Data Socket - Data provided (by configuration)
3	Job Socket - Call to get, set or control explicit data
4	Job Socket - Reply of the call

ISO SVI V-ITS Task Force

ISO TC-204 participants concerned with the state of TC22/TC204 collaborative development and the potential impact of ExVe adoption on the automotive aftermarket, formed an ad hoc workgroup to address the issue and overcome TC-22 WG6 objections.

- Scope & Objectives
- Define a single robust and fully secure technical solution using the ISO Secure Vehicle Interface (SVI) standards that's viable for both vehicle manufacturers and aftermarket solution providers.
- Educate aftermarket technologists on the benefits of SVI vs ExVe and provide an easy way for these engineers to incorporate SVI in their solutions.
- Inform legislators about our technical solution and educate them on the potential consequences resulting from a lack of a SVI and relying solely on manufacturers to protect the vehicle population.

A Single Standardized Solution Access Method

ISO 22900-2 defined in 2009 is the only international standard prescribing one set of access methods for data elements from an in-vehicle network. The absence of this led to product specific implementations which couldn't be shared across the automotive and ITS industry.

ISO 13185-2 UGP specifies the functional consolidation of the various vehicle interfaces into a "single solution" concept for an "SVI" which shall be connected to the ITS Vehicle Station (which consists of a Vehicle Mobile Host and a Vehicle Mobile Router).

Connection between Nomdic Device & SVI

Figure 4 illustrates the connection between the ND and V-ITS-SG. UGP provides application layer services to exchange the data between the Nomadic Device and the V-ITS-SG.

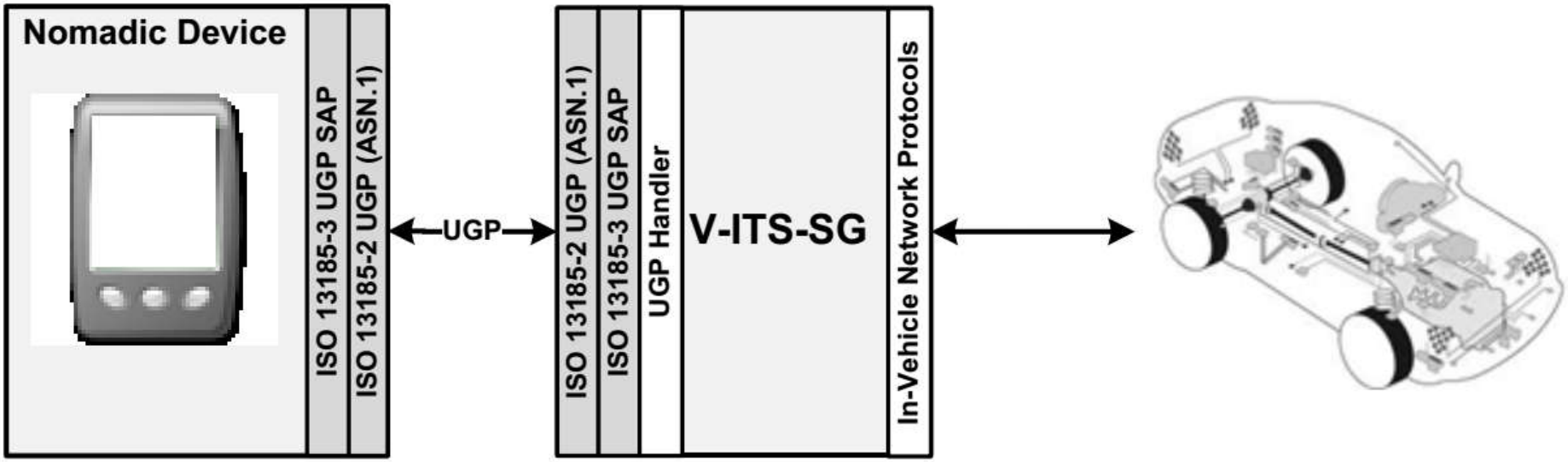
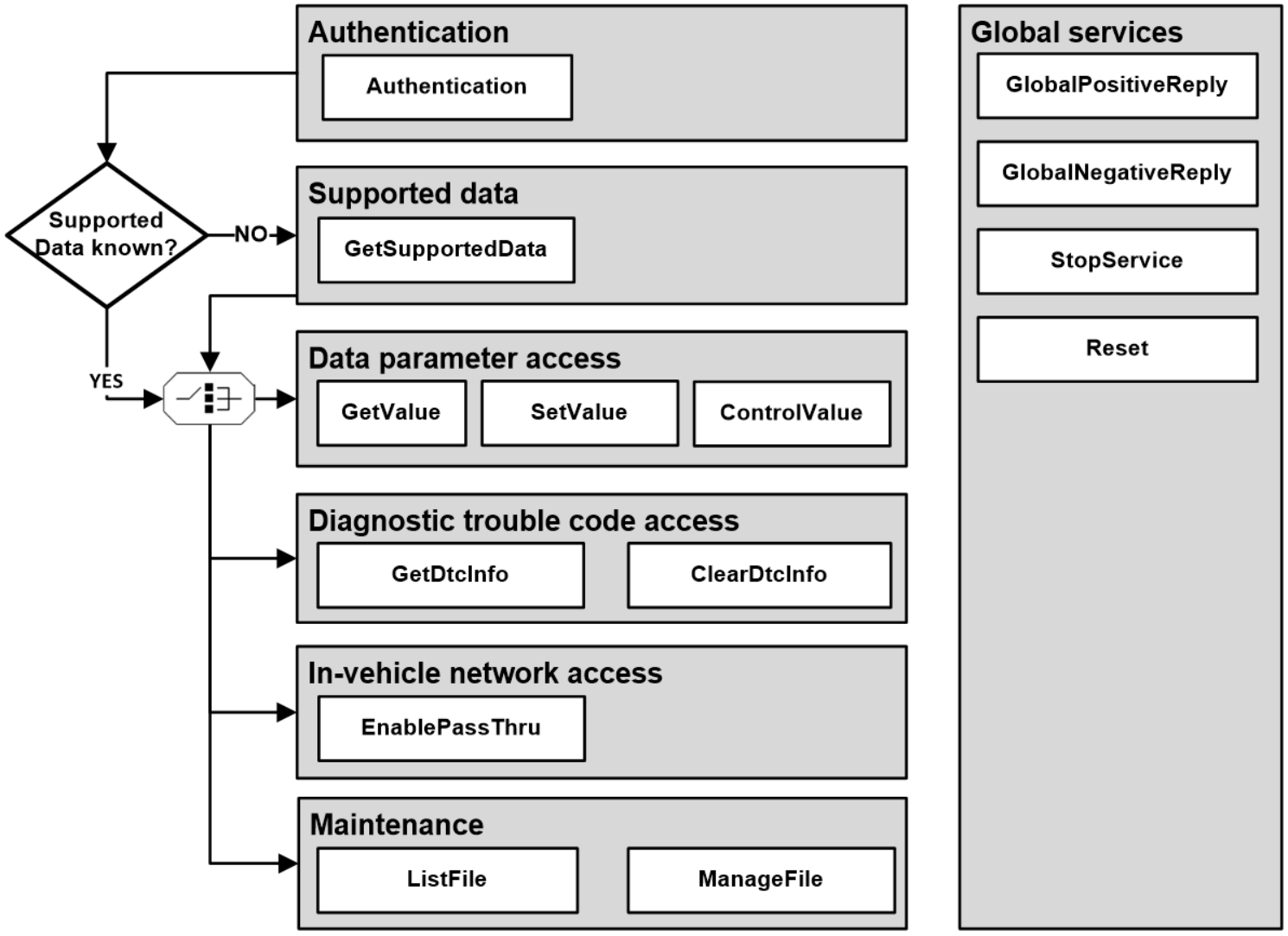


Figure 4 — Connection between ND and V-ITS-SG

ISO 13185-2 UGP Service Cluster and Services



UGP – Service Request and Indication primitives

6.4.2 Service request and service indication primitives

For each application layer service, service request and service indication primitives are specified according to the following general format:

```
service_name.request   (
                       callSequenceNumber,
                       timeInMillis,
                       length,
                       data[, parameter 1, ...],
                       )
```

The request primitive is used by the client function in the application, to initiate the service and pass data about the requested UGP service to the application layer.

```
service_name.indication   (
                       callSequenceNumber,
                       timeInMillis,
                       length,
                       data[, parameter 1, ...],
                       )
```

UGP - Messages

Table 1 — Definition of UGPMessage

Msg	**UGPMessage**		
Attributes	**Name**	**Description**	**Cvt**
	callSequenceNumber	This parameter includes the sequence number of the call. The receiver shall use the same number for the reply.	M
	timeInMillis	This parameter includes the time stamp of the reply data in milliseconds since 1970.	O

ASN.1

```
UGPMessage ::= SEQUENCE {
    callSequenceNumber  UNUM16,
    timeInMillis        UNUM64                  OPTIONAL,
    choiceUGP           CHOICE {
        authenticationCall          AuthenticationCall,
        authenticationReply         AuthenticationReply,
        getSupportedDataCall        GetSupportedDataCall,
        ...
        globalPositiveReply         GlobalPositiveReply,
        globalNegativeReply         GlobalNegativeReply,
        ...
    },
    ...
}
ugpVersion Version ::= 1
```

Table 15 — GetSupportedData example with ECU data

ASN.1

```
getSupportedDataCallWithEcuData UGPMessage ::= { callSequenceNumber 113,
    choiceUGP getSupportedDataCall: { supportedDataFilter with-ecu-data
} }
getSupportedDataReplyWithEcuData UGPMessage ::= { callSequenceNumber 113,
    choiceUGP getSupportedDataReply: { dataParamMapping {
        { rvId 1002, ecuId 17 }, { rvId 2341, ecuId 51 },
        { rvId 10020, ecuId 21 }, { rvId 10021, ecuId 21 },
        { rvId 10022, ecuId 21 }, { rvId 10023, ecuId 21 },
        { rvId 10024, ecuId 21 }, { rvId 10042, ecuId 51 },
        { rvId 10050, ecuId 51 }
} } }
```

U-PER

```
--getSupportedDataCallWithEcuData (4 byte):
00 1C 42 02

--getSupportedDataReplyWithEcuData (78 byte):
00 1C 43 21  28 00 00 3E  A8 00 00 01  14 00 00 49  2C 00 00 01  9A 00 00 9C
92 00 00 00  55 00 00 4E  4B 00 00 00  2A 80 00 27  26 80 00 00  15 40 00 13
93 C0 00 00  0A A0 00 09  CA 20 00 00  05 50 00 04  E7 50 00 00  06 68 00 02
74 28 00 00  03 30
```

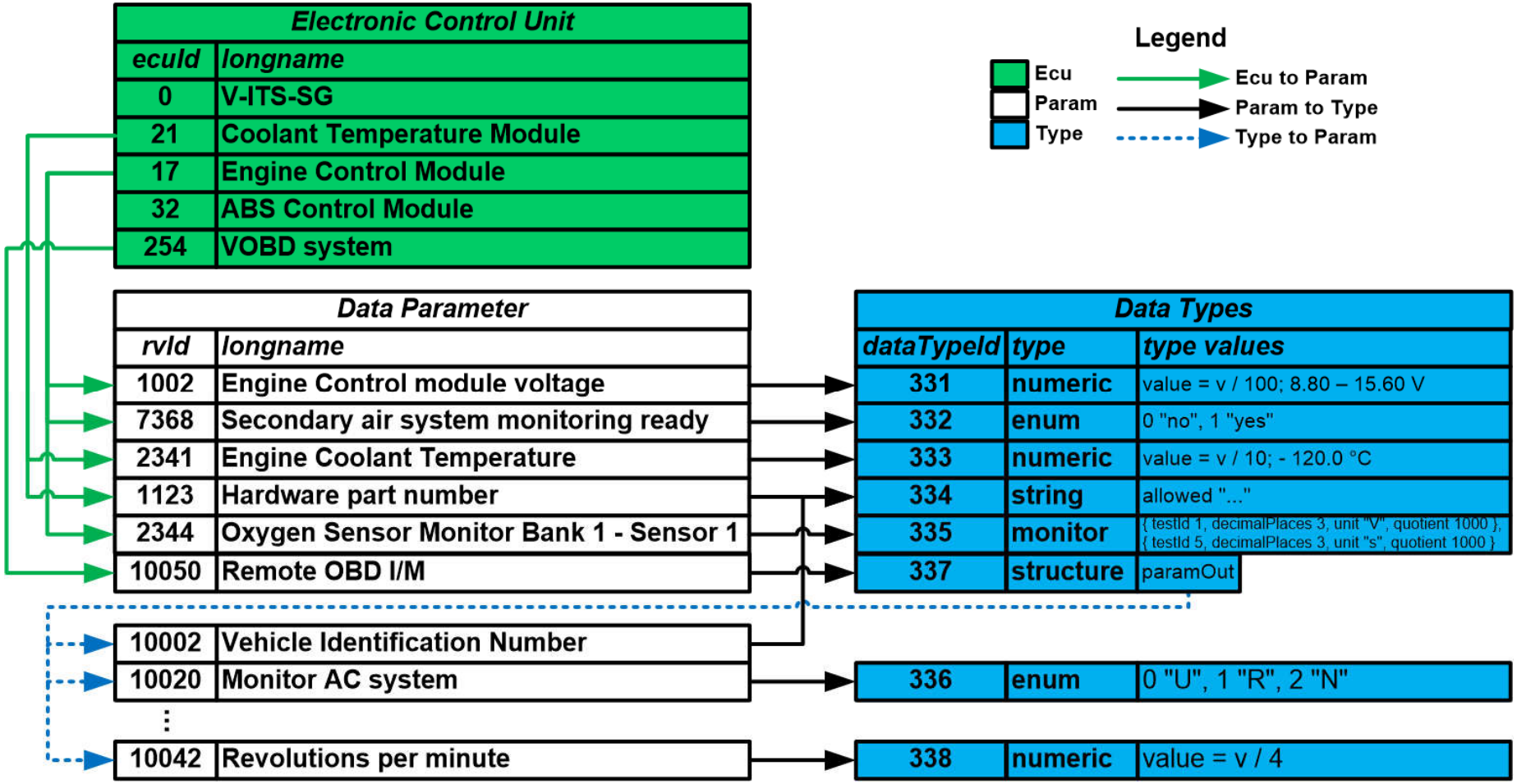
Table 18 — Definition of the message 'GetValueCall'

Msg	GetValueCall		Request data parameter values	
Attributes	**Name**		**Description**	**Cvt**
	test-Interval	condition	specifies the interpretation of the combined attributes `testInterval` and `condition`	O
	= 0	not set	reply message is sent only once	
		set	test the condition and if test condition is true then send one reply message	
	> 0	not set	reply message is sent each time the testInterval [ms] has expired	
		set	test the condition each time the testInterval [ms] has expired and if test condition is true then send a reply message	
	`dataParamList {}`		List of registered value identifiers to retrieve	O1[a]
	`dataParamMapping {}`		List of mappings between data parameter and ECU	O2[a]
	`condition`		A `ComplexCondition` element (see A.20). If the condition is set, the data parameter values should be responded only if the condition is true.	O
[a]:	Either O1 or O2 must be defined			

ASN.1

```
GetValueCall ::= SEQUENCE {
    testInterval        SNUM32,
    dataParamList       SEQUENCE OF Identifier                  OPTIONAL,
    dataParamMapping    SEQUENCE OF DataParamMapping            OPTIONAL,
    condition           ComplexCondition                        OPTIONAL,
    ...
}
```

VIDF Configuration Example



VIDF config 1

```
vidfConfigExample VIDFConfig ::= { configName "VIDFConfig example",
    unitType {
        { unitTypeId 7, name { textId 9007, longname "temperature" } },
        { unitTypeId 10, name { textId 9010, longname "electrical potential" } },
        { unitTypeId 21, name { textId 9021, longname "frequency" } },
        { unitTypeId 28, name { textId 9028, longname "velocity / speed" } }
    }, unit {
        { unitTypeId 7, unitId 11, name { textId 9111, shortname "°C",
            longname "degree celsius" }, formula 0, c0 1, c1 0, c2 0 },
        { unitTypeId 10, unitId 17, name { textId 9117, shortname "V",
            longname "volt" }, formula 0, c0 1, c1 0, c2 0 },
        { unitTypeId 21, unitId 23, name { textId 9123, shortname "rpm",
            longname "revolutions per minute" }, formula 0, c0 1, c1 0, c2 0 },
        { unitTypeId 28, unitId 24, name { textId 9124, shortname "kph",
            longname "kilometers per hour" }, formula 9, c0 10, c1 36, c2 0 },
        { unitTypeId 28, unitId 60, name { textId 9160, shortname "m/s",
            longname "meter per second" }, formula 0, c0 1, c1 0, c2 0 }
    }, provider {
        { providerId 0, name { textId 9200, longname "UNKNOWN" } },
        { providerId 1, name { textId 9201, shortname "ANY", longname "Any supplier" } }
    }, ecu {
        { ecuId 0, name { textId 13000, shortname "VSG", longname "Vehicle Station Gateway" }, providerId 1 },
        { ecuId 32, name { textId 13007, shortname "ABS", longname "Anti-Lock Brake System" }, providerId 0 },
        { ecuId 51, name { textId 13003, shortname "OBDII", longname "Emissions OBD II System" }, providerId 0 }
    },
```

formulaId	formula
0	y = C0 * x + C1
1	y = C0 * (x + C1)
2	y = C0 / (x + C1) + C2
3	y = x / C0 + C1
5	y = (x + C0) / C1 + C2
9	y = x * C0 / C1
10	y = x + C0 / C1
11	y = (x + C0) * C1 / C2

see ISO 14229-1 UDS
Table C.6 formulaIdentifier encoding

VIDF config 2

```
dataType {
    { dataTypeId 331, name { textId 10180, longname "Voltage in 1/100 V" }, type numeric: {
        decimalPlaces 2, unitId 17, factor 1, quotient 100, addend 0, min 880, max 1560 } },
    { dataTypeId 332, name { textId 10005, longname "Answer (no, yes)" }, type enumString: {
        { value 0, name { textId 10006, longname "no" } },
        { value 1, name { textId 10007, longname "yes" } } } },
    { dataTypeId 333, name { textId 10181, shortname "", longname "Temperature in 1/10 °C" }, type numeric: {
        decimalPlaces 1, unitId 11, factor 1, quotient 10, addend 0 } },
    { dataTypeId 334, name { textId 10000, longname "Unlimited string" }, type string: {} },
    { dataTypeId 360, name { textId 10183, longname "VIN" }, type string: {
        allowedCharacters "A..HJ..NPR..Z0..9", minLen 17, maxLen 17 } },
    { dataTypeId 338, name { textId 10182, longname "RPM" }, type lnumeric: {
        unitId 23, factor 1, quotient 4, addend 0, min 0, max 65535 } },
    { dataTypeId 7, name { textId 10030, longname "Speed in cm/s" }, type numeric: {
        decimalPlaces 2, unitId 60, factor 1, quotient 100, addend 0, min -32765, max 32765 } },
    { dataTypeId 374, name { textId 10008, longname "Switch (off, on)" }, type enumString: {
        { value 0, name { textId 10009, longname "off" } },
        { value 1, name { textId 10010, longname "on" } } } },
    { dataTypeId 337, name { textId 10230, longname "Continuous Remote OBD I/M {}" }, type structure: {
        { 1002, 7368, 2341, 10042 }, convention mandatory } }
},
```

Definition of formula for type numeric, lnumeric

```
realValue = value * factor / quotient + addend
```

DataTypeId 331

```
realValue = value * 1 / 100 + 0 = value / 100
=> min = 8.8 ; max = 15.6
```

VIDF config 3

```
dataParam {
    { rvId 1002, name { textId 11152, shortname "ECMB+", longname "Engine Control Module Voltage" },
        dataTypeId 331, accessType '10000'B, dataParamProperty sensor },
    { rvId 7368, name { textId 10134, shortname "AIR_RDY", longname "Secondary Air System Monitoring Ready" },
        dataTypeId 332, accessType '10000'B, dataParamProperty ecu-internal-monitor },
    { rvId 2341, name { textId 11157, shortname "ECT", longname "Engine Coolant Temperature" },
        dataTypeId 333, accessType '10000'B, dataParamProperty sensor },
    { rvId 1123, name { textId 10131, shortname "HWPNO", longname "Hardware Part Number" },
        dataTypeId 334, accessType '10000'B, dataParamProperty ecu-internal-signal },
    { rvId 461, name { textId 11104, shortname "VIN", longname "Vehicle Identification Number" },
        dataTypeId 360, accessType '10000'B, dataParamProperty ecu-internal-signal },
    { rvId 10042, name { textId 11158, shortname "RPM", longname "Engine RPM" },
        dataTypeId 338, accessType '10000'B, dataParamProperty sensor },
    { rvId 492, name { textId 11112, shortname "FLWS", longname "Front Left Wheel Speed" },
        dataTypeId 360, accessType '10000'B, dataParamProperty ecu-internal-signal },
    { rvId 499, name { textId 11113, shortname "IGSWST", longname "Ignition Switch Status" },
        dataTypeId 374, accessType '10000'B, dataParamProperty sensor },
    { rvId 10050, name { textId 11170, shortname "CROBDIM", longname "Continuous Remote OBD I/M" },
        dataTypeId 337, accessType '10000'B, dataParamProperty collection },
},
```

accessTypes

```
r (0) = read,
w (1) = write,
x (2) = execute,
i (3) = internal
u (4) = user
```

```
'10000'B
= read only
```

dataParamProperty

```
ecu-supported-info,
sensor,
actuator,
ecu-internal-signal,
ecu-internal-monitor,
collection,
routine,
fix,
other
```

VIDF config 4

```
dataParamMapping {
    { rvId 1002, ecuId 51 },
    { rvId 7368, ecuId 51 },
    { rvId 2341, ecuId 51 },
    { rvId 1123, ecuId 51 },
    { rvId 461, ecuId 51 },
    { rvId 10042, ecuId 51 },
    { rvId 492, ecuId 32 },
    { rvId 499, ecuId 0 },
    { rvId 10050, ecuId 0 }
}, fixedValue {
    { rvId 20001, ecuId 0, value string: "passcar_manufacturer_model_year" },
    { rvId 20002, ecuId 0, value enumString: 4 },
    { rvId 20003, ecuId 0, value enumString: 8 },
    { rvId 20004, ecuId 0, value displayName: "Manufacturer" },
    { rvId 20005, ecuId 0, value displayName: "Model" }
},
```

Vehicle Key

Vehicle Type = passcar

Vehicle Class = passenger vehicle

Vehicle Brand

Vehicle Model

VIDF config 5

```
dtcBase {
    { rDtcBaseId 1, ecuId 51, name { textId 20001, longname "Fuel Vol.Regul.Control Circuit/Open" } },
    { rDtcBaseId 2, ecuId 51, name { textId 20002, longname "Fuel Vol.Regul.Control Circuit Range/Performance" } },
    { rDtcBaseId 3, ecuId 51, name { textId 20003, longname "Fuel Vol.Regul.Control Circuit Low" } },
    { rDtcBaseId 4, ecuId 51, name { textId 20004, longname "Fuel Vol.Regul.Control Circuit High" } },
    { rDtcBaseId 5, ecuId 51, name { textId 20005, longname "Fuel Shutoff Valve 'A' Control Circuit/Open" },
        dataParamMapping {
            { rvId 1002, ecuId 17 },
            { rvId 2341, ecuId 17 }
    } },
    ...
    { rDtcBaseId 295, ecuId 51, name { textId 20006, longname "Intake Air Temperature Too High" } },
    ...
    { rDtcBaseId 16433, ecuId 51, name { textId 20007, longname "Left Front Wheel Speed Sensor" } },
    { rDtcBaseId 32825, ecuId 51, name { textId 20008, longname "Second Row Right Front.Stage 1 Deploym.Control" } },
    { rDtcBaseId 49280, ecuId 51, name { textId 20009, longname "Vehicle Communication Bus 'F'" } },
    ...
},
dtcSymptom {
    { rDtcSymptomId 0, name { textId 21000, longname "No Sub Type Information" } },
    { rDtcSymptomId 1, name { textId 21001, longname "General Electrical Failure" } } },
    { rDtcSymptomId 2, name { textId 21002, longname "General Signal Failure" } },
    { rDtcSymptomId 3, name { textId 21003, longname "FM (Frequency Modulated) / PWM (Pulse Width Module)" } },
    { rDtcSymptomId 4, name { textId 21004, longname "System Internal Failure" } },
    { rDtcSymptomId 5, name { textId 21005, longname "System Programming Failure" } },
    { rDtcSymptomId 6, name { textId 21006, longname "Algorithm Based Failure" } },
    { rDtcSymptomId 7, name { textId 21007, longname "Mechanical Failure" } },
    ...
    { rDtcSymptomId 255, name { textId 21255, longname "Manufacturer Defined" } }
}
}
```

Environment data

Replied data

```
{ rvId 461, ecuId 51, choice 2, stringValue "1M8GDM9A1KP042788" },
{ rvId 10042, ecuId 51, choice 1, iValue 5940 },
{ rvId 2341, ecuId 51, choice 0, iValue 1124 }, ...
```