A Service Oriented Architecture SOME/IP

Samir Bhagwat

Director [Engg. Services & Consulting]

sbhagwat@intrepidcs.com

Intrepid Control Systems

What is SOME/IP

An Automotive specific RPC mechanism and underlying Serialization format

- ECU A (Server) can provide a ‘function’ – a service
- ECU B (Client) can ‘consume’ that service by requesting methods with right parameters. Note - these parameters can be generic data structures.

Consumer ECU

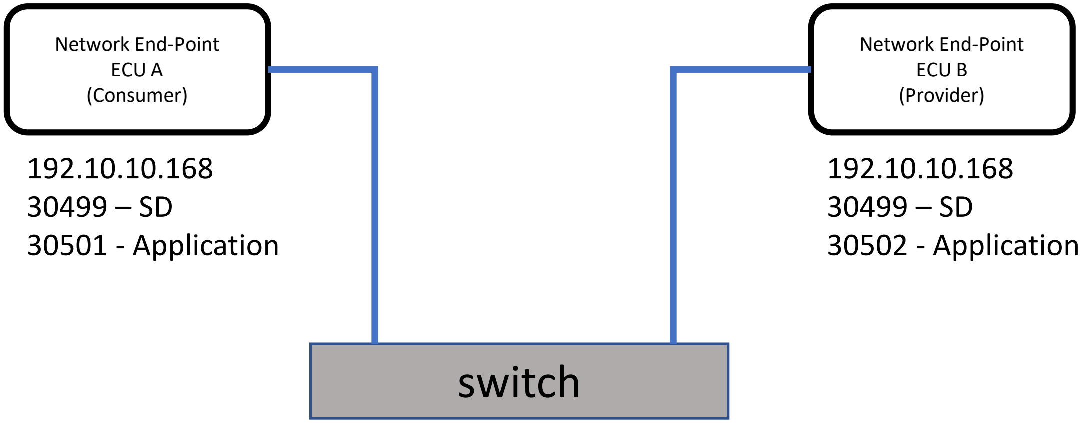
-Subscribes to a ‘Service’

- Receives Events Notification
- Can Invoke ‘Method’ e.g. get, set
- Receives Field Notify
- Can send REQ and receive RESP

Provider ECU

- Offers a ‘Service’ + Endpoint Details
- has Fields , Events
- has methods
- Fields can have getters, setters, notifiers

Scalable **S**ervice **O**riented **M**iddlewar**E** over **IP**



192.10.10.168
30499 – SD
30501 - Application

192.10.10.168
30499 – SD
30502 - Application

Why SOME/IP

- Classical Signal based (e.g CAN) communication - Insufficient for more complex data communication and control.

- Highly Computation intensive ECU's - can provide needed intelligence to other ECU's when required.

- Advent of Ethernet as In-Vehicle Network – Classical Signal based + Dynamic contract + bandwidth

- Complex Service Interfaces with Methods and Events can be transacted using Ethernet == Service Oriented Middleware Protocol using TCP/UDP over IP

Why SOME/IP

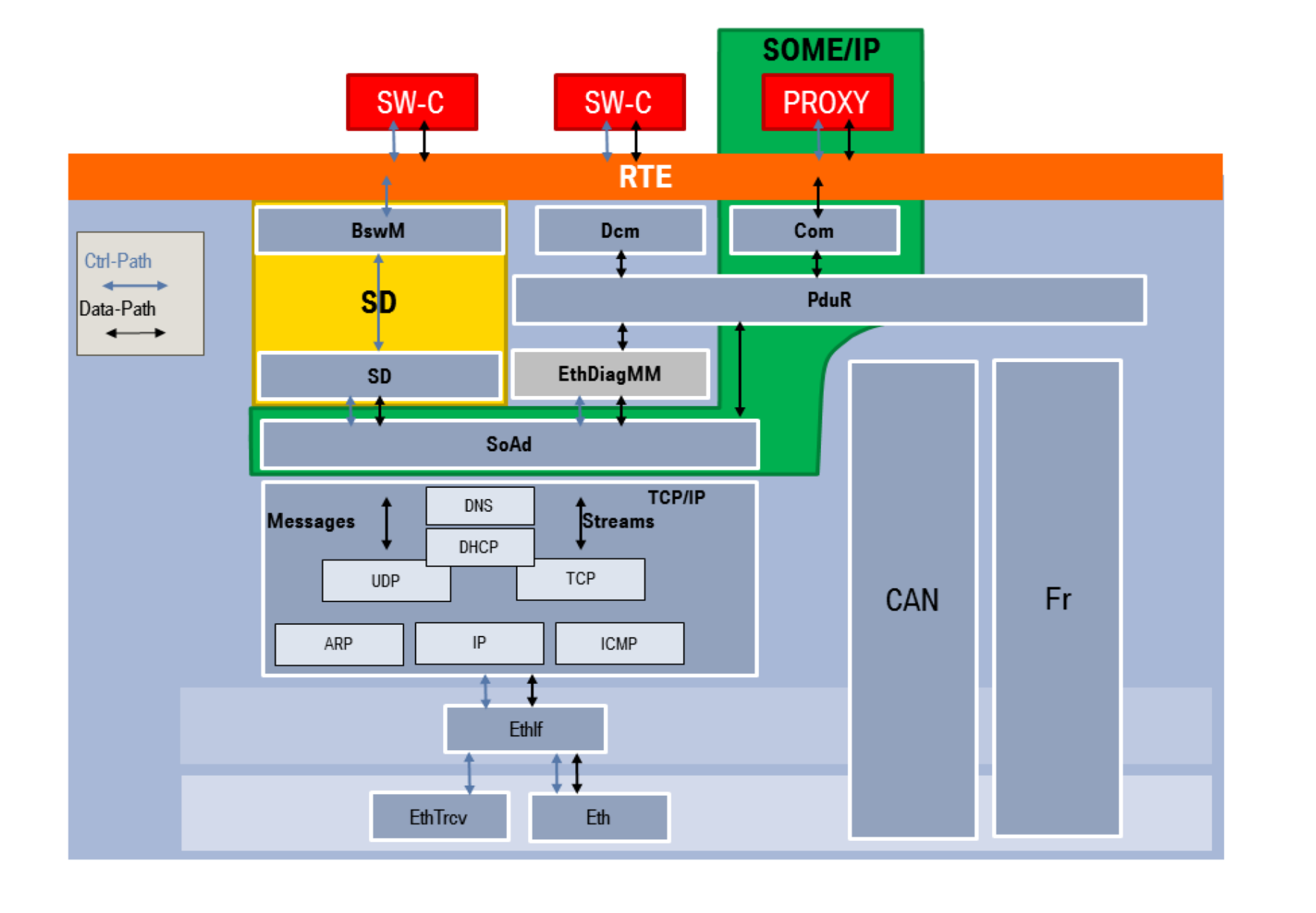
- Add new functions and features to vehicle

- Integration of new capabilities with minimal changes to existing functionality implementation

- Reduced problems of static defined interfaces and data exchange in complex modern architectures

- Enabler for vehicle as part of highly connected and digital world

SOME/IP layer in Autosar ECU Stack

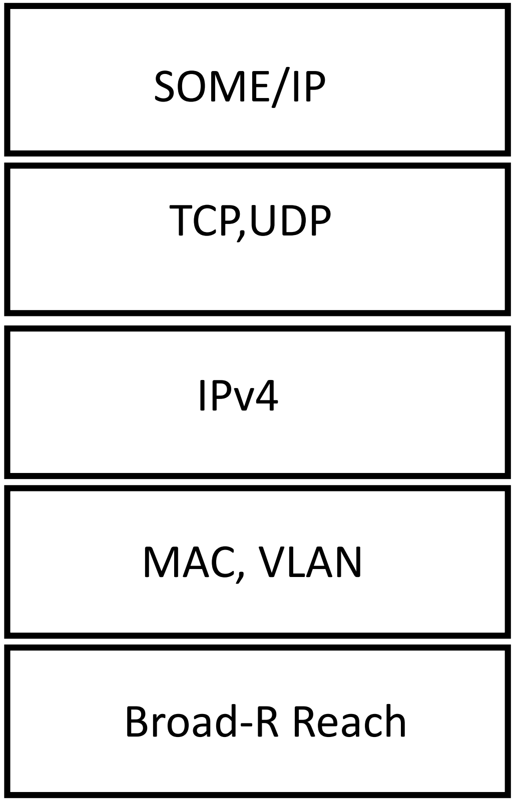


ref: autosar.org

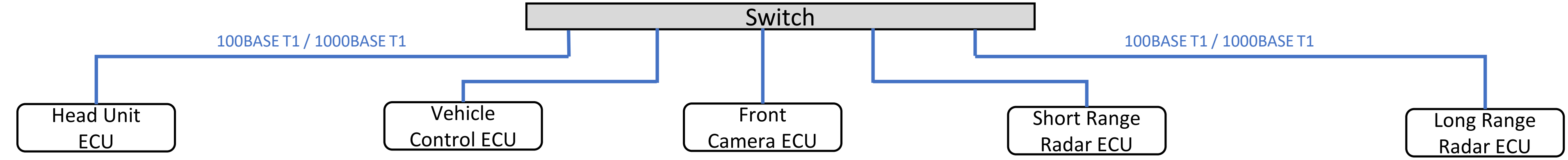
Just Another Middleware ?

Scalable Service
Oriented
Middlewar**E** over
IP

- ✓ Suited for future architectures
- ✓ Described and Specified
- ✓ Testable
- ✓ Can carry Autosar PDUs


SOME/IP
TCP,UDP
IPv4
MAC, VLAN
Broad-R Reach

Example of In-vehicle SOME/IP based communication



IP = 192.168.10.103
Port = 30505

Provided Service Interface :
Audio_Control
Event Group :
Front_Controls
Rear_Controls
Fields :
Cntl_Front_Side
(getter_1, notifier_1)
Cntl_Rear_Side
(getter_2, notifier_2)

IP = 192.168.10.102
Port = 30505

Provided Service Interface :
Obstacle_Avoidance
Events :
X_New_Position_Ev,
Y_New_Position_Ev
Event Group :
Azimuth_Data
Fields :
Obstacle_Type
Methods:
getter_1,setter_1,
notifier_1
Consumed Service Interface :
Object_Recognition
Consumed Service Interface :
Park_Assist_Info

IP = 192.168.10.100
Port = 30501

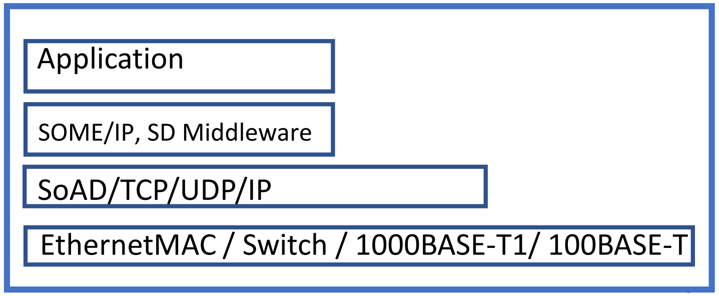
Provided Service Interface :
Object_Recognition
Event Group :
Object_Event_Grp_1
Events :
Object_New_Position,
Object_Now_Blurred

IP = 192.168.10.101
Port = 30501

Provided Service Interface :
Park_Assist_Info
Event Group :
Distance_Data
Fields :
Front_Distance (Notifier_1)
Rear_Distance (Notifier_2)

IP = 192.168.10.101
Port = 30501

Simplistic view of an ECU stack for SOME/IP

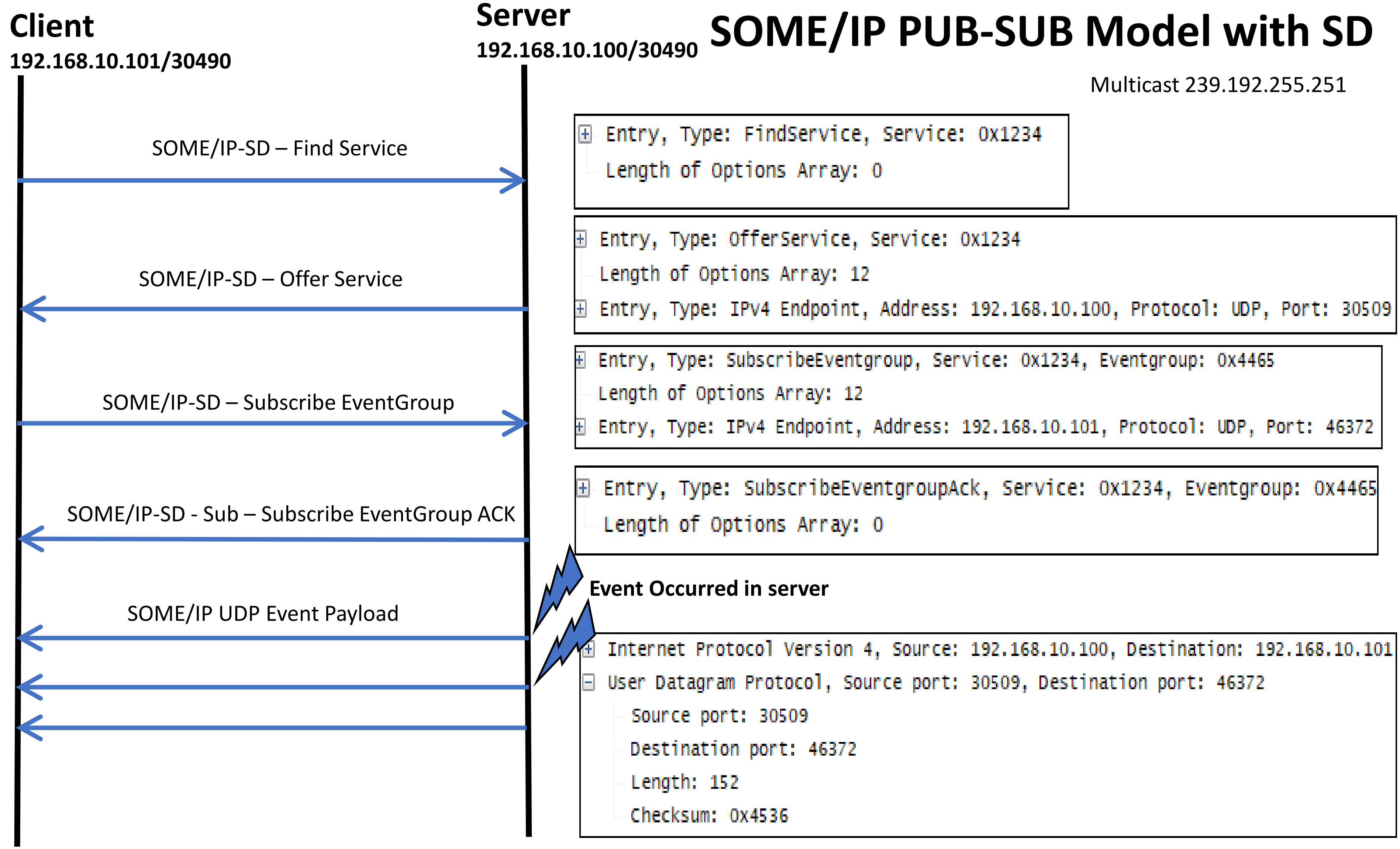

Application

SOME/IP, SD Middleware

SoAD/TCP/UDP/IP

EthernetMAC / Switch / 1000BASE-T1/ 100BASE-T

SOME/IP PUB-SUB Model with SD


Client
192.168.10.101/30490
Server
192.168.10.100/30490
Multicast 239.192.255.251
SOME/IP-SD – Find Service
SOME/IP-SD – Offer Service
SOME/IP-SD – Subscribe EventGroup
SOME/IP-SD - Sub – Subscribe EventGroup ACK
SOME/IP UDP Event Payload
Event Occurred in server

```
⊞ Entry, Type: FindService, Service: 0x1234
   Length of Options Array: 0
```

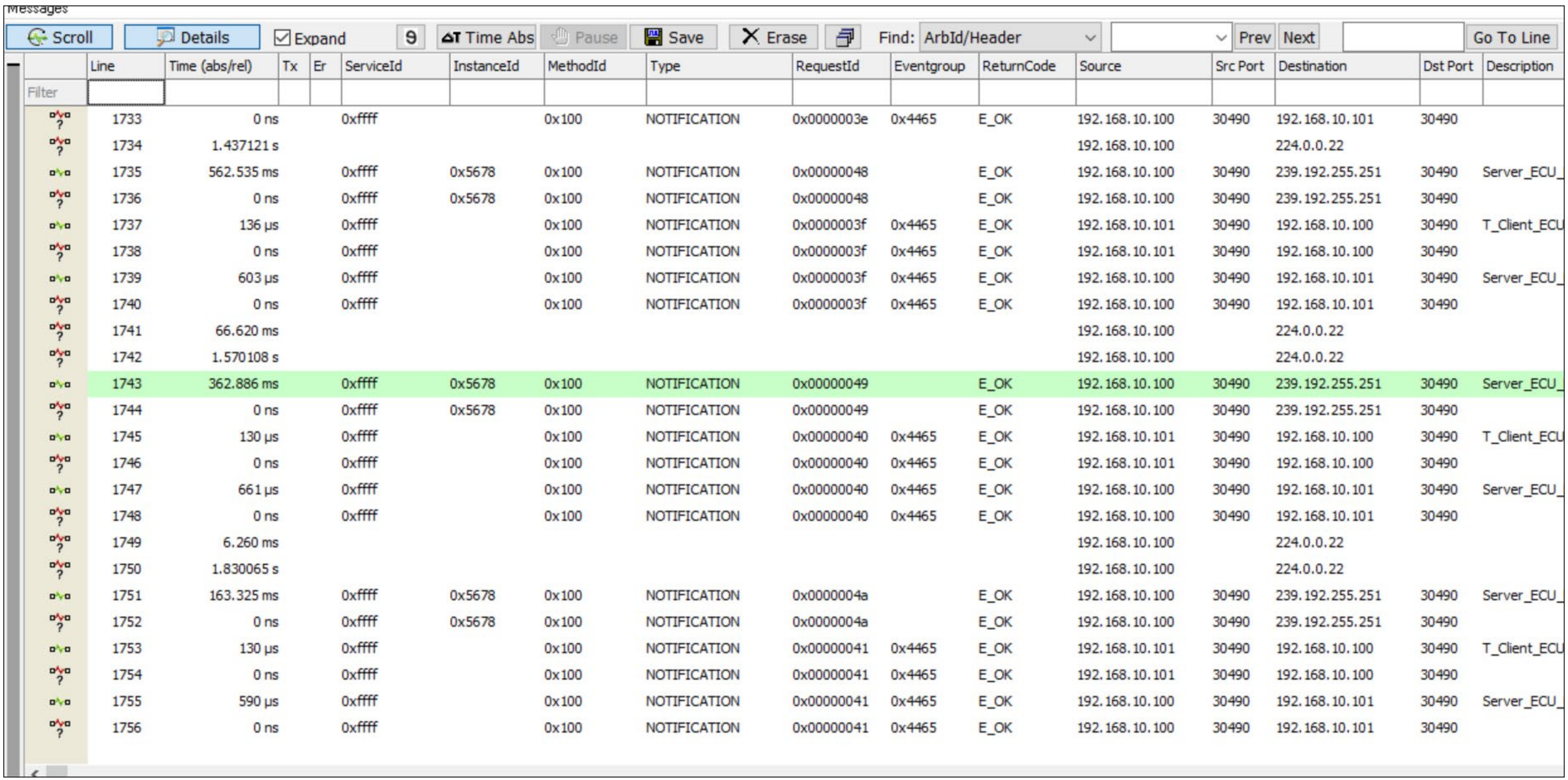
```
⊞ Entry, Type: OfferService, Service: 0x1234
   Length of Options Array: 12
⊞ Entry, Type: IPv4 Endpoint, Address: 192.168.10.100, Protocol: UDP, Port: 30509
```

```
⊞ Entry, Type: SubscribeEventgroup, Service: 0x1234, Eventgroup: 0x4465
   Length of Options Array: 12
⊞ Entry, Type: IPv4 Endpoint, Address: 192.168.10.101, Protocol: UDP, Port: 46372
```

```
⊞ Entry, Type: SubscribeEventgroupAck, Service: 0x1234, Eventgroup: 0x4465
   Length of Options Array: 0
```

```
⊞ Internet Protocol Version 4, Source: 192.168.10.100, Destination: 192.168.10.101
⊟ User Datagram Protocol, Source port: 30509, Destination port: 46372
     Source port: 30509
     Destination port: 46372
     Length: 152
     Checksum: 0x4536
```

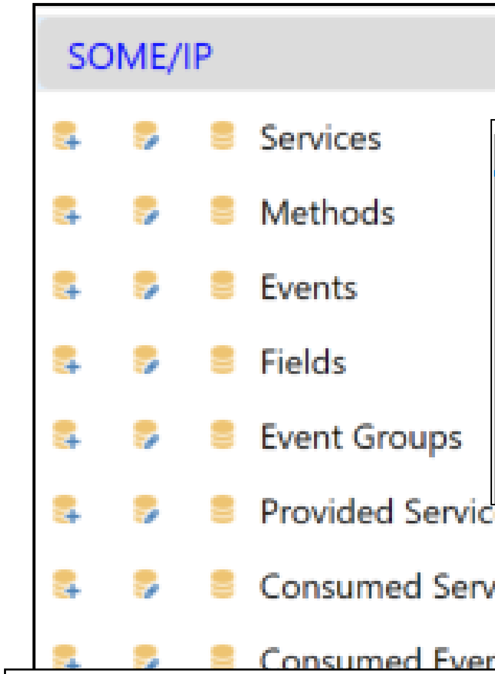
SOME/IP Messaging

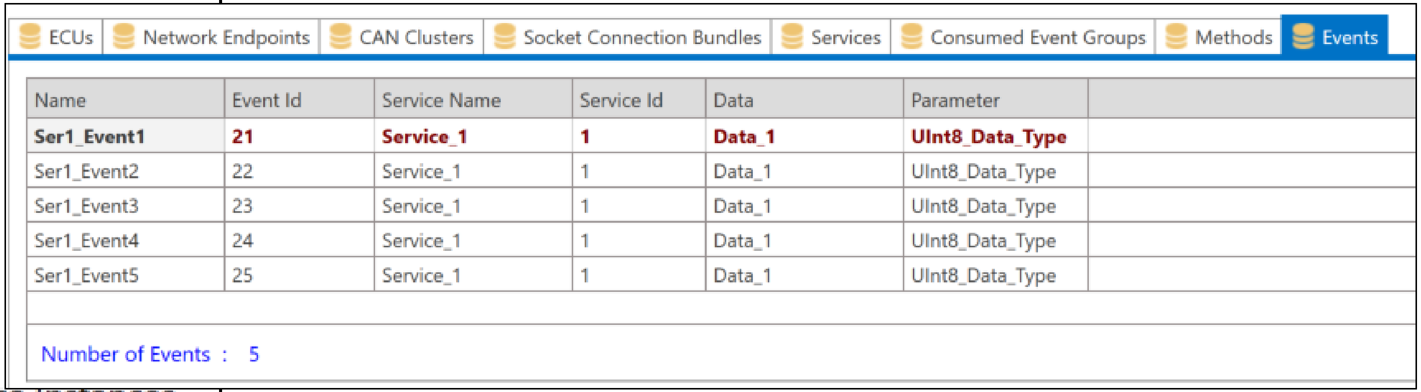


Messages

	Line	Time (abs/rel)	Tx	Er	ServiceId	InstanceId	MethodId	Type	RequestId	Eventgroup	ReturnCode	Source	Src Port	Destination	Dst Port	Description
Filter																
	1733	0 ns			0xffff		0x100	NOTIFICATION	0x0000003e	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	
	1734	1.437121 s										192.168.10.100		224.0.0.22		
	1735	562.535 ms			0xffff	0x5678	0x100	NOTIFICATION	0x00000048		E_OK	192.168.10.100	30490	239.192.255.251	30490	Server_ECU_
	1736	0 ns			0xffff	0x5678	0x100	NOTIFICATION	0x00000048		E_OK	192.168.10.100	30490	239.192.255.251	30490	
	1737	136 µs			0xffff		0x100	NOTIFICATION	0x0000003f	0x4465	E_OK	192.168.10.101	30490	192.168.10.100	30490	T_Client_ECU
	1738	0 ns			0xffff		0x100	NOTIFICATION	0x0000003f	0x4465	E_OK	192.168.10.101	30490	192.168.10.100	30490	
	1739	603 µs			0xffff		0x100	NOTIFICATION	0x0000003f	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	Server_ECU_
	1740	0 ns			0xffff		0x100	NOTIFICATION	0x0000003f	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	
	1741	66.620 ms										192.168.10.100		224.0.0.22		
	1742	1.570108 s										192.168.10.100		224.0.0.22		
	1743	362.886 ms			0xffff	0x5678	0x100	NOTIFICATION	0x00000049		E_OK	192.168.10.100	30490	239.192.255.251	30490	Server_ECU_
	1744	0 ns			0xffff	0x5678	0x100	NOTIFICATION	0x00000049		E_OK	192.168.10.100	30490	239.192.255.251	30490	
	1745	130 µs			0xffff		0x100	NOTIFICATION	0x00000040	0x4465	E_OK	192.168.10.101	30490	192.168.10.100	30490	T_Client_ECU
	1746	0 ns			0xffff		0x100	NOTIFICATION	0x00000040	0x4465	E_OK	192.168.10.101	30490	192.168.10.100	30490	
	1747	661 µs			0xffff		0x100	NOTIFICATION	0x00000040	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	Server_ECU_
	1748	0 ns			0xffff		0x100	NOTIFICATION	0x00000040	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	
	1749	6.260 ms										192.168.10.100		224.0.0.22		
	1750	1.830065 s										192.168.10.100		224.0.0.22		
	1751	163.325 ms			0xffff	0x5678	0x100	NOTIFICATION	0x0000004a		E_OK	192.168.10.100	30490	239.192.255.251	30490	Server_ECU_
	1752	0 ns			0xffff	0x5678	0x100	NOTIFICATION	0x0000004a		E_OK	192.168.10.100	30490	239.192.255.251	30490	
	1753	130 µs			0xffff		0x100	NOTIFICATION	0x00000041	0x4465	E_OK	192.168.10.101	30490	192.168.10.100	30490	T_Client_ECU
	1754	0 ns			0xffff		0x100	NOTIFICATION	0x00000041	0x4465	E_OK	192.168.10.101	30490	192.168.10.100	30490	
	1755	590 µs			0xffff		0x100	NOTIFICATION	0x00000041	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	Server_ECU_
	1756	0 ns			0xffff		0x100	NOTIFICATION	0x00000041	0x4465	E_OK	192.168.10.100	30490	192.168.10.101	30490	

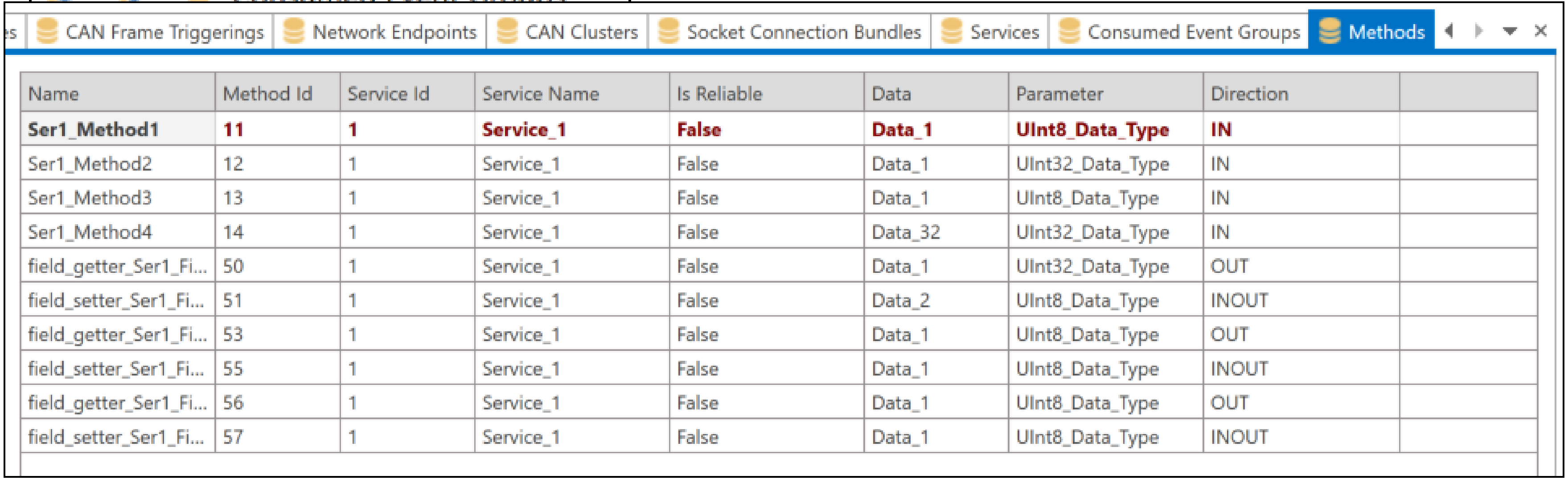
SOME/IP ARXML





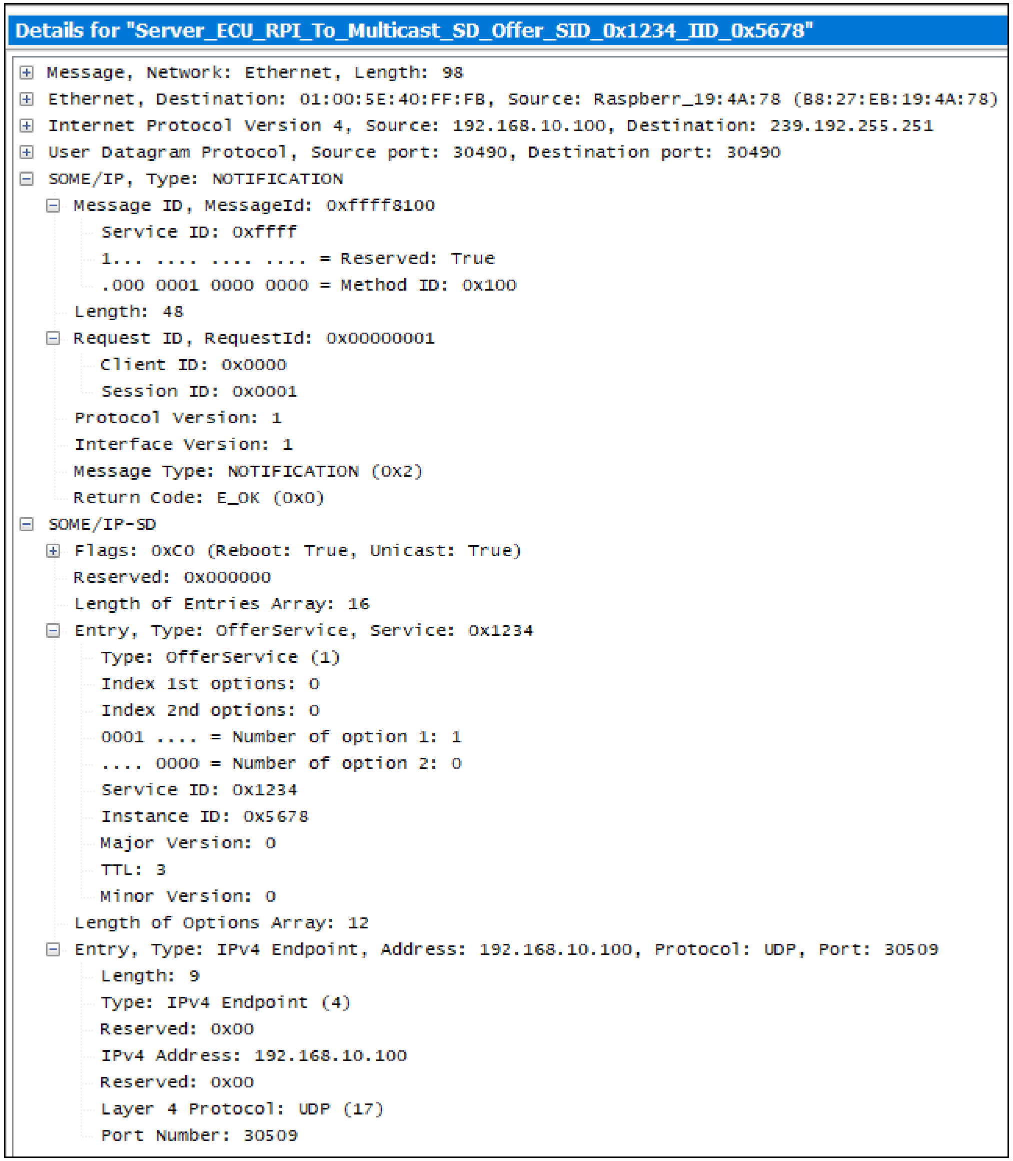
Name	Event Id	Service Name	Service Id	Data	Parameter
Ser1_Event1	**21**	**Service_1**	**1**	**Data_1**	**UInt8_Data_Type**
Ser1_Event2	22	Service_1	1	Data_1	UInt8_Data_Type
Ser1_Event3	23	Service_1	1	Data_1	UInt8_Data_Type
Ser1_Event4	24	Service_1	1	Data_1	UInt8_Data_Type
Ser1_Event5	25	Service_1	1	Data_1	UInt8_Data_Type

Number of Events : 5



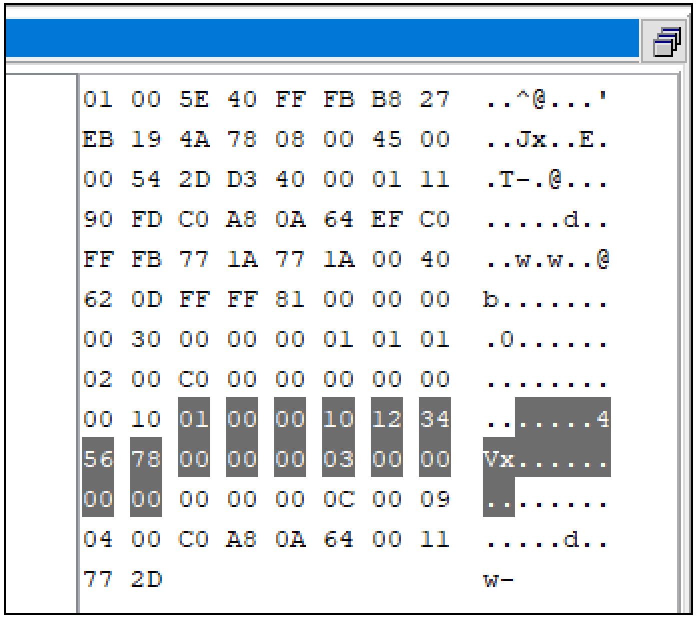
Name	Method Id	Service Id	Service Name	Is Reliable	Data	Parameter	Direction
Ser1_Method1	**11**	**1**	**Service_1**	**False**	**Data_1**	**UInt8_Data_Type**	**IN**
Ser1_Method2	12	1	Service_1	False	Data_1	UInt32_Data_Type	IN
Ser1_Method3	13	1	Service_1	False	Data_1	UInt8_Data_Type	IN
Ser1_Method4	14	1	Service_1	False	Data_32	UInt32_Data_Type	IN
field_getter_Ser1_Fi...	50	1	Service_1	False	Data_1	UInt32_Data_Type	OUT
field_setter_Ser1_Fi...	51	1	Service_1	False	Data_2	UInt8_Data_Type	INOUT
field_getter_Ser1_Fi...	53	1	Service_1	False	Data_1	UInt8_Data_Type	OUT
field_setter_Ser1_Fi...	55	1	Service_1	False	Data_1	UInt8_Data_Type	INOUT
field_getter_Ser1_Fi...	56	1	Service_1	False	Data_1	UInt8_Data_Type	OUT
field_setter_Ser1_Fi...	57	1	Service_1	False	Data_1	UInt8_Data_Type	INOUT

SOME/IP Messaging Deep Details



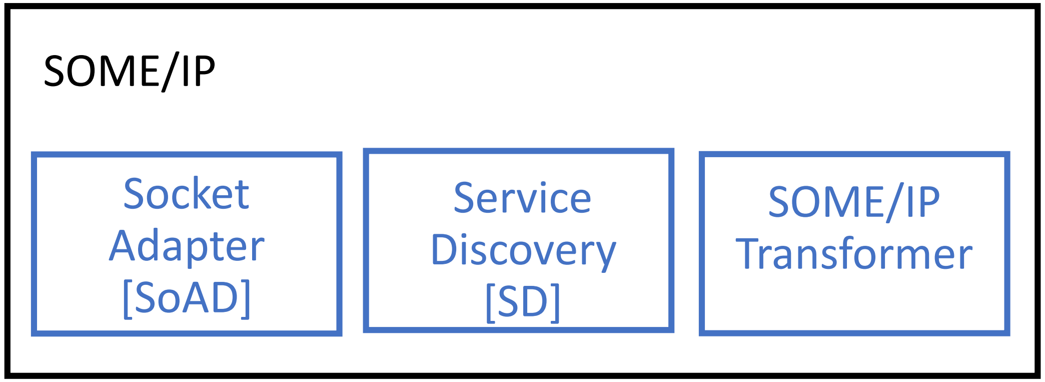
Details for "Server_ECU_RPI_To_Multicast_SD_Offer_SID_0x1234_IID_0x5678"

```
⊞ Message, Network: Ethernet, Length: 98
⊞ Ethernet, Destination: 01:00:5E:40:FF:FB, Source: Raspberr_19:4A:78 (B8:27:EB:19:4A:78)
⊞ Internet Protocol Version 4, Source: 192.168.10.100, Destination: 239.192.255.251
⊞ User Datagram Protocol, Source port: 30490, Destination port: 30490
⊟ SOME/IP, Type: NOTIFICATION
   ⊟ Message ID, MessageId: 0xffff8100
        Service ID: 0xffff
        1... .... .... .... = Reserved: True
        .000 0001 0000 0000 = Method ID: 0x100
     Length: 48
   ⊟ Request ID, RequestId: 0x00000001
        Client ID: 0x0000
        Session ID: 0x0001
     Protocol Version: 1
     Interface Version: 1
     Message Type: NOTIFICATION (0x2)
     Return Code: E_OK (0x0)
⊟ SOME/IP-SD
   ⊞ Flags: 0xC0 (Reboot: True, Unicast: True)
     Reserved: 0x000000
     Length of Entries Array: 16
   ⊟ Entry, Type: OfferService, Service: 0x1234
        Type: OfferService (1)
        Index 1st options: 0
        Index 2nd options: 0
        0001 .... = Number of option 1: 1
        .... 0000 = Number of option 2: 0
        Service ID: 0x1234
        Instance ID: 0x5678
        Major Version: 0
        TTL: 3
        Minor Version: 0
     Length of Options Array: 12
   ⊟ Entry, Type: IPv4 Endpoint, Address: 192.168.10.100, Protocol: UDP, Port: 30509
        Length: 9
        Type: IPv4 Endpoint (4)
        Reserved: 0x00
        IPv4 Address: 192.168.10.100
        Reserved: 0x00
        Layer 4 Protocol: UDP (17)
        Port Number: 30509
```



```
01 00 5E 40 FF FB B8 27   ..^@...'
EB 19 4A 78 08 00 45 00   ..Jx..E.
00 54 2D D3 40 00 01 11   .T-.@...
90 FD C0 A8 0A 64 EF C0   .....d..
FF FB 77 1A 77 1A 00 40   ..w.w..@
62 0D FF FF 81 00 00 00   b.......
00 30 00 00 00 01 01 01   .0......
02 00 C0 00 00 00 00 00   ........
00 10 01 00 00 10 12 34   .......4
56 78 00 00 00 03 00 00   Vx......
00 00 00 00 00 0C 00 09   ........
04 00 C0 A8 0A 64 00 11   .....d..
77 2D                     w-
```

SOME/IP - 3 Primary Components



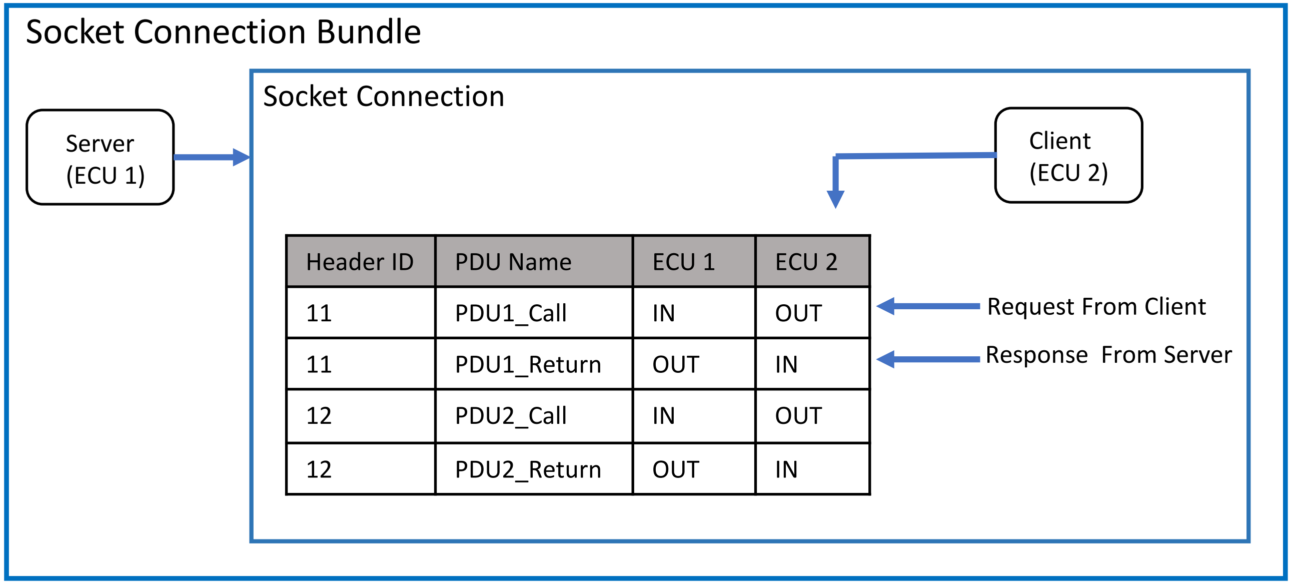
SOME/IP - Socket Adapter [SoAD]
What is it?

- The SoAd Layer enables PDU-based communication via the TCP/IP network.

- AUTOSAR I-PDUs are mapped to socket connections which are configured and maintained by SoAd.

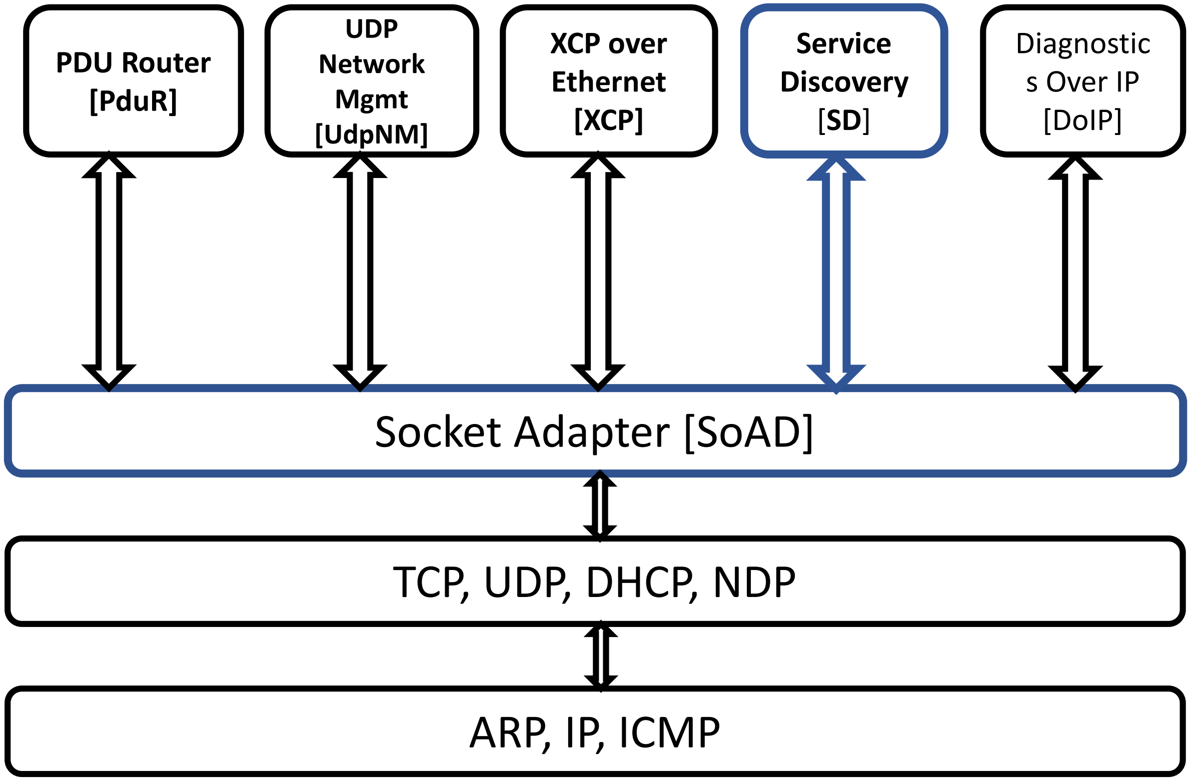
- To use a socket connection for more than one I-PDU a SoAd PDU Header can be added in front of each I-PDU.

- PDU routing groups Manage – Enable disable routing of PDU's from / to a socket



Header ID	PDU Name	ECU 1	ECU 2
11	PDU1_Call	IN	OUT
11	PDU1_Return	OUT	IN
12	PDU2_Call	IN	OUT
12	PDU2_Return	OUT	IN

SOME/IP - Socket Adapter [SoAD] – AUTOSAR MODEL
building block for generic upper layer support



SOME/IP - Service Discovery [SD]

Lets become familiar with some terms…what means what

Term	Description
Service	A functional entity that offers an interface. A logical combination of zero or more methods, zero or more events, and zero or more fields.
Service Instance	A single instance of the Service. Implementation of a service, which can exist more than once in the vehicle and more than once on an ECU
Offer Service	A message entry that declares offer of a Service Instance
Stop Offer	A message entry that declares stopping of a Service Instance
Find Service	A message entry used to find a Service Instance.
Event	A message send by an ECU implementing a Service Instance to an ECU using this Service Instance.
EventGroup	A logical grouping of 1 or more events. An EventGroup is part of a Service.
Field	A field does represent a status and thus has an valid value at all times on which getter, setter and notifier act upon.
Getter/Setter	A Request/Response call that allows Read/Write access to a field.
Method	A method, procedure, function, or subroutine that is called/invoked.
Notification Event	An event message of the notifier of a field.
Notifier	Sends out event message with a new value on change of the value of the field.
Remote Procedure Call (RPC)	A method call from one ECU to another that is transmitted using messages

SOME/IP - Service Discovery [SD]

SD – Server role

# Respond to ‘Find’

# Offer a service

# Take back the offer

SD – Client role

# Sends ‘Find’

# Listen for Offers

# Listen for stop offers

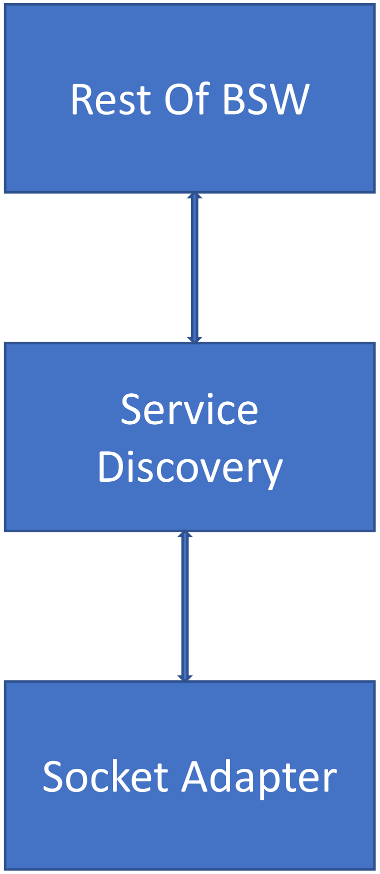
SOME/IP - Service Discovery [SD]

# Functions –

- Describes Service offered or consumed ,
- Describes Instance of the service
- Describes Event Group ID’s
- Transfer of IPV4 End Point Details (TCP or UDP, Port number)
- Receives and Transmits SD messages

# Messages (aka Service Discovery ‘Entry Types’) –

- FindService
- OfferService
- StopServiceOffer
- SubscribeEventgroup
- StopSubscribeEventgroup
- SubscribeEventgroupAck
- SubscribeEventgroupNack

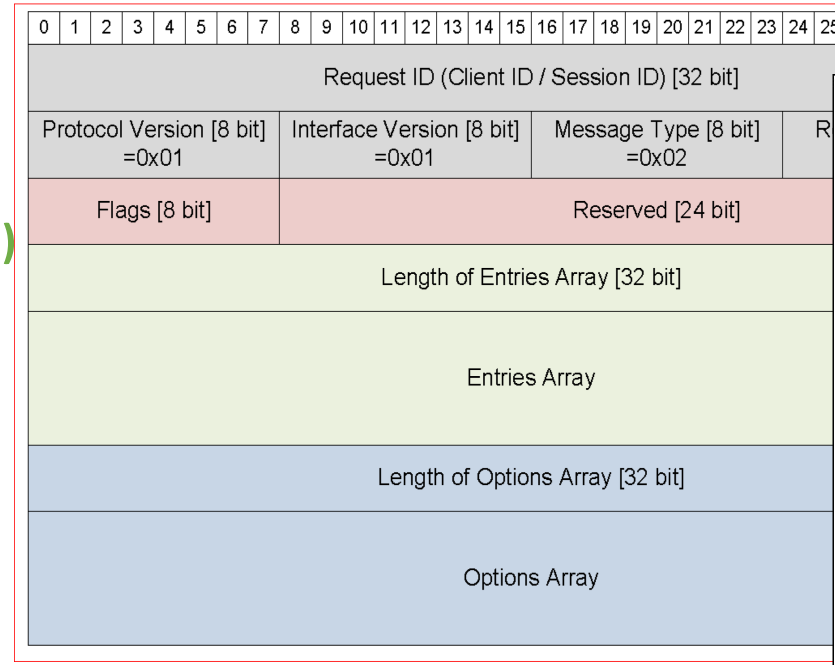


SOME/IP - Service Discovery [SD] - Frames

A (Message Header Area)

B (Entries Array Area)

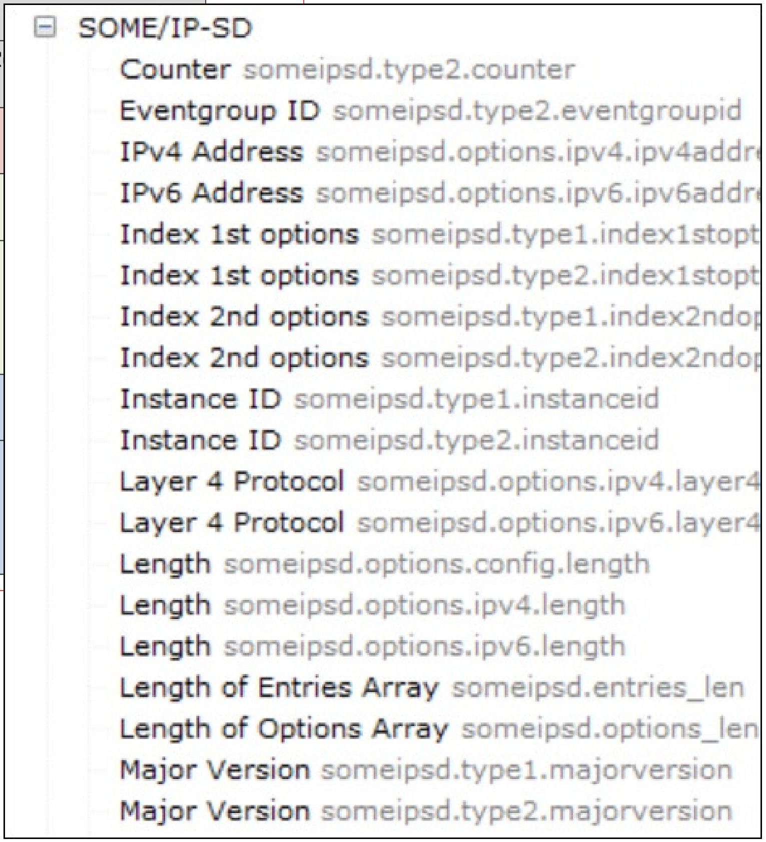
C (Options Array Area)



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	bit offset

Request ID (Client ID / Session ID) [32 bit]			
Protocol Version [8 bit] =0x01	Interface Version [8 bit] =0x01	Message Type [8 bit] =0x02	R
Flags [8 bit]	Reserved [24 bit]		
Length of Entries Array [32 bit]			
Entries Array			
Length of Options Array [32 bit]			
Options Array			

Autosar specs

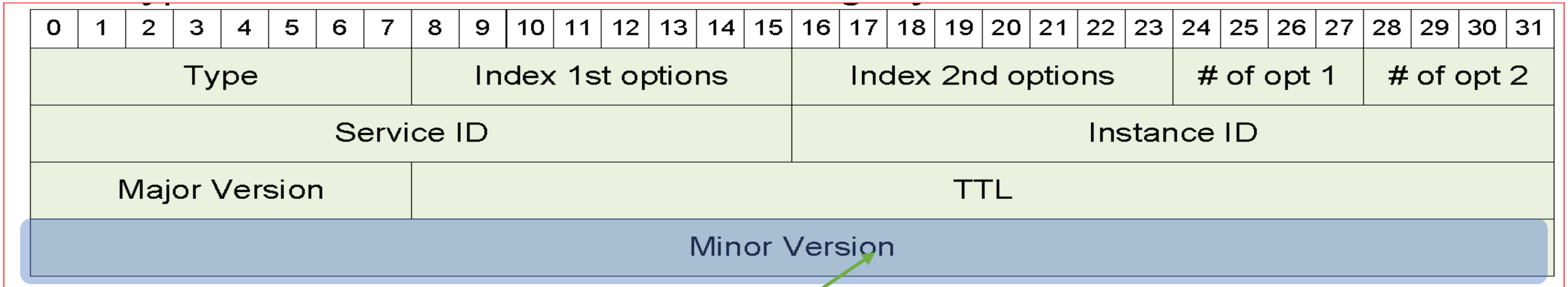


```
⊟ SOME/IP-SD
    Counter someipsd.type2.counter
    Eventgroup ID someipsd.type2.eventgroupid
    IPv4 Address someipsd.options.ipv4.ipv4addre
    IPv6 Address someipsd.options.ipv6.ipv6addre
    Index 1st options someipsd.type1.index1stopt
    Index 1st options someipsd.type2.index1stopt
    Index 2nd options someipsd.type1.index2ndop
    Index 2nd options someipsd.type2.index2ndop
    Instance ID someipsd.type1.instanceid
    Instance ID someipsd.type2.instanceid
    Layer 4 Protocol someipsd.options.ipv4.layer4
    Layer 4 Protocol someipsd.options.ipv6.layer4
    Length someipsd.options.config.length
    Length someipsd.options.ipv4.length
    Length someipsd.options.ipv6.length
    Length of Entries Array someipsd.entries_len
    Length of Options Array someipsd.options_len
    Major Version someipsd.type1.majorversion
    Major Version someipsd.type2.majorversion
```

Overview of the Service Discovery message format
From AUTOSAR

SOME/IP - Service Discovery [SD]

Entries Array Entry Format Types



<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="8">Type</td><td colspan="8">Index 1st options</td><td colspan="8">Index 2nd options</td><td colspan="4"># of opt 1</td><td colspan="4"># of opt 2</td></tr>
<tr><td colspan="16">Service ID</td><td colspan="16">Instance ID</td></tr>
<tr><td colspan="8">Major Version</td><td colspan="24">TTL</td></tr>
<tr><td colspan="32">Minor Version</td></tr>
</table>

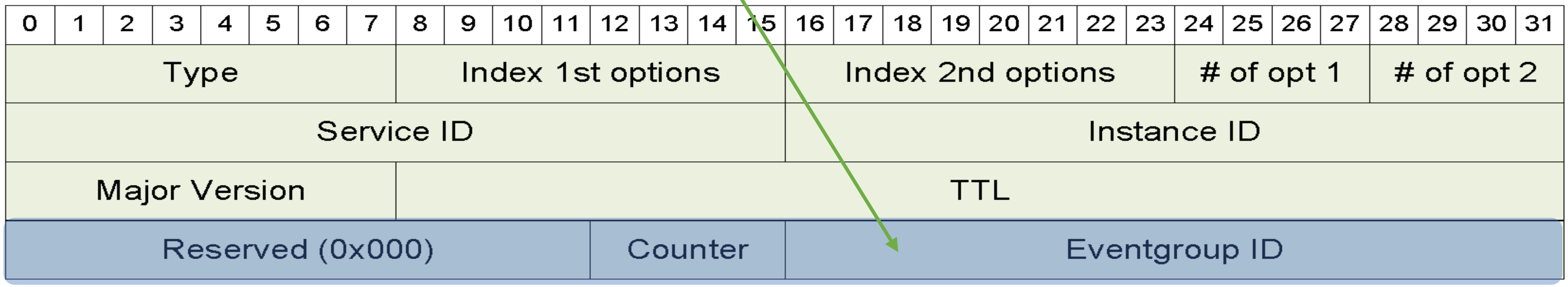
Services

Format Type 1 Entry

Fields differ only here

Format Type 2 Entry

Eventgroups

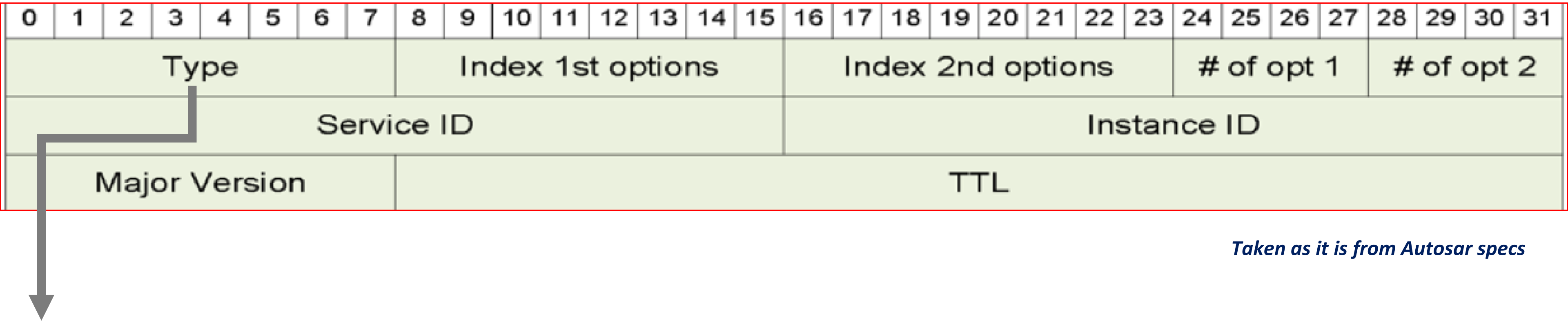


<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="8">Type</td><td colspan="8">Index 1st options</td><td colspan="8">Index 2nd options</td><td colspan="4"># of opt 1</td><td colspan="4"># of opt 2</td></tr>
<tr><td colspan="16">Service ID</td><td colspan="16">Instance ID</td></tr>
<tr><td colspan="8">Major Version</td><td colspan="24">TTL</td></tr>
<tr><td colspan="12">Reserved (0x000)</td><td colspan="4">Counter</td><td colspan="16">Eventgroup ID</td></tr>
</table>

Taken as it is from Autosar specs

SOME/IP - Service Discovery [SD]

Entries Array Entry Formats - Type encodings



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type								Index 1st options								Index 2nd options								# of opt 1				# of opt 2			
Service ID																Instance ID															
Major Version								TTL																							

Taken as it is from Autosar specs

Type = 0x00 encodes "FindService"

Type = 0x01 encodes "OfferService" And "StopOfferService"

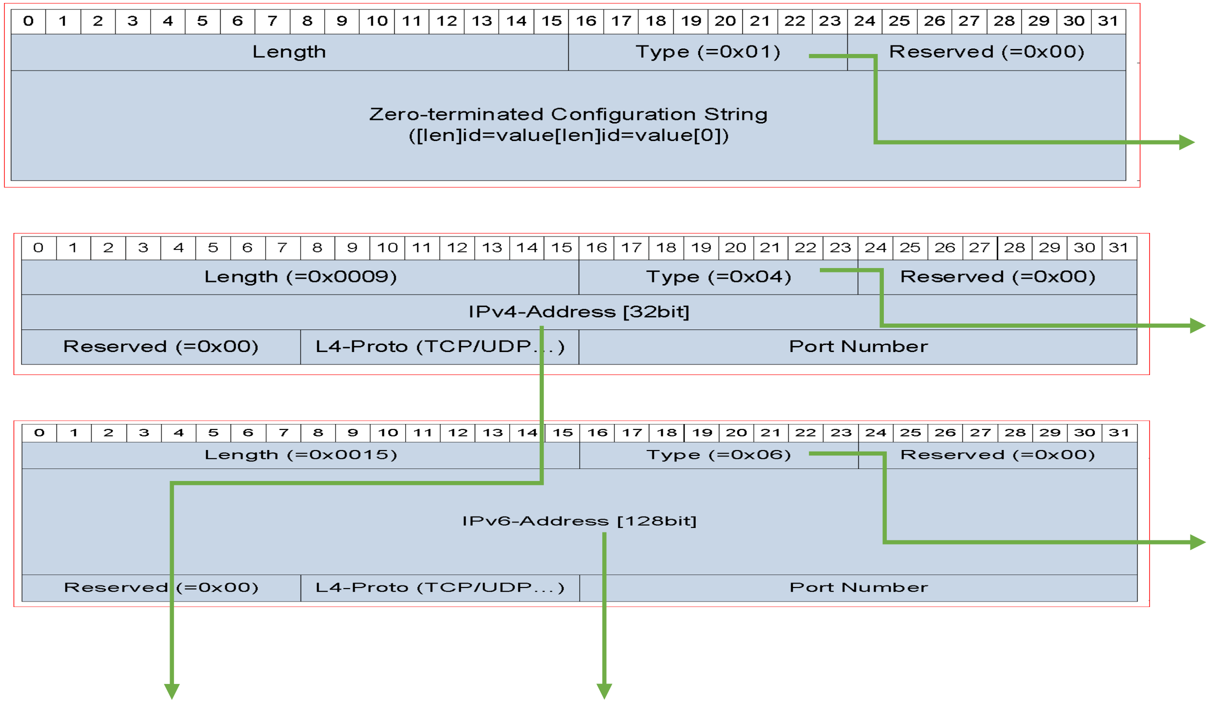
Type = 0x06 encodes "SubscribeEventGroup" And "StopSubscribeEventGroup"

Type = 0x07 encodes "SubscribeEventGroupAck" And "StopSubscribeEventGroupNack"

Type = 0x02, 0x03, 0x04, 0x05 not defined

SOME/IP - Service Discovery [SD]

Options Array Field - Transports additional information



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Length | Type (=0x01) | Reserved (=0x00)

Zero-terminated Configuration String
([len]id=value[len]id=value[0])

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Length (=0x0009) | Type (=0x04) | Reserved (=0x00)

IPv4-Address [32bit]

Reserved (=0x00) | L4-Proto (TCP/UDP…) | Port Number

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Length (=0x0015) | Type (=0x06) | Reserved (=0x00)

IPv6-Address [128bit]

Reserved (=0x00) | L4-Proto (TCP/UDP…) | Port Number

Type = 0x01 => option = “Configuration Option”

Type = 0x04 => option = “IPv4 Endpoint Option”

Type = 0x06 => option = “IPv6 Endpoint Option”

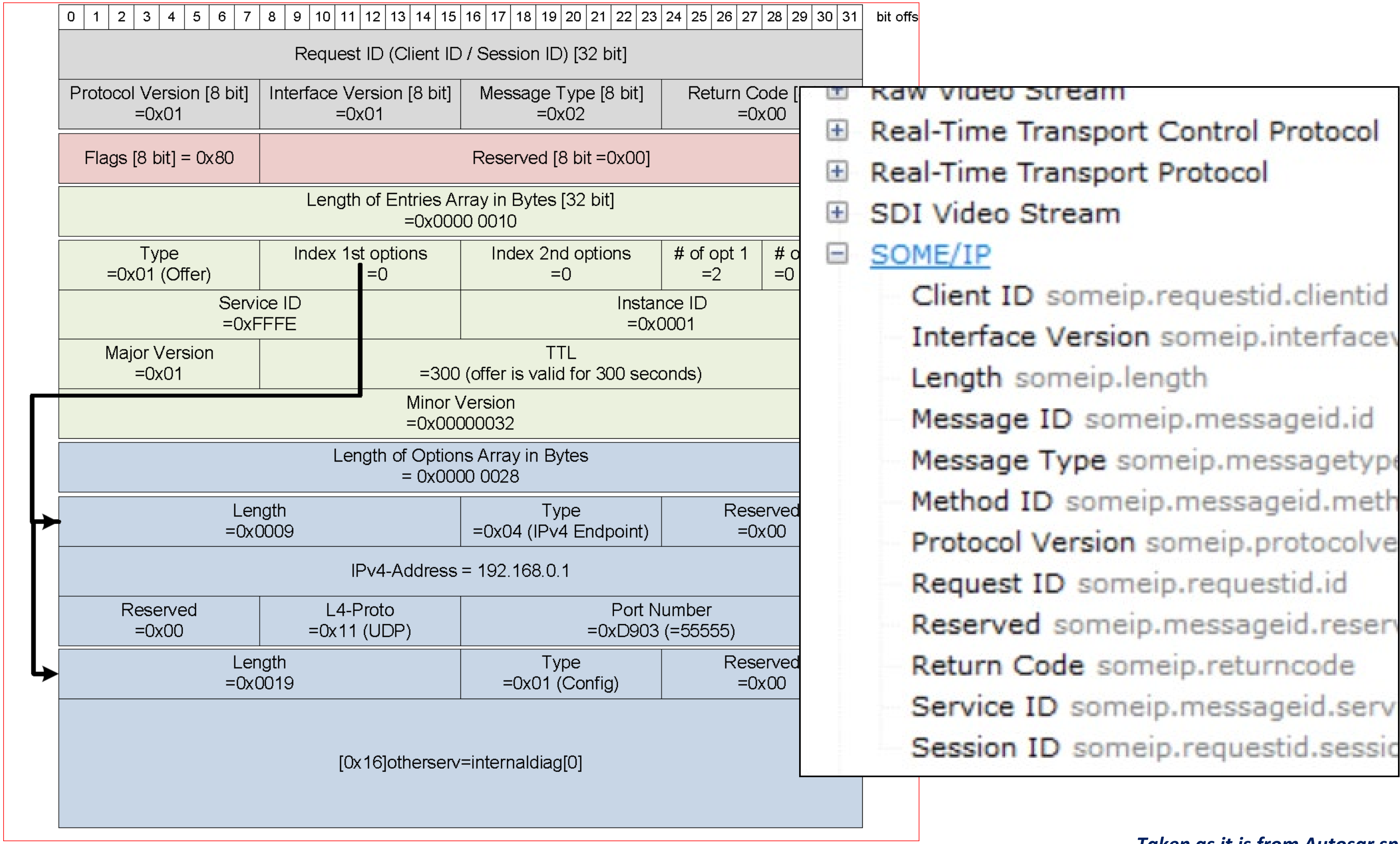
Type = 0x14 => option = “IPv4 Multicast Option”
Type = 0x16 => option = “IPv6 Multicast Option”

Note that length of IP address is 32 bits for IPv4 options and 128 bits for IPv6 Options

Options related to SD implementation itself

Type = 0x24 => option = “IPv4 SD Endpoint Option”
Type = 0x26 => option = “IPv6 SD Endpoint Option”

SOME/IP Protocol – Message



Request ID (Client ID / Session ID) [32 bit]

Protocol Version [8 bit] =0x01 | Interface Version [8 bit] =0x01 | Message Type [8 bit] =0x02 | Return Code [=0x00

Flags [8 bit] = 0x80 | Reserved [8 bit =0x00]

Length of Entries Array in Bytes [32 bit] =0x0000 0010

Type =0x01 (Offer) | Index 1st options =0 | Index 2nd options =0 | # of opt 1 =2 | # o =0

Service ID =0xFFFE | Instance ID =0x0001

Major Version =0x01 | TTL =300 (offer is valid for 300 seconds)

Minor Version =0x00000032

Length of Options Array in Bytes = 0x0000 0028

Length =0x0009 | Type =0x04 (IPv4 Endpoint) | Reserved =0x00

IPv4-Address = 192.168.0.1

Reserved =0x00 | L4-Proto =0x11 (UDP) | Port Number =0xD903 (=55555)

Length =0x0019 | Type =0x01 (Config) | Reserved =0x00

[0x16]otherserv=internaldiag[0]

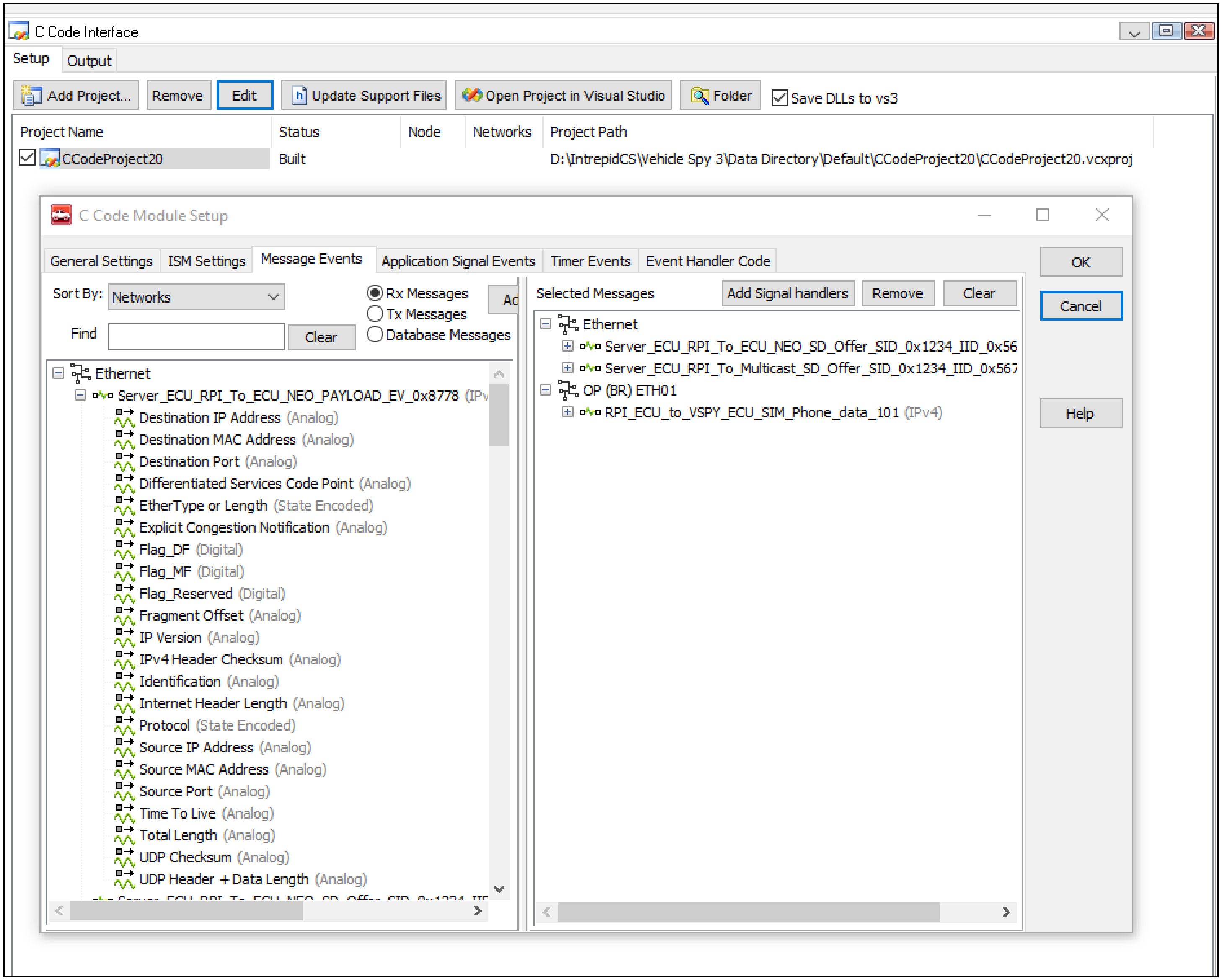
Raw Video Stream
Real-Time Transport Control Protocol
Real-Time Transport Protocol
SDI Video Stream
SOME/IP
Client ID someip.requestid.clientid
Interface Version someip.interfacev
Length someip.length
Message ID someip.messageid.id
Message Type someip.messagetype
Method ID someip.messageid.meth
Protocol Version someip.protocolve
Request ID someip.requestid.id
Reserved someip.messageid.reser
Return Code someip.returncode
Service ID someip.messageid.serv
Session ID someip.requestid.sessic

Taken as it is from Autosar specs

SOME/IP VehicleSpy Advanced Scripting Support

CCIF = C Code Interface

FB = Function Blocks



☑ Save DLLs to vs3

Project Name	Status	Node	Networks	Project Path
☑ CCodeProject20	Built			D:\IntrepidCS\Vehicle Spy 3\Data Directory\Default\CCodeProject20\CCodeProject20.vcxproj

Sort By: Networks

● Rx Messages
○ Tx Messages
○ Database Messages

```
⊟ Ethernet
  ⊟ Server_ECU_RPI_To_ECU_NEO_PAYLOAD_EV_0x8778 (IPv
      Destination IP Address (Analog)
      Destination MAC Address (Analog)
      Destination Port (Analog)
      Differentiated Services Code Point (Analog)
      EtherType or Length (State Encoded)
      Explicit Congestion Notification (Analog)
      Flag_DF (Digital)
      Flag_MF (Digital)
      Flag_Reserved (Digital)
      Fragment Offset (Analog)
      IP Version (Analog)
      IPv4 Header Checksum (Analog)
      Identification (Analog)
      Internet Header Length (Analog)
      Protocol (State Encoded)
      Source IP Address (Analog)
      Source MAC Address (Analog)
      Source Port (Analog)
      Time To Live (Analog)
      Total Length (Analog)
      UDP Checksum (Analog)
      UDP Header + Data Length (Analog)
```

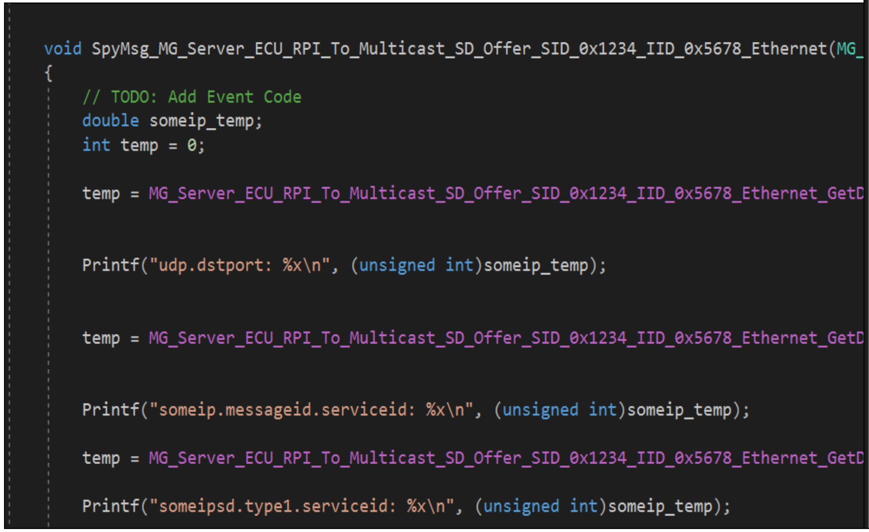
Selected Messages

```
⊟ Ethernet
  ⊞ Server_ECU_RPI_To_ECU_NEO_SD_Offer_SID_0x1234_IID_0x56
  ⊞ Server_ECU_RPI_To_Multicast_SD_Offer_SID_0x1234_IID_0x567
⊟ OP (BR) ETH01
  ⊞ RPI_ECU_to_VSPY_ECU_SIM_Phone_data_101 (IPv4)
```

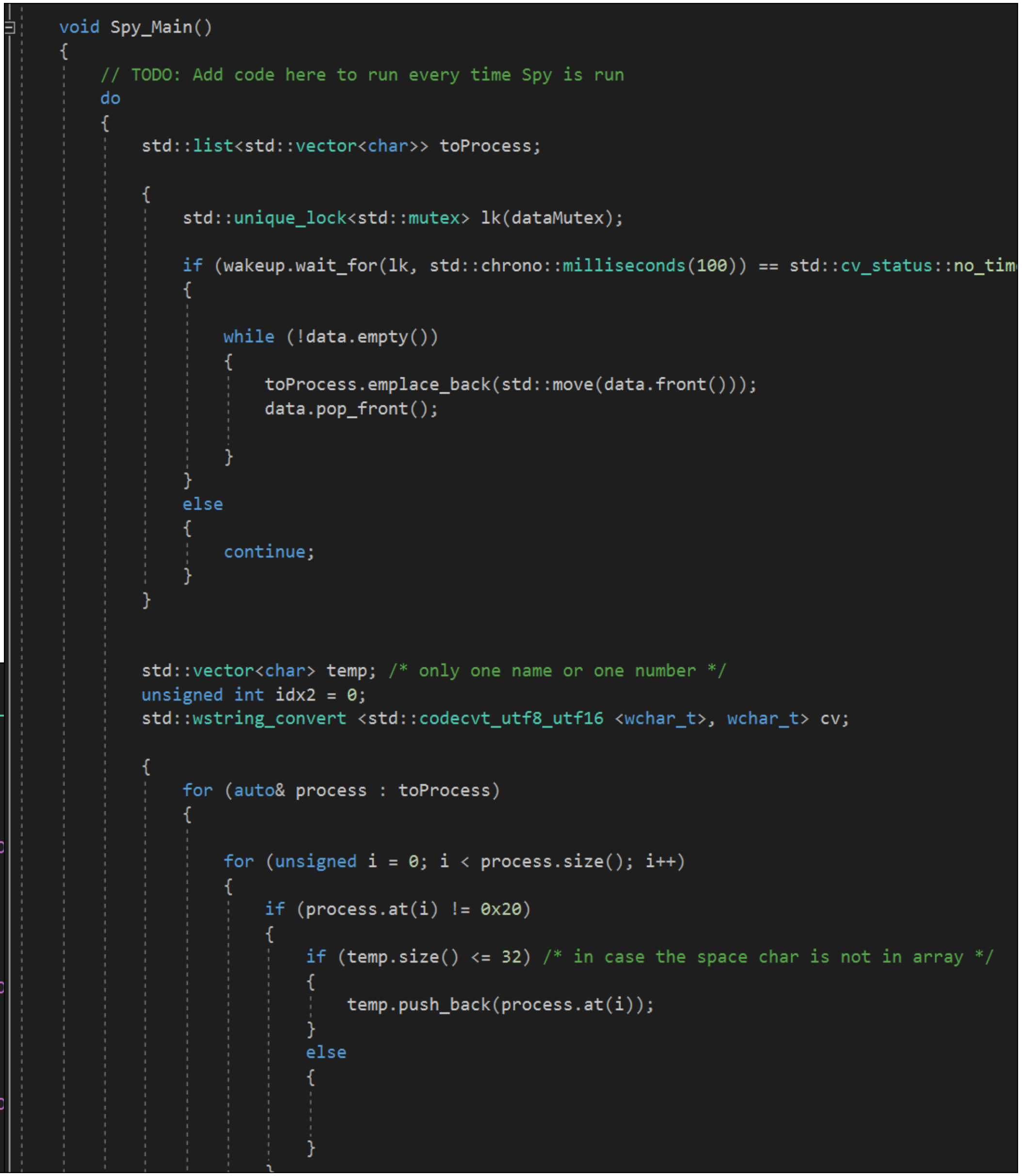
SOME/IP
VehicleSpy Advanced Scripting Support

CCIF = C Code Interface

FB = Function Blocks

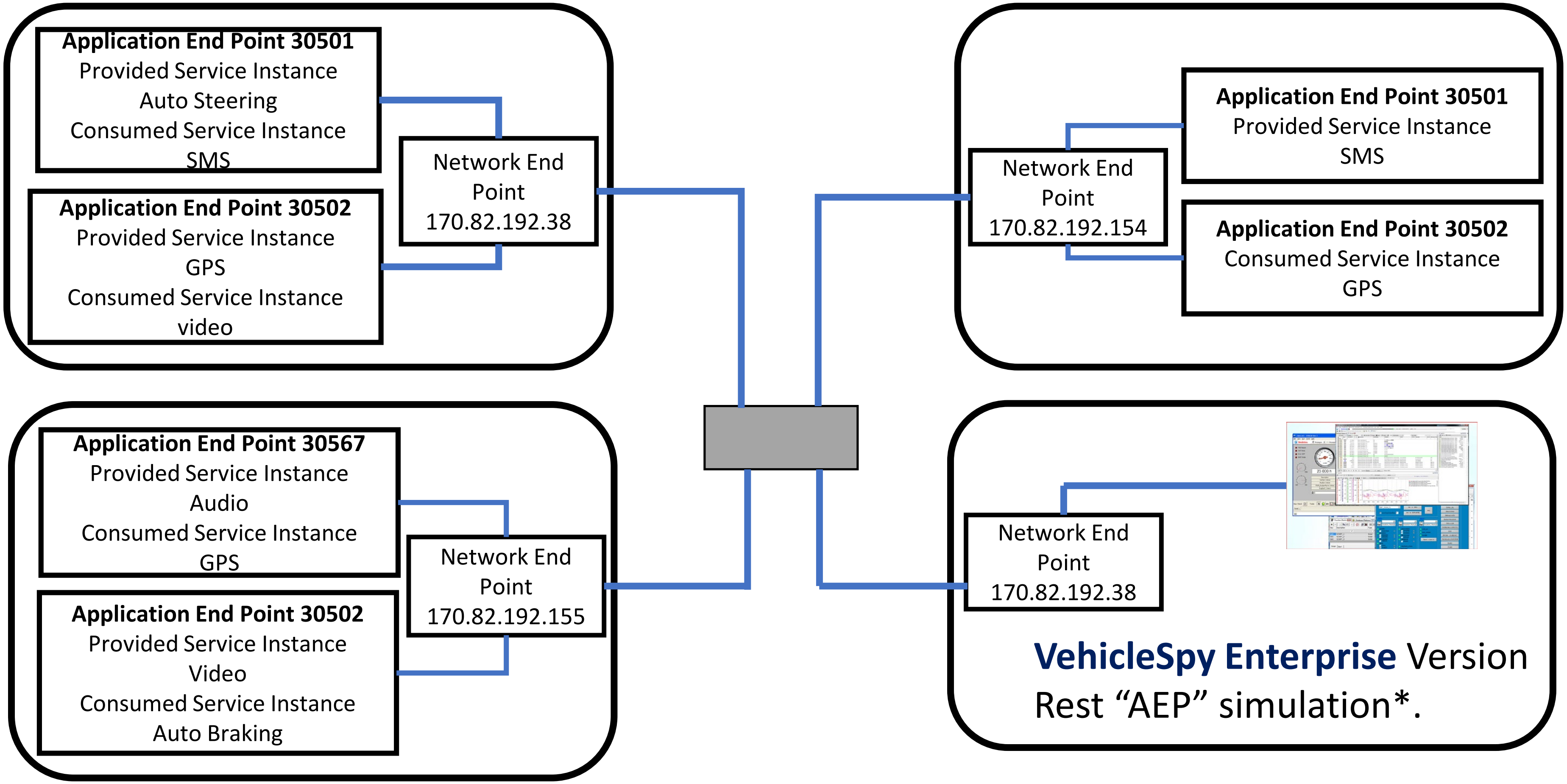


```
void SpyMsg_MG_Server_ECU_RPI_To_Multicast_SD_Offer_SID_0x1234_IID_0x5678_Ethernet(MG_
{
    // TODO: Add Event Code
    double someip_temp;
    int temp = 0;

    temp = MG_Server_ECU_RPI_To_Multicast_SD_Offer_SID_0x1234_IID_0x5678_Ethernet_GetD


    Printf("udp.dstport: %x\n", (unsigned int)someip_temp);


    temp = MG_Server_ECU_RPI_To_Multicast_SD_Offer_SID_0x1234_IID_0x5678_Ethernet_GetD


    Printf("someip.messageid.serviceid: %x\n", (unsigned int)someip_temp);

    temp = MG_Server_ECU_RPI_To_Multicast_SD_Offer_SID_0x1234_IID_0x5678_Ethernet_GetD

    Printf("someipsd.type1.serviceid: %x\n", (unsigned int)someip_temp);
```



```
void Spy_Main()
{
    // TODO: Add code here to run every time Spy is run
    do
    {
        std::list<std::vector<char>> toProcess;

        {
            std::unique_lock<std::mutex> lk(dataMutex);

            if (wakeup.wait_for(lk, std::chrono::milliseconds(100)) == std::cv_status::no_tim
            {

                while (!data.empty())
                {
                    toProcess.emplace_back(std::move(data.front()));
                    data.pop_front();

                }
            }
            else
            {
                continue;
            }
        }


        std::vector<char> temp; /* only one name or one number */
        unsigned int idx2 = 0;
        std::wstring_convert <std::codecvt_utf8_utf16 <wchar_t>, wchar_t> cv;

        {
            for (auto& process : toProcess)
            {

                for (unsigned i = 0; i < process.size(); i++)
                {
                    if (process.at(i) != 0x20)
                    {
                        if (temp.size() <= 32) /* in case the space char is not in array */
                        {
                            temp.push_back(process.at(i));
                        }
                        else
                        {

                        }
                    }
```

VehicleSpy Enterprise for Ethernet and SOME/IP



Application End Point 30501
Provided Service Instance
Auto Steering
Consumed Service Instance
SMS

Application End Point 30502
Provided Service Instance
GPS
Consumed Service Instance
video

Network End Point
170.82.192.38

Application End Point 30567
Provided Service Instance
Audio
Consumed Service Instance
GPS

Application End Point 30502
Provided Service Instance
Video
Consumed Service Instance
Auto Braking

Network End Point
170.82.192.155

Network End Point
170.82.192.154

Application End Point 30501
Provided Service Instance
SMS

Application End Point 30502
Consumed Service Instance
GPS

Network End Point
170.82.192.38

VehicleSpy Enterprise Version Rest “AEP” simulation*.

Intrepid Tools Support for SOME/IP

- **Intrepid SW VehicleSpy – Powerful bus analysis**

- **Intrepid SW EEA COM – Create, Edit, View SOME/IP Ethernet clusters**

- **Intrepid HW RAG GALAXY & RAD STAR 2 – Multi 100BASE T1, 1000BASE T1**

SOME/IP Updates and latest

AUTOSAR Classical ARXML Standard 4.3.0

ASAM MCD NET
FIBEX Standard 4.1.1

# Released 2016-11-30 in AUTOSAR 4.3.0 – SOME/IP Transport Protocol

# Segment SOME/IP packets which do not fit into 1 single UDP packet

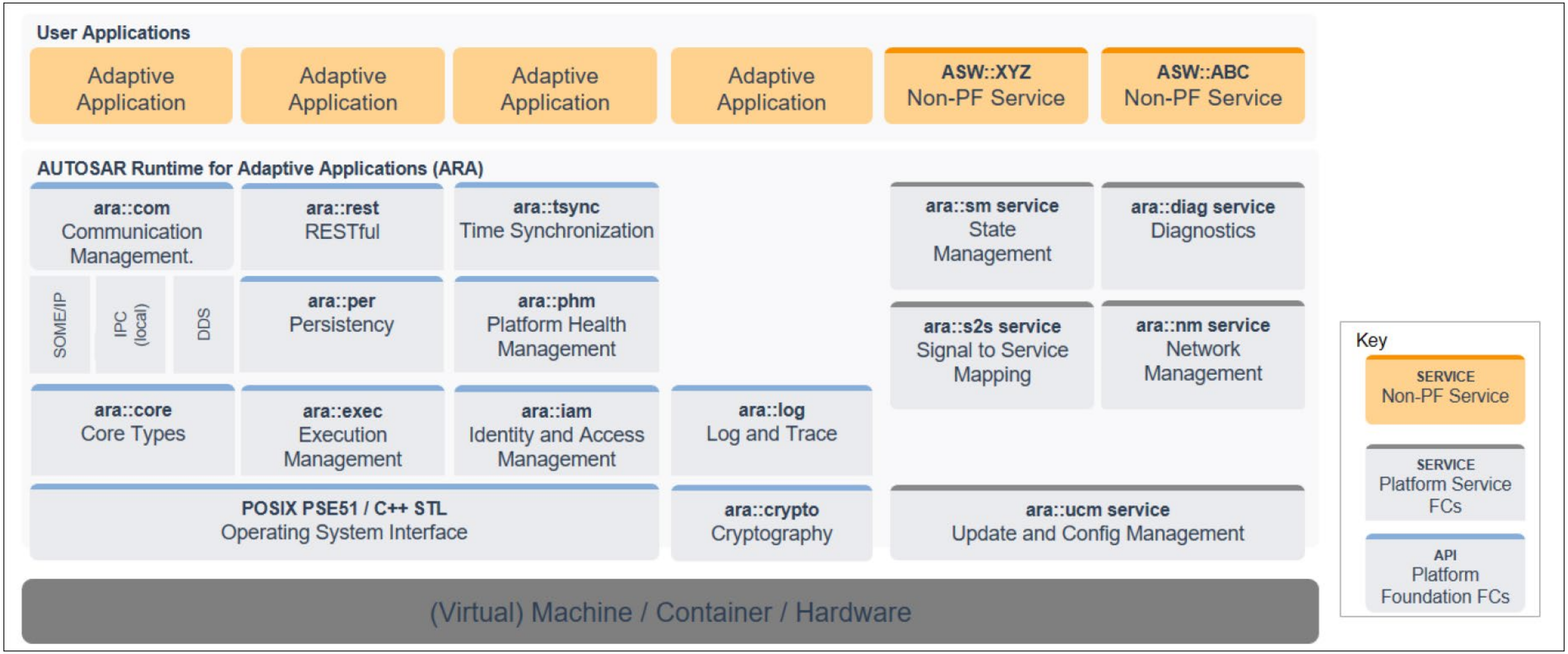
Adaptive AUTOSAR

ARA::COM API

SOME/IP as integral part

Adaptive Autosar

Ara::com – CM – Communications management



User Applications
Adaptive Application
Adaptive Application
Adaptive Application
Adaptive Application
ASW::XYZ Non-PF Service
ASW::ABC Non-PF Service
AUTOSAR Runtime for Adaptive Applications (ARA)
ara::com Communication Management.
ara::rest RESTful
ara::tsync Time Synchronization
ara::sm service State Management
ara::diag service Diagnostics
SOME/IP
IPC (local)
DDS
ara::per Persistency
ara::phm Platform Health Management
ara::s2s service Signal to Service Mapping
ara::nm service Network Management
ara::core Core Types
ara::exec Execution Management
ara::iam Identity and Access Management
ara::log Log and Trace
POSIX PSE51 / C++ STL Operating System Interface
ara::crypto Cryptography
ara::ucm service Update and Config Management
(Virtual) Machine / Container / Hardware
Key
SERVICE Non-PF Service
SERVICE Platform Service FCs
API Platform Foundation FCs

Ref: autosar.org

Thanks!

Hope you enjoyed learning SOME/IP concepts