CANOE ETH环境搭建

Davina Lu Senior SW Verification Engineer

6/9/2025

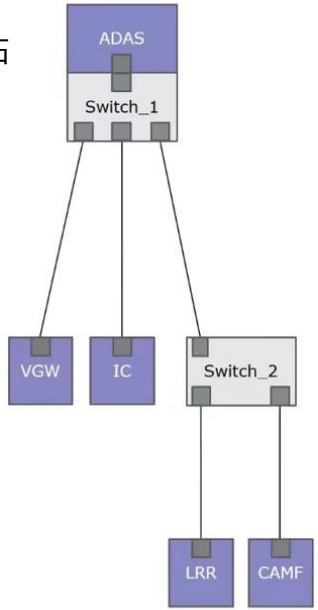
Content:

- Basic Concept
- Network-based Mode Overview
- Ethernet Hardware Configuration
- Canoe Configuration

Switched Network

Ethernet: Local Area Network (LAN) but no Bus System

以太网结构：
点对点网络拓扑结构

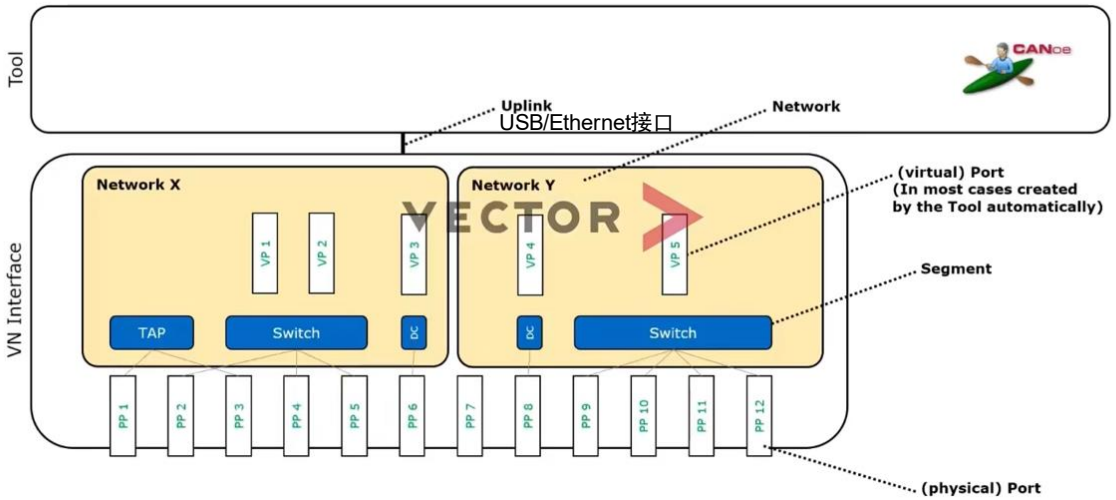


Network: Nodes, Links, Switches

- **Node** 收发报文，真实的/虚拟的
- **Link** 连接两个节点的phy
- **Switch** 两个以上的节点，使用switch进行报文转发

Network-based mode

Basic Concept



Port:

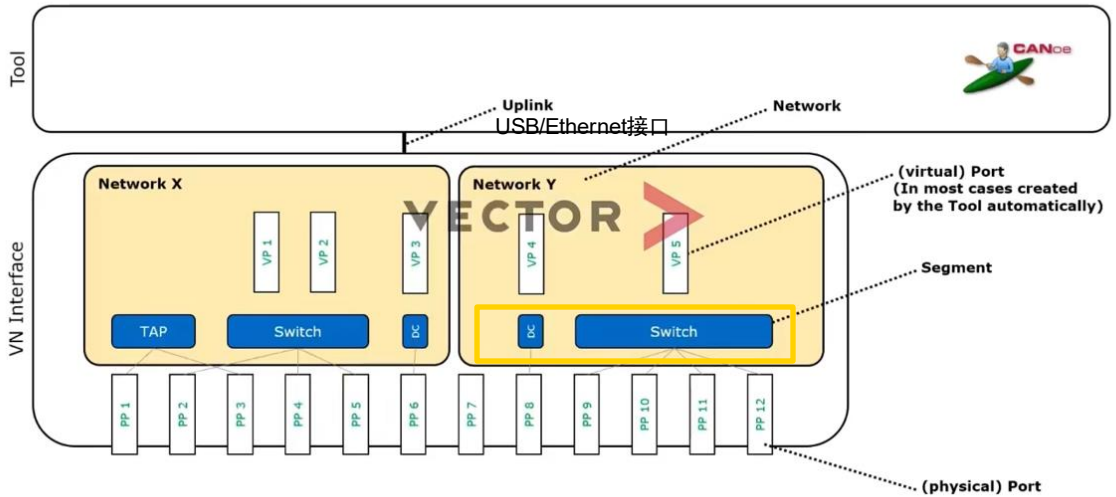
Physical port(PP):

- 连接到真实设备。
- 配置channel mode(Baud rate, master/slave)

Virtual Port(VP):

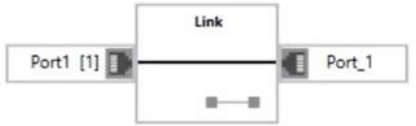
- Canoe仿真端口
- Canoe根据节点自动添加

Basic Concept

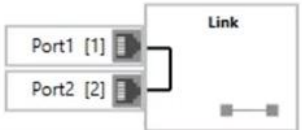


Segment: Link和Switch的统称
负责耦合不同的PP和VP

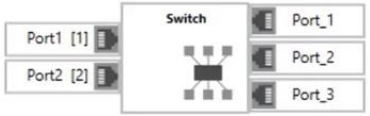
- **Link Segment**
  - Direct Connection
    - Connect one physical port and one virtual port



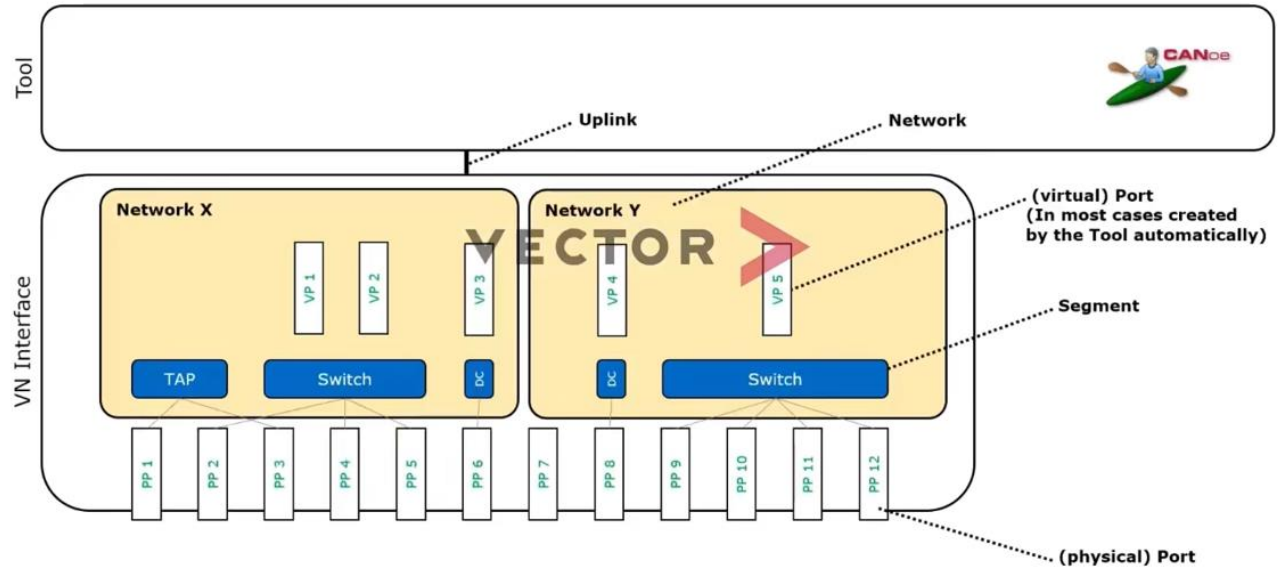
  - TAP
    - Connection of two physical ports
    - Very low and constant latency



- **Switch Segment**
  - Provides the basic functions of a layer-2 switch
    - Supports VLAN settings



Basic Concept



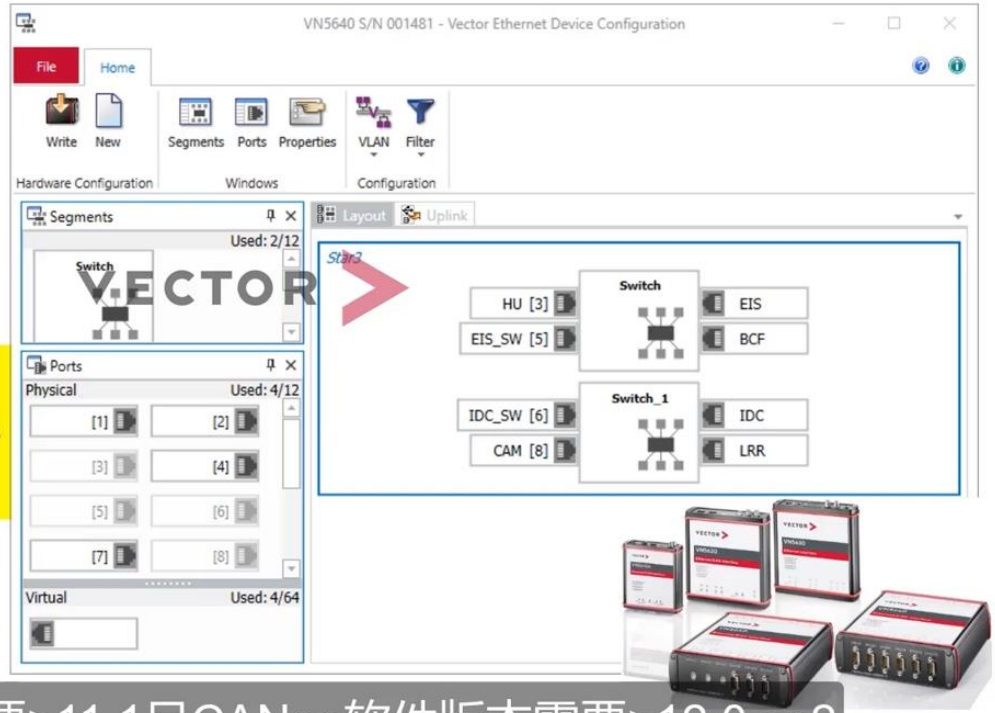
- ✓ 一个网络接口设备(VN5XXX)可以包含多个network.
- ✓ 一个network可以包含多个segment.
- ✓ 其中switch Segment又可以包含多个PP(至少一个)/VP

Network-based Mode Overview

Overview

- **Network-based Mode**
  - New software/firmware concept for the Vector Ethernet Network Interfaces
  - Configuration of the hardware is independent of CANoe

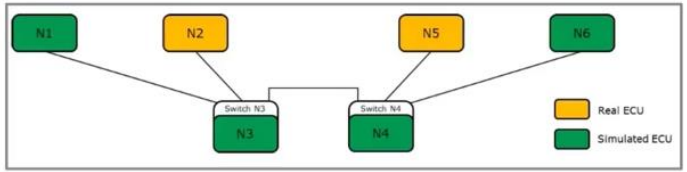
- **Precondition:**
  - Network Hardware Driver Version > **11.1**
  - CANoe Version >**12.0 sp2**



硬件驱动版本需要>11.1且CANoe软件版本需要>12.0 sp2

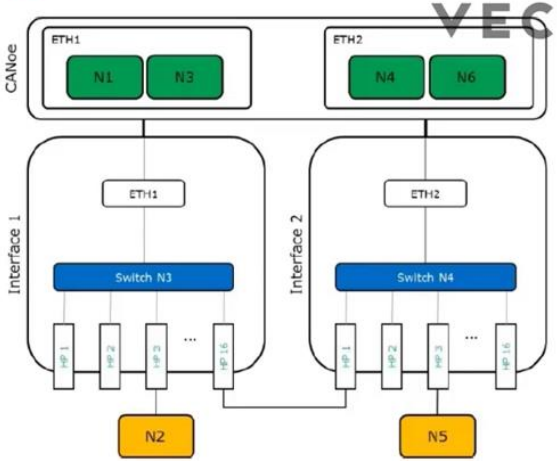
Network-based VS Channel-based Mode

Overview

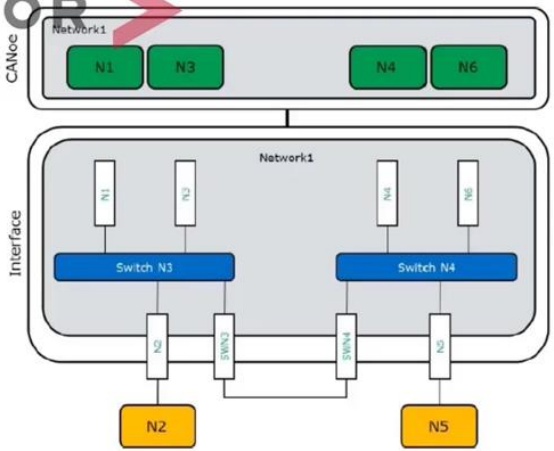


Q: 中间连线是物理还是设置上的？
定义两个不同的PP口连接到不同的switch，物理将两个PP口连接起来。

▶ Channel-based Mode



▶ Network-based Mode



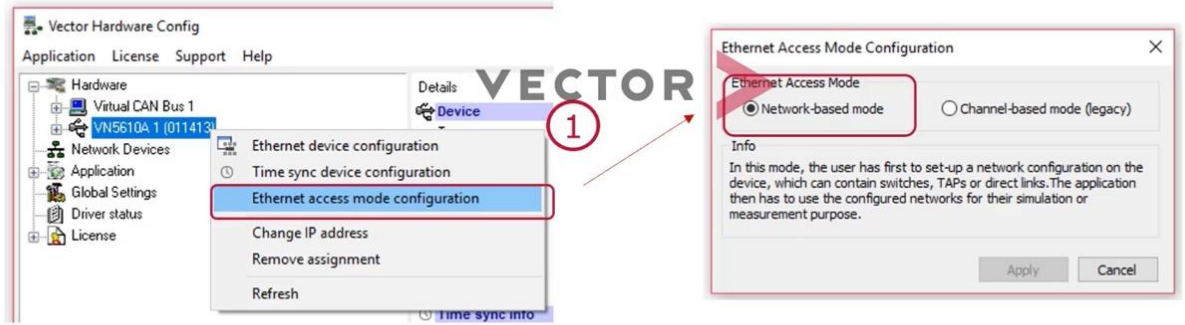
在channel-based模式下，每个VN设置只能支持一个switch segment

Ethernet Hardware Configuration

Ethernet Hardware Configuration

Network-based Mode

- **Switch to Network-based Mode**
  - The Network-based Mode can be activated separately for each network interface (context menu >> Ethernet access mode configuration)

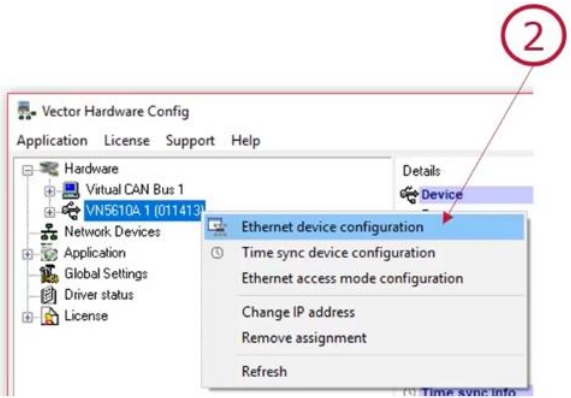


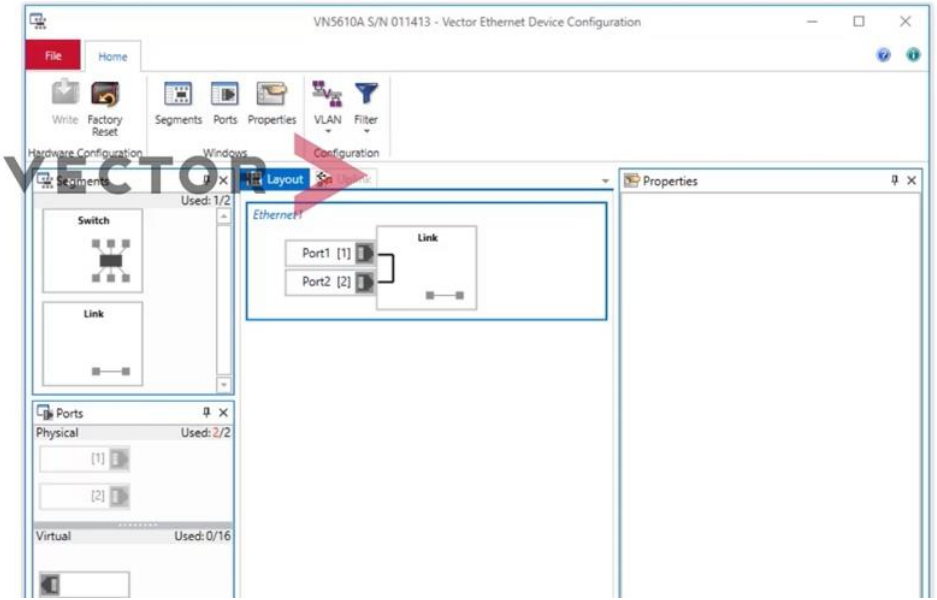
5610A/5640： 支持原有的channel-based mode，所以需要切换。

Ethernet Hardware Configuration

Ethernet Device Configuration Window

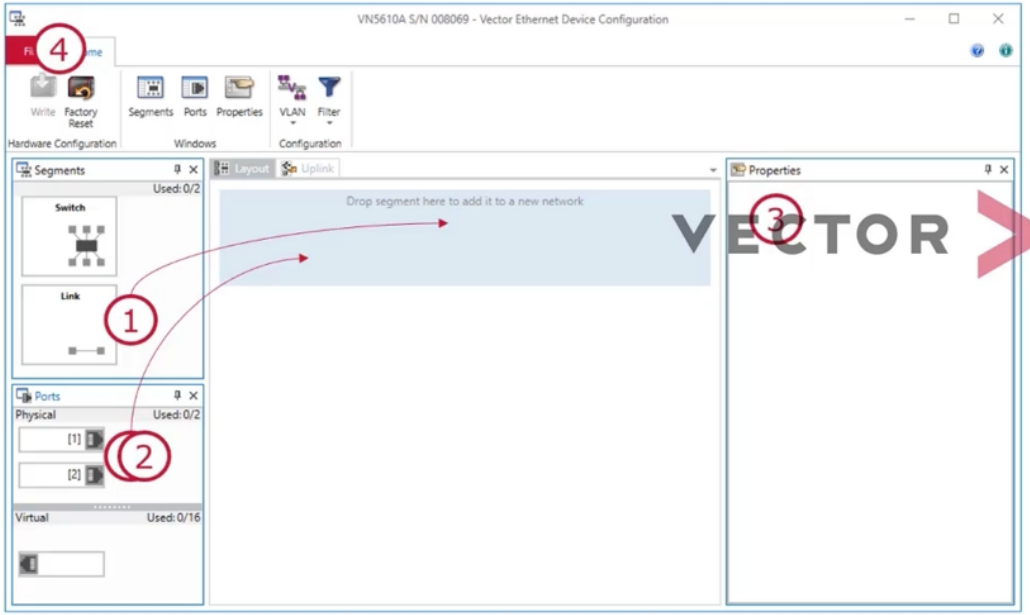
- Each device is configured separately using the new uniform "Ethernet Device Configuration"
  - Opened from the Vector Hardware Configuration
    (context menu >> Ethernet Device Configuration)





Ethernet Hardware Configuration

Ethernet Device Configuration Sequence



① Select Segments and name network

② Assign Physical Port

③ Set port properties

④ Click "Write" to save the configuration

...

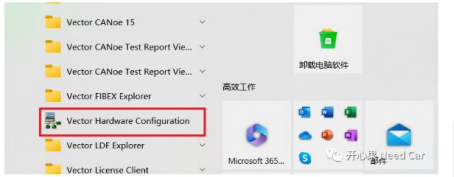
- The configuration can be saved to a XML file
- The configuration can be loaded from a XML file

2、CANoe配置

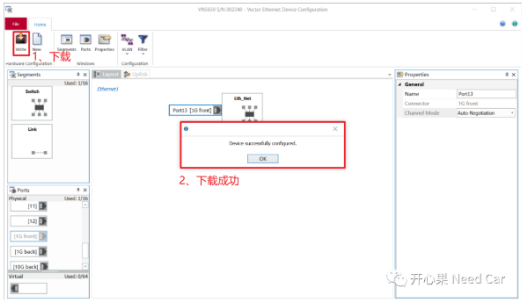
完成物理连接以后，进行CANoe配置，以便于以太网数据的监控和仿真。

（一）VN5650硬件网络（Network）配置

安装CANoe 15.0软件以及VN5650驱动以后，在开始菜单中可以找到"Vector Hardware Configuration"，如下所示：

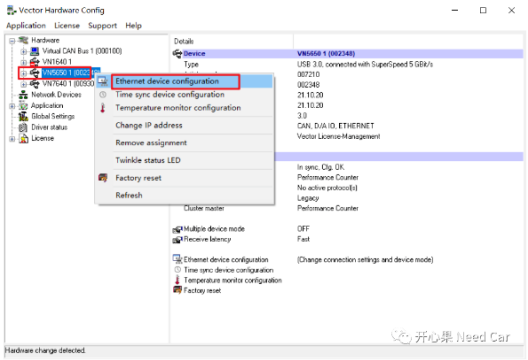


配置好网络以后，将配置好的网络下载到VN5650设备中，成功配置网络到VN5650后，提示如下：

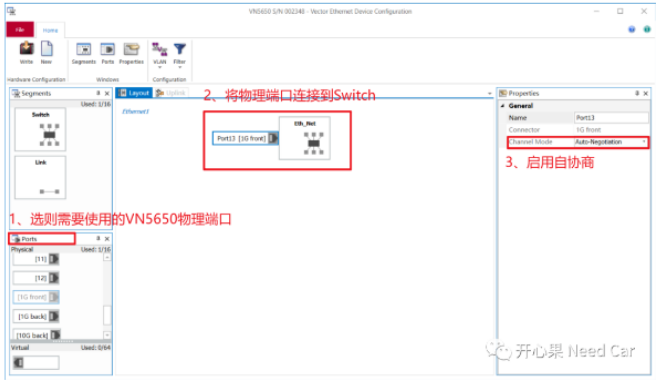


提示：配置好的Network信息可以保存为*.XML文件，方便复用。

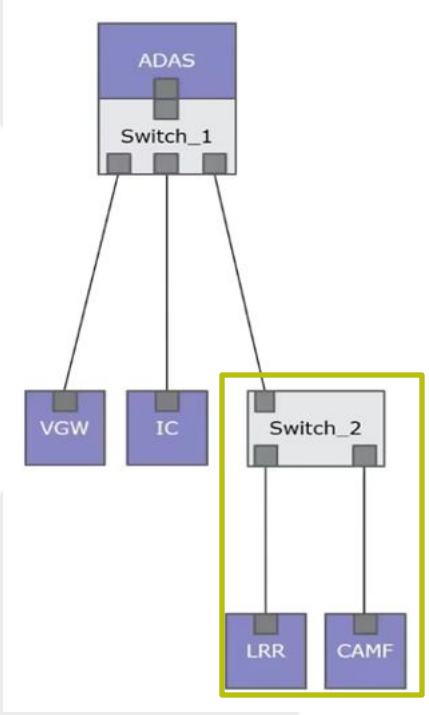
在Vector Hardware Config的配置界面选择VN5650，右击->选择"Ethernet device Configuration"，如下所示：

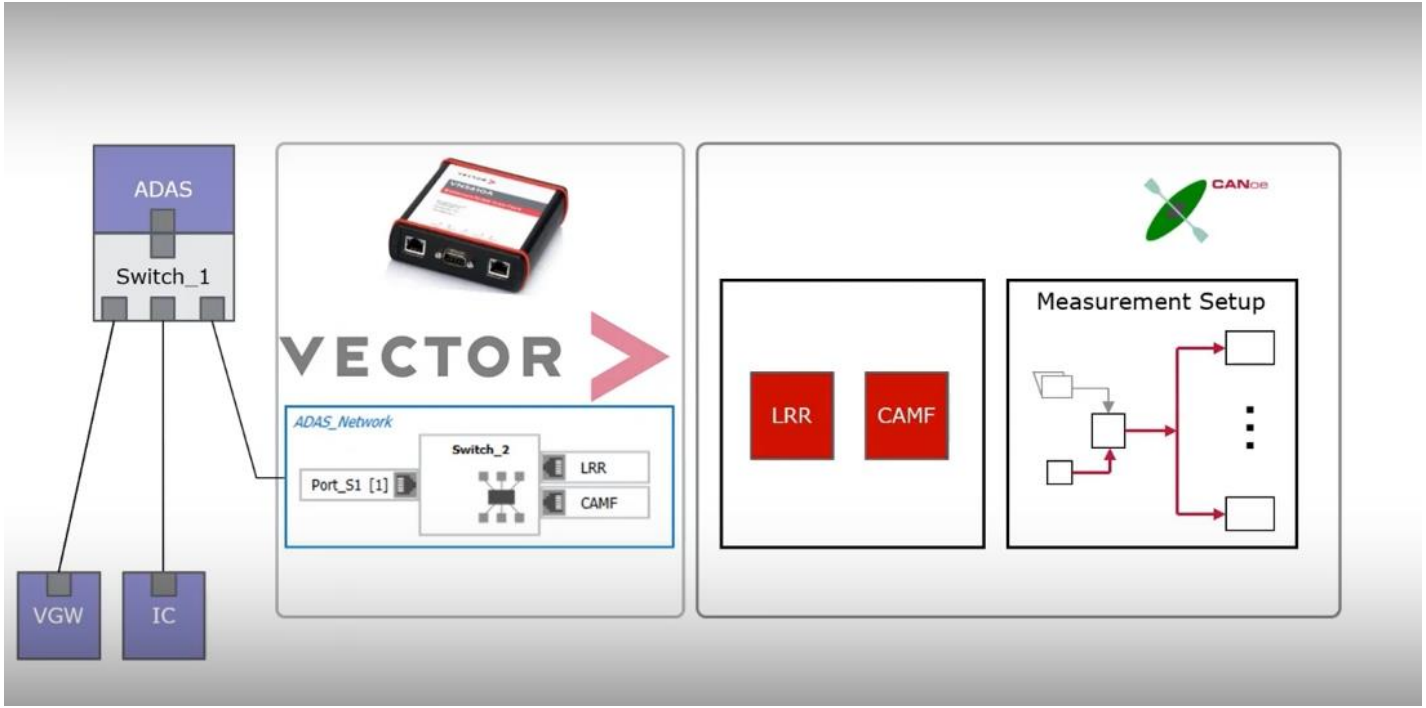


选择Port口，此处选择要使用的VN5650物理端口（PP），本文仅使用Port13[1G front]，即：目标DUT与VN5650通过RJ45连接，至于Ethernet的通信速率，使用默认的自协商（Auto-Negotiation）方式即可，配置如下：



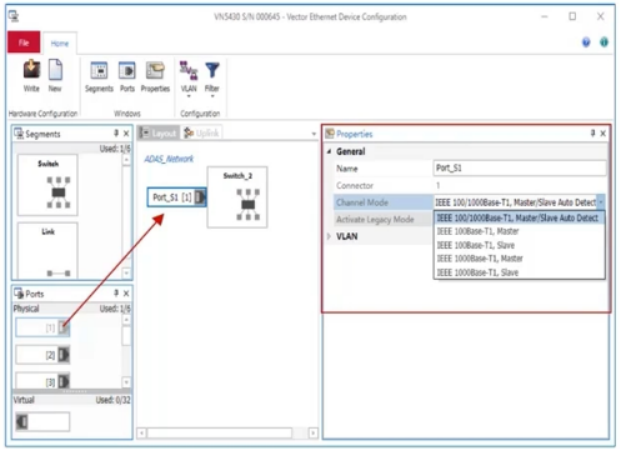
Use Case

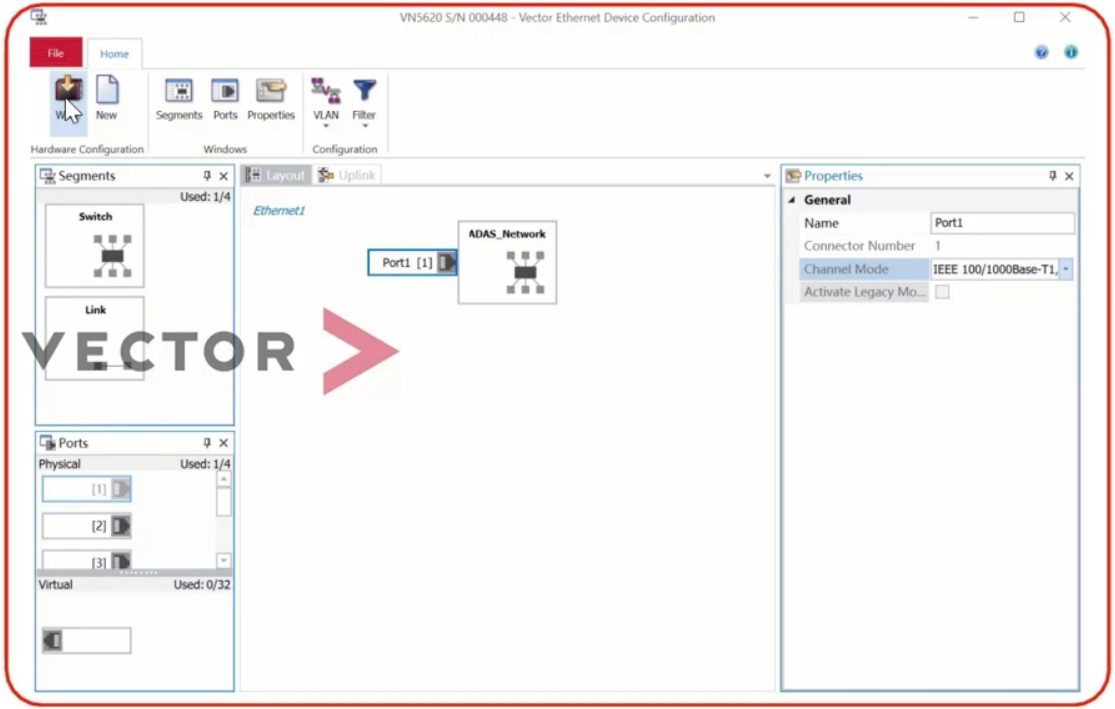




Use Case

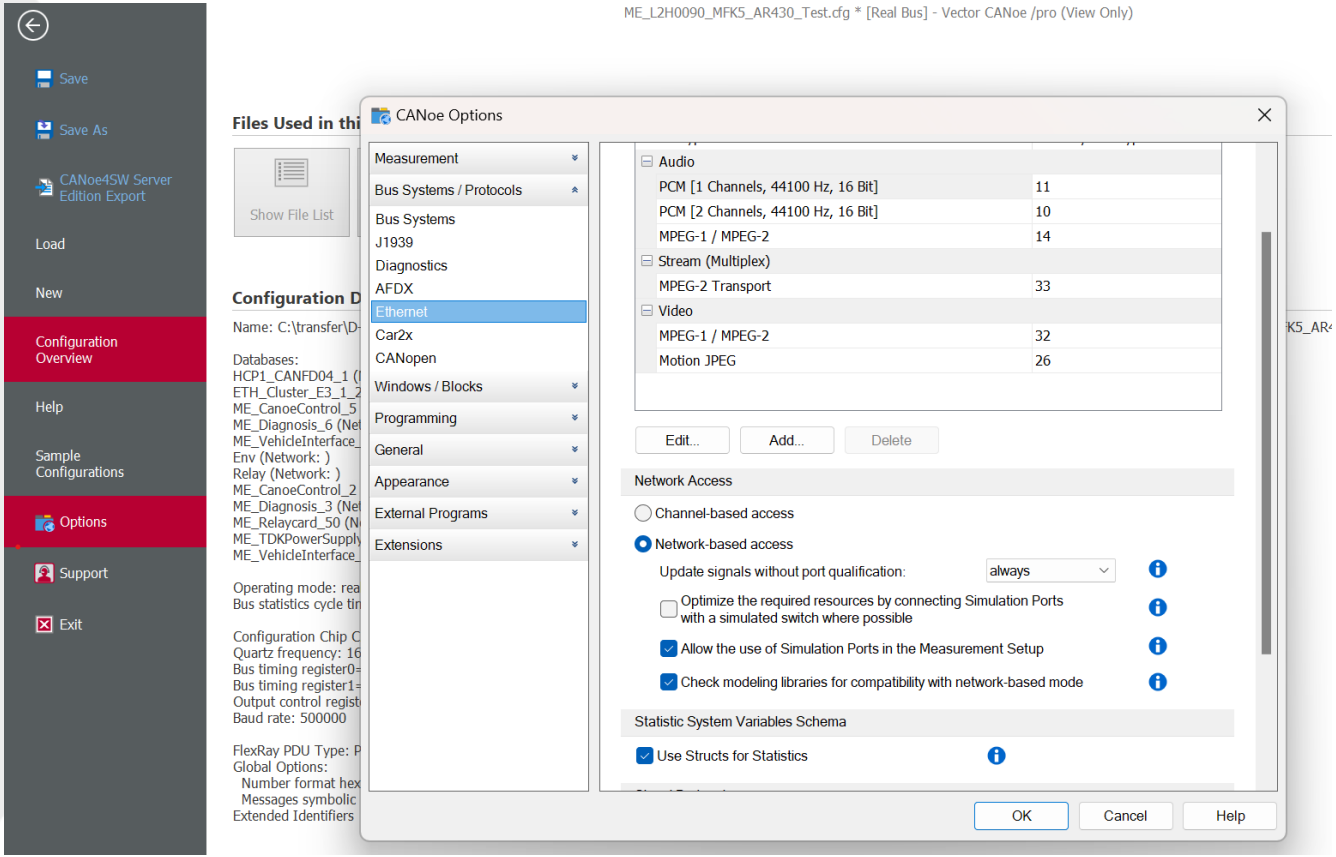
Switch for Multiple Senders



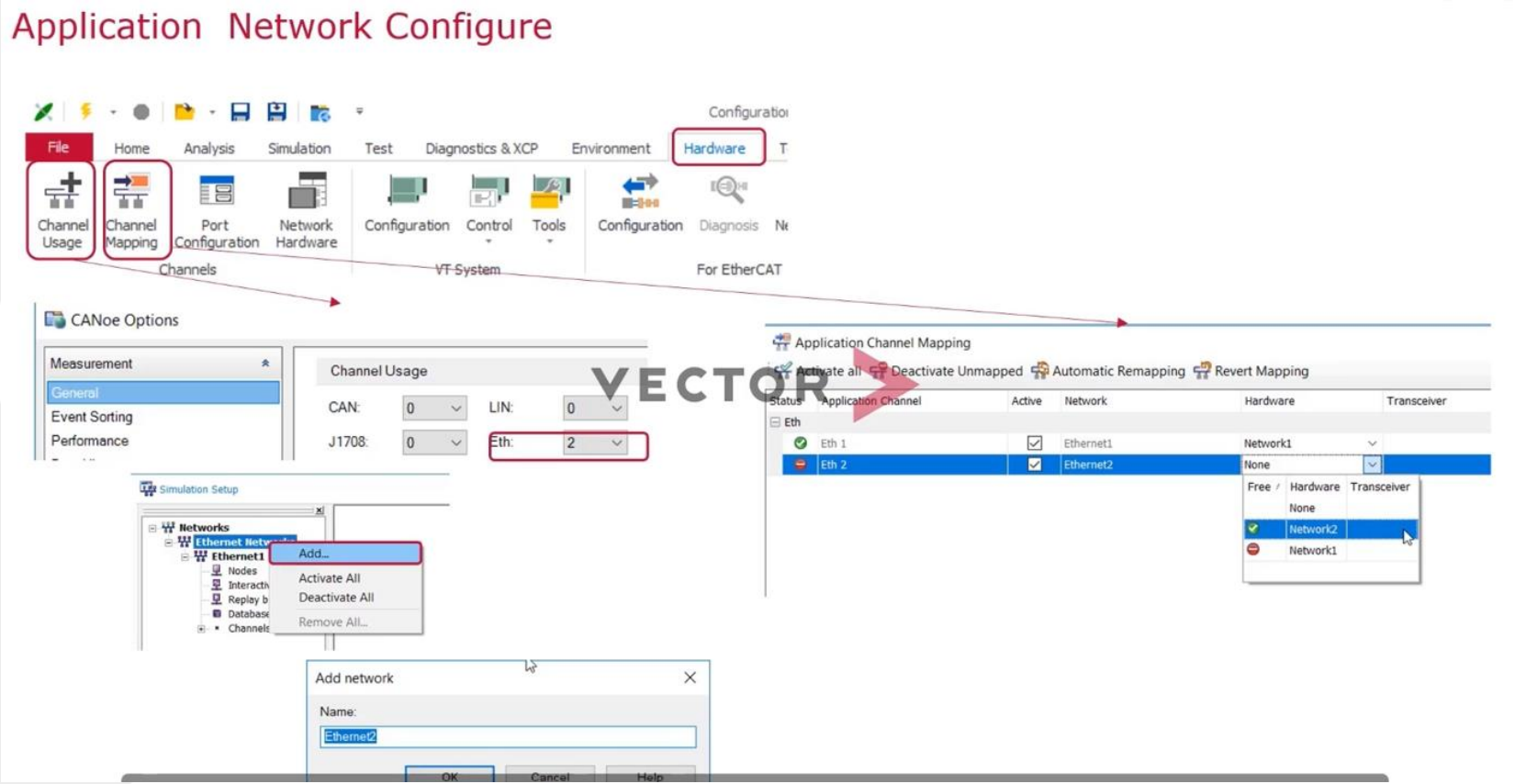


Canoe Configuration

Application Network Configure



Application Network Configuration

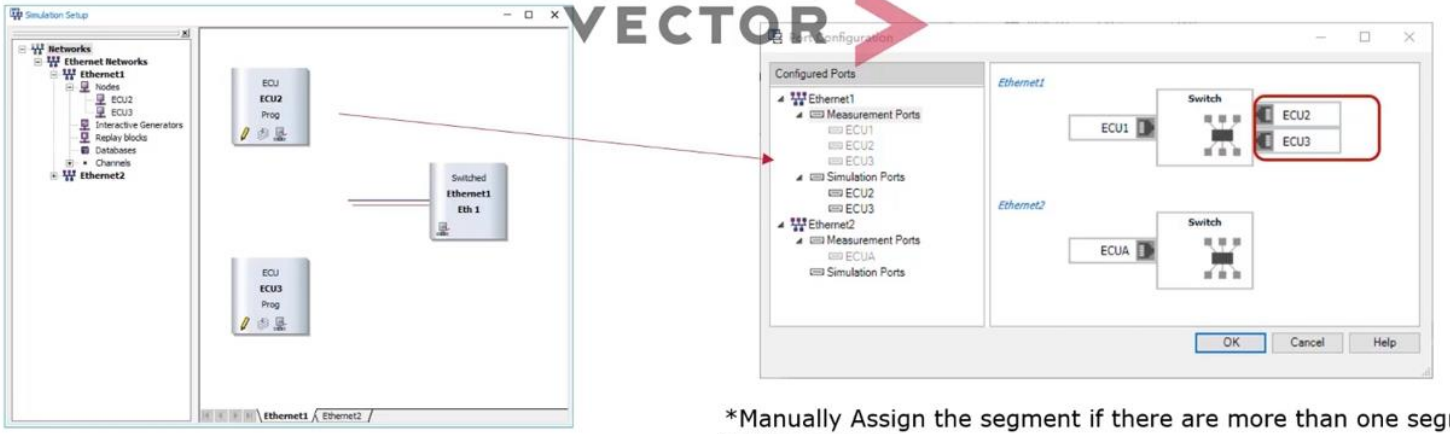


Port Configuration

Port Configuration

- **Simulation Ports:**
  - Used for simulation access (read and write access)
  - Represents a virtual port defined in the Ethernet Device Configuration in CANoe

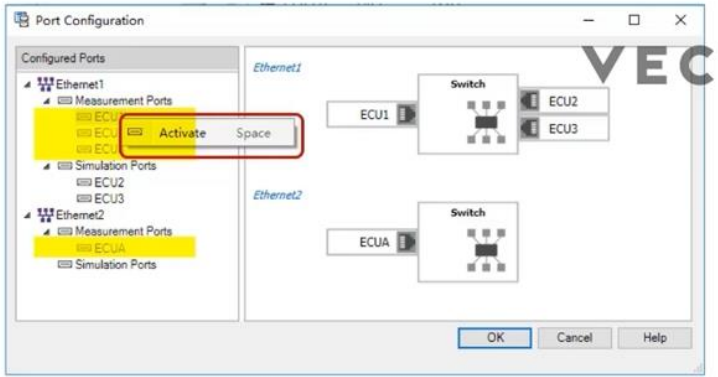
- A simulation port is created automatically by adding a node to the network in the simulation setup



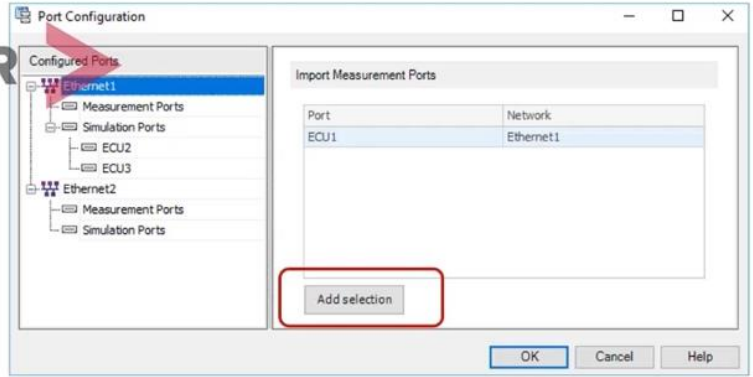
*Manually Assign the segment if there are more than one segments in the network

Port Configuration

- **Measurement Ports:**
  - All measurement ports whose data you want to measure are displayed here.
  - Import physical ports as measurement ports
  - Or manually add the measurement port by adding the name of the matching physical port



CANoe14.0

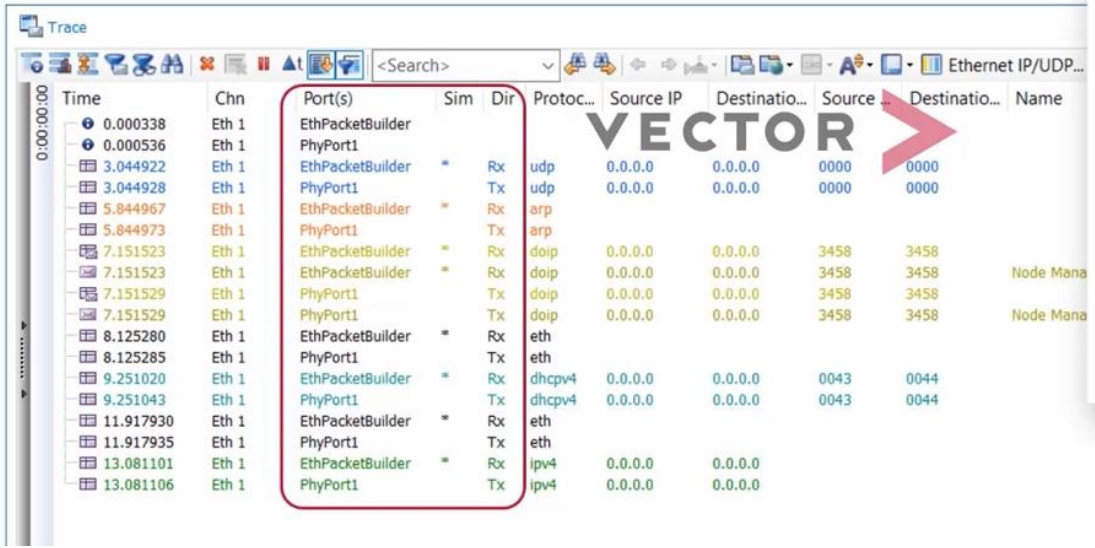


CANoe13.0

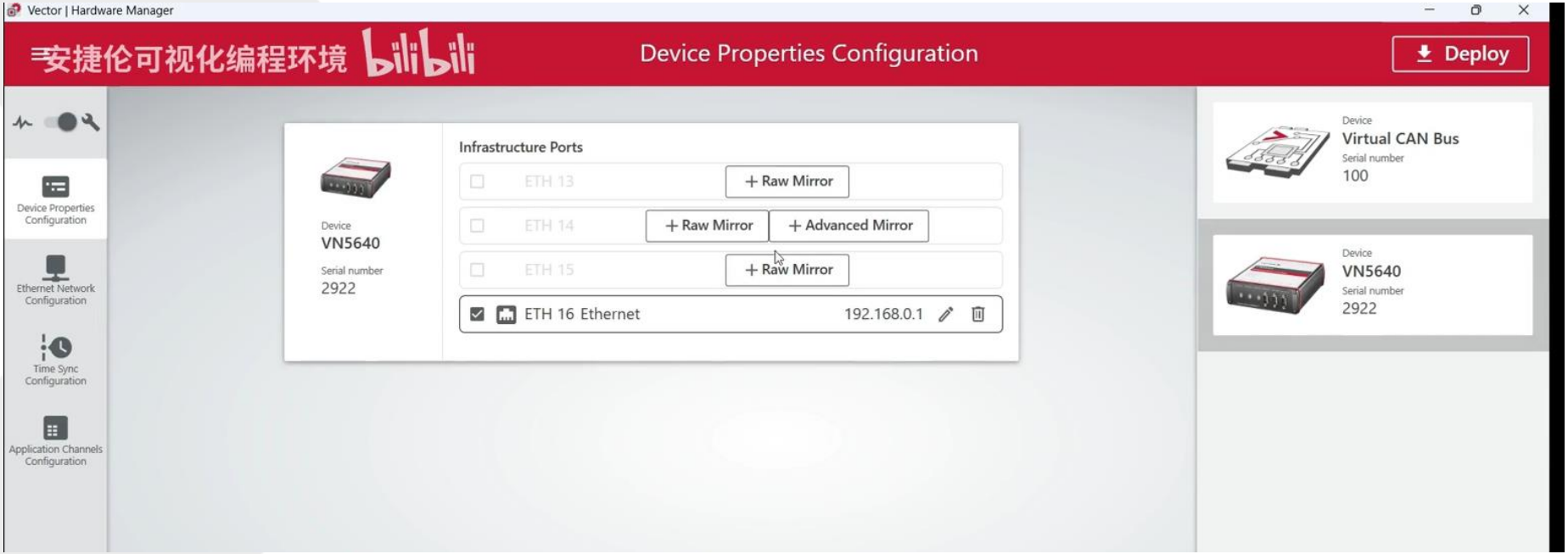
CANoe Trace

Obeserve in CANoe Trace

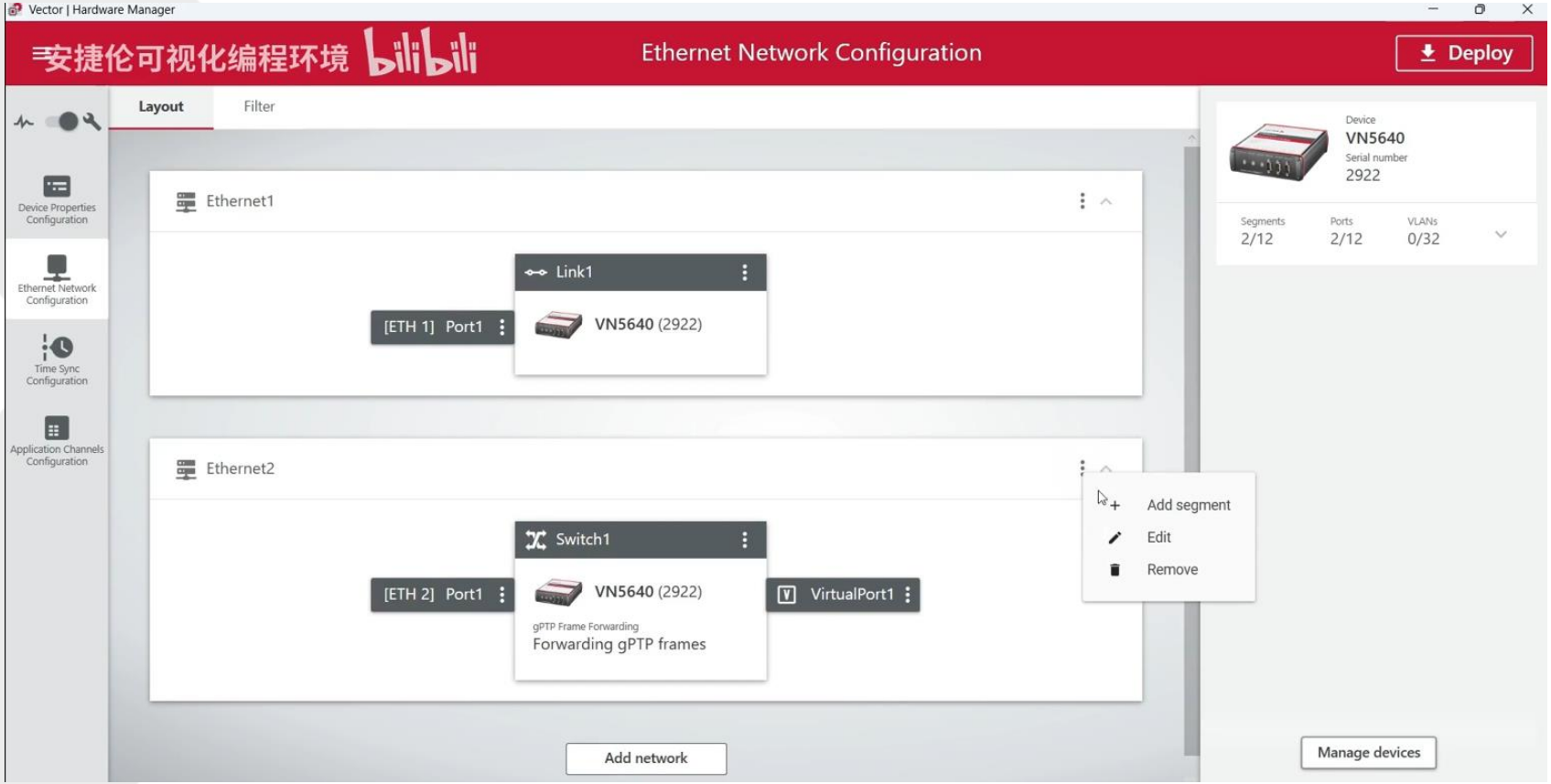
- The TX/RX direction is based on the hardware interface perspective.



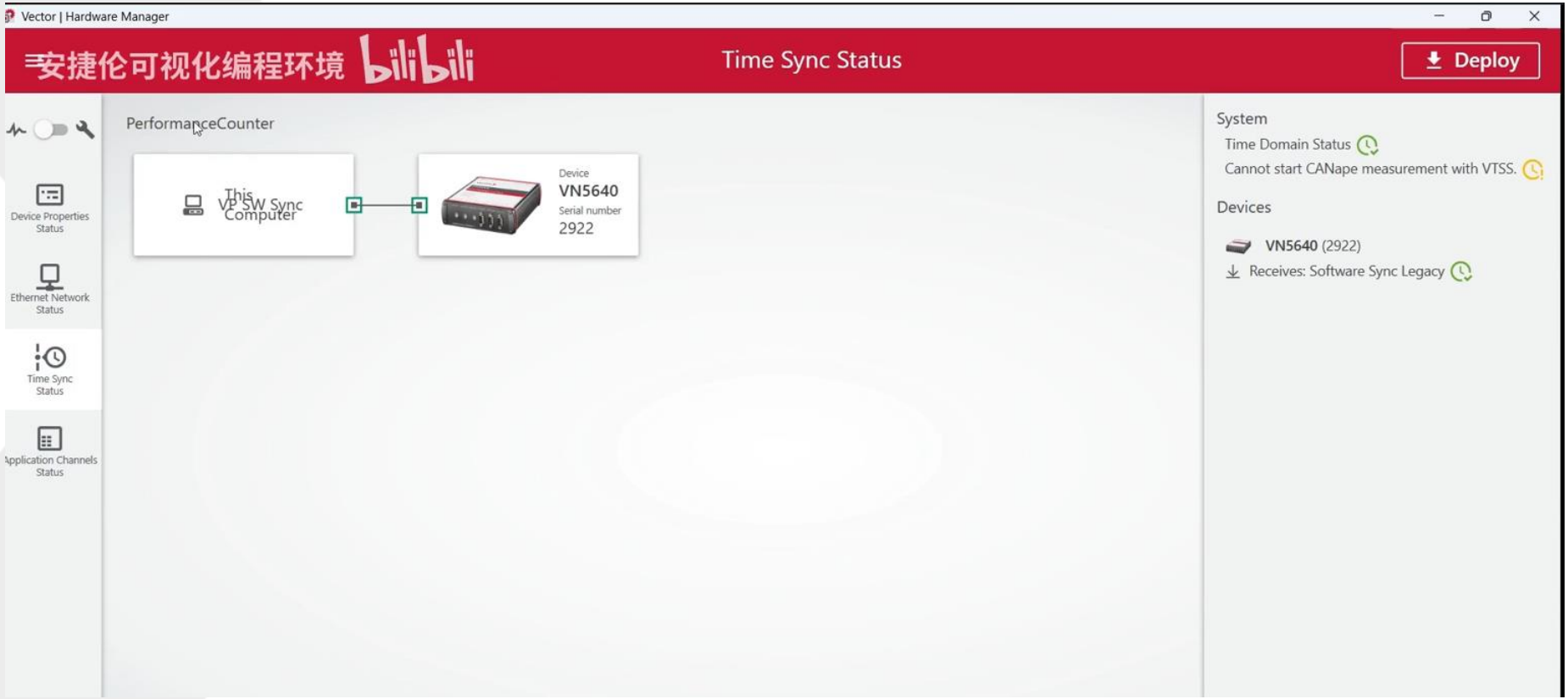
CANoe 16+

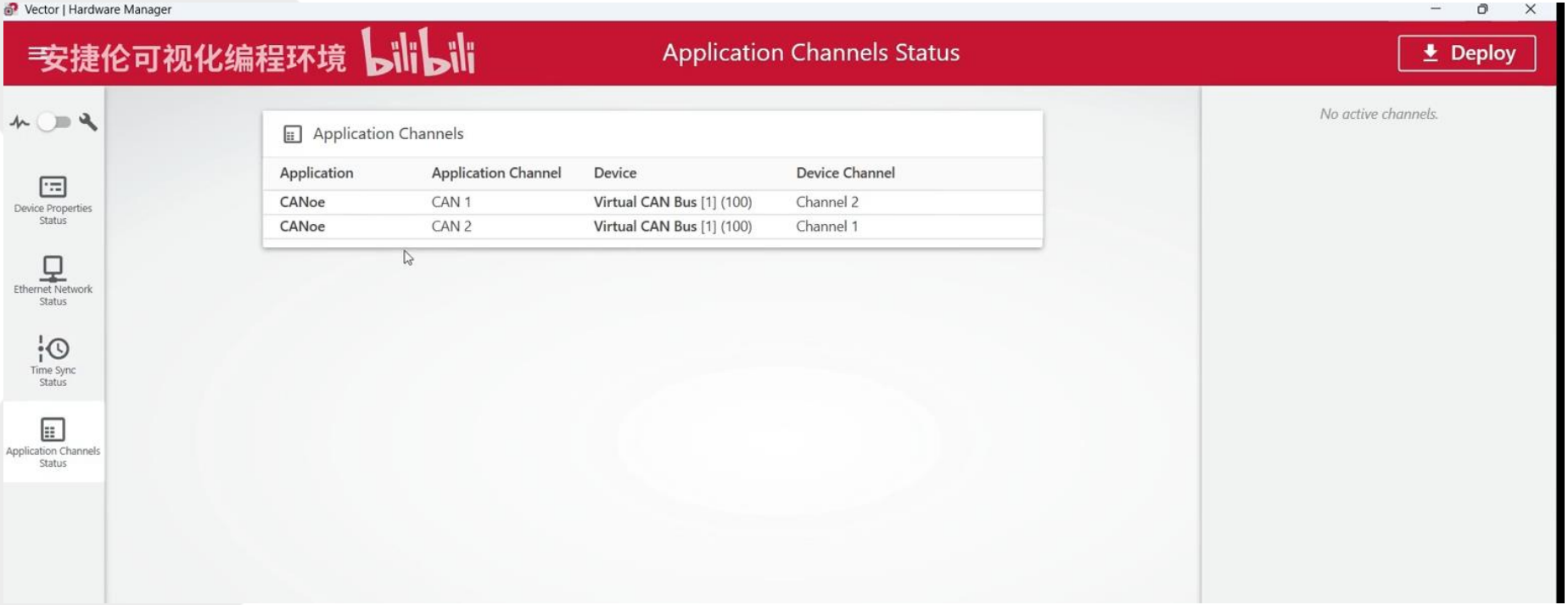


CANoe 16+

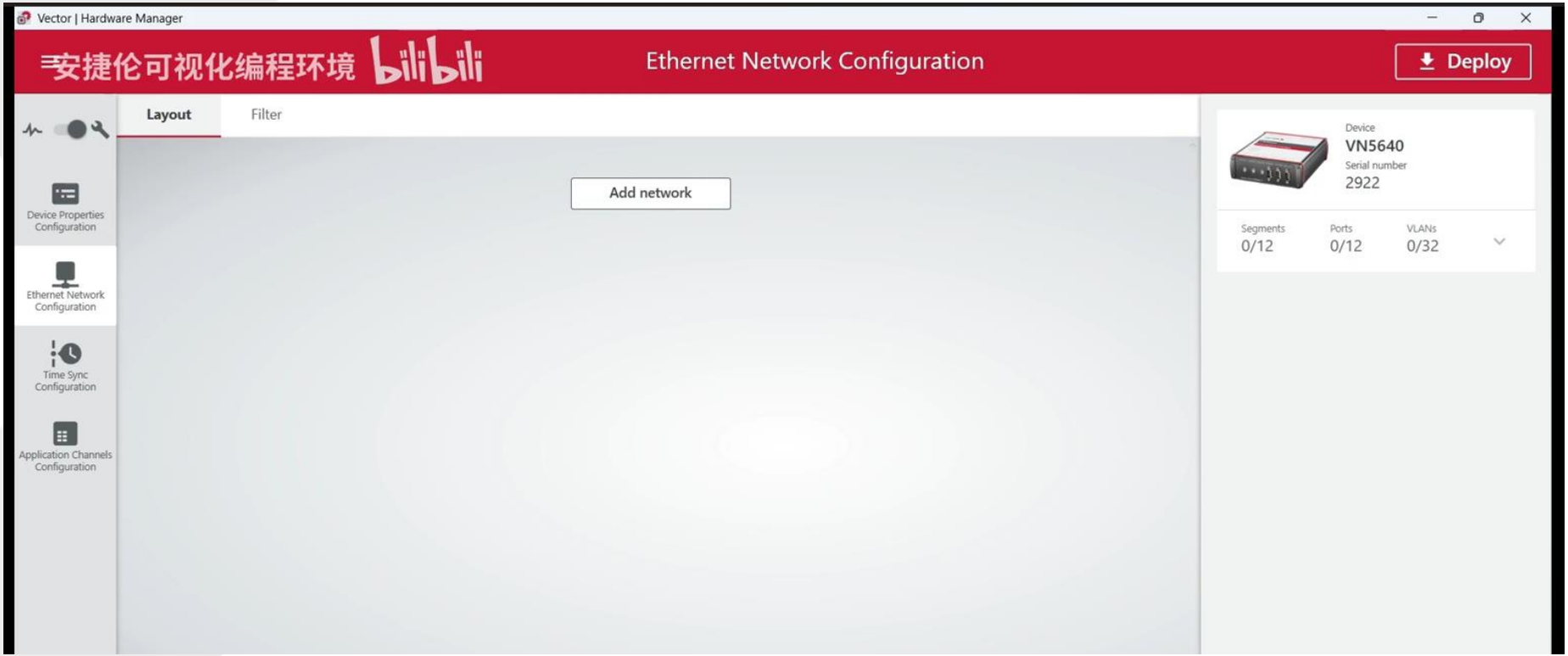


CANoe 16+

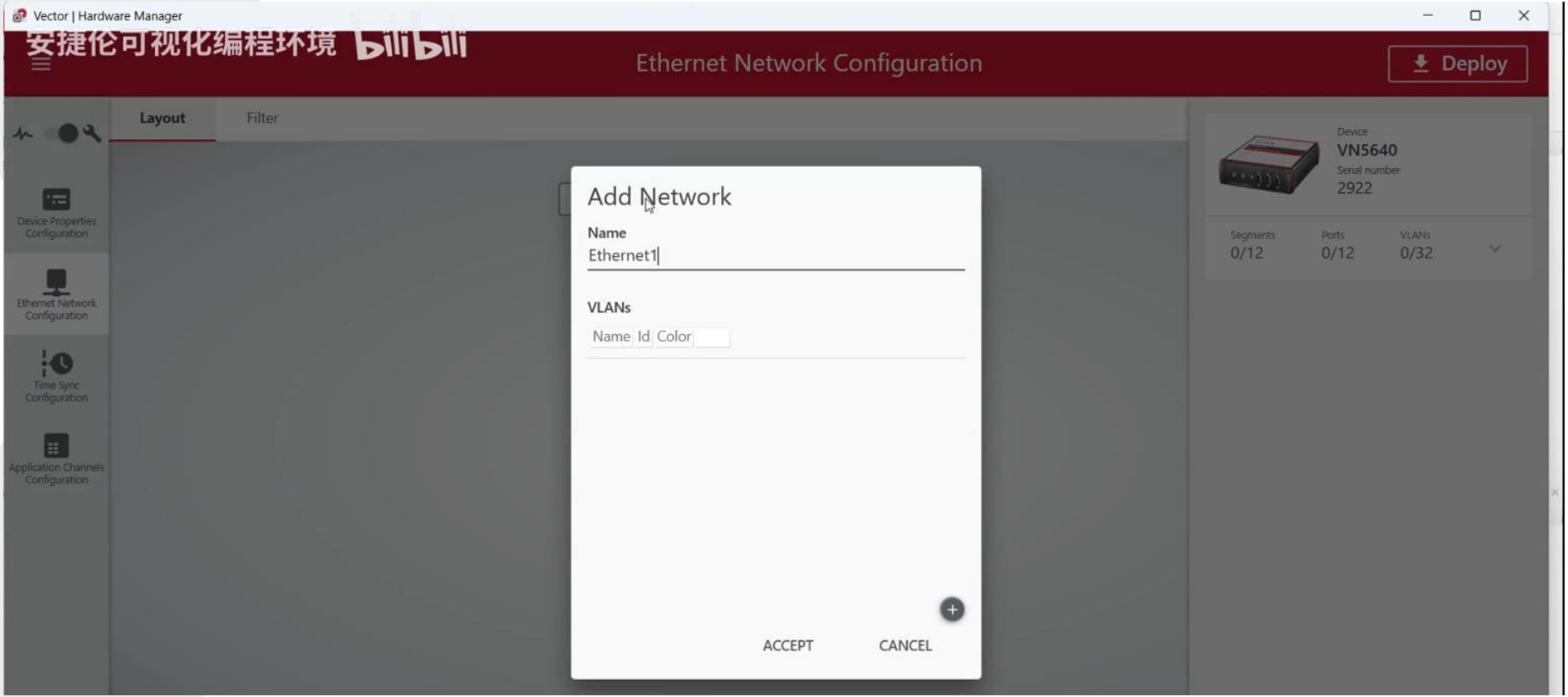




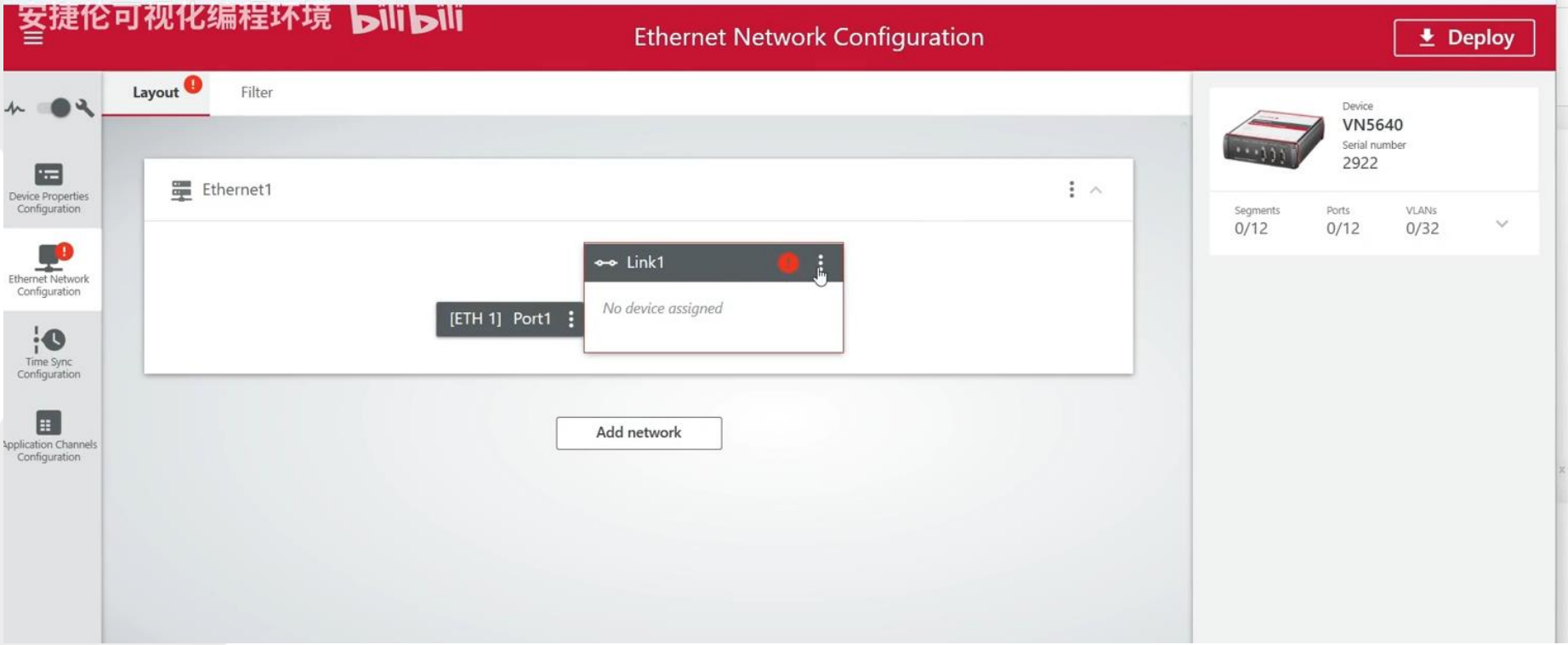
CANoe 16+



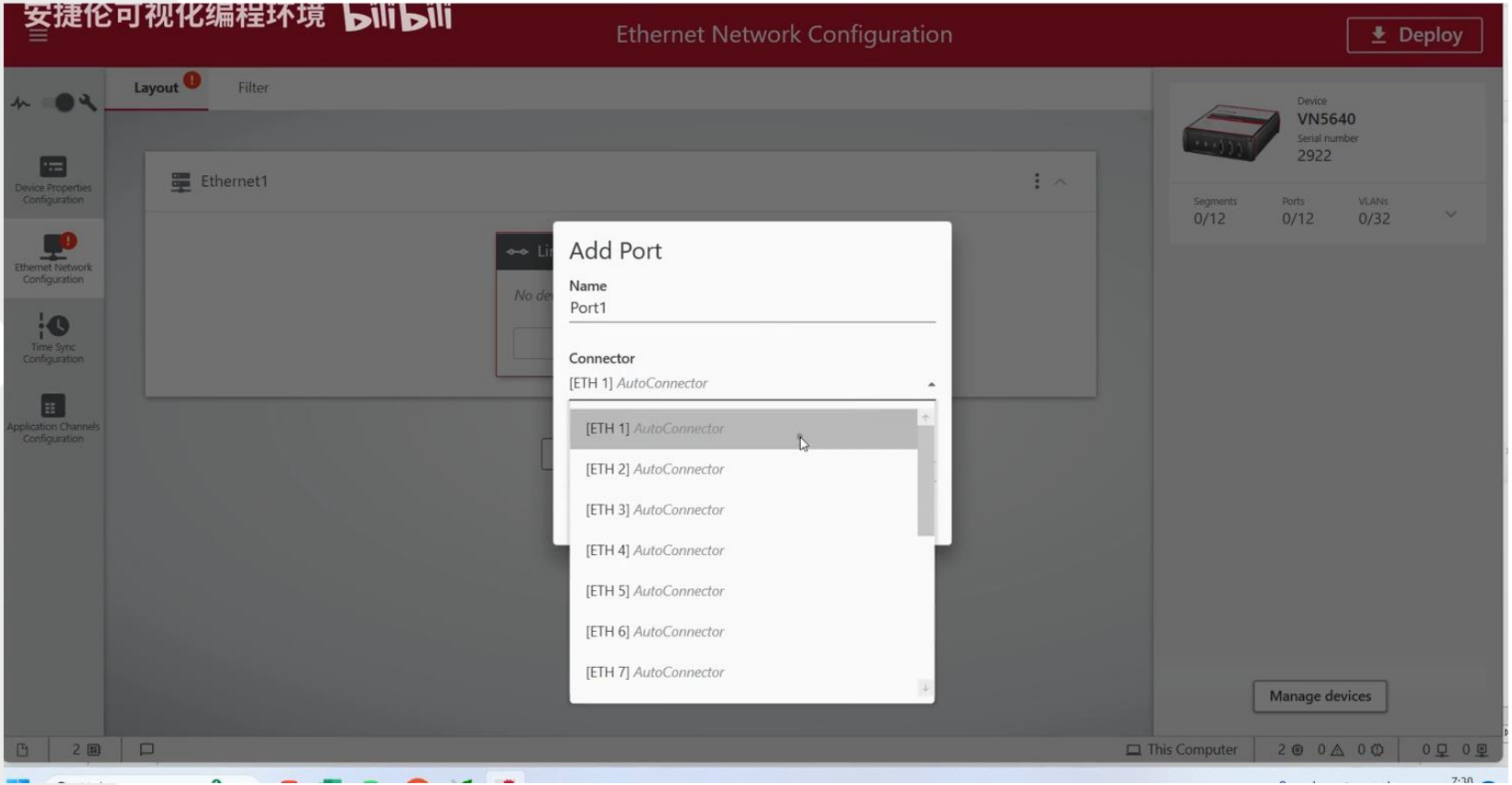
CANoe 16+



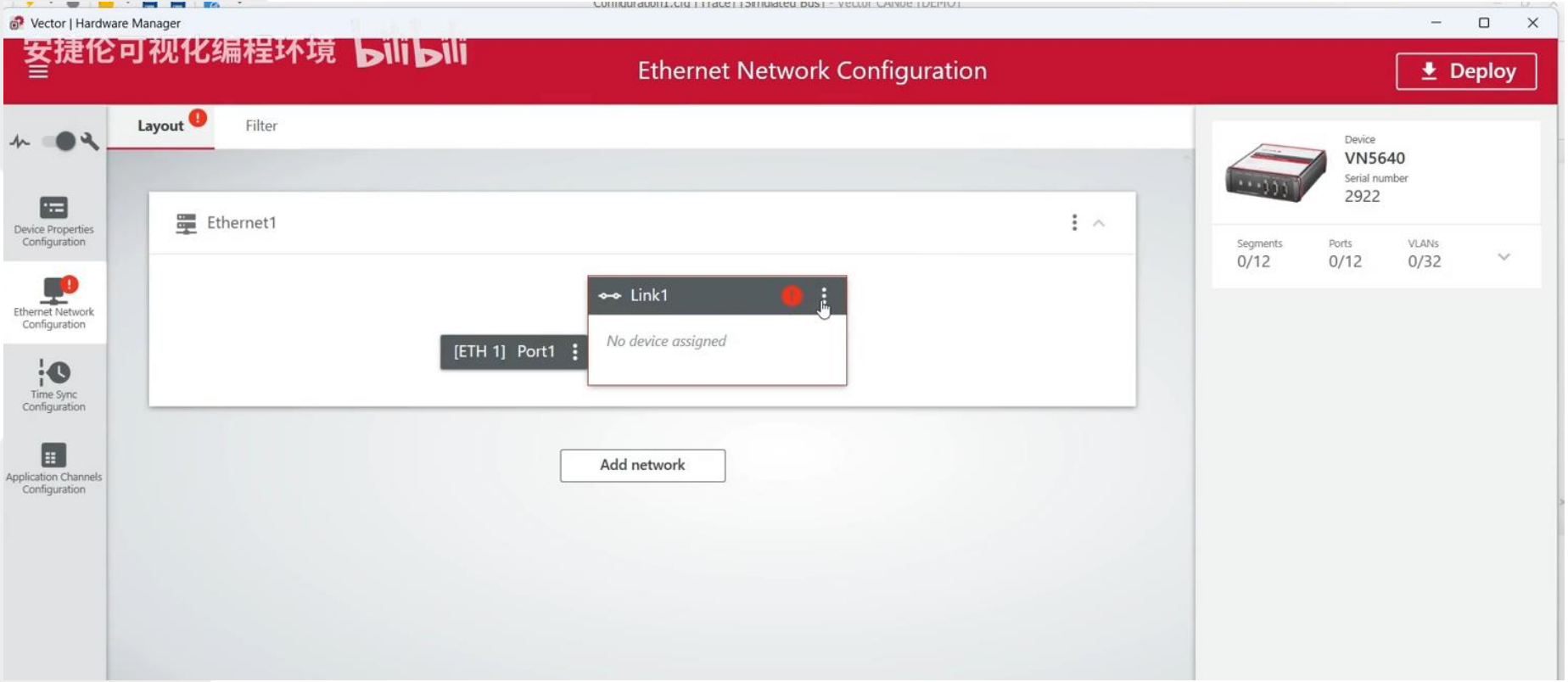
CANoe 16+



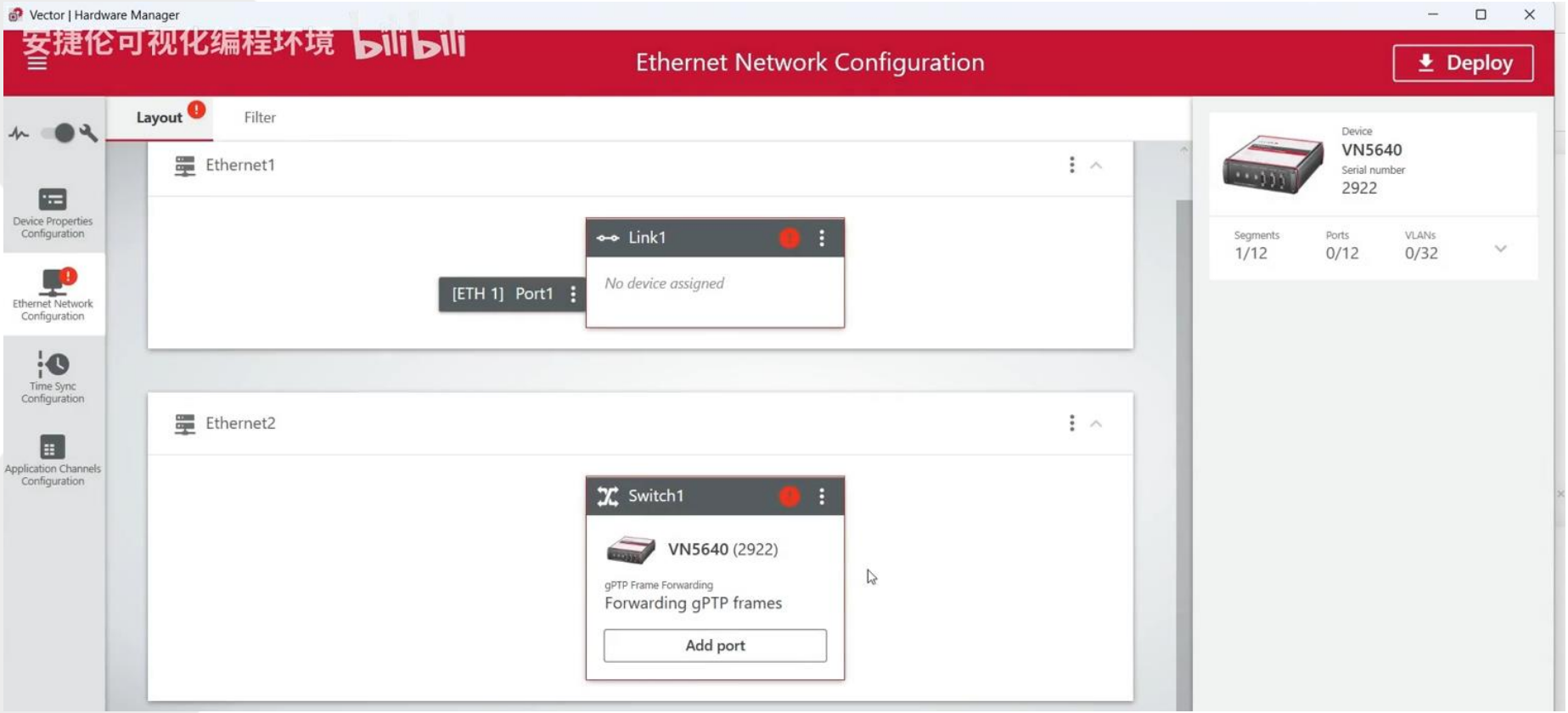
CANoe 16+



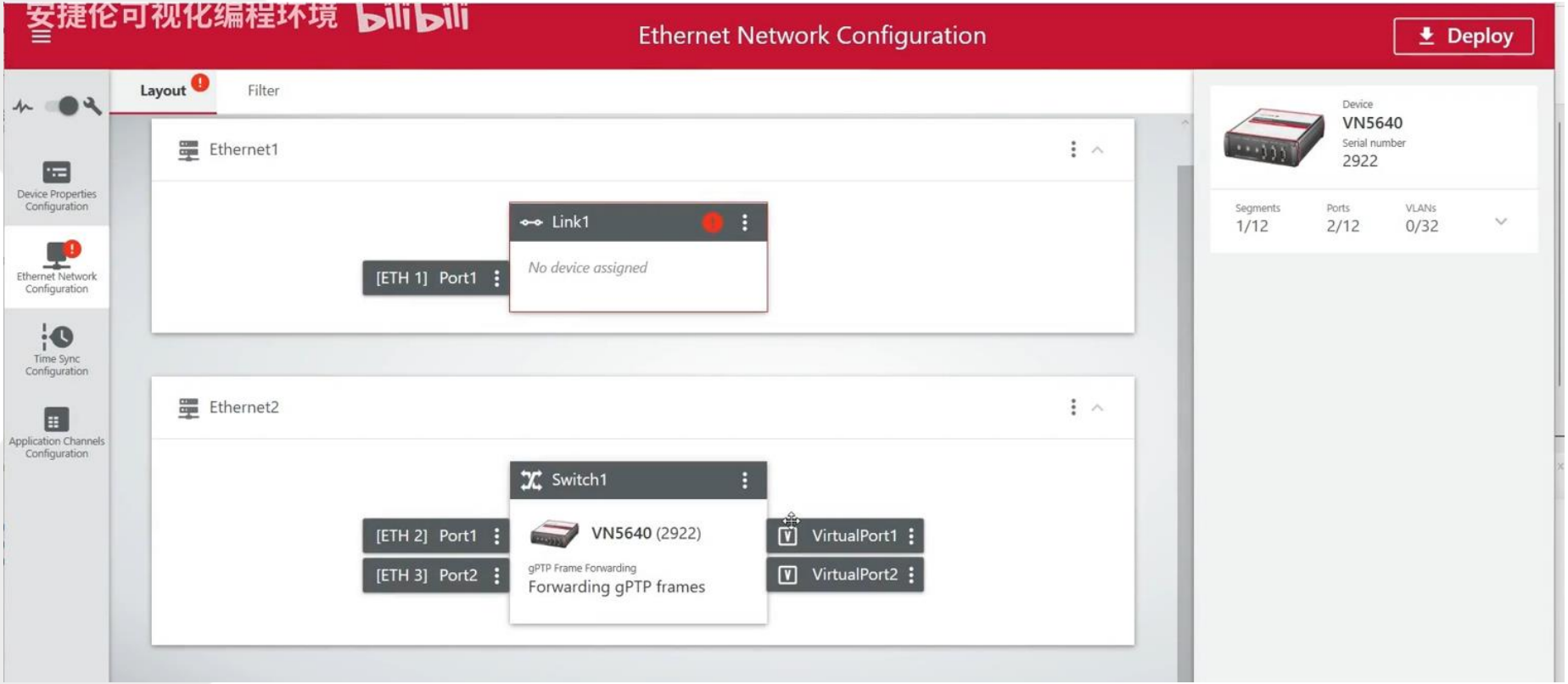
CANoe 16+



CANoe 16+



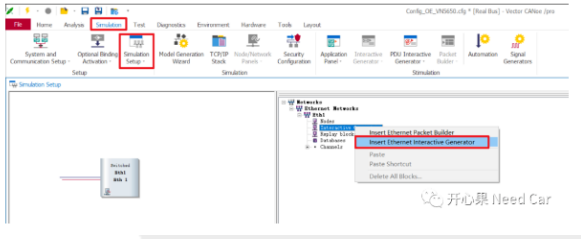
CANoe 16+



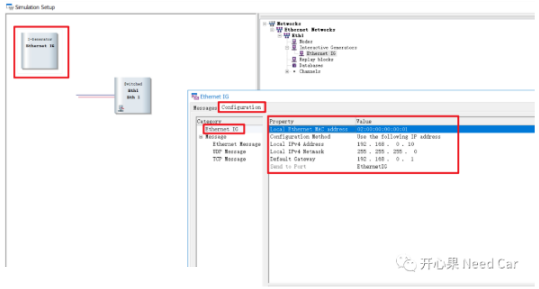
Ethernet IG configuration

3、创建仿真IG（Interactive Generator）模块

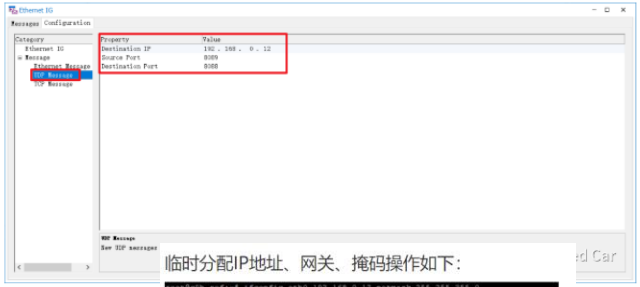
当需要模拟节点UDP/TCP报文发送时，可以通过IG等模块仿真，IG创建如下所示：



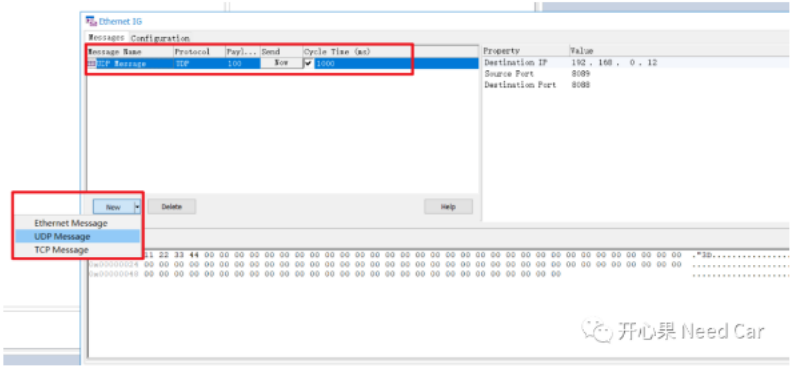
IG属性配置，本文设置虚拟以太网节点（IG）的IPv4地址：192.168.0.10，网络掩码：255.255.255.0，默认网关地址：192.168.0.1，如下所示：



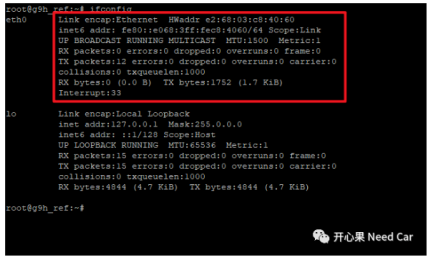
本文配置UDP报文发送属性，发送的目标IPv4地址：192.168.0.12，目标端口号：8088，IG端口号设置为：8089，如下所示：



添加仿真UDP报文，周期1000ms，如下所示：



提示：DUT的IP地址、网关、掩码以及端口等信息，开发过程中配置。 而本文的DUT型号是芯驰的G9H，基于Linux系统，可以通过命令行设置这些信息，本文只是临时配置，使用临时配置的方式修改DUT的IP地址、网关、掩码信息。 没有设置之前，网卡eth0并未分配IP地址、网关、掩码等信息，如下所示：

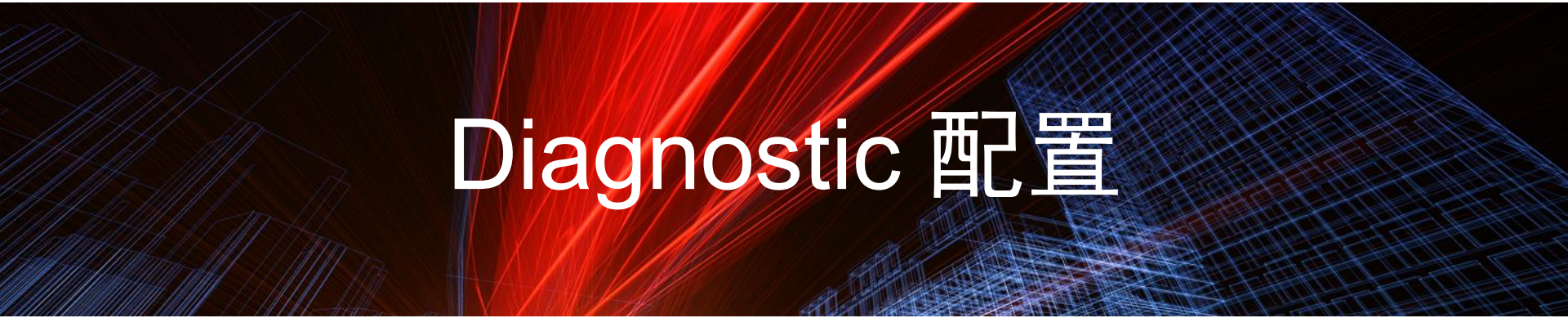


临时分配IP地址、网关、掩码操作如下：

上述命令如下：

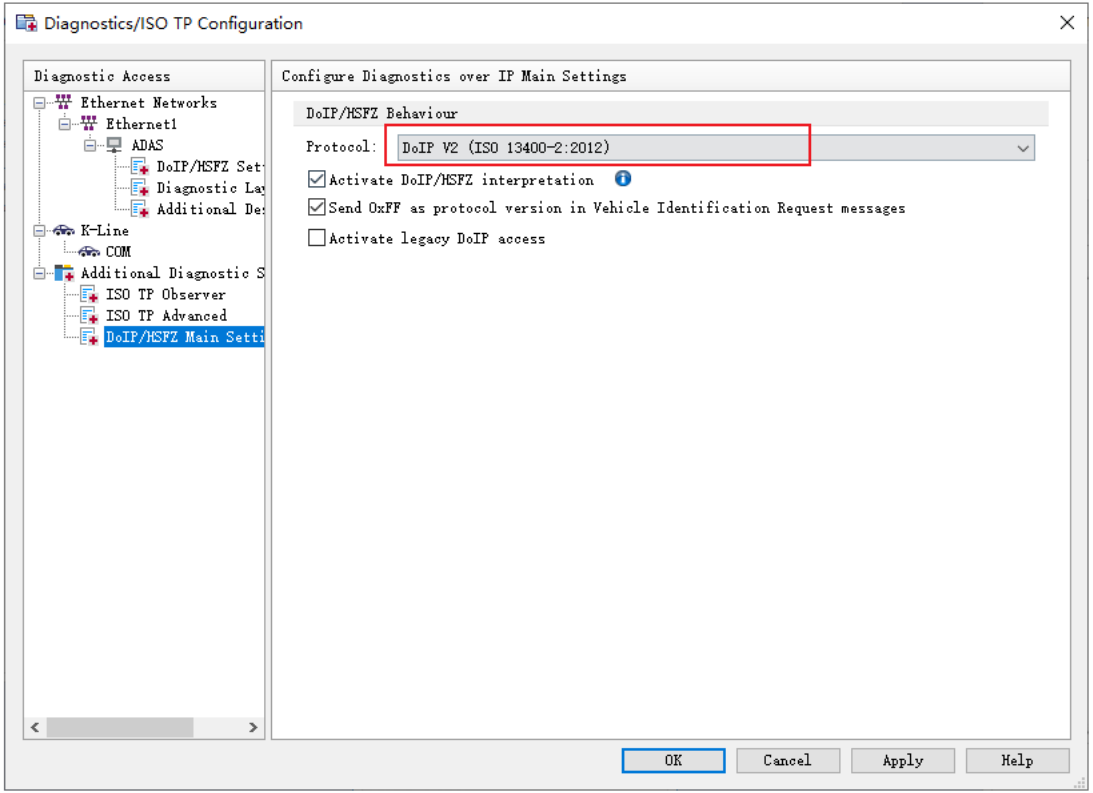
```
//配置网卡eth0的IPv4地址，网络掩码
ifconfig eth0 192.168.0.12 netmask
255.255.255.0
//配置默认路由地址
route add default gw 192.168.0.1
```

再次通过ifconfig确认DUT的IP地址、网关、掩码等信息，如下所示：

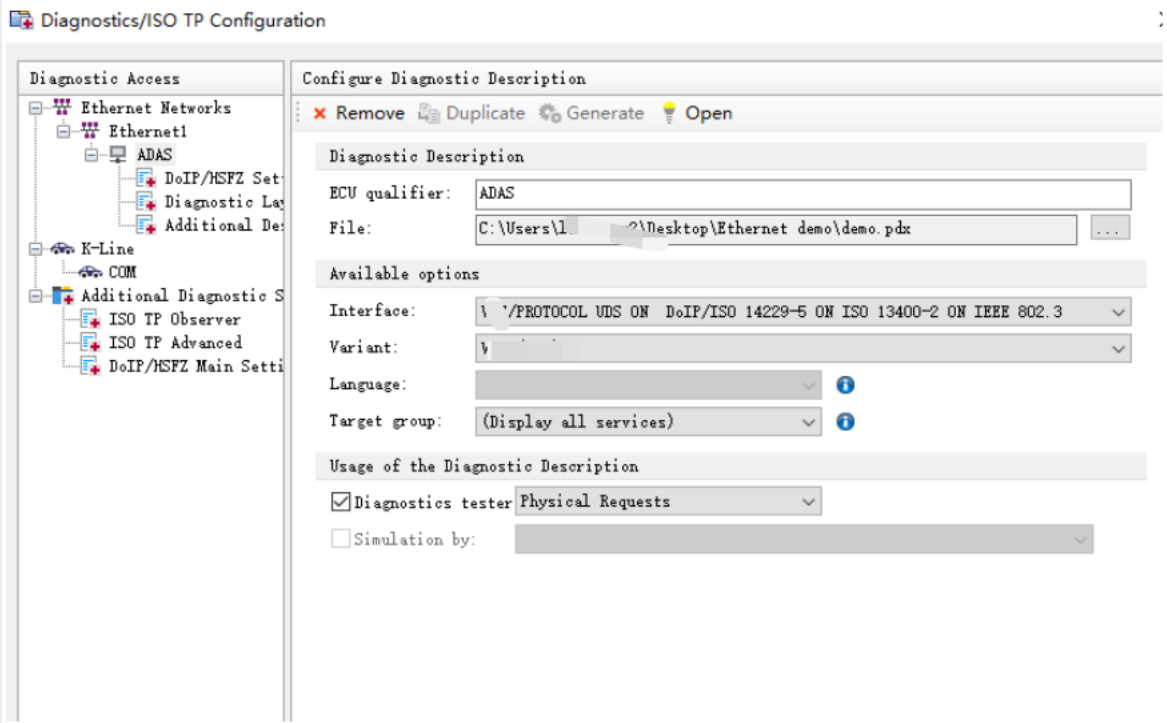


配置诊断数据库

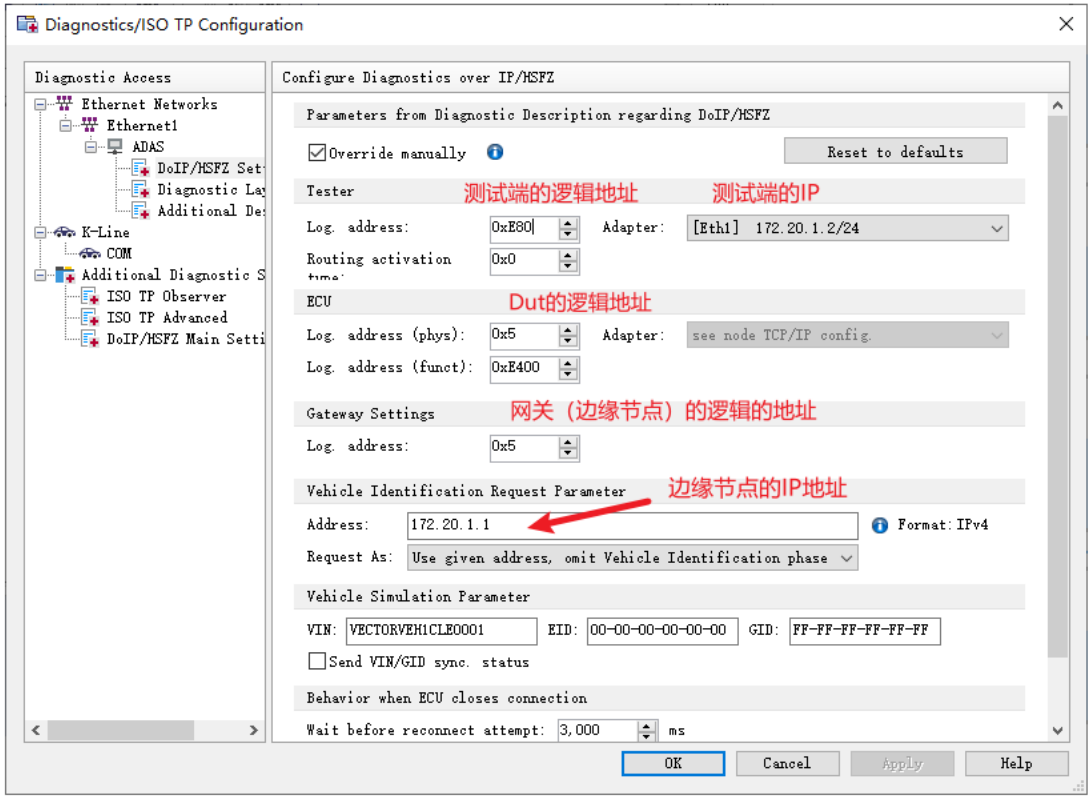
选择DUT支持的Doip协议版本。



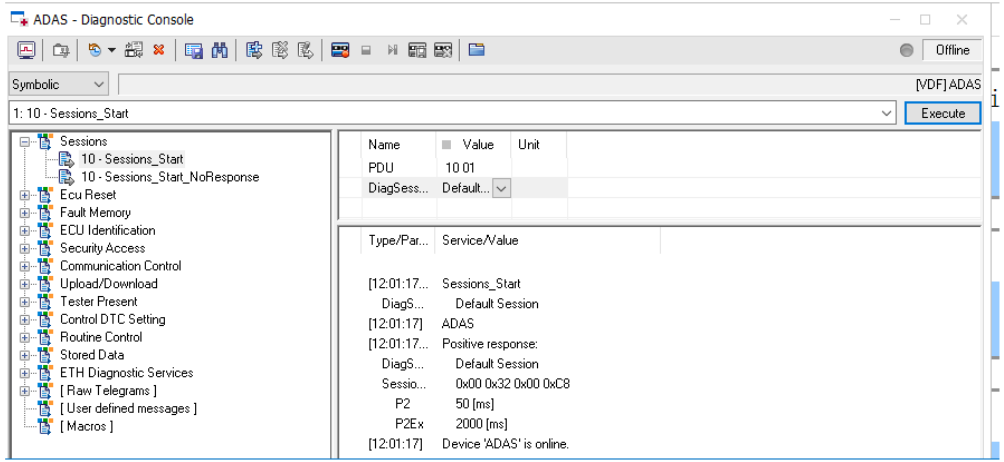
选择CDD/ODX/PDX等诊断数据库文件。

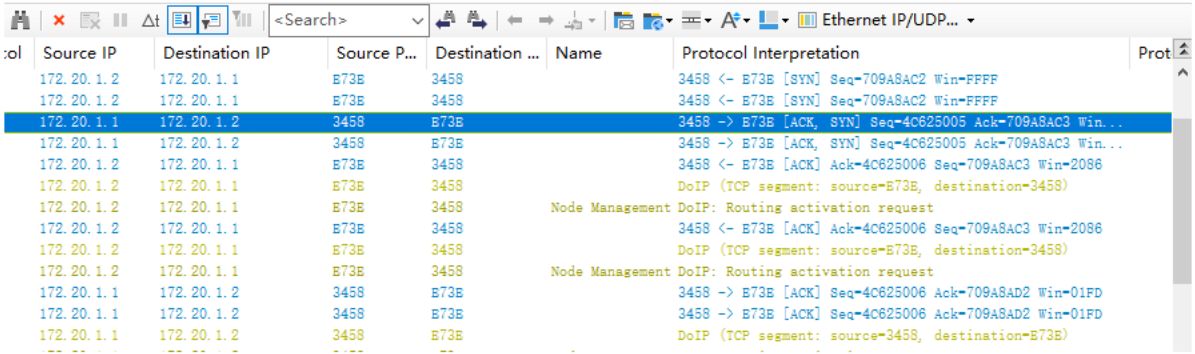


配置 IP地址

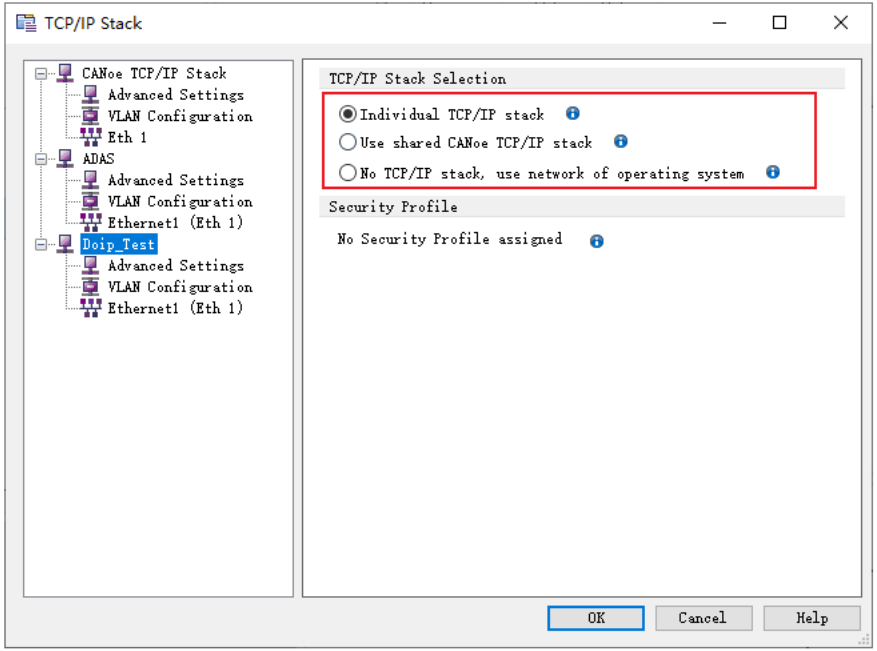


发送诊断测试



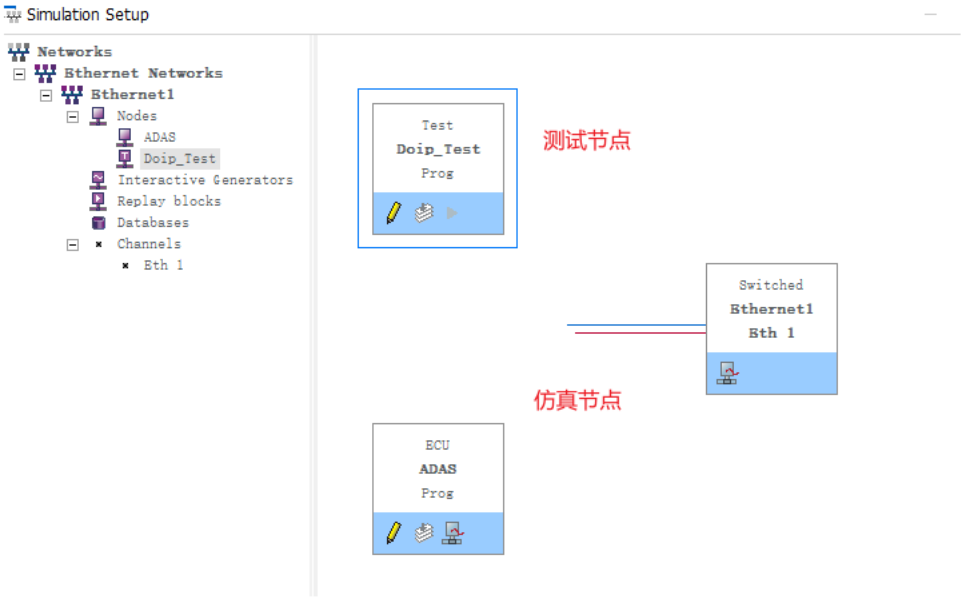


TCP/IP Stack配置的三种选项有什么不同？



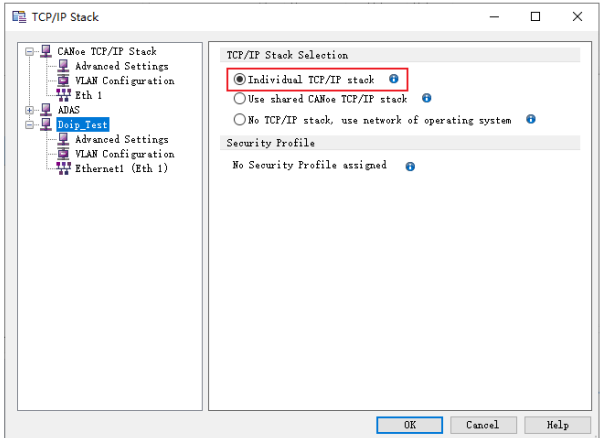
Action

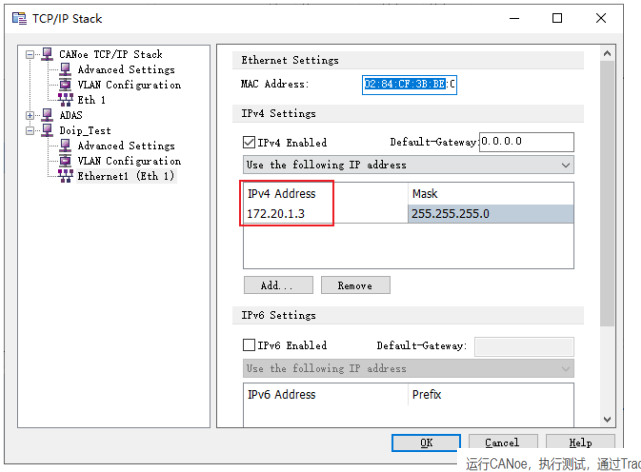
首先在一个空白工程中添加两个节点，一个是测试节点（Doip_Test），一个是网络节点（ADAS）



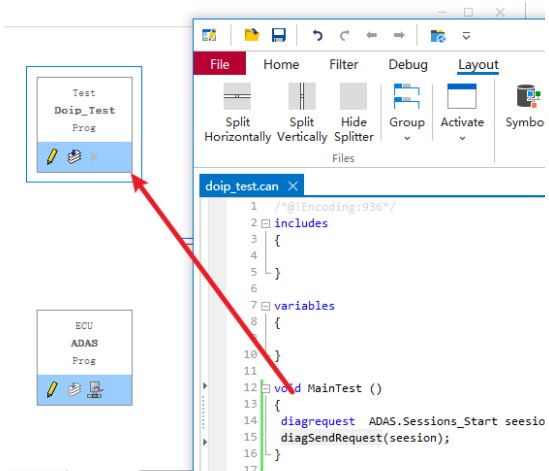
Individual TCP/IP Stack

为网络节点建立专有的TCP/IP栈。

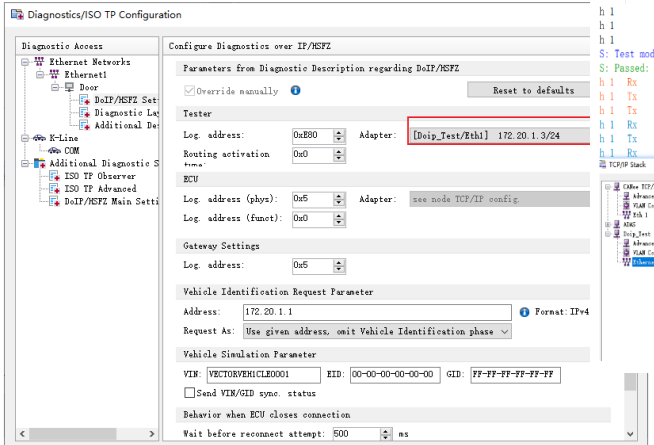




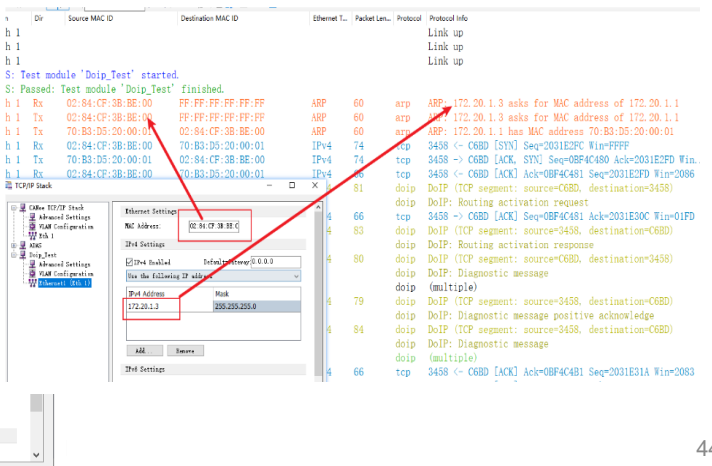
给Doip_Test 节点加载一个.can文件，就是简单的发送一个诊断请求



因为上面的.can代码是依赖诊断数据库发送的报文，所以这里还要选择下 测试段的IP

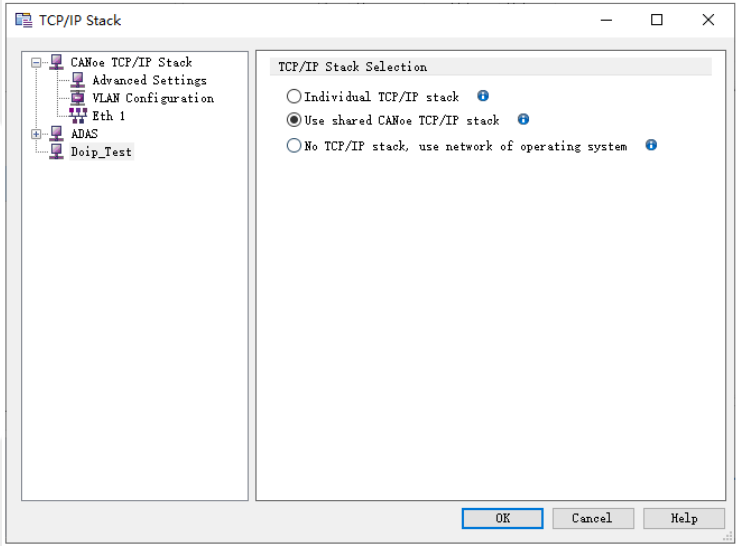


运行CANoe，执行测试，通过Trace可以看出，TCP/IP协议栈的MAC地址和IP地址都是Doip_Test节点自己的。

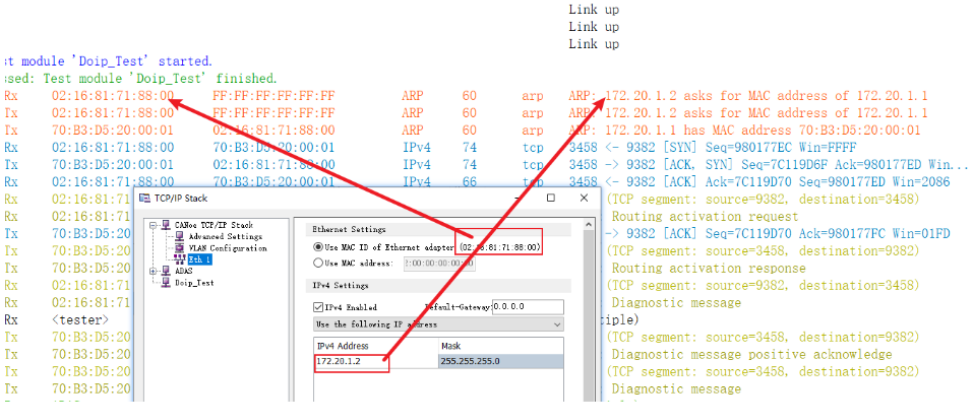


Use shared CANoe TCP/IP stack

CANoe有它自己的TCP/IP堆栈实例，它可以被几个共同的网络节点使用。使用此实例的节点在网络中仅作为一个节点可见，**CANoe TCP/IP Stack**.



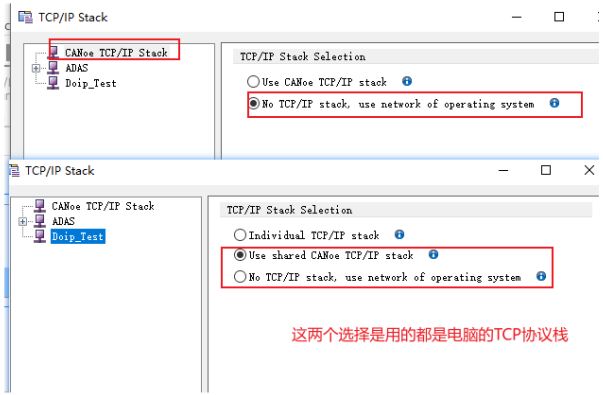
运行CANoe，执行测试，通过Trace可以看出，TCP/IP协议栈的MAC地址和IP地址都是**CANoe TCP/IP Stack**.



No TCP/IP stack, use network of operating system

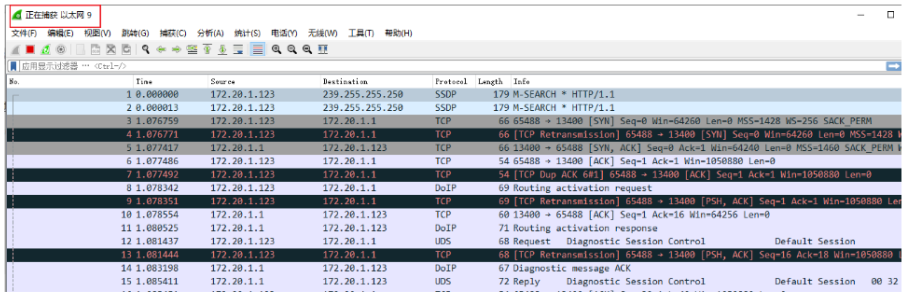
这个选项就是使用电脑的网卡或者拓展坞的网卡，CANoe不用做什么配置，**不过CANoe 中也检测不到任何的数据了。**

必须在**CANoe TCP/IP Stack**.选项中选择 No TCP/IP stack, use network of operating system，其它节点选择下图的两个选项都行。

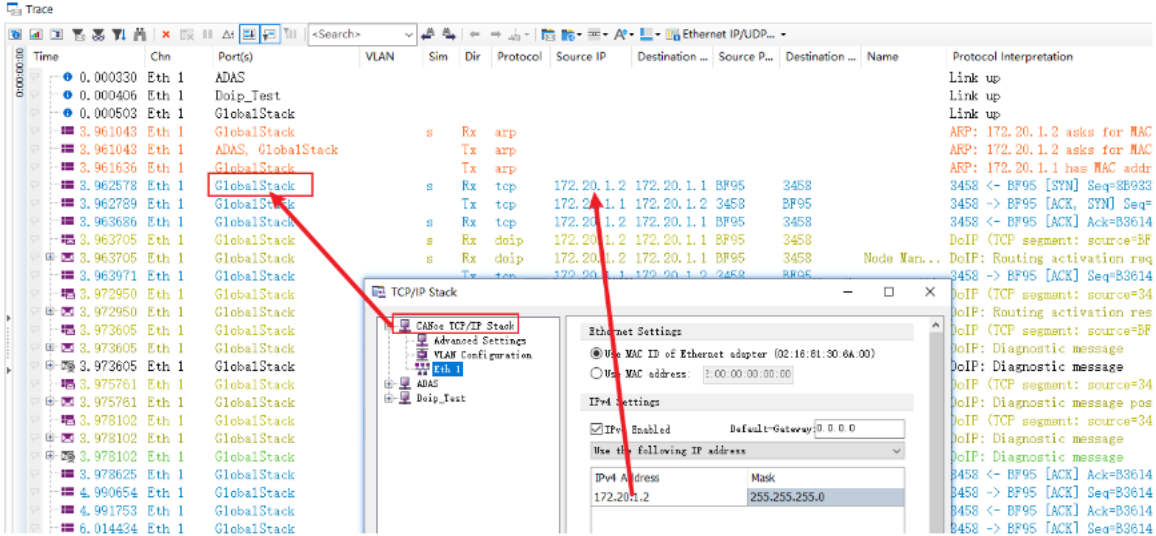


本次测试使用的是一个拓展坞的网卡，也要将拓展坞的IP地址设置到DUT的同一网段

运行CANoe，执行测试，通过CANoe中的Trace窗口看不到任何数据，这里是使用了wireShark 抓包工具在拓展坞的网卡上抓到的数据。



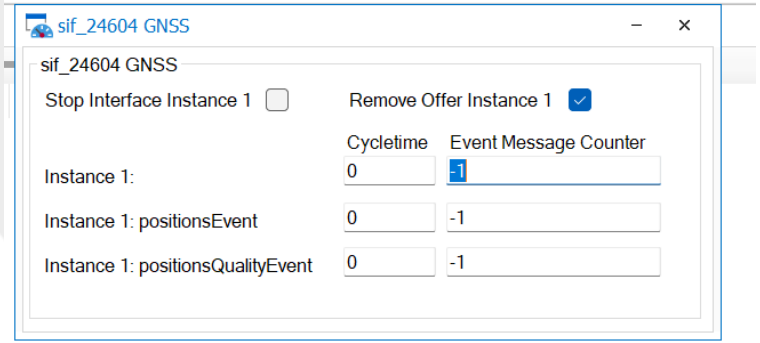
比如下图通过诊断数据库，在诊断控制台上发送诊断请求，trace窗口看到报文的发送端口是 GlobalStatck

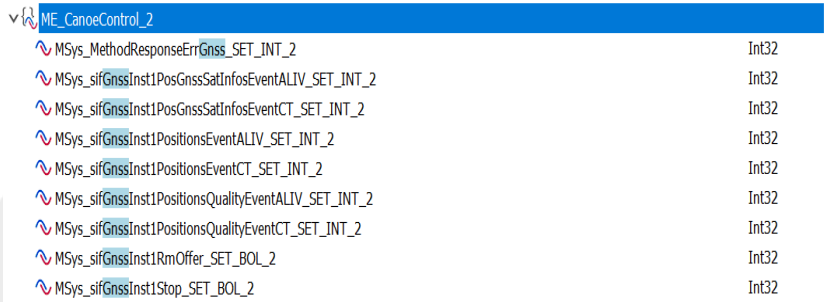


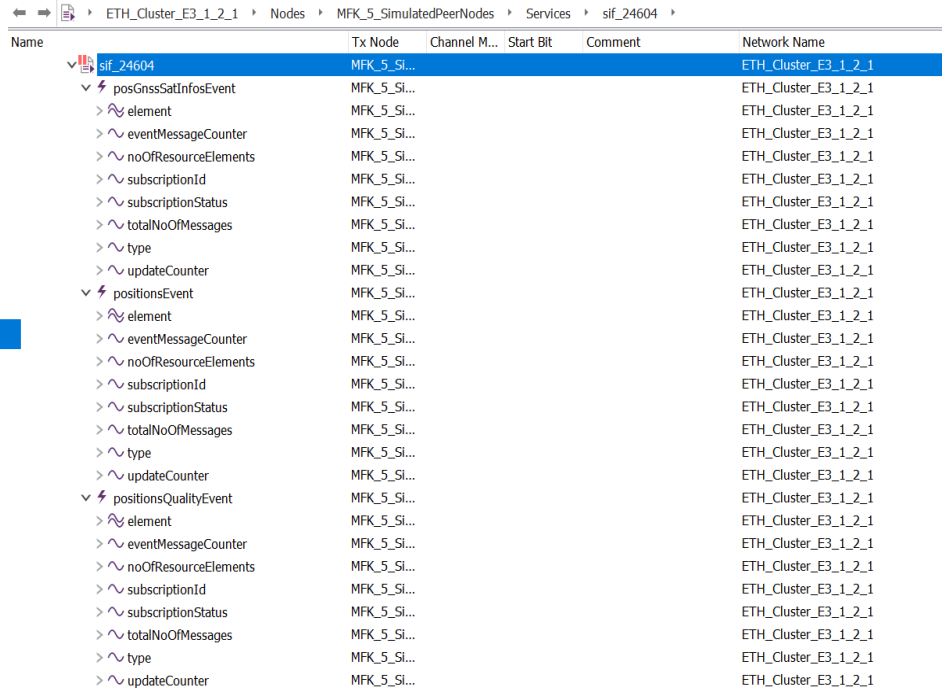
MFK5 Ethernet Test Case Example

Part1: GNSS Service - 24604

Configuration







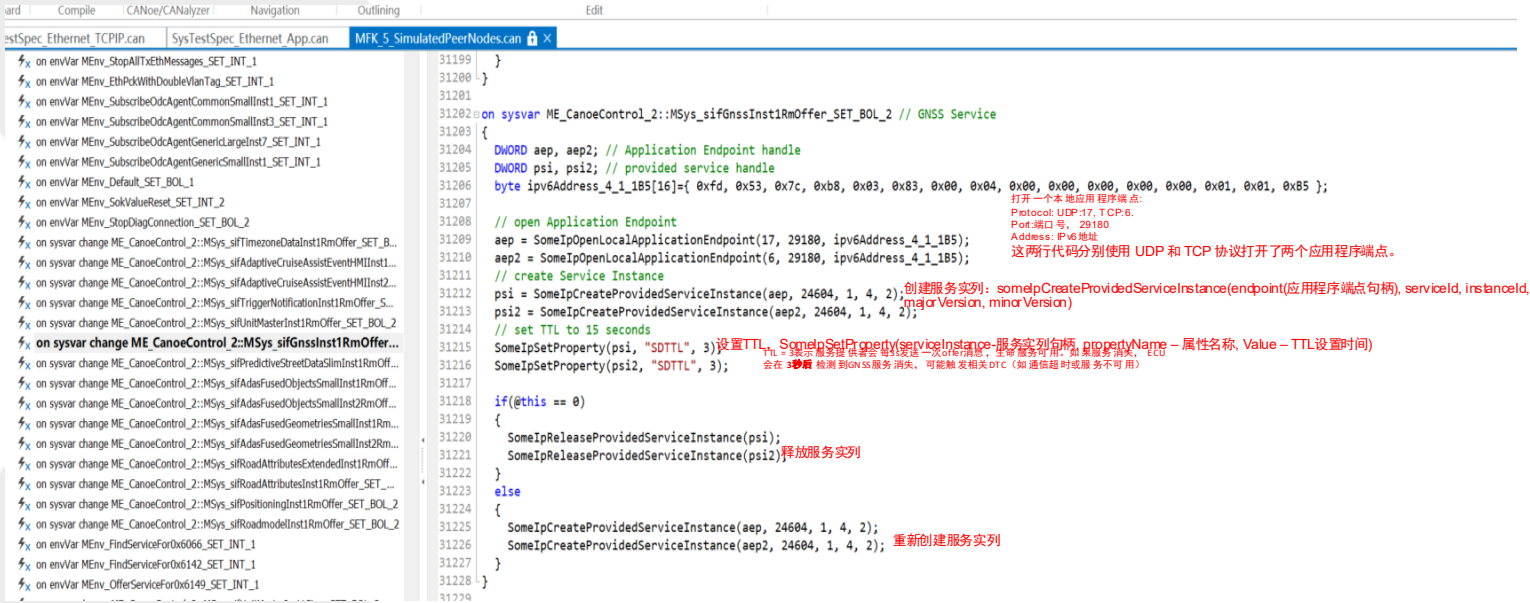
posGnssSatInfosEvent:
GMSS卫星的信息

positionsEvent:
当前位置信息的数据

positionsQualityEvent:
位置信息质量相关的数据

on sysvar

Summary:someip服务实列的创建和释放



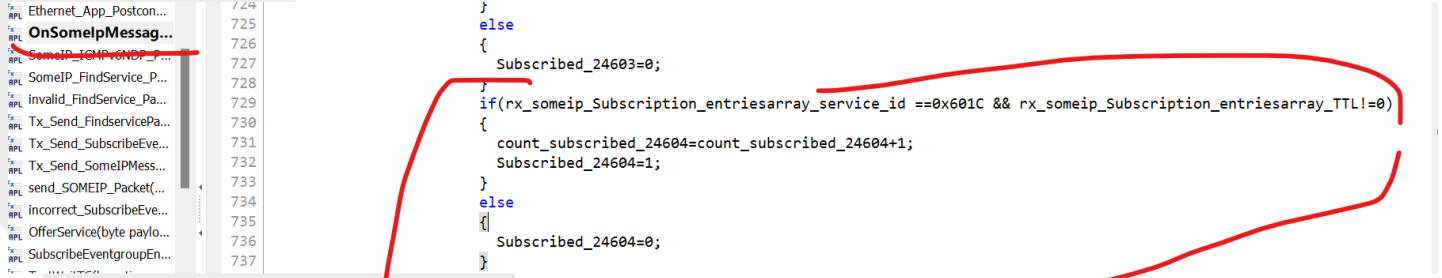
On sysvar

Summary:停止/使能24604服务和设置cycle time

MFK_5_SimulatedPeerNodes.can* | ME_SomeIpSubscribe.can

```
on sysvar ME_CanoeControl_2::MSys_sifGnssInst1Stop_SET_BOL_2
{
    DWORD hSrvcIf;
    hSrvcIf = SomeIpGetProvidedObjectHandle("sif_24604::4::2::1");
    SomeIpSDSetServiceStatus(hSrvcIf, (@this == 0));
}
on sysvar ME_CanoeControl_2::MSys_sifGnssInst1PosGnssSatInfosEventCT_SET_INT_2
{
    DWORD hEvnt;
    hEvnt = SomeIpGetProvidedObjectHandle("sif_24604::posGnssSatInfosEvent::4::2::1");
    @this = (@this >= 10) * @this;
    SomeIpSetProperty(hEvnt, "CycleTimeMs", @this);
}
on sysvar ME_CanoeControl_2::MSys_sifGnssInst1PositionsEventCT_SET_INT_2
{
    DWORD hEvnt;
    hEvnt = SomeIpGetProvidedObjectHandle("sif_24604::positionsEvent::4::2::1");
    @this = (@this >= 10) * @this;
    SomeIpSetProperty(hEvnt, "CycleTimeMs", @this);
}
on sysvar ME_CanoeControl_2::MSys_sifGnssInst1PositionsQualityEventCT_SET_INT_2
{
    DWORD hEvnt;
    hEvnt = SomeIpGetProvidedObjectHandle("sif_24604::positionsQualityEvent::4::2::1");
    @this = (@this >= 10) * @this;
```

Lib_common.cin::OnSomeIpMessage()



接收到someip pdu后判断为订阅服务 且
是24604后， 响应count累加。

Q：SomeipReleaseProvidedServiceInstance和SomeipSDSetServiceStatus区别在哪里？

对比维度	SomeipReleaseProvidedServiceInstance	SomeipSDSetServiceStatus
功能目的	永久释放服务实例，从网络中移除。	动态设置服务状态（启用/禁用），不移除服务实例。
SOME/IP-SD报文影响	发送 StopOffer 广播，删除服务注册表中的 OfferService 条目。	不修改 OfferEntry，仅内部标记状态（对端不可见）。
资源释放	释放服务实例所有资源（端口、内存等）。	不释放资源，仅更新状态标志。
重新激活方式	需重新调用 SomeipOfferService 注册服务。	直接调用 SomeipSDSetServiceStatus(SERVICE_STATUS_AVAILABLE)。
对订阅者的影响	订阅者立即收到服务下线通知，触发重寻址或故障处理。（TTL立即为0）	订阅者仍可见服务，但请求会失败（依赖超时检测 - TTL超时）。
网络流量	产生 StopOffer 广播流量。	无额外网络流量。
适用场景	- ECU关闭 - 服务永久下线 - 功能降级移除服务。	- 服务临时不可用（如信号丢失） - 测试/诊断手动控制状态。
AUTOSAR需求映射	SWS_SOMEIPSD_00423（服务生命周期管理）。	SWS_SOMEIPSD_00567（动态服务状态控制）。
典型代码示例	SomeipReleaseProvidedServiceInstance(GNSS, 1);	SomeipSDSetServiceStatus(GNSS, 1, UNAVAILABLE);

Callback Function

Summary:Alivecounter累加或设值

- OnSomeIpProcessTxMessage()

```
if (nMsgId == 0x601cc080) // sif_24604::posGnssSatInfosEvent
{
    if (@ME_CanoeControl_2::MSys_sifGnssInst1PosGnssSatInfosEventALIV_SET_INT_2 < 0)
    {
        nSifGnssI1PosGnssSatInfosEventEvtMsgCntr++;
    }
    else
    {
        nSifGnssI1PosGnssSatInfosEventEvtMsgCntr =
            (WORD) (@ME_CanoeControl_2::MSys_sifGnssInst1PosGnssSatInfosEventALIV_SET_INT_2 & 0x0000ffff);
    }
    SomeIpSetValueDWord(hMsg, "eventMessageCounter", (DWORD) nSifGnssI1PosGnssSatInfosEventEvtMsgCntr);
}
if (nMsgId == 0x601cc180) // sif_24604::positionsEvent
{
    if (@ME_CanoeControl_2::MSys_sifGnssInst1PositionsEventALIV_SET_INT_2 < 0)
    {
        nSifGnssI1PositionsEventEvtMsgCntr++;
    }
    else
    {
        nSifGnssI1PositionsEventEvtMsgCntr =
            (WORD) (@ME_CanoeControl_2::MSys_sifGnssInst1PositionsEventALIV_SET_INT_2 & 0x0000ffff);
    }
    SomeIpSetValueDWord(hMsg, "eventMessageCounter", (DWORD) nSifGnssI1PositionsEventEvtMsgCntr);
}
if (nMsgId == 0x601cc280) // sif_24604::positionsQualityEvent
{
    if (@ME_CanoeControl_2::MSys_sifGnssInst1PositionsQualityEventALIV_SET_INT_2 < 0)
    {
        nSifGnssI1PositionsQualityEventEvtMsgCntr++;
    }
    else
    {
        nSifGnssI1PositionsQualityEventEvtMsgCntr =
```

GNSS相关测试点

分类	子类	测试用例ID	测试点描述	可能的需求描述	验证手段
服务订阅验证	服务订阅行为验证	EthernetAPP_15760245	验证GNSS服务禁用后，DUT在KL15重启时不主动订阅，恢复服务后重新订阅	需求：DUT应在服务可用时订阅，不可用时停止订阅	1. 禁用服务后重启KL15 2. 检查订阅状态（预期：无订阅） 3. 恢复服务后验证订阅
	服务订阅稳定性验证	EthernetAPP_16548323	验证GNSS服务订阅在长时间（1分钟）内的稳定性	需求：DUT应持续订阅关键服务（如GNSS），避免意外断开	1. 监控订阅计数器count_subscribed_24604_MFK5 2. 持续60秒验证计数是否持续增加
	SOME/IP订阅响应验证	EthernetAPP_15213434	验证DUT对SIF 24604和24590的SubscribeEventgroupAck响应是否符合规范	需求：DUT应正确响应SOME/IP订阅请求，返回有效的SubscribeEventgroupAckEntry	1. 发送订阅请求 2. 检查响应计数器count_subscribeAck_xxx_MFK5是否递增
相关DTC验证	网络超时与DTC关联性验证	EthernetAPP_21377936	验证CAN总线OFF错误与GNSS服务超时DTC的独立性（确保不误报）	需求：不同通信通道（CAN/ETH）的故障应独立处理，避免交叉影响	1. 触发CAN Bus OFF 2. 禁用GNSS服务 3. 分别验证DTC 0x10004D和0x10026F状态
	网络状态与DTC触发条件验证	EthernetAPP_21377935	验证NWDF信号（网络唤醒标志）对GNSS服务超时DTC的影响	需求：DTC触发应依赖实际服务状态，而非网络唤醒信号	1. 强制NWDF=0 2. 禁用GNSS服务 3. 验证DTC状态（预期：不受NWDF影响）
	全局错误与DTC关联性验证	EthernetAPP_21377933	验证全局通信错误（如CAN消息停止）与GNSS服务超时DTC的独立性	需求：全局通信错误不应触发服务级DTC	1. 停止所有CAN消息 2. 禁用GNSS服务 3. 验证DTC 0x10026F状态（预期：无影响）
	DTC诊断功能验证	EthernetAPP_14277036	验证当GNSS服务不可用时，系统能正确记录DTC (0x10026F) 并在恢复后更新状态	需求：系统应检测SOME/IP服务可用性，并在服务丢失时生成DTC，恢复后更新DTC状态	1. 禁用GNSS服务 2. 触发睡眠唤醒 3. 读取DTC状态（预期：0x2F表示故障） 4. 恢复服务后验证DTC状态（预期：0x2E表示恢复）

EthernetAPP_15760245

验证GNSS服务禁用后，DUT在KL15重启时不主动订阅，恢复服务后重新订阅

```
testcase EthernetAPP_15760245(char TestDescription[], char TestRequirements[], char TestParameters[])
{
…

  ReportComment("--------------TEST SEQUENCE--------------");
  TestStep("1.0", " Disable the service sif_24604");
  @ME_CanoeControl_2::MSys_sifGnssInst1Stop_SET_BOL_2=1;

  TestStep("2.0", "Set KL_15 OFF ");
  SetKl15OutputTS(0);

  TestStep("3.0", "Set KL_15 ON ");
  SetKl15OutputTS(1);

  TestStep("4.0", "Check SIF should not be subscribed");
  Subscription_Check_Flag=1; // when flag is enabled then only check status so previous values are not stored.
  testWaitForTimeout(100); // wait for signal update in on someip function
  if(Subscribed_24604==0)
  {
    Subscription_Check_Flag=0; //make sure Flag is 0 so it can be triggered next time when required.
    testStepPass("SubscribeEventGroup Response","DUT does not responds with SubscribeEventGroup");
  }
  else
  {
Subscription_Check_Flag=0; //make sure Flag is 0 so it can be triggered next time when required.
    testStepFail("SubscribeEventGroup Response","DUT does responds with SubscribeEventGroup");
  }

  TestStep("5.0", "wait for 180 sec (3minutes)");
  testWaitForTimeout(180000);

  TestStep("6.0", " Enable the service sif_24604");
  @ME_CanoeControl_2::MSys_sifGnssInst1Stop_SET_BOL_2=0;

  TestStep("7.0", "Check SIF should not be subscribed");
  Subscription_Check_Flag=1; // when flag is enabled then only check status so previous values are not stored.
  testWaitForTimeout(500); // wait for signal update in on someip function
  if(Subscribed_24604==1)
  {
    Subscription_Check_Flag=0; //make sure Flag is 0 so it can be triggered next time when required.
    testStepPass("SubscribeEventGroup Response","DUT does responds with SubscribeEventGroup");
  }
  else
  {
Subscription_Check_Flag=0; //make sure Flag is 0 so it can be triggered next time when required.
    testStepFail("SubscribeEventGroup Response","DUT does not responds with SubscribeEventGroup");
  }
….
```

1. 禁用服务后重启KL15

2. 检查订阅状态（预期：无订阅）

3. 恢复服务后验证订阅

EthernetAPP_16548323

验证GNSS服务订阅在长时间（1分钟）内的稳定性

```
testcase EthernetAPP_16548323(char TestDescription[], char TestRequirements[], char TestParameters[])
{
  count_subscribed_24604_MFK5=0;
….

  ReportComment("---------------TEST SEQUENCE---------------");
  TestStep("1.0", " DUT: Sends SOME/IP Message with Options");
  testWaitForTimeout(10000); // to capture the trace

  TestStep("2.0", "Tester : Check the service 24604 ( GNSS - Position ) in trace ");
  testWaitForTimeout(60000); //Check if service 24604 subscribing in 1 min
  if(count_subscribed_24604_MFK5!=0){

    testStepPass("SubscriptionResponse","GNSS service consistently subscribed for more than 1minute ");
  }
  else
  {
    testStepFail("SubscriptionResponse","GNSS service  not consistently subscribed for more than 1minute ");
  }
…
}
```

1. 监控订阅计数器count_subscribed_24604_MFK5

2. 持续60秒验证计数是否持续增加

EthernetAPP_15213434

验证DUT对SIF 24604和24590的SubscribeEventgroupAck响应是否符合规范

```
testcase EthernetAPP_15213434(char TestDescription[], char TestRequirements[], char TestParameters[])
{
  count_subscribed_24604_MFK5=0;
  count_subscribeAck_24604_MFK5=0;
...


  ReportComment("---------------TEST SEQUENCE---------------");
  TestStep("1.0", "DUT : Sends SubscribeEventgroupEntry with service sif_24604");
  SubscriptionAck_Check_Flag=1;
  testWaitForTimeout(1000); // to capture the trace

  TestStep("2.0", "TESTER: Check SubscribeEventgroupAckEntry ");
  if(count_subscribed_24604_MFK5!=0 && count_subscribeAck_24604_MFK5!=0)
  {
   testStepPass("Result"," DUT responds SOME/IP SD Message with valid SubscribeEventgroupAck FOR SIF 24604");
  }
  else
  {
   testStepFail("Result"," DUT NOT responds with valid SubscribeEventgroupAck FOR SIF 24604");
  }
…
}
```

1. 使能订阅响应检查位

2. 检查响应计数器count_subscribeAck_xxx_MFK5是否递增

EthernetAPP_21377936

验证CAN总线OFF错误与GNSS服务超时DTC的独立性（确保不误报）

```
testcase EthernetAPP_21377936(char TestDescription[], char TestRequirements[], char TestParameters[])
{
…

Teststep("3.0","create CAN Bus OFF error");
canConfigureBusOff(1, 0x1B00004Fx, 1); // enable disturbance
  testWaitForTimeout(3000);

TestStep("4.0-5.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 00 4D 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 00 4D 01", "59 06 10 00 4D 27", "FF FF FF FF FF FF");

Teststep("6.0","Stop service GNSS");
@ME_CanoeControl_2::MSys_sifGnssInst1RmOffer_SET_BOL_2 = 0;

Teststep("7.0","Wait for maturity time");
testWaitForTimeout(3200);

TestStep("8.0-9.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF");
if(MFK5.diagResp[5] == 0x00 || MFK5.diagResp[5] == 0x50){
testStepPass("DTC 0x10026F is neither active nor passive");
}
else{
testStepFail("DTC 0x10026F is either active or passive");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

    Teststep("10.0","Remove CAN Bus OFF error");
canConfigureBusOff(1, 0x1B00004Fx, 0); //disable diaturnce
  testWaitForTimeout(500);

TestStep("11.0-12.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 00 4D 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 00 4D 01", "59 06 10 00 4D", "FF FF FF FF FF");
if(MFK5.diagResp[5] != 0x27){
testStepPass("DTC 0x10004D is not active");
}
else{
testStepFail("DTC 0x10004D is active");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

Teststep("13.0","Wait for maturity");
testWaitForTimeout(3200);

TestStep("14.0-15.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF");
if(MFK5.diagResp[5] == 0x00 || MFK5.diagResp[5] == 0x50){
testStepPass("DTC 0x10026F is neither active nor passive");
}
else{
testStepFail("DTC 0x10026F is either active or passive");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

Teststep("16.0","Wait for t_diagStart");
testWaitForTimeout(2100);
TestStep("17.0-18.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
```

1. 使能CAN Bus OFF

2. 停止GNSS服务, 释放服务

3. 分别验证DTC 0x10004D和0x10026F状态

4. 解除CAN Bus OFF

5. 分别验证DTC 0x10004D和0x10026F状态

EthernetAPP_21377935

验证 **DTC (Diagnostic Trouble Code) 0x10026F** 在不同网络状态（NWDF信号）和服务启停条件下的状态变化，特别是检查该DTC是否会因GNSS服务停止、NWDF信号变化或超时而触发或恢复。

```
testcase EthernetAPP_21377935(char TestDescription[], char TestRequirements[], char TestParameters[])
{
…
    Teststep("3.0","Set NWDF to 0");
    setSignal(SI_NWDF_XIX_Systeminfo_01_XIX_HCP1_CANFD04,0);
  testWaitForSignalMatch(SI_NWDF_XIX_Systeminfo_01_XIX_HCP1_CANFD04,0,1000);


Teststep("4.0","Stop service GNSS");
@ME_CanoeControl_2::MSys_sifGnssInst1RmOffer_SET_BOL_2 = 0;

Teststep("5.0","Wait for maturity");
testWaitForTimeout(3200);

TestStep("6.0-7.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF");
if(MFK5.diagResp[5] == 0x00 || MFK5.diagResp[5] == 0x50){
testStepPass("DTC 0x10026F is neither active nor passive");
}
else{
testStepFail("DTC 0x10026F is either active or passive");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

    Teststep("8.0","Set NWDF to 1");
    setSignal(SI_NWDF_XIX_Systeminfo_01_XIX_HCP1_CANFD04,1);
    testWaitForSignalMatch(SI_NWDF_XIX_Systeminfo_01_XIX_HCP1_CANFD04,1,1000);

Teststep("9.0","Wait for maturity");
testWaitForTimeout(3200);

TestStep("10.0-11.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF");
if(MFK5.diagResp[5] == 0x00 || MFK5.diagResp[5] == 0x50){
testStepPass("DTC 0x10026F is neither active nor passive");
}
else{
testStepFail("DTC 0x10026F is either active or passive");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

Teststep("12.0","Wait for t_diagStart");
testWaitForTimeout(2100);

TestStep("13.0-14.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F 27", "FF FF FF FF FF FF");

Teststep("15.0","Start sending service GNSS");
@ME_CanoeControl_2::MSys_sifGnssInst1RmOffer_SET_BOL_2 = 1;

Teststep("16.0","Wait for dematurity");
testWaitForTimeout(11000);

TestStep("17.0-18.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F 26", "FF FF FF FF FF FF");

Teststep("19.0","Clear_DTC");
```

1. 强制NWDF网络睡眠
2. 禁用GNSS服务
3. 验证DTC状态（预期：不受NWDF影响）
4. 唤醒网络
5. 验证DTC状态
6. 启用GNSS服务
7. 读取DTC状态

EthernetAPP_21377933

验证全局通信错误（如CAN消息停止）与GNSS服务超时DTC的独立性

```
testcase EthernetAPP_21377933(char TestDescription[], char TestRequirements[], char TestParameters[])
{
…

    Teststep("3.0","create Global error");
@MEnv_StopCanMessageswithoutNM_SET_BOL_5 = 1; // enable disturbance
  testWaitForTimeout(3000);

TestStep("4.0-5.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 00 4E 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 00 4E 01", "59 06 10 00 4E 27", "FF FF FF FF FF FF ");

Teststep("6.0","Stop service GNSS");
    @ME_CanoeControl_2::MSys_sifGnssInst1RmOffer_SET_BOL_2 = 0;

Teststep("7.0","Wait for maturity ");
testWaitForTimeout(3200);

TestStep("8.0-9.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF");
if(MFK5.diagResp[5] == 0x00 || MFK5.diagResp[5] == 0x50){
testStepPass("DTC 0x10026F is neither active nor passive");
}
else{
testStepFail("DTC 0x10026F is either active or passive");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

    Teststep("10.0","Remove global error");
@MEnv_StopCanMessageswithoutNM_SET_BOL_5 = 0; // diasable disturbance
CAN_checkPDUonCANbusWithWaitTime("ClampControl_01_XIX_HCP1_CANFD04", 1, "ClampControl_01_XIX_HCP1_CANFD04", 500);

Teststep("13.0","Wait for maturity");
testWaitForTimeout(3200);
TestStep("14.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF");
if(MFK5.diagResp[5] == 0x00 || MFK5.diagResp[5] == 0x50){
testStepPass("DTC 0x10026F is neither active nor passive");
}
else{
testStepFail("DTC 0x10026F is either active or passive");
TestStep("","Status = %x",MFK5.diagResp[5]);
}

Teststep("16.0","Wait for t_diagStart");
testWaitForTimeout(2100);

TestStep("17.0-18.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F 27", "FF FF FF FF FF FF");

Teststep("19.0","Start sending service GNSS");
@ME_CanoeControl_2::MSys_sifGnssInst1RmOffer_SET_BOL_2 = 1;

Teststep("20.0","Wait for dematurity");
testWaitForTimeout(11000);

TestStep("21.0-22.0","Send Diagnostic Request(Physical) Read DTC Information with Raw Bytes 0x19 06 10 02 6F 01");
SendDiagRequestTS(MFK5, "Read Time out DTC", "19 06 10 02 6F 01", "59 06 10 02 6F", "FF FF FF FF FF FF");
```

1. 禁止所有CAN消息

2. 禁用GNSS服务

3. 验证DTC 0x10026F状态

4. 恢复所有CAN消息

5. 验证DTC 0x10026F状态

6. 启动GNSS服务

EthernetAPP_14277036

验证当GNSS服务不可用时，系统能正确记录DTC (0x10026F) 并在恢复后更新状态

```
testcase EthernetAPP_14277036(char TestDescription[], char TestRequirements[], char TestParameters[])
{
…


  ReportComment("---------------TEST SEQUENCE---------------");
  TestStep("1.0","Create fault such that SOMEIP_Service_GNSS is not available( no management message with an OfferServiceEntry was received for the server service instance)");
  @sysvar::IL_ETH1::NODES::N_MFK_5_SimulatedPeerNodes::PROVIDED_SERVICES::sif_24604_4_2_1::CONTROLS::Enable=0;


  TestStep("2.0", "Perform sleep wake up cycle and wait for 3s");
  @MEnv_SwitchOne_SET_INT_2=1;
  testWaitForTimeout(3000);

  TestStep("3.0-4.0", " Send Diagnostic Request(Physical) Read DTC information 0x19 06 10 02 6F 01");
  SendDiagRequestTS(MFK5, "Check routine ", "19 06 10 02 6F 01", "59 06 10 02 6F 2F 01 06", "FF FF FF FF FF FF FF FF");

  TestStep("5.0", "Remove fault by making SOMEIP_Service_GNSS is available(  management message with an OfferServiceEntry was received for the server service instance)");
  @sysvar::IL_ETH1::NODES::N_MFK_5_SimulatedPeerNodes::PROVIDED_SERVICES::sif_24604_4_2_1::CONTROLS::Enable=1;

  TestStep("6.0-7.0", "Wait for 10 secsn and Send Diagnostic Request(Physical) Read DTC information 0x19 06 10 02 6F 01");
  testWaitFortimeout(10000);
  SendDiagRequestTS(MFK5, "Check routine ", "19 06 10 02 6F 01", "59 06 10 02 6F 2E 01 06", "FF FF FF FF FF FF FF FF");

…
}
```

1. 模拟节点端 制造**SOME/IP 服务不可用** 的故障场景

2. 触发睡眠唤醒

3. 读取DTC状态（预期：0x2F表示故障）

4. 恢复服务后验证DTC状态（预期：0x2E表示恢复）

补充测试点（Deepseek）

分类	子类	建议补充的测试点描述	关联需求推测	推荐验证手段
服务发现	服务注册超时验证	验证DUT在服务提供方异常离线（如崩溃）后，能否在TTL超时后正确清除服务注册信息。	需求：SD模块应严格遵循TTL机制维护服务列表。	1. 模拟服务提供方崩溃 2. 监控DUT的SD缓存是否在TTL超时后删除对应OfferEntry。
	多实例服务发现	验证DUT能正确处理同一服务多个实例的Offer/StopOffer消息（如负载均衡场景）。	需求：支持多实例服务的动态发现与选择逻辑。	1. 启动多个GNSS服务实例 2. 检查DUT是否轮询或按优先级选择实例。
订阅机制	订阅重试逻辑验证	验证订阅失败（如网络抖动）时，DUT是否按配置的重试间隔和次数重新发起订阅。	需求：订阅需具备重试机制以保证可靠性。	1. 模拟网络丢包 2. 检查DUT的订阅重试日志及计数器。
	事件组过滤验证	验证DUT能正确解析SubscribeEventgroup中的事件组过滤参数（如仅订阅特定事件）。	需求：支持SOME/IP事件组过滤功能（AUTOSAR SWS_SOMEIP_00372）。	1. 发送带过滤参数的订阅请求 2. 验证DUT仅接收符合条件的事件通知。
DTC与诊断	服务降级与DTC关联	验证服务降级（如带宽不足时关闭GNSS）是否会触发合理的DTC（非故障类，如0x10026F）。	需求：服务降级与故障应区分处理，避免误报DTC。	1. 模拟带宽限制 2. 检查DTC状态（预期：不触发故障DTC，或触发专用降级DTC）。
	DTC冻结帧验证	验证GNSS服务超时DTC的冻结帧（Freeze Frame）是否记录关键上下文（如服务ID、时间戳）。	需求：DTC需记录足够上下文供诊断分析（ISO 14229-1）。	1. 触发DTC后读取冻结帧 2. 验证字段完整性（如Service ID=0x24604）。
异常处理	恶意报文注入防护	验证DUT对非法的SOME/IP-SD报文（如伪造OfferEntry）是否具备过滤或拒绝能力。	需求：安全模块需防御SOME/IP-SD欺骗攻击（ISO 21434）。	1. 注入伪造Offer报文 2. 检查DUT是否忽略或告警。
	服务状态冲突验证	验证当服务端同时调用Release和SetStatus时，DUT能否处理状态冲突（如先SetStatus再Release）。	需求：状态管理需保证原子性，避免竞态条件。	1. 并行触发两种操作 2. 检查最终状态是否符合预期（以Release为准）。
性能与压力	高负载订阅稳定性	验证DUT在同时订阅100+服务时的资源占用和响应稳定性（如内存泄漏、响应延迟）。	需求：高负载下仍需保证关键服务（如GNSS）的订阅可靠性。	1. 模拟大规模服务订阅 2. 监控DUT内存、CPU及GNSS订阅响应时间。
	频繁服务启停压力	验证短时间内重复启停GNSS服务（如每秒1次）是否导致DUT资源耗尽或订阅异常。	需求：支持服务的快速动态启停（如诊断刷写场景）。	1. 循环调用Release/OfferService 2. 检查订阅状态是否一致。
兼容性	协议版本兼容性	验证DUT与不同SOME/IP-SD协议版本（如v1.0/v1.1）的服务端交互兼容性。	需求：向后兼容旧版本协议（AUTOSAR SOME/IP-SD规范）。	1. 配置服务端使用旧版本协议 2. 验证DUT能否正常发现和订阅服务。
	多节点拓扑验证	验证DUT在复杂网络拓扑（如网关跨子网转发SOME/IP消息）下的服务发现和订阅功能。	需求：支持跨子网或网关转发的SOME/IP通信（如车载以太网多域架构）。	1. 部署网关转发GNSS服务 2. 验证DUT能否通过网关发现和订阅服务。


MAGNA
Forward. For all.