PAIO 사용 설명서

매뉴얼과 튜토리얼

2026-11-26

PopcornSAR

PAIO

PDF 다운로드 (한국어) Download PDF (English)

PAIO는 AI를 내장한 올인원 AUTOSAR / SDV(Software Defined Vehicle) 개발 IDE이다. 데스크톱 애플리케이션 하나에서 AUTOSAR **Classic Platform**과 **Adaptive Platform**

섹션	다루는 내용
제품 소개	PAIO 개요, 매뉴얼 표기 규칙, 제품 기본 사양
설치	설치와 로그인, 라이선스 상태
사용자 가이드	시작하기, 프로젝트(창 하나에 프로젝트 하나), 화면 구성
편집기	System 편집기(옛 IVC), Signal, VFB, Interface, DataType, SYSTEM 패널, ECUC, Diagnostic
작업 흐름	임포트 · 익스포트, 검증, AI 어시스턴트, 코드 · 빌드, 원격 개발, AI 워크플로우
레퍼런스	단축키, 용어 사전, 문제 해결 · FAQ

PAIO를 처음 쓴다면 제품 소개부터 읽고 설치로 넘어간다.

제품 소개

PAIO는 AI를 내장한 올인원 AUTOSAR / SDV(Software Defined Vehicle) 개발 IDE이다. 데스크톱 애플리케이션 하나에서 AUTOSAR **Classic Platform**과 **Adaptive Platform**

항목	값
제품 페이지	https://autosar.io/products/paio
공급사	PopcornSAR Co., Ltd.
적용 버전	**PAIO 1.0.1**
문서 작성 기준일	2026-11-11

이 문서는 PAIO 1.0.1의 제품 매뉴얼이다. 현재 버전에서 실제로 사용할 수 있는 기능과, 화면에 보이지만 동작이 다른 지점을 다룬다.

먼저 표기 규칙을 확인한 뒤 제품 개요와 제품 기본 사양을 읽는다.

표기 규칙

문서 전체에서 다음 표기 규칙을 사용한다.

- `이런 표기`는 파일 경로, 확장자, 코드 식별자, 열거형 값, 단축키를 뜻한다.
- **굵은 글씨**는 UI에 실제로 보이는 레이블이다. PAIO의 UI 문자열은 영문이므로 번역하지 않고 원문 그대로 적는다. 예: **Add ECU**, **Save All**, **New Project...**.
- 레이블은 UI가 그리는 모양 그대로 적는다. 메뉴 막대, 섹션 머리글, 패널 제목, 옆 패널의 필드 레이블은 대문자로 보이고(**PROJECT**, **DESIGN**, **SYSTEM**, **NAME**), 메뉴 항목 · 행 · **Project...**, **System**, **Create**, **Location**).
- 메뉴 경로는 `▸`로 잇는다. 예: `PROJECT ▸ Open Project...`.
- 단축키는 Windows/Linux 기준이며, macOS에서는 `Ctrl`을 `Cmd`로 읽는다.
- 인용문(>) 블록은 주의가 필요한 사실을 뜻한다. 특히 **레이블과 실제 동작이 다른 경우**나 **UI에는 있으나 동작하지 않는 항목**을 이 블록으로 표시한다.
- 오류 메시지는 앱이 실제로 출력하는 영문 원문을 그대로 인용한다.

제품 개요

이 장에서 다루는 것

PAIO가 어떤 제품인지, 무엇을 할 수 있는지, 데이터가 어떤 구조로 흐르는지 안내한다. 개별 기능의 조작 절차는 다루지 않고, 어느 장으로 가면 되는지만 알려 준다.

- 대상 독자: PAIO를 처음 접하는 사용자
- 선행 조건: 없음

수치, 버전, 지원 목록은 02. 제품 기본 사양에, 용어는 19. 용어 사전에 있다.

1.1 PAIO란

PAIO는 **AI를 내장한 올인원 AUTOSAR / SDV(Software Defined Vehicle) 개발 IDE**이다. Electron 기반 데스크톱 애플리케이션 하나에서 AUTOSAR **Classic Platform**과 **Adap**

기존에는 네트워크 설계, 시그널 DB, SWC 설계, BSW 설정, 코드 IDE를 각각 다른 도구로 쓰면서 ARXML 파일로 오갔다. PAIO는 이 과정을 하나의 프로젝트 창, 하나의 데이터 모델 우

PAIO는 **로그인해야 사용할 수 있다.** 로그인 전에는 로그인 화면만 보이고, 라이선스 검증이 끝나기 전까지 기능은 **잠금(읽기 전용)** 상태다. 03. 설치 · 로그인 · 라이선스를 참고한다.

먼저 알아 둘 여덟 개 용어

01~05장을 읽는 데는 **ECU**, **HPC**, **버스 / 클러스터**, **시그널**, **프레임**, **SWC**, **포트**, **ARXML** 여덟 개면 충분하다. 뜻은 19. 용어 사전에 있다.

1.2 무엇을 할 수 있나 — 기능 지도

모델링 편집기 (8종)

하려는 일로 편집기를 고른다.

하려는 일	여는 편집기
차량의 ECU와 그것을 잇는 버스를 그린다	**System**
그 버스로 오가는 데이터를 정의한다	**Signal**
소프트웨어를 컴포넌트로 나눈다	**VFB**
컴포넌트끼리 주고받을 데이터의 형식을 정한다	**Interface**, **DataType**
컴포넌트를 ECU나 HPC에 올린다	**System**(**SW Mapping**) 또는 **VFB**(**ECU/HPC Mapping**)
진단 데이터(DEXT)를 담는다	**Diagnostic**
BSW 설정값을 만든다	**Module Definition**, **Module Configuration**

1. 창 맨 왼쪽 아이콘 레일의 첫 번째 아이콘 **Design** 을 누른다. **EDITOR** 뷰가 열린다.
2. **DESIGN**(**System**, **VFB**, **Interface**, **DataType**, **Diagnostic**) 또는 **MODULE**(**Definition**, **Configuration**) 아래 행을 누른다. 편집기가 `<project>:<행 이름>` 탭으로 열
3. **Signal** 은 행이 없다. System 편집기 캔버스 왼쪽 가장자리의 **Signal** 아이콘으로 연다.

모델이 만들어지는 시점

새 프로젝트의 `design` 폴더에는 `IVC.io.model`, `SYSTEM.io.model`, `VFB.io.model` 만 있다. 나머지 모델 파일은 해당 편집기에서 처음 저장하거나 임포트가 그 내용을 만들

개발 환경

모델링 외에 내장 코드 IDE, 통합 터미널, 문서 편집, 원격 개발, vECU 뷰, ECU 빌드 · 실행, 검색 기능이 있다.

변환 · 검증 · 자동화

ARXML 임포트/익스포트, DBC 임포트, ECU / Machine Extract, PRTE / PARA 코드 생성, PRTE Platform Import / Auto Configuration, Validation, AI 채팅, AI 워크플로우로

1.3 설계 철학 — IO-First 아키텍처

PAIO의 가장 중요한 구조적 특징은 **AUTOSAR가 원본이 아니라는 것**이다.

- IO Model 이 1차 데이터 모델이다. 사람이 읽을 수 있는 텍스트로 저장되므로 형상 관리에 그대로 올릴 수 있다.
- ARXML과 DBC는 그 위에 얹힌 **변환 계층**이다.
- 편집기, AI, 검증은 모두 이 모델만 읽고 쓴다.
- **임포트 → 편집 → 익스포트 → 재임포트** 왕복을 지원하지만, 임의의 ARXML이 무손실로 왕복한다는 뜻은 아니다.

다이어그램 좌표는 ARXML로 나가지 않는다

노드 위치와 크기는 프로젝트 안에만 남는다. **익스포트한 ARXML에는 좌표가 없다.**

- 빈 프로젝트에 처음 임포트하면 그리드 위에 **자동 배치**된다.
- 다시 임포트하면 **옮겨 둔 좌표가 유지**된다. 노드를 옮긴 것은 충돌이 아니다.

1.4 편집 이력과 창 하나에 프로젝트 하나

- 편집 이력은 단일 타임라인 이 관리한다. **EDIT** 메뉴의 **Undo** 는 어느 편집기의 편집이든 되돌린다.
- 창 하나에는 프로젝트가 정확히 하나 열리고, 제목 표시줄에 `PAIO [<project>]` 가 표시된다.
- 다른 프로젝트를 만들거나 열면 자기 창으로 열리고 지금 창은 닫힌다. `PROJECT ▸ New Window`(`Ctrl+Shift+N`)는 빈 창을 하나 더 연다.

1.5 AUTOSAR 지원 범위

Classic Platform 과 Adaptive Platform 을 다룬다. 출력은 AUTOSAR **R25-11**(`AUTOSAR_00054.xsd`)로 고정된다. FlexRay는 모델과 변환 계층에서는 다루지만 System 편집

1.6 대표 작업 흐름

자주 하는 네 가지 작업이다.

A. 기존 ARXML로 시작하는 경우

1. 프로젝트를 만든다. **Welcome** 탭의 **New Project** 또는 `PROJECT ▸ New Project...` 에서 **Name** 을 입력하고 **Location** 을 정한 뒤 **Create** 를 누른다.
2. **DESIGN** 머리글의 **Import** 를 누르고 ARXML(또는 DBC) 파일을 고른다.
3. 충돌 해결 대화상자가 열리면 항목별로 남길 쪽을 고른다.
4. 각 편집기에서 결과를 확인한다.
5. **DESIGN** 머리글의 **Export** 를 누르고 형식을 고른 뒤 **Export** 를 누른다. 다른 폴더를 고르지 않으면 파일은 `<프로젝트>/export/` 에 만들어진다.

B. 백지에서 시작하는 경우

1. A처럼 프로젝트를 만든다.
2. **System 편집기**에서 ECU / HPC 노드, 커넥터, 컨트롤러, 클러스터를 배치한다.
3. **Signal 편집기**에서 프레임, PDU, 시그널을 정의한다.
4. **Interface / DataType 편집기**에서 포트 인터페이스와 데이터 타입을 정의한다.
5. **VFB 편집기**에서 컴포지션과 SWC를 배치하고 포트를 연결한 뒤, 각 컴포지션을 ECU나 HPC에 매핑한다.
6. 상태 표시줄의 **V** 아이콘으로 **VALIDATION** 패널을 열고 위반을 정리한다.
7. **DESIGN** 머리글의 **Export** 로 파일을 내보낸다.

C. AI로 초안을 만드는 경우

1. AI 채팅을 열고 모델을 자연어로 설명한다.
2. 각 편집기에서 결과를 검토하고 수정한다.

15. AI 어시스턴트를 참고한다.

D. 한 ECU 분량만 뽑아 협력사에 전달하는 경우

1. **System 편집기**에서 대상 노드를 우클릭한다.
2. **Extract ECU**(ECU 노드) 또는 **Extract Machine**(HPC 노드)을 고른다.
3. ARXML이 `<프로젝트>/export/<노드명>/` 아래에 만들어진다.

13. 임포트 · 익스포트를 참고한다.

제품 기본 사양

이 장에서 다루는 것

PAIO의 확정된 수치, 목록, 제약을 요약한다. 조작 절차는 다루지 않는다.

- 대상 독자: 도입 검토자, 문제를 좁혀 가는 사용자
- 선행 조건: 없음

용어는 19. 용어 사전에 있다.

2.1 제품 식별 정보

PAIO 버전 1.0.1이며, PopcornSAR PAIO License에 따른 상용 제품이다.

2.2 지원 플랫폼과 설치 패키지

OS	아키텍처	배포 형식
Windows	x64	NSIS 설치 실행 파일, ZIP
macOS	x64, arm64 (Apple Silicon)	DMG, ZIP
Linux	x64	DEB, tar.gz

Windows 설치 프로그램(NSIS) 동작

설치 프로그램은 마법사 형태로 진행되고, 사용자 단위 설치가 기본이며, 설치 경로를 바꿀 수 있다. 제거해도 앱 데이터는 남는다.

macOS · Linux

- macOS 빌드는 코드 서명과 공증(notarize)을 하지 않는다.
- Linux에서 실행하는 프로그램은 런처 `paio` 이다. 비특권 사용자 네임스페이스를 허용하지 않는 시스템에서 `.tar.gz` 압축 해제본은 그 폴더에서 `sudo ./install.sh` 를 다. `.deb` 는 샌드박스를 스스로 설정한다.

2.3 실행 환경

창 하나는 프로세스 하나이자 프로젝트 하나다. 다른 PAIO 창이 이미 연 프로젝트 폴더를 열면 그 창에 포커스가 간다. 편집기 탭은 동시에 최대 **20개**까지 열린다.

2.4 기술 스택

PAIO는 Electron 기반 데스크톱 앱이다. 코드 편집은 Monaco, 터미널은 xterm을 쓴다.

2.5 AUTOSAR 지원 사양

스키마 버전

모든 익스포트는 AUTOSAR **R25-11** 스키마(`AUTOSAR_00054.xsd`)로 나간다.

지원 프로토콜

System 편집기에서는 CAN / CAN FD, LIN, Ethernet을 추가할 수 있고 FlexRay는 추가 메뉴가 없다. Signal 편집기의 탭은 **CAN**, **LIN**, **PDU**, **SIGNAL** 네 개다.

> **FlexRay 요소는 System 편집기에서 사라진다.** System 편집기는 FlexRay 요소를 그리지 않고, 저장할 때 화면에서 IVC 모델을 다시 만들기 때문에 그 요소가 파일에서 없어진다 집기에서 저장하지 않는다.

입출력 포맷

- 임포트: ARXML, DBC, ECUC Configuration ARXML.
- 익스포트: ARXML(R25-11)과 DBC(CAN 클러스터마다 파일 하나)를 기본으로 `<프로젝트>/export/` 에 쓰며, ECU / Machine Extract ARXML도 만든다.

프로젝트 익스포트가 만드는 파일은 최대 12종이다. 조건을 만족하지 못한 파일은 만들어지지 않으므로 12개보다 적은 것은 정상이다.

2.6 데이터 모델 사양

프로젝트에는 고정 폴더 6개(`doc`, `design`, `code`, `workflow`, `export`, `import`)가 있다. 새 프로젝트의 `design` 폴더에는 `IVC.io.model`, `SYSTEM.io.model`, `VFB.io` **EDITOR** 뷰에서 연다. 04. 워크스페이스와 프로젝트도 참고한다.

2.7 AI 프로바이더 사양

AI 채팅의 프로바이더는 **Claude [OPUS]**, **Claude [GLM]**, **Gemini**, **Codex**, **Ollama** 5종이며 이 순서로 표시된다. **Claude [GLM]** 은 이름과 달리 Anthropic Claude가 아니라 Zhi 만 쓰이고, AI 워크플로우는 이들을 쓰지 않는다(2.13).

2.8 검증(Validation) 사양

등록된 규칙은 **429개**이다. 검증은 프로젝트가 열릴 때와 편집 뒤 손을 멈추면 곧바로 실행된다. 상태바 배지가 위반 개수를 표시한다.

2.9 라이선스와 기능 제한

로그인과 라이선스 검증이 끝나기 전까지 PAIO는 **잠금(읽기 전용)** 상태다. 만료된 라이선스는 **읽기와 임포트만 허용**한다. 익스포트는 온라인 검증을 요구한다. 절차는 03. 설치 · 로그

2.10 보안 사양

무결성 검사 결과

서명이 유효하지 않은 설치본에서는 편집 기능이 세션 동안 읽기 전용으로 잠긴다. 앱은 종료되지 않는다. 20. 문제 해결 · FAQ 20.1과 03. 설치 · 로그인 · 라이선스 3.1.5절을 참고한다.

2.11 업데이트 알림 사양

Windows와 `.deb` 패키지로 설치한 Linux에서는 업데이트를 선택했을 때에만 내려받아 설치한다. macOS와 `tar.gz` 로 쓰는 Linux에서는 알리기만 하고, 요청하면 다운로드 페 인한다. 3.5도 참고한다.

2.12 알려진 제약 사항

용량 · 범위 제약

편집기 탭을 20개 연 뒤 새 파일을 열면 **아무 안내 없이 열리지 않는다**. 탭을 먼저 닫는다.

동작하지 않거나 제공되지 않는 항목

이 버전에서는 DBC 인코딩 선택, Signal 편집기의 **FlexRay** / **Ethernet** 탭, System 편집기의 FlexRay, QuickStart 워크플로가 동작하지 않거나 제공되지 않는다.

그 밖에 알아 둘 동작

문제로 오해하기 쉬운 정상 동작이 있다. 임포트는 저장하지 않은 편집을 먼저 저장하며, 내보낸 ARXML은 화면 배치를 복원하지 않는다.

2.13 AI 워크플로우 사양

AI 워크플로우의 AI 노드는 PAIO에 번들된 Claude Code 런타임으로 Claude를 실행하며, AI 채팅 프로바이더와는 별개다. 워크플로 문서는 프로젝트당 하나, `workflow/WORKFLO`

설치

PAIO를 설치하고 이메일 일회용 코드로 로그인한 뒤, 라이선스 상태를 확인하고 변경하는 방법을 다룬다.

설치 · 로그인 · 라이선스 문서를 참고한다.

설치 · 로그인 · 라이선스

이 장에서 다루는 것

PAIO를 설치하고 로그인하는 방법과, 라이선스 상태에 따라 어떤 기능이 열리고 막히는지를 설명한다. 업데이트 알림, 앱 버전, 이슈 신고도 다룬다. 대상 독자는 PAIO를 처음 설치하는 받은 설치 파일과 Google 계정 또는 이메일 주소가 필요하다. 별도의 계정 신청은 없다. 처음 로그인하면 계정이 만들어지고 체험판으로 시작한다.

설치 패키지 이름과 지원 플랫폼은 02. 제품 기본 사양에 있다.

3.1 설치

3.1.1 Windows — 설치 프로그램(NSIS)

1. `PAIO-win-x64.exe` 를 실행한다. 사용자 단위로 설치되므로 관리자 권한을 요구하지 않는다.
2. 이미 설치되어 있으면 **Program Maintenance** 화면이 나타난다. **Modify** 와 **Repair** 는 일반 설치 절차로 진행하고, **Remove** 는 제거 프로그램을 실행한 뒤 종료한다.
3. **I agree to the License Agreement** 를 선택한다.
4. 설치 폴더를 그대로 두거나 바꾼다(기본값 `%LOCALAPPDATA%\Programs\PAIO`).
5. 설치를 마친다. **Create a desktop icon** 은 기본으로 켜져 있다.

제거해도 사용자 데이터는 지워지지 않는다.

3.1.2 Windows — 설치 없이 사용(ZIP)

`PAIO-win-x64.zip` 을 원하는 위치에 풀고 `PAIO.exe` 를 실행한다. 바로가기와 시작 메뉴 항목은 만들어지지 않는다.

3.1.3 macOS

1. `PAIO-mac-x64.dmg`(Intel) 또는 `PAIO-mac-arm64.dmg`(Apple Silicon)를 열고 PAIO를 Applications 폴더로 끌어다 놓는다.
2. 첫 실행 때 Gatekeeper 경고가 나타난다. 시스템 설정 ▸ 개인정보 보호 및 보안에서 실행을 허용한다. 손상되어 열 수 없다고 나오면 `xattr -dr com.apple.quarantine`

3.1.4 Linux

- **Debian/Ubuntu**: `sudo apt install ./PAIO-linux-x64.deb` 로 설치한다. 패키지가 아이콘을 등록하고 Chromium 샌드박스를 설정한다.
- **tar.gz**: `PAIO-linux-x64.tar.gz` 를 풀고 그 폴더의 `paio` 를 실행한다. `.paio-bin` 을 직접 실행하지 않는다.

3.1.5 설치본 무결성 검사 (Windows 전용)

PAIO는 실행 직후 백그라운드에서 `PAIO.exe` 의 코드 서명을 검사한다. 실행 파일이 변조되었거나 다른 서명자로 재서명되었으면 편집 기능이 세션 동안 읽기 전용으로 고정되고 다

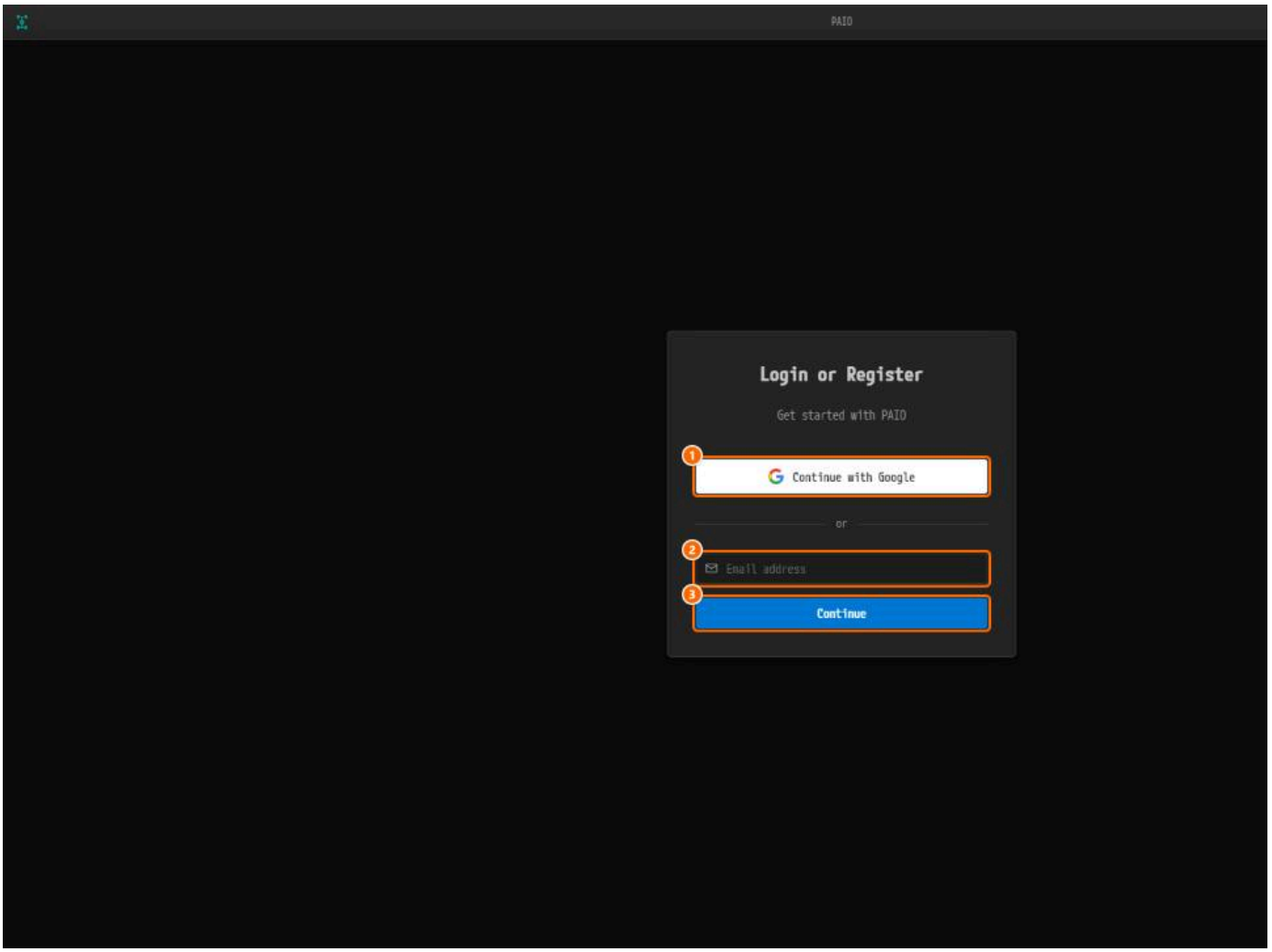
> `PAIO could not verify the integrity of its installation.` `Please reinstall PAIO. Editing is disabled for this session.`

이 잠금은 라이선스가 정상이어도 풀리지 않는다. 정상 배포본을 다시 설치한다. 진단 파일은 20. 문제 해결 · FAQ 20.1에 있다.

3.2 첫 실행 — 로그인

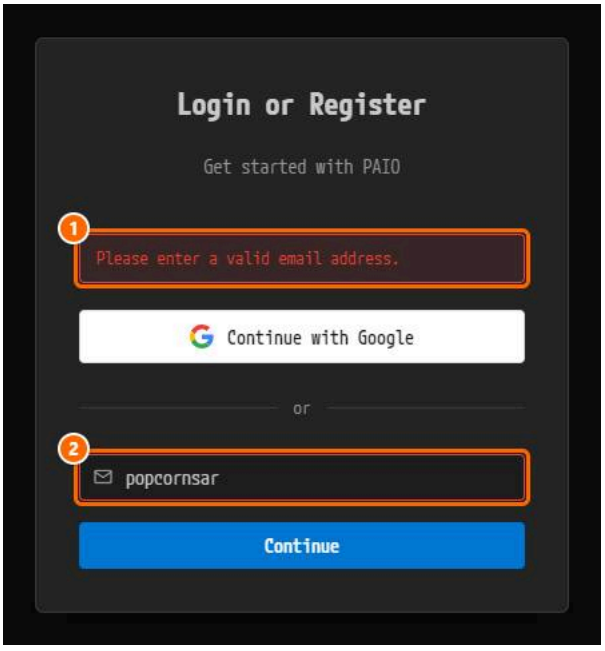
PAIO는 로그인한 뒤에야 메인 화면을 보여 준다. 비밀번호는 쓰지 않으며, Google 계정이나 이메일 일회용 코드(OTP)로 로그인한다.

3.2.1 로그인 화면



*로그인 화면. ① **Continue with Google**, ② 이메일 입력란, ③ **Continue**.*

이메일이 올바르지 않으면 `Please enter a valid email address.` 가 표시되고 아무것도 전송되지 않는다.



`popcornsar` *로 **Continue** 를 눌렀을 때. ① 오류 문구, ② 테두리가 붉게 바뀐 입력란.*

3.2.2 방법 1 — Google 계정

1. **Continue with Google** 을 누른다. 시스템 브라우저가 Google 로그인 페이지를 연다.
2. 브라우저에서 로그인을 마친다. PAIO가 로그인을 마치며, 붙여 넣을 것은 없다.

기다리기를 그만두려면 **Cancel** 을 누른다. **5분**이 지나면 PAIO는 기다리기를 멈추므로 **Continue with Google** 을 다시 누른다.

3.2.3 방법 2 — 이메일 일회용 코드(OTP)

1. 이메일 주소를 입력하고 **Continue** 를 누른다.
2. **Enter your code** 화면에서 메일로 받은 6자리 코드를 입력하고 **Continue** 를 누른다. 코드는 **10분** 동안 한 번만 쓸 수 있다.
3. 코드가 오지 않으면 **Resend code** 를 누른다(60초 뒤 활성화).

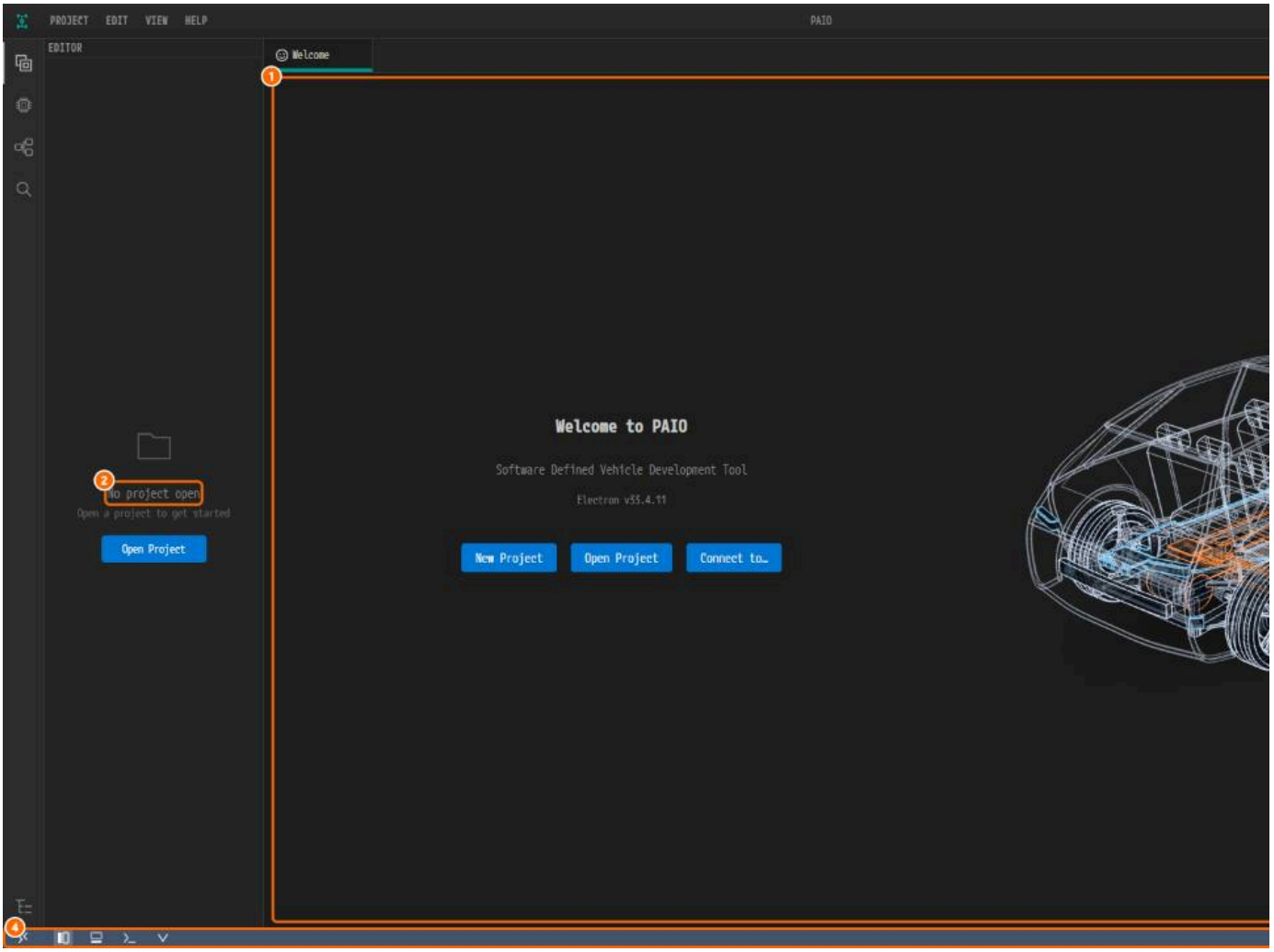


*Enter your code 화면(예시 주소). ① 주소, ② 코드 입력란, ③ **Continue**, ④ **Resend code** 카운트다운, ⑤ **Back to login**.*

코드가 틀리거나 만료되었으면 `Invalid or expired code` 가 표시된다.

3.2.4 로그인 후 처음 보이는 창

로그인하면 PAIO는 마지막으로 작업한 프로젝트를 그 폴더가 남아 있는 한 다시 연다. 그렇지 않으면 프로젝트가 열려 있지 않은 창이 된다.



*프로젝트가 열려 있지 않은 창. ① **Welcome** 탭, ② **EDITOR** 뷰의 `No project open`, ③ 상단바 오른쪽 버튼들, ④ 상태바.*

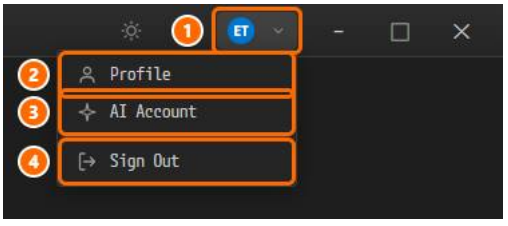
프로젝트를 만들고 여는 방법은 04. 워크스페이스와 프로젝트에서 다룬다.

3.2.5 로그인에 실패했을 때

로그인에 실패하면 카드에 `Can't reach the PAIO server (pars.popcornsar.com). Check your network or proxy.` 같은 한 줄이 표시된다. 회사 네트워크라면

3.3 사용자 메뉴와 프로필

상단바 오른쪽의 사용자 아바타를 누르면 **Profile**, **AI Account**, **Sign Out** 이 나온다.



*사용자 메뉴. ① 아바타, ② **Profile**, ③ **AI Account**, ④ **Sign Out**.*

- **Profile** 은 계정 이메일, 현재 플랜과 상태를 보여 준다. 편집할 수 있는 값은 없다. **Subscribe** 는 가격 페이지를 연다.
- **AI Account** 는 AI 워크플로우의 AI 노드가 쓰는 Claude 계정을 연결한다(21.11). AI 채팅 프로바이더는 채팅 안에서 설정한다(15.2).
- **Sign Out** 은 저장하지 않은 탭이 있을 때만 먼저 묻는다.



*체험판의 Profile 화면(예시 주소). ① **Account**, ② **Current Plan**, ③ **Subscribe**, ④ **Sign Out**, ⑤ 닫기 버튼.*

3.4 라이선스와 기능 제한

3.4.1 플랜과 상태

권한은 플랜이 아니라 라이선스 상태로 결정된다. 제한 상태는 `expired` 와 `readonly` 뿐이다. 체험 기간은 **앱에서 처음 로그인한 시점부터 30일**이다.

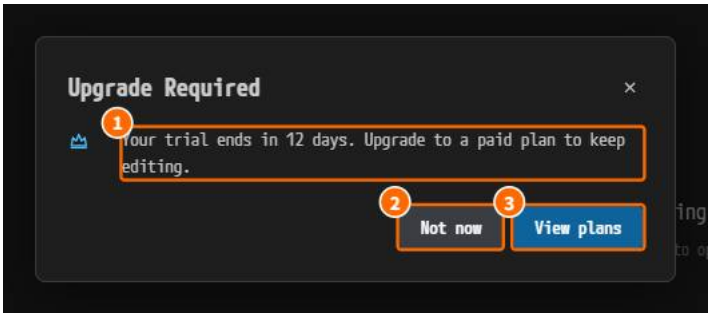
3.4.2 권한 표

상태	읽기	편집(Modify)	생성(Create)	삭제(Delete)	익스포트(Export)	임포트(Import)
`trial` / `active`	O	O	O	O	O	O
`expired` / `readonly`	O	X	X	X	X	O

임포트는 어떤 상태에서도 허용된다. 제한 상태에서는 AI 어시스턴트를 쓸 수 없다.

3.4.3 권한이 없을 때 보이는 화면

권한 없이 동작을 시도하면 **Upgrade Required** 다이얼로그가 열린다. **View plans** 는 가격 페이지를 열고, **Not now** 는 다이얼로그를 닫는다.



*체험 기간이 12일 남았을 때의 **Upgrade Required** 다이얼로그. ① 안내 문구, ② **Not now**, ③ **View plans**.*

> **구매는 웹사이트에서 한다.** 앱에는 결제 화면도, 라이선스 키 입력란도 없다. 구독하거나 갱신한 뒤 PAIO로 돌아오면, 창이 포커스를 얻을 때 새 플랜이 반영된다.

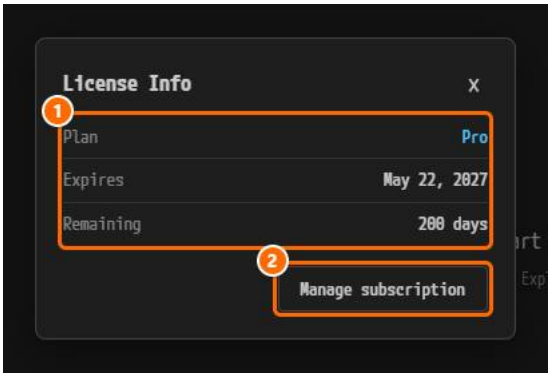
3.4.4 라이선스 상태 확인

라이선스는 상태바 오른쪽에 `Trial: 12d` 나 **Pro** 처럼 표시된다.



① 라이선스 표시. 그림은 Pro 라이선스다.

표시를 클릭한다. 체험판이면 **Upgrade Required** 다이얼로그가, 그 밖의 플랜이면 **Plan**, **Expires**, **Remaining** 을 보여 주는 **License Info** 다이얼로그가 열린다.



*Pro 라이선스의 **License Info** 다이얼로그. ① **Plan**, **Expires**, **Remaining**, ② **Manage subscription**.*

3.4.5 오프라인 동작과 익스포트

라이선스는 **4시간** 동안 캐시되므로 그동안은 오프라인에서도 작업할 수 있다. 익스포트는 실행할 때마다 라이선스를 다시 확인한다. 예를 들어 서버에 연결할 수 없고 쓸 수 있는 캐시!

3.4.6 세션 무효화

계정 하나는 한 번에 **기기 하나**에서만 쓸 수 있다. 다른 기기에서 로그인하면 이 기기는 다음 라이선스 재확인 때, 예를 들어 창을 클릭했을 때 로그아웃된다. 저장하지 않은 탭이 있으면

3.5 업데이트 알림

PAIO는 백그라운드에서 새 버전을 확인하고 오른쪽 아래에 **Software Update** 토스트로 알린다. 직접 확인하려면 **HELP** ▸ **Check for Updates** 를 고른다.

- Windows에서는 **Update** 를 누른다. PAIO가 다른 창과 실행 중인 작업이 끝나기를 기다린 뒤 업데이트를 내려받아 설치하고 다시 시작한다. 설치하는 동안 약 5분간 창이 사라짐
- Linux에서 `.deb` 패키지로 설치했으면 **Update** 가 똑같이 동작하며, 설치 전에 관리자 암호를 묻는다. 그때 다른 프로그램이 소프트웨어를 설치하고 있으면 PAIO 는 최대 2분 기 **Update** 를 다시 보여 주므로, 다른 설치가 끝난 뒤 다시 누른다.
- macOS와 `tar.gz` 로 쓰는 Linux에서는 다운로드 페이지로 안내하므로, 3.1의 절차대로 새 버전을 설치한다.

20.12도 참고한다.

3.6 앱 정보 확인

HELP ▸ **About PAIO** 를 고르면 버전을 볼 수 있다. 버전은 상태바 맨 오른쪽에도 `v<버전>` 으로 표시된다.

3.7 문의 / 이슈 신고

HELP ▸ **Report Issue** 를 고른다. 지원 Q&A 페이지(`https://autosar.io/support/qna`)가 브라우저로 열린다.

3.8 계정과 라이선스 데이터가 저장되는 위치

인증 토큰, 라이선스 캐시, 기기 식별자, AI API 키는 기기마다 한 곳에 저장되며, 토큰과 캐시와 키는 암호화된다. 그 밖의 앱 데이터는 프로젝트 폴더별로 저장된다. OS 보안 저장소를 사용하며, Linux에서는 `libsecret` 을 설치해야 한다.

사용자 가이드

PAIO에서 작업을 시작하는 방법을 다룬다. PAIO 창 하나는 프로젝트 하나를 연다. 창이 연 폴더가 곧 프로젝트이며, 제목 표시줄에 그 이름이 `PAIO [<project>]` 로 보인다.

문서	다루는 내용
시작하기	첫 실행, 프로젝트 만들기, 첫 편집기 열기
워크스페이스와 프로젝트	프로젝트를 만들고 여는 방법, 창과 프로젝트 폴더의 관계
화면 구성	제목 표시줄, 아이콘 막대, **EDITOR** 뷰, 탭, 상태 표시줄

시작하기

PAIO를 막 설치한 상태에서 System 편집기를 열기까지의 최소 절차를 설명한다.

설치와 로그인

설치 · 로그인 · 라이선스에 따라 PAIO를 설치하고 로그인한다.

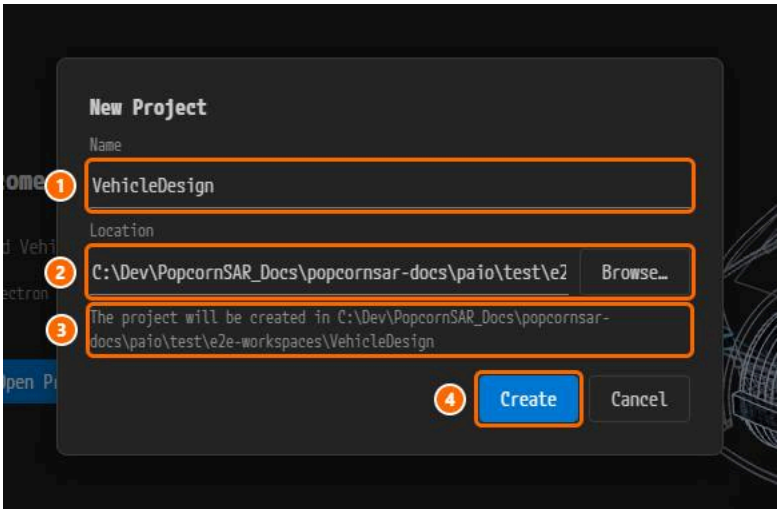
프로젝트가 열려 있지 않은 창

프로젝트가 열려 있지 않으면 제목 표시줄에 `PAIO` 만 보이고, 편집 영역에 **Welcome** 탭(`Welcome to PAIO`)이 열린다. 이 탭에는 **New Project**, **Open Project**, **Connect to...** 과 **Open Project** 버튼이 보인다.

같은 명령은 **PROJECT** 메뉴에도 있다. **New Project...**, **Open Project...**(`Ctrl+O`), **Open Recent**, **New Window**(`Ctrl+Shift+N`), **Save**(`Ctrl+S`), **Save All**, **Exit**(`Alt+F4`

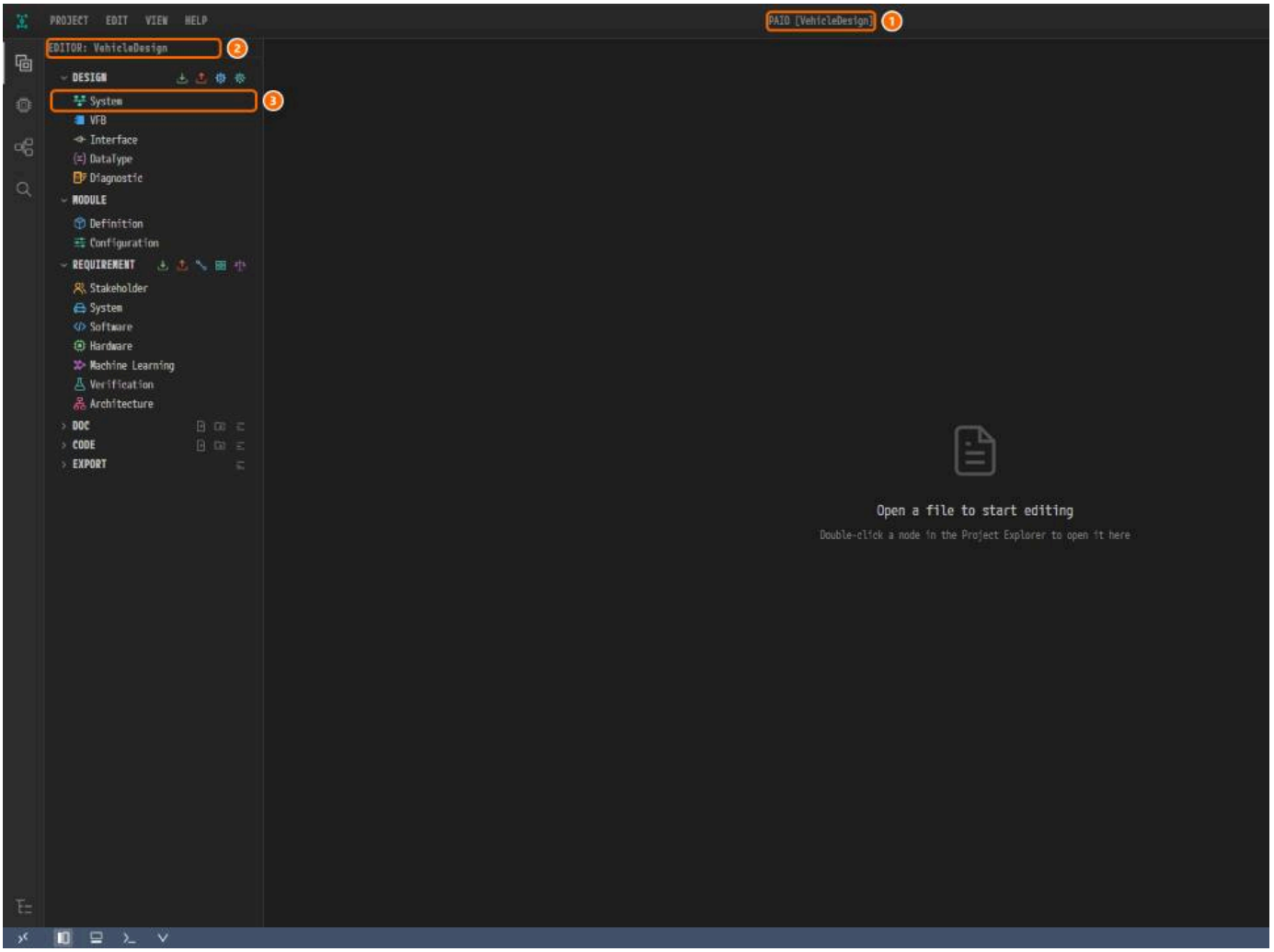
프로젝트 만들기

1. **Welcome** 탭에서 **New Project** 를 클릭한다(또는 `PROJECT ▸ New Project...` 를 고른다).
2. **New Project** 대화상자의 **Name** 에 프로젝트 이름을 입력한다.
3. **Location** 에 프로젝트를 담을 폴더를 지정하거나 **Browse...** 를 클릭해 고른다.
4. 입력란 아래 줄 `The project will be created in <Location>\<Name>` 을 확인하고 **Create** 를 클릭한다.



*새 프로젝트를 만드는 **New Project** 대화상자. ① **Name**, ② **Location** 과 **Browse...** 버튼, ③ 새 프로젝트 폴더를 알려 주는 줄, ④ **Create** 버튼.*

PAIO는 `<Location>\<Name>` 폴더를 만들고 그 폴더를 창의 프로젝트로 연다. 제목 표시줄이 `PAIO [<Name>]` 으로 바뀌고, **EDITOR** 뷰 머리글이 `EDITOR: <Name>` 이 되며 **D**



*`VehicleDesign` 프로젝트를 **Create** 한 직후의 창. ① 제목 표시줄 `PAIO [VehicleDesign]`, ② 뷰 머리글 `EDITOR: VehicleDesign`, ③ **DESIGN** 아래의 **System**. 편집기 `editing` 만 보인다.*

첫 편집기 열기

EDITOR 뷰의 **DESIGN** 아래에서 **System** 을 클릭한다. System 편집기가 탭 `<Name>:System` 으로 열린다. 다른 설계 편집기도 **DESIGN** 과 **MODULE** 아래의 행에서 같은 방법으로

다음 단계

- 워크스페이스와 프로젝트: 프로젝트를 만들고 여는 방법을 자세히 다룬다.
- 화면 구성: 애플리케이션 창의 각 부분을 다룬다.

워크스페이스와 프로젝트

이 장에서 다루는 것

이 장은 PAIO가 작업을 담는 방식을 설명한다. 창 하나는 폴더 하나를 열고, 그 폴더가 곧 프로젝트다. 프로젝트를 만들고 여는 방법, 프로젝트의 폴더, **EDITOR** 뷰, **Explorer**, 저장, 편집하는 사용자다. 선행 조건은 로그인과 활성 라이선스이며(03. 설치 · 로그인 · 라이선스), 만료 상태에서는 이 장의 생성, 삭제, 이름 변경, 저장이 모두 막힌다.

4.1 개념 정리

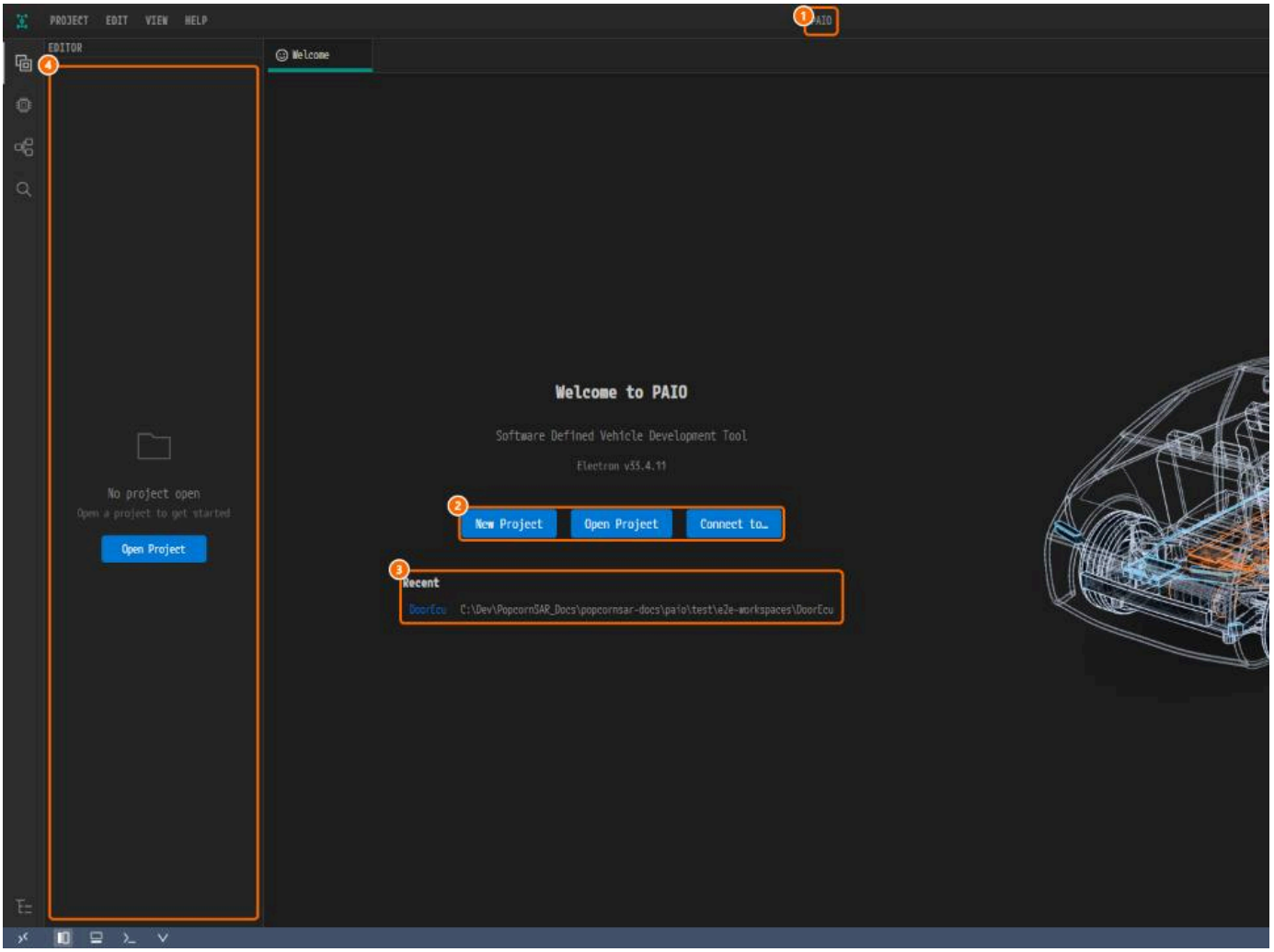
PAIO에서 워크스페이스와 프로젝트는 같은 것이다. 창이 연 폴더가 그대로 프로젝트다.

```
DoorEcu/                ← 창이 연 폴더 = 프로젝트
 ├── DoorEcu.io.prj     ← 프로젝트 메타 파일 (Explorer 에는 보이지 않는다)
 ├── code/  design/  doc/  export/  import/  workflow/
 └── ...
```

- **창 하나에 프로젝트 하나.** 다른 프로젝트를 다루려면 다른 창에서 연다(`PROJECT ▸ New Window`, `Ctrl+Shift+N`).
- **폴더 이름이 곧 프로젝트 이름이다.**
- **폴더를 열면 그 폴더가 프로젝트가 된다.** 메타 파일이 없으면 PAIO가 묻지 않고 메타 파일, 고정 폴더, 기본 모델을 추가한다. 이미 있는 것은 덮어쓰지 않는다.

4.2 프로젝트 열기

프로젝트가 열려 있지 않은 창은 제목이 `PAIO` 이고 **Welcome** 탭을 보여 준다.



1. **PROJECT ▸ Open Project...**(`Ctrl+O`)를 고르거나 **Welcome** 탭의 **Open Project** 를 클릭한다. 최근 프로젝트는 **Open Recent** 나 **Recent** 목록에서 연다.

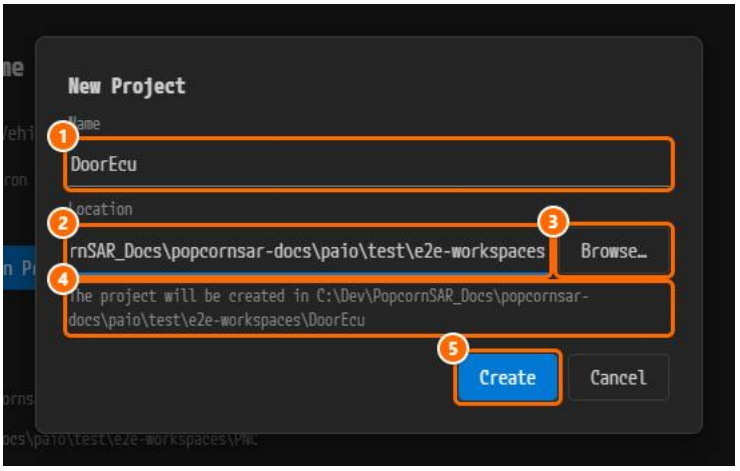
2. 폴더 선택 창에서 프로젝트 폴더를 고른다.

창 제목이 `PAIO [<프로젝트>]` 로 바뀌고 사이드바에 `EDITOR: <프로젝트>` 가 표시된다. 이미 프로젝트가 열린 창에서는 고른 폴더가 새 창으로 열리고 현재 창은 닫힌다. 원격 연

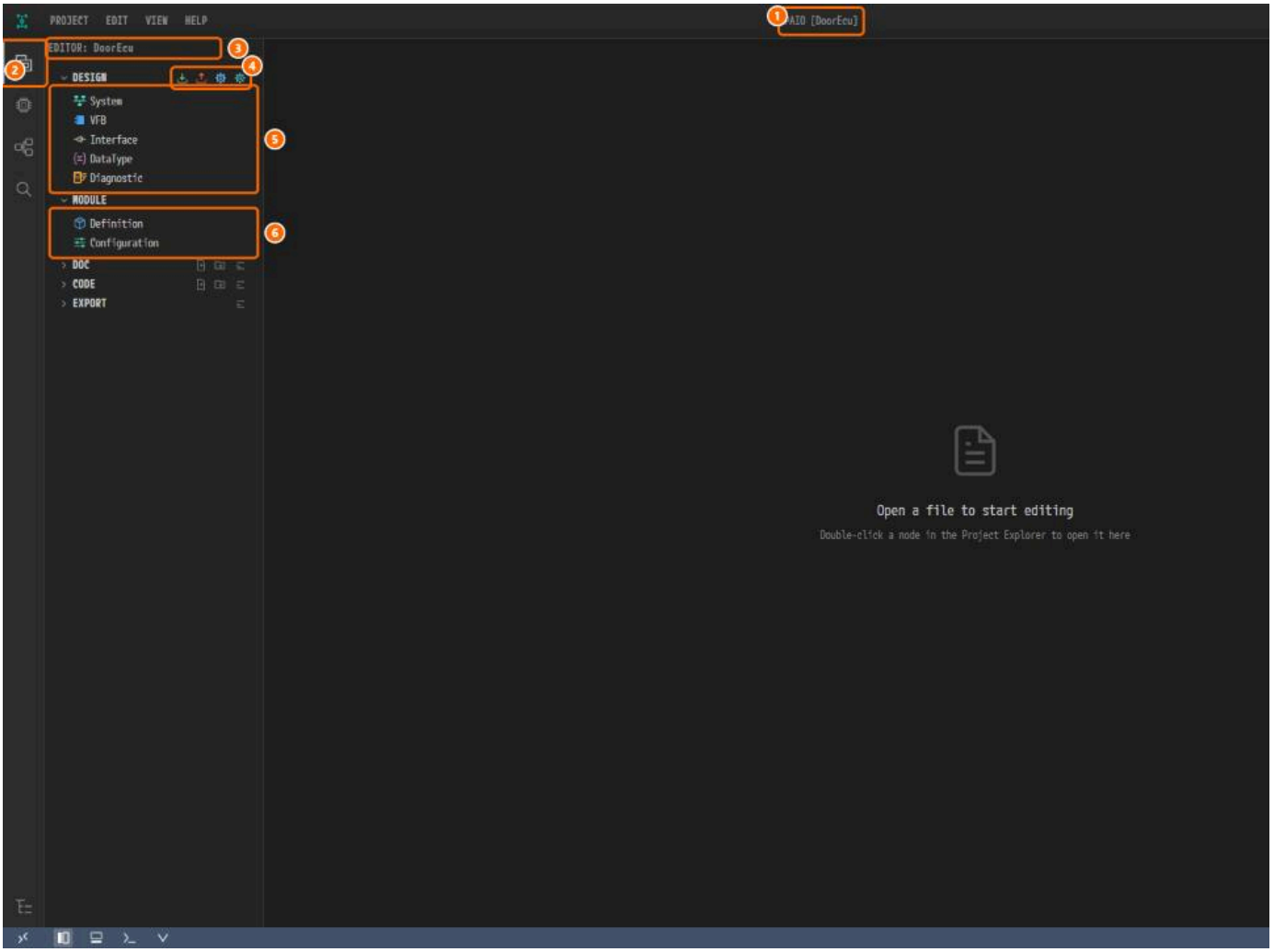
4.3 프로젝트 만들기

4.3.1 절차

1. **Welcome** 탭에서 **New Project** 를 클릭하거나 `PROJECT ▸ New Project...` 를 실행한다.
2. **Name** 에 프로젝트 이름을 입력한다.
3. 프로젝트를 만들 폴더인 **Location** 을 지정한다. 경로를 입력하거나 **Browse…** 를 클릭한다.
4. 입력란 아래 줄 `The project will be created in <Location>\<Name>` 을 확인한다.
5. **Create** 를 클릭한다.



창 제목이 `PAIO [<Name>]`, 사이드바가 `EDITOR: <Name>` 이 되고 **DESIGN**, **MODULE** 이 펼쳐진다. **DESIGN** 아래의 **System** 을 클릭하면 새로 만든 빈 시스템의 System 편집기



이미 프로젝트가 열린 창에서는 새 프로젝트가 새 창으로 열리고 현재 창은 닫힌다.

4.3.2 이름 규칙

영문자, 숫자, 하이픈, 밑줄만 쓴다. 예: `DoorEcu`, `body-domain`, `door_ecu_01`. Windows 예약어와, `<Location>\<Name>` 에 이미 있는 프로젝트는 거부된다.

4.3.3 생성되는 것

새 프로젝트에는 메타 파일 `<Name>.io.prj`, 고정 폴더 6개(`code`, `design`, `doc`, `export`, `import`, `workflow`), `design` 안의 기본 모델 3개(`SYSTEM.io.model`, `VFB.` 와 모델은 덮어쓰지 않는다.

4.3.4 Quick Start 프로젝트

PROJECT 메뉴에는 더 이상 **Quick Start** 가 없다. 프로젝트는 모두 **New Project** 로 만든다(4.3.1).

4.4 프로젝트 폴더 구조

폴더	용도
`code/`	C/C++/Rust 소스와 빌드 파일
`design/`	모델 파일(`*.io.model`)과 하위 폴더 `SWC/`, `Module/`
`doc/`	문서
`export/`	익스포트 결과
`import/`	임포트 원본을 두는 곳. ARXML · DBC 임포트는 원본을 여기로 복사하지 않는다
`workflow/`	AI 워크플로우의 파일

`code`, `design`, `doc`, `export`, `workflow` 는 이름을 바꾸거나 삭제할 수 없다.

4.5 design 폴더와 EDITOR 뷰

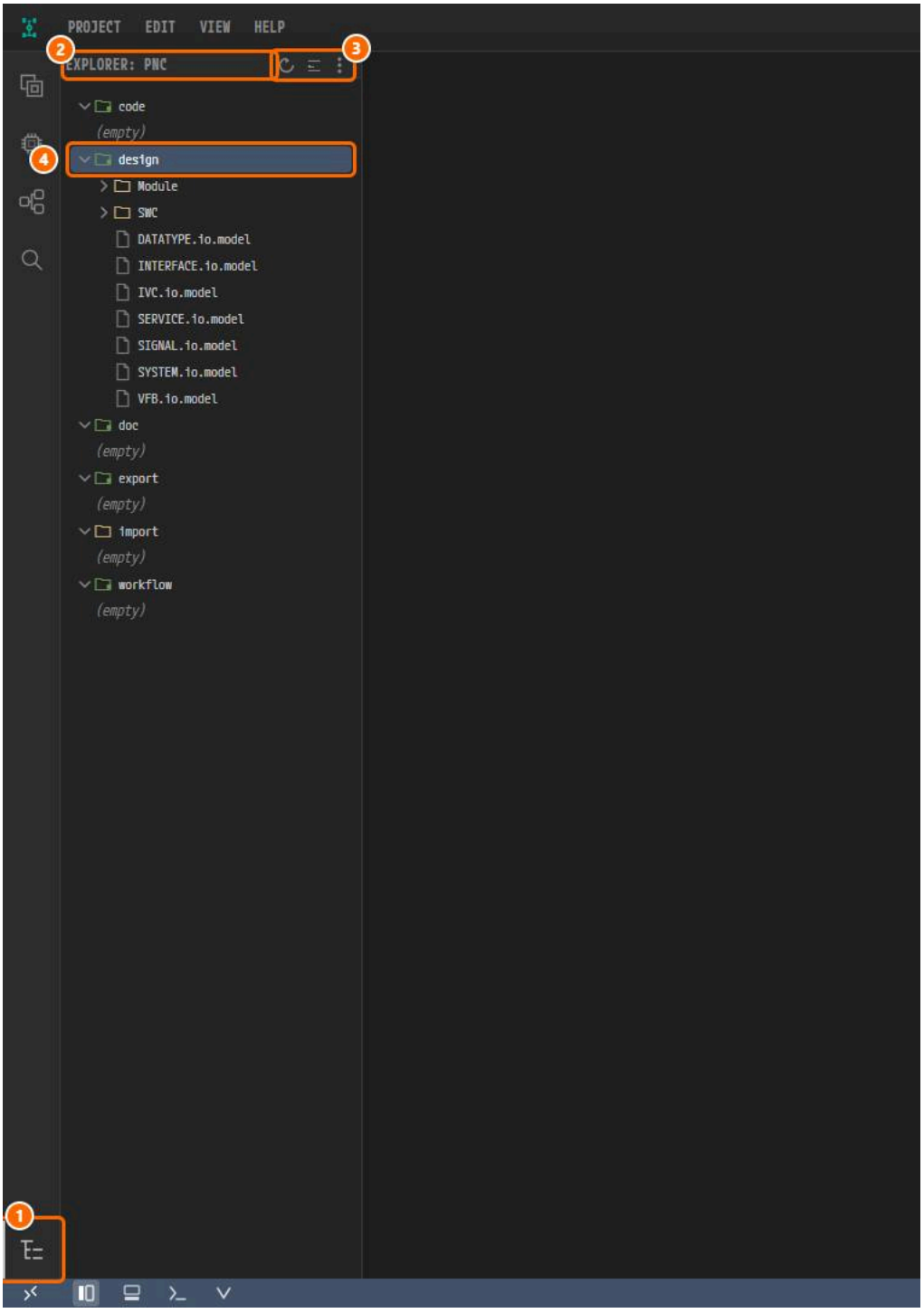
설계 편집기는 Explorer가 아니라 **EDITOR** 뷰에서 연다.

1. **DESIGN**(**System**, **VFB**, **Interface**, **DataType**, **Diagnostic**)이나 **MODULE**(**Definition**, **Configuration**) 아래의 행을 클릭한다.
2. 편집기가 `<프로젝트>:<행>` 이름의 탭으로 열린다. 예: `DoorEcu:System`.

- **Signal 에는 행이 없다.** Signal 편집기는 System 편집기 캔버스의 **Signal** 아이콘으로 연다(06. System 에디터, 07. Signal 에디터).
- **DESIGN** 헤더의 **Import**, **Export** 아이콘은 13. 임포트 · 익스포트에서, **MODULE** 은 12. ECUC에서 다룬다.
- Explorer에서 `*.io.model` 파일을 더블클릭하면 설계 편집기가 아니라 읽기 전용 JSON으로 열린다.

4.6 Explorer 조작

레일 맨 아래의 **Explorer** 아이콘을 클릭한다. 트리는 프로젝트 행 없이 프로젝트 폴더의 내용을 바로 나열한다.



4.6.1 선택

클릭으로 선택하고, `Ctrl` + 클릭이나 `Shift` + 클릭으로 여러 개를 선택하며, 더블클릭으로 파일을 연다. 탭은 최대 20개까지 열 수 있다.

4.6.2 Explorer 키보드 단축키

트리에 포커스가 있을 때 `Ctrl+C`, `Ctrl+X`, `Ctrl+V` 는 복사, 잘라내기, 붙여넣기를 하고, `Ctrl+A` 는 펼쳐진 노드를 모두 선택하며, `Delete` 는 삭제한다. 고정 폴더에는 적용되지

4.6.3 사이드바 헤더 버튼

헤더 버튼은 마우스 포인터가 헤더 위에 있을 때 나타난다: **Refresh Explorer**, **Collapse All**(접기 또는 펼치기), **Show Hidden Entries** 를 담은 **More Actions...**.

4.6.4 드래그 앤 드롭

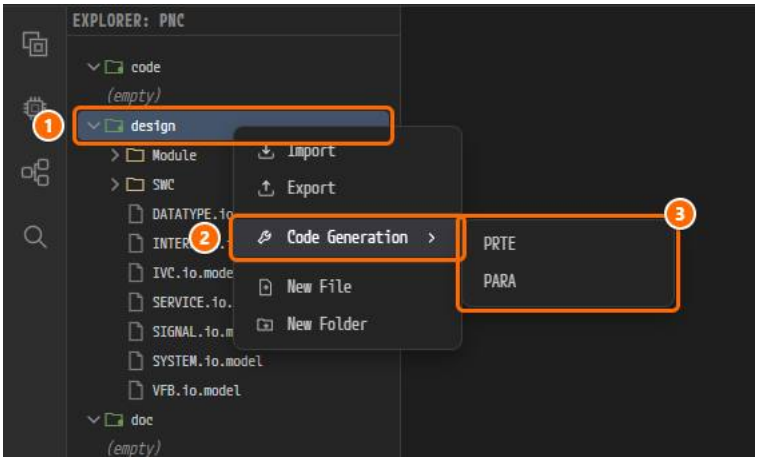
1. 파일이나 폴더를 드래그한다.
2. 폴더 행의 가운데에 올린다. 접힌 폴더는 약 0.6초 뒤 펼쳐진다.

3. 놓으면 그 폴더로 옮겨진다.

같은 이름이 이미 있으면 덮어쓰지 않고 이름을 바꾼다.

4.6.5 컨텍스트 메뉴 (우클릭)

메뉴는 우클릭한 대상(빈 영역, 고정 폴더, 일반 폴더, 파일)에 따라 다르다. `design` 폴더에는 **Import**, **Export**, **Code Generation** ▸ (**PRTE**, **PARA**)가 있다.



4.6.6 파일과 폴더 만들기

1. 폴더를 우클릭하고 **New File** 이나 **New Folder** 를 고른다.
2. 이름을 입력하고 **Create** 를 클릭한다. 새 파일은 에디터 탭으로 열린다.

`doc` 은 `.md` 와 `.markdown` 만, `code` 는 C, C++, Rust 소스 확장자만 받는다.

4.6.7 이름 변경과 삭제

- **Rename** 은 이름만 바꾸며, 파일의 확장자는 유지된다. 열린 탭과 편집 이력은 새 경로를 따라간다.
- **Delete**(또는 `Delete` 키)는 먼저 확인을 받는다. 삭제는 되돌릴 수 없다.
- 파일 시스템이 이름 변경, 삭제, 새 파일이나 폴더 만들기를 거부하면(쓰기 권한 없음) `Permission denied` 토스트가 나타난다.

4.6.8 잘라내기 · 복사 · 붙여넣기

Paste 는 선택한 폴더에, 파일을 선택했으면 그 파일이 든 폴더에 붙여 넣는다. 같은 이름이 있으면 `main_copy.c` 처럼 이름을 바꾼다.

4.7 프로젝트 이름 변경과 삭제

PAIO에는 프로젝트의 이름을 바꾸거나 삭제하는 명령이 없다. 프로젝트는 곧 폴더이므로 폴더의 이름을 바꾸거나 폴더를 삭제한다.

4.7.1 Rename Project

1. 그 프로젝트를 연 PAIO 창을 닫는다.
2. OS 파일 관리자로 프로젝트 폴더의 이름을 바꾼다. 새 이름은 4.3.2의 규칙을 따라야 한다.
3. 이름을 바꾼 폴더를 PAIO에서 연다(`PROJECT ▸ Open Project...`). PAIO가 메타 파일 이름을 폴더에 맞춘다.

4.7.2 Delete Project

1. 그 프로젝트를 연 PAIO 창을 닫는다.
2. OS 파일 관리자로 프로젝트 폴더를 삭제한다.

Recent 목록에 남은 항목은 `PROJECT ▸ Open Recent ▸ Clear Recently Opened` 로 지운다.

4.8 저장

- **Save**(`PROJECT ▸ Save`, `Ctrl+S`)는 활성 편집기를 저장한다.
- **Save All**(`PROJECT ▸ Save All`)은 프로젝트의 변경된 파일을 모두 저장한다.

- **Save As...** 는 없다. SYSTEM 패널의 변경은 하는 즉시 디스크에 기록된다.
- 저장하지 않은 변경이 있는 채로 창을 닫으면 **Save**, **Don't Save**, **Cancel** 이 있는 **Unsaved Changes** 대화상자가 나타난다.

라이선스가 만료되었거나 읽기 전용이면 저장 대신 업그레이드 대화상자가 뜬다.

4.9 편집 이력 (History)

하단 유틸리티 패널의 **History** 탭이 프로젝트의 편집 이력을 보여 준다. 상태바 왼쪽의 **Toggle Panel** 버튼으로 패널을 켜고 끈다.

- 가장 새로운 항목이 맨 위에 있다. **Current state** 위의 항목은 다시 실행할 수 있고, 아래의 항목은 실행 취소할 수 있다.
- 항목을 **더블클릭**하면 그 시점으로 이동한다.
- 프로젝트마다 항목을 최대 25개까지 보관한다.

4.10 외부 변경 감시와 자동 복구

PAIO는 프로젝트 폴더를 감시하므로 앱 밖에서 추가하거나 삭제한 파일이 트리에 반영된다. 앱 밖에서 삭제된 고정 폴더는 다시 만들어진다. 편집기에 저장하지 않은 변경이 있는 파일
장하면 디스크를 덮어쓴다.

4.11 파일을 직접 편집할 때

모델은 텍스트 JSON으로 저장되므로 형상 관리에 그대로 넣을 수 있다. PAIO 밖에서 편집하는 것은 권장하지 않는다. 참조가 깨질 수 있기 때문이다. 이름은 편집기에서 바꾸고, 문제

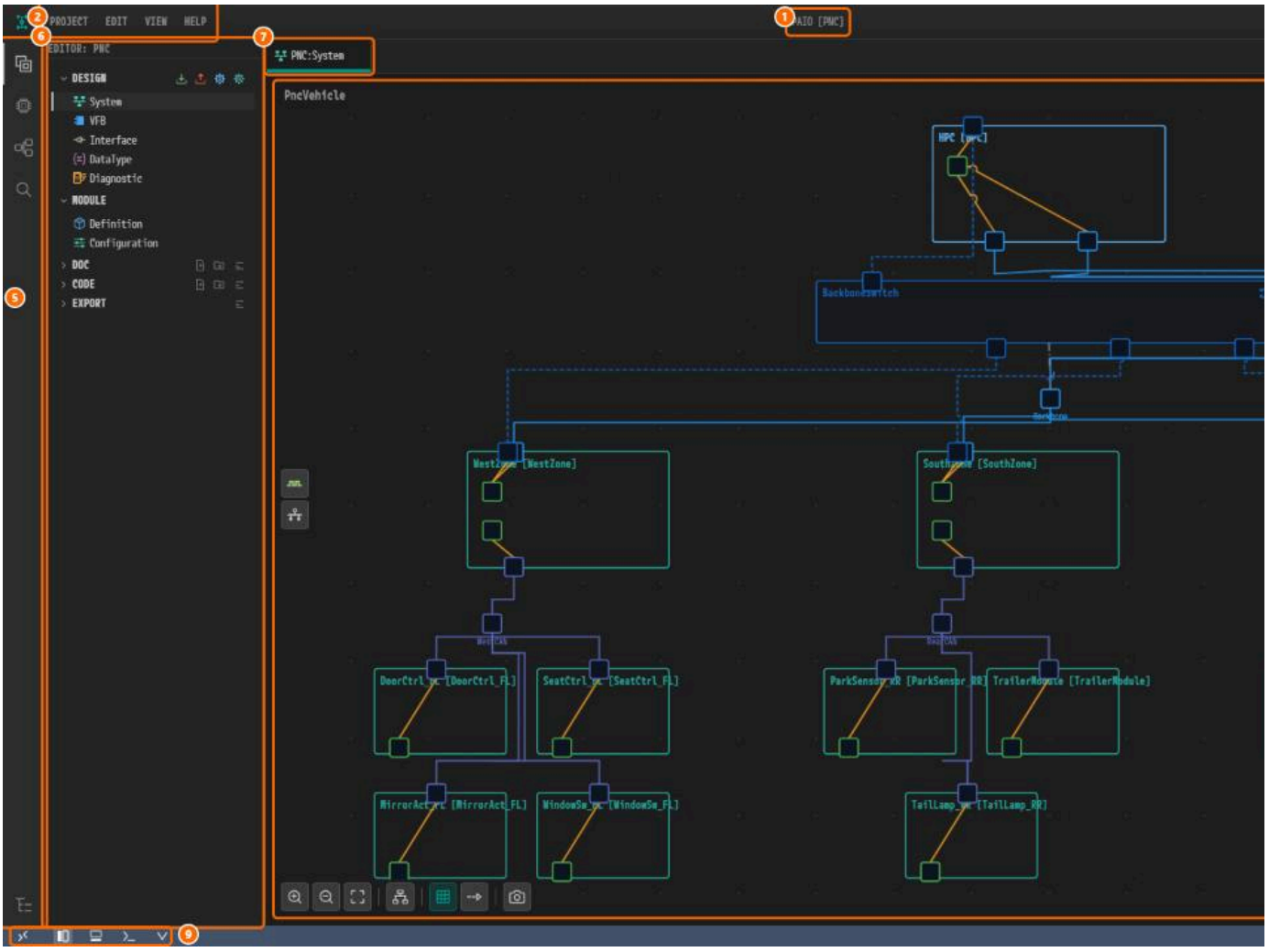
화면 구성과 조작

이 장에서 다루는 것

이 장에서는 PAIO 창의 각 영역이 무엇을 하는지 설명한다. 헤더, 아이콘 레일, **EDITOR** 뷰와 Explorer가 놓이는 사이드바, 에디터 영역, 하단 패널, 상태바, Search 뷰, Meta Class 검 찾는 사용자를 위한 장이며, 선행 조건은 없다.

- 설치와 로그인: 03. 설치 · 로그인 · 라이선스
- 폴더가 창의 프로젝트가 되는 방식: 04. 워크스페이스와 프로젝트
- 단축키만 볼 때: 18. 단축키 총람

PAIO 창 하나는 프로젝트 하나를 담는다. 창이 연 폴더가 곧 프로젝트이며, 그 이름이 창 제목에 표시된다.



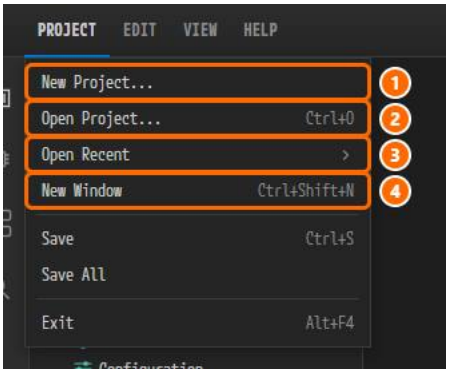
5.1 헤더 (상단바)

5.1.1 두 개의 메뉴

PAIO에는 메뉴가 두 벌 있다. 앱이 그린 메뉴 막대(**PROJECT**, **EDIT**, **VIEW**, **HELP**)는 헤더 왼쪽에 있다. 네이티브 애플리케이션 메뉴는 Windows와 Linux에서는 보이지 않고, macO 린 메뉴를 기준으로 설명한다.

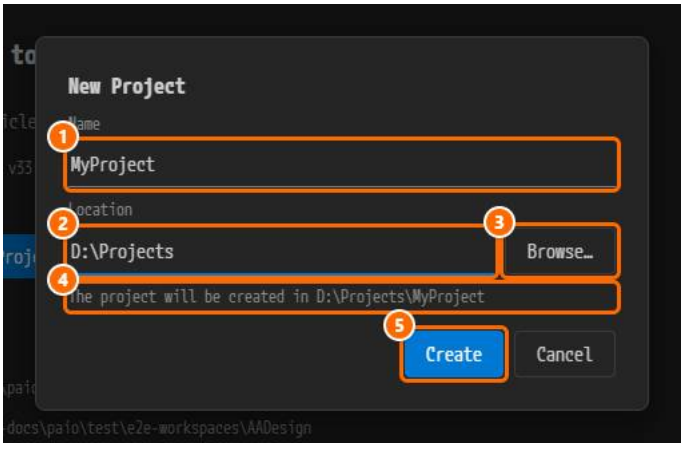
5.1.2 Project 메뉴

PROJECT 는 프로젝트를 만들고 열고 저장하며, 새 창을 열고(`Ctrl+Shift+N`) 종료한다.



프로젝트를 만드는 순서는 다음과 같다.

1. **New Project...** 를 고른다.
2. **Name** 에 프로젝트 이름을 입력한다. 영문자, 숫자, 하이픈, 밑줄만 쓸 수 있다.
3. **Location** 에 상위 폴더를 입력하거나 **Browse...** 를 클릭한다.
4. **Create** 를 클릭한다. PAIO가 `<Location>\<Name>` 폴더를 만들고 창의 프로젝트로 연다. 창에 이미 프로젝트가 열려 있으면 새 프로젝트는 새 창에서 열린다.



- **Save**(`Ctrl+S`)는 활성 에디터를, **Save All** 은 프로젝트의 변경된 에디터를 모두 저장한다. 라이선스가 만료되었거나 읽기 전용이면 둘 다 막힌다.
- 저장하지 않은 변경이 있는 채로 창을 닫으면 `Unsaved Changes` 다이얼로그 하나가 그 전부를 묻는다.

5.1.3 Edit 메뉴

EDIT 에는 **Undo**(`Ctrl+Z`), **Redo**(`Ctrl+Y`), **Cut**, **Copy**, **Paste**, **Select All**, **Go to Meta Class...**(`Ctrl+T`, 5.10)이 있다. Undo와 Redo는 프로젝트 이력에 작용하며 작업 이
스트 입력에만 작용하며, 캔버스 노드나 표의 행에는 작용하지 않는다.

5.1.4 View 메뉴

VIEW 는 사이드바(`Ctrl+B`)와 Utility 패널을 켜고 끄며, 창 배율을 바꾸고(5.7), 전체 화면(`F11`)을 전환하고, **Theme** 를 **Dark** 와 **Light** 사이에서 바꾼다.

> 메뉴에는 **Toggle Panel** 옆에 `Ctrl+J` 가 적혀 있지만, `Ctrl+J` 키는 **터미널 패널**을 토글한다. Utility 패널은 메뉴나 상태바 버튼으로 연다.

5.1.5 Help 메뉴

HELP 는 **Welcome** 탭을 다시 열고, 문의 페이지를 열고(**Report Issue**), 버전을 보여 주고(**About PAIO**), **Check for Updates** 를 실행한다. 업데이트 결과는 `Software Update`

5.1.6 헤더 오른쪽 영역

왼쪽부터 테마 전환, AI Chat 도크를 여는 로봇 아이콘, **Profile**, **AI Account**, **Sign Out** 을 담은 사용자 아이콘(3.3 참고), 창 조작 버튼이 있다. 라이선스가 만료되면 로봇 아이콘이 ㅂ

5.1.7 헤더 가운데 제목

제목은 프로젝트가 없으면 `PAIO`, 있으면 `PAIO [<프로젝트>]` 이다(예: `PAIO [PNC]`). 원격 연결 상태는 상태바 왼쪽에 표시된다.

5.2 아이콘 레일과 사이드바 (좌측)

창 왼쪽 가장자리의 아이콘 레일이 사이드바에 무엇을 보일지 고른다. **EDITOR** 뷰는 설계 에디터를 열고, Explorer는 프로젝트 폴더의 파일을 보여 준다.

5.2.1 아이콘 레일

레일 아이콘에는 글자가 없다. 마우스를 올리면 툴팁이 보인다.

위치	툴팁	보여 주는 것
1 (맨 위)	`Design`	**EDITOR** 뷰(5.2.2), 시작할 때의 뷰
2	`vECU`	vECU 뷰
3	`Workflow`	AI 워크플로우를 탭으로 연다(21. AI 워크플로우)
4	`Search`	Search 뷰(5.8)
맨 아래	`Explorer`	Explorer(5.2.3)

이미 보이는 뷰의 아이콘을 클릭하면 사이드바가 숨겨지고, 다시 클릭하면 나타난다.

5.2.2 EDITOR 뷰

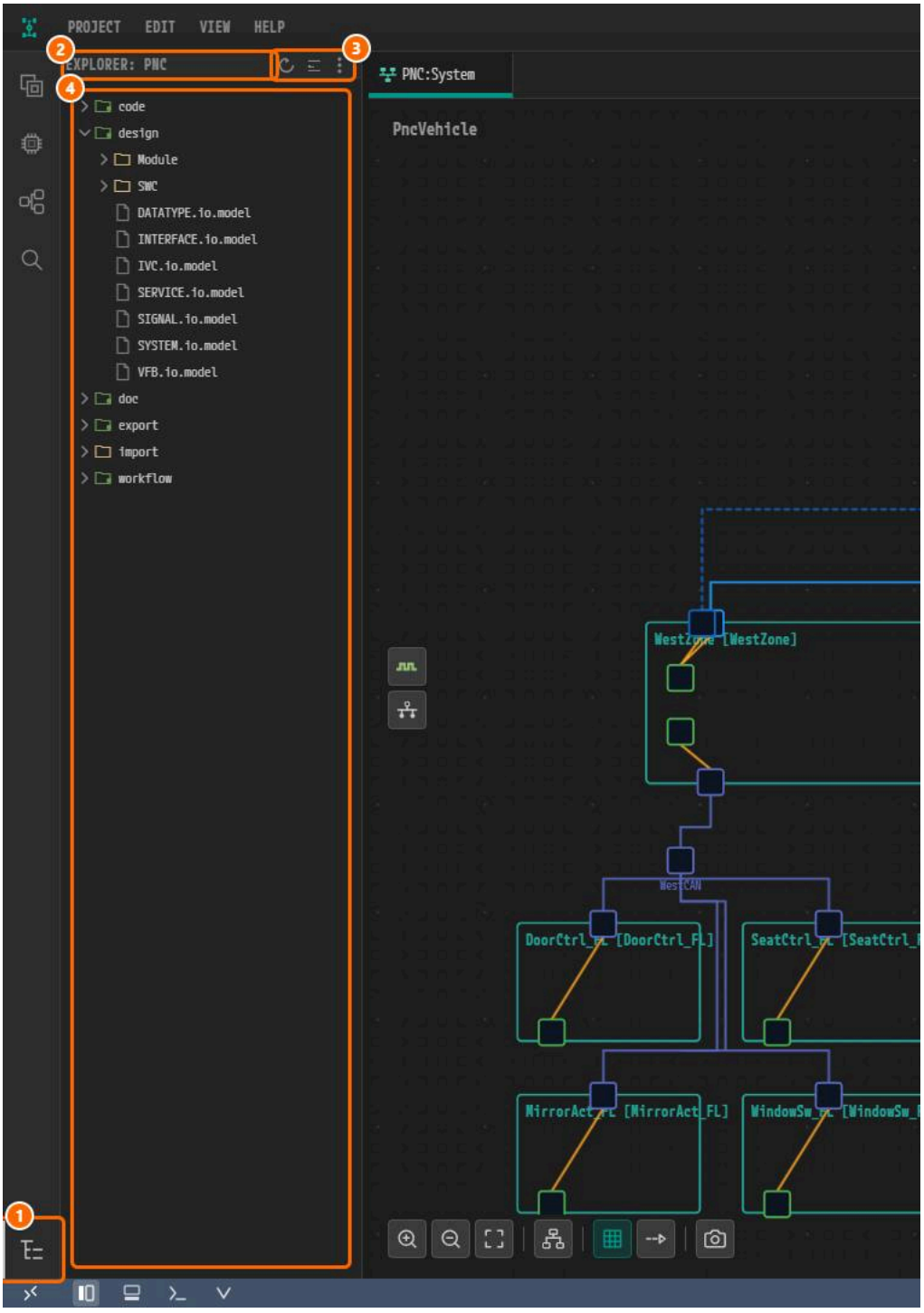
머리글은 `EDITOR: <프로젝트>` 이다. 그 아래에 **DESIGN**, **MODULE**, **DOC**, **CODE**, **EXPORT** 섹션 다섯 개가 있다.

1. **DESIGN** 이나 **MODULE** 아래의 행, 예를 들어 **System** 을 클릭한다. 한 번 클릭하면 그 에디터가 열리며, 여기서는 탭 `PNC:System` 이 열린다.

Signal 에디터에는 행이 없으며, System 에디터 캔버스의 **Signal** 아이콘으로 연다(06. System 에디터 참고). **DESIGN** 머리글의 **Import**, **Export** 아이콘은 13. 임포트 · 익스포트0

5.2.3 Explorer

레일 맨 아래의 **Explorer** 아이콘을 클릭한다. Explorer는 프로젝트 폴더의 내용을 바로 보여 준다: `code`, `design`, `doc`, `export`, `import`, `workflow`.



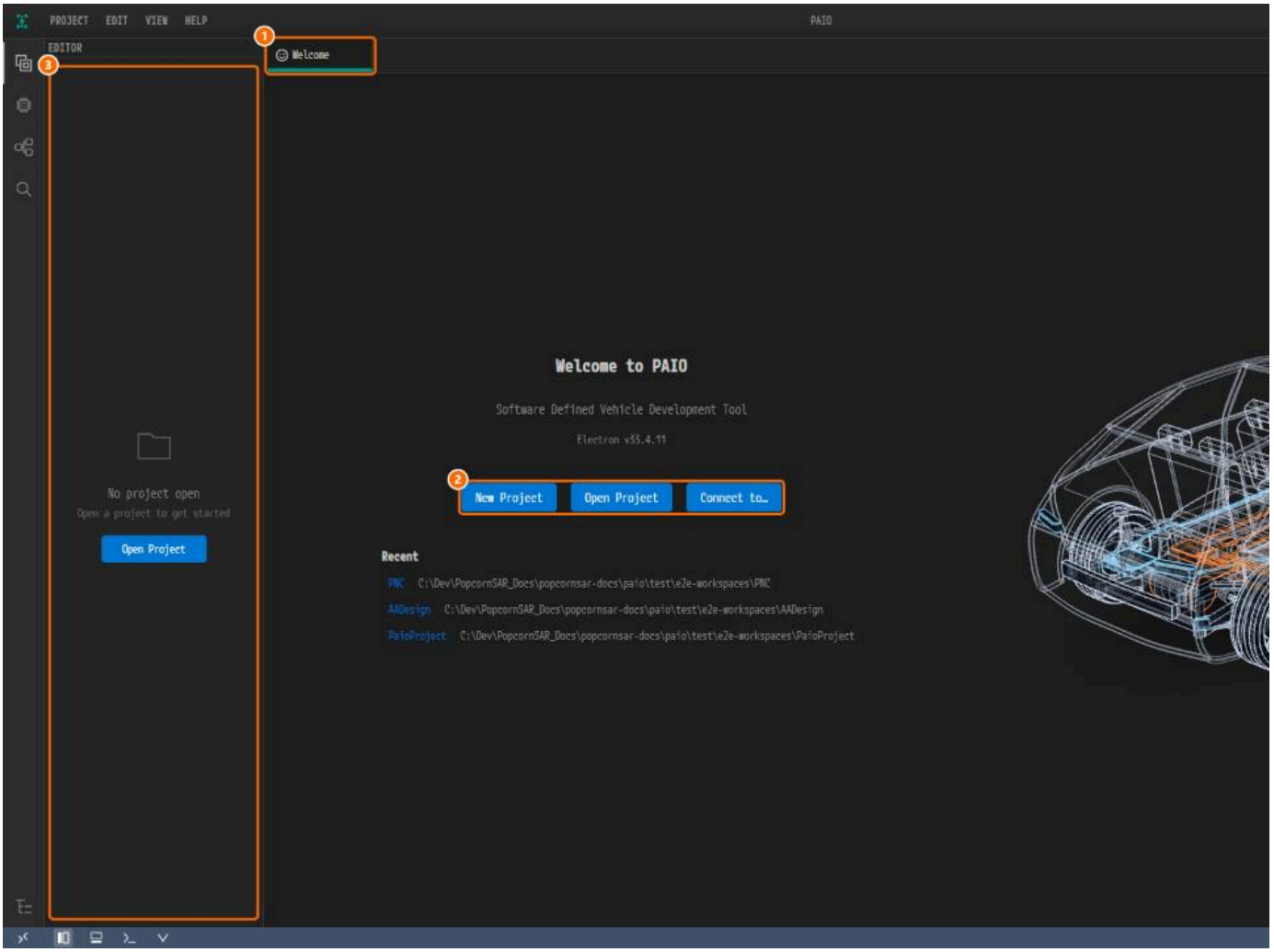
> **Explorer는 설계 에디터를 열지 않는다.** `*.io.model` 파일을 더블클릭하면 읽기 전용 JSON으로 열린다(5.3.8 참고). 설계는 **EDITOR** 뷰에서 편집한다.

머리글은 프로젝트 이름과 연결 종류(SSH, WSL, 컨테이너)를 보여 준다. 폴더 조작은 04. 워크스페이스와 프로젝트에, 키는 18.8 Explorer에 있다.

5.3 에디터 영역

5.3.1 환영 화면

Welcome 탭은 PAIO를 시작할 때마다 열린다.



- **New Project** 는 **New Project** 대화상자를 연다(5.1.2).
- **Open Project** 는 폴더를 프로젝트로 연다.
- **Connect to...** 는 원격 연결을 시작한다(17. 원격 개발 (SSH · vECU 컨테이너)).
- 아래 **Recent** 목록은 최근에 연 프로젝트를 보여 준다.

탭을 다시 열려면 **HELP** > **Welcome** 을 고른다.

5.3.2 탭

탭을 클릭하면 활성화된다. × 나 가운데 버튼 클릭으로 닫고, 우클릭하면 컨텍스트 메뉴가 열리며, 드래그하면 순서 변경, 이동, 분할을 한다. 제목 형식은 `프로젝트명 : 노드명` 이다(여
탭 너비는 이름 길이에 맞는다. 탭이 다 들어가지 않으면 오른쪽 끝의 꺾쇠 버튼 두 개(**Previous Tabs**, **Next Tabs**)로 가려진 탭을 하나씩 보이게 한다.

- 전체 그룹을 합쳐 탭은 최대 20개까지 열 수 있다. 그 이상은 새 탭이 열리지 않고 화면에 아무 안내도 나오지 않는다.
- 같은 파일을 다시 열면 이미 열린 탭으로 이동한다.

5.3.3 탭 컨텍스트 메뉴

탭을 우클릭하면 **Close**, **Close Others**, **Close Saved**, **Close All** 이 나온다. 뒤의 세 항목은 같은 그룹에 작용한다. 변경분이 있는 탭을 닫으면 저장 여부를 묻는다.

5.3.4 분할 보기

분할을 실행하는 메뉴 항목은 없다. 탭을 끌어다 놓아 분할한다.

1. 탭을 드래그한다.
2. **에디터 본문의 오른쪽 절반**에 놓는다. 그룹이 좌우로 분할되고 탭이 새 오른쪽 그룹으로 옮겨진다. 놓기 전에 세로 분할선이 그려진 사각형 표식이 나타난다.

왼쪽 절반이나 탭 바에 놓으면 분할하지 않고 탭만 옮긴다. 분할은 좌우 방향만 지원한다. 경계선을 드래그해 비율을 바꾼다. 이렇게 만든 창을 에디터 그룹이라고 하며, 빈 그룹은 저절

5.3.5 에디터 종류

탭은 노드 종류에 따라 모델 편집기 아홉 가지(System, Signal, VFB, Interface, Data Type, Diagnostic, Module Definition, Module Config, Workflow) 또는 파일 편집기와 지, HTML, 비교, Welcome, Search) 중 하나로 열린다. 이미지 파일은 항상 이미지 뷰어로 열린다.

5.3.6 저장 · 실행 취소 · 다시 실행

프로젝트마다 편집 이력이 하나 있고, 저장, 실행 취소, 다시 실행은 모두 이 이력으로 동작한다.

- **EDIT ▸ Undo** 는 어느 에디터에서 한 편집이든 되돌리고, 그 대상이 있는 에디터로 이동한다.
- 저장하면 "변경 없음" 지점이 기록되지만 이력은 지워지지 않으므로, 저장 뒤에도 계속 실행 취소할 수 있다.
- 그룹이 여럿이어도 메뉴의 Undo / Redo / Save는 활성 그룹에서 한 번만 실행된다.

5.3.7 외부 변경 감지

에디터가 열어 둔 파일이 디스크에서 바뀌면 PAIO가 다시 읽는다. **저장하지 않은 편집이 있거나 편집이 진행 중이면 다시 읽지 않으므로** 편집이 사라지지 않는다. 이때는 파일이 디스크 뜬다.

5.3.8 XML / ARXML 에디터

`.arxml`, `.xml`, `.dbc` 파일을 열면 Monaco 기반 에디터가 **보기 모드**로 열린다.

1. 눈 아이콘을 클릭해 편집 모드로 바꾼다. 저장, 실행 취소, 다시 실행, 서식 버튼이 활성화된다.
2. 파일을 편집한다. `<` 를 입력하면 그 위치에 올 수 있는 AUTOSAR 자식 요소를 제안한다.
3. 아래 PROBLEMS 패널을 확인한다. 검사는 파일을 열 때와 변경할 때마다 실행된다.

Explorer에서 연 PAIO 문서(`*.io.model` 등)는 여기서 **읽기 전용**이다.

5.4 하단 패널

에디터 영역 아래에는 서로 독립적인 패널 Utility, TERMINAL, VALIDATION이 있고 각각 따로 켜고 끈다. 세 패널과 AI Chat 도크는 기본으로 숨겨져 있다. 표시 여부와 크기는 앱을 다

5.4.1 Utility 패널 — History

History 탭은 프로젝트의 편집 이력을 최신 순으로 보여 준다. 행을 더블클릭하면 그 시점으로 이동한다. 이 패널에는 Problems 탭이 없다. 진단은 각 에디터의 PROBLEMS 패널과 V

5.4.2 TERMINAL 패널

터미널 패널에서 터미널을 추가, 분할, 종료하고 글자 크기를 바꾼다. `Ctrl+J` 나 `Ctrl+`` 로 열 때 터미널이 없으면 하나 만들어 준다. 16. 코드 편집 · 빌드 · 터미널 · 문서를 참고한다

5.4.3 VALIDATION 패널

AUTOSAR 제약 위반을 표로 보여 준다. 위반 행을 클릭하면 그 요소가 있는 에디터가 열리고 요소가 선택된다. 14. Validation을 참고한다.

5.5 상태바 (하단)

5.5.1 왼쪽

- 맨 왼쪽에는 원격 표시기(`><`)가 항상 있다. 연결되면 초록 배경이 되고 `SSH: <호스트>` 처럼 연결을 표시한다.
- 버튼 네 개가 사이드바, Utility 패널, 터미널, VALIDATION 패널을 토글한다.
- 검증 버튼에는 위반 개수 배지가 붙는다. `!` 는 검증을 실행하지 못했다는 뜻이다.
- **IntelliSense** 항목은 C/C++ 언어 서버가 한 번 시작된 뒤 그 상태를 보여 준다.

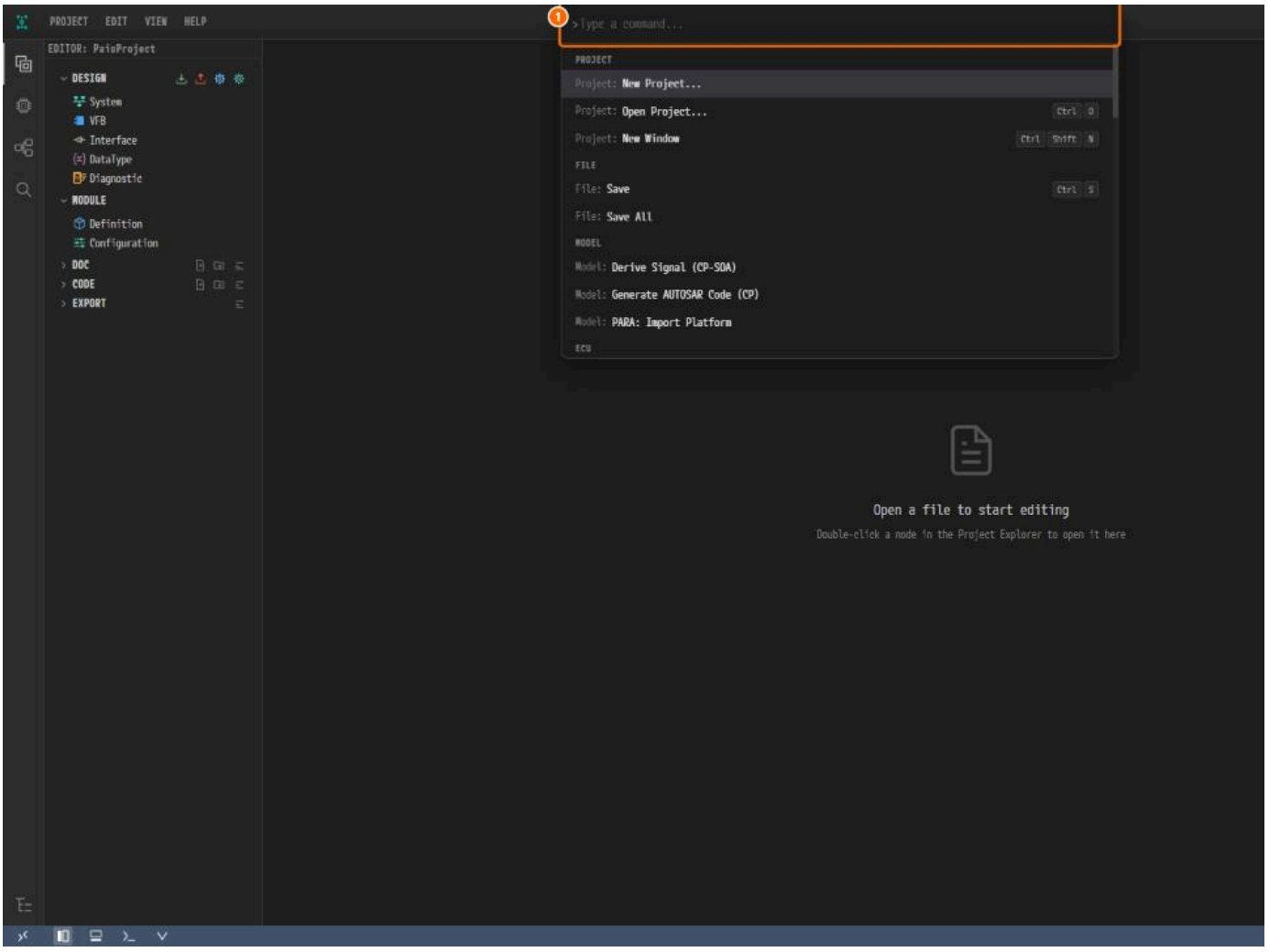
5.5.2 오른쪽

오른쪽에는 라이선스(예: `Pro`), 알릴 업데이트가 있을 때의 업데이트 표시, 버전 `v<버전>` 이 있다. 라이선스를 클릭하면 다이얼로그가 열린다. 3.4.4를 참고한다.

5.6 명령 팔레트

1. `Ctrl+Shift+P` 를 누른다.
2. 명령 일부를 입력한다. 퍼지 검색이므로 `tgt` 가 `View: Toggle Terminal` 에 걸린다.
3. `Enter` 를 누르거나 명령을 클릭한다.

`Escape` 로 닫는다.



5.6.1 등록된 명령

명령은 Project, File, Model, ECU, Edit, View, Search, AI, Help, Remote-SSH, Code 열한 개 카테고리에 모두 41개다.

5.6.2 실행해도 아무 일이 없는 명령

> 아홉 개 명령은 목록에 나오지만 실행해도 아무 일도 하지 않는다: **Cut**, **Copy**, **Paste**, **Select All**, **Toggle Sidebar**, **Toggle Panel**, **Toggle Full Screen**, **Toggle Terminal**, **Re**

5.7 화면 배율

VIEW ▸ Zoom In(`Ctrl+=`), **Zoom Out**(`Ctrl+-`), **Reset Zoom**(`Ctrl+0`)은 창 전체의 배율을 한 단계에 20%씩, 양쪽으로 여덟 단계까지 바꾼다. 배율은 모든 PAIO 창에 적용되
+ 휠로 하는 확대는 이와 별개다(18.2 그래픽 편집기 (System · VFB) 참고).

5.8 Search 뷰

Search 뷰는 프로젝트의 `design`, `doc`, `code` 폴더에 저장된 파일을 검색한다.

1. 레일의 **Search** 아이콘을 클릭하거나 `Ctrl+Shift+F` 를 누른다.
2. 검색어를 입력한다. 입력을 멈추면 잠시 뒤 검색이 저절로 실행된다.
3. 결과를 클릭해 그 위치로 간다.

> **검색은 저장된 파일만 읽는다.** 저장하지 않은 탭이 있으면 먼저 저장할지 묻는다.

5.8.1 검색 입력란

입력란 안에 토글 세 개가 있다: `Aa`(**Match Case**), `ab`(**Match Whole Word**), `.*`(**Use Regular Expression**). `Enter` 는 바로 검색하고, `ArrowUp` / `ArrowDown` 은 이 프로젝

5.8.2 머리글 버튼

머리글 버튼은 다시 검색, 지우기, Search Editor 열기(5.9), Meta Class 검색 열기(5.10), 목록, 트리 전환, 결과 접기, 펼치기를 한다.

5.8.3 결과

결과는 **Design**, **Doc**, **Code** 세 영역에 일치 개수와 함께 나온다. **Design** 은 결과를 여는 에디터별로 묶고, **Doc** 와 **Code** 는 파일과 일치하는 줄을 보여 준다. `...` 버튼은 결과를 좁

5.8.4 결과 열기

결과 행을 클릭한다. Doc, Code 줄은 일치한 부분을 선택한 채 파일을 연다. Design 결과는 값을 가진 에디터를 열고 그 필드를 표시한다. `F4` 와 `Shift+F4` 는 다음, 이전 결과로 긴

5.8.5 Find Usages

Design 의 요소 행을 우클릭하고 `Find Usages of '<name>'` 을 고른다. 결과에는 그 요소를 가리키는 설계 참조가 나온다. 막대의 `×` 를 클릭하면 텍스트 검색으로 돌아간다.

5.9 Search Editor

Search Editor는 에디터 탭 안에서 따로 실행하는 검색이다. Search 뷰 머리글의 **Open New Search Editor** 로 열면 뷰의 검색어를 이어받고, 명령 팔레트로 열면 빈 상태로 열린다. 읽기 전용이다.

5.10 Meta Class 검색 (Ctrl+T)

Meta Class 검색은 한 가지 물음에 답한다. 이 AUTOSAR 메타 클래스는 PAIO 어디에서 설계하는가? `ServiceInterface`, `TlsSecureComProps` 처럼 메타 클래스 이름을 입력

1. `Ctrl+T` 를 누르거나, **EDIT ▸ Go to Meta Class...** 를 고르거나, Search 뷰 머리글의 아이콘을 누르거나, 명령 팔레트에서 **Go to Meta Class...** 를 실행한다.
2. 메타 클래스 이름을 입력한다. 공백이나 `-` 는 단어를 나누므로 `service interface` 와 `SERVICE-INTERFACE` 모두 `ServiceInterface` 를 찾는다.
3. `ArrowUp` / `ArrowDown` 으로 줄을 고르고 `Enter` 를 누르거나 줄을 클릭한다.

한 줄은 메타 클래스 하나와 그것을 설계하는 위치 하나이다. 예를 들어 `ServiceInterface Designed in Interface › Service` 이다. 여러 곳에서 설계하는 클래스는 위치트에 이미 있는 그 클래스의 첫 요소로 간다. 그 위치에 아직 요소가 없으면 하나를 추가하는 절이나 버튼을 표시한다. 아무것도 만들지 않는다.

목록 아래에는 고른 줄에 대해, 그 클래스를 추가하기 전에 있어야 하는 것(`Needs first:`)과 추가한 뒤 채울 것(`Fill in after:`)이 나온다. 각 이름 뒤의 체크 표시나 `none y`

> **프로젝트의 요소를 찾을 때는 Search 뷰(5.8)를 쓴다.** Meta Class 검색은 메타 클래스만 나열한다. Markdown 에디터와 Doc 시작 화면에서는 `Ctrl+T` 가 원래 뜻(새 문서)을

편집기

모델 편집기를 보통 사용하는 순서대로 한 장씩 정리했다.

설계 편집기는 창 왼쪽의 **EDITOR** 뷰에서 연다. **DESIGN**(**System**, **VFB**, **Interface**, **DataType**, **Diagnostic**)이나 **MODULE**(**Definition**, **Configuration**) 아래의 행을 클릭하면 그을 붙인 탭(예: `<project>:System`)으로 열린다. Signal 편집기는 행이 없으며, System 편집기 캔버스 왼쪽 가장자리 아이콘 줄의 맨 위 아이콘 **Signal** 로 연다. SYSTEM 패널은 (PAIO 로고)으로 연다. Diagnostic 에디터는 Diagnostic 에디터에서 다룬다.

편집기	정의하는 대상
System 편집기	ECU · HPC 노드, 통신 클러스터, 연결. **DESIGN** ▸ **System** 으로 연다(탭 `<project>:System`). 예전의 IVC 편집기이다
Signal 편집기	시그널과 PDU, 비트 레이아웃, 초기값. System 편집기 캔버스의 **Signal** 아이콘으로 연다
VFB 편집기	소프트웨어 컴포넌트 컴포지션, 포트, 연결
SYSTEM 패널	시스템의 **NAME** · **VERSION** · **DESCRIPTION**, System 편집기의 **ECU MAPPING**, VFB 편집기의 **ROOT COMPOSITION** · **COMPOSITIONS**
Interface 편집기	포트 인터페이스, 데이터 요소, 오퍼레이션
DataType 편집기	애플리케이션 · 구현 데이터 타입, 단위
Diagnostic 에디터	진단 추출(DEXT): 세션 · 보안 · DID · 루틴 · 이벤트 · DTC와 각 ECU · HPC가 지원하는 항목. **DESIGN** ▸ **Diagnostic** 으로 연다(탭 `<project>:Diagnostic`)
ECUC	ECU 구성 모듈, 컨테이너, 파라미터. **MODULE** ▸ **Definition** 과 **MODULE** ▸ **Configuration** 으로 연다

System 에디터

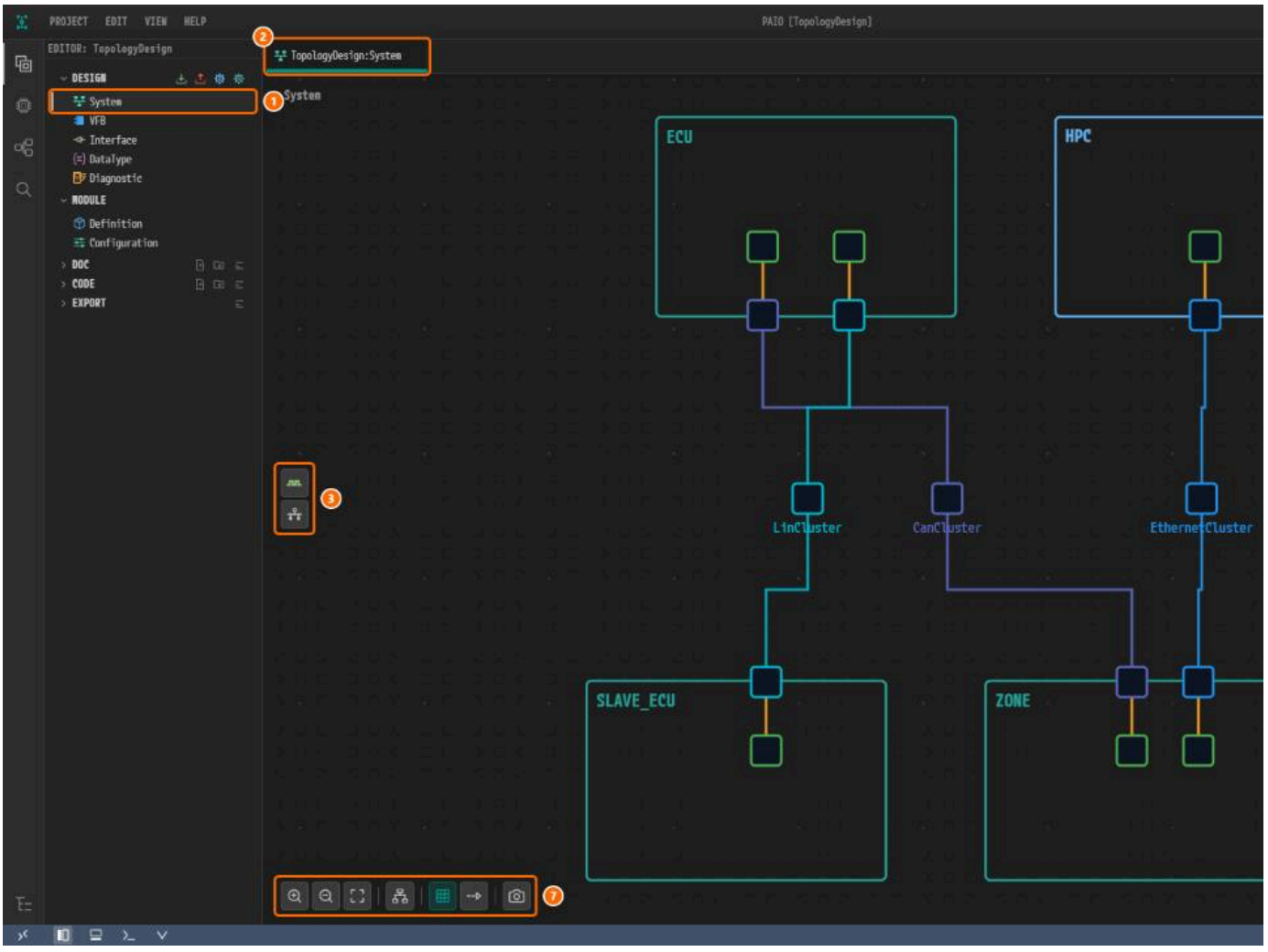
이 장에서 다루는 것

System 에디터는 차량 내 통신 토폴로지를 그림으로 설계하는 에디터이다. ECU, HPC 노드와 그 안의 커넥터, 컨트롤러를 놓고, 통신 클러스터와 물리 채널로 묶은 뒤, 이더넷 스위치(뜻은 19. 용어 사전에 있다.

- 대상 독자: AUTOSAR 네트워크 토폴로지를 직접 편집하는 사용자
- 선행 조건: 프로젝트의 창이 열려 있는 것 (제목 표시줄에 `PAIO [<프로젝트>]` 가 보인다)

6.1 에디터 열기와 저장 위치

1. 왼쪽 **EDITOR** 뷰의 **DESIGN** 아래에서 **System** 을 누른다. 탭 `<프로젝트>:System` 이 열리고 앞서 저장한 내용을 불러온다.
2. `Ctrl+S` 로 저장한다. 내용은 `design/IVC.io.model` 에 저장된다.



*① **DESIGN** 아래의 **System**. ② 탭 `TopologyDesign:System`. ③ 액션 줄: **Signal** 과 **Partial Network (PNC)**. ④ 속성 패널 토글. ⑤ 잠금. ⑥ **System**(PAIO 로고). ⑦ 캔버스 툴*

- 캔버스 왼쪽 위의 라벨은 시스템 **NAME** 이며, 에디터 이름이 아니다.
- 액션 줄의 **Signal** 은 Signal 에디터를 연다. **Partial Network (PNC)** 는 PNC 다이얼로그를 연다(6.14).
- 오른쪽 가장자리의 **System** 은 SYSTEM 패널을 연다(11. SYSTEM 패널).

노드 위치와 연결선 경로도 저장되지만, ARXML 로 익스포트할 때는 들어가지 않는다.

6.2 다룰 수 있는 요소

이 에디터는 여덟 종류의 요소를 놓는다. **ECU**, **HPC**, **Connector**, **Controller**, **Ethernet Switch**(Coupling Element), **Coupling Port**, **Communication Cluster**, **Physical Ch** 커넥터는 노드 경계 위에 있으며, 클러스터는 커넥터를 잇는 선과 그 가운데의 사각형이다.

> **FlexRay 요소는 이 에디터에 나타나지 않는다.** 임포트로 들어온 FlexRay 요소는 캔버스를 그리기 전에 걸러지고, 저장은 화면의 다이어그램만으로 내용을 다시 만든다. FlexRay 프로젝트에서 사라진다.

6.3 캔버스 조작

캔버스를 움직이고 요소를 선택하고 옮기는 기본 조작이다.

6.3.1 마우스

- 빈 곳을 드래그하면 화면을 옮기고, `Ctrl` + 휠로 확대, 축소한다.
- 노드를 누르면 선택하고, `Ctrl` + 클릭으로 선택에 더한다.
- 노드를 드래그하면 옮긴다. `Shift` + 드래그는 연결이다(6.5.5).
- 노드를 더블클릭하면 설정 다이얼로그가 열린다(6.7).

6.3.2 키보드

단축키는 에디터에 포커스가 있을 때만 동작한다.

키	동작
`Delete` / `Backspace`	확인을 거쳐 선택 항목 삭제
`Ctrl+Z` / `Ctrl+Y`	실행 취소 / 다시 실행
`Ctrl+A`	모두 선택
화살표 키	선택한 노드를 1px 이동 (`Shift`: 20px)
`Ctrl+S`	저장

`Ctrl+C` / `Ctrl+V` / `Ctrl+X` 는 동작하지 않는다.

6.3.3 툴바 (캔버스 왼쪽 아래)

툴바(6.1 그림의 ⑦)에는 **Zoom In**, **Zoom Out**, **Fit to View**, **Auto Layout**, **Toggle Grid**, **Toggle Connector Animation**, **Export as PNG** 가 있다. **Export as PNG** 는 파일 대화

다.

6.4 노드 만들기

노드는 우클릭 메뉴로 만든다. 모든 메뉴의 끝에는 **Delete** 가 있고, 삭제는 항상 먼저 확인을 묻는다.

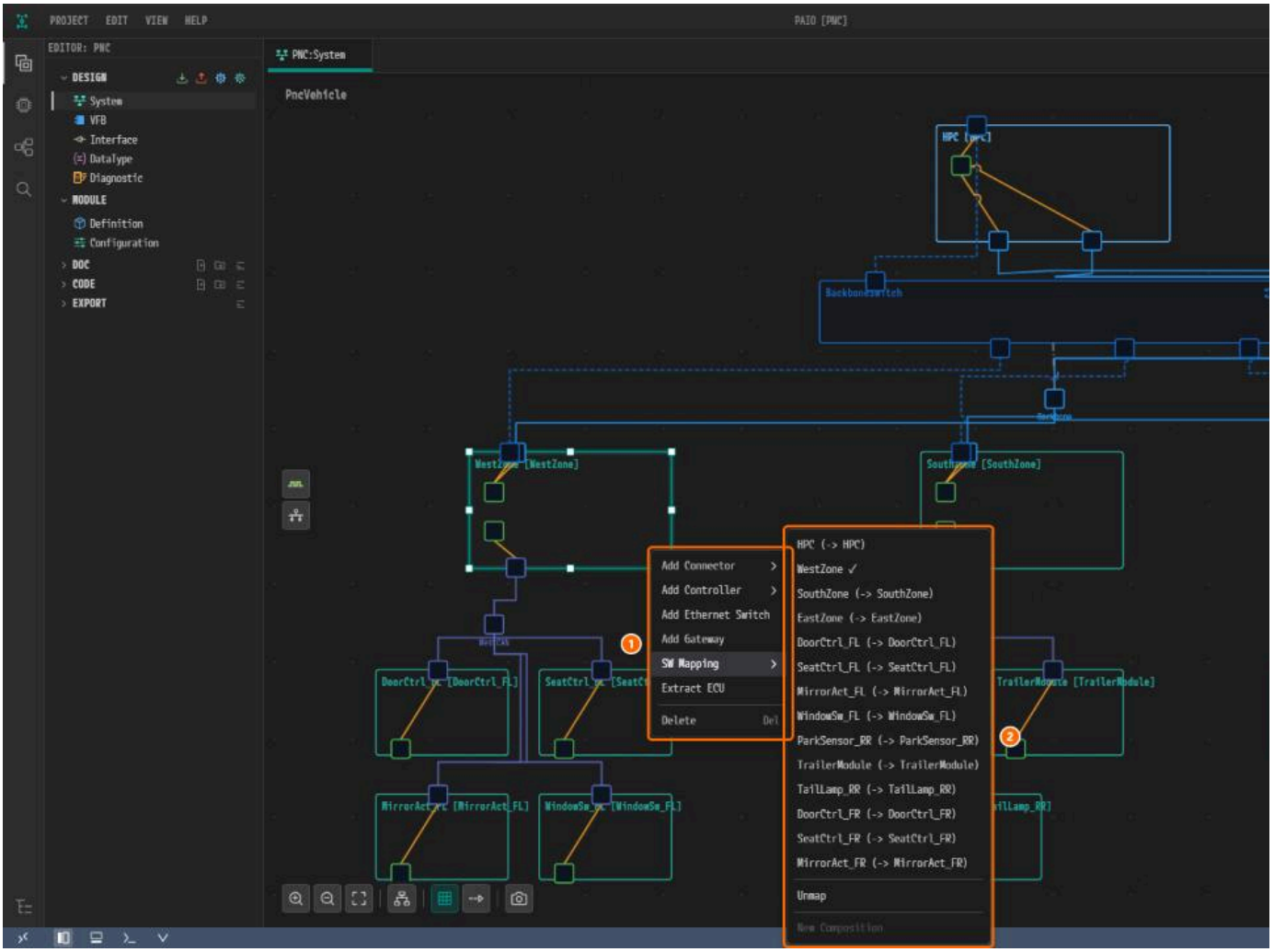
6.4.1 빈 캔버스 우클릭

빈 캔버스를 우클릭하고 **Add ECU** 또는 **Add HPC** 를 고른다. 누른 자리에 노드가 생긴다. 스위치는 ECU 나 HPC 노드의 메뉴에서 추가한다.

6.4.2 ECU / HPC 노드 우클릭

ECU 나 HPC 노드를 우클릭해 그 안에 들어갈 것을 추가한다.

- **Add Connector**, **Add Controller**: Ethernet, CAN, LIN 중에서 고른다.
- **Add Ethernet Switch**, **Add Gateway** / **Remove Gateway**.
- **SW Mapping**: VFB 컴포지션을 이 노드에 매핑한다(6.6).
- **Extract ECU** / **Extract Machine**: 이 노드만 ARXML 로 추출한다(13. 임포트 · 익스포트).



*① 노드 메뉴. ② **SW Mapping** 하위 메뉴: `WestZone ✓` 가 이 노드에 매핑된 컴포지션이다.*

노드를 지우면 그 안의 모든 것이 함께 지워진다. `Ctrl+Z` 로 되살린다.

6.4.3 Ethernet Switch(Coupling Element) 노드 우클릭

Add CouplingPort 로 스위치 포트를 추가한다.

6.4.4 Controller 노드 우클릭

Ethernet 컨트롤러에서는 **Add Coupling Port** 로 컨트롤러가 소유하는 포트를 추가한다. Ethernet 컨트롤러는 포트를 하나만 가질 수 있다. CAN, LIN 컨트롤러의 메뉴에는 **Delet**

6.4.5 Coupling Port 노드 우클릭

Connect to Port... 로 상대 포트를 골라 물리 케이블로 잇는다(6.5.4).

6.4.6 자동으로 붙는 기본 이름

새 노드에는 0 부터 번호가 붙은 이름이 붙는다(예: `ECU_0`, `DoorEcu_Can_Connector_0`). 이름은 속성 패널의 **NAME** 필드에서 바꾼다(6.12).

6.5 연결 만들기

커넥터, 클러스터, 컨트롤러, 포트를 잇고 그 연결을 끊는 방법이다.

6.5.1 커넥터에서 클러스터로 (커넥터 쪽에서 시작)

아직 클러스터에 연결되지 않은 커넥터에는 초록색 **[+] 배지**가 붙는다.

1. 커넥터의 **[+] 배지**를 누른다.
2. **Create CommunicationCluster [...]** 로 같은 프로토콜의 클러스터를 새로 만들거나, **Connect to Channel...** 에서 클러스터 ▸ 물리 채널을 고른다.

3. 커넥터와 클러스터 사이에 선이 그려지고 배지가 사라진다.

커넥터를 우클릭해도 **Connect to Channel...** 이 나오며, 이미 연결된 커넥터에서도 열린다.

6.5.2 클러스터에서 커넥터로 (클러스터 쪽에서 시작)

클러스터를 우클릭하고 **Add Connector to Channel...** 을 고른다(채널이 없으면 **Add Connector...**). 같은 프로토콜이면서 아직 연결되지 않은 커넥터만 나온다. CAN 과 LIN 은 달

6.5.3 컨트롤러와 커넥터 잇기 (내부 배선)

내부 배선은 같은 ECU 안에서 프로토콜이 같은 컨트롤러와 커넥터를 잇는다.

- 커넥터를 선택하고 속성 패널의 **Controller** 필드에서 컨트롤러를 고르거나,
- 컨트롤러에서 커넥터로 `Shift` + 드래그한다.

6.5.4 포트와 포트 잇기 (물리 케이블)

1. Coupling Port 노드를 우클릭한다.
2. **Connect to Port...** 에서 상대 포트를 고른다. 두 포트 사이에 선이 생긴다.

케이블은 1:1 이므로 이미 연결된 포트는 목록에 나오지 않는다.

6.5.5 `Shift` + 드래그로 직접 잇기

`Shift` 를 누른 채 **Connector**, **Controller**, **Cluster**, **Coupling Port** 에서 대상까지 드래그한다. 커서 모양으로 그 대상에 연결할 수 있는지 알 수 있다.

> 조건에 맞지 않는 대상에 놓으면 **아무 일도 일어나지 않고 메시지도 나오지 않는다.** 먼저 프로토콜과 소속 ECU 를 확인한다.

6.5.6 연결 끊기

- 커넥터 - 클러스터: 연결선 우클릭 ▸ **Disconnect**, 또는 커넥터 속성 패널의 **Connected Channel** 을 비운다.
- 컨트롤러 - 커넥터: 연결선 우클릭 ▸ **Disconnect**.
- 포트 - 포트: 연결선 우클릭 ▸ **Delete Connection**.

커넥터의 메뉴에는 **Disconnect** 가 없다.

6.5.7 연결선 우클릭 메뉴와 경로 편집

연결선을 선택하고 경로 점 핸들을 드래그하면 배선을 직접 고칠 수 있다. **To Step Edge** 는 자동 경로로 되돌리고, **To Straight Edge** 는 직선으로 만든다. 경로 편집은 커넥터-클러:

6.6 컴포지션 매핑 (SWC - ECU)

VFB 컴포지션을 ECU 나 HPC 노드에 놓는다.

1. 노드를 우클릭하고 **SW Mapping** 을 연다(6.4.2 의 그림 참고).
2. 컴포지션을 고르거나, **New Composition** 으로 노드 이름과 같은 컴포지션을 만든다.

매핑은 1:1 이다. 체크 표시가 있는 항목을 고르면 매핑이 풀린다. 매핑은 `design/SYSTEM.io.model` 에 기록되고 SYSTEM 패널의 **ECU MAPPING** 목록에 보인다. 반대 방향은 0E

6.7 설정 다이얼로그

노드를 더블클릭하면 그 종류의 설정 다이얼로그가 열린다.

6.7.1 어떤 노드가 어떤 다이얼로그를 여는가

ECU, HPC, 클러스터, Ethernet 스위치는 각자의 다이얼로그를 연다. Coupling Port 는 스위치의 다이얼로그를 열거나, 컨트롤러가 소유한 포트면 자신의 다이얼로그를 연다. 커넥터

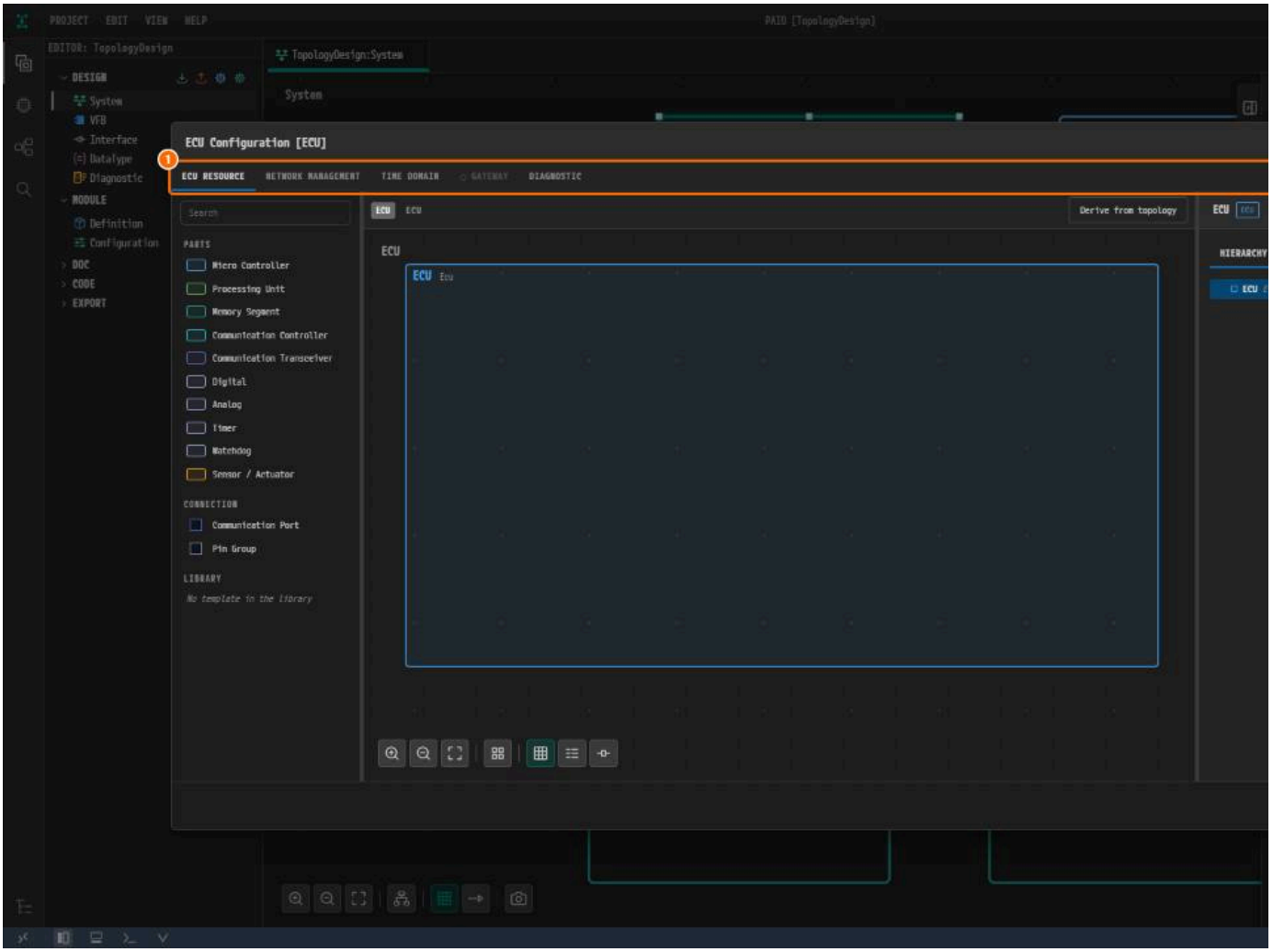
다.

6.7.2 모든 설정 다이얼로그의 공통 규칙

- 편집은 작업 사본에서 이루어진다. 다이얼로그에는 **Close** 버튼만 있고, `Ctrl+S` 로 저장한다.
- `Ctrl+Z` 는 먼저 다이얼로그 자체의 이력을 되돌린다.
- 클러스터 다이얼로그의 **CLUSTER** 탭과 **GATEWAY** 탭의 **Route** 는 프로젝트를 바로 바꾼다.
- 제목 앞의 점(•)은 저장하지 않은 변경이 있다는 뜻이다.

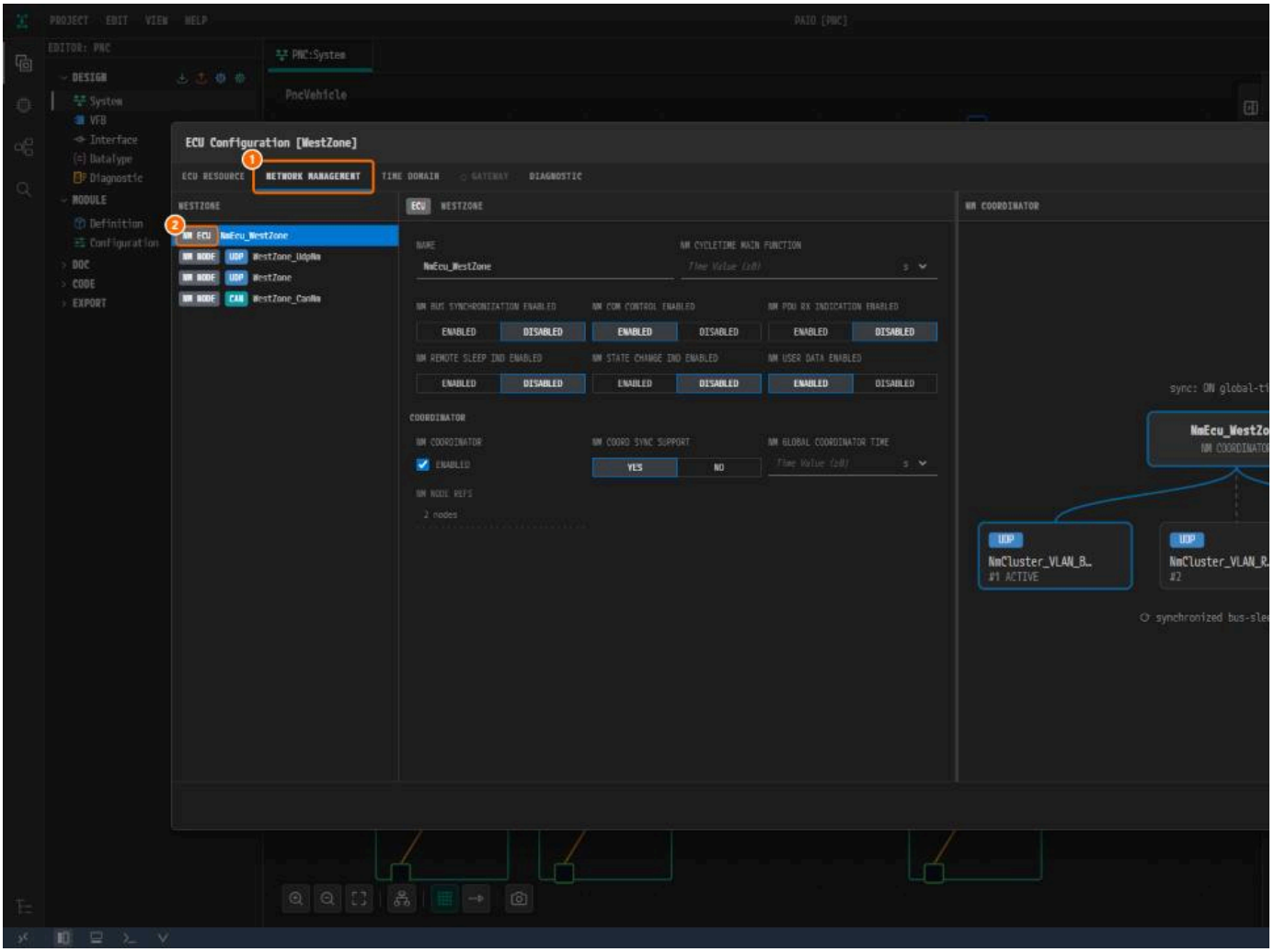
6.7.3 Node Configuration (ECU / HPC)

ECU 다이얼로그에는 **ECU RESOURCE**, **NETWORK MANAGEMENT**, **TIME DOMAIN**, **GATEWAY**, **DIAGNOSTIC** 탭이 있다. HPC 에는 **GATEWAY** 가 없다.



*① 탭 줄. 노드에 게이트웨이가 없으면 **GATEWAY** 는 흐리게 보인다.*

NETWORK MANAGEMENT 목록은 NM 을 켠 클러스터에서 자동으로 만들어지며, 직접 만들지 않는다.

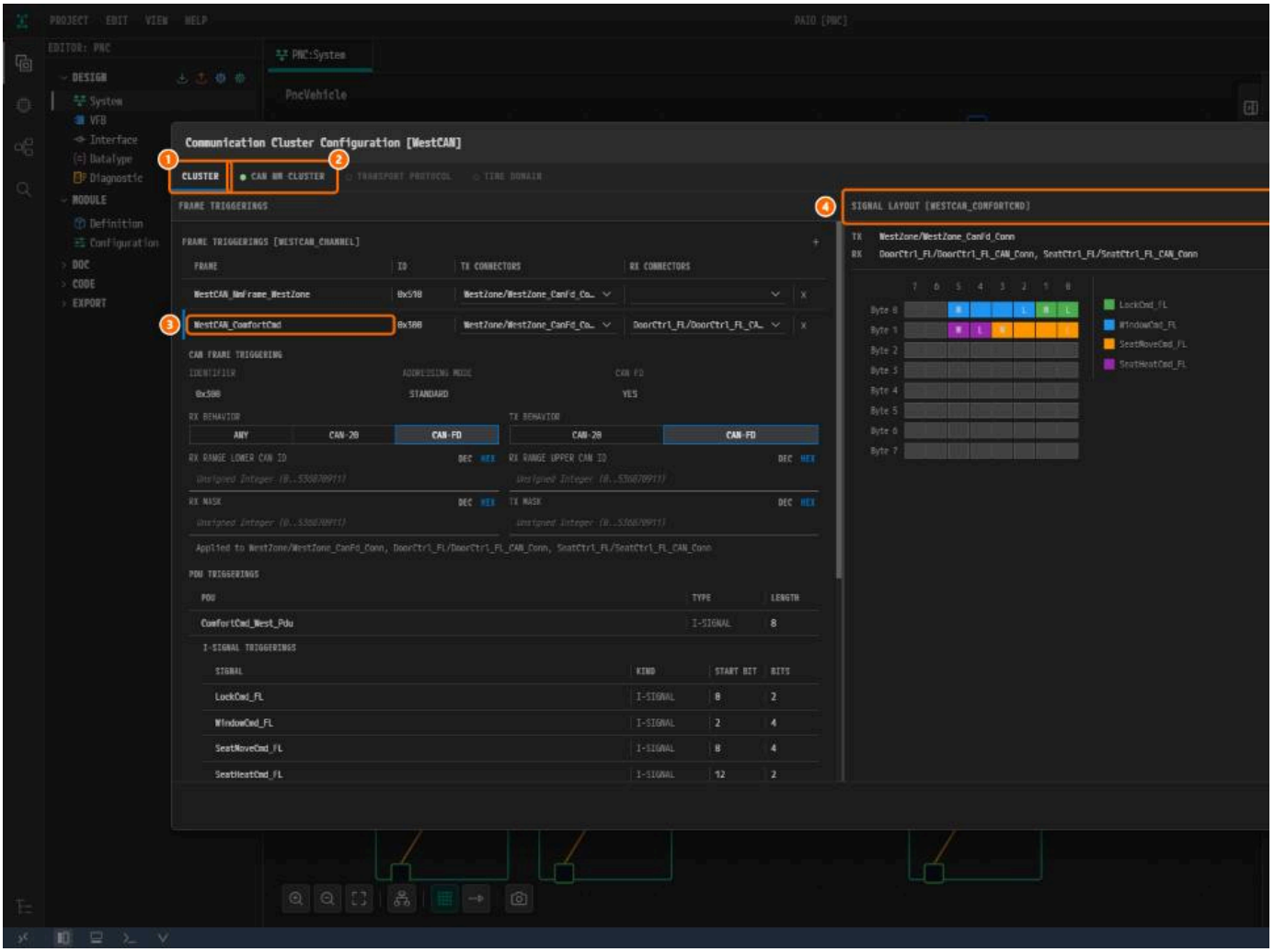


*① **NETWORK MANAGEMENT** 탭. ② `NM ECU` 항목 `NmEcu_WestZone` 과 그 아래의 `NM NODE` 항목.*

GATEWAY 를 쓰려면 노드 메뉴의 **Add Gateway** 를 누르거나, 탭을 우클릭하고 **Enable** 을 고른다.

6.7.4 Cluster Configuration

첫 탭은 항상 **CLUSTER** 이다. 나머지 탭은 프로토콜에 따라 다르다. NM 탭(CAN, Ethernet), **TRANSPORT PROTOCOL**(CAN, LIN), **TIME DOMAIN**(CAN, Ethernet)이다.



*① **CLUSTER** 탭. ② 켜져 있는 **CAN NM CLUSTER** 탭. ③ 펼친 프레임 행 `WestCAN_ComfortCmd`. ④ 그 프레임의 **SIGNAL LAYOUT**.*

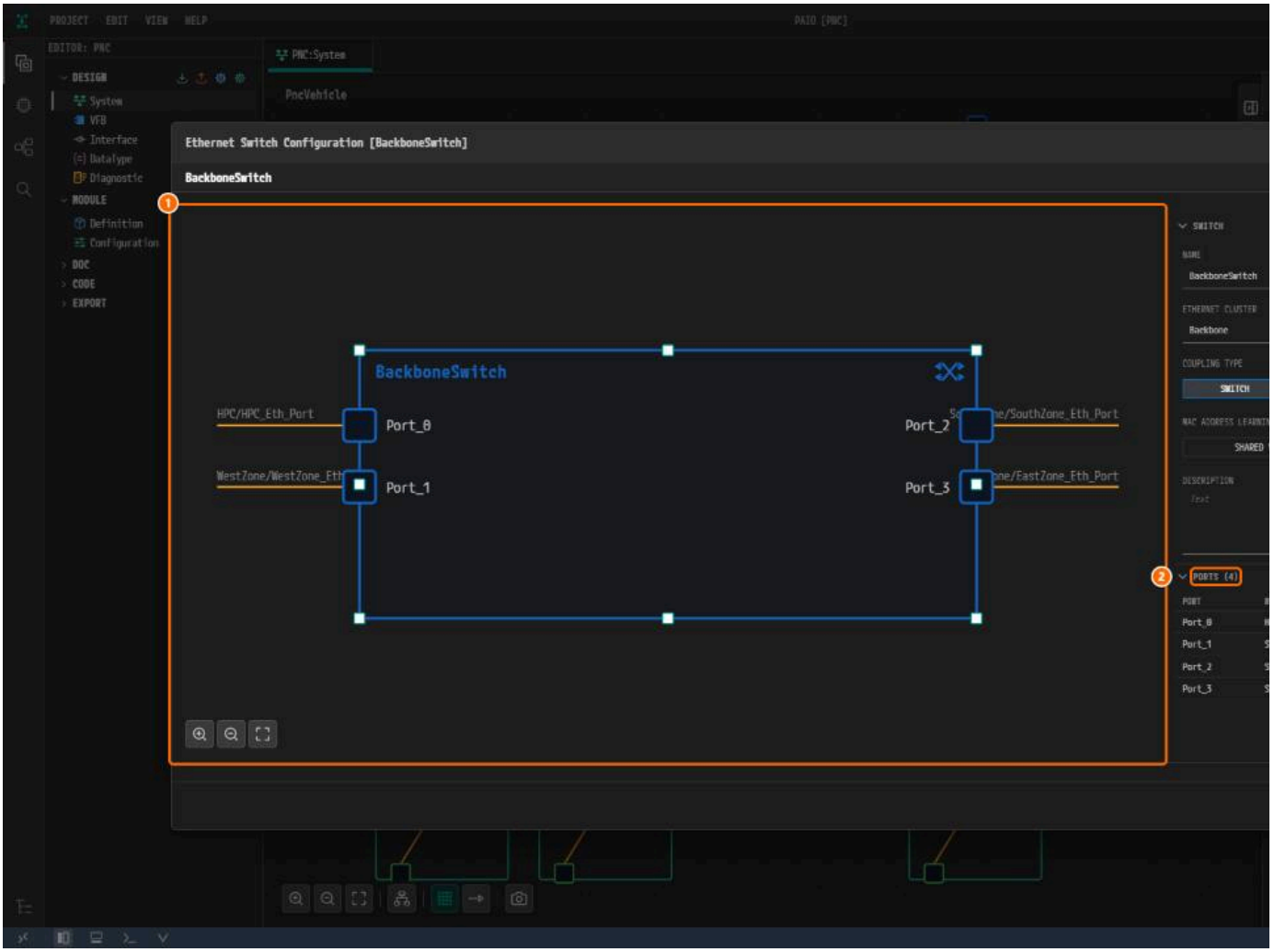
CLUSTER 탭에서 채널에 프레임을 추가하는 방법은 다음과 같다.

1. 채널 표 머리글의 **+** 를 누르고 Signal 에디터의 프레임이나 PDU 를 고른다.
2. **TX CONNECTORS** 와 **RX CONNECTORS** 칸에서 커넥터를 고른다.

편집은 바로 프로젝트에 반영된다. NM, **TRANSPORT PROTOCOL**, **TIME DOMAIN** 은 새 클러스터에서 꺼져 있다. 탭을 우클릭하고 **Enable** 을 고른다.

6.7.5 Ethernet Switch Configuration

다이얼로그는 두 단계이다. 스위치 페이스플레이트, 그리고 포트 하나의 파이프라인이다. 포트를 더블클릭하면 아래 단계로 내려가고, 브레드크럼(`Switch > Port X`)이나 `<` `>` 로



① `Port_0` ~ `Port_3` *가 있는 페이스플레이트. 포트를 선택하지 않으면 오른쪽에 스위치 자체가 보인다.* ② ***PORTS (4)***.

- 페이스플레이트에서 포트를 누르면 오른쪽에 상세가 보인다. 오류는 아래쪽 바에 나온다.
- 포트 단계에서는 블록(**PORT**, **PRIORITY REGENERATION**, **TRAFFIC CLASS ASSIGNMENT**, **EGRESS**)을 눌러 편집한다.
- EGRESS 에디터에서는 캔버스를 우클릭해 FIFO, 스케줄러, 프리셋을 추가한다.

ATS 셰이퍼는 UI 로 만들 수 없다.

6.7.6 Node DIAGNOSTIC 탭

DIAGNOSTIC 탭에는 노드의 진단 주소가 16진수로 들어간다. **ADDRESS** 는 ECU 와 HPC 에, **CONNECTIONS** 는 ECU 에, **DOIP** 는 Ethernet 커넥터가 있는 ECU 에 보인다.

6.7.7 Node GATEWAY 탭

ECU 의 **GATEWAY** 탭은 게이트웨이가 한 버스에서 다른 버스로 중계하는 경로를 편집한다. 왼쪽에 **FRAME**, **PDU**, **SIGNAL** 구역이 있고, 오른쪽은 **ROUTING MAP** 이다.

1. **ROUTING MAP** 에서 수신 항목을 누른 뒤 불이 켜진 목적지를 누르거나, 구역의 **+** 를 누른다. **ROUTE** 시트가 열린다.
2. **SOURCE** 와 **DESTINATION** 을 확인한다.
3. **Route** 를 누른다. 경로가 실행 취소 한 단계로 기록된다.

6.7.8 ECU RESOURCE 탭

ECU RESOURCE 탭은 노드의 하드웨어를 기술한다.

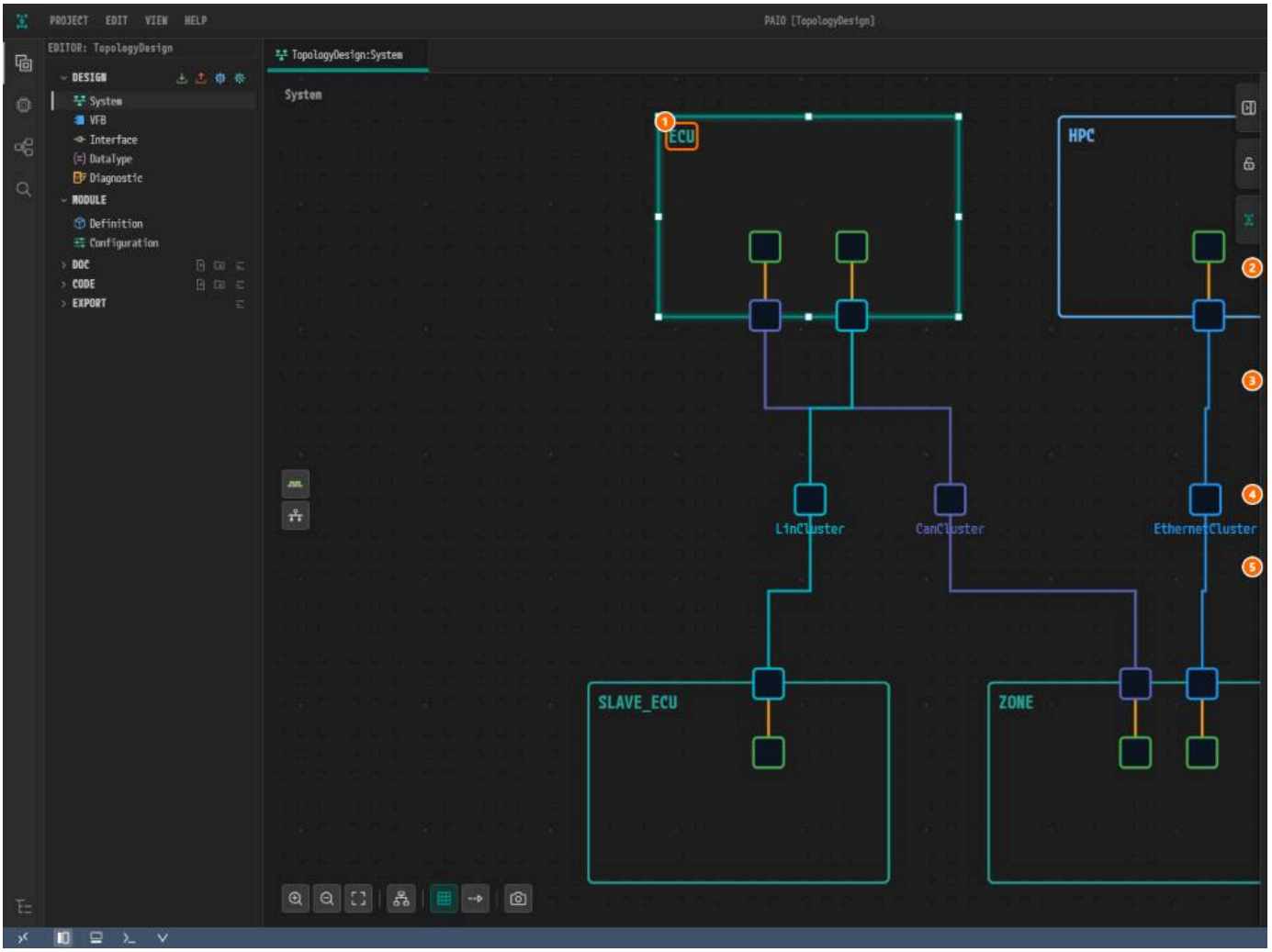
- 팔레트의 **PARTS** 나 **CONNECTION** 에서 부품을 노드 상자 안으로 드래그한다.
- **Derive from topology** 는 System 토폴로지로 상자를 한 번에 채운다.

6.8 속성 패널 (우측)

노드나 연결선을 선택하면 오른쪽 속성 패널에 그 항목이 아코디언으로 나온다.

6.8.1 노드 종류별 구성

ECU 는 **NAME**, **DESCRIPTION**, **CONNECTORS**, **CONTROLLERS**, **POWER MANAGEMENT**, **PARTIAL NETWORK**, **COM MAIN FUNCTION** 을 보여 준다. 클러스터는 프로토콜 설정:



*① 선택한 ECU. ② **CONNECTORS**. ③ **CONTROLLERS**. ④ **POWER MANAGEMENT**. ⑤ **PARTIAL NETWORK**.*

6.8.2 커넥터 선택 시

커넥터는 Name, Controller, Connected Channel, Description 과 프로토콜별 구역(예: **TX Frames** / **RX Frames**)을 보여 준다. 프레임과 PDU 목록은 Signal 에디터에서 읽어

6.8.3 컨트롤러 선택 시

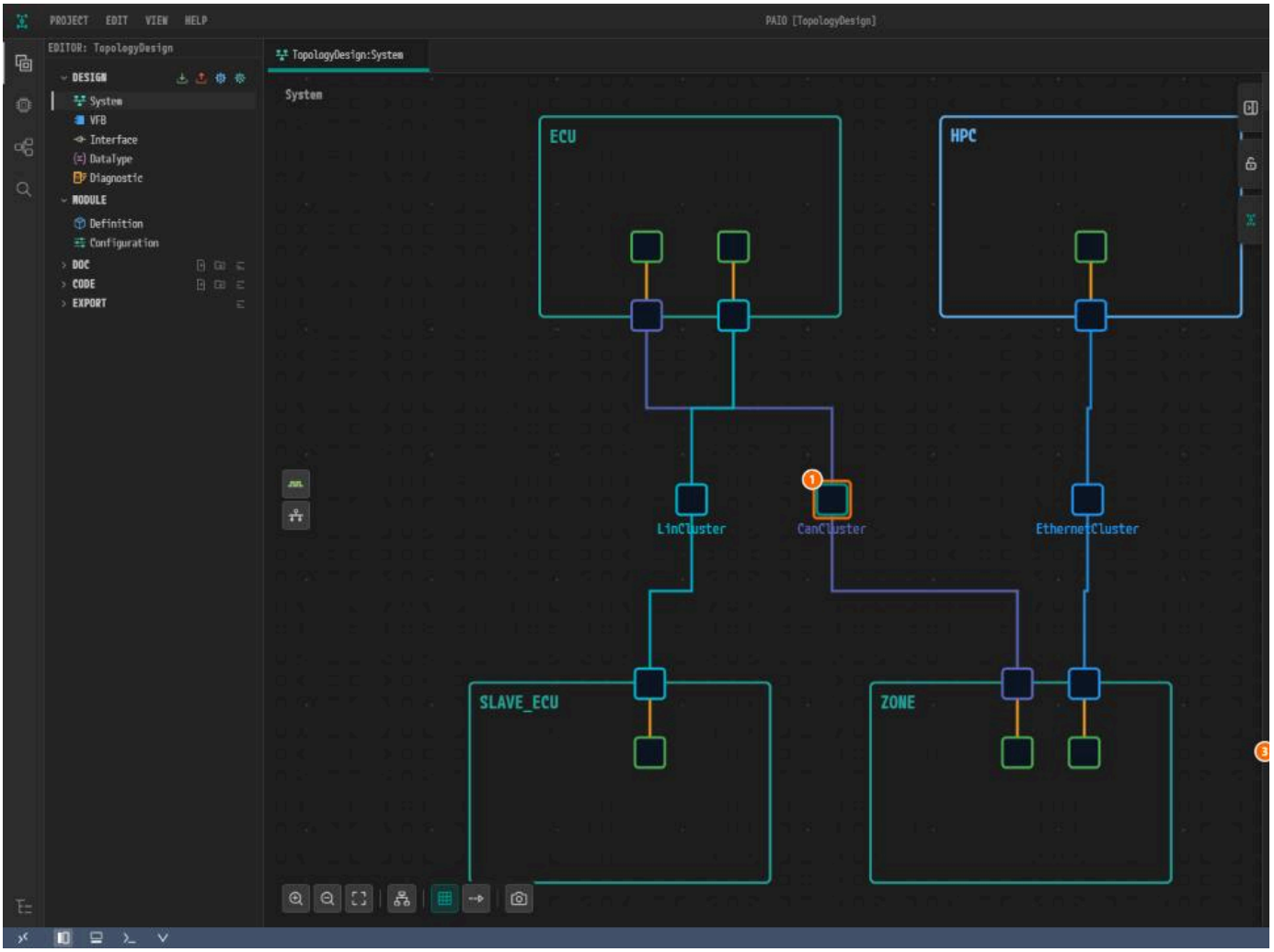
컨트롤러는 Name 과 Description 다음에 종류(Ethernet, CAN, LIN Master, LIN Slave)별 구역을 보여 준다.

6.8.4 연결선 선택 시

포트-포트 연결선은 **First Port** 와 **Second Port** 를 읽기 전용으로 보여 준다. 다른 연결선은 아무것도 보여 주지 않는다.

6.8.5 물리 채널 편집

클러스터를 선택하고 **PHYSICAL CHANNELS** 에서 채널을 편집한다. 채널 이름, VLAN ID(Ethernet, 1 ~ 4094 또는 빈 값), 유휴 타임아웃(LIN)이다.



① 선택한 클러스터. ② `500` *으로 설정된* ***BAUDRATE (ARBITRATION, KBPS)****. ③ 채널 아래의* `Connected Connectors (2)`*.*

채널에 커넥터를 추가하는 방법은 다음과 같다.

1. `Connected Connectors` 줄 오른쪽의 **Add Connector** 를 누른다.
2. `Select Connector` 팝업에서 커넥터를 고르고 **Add** 를 누른다.

6.8.6 LIN 스케줄 테이블

LIN 클러스터를 선택하고 **SCHEDULE TABLES** 에서 스케줄 테이블을 편집한다. **+** 로 테이블을 추가하고, `Entries (n)` 에 그 엔트리가 들어간다.

6.9 ECUC 생성

System 에디터에는 ECUC 구역이 없다. ECU 의 BSW 모듈 설정은 이 에디터의 토폴로지로부터 VFB 에디터의 **PRTE AUTO CONFIGURATION**(8. VFB 에디터)이나 Module Confi
(12.2.6)으로 생성한다. 12. ECUC를 참고한다.

6.10 자동 레이아웃

1. 툴바의 **Auto Layout** 을 누른다.
2. 손으로 고친 연결선 경로가 있으면 확인 창이 나온다. **Replace** 를 누르면 계속한다.
3. 끝나면 모든 노드가 보이도록 화면이 맞춰진다.

실행 취소 한 번으로 레이아웃 전체가 되돌아간다.

6.11 다른 에디터와의 연동

- 컴포지션 매핑은 SYSTEM 패널과 VFB 에디터에 나타난다.
- 커넥터의 프레임, PDU 목록은 Signal 에디터의 내용을 따른다.

- ECU 나 HPC 의 이름을 바꾸거나 지우면 다른 내용 속의 이름도 같은 실행 취소 단계로 함께 바뀐다.

6.12 이름 규칙

이름은 영문자로 시작하고 영문자, 숫자, 밑줄만 쓸 수 있으며 128 자 이하이다. 규칙에 맞지 않는 이름은 저장되지 않고 마지막으로 저장된 값으로 돌아간다.

6.13 표시와 실제 동작이 다른 항목

저장하지 않은 편집이 있는 동안 파일이 밖에서 바뀌면 다시 읽지 않고 경고만 띄운다. 편집을 지키려면 저장하고, 디스크의 내용을 받으려면 탭을 닫고 다시 연다.

6.14 Partial Network (PNC) Configuration

액션 줄의 **Partial Network (PNC)** 아이콘은 **Partial Network (PNC) Configuration** 다이얼로그를 연다. 시스템 전체의 부분 네트워크를 편집하며, `design/SYSTEM.io.mode`

1. 왼쪽 **PARTIAL NETWORKS** 의 **+** 로 PNC 를 추가한다.
2. 가운데 창의 **PHYSICAL CHANNELS** 에서 PNC 에 참여할 채널을 체크한다.
3. `Ctrl+S` 로 저장한다.

Signal 에디터

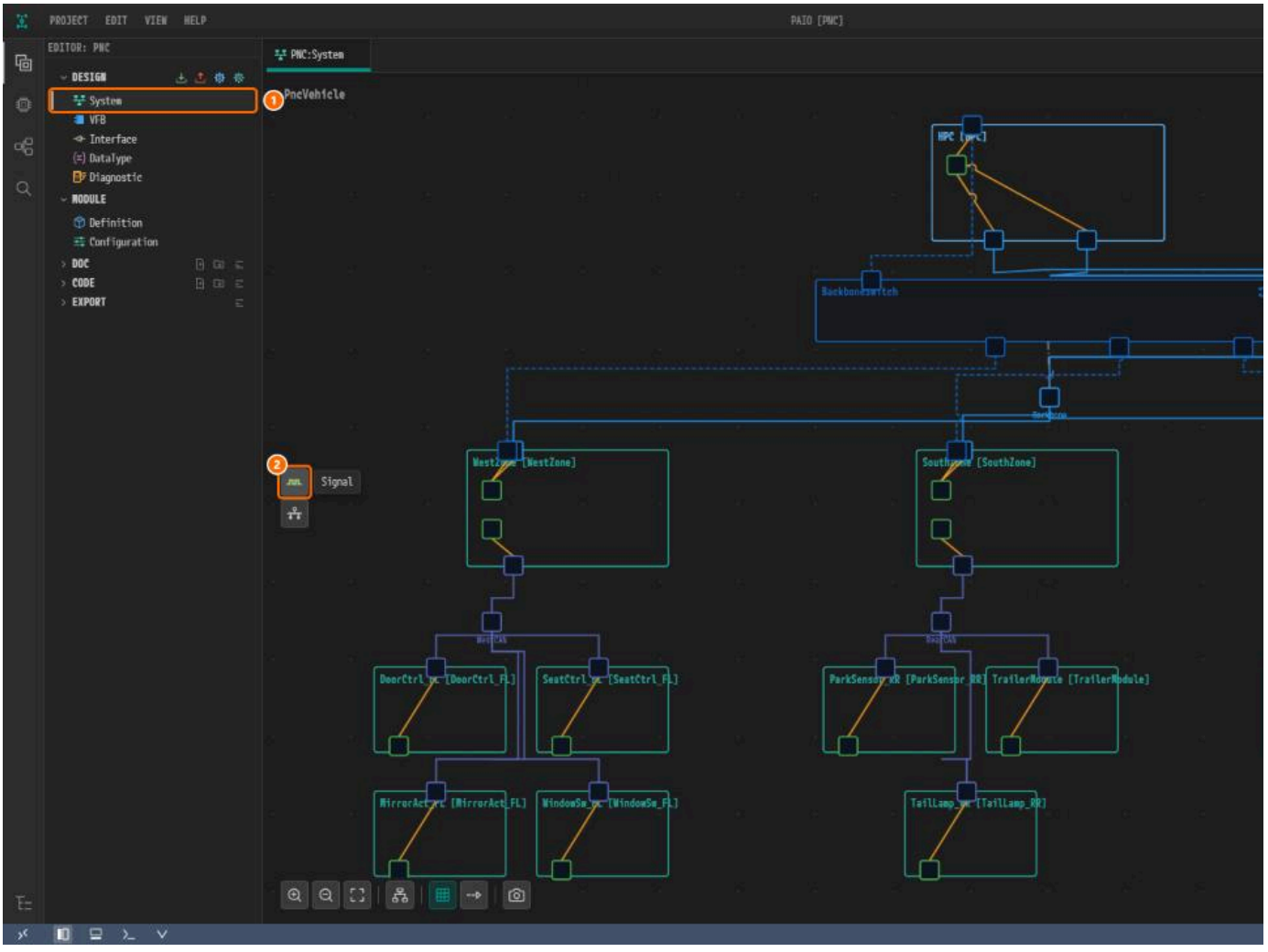
이 장에서 다루는 것

통신 데이터 사전, 즉 CAN/LIN 프레임, 6종 PDU, I-Signal, 시그널 그룹을 편집하는 Signal 에디터의 사용법을 설명한다. DBC/ARXML 임포트와 익스포트 절차는 13. 임포트 · 익스포

- 대상 독자: 버스 통신 매트릭스를 작성하거나, 임포트한 매트릭스를 검토하고 수정하는 사용자
- 선행 조건: PAIO 에서 프로젝트가 열려 있는 것

Signal 에디터는 **EDITOR** 뷰의 **DESIGN** 아래에 행이 없다. System 에디터에서 연다.

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System** 을 누른다. 탭 `<project>:System` 이 열린다.
2. 캔버스 왼쪽 가장자리 아이콘 줄의 맨 위 아이콘 **Signal** 을 누른다. 탭 `<project>:Signal` 이 열린다.

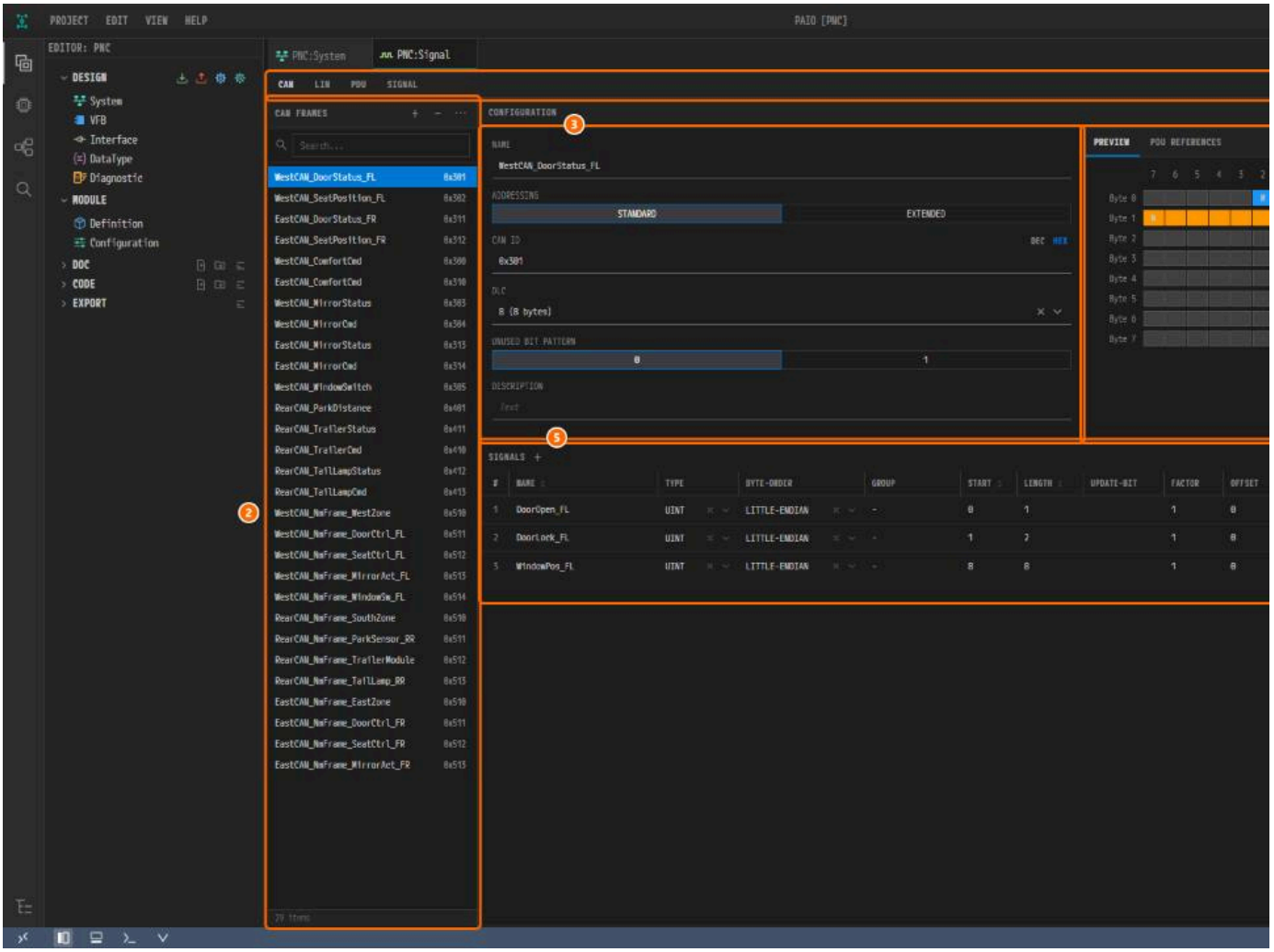


*① **DESIGN** 아래의 **System**, ② **Signal** 아이콘.*

에디터는 파일 하나, `design/SIGNAL.io.model` 에 저장한다. I-Signal과 System Signal처럼 헷갈리기 쉬운 용어는 19. 용어 사전에 있다.

7.1 화면 구성

맨 위에 프로토콜 탭이 있다. 그 아래 왼쪽에 목록 패널이, 나머지 폭에 Configuration 패널이 있다.



*① 프로토콜 탭, ② **CAN FRAMES** 목록, ③ Configuration 폼, ④ **PREVIEW** 와 **PDU REFERENCES** 탭, ⑤ **SIGNALS** 표.*

- 비트 레이아웃은 Configuration 폼 안의 **PREVIEW** 탭이다. 별도의 Preview 패널은 없다.
- 프레임, I-Signal I-PDU, NM-PDU 에서는 폼 아래에 **SIGNALS** 표가 나온다.
- **SIGNAL GROUPS** 패널은 SIGNAL 탭 오른쪽에만 나온다.

7.2 프로토콜 탭

탭	편집 대상
CAN	CAN / CAN FD 프레임
LIN	LIN 프레임
PDU	NM-PDU 를 포함한 6종 PDU 통합 목록
SIGNAL	시그널(I-SIGNAL) 사전

FlexRay 탭과 Ethernet 탭은 없다. 임포트한 FlexRay, Ethernet 데이터는 파일에 유지되고 익스포트에도 포함된다.

7.3 왼쪽 목록 패널

목록은 현재 탭의 항목을 보여 준다.

헤더 버튼

+ 는 항목을 추가하고, **−** 는 확인을 거쳐 선택한 항목을 삭제하고, **⋯** 는 필터와 정렬 팝오버를 연다(7.4).

여러 항목 선택

`Ctrl`+클릭하면 항목이 선택에 더해진다. 목록에 포커스를 두고 `Ctrl+A` 를 누르면 모든 항목이 선택된다.

검색과 상태 표시줄

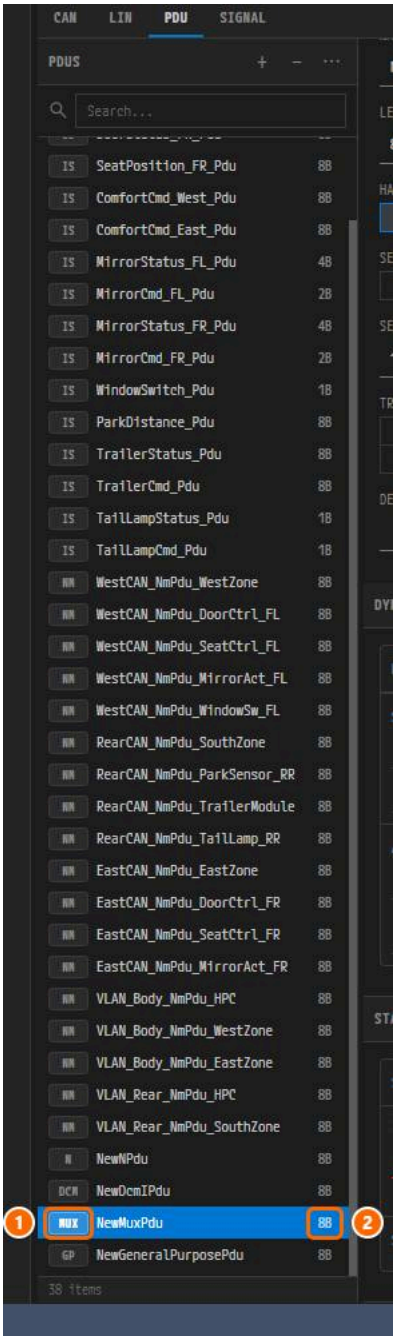
`Search...` 입력은 이름으로 목록을 거른다. `Esc` 를 누르면 지워진다. 목록 아래 상태 표시줄은 항목 수를 보여 준다.

빈 상태와 검색 결과 없음

빈 탭에는 `Add a frame` 같은 안내가 나온다. 검색에 맞는 것이 없으면 `No results found` 가 나온다.

목록 항목의 배지

각 항목에는 배지가 붙는다. CAN, LIN 탭은 프레임 ID, PDU 탭은 종류 태그와 길이, SIGNAL 탭은 비트 길이이다.

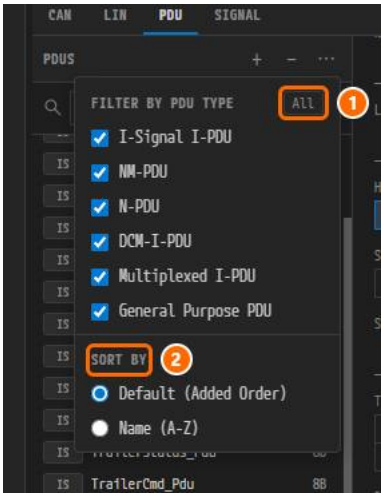


① `MUX` 종류 태그, ② 길이 배지 `8B`.

`SOC`, `COM` 배지는 도구가 만든 항목을 표시한다. 배지는 출처만 알리며, 그 항목도 편집하고 삭제할 수 있다.

7.4 More(⋯) 메뉴 — 필터와 정렬

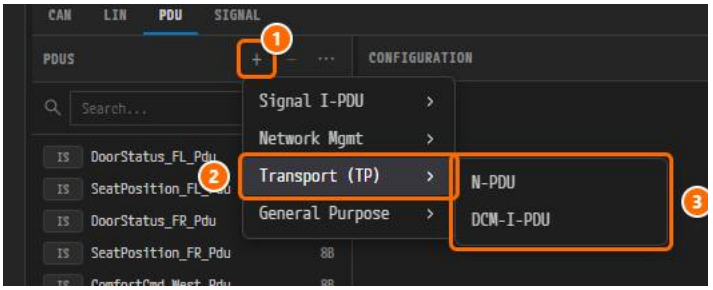
⋯ 를 누르면 PDU 종류로 거르고(PDU 탭만) 목록을 정렬할 수 있다. 정렬은 화면 표시 순서만 바꾸며 파일의 순서는 그대로이다.



*① **FILTER BY PDU TYPE**, ② **SORT BY**.*

7.5 PDU 6종

1. PDU 탭에서 **+** 를 누른다. 2단 메뉴가 열린다.
2. 왼쪽의 그룹, 예를 들어 **Transport (TP)** 에 마우스를 올린다.
3. 하위 메뉴에서 종류를 누른다. PDU 가 만들어진다.



*① **+**, ② **Transport (TP)**, ③ **N-PDU / DCM-I-PDU**.*

종류는 I-Signal I-PDU, Multiplexed I-PDU, NM-PDU, N-PDU, DCM-I-PDU, General Purpose PDU 이다. 이미 있는 이름에는 `NewPdu_1` 처럼 번호가 붙는다.

7.6 항목 추가와 삭제

프레임 추가 (CAN / LIN 탭)

1. CAN 또는 LIN 탭을 고르고 **+** 를 누른다.
2. 새 프레임이 그 프레임 전용 PDU 와 함께 만들어지고 선택된다.

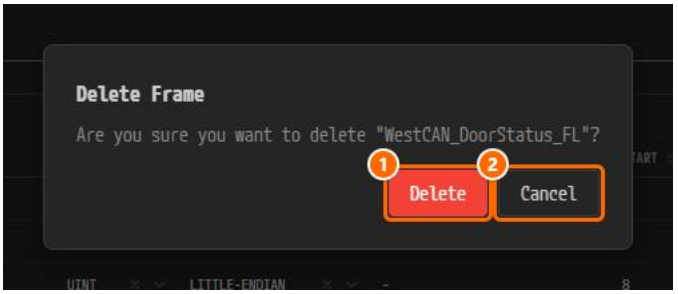
시그널 추가

- **SIGNAL 탭의 +** 는 사전에만 시그널을 추가한다. 어느 PDU 에도 매핑되지 않는다.
- **SIGNALS 표 머리의 +** 는 시그널을 만들고 선택한 프레임이나 PDU 에 매핑한다. PDU 가 짧으면 PDU 길이가 늘어나고, PDU 가 CAN 프레임의 DLC 를 넘으면 DLC 도 올라간다.

프레임이 가득 차면 추가가 조용히 무시된다. 프레임이나 PDU 배치를 먼저 정리한다.

삭제

1. 목록에서 항목을 선택한다.
2. **−** 를 누르거나 `Delete` 를 누른다.
3. 확인 대화상자에서 **Delete** 를 누른다.



*① **Delete**, ② **Cancel**.*

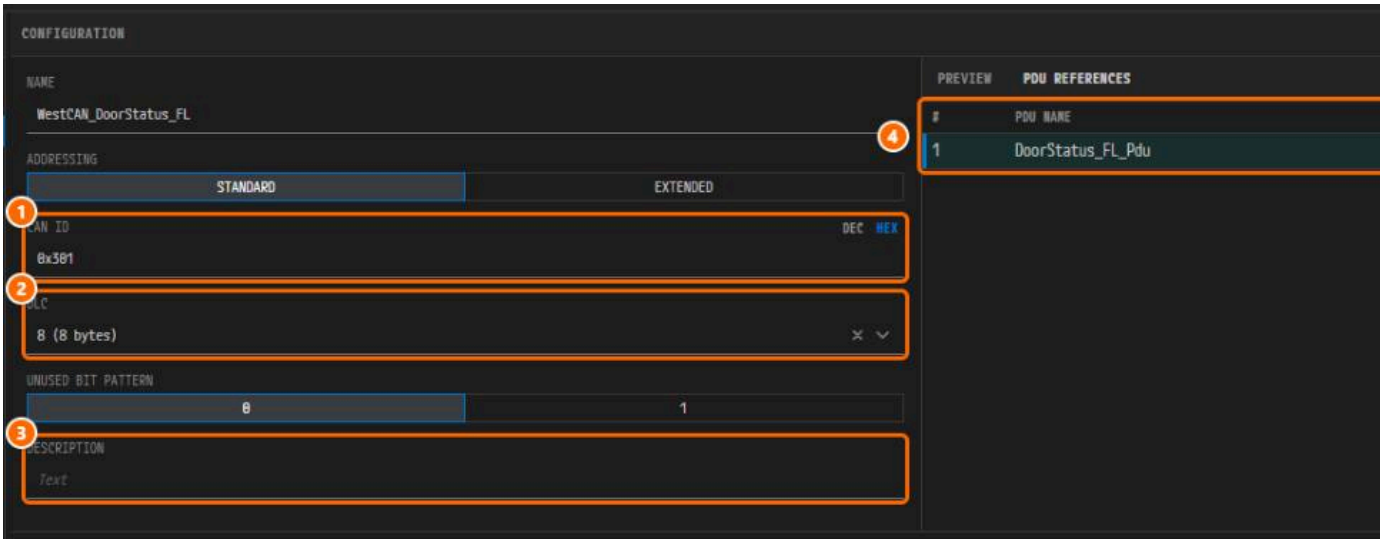
- **SIGNALS** 표 행 끝의 × 는 대화상자 없이 시그널을 바로 지우고, 사전에서도 지운다. 되돌리려면 **EDIT** > **Undo** 를 쓴다.
- 데이터 매핑이 걸린 시그널은 표에서 지운다. 표의 × 만 System 모델의 데이터 매핑 참조까지 정리한다.

7.7 Configuration 패널 — 항목 종류별 편집 필드

Configuration 패널은 선택한 항목의 폼을 보여 준다. 선택 필드는 대문자 글자의 세그먼트 버튼으로 그려진다.

7.7.1 CAN 프레임

Name, Addressing, CAN ID, DLC, Unused Bit Pattern, Description 을 정한다. 오른쪽 열에는 **PREVIEW** 와 **PDU REFERENCES** 탭이 있다.



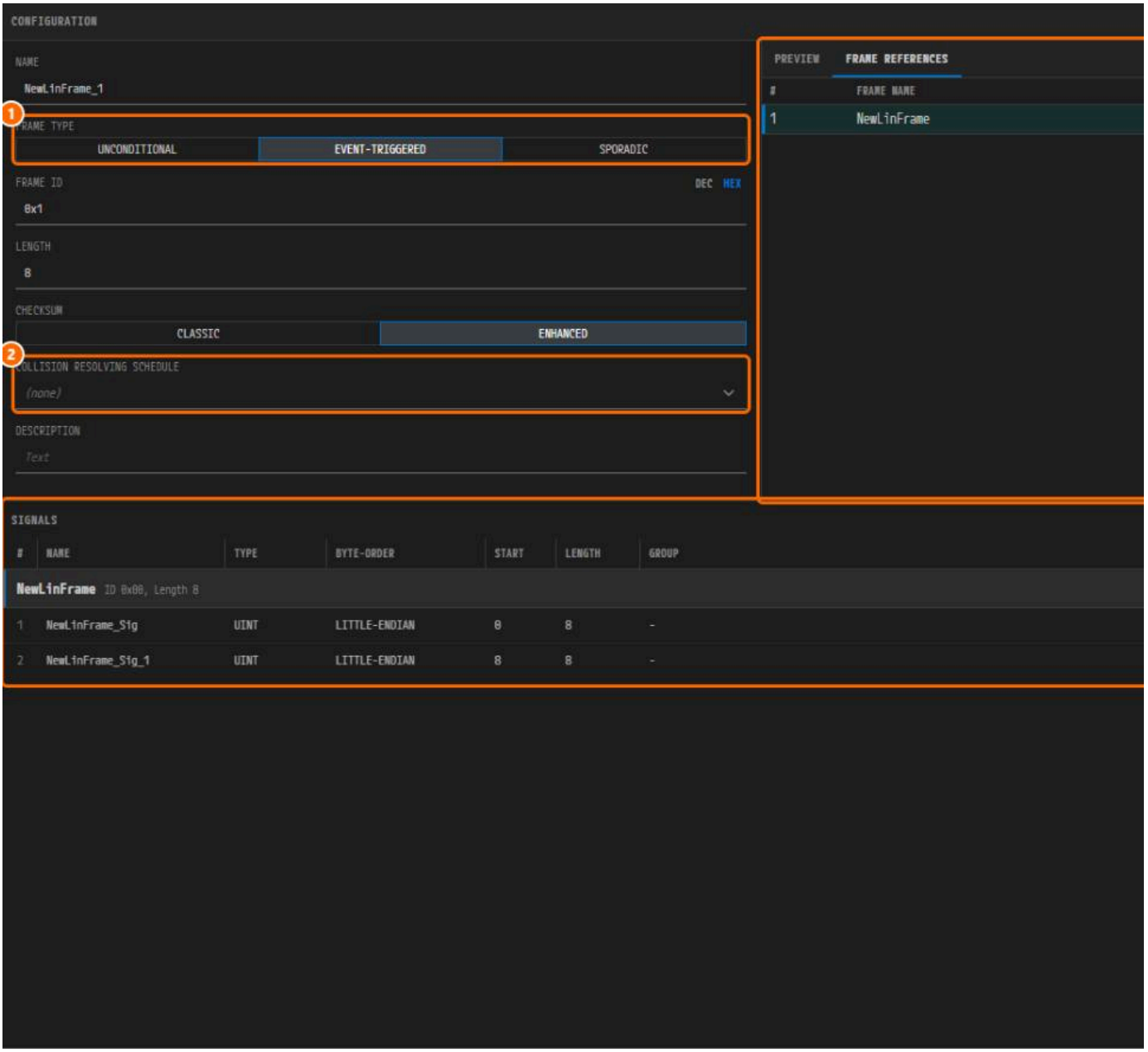
*① **DEC / HEX** 가 있는 **CAN ID**, ② **DLC**, ③ **Description**, ④ **PDU REFERENCES** 표, ⑤ PDU 선택기.*

- CAN ID 범위는 Addressing 을 따른다. `standard` 는 `0x7FF` 까지, `extended` 는 `0x1FFFFFFF` 까지이다.
- DLC 9~15 는 CAN FD 길이이다. 프레임이 PDU 하나만 참조하면 DLC 를 바꿀 때 PDU 길이도 바뀐다.

7.7.2 LIN 프레임

필드는 Frame Type(`unconditional`, `event-triggered`, `sporadic`)에 따라 다르다.

- `unconditional` 프레임에는 편집 가능한 **SIGNALS** 표와 **PDU REFERENCES** 탭이 있다.
- `event-triggered`, `sporadic` 프레임은 자체 시그널이 없다. **FRAME REFERENCES** 탭과, 참조한 프레임들의 시그널을 보여 주는 읽기 전용 목록이 있다. **Collision Resol**



*① **Frame Type**, ② **Collision Resolving Schedule**, ③ **FRAME REFERENCES** 탭과 선택기, ④ 읽기 전용 **SIGNALS** 목록.*

시그널을 고치려면 그 시그널을 가진 unconditional 프레임으로 간다.

7.7.3 I-Signal I-PDU

Name, Length, Unused Bit Pattern, Cycle Time (ms), Transmission Mode, Description 을 정한다. 폼에는 **PREVIEW** 탭과 **SIGNALS** 표가 있다. Cycle Time 은 밀리초로 저

7.7.4 NM-PDU

Name, Length, Unused Bit Pattern, NM Vote Information, NM Data Information, Description 을 정한다. NM-PDU 도 UserData 시그널을 실을 수 있으므로 같은 **PREVIE**

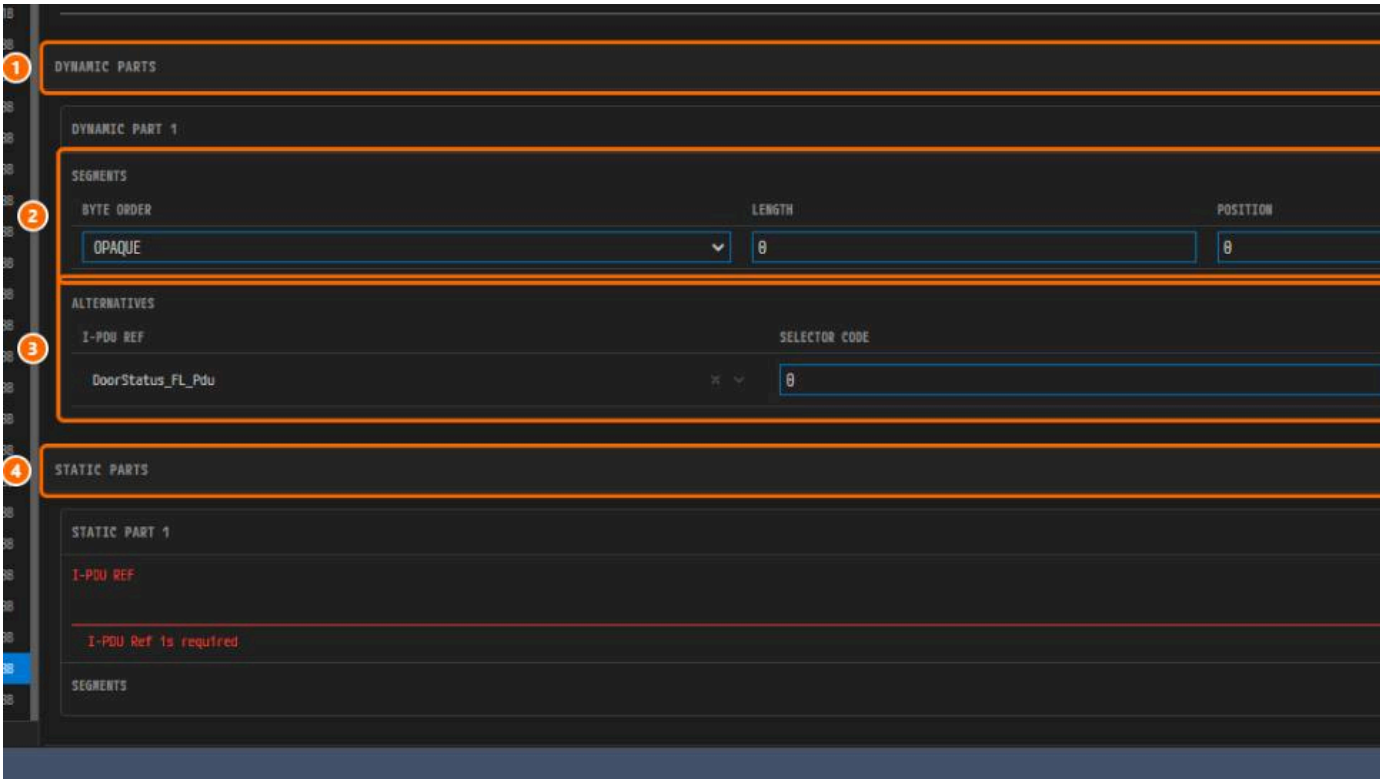
7.7.5 N-PDU · DCM-I-PDU · General Purpose PDU

이 세 종류는 시그널을 갖지 않는다. 폼에는 Name, Length, Description 과 종류별 필드 하나(DCM-I-PDU 는 Diag PDU Type, General Purpose PDU 는 Category)만 있다.

7.7.6 Multiplexed I-PDU

셀렉터 필드와 Trigger Mode 를 정한 뒤, 폼 아래에서 파트를 만든다.

1. `DYNAMIC PARTS` 의 **+** 를 눌러 동적 파트를 추가한다. 그 `SEGMENTS` 와 `ALTERNATIVES` 를 채운다.
2. `STATIC PARTS` 의 **+** 를 눌러 정적 파트를 추가한다. 필수인 `I-PDU REF` 를 지정한다.



① ***DYNAMIC PARTS***, ② ***SEGMENTS***, ③ ***ALTERNATIVES***, ④ *I-PDU Ref 가 비어 있는* `Static Part 1` *이 있는* ***STATIC PARTS***.

`I-PDU REF` 가 비어 있으면 빨간색으로 `I-PDU Ref is required` 가 나온다.

7.7.7 시그널 (SIGNAL 탭)

Name, Bit Length, Type, Unit, Factor, Offset, Min, Max, Init Value(7.11), Description 을 정한다.

- **Signal Groups** 는 표시 전용이다. 그룹은 **SIGNALS** 표의 `GROUP` 열에서 바꾼다(7.9).
- `array` 와 `struct` 는 여기서만 지정할 수 있다. 표는 `UINT`, `SINT`, `FLOAT` 만 제공한다.

7.8 PDU REFERENCES 탭과 FRAME REFERENCES 탭

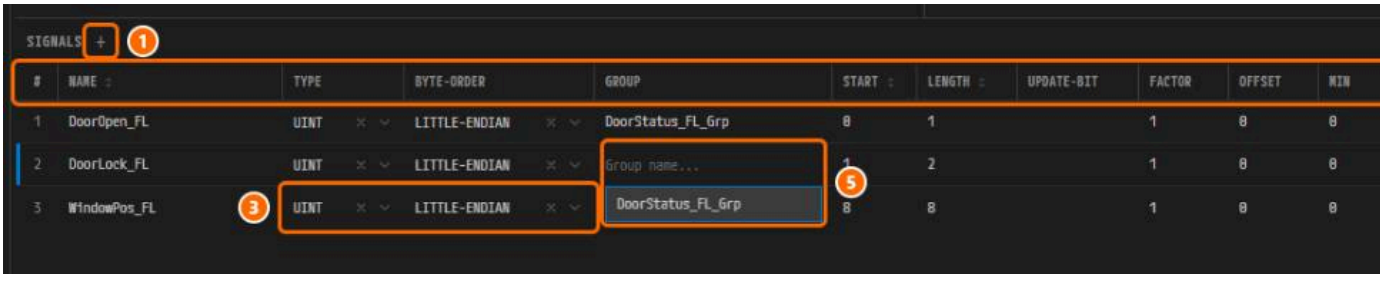
프레임의 참조는 폼 오른쪽 열의 두 번째 탭에서 지정한다.

1. **PDU REFERENCES**(CAN 프레임, LIN unconditional 프레임) 또는 **FRAME REFERENCES**(LIN event-triggered, sporadic 프레임)를 누른다. 탭 바 오른쪽 끝에 선택기가 나
2. 선택기를 누르고 필요한 항목을 체크한다.
3. 드롭다운을 닫는다. 항목이 적용되어 표에 나온다. 2개 이상이면 행을 드래그해 순서를 바꾼다.

두 참조 모두 필수이다. 비어 있으면 탭 레이블이 빨간색이 된다. PDU 가 프레임 크기를 넘거나 시그널 비트가 겹칠 때도 **PDU REFERENCES** 레이블이 빨간색이 되고, 툴팁에 메시지기

7.9 SIGNALS 표

폼 아래의 표는 선택한 프레임, I-Signal I-PDU, NM-PDU 에 매핑된 시그널을 보여 준다. 제목 뒤의 **+** 를 눌러 시그널을 추가한다.



① *+(*`Add Signal`*)*, ② *열 머리*, ③ `TYPE` *과* `BYTE-ORDER`, ④ `PDU` *셀*, ⑤ *편집 중인* `GROUP` *셀*.

열 구성

주요 열은 `NAME`, `TYPE`, `BYTE-ORDER`, `GROUP`, `START`, `LENGTH`, `INIT VALUE`, `PDU` 이다. `NAME`, `START`, `LENGTH` 는 정렬할 수 있다. 비트가 겹치는 행은 강조된다.

셀 편집 절차와 거부되는 입력

1. 셀을 누른다.
2. 값을 바꾸고 `Enter` 를 누르거나 포커스를 옮긴다.
3. 취소하려면 `Esc` 를 누른다.

비어 있거나, 중복되거나, 규칙을 어긴 `NAME` 은 저장되지 않는다. 잘못된 숫자는 원래 값으로 돌아간다.

GROUP 열 자동완성

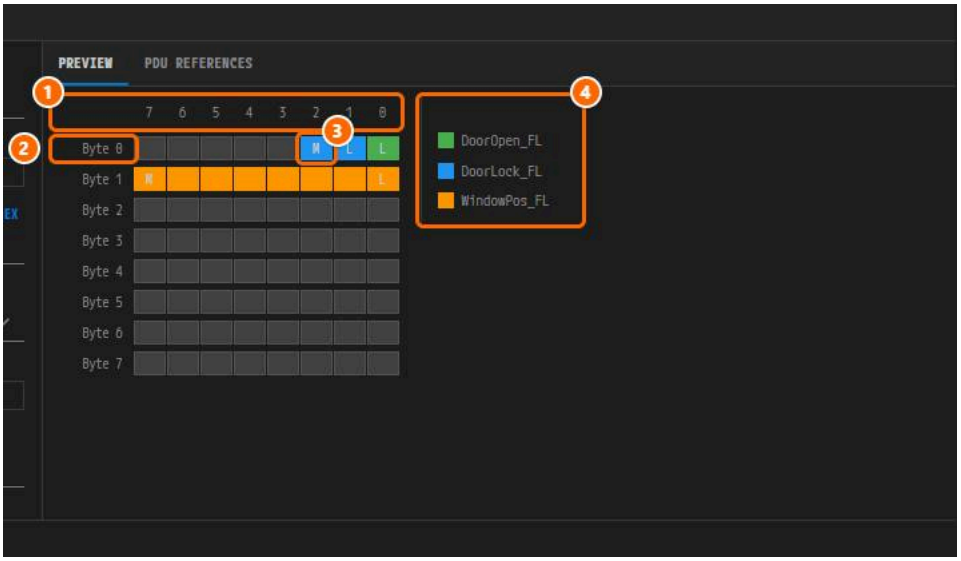
1. `GROUP` 셀을 누른다.
2. 그룹 이름을 입력한다. 후보를 고르거나, `Create "<이름>"` 을 골라 새 그룹을 만든다.
3. 그룹에서 빼려면 값을 비우거나 `-` 를 입력한다.

시그널은 한 번에 한 그룹에만 속한다.

7.10 PREVIEW 탭과 SIGNAL GROUPS 패널

비트 레이아웃 (PREVIEW 탭)

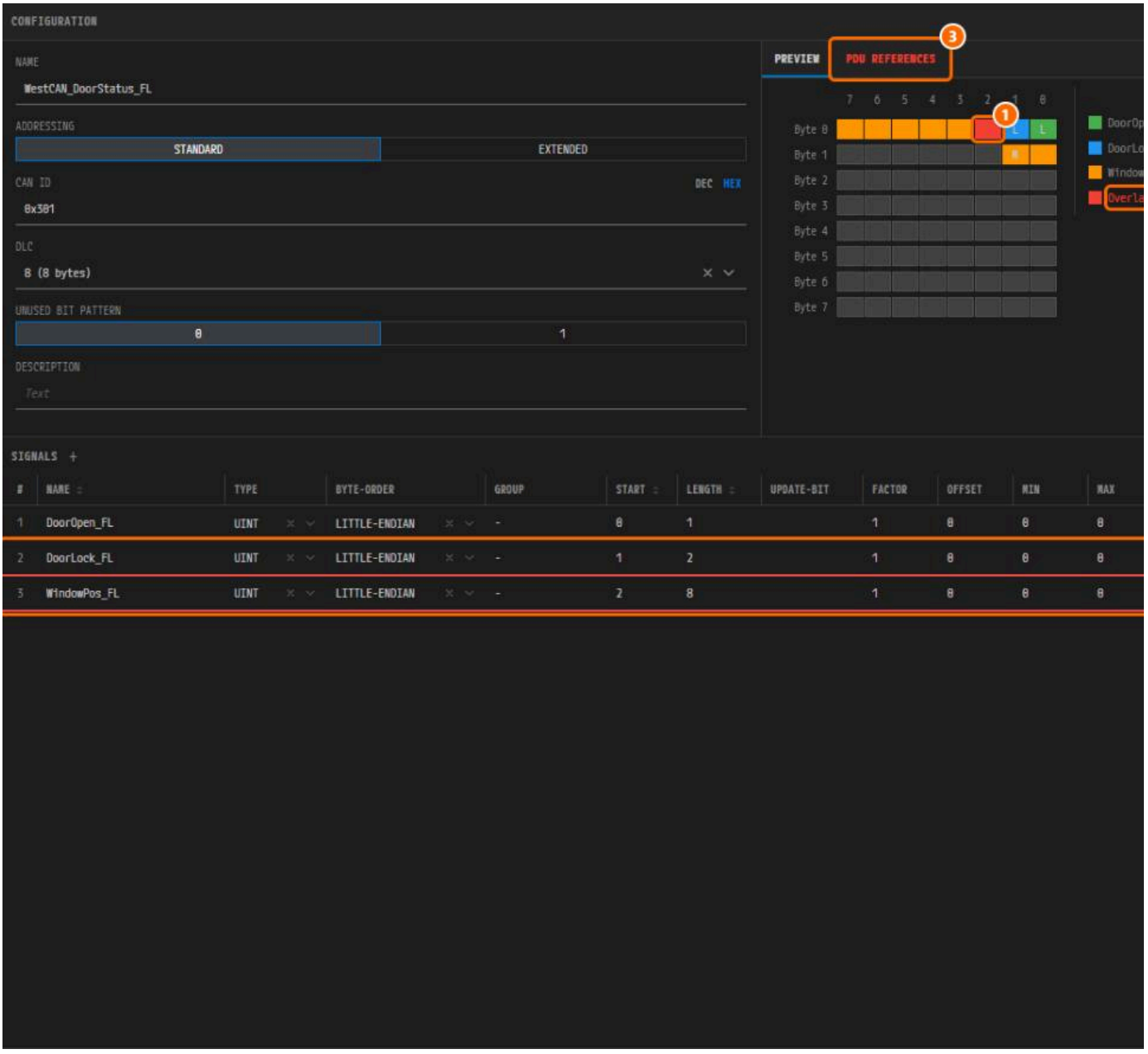
PREVIEW 는 바이트마다 비트를 그리고 시그널마다 색을 칠한다. `L` 은 LSB, `M` 은 MSB, `U` 는 업데이트 비트이다.



① 비트 번호 행, ② 바이트 행 레이블, ③ MSB(`M`) 표시, ④ 범례.

겹침은 오류이다. 다음과 같이 고친다.

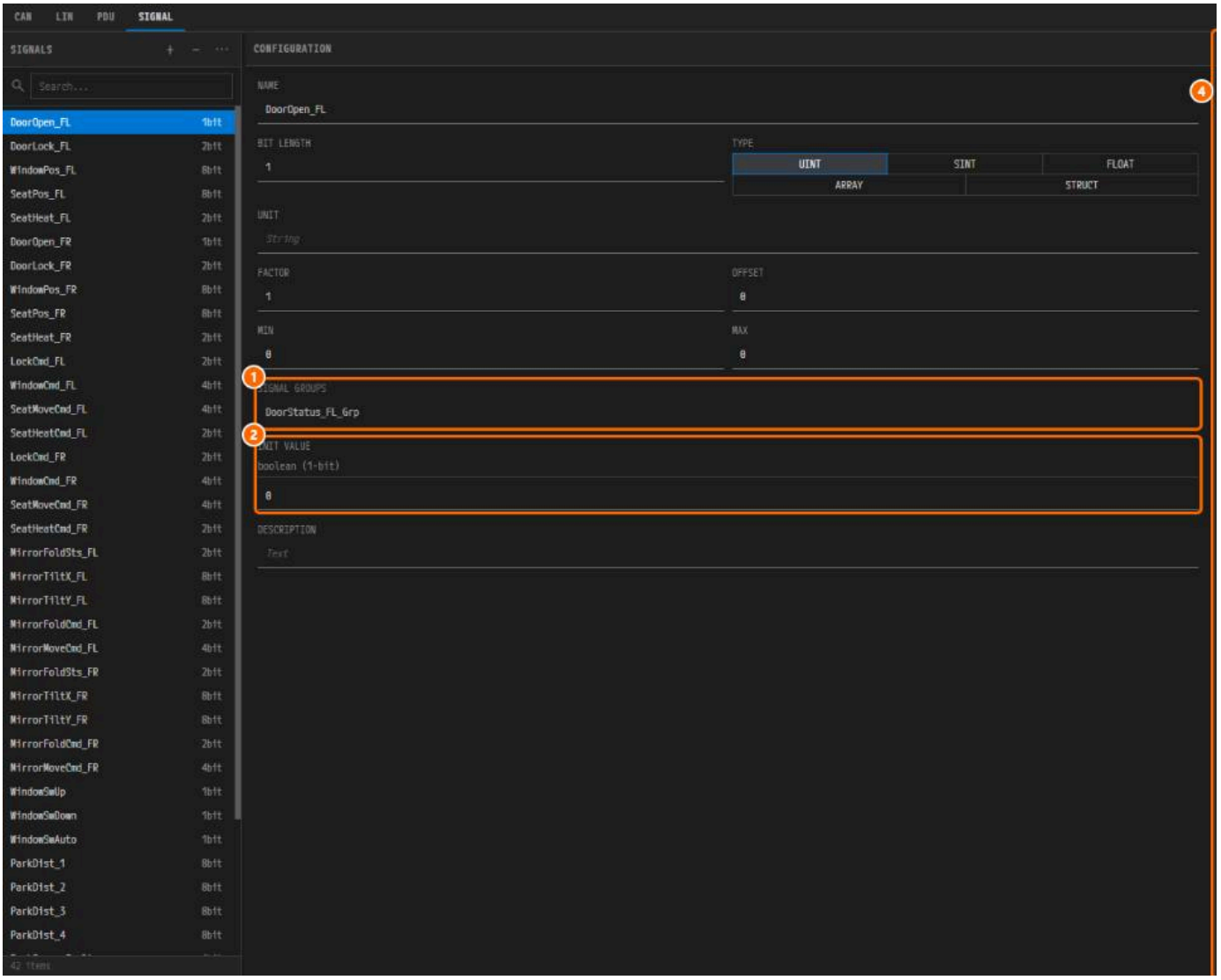
1. **PREVIEW** 에서 겹친 비트를 찾는다. 오류 색으로 칠해져 있다.
2. 두 시그널 중 하나의 **START** 를 바꾼다(7.9).
3. 그래도 자리가 모자라면 마지막 비트가 프레임의 **DLC**(LIN 은 **Length**)를 넘지 않는지 확인한다.



*① 겹친 비트, ② `Overlap (Error)`, ③ 빨간색 **PDU REFERENCES** 레이블, ④ 강조된 행.*

시그널 그룹 트리 (SIGNAL GROUPS 패널)

SIGNAL 탭 오른쪽의 읽기 전용 트리이다. 그룹을 누르면 펼쳐지고, 시그널을 누르면 Configuration 패널에 그 시그널이 열린다.



*① 읽기 전용 **SIGNAL GROUPS** 필드, ② **INIT VALUE**, ③ **SIGNAL GROUPS** 패널, ④ 펼친 그룹 행.*

7.11 초기값 (Init Value)

입력 형태는 시그널 자신의 속성에서 정해진다. Text Table 항목이 있으면 콤보 상자, 1비트 시그널은 0 또는 1, `uint` 와 `sint` 는 범위가 있는 정수이다.

- 모델에는 항상 숫자가 저장된다.
- 범위를 벗어난 값도 받아들이고 오류 표시만 붙인다.

표의 INIT VALUE 열

단순 숫자는 셀에서 바로 편집한다. 구조화된 값은 `STRUCT` 같은 배지로 나오며 편집할 수 없다.

7.12 Text Table (열거)

시그널의 열거 항목은 Configuration 패널의 `Text Table (N entries)` 절에 나온다. 읽기 전용이며 ARXML 이나 DBC 임포트로만 만들어진다.

7.13 저장과 실행 취소

조작	명령
저장	`Ctrl+S`
실행 취소	**EDIT** > **Undo** (`Ctrl+Z`)
다시 실행	**EDIT** > **Redo** (`Ctrl+Y`)

단축키는 에디터 안에 포커스가 있을 때 동작한다. 그렇지 않으면 **EDIT** 메뉴를 쓴다. 실행 취소는 되돌린 항목의 탭으로 바꾸고 그 항목을 선택한다.

7.14 외부 변경과 임포트

- 디스크의 파일이 바뀌면 에디터가 다시 읽는다.
- 저장하지 않은 변경이 있으면 편집을 유지하고 둘째 줄이 `Unsaved edits kept` 인 경고 `Changed on disk` 를 보여 준다.
- ARXML/DBC 임포트는 열린 모든 에디터를 먼저 저장하고, 끝나면 이 에디터를 다시 읽는다.

7.15 다른 에디터·기능과의 연동

- **System 에디터**: 그 IVC 모델이 여기서 정의한 프레임과 PDU 를 참조한다.
- **System 모델 데이터 매핑**: 시그널 이름을 바꾸면 함께 갱신된다.
- **ECUC**: Com, PduR 구성은 이 모델로부터 생성되지 않는다(12. ECUC).
- **Validation**: 결과는 14. Validation에 모인다.
- **DBC**: **DESIGN** 머리글의 **Import** 로 가져온다. **DESIGN** 머리글의 **Export** 나 System 에디터 CAN 클러스터의 **Export to DBC** 로 내보낸다(13. 임포트 · 익스포트).

7.16 편집 규칙 요약

- 이름은 AUTOSAR SHORT-NAME 규칙을 따른다. 첫 글자는 영문자, 나머지는 영문자, 숫자, 밑줄이다. 규칙을 어긴 이름은 반영되지 않는다.
- 비어 있는 필수 참조와 비어 있는 `TYPE`, `BYTE-ORDER` 는 빨갛게 표시된다.
- 정렬, 필터, 패널 폭, 열 폭은 화면 상태일 뿐이며 저장되지 않는다.

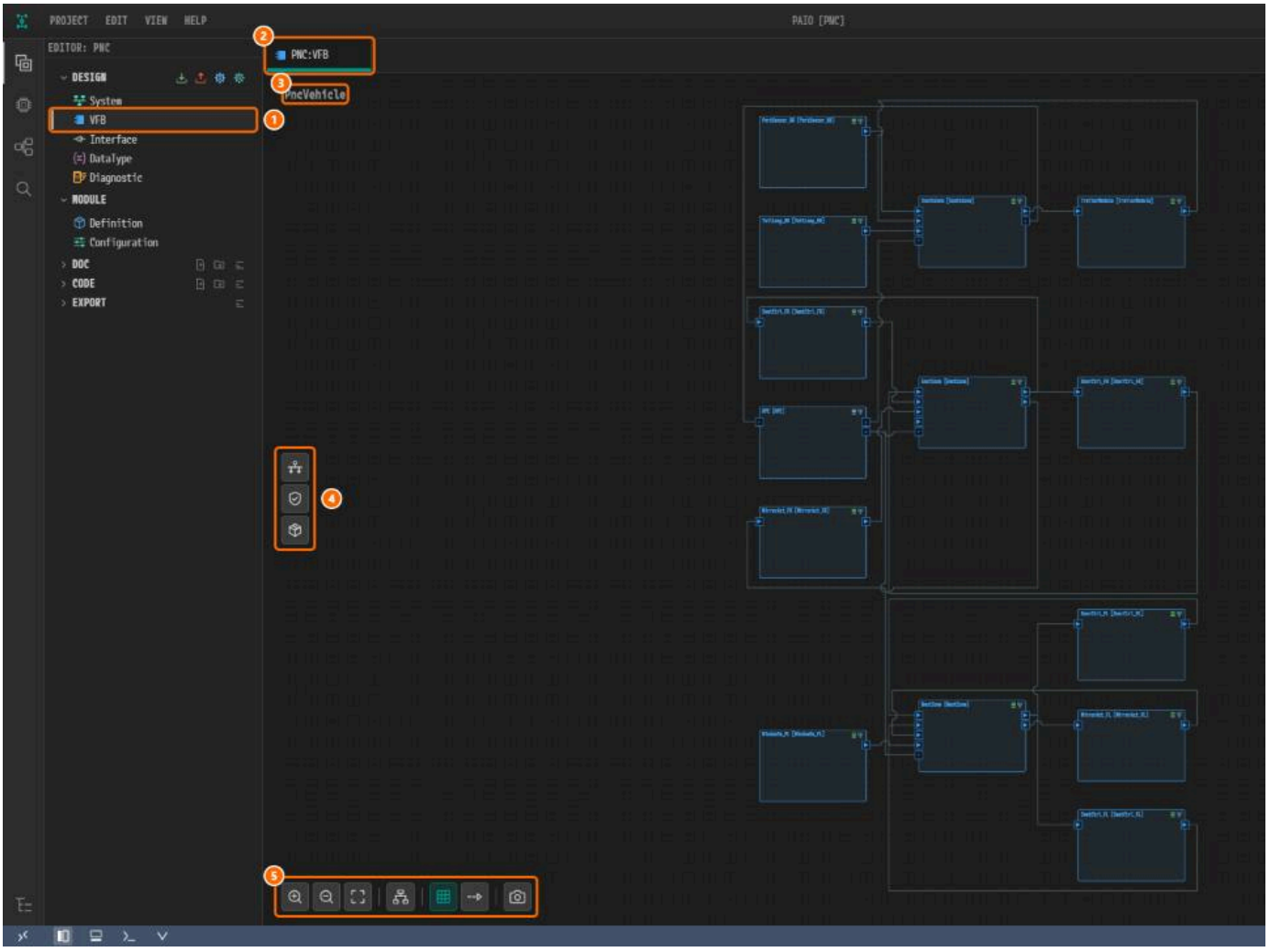
VFB 에디터

이 장에서 다루는 것

이 장에서는 Virtual Functional Bus 관점에서 소프트웨어 컴포넌트(SWC)와 컴포지션을 배치하고 포트를 연결하는 VFB 에디터의 사용법을 안내한다.

- 대상: AUTOSAR Classic 또는 Adaptive 플랫폼의 소프트웨어 아키텍처를 설계하는 사용자
- 전제: 창에 프로젝트가 열려 있어야 한다. 포트에 지정할 인터페이스는 09. Interface 에디터에서 미리 정의해 두면 편리하다. ECU/HPC 매핑에 쓰는 ECU, HPC 노드는 06. Sys

에디터를 열려면 왼쪽 **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 클릭한다. 탭 `<프로젝트>:VFB`(예: `PNC:VFB`)가 열린다. `design/VFB.io.model` 을 Explorer 에서 열지 않는다. 다.



*① **DESIGN** 아래의 **VFB**. ② 탭 `PNC:VFB`. ③ 시스템 라벨 `PncVehicle`. ④ 캔버스 왼쪽 가장자리의 액션 줄. ⑤ 캔버스 툴바.*

컨트롤	내용
액션 줄(캔버스 왼쪽 가장자리)	**VFC (PNC Overlay)**(8.3), **Intrusion Detection System**(8.19), **Software Cluster**(8.20)
캔버스 툴바(왼쪽 아래)	**Zoom In**, **Zoom Out**, **Fit to View**, **Auto Layout**, **Toggle Grid**, **Toggle Connector Animation**, **Export as PNG** (8.3)
오른쪽 가장자리 버튼	속성 패널 토글, 잠금, SYSTEM 패널을 여는 **System**(PAIO 로고) (8.7)

8.1 저장 대상 파일

한 번 저장하면 여러 파일이 함께 기록된다. VFB 모델, SWC마다의 SWC 정의, 컴포지션마다의 캔버스 배치(`design/SWC/Layout/<컴포지션>.vfb.layout`), Machine 설정이

- 커넥터는 VFB 모델이 아니라 컴포지션의 SWC 정의에 저장된다.
- `.vfb.layout` 은 화면 배치만 담는다. ARXML로 내보내지지 않는다.

- 이름을 바꾼 SWC의 예전 파일은 저장할 때에야 삭제된다. 이름을 바꾼 뒤 파일 목록이 이상해 보이면 먼저 `Ctrl+S` 로 저장한다.

8.2 SWC 종류

Add SWC 하위 메뉴가 제공하는 종류는 **Composition**, **Adaptive Application**, **Application**, **Complex Device Driver**, **Ecu Abstraction**, **Sensor Actuator**, **Service** 7가지

InterfaceMapping 노드는 SWC 종류가 아니다. 루트 레벨의 빈 캔버스에서 **Add Interface Mapping** 으로 만들며, AP 서비스 인터페이스와 CP SR/CS 인터페이스를 잇는 역할민

8.3 캔버스 조작

자주 쓰는 조작은 다음과 같다.

마우스

- 빈 곳에서 좌클릭, 가운데 버튼, 또는 `Space` + 좌클릭으로 드래그하면 화면이 이동한다. `Ctrl` + 휠로 확대/축소한다.
- 노드를 드래그하면 이동한다. 포트에서 `Shift` + 드래그하면 연결을 시작한다(8.6).
- 컴포지션을 더블클릭하면 그 안으로 드릴다운한다. Adaptive Application을 더블클릭하면 **AA Internal Behavior**(8.9)가, 그 밖의 Atomic SWC를 더블클릭하면 **SWC Inte**
- 우클릭하면 컨텍스트 메뉴(8.5)가 열린다.

키보드

- `Ctrl+S` 는 저장, `Ctrl+Z` 는 실행 취소, `Ctrl+Y` 는 다시 실행이다.
- `Delete` / `Backspace` 는 선택한 상자를 확인을 거쳐 삭제한다. 연결선은 지우지 않는다. 연결선은 그 연결선의 우클릭 메뉴에서 **Disconnect** 로 끊는다(8.5.7).
- 방향키와 `Ctrl+C` / `Ctrl+V` / `Ctrl+X` 는 VFB 캔버스에서 아무 일도 하지 않는다.

툴바 버튼 (캔버스 왼쪽 아래)

- **Fit to View** 는 전체 내용이 보이도록 맞춘다.
- **Auto Layout** 은 자동 배치를 실행한다. 손으로 조정한 연결선 경로를 모두 바꾸므로, 그런 경로가 있으면 먼저 확인을 받는다.
- **Export as PNG** 는 현재 다이어그램을 프로젝트의 `doc/capture/` 폴더에 저장한다.

VFC (PNC Overlay)

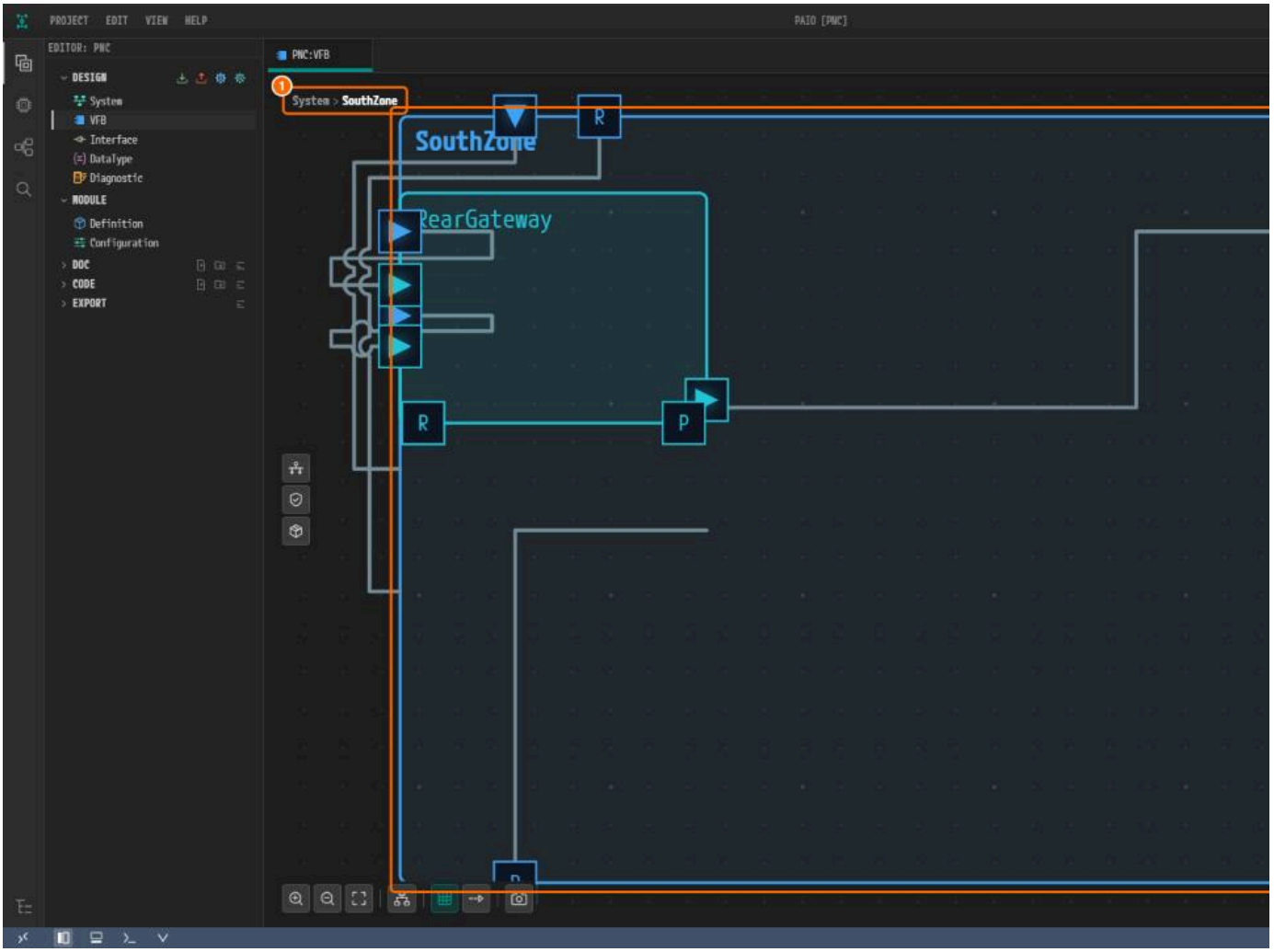
액션 줄의 첫 아이콘은 오버레이를 켜고 끈다. **PNC OVERLAY** 패널에서 VFC(Virtual Function Cluster, 포트 그룹)를 고르면 그 VFC에 속한 노드와 포트는 원래 밝기로, 나머지는 흐
네트워크가 나열된다.

오버레이는 보여 주기만 하고 모델을 바꾸지 않는다. VFC는 06. System 에디터의 **Partial Network (PNC) Configuration** 다이얼로그에서 선언한다.

8.4 루트 레벨과 드릴다운

- **루트 레벨**: 아무 컴포지션에도 들어가지 않은 상태이다. 최상위 컴포지션과 InterfaceMapping 노드가 놓인다.
- **드릴다운 상태**: 컴포지션 하나 안으로 들어간 상태이다. 그 컴포지션이 프레임으로 그려지고 자식 노드가 프레임 안에 놓인다.

1. 컴포지션을 더블클릭한다. 캔버스 왼쪽 위에 `System > SouthZone` 같은 브레드크럼이 나타난다.
2. 브레드크럼의 앞쪽 항목을 클릭하면 그 레벨로 돌아간다.



① 브레드크럼 `System > SouthZone`. ② 자식 SWC `RearGateway`, 경계 포트, 커넥터가 있는 컴포지션 프레임.

드릴다운 상태에서 노드를 만들 때는 프레임(컴포지션)을 우클릭한다. 프레임 바깥의 빈 캔버스에서는 메뉴가 나타나지 않는다.

8.5 컨텍스트 메뉴

우클릭한 대상에 따라 메뉴가 달라진다. 상자를 여러 개 선택한 상태에서 그중 하나를 우클릭하면 **Delete N items** 만 나타난다.

8.5.1 빈 캔버스 (루트 레벨에서만)

메뉴에는 **Add Composition**, **Attach Composition**(이미 정의된 컴포지션을 프로토타입으로 붙인다), **Add Interface Mapping** 이 있다.

8.5.2 컴포지션 프레임 (현재 드릴다운 중인 컴포지션)

메뉴에는 **Add SWC**(8.2의 7종 중 하나), **Attach SWC**, **Add Port**(**PPort**, **RPort**, **PRPort**), **Add Composition**, **Attach Composition** 이 있다.

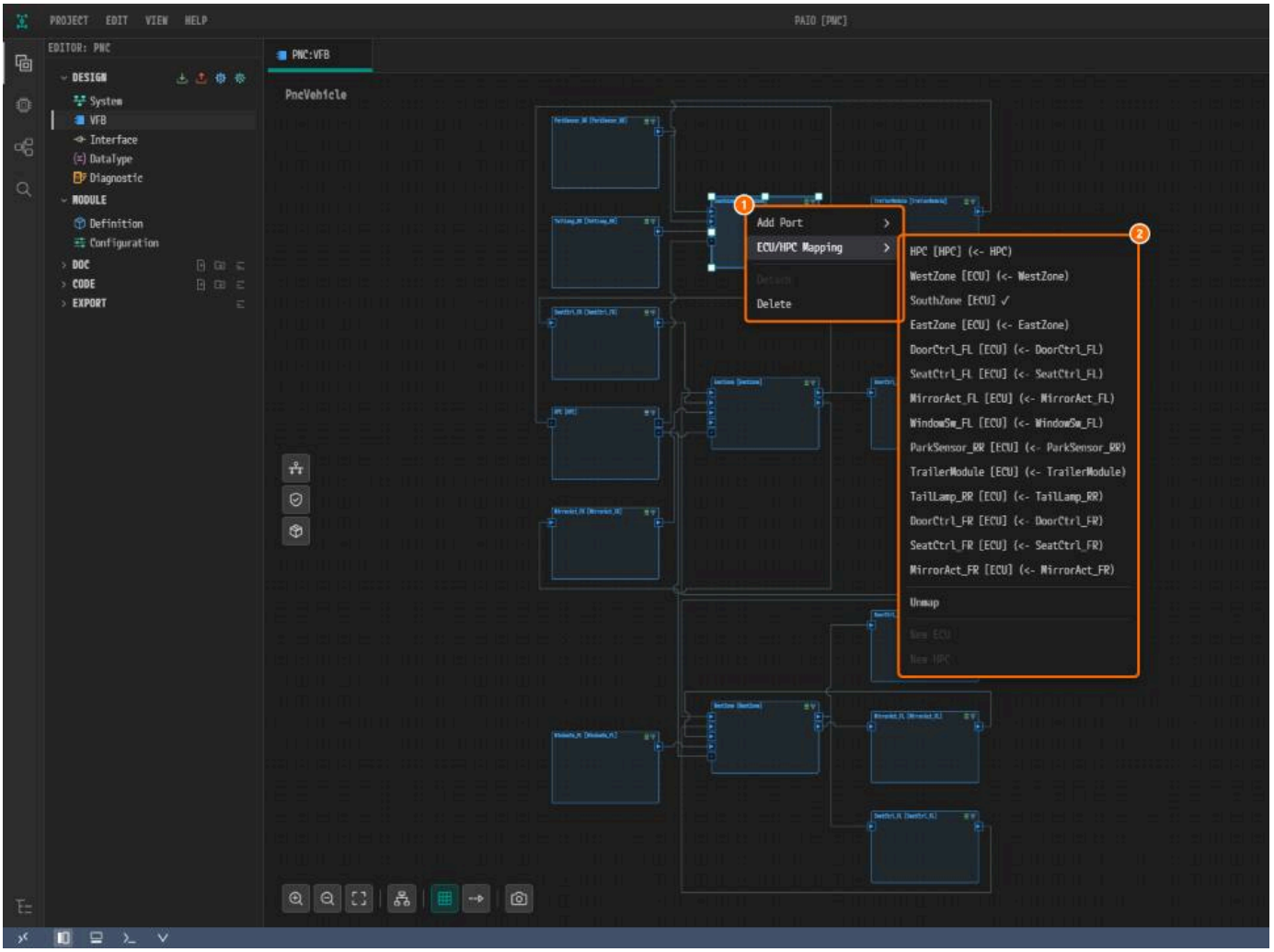
8.5.3 컴포지션 노드 (드릴다운 중이 아닌 컴포지션)

메뉴에는 **Add Port**, **ECU/HPC Mapping**(최상위 컴포지션에만), **Detach**, **Delete** 가 있다. 자식 노드는 그 컴포지션 안으로 드릴다운해서 관리한다.

- **Detach** 는 상위 컴포지션에서 그 프로토타입만 떼어낸다. SWC 타입은 남는다.
- **Delete** 는 **Delete SWC** 확인 창을 거친 뒤 그 SWC를 모든 컴포지션에서 제거하고 타입 파일까지 삭제한다. 파일 삭제는 저장할 때 반영된다.

컴포지션을 ECU 또는 HPC에 매핑하려면 다음과 같이 한다.

1. 최상위 컴포지션을 우클릭하고 **ECU/HPC Mapping** 을 연다.
2. System 에디터의 노드를 고르거나, **New ECU** / **New HPC** 로 같은 이름의 노드를 만들어 매핑한다. **Unmap** 은 매핑을 해제한다.



① `SouthZone` 의 노드 메뉴. ② **ECU/HPC Mapping** 하위 메뉴. `SouthZone [ECU] ✓` 가 이 컴포지션에 매핑된 노드이다.

매핑된 컴포지션은 머리글에 `SouthZone [SouthZone]` 처럼 노드 이름을 보여 주고, ECU 아이콘 또는 HPC 아이콘이 붙는다. 아이콘을 클릭하면 ECU Configuration(8.12) 또는

8.5.4 Atomic SWC 노드

메뉴에는 **Add Port**, **Detach SWC**, **Delete** 가 있다. **Delete** 는 8.5.3과 같은 **Delete SWC** 확인 창을 거치며 타입 파일까지 삭제한다.

8.5.5 Port 노드

메뉴에는 **Connect to**, **Connect to Mapping**, **Derive SR/CS Interface**, **Delete** 가 있다. **Derive SR/CS Interface** 는 루트 레벨에서, 인터페이스 카테고리가 `service` 인 포트

8.5.6 InterfaceMapping 노드

메뉴 항목은 **Delete** 하나뿐이다.

8.5.7 연결선

메뉴에는 **To Step Edge**(자동 직교 라우터가 다시 그린다), **To Straight Edge**, **Disconnect** 가 있다. 웨이포인트 핸들을 우클릭해도 같은 메뉴가 열린다.

8.6 포트 연결

1. 포트에서 `Shift` 를 누른 채 드래그한다. 놓을 수 있는 대상은 초록색, 놓을 수 없는 대상은 빨간색으로 강조된다.
2. 대상 위에서 버튼을 놓는다.
   - 호환되는 포트 위: 두 포트가 연결된다.
   - SWC 몸통 위: 짝이 되는 포트가 만들어지고 연결된다.
   - InterfaceMapping 노드 위: 그 노드로 연결된다.
3. 연결선을 클릭한다. 속성 패널에 **Basic Info** 섹션이 나타나면 연결이 만들어진 것이다.

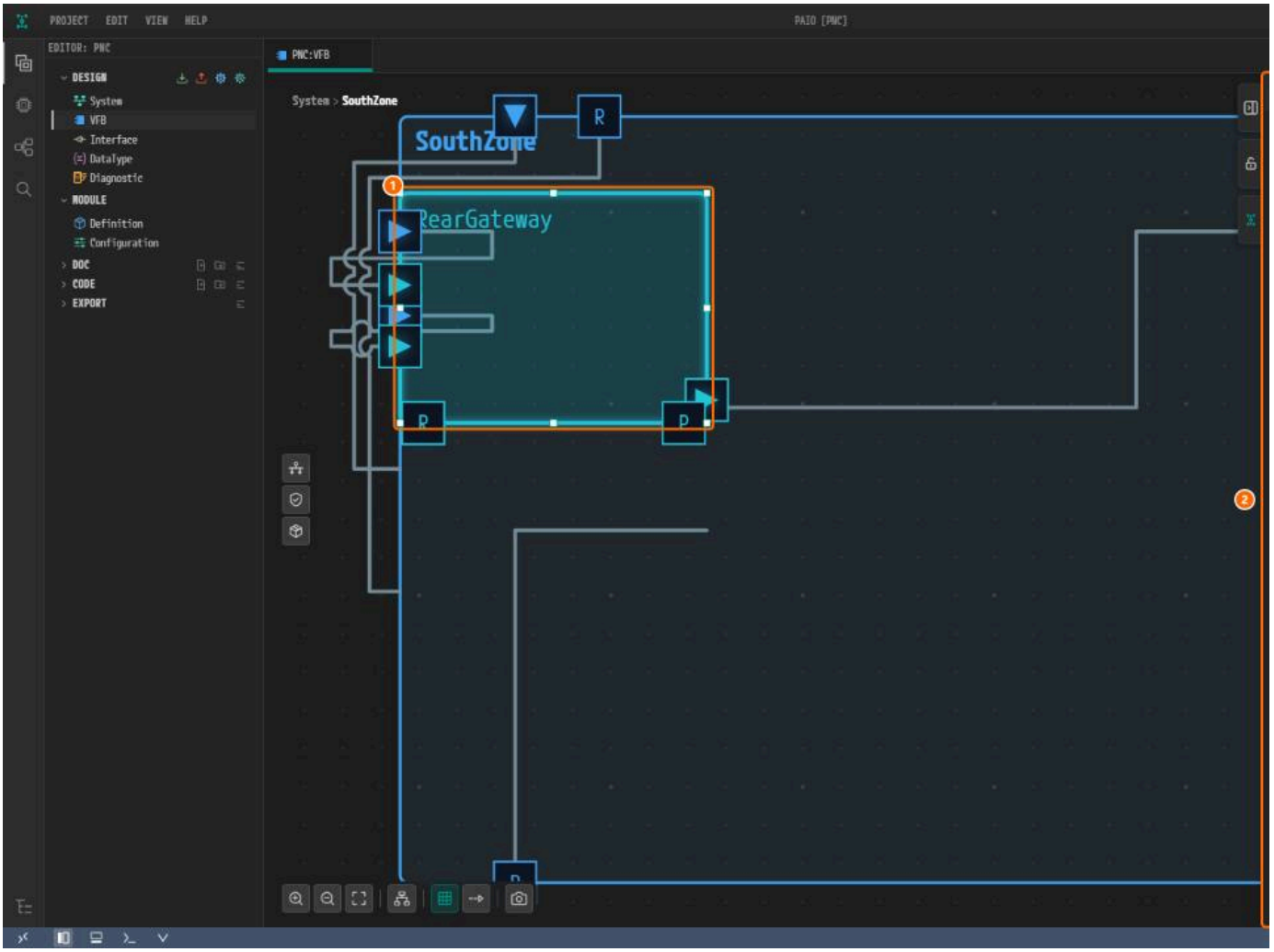
`Shift` 없이 드래그하면 이동이 된다. 포트를 우클릭해 **Connect to** / **Connect to Mapping** 에서 상대를 고를 수도 있다(8.5.5). 연결될 것으로 예상되는 쌍인데 **Connect to** 에로 연결한다.

8.7 속성 패널 (우측)

선택한 대상의 필드를 보여 준다. 캔버스 오른쪽 가장자리의 버튼 세 개로 패널을 켜고 끄고, 열린 채로 고정하고, SYSTEM 패널을 연다(**System**, PAIO 로고).

SWC / 컴포지션 노드

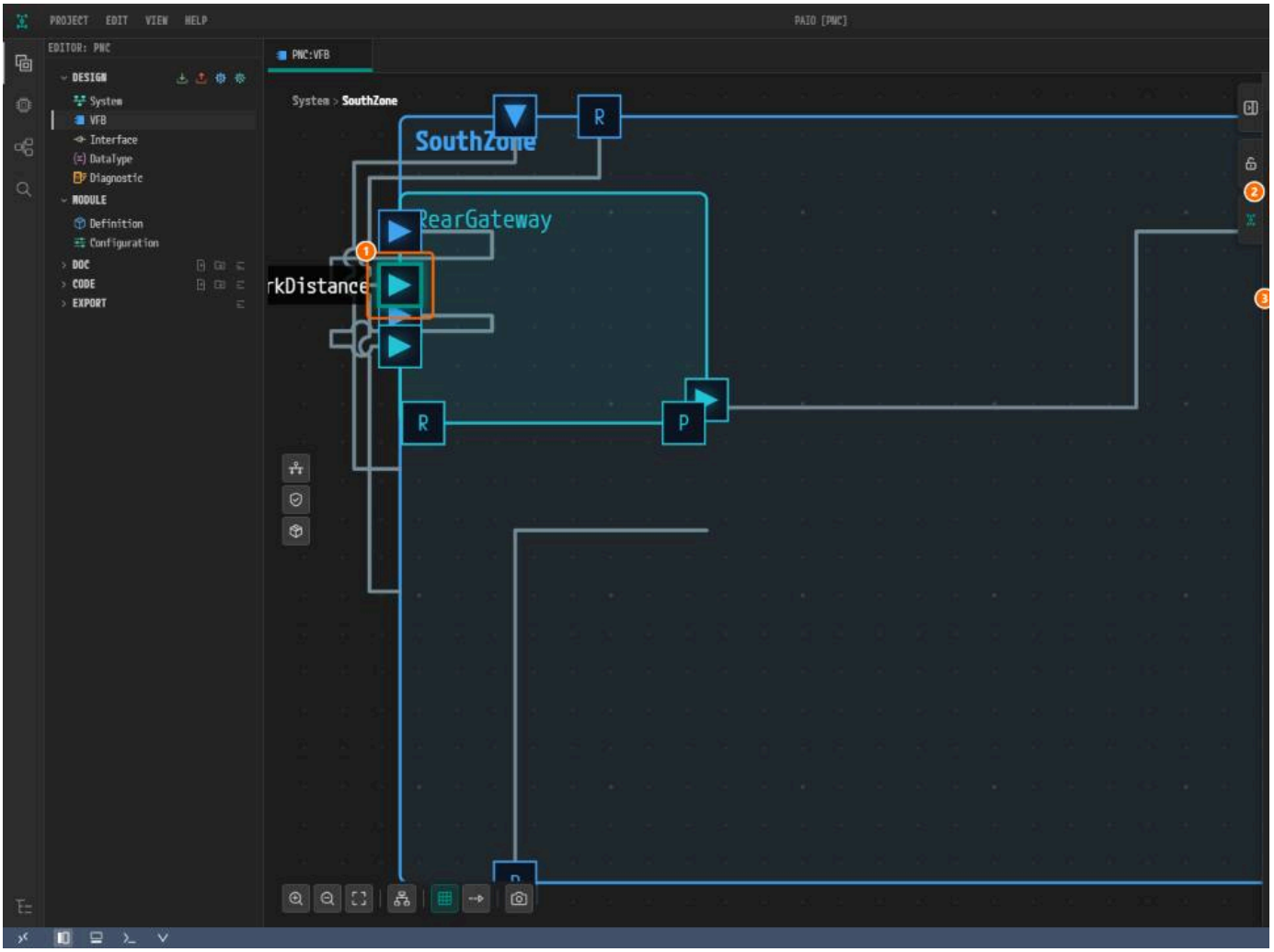
패널에 **Name**, **AS Prototype**, **Description**, **Ports** 가 나온다. 컴포지션에는 **Port Groups** 도 있다. Atomic SWC의 포트 그룹은 SWC Internal Behavior 다이얼로그(8.8)에서 표



*① 선택한 SWC `RearGateway`. ② 제목이 **APPLICATION** 인 속성 패널. **PORTS** 에 포트마다 PortInterface 가 함께 나온다.*

Port 노드

1. 포트를 클릭한다.
2. **Port Interface** 에서 인터페이스를 고르거나 `[+]` 로 새로 만든다. 후보는 부모 SWC 종류에 따라 달라진다.
3. 인터페이스 카테고리에 맞는 ComSpec 섹션을 편집한다. 예를 들어 Sender-Receiver이면 **Data Element** 이다.



① 선택한 포트. ② ***PORT INTERFACE*** `SenderReceiver: SRI_ParkDistance`. ③ ***DATA ELEMENT*** 섹션.

연결선

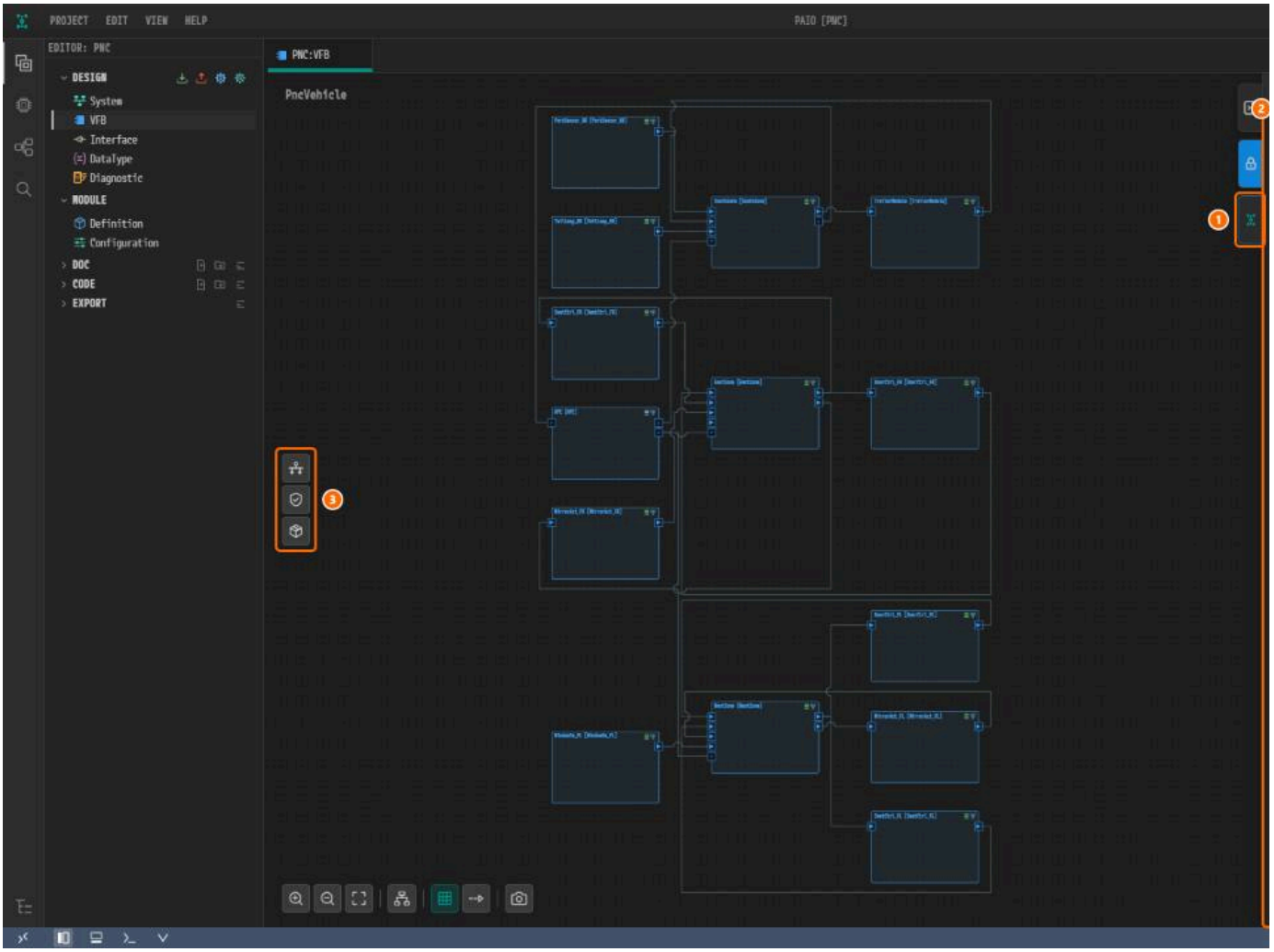
Basic Info 섹션에 **Name** 필드 하나가 있다.

InterfaceMapping 노드

서비스 인터페이스의 요소를 SR/CS 인터페이스에 **Event Mappings**, **Method Mappings**, **Field Mappings** 로 매핑한다. **Connected Ports** 에는 이 노드에 연결된 포트가 나열션이 나타나지 않는다.

SYSTEM 패널

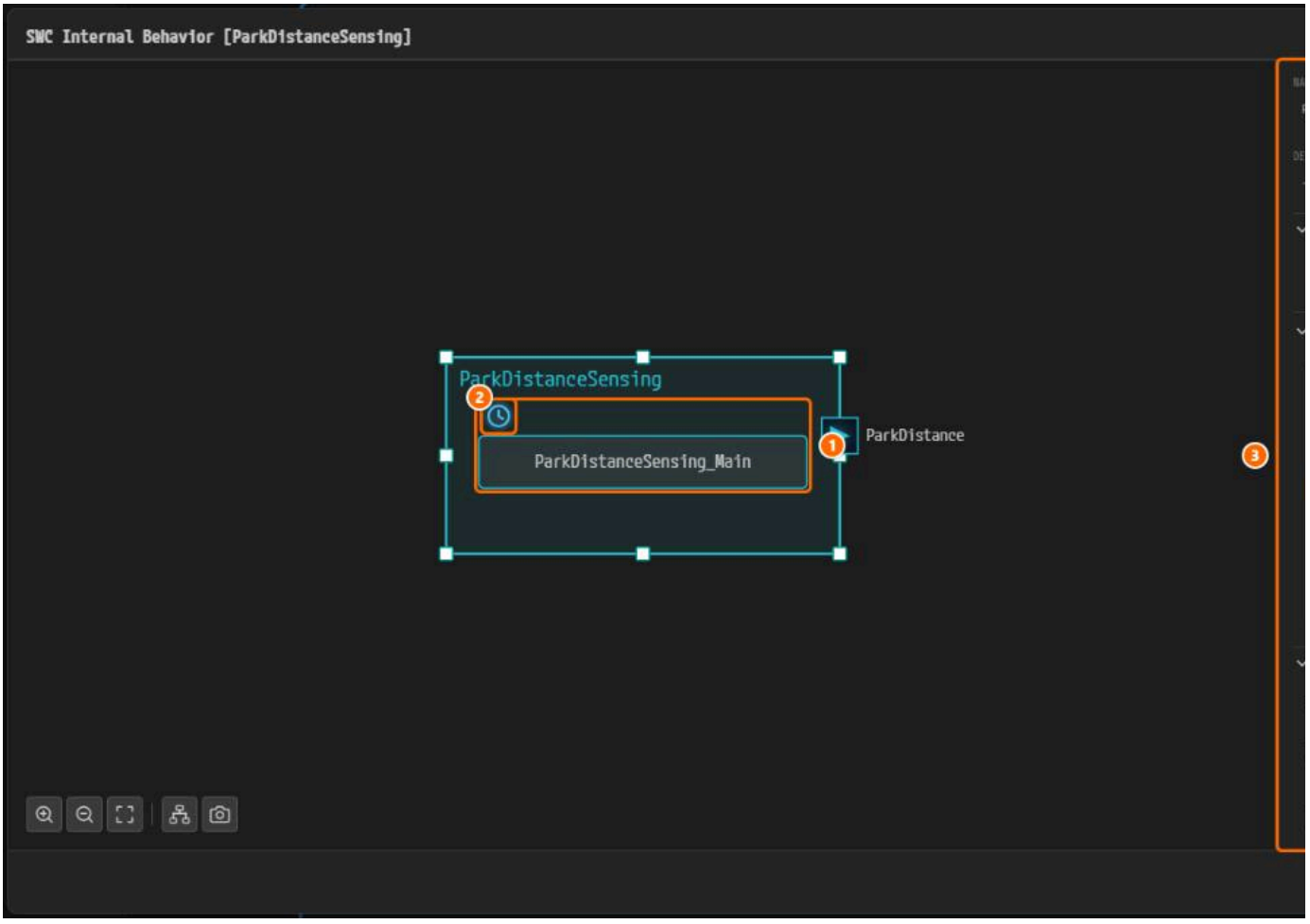
오른쪽 가장자리의 **System** 을 클릭하면 **NAME**, **VERSION**, **DESCRIPTION**, **ROOT COMPOSITION**, **COMPOSITIONS** 트리가 나타난다. 값은 바꾸는 즉시 프로젝트에 저장되므로 `Ct` **DESCRIPTION** 은 System 에디터와 공유한다(11. SYSTEM 패널).



① ***System***(PAIO 로고). ② SYSTEM 패널. ③ 액션 줄.

8.8 SWC Internal Behavior 다이얼로그

Application, Complex Device Driver, Service, Ecu Abstraction, Sensor Actuator 노드를 더블클릭하면 열린다. 왼쪽은 캔버스, 오른쪽은 포트, **Port Groups**, **Implementa**



*① Runnable `ParkDistanceSensing_Main`. ② 그 RTE 이벤트 아이콘(TimingEvent 시계). ③ **PORTS**, **PORT GROUPS**, **IMPLEMENTATION** 이 있는 오른쪽 패널.*

저장하지 않은 편집이 있으면 제목 앞에 • 가 붙고, **Close** 를 누를 때 저장 여부를 묻는다.

캔버스 우클릭

1. SWC 몸통을 우클릭하고 **Add Port**, **Add Runnable**, **Add IRV** 중 하나를 고른다.
2. Runnable을 우클릭하고 **Add RTEEvent** 아래에서 RTE 이벤트를 고른다. DataReceiveErrorEvent, DataSendCompletedEvent, AsynchronousServerCallReturnsE ExternalTriggerOccurredEvent 를 포함해 13종이 있다.

각 요소의 메뉴에는 삭제 항목도 있다.

Runnable 속성

Runnable을 클릭하면 **NAME**, **SYMBOL**(생성 코드의 C 함수 이름), **CAN BE INVOKED**, **MIN START INTERVAL** 을 편집한다. **Access Points** 와 **RTE Events** 머리글의 + 로 액세스

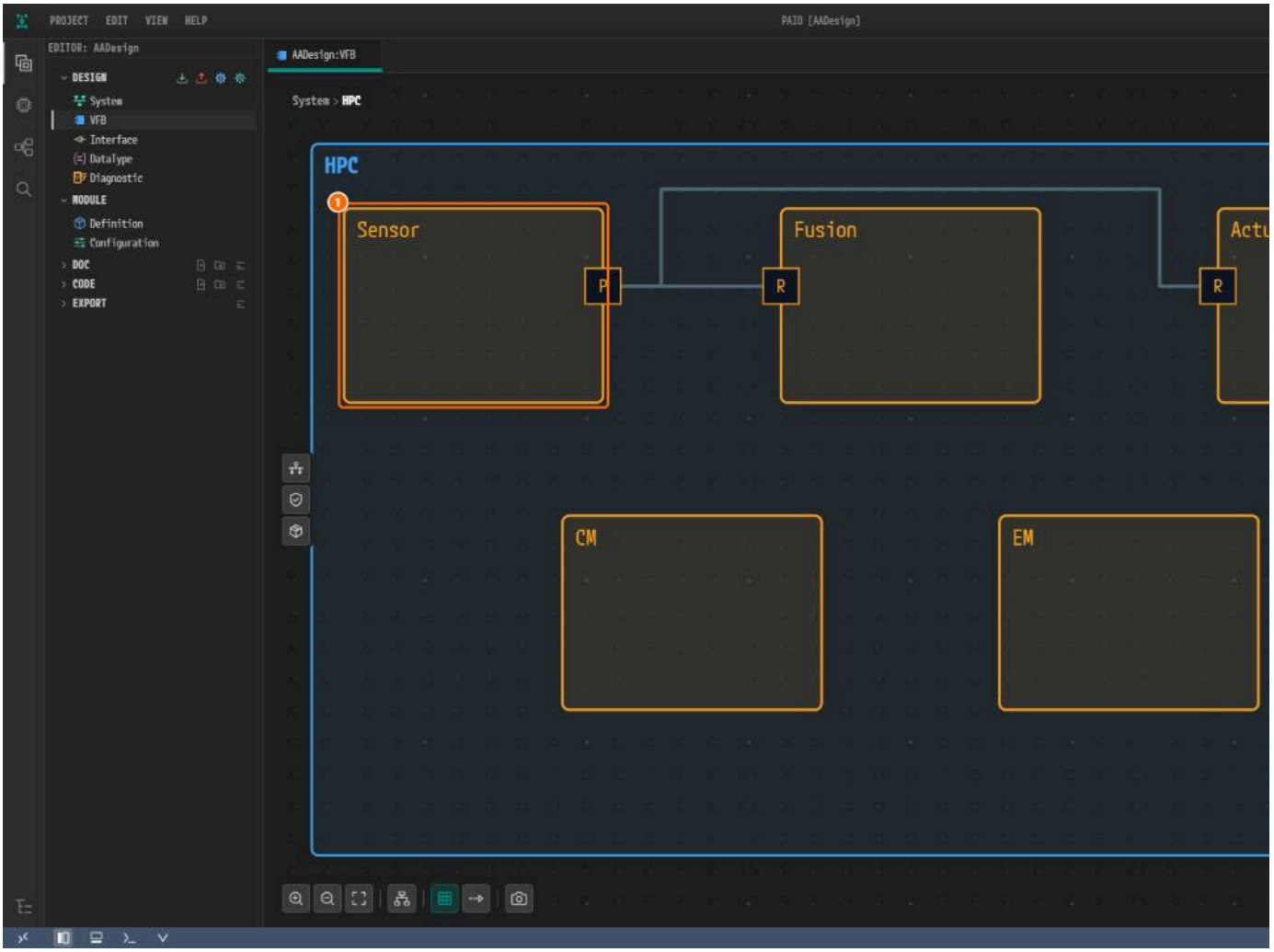
포트에서 Runnable로 `Shift` + 드래그하면 그 포트의 방향과 인터페이스 카테고리에 맞는 항목만 메뉴에 나온다. 예를 들어 Sender-Receiver P-Port이면 DataSendPoint와 D 쓰는 포트는 P-Port이면 ExternalTriggeringPoint, R-Port이면 ExternalTriggerOccurredEvent가 나온다. AsyncServerCallPoint에는 AsynchronousServerCallResultP

Port API Option

이 다이얼로그에서 포트를 클릭하면 **Enable Take Address**, **Indirect API**, **Error Handling**, **Port Arg Values** 가 있는 **Port API Option** 섹션이 추가된다. 이 섹션은 메인 캔버스에

8.9 AA Internal Behavior 다이얼로그 (Adaptive Application)

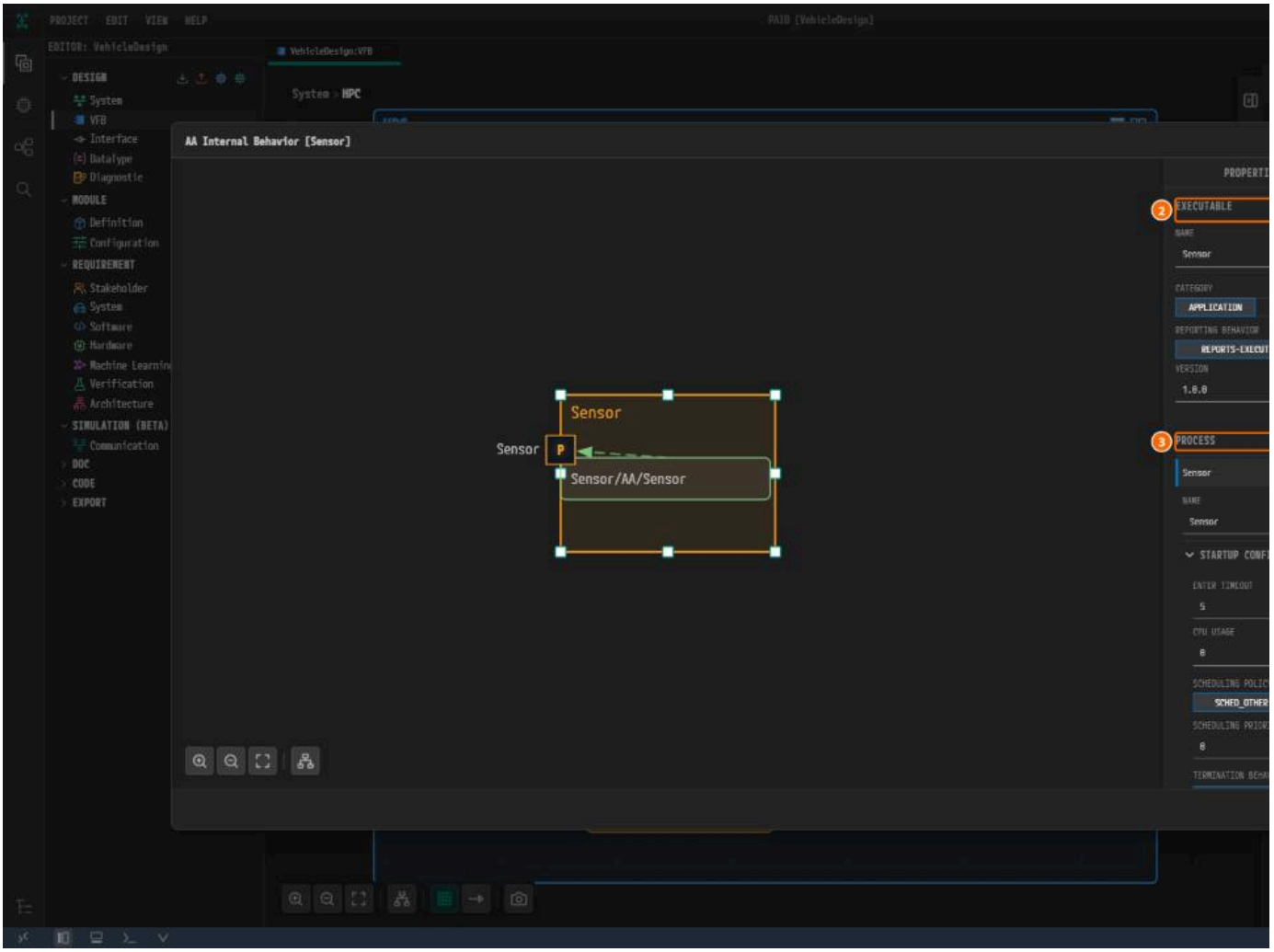
Adaptive Application 노드를 더블클릭하면 열린다. Adaptive Application 노드는 주황색으로 그려진다.



① 컴포지션 `HPC` 안의 노드 `Sensor`.

오른쪽 패널에는 탭이 두 개 있다.

- **PROPERTIES**: 노드의 **Name**, **AS Prototype**, **Description**, **Ports**.
- **EXECUTION**: **EXECUTABLE** 과, 프로세스마다 **PROCESS** 카드 하나. 카드에는 **STARTUP CONFIG**, **FUNCTION GROUP STATE**, **EXECUTION DEPENDENCY**, **PROGRAM ARG** 다. 프로세스 이름을 클릭하면 카드가 펼쳐지거나 접힌다. 목록이 비어 있으면 **FUNCTION GROUP STATE** 의 **+** 가 빨갛다.



① ***EXECUTION*** 탭. ② ***EXECUTABLE***. ③ ***PROCESS*** 카드 `Sensor`.

프로그램 인자와 환경 변수는 PARA가 프로세스의 실행 매니페스트에 기록한다.

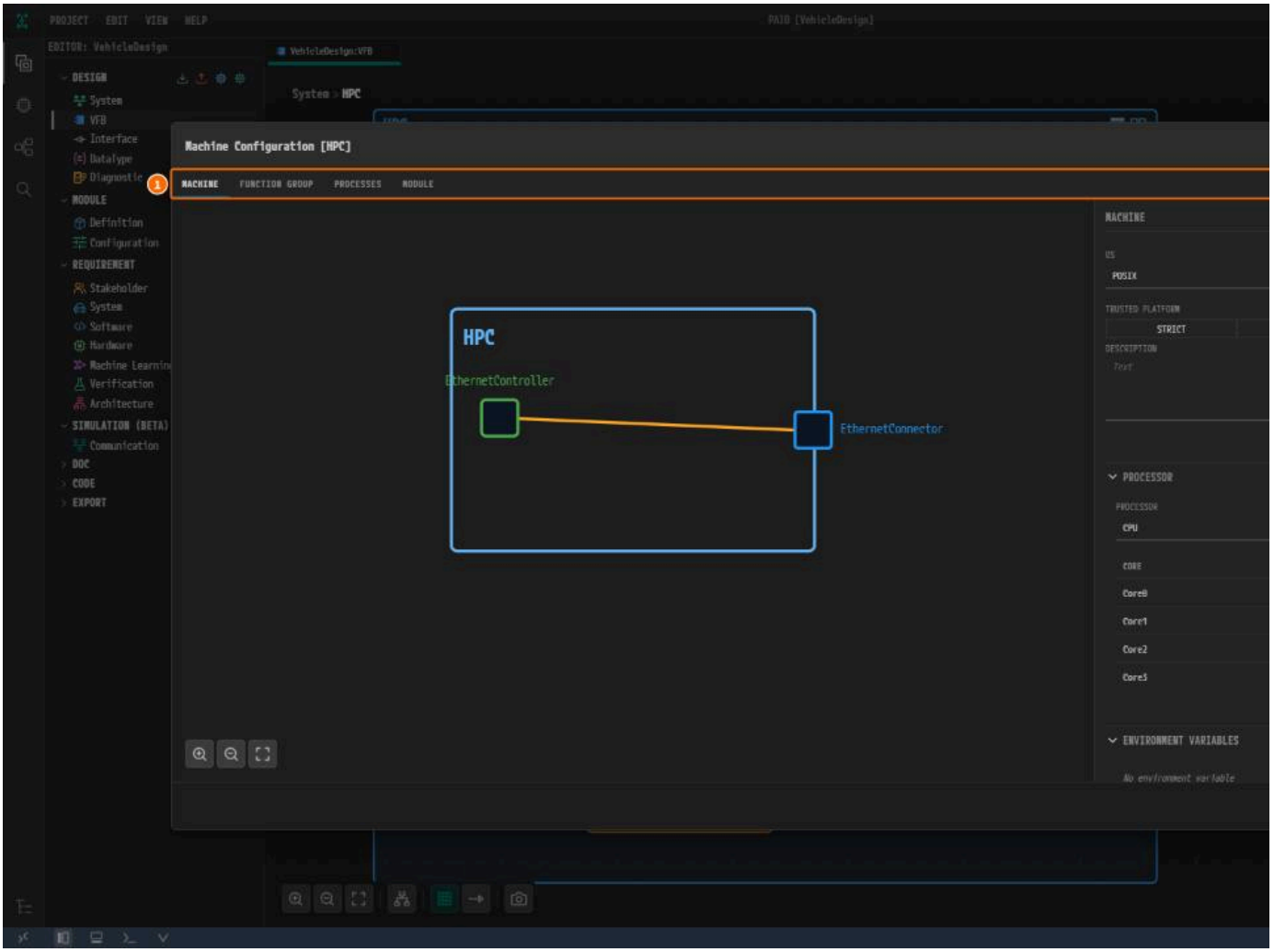
캔버스 우클릭 (AA 몸통)

메뉴에는 **Add Port** 와 **Add Instance** 가 있다. **Add Instance** 는 기능 클러스터(**ara::com**, **ara::log**, **ara::sm** 등)별로 묶인 AP 매핑 상자를 추가한다. 포트에서 `Shift` + 드래그

8.10 Machine Configuration 다이얼로그 (Adaptive)

HPC에 매핑된 컴포지션 노드의 HPC 아이콘을 클릭하면 열린다. 탭이 네 개 있다.

- **MACHINE**: OS, 신뢰 플랫폼, 프로세서와 코어, Machine의 환경 변수, HPC의 읽기 전용 네트워크 요약.
- **FUNCTION GROUP**: 함수 그룹과 그 상태 머신.
- **PROCESSES**: 이 컴포지션의 Adaptive Application 프로세스(8.10.2).
- **MODULE**: AP 기능 클러스터 모듈(8.10.1).



*① 탭 **MACHINE, FUNCTION GROUP, PROCESSES, MODULE**. ② **PARA Platform Import**.*

다이얼로그를 열면 없는 요소가 채워진다. **MachineFG** 함수 그룹, 11종 모듈 인스턴스, `CPU` / `Core0` 한 개이다. 처음에는 지금 필요한 클러스터만 편집하고 나머지는 그대로 두어도

8.10.1 MODULE 탭의 11개 모듈

MODULES 목록에서 모듈을 클릭하면 가운데에 편집 화면, 오른쪽에 개요가 나타난다. **Cryptography** 와 **Persistency** 에는 오른쪽 패널이 없다. 모듈 행을 우클릭하면 **Disable** 또
익스포트되지도, Validation 검사를 받지도 않는다.

8.10.2 PROCESSES 탭

Processes 목록에서 프로세스를 고른다. 편집할 수 있는 값은 **CORE AFFINITY** 하나이고, 나머지는 읽기 전용이다.

프로세스를 State Management 또는 Platform Health Management 프로세스로 지정하려면 다음과 같이 한다.

1. 프로세스를 선택한다.
2. 머리글 오른쪽의 **SM** 아이콘이나 **PHM** 아이콘을 누른다(툴팁 `Mark as State Management`, `Mark as Platform Health Management`). 회색 아이콘은 꺼진 상태, 싹
   소속은 프로세스 하나만 가질 수 있으며, 다른 프로세스를 지정하면 옮길지 묻는다.

SM으로 지정하면 함수 그룹마다 SM SWC에 StateClient R-Port와 FunctionGroupPortMapping도 함께 만든다. 켜진 SM 아이콘을 누르면 먼저 확인을 묻고, 지정을 해제하면
제되며, PHM은 포트나 매핑을 만들지 않는다. 지정 하나가 실행 취소 한 항목이다. 지정만 되돌리려면 다이얼로그를 먼저 닫고 `Ctrl+Z` 를 누른다.

8.11 PARA Base Configuration

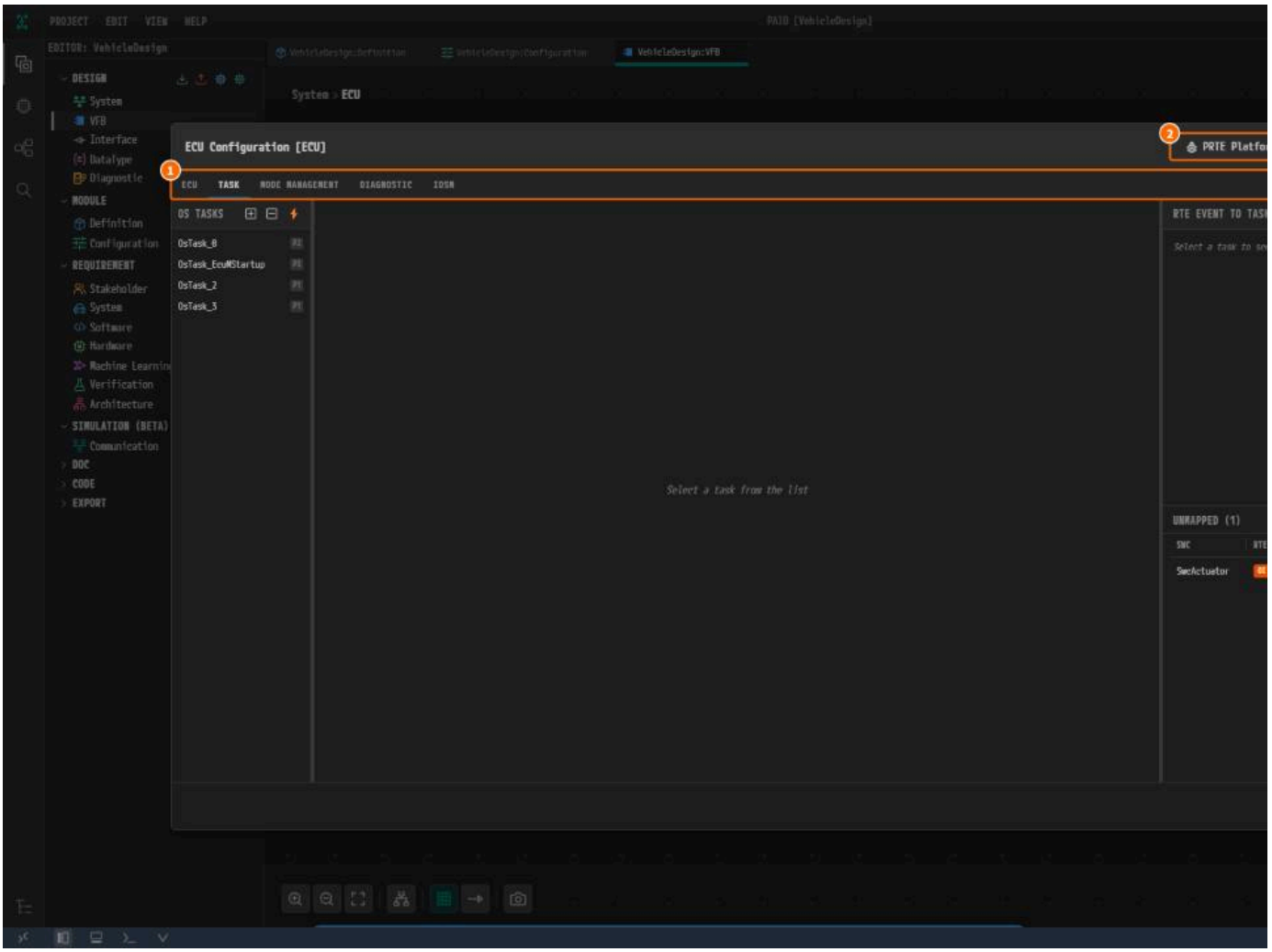
PopcornSAR AUTOSAR Adaptive Platform 기본 구성을 프로젝트에 주입한다.

1. Machine Configuration 다이얼로그 머리글의 **PARA Platform Import** 를 누르거나, 명령 팔레트에서 **PARA: Import Platform** 을 실행한다.
2. 확인 창에서 **Import** 를 누른다.
3. 충돌이 있으면 충돌 해결 다이얼로그가 열린다. 성공하면 Machine Configuration 다이얼로그가 다시 열려 MachineFG와 모듈이 합쳐진 결과를 보여 준다.

전체가 실행 취소 한 단계이다. 머신 이름, 프로세서, OS는 건드리지 않는다.

8.12 ECU Configuration 다이얼로그 (Classic)

ECU에 매핑된 컴포지션 노드의 ECU 아이콘을 클릭하면 열린다. 탭은 **ECU**, **TASK**, **MODE MANAGEMENT**, **DIAGNOSTIC**, **IDSM** 다섯 개이고, 머리글에 **PRTE Platform Import**(8.1 편집은 `Ctrl+S` 로 저장한다.



*① 탭 줄. ② **PRTE Platform Import**. ③ **PRTE AUTO CONFIGURATION**.*

TASK 탭에서 태스크를 구성하는 순서는 다음과 같다.

1. **OS TASKS** 에서 ⚡(Auto-Generate)를 눌러 RTE 이벤트로부터 태스크와 알람을 만들거나, `+` 로 태스크를 직접 추가한다.
2. 태스크를 선택하고 가운데 **TASK CONFIGURATION** 에서 편집한다.
3. 오른쪽 **RTE EVENT TO TASK MAPPING** 에서 RTE 이벤트를 배정한다. 매핑된 행을 위아래로 드래그하면 태스크 안의 실행 순서가 정해진다.

Auto-Generate 는 BSW 메인 함수를 주기마다 태스크 하나(`OsTask_Bsw_10ms` 등)에 두고, 모든 태스크의 우선순위를 정한다. 백그라운드가 가장 낮고, 그 위에 BSW, 그 위에 SV 출로 시작하는 이벤트는 태스크가 필요 없으므로 **DIRECT CALL** 에 따로 나오고, 매핑·미매핑 어느 쪽으로도 세지 않는다. 태스크의 **Autostart** 는 ON/OFF 토글이다.

TASK 탭은 `Os`, `Rte` 설정과 양방향으로 동기화되므로, 여기서 만든 태스크는 12. ECUC에서도 보인다.

8.12.1 ECU 탭

ECU의 **PARTITIONS** 를 편집하고(`+` 로 추가, `×` 로 제거), **NETWORKS** 와 **ECU-WIDE** 정보를 읽기 전용으로 보여 준다. 저장하면 파티션이 System 에디터 모델과 그에 맞는 EcuC 1 간다. 네트워크는 06. System 에디터에서 편집한다.

8.12.2 MODE MANAGEMENT 탭

ECU의 BswM 설정을 세 패널로 편집한다. **BSWM** 요소 트리, 선택한 요소의 파라미터, **Coherence** 목록이 있는 **SWC SIDE (BswM.swc)** 이다. **Propose SD rules** 는 ECU의 SOM 칙을 추가한다. 이 탭은 먼저 PRTE Platform Import(8.15)가 필요하다.

8.12.3 DIAGNOSTIC 탭

ECU 진단 사양의 개수를 보여 주고, 스위치 **Derive Dcm/Dem ECUC from DEXT on Apply** 가 있다. 스위치가 켜져 있으면 다이얼로그를 저장할 때 사양에서 Dcm, Dem 설정이 기다.

8.12.4 IDSM 탭

Intrusion Detection System 다이얼로그(8.19)와 같은 편집기로 ECU의 IdsM 인스턴스를 편집한다. **ADD IDSM INSTANCE** 로 인스턴스를 만들고 **REMOVE** 로 지운다.

8.13 저장 · 실행 취소 · 외부 변경

- `Ctrl+S` 는 8.1의 파일을 한 번에 기록한다.
- 실행 취소와 다시 실행은 한 프로젝트의 모든 에디터가 공유하는 하나의 타임라인을 쓴다. 노드 위치, 크기, 연결선 경로는 되돌릴 수 있고, 화면 이동과 확대/축소는 되돌리지 않는
- 저장하지 않은 편집이 있는 동안 디스크의 VFB 파일이 밖에서 바뀌면, PAIO는 편집 내용을 지키고 경고를 표시한다. 저장하면 외부 변경을 덮어쓴다.

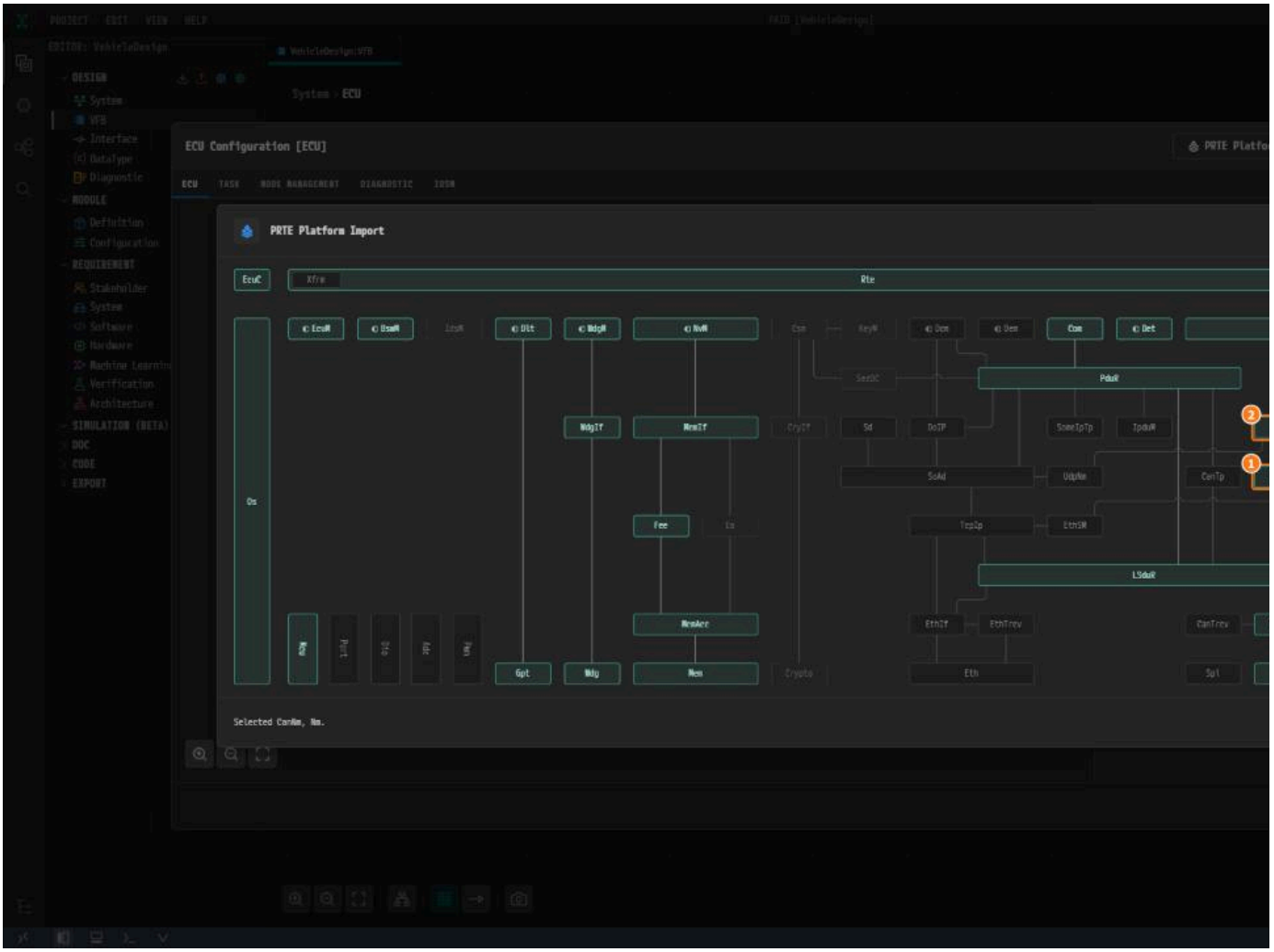
8.14 다른 에디터와의 연동

- 컴포지션과 ECU/HPC 노드의 매핑은 06. System 에디터와 공유된다.
- 포트의 인터페이스는 09. Interface 에디터의 정의를 참조한다. 인터페이스 이름을 바꾸면 VFB도 갱신된다.
- InterfaceMapping은 Interface 에디터의 **Mapping** 탭과 같은 데이터이다.
- ARXML 임포트/익스포트는 13. 임포트 · 익스포트, 단축키는 18. 단축키 총람에 있다.

8.15 PRTE Platform Import

PopcornSAR AUTOSAR Classic Platform 기본 세트를 ECU 하나에 대해 프로젝트로 가져온다. PRTE 모듈 정의(VSMD), 번들 서비스 인터페이스와 데이터 타입, 선택한 BSW 모듈

1. ECU Configuration 다이얼로그(8.12) 머리글의 **PRTE Platform Import** 를 누른다.
2. BSW 모듈 지도가 열린다. 설계가 필요로 하는 모듈은 선택되어 있고, Os, Rte, EcuM, BswM, Mcu는 항상 선택되어 있다. 모듈을 누르면 선택하거나 해제한다. 함께 필요한 모듈
   이름이 나온다. 점선 모듈은 이후 PRTE 릴리스에 들어온다. **Reset to suggestion** 은 제안된 선택으로 되돌린다.
3. **Import** 를 누른다. 충돌이 있으면 충돌 해결 다이얼로그가 열린다. 성공하면 ECU Configuration 다이얼로그가 다시 열린다.



① `CanNm`, ② `Nm`, ③ ***Import.***

다시 열면 임포트한 모듈이 선택된 상태로 보인다. 해제하고 **Apply** 를 누르면 그 모듈을 제거한다.

임포트 전체가 실행 취소 한 단계이다. 서비스 컴포넌트는 ECU 아래 `Composition/PRTE_<ECU>` 와 `Service/<ECU>/<Module>` 에 들어간다. 사용자가 고친 모듈 정의는 그대

8.16 PRTE Auto Configuration

ECU 하나의 BSW 모듈 설정을 설계로부터 기록한다. 설계가 정할 수 없는 값은 사용자나 AI에게 남긴다. 쓰기 권한이 필요하다.

1. ECU Configuration 다이얼로그(8.12) 머리글의 **PRTE AUTO CONFIGURATION** 을 누른다.
2. 미리보기를 읽는다. 아직 아무것도 기록되지 않는다. 영역별 파일과 **NOTES** 가 나온다. 파일을 누르면 바뀌는 내용이 보인다(12.2.6). 빠진 모듈이 있으면 `Not imported:` 에 ㅣ
3. **Configure** 또는 **Apply** 를 누른다. PAIO가 파일을 디스크에 바로 기록하므로 `Ctrl+S` 가 필요 없다.

실행 전체가 실행 취소 한 단계이다. 설계가 바뀌지 않았으면 다시 실행해도 아무것도 기록하지 않는다.

8.17 PRTE Code Generation

ECU 하나의 ECU Extract를 익스포트하고 그로부터 PRTE BSW 설정 코드를 생성한다. 쓰기 권한이 필요하다.

1. **EDITOR** 뷰 **DESIGN** 머리글의 **PRTE Code Generation** 을 누른다.
2. **Target ECU** 를 고른다.
3. **Generate** 를 누른다. 검증 오류가 있으면 생성이 멈춘다(14. Validation).
4. ECU Extract는 `export/<ECU>/` 에, 코드는 `<project>/code/<ECU>/` 에 기록된다.

호스트에서 실행된다. SSH 창에서는 PAIO가 프로젝트를 PC로 복사해 생성하고 결과를 서버에 다시 쓴다.

8.18 PARA Code Generation

HPC 하나의 Machine Extract를 익스포트하고 그로부터 PARA(AUTOSAR Adaptive Platform) 코드를 생성한다. 쓰기 권한이 필요하다.

1. **EDITOR** 뷰 **DESIGN** 머리글의 **PARA Code Generation** 을 누른다.
2. **Target HPC** 를 고른다.
3. **Generate** 를 누른다. 검증 오류가 있으면 생성이 멈춘다.
4. Machine Extract는 `export/<HPC>/` 에, 코드는 Machine마다 폴더 하나씩 `<project>/code/<Machine>/` 에 기록된다.

다시 실행하면 생성기가 소유한 파일은 덮어쓰고 사용자가 추가한 파일은 남긴다. 결과는 PARA vECU에서 빌드하고 실행한다(17. 원격 개발 (SSH · vECU 컨테이너)). PRTE처럼 SSH

8.19 Intrusion Detection System 다이얼로그

액션 줄의 두 번째 아이콘이 이 다이얼로그를 연다. 전용 에디터가 없는 `design/IDS.io.model` 을 편집한다.

- **IDSM INSTANCES**: ECU 또는 HPC 노드마다 IdsM 인스턴스 하나. 제한, 필터 체인, 서명, 컨텍스트 매핑 섹션이 있다.
- **SECURITY EVENTS**: 시스템 전체의 보안 이벤트. **HANDLED BY** 에는 그 이벤트를 보고하는 IdsM 인스턴스가 나열된다.

`Ctrl+S` 로 저장하며, 저장은 실행 취소 한 단계이다.

8.20 Software Cluster 다이얼로그

액션 줄의 세 번째 아이콘이 이 다이얼로그를 연다. Adaptive Platform의 소프트웨어 클러스터, 소프트웨어 패키지, 차량 패키지를 담은 `design/SOFTWARE_CLUSTER.io.mode` 열린다.

- **SOFTWARE CLUSTER**: `+` 로 노드에 클러스터를 추가한다. 하위 탭은 **CLUSTER**, **PACKAGE**, **LICENSE**, **RELEASE NOTES** 이고, **CONTENTS** 는 익스포트가 클러스터에 붙일 내
- **VEHICLE PACKAGE**: 하위 탭은 **CONFIGURATION**, **VEHICLE DESCRIPTION**, **ROLLOUT STEP** 이다.

`Ctrl+S` 로 저장하며, 저장은 실행 취소 한 단계이다.

SYSTEM 패널

이 장에서 다루는 것

SYSTEM 요소의 정보를 확인하고 편집하는 SYSTEM 패널을 설명한다. 이 패널에는 별도의 탭이 없다. 두 캔버스 오른쪽의 속성 패널 자리에 열린다.

- System 에디터(탭 `<project>:System`)에서는 **NAME**, **VERSION**, **DESCRIPTION** 과 SWC ↔ ECU 매핑의 읽기 전용 목록 **ECU MAPPING** 을 보여 준다.
- VFB 에디터(탭 `<project>:VFB`)에서는 **NAME**, **VERSION**, **DESCRIPTION**, **ROOT COMPOSITION** 과 컴포지션 계층의 읽기 전용 트리 **COMPOSITIONS** 를 보여 준다.

ECU, HPC 노드는 06. System 에디터에서, 컴포지션은 08. VFB 에디터에서 만든다.

11.1 여는 방법과 저장 위치

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System**(또는 **VFB**)을 누른다.
2. 캔버스 오른쪽 가장자리의 **System** 버튼을 누른다. 패널 토글과 잠금 버튼 아래의 세 번째 버튼이다.
3. 속성 패널이 제목 **SYSTEM** 으로 열린다.

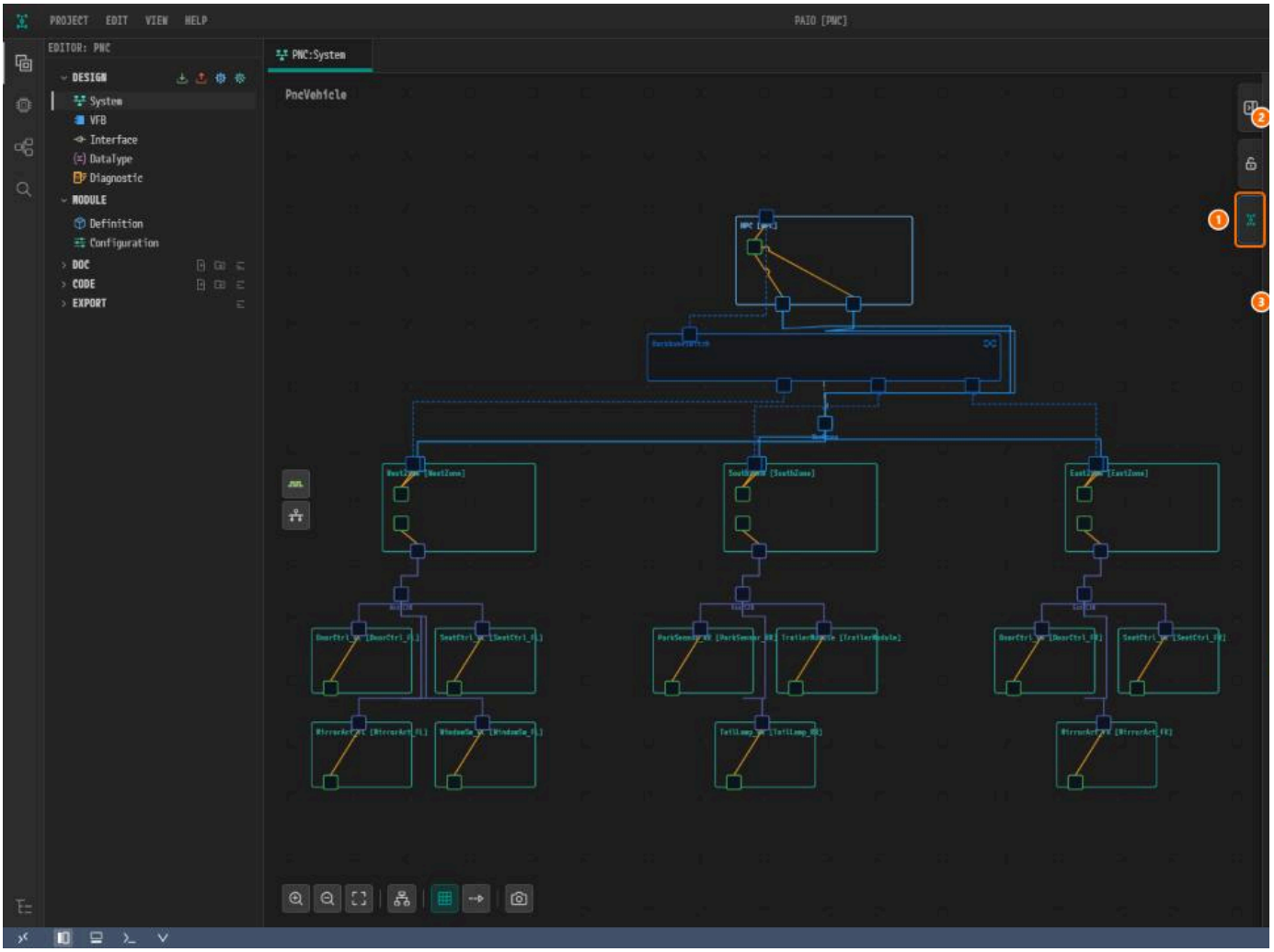


*① 패널 토글, ② 잠금, ③ **System** 버튼.*

저장 단계는 없다. 바꾼 값은 바로 `design/SYSTEM.io.model` 에 기록된다.

11.2 화면 구성

패널 위쪽에 필드가 있고, 그 아래에 목록(System 에디터) 또는 트리(VFB 에디터)가 있다.



*① **System** 버튼, ② **NAME, VERSION, DESCRIPTION**, ③ ECU, HPC 노드마다 한 행씩 있는 **ECU MAPPING**.*

- 노드를 선택하면 패널이 그 노드로 바뀐다. 돌아가려면 **System** 을 다시 누른다.
- 패널이 잠겨 있지 않으면 빈 캔버스를 누를 때 패널이 접힌다.

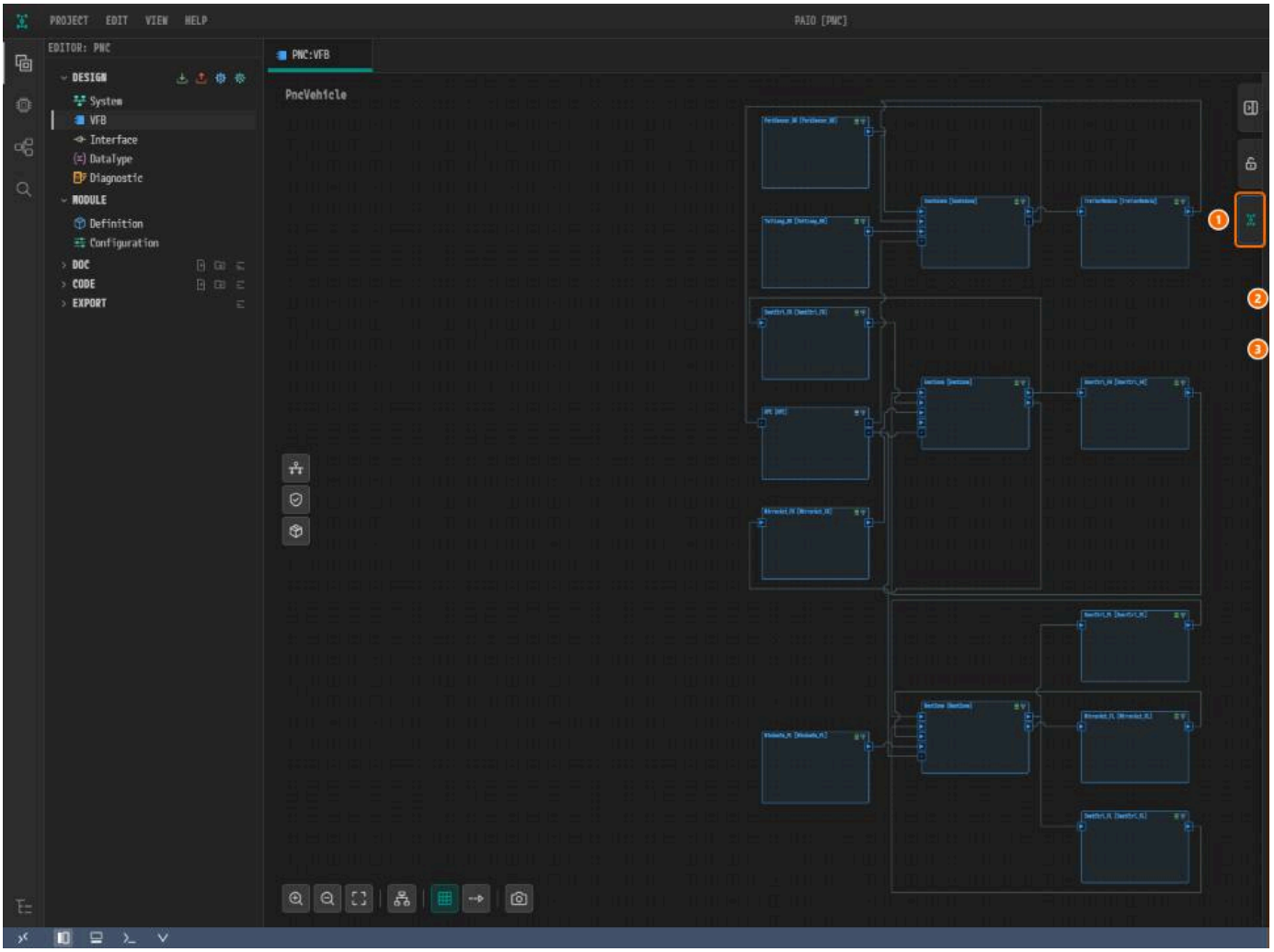
11.3 시스템 정보

1. **NAME** 에 SYSTEM 요소의 SHORT-NAME 을 입력한다. `Enter` 를 누르거나 포커스가 빠지면 확정된다.
2. **VERSION** 에 `1.0.0` 처럼 `major.minor.patch` 형식으로 입력한다.
3. **DESCRIPTION** 에 자유 텍스트를 입력한다.

- 비어 있거나 규칙에 어긋난 **NAME** 은 기록되지 않는다. 입력 칸이 저장된 이름으로 돌아가고 오류가 표시된다.
- 형식이 어긋난 **VERSION** 에는 경고가 나오지만 값은 그대로 기록된다.

11.4 컴포지션 트리 (VFB 에디터)

VFB 에디터 패널의 **COMPOSITIONS** 트리는 컴포지션 계층을 보여 준다. 읽기 전용이다. 컴포지션의 추가, 삭제, 이름 변경은 08. VFB 에디터에서 한다.



*① **System** 버튼, ② **ROOT COMPOSITION**, ③ **COMPOSITIONS** 트리.*

11.4.1 행 구성

각 행은 SWC 종류 아이콘, 프로토타입 이름, 그리고 이름이 다를 때 기울임꼴의 SWC 타입 이름을 보여 준다.

11.4.2 행 종류

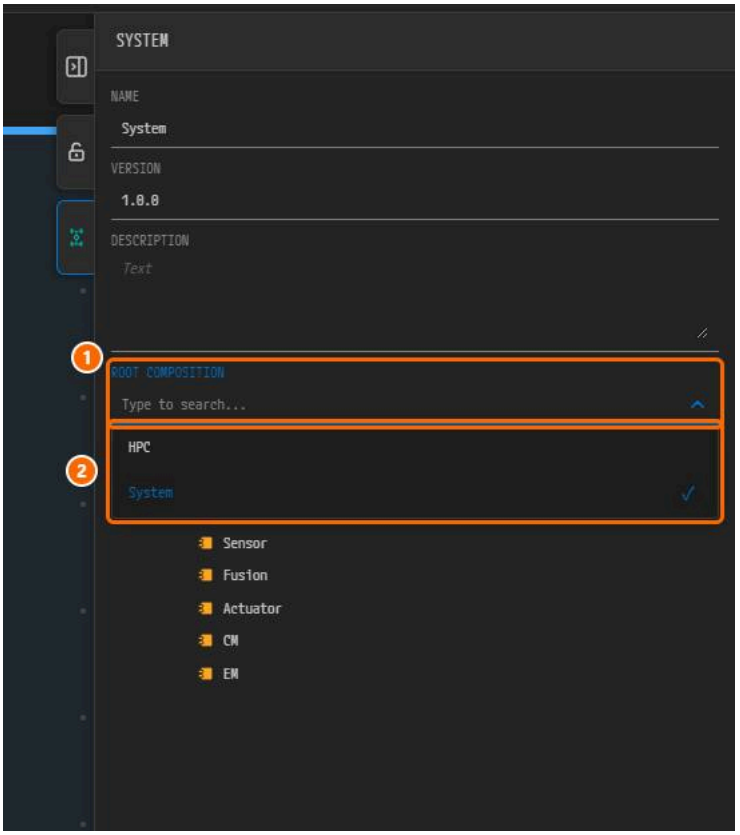
트리는 컴포지션과 그 프로토타입을 나열한다. 행의 아이콘에 포인터를 올리면 `COMPOSITION` 같은 종류가 표시된다.

11.4.3 트리에서 이동하기

컴포지션 행을 누르면 VFB 캔버스가 그 컴포지션 안으로 드릴다운된다. 트리에서는 이름을 편집할 수 없다.

11.4.4 루트 컴포지션 지정

1. **ROOT COMPOSITION** 을 누른다. 후보 목록이 열리고 현재 루트에 체크 표시가 있다.
2. 컴포지션을 고른다. VFB 모델이 새 루트를 기준으로 재구성된다.



*① 검색 입력이 나타난 **ROOT COMPOSITION**, ② 현재 루트에 체크 표시가 있는 후보.*

- 먼저 VFB 에디터를 저장한다. 저장하지 않은 편집이 있으면 선택이 거부된다.
- `Ctrl+Z` 를 한 번 누르면 이전 루트로 돌아간다.

11.4.5 ECU/HPC 배치

배치는 이 패널에서 편집하지 않는다. 다음 메뉴 가운데 하나를 쓴다. 결과는 **ECU MAPPING**(11.5)에 나타난다.

- **System 에디터**: ECU 또는 HPC 노드 우클릭 ▸ **SW Mapping**
- **VFB 에디터**: 최상위 컴포지션 우클릭 ▸ **ECU/HPC Mapping**

11.5 ECU Mapping

System 에디터 패널의 **ECU MAPPING** 은 ECU, HPC 노드와 각 노드에 매핑된 컴포지션을 보여 주는 읽기 전용 목록이다. 행을 누르면 캔버스에서 그 노드가 선택된다. ECU, HPC 가
다.

11.6 통계 (제거됨)

예전 System 대시보드의 통계 패널은 제거되었다. 규칙 검사는 14. Validation에서 한다.

11.6.1 System Completeness

제거되었다. 대신 **ECU MAPPING**(11.5)을 확인한다.

11.6.2 Architecture Quality

제거되었다. 대신 14. Validation을 쓴다.

11.7 저장 · 실행 취소 · 단축키

- 저장할 것이 없다. 바꾼 값은 이미 기록되어 있다.
- 되돌리려면 `Ctrl+Z` 를 누르거나 **EDIT** 메뉴를 쓴다.
- **VERSION** 은 키 입력마다 기록되므로 실행 취소도 키 입력 하나씩 되돌아간다.

11.7.1 외부 변경과 충돌

패널에는 저장하지 않은 상태가 없으므로 충돌 경고도 없다. 쓰기는 자기 필드만 바꾸므로 두 SYSTEM 패널이 서로의 값을 덮어쓰지 않는다.

11.8 다른 에디터와의 연동

- 두 SYSTEM 패널은 같은 시스템 정보를 편집한다.
- **ECU MAPPING** 은 System 에디터 캔버스에서, **COMPOSITIONS** 트리는 VFB 다이어그램에서 온다.
- 데이터 매핑(포트의 데이터 요소 ↔ 시그널)은 System 모델에 기록되지만 편집은 08. VFB 에디터에서 한다.

11.9 자주 겪는 상황

증상	할 일
SYSTEM 패널이 사라짐	**System** 을 다시 누르거나 패널을 잠근다
NAME 이 예전 이름으로 돌아옴	입력 칸 아래의 오류를 확인한다
ROOT COMPOSITION 선택이 거부됨	VFB 에디터를 저장한 뒤 다시 고른다

Interface 에디터

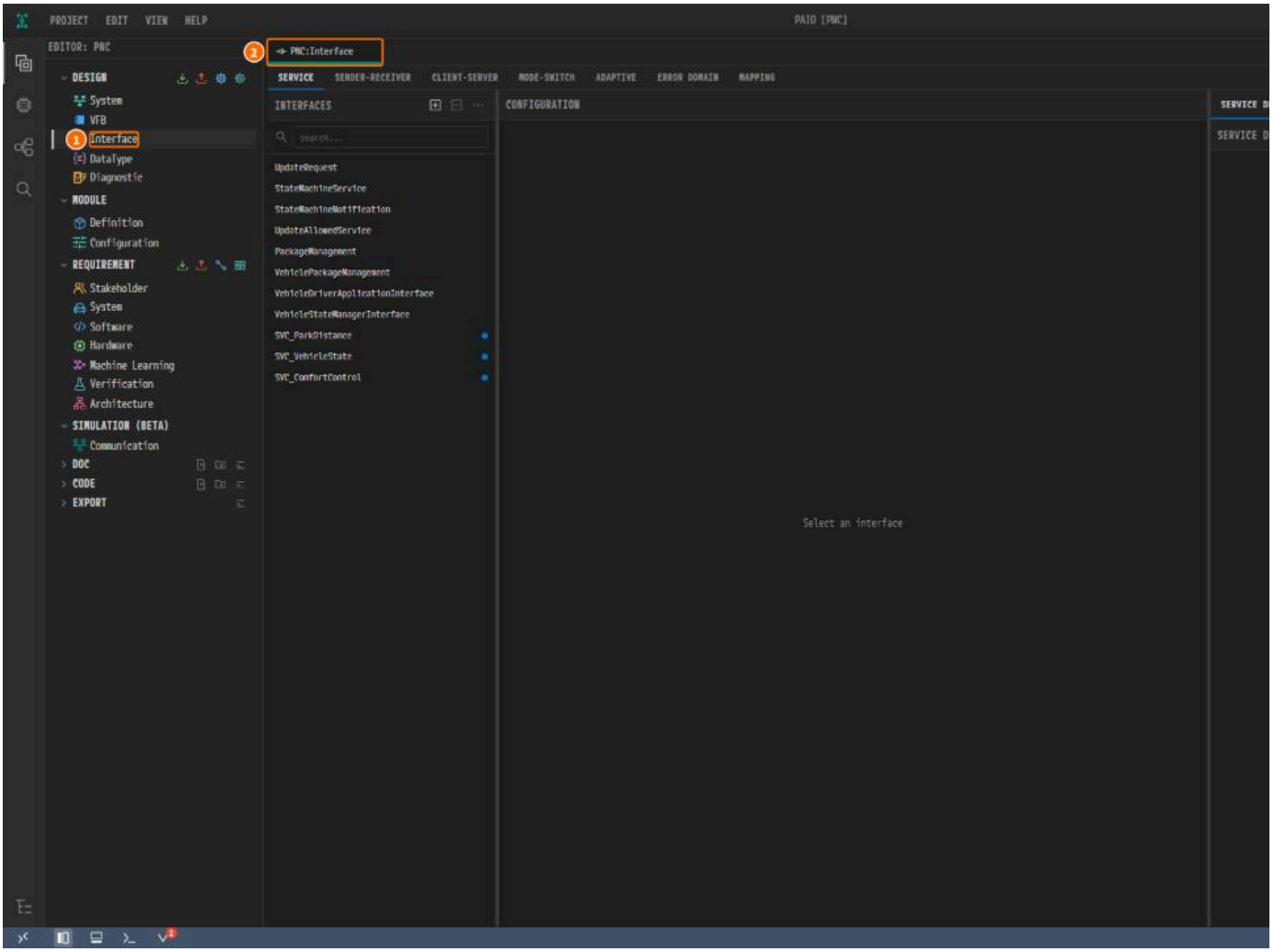
이 장에서 다루는 것

Interface 에디터의 사용법을 설명한다. 이 에디터는 SWC 포트가 참조하는 포트 인터페이스를 정의하고, AP 서비스 인터페이스의 배포(SOME/IP, DDS, IPC)와 서비스 인스턴스를 실

- 대상 독자: AUTOSAR 인터페이스와 서비스 배포를 정의하는 사용자
- 선행 조건: 프로젝트가 열려 있는 것. 파생 기능은 라이선스가 활성 상태여야 한다.

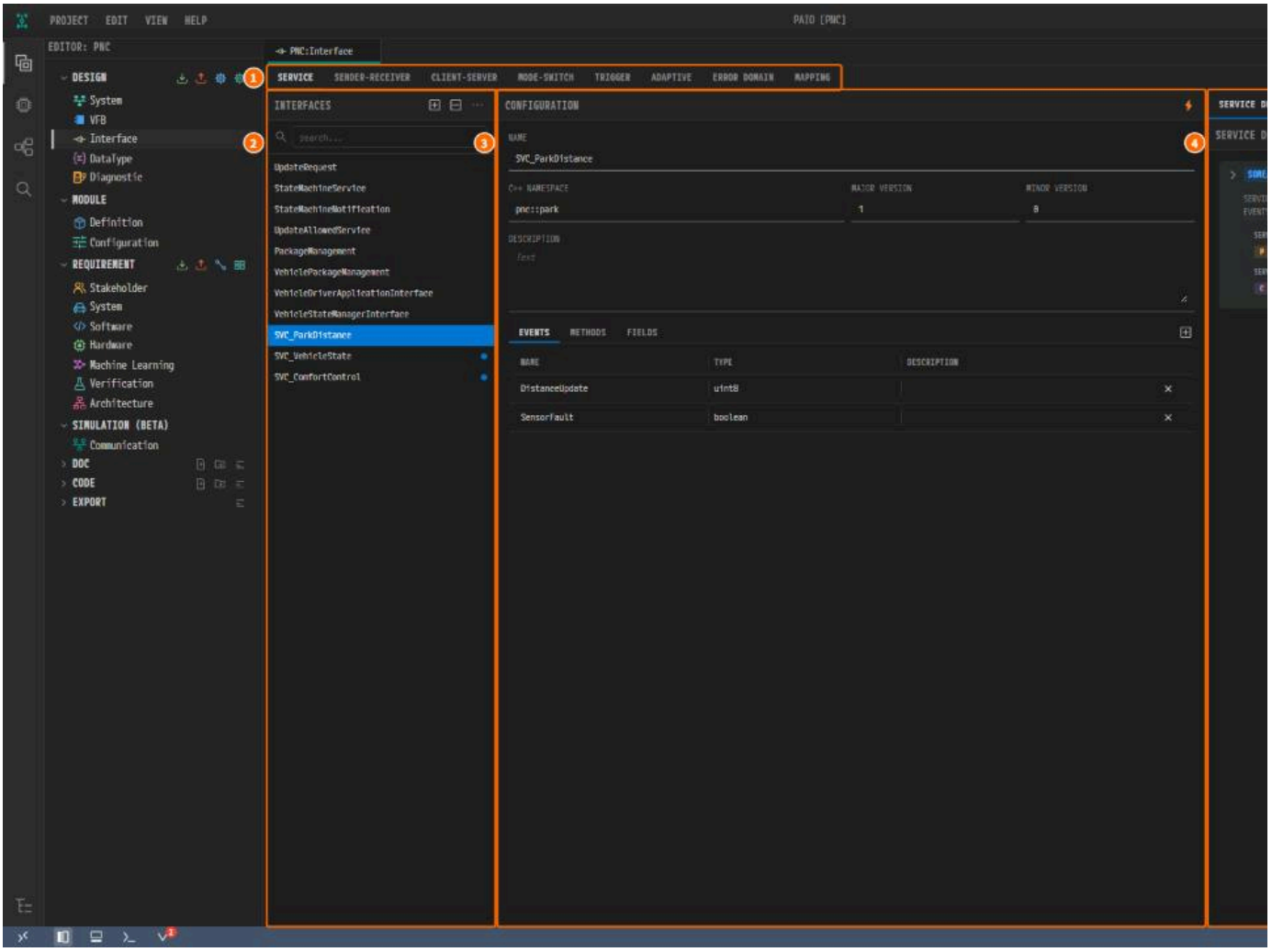
데이터 타입은 이 화면 안에서 정의할 수 있으므로 10. DataType 에디터를 미리 열 필요는 없다.

왼쪽 **EDITOR** 뷰의 **DESIGN** 아래에서 **Interface** 를 누르면 탭 `<project>:Interface` 가 열린다. `design/INTERFACE.io.model` 이나 `design/SERVICE.io.model` 을 E
로 열린다.



*① **DESIGN** 아래의 **Interface**. ② 탭 `PNC:Interface`.*

9.1 화면 구성

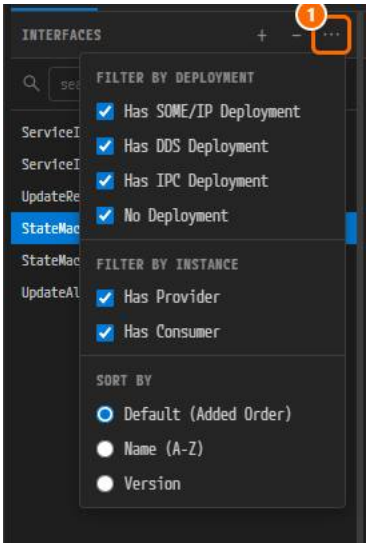


*① 카테고리 탭 8종. ② **INTERFACES** 패널. ③ **CONFIGURATION** 패널. ④ 오른쪽 패널.*

세 패널 사이 경계를 드래그해 너비를 조절할 수 있다.

왼쪽 INTERFACES 패널

- `+` 는 현재 탭에 맞는 항목을 추가하고, `-` 는 확인 대화상자를 거쳐 선택한 항목을 지운다.
- `…` 는 정렬·필터 팝오버를 열고, 검색 상자는 이름으로 거른다.
- `Ctrl`+클릭은 행을 선택에 더하고, `Ctrl+A` 는 모든 행을 선택한다.



① *Service 탭의 ⋯ (**Sort & Filter**) 버튼.*

가운데 CONFIGURATION 패널

선택한 인터페이스의 항목을 보여 준다. 머리글의 ⚡ 버튼은 파생 동작이다(9.5).

인터페이스는 Interface 모델에, 서비스 배포와 서비스 인스턴스는 Service 모델에 저장된다. `Ctrl+S` 한 번으로 둘이 DataType 모델과 함께 기록된다. InterfaceMapping 만은

9.2 카테고리 탭 (8종)



*카테고리 탭 8종. **SERVICE** 가 활성 상태이다.*

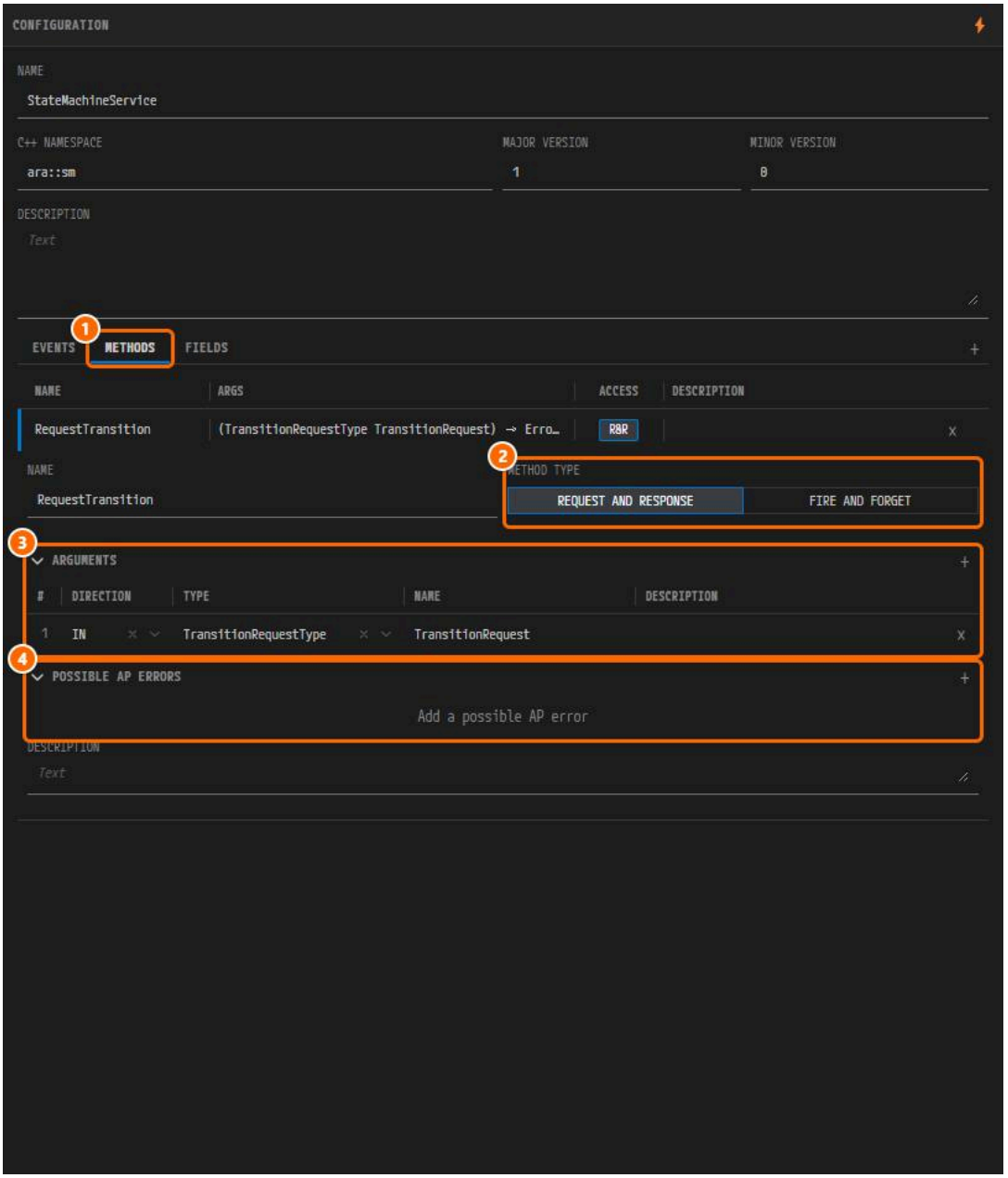
탭 레이블	다루는 것
SERVICE	AP ServiceInterface + 서비스 배포 + 서비스 인스턴스
SENDER-RECEIVER	CP SenderReceiverInterface
CLIENT-SERVER	CP ClientServerInterface
MODE-SWITCH	CP ModeSwitchInterface
TRIGGER	CP TriggerInterface
ADAPTIVE	AP 기능 클러스터별 전용 포트 인터페이스
ERROR DOMAIN	AP 오류 도메인
MAPPING	AP 서비스 인터페이스와 CP 인터페이스를 잇는 InterfaceMapping

9.3 카테고리별 편집 내용

모든 탭에 **NAME** 과 자유 입력 **DESCRIPTION** 이 있다. 아래에서는 탭마다 다른 것만 다룬다. 이름 규칙은 9.8 을 따른다.

9.3.1 Service

1. **SERVICE** 탭에서 `+` 를 누르고 새 인터페이스를 선택한다.
2. **NAME**, **C++ NAMESPACE**, **MAJOR VERSION**, **MINOR VERSION** 을 정한다.
3. **EVENTS**, **METHODS**, **FIELDS** 하위 탭을 열고 머리글의 `+` 로 요소를 추가한다.
4. 메서드 행을 펼쳐 **METHOD TYPE**, **ARGUMENTS**, **POSSIBLE AP ERRORS** 를 정한다.

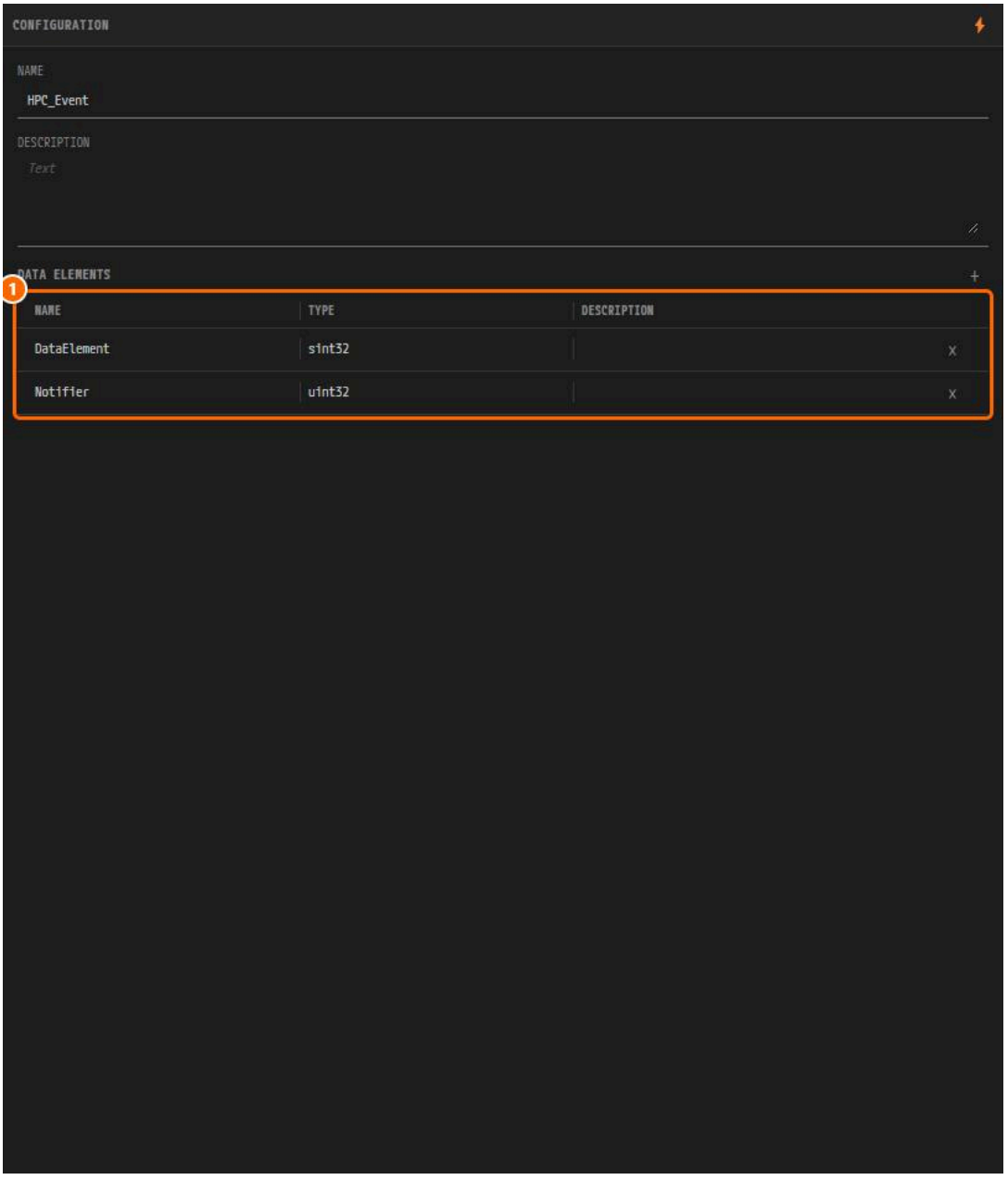


*① **METHODS** 하위 탭. ② **METHOD TYPE**. ③ **ARGUMENTS**. ④ **POSSIBLE AP ERRORS**.*

- **POSSIBLE AP ERRORS** 는 Error Domain 탭의 오류를 참조하므로, 도메인을 먼저 만든다.
- 인자 행의 `#` 칸을 끌어 인자 순서를 바꾼다.

9.3.2 Sender-Receiver

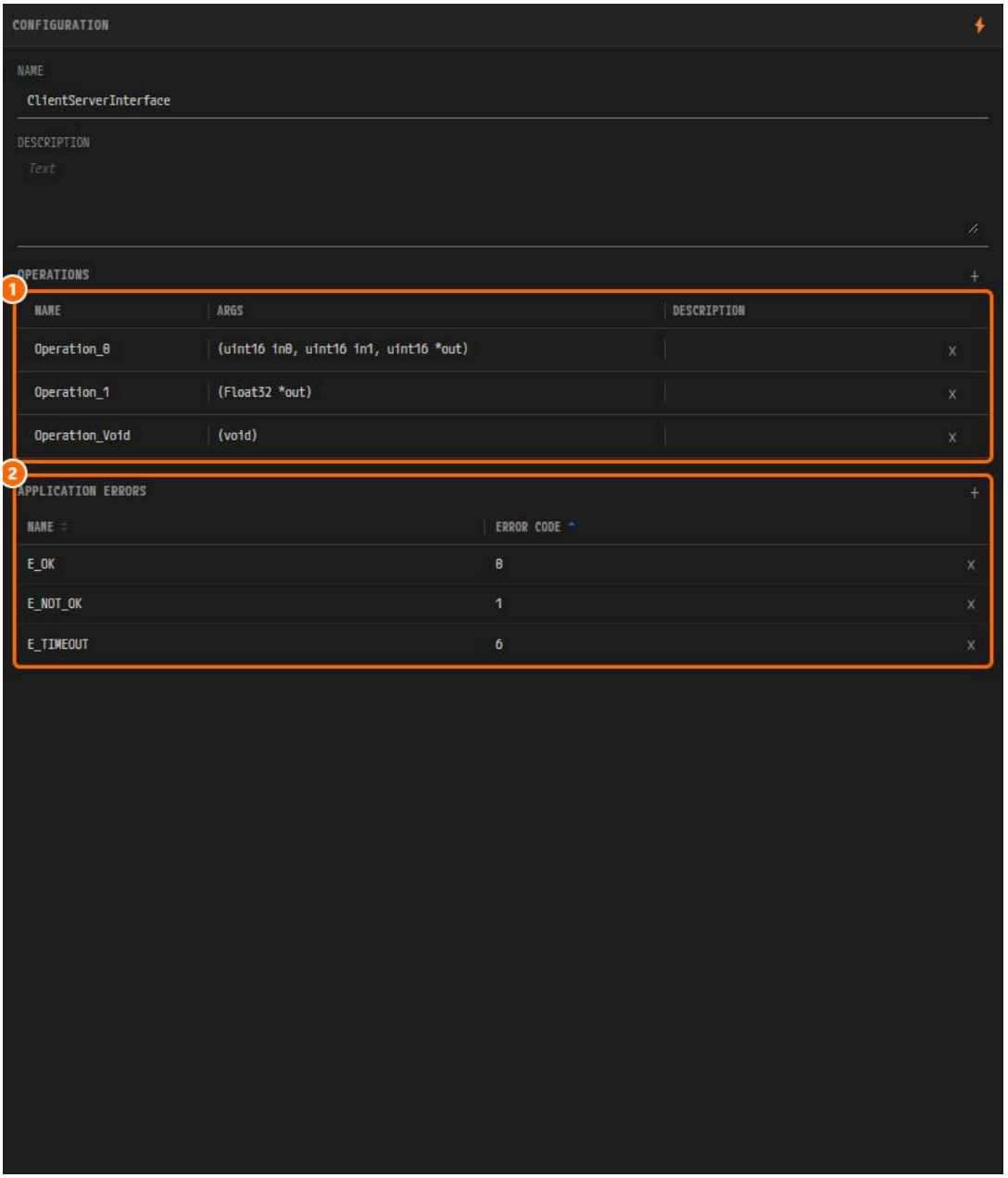
DATA ELEMENTS 표에 데이터 요소를 추가하고, 각각 **NAME** 과 **TYPE** 을 정한다.



① ***DATA ELEMENTS 표.***

9.3.3 Client-Server

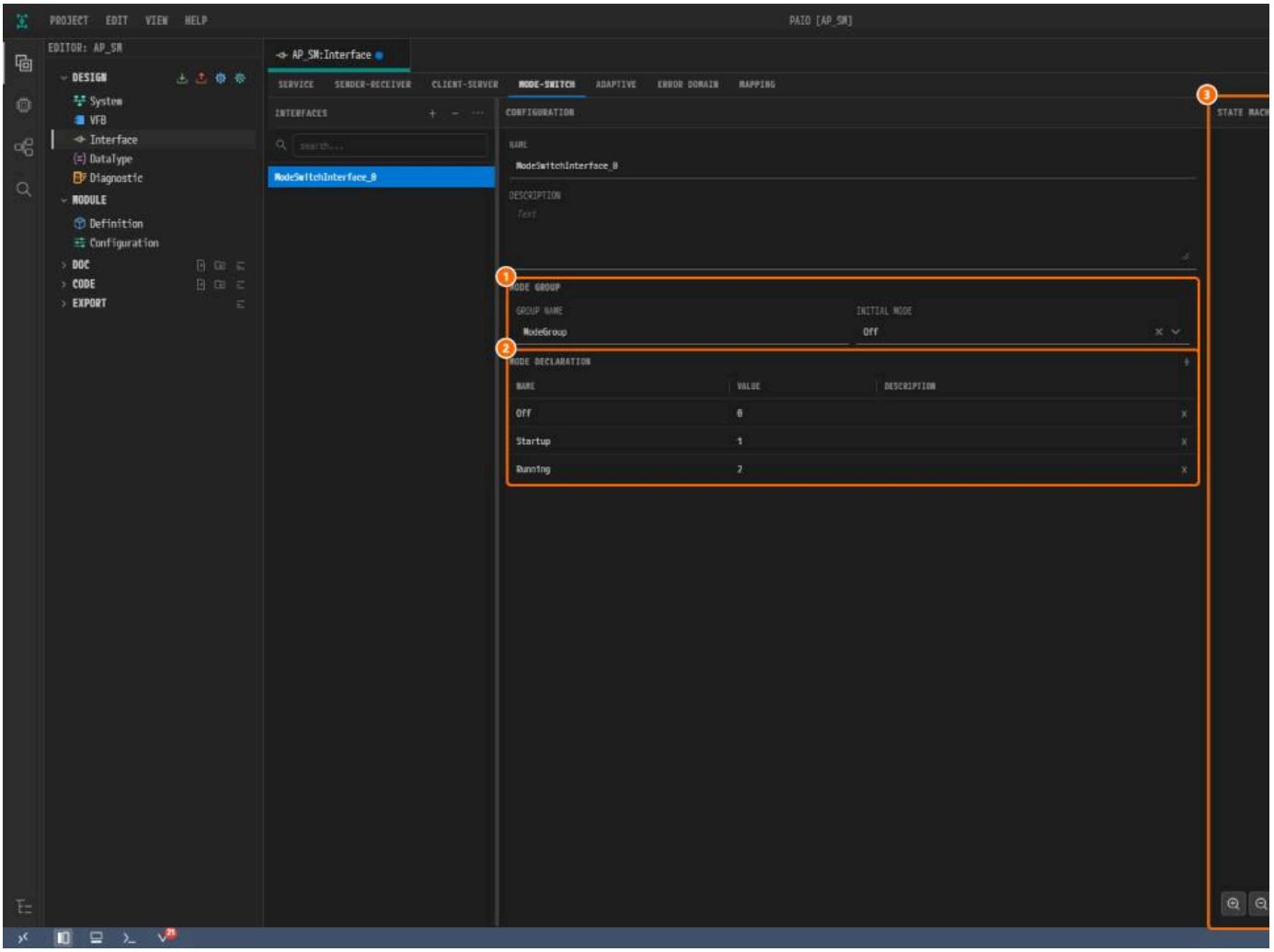
OPERATIONS 에 오퍼레이션을 추가하고, 펼쳐서 **ARGUMENTS**(**IN** / **OUT** / **INOUT**)를 정한다. 아래의 **APPLICATION ERRORS** 는 인터페이스 전체에 걸린다.



*① **ARGS** 에 오퍼레이션이 C 매개변수 목록으로 나오는 **OPERATIONS**. ② **APPLICATION ERRORS**.*

9.3.4 Mode-Switch

1. **MODE DECLARATION** 의 `+` 로 모드 선언을 추가한다.
2. **MODE GROUP** 에서 **INITIAL MODE** 를 고른다.
3. 오른쪽 **STATE MACHINE** 다이어그램에서 전이를 그린다. 노드를 `Shift` + 드래그해 다른 노드 위에 놓는다. 전이는 자동으로 만들어지지 않는다.

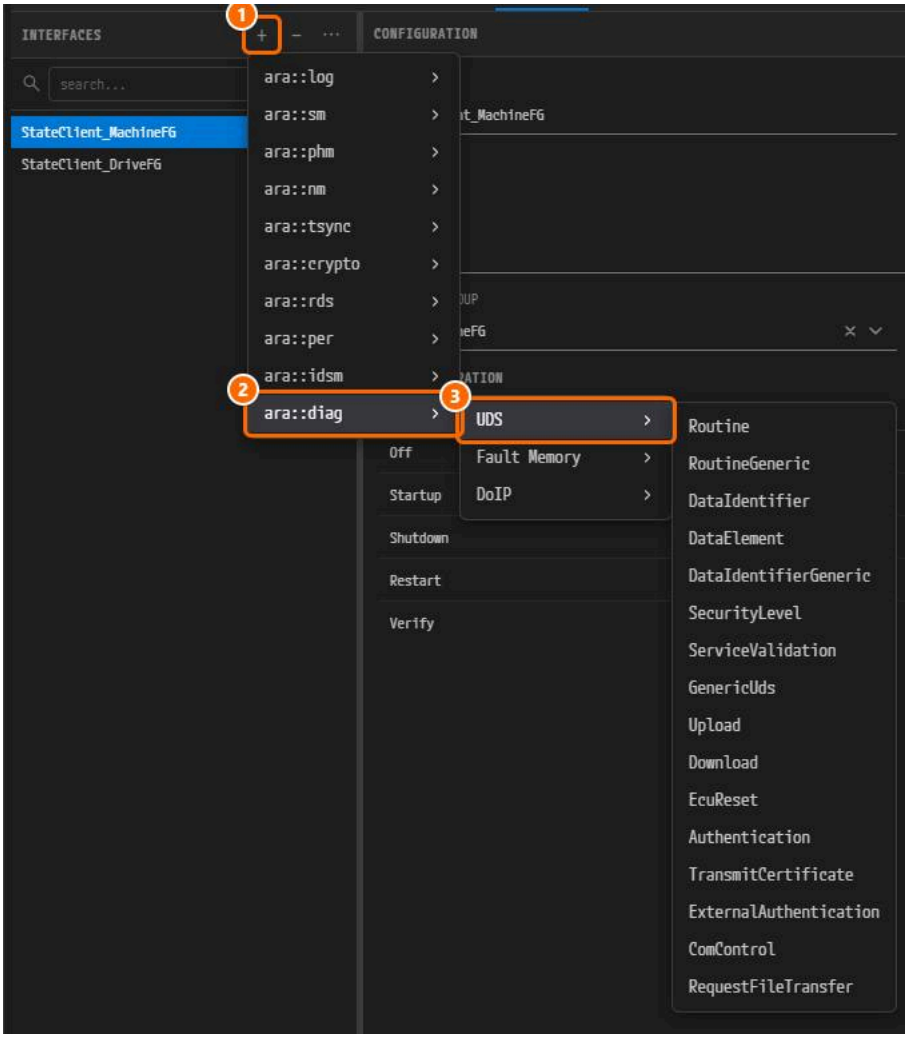


*① **INITIAL MODE** 가 `Off` 인 **MODE GROUP**. ② **MODE DECLARATION**. ③ **STATE MACHINE** 다이어그램.*

9.3.5 Adaptive

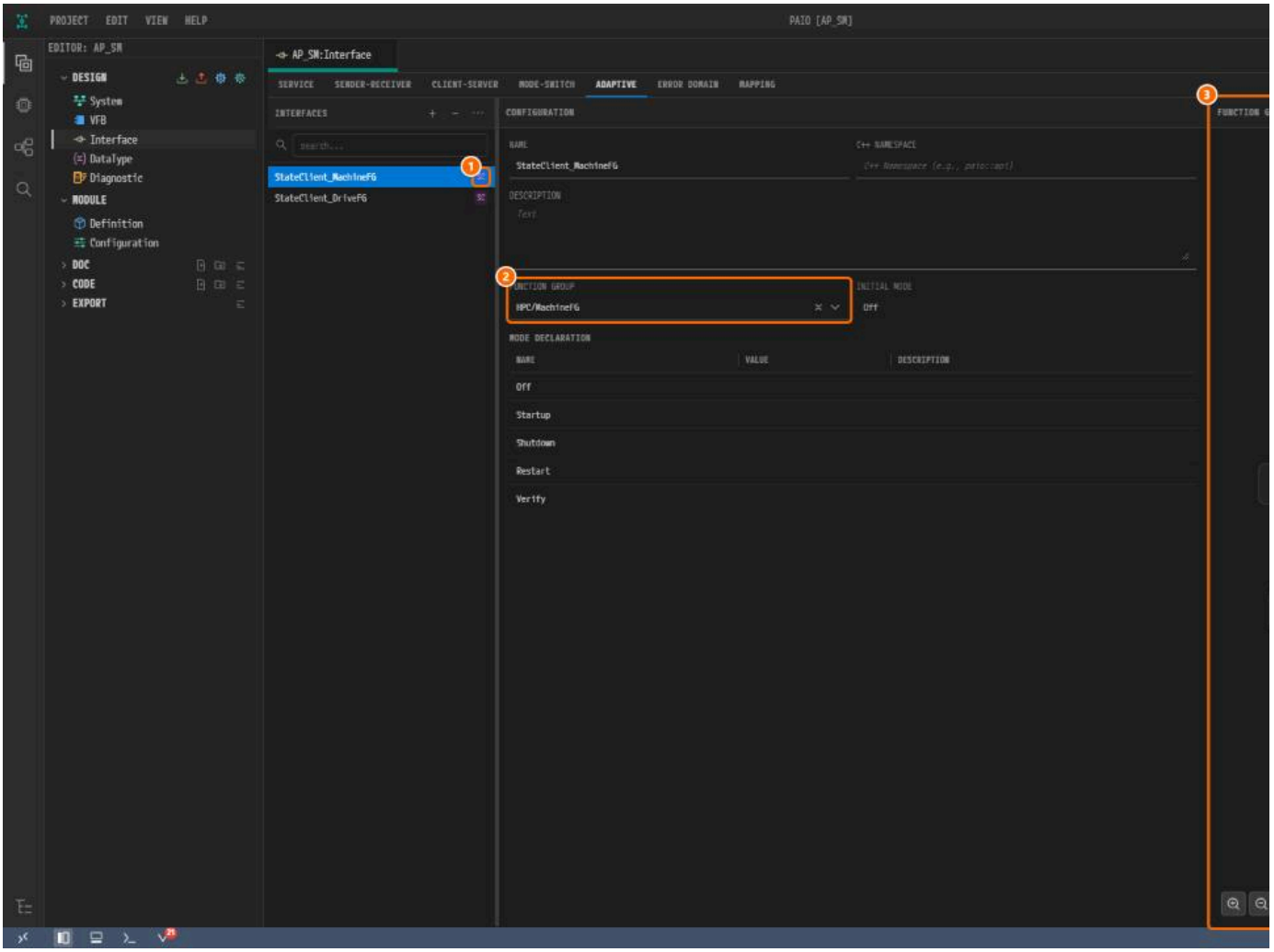
AP 기능 클러스터의 전용 인터페이스를 만든다.

1. `+` 를 누른다. 클러스터 10개의 목록이 열린다.
2. 클러스터에 마우스를 올리고 인터페이스 종류를 누른다. **ara::diag** 는 **UDS**, **Fault Memory**, **DoIP** 로 한 단계 더 들어간다.



① `+` 버튼. ② ***ara::diag***. ③ ***UDS*** 하위 메뉴.

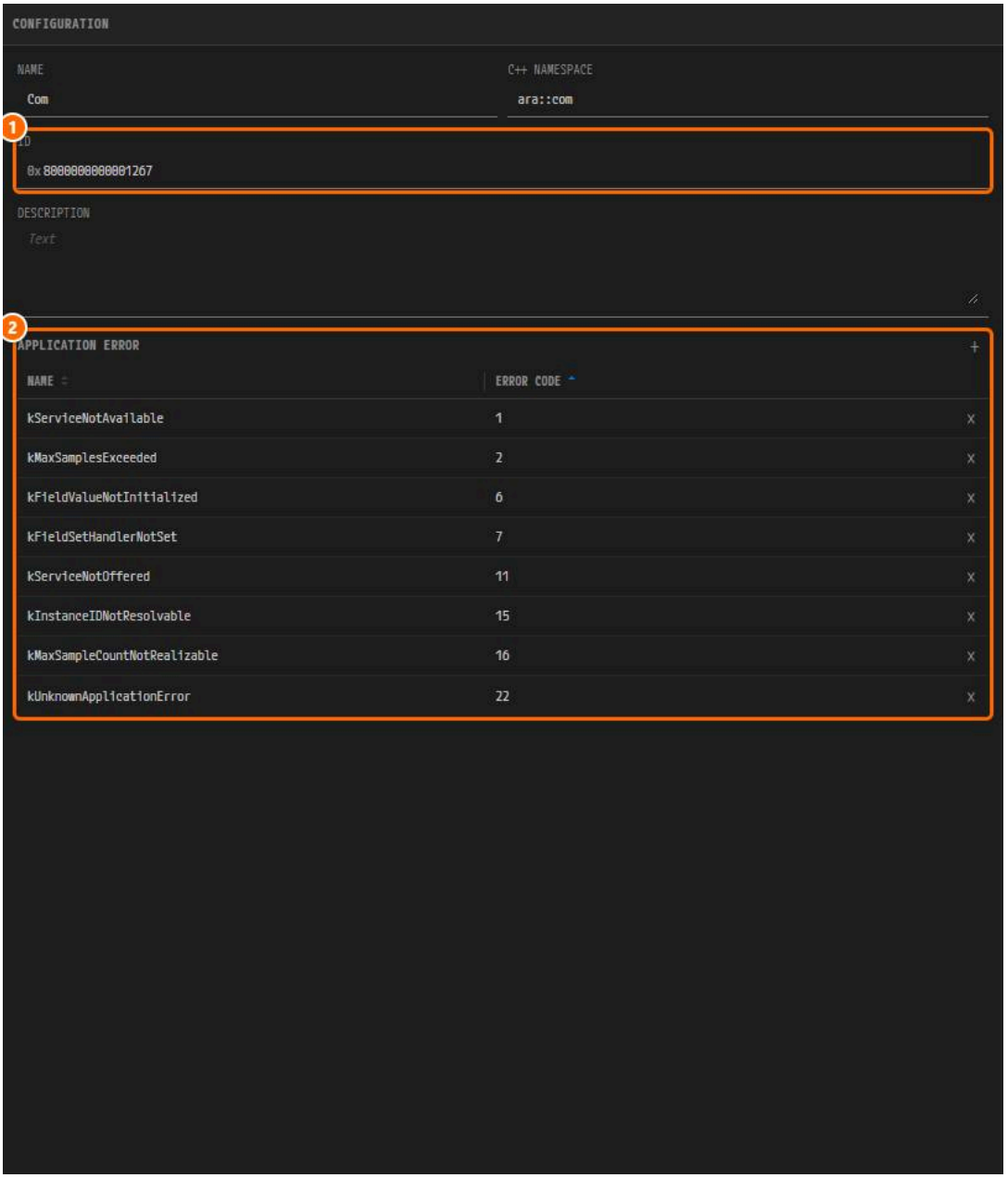
이름 뒤의 배지가 종류를 나타낸다. **StateClient** 는 Machine Configuration 다이얼로그에서 정의한 **FUNCTION GROUP** 을 고르며, 그러면 오른쪽 패널에 그 상태 머신이 나온다.



① `SC` 배지. ② `HPC/MachineFG` 로 지정된 ***FUNCTION GROUP***. ③ ***FUNCTION GROUP STATE MACHINE*** 뷰어.

9.3.6 Error Domain

도메인 **ID** 를 16진수로 정하고(`0x` 접두사는 고정), **APPLICATION ERROR** 에 `NAME` 과 `ERROR CODE` 로 오류를 추가한다.

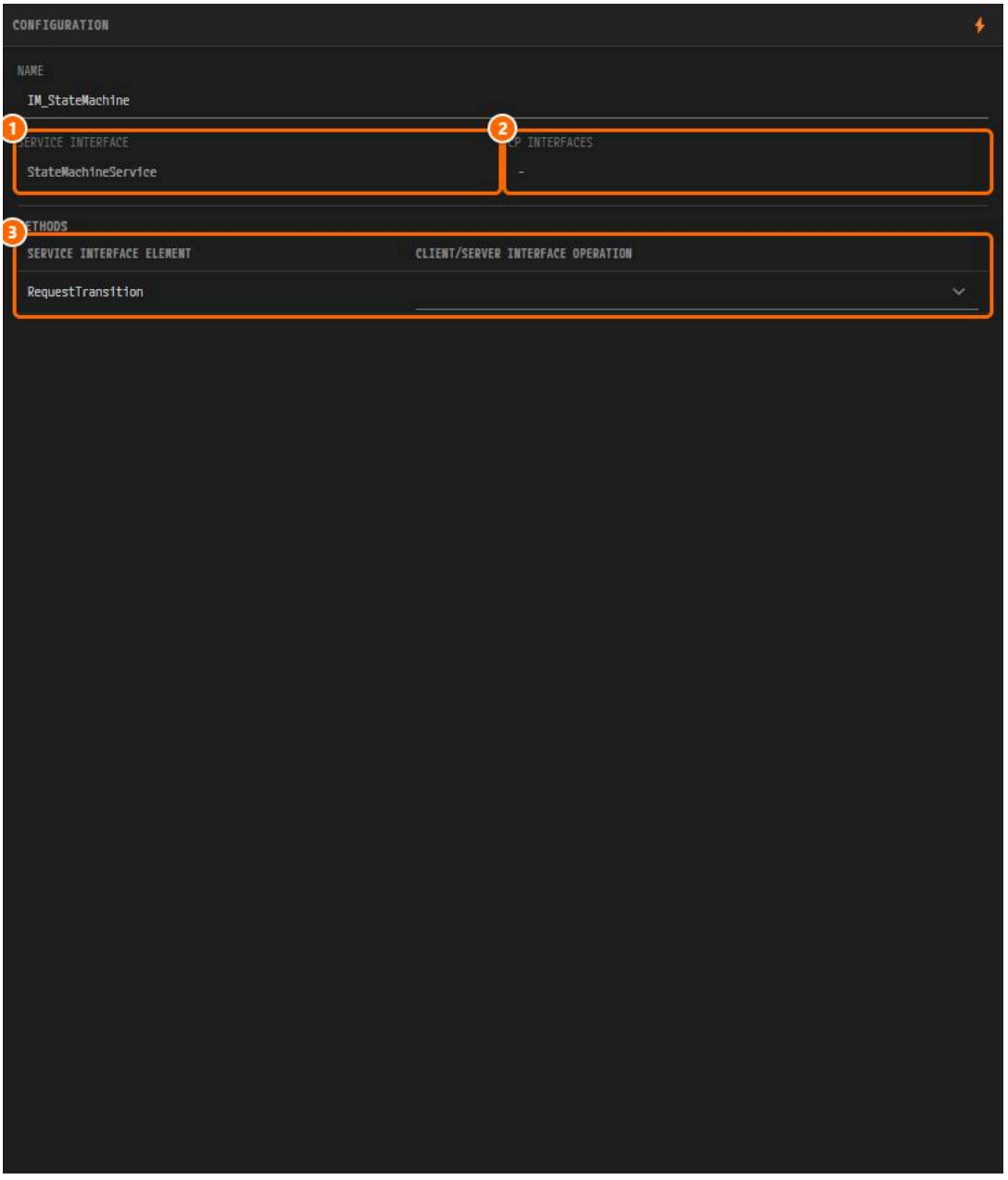


*① **ID**. ② **APPLICATION ERROR**.*

9.3.7 Mapping

AP 서비스 인터페이스와 CP 인터페이스를 잇는 InterfaceMapping 을 정의한다.

1. `+` 를 누른다. 첫 **Add Interface Mapping** 창에 이름을 넣고, 둘째 창에서 서비스 인터페이스를 고르거나 비워 둔다.
2. 매핑을 선택한다. **SERVICE INTERFACE** 와 **CP INTERFACES** 는 읽기 전용이다.
3. **EVENTS**, **METHODS**, **FIELDS** 섹션에서 서비스 요소마다 오른쪽 열의 CP 요소를 고른다.



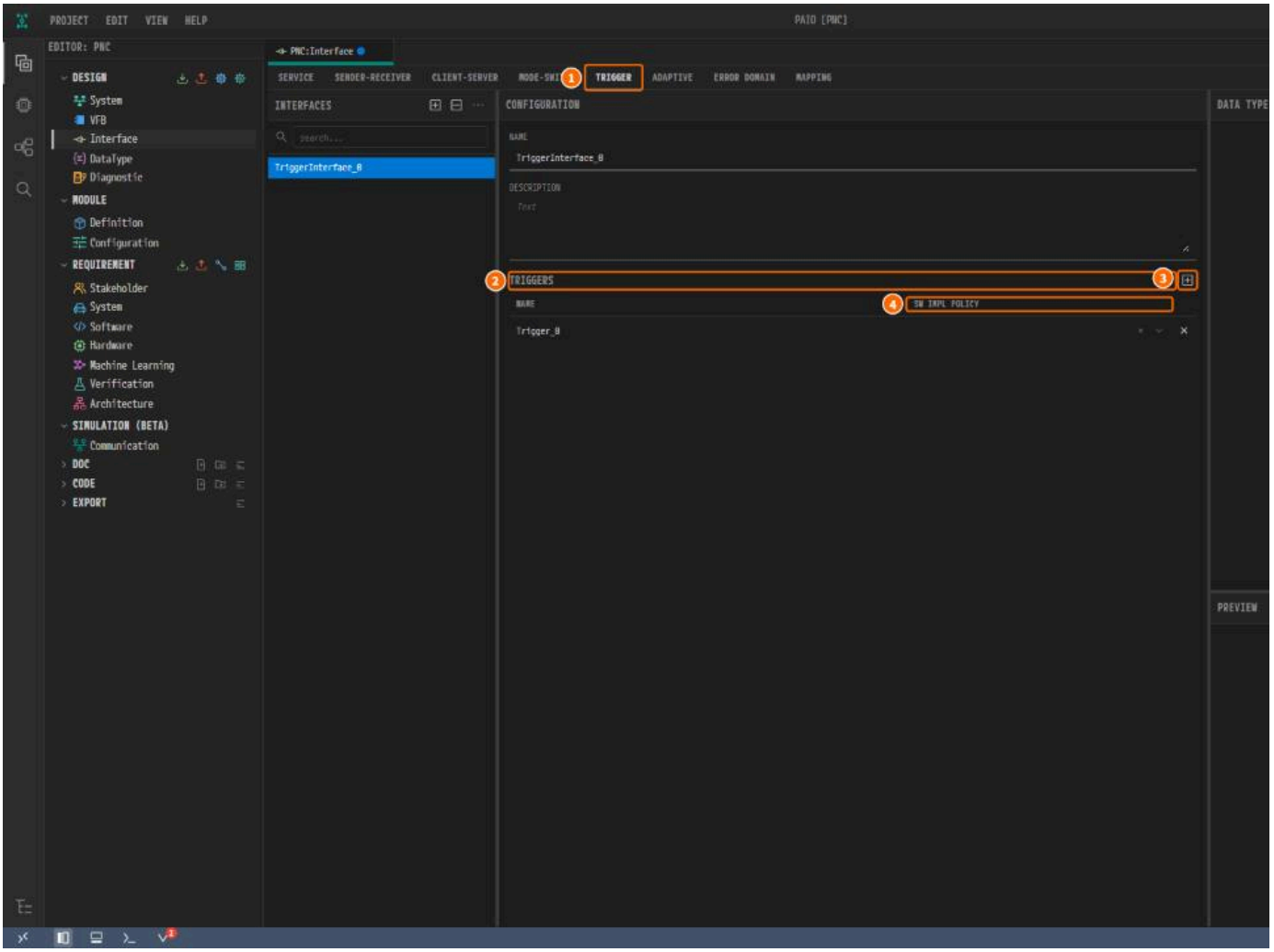
*① **SERVICE INTERFACE**. ② **CP INTERFACES**. ③ **METHODS** 섹션.*

InterfaceMapping 은 Interface 모델이 아니라 VFB 모델에 저장되며, 편집하는 즉시 반영된다.

9.3.8 Trigger

Classic SW-C 포트가 트리거를 보내거나 받을 때 쓰는 트리거 인터페이스를 정의한다.

1. **TRIGGER** 탭에서 `+` 를 누르고 새 인터페이스를 선택한다.
2. **TRIGGERS** 에서 `+` (**Add Trigger**)를 눌러 트리거를 추가한다.
3. 트리거의 **Name** 을 정하고, 필요하면 **Sw Impl Policy**(`STANDARD` 또는 `QUEUED`)를 고른다.



① TRIGGER 탭. ② TRIGGERS. ③ +(Add Trigger). ④ Sw Impl Policy.

트리거 이름은 인터페이스 안에서 겹치면 안 된다. VFB 에디터에서 트리거 인터페이스를 쓰는 포트에는 트리거 아이콘이 나온다(8.8 참조).

9.4 오른쪽 패널

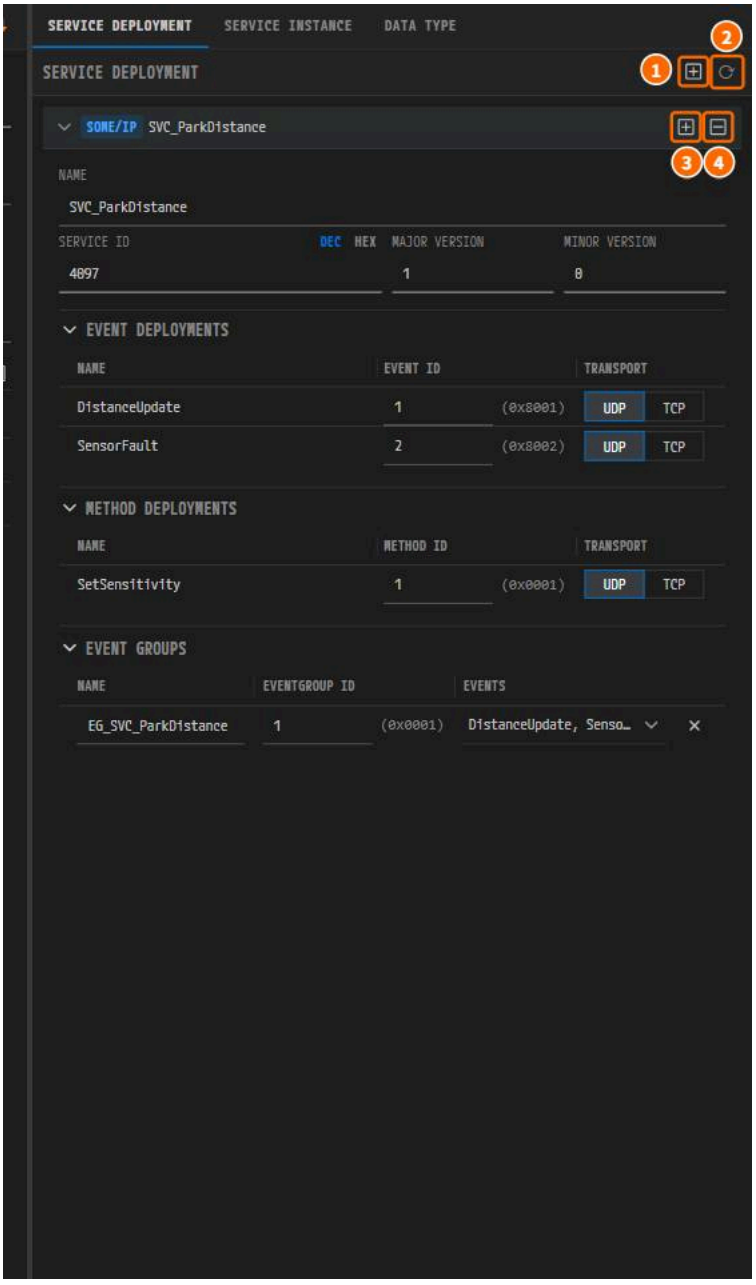
오른쪽 패널은 탭과 선택에 따라 바뀐다.

상황	오른쪽 패널
SERVICE / **SENDER-RECEIVER** / **CLIENT-SERVER** 탭	**SERVICE DEPLOYMENT** / **SERVICE INSTANCE** / **DATA TYPE** 하위 탭
MODE-SWITCH 탭	**STATE MACHINE** 다이어그램
ADAPTIVE 탭 + StateClient 선택	**FUNCTION GROUP STATE MACHINE** 뷰어 (읽기 전용)
그 밖의 모든 경우	내장 DataType 편집 패널

Sender-Receiver 와 Client-Server 탭에서는 하위 탭이 선택한 CP 인터페이스에 매핑된 ServiceInterface 를 대상으로 동작한다.

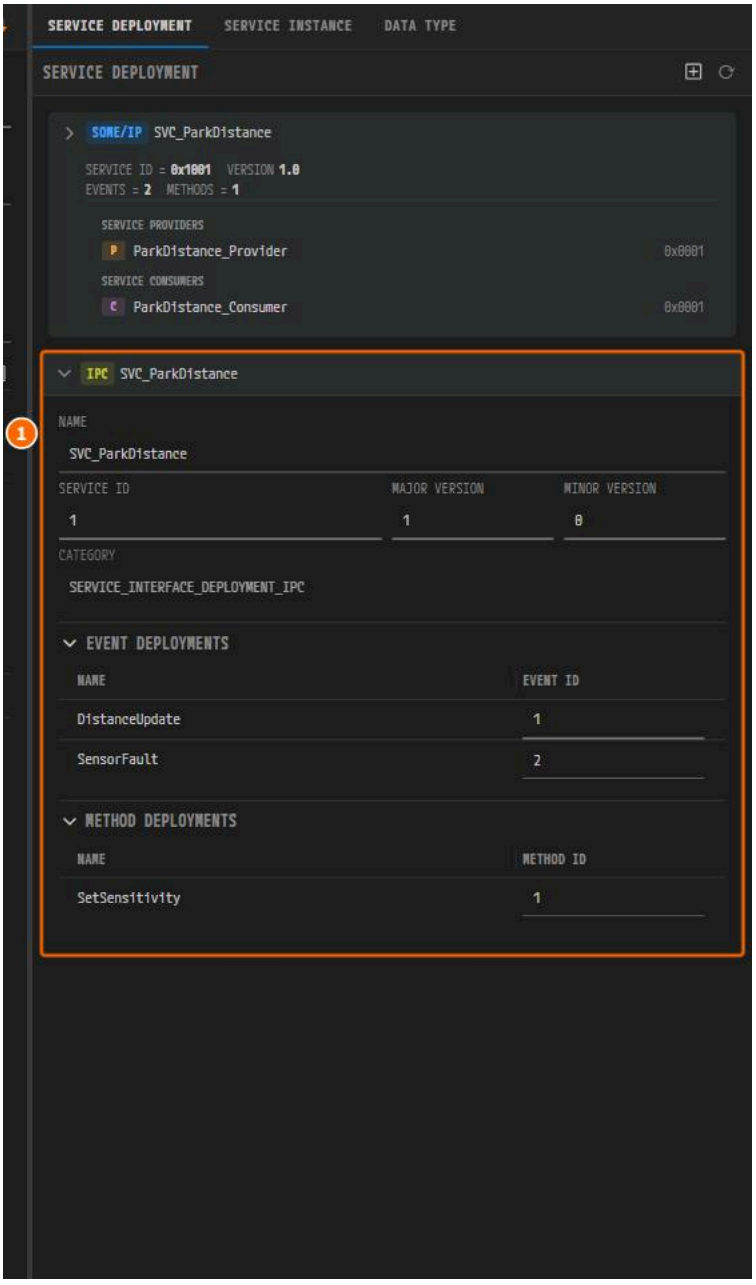
9.4.1 SERVICE DEPLOYMENT

1. 서비스 인터페이스를 선택하고 **SERVICE DEPLOYMENT** 를 연다.
2. +(**Add Deployment**)를 누르고 **SOME/IP**, **DDS**, **IPC** 중 하나를 고른다. 배포 카드가 생긴다.
3. 카드에 **SERVICE ID**, 버전 등을 채운다.



① `+` *(**Add Deployment**).* ② `⟳` *(**Derive Signal**).* ③ ④ *카드 머리글의* `+` *(**Add service instance**)와* `-` *(**Remove deployment**).*

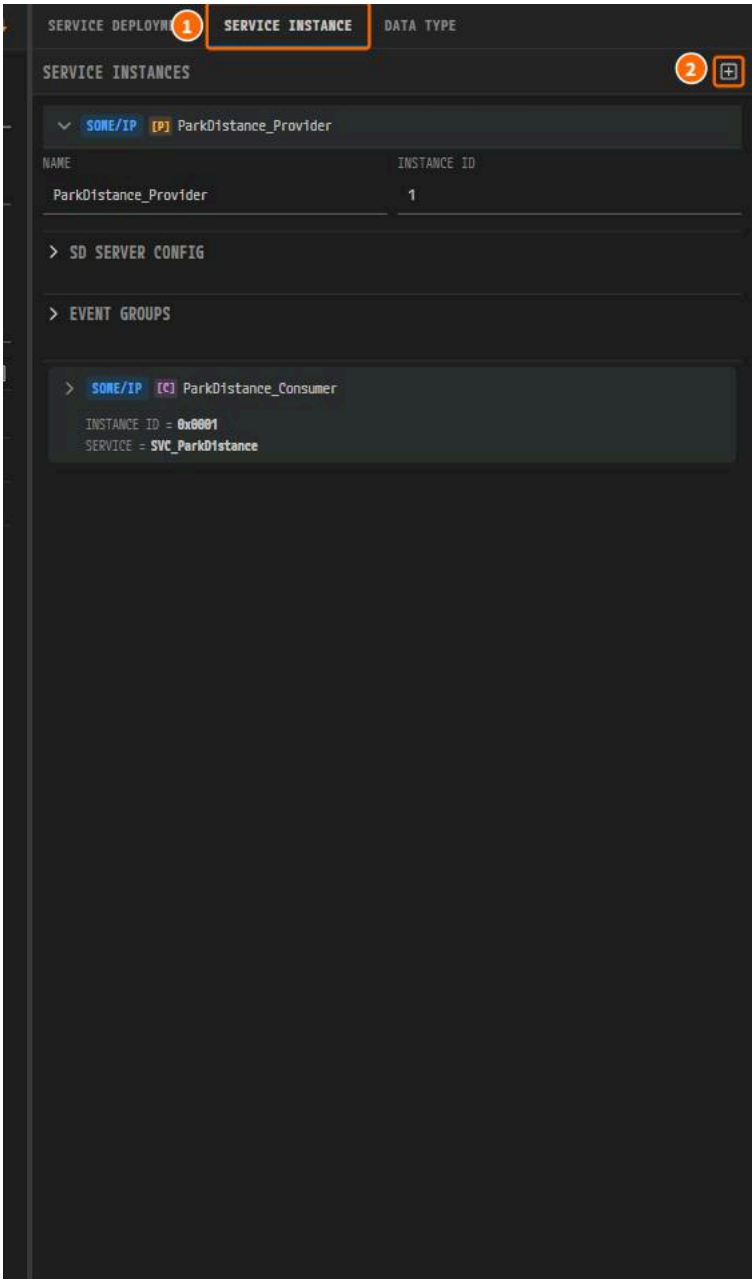
- `⟳` 는 CP-SOA 파생을 수동으로 다시 실행한다. 평소에는 자동으로 실행된다.
- 배포를 지우면 그 배포를 쓰는 서비스 인스턴스도 함께 지워진다.



① 새 IPC 카드.

9.4.2 SERVICE INSTANCE

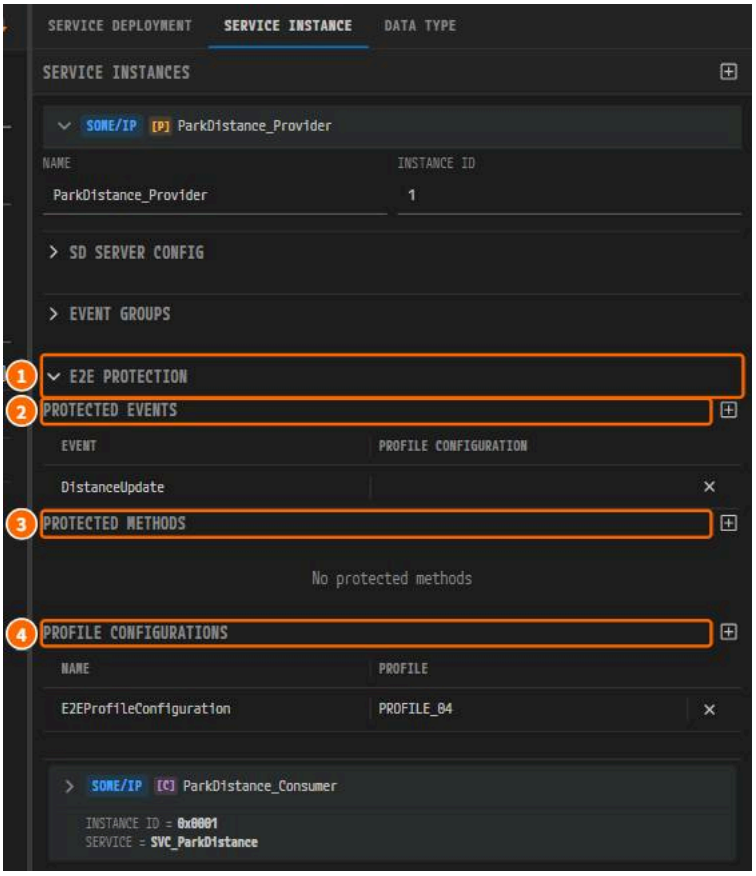
`+` 를 누르고 배포의 제공자나 소비자(예: **SOME/IP Provider**)를 고른다. 제공자에는 `[P]`, 소비자에는 `[C]` 배지가 붙는다.



*① **SERVICE INSTANCE** 하위 탭. ② 인스턴스를 추가하는 `+`.*

SOME/IP 제공자·소비자 카드에는 종단 간(E2E) 보호를 설정하는 **E2E PROTECTION** 섹션도 있다.

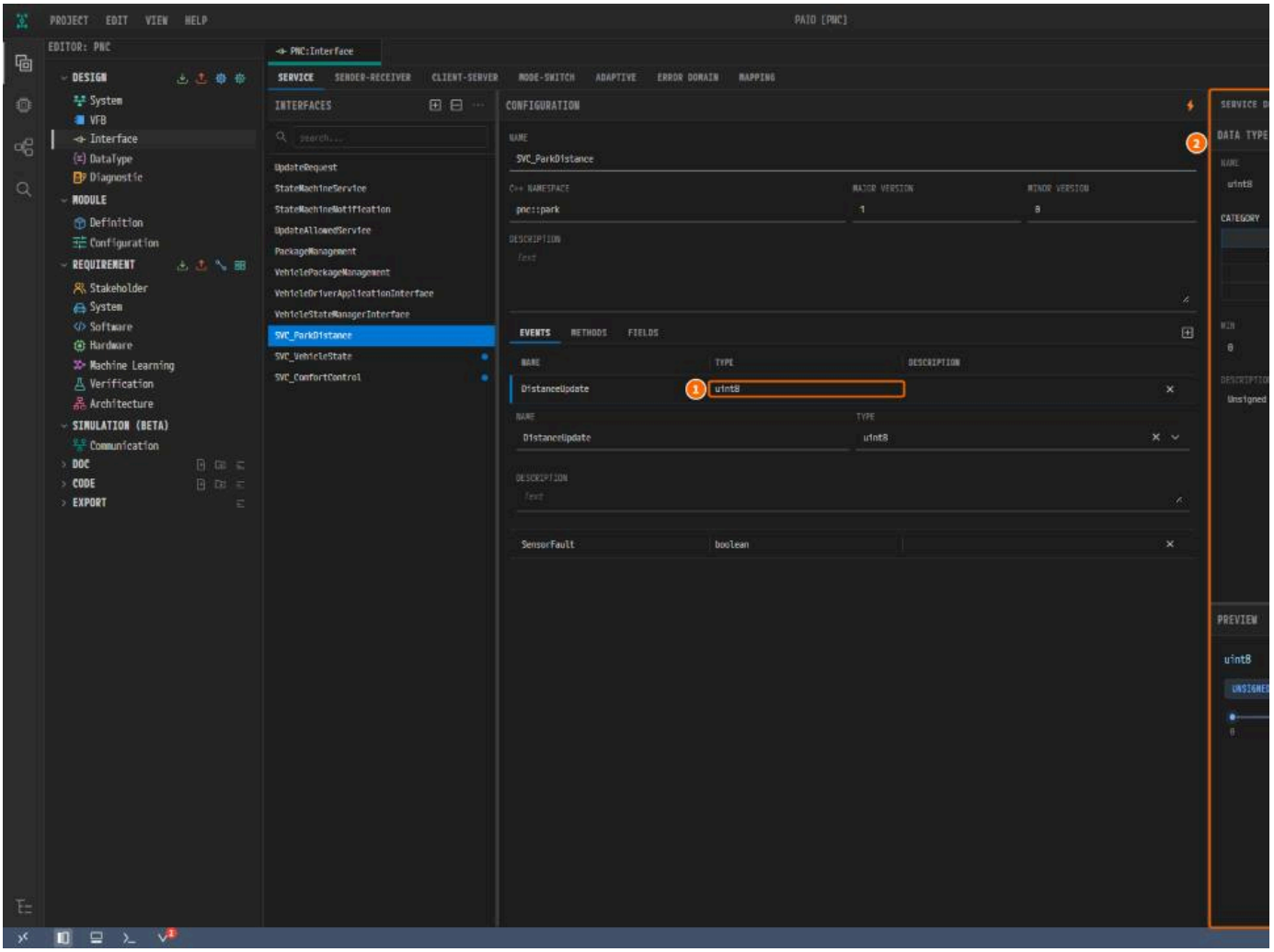
- **PROFILE CONFIGURATIONS**: `+` 로 프로파일 설정을 추가한다. 그 행을 펼쳐 **Name**, **Profile**(`PROFILE_04`, `PROFILE_05` 등), **Data ID Mode**, **Max Delta Counter** 등의 값
  카드에 같은 목록이 나온다.
- **PROTECTED EVENTS**, **PROTECTED METHODS**: `+` 를 누르면 서비스에서 보호할 수 있는 이벤트나 메서드 목록이 나오고, 그중 하나를 고른다. 새 행을 펼쳐 **Profile Configu**



*① **E2E PROTECTION**. ② 이벤트 `DistanceUpdate` 를 보호하도록 추가한 **PROTECTED EVENTS**. ③ **PROTECTED METHODS**. ④ **PROFILE CONFIGURATIONS**.*

9.4.3 DATA TYPE

인터페이스 요소의 **TYPE** 을 누르면 오른쪽 패널이 **DATA TYPE** 으로 바뀌어 그 타입을 보여 준다. 편집 내용은 10. DataType 에디터와 같은 DataType 모델에 반영된다.

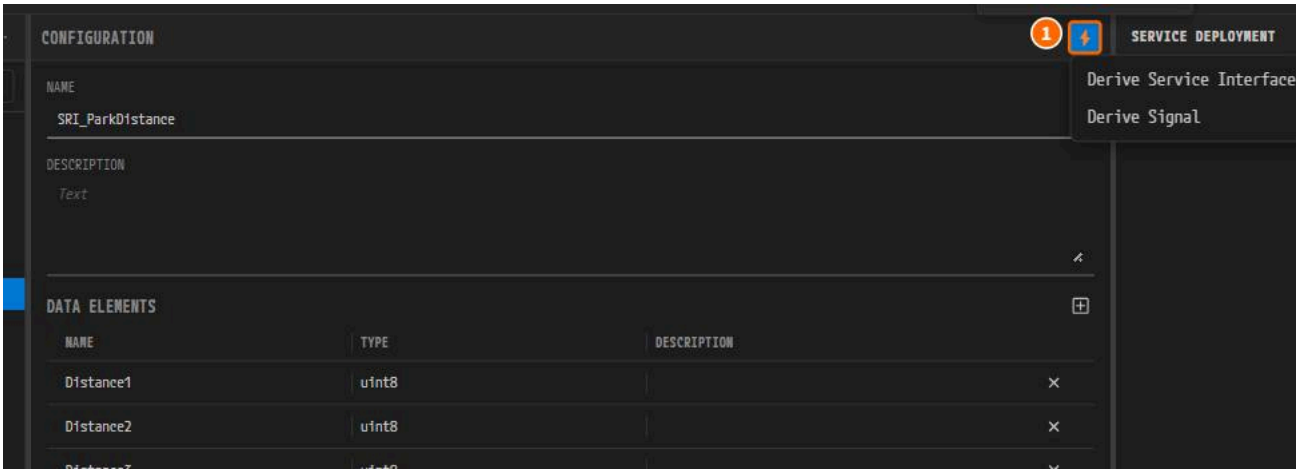


① 이벤트의 타입 `uint8` 를 누른다. ② 오른쪽 패널에 `uint8` 가 나온다.

9.5 인터페이스 파생 (Derive)

CONFIGURATION 머리글의 ⚡ 버튼은 선택한 인터페이스에서 짝이 되는 인터페이스를 만든다. 조건이 맞지 않으면 버튼이 숨겨진다.

- **SERVICE** 나 **MAPPING** 탭에서는 `{이름}_SR`, `{이름}_CS` SR/CS 인터페이스를 곧바로 만든다.
- **SENDER-RECEIVER** 나 **CLIENT-SERVER** 탭에서는 **Derive Service Interface** 가 서비스 인터페이스 이름을 묻는다. 기존 이름을 고르면 그 인터페이스에 요소가 추가된다.
- **SENDER-RECEIVER** 탭의 **Derive Signal** 은 Signal 에디터에 시그널과 PDU 를 만든다.



① CONFIGURATION 머리글의 ⚡.

파생은 VFB 모델의 InterfaceMapping 도 함께 채우며, `Ctrl+Z` 한 번으로 전부 되돌아간다.

9.6 단축키

단축키는 공통 단축키를 따른다(18.4 참조). `Ctrl+S` 한 번이 INTERFACE, SERVICE, 내장 DataType 을 함께 기록한다. `Delete` 는 목록에서 선택한 행을 확인을 거쳐 지운다.

9.7 저장 · 실행 취소 · 외부 변경

- `Ctrl+S` 한 번으로 Interface, Service, DataType 모델이 함께 저장된다.
- 실행 취소의 단위는 편집 하나이다. 두 파일에 걸친 편집도 `Ctrl+Z` 한 번으로 되돌아간다.
- 저장하지 않은 편집이 있을 때 디스크의 파일이 바뀌면, PAIO 는 편집 내용을 지키고 경고를 표시한다.

9.8 이름 규칙

이름은 영문자로 시작하고, 이후는 영문자, 숫자, 밑줄이다. 규칙을 어긴 이름에는 주황색 경고가 나온다. 거부하는 자리와 경고만 하고 받아들이는 자리가 있으므로, 익스포트 전에 경고

9.9 다른 에디터와의 연동

- VFB 에디터에서 포트에 인터페이스를 지정할 때 여기서 정의한 인터페이스가 후보로 나온다(08. VFB 에디터 참조).
- 인터페이스 이름을 바꾸면 그것을 참조하는 SWC 포트와 ECUC 참조도 함께 바뀐다.
- ARXML 임포트·익스포트는 13. 임포트 · 익스포트에서 다룬다.

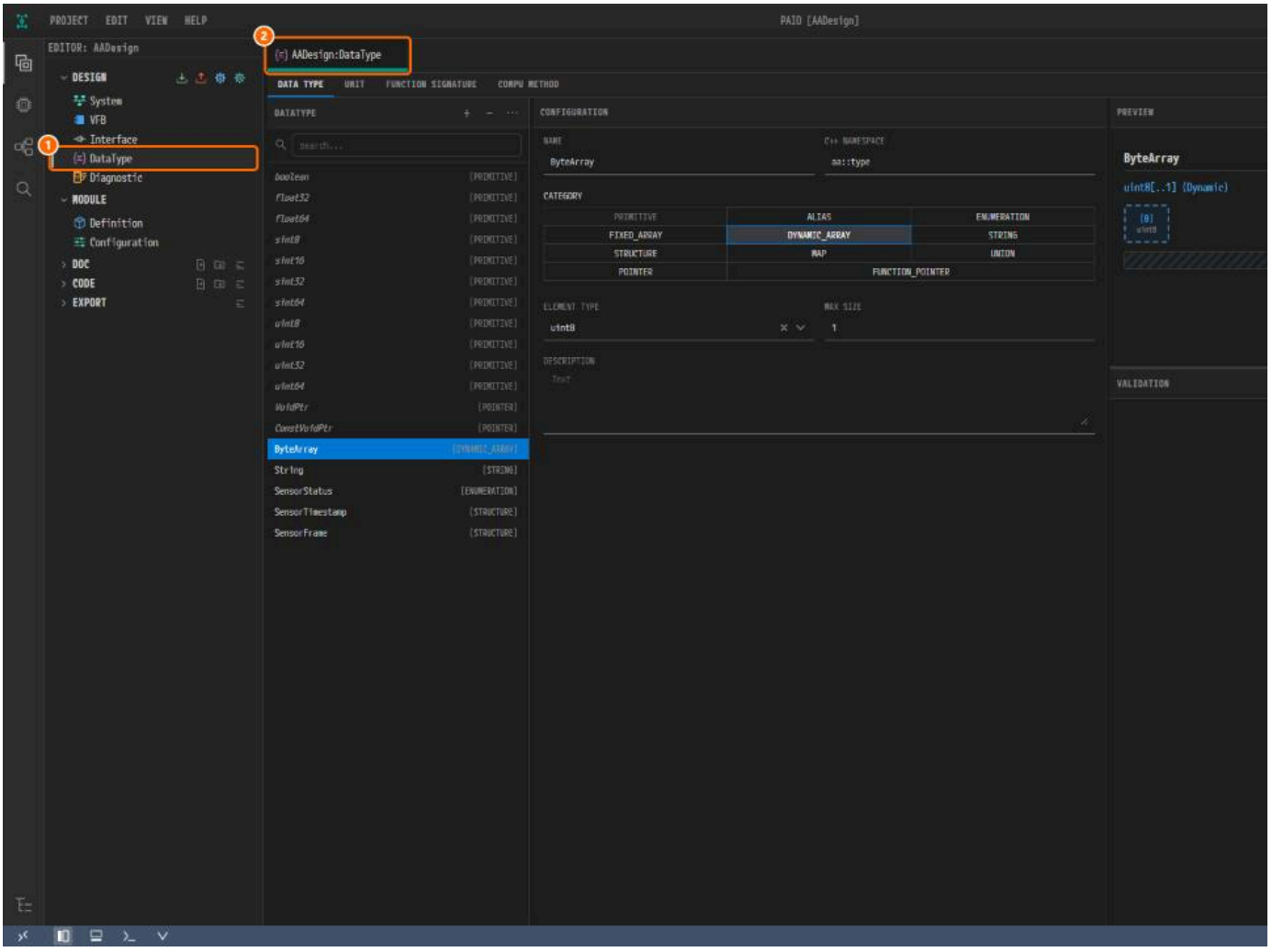
DataType 에디터

이 장에서 다루는 것

DataType 에디터의 사용법을 설명한다. 이 에디터는 애플리케이션 데이터 타입과 구현 데이터 타입, 물리 단위, 함수 시그니처, 변환 방법(Compu Method)을 정의한다. ARXML 임

- 대상 독자: 인터페이스나 SWC 가 쓸 데이터 타입을 정의하는 사용자
- 선행 조건: 창에 프로젝트가 열려 있는 것

왼쪽 **EDITOR** 뷰의 **DESIGN** 아래에서 **DataType** 을 누르면 탭 `<프로젝트>:DataType` 이 열린다. 데이터 타입, 단위, 함수 시그니처, Compu Method 는 모두 DataType 모델 하

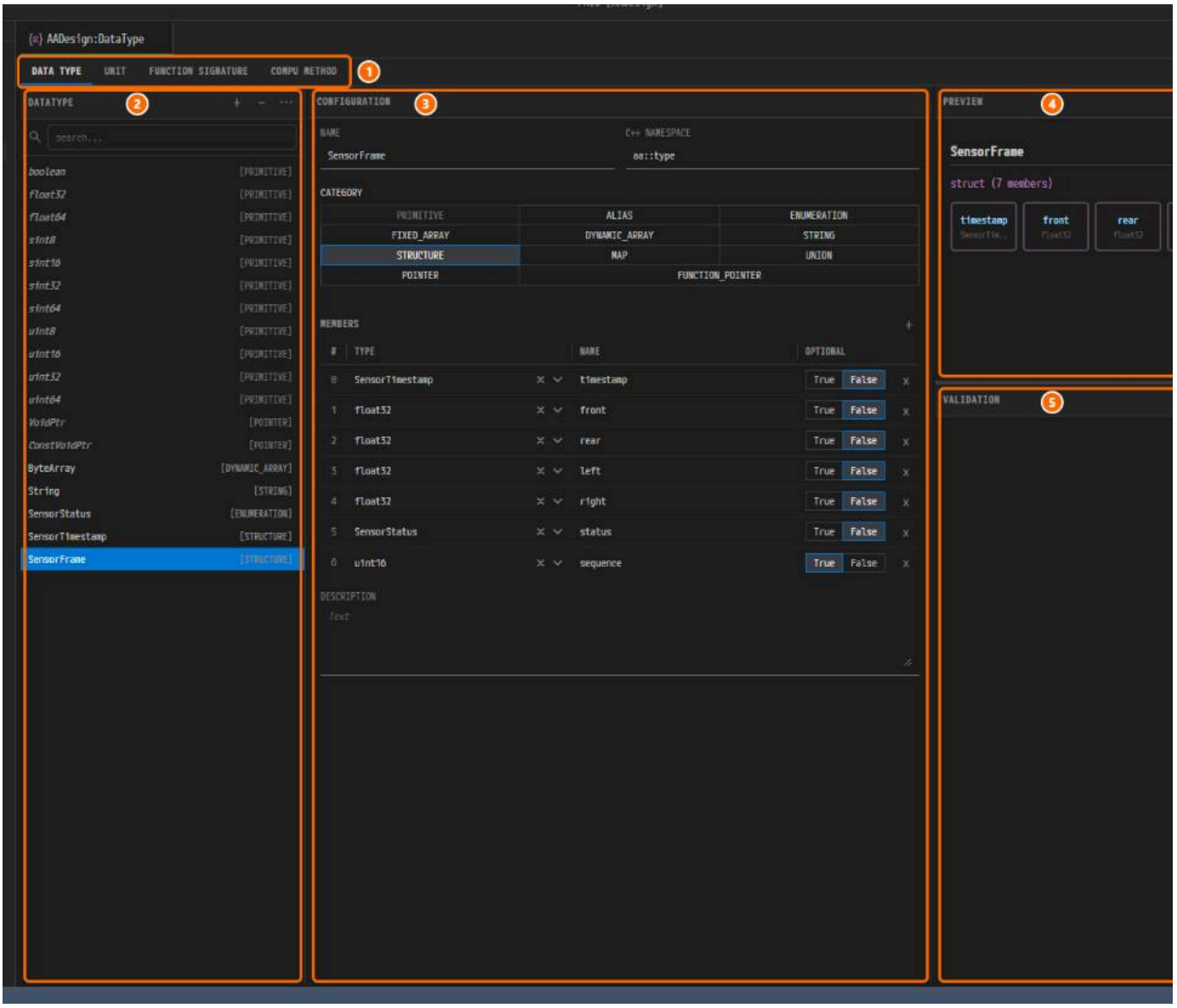


*① **DESIGN** 아래의 **DataType**. ② 그것을 눌러 열린 탭 `AADesign:DataType`.*

10.1 화면 구성

에디터 맨 위에 탭 버튼 네 개, **DATA TYPE**, **UNIT**, **FUNCTION SIGNATURE**(10.12), **COMPU METHOD**(10.13)가 있다. 탭마다 목록과 상세 편집 폼이 나온다.

DataType 탭 배치

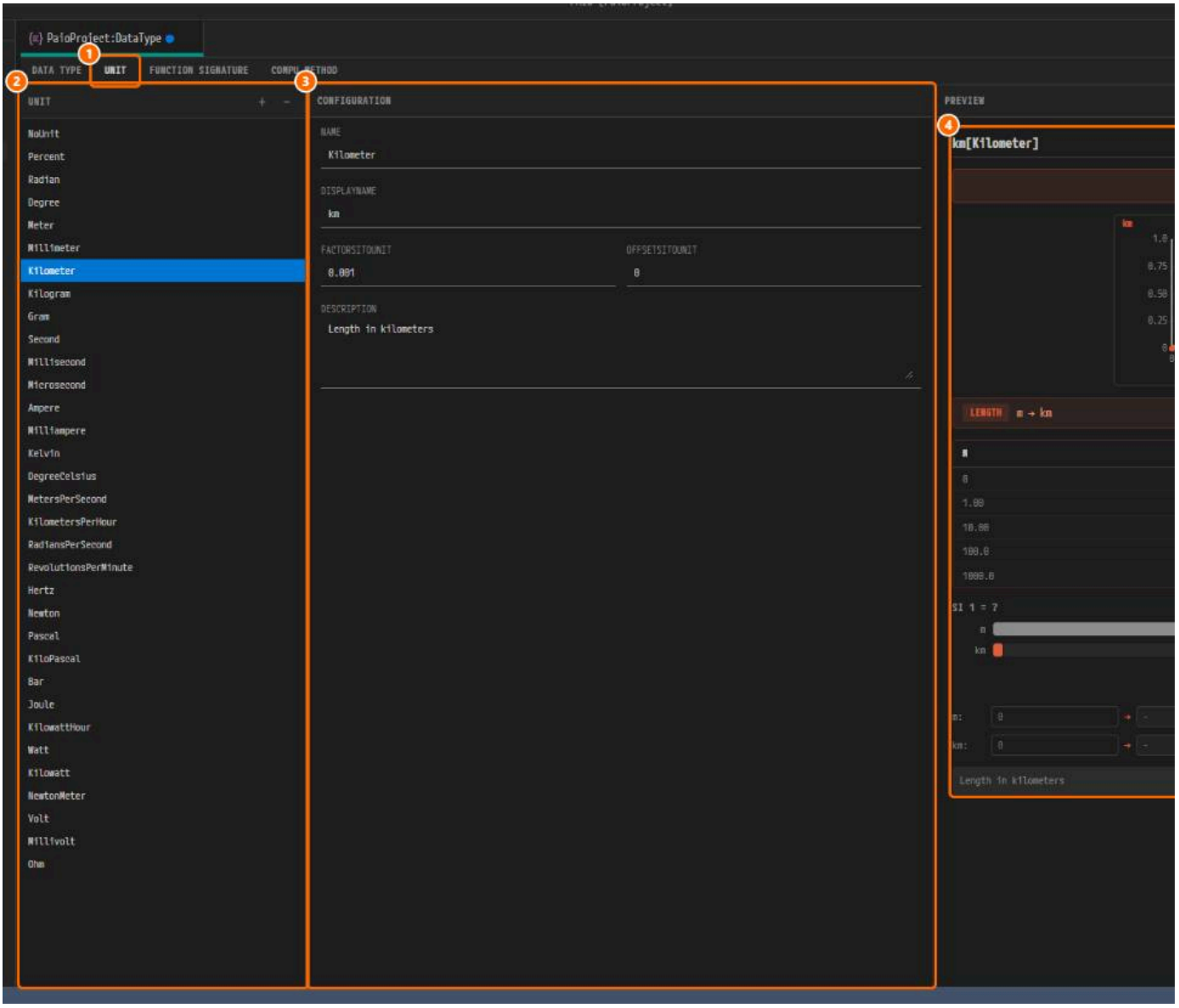


① 탭 버튼 네 개. ② DATATYPE 목록. ③ CATEGORY 세그먼트와 MEMBERS 표가 있는 CONFIGURATION. ④ PREVIEW. ⑤ VALIDATION.

패널 사이 경계를 드래그해 크기를 바꿀 수 있다.

Unit 탭 배치

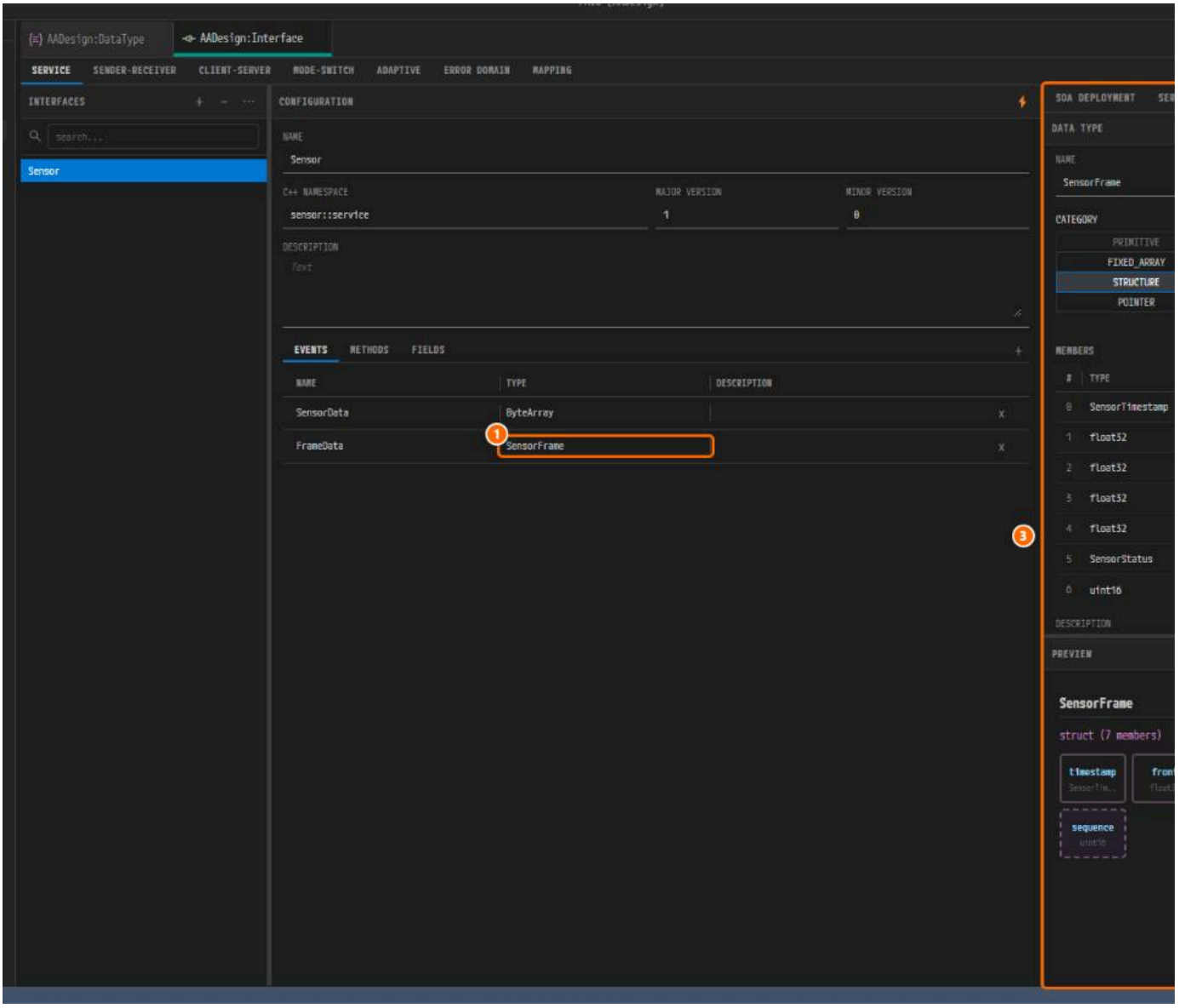
왼쪽에 단위 목록이, 오른쪽에 CONFIGURATION 폼과 PREVIEW 가 나란히 놓인다. Unit 탭에는 검색, 필터, Validation 영역이 없다.



① UNIT 탭 버튼. ② UNIT 목록. ③ CONFIGURATION. ④ Preview.

Interface 에디터에 들어가는 축소 모드

09. Interface 에디터에도 같은 편집기가 축소 모드로 들어 있다. 이벤트의 타입을 누르면 **DATA TYPE** 하위 탭에 속성 폼과 Preview 만 나온다. 편집 결과는 같은 DataType 모델에



① 이벤트 FrameData 의 타입 SensorFrame. ② DATA TYPE 하위 탭. ③ 축소 편집기.

10.2 데이터 타입 카테고리 (11종)

타입은 다음 순서로 정의한다.

1. 목록의 + 를 누른다. Category 가 비어 있는 새 타입이 생긴다.
2. 세그먼트에서 Category 를 고른다. 폼에 그 카테고리의 항목이 나온다.
3. 항목을 채운다. BaseType, Element Type 같은 참조 항목은 검색형 선택 상자이다.



① PRIMITIVE 는 흐리게 표시되고 선택할 수 없다.

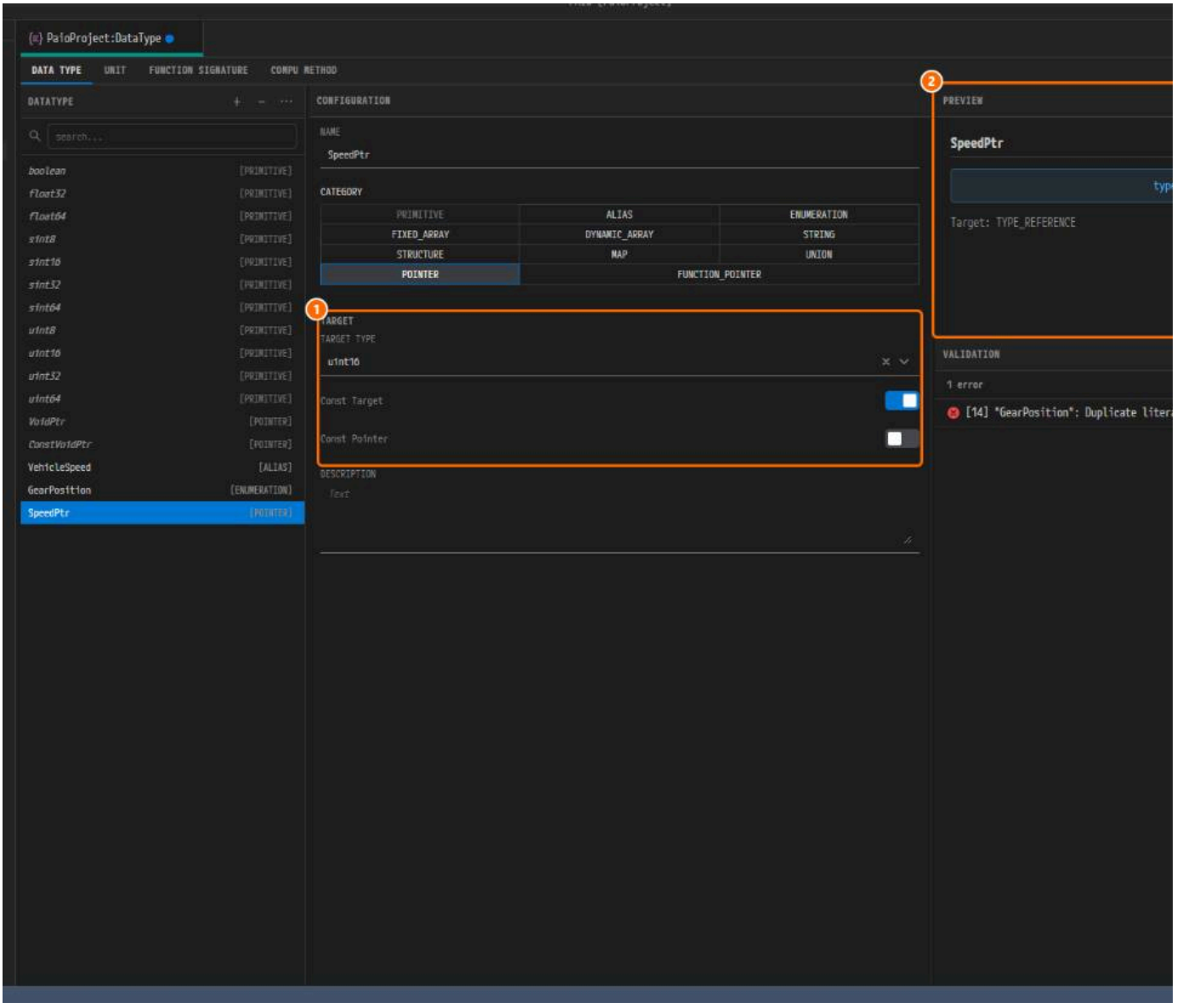
- 모든 카테고리에 `NAME`, `Category`, `Description` 이 있다.
- ALIAS 와 ENUMERATION 의 BaseType 은 **내장 프리미티브 11종만** 받는다.
- STRING 과 MAP 은 AP 전용, POINTER 는 CP 전용이다.



① ALIAS 의 BaseType 목록에는 내장 프리미티브 11종만 있다.

POINTER와 FUNCTION_POINTER

POINTER 는 가리키는 타입인 **Target Type** 과 두 토글 **Const Target**, **Const Pointer** 를 정한다. FUNCTION_POINTER 는 **Signature** 에 함수 시그니처 하나를 지정하므로, 시그니
다.

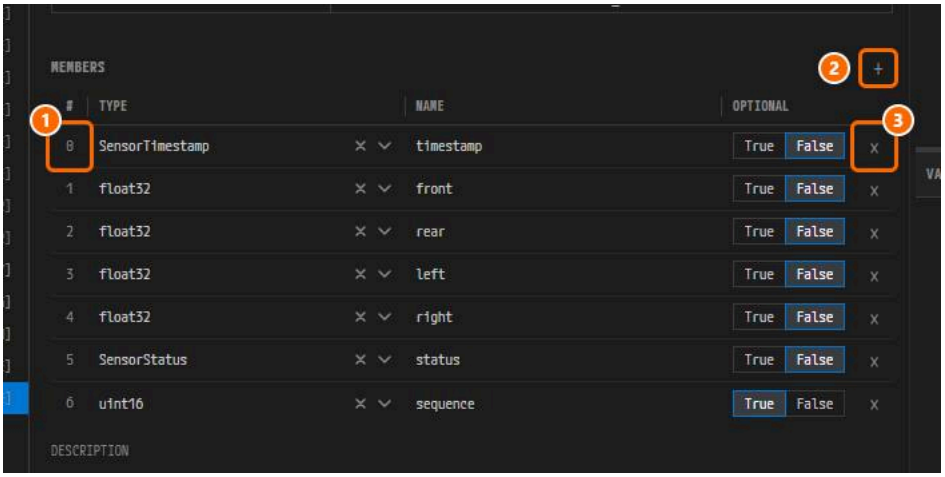


① Target Type `uint16`, Const Target 켜짐인 Target 구역. ② typedef 줄이 나온 Preview.

10.3 MEMBERS / ENUMERATION 하위 표

STRUCTURE, UNION, ENUMERATION 을 고르면 폼 아래에 하위 표가 붙는다. 앞의 둘은 `MEMBERS`, ENUMERATION 은 `ENUMERATION` 이다.

1. **+**(`Add Member` 또는 `Add Enumeration`)를 눌러 행을 추가한다.
2. 타입과 이름을 정한다. ENUMERATION 이면 이름과 값을 정한다.
3. `#` 칸을 끌어 순서를 바꾸고, **×** 로 행을 지운다.



*① `#` 드래그 손잡이. ② **+**(Add Member). ③ **×**(Remove Member).*

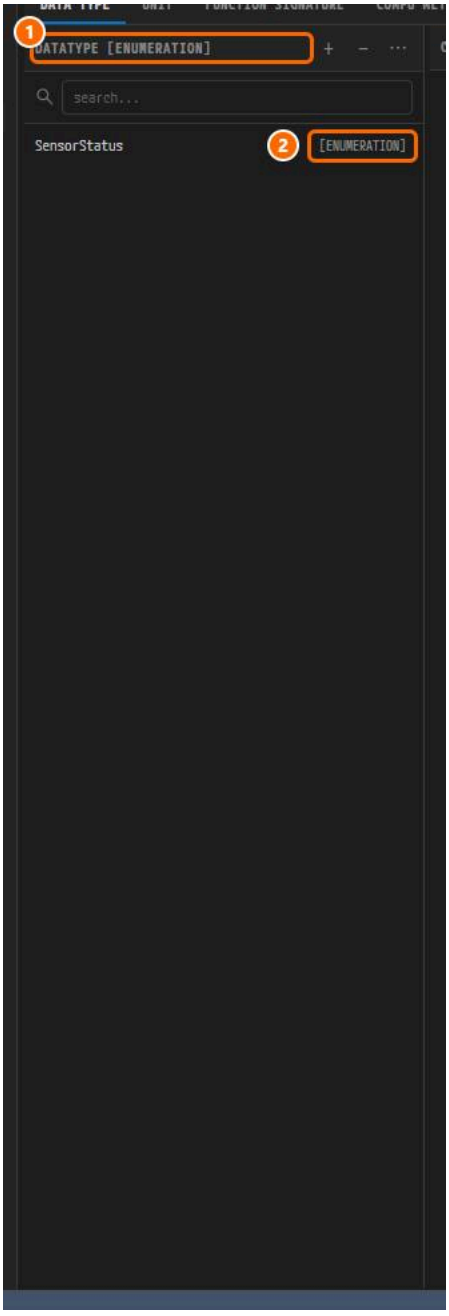
- AUTOSAR 규칙을 어긴 멤버·리터럴 이름은 저장되지 않고 이전 이름이 유지된다.
- ENUMERATION 값은 내장 BaseType 의 범위로 자동 보정된다. `uint8` 이면 300 이 255 가 된다.

10.3.1 포인터 멤버

STRUCTURE 나 UNION 의 멤버가 포인터일 수 있으며, 대개 임포트한 ARXML 파일에서 온다. `Type` 칸은 C 타입을 글자로 보여 주고, 대상은 행 아래의 **Target Type** 과 **Const Tar**

10.4 목록 조작 (DataType 탭)

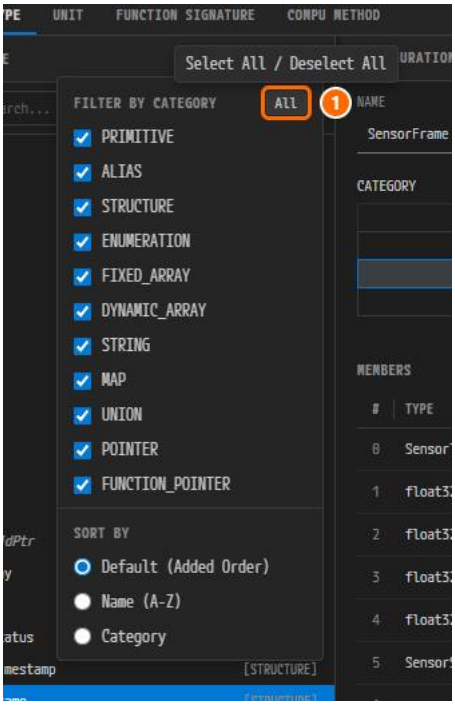
- **+** 로 타입을 만들고, **−** 로 선택한 타입을 확인을 거쳐 지운다. `Ctrl`+클릭으로 여러 개를 고르고, `Delete` 키도 **−** 와 같이 동작한다.
- 타입을 지우면 다른 타입에서 그 타입을 가리키던 참조가 비워진다.
- 내장 타입은 지울 수 없다.
- 검색 상자로 이름을 거른다.



① ENUMERATION 만 선택해 제목이 `DATATYPE [ENUMERATION]` 이 된 상태. ② `[ENUMERATION]` 배지.

⋯ 메뉴

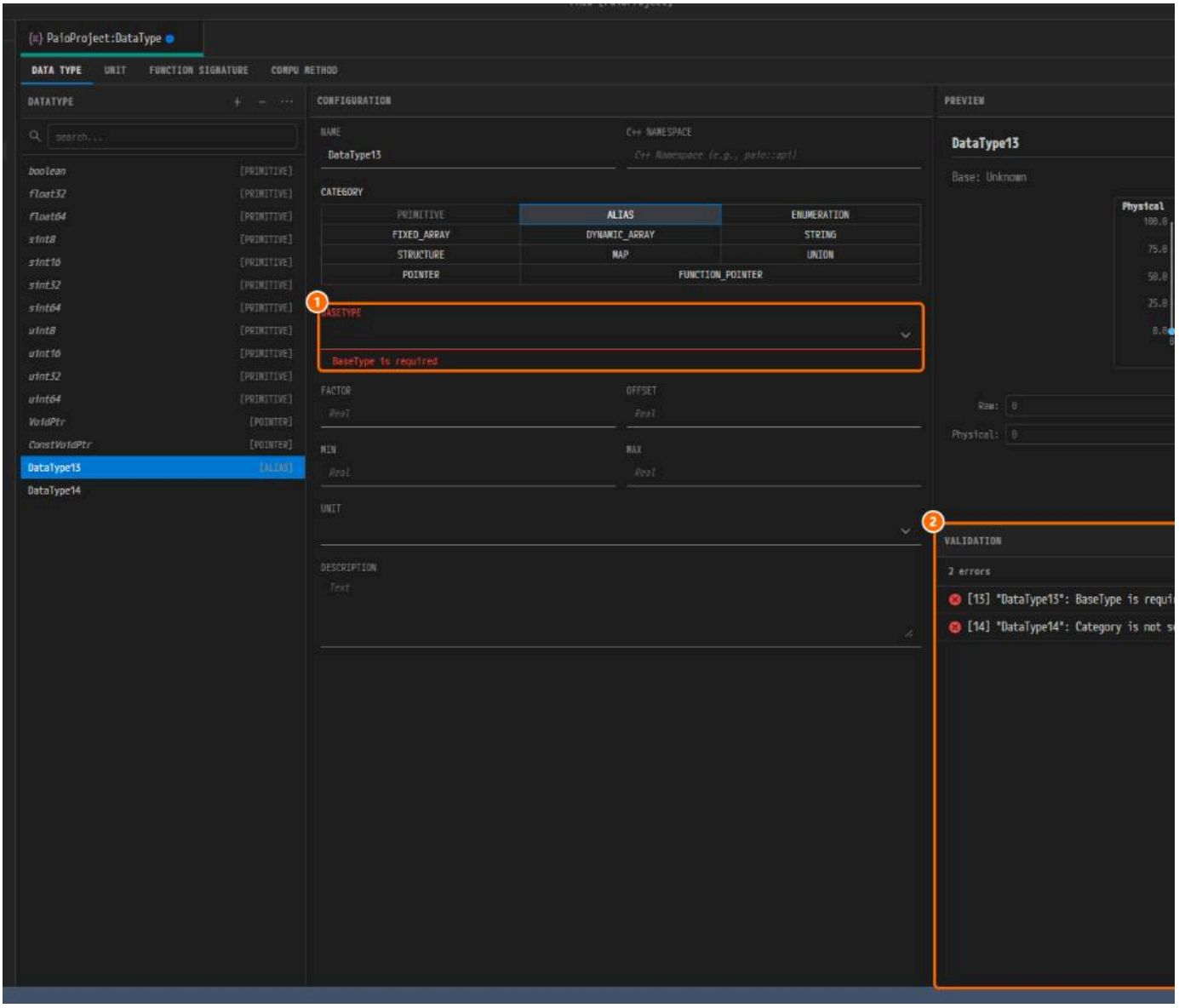
⋯ 를 누르면 카테고리로 목록을 거르고 정렬하는 팝오버가 열린다. 정렬은 화면 표시 순서만 바꾸고 파일은 바꾸지 않는다.



*① **All** 버튼. 그 아래에 FILTER BY CATEGORY 와 SORT BY 항목이 있다.*

10.5 Validation 패널

오른쪽 아래 `VALIDATION` 영역이 편집 결과를 실시간으로 검사한다. 항목을 누르면 해당 타입이 목록에서 선택된다.



① `BaseType is required` 가 붙은 BASETYPE 필드. ② 요약 `2 errors` 가 나온 VALIDATION.

메시지는 `[인덱스] "타입이름": 내용` 형식이다. 예: `[13] "DataType13": BaseType is required`.

오류 (error)

이름 없음, 필수 참조 없음, 자기 참조, 순환 참조 등이 오류로 보고된다.

구조체 · 열거형 세부 오류

멤버·리터럴 이름이 비었거나 겹치는 경우, 리터럴 값이 BaseType 범위를 벗어난 경우도 보고된다.

경고 (warning)

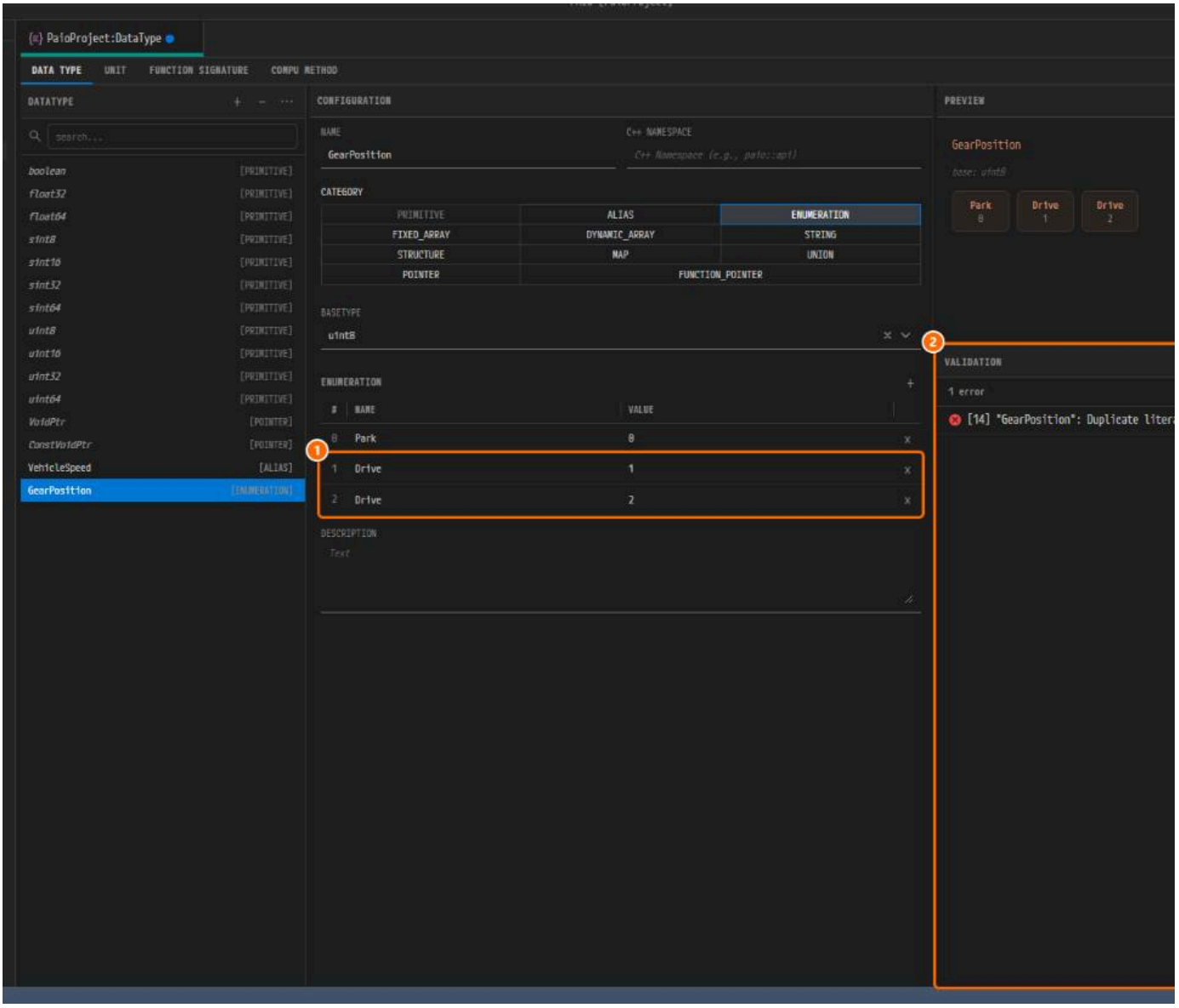
STRING 에 Max Size 가 없으면 경고가 나오고 기본값 255 가 적용된다.

입력 즉시 나타나는 필드 오류

ALIAS 의 Factor 가 0 이거나 Min > Max 인 경우 같은 일부 문제는 필드 바로 아래에도 표시된다.

MEMBERS / ENUMERATION 행의 문제

하위 표 행의 문제는 Validation 패널에 보고된다. 예: `[14] "GearPosition": Duplicate literal name: "drive"`.

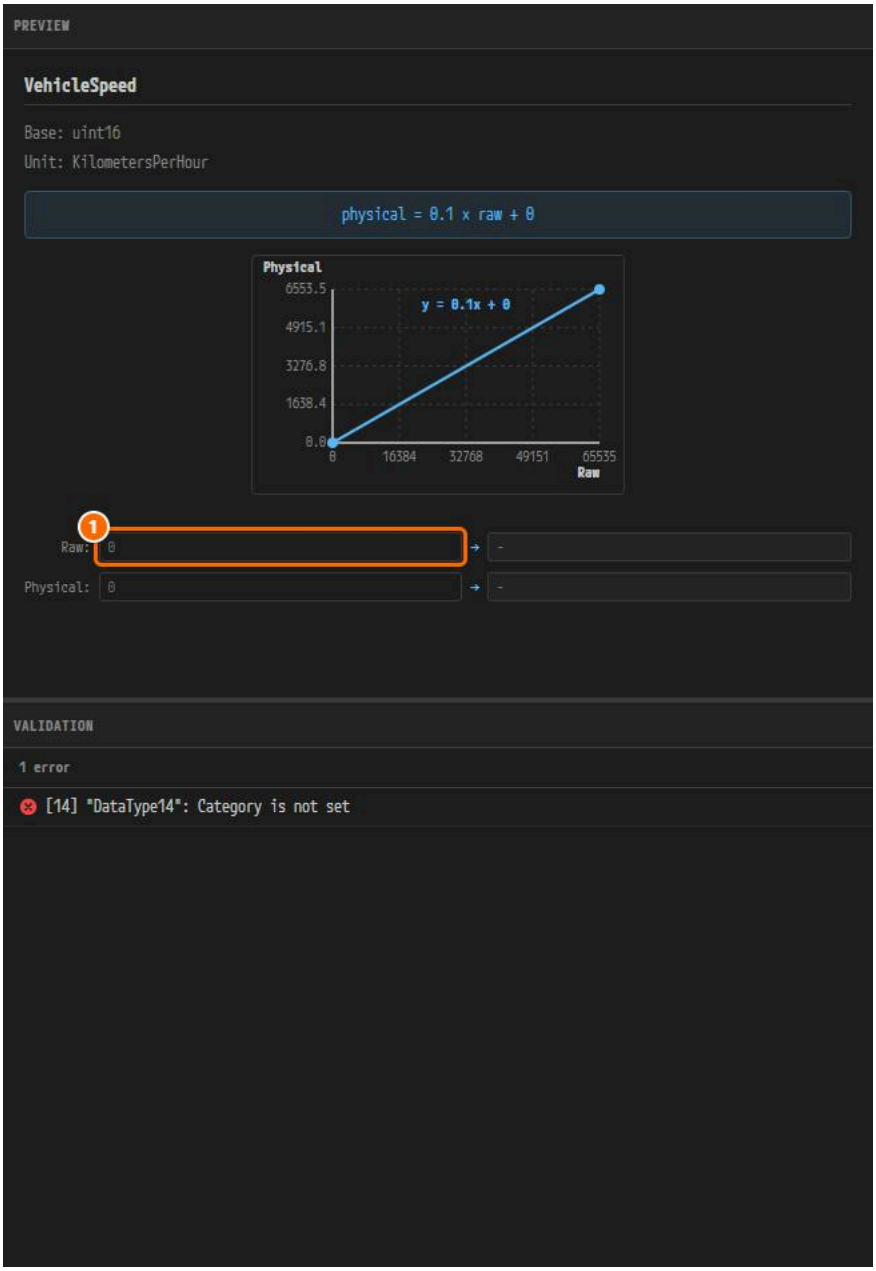


① Drive 두 행. ② VALIDATION 의 Duplicate literal name 항목.

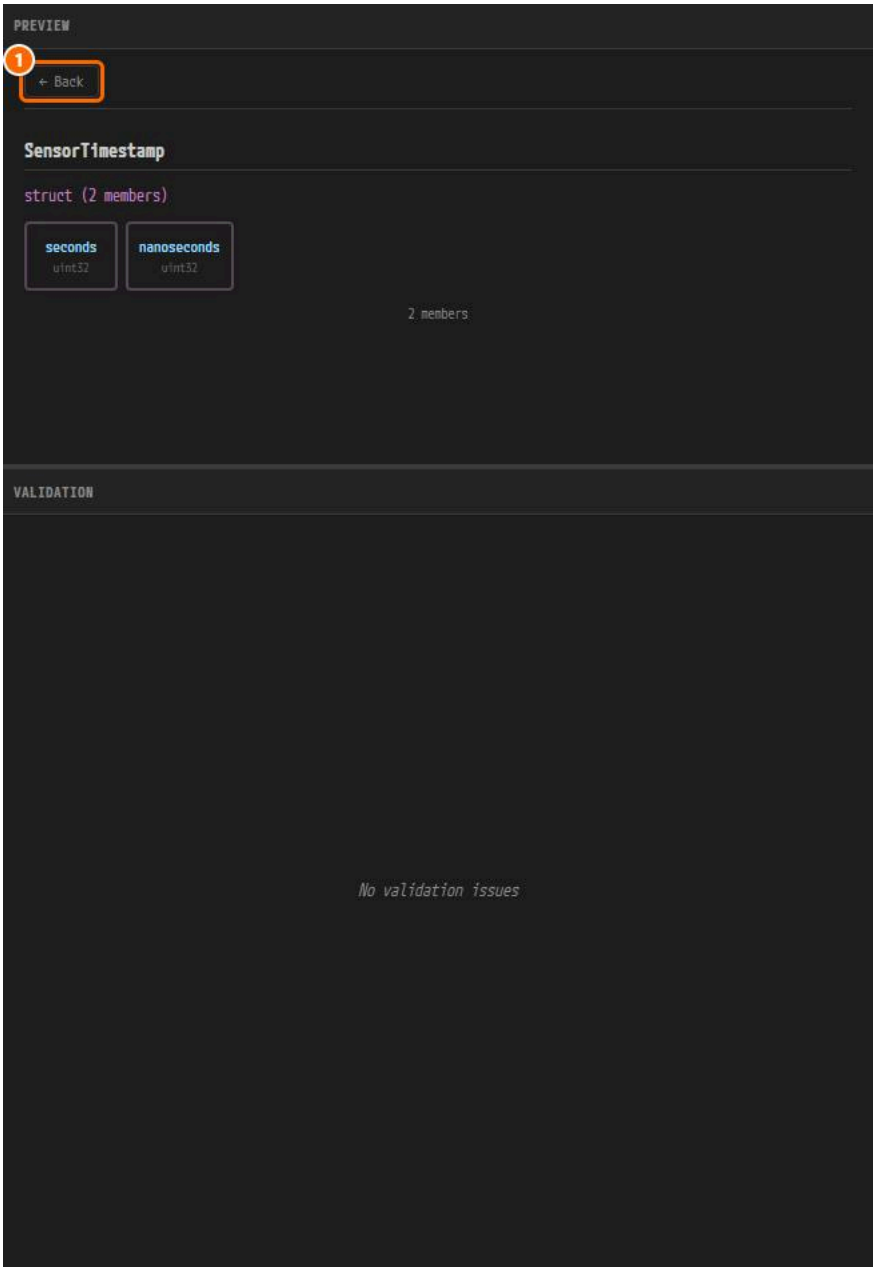
10.6 Preview 패널

Preview 는 선택한 타입의 구조를 카테고리에 맞게 그린다. 값 범위 막대, 구조체 그리드, 리터럴 칩, C `typedef` 등이다.

- ALIAS Preview 에는 Raw 와 Physical 을 서로 바꿔 보는 **값 변환기**가 있다. 계산기일 뿐 모델을 바꾸지 않는다.
- 자체 구조를 가진 타입의 멤버 상자를 누르면 그 타입으로 들어간다. **← Back** 으로 돌아온다.



① 값 변환기의 Raw 입력란.



*① SensorFrame 의 timestamp 상자를 누른 뒤 SensorTimestamp 와 **← Back** 이 나온 상태.*

10.7 내장 타입 (프리미티브 11종과 플랫폼 포인터 2종)

목록은 언제나 기울임꼴의 내장 타입 13종으로 시작한다. 프리미티브 11종(`boolean`, `float32`, `float64`, `sint8` ~ `sint64`, `uint8` ~ `uint64`)과 포인터 타입 `VoidPtr`, `C`

- 이름과 속성을 바꾸거나 지울 수 없으며, 이 이름들은 예약어이다.
- 모델 파일에 저장되지 않는다. 손실이 아니라 정상이다.
- 모델 안에서는 AUTOSAR 표기(`uint8`)를 쓴다. `uint8_t` 같은 C++ 표기는 C++ ARXML 을 내보낼 때만 나타난다.

익스포트 시 만들어지는 구조

ARXML 로 익스포트하면 프리미티브 하나가 네 가지 AUTOSAR 표현으로 나가고, DATA-TYPE-MAPPING-SET 두 개가 붙는다.

10.8 Unit 탭

단위는 ALIAS 타입의 `Unit` 항목이 참조한다.

목록

+ 로 단위를 만들고, **−** 로 선택한 단위를 확인을 거쳐 지운다.

편집 항목

1. + 를 누르거나 단위를 고른다.
2. `NAME`, `DisplayName`(화면 표시 기호, 예: `km/h`), `FactorSiToUnit`, `OffsetSiToUnit` 을 입력한다.

단위 이름을 바꾸면 그 단위를 쓰는 데이터 타입도 갱신되고, 단위를 지우면 그 `Unit` 값이 비워진다.

Preview

Preview 는 변환식, 그래프, 변환 표, 비교 막대, 값 변환기를 보여 준다. `LENGTH` 같은 도메인 배지는 Factor 와 Offset 이 알려진 변환과 일치할 때만 나온다.

기본 단위 33종

프로젝트는 `NoUnit` 부터 `Ohm` 까지 자주 쓰는 단위 33개로 시작한다. 기본값 그대로인 단위는 파일에 저장되지 않으므로, 파일의 단위 목록이 짧아도 정상이다.

10.9 저장 · 실행 취소

- `Ctrl+S` 한 번으로 네 탭이 모두 같은 DataType 모델에 저장된다.
- `Ctrl+Z` 로 실행 취소, `Ctrl+Y` 로 다시 실행한다. 실행 취소는 프로젝트 전체가 공유하는 타임라인 하나를 쓴다.

10.10 다른 화면과의 연동

- Interface 에디터에 이 편집기가 축소 모드로 들어 있다.
- 초기값(ValueSpec) 편집기는 여기서 정의한 타입으로 입력 형태를 만든다.
- 타입 이름을 바꾸면 Interface 모델, ECUC 설정, 함수 시그니처의 타입 참조가 함께 바뀐다. SWC 정의는 바뀌지 않는다.
- 프로젝트 전체 검사는 14. Validation에서 본다.

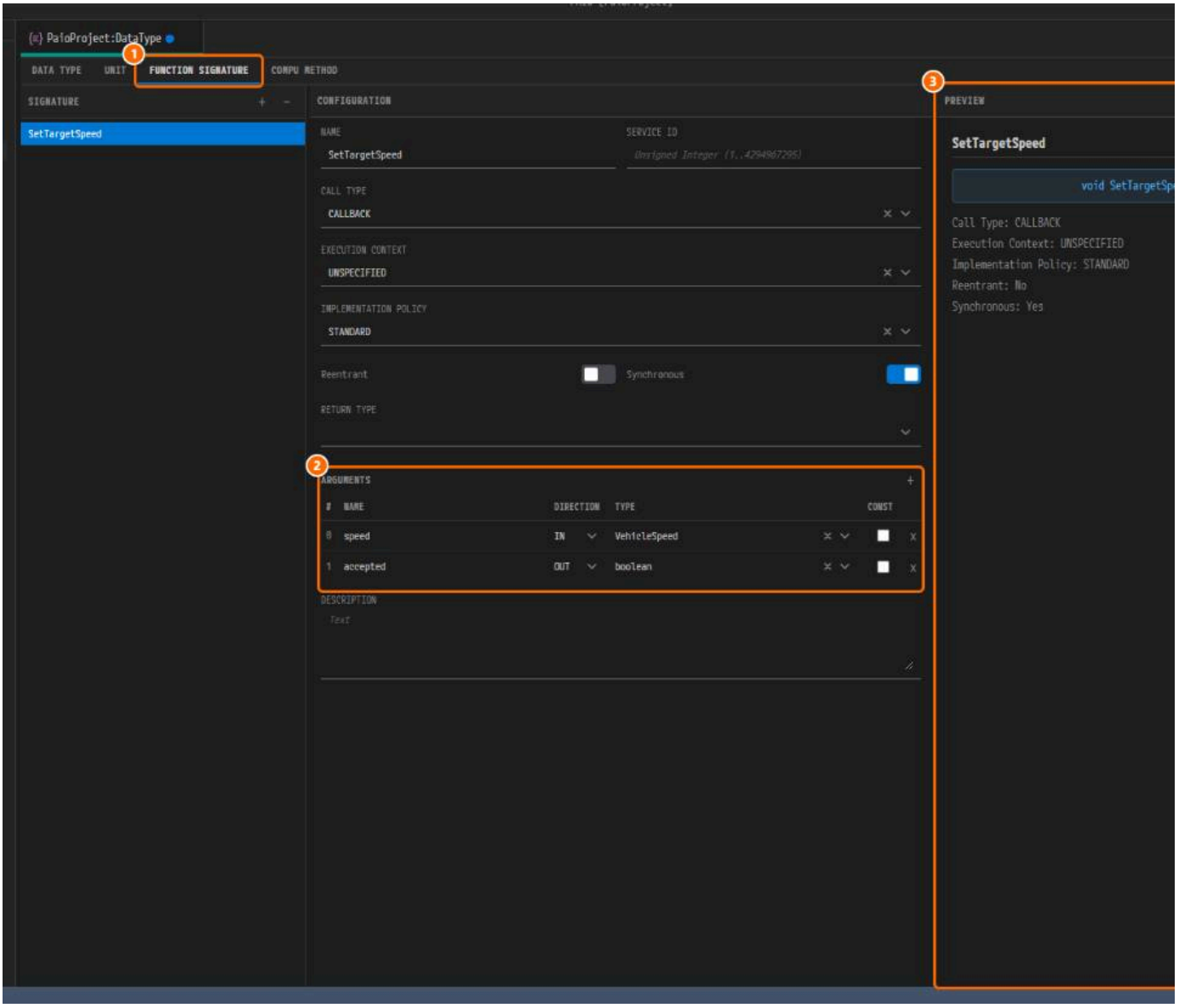
10.11 편집 규칙 요약

- 이름은 영문자로 시작하고, 이후는 영문자, 숫자, 밑줄이다.
- 비어 있거나 예약어인 타입 이름은 저장되지 않는다. 그 밖의 규칙 위반은 경고만 나오므로 익스포트 전에 고친다.
- 삭제는 모두 먼저 확인을 묻는다.

10.12 Function Signature 탭

FUNCTION_POINTER 타입이 지정하는 함수 프로토타입을 정의한다.

1. `SIGNATURE` 목록의 + 를 누른다. `FunctionSignature0` 항목이 생긴다.
2. `RETURN TYPE` 과 그 밖의 항목을 정한다.
3. `ARGUMENTS` 오른쪽의 + 로 인자를 추가하고, Direction(`IN`, `OUT`, `INOUT`)과 Type 을 정한다.

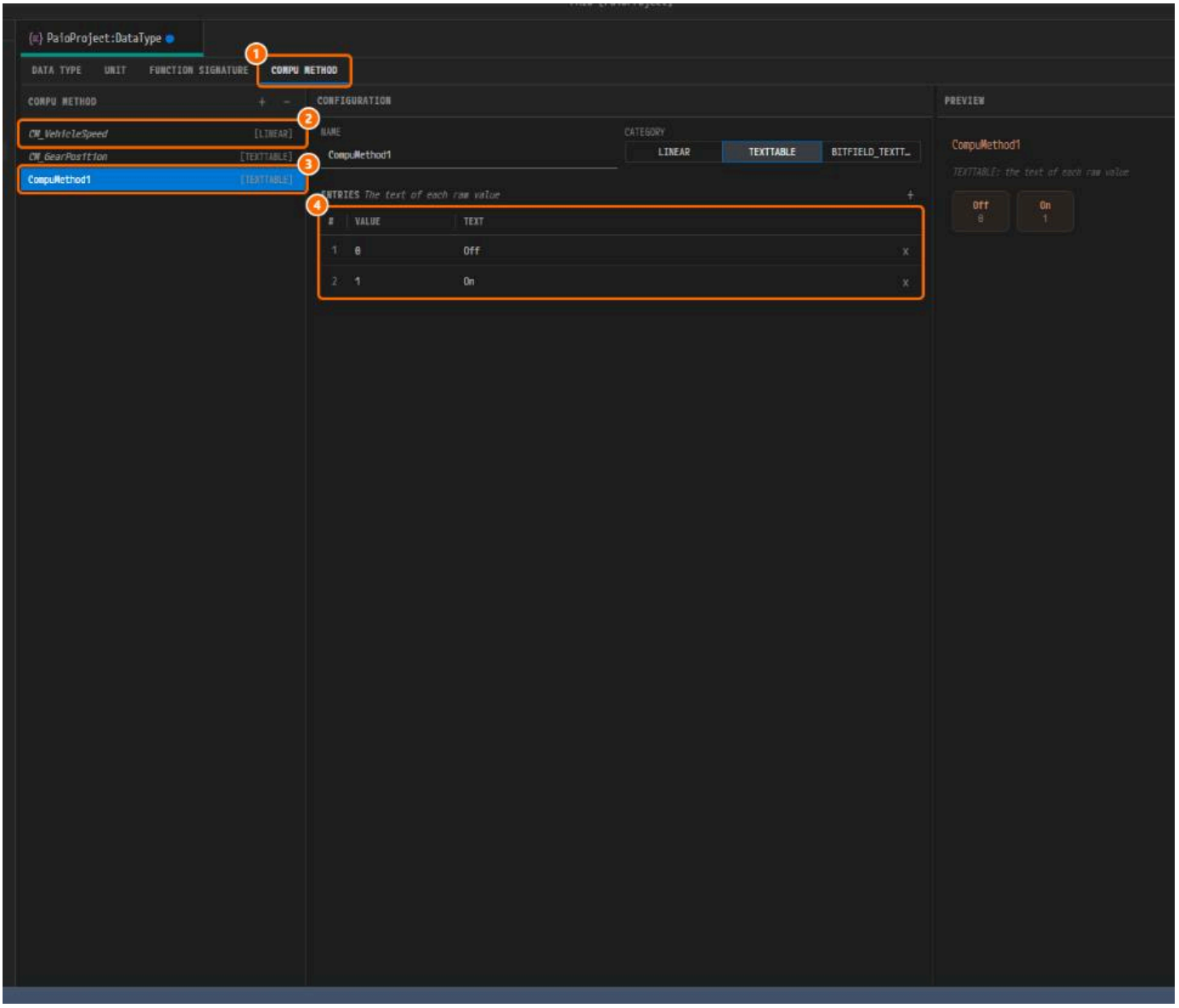


① SetTargetSpeed 를 선택한 상태. ② ARGUMENTS 표. ③ C 프로토타입이 나온 Preview.

10.13 Compu Method 탭

프로젝트의 변환 방법을 나열한다.

- **파생 항목**은 데이터 타입(Factor 가 있는 ALIAS, ENUMERATION)에서 만들어진다. 읽기 전용이며, **Open DataType** 으로 DataType 탭에서 고친다.
- **독립 항목**은 **+** 로 추가한 항목이다. `CATEGORY` 에서 `LINEAR`, `TEXTTABLE`, `BITFIELD_TEXTTABLE` 중 하나를 고른다.



① COMPU METHOD 탭. ② 파생 항목 CM_VehicleSpeed. ③ 독립 항목 CompuMethod1. ④ 그 ENTRIES 표.

Diagnostic 에디터

이 장에서 다루는 것

프로젝트의 진단 추출(DEXT)을 편집하는 Diagnostic 에디터를 설명한다. 에디터는 두 단계로 되어 있다.

- **SYSTEM COMMON**: 모든 노드가 공유하는 카탈로그이다. 세션, 보안 레벨, 접근 권한, 인증 역할, 데이터 식별자(DID), 루틴, 이벤트, DTC 와 그 레코드가 있다.
- **DIAGNOSTIC ECUs**: System 에디터의 ECU, HPC 노드가 나열된다. 노드마다 어떤 카탈로그 항목을 지원하는지, 어떤 UDS 서비스를 제공하는지, 어떤 접근 권한으로 제공하는

노드 자체는 06. System 에디터에서 만든다. 노드의 진단 주소 지정(물리 주소와 기능 주소, 연결, DoIP)은 그곳 노드의 **DIAGNOSTIC** 탭(6.7.6)에서 편집하며, 이 에디터에서는 편집

22.1 여는 방법과 저장 위치

1. 왼쪽 **EDITOR** 뷰의 **DESIGN** 아래에서 **Diagnostic** 을 누른다. 탭 `<project>:Diagnostic` 이 열린다.
2. 편집한 뒤 `Ctrl+S` 로 저장한다.

에디터는 `design/DIAGNOSTIC.io.model` 을 읽고 쓰며, 이 파일은 처음 저장할 때 만들어진다. 파일을 읽을 수 없으면, 고치거나 지운 뒤 에디터를 다시 열 때까지 편집이 막힌다.

22.2 화면 구성

에디터 왼쪽에는 트리가, 오른쪽에는 편집 영역이 있다.

- **SYSTEM COMMON** 에는 **SERVICE**(탭 **ACCESS**, **DATA IDENTIFIER**, **ROUTINE**)와 **EVENT** 두 행이 있다.
- **DIAGNOSTIC ECUs** 에는 ECU, HPC 마다 한 행이 있고 **ECU** 또는 **HPC** 배지가 붙는다. 노드에 진단 명세가 있으면 배지가 강조된다.

System 에디터에서 노드를 지우거나 이름을 바꾼 명세는 `<name> (missing)` 으로 목록에 남는다.

22.3 노드의 진단 활성화

노드는 활성화하기 전까지 진단 명세가 없다. 그동안 노드의 탭은 읽기 전용이다.

1. **DIAGNOSTIC ECUs** 아래에서 노드를 선택한다. `Diagnostics are not enabled for <node>.` 배너가 나온다.
2. **ENABLE DIAGNOSTICS** 를 누른다. 배너가 사라지고 노드의 배지가 강조된다.

새로 활성화한 노드는 카탈로그의 모든 세션과 보안 레벨을 지원하고, 서비스, DID, 루틴, 이벤트는 하나도 지원하지 않는다.

22.4 표 다루기

에디터의 목록 대부분은 같은 방식으로 다루는 행 표이다.

- 구역 머리글의 **+** 는 행을 추가하거나 선택지 메뉴를 연다. 행을 누르면 필드가 펼쳐진다.
- 행 끝의 **×** 는 행을 지운다. 다른 항목이 아직 참조하는 항목은 **Delete with References** 가 열리며, 항목을 지우고 그 참조를 한 단계로 함께 없앤다.
- 이름은 필드에서 포커스가 빠질 때 검사한다. 빈 이름이나 AUTOSAR 이름 규칙을 어긴 이름은 거부된다. 이름을 바꾸면 모든 참조도 함께 바뀐다.

22.5 SYSTEM COMMON ▸ SERVICE

SERVICE 의 세 탭은 공유 서비스 카탈로그를 만들고 편집하는 유일한 곳이다. 같은 이름의 노드 탭(22.7)은 여기서 고르기만 한다.

22.5.1 ACCESS

ACCESS 에는 **SESSIONS**, **SECURITY LEVELS**, **ACCESS PERMISSIONS**, **AUTHENTICATION (0x29)** 네 구역이 있다.

1. **SESSIONS** 의 **+** 를 누르고 ISO 14229-1 세션이나 **Custom session** 을 고른다.
2. **SECURITY LEVELS** 의 **+** 로 보안 레벨을 추가한다.
3. 접근 권한을 추가하고, 그 권한이 허용하는 세션과 보안 레벨, **AUTHENTICATION** 필요 여부를 고른다.

22.5.2 DATA IDENTIFIER

표는 프로젝트의 DID 를 나열한다.

1. **+** 를 누르고 **Custom DID** 나 ISO 14229-1 Table C.1 의 DID 를 고른다.

2. DID 를 펼쳐 **NAME**, **ID** 와 나머지 필드를 정한다.
3. **PARAMETERS** 표에서 **+** 로 파라미터를 추가한다. **Insert DID** 는 다른 DID 의 파라미터를 뒤에 붙인다. **#** 칸을 끌면 행 순서가 바뀐다.

COMPU METHOD 는 DataType 에디터의 CompuMethod 를 고른다.

22.5.3 ROUTINE

1. **+** 를 누른다. 이름 `Routine1`, START 서브펑션을 가진 루틴이 추가된다.
2. 루틴을 펼쳐 **SUB-FUNCTIONS** 의 **START**, **STOP**, **RESULT** 를 켜거나 끈다.
3. 켜진 서브펑션마다 요청, 응답 파라미터 표를 채운다. DID 의 **PARAMETERS** 표와 같이 동작한다.

22.6 SYSTEM COMMON ▸ EVENT

EVENT 에서는 진단 이벤트, DTC, 그리고 이들이 쓰는 정의를 편집한다. 위쪽 하위 탭 막대에 **EVENT** 하위 탭과 정의 하위 탭들이 있다.

22.6.1 EVENT 하위 탭

하위 탭은 패널 셋으로 되어 있다. 왼쪽은 **EVENTS**, 가운데는 선택한 **EVENT**, 오른쪽은 그 **DTC** 이다.

1. **EVENTS** 의 **+** 를 누른다. `Event1` 이 추가된다.
2. 가운데에서 이벤트 필드를 정한다. **OPERATION CYCLE** 과 **DEBOUNCE** 는 쉐브론 목록에서 정의를 고르거나, **+** 로 새 정의를 만들어 연결한다.
3. 오른쪽 패널의 **DTC** 머리글에서 **+** 를 누른다. `0x500000` 부터 다음 빈 3바이트 값으로 DTC 가 만들어져 이벤트에 연결된다.

이벤트를 지우면 그 DTC 도 함께 지워진다.

22.6.2 정의 하위 탭

나머지 하위 탭(**OPERATION CYCLE**, **DEBOUNCE**, **AGING**, **FREEZE FRAME**, **EXTENDED DATA**, **MEMORY**, **DTC GROUP**)은 이벤트와 DTC 가 쓰는 정의를 나열한다. 모두 행 표이며

22.7 노드 탭

DIAGNOSTIC ECUs 아래에서 노드를 선택해 그 노드가 지원하는 항목을 편집한다. 노드 탭은 카탈로그 항목을 만들지 않으며, SYSTEM COMMON 에서 고르기만 한다.

22.7.1 OVERVIEW

노드의 프로토콜, 타이밍, 응답, DTC 상태, 보안 값을 정한다. 예를 들면 **DEFAULT ENDIANNESS**, **S3 SERVER TIMEOUT (S)**(비워 두면 5초)이다.

22.7.2 ACCESS

노드의 세션 제어(`0x10`), 보안 접근(`0x27`), 인증(`0x29`) 서비스를 담당한다.

1. 세션이나 보안 레벨의 **SUPPORTED** 를 체크한다. 그 서비스 인스턴스와 함께 노드에 추가된다.
2. 행을 펼쳐 **ACCESS PERMISSION**(**Unrestricted** 또는 카탈로그의 접근 권한)을 고른다.
3. 기능 주소 지정으로 부를 수 있는 인스턴스면 **FUNC** 를 체크한다.

노드의 진단 연결 중 기능 수신 PDU 가 있는 것이 없으면 **FUNC** 는 비활성이다(System 에디터의 **DIAGNOSTIC** 탭, 6.7.6).

22.7.3 SERVICES

노드의 ISO 14229-1 서비스를 SID 별로 묶어, 서브펑션마다 한 행으로 나열한다.

1. 노드가 제공하는 행의 **SUPPORTED** 를 체크한다.
2. **PERMISSION** 을 고르고, 필요하면 **FUNC** 를 체크한다.

필터 칸(`Filter SID / name`)에 입력하면 목록이 좁혀진다.

22.7.4 DATA IDENTIFIER

USED 를 체크하면 DID 가 제한 없는 읽기 접근으로 노드에 배정된다. 행을 펼쳐 읽기, 쓰기, IO 제어 접근을 **Unsupported**, **Unrestricted**, 또는 접근 권한으로 정한다.

22.7.5 ROUTINE

USED 를 체크하면 루틴에 있는 서브펑션 모두에 제한 없는 접근이 주어진다. 행을 펼쳐 서브펑션마다 권한을 고른다.

22.7.6 EVENT

USED 를 체크하면 이벤트가 노드에 적용되며, 이벤트에 DTC 가 있으면 DTC 와 그 레코드도 따라간다. 이벤트와 DTC 는 여기서 읽기 전용이며, 편집은 SYSTEM COMMON ▸ **EVENT**

22.8 저장 · 실행 취소 · 외부 변경

`Ctrl+S` 로 저장하고, `Ctrl+Z` / `Ctrl+Y` 로 실행 취소, 다시 실행한다. 파일이 디스크에서 바뀌면 에디터가 다시 읽되, 저장하지 않은 편집이 있으면 그 편집을 지킨다.

22.9 검증과 제한

14. Validation은 진단 모델도 검사한다. 진단 항목을 누르면 이 에디터가 해당 노드, 탭, 행으로 열린다. 아무것도 가리키지 않는 참조는 경고 색으로 표시된다.

22.10 다른 에디터와의 연동

- 노드 목록은 System 에디터에서 온다.
- VFB 에디터의 ECU Configuration 대화상자에는 **DIAGNOSTIC** 탭(8.12.3)이 있다. 그 안의 **Derive Dcm/Dem ECUC from DEXT on Apply** 스위치는 이 모델에서 Dcm, De
- DID, 루틴 파라미터의 CompuMethod 는 DataType 에디터에서 온다.

22.11 자주 겪는 상황

가장 흔한 경우는 노드에 진단 명세가 없어 노드의 탭이 읽기 전용인 경우이다. **ENABLE DIAGNOSTICS** 를 누른다(22.3).

ECUC

이 장에서 다루는 것

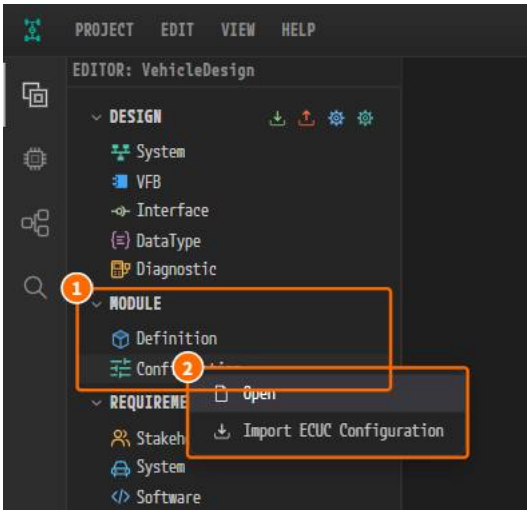
AUTOSAR Classic Platform ECU 의 BSW 모듈 설정(ECUC)을 다루는 두 에디터의 사용법을 설명한다. 이 장의 예시는 AUTOSAR CP 튜토리얼의 ECU `ECU` 이다.

- 대상 독자: ECU 의 BSW 모듈(Os, Rte, EcuM, BswM 등)을 설정하는 사용자
- 선행 조건: 프로젝트가 열려 있고, System 에디터에 ECU 노드가 있는 것

두 에디터는 왼쪽 **EDITOR** 뷰의 **MODULE** 아래에서 연다.

에디터	다루는 것	여는 곳
Module Definition	모듈에 어떤 파라미터가 있는가 (정의, `.def`)	**MODULE ▸ Definition**
Module Configuration	그 파라미터의 값 (설정, `.conf`)	**MODULE ▸ Configuration**

설정은 반드시 어떤 정의를 따른다. 대부분의 작업은 Module Configuration 에디터에서 한다. PRTE 모듈의 정의는 **PRTE Platform Import** 가 넣어 주므로, 정의를 직접 편집할 일



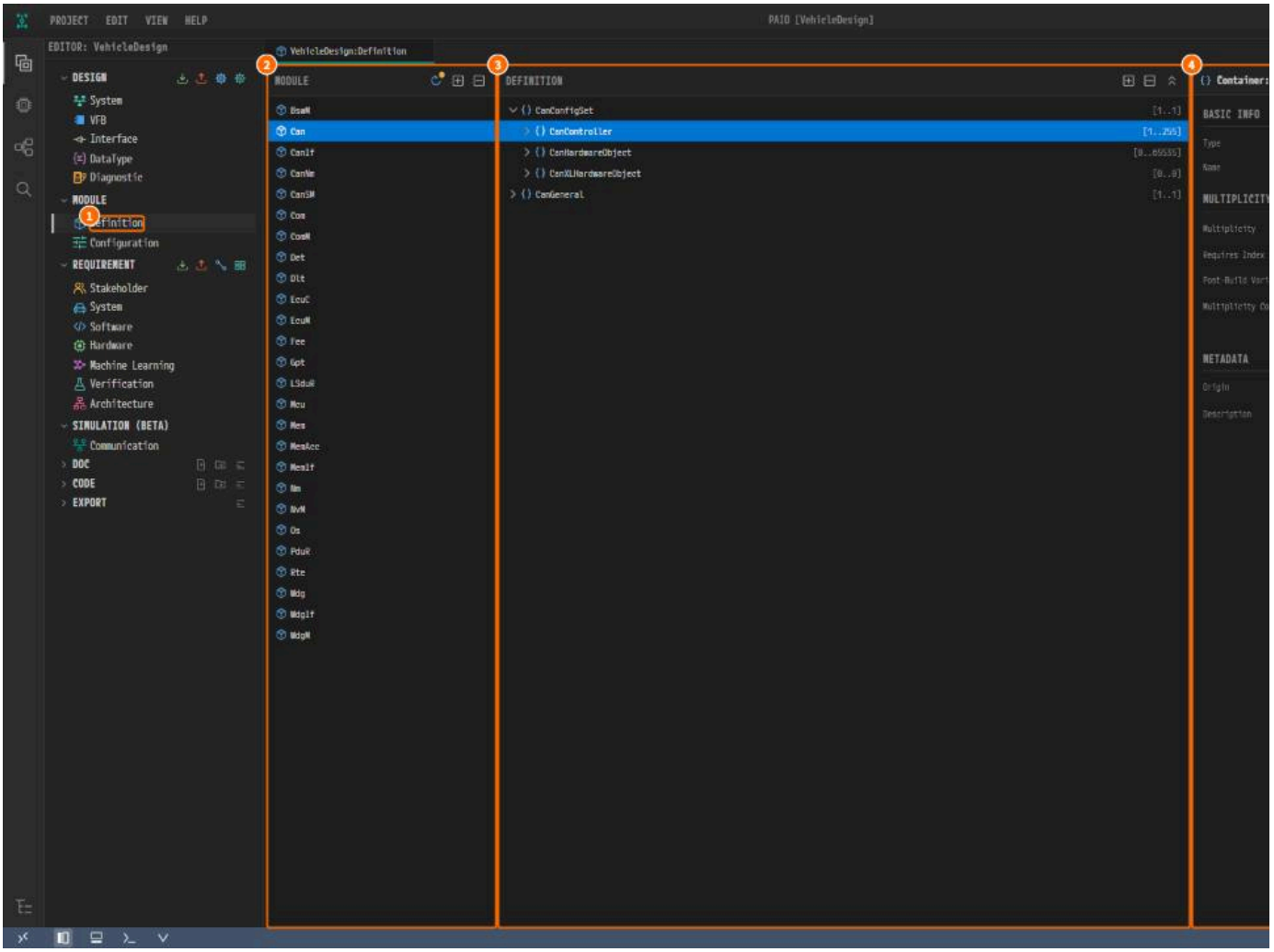
*① **EDITOR** 뷰의 **MODULE** 과 그 아래 **Definition**, **Configuration**. ② **Configuration** 을 우클릭하면 **Open** 과 **Import ECUC Configuration** 이 있다.*

12.1 Module Definition 에디터

모듈의 구조, 즉 컨테이너·파라미터·레퍼런스를 정의한다. 벤더 모듈을 직접 만들 때만 쓴다.

12.1.1 여는 방법과 화면 구성

1. **EDITOR** 뷰의 **MODULE** 아래에서 **Definition** 을 누른다. 탭 `<프로젝트>:Definition` 이 열린다.
2. 왼쪽 ② **MODULE** 패널에서 모듈을 고른다. 가운데 ③ **DEFINITION** 트리와 오른쪽 ④ 속성 패널이 채워진다.



*① **Definition**, ② `Can` 을 고른 **MODULE** 패널, ③ **DEFINITION** 트리, ④ 선택한 `CanController` 의 속성.*

12.1.2 MODULE 패널

- **+** 로 모듈을 만들고, **-** 로 고른 모듈을 지운다.
- 프로젝트의 PRTE 정의가 PAIO 에 들어 있는 것보다 오래되면 점이 붙은 새로 고침 아이콘(**Update to current PRTE VSMD**)이 나타난다. 누르고 열린 창에서 모듈을 고른 뒤 U
  다. 같은 아이콘이 Module Configuration 에디터의 **ECU** 목록에도 있다.
- 새 모듈은 `Ctrl+S` 로 저장해야 파일이 생긴다.

12.1.3 DEFINITION 트리

1. 컨테이너 행을 우클릭한다.
2. **Add Container**, **Add Parameter**, **Add Reference** 에서 만들 종류를 고른다.
3. 새 항목을 누르고 오른쪽 속성 패널에서 이름과 값을 정한다.

12.1.4 속성 패널

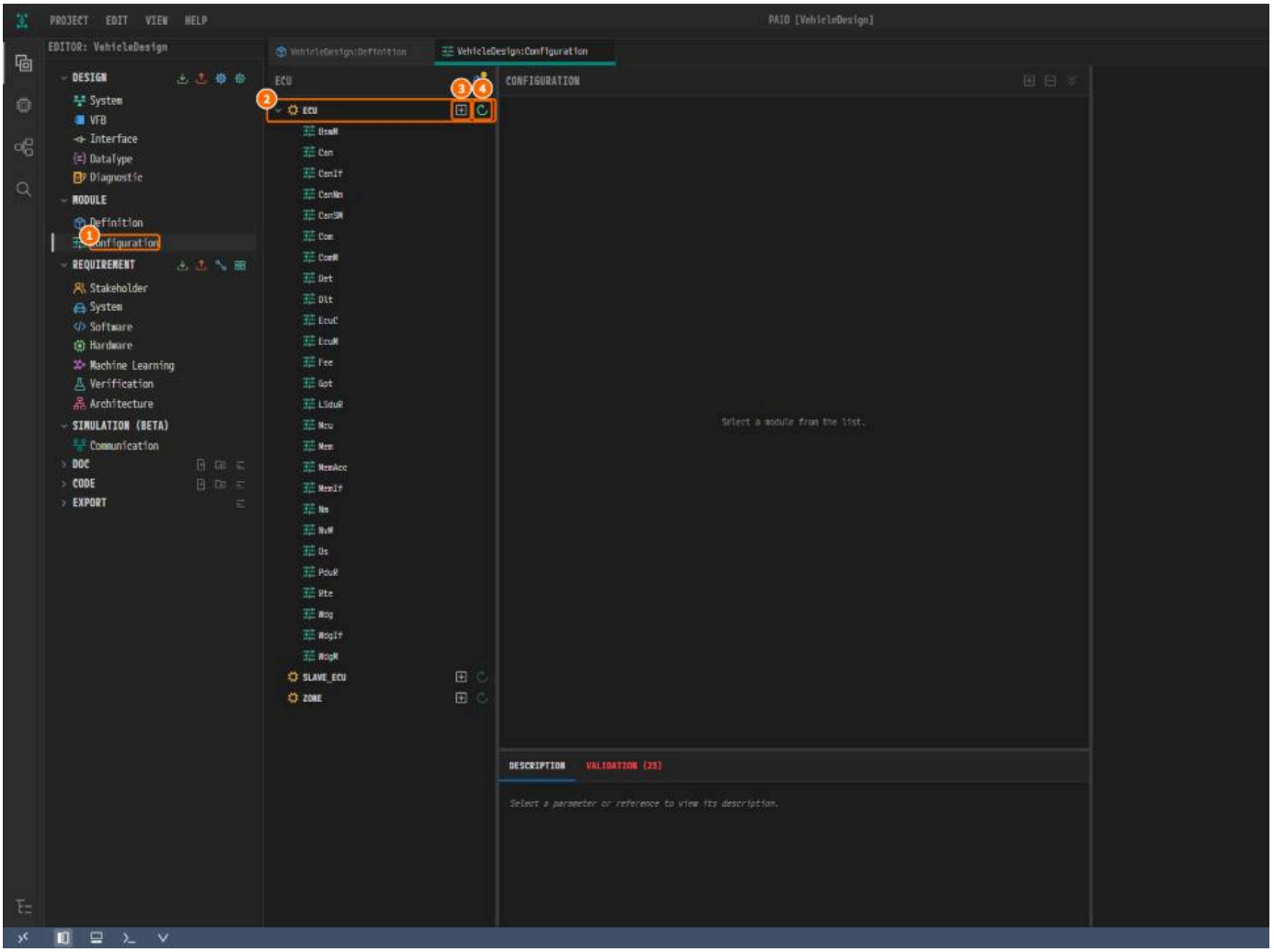
선택한 항목의 이름, 다중도(**Multiplicity**), 타입별 옵션(범위, 기본값, 리터럴, 레퍼런스 대상)을 편집한다.

12.2 Module Configuration 에디터

ECU 의 BSW 모듈 값을 넣는 에디터이다.

12.2.1 여는 방법과 화면 구성

1. **EDITOR** 뷰의 **MODULE** 아래에서 **Configuration** 을 누른다. 탭 `<프로젝트>:Configuration` 이 열린다.
2. 왼쪽 **ECU** 목록에 설계의 ECU 가 있고, 각 ECU 아래에 그 ECU 의 모듈 설정이 나열된다.



*① **Configuration**, ② ECU `ECU` 와 모듈 목록, ③ **Add Module**, ④ **PRTE Auto Configuration**.*

12.2.2 ECU 목록

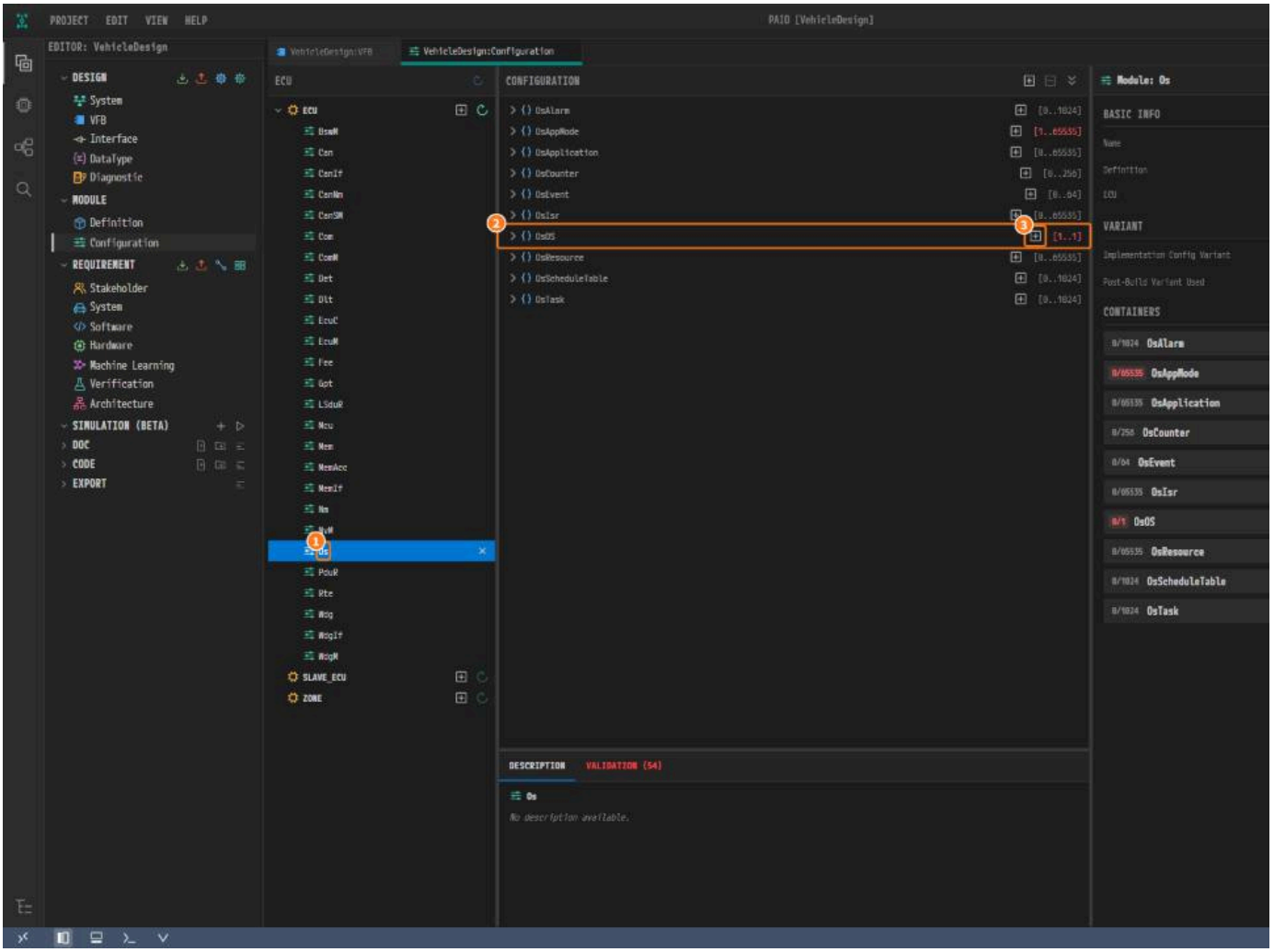
- ECU 행의 **+(Add Module)**를 누르면 모듈 목록이 나온다. 고르면 그 ECU 에 설정이 생긴다. 이름은 모듈 이름이고, 두 번째 설정은 `<모듈>_1` 이다.
- 설정 행의 **×** 로 설정을 지운다. 이름을 더블클릭하면 이름을 바꾼다.
- 프로젝트에 모듈 정의가 없으면 **Add Module** 이, 그 ECU 에 설정이 없으면 **PRTE Auto Configuration** 이 꺼져 있다(`Import PRTE modules first.`).
- 새 설정은 `Ctrl+S` 로 저장해야 파일이 생긴다.

12.2.3 CONFIGURATION 트리

트리는 정의를 따라 그려진다. 정의의 컨테이너가 그룹 행으로 나오고, 그 아래에 실제 인스턴스가 붙는다. 하한을 채우지 못한 그룹은 다중도(`[1..1]`)가 붉게 보인다.

컨테이너를 추가하는 방법은 다음과 같다. 예시는 `Os` 의 `OsOS` 이다.

1. 목록의 ECU 아래에서 ① **Os** 를 누른다.
2. 붉게 보이는 `OsOS` 그룹 행 ②의 ③ **+** 를 누른다. 인스턴스 `OsOS` 가 생긴다.

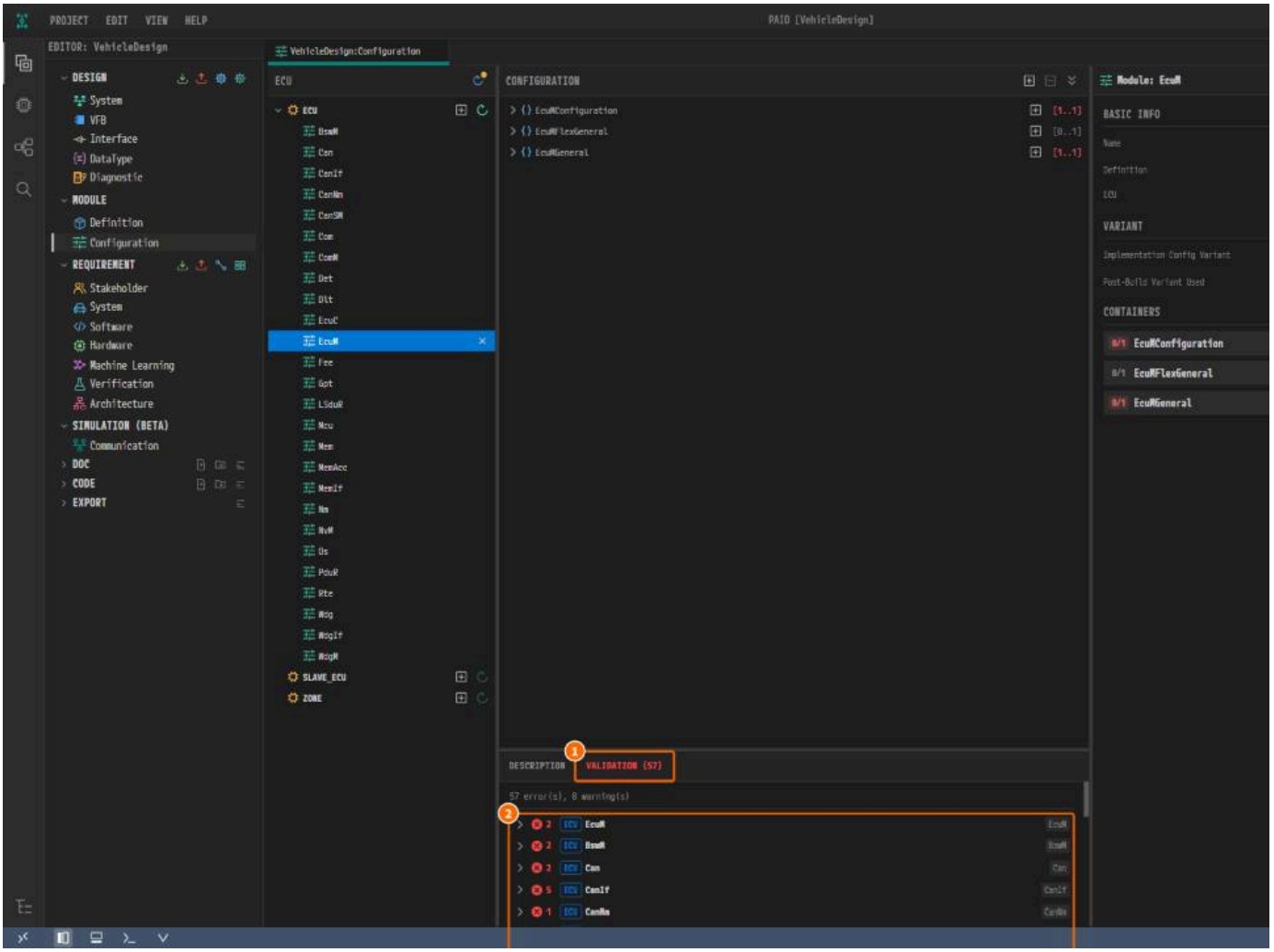


*① **Os**, ② `OsOS` 행, ③ **+**. `OsOS` 는 하한이 1 이라 다중도가 붉게 보인다.*

12.2.4 Description 탭과 Validation 탭

가운데 패널 아래에 두 탭이 있다.

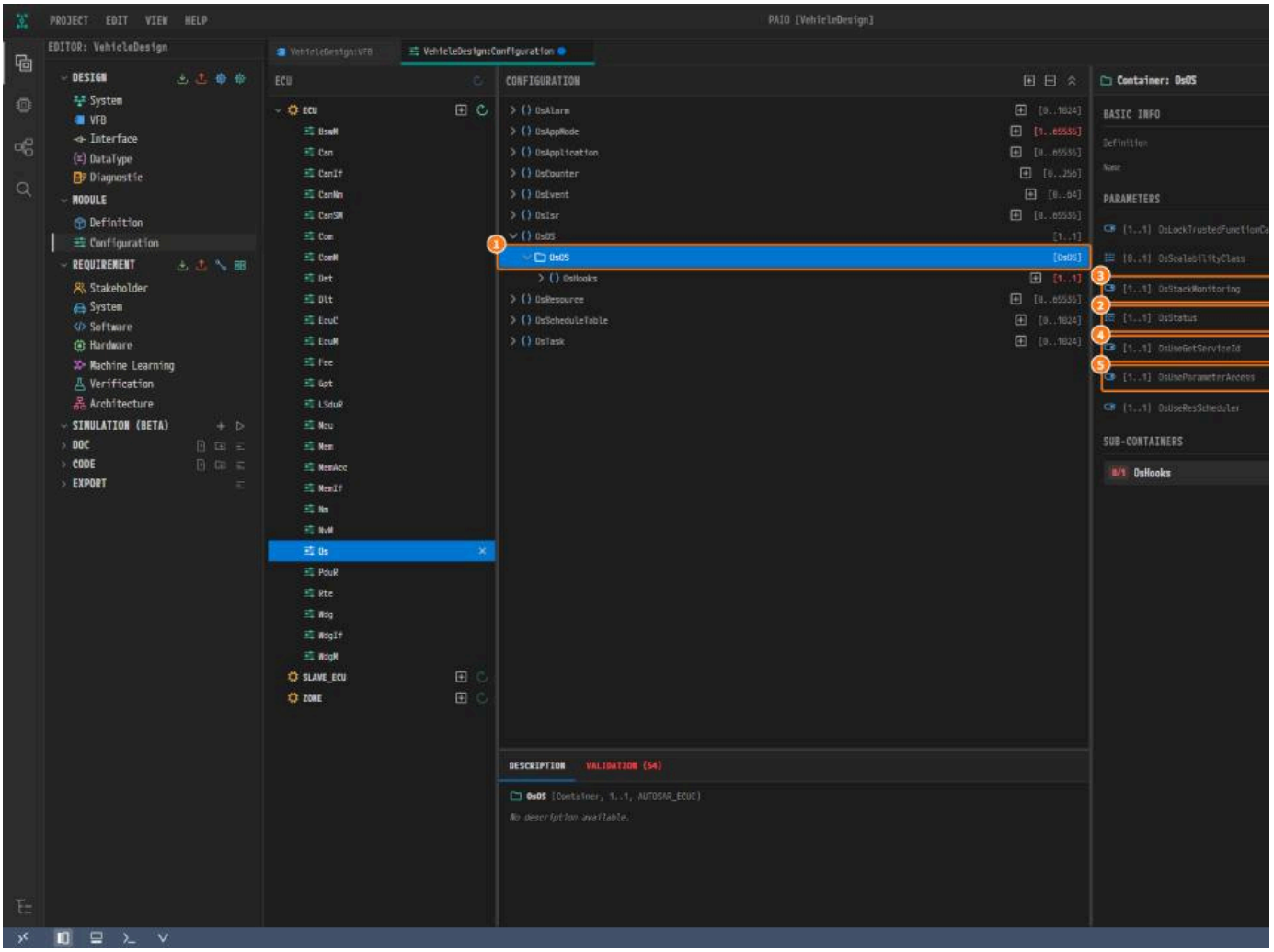
- **DESCRIPTION**: 방금 누른 파라미터의 정의 설명을 보여 준다.
- **VALIDATION**: 아직 값이 없는 필수 파라미터, 범위를 벗어난 값 같은 문제를 모아 보여 준다. 항목을 누르면 그 위치로 이동한다. 탭 이름의 숫자는 오류 개수이다.



① 탭 라벨 `VALIDATION (57)`. ② 오류와 경고가 모듈별로 묶여 있다.

12.2.5 값 편집 패널

1. 트리에서 인스턴스(예: `OsOS`)를 누른다.
2. 오른쪽 패널의 **PARAMETERS** 에서 값을 넣는다. 숫자·문자는 입력 칸, 열거형은 콤보, 불리언은 **TRUE** / **FALSE** 버튼이다.
3. `Ctrl+S` 로 저장한다.



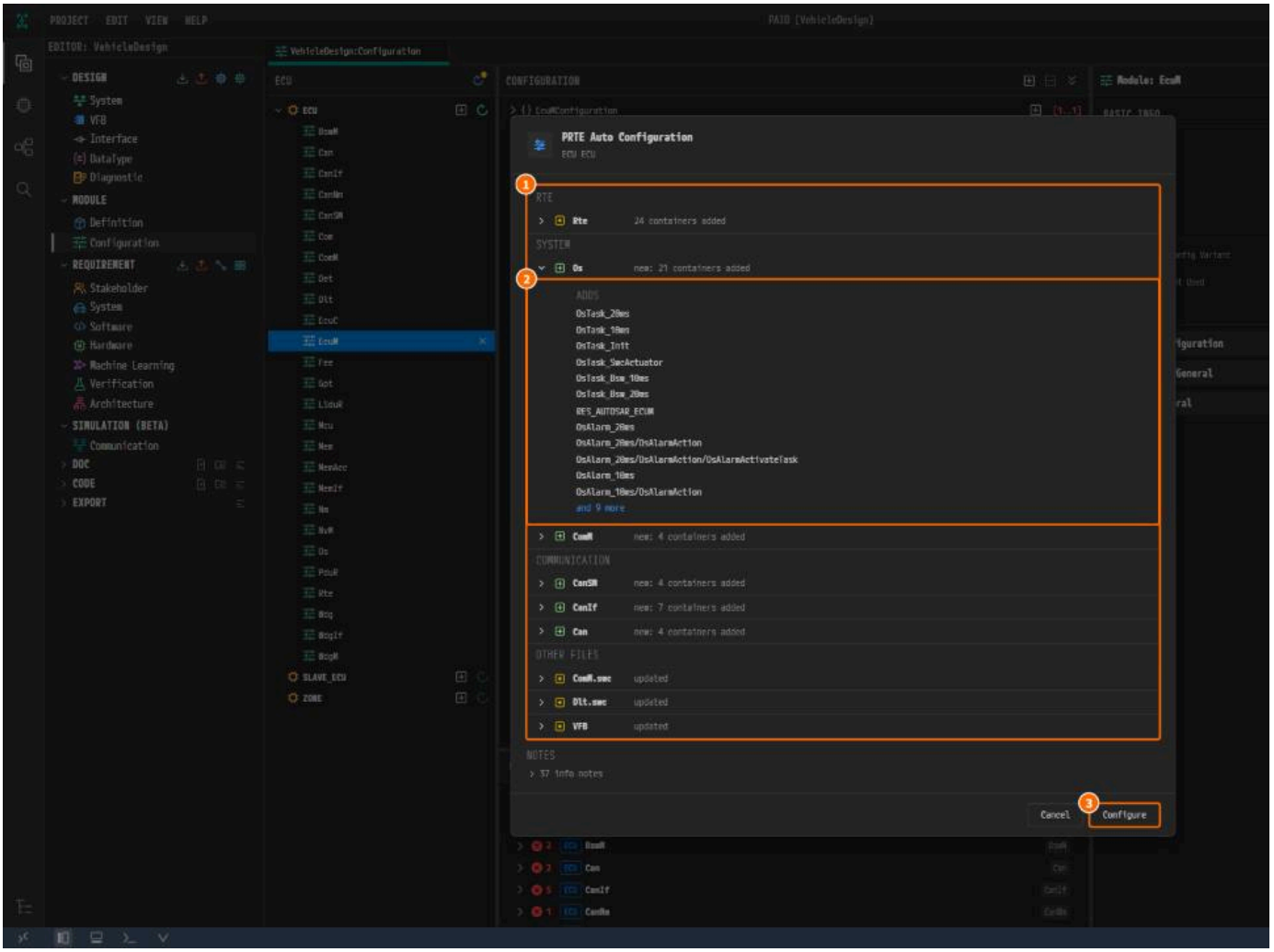
① `OsOS`, ②~⑤ *값을 넣은 파라미터. 줄 앞의* `[1..1]` *은 반드시 값이 있어야 한다는 뜻이다.*

- 필수 파라미터가 비어 있으면 입력 칸에 붉은 밑줄이 그어진다.
- 줄 오른쪽의 되돌리기 아이콘은 값을 정의의 기본값으로 돌린다.
- 입력 칸에 보이는 회색 값은 기본값일 뿐, 파일에 저장된 값이 아닐 수 있다.

12.2.6 설계로 ECU 구성하기

설계에서 정해지는 값(태스크, RTE 매핑, 통신 스택 등)은 손으로 넣지 않고 설계에서 채운다.

1. **ECU** 목록의 ECU 행에서 **PRTE Auto Configuration** 아이콘을 누른다.
2. **PRTE Auto Configuration** 창에 미리보기가 나온다. 아직 아무것도 기록되지 않는다. 파일이 영역별로 묶여 있다. 파일을 누르면 추가할 것(**Adds**), 설계에서 온 값(**Values fro you set them**), 지울 것(**Removed**)이 보인다. **NOTES** 에는 참고 사항이, `Not imported:` 에는 아직 임포트해야 할 모듈이 나온다.
3. **Configure** 를 누른다. 이후 실행에서는 버튼이 **Apply** 이다.



*① 이번에 기록할 파일, ② 펼친 파일 `Os`, ③ **Configure**.*

> **주의.** 사용자가 넣은 값이라도 설계가 정하는 값이면 바뀐다. **Apply** 를 누르기 전에 미리보기를 읽는다.

- 실행 전체가 실행 취소 한 단계이다.
- 설계를 바꾸면 아이콘에 점이 붙고, 툴팁에 바뀔 파일이 나온다. 아이콘을 눌러 다시 맞춘다.
- 설계가 모든 값을 정하지는 않는다. 아직 없는 값은 **VALIDATION** 탭에 나온다.

12.3 레퍼런스가 해석되는 방식

레퍼런스 값은 목록에서 고른다. 목록에는 정의가 가리키는 대상 컨테이너만 나온다. 대상이 없는 값은 강조 표시되고 **VALIDATION** 에 경고로 보고된다. 다른 에디터에서 이름을 바꾸

12.4 ECUC 자동 생성

설계에서 ECUC 를 만드는 기능은 두 가지이며, 모두 VFB 의 ECU Configuration 창에서도 열 수 있다.

기능	하는 일
PRTE Platform Import	고른 BSW 모듈의 PRTE 정의와 빈 설정을 넣는다
PRTE AUTO CONFIGURATION (**PRTE Auto Configuration** 아이콘)	설계가 정하는 값을 쓴다. 설계가 정하지 않는 값은 남겨 둔다. 모듈은 추가하지 않는다

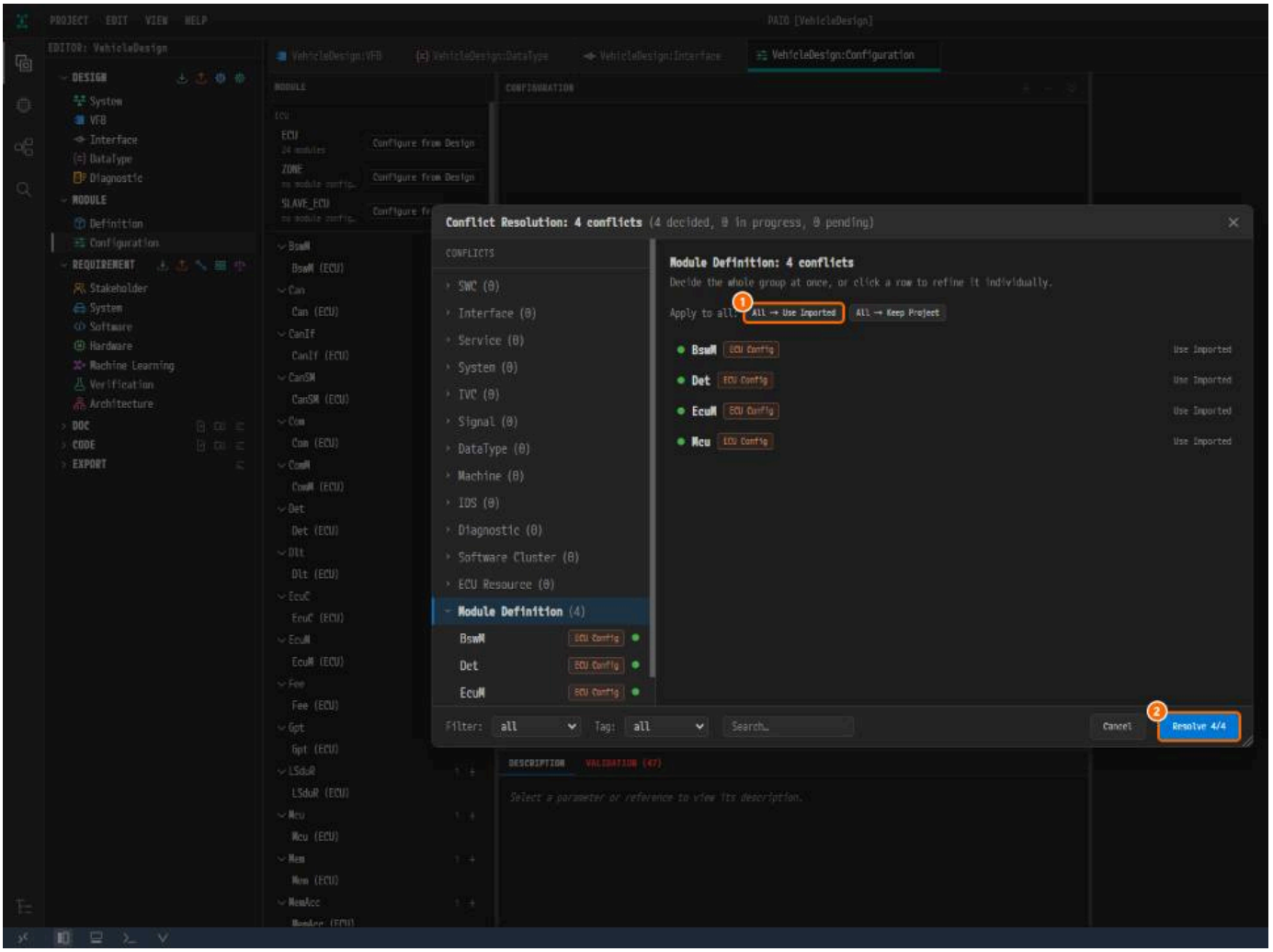
PRTE 가 코드를 만들려면 `Os`, `Rte`, `EcuM`, `BswM`, `Mcu` 다섯 모듈의 설정이 있어야 한다. 실제 순서는 AUTOSAR CP 튜토리얼의 6장부터 10장까지를 따라 하면 된다. 이 장들은 C
해 PRTE AUTO CONFIGURATION 없이 손으로 설정한다.

12.5 ECUC 임포트 · 익스포트

12.5.1 임포트

다른 도구에서 만든 ECUC 값을 ARXML 파일로 가져온다.

1. **MODULE** 아래의 **Configuration** 을 우클릭하고 **Import ECUC Configuration** 을 누른다.
2. `.arxml` 파일을 고른다.
3. 이미 있는 설정과 겹치면 **Conflict Resolution** 창이 열린다. **Module Definition** 을 누르고 **All → Use Imported** 를 고른 뒤 **Resolve** 를 누른다.



① All → Use Imported, ② Resolve 4/4.

끝나면 `ECUC Configuration Import Complete: imported <n> module configuration(s).` 가 보인다. 정의를 찾지 못한 모듈은 건너뛴다.

12.5.2 익스포트

EDITOR 뷰 **DESIGN** 머리글의 **Export** 를 누르면 ECUC 도 함께 내보낸다. **ECU** 가 지정된 설정만 내보내므로, 내보내기 전에 설정의 **ECU** 를 확인한다.

12.6 편집 규칙 · 단축키

12.6.1 이름 규칙

이름은 영문자로 시작하고 영문자·숫자·밑줄만 쓸 수 있으며 128 자 이하이다. 규칙에 맞지 않는 이름은 저장되지 않는다.

12.6.2 단축키

조작	단축키
저장	`Ctrl+S`
실행 취소 / 다시 실행	`Ctrl+Z` / `Ctrl+Y`
선택 항목 삭제	`Delete`
컨테이너 인스턴스 복제 (Configuration 에디터)	`Ctrl+D`

12.6.3 외부 변경

다른 도구가 열린 파일을 바꾸면, 저장하지 않은 편집이 있을 때는 그 편집을 지키고 경고만 띄운다. 편집이 없으면 조용히 다시 읽는다.

12.6.4 라이선스 상태와 편집 제한

읽기 전용 세션에서는 편집이 막힌다. 03. 설치 · 로그인 · 라이선스를 참고한다.

작업 흐름

모델을 만든 뒤에 하는 일 — 가져오고 내보내기, 검증, 코드 작성과 빌드를 다룬다.

- 임포트 · 익스포트
- Validation
- AI 어시스턴트
- 코드 편집 · 빌드 · 터미널 · 문서
- 원격 개발
- AI 워크플로우

임포트 · 익스포트

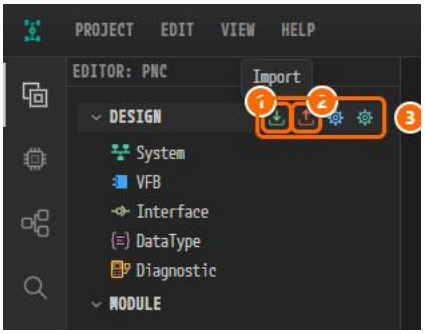
이 장에서 다루는 것

PAIO 프로젝트로 외부 파일(ARXML, DBC)을 가져오는 방법과, 프로젝트 모델을 AUTOSAR ARXML 이나 DBC 로 내보내는 방법을 설명한다. 충돌 해결, ECU 하나만 잘라 내보내는 E

- 대상 독자: 다른 도구에서 만든 ARXML/DBC 를 PAIO 로 옮기거나, PAIO 모델을 다른 도구로 넘기려는 사용자
- 선행 조건: 프로젝트가 열려 있는 것(04. 워크스페이스와 프로젝트)

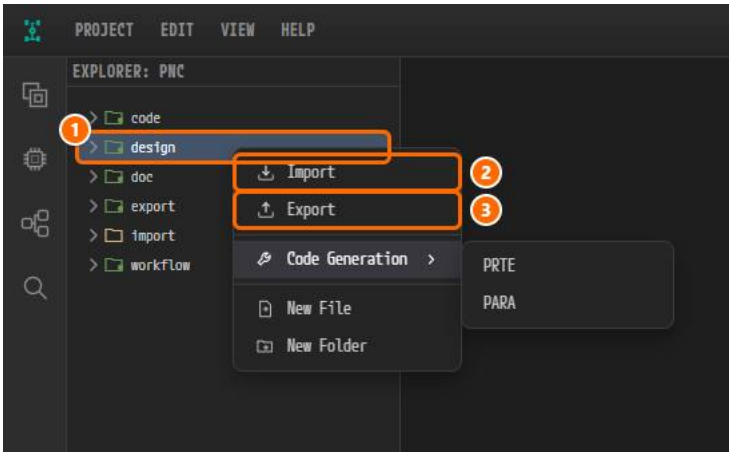
13.1 한눈에 보기

하려는 일	시작 위치
ARXML/DBC 를 프로젝트로 가져오기	**EDITOR** 뷰 **DESIGN** 헤더의 **Import** 아이콘
프로젝트 전체를 ARXML, DBC 또는 둘 다로 내보내기	**EDITOR** 뷰 **DESIGN** 헤더의 **Export** 아이콘
CAN 클러스터 하나를 DBC 로 내보내기	System 에디터에서 CAN 클러스터 우클릭 ▸ **Export to DBC**
ECU 나 HPC 하나만 잘라 내보내기	System 에디터에서 노드 우클릭 ▸ **Extract ECU** / **Extract Machine**
ECUC 설정값만 가져오기	**MODULE** 아래의 **Configuration** 우클릭 ▸ **Import ECUC Configuration**
PopcornSAR AP 베이스 설정 넣기	Machine Configuration 다이얼로그의 **PARA Platform Import** 버튼



*① **Import**, ② **Export**, ③ 헤더 아이콘 네 개: **Import**, **Export**, **PRTE Code Generation**, **PARA Code Generation**.*

같은 두 명령은 Explorer 의 `design` 폴더 컨텍스트 메뉴에도 있다.



*① `design` 폴더, ② **Import**, ③ **Export**.*

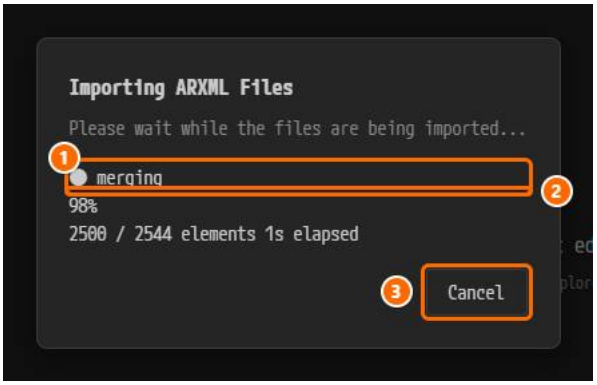
라이선스 조건

익스포트(ARXML, DBC)와 Extract 는 **활성 라이선스와 서버 온라인 검증**을 모두 요구한다. 임포트는 라이선스가 만료된 상태에서도 동작한다.

13.2 ARXML / DBC 임포트

13.2.1 절차

1. **DESIGN** 헤더의 **Import** 아이콘을 누른다. 다이얼로그 `Select files to import` 가 열린다.
2. 파일을 고른다. 여러 개를 한 번에 고를 수 있고, ARXML 과 DBC 를 섞어도 된다.
3. 확정한다. 프로젝트의 저장되지 않은 편집이 먼저 디스크에 저장된다.
4. 진행 다이얼로그 `Importing ARXML Files` 가 나타난다. **Cancel** 을 누르면 임포트가 멈추고, 이미 쓴 파일은 모두 되돌려진다.
5. 결과에 따라 세 갈래(13.2.3 ~ 13.2.5)로 이어진다.



*① 현재 단계 이름, ② 백분율과 요소 카운터가 붙은 단계 줄, ③ **Cancel**.*

원본 파일은 선택한 자리에서 그대로 읽으며, 프로젝트의 `import/` 폴더로 복사하지 않는다.

13.2.2 임포트가 거부되는 경우

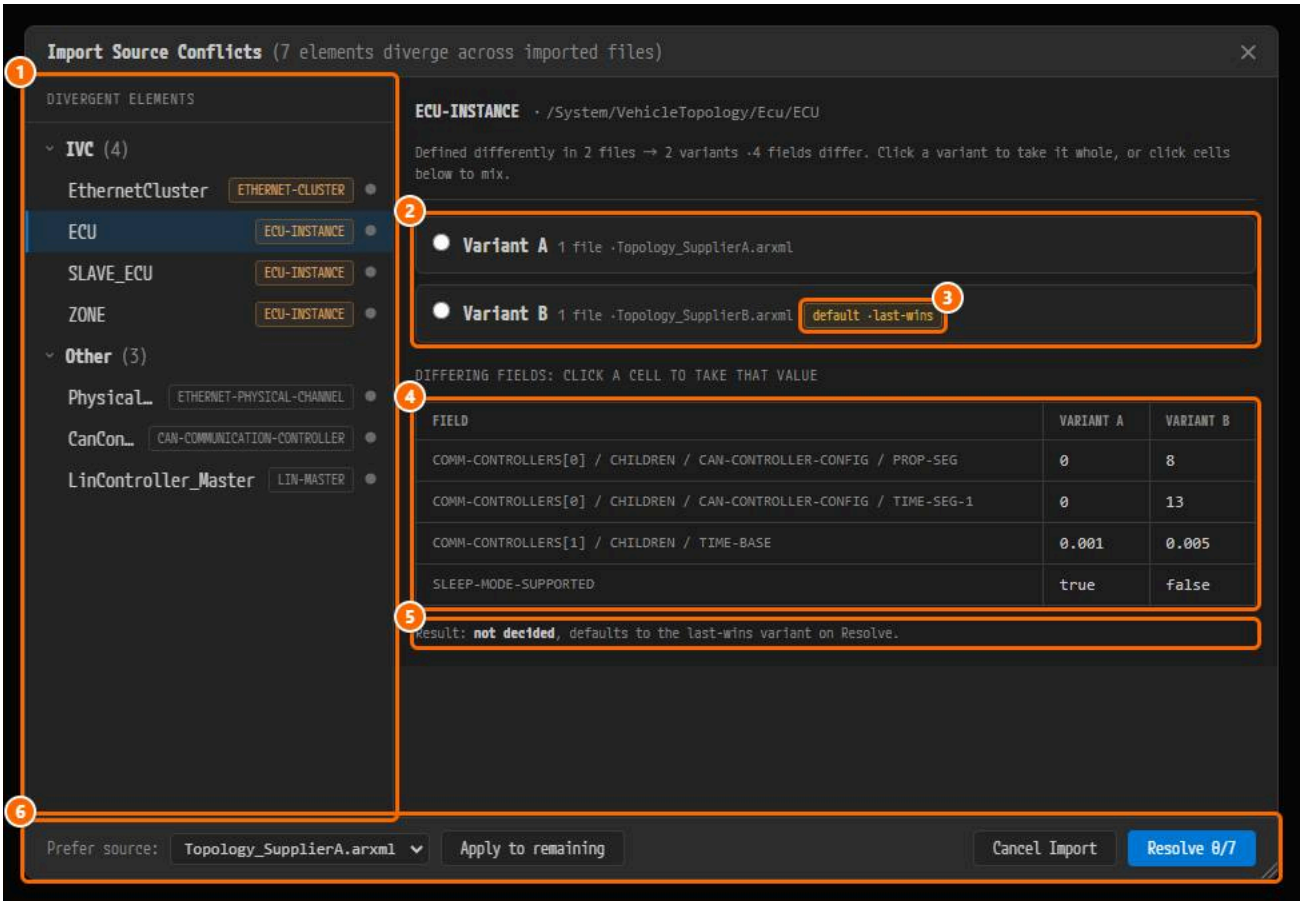
임포트는 한 번에 하나만 진행할 수 있다. 이전 임포트가 끝나지 않았으면 새 임포트는 `Import rejected: <사유>. Finish or cancel it first.` 로 거부된다.

13.2.3 결과 1 — 성공

`Import complete: N elements imported from M file(s)` 가 나타나고 다이얼로그가 저절로 닫힌다. 열려 있던 에디터들은 새 내용으로 다시 읽어 들인다.

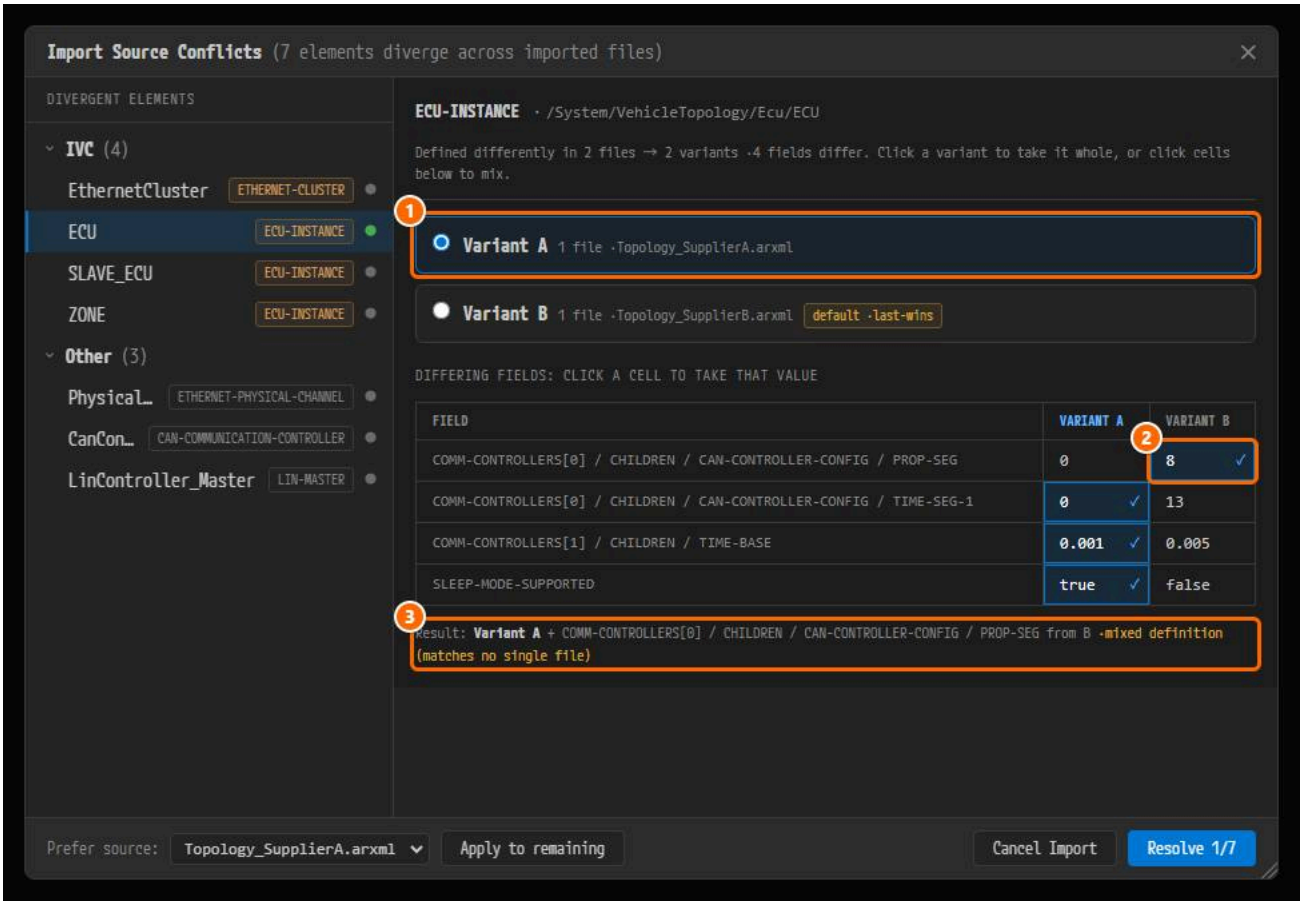
13.2.4 결과 2 — 원본 충돌 (Import Source Conflicts)

한 번에 고른 여러 파일이 같은 요소를 서로 다르게 정의하고 있으면, 디스크에 아무것도 쓰기 전에 **Import Source Conflicts** 다이얼로그가 나타난다.

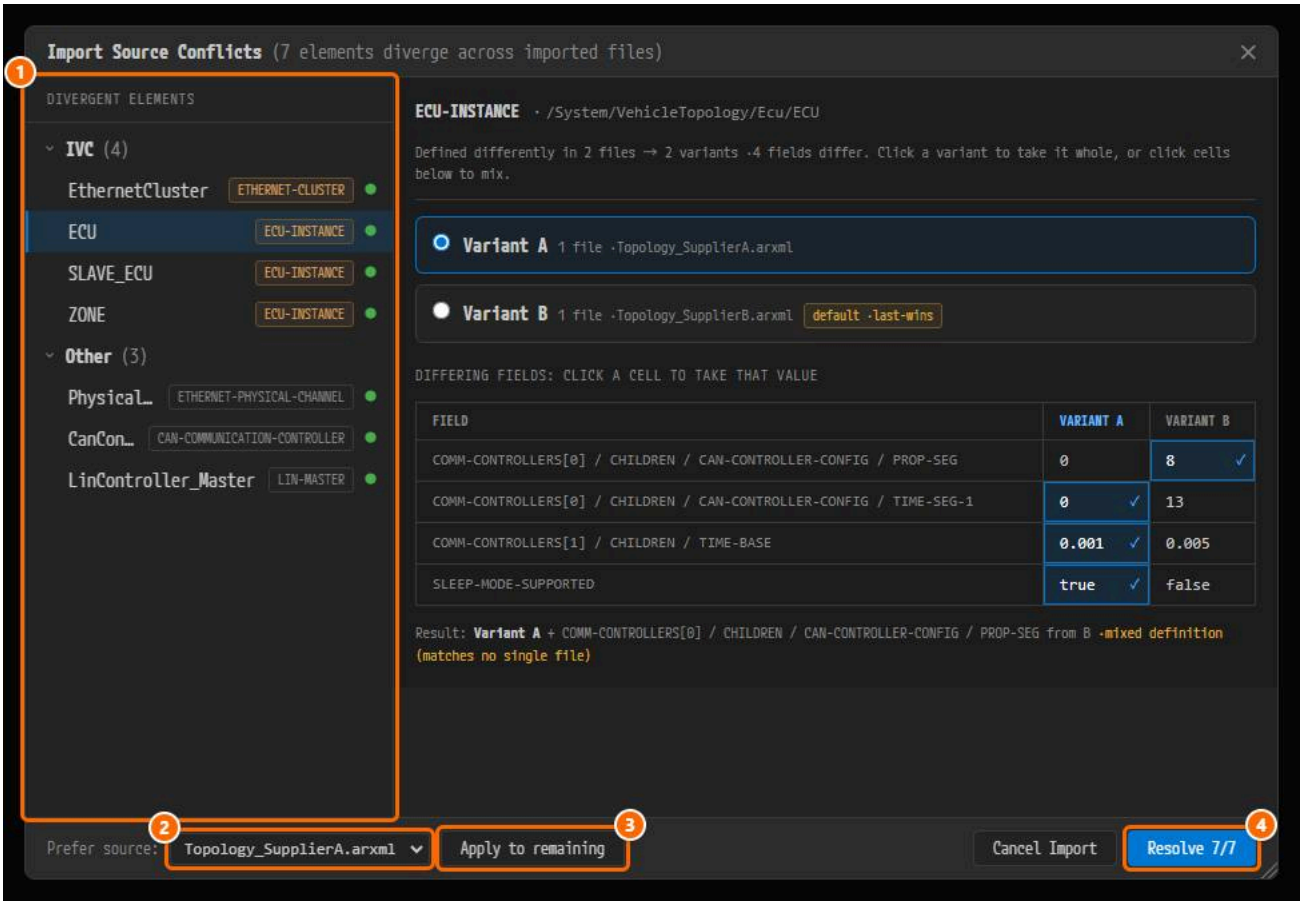


*① **Divergent Elements**, ② 변형 카드, ③ 기본값 배지, ④ 필드 표, ⑤ `Result:` 줄, ⑥ 하단 영역.*

1. 왼쪽 **Divergent Elements** 목록에서 요소를 고른다.
2. 변형 카드를 누르면 그 변형을 통째로 채택하고, 필드 표의 셀을 누르면 그 필드만 채택한다.
3. 남은 요소를 한꺼번에 정하려면 **Prefer source:** 에서 파일을 고르고 **Apply to remaining** 을 누른다.
4. **Resolve d/N** 을 누른다. 정하지 않은 요소는 기본값 배지가 붙은 변형이 채택된다.



① 통째로 채택한 `Variant A`, ② `Variant B` 에서 가져온 셀 하나, ③ `· mixed definition (matches no single file)` 이 붙은 `Result:` 줄.

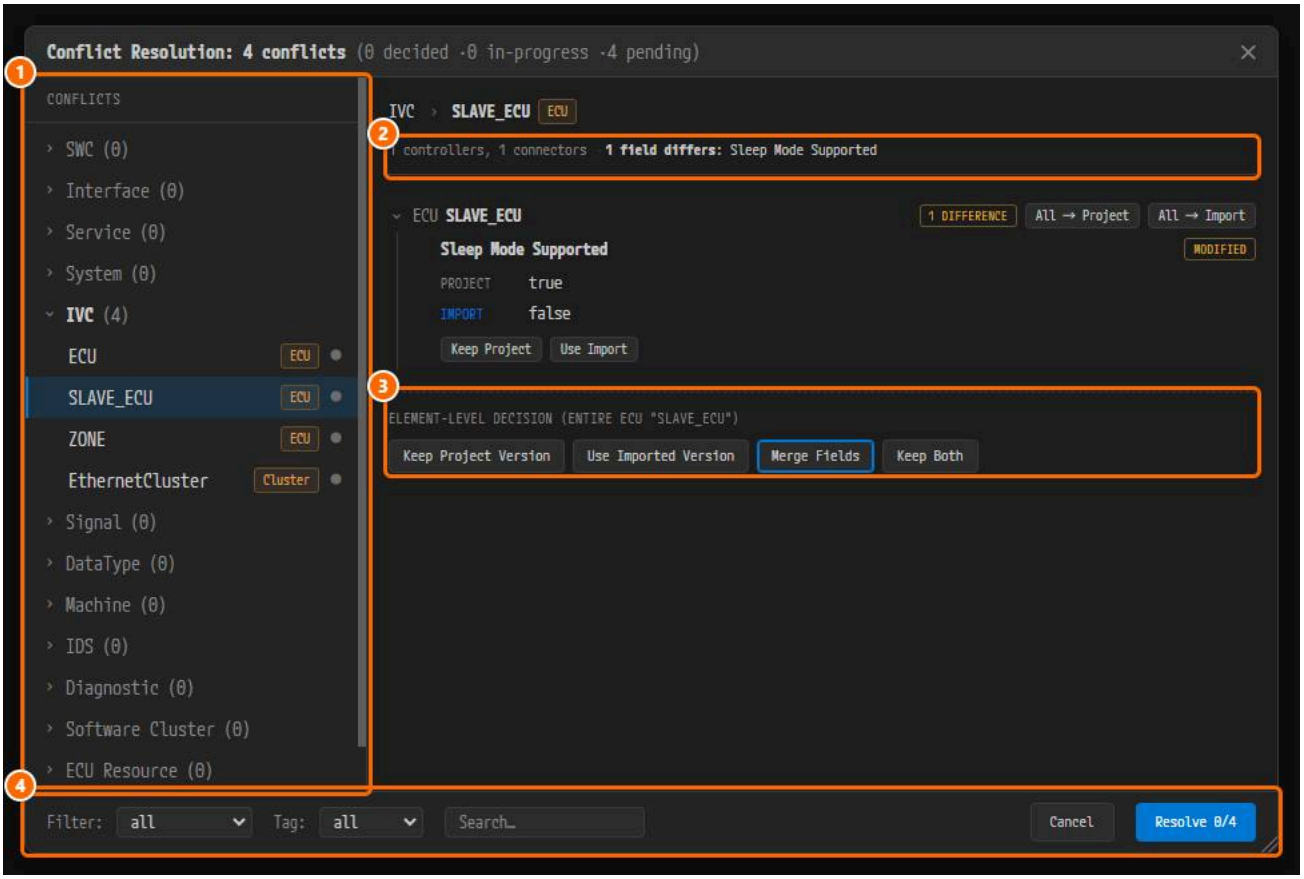


① 모두 켜진 점, ② **Prefer source:** 에서 고른 파일, ③ **Apply to remaining**, ④ **Resolve 7/7.**

Cancel Import 를 누르면 임포트를 멈춘다. 아직 아무것도 쓰지 않은 상태이다.

13.2.5 결과 3 — 모델 충돌 (Conflict Resolution)

이미 프로젝트에 있는 요소와 새로 들어온 요소가 서로 다른 값을 가지면 **Conflict Resolution** 다이얼로그가 나타난다.



*① **Conflicts** 트리, ② 요약 줄, ③ 요소 단위 결정 버튼, ④ 하단 영역.*

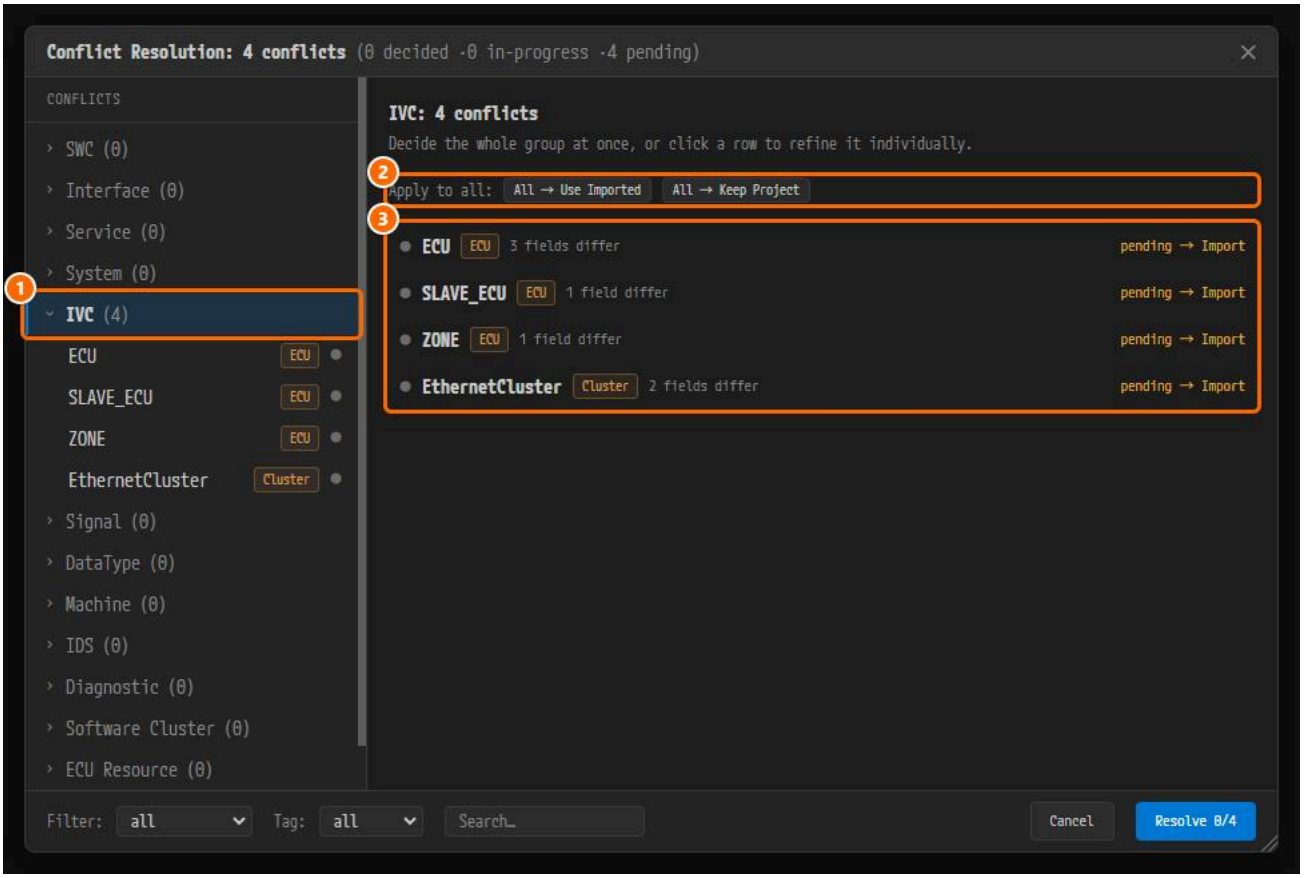
1. 왼쪽 **Conflicts** 트리에서 항목을 고르고, 프로젝트 값과 임포트 값을 비교한다.
2. `Element-level decision` 아래의 네 버튼 중 하나를 고른다.

버튼	동작
Keep Project Version	프로젝트에 있던 값을 그대로 둔다
Use Imported Version	임포트한 값으로 교체한다
Merge Fields	필드 단위로 어느 쪽 값을 쓸지 직접 고른다
Keep Both	임포트한 쪽에 새 이름을 붙여 둘 다 남긴다

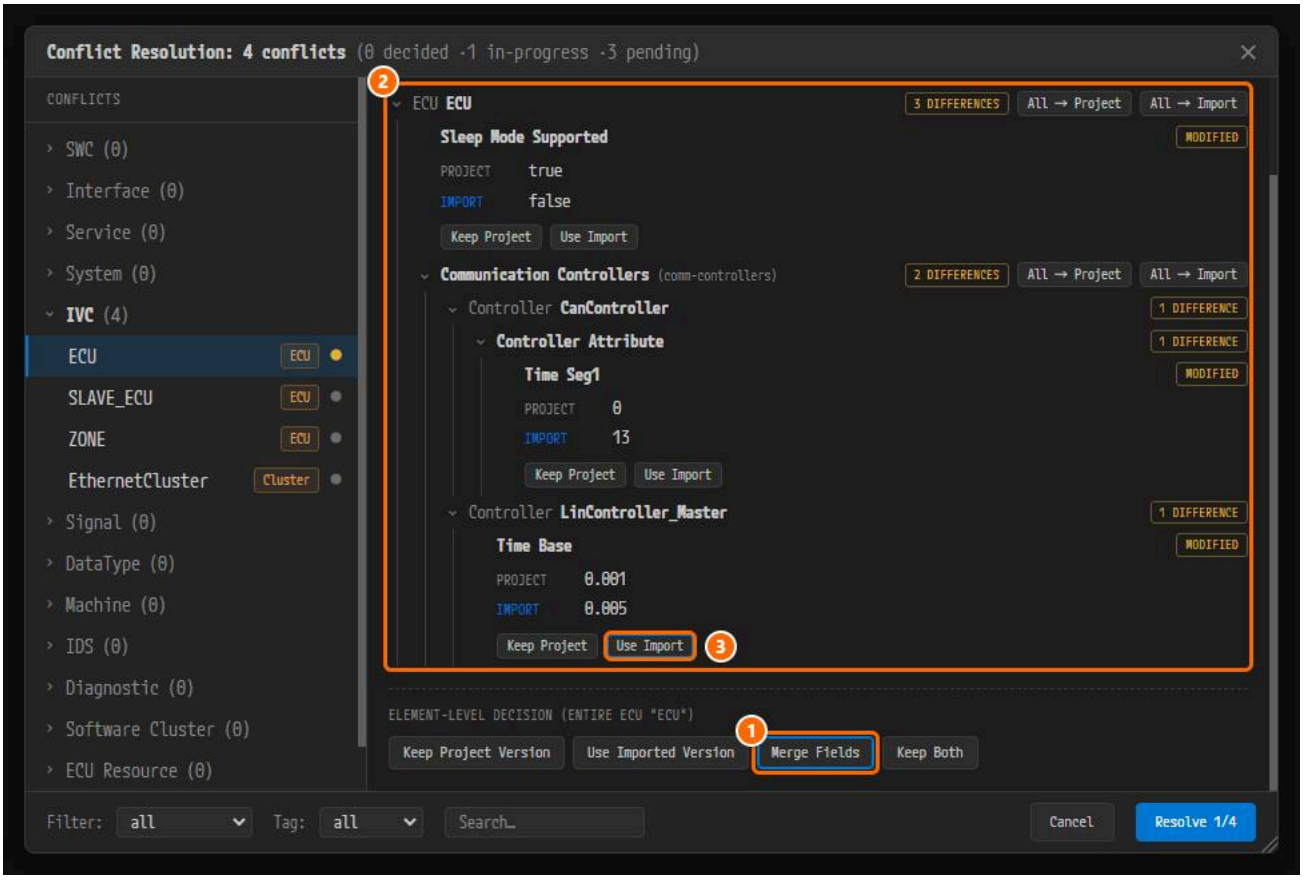
3. **Resolve n/N** 을 눌러 확정한다.

> **정하지 않은 항목은 유지되지 않고 임포트 값으로 덮어써진다.** 미결정 항목이 남아 있으면 `Unresolved Conflicts` 확인 창이 먼저 묻는다.

영역 전체를 한 번에 처리하려면 영역 머리글을 누르고 **All → Use Imported** 또는 **All → Keep Project** 를 쓴다.



*① 영역 머리글(**IVC**), ② 일괄 버튼, ③ 현재 결정 상태가 붙은 행.*

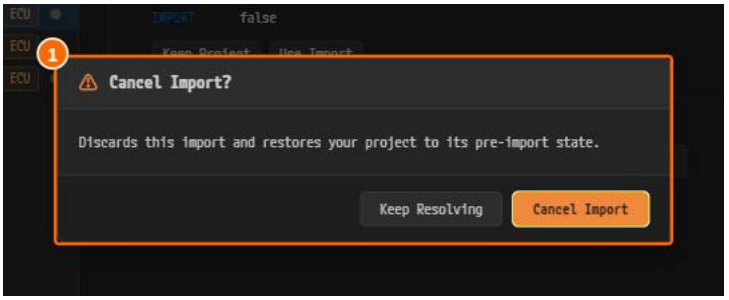


*① **Merge Fields**, ② 항목의 필드 트리, ③ 한 필드에 고른 **Use Import**.*



① `Unresolved Conflicts` *확인 창(**Cancel**, **Overwrite & Continue**).*

Cancel 을 누르면 임포트를 멈추고 프로젝트를 임포트 전 상태로 되돌린다.



① `Cancel Import?` *확인 창(**Keep Resolving**, **Cancel Import**).*

13.2.6 임포트가 함께 처리하는 것

한 번의 ARXML 임포트가 다음을 함께 처리한다. 네트워크 토폴로지, 시그널, SWC, 인터페이스, 데이터 타입, 서비스, 시스템, 보안 이벤트, 진단, 소프트웨어 클러스터, ECU 리소스, M 값보다 먼저 읽으므로 둘이 같은 파일에 있어도 된다.

임포트 전체는 **하나의 실행 취소 단위**이다. `Ctrl+Z` 한 번으로 임포트 직전 상태로 돌아간다.

13.2.7 재임포트는 덮어쓰기가 아니라 병합이다

프로젝트에 이미 있는 요소는 통째로 바꾸지 않고 필드 단위로 합친다.

- 한쪽에만 값이 있으면 그 값을 쓴다.
- 양쪽 모두 값이 있고 서로 다를 때만 충돌로 올라온다.
- 다이어그램 좌표와 크기는 내용으로 보지 않으므로 기존 배치가 유지된다.

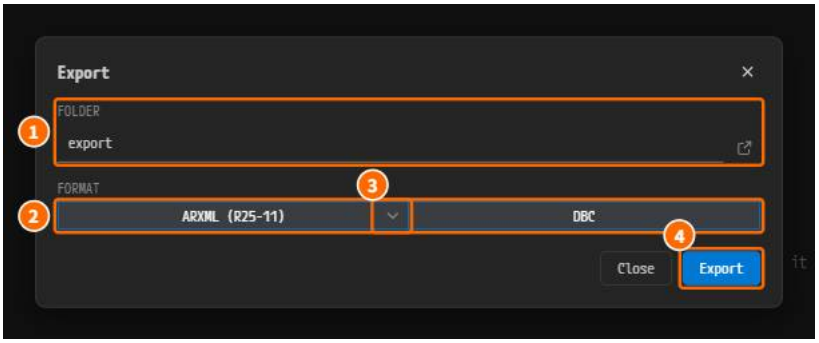
13.2.8 DBC 인코딩 주의

DBC 파일은 **UTF-8 BOM 이 있을 때만** UTF-8 로 읽고, 그 외에는 ISO-8859-1 로 읽는다. 문자가 깨지면 파일에 BOM 을 넣거나 ISO-8859-1 로 다시 저장한다.

13.3 프로젝트 익스포트 (ARXML, DBC)

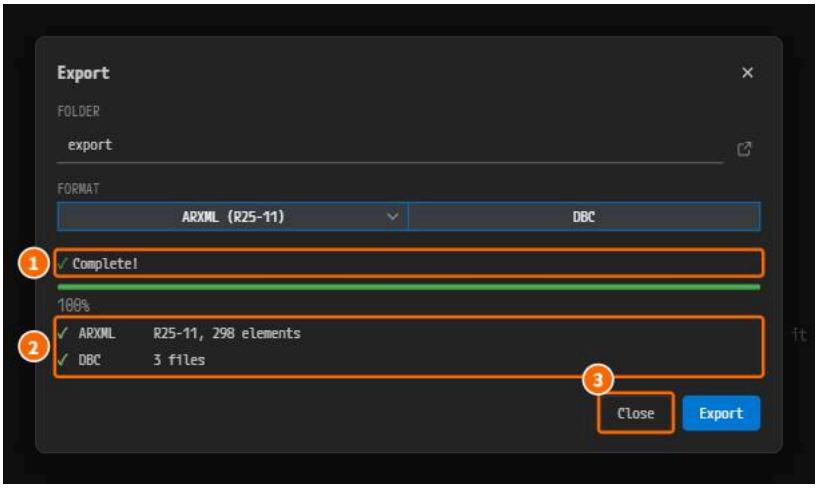
13.3.1 절차

1. 편집을 저장한다(**PROJECT** ▸ **Save All**).
2. **DESIGN** 헤더의 **Export** 아이콘을 누른다. **Export** 창이 열린다.
3. **FOLDER** 를 확인한다. `export` 는 프로젝트의 `export` 폴더이다. 프로젝트 안의 다른 폴더나 전체 경로를 입력하거나, 오른쪽 끝 아이콘으로 고른다. 원격 창에는 이 아이콘이
4. **FORMAT** 에서 **ARXML** 과 **DBC** 중 하나나 둘을 고른다. 적어도 하나는 선택되어 있다. **ARXML** 버튼의 화살표는 AUTOSAR 버전 목록이며, 지금은 `R25-11` 만 있다.



*① **FOLDER**, ② 두 형식을 모두 고른 **FORMAT**, ③ AUTOSAR 버전, ④ **Export**.*

5. **Export** 를 누른다. 형식을 차례로 쓰고, 같은 이름의 파일은 덮어쓴다. **Cancel** 을 누르면 중단한다.
6. `✓ Complete!` 와 형식별 결과가 한 줄씩 나타난다. **Close** 를 누른다.

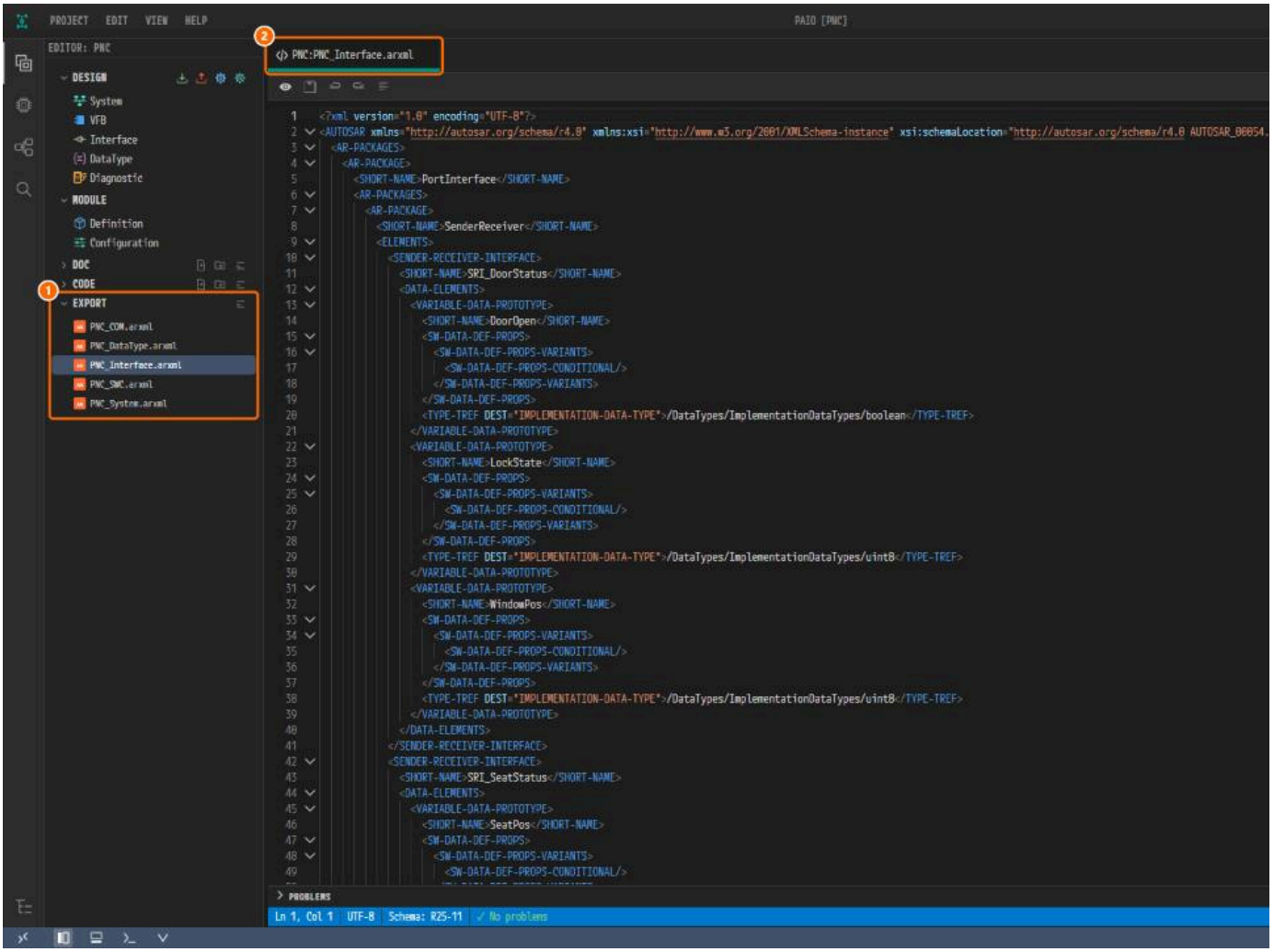


*① `✓ Complete!`, ② 형식별 결과, ③ **Close**.*

실패한 형식이 있으면 `X Finished with errors` 가 나타나고 그 형식의 줄에 사유가 보인다. 다른 형식은 그대로 만들어진다.

13.3.2 만들어지는 파일

ARXML 익스포트는 `<프로젝트>_<접미사>.arxml` 형태의 파일 여러 개를 쓴다(예: `MyProject_System.arxml`, `MyProject_COM.arxml`). 파일은 해당 내용이 있을 때만 만
DBC 익스포트는 CAN 클러스터마다 `<클러스터>.dbc` 파일을 하나씩 쓴다(13.6).



*① **EXPORT** 섹션의 익스포트한 파일, ② 더블클릭해서 연 파일.*

13.3.3 익스포트가 읽는 데이터

ARXML 익스포트는 `design/` **폴더에 저장된 파일만 읽는다.** 저장하지 않은 편집은 그 에디터가 열려 있어도 내보내지 않는다. 익스포트 전에 저장한다.

13.3.4 출력 규격

AUTOSAR 네임스페이스는 **R25-11 로 고정**된다. 값이 지정되지 않은 참조는 내보내지 않는다.

13.3.5 결과의 패키지 배치

결과의 AR-PACKAGE 구조는 규칙으로 고정되어 있다.

13.3.6 AUTOSAR로 나가지 않는 정보

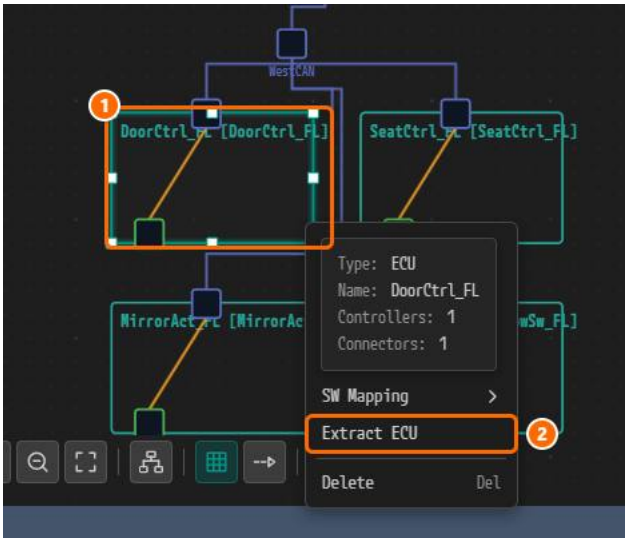
- ECUC 설정이 어느 ECU 의 것인지 같은 소유 관계는 ARXML 의 ADMIN-DATA SDG로 함께 나가고, 다시 임포트해도 살아난다.
- **다이어그램 좌표와 화면 배치는 ARXML 로 나가지 않는다.** 빈 프로젝트로 다시 임포트하면 노드가 자동 배치된다. 같은 프로젝트로 다시 임포트하면 기존 배치가 유지된다(13.2.

13.4 ECU / Machine Extract

한 노드(ECU 또는 HPC)와 그 노드가 참조하는 것만 잘라 별도 ARXML 로 만든다. 특정 ECU 담당자에게 필요한 부분만 넘길 때 쓴다.

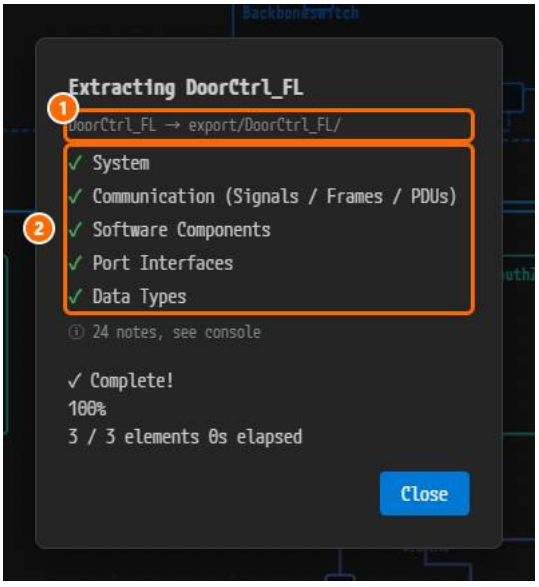
13.4.1 절차

1. 편집을 저장하고 System 에디터를 연다(**EDITOR** 뷰 **DESIGN** 목록의 **System**).
2. 대상 노드를 우클릭하고 **Extract ECU**(Classic ECU) 또는 **Extract Machine**(Adaptive HPC)을 고른다.



*① ECU 노드 `DoorCtrl_FL`, ② **Extract ECU**.*

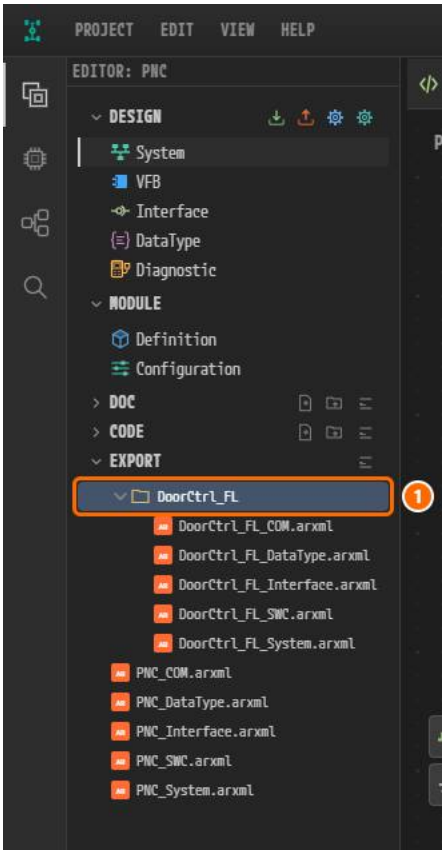
3. 성공하면 만들어진 파일이 영역 이름으로 나열된다. **Close** 로 닫는다.



① 부제 `DoorCtrl_FL → export/DoorCtrl_FL/`, ② 실제로 쓴 영역의 체크리스트.

13.4.2 결과

파일은 `<프로젝트>/export/<노드이름>/` 에 놓이고, **노드 이름**을 접두사로 쓴다(`DoorEcu_System.arxml`). 저장된 모델만 읽는다.



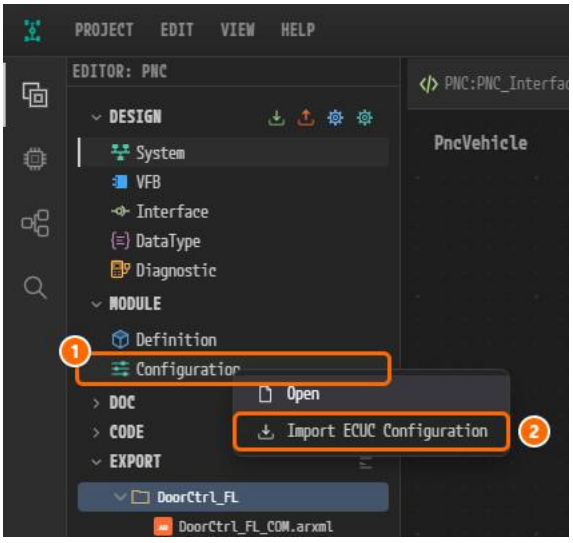
① 행 왼쪽의 펼침 화살표로 펼친 `DoorCtrl_FL` 폴더.

13.4.3 잘라내는 범위

Extract 는 대상 노드, 그 노드가 쓰는 프레임, PDU, 시그널, 그 노드의 ECUC 설정값을 남긴다. ECUC 모듈 정의는 전부 내보낸다. 남은 요소는 끊어진 참조가 없도록 정리된다.

13.5 ECUC Configuration 개별 임포트

EDITOR 뷰 **MODULE** 아래의 **Configuration** 을 우클릭하고 **Import ECUC Configuration** 을 고른다.



*① **MODULE** 아래의 **Configuration** 행, ② **Import ECUC Configuration**.*

13.5.1 절차

1. `Select ARXML File to Import ECUC Configuration` 에서 ARXML 파일을 **한 개** 고른다.
2. 파일에 `ECUC-MODULE-DEF` 가 있으면 모듈 정의부터 임포트한다.
3. 같은 경로에 설정이 이미 있으면 13.2.5의 충돌 다이얼로그가 나타난다.

4. 결과를 보고한다. 건너뛰거나 거부된 모듈은 사유와 함께 나열된다.

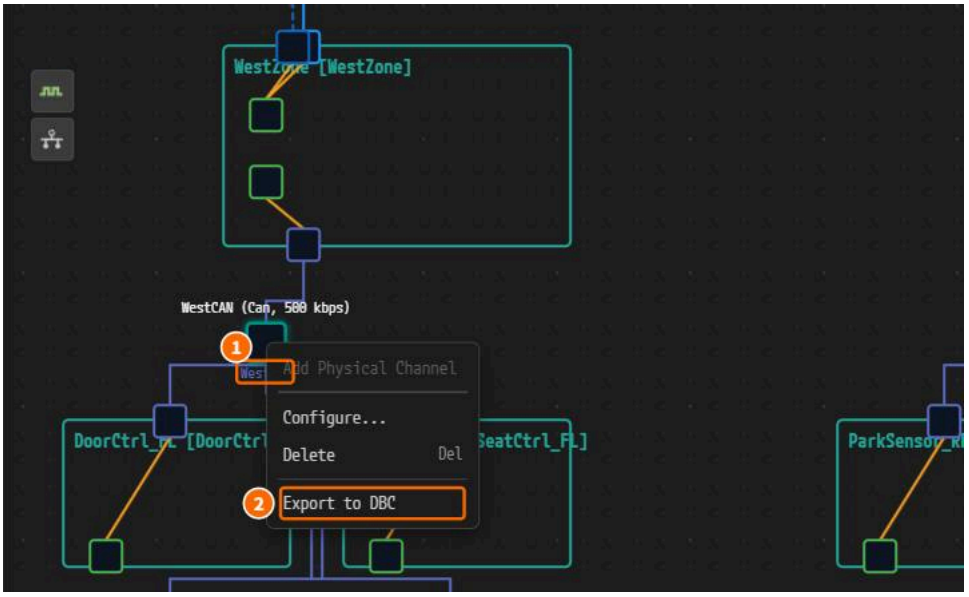
임포트 전체는 하나의 실행 취소 단위이다.

13.5.2 거부되는 이유

설정값의 컨테이너, 파라미터, 레퍼런스가 정의에 없으면 그 모듈 전체가 거부된다.

13.6 DBC

- **임포트**: 13.2와 같은 **Import** 다이얼로그에서 `.dbc` 파일을 고른다. 멀티플렉스 메시지는 MULTIPLEXED-I-PDU로 들어오며, 버스가 CAN인지 CAN FD인지는 파일에서 읽는다
- **모든 CAN 클러스터 익스포트**: 13.3의 **Export** 창에서 **DBC** 를 고른다. CAN 클러스터마다 `<클러스터>.dbc` 가 하나씩 만들어진다. CAN 클러스터가 없는 프로젝트는 `<프로젝`
- **CAN 클러스터 하나 익스포트**: System 에디터에서 CAN 클러스터를 우클릭하고 **Export to DBC** 를 고른다. **Exporting to DBC** 창에 `DBC export complete: 1 file` `>.dbc` 이다.



① *CAN 클러스터* `WestCAN`, ② ***Export to DBC***.

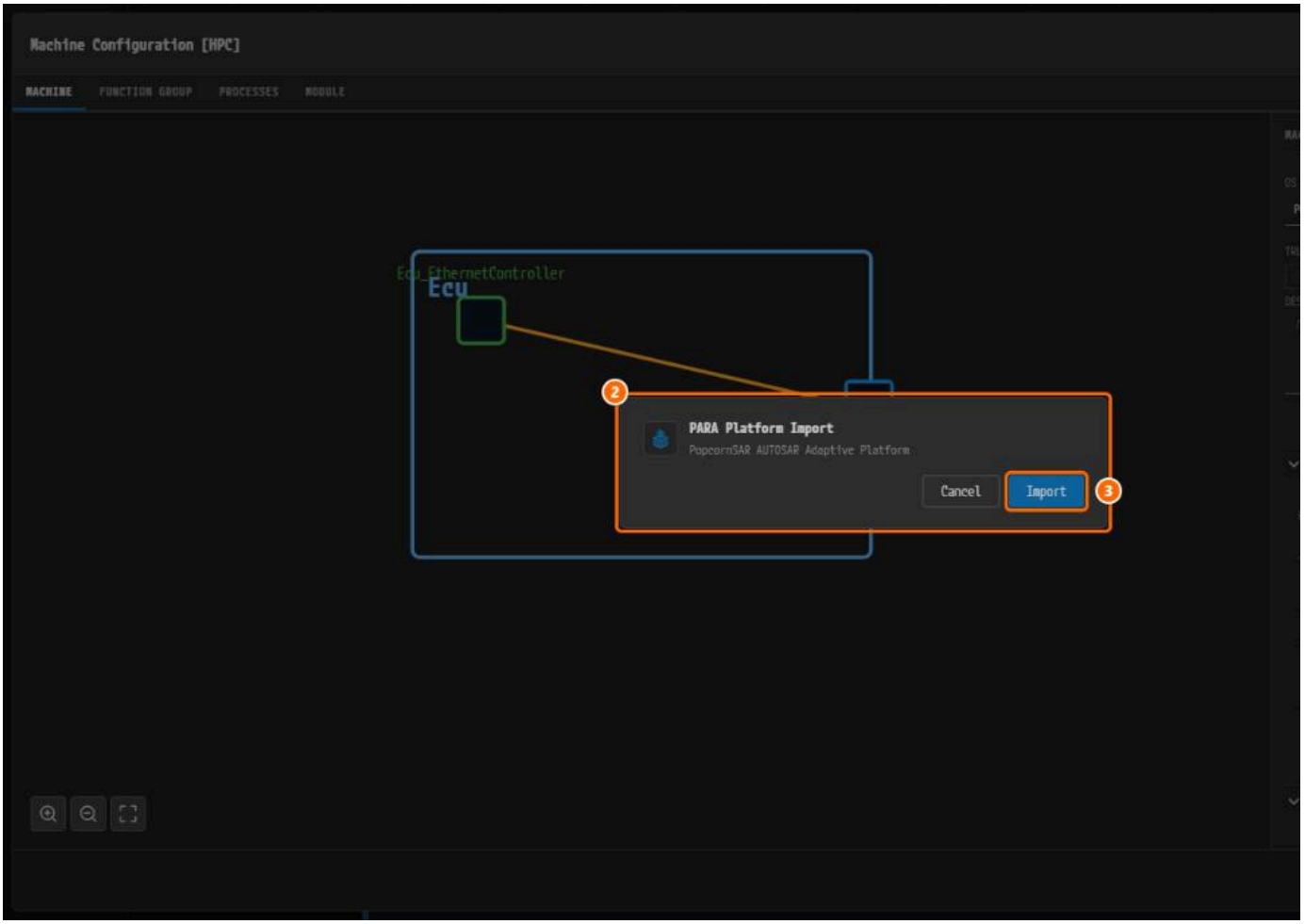
DBC 파일에는 CAN 커넥터가 있는 노드만 들어간다. 프로젝트에 CAN 클러스터가 있으면, 어떤 CAN 클러스터에도 실리지 않는 프레임은 DBC 파일에 들어가지 않는다.

13.7 PARA Platform 베이스 설정 가져오기

PopcornSAR AUTOSAR Adaptive Platform 의 기본 구성을 프로젝트에 넣는다. 파일을 고르지 않고 제품에 동봉된 세트를 쓴다.

13.7.1 절차

1. VFB 에디터에서 HPC 에 매핑된 컴포지션의 **Machine Configuration** 다이얼로그를 연다.
2. 헤더의 **PARA Platform Import** 를 누른다.
3. 확인 창에서 **Import** 를 누른다. 이후는 ARXML 임포트와 같다.



*① **PARA Platform Import**, ② 확인 창, ③ **Import**.*

13.7.2 제약

다른 임포트가 진행 중이거나 라이선스가 만료된 상태면 거부된다.

13.8 자주 겪는 상황

- Export 를 눌렀는데 업그레이드 안내만 나오는 경우: 라이선스가 만료된 상태이다. 임포트는 계속 쓸 수 있다.
- **Resolve** 뒤에 기존 값이 바뀐 경우: 정하지 않은 항목은 임포트 값으로 덮어쓴다(13.2.5).
- 익스포트나 Extract 결과에 최신 편집이 빠진 경우: 먼저 저장한다.

13.9 SSH 창에서 임포트

SSH로 연결된 창에서는 임포트가 SSH 서버에 쓴다.

- 파일은 서버에서 먼저 고른다. 창 위쪽에 프로젝트 폴더를 보여 주는 선택 창이 열린다. 파일을 클릭하고 **Open** 을 누른다. **Import** 에서는 `Ctrl` +클릭으로 더 고른다. **Show Lo**
  ECUC Configuration 이 이렇게 동작한다.
- 임포트 중에 연결이 끊기면 임포트는 `Import failed: The remote connection closed during the import. Part of it may be on the server; rec`
  무것도 쓰이지 않는다. 다시 연결하고 다시 임포트한다.
- 임포트 중에 직접 연결을 끊으려 하면 먼저 묻는다. **Close Connection** 은 임포트를 취소한다. 17.7을 참고한다.

Validation

이 장에서 다루는 것

PAIO 의 실시간 검증을 설명한다. 검증 패널을 여는 방법, 검증이 언제 실행되는지, 목록을 읽는 방법, 문제 지점으로 이동해 고치는 방법을 다룬다. 검사 규칙은 AUTOSAR 표준 제약을 가한 것이다. 검사는 네트워크 구성과 통신, 소프트웨어 구조와 인터페이스, 데이터 타입, Adaptive Platform 설정과 서비스 배포, 진단, Software Cluster 와 Vehicle Package, I 다. 심각도는 규칙마다 정해져 있어 대부분 **error**, 나머지는 **warning** 이다.

- 대상 독자: 모델을 편집하고 익스포트 전에 오류를 정리하려는 사용자
- 선행 조건: 프로젝트가 열려 있어야 한다. 04. 워크스페이스와 프로젝트를 참고한다.

검증에는 실행 버튼이 없다. 프로젝트가 열리는 즉시 실행되고, 모델을 편집할 때마다 다시 실행된다.

14.1 검증 패널 열기와 닫기

1. 화면 맨 아래 상태바의 **V** 버튼(툴팁 `Toggle Validation`)을 누른다. 에디터 영역 아래에 패널이 열린다.
2. 닫으려면 **V** 를 다시 누르거나 패널 헤더의 **Close Panel** 을 누른다.

V 가 유일한 진입점이며, 메뉴, 명령 팔레트, 단축키에는 검증 항목이 없다.

상태바 배지 읽는 법

- 배지 없음: 위반이 없다.
- 숫자: error 와 warning 을 합친 건수다. error 가 하나라도 있으면 빨간색이다.
- `!`: 검증이 실행되지 못했다. "문제 없음"이 **아니다**.

패널 헤더의 `VALIDATION` 제목 옆 배지는 error 만 센다.

14.2 어떤 프로젝트를 검증하나

PAIO 창 하나에는 프로젝트가 하나 열리고, 검증은 항상 그 프로젝트를 대상으로 한다. 다른 프로젝트를 검증하려면 그 프로젝트를 별도의 창에서 연다. 프로젝트가 열려 있지 않으면 `run validation.` 이 표시된다.

14.3 언제 실행되고 무엇을 읽나

검증은 다음 경우에 자동으로 실행된다.

- 프로젝트가 열릴 때
- 모델을 편집할 때마다
- ARXML 을 임포트한 뒤
- 프로젝트의 설계 파일이 디스크에서 바뀌었을 때

저장하지 않은 편집을 포함해 프로젝트의 설계 데이터 전체를 읽는다. 아직 만들지 않은 부분은 오류로 처리하지 않는다.

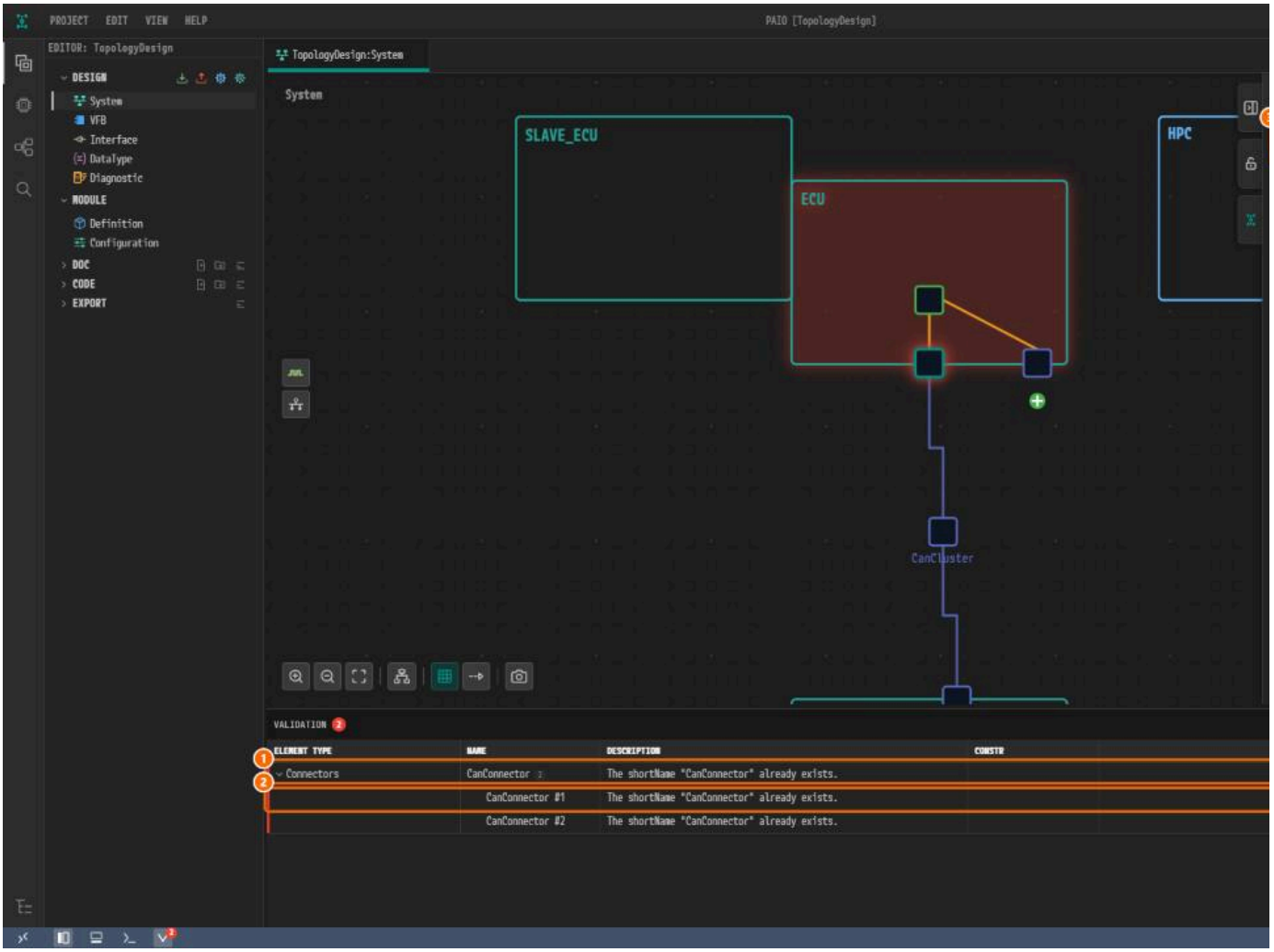
14.4 목록 보는 법

목록은 **Element Type**, **Name**, **Description**, **Constr**(AUTOSAR 제약 번호) 네 열로 된 표다. 행의 빨간 막대는 error, 주황 막대는 warning 이다. 표준 제약 번호가 없는 검사는 **Co**

그룹 행

한 부모 아래 형제 요소끼리 SHORT-NAME 이 겹치면 건수 배지가 붙은 그룹 행 하나로 묶인다.

1. 그룹 행을 누른다. 자식 행 `#1`, `#2` 등이 나타난다.
2. 자식 행을 누르면 그 요소에 표시가 붙는다.



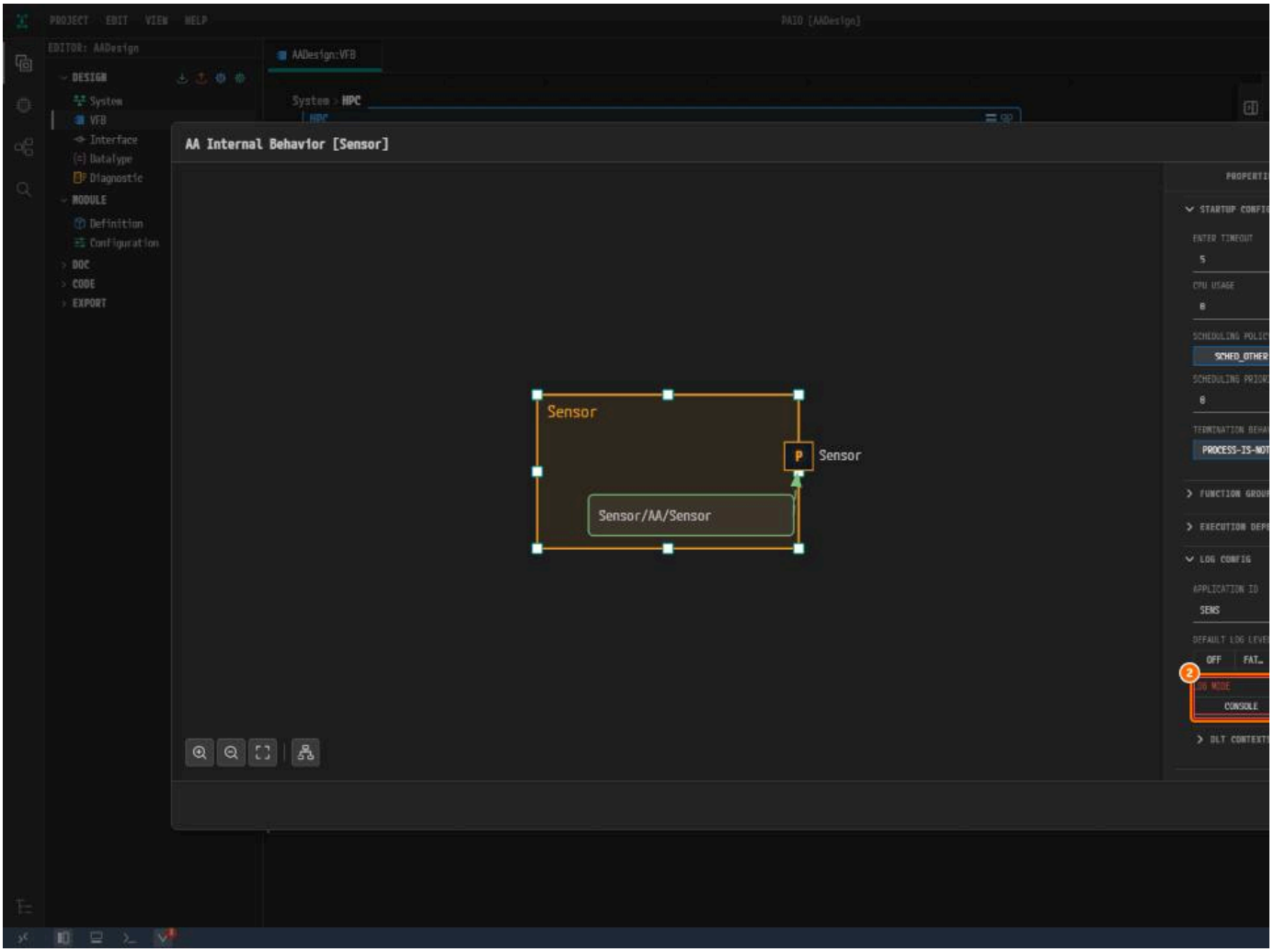
① 건수 배지 `2` 가 붙은 그룹 행 `Connectors` / `CanConnector` 가 펼쳐져 있다. ② 누른 자식 행 `CanConnector #1`. ③ 그 커넥터의 NAME 필드가 빨갛게 표시된다.

목록이 비었을 때

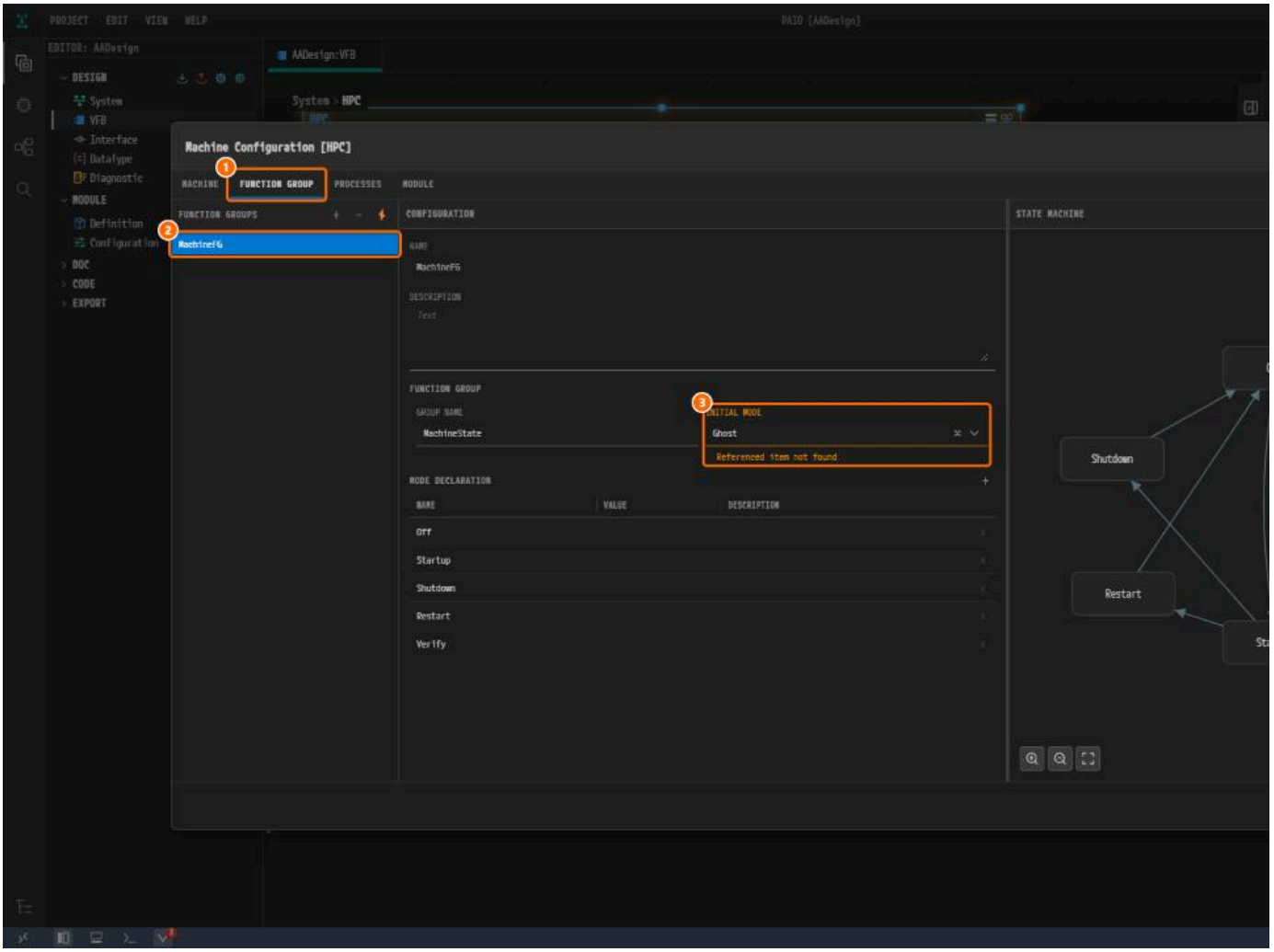
`No validation issues.` 는 위반이 없다는 뜻이다. `Validation could not run. See the developer console for details.` 는 검증 실행이 실패했다는 뜻이다

14.5 위반 위치로 이동

1. 행을 누른다. 그 요소를 소유한 에디터 탭이 열리고 요소가 선택된다.
2. 요소에 행의 심각도 색으로 위반 표시가 남는다. error 는 빨강, warning 은 주황이다. 대상이 입력 필드면 필드에 포커스가 들어가고 값이 선택된다.
3. 다이얼로그 안에서 편집하는 요소(**Machine Configuration**, **AA Internal Behavior** 등)는 해당 다이얼로그까지 열린다.

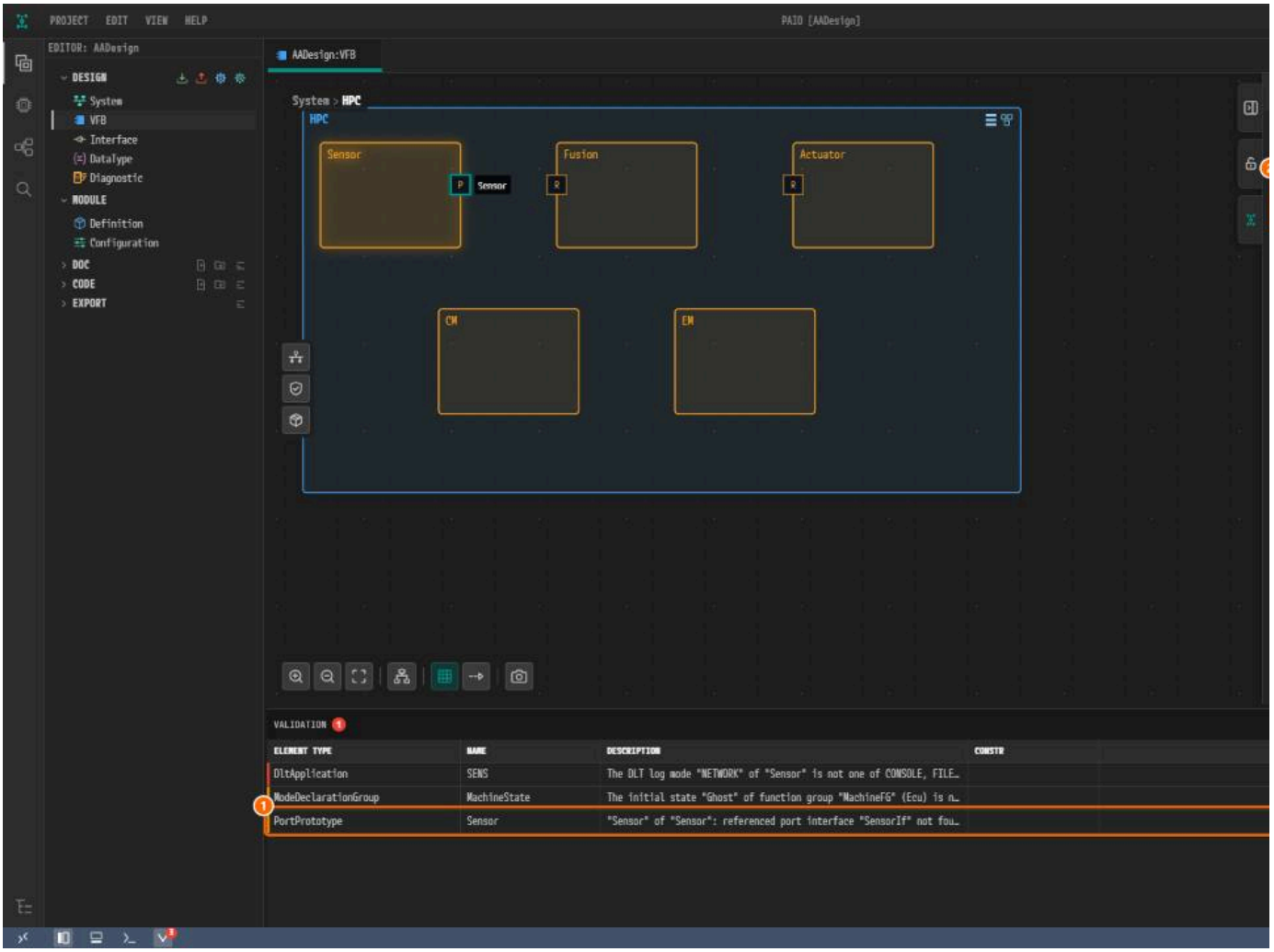


*다이얼로그 안의 필드다. ① **LOG CONFIG** 가 펼쳐진 **EXECUTION** 탭. ② 허용되지 않은 값을 가진 `LOG MODE` 가 빨갛게 표시된다.*

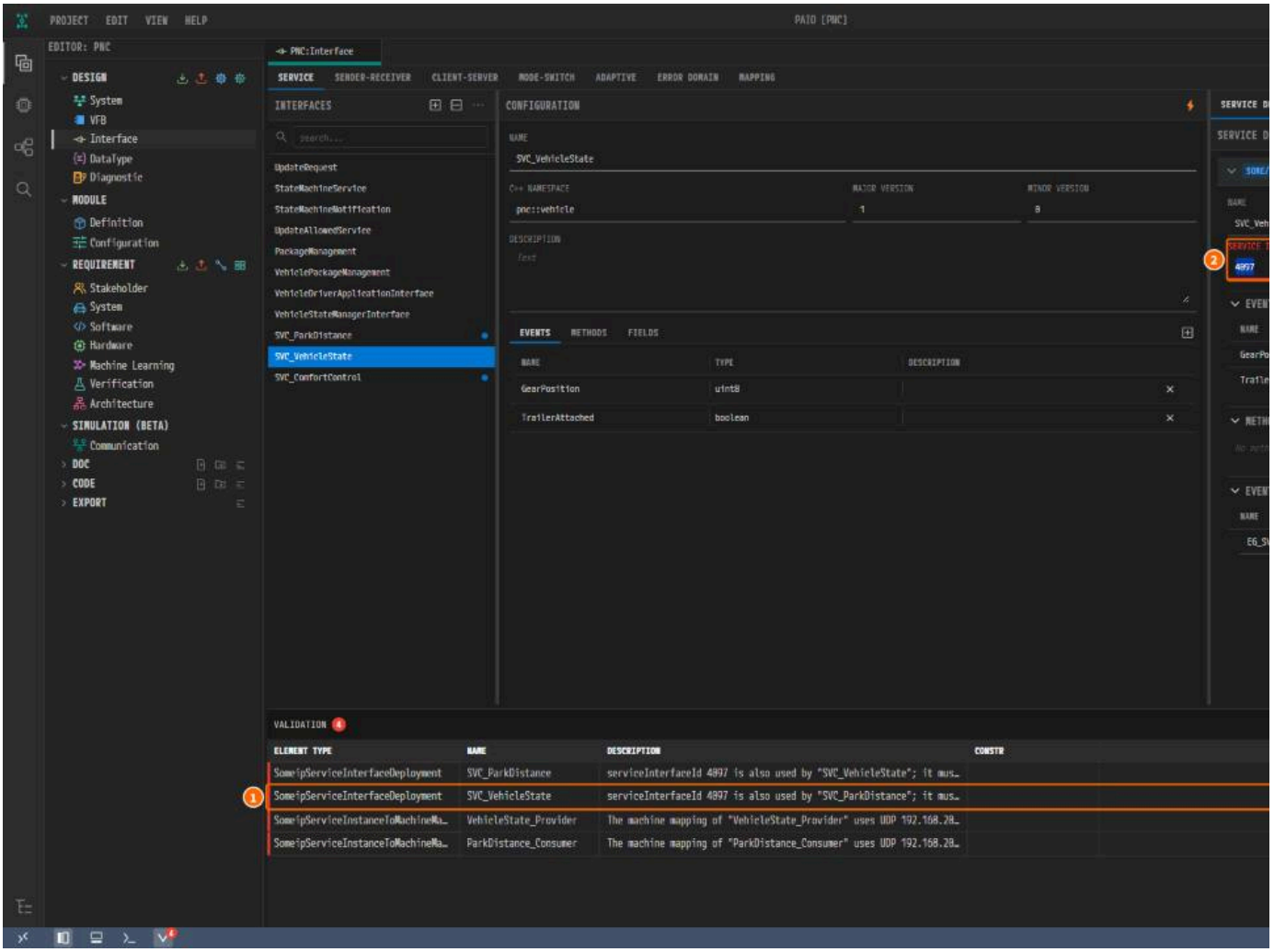


*필드를 표시하지 않고 다이얼로그만 여는 행도 있다. ① **FUNCTION GROUP** 탭. ② `MachineFG` 를 직접 선택한다. ③ `INITIAL MODE` 에 `Ghost` 와 `Referenced item not f*

4. 값을 고친다. 검증이 다시 실행되고, 고친 행은 목록에서 사라진다.

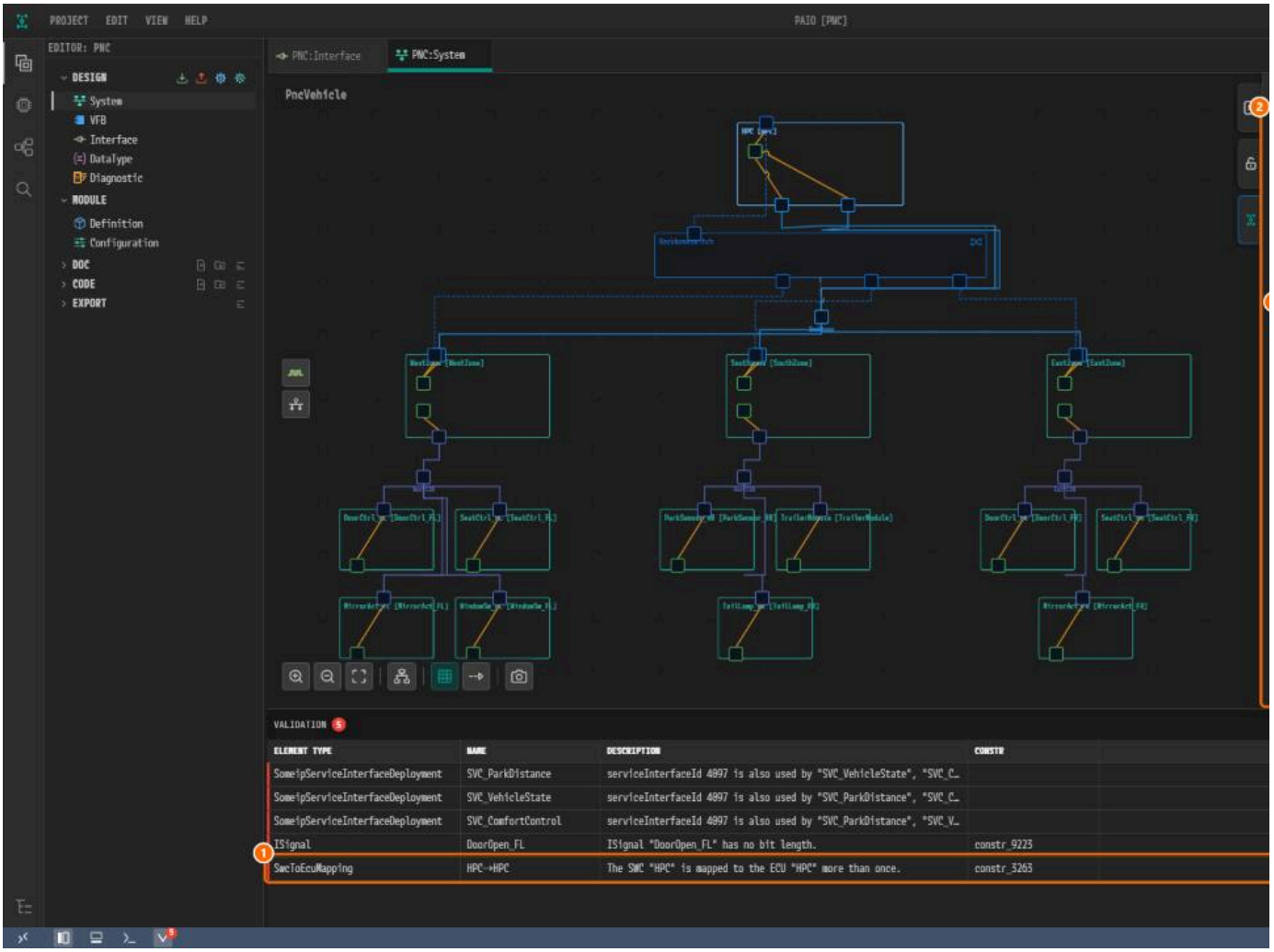


① VFB 에디터가 포트 `Sensor` 를 선택한다. ② 대상이 없는 `PORT INTERFACE` 필드가 주황으로 표시된다.



다이어그램 에디터가 아닌 곳에서도 같다. ① 이 행을 누르면 인터페이스를 선택한 채 `PNC:Interface` 탭이 열린다. ② `SERVICE ID` 필드만 빨갛게 표시된다.

한 번에 한 요소만 표시된다. 표시할 필드가 하나로 정해지지 않는 행은 에디터가 요소를 선택하기만 한다. 예를 들어 SWC↔ECU 매핑에 관한 행은 System 에디터의 SYSTEM 패널을



① `SwcToEcuMapping` 행. ② SYSTEM 패널. ③ **ECU MAPPING**. 표시된 필드는 없다. *11. SYSTEM 패널*을 참고한다.

이동할 수 있는 에디터

열리는 에디터는 요소가 속한 영역에 따라 다르다. 예를 들어 데이터 타입은 DataType 에디터, 포트 인터페이스는 Interface 에디터, SWC 와 Machine 은 VFB 에디터, 노드와 커넥 열린다.

클릭해도 이동하지 않는 행

ECUC 행(`EcucParameterValue` 등), `RuleFailure` 행, 그룹 헤더 행은 에디터를 열지 않는다.

검증을 통과해도 내보낸 ARXML 이 스키마에 맞는다는 뜻은 아니다. 13. 임포트 · 익스포트를 참고한다.

14.6 필수 참조와 깨진 참조

모든 에디터와 다이얼로그의 참조 필드는 두 가지 상태를 표시하고, 검증 패널도 같은 두 가지를 보고한다.

- 설계상 필수인 참조가 비어 있음: 필드가 빨간색이고, 패널은 **error** 로 보고한다.
- 값이 존재하지 않는 요소를 가리킴: 필드가 주황색이고 `Referenced item not found` 가 나오며, 패널은 **warning** 으로 보고한다.

참조 필드의 지우기 버튼(×)은 화면만이 아니라 모델의 값을 비운다.

AI 어시스턴트

이 장에서 다루는 것

PAIO 의 AI 채팅은 자연어로 모델을 만들고 고친다. AI 채팅이 무엇을 해 주는지, 요청을 어떻게 쓰는지, AI 가 쓴 결과를 어떻게 확인하고 되돌리는지를 설명한다.

- 대상 독자: 자연어로 모델을 만들거나 고치려는 사용자
- 선행 조건: 활성 라이선스와 LLM 프로바이더 하나 이상(15.2). 모델을 고치려면 창에 프로젝트가 열려 있어야 한다.

> **AI 가 만들고, 엔지니어가 결정한다.** AI 가 내놓는 것은 언제나 초안이다. PAIO 는 형식이 깨진 결과를 막고 되돌릴 수단을 주지만, 설계로서 옳은지는 판단하지 않는다. 결과는 각 에

15.1 무엇을 해 주나

PAIO 의 AI 기능은 두 가지다. 연결하고 선택한 프로바이더를 쓰는 **AI 채팅**(이 장)과, 항상 Claude 를 쓰는 **AI 워크플로우**의 AI 노드다(21. AI 워크플로우). AI 채팅에서 고른 프로바이더

AI 채팅은 요청한 모델 부분만 바꾼다. 손댈 수 있는 것은 다음과 같다.

- 네트워크 토폴로지: ECU, HPC, 통신 클러스터, 커넥터
- 시그널, PDU, 프레임
- 데이터 타입과 단위
- 포트 인터페이스와 서비스 배포
- SWC 와 컴포지션, SWC ↔ ECU 매핑
- ECUC 모듈 정의와 설정값
- AI 워크플로우 그래프: 노드, 연결, 노트, 설정(21. AI 워크플로우)

모델을 고치지 않는 질문도 할 수 있다.

15.2 준비 — 프로바이더 연결

1. 채팅 헤더의 톱니 버튼(툴팁 `Manage AI accounts`)을 누른다. **AI Providers** 다이얼로그에 프로바이더 5종이 나온다.
2. PAIO 밖에서 프로바이더를 준비한다. 예를 들어 터미널에서 `claude auth login` 으로 Claude CLI 에 로그인한다. PAIO 자체에는 로그인 버튼이 없다.
3. 사용 가능한 프로바이더의 **Use** 를 누르면 활성이 된다. 활성 프로바이더에는 **ACTIVE** 배지가 붙는다.



*① 상태 점. 초록은 사용 가능, 회색은 사용 불가다. ② **ACTIVE** 가 붙은 활성 프로바이더. ③ 상태 줄. ④ 사용 가능한 프로바이더의 **Use**. ⑤ **Set API key**. ⑥ 연결할 수 있는 Ollama 서*

15.2.1 Claude 모델과 Effort

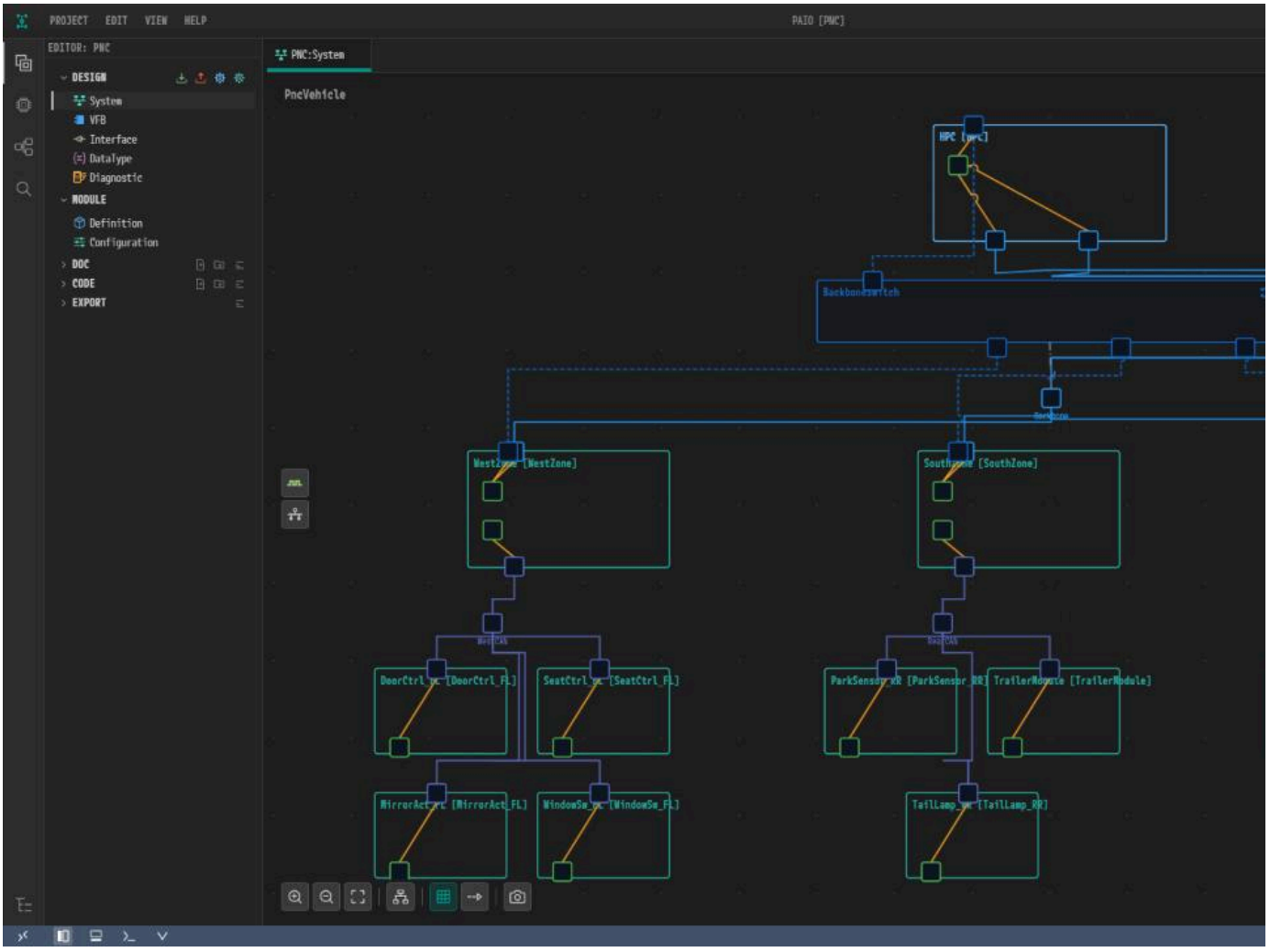
Claude [OPUS] 행을 캐럿으로 펼치고 **MODEL** 과 **EFFORT** 를 고른다. 기본값은 **Opus** 와 **Medium** 이다. 바꾼 값은 다음 메시지부터 적용된다.



*실행 설정이다. **MODEL** 은 **Opus**, **EFFORT** 는 **Medium** 이다.*

15.3 AI 채팅 쓰기

1. 상단바 오른쪽의 로봇 버튼(툴팁 `AI Chat Panel`)을 누른다. 오른쪽에 채팅 패널 **PAIO Robo** 가 열린다.
2. 요청을 쓰고 `Enter` 를 누른다. `Shift+Enter` 는 줄을 바꾼다.
3. 답변을 중단하려면 **Stop response** 를 누른다.

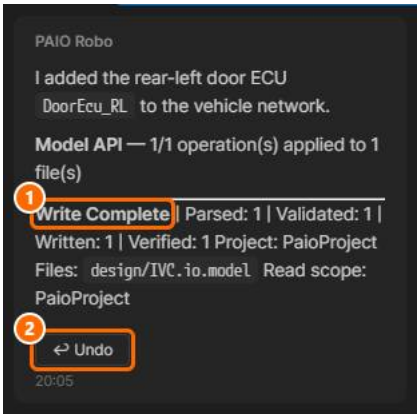


*① 로봇 버튼. ② 제목 **PAIO Robo**. ③ 답변 언어. ④ 지금 쓰는 프로바이더. ⑤ 입력란. ⑥ 보내기 버튼.*

언어 선택에는 **KO**, **EN**, **JA**, **ZH** 가 있으며, 고른 언어로 답한다.

15.3.1 기본 흐름

1. 프로젝트를 열어 둔 상태에서 요청을 쓴다.
2. 답변을 읽는다. 모델을 고치는 요청이었다면 결과가 파일에 바로 반영되고, 답변 끝에 요약(**Model API**, **Write Complete**, `Project:`, `Files:`)이 붙는다.
3. 각 에디터를 열어 결과를 확인한다. 상태바의 **V** 를 눌러 검증 패널도 함께 확인한다.
4. 마음에 들지 않으면 되돌린다(15.3.3).



*① 개수와 쓴 파일이 붙은 **Write Complete**. ② **↩ Undo** 버튼.*

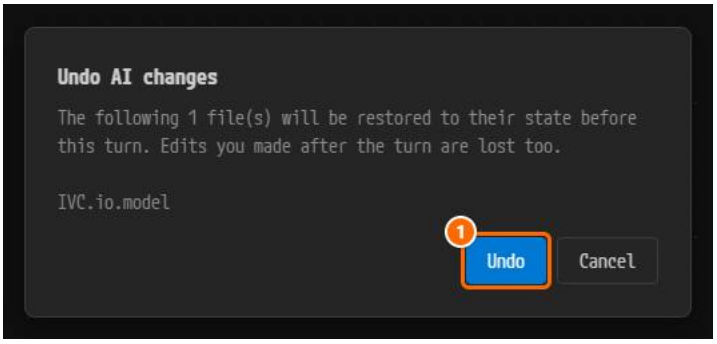
이름이 비었거나 맞지 않는 문서, 구조가 맞지 않는 문서는 거부되며, PAIO 가 최대 2회까지 자동으로 다시 요청한다.

15.3.2 어느 프로젝트에 쓰이나

AI 는 어떤 에디터 탭이 열려 있든 그 창에 열린 프로젝트에 쓴다. 요약의 `Project:` 줄이 실제로 쓴 프로젝트를 알려 준다. 프로젝트가 열려 있지 않으면 아무것도 쓰지 않는다.

15.3.3 되돌리기

1. 답변 아래의 **↩ Undo** 를 누른다.
2. 확인 창 **Undo AI changes** 가 복원될 파일을 보여 준다. 그 답변 뒤에 직접 고친 내용도 함께 사라진다.
3. **Undo** 를 누른다.



*① **Undo** 는 파일을 복원하고, **Cancel** 은 그대로 둔다.*

`Ctrl+Z` 로도 되돌릴 수 있다. AI 가 쓴 변경은 실행 취소 한 단계다. **↩ Undo** 가 쓰는 백업은 30분 동안만 보관되며, PAIO 를 다시 켜면 남지 않는다.

15.4 요청을 잘 쓰는 법

결과의 품질은 대부분 요청 문장에서 갈린다. 네 가지만 지키면 된다.

대상을 이름으로 지목한다

AI 는 화면에서 무엇을 선택했는지 보지 못한다. 무엇을 어디에 만들지 문장으로 알려 준다. 예: CAN 클러스터 `CAN_Body` 에 `DoorEcu_RL` 이라는 도어 ECU 를 추가해 줘.

한 번에 한 가지만 요청한다

여러 관심사가 담긴 요청은 나눠서 하고, 각각을 확인한 뒤 다음으로 넘어간다.

조건을 함께 준다

개수, 비트 길이, 주기, 플랫폼처럼 정해진 값은 미리 적는다. 적지 않으면 AI 가 임의로 고른다.

확신이 없으면 먼저 물어본다

모델을 고치라고 하지 않고 묻기만 하면 파일은 그대로 있다.

15.5 QuickStart 워크플로

QuickStart 는 PAIO 1.0.1 에서 제공하지 않는다. 백지에서 시작하려면 **New Project** 로 프로젝트를 만들고(4.3.1), 에디터에서 모델을 만들거나 AI 채팅으로 한 단계씩 요청한다(15.

15.6 알아 둘 한계

- 활성 라이선스가 필요하다(20.8 참조).
- 대화 내용과 모델 내용은 선택한 프로바이더로 전송된다. Ollama 는 로컬 서버로만 통신한다.
- 큰 프로젝트는 전부 전달되지 않으므로 대상을 구체적으로 적는다.
- 생성 결과는 초안이며 반드시 사람이 검토한다.
- AI 워크플로우의 AI 노드는 항상 Anthropic Claude 로 전송한다(21).

15.7 긴 요청, 중지, 링크

15.7.1 여러 라운드로 진행되는 요청

VFB 전체 설계처럼 큰 요청은 여러 라운드에 걸쳐 쓰인다. PAIO 가 나머지를 스스로 이어서 요청하며, 요청 하나당 최대 10라운드까지 이어 간다. 각 라운드는 `[Continuing <n>/1` 답한다. AI가 쓴 내용이 새 검증 결과를 만들면 PAIO가 그것을 고치도록 AI에 요청하며, 남은 것은 응답에 나열된다.

15.7.2 요청 중지

Stop response 를 누르면 그 순간부터 더 쓰지 않는다. 이미 쓰인 라운드는 파일에 남는다. 응답 아래의 **↩ Undo** 는 모든 라운드를 포함해 요청 전체를 되돌린다.

15.7.3 응답 안의 링크

프로젝트 파일 링크는 Explorer 에서 더블클릭한 것처럼 파일을 연다. `http`, `https` 주소는 시스템 브라우저에서 열린다.

15.8 SSH 창에서의 AI 채팅

SSH로 연결된 창(17. 원격 개발)에서는 AI 채팅이 PC가 아니라 SSH 서버에 쓴다.

1. 평소처럼 요청을 보낸다. PAIO가 서버에서 프로젝트를 복사하는 동안 응답 영역에 `Reading the project from the SSH server…` 가 표시된다.
2. AI가 쓰는 모든 파일은 서버에 먼저 쓰인다. **↩ Undo** 와 `Ctrl+Z` 는 서버의 파일을 되돌린다.
3. 대화는 서버 프로젝트에 저장된다.

코드 편집 · 빌드 · 터미널 · 문서

이 장에서 다루는 것

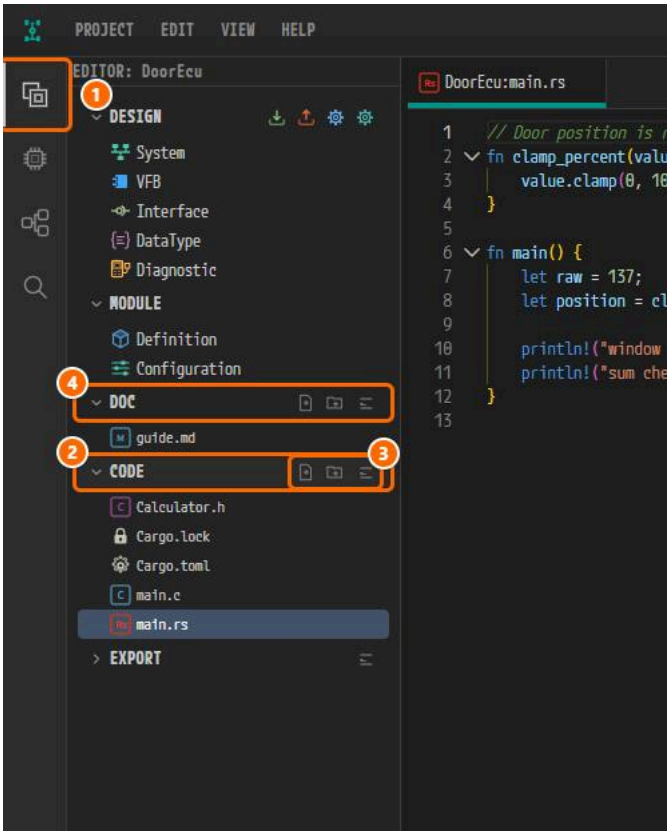
일반 개발 도구를 다룬다. 프로젝트의 `code` 폴더에 있는 C, C++, Rust 소스를 편집하고 빌드 · 실행하는 방법, 통합 터미널, `doc` 폴더의 Markdown 문서 편집기다.

- 대상 독자: PAIO 안에서 애플리케이션 코드를 직접 작성하고 빌드하는 개발자
- 선행 조건: 프로젝트가 열려 있는 것
- 원격(SSH 호스트, vECU 컨테이너)에서 작업한다면 17. 원격 개발 (SSH · vECU 컨테이너)을 함께 본다.

16.1 코드 에디터 (C / C++ / Rust)

16.1.1 어떤 파일이 열리나

1. 왼쪽 레일 맨 위 아이콘(툴팁 **Design**)을 눌러 **EDITOR** 뷰를 연다.
2. **CODE** 섹션(`code/` 의 파일)이나 **DOC** 섹션(`doc/` 의 파일)을 펼친다.
3. 파일을 더블클릭해 연다.

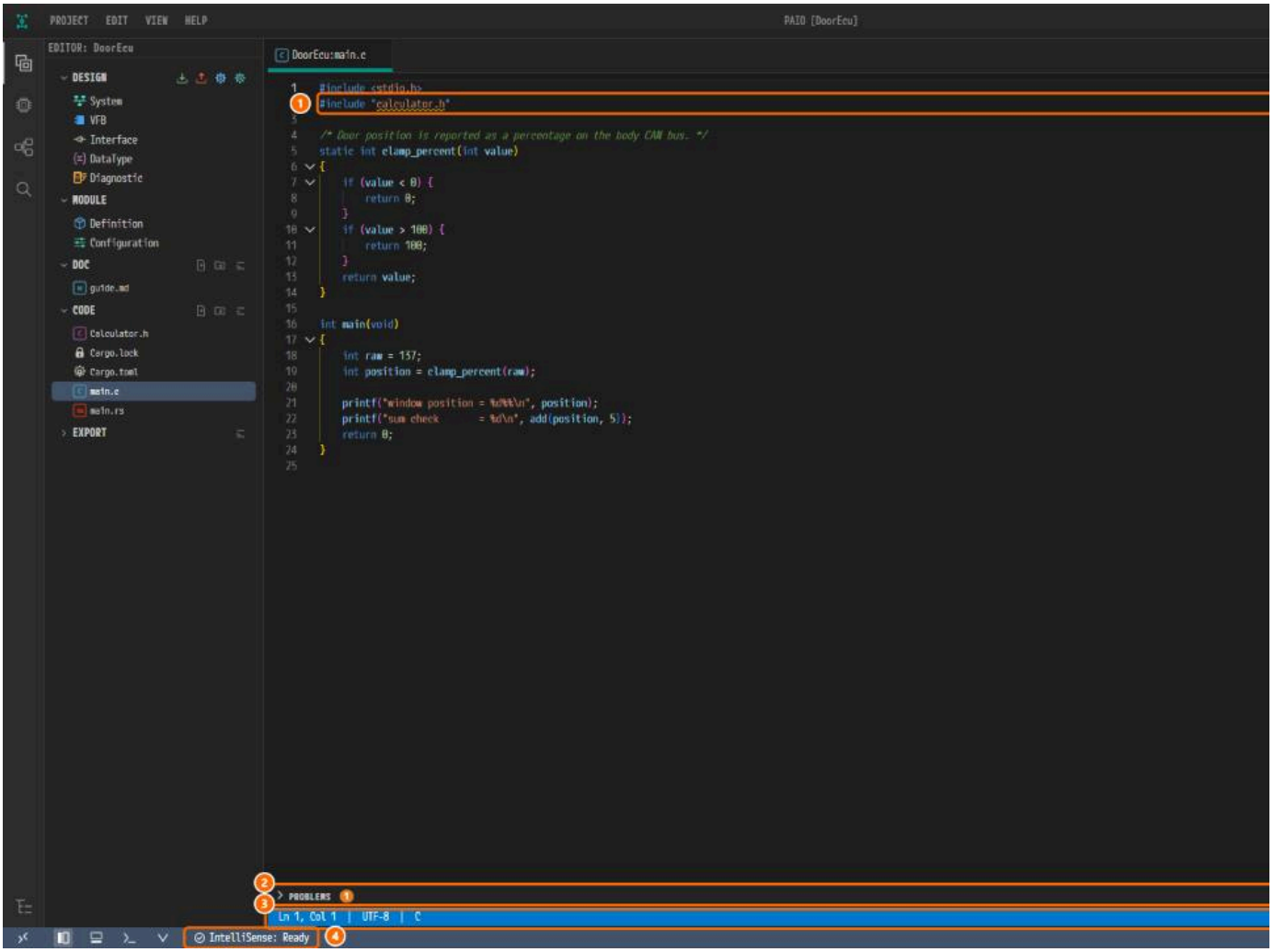


*① 레일 아이콘, ② **CODE** 머리글, ③ **New File, New Folder, Collapse All**, ④ **DOC** 머리글.*

어느 폴더에 있든 확장자만으로 편집기가 정해진다.

16.1.2 화면과 단축키

편집 영역 아래에 **PROBLEMS** 패널이 있다. 빌드를 시작하면 그 아래에 **Output** 패널이 나타난다.



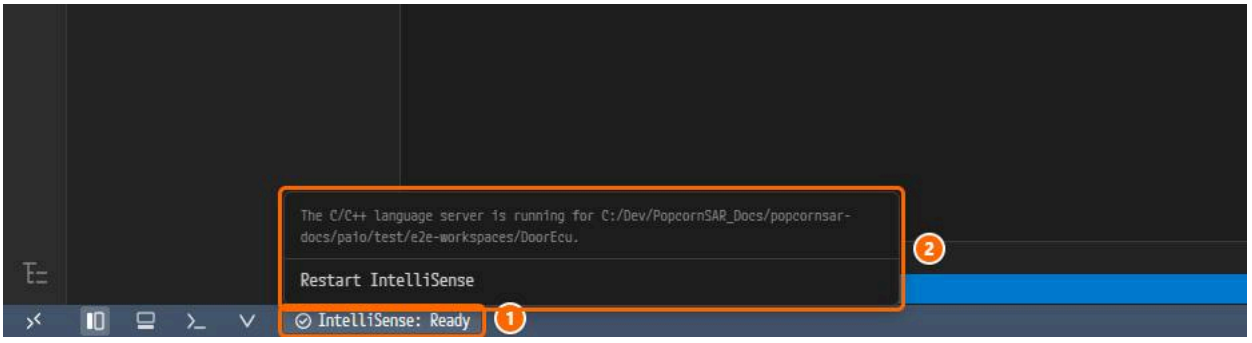
*① 2행의 경고 물결선, ② **PROBLEMS**, ③ 에디터 바, ④ `IntelliSense: Ready`.*

단축키	동작
`Ctrl+B`	**Build**: 현재 파일 하나만 컴파일한다
`Ctrl+F5`	**Build & Run**: 프로젝트 형태를 판별해 빌드하고 실행한다
`Ctrl+S`	저장
`F12`	정의로 이동

16.1.3 언어 서버 (LSP)

자동 완성, 호버, 정의로 이동, 서식 정리는 언어 서버가 제공한다. C/C++는 clangd, Rust는 rust-analyzer이며 둘 다 앱에 함께 들어 있다. 이름 바꾸기(`F2`)와 quick fix는 제공되

C나 C++ 파일을 열면 상태바에 **IntelliSense** 항목이 나타난다. 자동 완성과 진단이 조용히 동작하지 않으면 이 항목을 누르고 **Restart IntelliSense** 를 고른다.

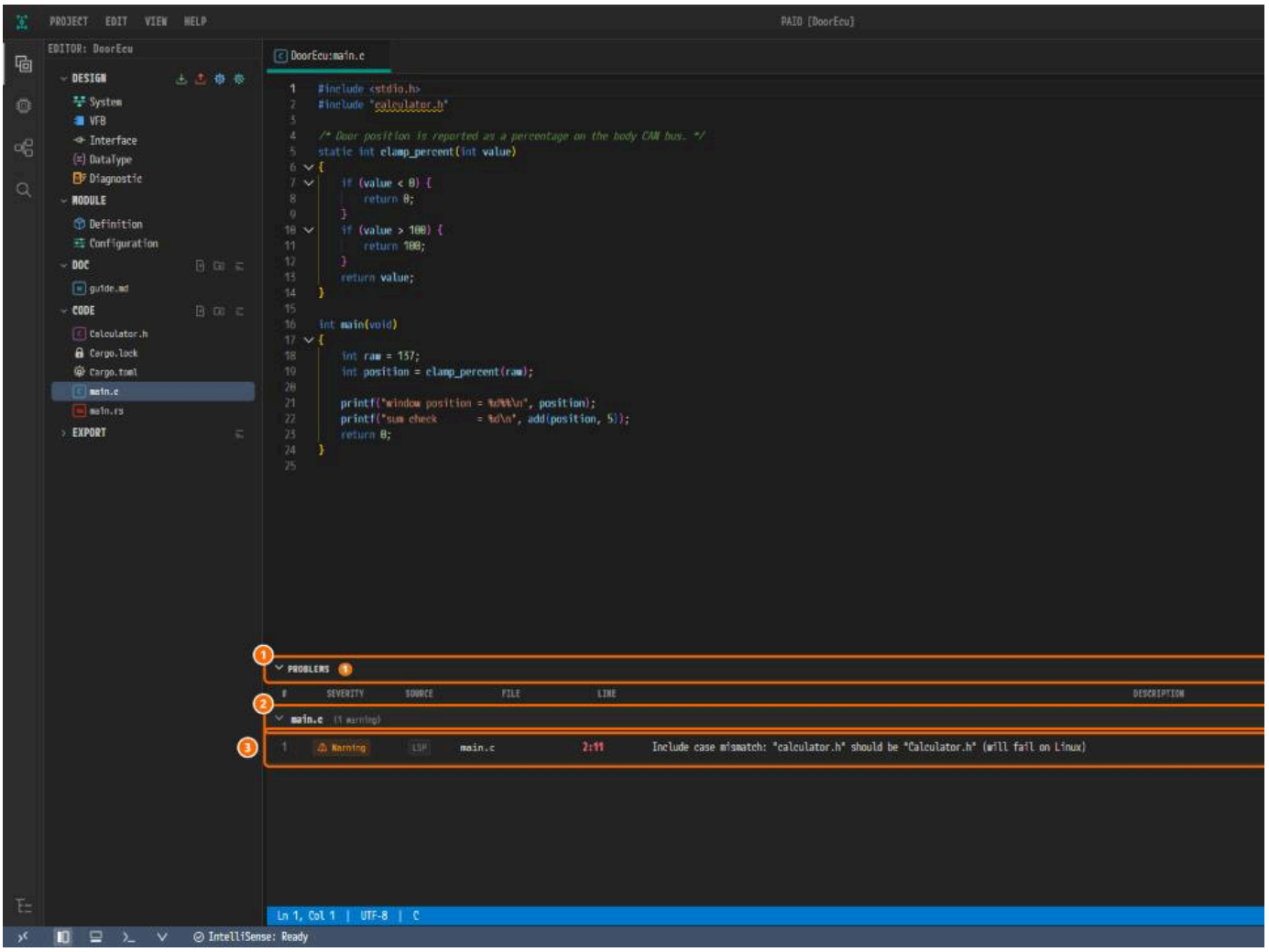


① `IntelliSense: Ready`, ② ***Restart IntelliSense*** *가 있는 메뉴.*

`code/Cargo.toml` 은 자동으로 만들어지고 덮어써지므로, 손으로 관리하는 Cargo 프로젝트는 하위 폴더에 둔다.

16.1.4 PROBLEMS 패널

파일별로 묶인 진단 목록이다. 항목을 누르면 그 위치로 이동한다.



① 개수가 있는 머리글, ② 파일 묶음, ③ 항목.

코드 에디터의 **Build** 와 **Build & Run** 오류는 여기에 오지 않고 **Output** 패널에 나타난다. **Build ECU** 의 컴파일러 진단은 `BUILD` 태그를 달고 여기에 나온다.

16.1.5 저장과 외부 변경

`Ctrl+S` 로 저장한다. 저장하지 않은 편집이 있는 파일이 디스크에서 바뀌면 에디터는 편집 내용을 지키고 경고를 띄운다. 이어서 저장하면 디스크의 변경 위에 덮어쓴다.

16.2 빌드와 실행

Build 와 **Build & Run** 은 열린 파일, 또는 그 파일이 속한 CMake · Cargo 프로젝트를 그 창의 환경에서 빌드한다. ECU는 그 vECU에서 **Build ECU** 로 빌드한다. 17.9를 참고한다.

16.2.1 Build와 Build & Run은 다르다

- `Ctrl+B` (**Build**)는 현재 열려 있는 파일 하나만 gcc / g++로 컴파일한다.
- `Ctrl+F5` (**Build & Run**)가 프로젝트 단위 빌드다.

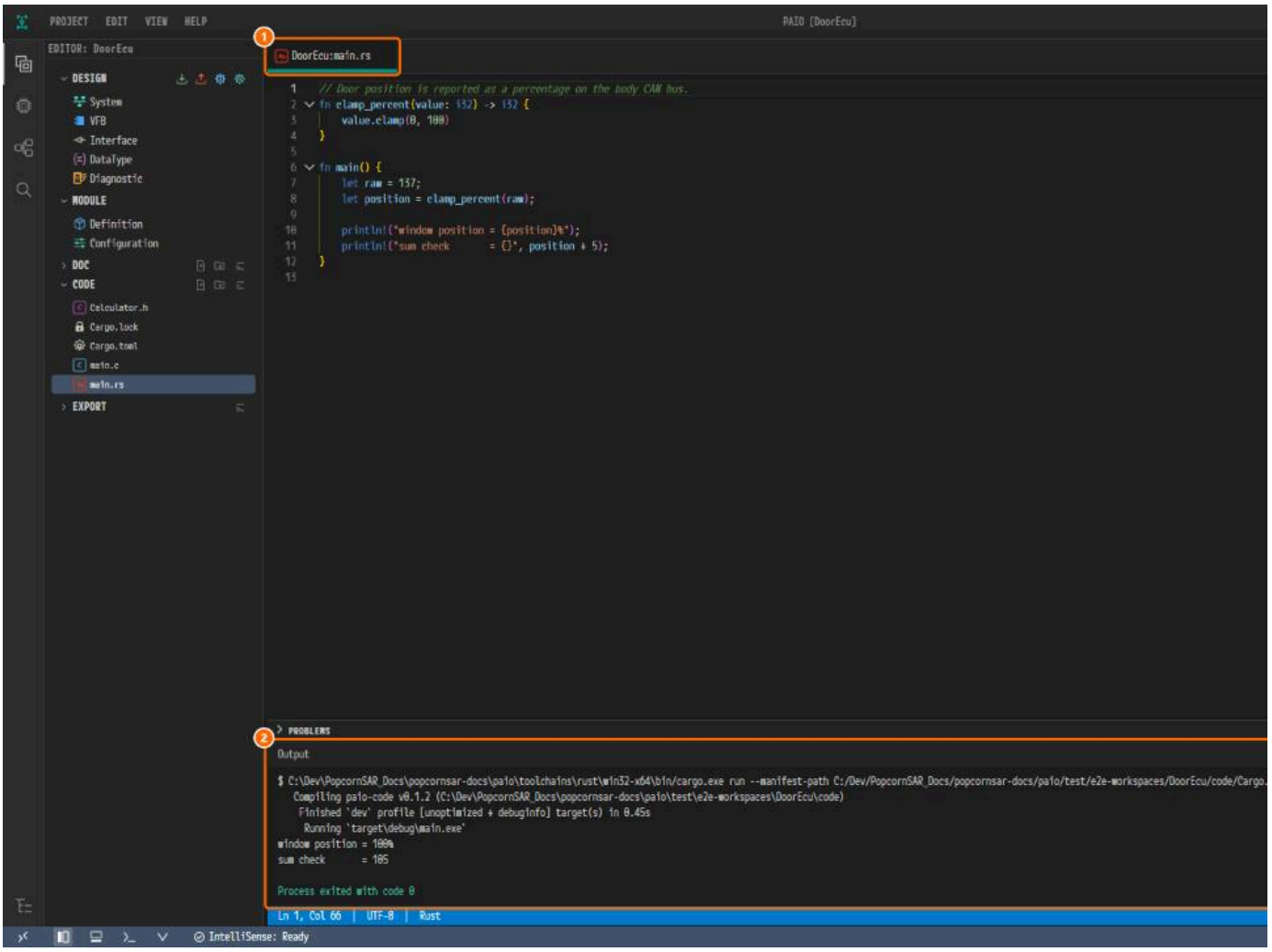
둘 다 저장하지 않은 편집을 먼저 저장한다.

16.2.2 Build & Run이 판별하는 순서

현재 파일의 폴더에서 위로 올라가며 먼저 맞는 것을 쓴다. `CMakeLists.txt` (구성과 빌드, 실행은 하지 않음), `Cargo.toml` (`cargo run`), `.c` / `.cpp` 단일 파일(컴파일한 뒤 실

16.2.3 실행되는 명령과 출력

빌드 타입은 `Debug` 로 고정이다. **Output** 패널에 실행한 명령, 실시간 결과, `Process exited with code <숫자>` 가 나온다.



*① `Ctrl+F5` 를 누른 탭, ② **Output** 패널.*

진행 중인 빌드를 중단하는 버튼은 없다.

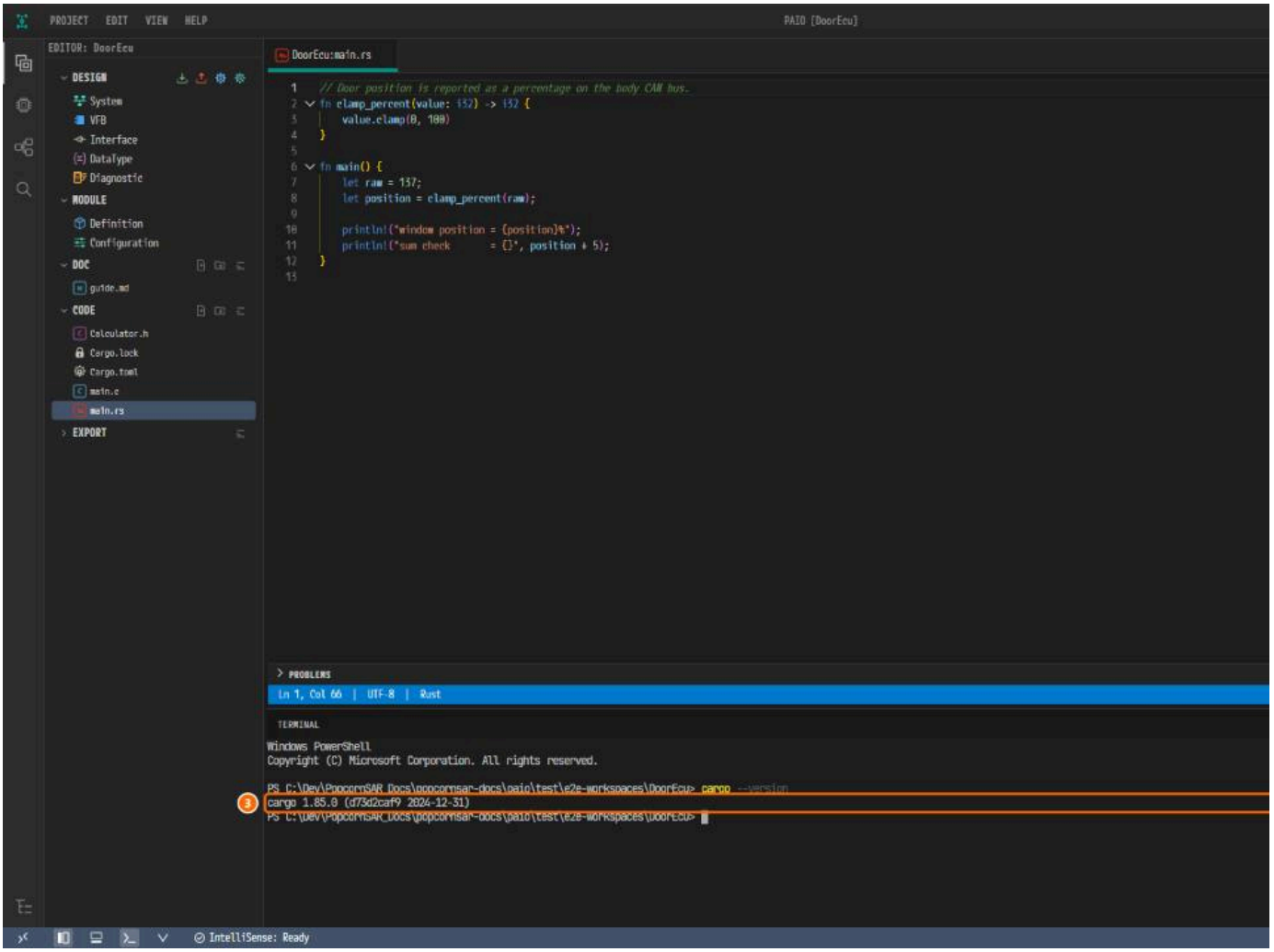
16.2.4 사용하는 툴체인

Rust 툴체인은 앱에 함께 들어 있다. 번들 도구가 있으면 그것을 먼저 쓰고, 없으면 시스템 `PATH` 의 도구로 넘어간다.

16.3 통합 터미널

16.3.1 여는 방법과 조작

1. `Ctrl+`` 로 터미널 패널을 보이거나 숨긴다. 새 터미널은 프로젝트 폴더에서 시작한다.
2. 특정 폴더에서 시작하려면 **Explorer** 에서 그 폴더를 우클릭해 **Open in Integrated Terminal** 을 고른다.



① ***Decrease, Increase, New Terminal, Split Terminal, Kill Terminal, Close Panel.*** *② 터미널 목록. ③ 터미널에서* `cargo --version` *이 답한 줄.*

16.3.2 어떤 셸이 열리나

vECU 컨테이너 창은 컨테이너 안의 셸을, SSH 창은 원격 SSH 셸을 연다. 로컬 기본 셸은 Windows에서 PowerShell이다.

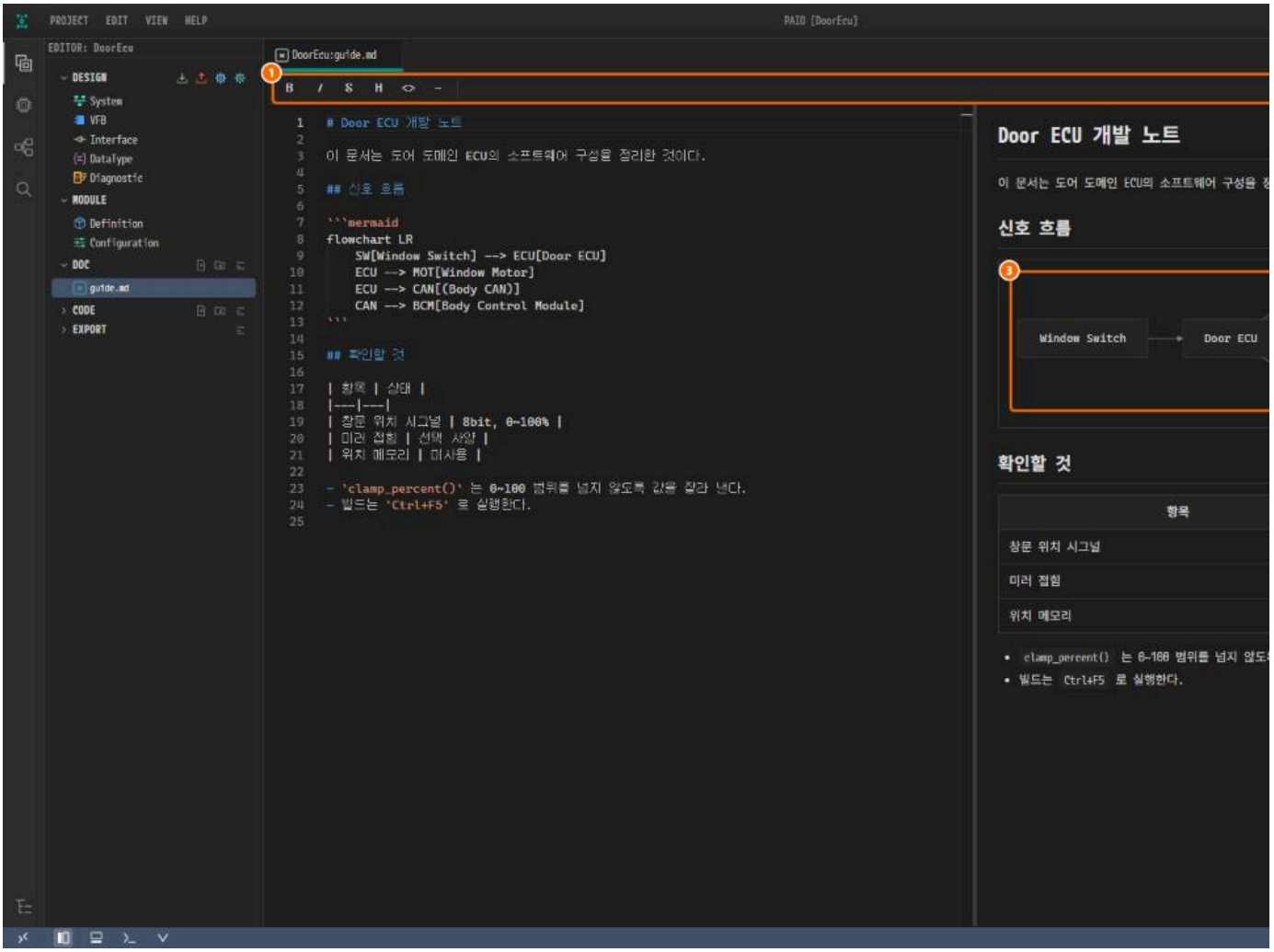
16.3.3 번들 툴체인과 PATH

로컬 터미널의 `PATH` 에는 번들 툴체인 경로가 들어 있어 `cargo` 를 바로 입력할 수 있다. SSH 셸에는 적용되지 않는다. 원격에서 쓰는 빌드 도구는 17.6에서 다룬다.

16.4 Markdown 문서 에디터

16.4.1 편집과 미리보기

`.md` 파일은 Markdown 에디터로 열리고, 옆에 실시간 미리보기가 열린다.



① 서식 툴바, ② 보기 버튼 세 개, ③ 미리보기에 그려진 `mermaid` 블록.

굵게와 기울임은 툴바 버튼으로 적용한다. 여기서 `Ctrl+B` 와 `Ctrl+I` 는 서식을 적용하지 않는다.

16.4.2 다이어그램

코드 블록의 언어를 `mermaid`, `drawio`, `zenuml`, `plantuml` 중 하나로 지정하면 미리보기에 그림으로 그려진다. PlantUML만 인터넷 연결이 필요하다.

16.5 그 밖의 뷰어와 편집기

확장자로 편집기가 정해진다. HTML 미리보기, 이미지 뷰어, XML 에디터, PAIO 자체 문서의 읽기 전용 JSON 보기, 그 밖의 텍스트 파일용 텍스트 에디터가 있다. 설계 편집기는 **Explo MODULE** 섹션에서 연다.

원격 개발 (SSH · vECU 컨테이너)

이 장에서 다루는 것

로컬 파일 시스템에 있지 않은 프로젝트를 다루는 방법을 설명한다. 대상은 **SSH 호스트**, 그리고 로컬 Docker에서 실행되는 프로젝트의 **vECU 컨테이너** 두 가지다. **PRTE vECU** 는 AU
는 AUTOSAR Adaptive Platform용 컨테이너다.

- 창 하나가 프로젝트 하나를 가진다. 연 원격 폴더가 그 창의 프로젝트가 된다.
- 프로젝트가 없는 창은 그 자리에서 연결한다. 이미 프로젝트가 있는 창은 프로젝트를 그대로 두고, 연결은 별도의 창으로 열린다.
- SSH는 접속 가능한 서버가 필요하다. vECU는 로컬에 설치되어 실행 중인 Docker와, 이 컴퓨터에 받아 둔 PRTE vECU 또는 PARA vECU 이미지가 필요하다.
- 코드 편집기, 빌드 명령, 터미널 자체의 사용법은 16. 코드 편집 · 빌드 · 터미널 · 문서에 있다.

17.1 진입점

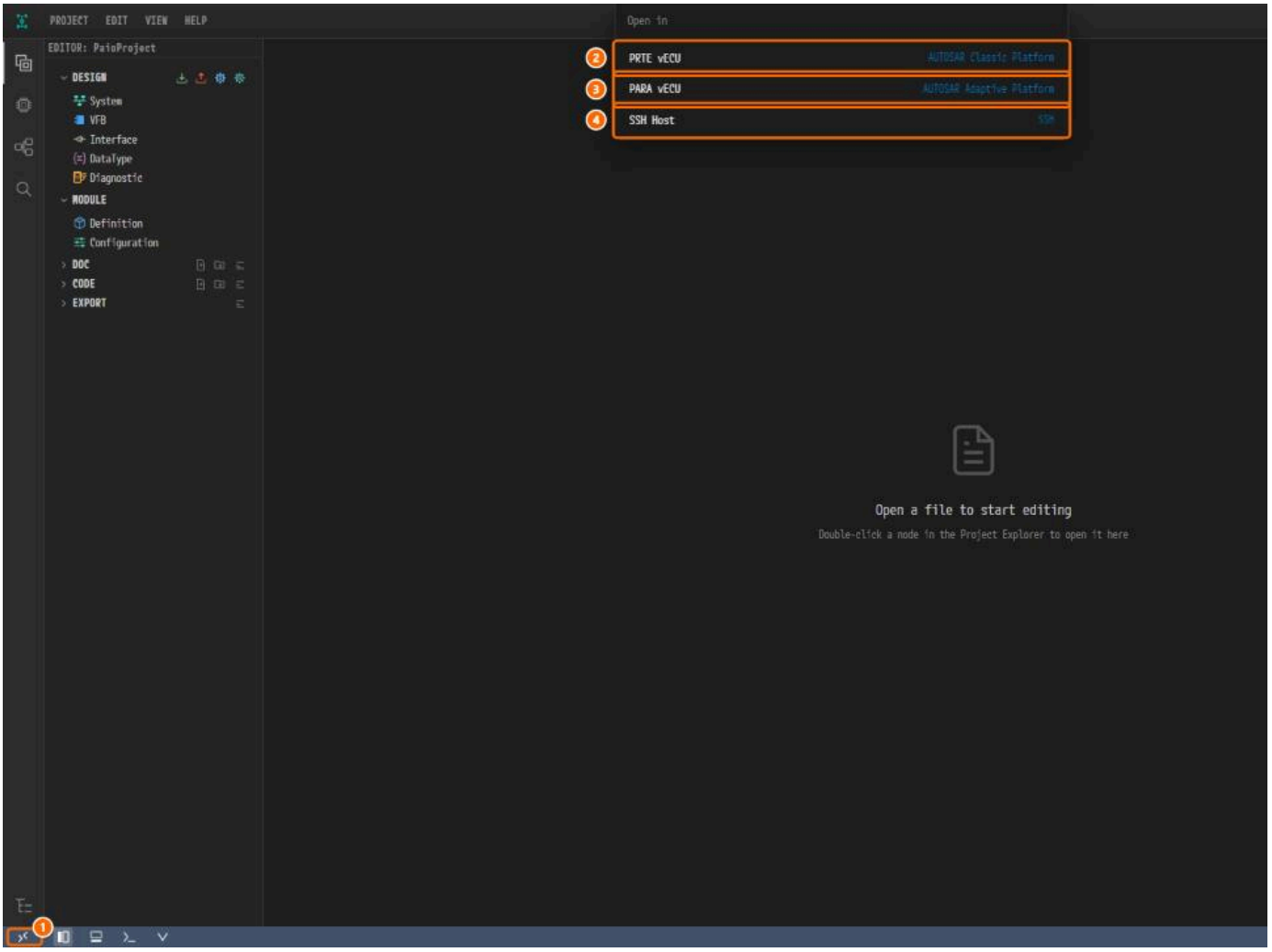
원격 연결은 상태바 왼쪽 끝의 `><` 표시기, **Welcome** 탭의 **Connect to...** 버튼, 명령 팔레트(SSH만), 그리고 vECU 컨테이너의 경우 왼쪽 레일의 **vECU** 항목에서 시작한다.

17.1.1 원격 표시기

상태바 가장 왼쪽 항목이 `><` 표시기다. 연결 상태를 보여 주며, 연결되면 초록 배경에 `SSH: <user>@<host>` 또는 `Container: <이름> (<이미지 태그>)` 가 표시된다. 클릭히

17.1.2 연결되지 않은 상태의 메뉴

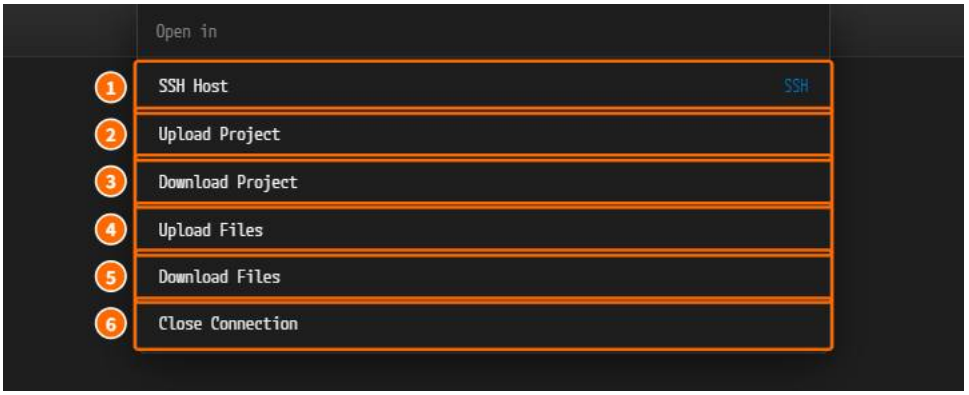
메뉴에는 **PRTE vECU**, **PARA vECU**, **SSH Host** 세 항목이 있다. 두 vECU 항목은 프로젝트의 컨테이너를 연다(17.3.2). **SSH Host** 는 SSH 빠른 연결 입력 창을 연다(17.2.1).



*① `><` 표시기. ② **PRTE vECU**, ③ **PARA vECU**, ④ **SSH Host**.*

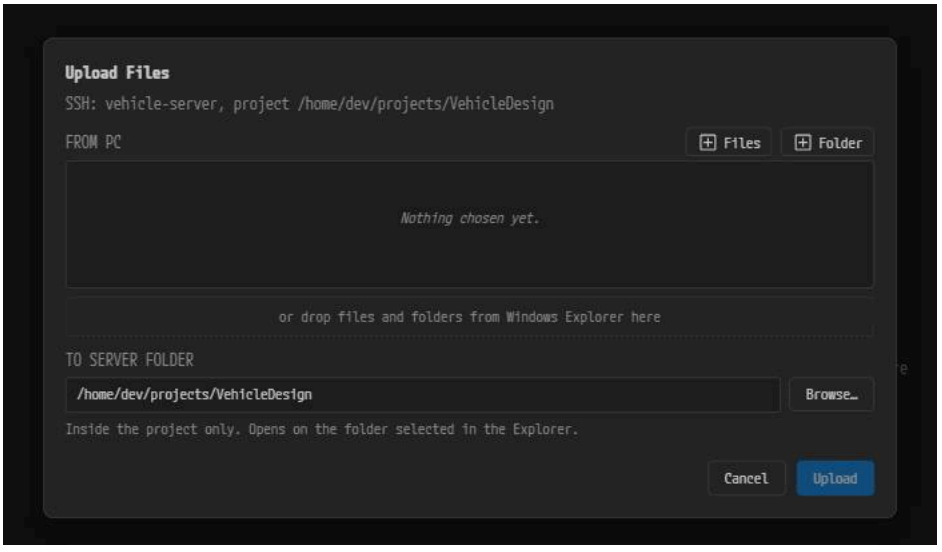
17.1.3 연결된 상태의 메뉴

SSH로 연결된 창의 메뉴에는 **SSH Host**, **Upload Project**, **Download Project**, **Upload Files**, **Download Files**, **Close Connection** 이 있다. vECU 컨테이너 창의 메뉴에는 **Clo**

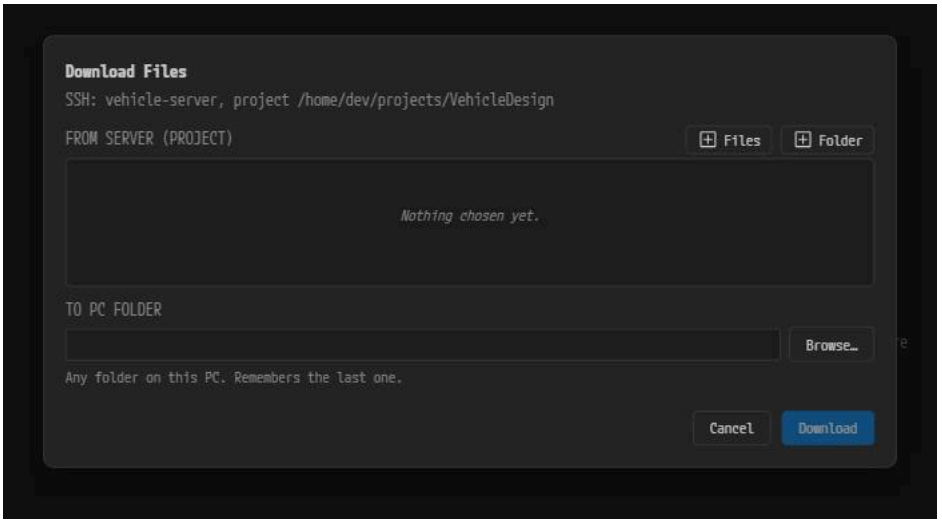


*① **SSH Host**, ② **Upload Project**, ③ **Download Project**, ④ **Upload Files**, ⑤ **Download Files**, ⑥ **Close Connection**.*

- **Upload Project** 는 창의 서버 폴더를 PC의 프로젝트로 바꾸고, **Download Project** 는 서버 프로젝트를 PC 폴더로 복사한다. 이미 있는 내용은 확인을 받은 뒤에만 바꾼다.
- **Upload Files** 와 **Download Files** 는 고른 파일과 폴더를 복사한다. **+ Files** 와 **+ Folder** 로 고르고, **Browse…** 로 대상 폴더를 정한 뒤 **Upload** 나 **Download** 를 누른다. Win
  다.
- 같은 이름의 파일이 있으면 바꿀지 건너뛸지 묻는다.
- 서버 쪽은 프로젝트 폴더 안으로 제한된다.



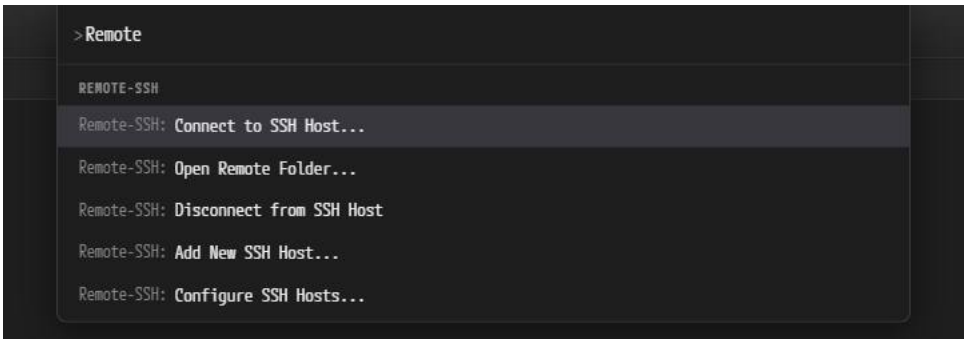
*Upload Files 창. **FROM PC** 는 복사할 파일 목록, **TO SERVER FOLDER** 는 대상 폴더다.*



*Download Files 창. **FROM SERVER (PROJECT)** 는 복사할 파일 목록, **TO PC FOLDER** 는 대상 폴더다.*

17.1.4 명령 팔레트

명령 팔레트에는 `Remote-SSH` 분류의 SSH 항목만 있다. vECU 컨테이너는 **vECU** 뷰나 표시기 메뉴에서 연다.

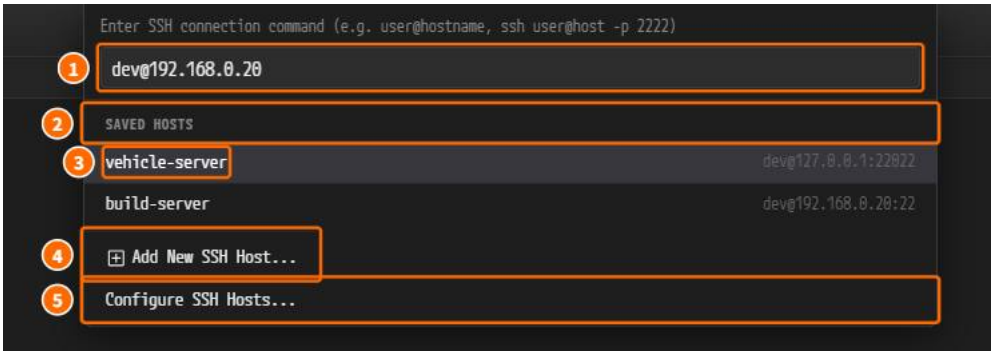


`Ctrl+Shift+P` 를 누르고 `Remote` 를 입력한다. 다섯 항목이 `REMOTE-SSH` 아래에 모여 있다.

17.2 SSH

17.2.1 빠른 연결

1. 표시기 메뉴에서 **SSH Host** 를 고른다. 화면 위쪽에 입력 창이 열린다.
2. `user@host` 나 `user@host:2222` 같은 접속 정보를 입력하고 `Enter` 를 누른다. `SAVED HOSTS` 에서 호스트를 골라도 된다.
3. 프로젝트가 없는 창이면 창이 다시 로드되고 원격 폴더 선택 창이 열린다(17.5). 폴더를 고르고 **OK** 를 누른다.



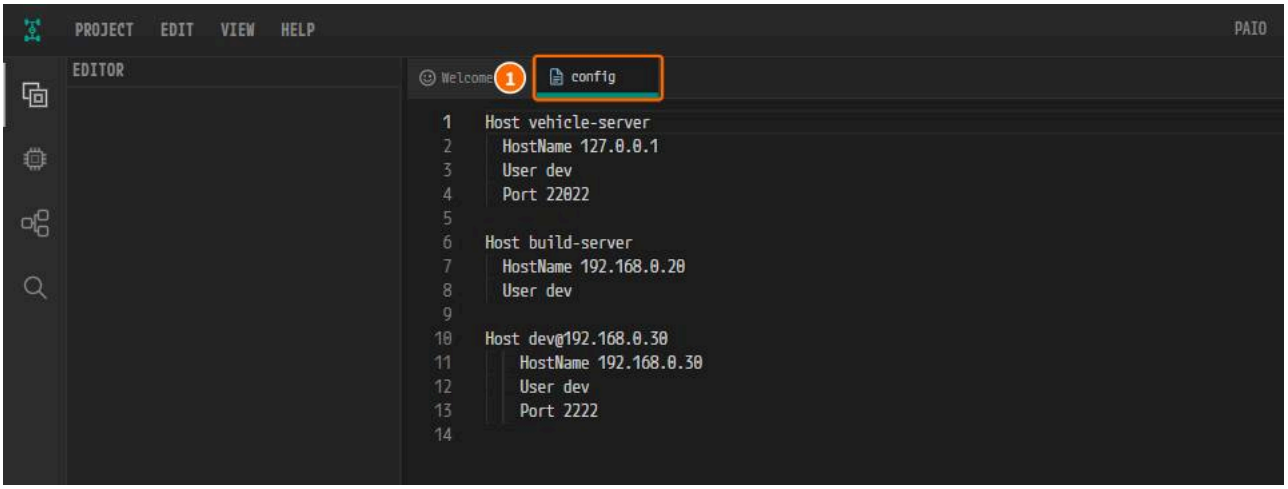
*① 입력 창. ② `SAVED HOSTS`, ③ 저장된 호스트. ④ **Add New SSH Host...**(17.2.3), ⑤ **Configure SSH Hosts...**(17.2.2).*

입력한 정보는 `user@host` 레이블로 사용자의 `~/.ssh/config` 에 저장된다. 같은 대상으로 이미 저장된 호스트는 그대로 쓰고 다시 쓰지 않는다.

17.2.2 호스트 설정

호스트 목록은 사용자의 `~/.ssh/config` 에 있는 것이다. 호스트를 고치거나 지우려면 이 파일을 편집한다.

1. 빠른 연결 입력 창이나 명령 팔레트에서 **Configure SSH Hosts...** 를 고른다.
2. `Select SSH configuration file to update` 아래의 파일을 고른다. 파일이 편집기 탭으로 열린다.
3. 파일을 고치고 `Ctrl+S` 를 누른다.



① 탭 `config`. 호스트마다 `Host` 블록이 하나 있다.

원격 대상에 연결된 창에서는 이 파일이 열리지 않는다. 로컬 창에서 연다.

17.2.3 호스트 추가

1. 빠른 연결 입력 창이나 명령 팔레트에서 **Add New SSH Host...** 를 고른다.
2. `ssh dev@192.168.0.30 -p 2222` 같은 연결 명령을 입력하고 `Enter` 를 누른다. `user@host` 와 `user@host:port` 도 되며, 포트 기본값은 `22` 이다.
3. `Select SSH configuration file to update` 아래의 파일을 고른다. 토스트 `Host added` 가 나타나고, 호스트가 `user@host` 레이블로 저장된다.



① 연결 명령.

이미 있는 레이블은 그대로 두며 토스트 `Host already added` 가 나타난다.

17.2.4 인증이 시도되는 순서

PAIO는 먼저 자격 증명 없이 연결하므로 `~/.ssh` 의 키와 ssh-agent가 시도된다. 실패하면 비밀번호를 묻는다.

17.2.5 호스트 키 검증

PAIO는 호스트 키 지문을 `~/.ssh/known_hosts` 가 아니라 자체 저장소에 기록한다. 처음 보는 호스트는 수락하고 기록한다. 나중에 지문이 바뀌면 연결을 거부하고 `Host Key C`

17.2.6 연결 유지와 자동 재연결

연결이 끊기면 최대 3회까지 자동으로 재연결하며, 그동안 표시기는 `Connecting...` 이다.

17.2.7 연결 실패 메시지

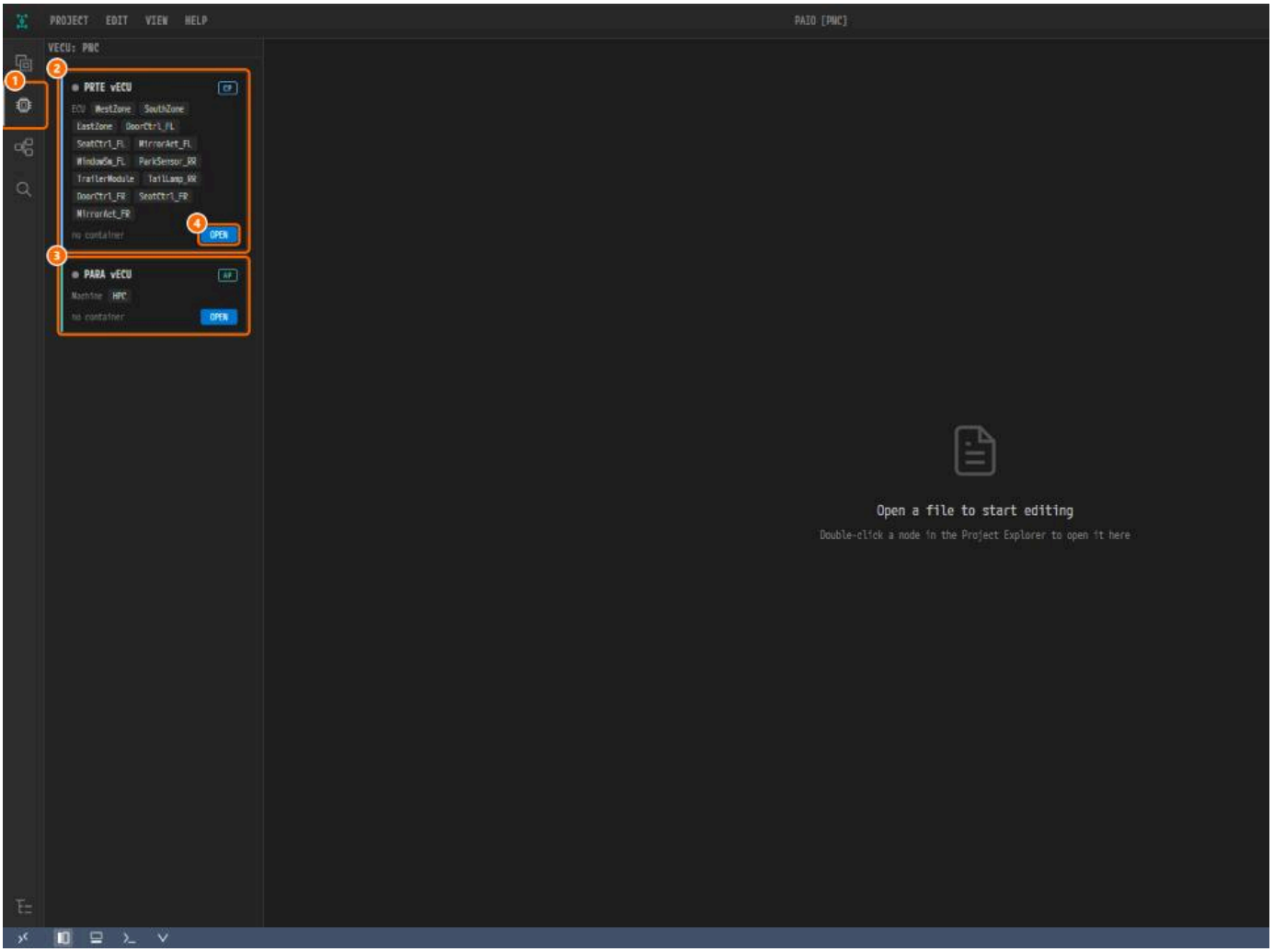
연결에 실패하면 무엇을 확인할지 알려 주는 문구로 바꿔 보여 준다.

17.3 vECU 컨테이너 (Docker)

vECU는 프로젝트의 실행 환경이다. PRTE vECU 또는 PARA vECU 이미지로 만든 Docker 컨테이너이며, 프로젝트의 ECU를 그 안에서 빌드하고 실행한다. 왼쪽 레일의 **vECU** 항목을

- **PRTE vECU** 카드 하나(배지 `CP`)에 프로젝트의 Classic Platform ECU가 표시된다.
- **PARA vECU** 카드(배지 `AP`)는 HPC마다 하나씩 있고, 아직 HPC가 없으면 하나만 있다.

카드마다 컨테이너 상태와 동작 하나, **OPEN** 이 표시된다.



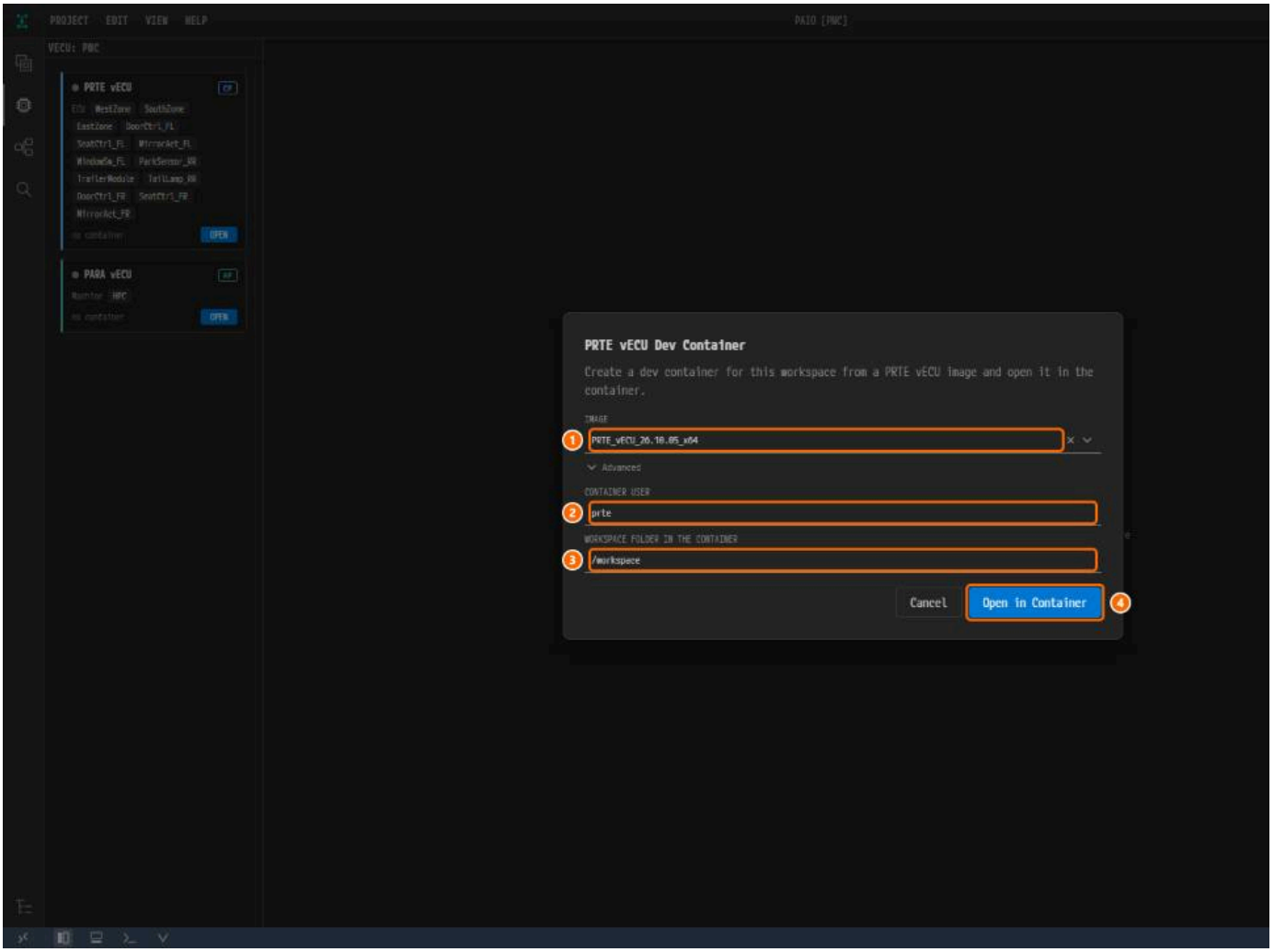
*① 레일의 **vECU** 항목. ② **PRTE vECU** 카드, ③ **PARA vECU** 카드, ④ **OPEN**.*

17.3.1 사전 조건

로컬에 Docker가 설치되어 실행 중이어야 하고, 이 컴퓨터에 PRTE vECU 또는 PARA vECU 이미지가 있어야 한다. 이미지는 `autosar.io/support` 에 로그인해 페이지가 보여 주는다.

17.3.2 vECU 컨테이너 열기

1. 카드의 **OPEN** 을 누른다.
2. 슬롯에 아직 컨테이너가 없으면 `PRTE vECU Dev Container` 대화상자(PARA vECU는 그에 대응하는 대화상자)가 열린다. **Image** 에서 이미지를 고른다.
3. **Open in Container** 를 누른다.
4. PAIO가 컨테이너를 만들고, 프로젝트가 컨테이너 안에서 호스트 창과 별도의 창으로 열린다.



*① **Image**, ② **Container user**, ③ **Workspace folder in the container**(**Advanced** 아래), ④ **Open in Container**.*

슬롯에 이미 컨테이너가 있으면 **OPEN** 은 바로 그 컨테이너로 들어간다.

17.3.3 진행 토스트

진행 상황은 오른쪽 아래 토스트로 표시되며, 그동안 다른 작업을 할 수 있다. **Show log** 는 로그를 보여 주고 **Cancel** 은 작업을 중단한다.

17.3.4 이미 있는 컨테이너로 들어가기

슬롯에 컨테이너가 생긴 뒤에는 그 카드의 **OPEN** 이 대화상자 없이 그 컨테이너로 다시 들어간다. 임의의 실행 중인 컨테이너에 연결하는 기능은 제공하지 않는다.

17.3.5 vECU 설정 파일

vECU 대화상자는 프로젝트의 `.devcontainer/` 아래에 슬롯마다 설정 파일을 하나씩 쓴다.

17.3.6 새 컨테이너로 다시 시작하기

재빌드 명령은 없다. 대신 슬롯의 컨테이너를 제거한다. 워크스페이스 볼륨과 호스트 프로젝트는 남는다.

1. **vECU** 뷰에서 카드 오른쪽 위의 점 세 개 버튼(툴팁 `More`)을 누른다.
2. **REMOVE CONTAINER** 를 고르고 **Remove** 를 누른다.
3. 다음 **OPEN** 이 새 컨테이너를 만든다.

17.3.7 컨테이너 창에서 나오기

Close Connection 은 컨테이너의 작업 내용을 호스트 프로젝트 폴더로 복사하고(충돌하는 파일은 하나씩 묻는다, 17.10), 호스트 프로젝트의 창으로 돌아간다. 컨테이너 창을 X로 닫

17.3.8 컨테이너가 밖에서 멈춘 경우

PAIO 밖에서(예: `docker stop`) 컨테이너가 멈추면 연결 상태가 정리되고 토스트가 표시된다. 컨테이너를 다시 쓰려면 **vECU** 뷰에서 연다.

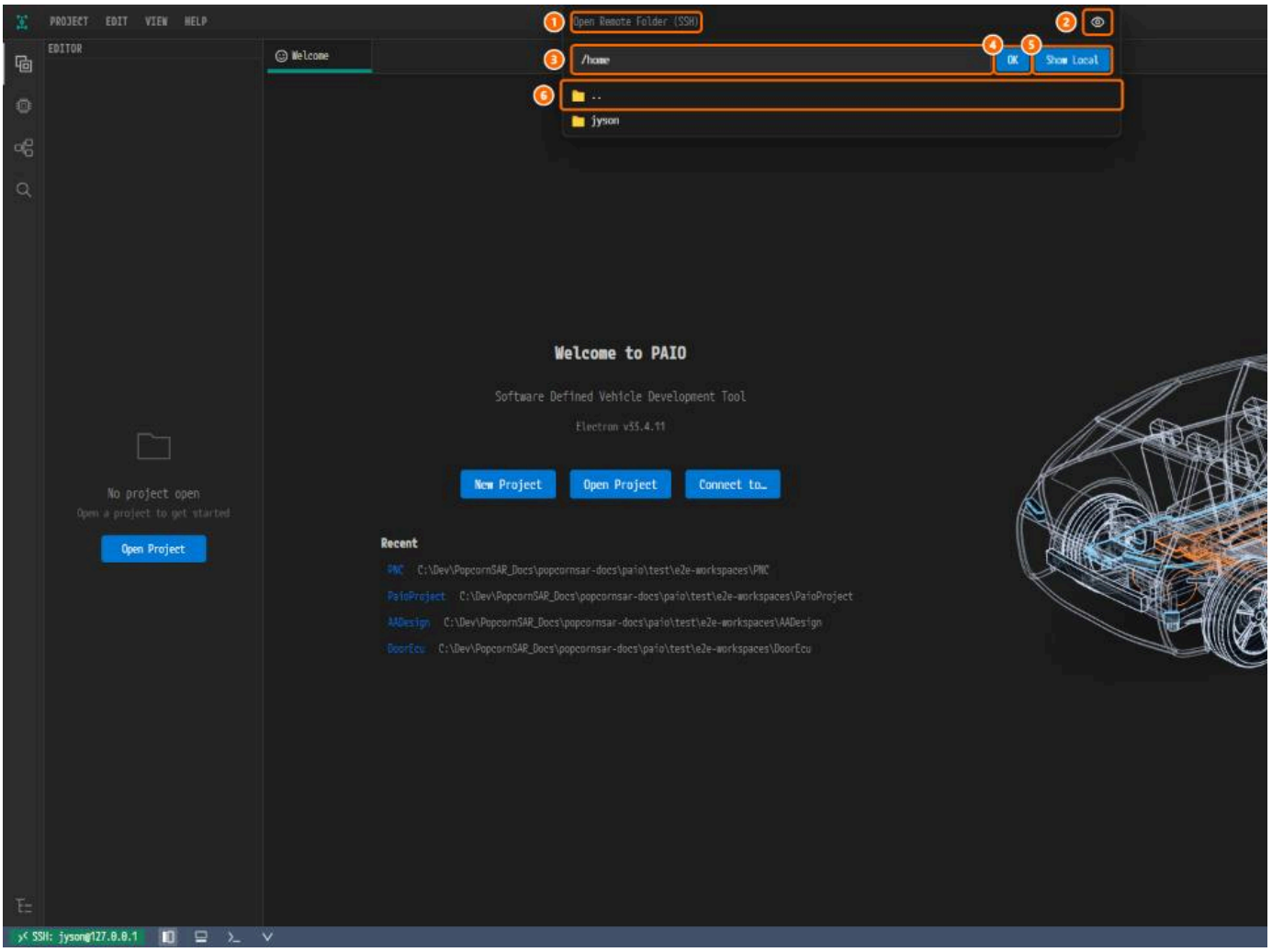
17.4 WSL (Windows)

WSL은 연결 대상이 아니다. `\\wsl$\<배포판>` 아래의 폴더는 일반 로컬 창으로 열린다. 프로젝트의 ECU 빌드와 실행은 vECU(17.3)에서 한다.

17.5 원격 폴더 선택 창 (SSH)

SSH 호스트에서 열 폴더를 고르는 창이다. 프로젝트가 없는 창에서 연결한 직후 저절로 열리고, SSH로 연결된 창에서는 **PROJECT ▸ Open Project...** 로도 열린다.

1. 폴더를 클릭해 들어가고, `..` 로 위로 올라간다. 경로를 입력해도 되며, 목록이 입력한 경로를 따라간다. 화살표 키로 목록을 이동한다.
2. **OK** 를 누르면 경로 입력란에 표시된 폴더가 창의 프로젝트로 열린다.
3. 폴더를 만들려면 목록 아래의 **+** 행(툴팁 `New Folder`)을 클릭하고 이름을 입력한 뒤 `Enter` 를 누른다.



*① 머리글 `Open Remote Folder (SSH)`, ② 눈 아이콘(점 파일), ③ 경로 입력란, ④ **OK**, ⑤ **Show Local**, ⑥ `..`.*

목록에는 폴더만 표시된다.

17.6 원격 언어 서버와 빌드 도구

17.6.1 자동 배포

SSH 창이나 컨테이너 창에서 C/C++/Rust 파일을 열거나 빌드하면, PAIO가 함께 배포한 Linux x64 바이너리(clangd, rust-analyzer, CMake, gcc, Rust 툴체인)를 `~/.paio-` 없다.

17.6.2 지원 범위

번들 바이너리는 glibc 기반 Linux x86_64용이다. arm64, macOS, Alpine 같은 musl 계열 배포판은 지원되지 않는다.

17.7 연결 종료

표시기 메뉴에서 **Close Connection** 을 고른다.

- **SSH**: 연결이 끊기고 창이 프로젝트 없는 창으로 다시 로드된다.
- **vECU 컨테이너**: 컨테이너의 작업 내용이 호스트로 복사되고 호스트 프로젝트의 창으로 돌아간다.

두 경우 모두 다음 실행 시 자동으로 다시 연결하지 않는다.

ARXML 임포트가 진행 중이면 Close Connection 은 먼저 `An ARXML import is running. Closing the connection cancels it.` 라고 묻는다. SSH 창에서는 워크

17.8 제한 사항과 주의점

- 프로젝트가 있는 창에서 연결하면 새 창이 열린다. 연결하거나 끊기 전에 저장하지 않은 편집을 저장한다.
- 포트 포워딩은 제공되지 않는다.
- ECU 빌드와 실행은 vECU에서 이뤄진다. 로컬 Windows 창과 SSH 창에서는 거부된다.
- 원격 빌드는 120초에서 강제 종료된다.
- 한 창은 한 번에 하나의 원격 모드만 가진다.

17.9 vECU에서 ECU 빌드 · 실행

프로젝트의 ECU는 네 명령 **Build ECU**, **Run ECU**, **Stop ECU**, **ECU Doctor** 로 빌드하고 실행한다. 코드 에디터의 **Build**, **Build & Run**(16.2)과는 다르다.

17.9.1 명령이 있는 곳

1. vECU 컨테이너 창의 **CODE** 섹션이나 **Explorer** 에서 `code/` 바로 아래 폴더(예: `code/SwcDemo`)를 우클릭한다.
2. **Build ECU** 를 고르고, 이어서 **Run ECU** 를 고른다.

명령 팔레트에도 네 명령이 `ECU` 분류로 들어 있다.

17.9.2 실행되는 곳

명령은 그 창의 vECU 컨테이너에서 실행된다. 로컬 Windows 창과 SSH 창에서는 거부된다.

17.9.3 실행되는 내용

PAIO는 폴더를 보고 플랫폼(PRTE ECU 또는 PARA Machine)을 정하고, vECU의 SDK로 CMake를 실행한다. 출력에 버전 불일치 경고가 보이면 vECU의 SDK에 맞춰 Code Gener

17.9.4 출력과 결과

Build ECU 나 **Run ECU** 를 실행할 때마다 터미널 패널에 읽기 전용 **Output** 채널이 열린다. 마지막 줄이 결과이며(예: `Build ECU SwcDemo: succeeded (exit code 0)`), `succeeded`). 파일을 가리키는 컴파일러 진단은 **PROBLEMS** 에도 `BUILD` 태그로 나타난다.

17.9.5 ECU RUNS 목록과 중지

터미널 목록 위의 **ECU RUNS** 영역에 진행 중인 빌드 · 실행과 최근에 끝난 것이 나열된다. 행의 **Stop** 이나 **Stop ECU** 로 실행을 끝낸다.

17.9.6 ECU Doctor

ECU Doctor 는 빌드에 필요한 도구와 SDK를 점검하며 아무것도 바꾸지 않는다. **Build ECU** 가 실패하면 먼저 **ECU Doctor** 를 실행한다.

17.9.7 ECU Settings

ECU Settings… 에서 ECU 하나의 SDK 루트, 실행 환경 변수, 실행 인수를 정한다. 호스트에서 저장한 설정은 전달되지 않으므로 ECU를 실행하는 vECU 창에서 설정한다.

17.10 호스트와 컨테이너 동기화

vECU 컨테이너는 프로젝트 사본으로 작업한다. PAIO는 들어갈 때, 저장할 때, **Close Connection** 때나 컨테이너 창을 닫을 때 그 사본과 호스트 프로젝트 폴더 사이에서 파일을 복사

17.10.1 진행과 결과

프로젝트 전체를 옮기는 동안 오른쪽 아래 토스트가 진행 상황을, 끝나면 결과를 알린다.

17.10.2 어느 쪽이 무엇을 가지나

- `design/`, `doc/`, `workflow/`, `export/`, `.devcontainer/` 는 호스트 소유다.
- `code/<node>/` 아래의 코드는 vECU 창에서 저장하는 즉시 호스트로 복사된다.
- ECU의 코드는 그 vECU 창에서 편집한다. 같은 파일을 호스트 창에서 저장하면 실행 중인 컨테이너에는 다시 들어갈 때까지 전달되지 않는다.

17.10.3 충돌 질문

결정이 필요한 파일이 있으면 PAIO는 파일마다 대화상자 하나로 묻는다.

17.10.4 vECU 두 개를 함께

같은 프로젝트의 PRTE vECU와 PARA vECU는 각자의 창에서 동시에 실행할 수 있다.

17.11 vECU 컨테이너의 코드 인텔리전스

vECU 컨테이너 창에서는 C/C++ 코드 인텔리전스가 이미지 안의 SDK를 기준으로 동작하므로, 로컬 창에서 해석되지 않던 SDK 헤더가 여기서는 해석된다. 언어 서버는 프로젝트를 돌다.

17.12 SSH 창에서 작업

SSH 창에서 만든 결과는 SSH 서버에 저장된다.

- **Code Generation**(PRTE, PARA)은 서버 프로젝트의 사본을 PC에서 처리하고, `export/<node>/` 와 `code/<node>/` 를 서버에 다시 쓴다.
- **AI 채팅**은 서버 프로젝트의 사본으로 작업하고 모든 파일을 서버에 먼저 쓴다. 대화는 서버 프로젝트에 저장된다(15.8).
- **AI 워크플로우** 실행은 서버 프로젝트의 사본으로 작업하며, 각 노드의 파일은 노드가 끝날 때 서버에 쓰인다(21.13).
- **Build ECU** 와 **Run ECU** 는 여전히 거부된다(17.9.2).

AI 워크플로우

이 장에서 다루는 것

AI 워크플로우를 설명한다. AI 워크플로우는 노드 그래프로, 프로젝트를 입력 문서에서 출발해 설계, AUTOSAR 익스포트와 코드 생성, 사용자 코드, 테스트, 빌드, vECU 실행, 검증 보고
하고 그래프를 실행한다. AI 노드는 각각 Claude 세션을 실행해 프로젝트에 파일을 쓰고, 나머지 노드는 PAIO 자체 도구를 실행한다.

- 대상 독자: 여러 단계를 한 번에 PAIO 에 맡기려는 사용자
- 선행 조건: 열린 프로젝트, 활성 라이선스, AI 노드용 Claude 계정(21.11), 코드를 빌드하고 실행하는 노드용 Docker 와 PRTE vECU 이미지(21.2)

> **워크플로가 쓴 것은 초안이다.** PAIO 는 노드가 쓴 것을 검사하고 결함을 그것을 만든 노드로 되돌려 보내지만, 설계가 옳은지는 판단하지 않는다. 결과에 기대기 전에 각 에디터와 14

AI 채팅은 별개의 기능이다. 15. AI 어시스턴트에서 고른 프로바이더는 AI 워크플로우에 적용되지 않는다.

21.1 AI 워크플로우가 하는 일

워크플로는 방향 그래프이다. 노드는 각각 한 단계를 수행하고, 그 결과를 출력에 연결된 노드로 넘긴다. 일반적인 흐름은 다음과 같다.

1. **Input Analysis** 가 프로젝트의 문서, 모델 파일, 코드를 읽어 `workflow/knowledge/` 아래에 지식 베이스를 만든다.
2. 설계 노드가 `design/` 아래에 설계를 쓴다.
3. **AUTOSAR Export** 가 ECU Extract 를 쓰고, **AUTOSAR Code Generation** 이 PRTE(CP) 또는 PARA(AP) 생성기를 실행한다.
4. **User Code Implementation** 과 **Test Code Implementation** 이 사용자 코드와 테스트를 쓴다.
5. **Compile/Build**, **Test Execution**, **Runtime Check** 가 vECU 에서 코드를 빌드하고 실행하며, **Verification Report** 가 요구사항 추적 표를 쓴다.

결함을 찾은 노드는 **Rejected** 출력으로 앞쪽 노드에 결함을 돌려보낼 수 있고, 그 노드가 다시 실행된다(21.8.4). 프로젝트마다 워크플로 문서는 정확히 하나, `workflow/WORKFLOW`

21.2 사전 조건

열린 프로젝트, 활성 라이선스, 모든 AI 노드용 Claude 계정(21.11), 빌드 노드와 실행 노드용 Docker 와 로컬 PRTE vECU 이미지가 필요하다. PAIO 는 이미지를 직접 받지 않는다. 17

21.3 에디터 열기와 화면 구성

1. 레일의 세 번째 아이콘인 **Workflow** 를 누른다. 에디터 탭 `Workflow` 가 열린다.
2. 새 프로젝트에서는 캔버스가 비어 있다. 처음 `Ctrl+S` 를 누를 때 `workflow/WORKFLOW.io.workflow` 가 만들어진다.

에디터는 왼쪽에 노드 팔레트, 가운데에 캔버스와 그 아래 실행 로그, 오른쪽에 인스펙터가 있다. 탐색기에서 `WORKFLOW.io.workflow` 를 더블클릭하면 에디터가 아니라 읽기 전용

21.4 노드 팔레트와 노드 유형

21.4.1 팔레트

팔레트는 노드 유형을 범주별로 나열한다. `Search nodes` 에 입력하면 목록을 거르고, 항목을 캔버스로 끌어다 놓으면 노드가 추가된다. 항목을 클릭해서는 아무것도 추가되지 않는

21.4.2 노드 카드

카드에는 노드 이름과 유형 레이블이 보인다. `(OFF)` 는 꺼진 노드를 뜻하고, `START`, `END`, `PINNED` 표식은 21.5.3 의 설정을 보여 준다. 실행 중에는 상태 배지가 노드가 실행 중인지

21.4.3 노드 유형

노드는 요구사항, 설계, 생성, 구현, 빌드와 검증, 제어(**Approval**, **Condition Branch**, **Sticky Note**)를 다룬다. 입력 포트는 모두 필수이다.

21.5 워크플로 만들기

21.5.1 노드 연결

출력 포트(노드 오른쪽 가장자리)에서 다른 노드의 입력 포트(왼쪽 가장자리)까지 끈다. 출력 하나가 여러 노드에 이어질 수 있다. 루프는 점선으로 그려지는 피드백 연결로만 허용된다
노드로 돌아가는 연결이다.

21.5.2 선택과 이동

노드를 클릭하면 선택되고 설정이 보이며, 빈 캔버스를 클릭하면 워크플로 설정이 보인다. 빈 캔버스를 끌어 화면을 옮기고, `Ctrl`+휠로 확대하거나 축소한다.

21.5.3 컨텍스트 메뉴

노드를 오른쪽 클릭하면 노드를 끄거나, 그 노드부터 실행하거나, 출력을 고정하거나, 실행의 START 또는 END 노드로 정할 수 있다.

21.5.4 스티키 노트

빈 캔버스를 오른쪽 클릭하고 **Add Sticky Note** 를 고른다. 메모를 선택하고 인스펙터의 **NOTE** 필드에 내용을 입력한다.

21.5.5 플로팅 툴바

캔버스 왼쪽 아래 툴바에는 `Run Workflow`, `Cancel Run`, `Auto Layout`, `Templates`, `Show/Hide Run Log` 가 있다.

21.5.6 템플릿

템플릿은 워크플로 그래프 전체를 이 컴퓨터의 라이브러리에 보관한 것으로, 모든 프로젝트가 함께 쓴다. **Save as Template** 로 현재 그래프를 저장하고, `Load: <name> (<n> no`

21.5.7 저장과 실행 취소

`Ctrl+S` 는 `workflow/WORKFLOW.io.workflow` 를 저장한다. `Ctrl+Z` 는 실행 취소, `Ctrl+Y` 는 다시 실행이다.

21.6 인스펙터

21.6.1 워크플로 설정

빈 캔버스를 클릭하면 **Workflow** 설정이 나온다.

- **RUN MODE** > **NEXT RUN** 은 다음 실행이 기존 결과를 어떻게 다룰지 정한다. `UPDATE`(기본값), `REBUILD`, `RESET` 중 하나이며, 한 번의 실행에만 적용된다.
- **AI DEFAULTS** 는 모든 AI 노드가 쓰는 Claude 모델과 effort 를 정한다.
- **PER-ECU NODES** > **PARALLEL ECUS** 는 일부 노드가 동시에 다루는 ECU 수를 정한다.

> `RESET` 은 먼저 `design/`(손으로 만든 문서를 포함한 모든 설계 문서)과 다른 폴더를 지운다. 삭제는 되돌릴 수 없다.

21.6.2 노드 설정

노드를 선택하면 **ENABLED**, **NAME**, **AI SETTINGS**, **PARAMETERS**(21.7), **LAST RUN**(21.9.1) 같은 설정이 나온다.

21.6.3 여러 노드가 함께 쓰는 설정

ADDITIONAL INSTRUCTIONS, **INSUFFICIENT INFORMATION**, **FEASIBILITY CHECK**, **CONNECT TO VECU**, **TARGET ECU / TARGET ECUS** 는 여러 노드에 나온다.

21.6.4 PRTE vECU 이미지

PRTE vECU 이미지 하나가 워크플로 전체에 적용된다. 목록은 로컬 `docker images` 에서 읽고, 기본값은 실행 시점의 가장 새 로컬 이미지이다. 자동으로 받지는 않는다.

21.7 노드 설정 참조

노드마다 **PARAMETERS** 아래에 고유 필드가 있다. 아래 각 절은 노드 설정이 무엇을 정하는지 설명한다.

21.7.1 Input Analysis

분석할 소스(문서, 모델 파일, 코드)와, Feasibility Check 실행, 지식 베이스 빌드, 운영 시나리오 작성 여부를 정한다.

21.7.2 System Design

시스템 이름, 통신 범위, 각 ECU / HPC 노드를 어떻게 할지 정한다.

21.7.3 Diagnostic Design

TARGET ECU / HPC 에 설계할 노드 이름을 적는다. 비우면 모든 ECU 와 HPC 이다.

21.7.4 SWC Design

대상 ECU, 기존 SW-C 를 어떻게 할지, 설계 방식, 기존 인터페이스를 바꿀 수 있는지를 정한다.

21.7.5 BSW Config

대상 ECU, Base Set(`PRTE` 또는 `Other Vendor`), 모듈을 고르는 방식을 정한다.

21.7.6 SWC + BSW Integration과 Functional Cluster Config

둘 다 대상 목록(**TARGET ECU** / **TARGET HPC**)이 있다.

21.7.7 AA Design

TARGET COMPOSITION 에 컴포지션 이름을 적는다. 이 버전에서는 노드가 실행되지 않는다(21.12).

21.7.8 Design Review

REVIEWER 는 `AI`(기본값) 또는 `HUMAN` 이다. `HUMAN` 은 사람이 승인하거나 거부할 때까지 실행을 멈춘다(21.8.3).

21.7.9 AUTOSAR Code Generation

대상 ECU 와 생성기 결과를 고치는 방식을 정한다.

21.7.10 User Code Implementation

플랫폼, 대상, 코드 범위, MISRA C:2025 검사, 기존 코드 처리 방식을 정한다.

21.7.11 Test Code Implementation

TEST LEVEL 은 `Unit`(기본값) 또는 `Integration` 이다.

21.7.12 Compile/Build

경고를 오류로 볼지, AI 자동 수정, 병렬 작업 수를 정한다.

21.7.13 Runtime Check

실행 시간, 버스, 수정 방식, 확인할 기대 동작을 정한다.

21.7.14 Test Execution과 Verification Report

Test Execution 에는 타임아웃과 AI 실패 분석이, **Verification Report** 에는 AI 요약이 있다.

21.7.15 Approval과 Condition Branch

Approval 에는 실행이 기다리는 동안 보여 줄 문구인 **APPROVAL MESSAGE** 가 있다. **Condition Branch** 에는 **CONDITION** 이 있지만 이 버전에서는 실행되지 않는다(21.12).

21.8 워크플로 실행

21.8.1 실행 시작

1. 툴바의 `Run Workflow` 를 누른다. 저장하지 않은 편집을 포함해 보이는 그대로의 문서가 실행된다.
2. 중간부터 시작하려면 노드를 오른쪽 클릭하고 **Run from Here** 를 고른다. 위쪽 값은 가장 최근에 기록된 실행에서 가져온다.

프로젝트마다 한 번에 하나의 실행만 가능하다.

21.8.2 실행 순서

입력이 준비된 노드들은 병렬로 실행된다. 입력이 여럿인 노드는 모두 도착할 때까지 기다린다. 꺼진 노드는 입력을 그대로 넘긴다.

21.8.3 사용자를 기다리는 노드

REVIEWER 가 `HUMAN` 인 **Design Review** 와 **Approval** 노드는 실행을 멈춰 둔다.

1. 노드를 선택한다. **LAST RUN** 에 `Waiting for response…` 가 보인다.
2. 필요하면 `Comment (optional)` 에 주석을 입력한다.
3. 체크(`Approve`) 또는 X(`Reject`)를 누른다.

프로젝트에서 얻을 수 없는 정보가 필요한 노드는 `Waiting for user input: <question>` 으로 묻는다. 요청한 것을 한 뒤 Approve 를 누른다.

21.8.4 Rejected 루프

노드가 **Rejected** 포트로 결함을 보고하고 그 포트가 앞쪽 노드로 연결되어 있으면, 실행은 그 노드를 결함 내용과 함께 다시 실행하고 이어서 그 뒤 노드들을 실행한다. 되돌아가는 연

21.8.5 실행 지켜보기

캔버스 아래 실행 로그에 단계마다 한 줄씩 보인다. 에디터 탭을 닫아도 실행은 백그라운드에서 계속되고, 끝나면 토스트가 알린다.

21.8.6 취소와 닫기

툴바의 `Cancel Run` 은 실행을 멈추고, 그 실행이 쓴 파일을 모두 되돌린다. 실행 중에 탭이나 창을 닫으면 동작이 다르다.

21.8.7 중단된 실행 이어 하기

실행 중에 PAIO 가 종료되었다면, 에디터를 다시 열 때 **Interrupted Workflow Run** 이 묻는다. **Resume** 은 완료된 노드를 재사용하고 나머지를 실행한다. **Discard** 는 기록을 버린다

21.9 실행 결과와 저장 위치

21.9.1 Last Run

노드를 선택하면 **LAST RUN** 이 보인다. 상태가 먼저 나오고, 설계한 요소, 결과(Findings), 바뀐 파일 같은 행이 이어진다. 현재 세션의 실행만 보여 준다.

21.9.2 실행 전후 비교

LAST RUN 의 `Compare before/after of this run` 은 실행 전 파일과 현재 파일을 나란히 보여 주는 읽기 전용 비교를 연다.

21.9.3 결과가 쓰이는 곳

노드는 `workflow/`, `doc/`, `design/`, `export/`, `code/` 아래에 쓴다. 보고서는 `doc/report/` 에 쓰인다.

21.9.4 실행 폴더

모든 실행은 `<project>/.paio/workflow-runs/<runId>/` 에 기록되며, 되돌리기와 비교에 쓰는 백업도 여기에 있다. `.paio` 폴더는 탐색기에서 기본으로 숨겨진다.

21.10 Knowledge 뷰

BUILD KNOWLEDGE (LLM WIKI) 가 켜진 **Input Analysis** 는 `workflow/knowledge/` 아래에 페이지를 만들고, 이후의 모든 노드는 이 페이지를 바탕으로 작업한다. 캔버스 위의 **WORKFLOW** 를 누르면 돌아간다. 페이지를 클릭하면 미리 볼 수 있고, 요구사항 페이지에서는 검증 방법을 정할 수 있다.

21.11 Claude 계정 연결 (AI Account)

워크플로 AI 노드는 언제나 PAIO 에 번들된 Claude Code 런타임으로 Claude 를 쓴다. 기본값은 이 컴퓨터의 Claude Code 로그인이다.

1. 상단 바 오른쪽의 사용자 아바타 > **AI Account** 를 연다(3.3).
2. 상태 줄을 확인한다. `Not connected` 이면 AI 노드를 실행할 수 없다.
3. **Connect** 로 브라우저 로그인을 하거나, **Paste Token** 으로 토큰을 붙여 넣는다.

21.12 제한 사항과 구현되지 않은 노드

- **AA Design** 과 **Condition Branch** 는 배치할 수 있지만 이 버전에서는 실행되지 않는다. 끄거나 빼 둔다.
- **Runtime Check** > **BUS** = `REAL BUS` 와 **User Code Implementation** > **PLATFORM** = `AP` 는 아직 쓸 수 없다.
- 프로젝트마다 한 번에 하나의 실행만 가능하다.

21.13 SSH 창에서의 워크플로 실행

SSH로 연결된 창(17. 원격 개발)에서는 실행이 PC에 둔 서버 프로젝트의 사본으로 작업하며, 각 노드의 파일은 노드가 끝날 때 서버에 쓰인다.

- **User Code Implementation**, **Test Code Implementation**, **Compile/Build**, **Test Execution**, **Runtime Check** 는 그곳에서 실행되지 않는다. 이 노드들은 vECU에서 빌
- 실행 중의 **Close Connection** 은 먼저 묻는다. 끊으면 파일을 되돌리지 않고 실행이 취소된다.
- 실행 기록은 PC에 남는다.

워크플로 그래프는 AI 채팅으로도 만들거나 바꿀 수 있다(15. AI 어시스턴트).

레퍼런스

문서	다루는 내용
단축키 총람	동작하는 위치별로 묶은 단축키
용어 사전	PAIO와 이 매뉴얼에서 쓰는 용어
문제 해결 · FAQ	자주 겪는 문제와 원인, 대처 방법

단축키 총람

이 장에서 다루는 것

PAIO에서 동작하는 키보드 · 마우스 조작을 화면 영역별로 정리하고, 상황에 따라 다르게 동작하는 키도 함께 표시한다. 영역 이름은 19. 용어 사전을 따르고, 화면 구성은 05. 화면 구성

- macOS에서는 `Ctrl` 을 `Cmd` 로 읽는다.
- 표에 없는 키는 등록되어 있지 않다. 텍스트 영역은 브라우저 기본 동작을 따른다.
- 일부 키는 보이지 않는 네이티브 메뉴의 가속기이다(`Ctrl+O`, `Ctrl+Shift+N`, `Ctrl+S`, `F11`). 확대/축소 키를 포함한 나머지는 앱과 각 편집기가 자기 영역에서 처리한다. (

18.1 전역

앱 어느 화면에서나 동작하는 키이다.

단축키	동작	비고
`Ctrl+O`	Open Project...	네이티브 메뉴가 등록한 가속기이다. 고른 폴더가 그 창의 프로젝트가 된다. System 편집기 캔버스와 Markdown 편집기 안에서는 다르
`Ctrl+Shift+N`	New Window	네이티브 메뉴가 등록한 가속기이다. 프로젝트가 열려 있지 않은 PAIO 창을 하나 더 연다
`Ctrl+S`	활성 편집기 저장	편집 가능한 편집기가 각자 처리한다. 어느 편집기도 처리하지 않은 `Ctrl+S` 는 메뉴 Save와 같은 경로로 저장한다
`Ctrl+Shift+S`	활성 편집기 저장	`Ctrl+S` 와 똑같이 동작한다. 다른 이름으로 저장(Save As)은 없다
`Ctrl+Z`	Undo	편집 가능한 편집기가 각자 처리해 프로젝트에 하나뿐인 편집 이력을 되돌린다. 편집 불가 탭에서는 물리 키가 아무 일도 하지 않는다 (18.
`Ctrl+Y`	Redo	System · VFB 캔버스, 설정 다이얼로그, DataType 편집기는 `Ctrl+Shift+Z` 도 같은 동작으로 받는다
`Ctrl+X` / `Ctrl+C` / `Ctrl+V`	잘라내기 / 복사 / 붙여넣기	텍스트 입력 대상이다. **EDIT** 메뉴 항목은 브라우저 클립보드 명령을 실행한다
`Ctrl+A`	전체 선택	텍스트 입력 대상이다. Explorer, System 편집기 캔버스, VFB 캔버스에는 별도 동작이 있다
`Ctrl+B`	사이드바 토글	왼쪽 아이콘 막대에서 고른 사이드바 뷰(**EDITOR** 뷰, Explorer 등)를 보이거나 숨긴다. 코드 편집기는 같은 키를 Build로 등록한다
`Ctrl+``	터미널 패널 토글	터미널이 없으면 하나를 만들어 준다
`Ctrl+J`	**터미널 패널 토글**	메뉴 표기는 `Toggle Panel` 이지만 실제 동작은 터미널이다
`Ctrl+Shift+P`	명령 팔레트 열기/닫기	
`Ctrl+Shift+F`	Search 뷰 열기	터미널에 포커스가 있어도 동작한다. 모달 다이얼로그가 창을 덮고 있으면 동작하지 않는다
`F4` / `Shift+F4`	다음 / 이전 검색 결과로 이동	Search 뷰에 결과가 있고, 편집기나 다이얼로그가 그 키를 가져가지 않았을 때만 동작한다
`Ctrl+T`	Meta Class 검색 열기	AUTOSAR 메타 클래스를 어디에서 설계하는지 찾는다(05.10). 모달 다이얼로그가 열려 있으면 동작하지 않는다. Markdown 편집기와 I
`Ctrl+=` / `Ctrl+-` / `Ctrl+0`	확대 / 축소 / 배율 초기화	`Ctrl++` 와 숫자 키패드의 `+`, `-`, `0` 도 `Ctrl` 과 함께 쓸 수 있다. 키, 메뉴 항목, 명령 팔레트가 모두 같은 방식으로 배율을 바꾸며, 배율
`F11`	전체 화면 토글	네이티브 메뉴가 등록한 가속기이다
`Alt+F4`	창 닫기	OS가 처리하는 창 닫기이며 앱이 등록한 키가 아니다. 변경분이 있으면 저장 확인이 먼저 뜬다

New Project... 에는 단축키가 없다. **Welcome** 탭의 **New Project** 버튼이나 **PROJECT** 메뉴에서 시작한다.

18.1.1 메뉴 표기와 실제 동작이 다른 키

메뉴에 적혀 있지만 눌렀을 때 메뉴 항목과 다른 일을 하는 키이다.

단축키	메뉴 표기	실제
`Ctrl+J`	Toggle Panel	메뉴 항목을 클릭하면 Utility 패널이 토글되지만, 키를 누르면 터미널 패널이 토글된다. Utility 패널을 여는 단축키는 없다

18.2 그래픽 편집기 (System · VFB)

System 편집기(탭 `<project>:System`)와 VFB 편집기는 같은 키를 받지만 **실제로 동작하는 범위가 다르다.** 표의 "동작하지 않음"은 눌러도 아무 일도 일어나지 않고 브라우저 기

18.2.1 키보드

단축키	System	VFB
`Ctrl+S` / `Ctrl+Shift+S`	저장	저장
`Ctrl+Z`	실행 취소	실행 취소
`Ctrl+Y` / `Ctrl+Shift+Z`	다시 실행	다시 실행
`Esc`	선택 해제	선택 해제
`Delete` / `Backspace`	확인을 거쳐 선택한 노드를 삭제한다. 연결선은 이 키로 지워지지 않으며, 연결선 자체의 컨텍스트 메뉴에서 지운다	컨텍스트 메뉴의 Delete와 같이 선택 항목을 삭제한
`Ctrl+A`	전체 선택	현재 드릴다운 단계의 노드 전체 선택
방향키	선택 항목을 1px 이동	동작하지 않음
`Shift` + 방향키	선택 항목을 20px(그리드 한 칸) 이동	동작하지 않음
`Ctrl+O`	파일 선택 대화상자를 열고, 고른 IVC 모델 파일을 현재 System 편집기에 불러들인다	동작하지 않음
`Ctrl+C` / `Ctrl+V` / `Ctrl+X`	동작하지 않음 (노드 복사·붙여넣기는 제공되지 않는다)	동일

- `Ctrl+S`, `Ctrl+Shift+S`, `Ctrl+O`, `Ctrl+Z`, `Ctrl+Y` 는 속성 패널의 입력란에 포커스가 있어도 캔버스가 처리한다. 실행 취소 계열은 입력란의 포커스를 먼저 해제하므로 그 밖의 키(`Delete`, `Ctrl+A`, 방향키 등)는 입력란에 포커스가 있으면 무시된다.
- 방향키로 여러 노드를 한꺼번에 옮겨도 실행 취소는 **한 번**으로 되돌아간다.
- 경로 꺾임점(waypoint)을 드래그하는 중에 `Esc` 를 누르면 선택 해제가 아니라 드래그가 취소되고 원래 모양으로 돌아간다. 편집 이력에는 아무 작업도 남지 않는다.

> **System 편집기의 `Ctrl+O` 는 프로젝트를 여는 키가 아니다.** System 편집기 캔버스에 포커스가 있으면 IVC 모델 파일 선택 대화상자가 열리고, 고른 파일이 **현재 System 편집**

18.2.2 마우스

캔버스에서의 마우스 조작이다.

조작	동작
빈 곳 좌클릭 드래그	화면 이동
`Ctrl` + 빈 곳 좌클릭 드래그	사각형 선택. 사각형 안에 완전히 들어간 노드가 선택되며, 이전 선택을 대신한다
가운데 버튼 드래그	화면 이동
`Space` + 좌클릭 드래그	화면 이동
휠 (수정자 없음)	세로 화면 이동 (문서 스크롤처럼 동작한다)
`Shift` + 휠	가로 화면 이동
`Ctrl` + 휠	확대/축소 (0.1배 ~ 4.0배, 커서 위치 기준)
`Shift` + 드래그	연결 모드
`Ctrl` + 클릭	다중 선택 (토글)
더블클릭	노드별 설정 다이얼로그 / 컴포지션 드릴다운
우클릭	컨텍스트 메뉴

18.3 설정 다이얼로그

System 편집기의 노드 · 클러스터 · 스위치 설정 다이얼로그와 **Partial Network (PNC) Configuration** 다이얼로그, VFB의 **Machine Configuration** · **ECU Configuration** · **SW Intrusion Detection System** · **Software Cluster** 다이얼로그에서 쓰는 키이다.

단축키	동작
`Ctrl+S` / `Ctrl+Shift+S`	저장
`Ctrl+Z`	다이얼로그 내부 실행 취소
`Ctrl+Y` / `Ctrl+Shift+Z`	다이얼로그 내부 다시 실행
`Esc`	가장 안쪽에 열린 UI를 닫고, 열린 것이 없으면 다이얼로그를 닫는다

`Esc` 는 드롭다운 · 팝업 메뉴 · 인라인 편집기 같은 일시적 UI부터 닫고, 한 번 더 누르면 다이얼로그를 닫는다. 변경분이 있으면 `Do you want to save the changes?` 확인이 뜬

> **`Ctrl+Z` 의 동작 범위는 다이얼로그마다 다르다.**
>
> - VFB의 **Machine Configuration** · **ECU Configuration** · **SWC Internal Behavior** · **AA Internal Behavior** · **Intrusion Detection System** · **Software Cluster** 다이얼로그 · **Partial Network (PNC) Configuration** 다이얼로그는 내부 이력이 비면 **프로젝트 편집 이력으로 넘어간다.** 저장 직후처럼 내부 이력이 비워진 상태에서도 `Ctrl+Z` 가 동작한다.
> - System 편집기의 클러스터 설정 다이얼로그는 클러스터 탭에서만 넘어간다. 이 탭의 편집은 곧바로 프로젝트 편집 이력에 기록되기 때문이다.
> - System 편집기의 노드 · 스위치 설정 다이얼로그는 넘어가지 않는다. 내부 이력이 비면 `Ctrl+Z` 는 아무 일도 하지 않고, 이 키는 바깥으로도 전달되지 않는다. 이전 편집을 되

설정 다이얼로그의 내부 이력은 System 편집기 · VFB 모두 최대 50단계까지 보관된다.

18.4 표 기반 모델 편집기

Signal · Interface · DataType · Diagnostic · Module Definition · Module Configuration 여섯 편집기는 캔버스가 아니라 목록과 속성 패널로 이루어져 있다. Interface · DataType · Diagnostic · Module Definition과 Module Configuration은 **MODULE** > **Definition** · **Configuration** 에서 열고, Signal은 System 편집기 캔버스 왼쪽 가장자리의 **Signal** 아이콘으로만 연다.

단축키	동작
`Ctrl+S` / `Ctrl+Shift+S`	저장
`Ctrl+Z`	실행 취소 (프로젝트 편집 이력)
`Ctrl+Y`	다시 실행. DataType 편집기는 `Ctrl+Shift+Z` 도 받는다

실행 취소 계열은 입력 중이던 값을 먼저 하나의 작업으로 확정한 뒤 되돌린다. 그래서 필드 편집이 통째로 되돌아가고, 브라우저의 글자 단위 되돌리기가 끼어들지 않는다.

편집기별로 다음 키가 더 있다.

편집기	단축키	동작
Interface	`Delete` / `Backspace`	확인을 거쳐 선택한 인터페이스를 삭제한다
Module Definition	`Delete` / `Backspace`	확인을 거쳐 선택한 정의 노드를 삭제한다 (모듈 노드는 제외)
Module Configuration	`Delete` / `Backspace`	확인을 거쳐 선택한 컨테이너 인스턴스를 모두 삭제한다
Module Configuration	`Ctrl+D`	선택한 컨테이너 인스턴스를 복제한다
Module Configuration	`F2`	속성 패널의 첫 입력란(Name)으로 포커스를 옮기고 값을 선택한다

- `Delete` 계열은 입력란 · 텍스트 영역 · 콤보에 포커스가 있으면 무시된다.
- 목록에서는 `ArrowUp` / `ArrowDown` 으로 행을 옮긴다. DataType의 타입 목록과 단위 목록, Interface의 인터페이스 목록, Signal의 프레임 목록과 시그널 표가 이 방식을 쓴다.
- DataType은 화면에 보이는 동안 창 전체에서 키를 받고, 나머지 다섯은 편집기 영역 안에 포커스가 있어야 한다. 포커스가 편집기 영역 밖에 있을 때 누른 `Ctrl+S` 도 메뉴 Save

18.5 코드 편집기 (C / C++ / Rust)

코드 편집기에서 쓰는 키이다.

단축키	동작
`Ctrl+S`	저장
`Ctrl+B`	Build
`Ctrl+F5`	Build & Run
`Shift+Alt+F`	Format Document
`Alt+O`	Switch Header/Source (`.c` ↔ `.h`, `.cpp` ↔ `.hpp` 등)
`F12`	정의로 이동. `Ctrl` + 클릭과 같다
`Ctrl` + 클릭	정의로 이동. 언어 서버가 없으면 따옴표 `#include "..."` 만 열리며, 현재 파일의 폴더를 기준으로 그 경로에 헤더가 있을 때만 열린다

그 밖의 편집 단축키는 Monaco 기본값을 따른다. 빌드 흐름은 16. 코드 편집 · 빌드 · 터미널 · 문서를 참고한다.

18.6 XML / ARXML 편집기

XML / ARXML 편집기에서 쓰는 키이다.

단축키	동작
`Ctrl+S`	저장
`Ctrl+Z` / `Ctrl+Y`	실행 취소 / 다시 실행 (Monaco 자체 이력)
`Shift+Alt+F`	Format XML

- `Ctrl+S` 와 실행 취소 계열은 **편집 모드에서만** 동작한다. 파일은 보기 모드로 열리므로, 먼저 툴바 왼쪽의 눈 아이콘을 눌러 편집 모드로 바꾼다. 보기 모드에서는 툴바의 저장 · 실
- Explorer에서 여는 설계 파일처럼 PAIO 자체 문서(`*.io.model`, `*.io.workflow`, `*.io.prj`, `*.vfb.layout`)는 읽기 전용으로 표시되고 편집 모드로 바꿀 수 없으므로 고친다.
- `.dbc` 파일에서는 서식 버튼이 숨겨지지만 `Shift+Alt+F` 키는 그대로 동작해, 편집 모드에서 누르면 XML 서식 함수가 실행된다.
- 편집 모드에서 `<` 를 입력하면 AUTOSAR 스키마상 허용된 자식 요소가 자동 완성으로 제안된다.

18.7 Markdown 편집기

Markdown 편집기에서 쓰는 키이다. 세 키는 Markdown 편집기가 화면에 보이는 동안에만 동작한다.

단축키	동작
`Ctrl+S`	저장한다. 아직 저장한 적 없는 문서면 저장 위치 대화상자가 뜬다
`Ctrl+T`	새 문서를 연다 (문서를 이미 편집 중이면 별도 탭으로 연다)
`Ctrl+O`	Markdown 파일 선택 대화상자를 열고, 고른 파일을 별도 탭으로 연다

이 세 키는 Markdown 편집기가 보일 때만 동작한다. `Ctrl+T` 는 포커스가 Markdown 편집기나 Doc 시작 화면에 있을 때만 새 문서를 열고, 그 밖에서는 Meta Class 검색을 연다

굵게 · 기울임 버튼의 툴팁에는 `Bold (Ctrl+B)` / `Italic (Ctrl+I)` 가 적혀 있지만 **이 편집기에서 그 키는 동작하지 않는다.** 서식은 툴바 버튼 여섯 개(`B`, `I`, `S`(취소선), `H`(제

18.8 Explorer

Explorer는 창 맨 왼쪽 아이콘 막대 맨 아래의 **Explorer** 아이콘으로 연다. 프로젝트 폴더의 내용을 그대로 나열하며, 그 위에 프로젝트 행은 없다. 마우스 조작은 다음과 같다.

조작	동작
클릭	선택
`Ctrl` + 클릭	다중 선택 (토글)
`Shift` + 클릭	범위 선택
더블클릭	파일 형식에 맞는 편집기로 파일을 연다. 설계 파일(`*.io.model`)은 읽기 전용으로 열리므로(18.6 참고), 설계 편집기는 **EDITOR** 뷰에서 연다
우클릭	컨텍스트 메뉴
화살표(꺾쇠) 클릭	펼치기 / 접기
드래그	이동. **파일과 폴더 노드만** 끌 수 있다

노드에 포커스가 있을 때 쓰는 키는 다음과 같다.

단축키	동작
`Enter` / `Space`	자식이 있으면 펼치기/접기, 없으면 편집기로 연다
`ArrowRight`	접힌 노드를 펼친다
`ArrowLeft`	펼친 노드를 접는다
`Delete`	선택한 **파일 · 폴더**를 삭제한다 (확인 대화상자가 뜬다)
`Ctrl+C`	복사
`Ctrl+X`	잘라내기
`Ctrl+V`	붙여넣기
`Ctrl+A`	포커스가 있는 폴더와 그 펼쳐진 내용을 선택한다(파일이면 그 파일이 든 폴더의 펼쳐진 내용). `doc`, `code`, `export` 폴더 안에서만 동작하며, 그 밖에서는 선택이 바뀌지 않는다

`Delete` 는 고정 폴더가 실수로 지워지지 않도록 고정 폴더(`code`, `design`, `doc`, `export`, `workflow`)에는 동작하지 않는다.

18.9 탭

편집기 탭에서 할 수 있는 조작이다.

조작	동작
클릭 / `Enter` / `Space`	탭 활성화
가운데 버튼 클릭	탭 닫기
우클릭	Close / Close Others / Close Saved / Close All
탭 바 위로 드래그	순서 변경 (같은 그룹) 또는 그룹 간 이동
본문 오른쪽 절반으로 드래그	좌우 분할 후 새 그룹으로 이동
본문 왼쪽 절반으로 드래그	분할 없이 그 그룹으로 이동

- 탭 컨텍스트 메뉴에서는 `ArrowUp` / `ArrowDown` / `Home` / `End` 로 항목을 옮기고, `Enter` 또는 `Space` 로 실행하고, `Esc` 로 닫는다.
- 분할은 좌우 방향만 지원한다.
- **Close Others** · **Close Saved** · **Close All** 은 **Close** 와 같은 방식으로 탭을 하나씩 닫으므로, 변경분이 있는 탭은 먼저 저장 여부를 묻는다(05.3.3 참고).

18.10 명령 팔레트

명령 팔레트에서 쓰는 키이다.

단축키	동작
`Ctrl+Shift+P`	열기 / 닫기
`ArrowDown` / `ArrowUp`	항목 이동
`Tab` / `Shift+Tab`	항목 이동
`Enter`	선택한 명령 실행
`Esc`	닫기

팔레트에 등록된 명령은 05.6 명령 팔레트에 있다.

18.11 알아 둘 충돌과 함정

같은 키가 화면 영역에 따라 다르게 동작하는 경우이다.

키	상황에 따른 동작
`Ctrl+B`	코드 편집기에서는 **Build** 로 등록되어 있다. 그 밖에서는 사이드바를 토글한다
`Ctrl+O`	기본 → Open Project... / System 편집기 캔버스 → **IVC 모델 파일 불러오기** / VFB 캔버스 → **아무 동작 없음** / Markdown 편집기 → Markdown 파일 열기
`Ctrl+J`	메뉴 항목은 Utility 패널을 토글하지만, 키는 실제로 **터미널 패널**을 토글한다
`Ctrl+T`	기본 → Meta Class 검색 열기 / 포커스가 Markdown 편집기나 Doc 시작 화면에 있음 → 새 문서 / 모달 다이얼로그가 열려 있음 → 아무 동작 없음
`Ctrl+A`	텍스트 입력 → 전체 선택 / Explorer → `doc` · `code` · `export` 안 폴더의 펼쳐진 내용 / System 편집기 캔버스 → 전체 선택 / VFB 캔버스 → 현재 드릴다운 단계 전체 선택
`Delete`	Explorer → 파일·폴더 삭제 / System 편집기 캔버스 → 선택한 노드 삭제 (연결선 제외) / VFB 캔버스 → 선택 항목 삭제 / Interface · Module Definition · Module Configuration → 선택
`Esc`	열린 드롭다운·메뉴 → 그것을 닫음 / 없으면 다이얼로그 닫기 / 캔버스에서는 선택 해제 (꺾임점 드래그 중이면 드래그 취소)
`Ctrl+Z`	편집 가능한 편집기 → 프로젝트 편집 이력 / 설정 다이얼로그 → 다이얼로그 내부 이력이 먼저 (18.3 참고) / 코드 · XML 편집기 → Monaco 자체 이력 / 이미지 · 텍스트 뷰어, 읽기 전용 문서 → 동

모델 편집은 하나의 프로젝트 이력으로 모이지만, **물리 `Ctrl+Z` 는 각 편집기가 직접 처리**한다. 이미지 뷰어처럼 편집 이력이 없는 탭에서는 키가 아무 일도 하지 않는다. 이때는 **EDIT** 탭에서든 가장 최근 모델 편집을 되돌리고 그 편집의 편집기로 화면을 옮긴다. 되돌릴 작업의 이름은 **EDIT** 메뉴에, 전체 목록은 Utility 패널의 History 탭에 있다(05.4.1 참고).

18.12 Search 뷰

Search 뷰와 Search Editor 탭에서 동작하는 키이다(05.8, 05.9). `Ctrl+Shift+F` 와 `F4` / `Shift+F4` 는 18.1에 있다.

검색 입력란에서:

단축키	동작	비고
`Enter`	곧바로 검색	검색어를 검색 기록에도 남긴다
`Escape`	검색 입력란 비우기	Search 뷰에서만, 입력란에 글자가 있을 때만
`Ctrl+Down`	결과 목록으로 이동	Search 뷰에서만
`ArrowUp` / `ArrowDown`	이 프로젝트의 이전 검색어 오가기	제안 목록이 열려 있으면 화살표는 목록 안을 이동하고 `Enter` 는 고른 검색어를 검색한다

Search 뷰의 결과 목록에서:

단축키	동작	비고
`ArrowUp` / `ArrowDown`	이전 / 다음 행으로 이동	첫 행에서 `ArrowUp` 을 누르면 검색 입력란으로 돌아간다
`Home` / `End`	첫 행 / 마지막 행으로 이동	
`ArrowRight` / `ArrowLeft`	그룹 펼치기 / 접기	펼친 그룹에서 `ArrowRight` 는 그 첫 행으로, 그룹 안의 행에서 `ArrowLeft` 는 그 그룹으로 이동한다
`Enter` / `Space`	결과 열기. 클릭과 같다	열 대상이 없는 그룹 행은 펼치거나 접는다. Search Editor 탭의 결과 행도 이 키를 받는다

18.13 AI 워크플로우

AI 워크플로우의 키이다(21).

단축키	동작	비고
`Ctrl+S`	워크플로 문서 저장	처음 저장할 때 `workflow/WORKFLOW.io.workflow` 가 만들어진다
`Ctrl+Z`	실행 취소	포커스가 텍스트 필드에 있을 때는 제외(필드가 입력을 되돌린다)
`Ctrl+Y` / `Ctrl+Shift+Z`	다시 실행	`Ctrl+Z` 와 같다
`Delete` / `Backspace`	선택한 노드, 메모, 연결 삭제	캔버스에 포커스가 있어야 한다
`Shift`+클릭	노드를 선택에 넣거나 뺀다	
`Ctrl`+휠	포인터 위치에서 확대·축소	
휠 / `Shift`+휠	세로 / 가로로 화면 이동	빈 캔버스를 끌어도 이동한다

18.14 Meta Class 검색

Meta Class 검색 상자의 키이다(05.10). `Ctrl+T` 자체는 18.1에 있다.

단축키	동작	비고
`ArrowDown` / `ArrowUp`	선택 이동	포인터는 강조만 옮긴다
`Enter`	선택한 줄 열기	클릭과 같다
`Escape`	상자 닫기	상자 바깥을 클릭해도 닫힌다
`Ctrl+T`	캐럿을 검색 상자로 되돌리기	상자가 열려 있을 때

용어 사전

이 장은 PAIO 문서에 나오는 AUTOSAR 표준 용어와 PAIO 고유 용어를 모으고, 각 용어가 **PAIO의 어느 화면과 어느 파일에 대응하는지** 정리한다.

이 사전을 읽는 방법

- 표는 **용어 · 뜻 · PAIO에서 보는 곳** 세 열이다. 세 번째 열의 링크는 그 용어를 다루는 문서로 이어진다.
- 첫 열의 굵은 글씨는 표제어이고, 그 밖의 굵은 글씨는 번역하지 않는 UI 레이블이다. `백틱`은 파일 경로, 확장자, 키 이름, 열거값, 단축키를 뜻한다.
- AUTOSAR 요소 이름은 ARXML에 나오는 대문자 표기(`ECU-INSTANCE`)를 그대로 사용한다.
- UI에 보이지만 실제로는 동작하지 않는 항목은 그 사실을 함께 적었다.
- 처음 읽을 때는 19.1부터 본다. 19.2~19.7은 필요할 때 찾아본다.

19.1 PAIO 고유 용어

PAIO가 스스로 정의한 개념이다. AUTOSAR 표준에는 없다.

용어	뜻	
IO Model	PAIO의 1차 데이터 모델. 포맷 중립이며 사람이 읽을 수 있는 JSON이다. ARXML/DBC는 이 위에 얹힌 변환 계층이고 원본이 아니다	`design/` 아래 모든 모
IO-First	AUTOSAR가 아니라 IO Model이 진실의 원천이라는 PAIO의 설계 원칙. 덕분에 AUTOSAR가 표현하지 못하는 정보도 모델에 남는다	01. 제품 개요
워크스페이스 (Workspace)	PAIO 창 하나가 열어 둔 폴더. PAIO에서는 워크스페이스와 프로젝트가 같은 것이다. 창 하나가 워크스페이스 하나를 열고, 그 워크스페이스가 곧 프로젝트 하나이다. 창은 연 폴더에 단일 인스턴스 락을 건다	**PROJECT** > **Open Proj** **New Project** / **Open P** 스와 프로젝트
프로젝트 (Project)	설계 산출물 한 벌이며 폴더 하나에 담긴다. 폴더 이름이 프로젝트 이름이다. 프로젝트 메타 파일(`<프로젝트명>.io.prj`)과 6개 고정 폴더로 이뤄진다. 다른 프로젝트를 다루려면 다른 창에서 연다(**PROJECT** > **New Window**)	**New Project** 다이얼로 젝트명> · 04. 워크스페
고정 폴더 (Fixed Folder)	프로젝트 생성 시 반드시 만들어지는 6개 폴더: `doc`, `design`, `code`, `workflow`, `export`, `import`. `import` 를 뺀 나머지는 이름 변경·삭제가 불가능하다. `import` 는 일반 폴더 메뉴를 모두 갖고, 삭제되면 PAIO가 다시 만든다	**Explorer** 의 최상위 폴더 **EDITOR** 뷰의 **DOC**, **COI**
EDITOR 뷰 (예전 이름: design 트리)	설계 에디터를 여는 사이드바 뷰. **DESIGN** 절에 **System**, **VFB**, **Interface**, **DataType**, **Diagnostic** 이, **MODULE** 절에 **Definition**, **Configuration** 이 있고, 행을 클릭하면 그 에디터가 열린다. 그 아래 **DOC**, **CODE**, **EXPORT** 절은 파일을 보여 준다. **Signal** 행은 없다. Signal 에디터는 System 에디터 캔버스에서 연다	레일 맨 위의 **Design** 아
편집 이력 (History 패널)	창에 열린 프로젝트의 Undo/Redo/변경 여부/저장을 관리하는 단일 타임라인. 에디터마다 별도 Undo 스택을 두지 않으므로 어느 에디터의 편집이든 **EDIT** > **Undo** 로 되돌아간다	하단 유틸리티 패널의 **Hi**
에디터 탭	모델 노드나 파일 하나를 여는 단위. 제목은 `프로젝트명:노드명` 형식이고(예: `<프로젝트명>:System`, `<프로젝트명>:VFB`, `<프로젝트명>:Signal`) 동시에 최대 **20개**까지 열린다	에디터 영역 상단 · 05. 화
에디터 그룹 / 분할 보기	탭을 에디터 영역 가장자리로 끌어 만든 분할 창	에디터 영역 · 05. 화면 구
더티 표시 (●)	저장하지 않은 변경이 있는 탭에 붙는 점. 편집 이력의 기록에서 파생되므로, 기록이 남지 않은 변경은 표시되지 않는다	탭 제목 옆 · 05. 화면 구
라이선스 잠금	라이선스 상태에 따라 쓰기 · 생성 · 삭제 · 익스포트를 켜고 끄는 관문. 읽기와 임포트는 어떤 상태에서도 허용된다. 기본값은 **잠금(읽기 전용)** 이고 로그인과 라이선스 검증 후 열린다	차단 시 뜨는 업그레이드
ECU Extract / Machine Extract	한 노드(ECU 또는 HPC)와 그 노드가 참조하는 것만 잘라낸 부분 ARXML을 만드는 기능. 남은 요소가 끊어진 참조를 갖지 않도록 정리한다	System 에디터 캔버스 **Extract Machine** · 13.
QuickStart 워크플로	도메인 템플릿(7 도메인 24종)과 변수로부터 AI가 단계별로 프로젝트 전체 모델을 생성하는 기능. **v1.0.1에서는 이것을 여는 메뉴가 없다.** **PROJECT** 메뉴에 **Quick Start** 가 더는 없다	v1.0.1에서는 UI로 열 수
PARA Base Set	PopcornSAR AP 기본 구성(서비스 인터페이스, 데이터 타입, 플랫폼 Adaptive Application, `PARA` 컴포지션, 머신 내용)을 프로젝트에 주입하는 번들. ARXML 임포트와 같은 병합 파이프라인을 탄다	**Machine Configurati** **PARA: Import Platfor**
PRTE Base Set	PRTE Platform Import가 ECU 하나에 대해 프로젝트에 넣는 번들. PRTE 모듈 정의(VSMD), PRTE 서비스 인터페이스와 데이터 타입, 선택한 BSW 모듈마다의 서비스 컴포넌트이며 `Composition/PRTE_<ECU>` 와 `Service/<ECU>/<Module>` 아래에 놓인다. ARXML 임포트와 같은 병합 파이프라인을 탄다	**ECU Configuration** 다 (8.15)
PRTE Auto Configuration	설계가 답을 하나로 정해 주는 값만 써서 ECU 하나의 BSW 모듈 설정을 기록하는 엔진. 모든 변경을 미리 보여 주며 실행 취소 한 단계로 실행된다	**ECU Configuration** 다 (8.16), Module Config
CP-SOA	Classic Platform에서 SOME/IP 서비스 통신을 쓰기 위해 서비스 정의로부터 시그널·PDU·인터페이스를 파생하는 계산. 파생 결과는 모델 파일에 저장되며 출처 표시(`SOC` 배지)만 따로 붙는다. 표시일 뿐이므로 파생 항목도 이름·속성 변경과 삭제가 모두 허용된다	명령 팔레트 **Derive Sig** **DEPLOYMENT** 의 ⟳ 버 07.3
DEST 카탈로그	ECUC Foreign/Instance Reference의 `DESTINATION-TYPE`(AUTOSAR 메타타입 문자열)이 IO Model의 어떤 요소를 가리키는지 적어 둔 대응 표. 62종 메타타입을 세 등급(후보 제공 / 이름 합성 / 불투명)으로 나눈다	Module Configuratio
충돌 해결 (Conflict Resolution)	임포트한 요소가 기존 요소와 다른 값을 가질 때 항목별로 **Keep Project Version** / **Use Imported Version** / **Merge Fields** / **Keep Both** 중 하나를 고르는 절차. 정하지 않은 항목은 임포트 값으로 덮어써진다	임포트 중 뜨는 **Conflict**
원본 충돌 (Import Source Conflicts)	한 번에 고른 **여러 임포트 파일이** 같은 요소를 서로 다르게 정의할 때 뜨는 별도 화면. 기준 파일을 고르고 필드 단위로 덮어쓴다	**Import Source Confli**
`paio_*` 규칙	AUTOSAR 표준 제약으로 커버되지 않는 실무적 오류를 잡는 PAIO 자체 검증 규칙(예: `paio_ref_dangling`, `paio_someip_service_id_unique`)	**VALIDATION** 패널의 `Co`
`.vfb.layout`	VFB 캔버스의 배치 정보(위치·크기·뷰포트)만 따로 담는 파일. 모델 데이터와 분리되어 있다	`design/SWC/Layout`

용어	뜻	
IVC	In-Vehicle Communication. 차량 내 통신 토폴로지 모델을 가리키는 PAIO의 이름이며 `design/IVC.io.model` 에 저장된다. 이 모델은 System 에디터에서 편집하고, 에디터 자체는 더 이상 IVC라고 부르지 않는다	`design/IVC.io.mod`
System 에디터 (예전 이름: IVC 에디터)	네트워크 토폴로지(ECU/HPC 노드, 컨트롤러, 커넥터, 클러스터, 스위치)를 편집하는 에디터. 캔버스 왼쪽 가장자리에 **Signal**, **Partial Network (PNC)** 아이콘이, 오른쪽 가장자리에 **System** 버튼이 있다	**EDITOR** 뷰 ▸ **DESIGN**
SYSTEM 패널	SYSTEM 요소의 정보를 보여 주는 패널. **NAME**, **VERSION**, **DESCRIPTION** 에 더해 System 에디터에서는 **ECU MAPPING** 을, VFB 에디터에서는 **ROOT COMPOSITION** 과 **COMPOSITIONS** 를 보여 준다. 예전 System 대시보드를 대신한다	System 에디터 또는 VF 11. SYSTEM 패널
레일 (Activity Bar)	창 맨 왼쪽의 아이콘 열. 위에서부터 **Design**(**EDITOR** 뷰), **vECU**, **Workflow**, **Search** 이고 맨 아래에 **Explorer** 가 있다	창 맨 왼쪽 · 05. 화면 구
Diagnostic 에디터	진단을 설계하는 에디터. 다른 설계 에디터처럼 **DESIGN** 절에서 연다	**EDITOR** 뷰 ▸ **DESIGN** Diagnostic 에디터
New Project 다이얼로그	프로젝트를 만드는 다이얼로그. **Name** 은 프로젝트 이름이자 폴더 이름이고, **Location** 은 프로젝트를 만들 폴더이며(**Browse…** 로 고른다), **Create** 로 만든다	**Welcome** 탭의 **New Pr** 트
Search 뷰 (Search View)	프로젝트의 `design`, `doc`, `code` 폴더에 저장된 파일을 검색하는 사이드바 뷰. Design 결과는 그 값을 보여 주는 에디터 필드를 연다	레일의 **Search** 아이콘,
Search Editor	에디터 탭 안에서 따로 실행하는 검색. 제목은 `Search: <검색어>` 이고 읽기 전용이다	**Open New Search Ed**
Find Usages	한 요소를 가리키는 설계 참조를 나열한다	Design 결과 행 우클릭
Meta Class 검색 (Meta Class search)	AUTOSAR 메타 클래스 이름을 받아, PAIO가 그 클래스를 어디에서 설계하는지 위치마다 한 줄로 나열하는 상자(`Designed in <에디터> › …`). 추가하기 전에 있어야 하는 것도 함께 보여 준다. 프로젝트의 요소는 나열하지 않는다	`Ctrl+T`, 명령 팔레트 **G**
IntelliSense 항목	창의 C/C++ 언어 서버 상태(`Starting`, `Ready`, `Failed`, `Stopped`)를 보여 주는 상태바 항목. 그 메뉴에 **Restart IntelliSense** 가 있다	상태바 왼쪽 · 05.5.1
AI 워크플로우	입력 문서에서 설계, 코드 생성, 빌드, 검증까지 프로젝트를 이끄는 AI 노드와 도구 노드의 그래프. 프로젝트마다 문서 하나, `workflow/WORKFLOW.io.workflow`	레일의 **Workflow** 아이
지식 베이스 (LLM Wiki)	Input Analysis 노드가 프로젝트의 문서, 모델, 코드로 만드는 유형별·근거 기반 페이지. 이후의 워크플로 노드는 모두 이를 바탕으로 작업한다	`workflow/knowledg`
Rejected 루프	노드의 **Rejected** 출력에서 앞쪽 노드로 돌아가는 워크플로 연결. 실행은 결함을 소유한 노드를 결함 내용과 함께 다시 실행한다	Workflow 캔버스의 점

19.2 AUTOSAR 공통

플랫폼과 무관하게 ARXML 전반에 나오는 용어이다.

용어	뜻	
ARXML	AUTOSAR의 XML 교환 포맷. PAIO는 임포트와 익스포트 양방향을 모두 지원한다	익스포트: **EDITOR** 뷰 **DES** `export` 폴더에 생기고 **E**
AR-PACKAGE	ARXML 안의 요소를 담는 패키지(폴더에 해당). 요소의 절대 경로가 여기서 만들어진다. IO Model에는 AR-PACKAGE 개념이 없으므로 임포트 시 떼고 익스포트 시 규칙대로 다시 붙인다	익스포트 결과 ARXML · 13
SHORT-NAME	AUTOSAR 요소의 식별자. 첫 글자는 영문자, 이후 영문자/숫자/밑줄만 허용하며 최대 128자이다. PAIO는 규칙에 맞지 않는 이름을 **모델에 반영하지 않는다**(입력란은 오류만 표시하고 이전 값을 유지)	모든 에디터의 Name 항목
DESC	요소의 설명 텍스트. PAIO의 Description 항목이 여기에 대응한다	각 에디터의 Description
ADMIN-DATA	요소에 붙는 관리 정보 블록. PAIO는 여기에 SDG를 실어 도구 전용 정보를 기록한다	익스포트 결과 ARXML
SDG (Special Data Group)	AUTOSAR가 허용하는 확장 슬롯. PAIO는 표준 요소로 표현할 수 없는 소유 관계를 여기에 기록한다. 예: ECUC 설정이 어느 ECU의 것인지(`IO-ECUC-CONFIG-TO-ECU-MAPPING`), 컴포지션이 어느 Machine에 매핑됐는지(`IO-SWC-TO-MACHINE-MAPPING`)	익스포트 결과 ARXML · 12
ECU-INSTANCE	Classic Platform의 전자 제어 장치 요소. PAIO의 **ECU** 노드가 이것으로 나간다	System 에디터 캔버스의
MACHINE-DESIGN	Adaptive Platform 머신의 네트워크 측면(커넥터·주소 등)을 담는 요소. PAIO의 **HPC** 노드가 이것으로 나간다	System 에디터 캔버스의
SYSTEM	시스템 전체를 묶는 최상위 요소. FIBEX-ELEMENTS(구성 요소 목록), MAPPINGS(매핑), ROOT-SOFTWARE-COMPOSITIONS(루트 컴포지션)를 담는다	SYSTEM 패널(System 에 널
FibexElement	SYSTEM의 FIBEX-ELEMENTS 목록이 참조해야 하는 통신 관련 요소들의 총칭. 클러스터, 프레임, PDU, 시그널, ECU-INSTANCE, TP 설정 등이 해당한다	익스포트 결과 `<Project`
CATEGORY	요소의 종류를 표시하는 자유 문자열. PAIO는 SYSTEM에 `SYSTEM_DESIGN_DESCRIPTION`(전체) / `ECU_EXTRACT`(ECU 추출) / `MACHINE_DESIGN_EXTRACT`(Machine 추출) 중 하나를 넣는다	익스포트·Extract 결과 AR
DEST	참조(`*-REF`) 태그에 붙는 속성으로, 가리키는 대상의 요소 종류를 적는다	익스포트 결과 ARXML, EC
XSD (스키마)	ARXML의 구조를 정의하는 XML 스키마. PAIO는 R25-11(`AUTOSAR_00054.xsd`)과 R20-11(`AUTOSAR_00049.xsd`)을 번들한다	익스포트 결과 ARXML · 13
R25-11	PAIO가 **고정 출력**하는 AUTOSAR 스키마 릴리스. 모든 익스포트가 이 네임스페이스로 나간다	익스포트 결과 ARXML 헤
AUTOSAR 기능 버전	변환 계층이 별도로 다루는 축으로 `4.3` ~ `4.7`(기본 `4.7`)이다. 위의 스키마 릴리스와 독립이다	화면에 노출되지 않는다 · C
Classic Platform (CP)	ECU·BSW·RTE 기반의 전통적 AUTOSAR 플랫폼	System 에디터 캔버스의
Adaptive Platform (AP)	POSIX 기반 고성능 컴퓨팅용 AUTOSAR 플랫폼	System 에디터 캔버스의
constr_xxxx	AUTOSAR 명세가 부여한 제약 번호. PAIO의 검증 규칙 다수가 이 번호를 그대로 쓴다	**VALIDATION** 패널의 `Cons`

19.3 통신

System 에디터와 Signal 에디터가 다루는 영역이다.

토폴로지

용어	뜻	
Communication Cluster	같은 프로토콜을 쓰는 노드들이 붙는 논리 버스. CAN / Ethernet / LIN 세 종류가 있다	System 에디터 캔버스의 클러스터
Physical Channel	클러스터 내부의 물리 채널. 어떤 커넥터들이 연결됐는지를 담고, 이더넷 채널은 VLAN ID를 갖는다. 편집 지점은 Cluster Configuration 다이얼로그가 아니라 속성 패널이다	클러스터 선택 시 속성 패널의 **Phys**
Connector	노드(ECU/HPC)가 채널에 붙는 접점. 프로토콜별로 Ethernet / CAN / LIN이 있고, 각 커넥터는 컨트롤러 하나를 참조한다	System 에디터 캔버스의 ECU/HP
Communication Controller	노드의 통신 하드웨어에 해당하는 요소. 종류는 `ethernet`, `can`, `lin-master`, `lin-slave` 넷이다	System 에디터 캔버스의 ECU/HP
Coupling Element	이더넷 스위치/허브/라우터를 나타내는 요소. 종류는 `switch` / `hub` / `router` 이다	System 에디터 캔버스의 ECU/HP
Coupling Port	스위치의 포트, 또는 이더넷 컨트롤러가 소유한 포트. 포트끼리 잇는 것이 물리 케이블에 해당한다	스위치 우클릭 ▸ **Add CouplingP**
Network Endpoint	이더넷 커넥터의 IP 계층 설정(IPv4/IPv6 주소, 넷마스크, 게이트웨이, DNS, TTL 등)	커넥터 선택 시 속성 패널의 **Netwo**
Application Endpoint	TCP 또는 UDP 포트 하나의 정의. 서비스 인스턴스가 이 포트를 통해 통신한다	**이더넷 커넥터**를 선택했을 때만 나오 에는 나오지 않는다) · 06.8.2
Service Instance	어떤 서비스를 이 노드가 제공(`provider`)하는지 소비(`consumer`)하는지, 어느 포트를 쓰는지를 잇는 매핑. Interface 에디터에서 정의한 인스턴스를 참조한다	이더넷 커넥터 선택 시 속성 패널의
PLCA	10BASE-T1S 이더넷의 물리 계층 충돌 회피 설정. 임포트된 값을 보여 주기만 하고 편집 수단이 없다	포트-포트 연결선 선택 시 속성 패널

프레임 · PDU · 시그널

용어	뜻	
Frame	버스 위를 오가는 물리 전송 단위. PAIO는 CAN / LIN / FlexRay / Ethernet 네 종류를 모델로 갖지만 편집 탭은 **CAN** 과 **LIN** 두 개뿐이다	Signal 에디터(System 에디터 명 `>:Signal`)의 **CAN** / **LIN** 탭
DLC	CAN 프레임의 데이터 길이 코드. 고를 수 있는 값은 `0 (0 bytes)` ~ `15 (64 bytes)` 16개이며, 9~15는 CAN FD 확장 길이(12/16/20/24/32/48/64바이트)를 뜻한다	CAN 프레임의 **DLC** 항목 · 07.7
CAN 주소 방식 (Addressing)	`standard`(11비트 ID)와 `extended`(29비트 ID)	CAN 프레임의 **Addressing** 항
LIN Frame Type	`unconditional`(무조건) / `event-triggered`(이벤트) / `sporadic`(산발). 뒤 둘은 다른 unconditional 프레임을 참조한다	LIN 프레임의 **Frame Type** 항
LIN Checksum	`classic` 또는 `enhanced`	LIN 프레임의 **Checksum** 항목
PDU (Protocol Data Unit)	프레임 안에 실리는 프로토콜 데이터 단위. PAIO는 6종을 구분해 다룬다	Signal 에디터의 **PDU** 탭 · 07.5
I-Signal I-PDU	시그널들을 담아 주기적으로 보내는 가장 일반적인 PDU. 길이·주기(Cycle Time)와 시그널 매핑을 갖는다	**PDU** 탭 ▸ + ▸ **I-Signal I-PDU**
Multiplexed I-PDU	선택자(selector) 값에 따라 내부 배치가 바뀌는 PDU. 항상 있는 STATIC 부분과 전환되는 DYNAMIC 부분으로 나뉜다	**PDU** 탭 ▸ + ▸ **Multiplexed I**
NM-PDU	네트워크 관리 전용 PDU. NM 노드가 송신/수신 대상으로 참조한다	**PDU** 탭 ▸ + ▸ **NM-PDU** · 07.5
N-PDU	전송 계층(Transport Protocol)이 쓰는 세그먼트 운반용 PDU. CanTp/LinTp 연결이 데이터 PDU 또는 흐름 제어 PDU로 참조한다	**PDU** 탭 ▸ + ▸ **N-PDU** · 07.5
DCM-I-PDU	진단(Dcm) 데이터를 담는 PDU. 전송 계층이 분할해 나르는 대상이다	**PDU** 탭 ▸ + ▸ **DCM-I-PDU** ·
General Purpose PDU	용도를 CATEGORY로 구분하는 범용 PDU. SOME/IP 서비스 디스커버리 PDU가 여기에 해당한다	**PDU** 탭 ▸ + ▸ **General Purp**
I-Signal	PDU 안에 배치되는 시그널의 통신 계층 정체성. Signal 에디터의 **Name** 이 이 이름이다	Signal 에디터의 **SIGNAL** 탭 ·
System Signal	같은 시그널의 시스템 계층 정체성. 하나의 논리 시그널은 I-Signal과 System Signal 두 이름을 함께 갖고 익스포트된다. 데이터 매핑이 참조하는 쪽은 System Signal이다	Signal 모델의 `systemSigna`
Signal Group	함께 다뤄야 하는 시그널들의 논리 묶음. 멤버는 System Signal 이름으로 기록된다	Signals 그리드의 `GROUP` 열 ·
Text Table	시그널 값과 표시 레이블을 잇는 열거 표. 이 표가 있으면 초기값 입력이 콤보 상자가 된다. **현재 버전에서는 읽기 전용이며 에디터에서 바꿀 수 없다** — 항목은 ARXML 또는 DBC 임포트로만 만들어진다	Configuration 패널의 `Text`
Factor / Offset	원시값을 물리값으로 바꾸는 계수. `물리값 = 원시값 × Factor + Offset` 이다	Signals 그리드의 `FACTOR` /
Byte Order	바이트 순서. `big-endian`(상위 바이트 우선) 또는 `little-endian`	Signals 그리드의 `BYTE-ORDE`
Update Bit	시그널이 갱신됐음을 알리는 별도 비트의 위치	Signals 그리드의 `UPDATE-BI`
비트 레이아웃 (Preview)	PDU/프레임 안의 비트 배치를 색으로 보여 주는 그림. 겹치는 비트는 오류 색으로 표시된다	Signal 에디터 **CONFIGURATIO**

네트워크 서비스

용어	뜻	PAIO에
NM (Network Management)	버스의 절전/기동을 조율하는 네트워크 관리. PAIO는 CAN NM과 UDP NM을 다룬다	Node Configuration ▸ **NETWORK MANAGEMENT** 탭, Cl **CLUSTER** 탭 · 06.7.3, 06.7.4
NM Cluster	같은 NM 규칙을 공유하는 노드 집합. 클러스터 수준 NM 파라미터를 담는다	Cluster Configuration의 NM 탭 · 06.7.4
NM Node	한 ECU를 한 NM Cluster에 붙이는 요소	Node Configuration ▸ **NETWORK MANAGEMENT** 탭 · 0
NM Coordinator	여러 NM 클러스터의 절전을 조율하는 역할. 채널별로 `ACTIVE` / `PASSIVE` / `NONE` 을 갖는다	Node Configuration의 NM 코디네이터 다이어그램 · 06.7.3
Transport Protocol (TP)	한 프레임에 담기지 않는 큰 데이터를 나눠 보내는 계층. PAIO는 **CanTp** 와 **LinTp** 를 다룬다	Cluster Configuration ▸ **TRANSPORT PROTOCOL** 탭 · 0
주소 지정 방식 (Addressing Format)	CanTp 연결의 ISO 15765-2 주소 모드 — `STANDARD` / `EXTENDED` / `MIXED` / `MIXED29BIT` / `NORMALFIXED`	**TRANSPORT PROTOCOL** 탭 · 06.7.4
통신 유형 (Target Address Type)	`PHYSICAL` (1:1) 또는 `FUNCTIONAL` (1:N)	**TRANSPORT PROTOCOL** 탭 · 06.7.4
Global Time	노드 간 시각 동기화. 이더넷은 IEEE 802.1AS(gPTP), CAN은 AUTOSAR 고유 방식이다	Node Configuration / Cluster Configuration ▸ **TIME D**
Time Domain	하나의 동기화 영역. 마스터·슬레이브·게이트웨이 역할과 CRC 정책을 갖는다	**TIME DOMAIN** 탭 · 06.7.4
SOME/IP	Adaptive Platform의 서비스 지향 통신 프로토콜. 서비스 ID와 이벤트/메서드/필드별 식별자를 배포 정보로 갖는다	Interface 에디터 **SERVICE DEPLOYMENT** 패널의 **SOME/IP**
DDS	또 다른 서비스 지향 통신 미들웨어. 토픽 이름과 서비스 ID 등을 배포 정보로 갖는다	Interface 에디터 **SERVICE DEPLOYMENT** 패널의 **DDS Dep**

TSN 이더넷 스위치

용어	뜻	
VLAN	하나의 물리 이더넷을 논리적으로 나누는 가상 랜. 채널마다 VLAN ID를 갖고, 포트마다 송신 방식(`TAGGED` / `UNTAGGED` / `NOT SENT`)을 정한다	**Ethernet Switch Configu MEMBERSHIP** 절 · 06.7.5
PCP (Priority Code Point)	VLAN 태그에 실리는 0~7의 우선순위 값	Port 파이프라인의 **PRIORIT**
Priority Regeneration	수신한 프레임의 PCP를 다른 값으로 바꿔 쓰는 매핑	Port 파이프라인의 **PRIORIT**
Traffic Class	PCP 값들을 묶어 배정하는 송신 큐의 등급	Port 파이프라인의 **TRAFFIC**
FIFO	하나의 트래픽 클래스를 담당하는 송신 큐. 셰이퍼를 붙일 수 있다	EGRESS 큐 그래프의 FIFO 노
CBS (Credit-Based Shaper)	IEEE 802.1Qav의 크레딧 기반 셰이퍼. `idleSlope` (초당 크레딧 증가량)를 갖는다	FIFO 우클릭 ▸ **Attach Shap**
ETS (Enhanced Transmission Selection)	IEEE 802.1Qaz의 대역 배분 셰이퍼/스케줄러. 가중치를 퍼센트 또는 weight 값으로 지정한다	FIFO 우클릭 ▸ **Attach Shap**
ATS (Asynchronous Traffic Shaper)	IEEE 802.1Qcr의 셰이퍼. 모델과 임포트는 지원하지만 **UI로 새로 만들 수는 없다**(스위치 수준 TSN 설정으로 대체됨). 임포트된 값은 표시·보존된다	Port 파이프라인에 `ATS` 로
Frame Preemption	긴 프레임을 잘라 급한 프레임을 먼저 보내는 기능. 트래픽 클래스마다 `EXPRESS` 또는 `PREEMPTABLE` 을 지정한다	EGRESS 큐 그래프의 FIFO 설
Scheduler	여러 FIFO 중 어느 것을 먼저 내보낼지 정하는 요소. `STRICT PRIORITY` 또는 `ETS` 이다	EGRESS 큐 그래프의 스케줄

19.4 소프트웨어 구조

VFB 에디터와 Interface 에디터가 다루는 영역이며, 컴포지션을 ECU/HPC 노드에 매핑하는 일도 포함한다.

용어	뜻	
VFB (Virtual Functional Bus)	하드웨어 배치를 생각하지 않고 소프트웨어 컴포넌트의 연결만 다루는 추상 계층	**EDITOR** 뷰 ▸ **DESIGN** ▸ **VFB**, 탭 <프
SWC (Software Component)	소프트웨어 컴포넌트. 포트를 통해서만 외부와 통신한다	VFB 캔버스의 노드 · 08.2
Composition	다른 컴포넌트를 담는 컨테이너형 SWC. 자기 자신도 포트를 갖는다	빈 캔버스 우클릭 ▸ **Add Compositio**
Atomic SWC	더 쪼갤 수 없는 실행 단위 SWC. 종류는 **Application**, **Complex Device Driver**, **Ecu Abstraction**, **Sensor Actuator**, **Service**(BSW 서비스), **Adaptive Application** 이다. **Add SWC** 메뉴는 여기에 **Composition** 을 더해 7종을 제공한다	컴포넌트 안쪽 우클릭 ▸ **Add SWC** · 0
Interface Mapping 노드	AP 서비스 인터페이스와 CP의 SR/CS 요소를 잇는 전용 노드. SWC 종류가 아니라 별도의 노드 종류이다	루트 레벨 빈 캔버스 우클릭 ▸ **Add Int**
Prototype	SWC **타입**을 컴포지션 안에 실제로 놓은 인스턴스. 같은 타입을 여러 프로토타입으로 여러 번 놓을 수 있다	컴포넌트 안쪽 우클릭 ▸ **Attach SWC** 로토타입 이름, 타입 이름이 다르면 타입
드릴다운	컴포지션 안으로 들어가 내부를 편집하는 것. 화면 상단 브레드크럼이 현재 위치를 보여 준다	컴포넌트 더블클릭 · 08.4
Port	SWC가 외부와 주고받는 접점. 방향은 **PPort**(제공) / **RPort**(요구) / **PRPort**(양방향) 셋이다	노드 우클릭 ▸ **Add Port** · 08.5
Port Interface	포트가 주고받는 내용의 계약. 포트는 인터페이스 하나를 참조한다	Interface 에디터 · 09. Interface 에디
Sender-Receiver (SR)	데이터 요소(Data Element)를 단방향으로 보내고 받는 인터페이스	Interface 에디터의 **Sender-Receive**
Client-Server (CS)	오퍼레이션을 호출하고 결과를 받는 인터페이스. 인자는 **IN** / **OUT** / **INOUT** 방향을 갖는다	Interface 에디터의 **Client-Server** 탭
Mode-Switch	모드 전환을 알리는 인터페이스. ModeDeclarationGroup 이름, 초기 모드, 모드 선언 목록을 갖는다	Interface 에디터의 **Mode-Switch** 탭
Service Interface	AP의 서비스 인터페이스. 이벤트(Events) / 메서드(Methods) / 필드(Fields)와 C++ 네임스페이스, 버전을 갖는다	Interface 에디터의 **Service** 탭 · 09.3
Data Element	Sender-Receiver 인터페이스가 나르는 데이터 하나. 이름과 데이터 타입을 갖는다	**Sender-Receiver** 탭 · 09.3.2
Operation	Client-Server 인터페이스가 제공하는 호출 하나	**Client-Server** 탭 · 09.3.3
ComSpec	포트별 통신 속성. 송/수신, 큐 사용 여부에 따라 종류가 나뉘고 초기값·타임아웃·전송 모드 등을 갖는다	포트 선택 시 속성 패널의 ComSpec 섹
Internal Behavior	Atomic SWC의 내부 동작 정의. Runnable, RTE 이벤트, Inter-Runnable Variable을 담는다	**SWC Internal Behavior** 다이얼로그 ·
Runnable (Runnable Entity)	RTE가 호출하는 실행 단위. `symbol`(생성 코드의 C 함수 이름)과 최소 시작 간격 등을 갖는다	**SWC Internal Behavior** 다이얼로그의
RTE Event	Runnable을 깨우는 계기. 메뉴가 제공하는 종류는 7종 — **TimingEvent**, **BackgroundEvent**, **DataReceivedEvent**, **OperationInvokedEvent**, **SwcModeSwitchEvent**, **InitEvent**, **InternalTriggeredEvent**(AUTOSAR 이름은 `InternalTriggerOccurredEvent`)	**SWC Internal Behavior** 다이얼로그의
Access Point	Runnable이 포트나 변수에 접근하는 지점. 9종 — **DataSendPoint**, **DataReceivedPoint**, **SyncServerCallPoint**, **DataWriteAccess**, **DataReadAccess**, **WrittenLocalVariable**, **ReadLocalVariable**, **ModeSwitchPoint**, **ModeAccessPoint**	**SWC Internal Behavior** 다이얼로그의
InternalTriggeringPoint	SWC 내부에서 다른 Runnable을 깨우는 트리거 발원지. 소유 Runnable에 속한다. **Access Points** 메뉴에 함께 들어 있지만 VARIABLE-ACCESS가 아닌 별도 요소로 저장된다	**SWC Internal Behavior** 다이얼로그의
Inter-Runnable Variable (IRV)	한 SWC 안의 Runnable들이 공유하는 변수	**SWC Internal Behavior** 다이얼로그
SWC ↔ ECU 매핑	어떤 컴포지션을 어떤 ECU/HPC에 배치할지 정하는 관계. 어느 쪽에서든 편집할 수 있고 두 쪽이 서로 동기화된다	System 에디터: 노드 우클릭 ▸ **SW M** 터: 컴포넌트 우클릭 ▸ **ECU/HPC Map** **MAPPING** 에 나온다 · 11. SYSTEM 패
Data Mapping	포트의 데이터 요소를 통신 시그널(또는 시그널 그룹)에 잇는 매핑. System 모델에 기록된다	VFB에서 **컴포지션 경계 포트**를 선택했
InterfaceMapping	AP 서비스 인터페이스의 이벤트/메서드/필드를 CP의 데이터 요소/오퍼레이션에 대응시키는 정의. **저장 위치는 VFB 모델이다** (Interface 모델이 아니다)	Interface 에디터의 **Mapping** 탭 · 09
Root Composition	시스템의 최상위 컴포지션. SYSTEM 요소가 이것을 참조한다	VFB 에디터 SYSTEM 패널의 **ROOT CO**

19.5 데이터 타입

DataType 에디터가 다루는 영역이다.

용어	뜻	
Application Data Type	물리 의미(단위, 범위) 기준의 데이터 타입. 익스포트 시 `APPLICATION-PRIMITIVE-DATA-TYPE` 등으로 나간다	DataT
Implementation Data Type	구현(C 언어) 기준의 데이터 타입. `IMPLEMENTATION-DATA-TYPE` 으로 나간다	익스포
StdCpp Implementation Data Type	C++ 기준의 구현 데이터 타입. `STD-CPP-IMPLEMENTATION-DATA-TYPE` 으로 나가며, 프리미티브는 이 자리에서만 C++ 표기(`uint8_t` 등)로 바뀐다	익스포
SW-BASE-TYPE	비트 크기·인코딩·네이티브 C 선언을 담는 기반 타입	익스포 지 · 13.
CompuMethod	원시값과 물리값(또는 텍스트)을 잇는 변환 규칙. PAIO는 ENUMERATION에서 `TEXTTABLE`, ALIAS에서 `LINEAR` 형태로 **자동 생성**하며 이름은 `CM_<타입명>` 이다	익스포
Unit	물리 단위. 표시 이름과 SI 변환 계수/오프셋을 갖는다. 기본 단위 33종이 미리 들어 있다	DataT
DataConstr	값 범위 제약. ALIAS 타입에 최소/최대가 있을 때만 `DC_<타입명>` 으로 자동 생성된다	익스포
DataTypeMappingSet	Application 타입과 구현 타입을 짝지어 두는 집합. PAIO는 `DataTypeMappingsCP`(C용)와 `DataTypeMappingsAP`(C++용) 둘을 자동으로 만든다	익스포
카테고리 (Category)	데이터 타입의 종류. 9종 — `PRIMITIVE`, `ALIAS`, `STRUCTURE`, `UNION`, `ENUMERATION`, `FIXED_ARRAY`, `DYNAMIC_ARRAY`, `STRING`, `MAP`	DataT
내장 프리미티브	PAIO가 예약해 둔 11개 기본 타입 — `boolean`, `float32`, `float64`, `sint8/16/32/64`, `uint8/16/32/64`. 프로젝트를 열 때 메모리에 자동 주입되며 **모델 파일에는 저장되지 않는다**	DataT
ALIAS	다른 타입에 붙이는 별칭. 여기에서만 Factor/Offset/Min/Max/Unit을 함께 지정한다	DataT
Init Value (초기값)	요소의 초기값. PAIO는 모든 초기값 화면에서 하나의 공용 위젯을 쓰며, 입력 형태는 대상 타입 구조에서 자동으로 결정된다	Signa 섹션 · (
ValueSpecification	단일 값뿐 아니라 구조체·배열·맵 형태의 초기값을 표현하는 AUTOSAR 구조. 알 수 없는 형태는 **원문 그대로 보존**되고 읽기 전용 뷰어로 표시된다	초기값

19.6 Adaptive Platform

VFB 에디터의 **AA Internal Behavior · Machine Configuration** 다이얼로그와 Interface 에디터의 **Adaptive** 탭이 다루는 영역이다.

기본 개념

용어	뜻	
HPC	High-Performance Computer. PAIO에서 Adaptive Platform 노드를 뜻하며 `MACHINE-DESIGN` 으로 익스포트된다	System 에디터 빈 캔버스
Machine	하나의 AP 실행 환경. 프로세서/코어, OS, Function Group, 플랫폼 모듈 인스턴스를 담는다	컴포지션의 HPC 아이콘 클 **FUNCTION GROUP**, **PROC**
Function Group	머신의 동작 상태를 묶어 정의하는 그룹. 각 그룹은 상태 머신(모드 선언 + 전이)을 갖는다	Machine Configuration
MachineFG	모든 머신이 반드시 갖는 기본 Function Group. 기본 상태 머신 이름은 `MachineState` 이고 초기 모드는 `Off`, 선언 상태는 `Off` / `Startup` / `Shutdown` / `Restart` / `Verify` 이다	Machine Configuration
Mode Declaration	상태 머신의 한 상태	**FUNCTION GROUP** 탭의 상
Adaptive Application (AA)	AP의 애플리케이션 SWC. Executable과 Process 설정을 함께 갖는다	VFB 캔버스에서 더블클릭
Executable	빌드 산출물 하나. 버전, 카테고리, 빌드 타입(`BUILD-TYPE-DEBUG` / `BUILD-TYPE-RELEASE`), 루트 SWC 프로토타입, 최소 타이머 해상도를 갖는다	**AA Internal Behavior** ▸
Process	Executable을 실제로 실행하는 프로세스. 시작 조건(Function Group 상태), 실행 의존성, 로그 설정, 코어 친화도를 갖는다	**AA Internal Behavior** ▸ 08.9
실행 의존성 (Execution Dependency)	다른 프로세스가 특정 상태(`Idle` / `Starting` / `Running` / `Terminated`)일 때만 시작하도록 하는 조건	**AA Internal Behavior** ▸
Module Instantiation	머신에 올린 플랫폼 모듈 인스턴스. 모듈 종류마다 전용 설정 화면을 갖는다	Machine Configuration
Error Domain	AP 애플리케이션 오류 코드를 묶는 도메인	Interface 에디터의 **Error**
Software Cluster	Adaptive Platform 배포 단위. 함께 배포되는 Adaptive Application을 묶고, 소프트웨어 패키지로 전달되며 차량 패키지로 롤아웃된다. `design/SOFTWARE_CLUSTER.io.model` 에 저장된다	VFB 액션 줄의 **Software**
IdsM instance	노드 하나의 Intrusion Detection System Manager 인스턴스. 필터, 한도, 보고하는 보안 이벤트를 담는다. `design/IDS.io.model` 에 저장된다	VFB 액션 줄의 **Intrusion** 로그의 **IDSM** 탭 · 08.12.4

기능 클러스터

기능 클러스터별 포트 인터페이스는 Interface 에디터의 **Adaptive** 탭에서 만들고, 머신 쪽은 Machine Configuration 다이얼로그의 **MODULE** 탭에서 설정한다. **Adaptive** 탭의
음), **MODULE** 탭의 모듈은 **10종**(RDS는 대응 모듈이 없음)이다. 표는 둘을 함께 나열한다.

클러스터	뜻	
ara::log (LOG, Log and Trace)	로그와 트레이스. DLT 애플리케이션 ID, 로그 모드(`CONSOLE`/`FILE`/`NETWORK`), 로그 레벨(`OFF`~`VERBOSE`)을 다룬다	Interface ▸ **Ac** **LOG** · 09.3.5, 08
CM (Communication Management)	서비스 지향 통신 관리	Machine ▸ **MC**
ara::sm (SM, State Management)	상태 관리. 인터페이스 종류는 StateClient, DiagTrigger, EmError, PhmError 넷이다	Interface ▸ **Ac** 션·요청/알림·배신
ara::phm (PHM, Platform Health Management)	플랫폼 상태 감시. 인터페이스는 PhmSupervisedEntity와 PhmRecoveryNotification 둘이다	Interface ▸ **Ac** 트·전이 다이어그
ara::nm (NM, Network Management)	AP의 네트워크 관리. 인터페이스는 NetworkManagement 하나이며 머신 쪽에서는 NetworkHandle을 다룬다	Interface ▸ **Ac** 08.10
ara::tsync (TS, Time Synchronization)	시각 동기화. 인터페이스는 SynchronizedTimeBaseConsumer와 SynchronizedTimeBaseProvider 둘이다	Interface ▸ **Ac** 08.10
ara::crypto (CRY, Cryptography)	암호. 인터페이스는 CryptoKeySlot, CryptoKeySlotClient, CryptoCertificate, CryptoCertificateGroup, CryptoProvider 다섯이다	Interface ▸ **Ac** 08.10
ara::per (PER, Persistency)	영속 저장. 인터페이스는 PersistencyKeyValueStorage와 PersistencyFileStorage 둘이고, 머신이 실제 저장소를 소유한다	Interface ▸ **Ac** 08.10
ara::rds (Raw Data Stream)	원시 데이터 스트림. 인터페이스는 RawDataStreamClient, RawDataStreamServer, Ieee1722RawDataStreamConsumer, Ieee1722RawDataStreamProducer 넷이다. **MODULE** 탭에는 대응 모듈이 없고 AA 쪽 매핑으로 다룬다	Interface ▸ **Ac** 09.3.5, 08.9
ara::idsm (IDS, Intrusion Detection)	침입 탐지. 인터페이스는 SecurityEventReport, IdsmTimestampProvider, IdsmContextProvider, IdsmReportingModeProvider, IdsmQualifiedEventReceiver 다섯이다	Interface ▸ **Ac** 결과 `<Project`
ara::ucm (UCM, Update and Configuration Management)	갱신·구성 관리. 전용 포트 인터페이스는 없고 머신 모듈 설정으로만 다룬다(Master / Subordinate)	Machine ▸ **MC**
DLT	Diagnostic Log and Trace. 로그 전송 규격이며 애플리케이션 ID와 컨텍스트 ID로 로그를 분류한다	Machine ▸ **MC** ▸ **EXECUTION**

Adaptive 인터페이스 배지

Interface 에디터의 **Adaptive** 탭 목록에서 인터페이스 이름 뒤에 붙는 배지이다 (09.3.5).

배지	인터페이스
`SC` / `DT` / `EM` / `PHM`	StateClient / DiagTrigger / EmError / PhmError
`PHM/SE` / `PHM/RN`	PhmSupervisedEntity / PhmRecoveryNotification
`NM`	NetworkManagement
`TS/C` / `TS/P`	SynchronizedTimeBaseConsumer / Provider
`CR/KS` / `CR/KSC` / `CR/CERT` / `CR/CG` / `CR/PRV`	CryptoKeySlot / KeySlotClient / Certificate / CertificateGroup / Provider
`RDS/C` / `RDS/S` / `RDS/1722C` / `RDS/1722P`	RawDataStream 4종
`PER/KV` / `PER/FS`	PersistencyKeyValueStorage / FileStorage
`IDSM/SEV` / `IDSM/TS` / `IDSM/CTX` / `IDSM/RM` / `IDSM/QEV`	IDSM 5종
`LnT`	LogAndTrace

19.7 ECUC

Classic Platform BSW 모듈 설정을 다루는 영역이다. 두 개의 에디터가 짝을 이룬다.

용어	뜻	
ECUC (ECU Configuration)	BSW 모듈의 설정 체계. "어떤 파라미터가 있는가"(정의)와 "그 값이 무엇인가"(설정)를 분리해 다룬다	**EDITOR** 뷰 ▸ **MODULE** ▸
Module Definition (`.def`)	모듈이 가질 수 있는 컨테이너·파라미터·레퍼런스의 구조를 선언한 문서. 값은 담지 않는다	Module Definition 에디
Module Configuration (`.conf`)	정의를 따라 실제 값을 채운 문서. 반드시 어떤 정의를 참조하며 ECU별로 보관된다	Module Configuration
Container	파라미터와 레퍼런스를 담는 묶음. 정의에는 `ParamDef`(일반)와 `Choice`(택일) 두 종류가 있다	DEFINITION 트리 우클릭
Container Instance	정의된 컨테이너를 설정에서 실제로 하나 만든 것. 이름을 붙이고 복제할 수 있다	컨테이너 트리 ▸ **+** (Add
Parameter	값을 담는 항목. 정의에서 고를 수 있는 종류는 9가지 — Integer, Float, Boolean, Enumeration, String, Function Name, Linker Symbol, Multiline String, Add Info	DEFINITION 트리 우클릭
Reference	다른 컨테이너 인스턴스를 가리키는 항목. 정의 화면에서 만들 수 있는 종류는 4가지이다(아래 4행)	DEFINITION 트리 우클릭
Reference (단순)	`destination` 에 적힌 컨테이너 정의 경로에 맞는 인스턴스 하나를 가리킨다	CONFIGURATION 패널의
Choice Reference	`destination` 을 여러 개 갖는다. 각각을 따로 해석해 후보를 합쳐 보여 준다	CONFIGURATION 패널의
Foreign Reference	ECUC 바깥의 AUTOSAR 요소를 `DESTINATION-TYPE`(메타타입 이름)으로 가리킨다	DEST 카탈로그가 아는 타
Instance Reference	바깥 요소를 **문맥 체인과 함께** 가리킨다. `DESTINATION-CONTEXT` 에 거쳐 가는 요소 종류들이 공백으로 구분되어 적힌다	CONFIGURATION 패널의
Symbolic Name Reference	AUTOSAR 4.4.0에서 제거된 옛 레퍼런스 종류. **임포트만 지원**하며 정의 에디터의 **Add Reference** 메뉴에는 나오지 않는다. 익스포트할 때는 일반 Reference로 나간다	임포트된 모듈 정의 안에서
destination	레퍼런스가 가리키는 대상의 경로. PAIO 안에서는 AR-PACKAGE 없이 `모듈명/컨테이너명[/하위 ...]` 형식이다	PROPERTIES 패널의 des
DESTINATION-TYPE	Foreign/Instance Reference가 가리키는 AUTOSAR 메타타입 이름(예: `RUNNABLE-ENTITY`). PAIO는 DEST 카탈로그로 이를 IO Model 요소와 연결한다	PROPERTIES 패널, CONF
DESTINATION-CONTEXT	Instance Reference에서 대상까지 거쳐 가는 요소 종류의 체인	PROPERTIES 패널 · 12.1
Multiplicity (다중도)	항목이 몇 개까지 존재할 수 있는지. 하한/상한으로 적으며 상한 무제한은 `inf` 이다	PROPERTIES 패널의 다중
Origin	그 정의 항목을 만든 주체를 적는 문자열(예: 표준명 또는 벤더명)	PROPERTIES 패널의 Orig
Config Class	그 값을 언제 확정해야 하는지 나타내는 등급 — `PRE-COMPILE`, `LINK`, `POST-BUILD`, `PUBLISHED-INFORMATION`. 변형(Variant)과 짝지어 기록된다	PROPERTIES 패널의 Valu 배지(표시 전용) · 12.1, 12.
Config Variant	설정이 확정되는 시점의 구성 방식 — `VARIANT-PRE-COMPILE`, `VARIANT-LINK-TIME`, `VARIANT-POST-BUILD`, `PRECONFIGURED-CONFIGURATION`, `RECOMMENDED-CONFIGURATION`	정의: **Supported Config** 12.2
Post-Build Variant Used	이 설정이 포스트빌드 변형을 쓰는지 여부. 설정 안 함 / `true` / `false` 3상태이며 설정 안 함이 임의로 `false` 로 바뀌지 않는다	Module Configuration
`EcucDefinitionCollection`	익스포트 시 모듈 정의들을 묶는 요소	익스포트 결과 `<Project`

문제 해결 · FAQ

이 장에서 다루는 것

PAIO에서 자주 마주치는 증상을 원인, 화면 메시지, 해결 방법과 함께 정리한다. 예상과 다르게 동작할 때 이 장을 본다. 각 기능의 정상적인 사용법은 해당 기능 문서에 있다. 화면과 실
은 19. 용어 사전에 있다.

> 인용된 메시지는 앱이 출력하는 영문 원문 그대로이다. 화면에서 본 문구로 이 장을 검색하면 된다. 권한 관련 증상은 라이선스 잠금 항목을, 임포트 증상은 충돌 해결 항목을 함께 본

20.0 증상으로 찾기

아래 표에서 증상을 찾은 뒤 안내한 절로 이동한다. 다른 증상은 20.1부터 영역별로 정리되어 있다. 영문 메시지로 찾으려면 20.13을 본다.

증상	짧은 답	자세히
컴포지션 안 빈 캔버스를 우클릭해도 메뉴가 없다	정상이다. 빈 곳이 아니라 **컴포지션 프레임**을 우클릭한다	08. VFB 에디터 8.4
검증 실행 버튼을 못 찾겠다	버튼이 없다. 편집할 때마다 자동으로 실행된다	14. Validation 14.1
저장을 눌렀는데 탭의 점(●)이 남아 있다	라이선스 만료이거나 쓰기 실패이다	20.3
`design` · `workflow` · `export` 폴더의 이름을 바꾸거나 지울 수 없다	정상이다. 프로젝트의 고정 폴더에는 Rename · Delete 항목이 없다	20.2
`export` 폴더의 ARXML을 고쳤는데 반영되지 않는다	`export` 는 결과물 폴더이다. 에디터에서 고치고 다시 내보낸다	20.5
IVC 에디터를 찾을 수 없다	지금은 **System 에디터**이다. **DESIGN** 목록의 **System** 을 누르며, 탭 제목은 `<프로젝트>:System` 이다	06. System 에디터
DESIGN 목록에 **Signal** 이 없다	Signal 에디터는 System 에디터 캔버스 왼쪽 가장자리의 **Signal** 아이콘으로 연다	07. Signal 에디터
시그널이 빨갛게 `Overlap` 으로 표시된다	두 시그널이 같은 비트를 쓴다. **START** 를 조정한다	07. Signal 에디터 7.
커넥터를 이으려 했는데 아무 반응이 없다	프로토콜이나 소속 ECU가 맞지 않는다. 메시지는 나오지 않는다	06. System 에디터
FlexRay 프레임을 만들 수 없다	현재 버전에는 FlexRay·Ethernet 탭이 없다	02. 제품 기본 사양 2
이름을 입력했는데 자꾸 원래대로 돌아간다	AUTOSAR 이름 규칙 위반이다	20.3
탭이 더 이상 열리지 않는다	동시에 열 수 있는 탭은 20개이다	20.3
화면을 좌우로 나누고 싶은데 메뉴가 없다	메뉴가 없다. 탭을 본문 오른쪽 절반으로 끌어다 놓는다	05. 화면 구성과 조작
AI가 만든 모델을 그대로 써도 되는지 모르겠다	초안이다. 각 에디터와 검증 패널로 반드시 확인한다	15. AI 어시스턴트 15.
ECU 속성 패널에 **ECUC GENERATION** 섹션이 없다	BSW 설정은 **PRTE AUTO CONFIGURATION** 이나 **PRTE Auto Configuration** 아이콘으로 생성한다	20.6
방금 입력한 값이 검색되지 않는다	검색은 저장된 파일만 읽는다. 저장하거나 **Save All & Search** 를 고른다	20.15
검색 결과를 클릭했는데 에디터 대신 읽기 전용 `*.io.model` 이 열렸다	그 값을 보여 주는 에디터 필드가 없다. 의도된 동작이다	20.15
워크플로 실행이 시작되지 않거나 워크플로 노드가 실패한다	토스트나 노드의 결과를 읽는다. 자주 보는 메시지를 정리했다	20.14
로컬 Windows 창에서 **Build ECU** 가 거부된다	ECU는 vECU에서 빌드하고 실행한다. 먼저 vECU를 연다	20.10

20.1 실행 · 로그인 · 라이선스

앱을 켰는데 작업 화면이 나오지 않고 로그인 화면만 보인다

정상 동작이다. PAIO는 로그인한 뒤에야 메인 레이아웃을 만든다. 다음 중 하나를 쓴다.

1. **Continue with Google** 을 누른다. 기본 브라우저에서 Google 로그인이 열린다.
2. `Email address` 에 이메일을 입력하고 **Continue** 를 누른다. 6자리 OTP 화면이 열린다.

로그인이 끝나면 오버레이가 사라지고 헤더, 왼쪽 사이드바, 에디터 영역이 나타난다.

로그인은 됐는데 편집이 되지 않고 업그레이드 다이얼로그만 나타난다

라이선스가 만료(`expired`) 또는 읽기 전용(`readonly`) 상태이다. 두 상태에서는 읽기와 임포트만 된다.

라이선스 상태	읽기	임포트	편집·생성·삭제	익스포트
`active` (Pro / Enterprise)	O	O	O	O
`trial` (체험 기간 내)	O	O	O	O
`expired`	O	O	X	X
`readonly` (무결성 검사 실패)	O	O	X	X

상태바 오른쪽의 라이선스 표시가 상태를 보여 준다.

표시	의미
`License Expired` (경고 아이콘)	만료 또는 읽기 전용
`Trial: 12d` (또는 `Trial` 만)	체험판, 남은 일수
`Pro - 5 days` (경고 아이콘)	만료 7일 이내
`Pro` / `Enterprise` (왕관 아이콘)	정상

이 표시는 클릭할 수 있다(툴팁 `Click to manage license`). 03. 설치 · 로그인 · 라이선스를 본다.

거부 다이얼로그의 제목은 라이선스에 따라 다르다.

라이선스	제목
체험 기간 중이거나, 개별 작업 하나가 거부됨	**Upgrade Required**
체험 기간이 끝남	**Trial Ended**
플랜 없음	**Subscription Required**
유료 플랜이 만료됨	**Subscription Expired**

Not now 나 닫기 버튼(×)으로 닫는다. **View plans** 는 플랜 안내 페이지를 연다.

> **다이얼로그가 반복해서 나타나는 경우**: 만료 상태에서는 편집을 시도할 때마다 나타난다. 한 번에 하나만 열리므로 쌓이지는 않는다.

오프라인으로 오래 두었더니 익스포트가 거부된다

라이선스 캐시의 유효 기간은 **4시간**이다. 그동안 서버에 연결하지 못하면 익스포트가 `OFFLINE_EXPORT_DENIED: A server connection is required` 로 거부된다. 이미
오프라인으로 시작했고 캐시도 만료되었다면 편집과 저장도 막힌다. 네트워크를 복구하고 창을 다시 활성화(포커스)하면 재검증이 실행되어 익스포트가 다시 된다.

작업 중에 만료로 바뀌면 다른 형태의 다이얼로그가 나타난다. 제목은 **Trial Ended**, **Subscription Required**, **Subscription Expired** 중 하나이고, ×와 **Not now** 가 없어 닫을 수
에 **Sign out** 버튼이 있다. 라이선스가 다시 유효해지면 저절로 닫히므로 앱을 다시 시작할 필요는 없다.

갑자기 로그아웃되었다

다른 기기에서 같은 계정으로 로그인한 경우이다. 다이얼로그 제목은 **Signed in on Another Device**, 본문은 `You've been signed out because your account was us`

- 저장하지 않은 탭이 없으면 **OK** 버튼 하나만 있고, 누르면 로그아웃된다.
- 저장하지 않은 탭이 있으면 그 이름이 표시되고 **Save & Sign Out** / **Sign Out** 중에서 고른다. 저장에 실패하면 **Save Failed** 다이얼로그가 재시도 여부를 묻는다.

아이콘을 눌러도 앱이 실행되지 않는다 (Windows)

설치 파일 일부가 손상되었거나 백신이 격리한 경우이다. 실행 파일 안의 프로그램 묶음이 설치 당시와 다르면 PAIO는 **오류 창도 로그도 없이 즉시 종료된다.** 창이 뜨는 아래의 무결성

PAIO를 제거한 뒤 정식 배포본을 다시 설치한다. 사용자 데이터는 그대로 남는다.

무결성 검사에 걸려 기능이 잠긴다 (Windows)

PAIO는 실행 직후 백그라운드에서 `PAIO.exe` 를 자신의 Authenticode 서명과 대조한다. 이 검사는 **Windows 설치본에서만** 수행된다.

검사 결과	동작
서명 유효 + 서명자 일치	정상. 아무것도 잠그지 않는다
서명 없음	아무것도 잠그지 않는다
판정 불가	아무것도 잠그지 않는다. 인증서 만료 · 폐기 서버 접속 불가 등
해시 불일치 (`hash-mismatch`)	잠금
서명자 불일치 (`wrong-signer`)	잠금

잠금이 걸리면 다음 다이얼로그가 한 번 표시된다.

```
PAIO could not verify the integrity of its installation.
Please reinstall PAIO. Editing is disabled for this session.
```

앱은 종료되지 않는다. 프로젝트를 열어 읽을 수 있고 편집만 막힌다. 진단 정보는 `%APPDATA%\PAIO\tamper-report.log` 에 기록된다(`workspace-<해시>` 하위 폴더가 아니라 로 지원팀에 그대로 보내면 된다. 해결책은 정식 배포본을 다시 설치하는 것이다.

로그인이 Can't reach the PAIO server 로 실패한다

카드에 `Can't reach the PAIO server (pars.popcornsar.com). Check your network or proxy.` 가 표시된다. 네트워크를 확인하고, 회사 네트워크라면 IT 부서 용해 달라고 요청한다. 다른 로그인 실패 메시지는 3.2.5에 있다.

Linux: PAIO가 Chromium 샌드박스 없이 실행된다

`.tar.gz` 압축 해제본을 실행하면 터미널에 `[paio] This system refuses unprivileged user namespaces and <folder>/chrome-sandbox is not setuid sandbox. ...` 가 출력된다. 압축을 푼 폴더에서 `sudo ./install.sh` 를 한 번 실행하거나(새로 풀 때마다 다시), `.deb` 패키지로 설치한다. 3.1.4를 본다.

20.2 워크스페이스 · 프로젝트

PAIO 창 하나에는 프로젝트 하나, 즉 창이 연 폴더가 들어간다. 창 제목에 `PAIO [<프로젝트>]` 가 표시된다. 다른 프로젝트는 새 창(**PROJECT ▸ New Window**)에서 연다.

프로젝트를 만들려는데 이름이 거부된다

Welcome 탭의 **New Project** 와 **PROJECT ▸ New Project...** 는 같은 다이얼로그를 연다. **Name** 칸과 **Location** 칸(**Browse...** 포함)을 모두 채우면 `The project will be c` **Create** 를 누를 수 있다.

조건	화면에 보이는 것
비어 있음	**Create** 가 비활성 상태로 남는다
영문자·숫자·하이픈·밑줄 외 문자 포함	`Project name can only contain letters, numbers, hyphens, and underscores`
Windows 예약 디바이스 이름	`"<이름>" is a reserved system name and cannot be used`
위치에 같은 이름의 폴더가 이미 있음	`Project "<이름>" already exists` (**Create** 를 누른 뒤)

예약 이름은 `CON`, `PRN`, `AUX`, `NUL`, `COM0` ~ `COM9`, `LPT0` ~ `LPT9` 이며 대소문자를 구분하지 않는다. 예를 들어 `DoorEcu` 와 `Door_Ecu-01` 은 통과하고, `Door Ecu` 와 `con` 치를 고친 뒤 **Create** 를 다시 누른다.

Rename Project 항목이 없다

프로젝트 이름은 폴더 이름이다. Explorer는 프로젝트 행 없이 폴더 내용을 바로 나열하므로 이름을 바꿀 항목이 없다.

Explorer에 파일이 보이지 않는다

Explorer는 디스크의 모든 파일과 폴더를 나열한다. 기본적으로 숨겨지는 것은 다음뿐이며 트리의 모든 단계에 적용된다.

숨겨지는 항목	정체
`.paio`	PAIO 자체의 작업 데이터
`.devcontainer`	PAIO가 쓰는 Dev Container 설정
`.git`	Git 저장소 데이터
`<프로젝트>.io.prj`	프로젝트 표지 파일

보려면 Explorer 헤더의 **More Actions...** ▸ **Show Hidden Entries** 를 고른다. **EDITOR** 뷰의 **DOC**, **CODE**, **EXPORT** 섹션은 Explorer보다 적은 파일만 보여 준다. 트리를 다시 읽으

`import` 폴더를 찾고 있다

프로젝트에는 디스크에 `code`, `design`, `doc`, `export`, `import`, `workflow` 여섯 폴더가 있다. Explorer는 여섯 개를 모두 이름순으로 보여 준다.

`design` 폴더를 펼치면 에디터가 아니라 모델 파일이 나온다

의도된 동작이다. Explorer에서 `design` 은 모델 파일(`IVC.io.model`, `SYSTEM.io.model`, `VFB.io.model`, `SWC` · `Module` 폴더 등)이 든 일반 폴더이다. `*.io.model` 파다.

에디터는 **EDITOR** 뷰에서 연다. **DESIGN** 아래 **System**, **VFB**, **Interface**, **DataType**, **Diagnostic**, **MODULE** 아래 **Definition**, **Configuration** 이다. 프로젝트 폴더를 OS 파일 탐색 **File Explorer** 를 고른다.

고정 폴더 이름을 바꿀 수 없다

`code`, `design`, `doc`, `export`, `workflow` 에는 이름 변경 · 삭제 항목이 없다. `import` 에는 폴더 메뉴 전체가 있다(4.6.5).

폴더	컨텍스트 메뉴
`design`	**Import**, **Export**, **Code Generation** (**PRTE** / **PARA**), **New File**, **New Folder**
`doc`, `code`	**New File**, **New Folder**
`workflow`, `export`	**New File**, **New Folder**
`import`	**Rename** 과 **Delete** 를 포함한 폴더 메뉴 전체

고정 폴더를 OS 탐색기에서 지웠는데 다시 생긴다

의도된 동작이다. 파일 감시자가 고정 폴더의 삭제(`unlinkDir`)를 감지하면 다시 만든다. 프로젝트 자체를 삭제하는 중일 때는 예외이다.

폴더를 열거나 Explorer에서 작업할 때 Permission denied 가 나타난다

파일 시스템이 PAIO의 요청을 거부했다(예: 사용자가 쓸 수 없는 SSH 폴더, 읽을 수 없는 폴더). PAIO는 `Permission denied` 토스트를 띄운다.

위치	표시
읽을 수 없거나 그 안에 프로젝트 폴더를 만들 수 없는 폴더를 열 때	토스트, 그리고 **EDITOR** 뷰와 Explorer에 폴더 경로와 함께 `Permission denied` 와 **Open Project** 버튼
폴더를 읽을 수 없는 **DOC**, **CODE**, **EXPORT** 섹션	폴더 이름이 든 토스트, 그리고 섹션에 폴더 경로와 함께 `Permission denied`
Explorer의 **New File**, **New Folder**, **Rename**, **Delete**	토스트
저장	저장되지 않은 파일의 이름이 든 토스트

폴더 권한을 고치거나 **Open Project** 로 다른 폴더를 연다. 거부된 섹션 폴더는 **EDITOR** 뷰 헤더의 **Refresh** 로 다시 읽는다.

20.3 편집 · 저장

탭에 점(●)이 계속 남아 있다

저장하지 않은 변경이 있다는 뜻이다. 저장 방법은 세 가지이다.

1. `Ctrl+S` 는 활성 에디터를 저장한다. 포커스가 에디터 밖에 있어도 메뉴 **Save** 와 같은 경로로 저장한다.
2. **PROJECT ▸ Save** (`Ctrl+S`)도 같다.
3. **PROJECT ▸ Save All** 은 프로젝트의 저장하지 않은 탭을 모두 저장한다.

저장에 성공하면 점이 사라진다. 남아 있다면 다음 중 하나이다.

1. 라이선스가 만료되었다. 대신 업그레이드 다이얼로그가 나타난다(20.1 참고).
2. 쓰기가 실패했다. 파일 시스템이 거부하면(쓰기 권한 없음) `Permission denied` 토스트가 저장되지 않은 파일의 이름을 보여 준다(20.2 참고). 그 밖의 쓰기 실패(원격 연결

> **다른 이름으로 저장(Save As)은 없다.** **PROJECT** 메뉴에는 **Save** 와 **Save All** 만 있고, `Ctrl+Shift+S` 는 `Ctrl+S` 와 같다. 모델 파일은 이름이 정해져 있어 다른 이름의 사본은

이름을 입력했는데 반영되지 않고 오류 표시만 나타난다

AUTOSAR SHORT-NAME 규칙 위반이다. 첫 글자는 영문자, 그 뒤는 영문자 · 숫자 · 밑줄만 쓴다. 이름을 거부하는 자리에서는 128자 제한도 걸린다.

조건	표시되는 문구
비어 있거나 공백뿐	`Name is required`
규칙에 맞지 않는 문자	`Must start with a letter, then letters/digits/underscores only`

빈 이름은 붉은색으로 표시되고 모델에 쓰이지 않는다. 다른 파일의 참조까지 빈 문자열로 덮어써 되돌릴 수 없게 되기 때문이다.

규칙에 어긋나는 문자는 주황색 경고만 내며, 값을 막을지는 화면마다 다르다. Adaptive 탭과 Persistency 인터페이스는 이전 이름을 남기고, Service · Sender-Receiver · Client 스포트 전에 경고가 남은 필드가 없는지 확인한다.

예를 들어 Adaptive 탭에서 `1stPort` 로 바꾸면 거부되고, `Port1` 은 즉시 반영된다.

Undo는 이 창의 프로젝트에만 적용된다

Undo · Redo와 **Save All** 은 창의 프로젝트 하나에만 적용된다. 두 번째 창의 프로젝트는 자기 이력과 저장하지 않은 변경을 따로 가진다. 탭의 점(●)은 파일 단위로 표시된다.

외부에서 파일을 고쳤는데 에디터에 반영되지 않는다

그 파일에 저장하지 않은 편집이 있거나 편집이 진행 중이면, 편집을 지키기 위해 리로드를 건너뛰고 제목이 `Changed on disk` 인 경고 토스트를 띄운다. 설계 에디터와 Workflow 코드 에디터에서는 `<파일이름>: unsaved edits kept` 이다.

같은 파일에 대한 경고는 5초 안에는 다시 나오지 않으므로, 파일을 연속으로 쓰는 도구가 있어도 반복되지 않는다. 다음은 모두 편집으로 본다.

- 그 에디터가 다루는 파일 중 하나라도 저장되지 않았을 때
- 입력 중인 값이 아직 반영 전일 때(타자 입력 중, 그리드 셀 편집 중 등)
- 그 에디터의 설정 다이얼로그가 열려 있고 그 안에 변경이 있을 때

저장해서 디스크를 덮어쓰거나, 변경을 버리려면 탭을 닫았다가 다시 연다.

창이 닫히거나 탭이 닫힐 때 저장 여부를 묻는 창

상황	제목	본문
탭 하나 닫기	**Unsaved Changes**	`Do you want to save the changes to <탭 이름>?`
창 닫기	**Unsaved Changes**	`Do you want to save the changes to <탭 이름들>?`

Close Others, **Close Saved**, **Close All** 은 탭을 하나씩 닫으며 저장하지 않은 탭마다 **Close** 와 똑같이 묻는다. 창을 닫을 때 워크플로 실행 같은 작업이 진행 중이면 **Close PAIO** 확

`Ctrl+B` 를 눌렀더니 사이드바 대신 빌드가 실행된다

코드 에디터(Monaco)에서 `Ctrl+B` 는 **Build** 이다. 밖을 클릭하고 누르면 사이드바를 토글한다. 코드 에디터에서는 `Ctrl+F5` 가 **Build & Run**, `Shift+Alt+F` 가 **Format Docu**
축키 총람을 본다.

탭이 더 이상 열리지 않는다

동시에 열 수 있는 탭은 **20개**이다. 넘으면 새 탭이 조용히 열리지 않는다.

> **표시와 동작이 다른 지점**: 아무 메시지도 나오지 않는다. 개발자 콘솔에만 `Maximum number of tabs (20) reached` 가 남는다. 파일을 눌러도 아무 일이 없으면 이 한도를

정리는 위의 탭 컨텍스트 메뉴 항목을 참고한다.

20.4 임포트

임포트는 **EDITOR** 뷰의 **DESIGN** 헤더 오른쪽 첫 번째 아이콘 **Import** 로 시작하거나, Explorer에서 `design` 우클릭 ▸ **Import** 로 시작한다. 파일 선택 다이얼로그(**Select files to i**
`.dbc`) / `ARXML Files` / `DBC Files` 이고, 여러 파일을 한 번에 고를 수 있다. 13. 임포트 · 익스포트를 본다.

임포트 전에 하던 편집이 사라질까 걱정된다

사라지지 않는다. 임포트는 작업을 시작하기 전에 **열린 모든 에디터의 저장하지 않은 편집을 저장한다.** 임포트가 모델 파일을 다시 쓰고 에디터가 디스크에서 다시 읽기 때문이다.

임포트 전체가 실행 취소 한 단계이므로 `Ctrl+Z` 한 번으로 되돌린다. **EDIT** ▸ **Undo** 에는 **첫 번째로** 고른 파일 이름으로 `Undo Import <파일이름>` 이 표시되지만, 그 임포트가 쓴

임포트를 다시 실행했더니 거부되었다

앞의 임포트가 진행 중이거나 충돌 다이얼로그에서 기다리는 동안에는 두 번째 임포트가 시작되지 않는다. 먼저 끝내거나 취소한다. 진행 다이얼로그에는 다음 중 하나가 표시된다.

```
Import rejected: an import is already running. Finish or cancel it first.
Import rejected: an import is waiting on the source-conflict dialog. Finish or cancel it first.
Import rejected: an import is waiting on conflict resolution. Finish or cancel it first.
```

두 번째 임포트가 첫 번째가 쓴 파일과 되돌리기 정보를 지우지 않게 하려는 동작이다. Machine Configuration 다이얼로그의 **PARA Platform Import** 도 같은 경로라 같은 조건에
다. PRTE 베이스 세트 추가는 `PRTE Base Set rejected: ...` 로 거부된다.

SSH 창의 임포트가 연결이 닫혀 실패했다

원격 창에서 ARXML 임포트가 진행되는 동안 연결을 닫으려 하면 먼저 묻는다 (**Close Connection**: `An ARXML import is running. Closing the connection cancel`
Connection 은 임포트를 취소한 뒤 임포트가 멈추면 연결을 닫는다. 임포트 내용은 PC에 쓰이지 않는다.

임포트 도중 연결이 끊기면 임포트가 `The remote connection closed during the import. Part of it may be on the server; reconnect and import a`
다.

임포트가 멈추고 "Import Source Conflicts" 창이 나타났다

고른 **여러 파일이 같은 요소를 서로 다르게 정의**하고 있다. 프로젝트가 아니라 입력 파일 사이의 충돌이다. 제목 옆에 `(N elements diverge across imported files)` 가 붙

1. 왼쪽 트리에서 충돌 요소를 고른다. 오른쪽에 파일별 값이 나타난다.
2. 요소마다 기준 파일을 고른다. 아직 정하지 않은 항목 전부에 한 파일을 적용하려면 **Prefer source:** 에서 그 파일을 고르고 **Apply to remaining** 을 누른다.
3. 오른쪽 아래 **Resolve 3/7**(결정한 수 / 전체 수)을 누르면 임포트가 이어진다. 이 버튼은 항상 누를 수 있다. 정하지 않은 항목은 마지막 파일의 값을 쓰고 `DUPLICATE_ELEMENT`

멈추려면 **Cancel Import** 를 누른다. 확인 창(**Cancel Import?**)의 본문은 `Nothing has been written yet; your project files stay untouched.` 이며, 되돌릴 것

임포트 후 충돌 창이 나타났다

프로젝트의 요소와 들어오는 요소가 충돌한다. 창 제목은 `Conflict Resolution: N conflicts` 이고, 요소마다 다음 중 하나를 고른다.

버튼	동작
Keep Project Version	기존 프로젝트 값을 유지한다
Use Imported Version	임포트한 값으로 덮어쓴다
Merge Fields	필드별로 각각 어느 쪽을 쓸지 선택한다(양쪽에 차이가 있을 때만 표시된다)
Keep Both	이름을 바꿔 둘 다 남긴다(해당 요소가 허용할 때만 표시된다)

Merge Fields 에서 **Keep Project Version** / **Use Imported Version** 으로 바꾸면 `Discard Field Edits?` 를 묻고, 확인하면 필드별 선택이 버려진다. 다 정했으면 오른쪽 아

> **결정하지 않고 Resolve를 누르면 임포트한 값으로 덮어쓴다.** 앞의 소스 충돌 창과 다르다. 남은 항목이 있으면 **Unresolved Conflicts** 가 먼저 `N undecided conflicts wi values.` 를 묻는다. **Overwrite & Continue** 는 남은 항목을 모두 임포트 값으로 바꾸고, **Cancel** 은 충돌 창으로 돌아간다.

취소하려면 **Resolve** 왼쪽의 **Cancel** 을 누른다(이 창에는 `Cancel Import` 버튼이 없다). `Cancel Import?` 확인 창의 본문은 `Discards this import and restores y` **Cancel Import** 는 임포트 전 상태로 되돌리고, **Keep Resolving** 은 창으로 돌아간다. 성공하면 `Project restored` 줄이 든 `Import cancelled` 토스트가, 되돌리기에 실패하

Import ECUC Configuration 에서 열린 충돌 창은 취소가 다르다. 확인 창 본문은 `Conflicting files keep your current values; already-imported files st` 는다.

결정을 마치면 임포트가 이어지고 `3 replaced, 1 kept` 같은 줄이 든 `Import complete` 토스트가 나타난다. 반영에 실패하면 프로젝트 전체가 되돌아가고 `Import failed`

임포트는 끝났는데 일부 데이터가 보이지 않는다

> **표시와 동작이 다른 지점**: 일반 **Import** 는 개수만 보고한다 (`Import complete: N elements imported from M file(s)`). **경고는 전혀 표시되지 않는다**: 해소되지 않은 모듈 직속 파라미터.

- ECUC 설정만 다시 가져오려면 **EDITOR** 뷰의 **MODULE** 아래 **Configuration** 을 우클릭한 뒤 **Import ECUC Configuration** 을 고른다. 이 경로는 건너뛴 모듈과 거부된 모듈을

ECUC 설정이 임포트되지 않았다

모듈 정의가 프로젝트에 없으면 그 설정은 건너뛴다. 같은 ARXML에 `ECUC-MODULE-DEF` 도 있으면 정의를 **먼저** 임포트하므로, 그런 파일은 따로 준비할 필요가 없다.

Import ECUC Configuration 은 결과를 **ECUC Configuration Import (Partial)** 다이얼로그로 보고한다.

```
Imported 3 module configuration(s).

Skipped, Module Definition not found:
  • Dcm
  • Dem

Rejected, definition mismatch:
  • Com: Container definition not found in "Com": ComSignalX (instance "ComSignal_0")
```

분류	의미	대응
Skipped	그 모듈의 정의 파일이 프로젝트에 없다	정의 ARXML을 먼저 임포트한다
Rejected	정의는 있는데 컨테이너 · 파라미터 · 레퍼런스가 정의 트리와 구조적으로 맞지 않는다	정의 버전을 맞추거나 설정 파일을 확인한다

일반 **Import** 로 넣으면 이 두 분류는 화면에 나오지 않고 경고로만 남는다.

ECUC 모듈 정의를 임포트했더니 일부 항목이 빠졌다

`ECUC-MODULE-DEF` 바로 아래의 파라미터 · 레퍼런스는 **의도적으로 버린다.** R25-11 XSD에는 그 자리에 슬롯이 없어 익스포트가 스키마를 위반하게 되기 때문이다. 지원하지 않는

모듈 정의는 일반 **Import** 로 함께 들어온다. ARXML 안의 `ECUC-MODULE-DEF` 를 찾아 모듈 정의로 쓴다. 따로 실행할 메뉴는 없다.

> **표시와 동작이 다른 지점**: 정의만 임포트하는 메뉴는 없고, "버림" 경고는 어디에도 표시되지 않는다. 항목이 빠졌다면 ARXML에서 `ECUC-MODULE-DEF` 직속 파라미터 · 레퍼런스

DBC의 한글·특수문자가 깨진다

자동 감지는 **UTF-8 BOM만** 인식하고, BOM이 없으면 ISO-8859-1(Latin-1)로 읽는다. 파일을 UTF-8 with BOM으로 다시 저장한다.

20.5 익스포트

익스포트는 **EDITOR** 뷰의 **DESIGN** 헤더 오른쪽 두 번째 아이콘 **Export** 로 실행하거나, Explorer에서 `design` 우클릭 ▸ **Export** 로 실행한다. 13. 임포트 · 익스포트를 본다.

Export를 눌렀는데 실패한다

익스포트에는 **활성 라이선스와 온라인 검증**이 필요하다. 실패하면 **Export** 창의 해당 형식 줄에 다음 중 하나가 나타난다.

표시되는 원인	의미
`OFFLINE_EXPORT_DENIED: A server connection is required`	서버에 연결하지 못했다
`PLAN_EXPIRED: Export is not permitted on this plan`	라이선스가 만료 상태이다
`INTEGRITY_EXPORT_DENIED: Installation integrity check failed`	무결성 검사가 실패한 설치본이다

라이선스가 만료되면 보통 업그레이드 다이얼로그가 메뉴를 먼저 막으므로, `PLAN_EXPIRED` 는 그 뒤 서버가 거부한 경우에만 나온다. 같은 검사가 **Extract ECU**, **Extract Machin**

결과 파일이 어디에 생기는지 모르겠다

Export 창의 **FOLDER** 에 적힌 폴더이며, 기본값은 `<프로젝트>/export/` 이다. ARXML은 프로젝트에 있는 내용 종류마다 `<프로젝트이름>_*.arxml` 파일을, DBC는 CAN 클러스
은 `R25-11, 298 elements` 나 `3 files` 처럼 나온다.

System 에디터의 **Extract ECU**(ECU 노드) 또는 **Extract Machine**(HPC 노드)은 `<프로젝트>/export/<노드이름>/` 에 쓰고, 완료 메시지는 `<노드이름> → export/<노드이`

`export` 폴더의 ARXML을 고쳤는데 설계에 반영되지 않는다

의도된 동작이다. `export` 는 **단방향 결과 폴더**이다. PAIO는 이 폴더를 읽지 않고, 다음 익스포트가 같은 이름의 파일을 덮어쓴다. 해당 에디터에서 모델을 고치고 저장한 뒤 **Export**

> 외부 ARXML은 `export` 에 두지 말고 **DESIGN** 헤더의 **Import** 로 가져온다 (13. 임포트 · 익스포트).

익스포트 결과에 최근 편집이 빠졌다

> **표시와 동작이 다른 지점**: ARXML 익스포트는 **Extract ECU**, **Extract Machine** 을 포함해 **저장된** `design/` **파일을 읽는다.** 저장하지 않은 내용은 빠지므로 익스포트 전에 저장

내보낸 ARXML을 다시 임포트했더니 다이어그램 배치가 달라졌다

의도된 동작이다. 노드의 좌표와 크기는 **ARXML로 나가지 않는다.** System 에디터 좌표는 `design/IVC.io.model` 에, VFB 배치는 `design/SWC/Layout/<컴포지션이름>.vf`
격자 위에 새로 생성된다. 배치를 유지하려면 프로젝트 폴더 자체를 옮기거나 복사한다.

DBC로 내보내고 싶다

Export 창에서 **DBC** 를 고르거나, System 에디터에서 CAN 클러스터 하나를 우클릭하고 **Export to DBC** 를 고른다(13. 임포트 · 익스포트 13.6 참조).

20.6 ECUC 생성

ECUC GENERATION 이나 Generate ECUC 를 찾을 수 없다

System 에디터의 ECU 속성 패널에는 **ECUC GENERATION** 섹션도 **Generate ECUC** 버튼도 없다. Classic ECU의 BSW 설정은 PRTE Auto Configuration이 설계로부터 생성하
같다.

- VFB 에디터 ECU Configuration 다이얼로그 머리글의 **PRTE AUTO CONFIGURATION** (08. VFB 에디터 8.16).
- Module Configuration 에디터 **ECU** 목록의 ECU 행에 있는 **PRTE Auto Configuration** 아이콘 (12. ECUC 12.2.6).

설정은 임포트로도 들어온다.

- 모듈 설정이 들어 있는 ARXML로 **DESIGN** 헤더의 **Import** 를 실행한다.
- **EDITOR** 뷰의 **MODULE** 아래 **Configuration** 을 우클릭한 뒤 **Import ECUC Configuration** 을 선택한다(20.4 참고).

MODULE ▸ **Configuration** 이 여는 에디터에서 확인하고 편집한다. 12. ECUC를 본다.

20.7 검증

실행 버튼은 없다. 창이 열리면 바로 그 창의 프로젝트를 검증하고, 다음 시점에 다시 실행한다.

- 모델을 편집했을 때(입력 중에도 갱신된다)
- 활성 에디터 탭이 바뀌었을 때
- ARXML 임포트가 끝났을 때(이때는 디스크 내용을 읽는다)
- 창에서 다른 폴더를 열었을 때

14. Validation을 본다.

결과가 비어 있을 때 표시되는 세 가지 문구

다음 셋을 구별한다.

표시되는 문구	의미
`No validation issues.`	검증이 정상적으로 실행되었고 위반이 없다
`Select a project in the Explorer to run validation.`	대상 프로젝트가 정해지지 않았다
`Validation could not run. See the developer console for details.`	검증이 실행되지 못하고 실패했다

세 번째는 "문제 없음"이 아니다. 이때 상태바 Validation(`V`) 배지는 `!` 를 표시하고, 툴팁은 `Validation could not run. Open the Validation panel` 로 바뀐다.

두 번째라면 프로젝트가 열려 있는지(타이틀바 `PAIO [<프로젝트>]`) 확인한 뒤 그 에디터를 하나 연다.

상태바 배지가 무엇을 뜻하는지 모르겠다

배지	의미
없음	위반 0건
숫자	전체 위반 수이다(오류 > 경고 > 정보 순으로 색이 정해진다). 99를 넘으면 `99+` 로 표시된다
`!`	검증 실행 실패

툴팁은 `Validation: N error(s), M warning(s), K info(s)` 형식이다. 패널 제목 옆 빨간 배지는 **오류 수만** 표시하고, 오류가 없으면 숨겨진다.

위반 행을 눌렀는데 아무 데도 이동하지 않는다

원인은 두 가지이다.

1. 그 규칙에 이동 대상이 없다. 클릭해도 아무 일 없이 끝난다.
2. 요소가 대상 에디터가 없는 영역에 있다. 이동이 되는 영역은 다음과 같다.

위반 영역	열리는 에디터
`datatype.*`	DataType 에디터
`interface.*`	Interface 에디터
`vfb.*`	VFB 에디터
`ivc.*`, `system.*`	System 에디터
`signal.*`	Signal 에디터
`diagnostic.*`	Diagnostic 에디터

ECUC 설정 · 모듈 정의 위반은 클릭 대상이 없으므로 **MODULE** 아래 에디터를 직접 연다.

이동하면 요소가 선택되고 위반 필드가 심각도 색으로 표시된다. 다른 행을 누르면 표시가 지워진다.

20.8 AI

AI 채팅 버튼이 비활성 상태이다

채팅 토글은 헤더 오른쪽에 있다(툴팁 `AI Chat Panel`). 툴팁이 `Plan expired` 이면 라이선스가 만료 또는 읽기 전용 상태이고 버튼이 눌리지 않는다. 이때 입력창 안내 문구는 `

프로바이더 목록에 아무것도 활성이 아니다

목록 표시 이름	종류	사용 조건
Claude [OPUS]	CLI	`claude` CLI 설치 + 로그인
Claude [GLM]	API	API 키 등록
Gemini	CLI	`gemini` CLI 설치 + 로그인
Codex	CLI	`codex` CLI 설치 + 로그인
Ollama	로컬 서버	Ollama 로컬 서버 실행

활성 프로바이더가 없으면 전송이 `LLM provider is not available` 로 실패한다.

프로바이더 다이얼로그는 채팅 패널 헤더의 톱니바퀴(툴팁 **Manage AI accounts**)나 명령 팔레트(`Ctrl+Shift+P`)의 **Manage AI Accounts...** 로 연다. 헤더 메뉴 막대(**PROJEC

가용성은 30초마다 다시 확인한다. 다이얼로그에서 로그인을 시작하면 **5초 간격으로 최대 2분** 동안 확인하고, 상태가 바뀌면 멈춘다. 2분 뒤에도 그대로면 다이얼로그를 닫았다가 다

AI가 만든 모델이 마음에 들지 않는다

`Ctrl+Z` 를 누른다. AI 채팅의 쓰기는 파일이 여럿이어도 실행 취소 한 단계이다. **EDIT ▸ Undo** 에는 `Undo AI: 3 file(s)` 처럼 파일 수만 표시되고 요소 이름은 나오지 않는다.

Quick Start 를 찾을 수 없다

이 버전의 메뉴에는 **Quick Start** 가 없다. **PROJECT ▸ New Project...** 로 프로젝트를 만들고(20.2 참고), 모델은 에디터나 AI 채팅으로 만든다. 15. AI 어시스턴트를 본다.

20.9 코드 · 빌드

Build를 눌렀더니 컴파일러를 못 찾는다는 메시지가 나타난다

로컬 빌드는 번들 `gcc` / `g++` 를 먼저, 그다음 `PATH` 의 컴파일러를 쓴다. 둘 다 없으면 OS 오류가 출력 패널에 그대로 나온다. 컴파일러를 설치하거나 `PATH` 를 확인한다.

원격(WSL · 컨테이너 · SSH) 빌드에서 종료 코드 127이나 `command not found` 가 나오면 다음 안내가 추가된다.

```
Hint: 'gcc' was not found in this environment. Install a C/C++ toolchain
(e.g. 'sudo apt install build-essential' on Debian/Ubuntu).
```

기본값: 옵션 `-Wall -g -std=c++17`, `.cpp` `.hpp` `.cc` `.cxx` 는 `g++`, 그 외는 `gcc`, 제한 시간은 로컬 60초 · 원격 120초이다.

`.rs` 파일 하나만 빌드하려는데 되지 않는다

Rust는 `Cargo.toml` 프로젝트가 필요하다. 단일 `.rs` 파일에서 **Build & Run** 을 실행하면 다음이 출력된다.

```
Rust files require a Cargo.toml project for Build & Run.
Create a Cargo project (cargo new <name>) or use the terminal:
  rustc "<파일경로>" && ./<output>
```

Build & Run 은 파일 위치에서 최대 12단계까지 올라가며 `CMakeLists.txt`, 그다음 `Cargo.toml` 을 찾는다. 둘 다 없을 때만 위 안내가 나온다.

`code/Cargo.toml` 을 직접 고쳤는데 되돌아간다

의도된 동작이다. `code`(하위 폴더 포함, `target` 과 `.git` 제외)에 `.rs` 파일이 하나라도 있으면 PAIO가 **`code/Cargo.toml` 을 다시 만들어 덮어쓴다.** rust-analyzer가 진단어

실행 시점	생성되는 `version`
앱 시작 직후, 그리고 창의 프로젝트가 바뀔 때	`0.1.0`
코드 에디터에서 `.rs` 파일을 열어 rust-analyzer가 시작될 때	`0.1.2`

둘 다 패키지 이름 `paio-code`, 에디션 `2021`, `.rs` 파일마다 `[[bin]]` 하나를 쓴다. 버전만 다르므로 그 줄이 왔다 갔다 할 수 있지만 빌드에는 영향이 없다. 직접 관리하는 Cargo 다.

코드 완성이 동작하지 않는다

언어 서버가 아직 시작 중이거나 실패했을 수 있다. 코드 에디터 아래 **PROBLEMS** 패널에 진단이 올라오는지 확인한다. Rust라면 위의 `Cargo.toml` 항목을, 원격(SSH · 컨테이너 · \

C/C++ 라면 상태바 왼쪽 **IntelliSense** 를 확인한다. `IntelliSense: Failed` 나 `IntelliSense: Stopped` 이면 마우스를 올려 이유를 읽고, 클릭해서 **Restart IntelliSens**

20.10 원격

원격 언어 서버가 배포되지 않는다

번들 바이너리는 **glibc 기반 linux-x64** 전용이다. 다음 중 하나라도 해당하면 거부된다.

조건	판정
`uname -s -m` 이 `Linux` + `x86_64` 가 아님	거부 (`not-linux-x64`)
`/lib/ld-musl-*` 존재 (Alpine 등 musl 계열)	거부 (`not-linux-x64`)
번들 바이너리가 패키지에 없음	거부 (`bundle-missing`)
원격 `$HOME` 을 확인하지 못함	거부 (`home-unresolved`)

SSH에서는 `Remote is not glibc Linux x64` 줄이 든 `No code intelligence` 토스트가 나온다(워크스페이스당 한 번).

WSL에서는 배포 배너에 오류가 표시된다.

조건	오류 문자열
배포판이 x86_64가 아님	`WSL distro is not x86_64 — only Linux x64 is supported`
musl(Alpine) 계열 배포판	`WSL distro uses musl (Alpine) — only glibc is supported`

연결·해제 후 화면이 새로 고쳐진다

의도된 동작이다. 원격 연결, 해제, 로컬 폴더로 복귀는 모두 창을 다시 로드해 상태를 초기화한다.

> **주의**: 이 재로드는 저장 여부를 묻지 않는다. 연결이나 해제 전에 저장한다.

컨테이너 메뉴에 Rebuild나 Reopen Locally가 없다

vECU 컨테이너 창의 원격 표시기 메뉴에는 **Close Connection** 만 있다 (17.1.3). 이것은 컨테이너의 작업을 호스트 프로젝트 폴더로 복사하고 호스트 프로젝트의 창으로 돌아가며, 컨 로 (17.3.7, 17.10) **Reopen Folder Locally** 는 없다. 실행 중인 임의의 컨테이너에 붙는 기능은 제공하지 않는다.

새 컨테이너로 시작하려면 **vECU** 뷰의 카드에서 세 점 버튼(툴팁 `More`) ▸ **REMOVE CONTAINER** 로 제거한다(17.3.6). 워크스페이스 볼륨과 호스트 프로젝트는 남고, 다음 **OPEN** 이

Build ECU 가 "A Windows host is not an execution target" 로 거부된다

ECU는 프로젝트의 vECU에서 빌드하고 실행한다. 로컬 Windows 창에서는 **Build ECU** 와 **Run ECU** 가 `Not available here` 토스트와 줄 `A Windows host is not an e` `Build and run the ECU in its vECU: open it from the vECU view.` 로 거부되고, SSH 창에서는 그 줄이 `ECU build/run over SSH is not supported` 다. **vECU** 뷰에서 vECU를 연 뒤(17.3) 그 창에서 **Build ECU** 를 쓴다. 17.9.2와 17. 원격 개발 (SSH · vECU 컨테이너)를 본다.

20.11 성능

노드가 많은 다이어그램이 느리다

System 에디터와 VFB 에디터의 캔버스 왼쪽 아래 툴바에 아이콘 토글이 두 개 있다(이름은 툴팁에 있다).

툴팁	끄면 사라지는 것
Toggle Grid	배경 격자
Toggle Connector Animation	커넥터를 따라 흐르는 애니메이션

두 토글은 표시만 바꾸고 모델은 건드리지 않는다. 같은 툴바의 **Fit to View** 는 전체를 한 번에 그리므로, 작업 영역만 확대하면 그리는 양이 줄어든다.

탭이 많아 느리다

탭은 최대 20개까지 열린다(20.3 참고). 탭을 우클릭해 **Close Saved** 로 저장된 탭만 닫거나, 저장하지 않은 탭마다 묻는 **Close Others** / **Close All** 을 쓴다.

20.12 그 밖의 확인 사항

확인	방법
앱 버전	상태바 오른쪽의 `v1.0.1` 표시, 또는 **HELP** ▸ **About PAIO**
제품 정보 · 연락처	**HELP** ▸ **About PAIO** 에 버전과 `contact@popcornsar.com` 이 표시된다
업데이트 확인	**HELP** ▸ **Check for Updates**
문의	**HELP** ▸ **Report Issue** 를 누르면 기본 브라우저에서 `https://autosar.io/support/qna` 가 열린다
Welcome 탭으로 돌아가기	**HELP** ▸ **Welcome**
명령 이름으로 기능 찾기	명령 팔레트(`Ctrl+Shift+P`)

macOS에서는 OS 메뉴 막대에도 Help 메뉴가 있고, 그쪽 **Report Issue** 는 `https://github.com/popcornsar/PAIO/issues` 를 연다. Windows와 Linux는 테두리 없는 커

업데이트 확인 결과가 무엇을 뜻하는지 모르겠다

Windows와 `.deb` 패키지로 설치한 Linux에서는 PAIO가 새 버전을 스스로 내려받아 설치할 수 있다. 알림 문구를 다음 표에서 찾는다.

결과	표시
새 버전 있음	`PAIO <버전> is available.`
확인 중	`Checking for updates…`
내려받는 중	`Downloading PAIO <버전>… <n>%`
이미 내려받음	알림 아래의 `PAIO <버전> has already been downloaded.`
준비 중	`Updating PAIO <버전>.`
설치 중	`Updating PAIO <버전>. PAIO restarts when the update finishes.`
설치 중 (Linux `.deb`)	`Enter an administrator password to install PAIO <버전>.`
창이 닫히고 설치하는 중(시스템 알림)	`Installing PAIO <버전>: This takes about 5 minutes with nothing on screen. PAIO opens again when it finishes.` (Windows) `opens again when the install finishes.` (Linux `.deb`)
암호 창을 닫았거나 인증 실패 (Linux `.deb`)	`The install was cancelled. Press Update to try again, or install it from the download page.`
다른 프로그램이 2분 넘게 설치 중 (Linux `.deb`)	`Another program is installing software. Press Update again when it finishes.`
다른 PAIO 창이 열려 있음	`Close the other PAIO windows to update (<n> open).`
실행 중인 작업이 있음	`PAIO <버전> installs when these tasks finish: <작업>.`
다른 창과 작업을 확인할 수 없음	`PAIO cannot tell whether other PAIO windows or tasks are running, so it cannot update yet.`
설치 실패	`PAIO <버전> could not be installed automatically. Install it from the download page.`
설치 파일 서명 확인 실패	`The downloaded PAIO <버전> installer could not be verified and was not installed. Install it from the download page.`
앱에서 설치할 수 없음 (Linux `.deb`)	`PAIO <버전> cannot be installed from the app on this system. Install it from the download page.`
패키지 확인 실패 (Linux `.deb`)	`The downloaded PAIO <버전> package could not be verified and was not installed. Install it from the download page.`
설치 실패 (Linux `.deb`)	`PAIO <버전> could not be installed. Install it from the download page.`
업데이트가 적용되지 않음	`PAIO <버전> was not installed. You are still on PAIO <현재 버전>. Install it from the download page.`
업데이트 설치됨	`Updated to PAIO <버전> (from <이전 버전>).`
업데이트로 프로젝트가 닫힘	`Projects closed for the update: <프로젝트>.`
다시 열기 실패	`Could not reopen: <프로젝트>.`
최신	`PAIO is up to date.`
로그인하지 않음	`Sign in to check for updates.`
로그인 정보를 저장할 수 없음 (Linux)	`Cannot check for updates: your sign-in could not be stored securely on this computer. Unlock or set up your keyring, t` `restart PAIO and sign in.`
이 플랫폼의 릴리스 없음	`No update is available for this platform yet.`
확인 실패	`The update check did not complete.`
개발 모드	`Auto-update is disabled in development mode.`

Update 가 거부되면 알림 아래에 다음 안내가 붙는다.

안내	할 일
`Some changes could not be saved. PAIO was not updated.`	열린 파일을 저장한 뒤 **Update** 를 다시 누른다
`Close the open dialog first, then press Update again.`	열린 다이얼로그를 먼저 닫고 **Update** 를 다시 누른다
`Another PAIO window is updating.`	그 창의 업데이트를 기다린다
`The update could not start.`	**Update** 를 다시 누르거나 다운로드 페이지를 쓴다

Download page 를 눌렀는데 브라우저가 열리지 않으면 `Could not open the browser. Download PAIO <버전> from <주소>.` 안내에 주소가 나온다.

Download page 는 기본 브라우저에서 다운로드 페이지를 열어 직접 설치하게 한다. 백그라운드 확인 시점은 3.5 에 있다.

20.13 자주 보는 메시지 모음

화면에서 본 문구로 찾을 때 쓴다.

라이선스 · 무결성

문구	항목
`Plan expired`	20.8
`Plan expired, upgrade to continue`	20.8
`License Expired`	20.1
`Upgrade Required` / `Trial Ended`	20.1
`Subscription Required` / `Subscription Expired`	20.1
`Signed in on Another Device`	20.1
`PAIO could not verify the integrity of its installation.`	20.1
`Can't reach the PAIO server (pars.popcornsar.com). Check your network or proxy.`	20.1
`Sign-in wasn't completed. Try again.`	3.2.5
`Too many attempts. Try again in a few minutes.`	3.2.5
`Couldn't sign in. Try again, or contact PopcornSAR support.`	3.2.5

임포트 · 익스포트

문구	항목
`Import rejected: an import is already running. Finish or cancel it first.`	20.4
`Import Source Conflicts`	20.4
`Nothing has been written yet; your project files stay untouched.`	20.4
`Conflict Resolution: N conflicts`	20.4
`Unresolved Conflicts`	20.4
`N undecided conflicts will be overwritten with the imported values.`	20.4
`Discard Field Edits?`	20.4
`Discards this import and restores your project to its pre-import state.`	20.4
`Conflicting files keep your current values; already-imported files stay applied.`	20.4
`Rollback failed`	20.4
`Import complete` / `Import failed` (토스트)	20.4
`An ARXML import is running. Closing the connection cancels it.`	20.4
`The remote connection closed during the import. ...`	20.4
`ECUC Configuration Import (Partial)`	20.4
`OFFLINE_EXPORT_DENIED: A server connection is required`	20.5
`PLAN_EXPIRED: Export is not permitted on this plan`	20.5
`INTEGRITY_EXPORT_DENIED: Installation integrity check failed`	20.5

편집 · 검증 · 프로젝트

문구	항목
`Changed on disk` / `Unsaved edits kept`	20.3
`Permission denied`	20.2
`Do you want to save the changes to ...?`	20.3
`Name is required`	20.3
`Must start with a letter, then letters/digits/underscores only`	20.3
`Project name can only contain letters, numbers, hyphens, and underscores`	20.2
`Project "..." already exists`	20.2
`"..." is a reserved system name and cannot be used`	20.2
`Validation could not run. See the developer console for details.`	20.7
`Select a project in the Explorer to run validation.`	20.7
`No validation issues.`	20.7

코드 · 원격

문구	항목
`Rust files require a Cargo.toml project for Build & Run.`	20.9
`Hint: 'gcc' was not found in this environment.`	20.9
`No code intelligence` / `Remote is not glibc Linux x64`	20.10
`WSL distro uses musl (Alpine) — only glibc is supported`	20.10
`WSL distro is not x86_64 — only Linux x64 is supported`	20.10
`Not available here` / `A Windows host is not an execution target ...`	20.10
`ECU build/run over SSH is not supported yet ...`	20.10
`PAIO is up to date.` / `The update check did not complete.`	20.12

AI 워크플로우

문구	항목
`Already running`	20.14
`Nothing to run` / `No nodes`	20.14
`Node '<name>' input 'in' is required but not connected`	20.14
`The workflow graph contains a cycle — execution order cannot be determined`	20.14
`No cached outputs for this upstream node ...`	20.14
`Node type '<type>' is not implemented yet`	20.14
`stopped by the provider's usage limit ...`	20.14
`stopped because the AI provider account is not logged in ...`	20.14
`No PRTE vECU image on this machine ...`	20.14
`PAIO is installing an update; nothing new can start.`	20.14

검색

문구	항목
`Search cancelled: there are unsaved changes. Save them, or press Refresh to choose again.`	20.15
`The result set only contains a subset of all matches; refine the search.`	20.15
`<n> files were not searched (too large, binary or unreadable).`	20.15

20.14 AI 워크플로우

Workflow 아이콘을 눌러도 아무 일도 일어나지 않는다

레일 아이콘은 창의 프로젝트 에디터를 연다. 프로젝트가 없으면 열 것이 없으므로 먼저 프로젝트를 연다(21.3).

Run Workflow 가 시작되지 않는다

토스트를 확인한다.

- `Already running`: 프로젝트마다 실행은 하나뿐이므로 `Cancel Run` 으로 멈춘다.
- `No nodes` 줄이 든 `Nothing to run`: 노드를 추가한다.
- 플랜이 만료되면 실행이 거부된다(`PLAN_EXPIRED`).
- PAIO가 업데이트를 설치하는 동안에는 새 작업이 시작되지 않는다: `PAIO is installing an update; nothing new can start.`

입력이 연결되지 않은 노드는 `Node '<name>' input 'in' is required but not connected` 로, 피드백 연결이 아닌 루프는 `The workflow graph contains a c determined` 로 실패한다(21.5.1).

실행이 프로바이더 사용 한도에서 멈췄다

실행 로그의 `Finished:` 줄이 `stopped by the provider's usage limit — <이유>` 로 끝난다. 계정의 AI 프로바이더 사용량이 소진되었다. 실행이 쓴 파일은 그대로 남는 **Run from Here** 를 쓰면 위쪽 값은 기록된 실행에서 가져온다(21.8.1).

Run from Here 가 No cached outputs 로 실패한다

위쪽 노드가 기록된 출력을 낸 적이 없다. 워크플로(또는 그 노드)를 한 번 실행한 뒤 다시 **Run from Here** 를 쓴다(21.8.1).

노드가 Node type ... is not implemented yet 로 실패한다

AA Design 과 **Condition Branch** 는 이 버전에서 실행되지 않는다. 끄거나 뺀다(21.12).

AI 노드가 로그인하지 못한다

실행 로그의 `Finished:` 줄이 `stopped because the AI provider account is not logged in — <이유>` 로 끝난다. 아바타 › **AI Account** 에서 상태 줄을 확인한다 을 쓴다(21.11). 그다음 다시 실행한다.

빌드·실행 노드가 vECU 이미지를 찾지 못한다

인스펙터의 PRTE vECU 이미지 필드에 `No PRTE vECU image on this machine. ...` 가 보인다. Docker를 실행하고 PRTE vECU 이미지를 받는다. PAIO는 직접 받지 않는

실행을 취소한 뒤 파일이 바뀌었다

툴바의 `Cancel Run` 은 실행이 쓴 파일을 모두 되돌린다(`Workflow cancelled` 토스트에 `Files rolled back` 줄이 나온다). 탭을 닫으며 **Cancel Run** 을 고르거나 창을 닫

에디터를 다시 열었더니 Last Run 이 비어 있다

LAST RUN 은 현재 세션만 보여 준다. 이전 실행은 `doc/report/` 와 `.paio/workflow-runs/<runId>/` 에서 확인한다(21.9).

20.15 검색

방금 입력한 값이 검색되지 않는다

검색은 디스크에 저장된 파일만 읽는다(5.8). 탭에 저장하지 않은 변경이 있으면 `Search` 다이얼로그가 먼저 묻는다. **Save All & Search** 는 저장한 뒤 검색하고, **Search Saved Fil** 메시지 줄에 `Search cancelled: there are unsaved changes. Save them, or press Refresh to choose again.` 가 표시된다. 뷰 머리글의 **Refresh** 버튼을

`export`, `import`, `workflow` 에서 아무것도 찾지 못한다

의도된 동작이다. 검색 대상은 `design`, `doc`, `code` 폴더뿐이다.

The result set only contains a subset of all matches 가 표시된다

검색은 일치 20,000개에서 멈추고, 메시지 줄에 `The result set only contains a subset of all matches; refine the search.` 가 표시된다. 검색어를 더 구체 를 켠다. 필터 칩은 이미 찾은 결과를 좁힐 뿐이며 한도를 넘은 일치를 되살리지 않는다.

검색되지 않은 파일이 있다

`<n> files were not searched (too large, binary or unreadable).` 는 PAIO가 5 MB보다 큰 파일, 바이너리 파일, 읽을 수 없는 파일을 건너뛰었다는 뜻이다.

검색 결과를 클릭했는데 에디터 대신 읽기 전용 `*.io.model` 이 열렸다

의도된 동작이다. PAIO는 어떤 에디터 필드에도 보이지 않는 값도 저장하고 내보낸다(예: SW-C 구현의 리소스 소비 이름). 이런 결과는 `*.io.model` 파일만 읽기 전용으로 열리고, '

튜토리얼

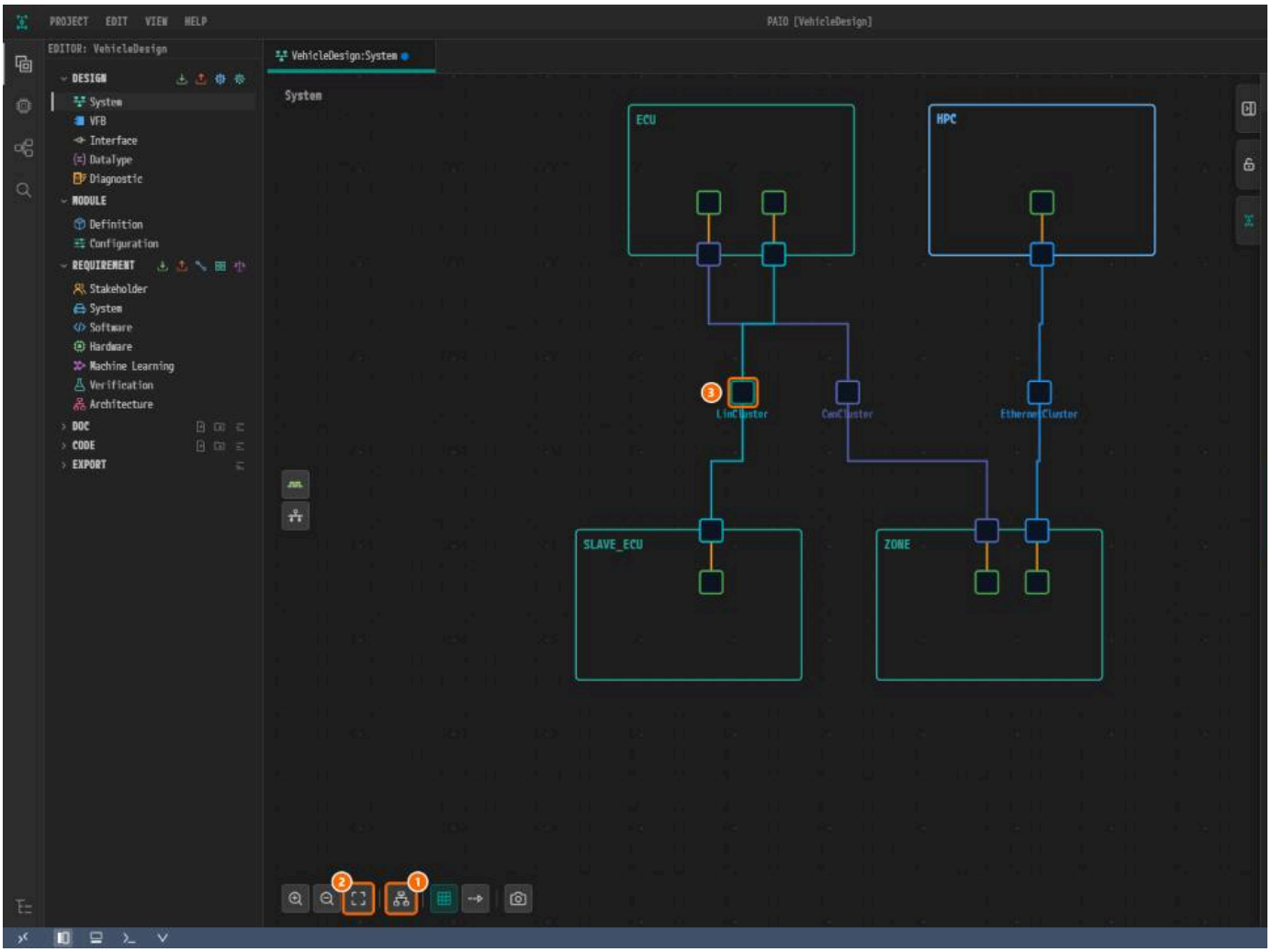
각 튜토리얼은 모델 하나를 처음부터 끝까지 완성하는 장 묶음이다. 편집기별 레퍼런스와 달리, 하나의 예제를 따라가며 어느 편집기에서 무엇을 누르는지를 순서대로 안내한다.

튜토리얼	만드는 것
AUTOSAR System	ECU 노드 3개와 HPC 1개를 CAN, 이더넷(VLAN, SOME/IP SD 멀티캐스트 포함), LIN 으로 잇는 차량 네트워크 `VehicleDesign` 을 프로젝트가 열려 있지 않은 PAIO 창에서 시작해 A
AUTOSAR CP	AUTOSAR System 튜토리얼의 프로젝트 `VehicleDesign` 에서 노드 `ECU` 에 소프트웨어 컴포넌트 두 개를 올리고, CAN 스택을 포함한 BSW 를 설정해 PRTE 코드 생성까지 진행한
AUTOSAR AP	AUTOSAR CP 튜토리얼의 프로젝트 `VehicleDesign` 에서 노드 `HPC` 에 Adaptive Application 3개(SOME/IP 서비스 `Sensor`, PARA 플랫폼 가져오기, Executable·Process, A
PNC 설계	조널 차량에 PNC(Partial Network Cluster) 3개를 설계하고, System 편집기의 토폴로지부터 검증을 거쳐 ARXML 익스포트까지

편집기별 설명은 에디터에, 임포트·익스포트와 검증은 작업 흐름에 정리되어 있다.

AUTOSAR System 튜토리얼

이 튜토리얼은 ECU 노드 3개와 HPC 노드 1개를 CAN, 이더넷, LIN 으로 잇는 차량 네트워크 `VehicleDesign` 을 PAIO 로 만든다. 프로젝트가 열려 있지 않은 PAIO 창에서 시작해 AP 세 편 가운데 첫째 편이다: AUTOSAR System → AUTOSAR CP → AUTOSAR AP.



이 튜토리얼에서 만드는 네트워크. ECU 노드 `ECU`, `ZONE`, `SLAVE_ECU`, HPC 노드 `HPC` 와 클러스터 `CanCluster`, `EthernetCluster`, `LinCluster` 이다.

무엇을 만드는가

버스마다 노드 두 개를 잇는다. 이후 CP 편은 `ECU` 에 소프트웨어 컴포넌트를, AP 편은 `HPC` 에 Adaptive Application 을 올린다.

클러스터	노드	주요 값
`CanCluster`	`ZONE`, `ECU`	버스오프 복구 L1 `100` ms, L2 `200` ms
`EthernetCluster`	`ZONE`, `HPC`	VLAN ID `13`, `HPC` 주소 `192.168.0.11`
`LinCluster`	`ECU`(마스터), `SLAVE_ECU`	`LIN 2.0`, 타임 베이스 `1` ms

준비물

- 설치되어 실행할 수 있는 PAIO.
- 미리 만든 프로젝트는 필요 없다. 1장은 프로젝트가 열려 있지 않은 PAIO 창에서 시작한다.
- 1920×1080 이상의 화면.
- 버스, 컨트롤러, 커넥터, 클러스터를 대략 알면 충분하다. 어느 화면에 무엇을 입력하는지는 장마다 보여 준다.

장 구성

장	하는 일	사용하는 편집기
1. 프로젝트와 노드 만들기	프로젝트 `VehicleDesign` 을 만들고 노드 4개를 놓는다	System 편집기
2. ECU 속성	ECU 노드의 슬립·웨이크업 속성과 시스템 버전을 정한다	System 편집기, SYSTEM 패널
3. CAN 설정	`ECU` 와 `ZONE` 을 `CanCluster` 로 잇고 버스오프 복구 값을 넣는다	System 편집기, 통신 클러스터 대화상자
4. ETHERNET 설정	`ZONE` 과 `HPC` 를 `EthernetCluster` 로 잇고 주소를 넣는다	System 편집기, 통신 클러스터 대화상자
5. LIN 설정	`ECU` 와 `SLAVE_ECU` 를 `LinCluster` 로 잇는다	System 편집기
6. 검증	VALIDATION 패널이 보고하는 일부러 넣은 결함 3개를 고친다	VALIDATION 패널, System 편집기
7. ARXML 익스포트	설계를 내보내고 XML 편집기로 결과를 확인한다	EDITOR 뷰, XML 편집기
부록	설계 체크리스트, 값 표, 파일 구성, 자주 겪는 문제	—

이 튜토리얼을 읽는 법

- **장은 순서대로 따라간다.** 각 장은 앞 장의 결과에서 시작한다.
- **반복 작업은 한 번만 보여 준다.** 하나를 절차로 보여 주고, 나머지는 값 표로 주며 같은 방법으로 입력한다.
- UI 라벨은 화면에 보이는 대로 **굵게**, 모델 요소 이름과 값은 `code` 로 적는다.

다음 단계

프로젝트와 노드 만들기부터 시작한다.

프로젝트와 노드 만들기

이 장에서 할 일

이 장에서는 프로젝트 `VehicleDesign` 을 만들고 System 편집기에 노드 네 개를 2x2 로 놓는다. ECU 노드 `ECU`, `ZONE`, `SLAVE_ECU` 와 HPC 노드 `HPC` 이다. 버스는 3장부터 붙

시작하기 전에

- 프로젝트가 열려 있지 않은 PAIO 창에서 시작한다. **EDITOR** 뷰에 `No project open` 이 보인다. PAIO 가 전의 프로젝트를 열면 **PROJECT** > **New Window** 를 선택한다.
- 이후 장에서 이 이름들을 그대로 쓰므로 바꾸지 않는다.

요소	입력하는 값	어디서
프로젝트 이름	`VehicleDesign`	**New Project** 대화상자
ECU 노드 이름 (세 개)	`ECU`, `ZONE`, `SLAVE_ECU`	System 편집기 속성 패널 **NAME**
HPC 노드 이름	`HPC`	System 편집기 속성 패널 **NAME**

새 프로젝트 만들기

1. 메뉴 막대의 **PROJECT** 를 누르고 **New Project...** 를 누른다.

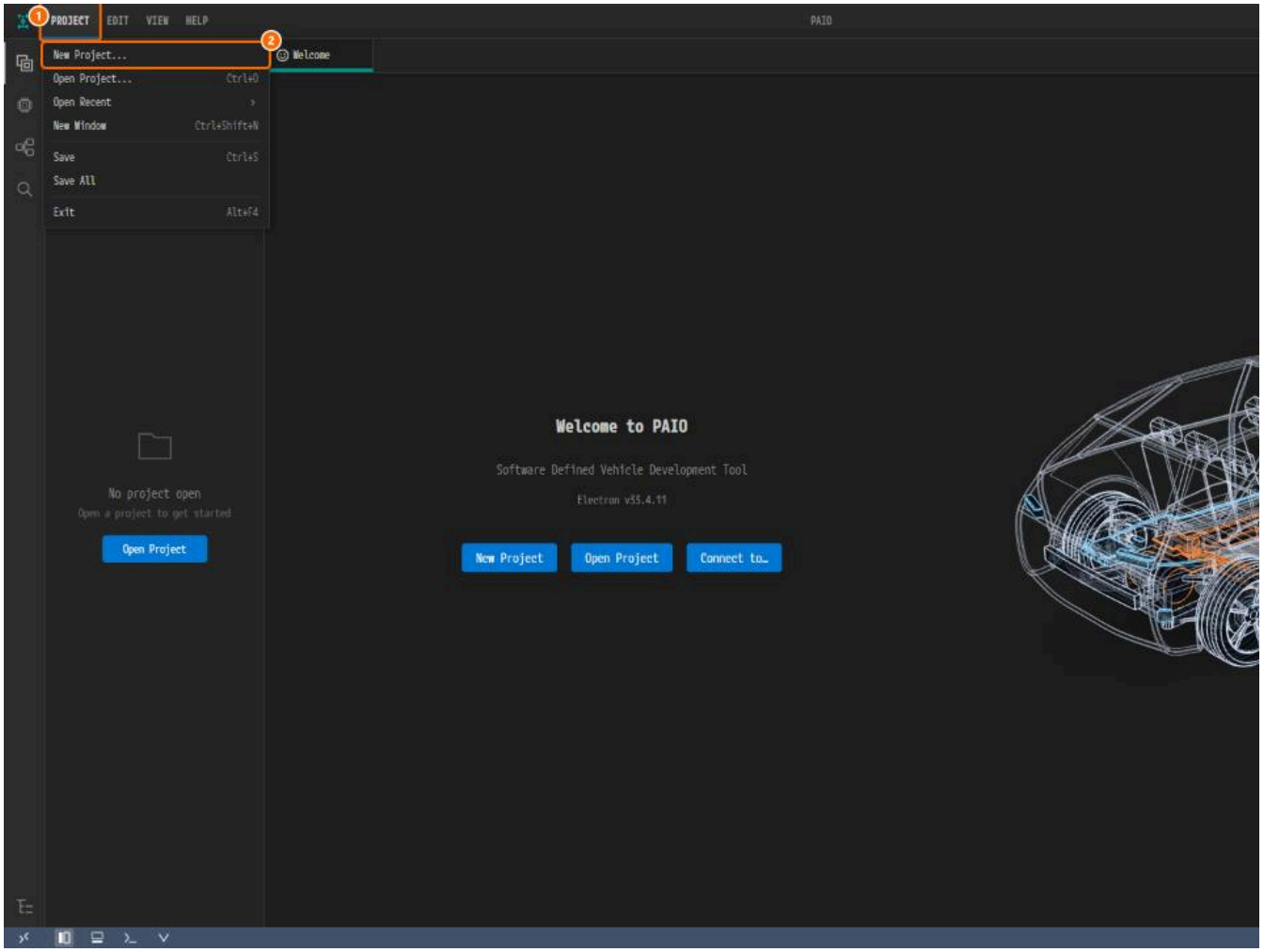


*그림 1-1. ① **PROJECT**, ② **New Project...**.*

2. **Name** 에 `VehicleDesign` 을 입력한다. **Location** 에 프로젝트를 둘 폴더를 입력하거나 **Browse...** 로 고른다. **Create** 를 누른다.

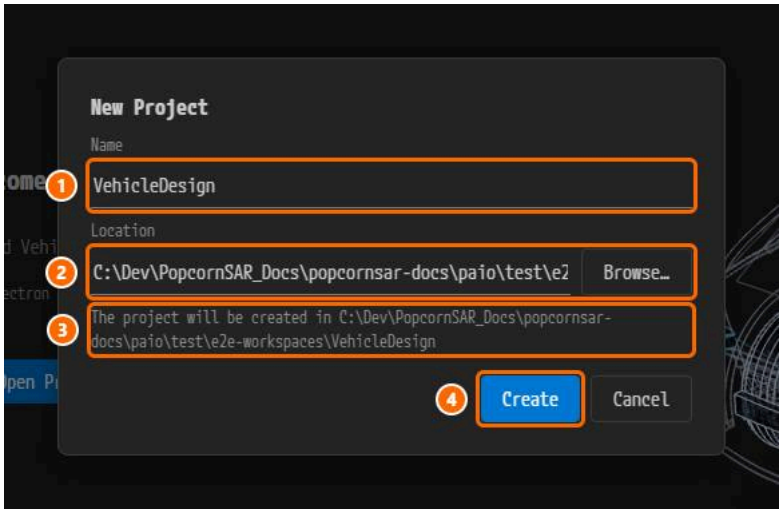


*그림 1-2. ① **Name**, ② **Location** 과 **Browse...**, ③ 만들어질 폴더, ④ **Create**. 그림의 폴더는 예시일 뿐이다.*

3. 창 제목이 `PAIO [VehicleDesign]` 이고 **EDITOR** 뷰의 **DESIGN** 아래에 **System** 이 있는 것을 확인한다.

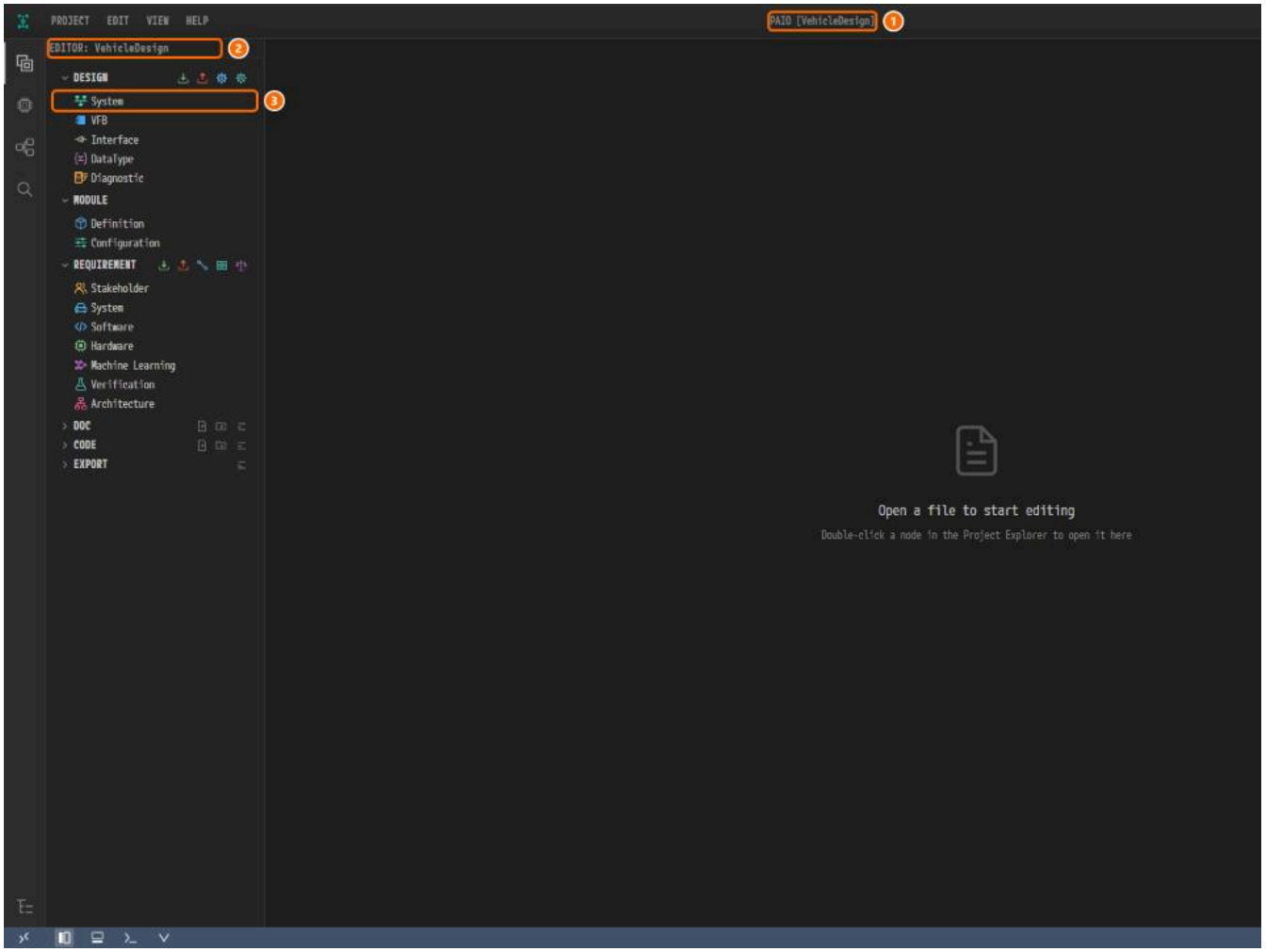


*그림 1-3. ① 창 제목, ② 뷰 머리글 `EDITOR: VehicleDesign`, ③ **System**.*

첫 번째 ECU 노드 만들기

1. **DESIGN** 아래의 **System** 을 누른다. 빈 캔버스와 함께 탭 `VehicleDesign:System` 이 열린다.

2. 캔버스의 **왼쪽 위** 빈 곳을 오른쪽 클릭하고 **Add ECU** 를 누른다.

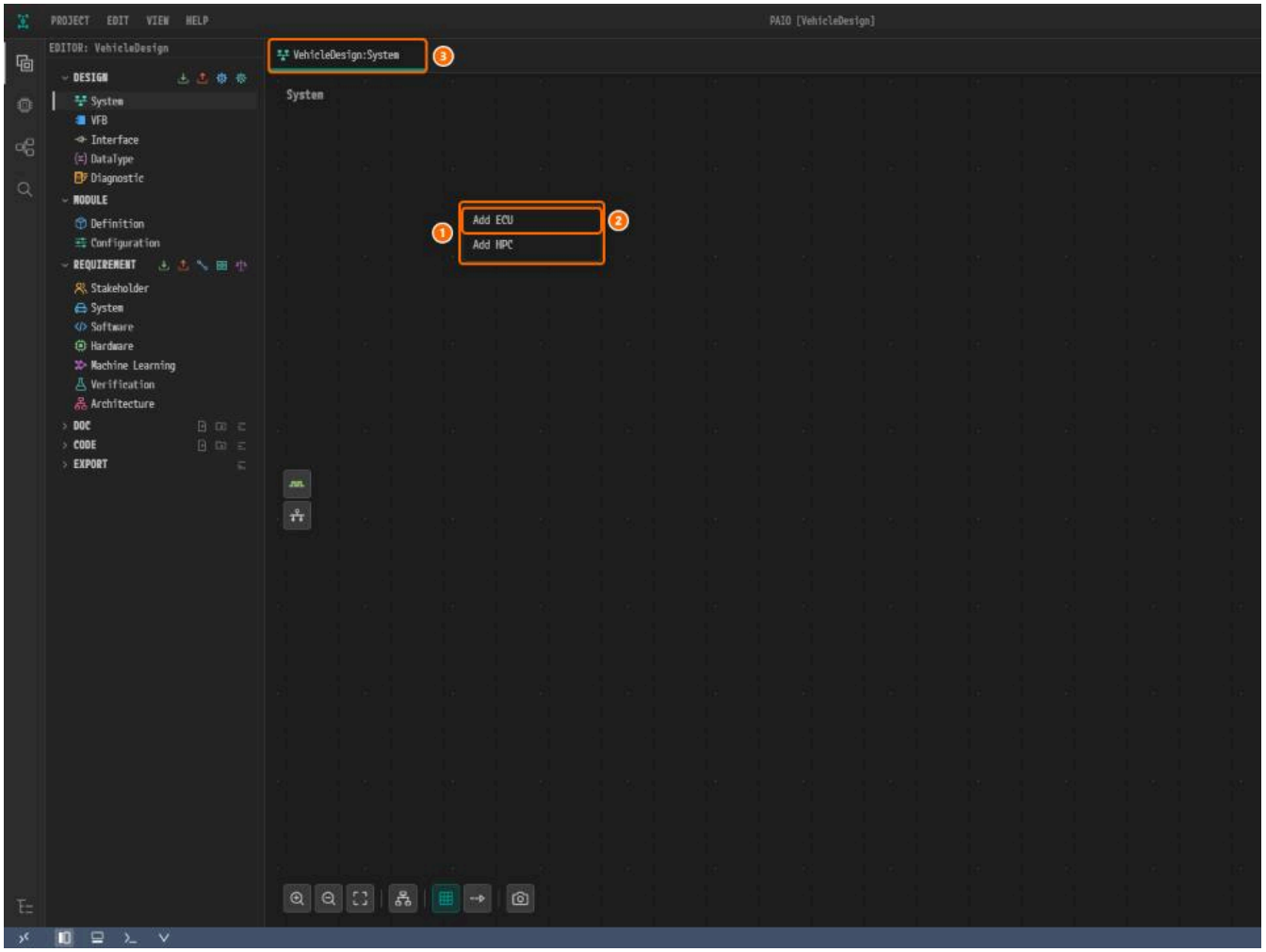


*그림 1-4. ① 캔버스 메뉴, ② **Add ECU**, ③ 탭 `VehicleDesign:System`.*

3. 노드 `ECU_0` 이 생긴다. 헤더를 누르면 제목이 **ECU** 인 속성 패널이 열린다.

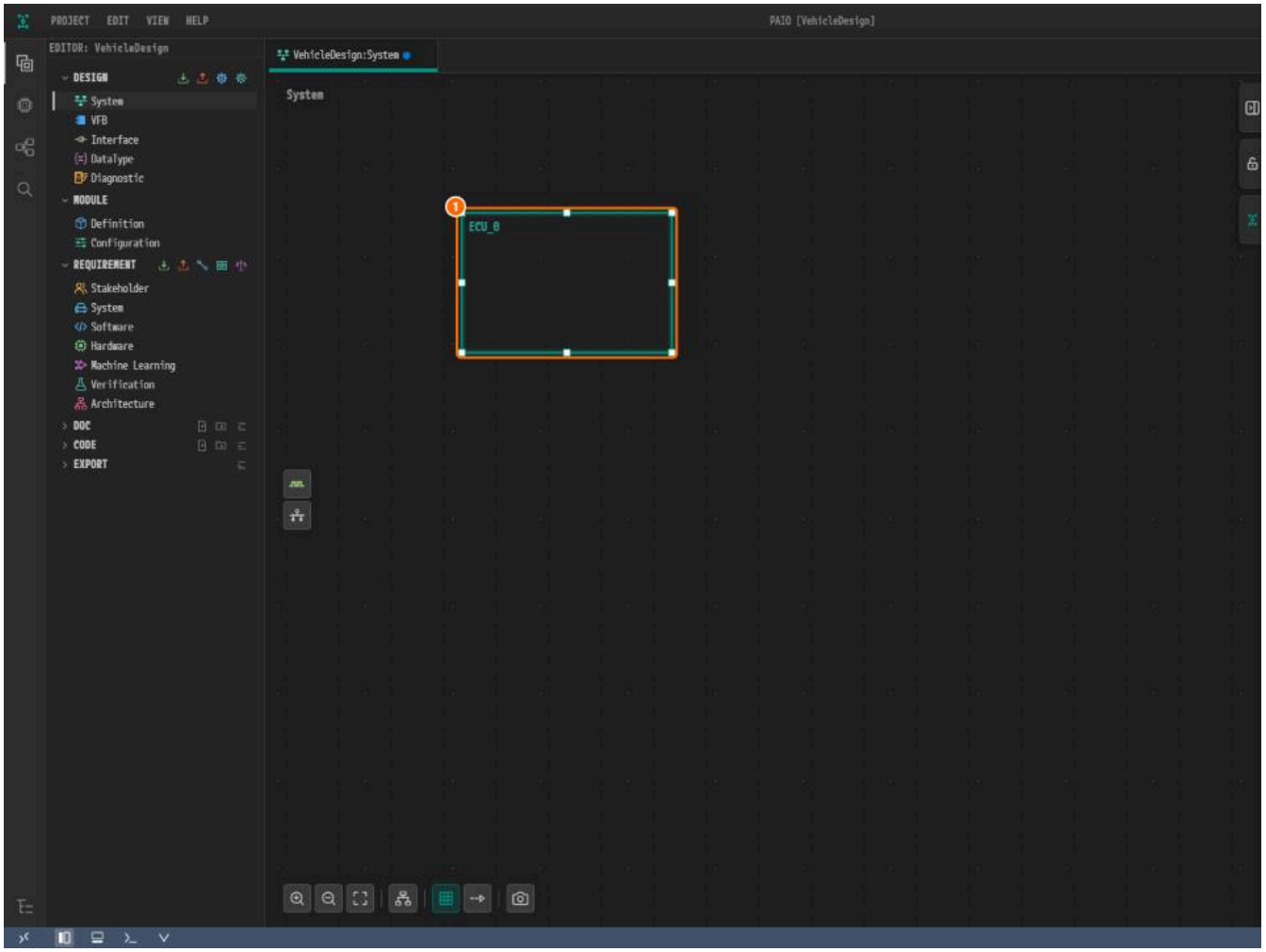


*그림 1-5. ① 노드 `ECU_0`, ② **NAME**.*

4. **NAME** 의 `ECU_0` 을 `ECU` 로 바꾸고 `Tab` 을 누른다. 노드 헤더가 `ECU` 로 바뀐다.

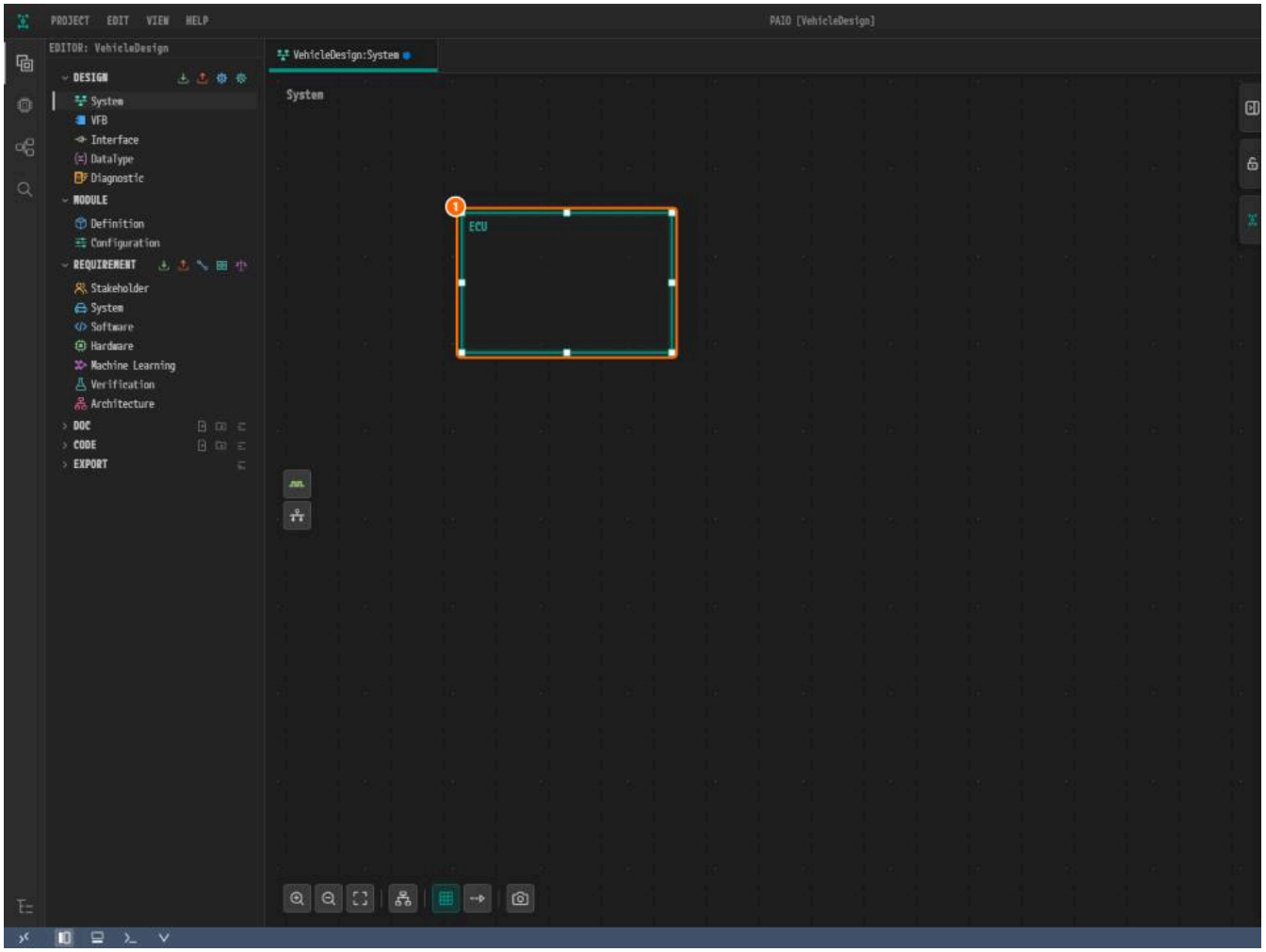


*그림 1-6. ① 노드 `ECU`, ② **NAME**.*

나머지 노드 만들기

1. **오른쪽 위**를 오른쪽 클릭하고 **Add ECU** 를 누른다. 새 노드의 이름은 다시 `ECU_0` 이다.

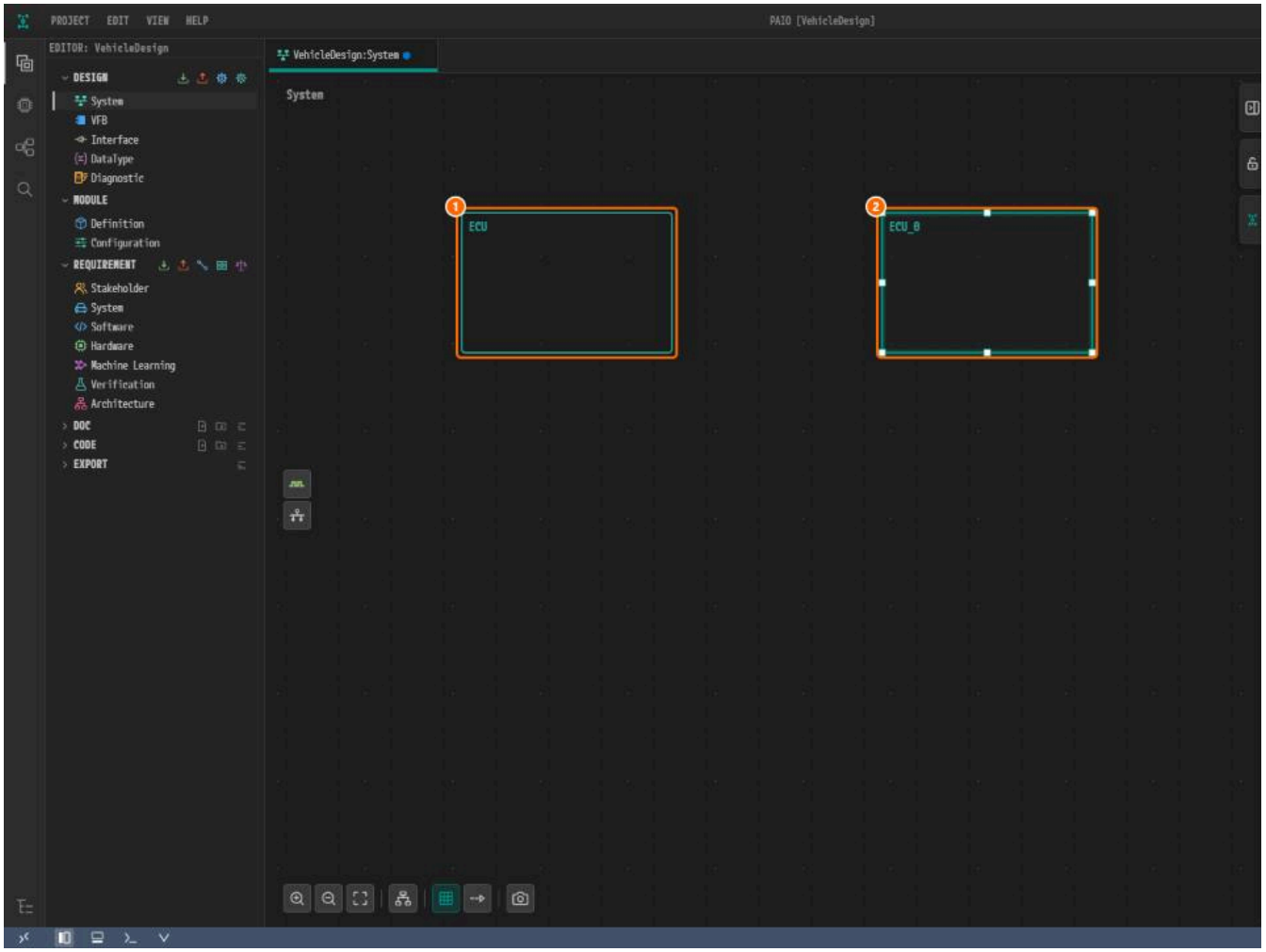


*그림 1-7. ① `ECU`, ② 새 노드 `ECU_0`, ③ **NAME**.*

2. **NAME** 에 `ZONE` 을 입력하고 `Tab` 을 누른다.

3. **왼쪽 아래**에 같은 방법으로 `SLAVE_ECU` 를 만든다.

4. **오른쪽 아래**를 오른쪽 클릭하고 **Add HPC** 를 누른다.

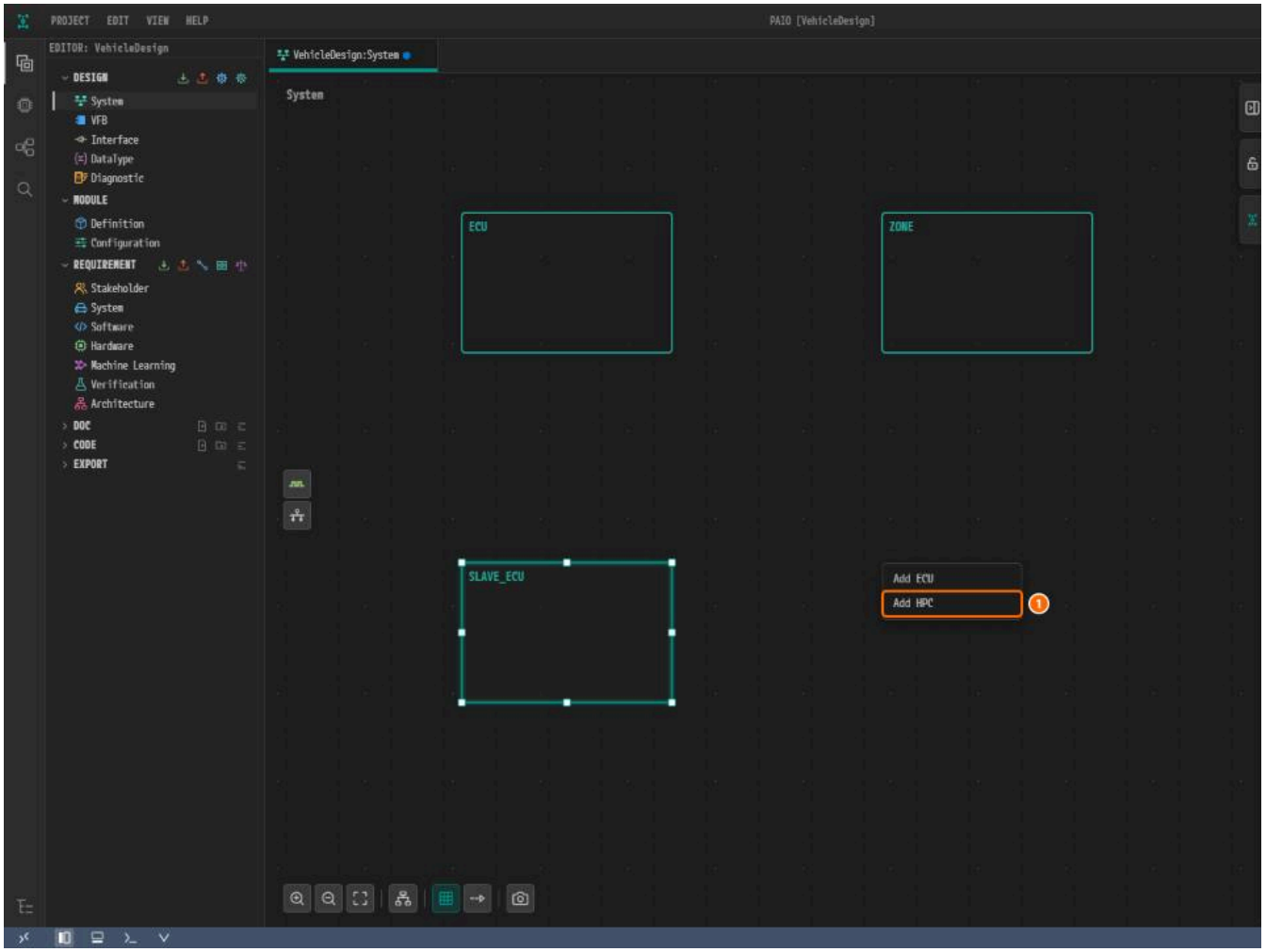


*그림 1-8. ① **Add HPC**.*

5. 새 노드 `HPC_0` 의 헤더를 누른다. 제목이 **HPC** 인 패널이 열린다. **NAME** 에 `HPC` 를 입력하고 `Tab` 을 누른다.

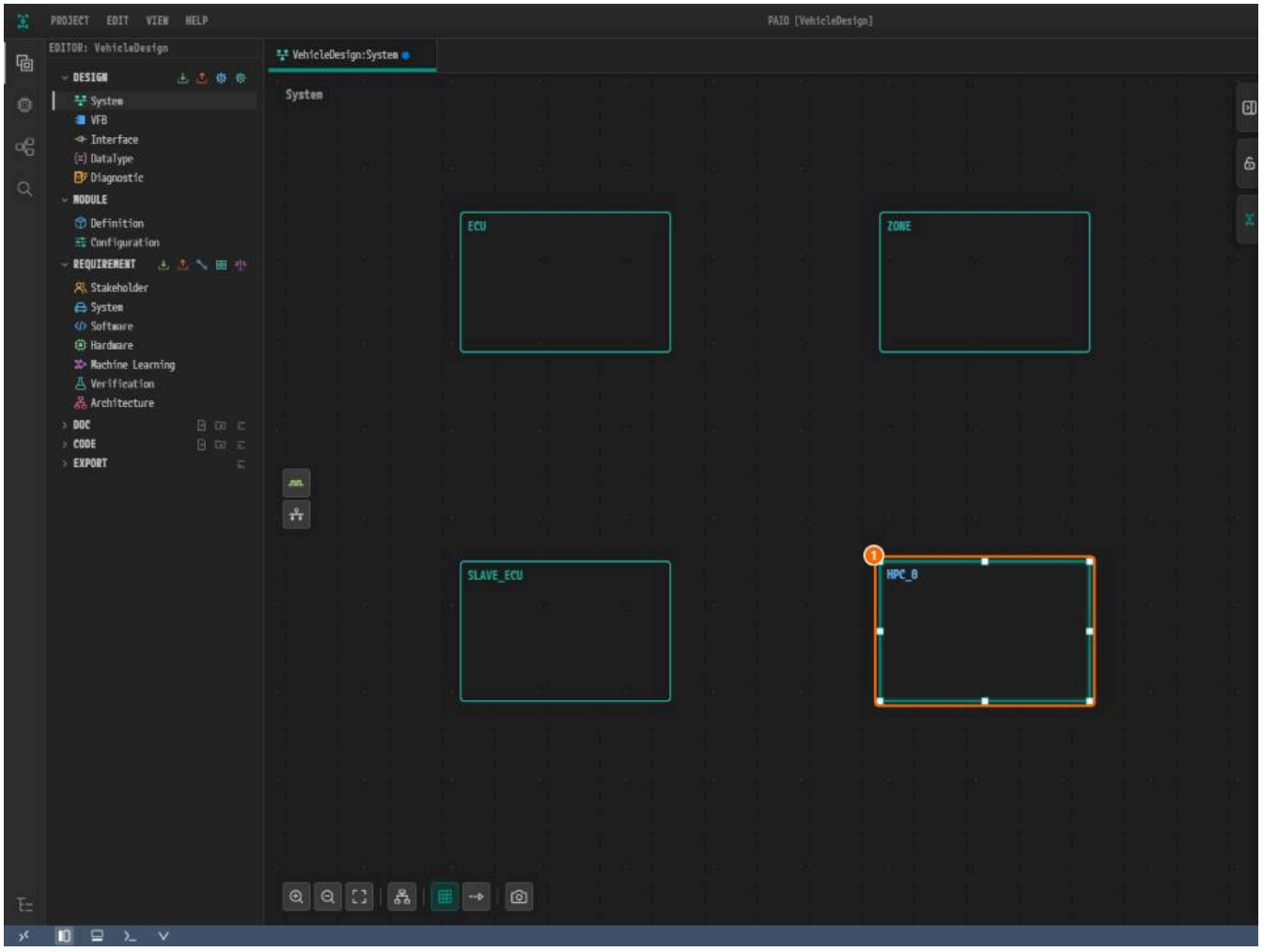


*그림 1-9. ① 노드 `HPC_0`, ② 패널 제목 **HPC**, ③ **NAME**. HPC 패널에는 **POWER MANAGEMENT** 섹션이 없다.*

노드는 이 표의 순서대로 추가한다.

순서	오른쪽 클릭 위치	메뉴 항목	생기는 노드	NAME 에 입력
1	캔버스 왼쪽 위	**Add ECU**	`ECU_0`	`ECU`
2	캔버스 오른쪽 위	**Add ECU**	`ECU_0`	`ZONE`
3	캔버스 왼쪽 아래	**Add ECU**	`ECU_0`	`SLAVE_ECU`
4	캔버스 오른쪽 아래	**Add HPC**	`HPC_0`	`HPC`

> **주의.** ECU 노드는 추가한 순서대로 익스포트되므로, ARXML 익스포트에 나오는 줄 번호는 이 순서일 때만 맞다.

저장하기

1. 네 노드 사이의 빈 캔버스를 눌러 선택을 푼다.
2. `Ctrl+S` 를 누른다. 탭의 수정 표시가 사라진다.

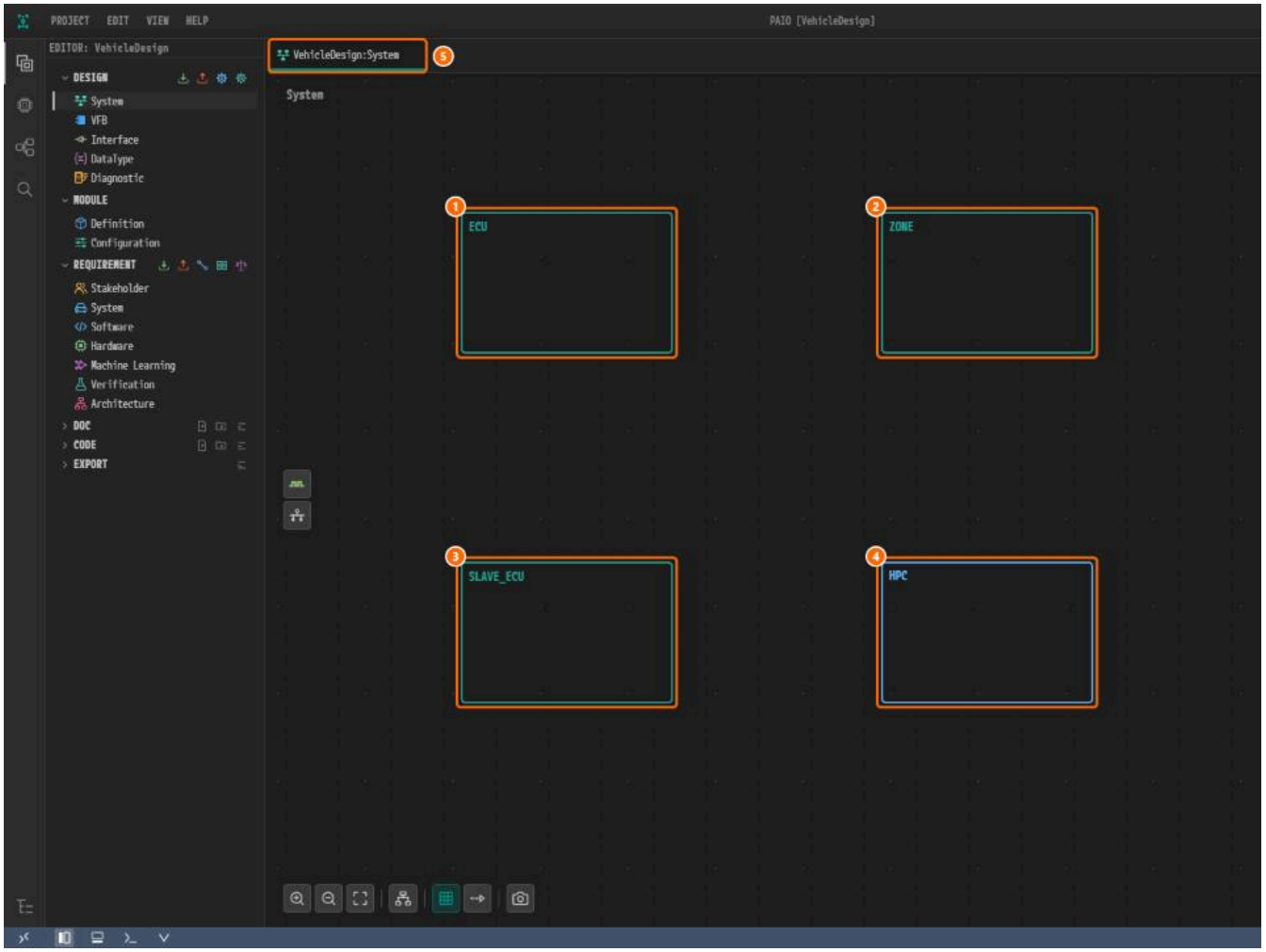


그림 1-10. ① `ECU`, ② `ZONE`, ③ `SLAVE_ECU`, ④ `HPC`, ⑤ 수정 표시가 없는 탭.

검증 결과 보기

1. 푸터의 **V** 아이콘(`Toggle Validation`)을 누른다. **VALIDATION** 패널에 `No validation issues.` 가 보인다.

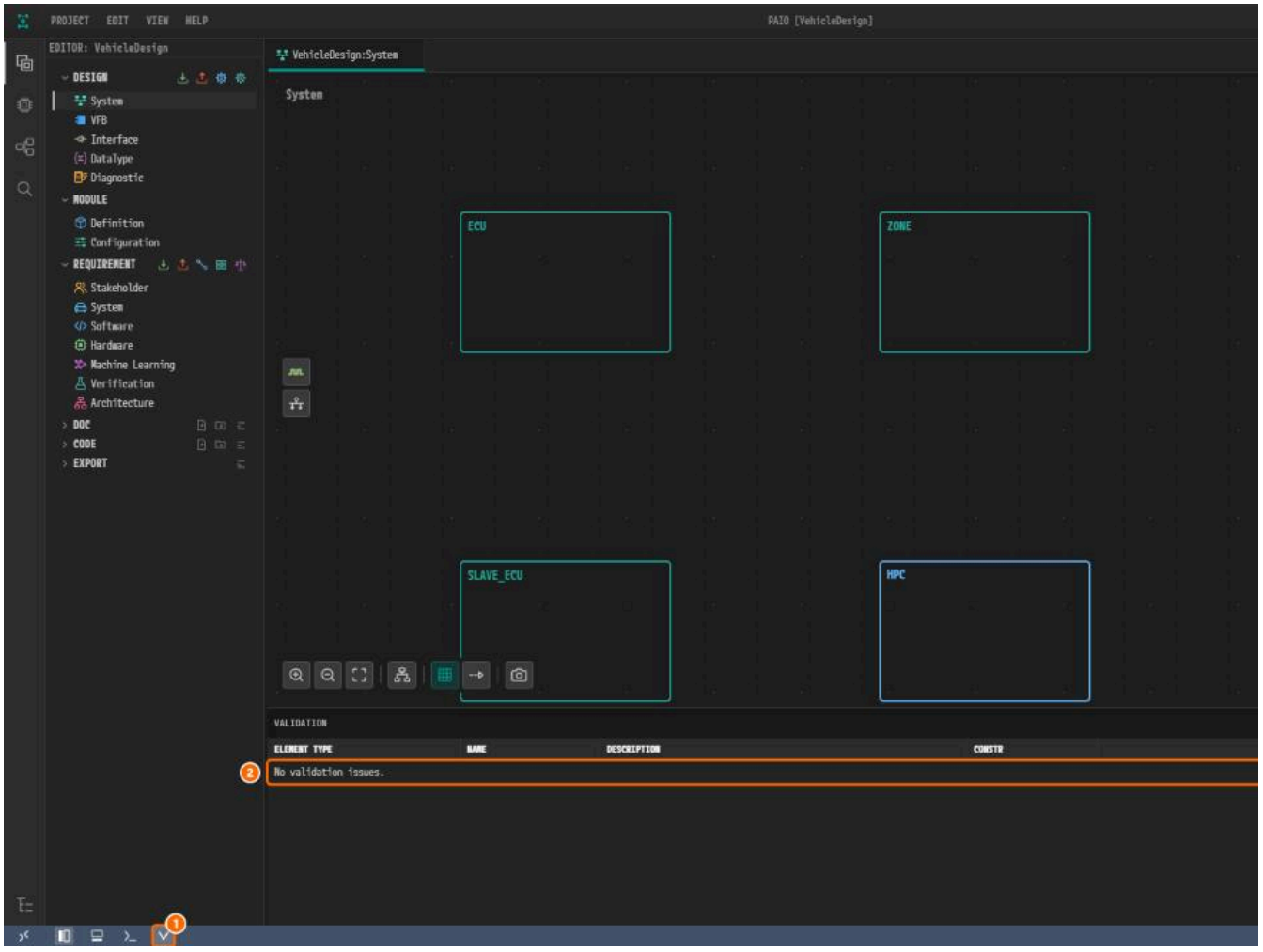


*그림 1-11. ① **V** 아이콘, ② `No validation issues.`.*

왜 이렇게 설계하는가

- **이름이 역할을 담는다.** `ECU` 는 CAN 과 LIN 마스터, `ZONE` 은 CAN 과 이더넷을 가지고, `SLAVE_ECU` 는 LIN 슬레이브, `HPC` 는 이더넷에만 연결된다.
- `HPC` **만 HPC 노드다.** 머신 설계로 저장되고 **POWER MANAGEMENT** 섹션이 없으므로, 다음 장에서 전원 관리는 ECU 노드 세 개에만 켠다.
- **노드 위치를 직접 고른다.** 노드는 오른쪽 클릭한 자리에 생기므로, 2x2 로 놓으면 이후 장의 그림과 같은 화면이 된다.

확인하기

- [ ] 창 제목이 `PAIO [VehicleDesign]` 이다.
- [ ] 탭 `VehicleDesign:System` 에 `ECU`, `ZONE`, `SLAVE_ECU`, `HPC` 가 2x2 로 놓여 있다.
- [ ] `HPC` 를 선택하면 패널 제목이 **HPC** 이고, 나머지 세 노드는 **ECU** 이다.
- [ ] 탭 `VehicleDesign:System` 에 수정 표시가 없다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

ECU 속성에서 ECU 노드 세 개의 전원 관리를 켜고 시스템 버전을 바꾼다.

ECU 속성

이 장에서 할 일

이 장에서는 ECU 노드 세 개에 슬립과 버스 웨이크업을 켜고, 시스템 버전을 `1.0.0;POPCORNSAR` 로 바꾼다.

시작하기 전에

- 프로젝트와 노드 만들기를 마친 뒤 시작한다. 탭 `VehicleDesign:System` 에 저장된 노드 네 개가 있다.
- 그림과 같은 화면으로 따라가려면 푸터의 **V** 아이콘으로 **VALIDATION** 패널을 닫는다.

어디서	항목	값
System 편집기, 노드 `ECU`, `ZONE`, `SLAVE_ECU`	**POWER MANAGEMENT > SLEEP MODE SUPPORTED**	켬
System 편집기, 노드 `ECU`, `ZONE`, `SLAVE_ECU`	**POWER MANAGEMENT > WAKE-UP OVER BUS**	켬
System 편집기, 노드 `HPC`	—	바꾸지 않는다 (확인만)
System 편집기, SYSTEM 패널	**VERSION**	`1.0.0;POPCORNSAR`

ECU 노드의 전원 관리 켜기

1. 노드 `ECU` 의 헤더를 누른다. 패널의 **POWER MANAGEMENT** 섹션에 두 버튼이 꺼진 채 보인다.

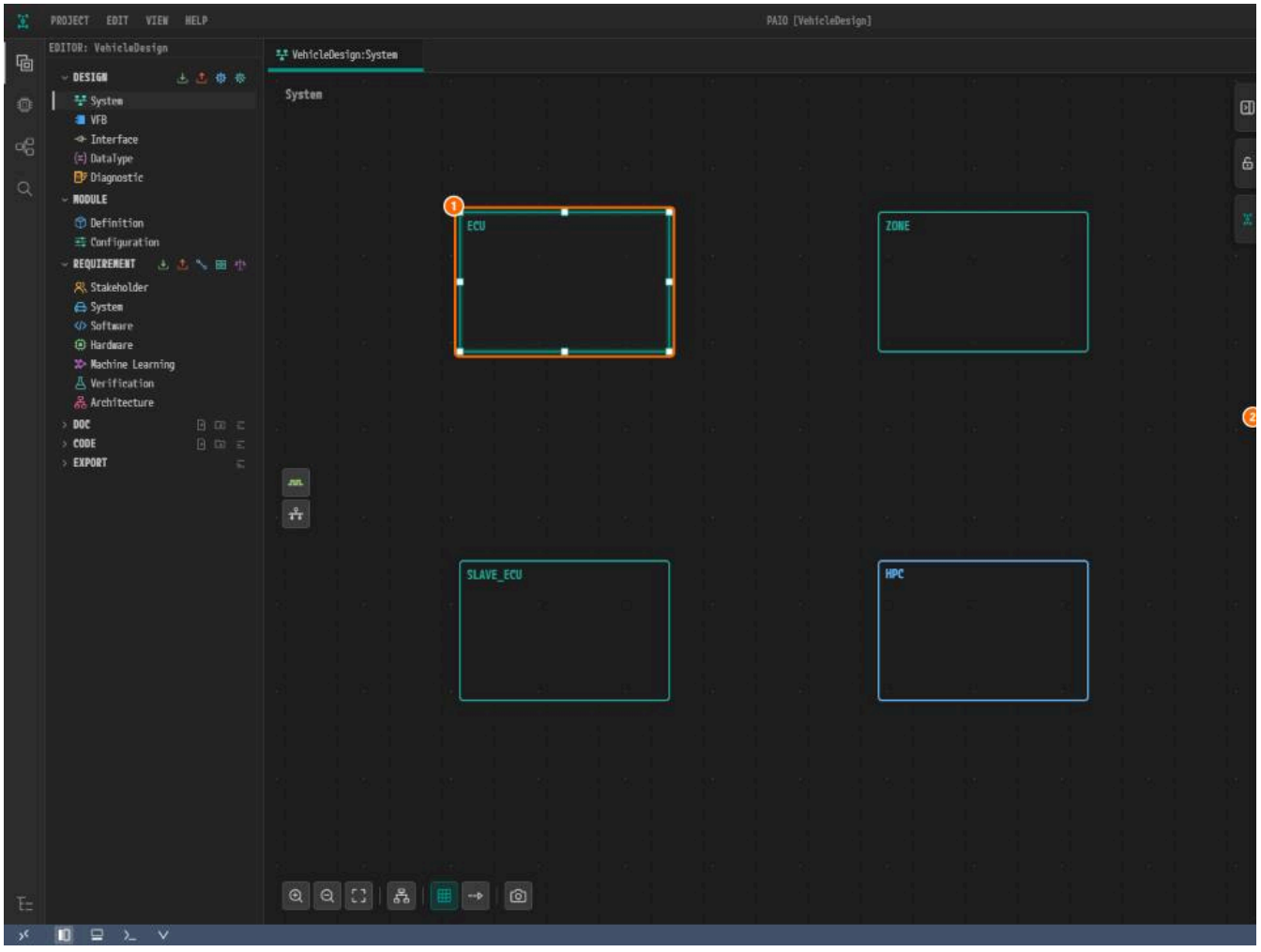


*그림 2-1. ① 노드 `ECU`, ② **POWER MANAGEMENT**, ③ **SLEEP MODE SUPPORTED**, ④ **WAKE-UP OVER BUS**.*

2. **SLEEP MODE SUPPORTED** 와 **WAKE-UP OVER BUS** 를 누른다. 둘 다 켜진다(배경이 채워진다).

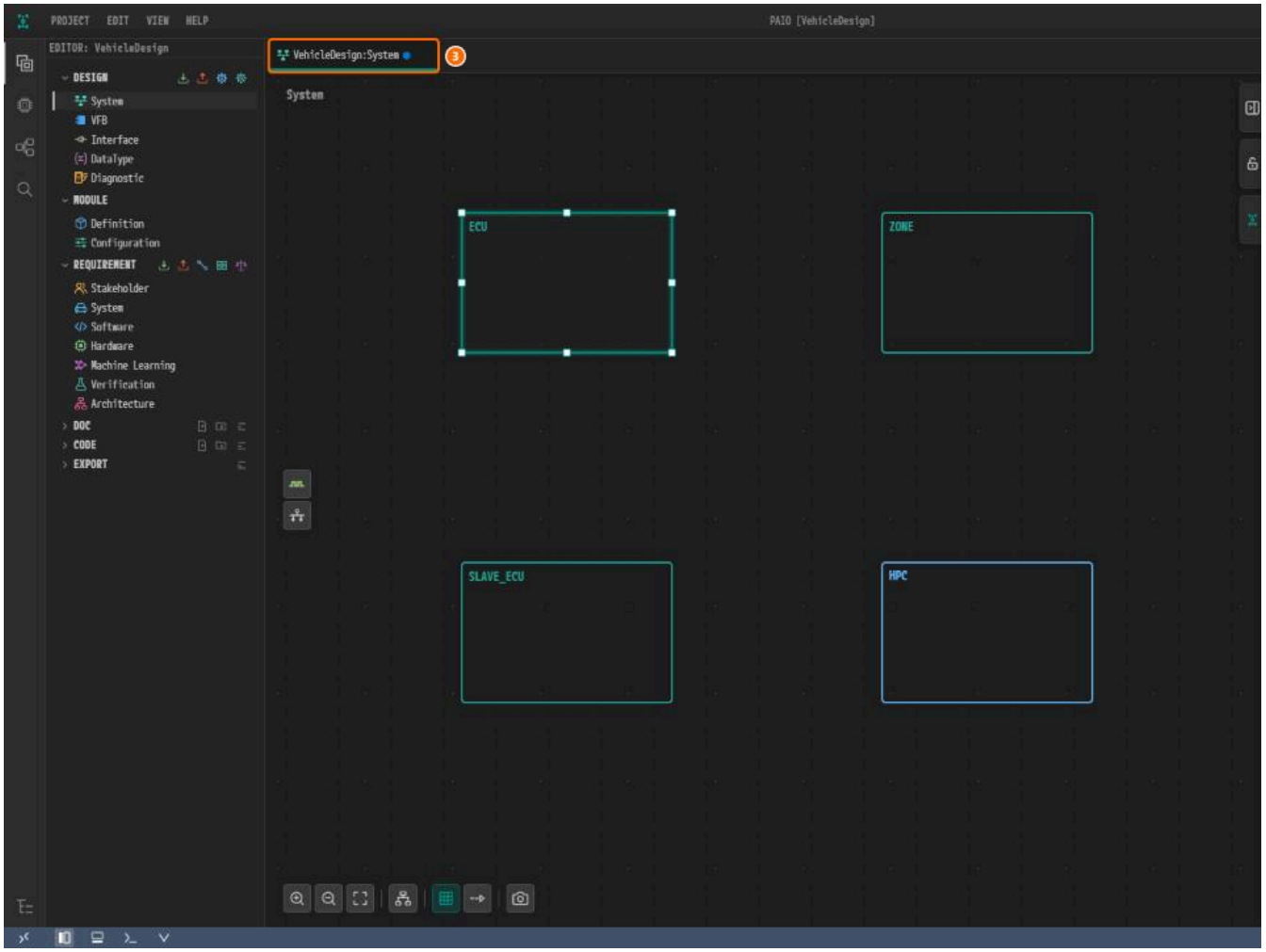


*그림 2-2. ① **SLEEP MODE SUPPORTED**, ② **WAKE-UP OVER BUS**, ③ 수정 표시가 붙은 탭.*

3. `ZONE` 과 `SLAVE_ECU` 에서도 같은 방법으로 두 토글을 켠다. 세 패널의 제목이 모두 **ECU** 이므로 어느 노드인지는 **NAME** 으로 확인한다.

4. `HPC` 의 헤더를 누른다. 패널 제목이 **HPC** 이고 **POWER MANAGEMENT** 섹션이 없는 것을 확인한다. 아무것도 바꾸지 않는다.

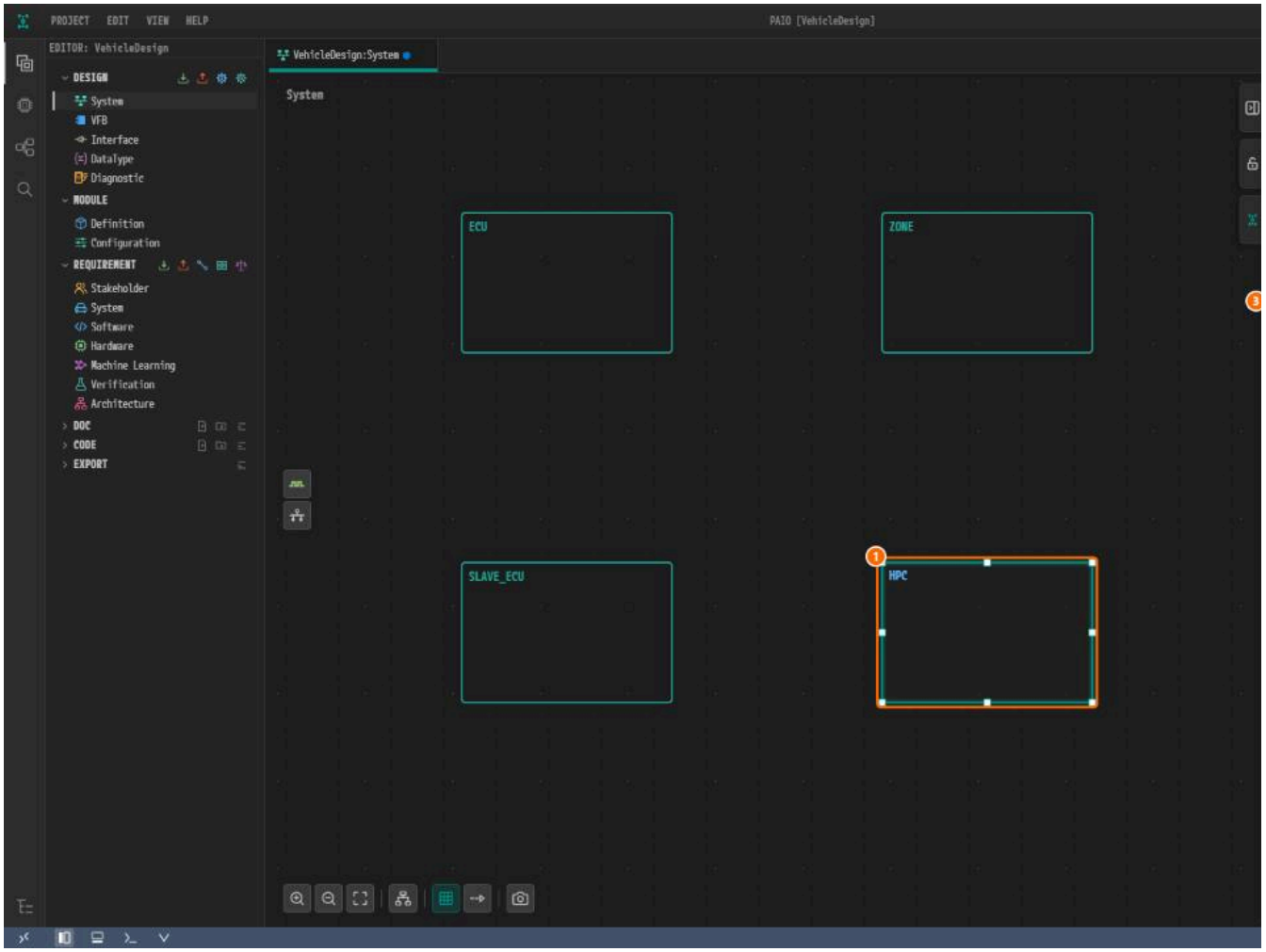


*그림 2-3. ① 노드 `HPC`, ② 패널 제목 **HPC**, ③ **POWER MANAGEMENT** 가 없는 패널 항목.*

5. 노드 사이의 빈 캔버스를 누르고 `Ctrl+S` 를 누른다.

시스템 버전 바꾸기

1. 캔버스 오른쪽 가장자리의 **System** 버튼(PAIO 로고)을 누른다. 그곳 버튼 세 개 가운데 세 번째다. **SYSTEM** 패널이 열린다.

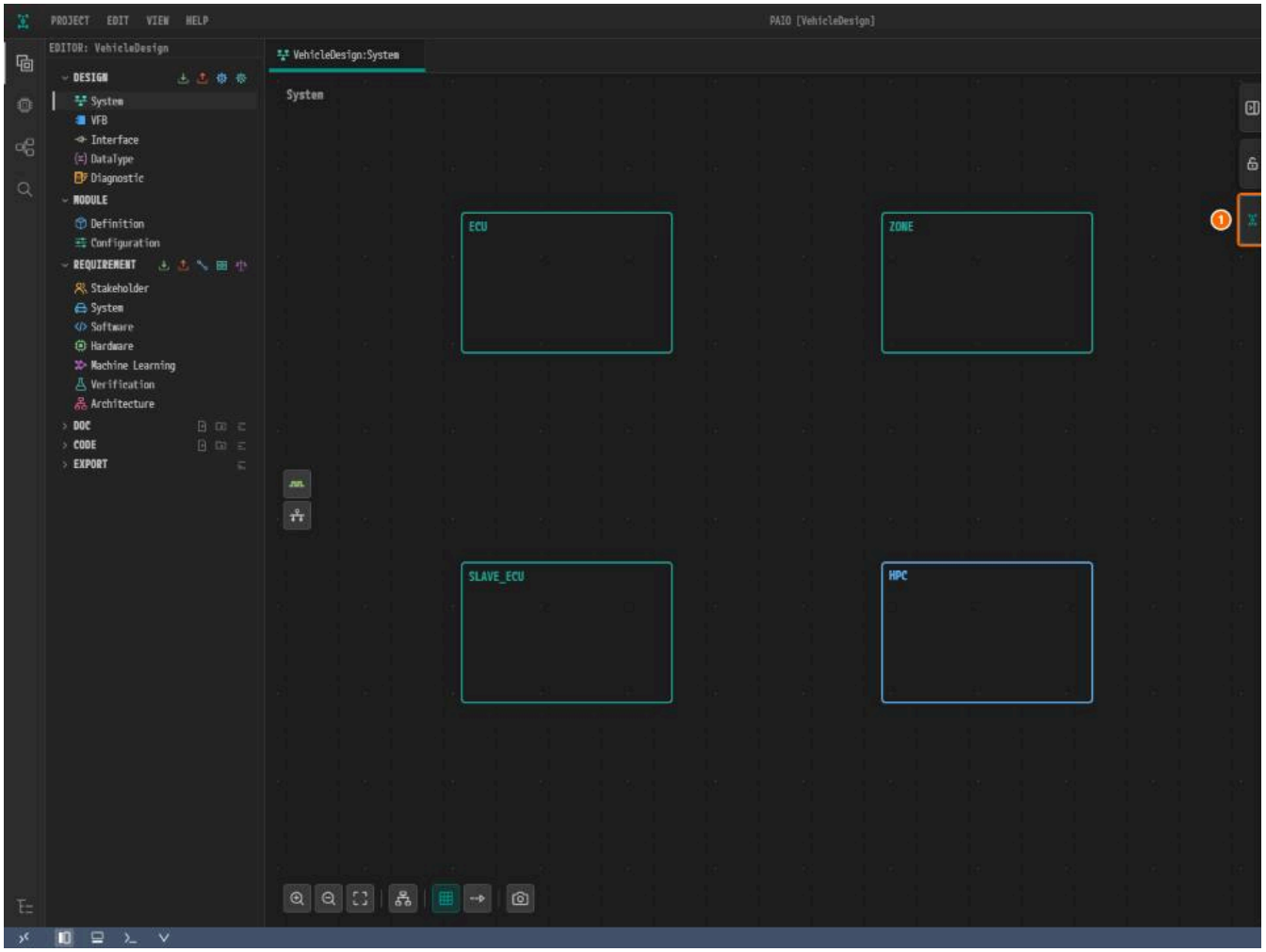


*그림 2-4. ① **System** 버튼, ② 패널 제목 **SYSTEM**, ③ 기본값 `1.0.0` 이 든 **VERSION**.*

2. **VERSION** 의 `1.0.0` 을 `1.0.0;POPCORNSAR` 로 바꾸고 `Tab` 을 누른다. 값은 입력하는 대로 저장되므로 수정 표시가 붙지 않고 저장할 것도 없다.

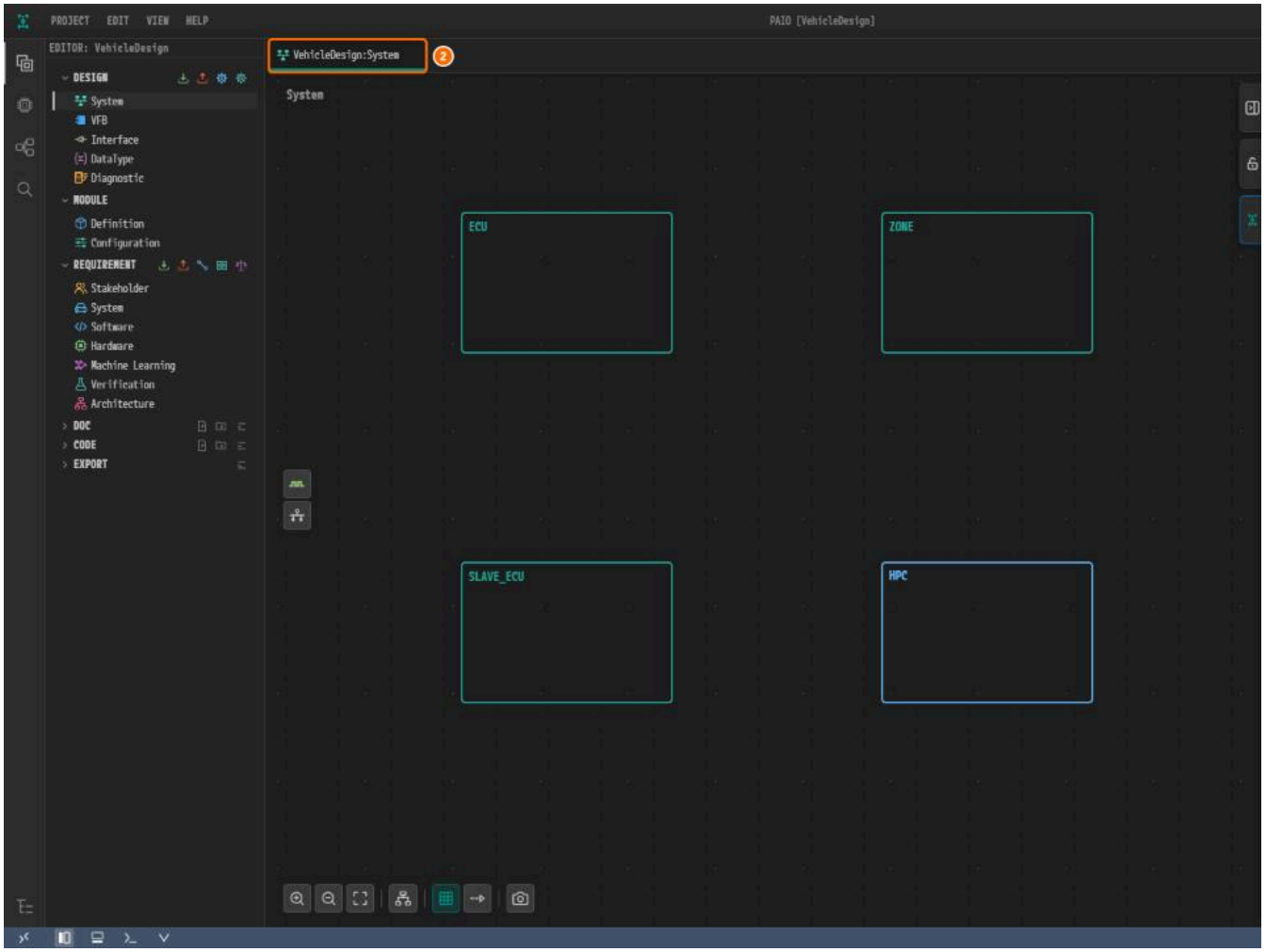


*그림 2-5. ① **VERSION**, ② 수정 표시가 없는 탭.*

3. **ECU MAPPING** 에 `ECU`, `ZONE`, `SLAVE_ECU`, `HPC` 가 모두 `No composition mapped` 로 있는 것을 확인한다.

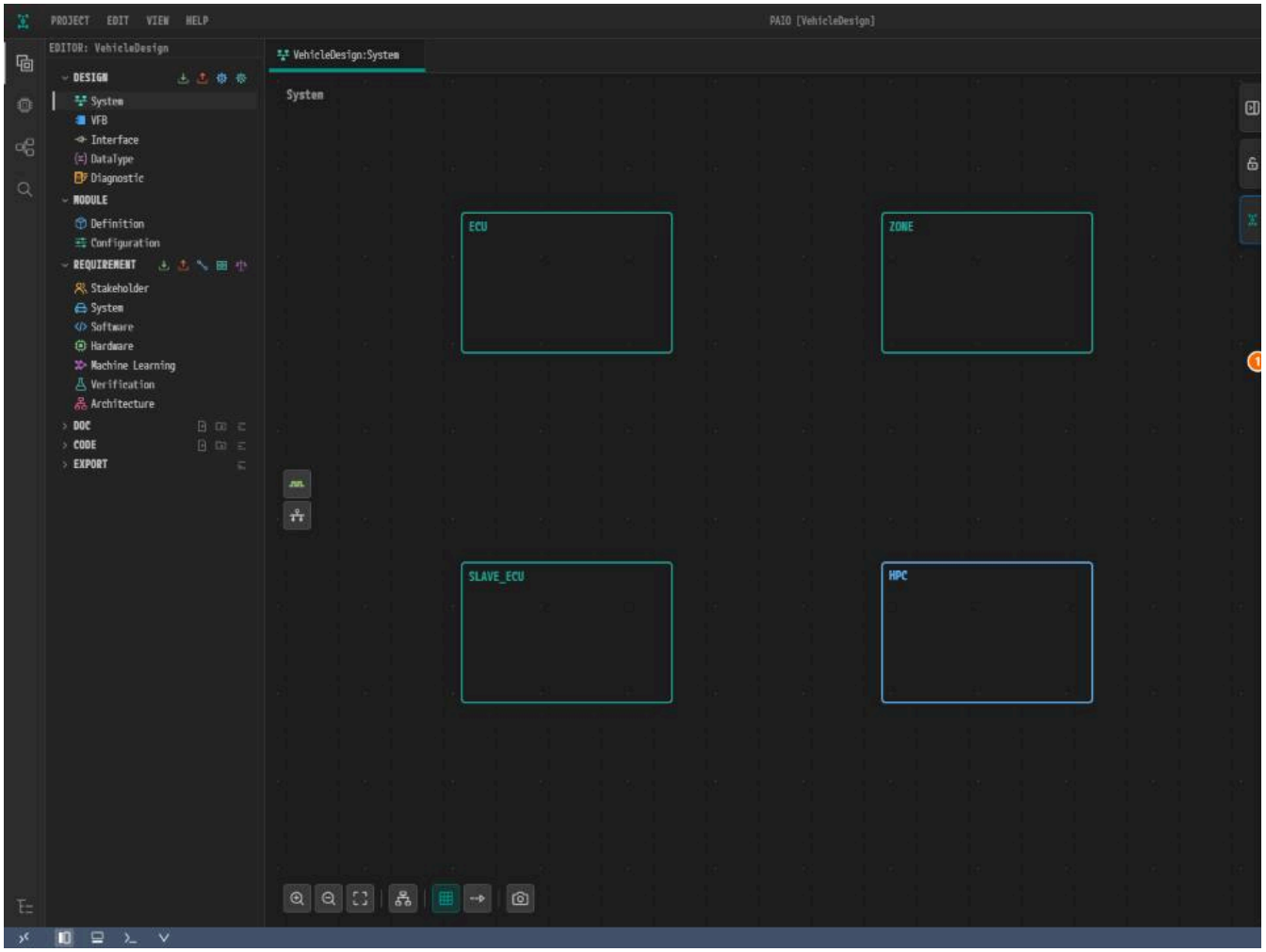


*그림 2-6. ① **ECU MAPPING** 의 네 행.*

검증 결과 보기

1. 푸터의 **V** 아이콘을 누른다. **VALIDATION** 패널에 `No validation issues.` 가 보인다.

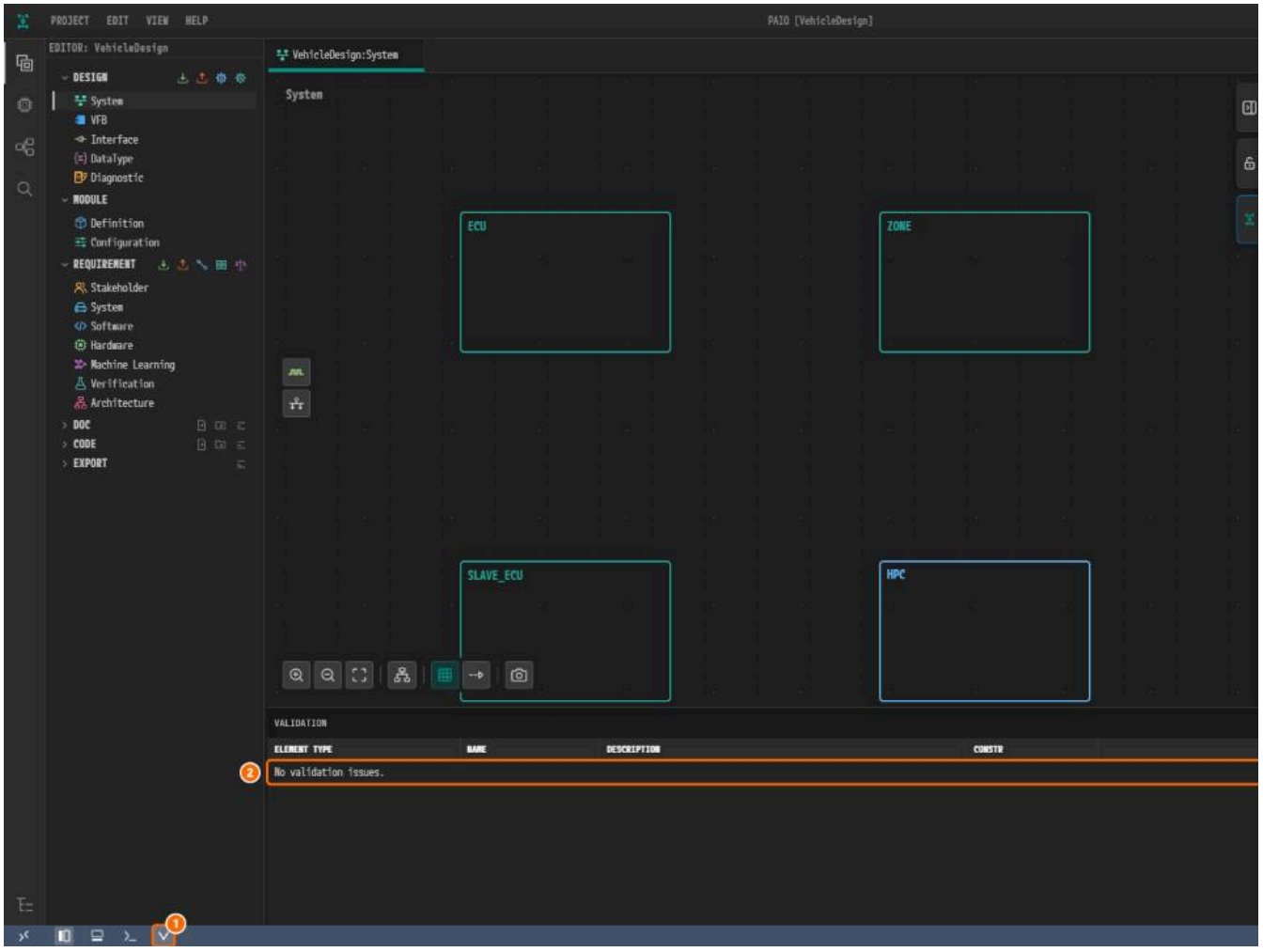


*그림 2-7. ① **V** 아이콘, ② `No validation issues.`.*

왜 이렇게 설계하는가

- **세 ECU 노드의 슬립과 웨이크업.** 참조 설계는 `ECU`, `ZONE`, `SLAVE_ECU` 의 `SLEEP-MODE-SUPPORTED` 와 `WAKE-UP-OVER-BUS-SUPPORTED` 를 `true` 로 둔다.
- `HPC` **에는 전원 관리가 없다.** HPC 노드는 머신 설계이고, 머신 설계에는 두 속성이 없다.
- **버전** `1.0.0;POPCORNSAR` 는 익스포트에서 `SYSTEM-VERSION` 이 되며 참조 설계와 같다.
- **파티션은 만들지 않는다.** 참조 설계는 `ECU` 에 파티션 `ECU_Partition0`, `ECU_Partition1` 을 두지만, System 편집기에 입력란이 없고 익스포트에도 나오지 않는다.

확인하기

- [ ] `ECU`, `ZONE`, `SLAVE_ECU` 에서 **SLEEP MODE SUPPORTED** 와 **WAKE-UP OVER BUS** 가 모두 켜져 있다.
- [ ] `HPC` 의 패널 제목이 **HPC** 이고 **POWER MANAGEMENT** 섹션이 없다.
- [ ] SYSTEM 패널의 **VERSION** 이 `1.0.0;POPCORNSAR` 이다.
- [ ] 탭 `VehicleDesign:System` 에 수정 표시가 없다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

CAN 설정에서 노드 `ECU` 와 `ZONE` 을 CAN 클러스터 `CanCluster` 로 잇는다.

CAN 설정

이 장에서 할 일

노드 `ECU` 와 `ZONE` 을 CAN 버스로 잇는다. 두 노드에 CAN 컨트롤러와 CAN 커넥터를 하나씩 만들고, 두 커넥터를 CAN 클러스터 `CanCluster` 로 연결한다. 클러스터에는 버스 오프 CAN 상태 관리자(CanSM)가 이 값을 읽는다. NM(네트워크 관리)과 TP(전송 프로토콜)는 꺼 둔다.

시작하기 전에

- ECU 속성을 마친 뒤 시작한다. 노드 네 개에는 컨트롤러, 커넥터, 클러스터가 없고, **VALIDATION** 패널은 `No validation issues.` 이다.
- 탭 `VehicleDesign:System` 을 연다(**EDITOR** > **DESIGN** > **System**). **VALIDATION** 패널은 푸터의 **V** 아이콘으로 닫아 둔다.
- 속성 패널의 입력란은 칸에서 빠져나와야 값이 확정되므로, 값마다 `Tab` 키를 누른다.

어디서	항목	값
CAN CONTROLLER (노드 `ECU`)	**NAME**	`CanController`
CAN CONTROLLER (노드 `ECU`)	**SYNC JUMP WIDTH**, **PROPAGATION SEGMENT**, **TIME SEGMENT 1**, **TIME SEGMENT 2**	네 칸 모두 `0`
CAN CONTROLLER (노드 `ZONE`)	**NAME**	`CanCommController`
CAN CONNECTOR (노드 `ECU`)	**NAME** / **CONTROLLER**	`CanConnector` / `CanController`
CAN CONNECTOR (노드 `ZONE`)	**NAME** / **CONTROLLER**	`CanConnector` / `CanCommController`
CAN CLUSTER	**NAME**	`CanCluster`
CAN CLUSTER	**CHANNEL NAME**	`PhysicalChannel` (기본값, 그대로 둔다)
CAN CLUSTER	**BOR TIME L1** / **BOR TIME L2**	`100` `ms` / `200` `ms`
CAN CLUSTER	**BOR COUNTER L1 TO L2** / **BOR TIME TX ENSURED**	`2` / `100` `ms`

CAN 컨트롤러 만들기

컨트롤러 추가하기

1. 노드 `ECU` 안쪽의 빈 곳을 오른쪽 클릭한다. 노드 이름이나 이미 놓인 요소 위는 클릭하지 않는다.

2. **Add Controller** > **CAN** 을 클릭한다.

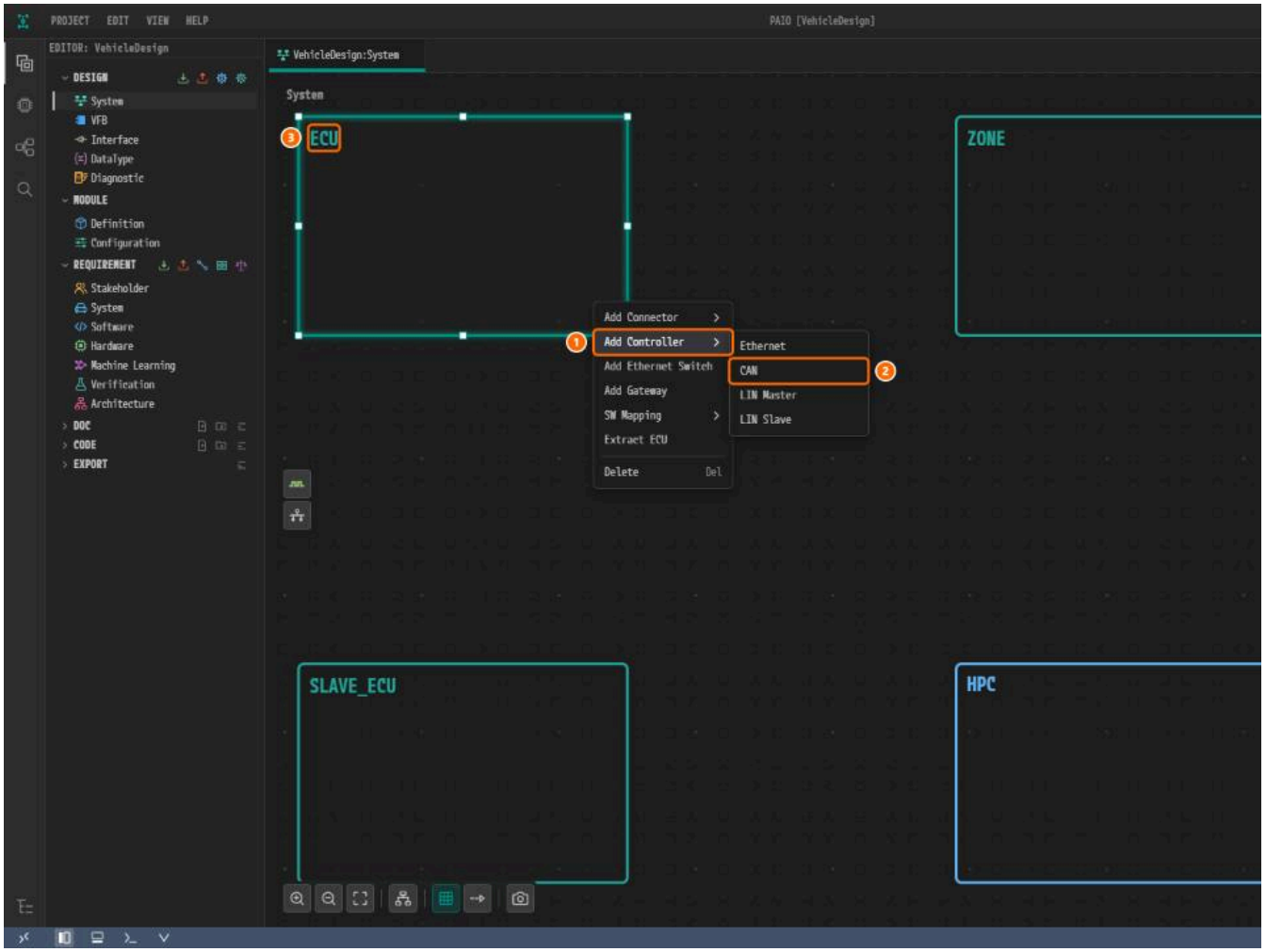


*그림 3-1. ① **Add Controller**, ② **CAN**, ③ 노드 `ECU`.*

3. 노드 안에 생긴 작은 사각형을 클릭한다. **NAME** 이 `ECU_Can_Controller_0` 인 **CAN CONTROLLER** 패널이 열린다.

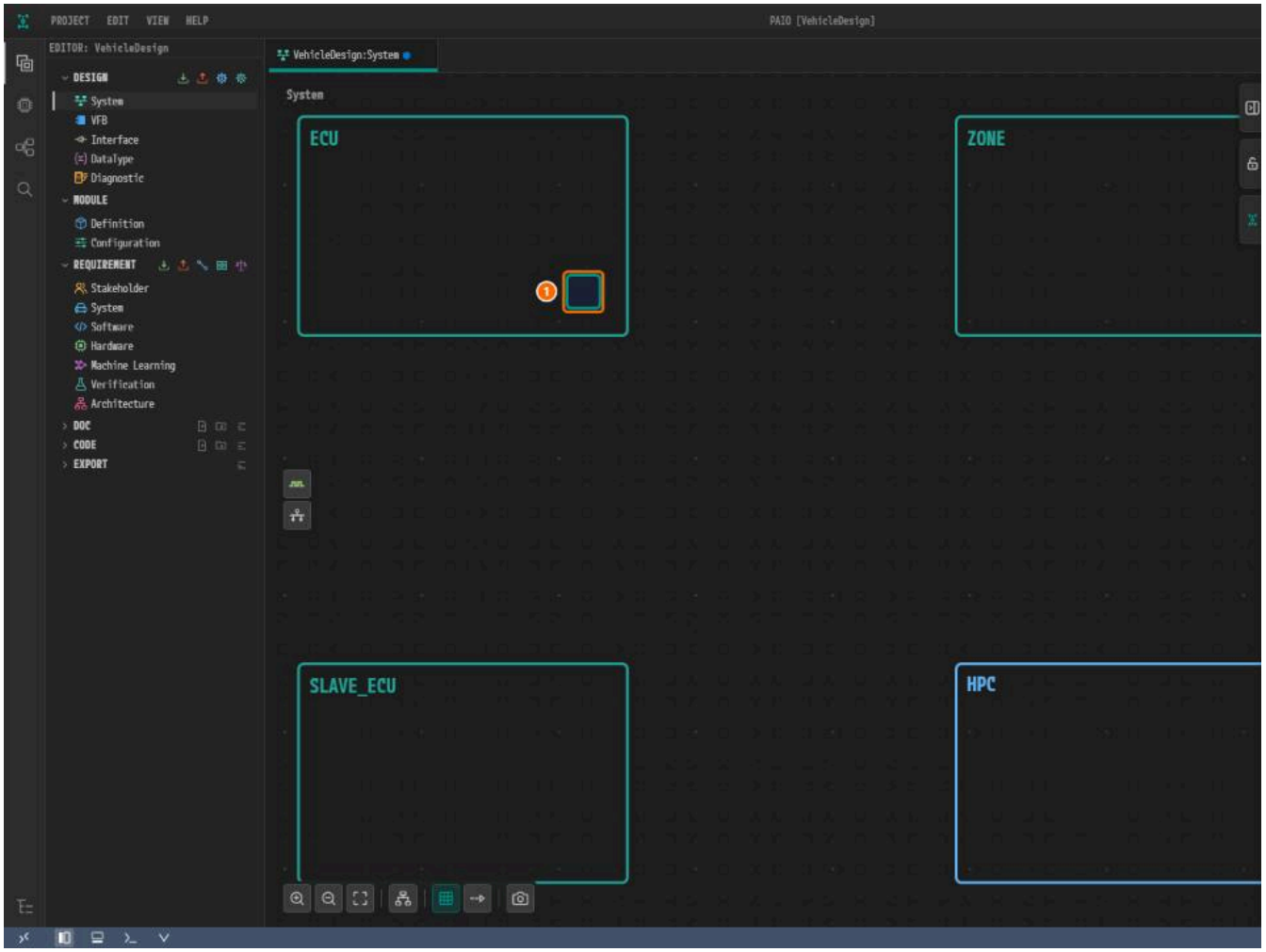


*그림 3-2. ① 컨트롤러, ② **CAN CONTROLLER**, ③ **NAME**, ④ **CAN / CAN FD** 토글, ⑤ **BIT TIMING CONFIGURATION**.*

이름과 비트 타이밍 입력하기

1. **NAME** 에 `CanController` 를 입력한다.
2. 토글은 **CAN** 그대로 둔다.
3. **SYNC JUMP WIDTH**, **PROPAGATION SEGMENT**, **TIME SEGMENT 1**, **TIME SEGMENT 2** 에 `0` 을 입력한다. **BIT TIMING REQUIREMENTS** 는 비워 둔다.

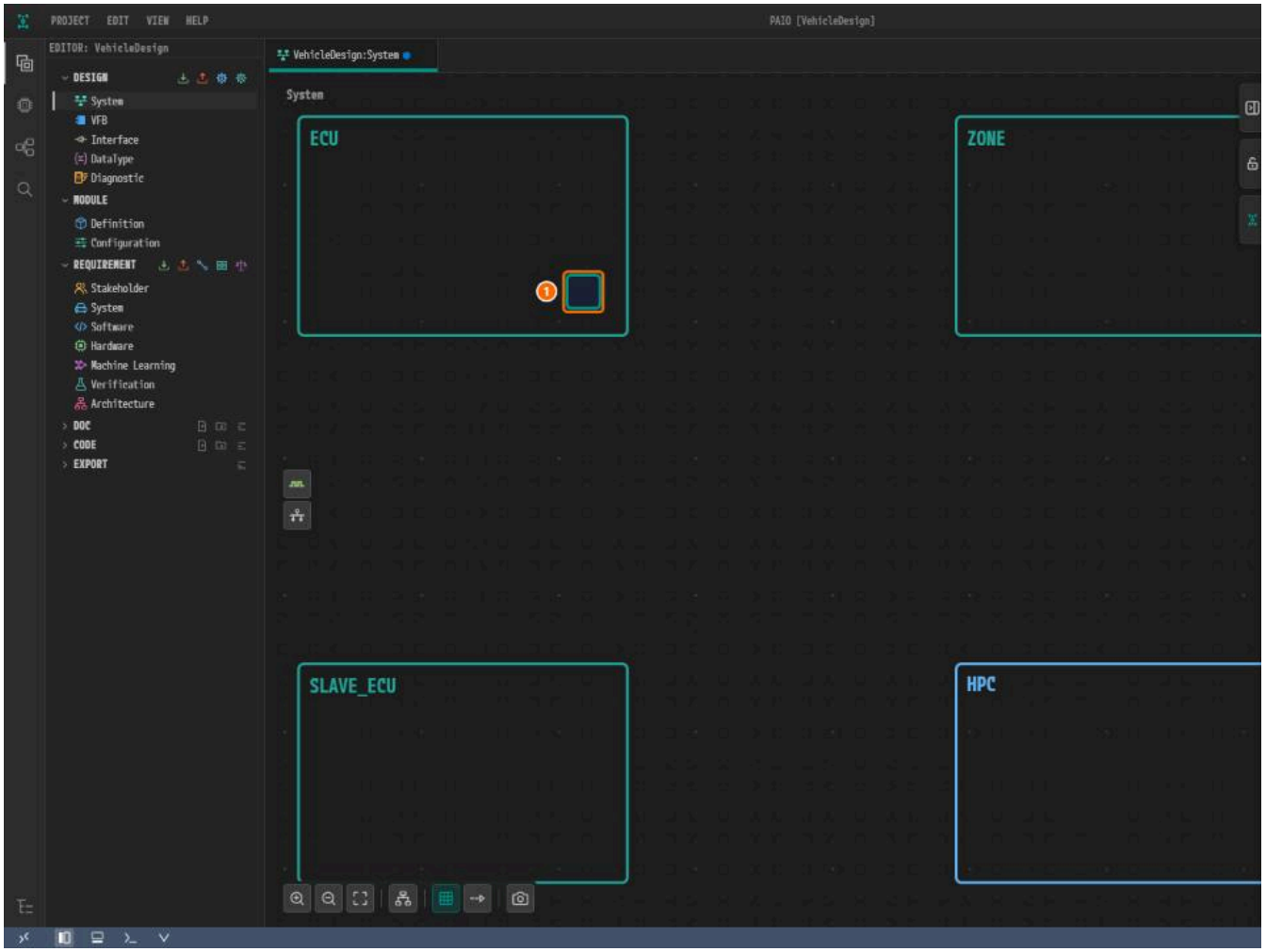


*그림 3-3. ① 컨트롤러, ② `0` 을 넣은 비트 타이밍 네 칸, ③ **NAME** = `CanController`.*

ZONE 에도 컨트롤러 만들기

1. `ZONE` 안쪽의 빈 곳을 오른쪽 클릭해 **Add Controller > CAN** 을 클릭하고, 새 사각형을 클릭한다.
2. **NAME** 에 `CanCommController` 를 입력한다. 비트 타이밍 칸은 비워 둔다.

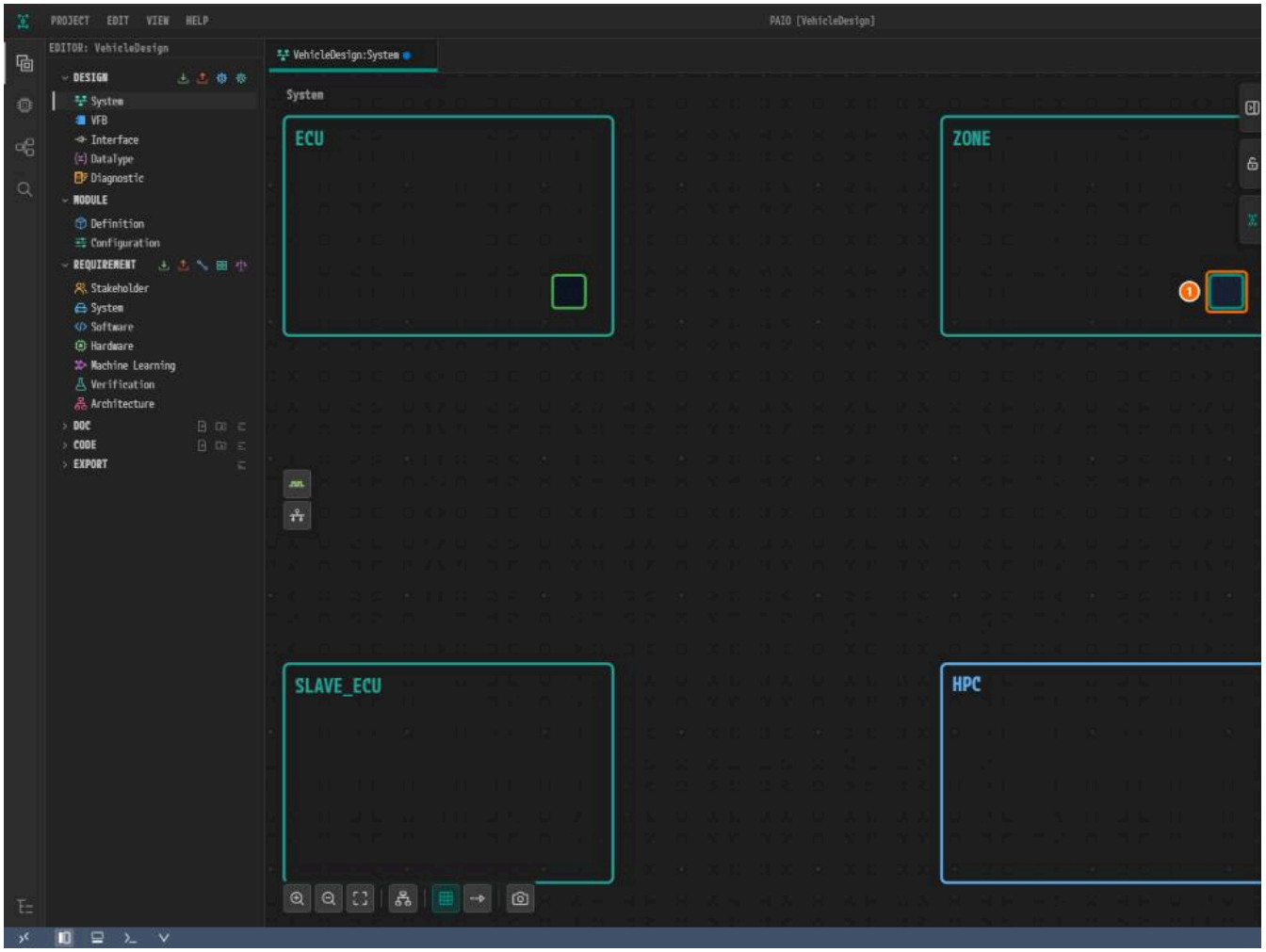


*그림 3-4. ① `ZONE` 안의 컨트롤러, ② **NAME** = `CanCommController`.*

CAN 커넥터 만들기

커넥터 추가하기

1. `ECU` 안쪽의 빈 곳을 오른쪽 클릭해 **Add Connector > CAN** 을 클릭한다.

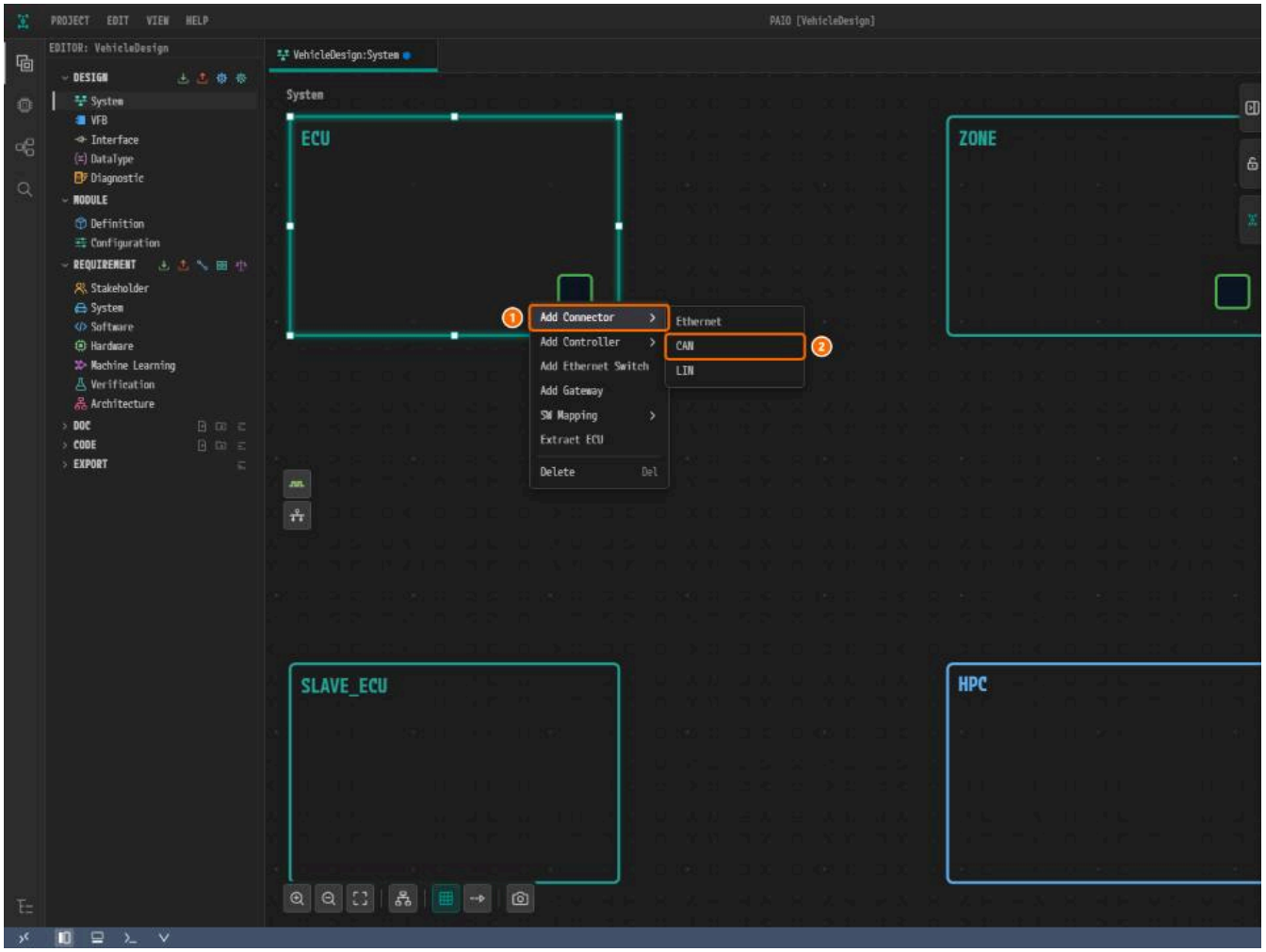


*그림 3-5. ① **Add Connector**, ② **CAN**.*

2. `ECU` 아래 가장자리에 생긴 커넥터 사각형을 클릭한다. **CAN CONNECTOR** 패널이 열린다. 컨트롤러를 고를 때까지 **CONTROLLER** 는 빨갛고 `Controller is required` 가

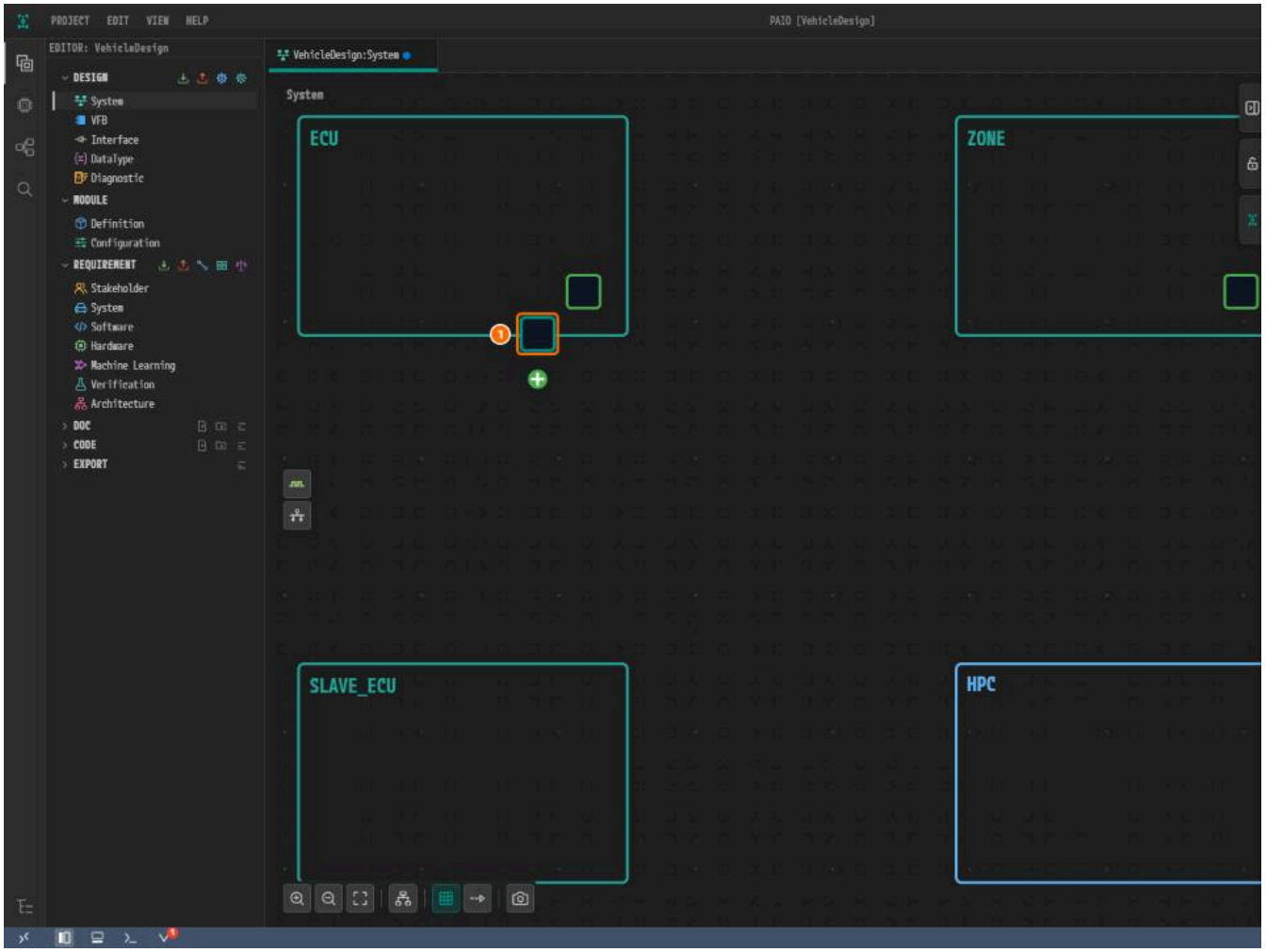


*그림 3-6. ① 커넥터, ② **NAME**, ③ **CONTROLLER**, ④ **CONNECTED CHANNEL**.*

이름과 컨트롤러 지정하기

1. **NAME** 에 `CanConnector` 를 입력한다.
2. **CONTROLLER** 를 클릭하고 목록에서 `CanController` 를 클릭한다. 커넥터와 컨트롤러가 주황 선으로 이어진다.

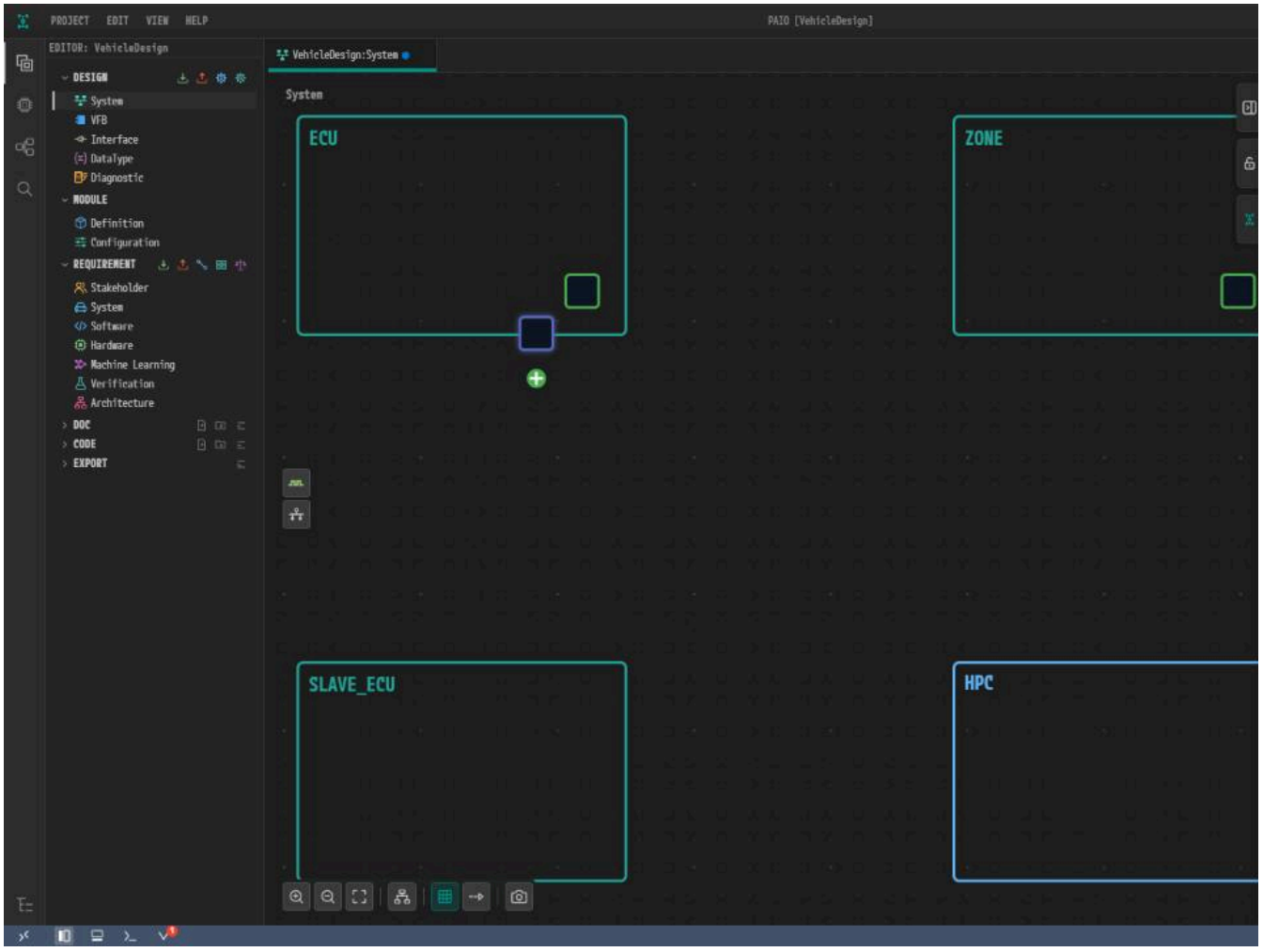


*그림 3-7. ① `CanController`, ② **NAME**, ③ **CONTROLLER**.*

ZONE 에도 커넥터 만들기

1. `ZONE` 안쪽의 빈 곳을 오른쪽 클릭해 **Add Connector > CAN** 을 클릭하고, 새 커넥터를 클릭한다.
2. **NAME** 에 `CanConnector` 를 입력하고 **CONTROLLER** 에서 `CanCommController` 를 고른다.

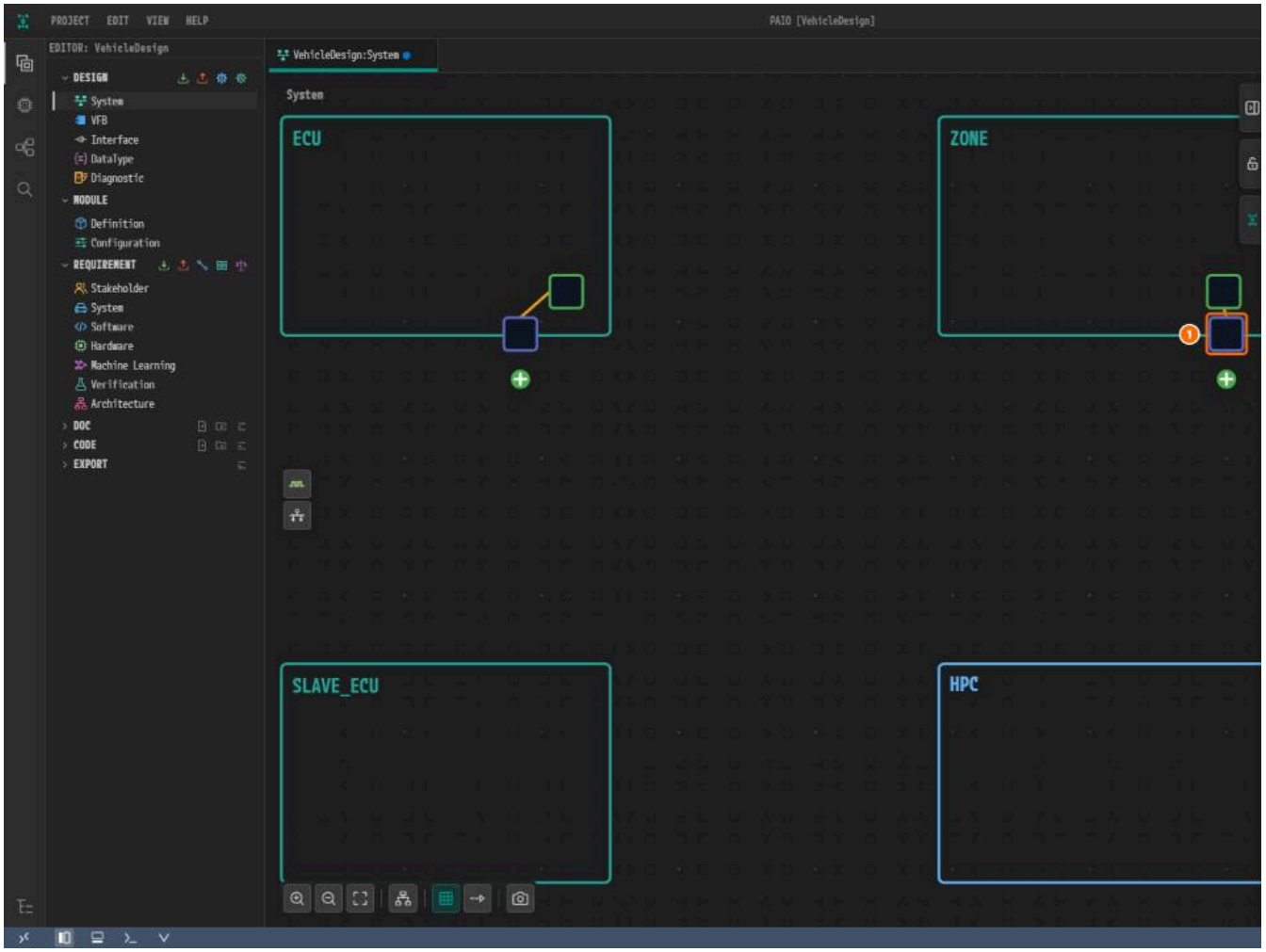


*그림 3-8. ① `ZONE` 의 커넥터, ② **NAME**, ③ **CONTROLLER**.*

커넥터는 각자 자기 노드에 속하므로 두 커넥터가 같은 이름 `CanConnector` 를 써도 된다.

CAN 클러스터 만들기

ZONE 커넥터에서 클러스터 만들기

참조 설계처럼 채널에서 `ZONE` 의 커넥터가 먼저 오도록 `ZONE` 에서 클러스터를 만든다.

1. `ZONE` 의 `CanConnector` 아래 초록 **+** 배지를 클릭하고 **Create CommunicationCluster [Can]** 을 클릭한다.

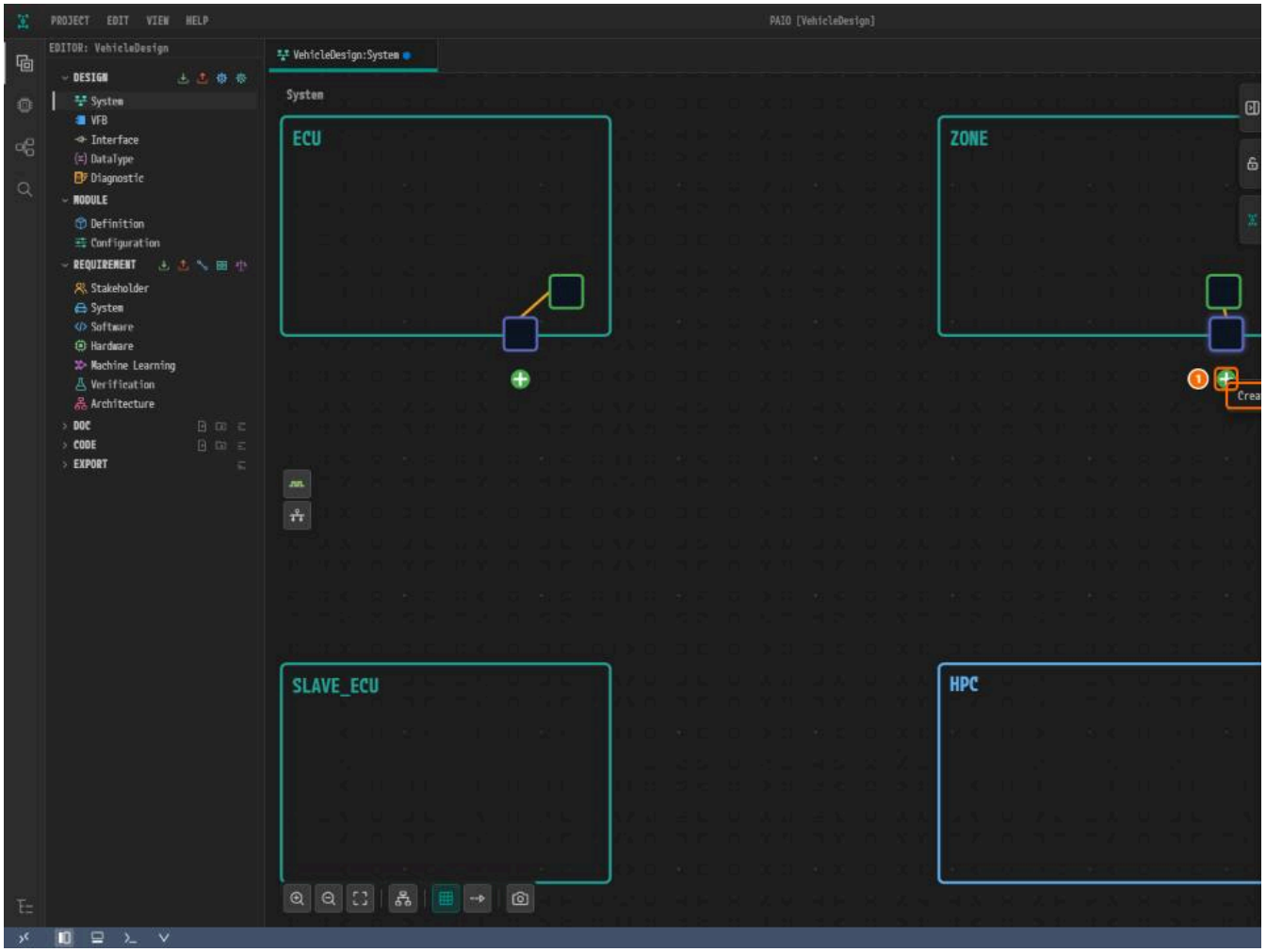


*그림 3-9. ① + 배지, ② **Create CommunicationCluster [Can]**.*

2. `ZONE` 아래에 생긴 클러스터 사각형을 클릭한다(보이지 않으면 **Fit to View** 를 클릭한다). **CAN CLUSTER** 패널이 열린다.

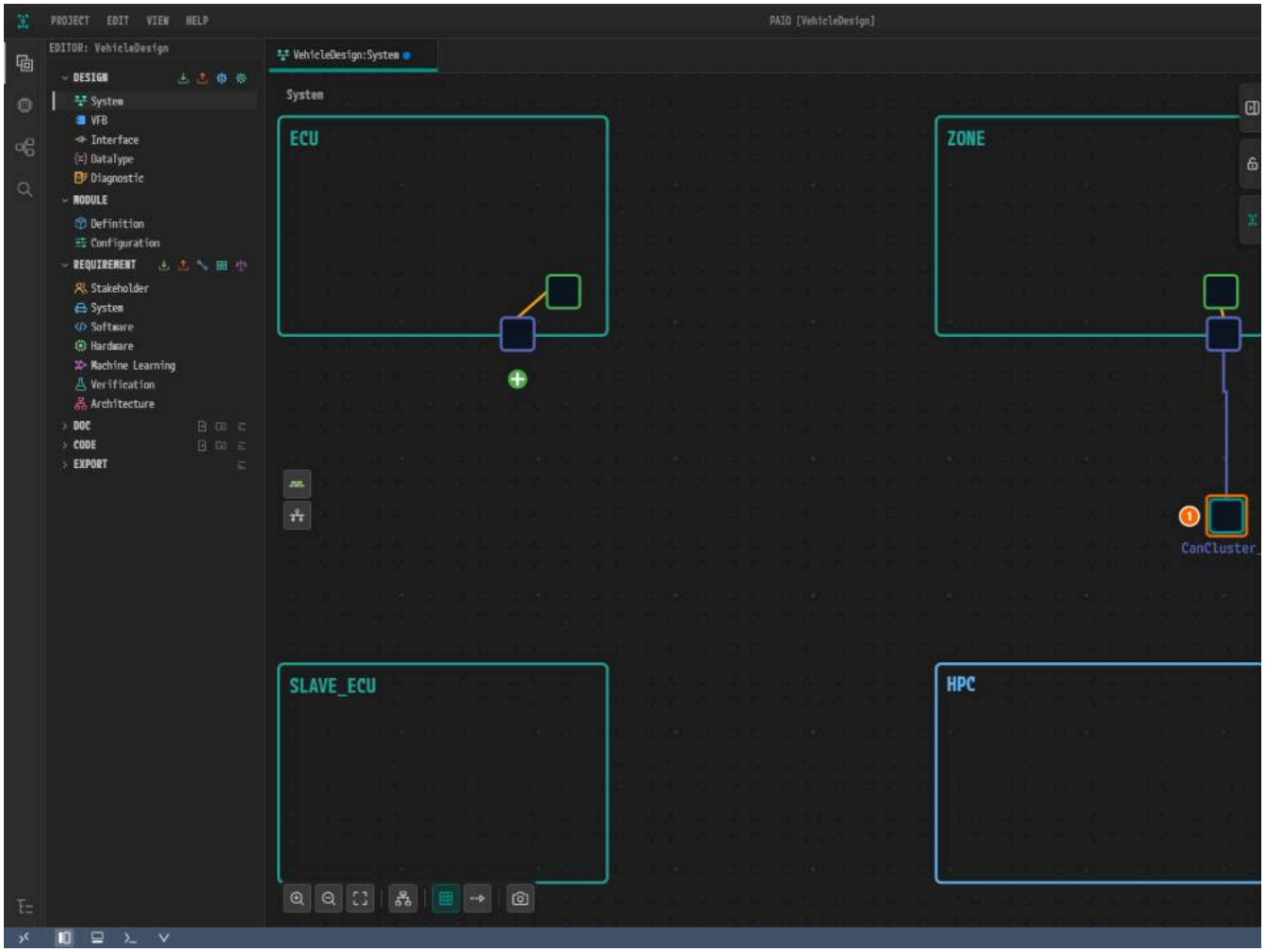


*그림 3-10. ① 클러스터, ② **NAME**, ③ **BAUDRATE**, ④ **CHANNEL NAME**.*

3. **NAME** 에 `CanCluster` 를 입력한다. **PROTOCOL** 과 **CHANNEL NAME** 은 그대로 둔다.

> **참고. BAUDRATE** 버튼은 하나도 켜져 있지 않지만 저장된 클러스터에는 `500000` 이 들어간다. ARXML 익스포트를 본다.

ECU 커넥터를 채널에 연결하기

1. `ECU` 의 `CanConnector` 아래 **+** 배지를 클릭하고 **Connect to Channel... > CanCluster > PhysicalChannel** 을 클릭한다. **Create CommunicationCluster [Can]** 은 클

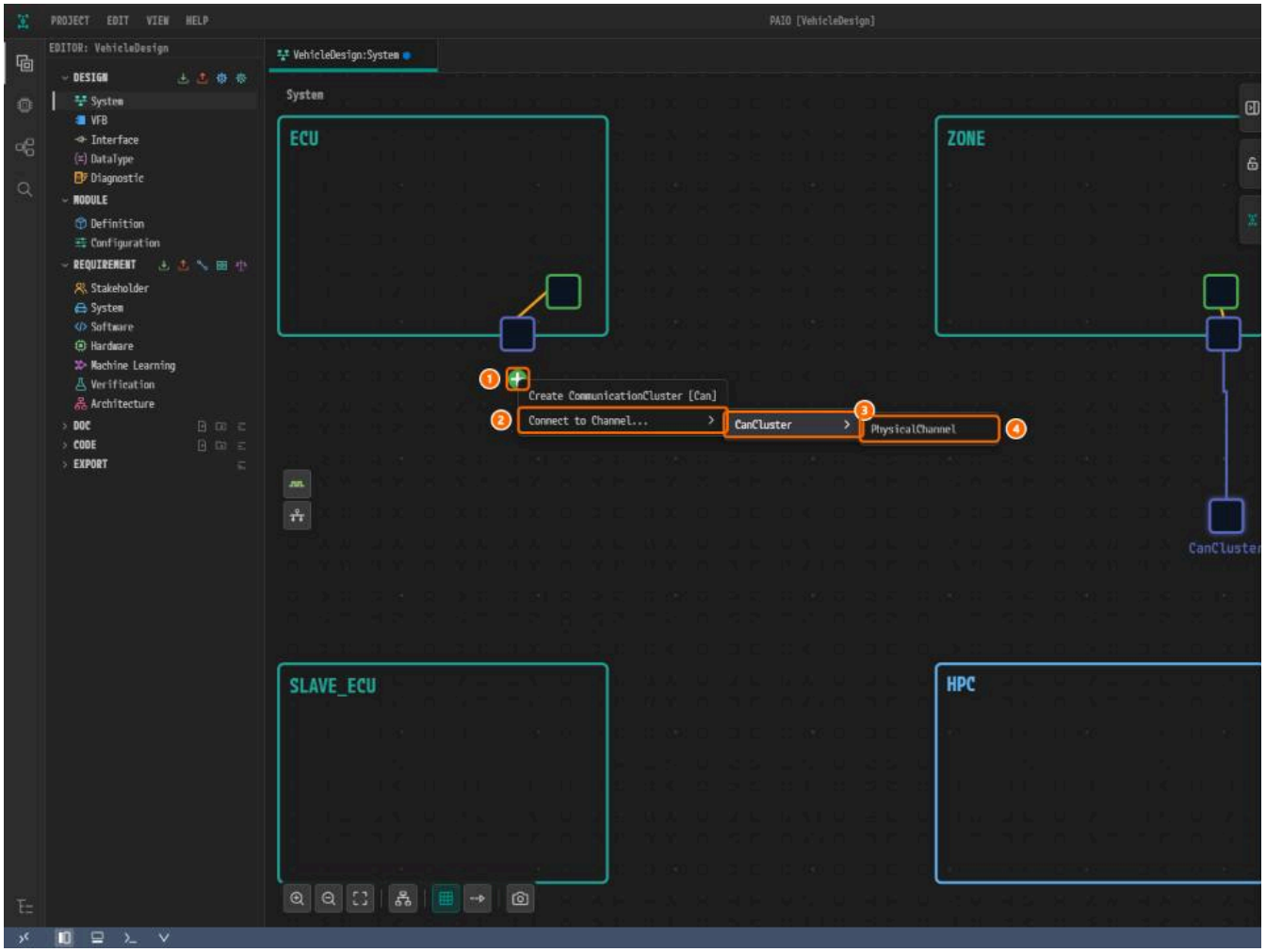


*그림 3-11. ① + 배지, ② **Connect to Channel...**, ③ **CanCluster**, ④ **PhysicalChannel**.*

배치 정리하고 저장하기

1. 캔버스 아래 툴바의 **Auto Layout** 을 클릭하고 이어서 **Fit to View** 를 클릭한다.
2. 클러스터를 클릭해 **Connected Connectors (2)** 에 `ZONE/CanConnector`, `ECU/CanConnector` 가 있는지 확인한다.

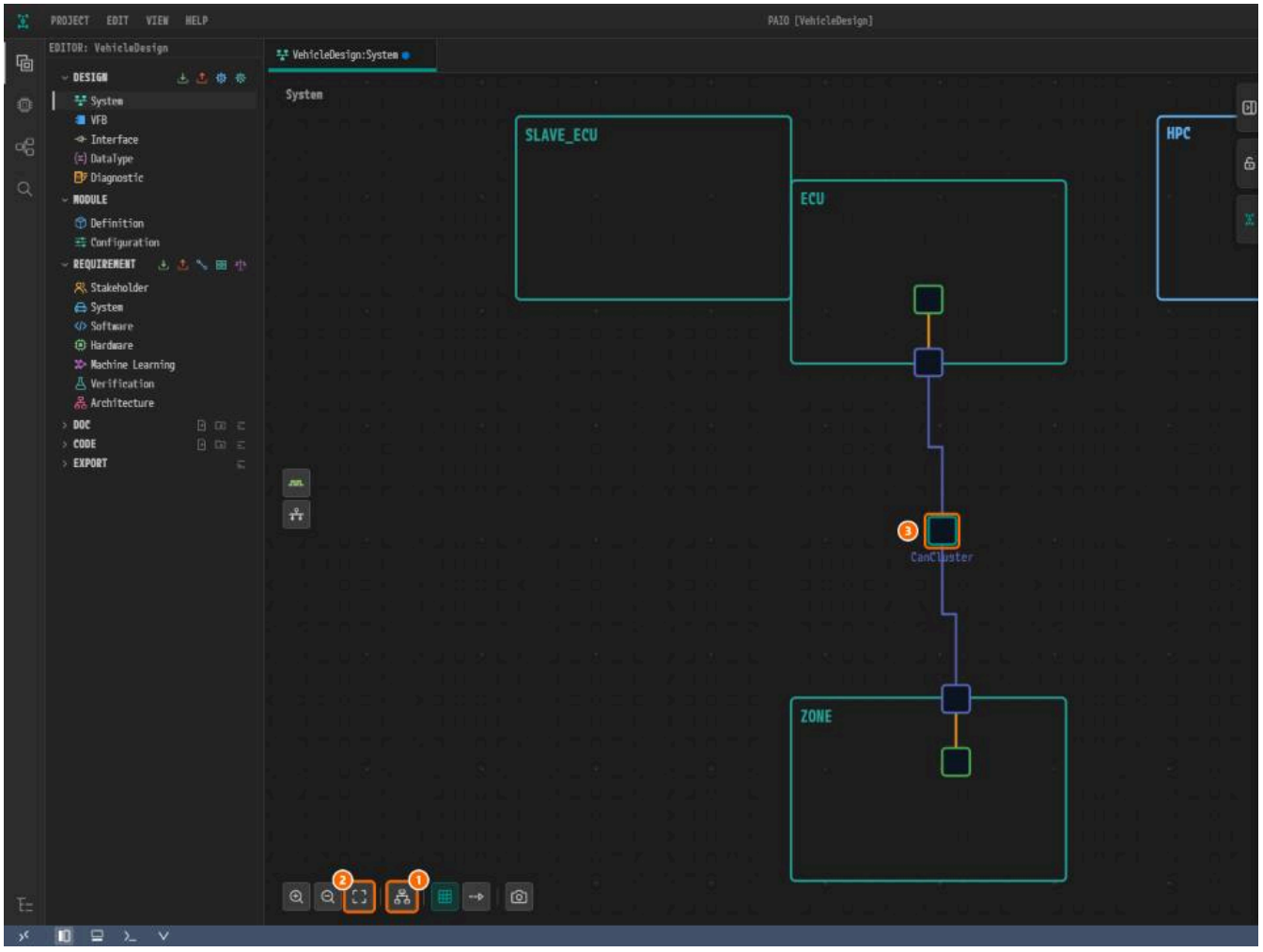


*그림 3-12. ① **Auto Layout**, ② **Fit to View**, ③ 클러스터, ④ **CHANNEL NAME** 과 두 커넥터.*

3. **BUS-OFF RECOVERY** 에서 단위를 먼저 `ms` 로 바꾼 뒤 **BOR TIME L1** `100`, **BOR TIME L2** `200`, **BOR TIME TX ENSURED** `100` 을 넣는다. **BOR COUNTER L1 TO L2** 에는 `2`

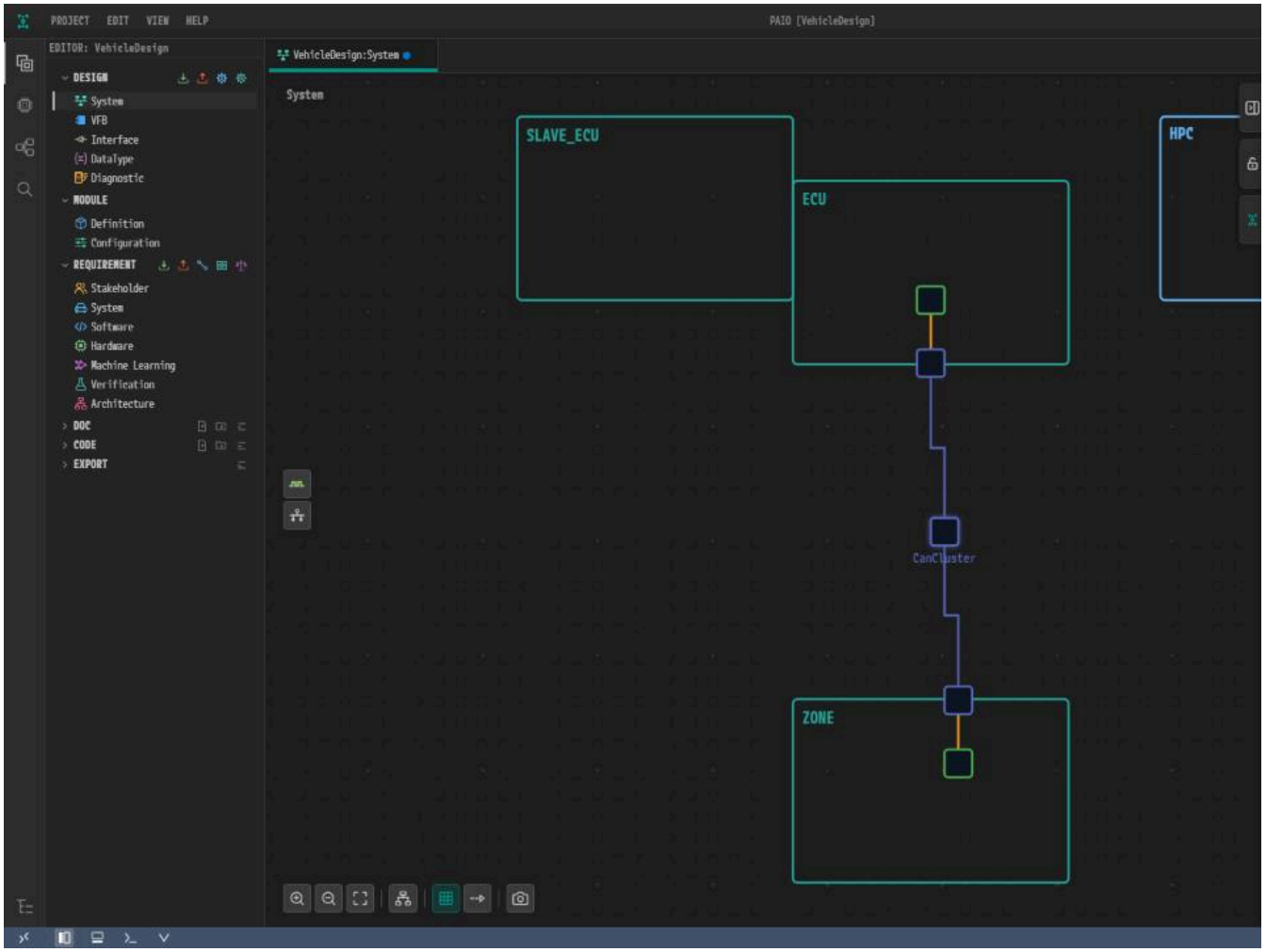


*그림 3-13. ① **BOR TIME L1**, ② **BOR TIME L2**, ③ **BOR COUNTER L1 TO L2**, ④ **BOR TIME TX ENSURED**.*

4. 빈 캔버스를 클릭하고 `Ctrl+S` 를 누른다.

검증 결과 보기

1. **V** 아이콘을 클릭해 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

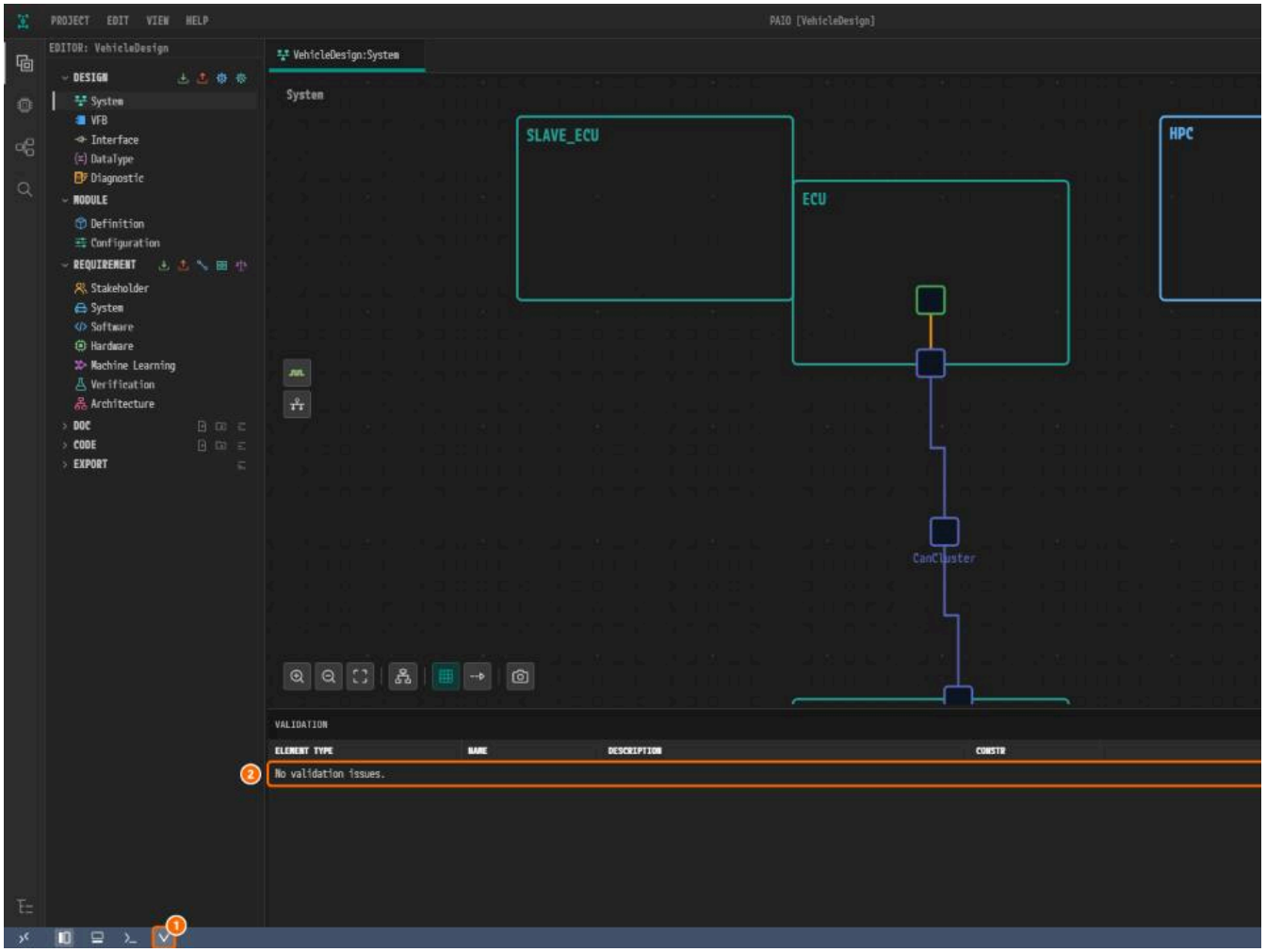


*그림 3-14. ① **V** 아이콘, ② `No validation issues.`.*

네트워크 관리 탭 확인하기

클러스터 설정 창 열기

1. **VALIDATION** 패널을 닫는다.
2. `CanCluster` 를 더블클릭한다. 창 **Communication Cluster Configuration [CanCluster]** 가 열린다.
3. **NM CLUSTER** 탭을 클릭한다.

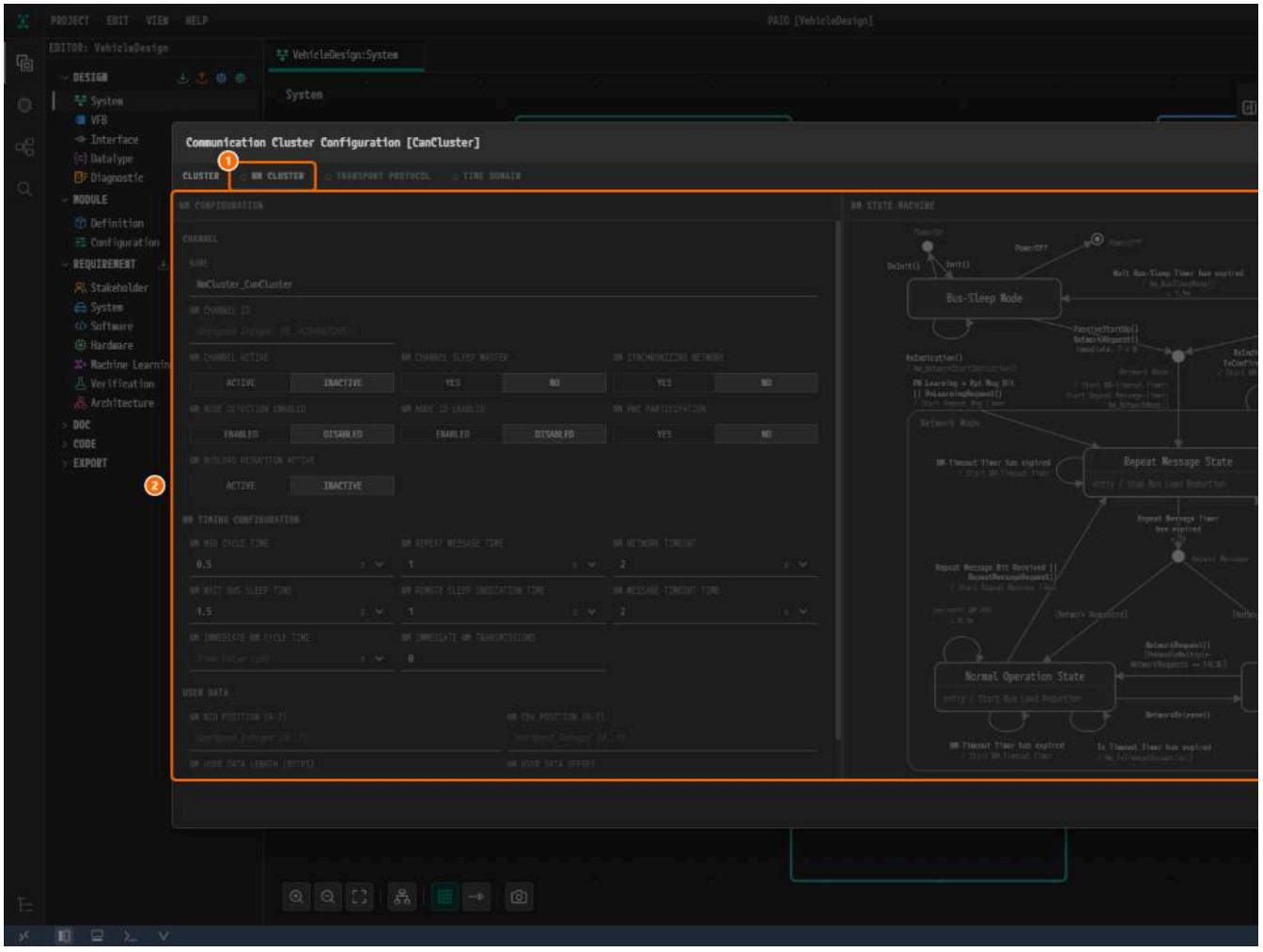


*그림 3-15. ① 빈 동그라미가 있는 **NM CLUSTER**, ② 흐린 편집 영역.*

빈 동그라미는 그 기능이 꺼져 있다는 뜻이며, **TRANSPORT PROTOCOL** 과 **TIME DOMAIN** 에도 있다. 흐린 영역은 미리보기일 뿐 저장되지 않는다.

> **주의.** 탭을 오른쪽 클릭해 **Enable** 을 고르지 않는다. 참조 설계에는 NM 이 없다.

전송 프로토콜 설정 확인하기

1. **TRANSPORT PROTOCOL** 탭을 클릭한다. 이 탭에도 빈 동그라미가 있다.

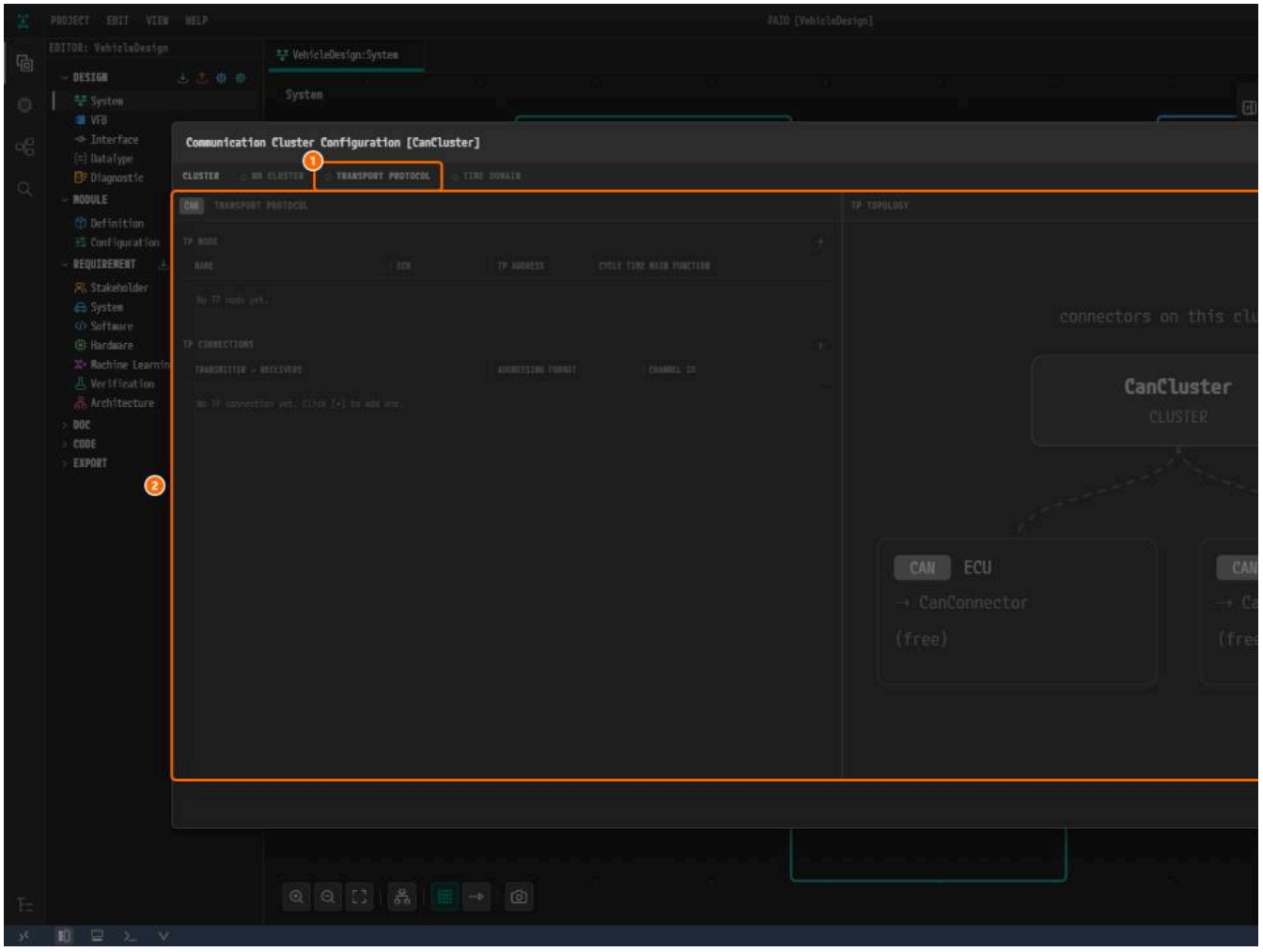


*그림 3-16. ① 빈 동그라미가 있는 **TRANSPORT PROTOCOL**, ② 흐린 편집 영역.*

2. **Close** 를 클릭한다.

다시 검증하기

1. **VALIDATION** 패널을 연다. 여전히 `No validation issues.` 가 보이고, 탭 `VehicleDesign:System` 에 수정됨 표시가 없다.

왜 이렇게 설계하는가

- **커넥터 순서.** 참조 설계는 `ZONE`, `ECU` 순서이므로 클러스터를 `ZONE` 에서 만든다.
- **이름.** 참조 설계대로 컨트롤러는 `CanController`, `CanCommController` 이고, 두 커넥터는 모두 `CanConnector` 이다.
- **비트 타이밍은 `ECU` 에만.** `CanController` 에는 `0` 네 개가 있고, `CanCommController` 에는 없다.
- **버스 오프 복구 값.** 참조 설계에는 없지만 PRTE 가 `ECU` 의 코드를 만들 때 CanSM 에 이 값이 필요하다. 값은 PRTE 예제가 쓰는 값이다.
- **NM 과 TP 는 꺼 둔다.** 참조 설계에 둘 다 없고, 새 클러스터에서는 둘 다 꺼져 있다.

확인하기

- [ ] `ECU` 에 `CanController`(비트 타이밍 `0`, **CAN**)와 `CanConnector`(**CONTROLLER** = `CanController`)가 있다.
- [ ] `ZONE` 에 `CanCommController` 와 `CanConnector`(**CONTROLLER** = `CanCommController`)가 있다.
- [ ] `CanCluster` 의 `PhysicalChannel` 에 `ZONE/CanConnector`, `ECU/CanConnector` 가 이 순서로 있다.
- [ ] **BUS-OFF RECOVERY** 가 `100` `ms`, `200` `ms`, `2`, `100` `ms` 이다.
- [ ] **NM CLUSTER**, **TRANSPORT PROTOCOL**, **TIME DOMAIN** 에 빈 동그라미가 있다.
- [ ] 수정됨 표시가 없고 `No validation issues.` 가 보인다.

다음 단계

ETHERNET 설정에서 `ZONE` 과 `HPC` 를 이더넷 클러스터 `EthernetCluster` 로 잇는다.

ETHERNET 설정

이 장에서 할 일

노드 `ZONE` 과 `HPC` 를 이더넷 버스로 잇는다. 두 노드에 이더넷 컨트롤러와 이더넷 커넥터를 하나씩 만들고, 채널 `PhysicalChannel` 의 VLAN ID 가 `13` 인 클러스터 `EthernetC`

시작하기 전에

- CAN 설정을 마친 상태에서 시작한다. **VALIDATION** 패널에는 `No validation issues.` 가 보인다.
- 탭 `VehicleDesign:System` 이 열려 있지 않으면 **EDITOR** 뷰의 **DESIGN** 아래에서 **System** 을 클릭한다. **VALIDATION** 패널은 푸터의 **V** 아이콘으로 닫는다.
- 노드 `HPC` 가 속성 패널에 가려지면 캔버스 아래 툴바의 **Fit to View** 를 클릭한 뒤 네 노드가 모두 보일 때까지 **Zoom Out** 을 클릭한다.
- 속성 패널의 입력란은 칸에서 빠져나와야 값이 확정된다. 값을 넣을 때마다 `Tab` 키를 누른다.

어디서	항목	값
ETHERNET CONTROLLER 패널 (노드 `ZONE`)	**NAME** / **MAC ADDRESS**	`EthernetCommController` / `11:22:EE:44:55:66`
ETHERNET CONTROLLER 패널 (노드 `HPC`)	**NAME** / **MAC ADDRESS**	`EthernetController` / `00:11:22:33:44:55`
ETHERNET CONNECTOR 패널 (노드 `ZONE`)	**NAME** / **CONTROLLER**	`EthernetConnector` / `EthernetCommController` 선택
ETHERNET CONNECTOR 패널 (노드 `ZONE`)	**UNICAST IPV4 ADDRESS** / **ADDRESS SOURCE**	`192.168.0.19` / **FIXED** 클릭
ETHERNET CONNECTOR 패널 (노드 `HPC`)	**NAME** / **CONTROLLER**	`EthernetConnector` / `EthernetController` 선택
ETHERNET CONNECTOR 패널 (노드 `HPC`)	**UNICAST IPV4 ADDRESS** / **ADDRESS SOURCE**	`192.168.0.11` / **FIXED** 클릭
ETHERNET CONNECTOR 패널 (노드 `HPC`)	**MULTICAST IPV4 ADDRESS**	`224.0.0.1`
ETHERNET CLUSTER 패널	**NAME** / **VLAN ID**	`EthernetCluster` / `13`
ETHERNET CLUSTER 패널	**PHYSICAL CHANNELS** > **CHANNEL NAME**	`PhysicalChannel` (기본값, 그대로 둔다)
Communication Cluster Configuration 창	**NM CLUSTER**, **TIME DOMAIN** 탭	꺼짐 (기본값, 그대로 둔다)

이더넷 컨트롤러 만들기

1. 노드 `ZONE` 안쪽의 빈 곳을 오른쪽 클릭하고 **Add Controller** > **Ethernet** 을 클릭한다.

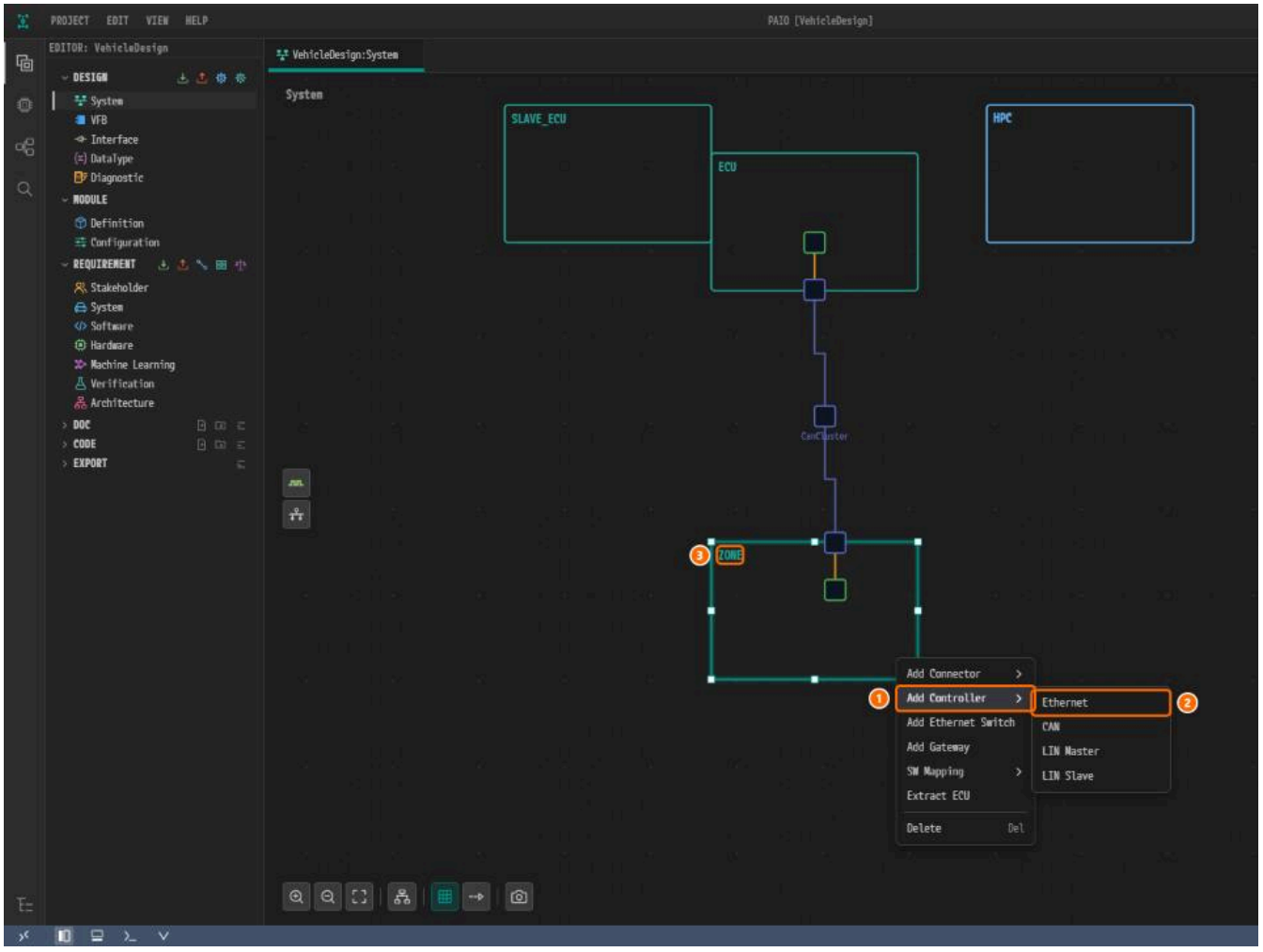


*그림 4-1. ① **Add Controller**, ② **Ethernet**, ③ 노드 `ZONE`.*

2. 노드 안에 작은 사각형, 곧 컨트롤러가 생긴다. 이 사각형을 클릭하면 **ETHERNET CONTROLLER** 패널이 열린다.

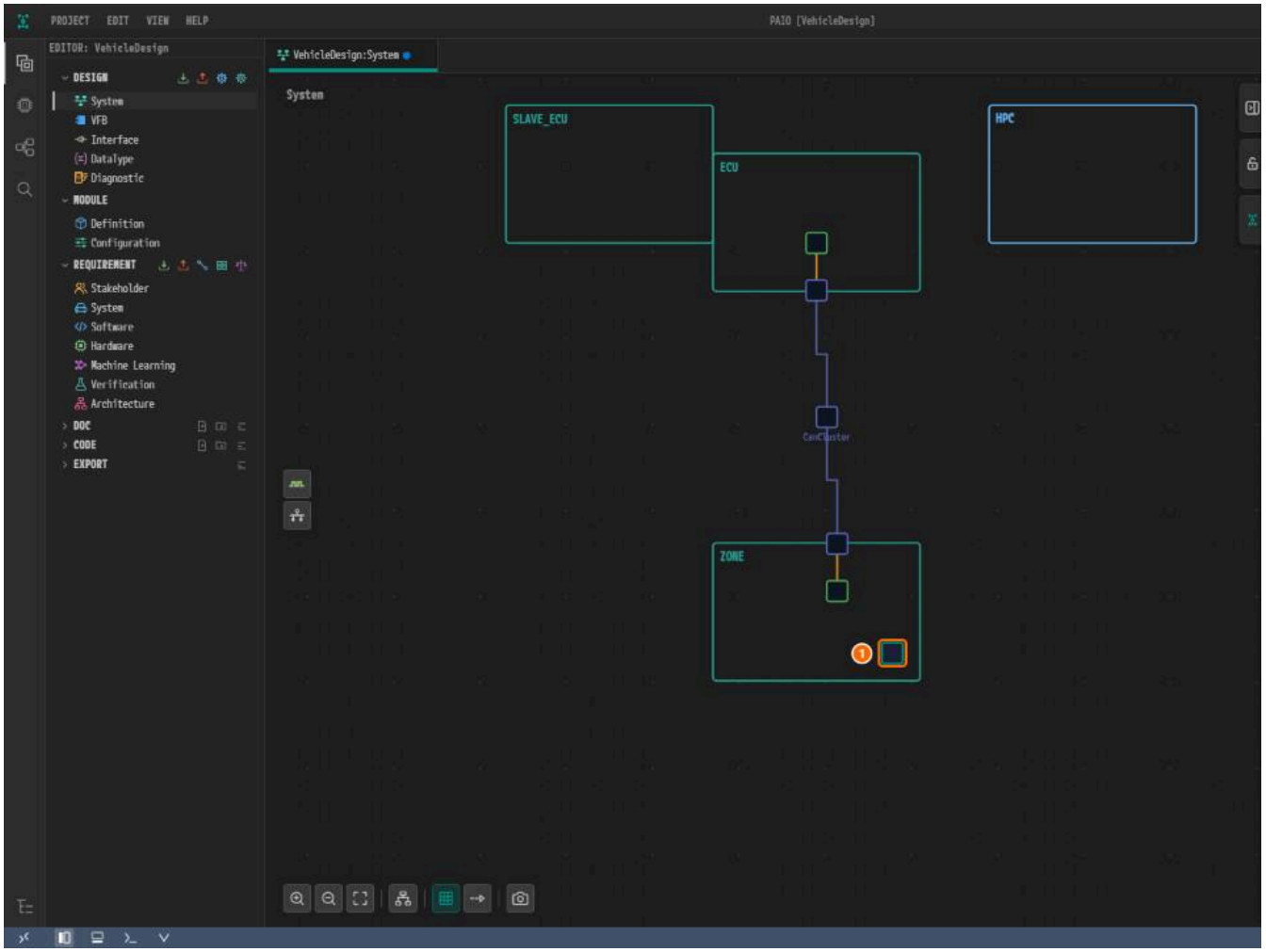


*그림 4-2. ① 컨트롤러, ② **ETHERNET CONTROLLER**, ③ **NAME**, ④ **MAC ADDRESS**, ⑤ **MAC LAYER TYPE**.*

3. **NAME** 에 `EthernetCommController`, **MAC ADDRESS** 에 `11:22:EE:44:55:66` 을 입력한다. 주소는 소문자로 보이지만 같은 주소이다. **MAC LAYER TYPE** 은 누르지 않

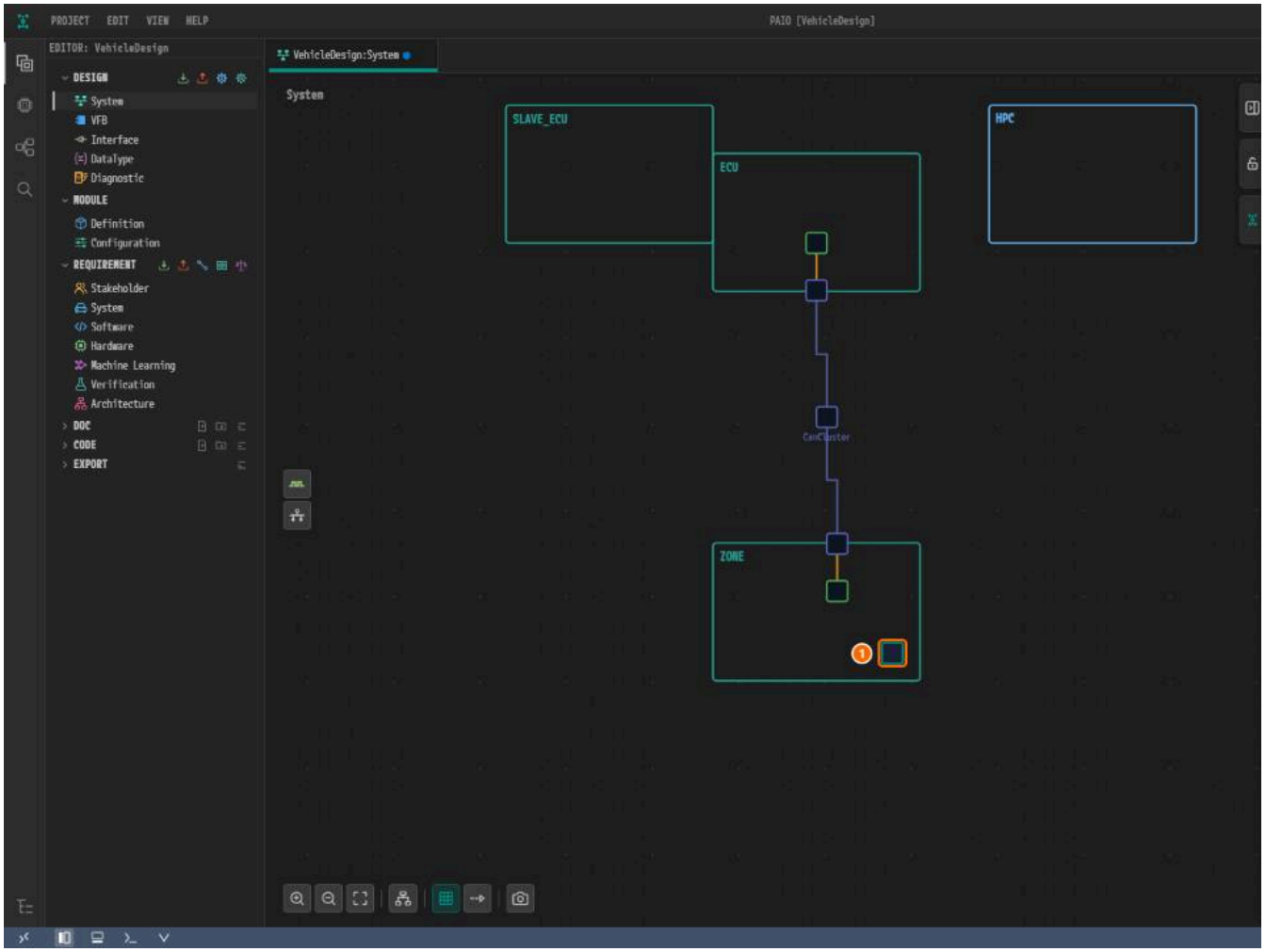


*그림 4-3. ① 컨트롤러, ② **NAME**, ③ **MAC ADDRESS**, ④ 누르지 않고 둔 **MAC LAYER TYPE**.*

> **주의.** 토글 값 `XMII`, `XGMII`, `XXGMII` 는 AUTOSAR 스키마의 값(`X-MII`, `XG-MII`, `XXG-MII`)이 아니어서, 하나라도 누르면 스키마가 거부하는 값이 익스포트된다. AR›

4. 노드 `HPC` 안쪽의 빈 곳을 오른쪽 클릭하고 **Add Controller > Ethernet** 을 클릭한다.

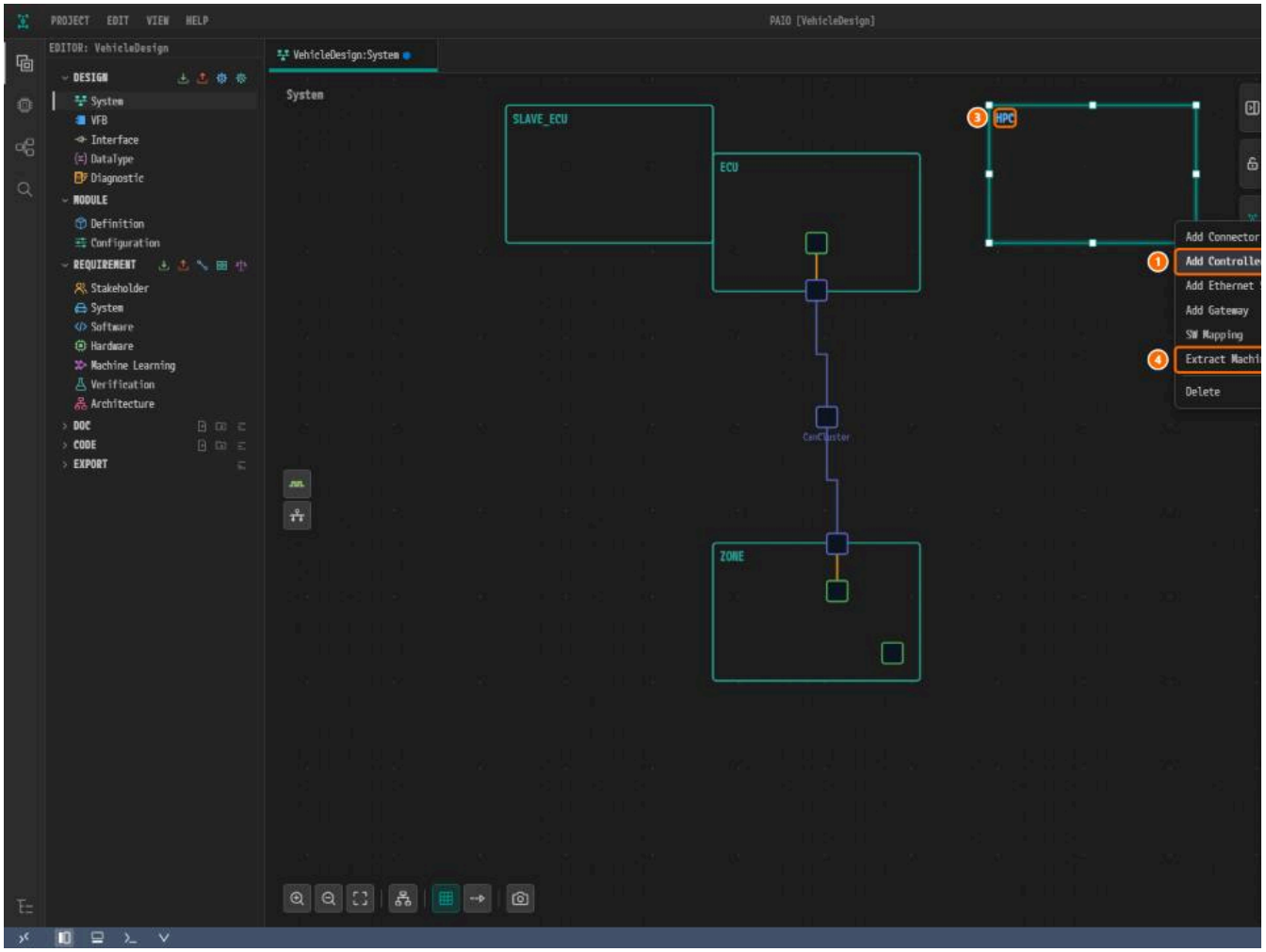


*그림 4-4. ① **Add Controller**, ② **Ethernet**, ③ 노드 `HPC`, ④ **Extract Machine**.*

5. `HPC` 안에 생긴 컨트롤러를 클릭한다. **NAME** 에 `EthernetController`, **MAC ADDRESS** 에 `00:11:22:33:44:55` 를 입력한다. **MAC LAYER TYPE** 은 이번에도 누르지 않

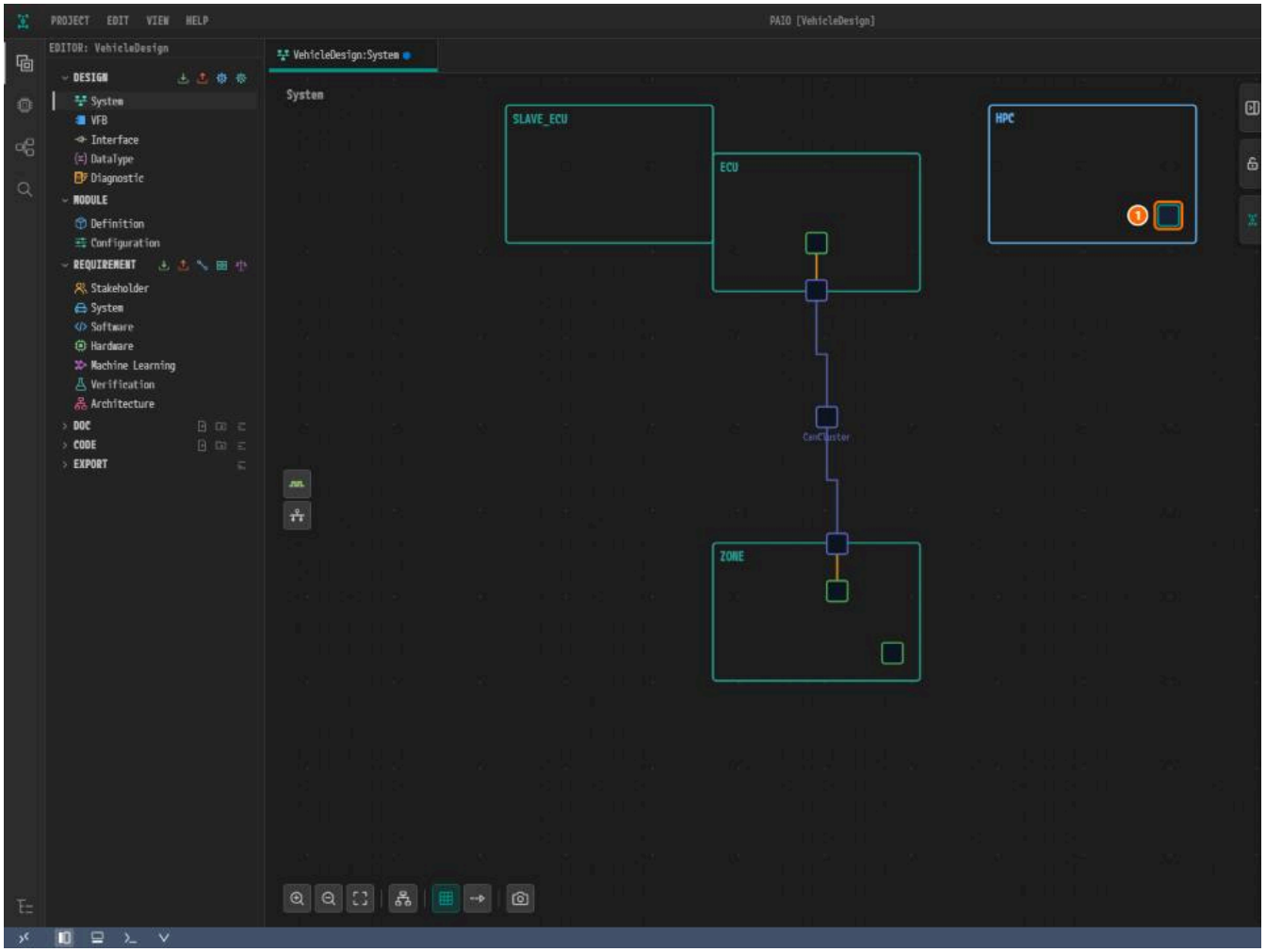


*그림 4-5. ① `HPC` 의 컨트롤러, ② **NAME**, ③ **MAC ADDRESS**.*

이더넷 커넥터 만들기

1. `ZONE` 안쪽의 빈 곳을 오른쪽 클릭하고 **Add Connector > Ethernet** 을 클릭한다. 노드 아래 가장자리에 커넥터가 생기고 초록 **+** 배지가 붙는다. 커넥터를 클릭한다. 컨트롤러 `is required`) 보인다.

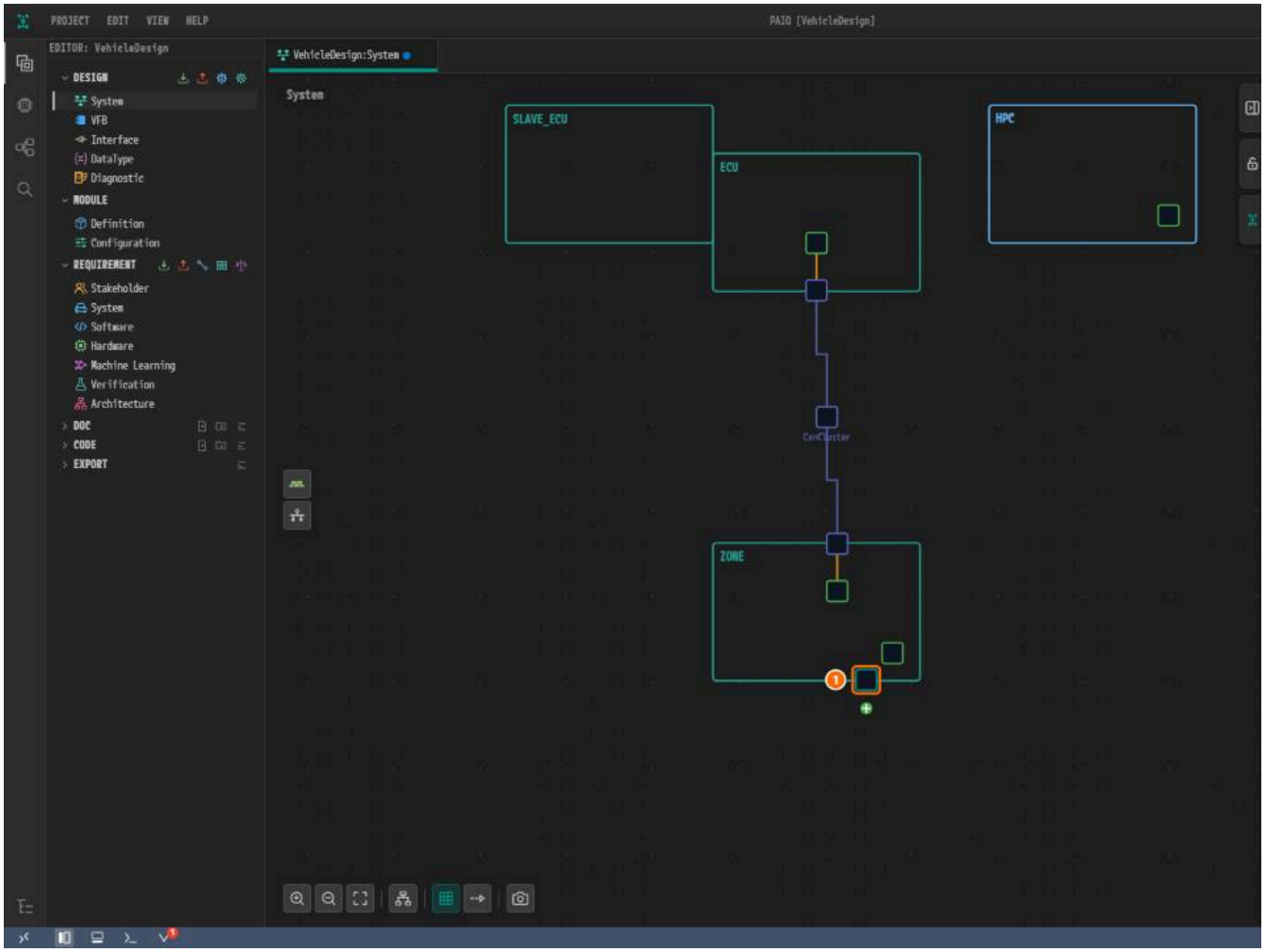


*그림 4-6. ① 커넥터, ② **ETHERNET CONNECTOR**, ③ **NAME**, ④ **CONTROLLER**, ⑤ **NETWORK ENDPOINT**.*

2. **NAME** 에 `EthernetConnector` 를 입력하고, **CONTROLLER** 목록에서 `EthernetCommController` 를 고르고, **UNICAST IPV4 ADDRESS** 에 `192.168.0.19` 를 입력한 컨트롤러를 잇는 주황 선이 생긴다.

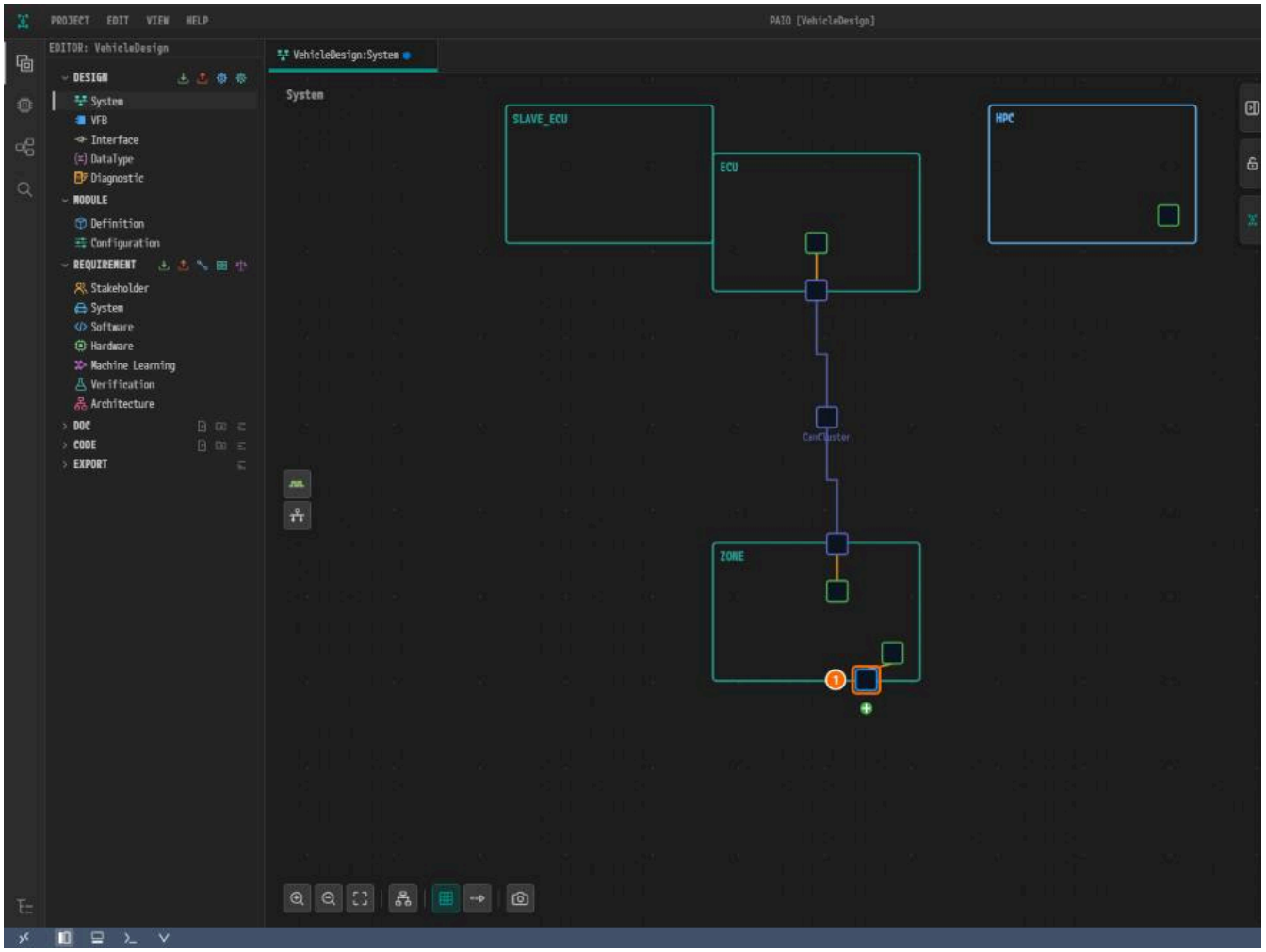


*그림 4-7. ① 커넥터, ② **NAME**, ③ **CONTROLLER**, ④ **UNICAST IPV4 ADDRESS**, ⑤ **FIXED**.*

> **주의.** **ADDRESS SOURCE** 는 처음에 켜진 값이 없다. **FIXED** 를 누르지 않으면 주소 방식이 빈 채로 저장된다.

3. `HPC` 의 커넥터도 같은 방법으로 만든다. **NAME** `EthernetConnector`, **CONTROLLER** `EthernetController`, **UNICAST IPV4 ADDRESS** `192.168.0.11`, **FIXED** 이다 **ADDRESS** 에 `224.0.0.1` 을 입력한다. **MULTICAST PORT** 는 비워 둔다.

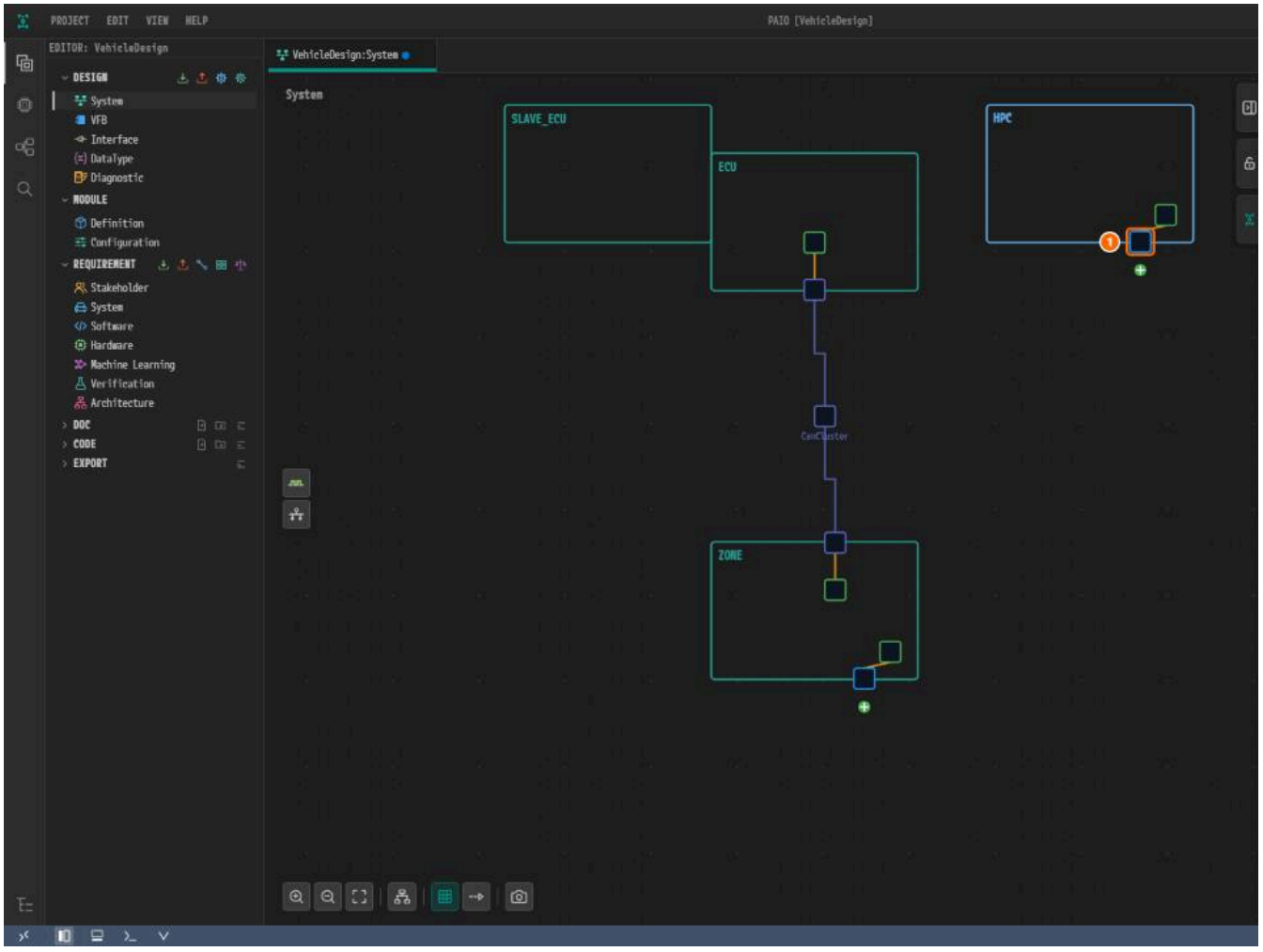


*그림 4-8. ① `HPC` 의 커넥터, ② **NAME**, ③ **CONTROLLER**, ④ **UNICAST IPV4 ADDRESS**, ⑤ **FIXED**, ⑥ **MULTICAST IPV4 ADDRESS**.*

이더넷 클러스터 만들기

1. `ZONE` 커넥터 아래의 초록 **+** 배지를 클릭하고 **Create CommunicationCluster [Ethernet]** 을 클릭한다.

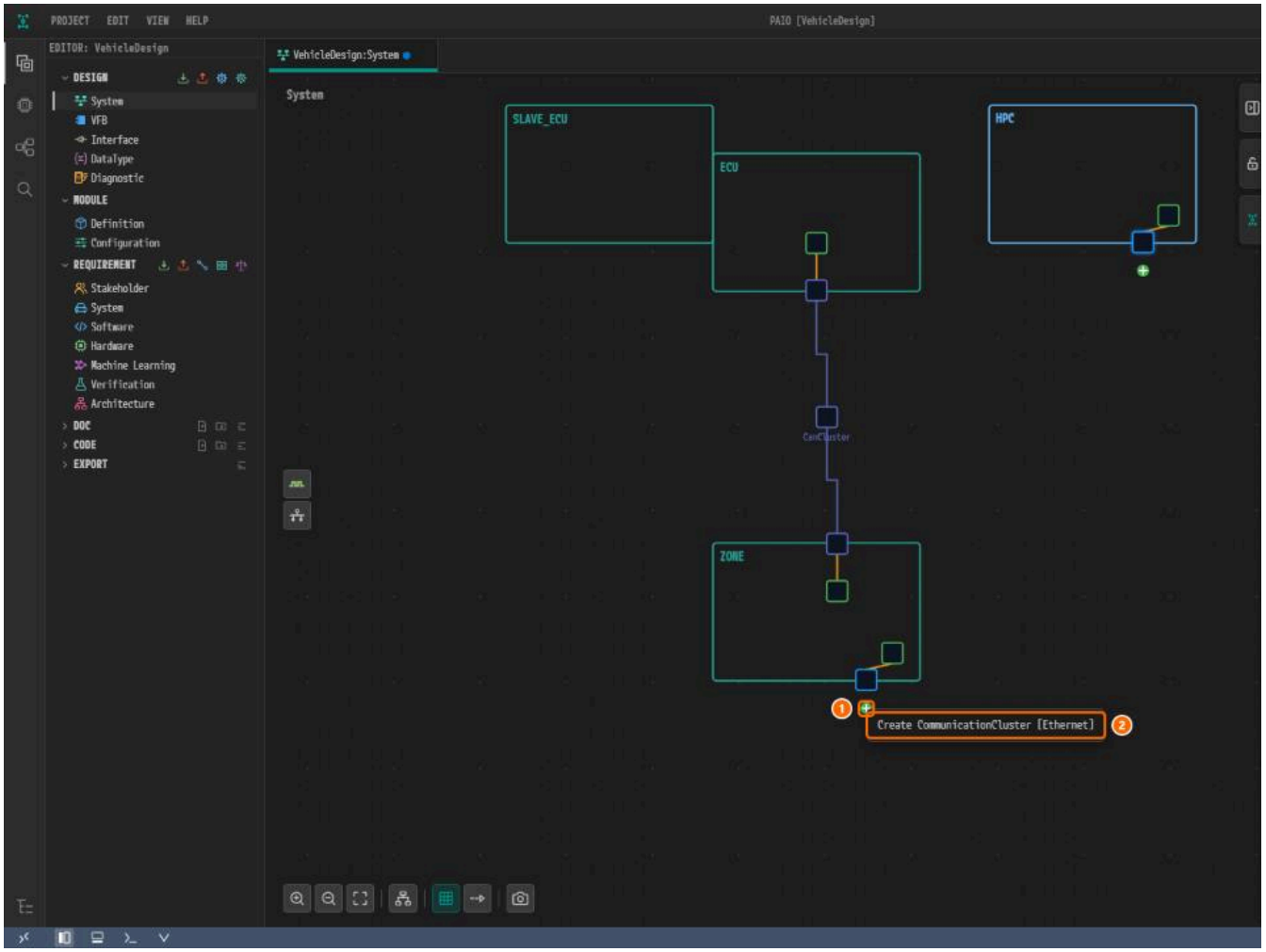


*그림 4-9. ① + 배지, ② **Create CommunicationCluster [Ethernet]**.*

2. `ZONE` 아래에 생긴 클러스터 사각형을 클릭한다. 보이지 않으면 **Fit to View** 를 클릭한다.

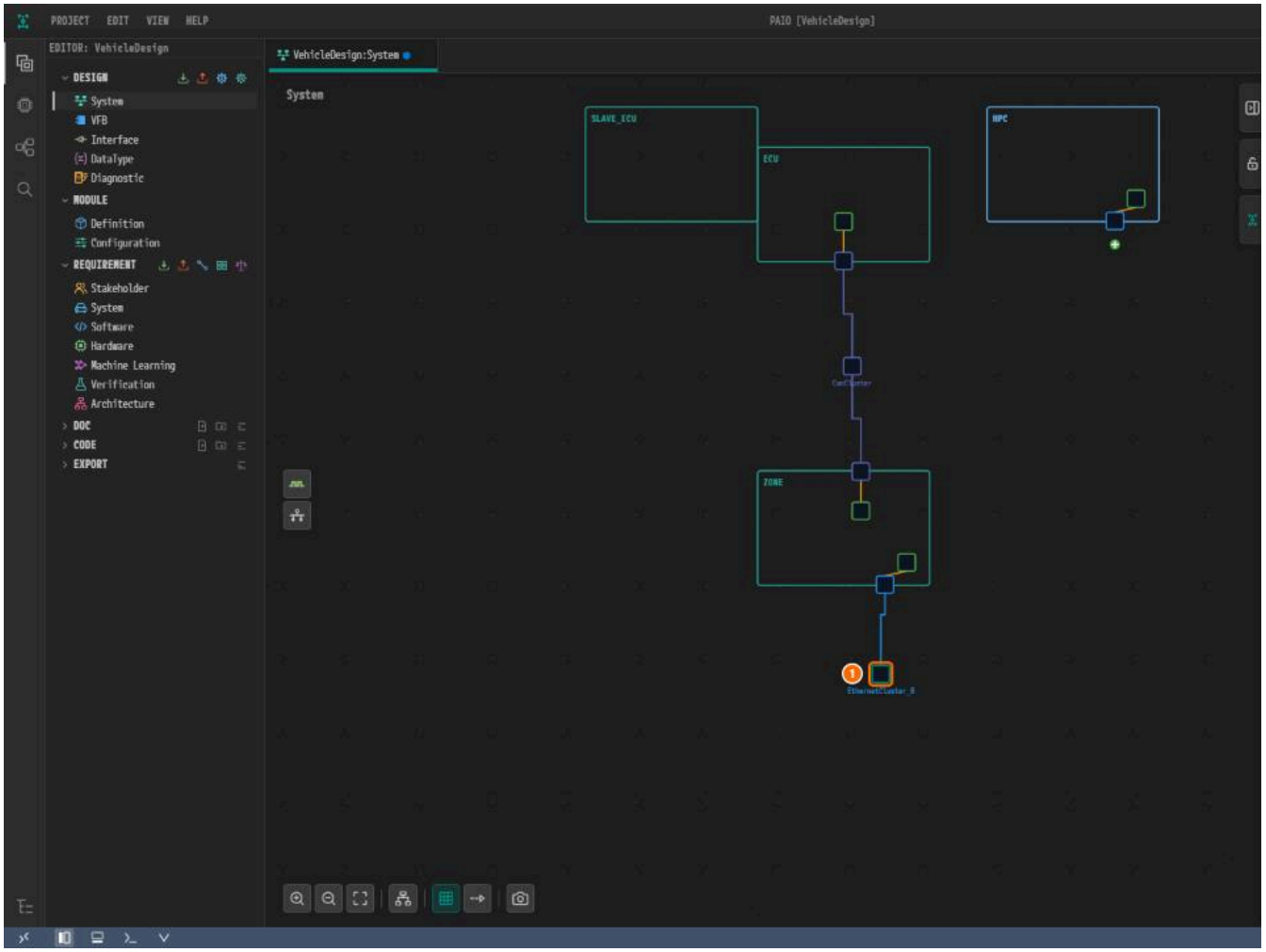


*그림 4-10. ① 클러스터, ② **NAME**, ③ **CHANNEL NAME**(`PhysicalChannel`), ④ **VLAN ID**.*

3. **NAME** 에 `EthernetCluster`, **VLAN ID** 에 `13` 을 입력한다. **CHANNEL NAME** 은 그대로 둔다.

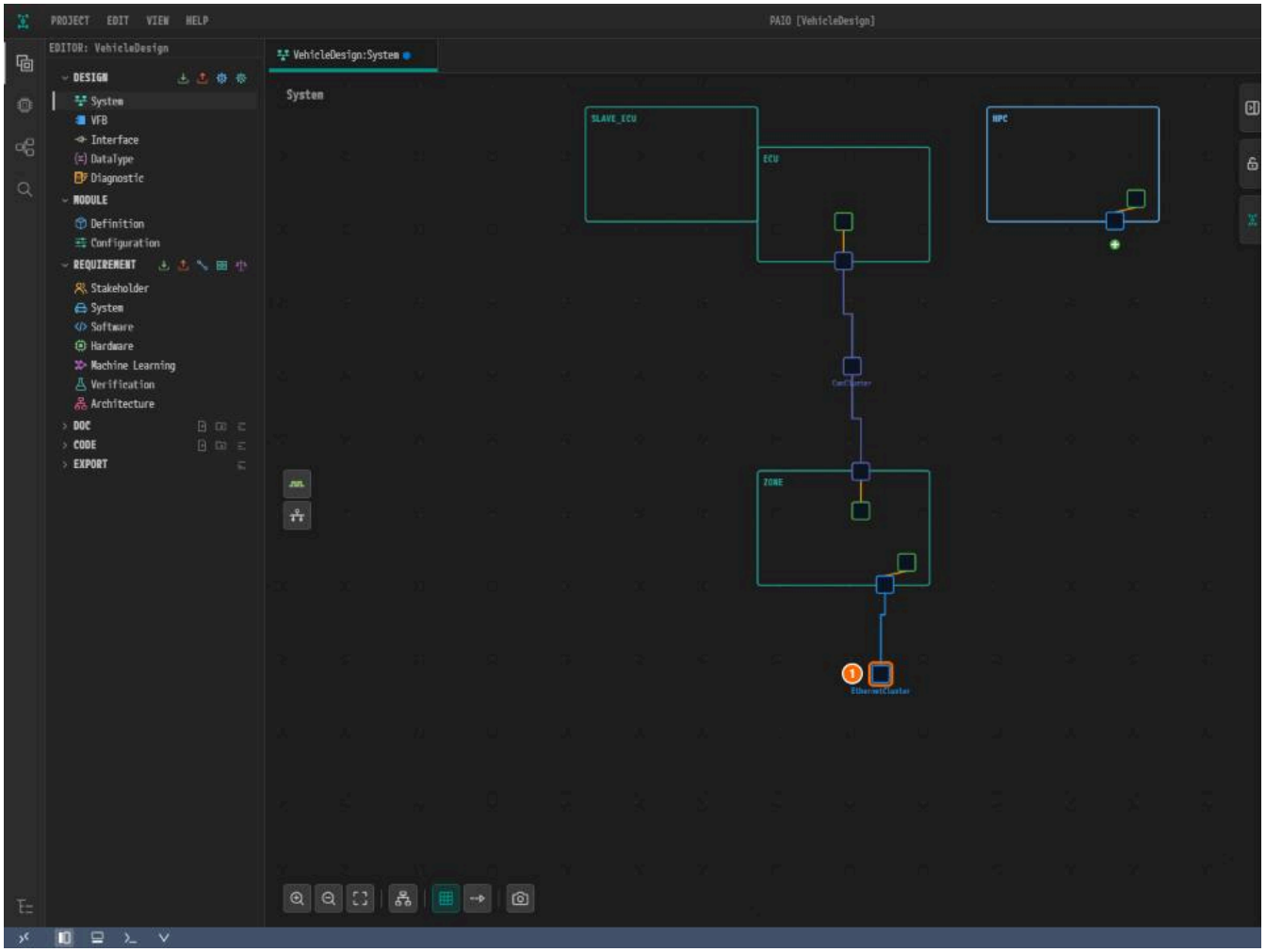


*그림 4-11. ① 클러스터, ② **NAME**, ③ **VLAN ID**.*

4. `HPC` 커넥터 아래의 **+** 배지를 클릭하고 **Connect to Channel... > EthernetCluster > PhysicalChannel (VLAN: 13)** 을 클릭한다. 배지를 누르기 어려우면 먼저 화면을 확대

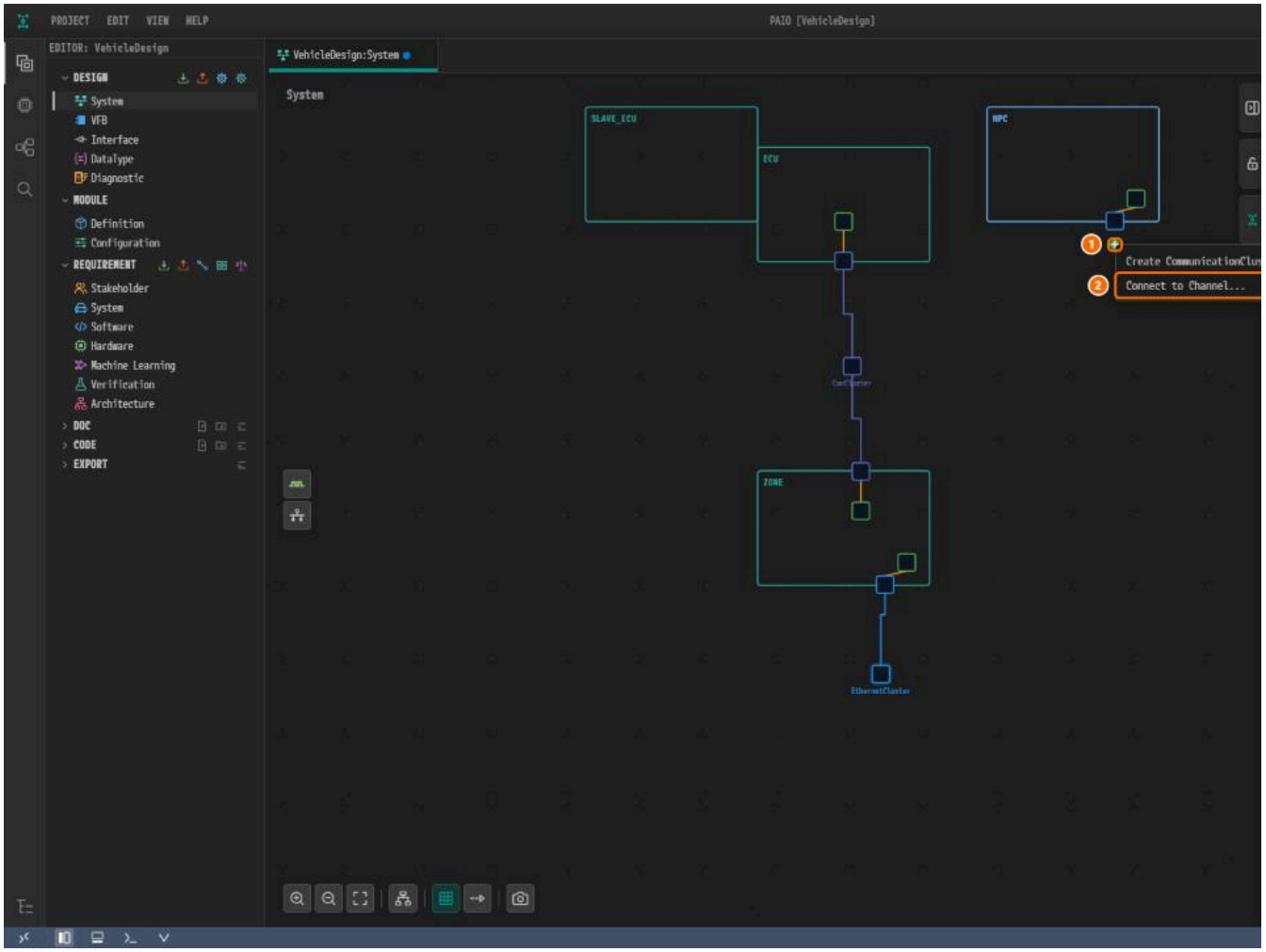


*그림 4-12. ① + 배지, ② **Connect to Channel...**, ③ **EthernetCluster**, ④ **PhysicalChannel (VLAN: 13)**.*

> **주의.** 여기서 **Create CommunicationCluster [Ethernet]** 을 누르지 않는다. 클러스터가 하나 더 생긴다.

5. **Auto Layout** 을 클릭하고 이어서 **Fit to View** 를 클릭한다. 클러스터를 클릭해 **Connected Connectors (2)** 에 `ZONE/EthernetConnector` 와 `HPC/EthernetConne`

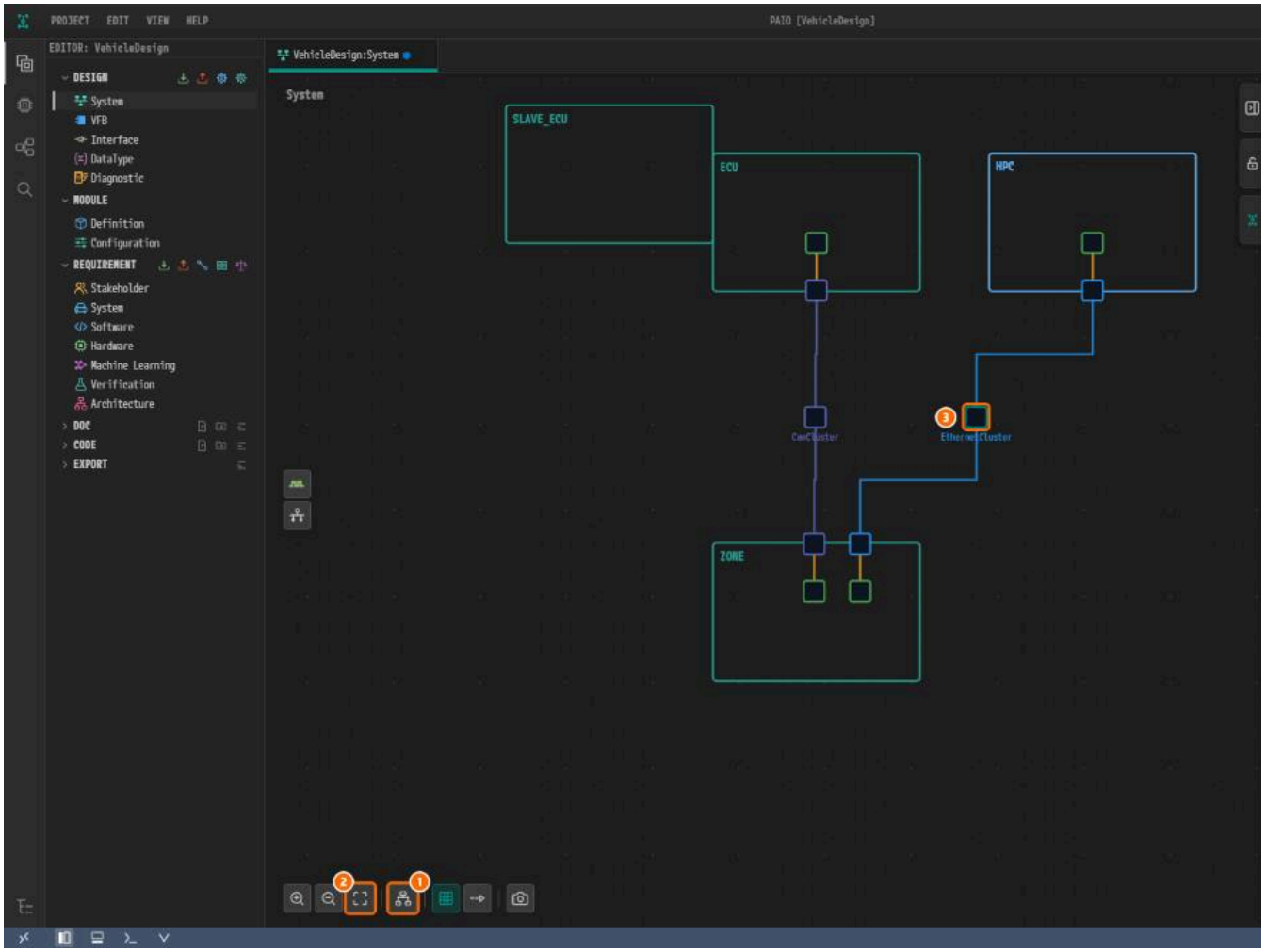


*그림 4-13. ① **Auto Layout**, ② **Fit to View**, ③ 클러스터, ④ 채널과 연결된 커넥터. 배치는 그림과 조금 다를 수 있다.*

6. 캔버스의 빈 곳을 클릭하고 `Ctrl+S` 를 누른다.

검증과 네트워크 관리 확인하기

1. 푸터의 **V** 아이콘을 클릭한다. **VALIDATION** 패널에 `No validation issues.` 가 보인다.

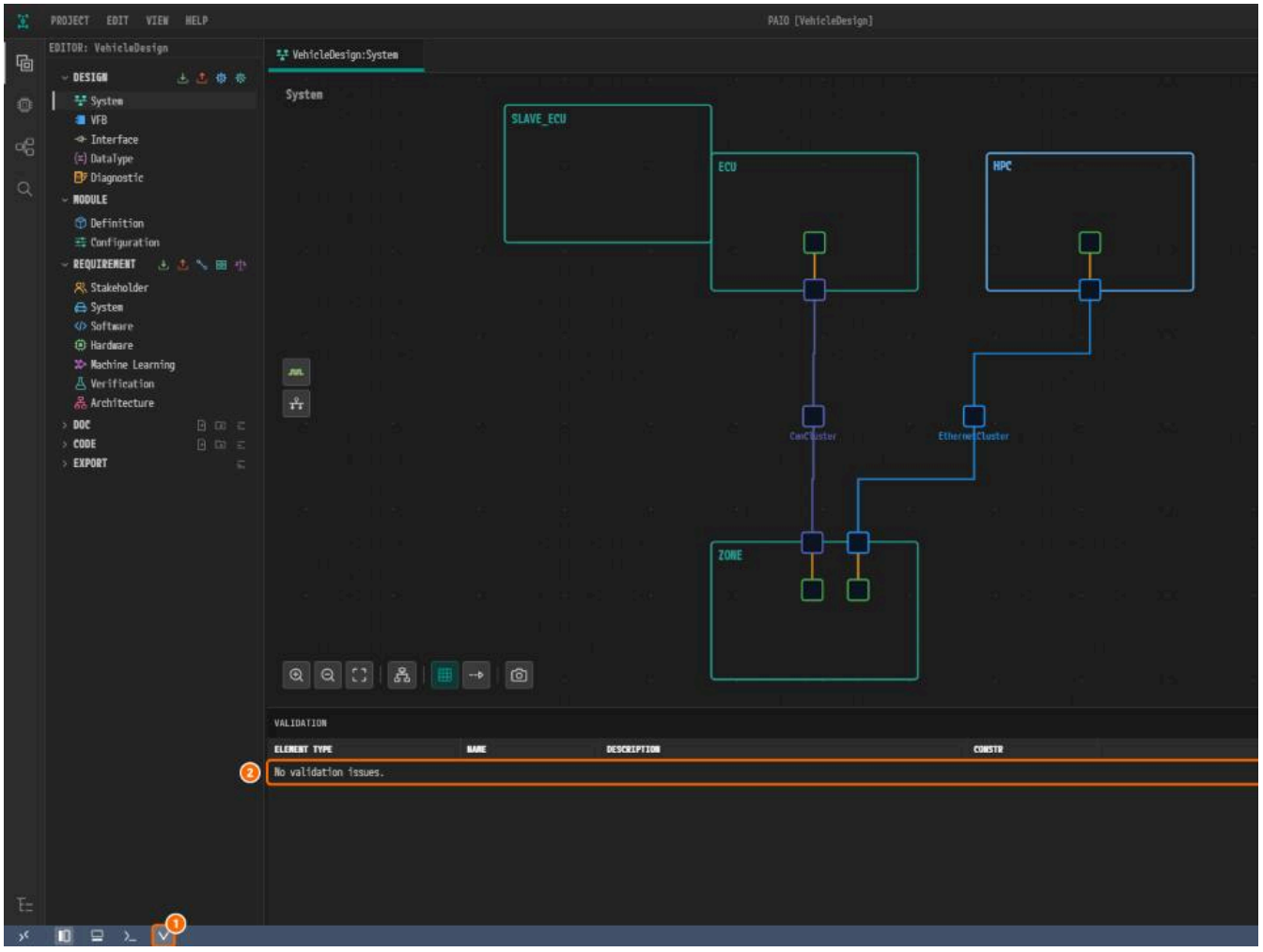


*그림 4-14. ① **V** 아이콘, ② `No validation issues.`.*

2. 패널을 닫는다. `EthernetCluster` 사각형을 더블클릭해 **Communication Cluster Configuration [EthernetCluster]** 를 열고 **NM CLUSTER** 탭을 클릭한다.

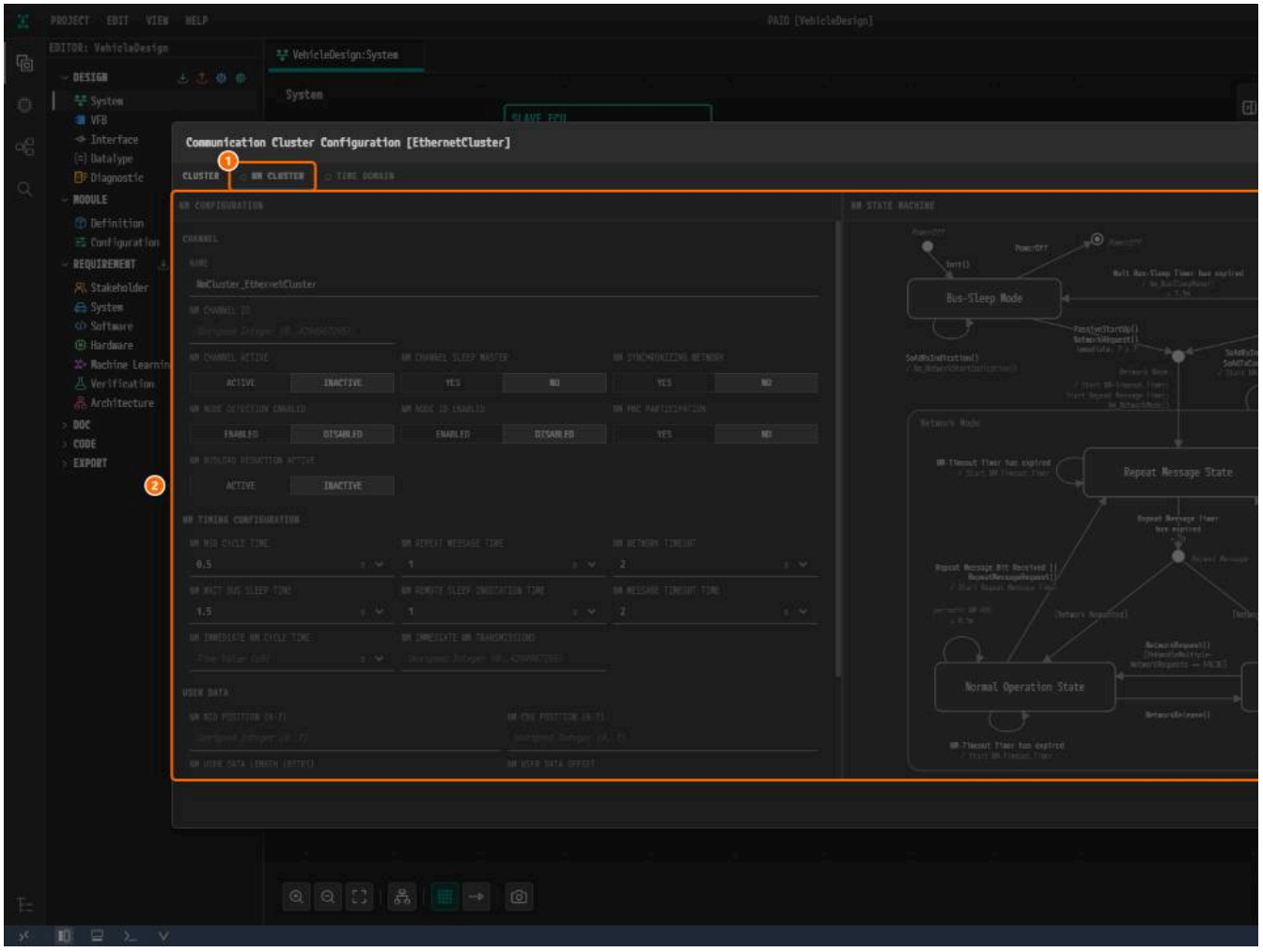


*그림 4-15. ① 빈 동그라미(꺼짐)가 있는 **NM CLUSTER**, ② 저장되지 않는 흐린 미리보기.*

> **주의.** 탭의 오른쪽 클릭 메뉴에서 **Enable** 을 고르지 않는다. 참조 설계에는 NM 이 없다.

3. **Close** 를 클릭한다. **VALIDATION** 패널에는 여전히 `No validation issues.` 가 보인다.

왜 이렇게 설계하는가

- **클러스터는 `ZONE` 에서 만든다.** 시작한 쪽의 커넥터가 채널의 첫 번째 커넥터가 되므로, 순서가 참조 설계처럼 `ZONE`, `HPC` 가 된다.
- **주소는 커넥터에 넣는다.** 편집기에는 엔드포인트 칸이 없다. 참조 설계의 엔드포인트 `ZONE_Endpoint`(`192.168.0.19`), `HPC_Endpoint`(`192.168.0.11`), `SOMEIP_SD` 리)가 입력한 세 주소이다.
- `HPC` **커넥터는 참조 설계를 바로잡는다.** 참조 설계에서는 컨트롤러가 없고 `ZONE` 의 주소를 가리킨다. 이 튜토리얼은 `HPC` 자신의 컨트롤러와 주소를 쓴다.
- **NM 은 꺼 둔다.** 새 클러스터는 NM 이 꺼져 있고, 참조 설계에도 NM 이 없다.
- **입력란이 없는 값이 있다.** 프로토콜 이름 `IEEE 802.3`, 종류 `WIRED`, `WAKE-UP-BY-CONTROLLER-SUPPORTED` `false`, VLAN 이름 `Vlan_13`, 엔드포인트 이름이다. 차이

확인하기

- [ ] `ZONE` 에 `EthernetCommController`(`11:22:ee:44:55:66`)와 `EthernetConnector`(`192.168.0.19`, **FIXED**)가 있다.
- [ ] `HPC` 에 `EthernetController`(`00:11:22:33:44:55`)와 `EthernetConnector`(`192.168.0.11`, **FIXED**, 멀티캐스트 `224.0.0.1`)가 있다.
- [ ] 두 컨트롤러의 **MAC LAYER TYPE** 이 꺼져 있다.
- [ ] `EthernetCluster` 의 **VLAN ID** 가 `13` 이고 `ZONE/EthernetConnector` 와 `HPC/EthernetConnector` 가 보인다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보이고, 탭 `VehicleDesign:System` 에 수정됨 표시가 없다.

다음 단계

LIN 설정에서 노드 `ECU` 와 `SLAVE_ECU` 를 LIN 클러스터 `LinCluster` 로 잇는다.

LIN 설정

이 장에서 할 일

노드 `ECU` 와 `SLAVE_ECU` 를 LIN 버스로 잇는다. `ECU` 에 LIN 마스터 컨트롤러와 커넥터를, `SLAVE_ECU` 에 LIN 슬레이브 컨트롤러와 커넥터를 만들고, 두 커넥터를 LIN 클러스터 `L` ms 는 System 편집기가 받는 최솟값보다 작으므로, 채널에는 유휴 타임아웃이 들어가지 않는다.

시작하기 전에

- ETHERNET 설정을 마친 뒤 시작한다. `SLAVE_ECU` 는 아직 비어 있고, **VALIDATION** 패널에는 `No validation issues.` 가 보인다.
- 탭 `VehicleDesign:System` 을 열고 **VALIDATION** 패널을 닫아 둔다.
- `HPC` 가 속성 패널에 가려지면 **Fit to View** 를 클릭한 뒤 **Zoom Out** 을 몇 번 클릭한다. 그림은 이렇게 축소한 화면이다.
- 속성 패널에 값을 넣을 때마다 `Tab` 키를 누른다.

어디서	항목	값
LIN MASTER (노드 `ECU`)	**NAME** / **PROTOCOL VERSION**	`LinController_Master` / `LIN 2.0`
LIN MASTER (노드 `ECU`)	**TIME BASE** / **TIME BASE JITTER**	`1` `ms` / `1` `ms`
LIN SLAVE (노드 `SLAVE_ECU`)	**NAME** / **PROTOCOL VERSION**	`LinSlaveController` / `LIN 2.0`
LIN CONNECTOR (노드 `ECU`)	**NAME** / **CONTROLLER**	`LinConnector_Master` / `LinController_Master`
LIN CONNECTOR (노드 `SLAVE_ECU`)	**NAME** / **CONTROLLER**	`LinConnector` / `LinSlaveController`
LIN CLUSTER	**NAME** / **IDLE TIMEOUT**	`LinCluster` / `10` `ms` (거부되어 저장되지 않는다)
LIN CLUSTER	**CHANNEL NAME**	`PhysicalChannel` (기본값, 그대로 둔다)

LIN 컨트롤러 만들기

마스터 컨트롤러 추가하기

1. 노드 `ECU` 안쪽의 빈 곳을 오른쪽 클릭한다.

2. **Add Controller > LIN Master** 를 클릭한다.

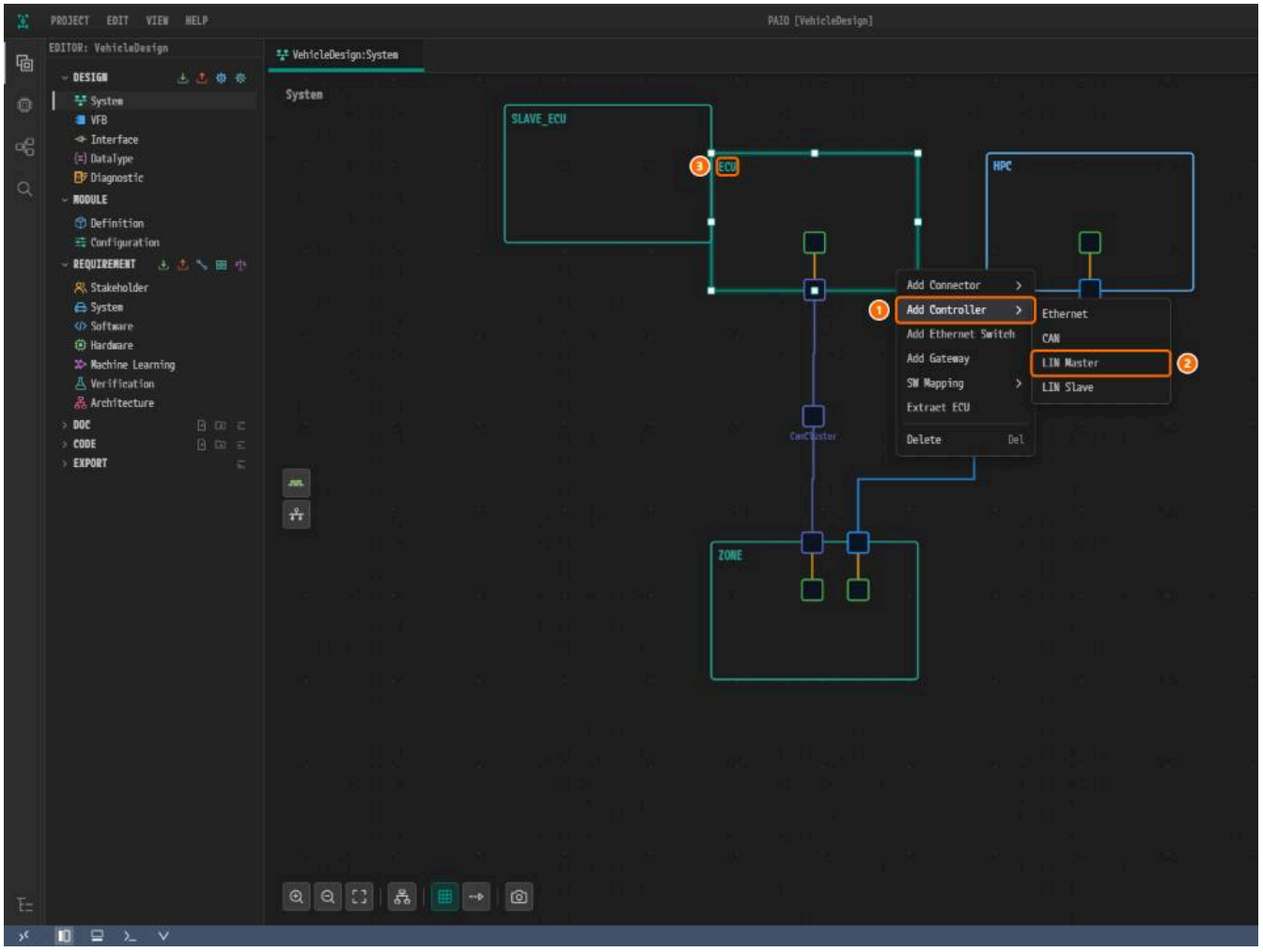


*그림 5-1. ① **Add Controller**, ② **LIN Master**, ③ 노드 `ECU`.*

3. 노드 안에 생긴 작은 사각형을 클릭한다. **NAME** 이 `ECU_LinMaster_Controller_0` 인 **LIN MASTER** 패널이 열린다. 프로토콜 버전을 고를 때까지 **V** 아이콘에 배지 `1` 이 붙

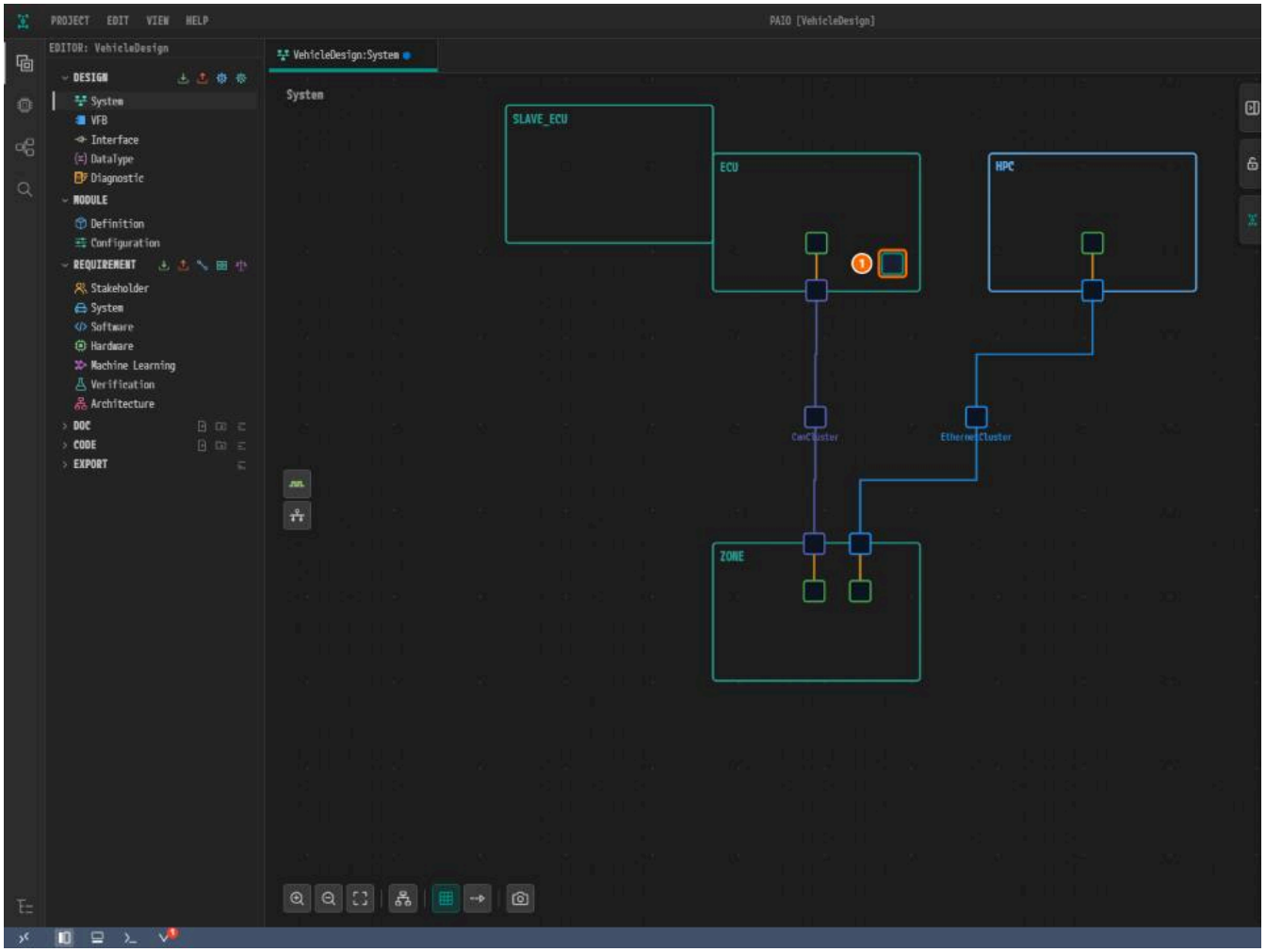


*그림 5-2. ① 컨트롤러, ② **LIN MASTER**, ③ **NAME**, ④ **PROTOCOL VERSION**, ⑤ **TIME BASE**, ⑥ **TIME BASE JITTER**.*

이름, 프로토콜 버전, 타임 베이스 입력하기

1. **NAME** 에 `LinController_Master` 를 입력한다.
2. **PROTOCOL VERSION** 아래의 `LIN 2.0` 을 클릭한다. **V** 아이콘의 배지가 사라진다.
3. **TIME BASE** 의 단위를 먼저 `s` 에서 `ms` 로 바꾼 뒤 `1` 을 입력한다.
4. **TIME BASE JITTER** 도 같은 방식으로 단위 `ms`, 값 `1` 을 넣는다. **WAKE-UP BY CONTROLLER** 와 **LIN SLAVES** 는 건드리지 않는다.

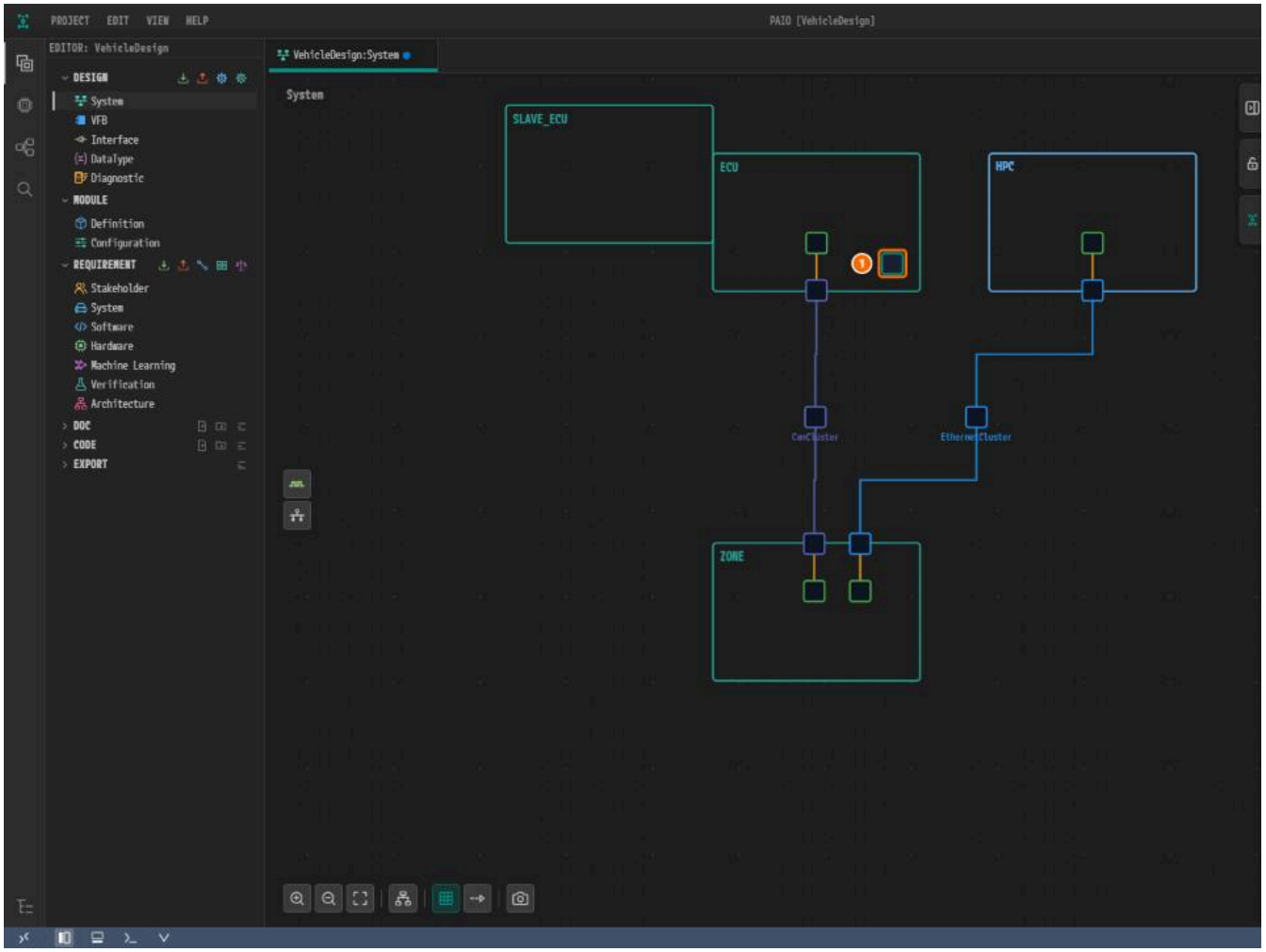


*그림 5-3. ① 컨트롤러, ② **NAME**, ③ **PROTOCOL VERSION**, ④ **TIME BASE**, ⑤ **TIME BASE JITTER**.*

> **주의.** 숫자보다 단위를 먼저 바꾼다. 단위의 기본값은 `s` 이고, 숫자를 넣은 뒤 단위를 바꾸면 숫자가 다시 계산될 수 있다.

SLAVE_ECU 에 슬레이브 컨트롤러 만들기

1. `SLAVE_ECU` 안쪽의 빈 곳을 오른쪽 클릭한다.
2. **Add Controller > LIN Slave** 를 클릭한다.

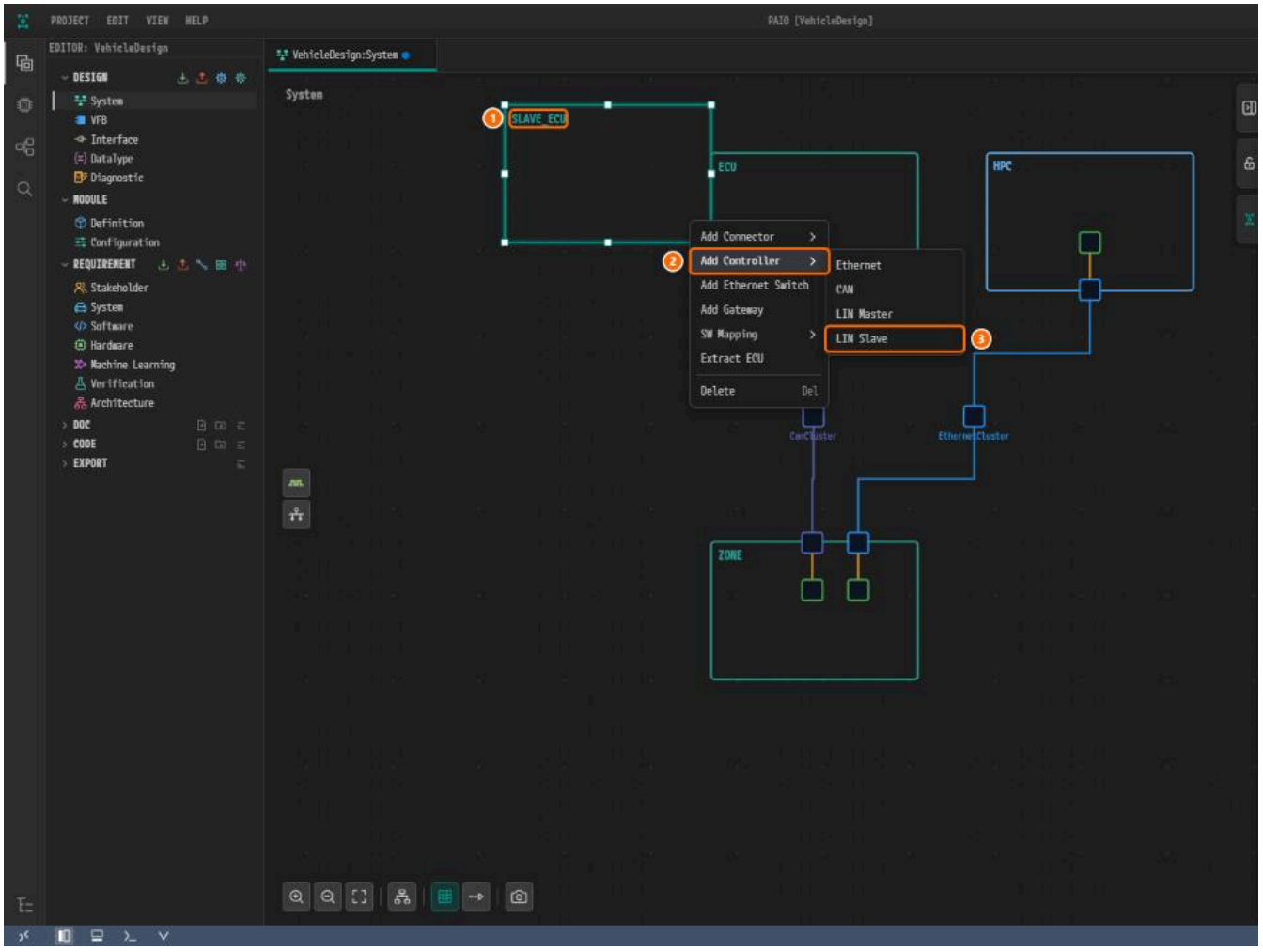


*그림 5-4. ① 노드 `SLAVE_ECU`, ② **Add Controller**, ③ **LIN Slave**.*

3. 새 사각형을 클릭한다. **LIN SLAVE** 패널이 열린다.

4. **NAME** 에 `LinSlaveController` 를 입력한다.

5. **PROTOCOL VERSION** 아래의 `LIN 2.0` 을 클릭한다. 나머지는 그대로 둔다.

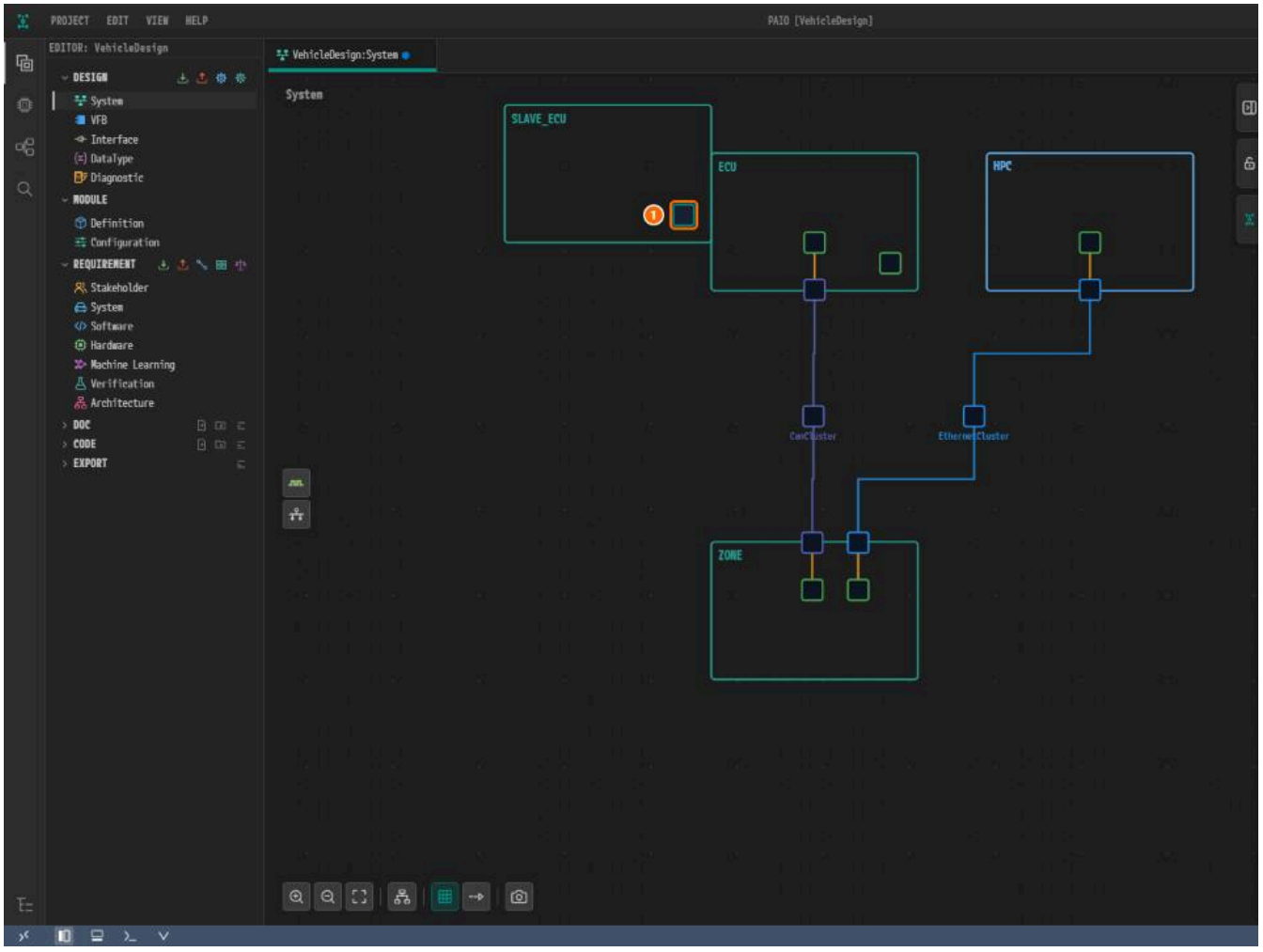


*그림 5-5. ① 컨트롤러, ② **NAME**, ③ **PROTOCOL VERSION**.*

슬레이브 패널에는 **TIME BASE** 칸이 없다.

LIN 커넥터 만들기

커넥터 추가하기

1. `ECU` 안쪽의 빈 곳을 오른쪽 클릭한다.
2. **Add Connector > LIN** 을 클릭한다.

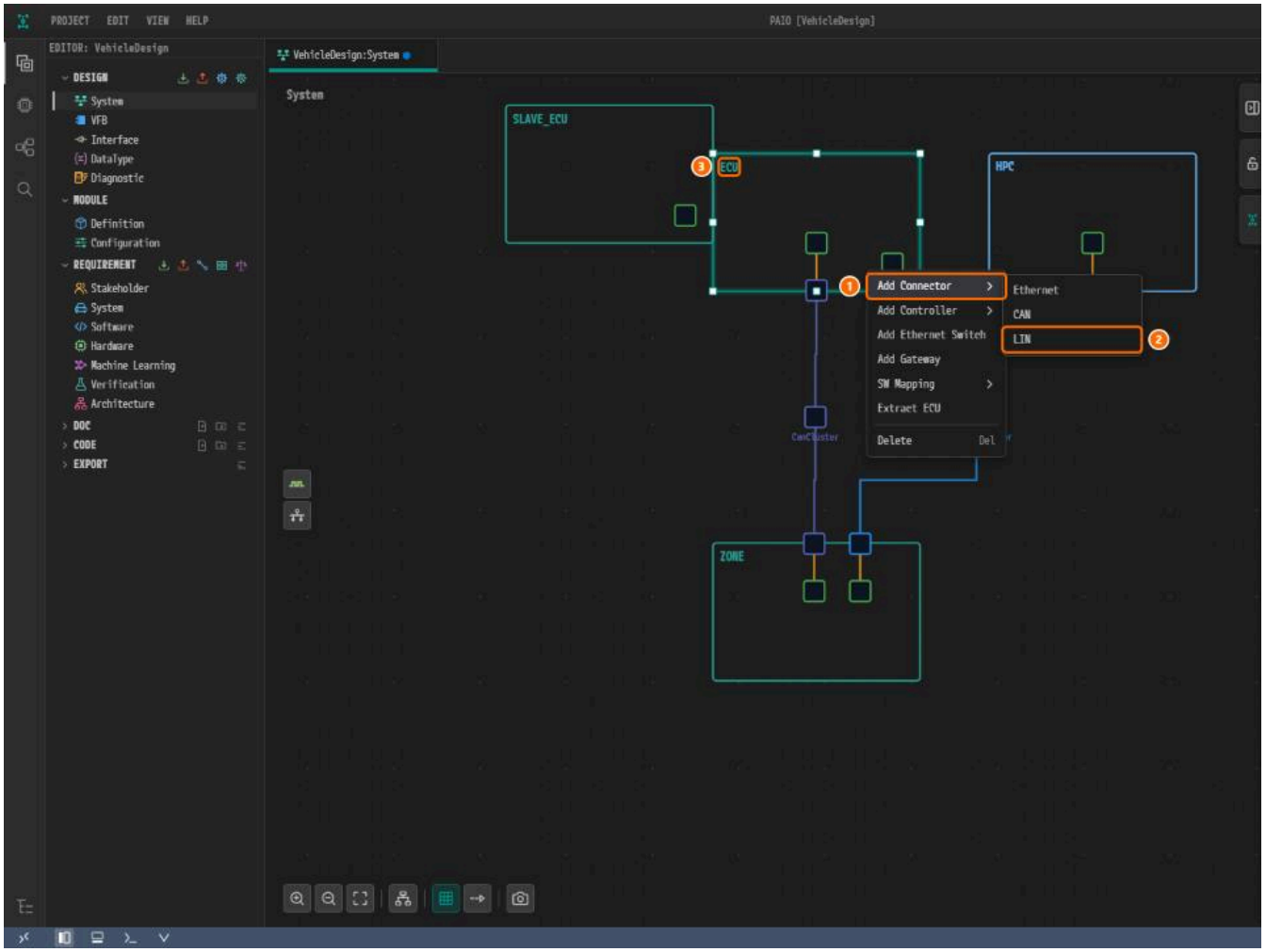


*그림 5-6. ① **Add Connector**, ② **LIN**, ③ 노드 `ECU`.*

3. `ECU` 아래 가장자리, `CanConnector` 오른쪽에 생긴 커넥터 사각형을 클릭한다. **LIN CONNECTOR** 패널이 열린다. 컨트롤러를 고를 때까지 **CONTROLLER** 는 빨갛고 `Contro`

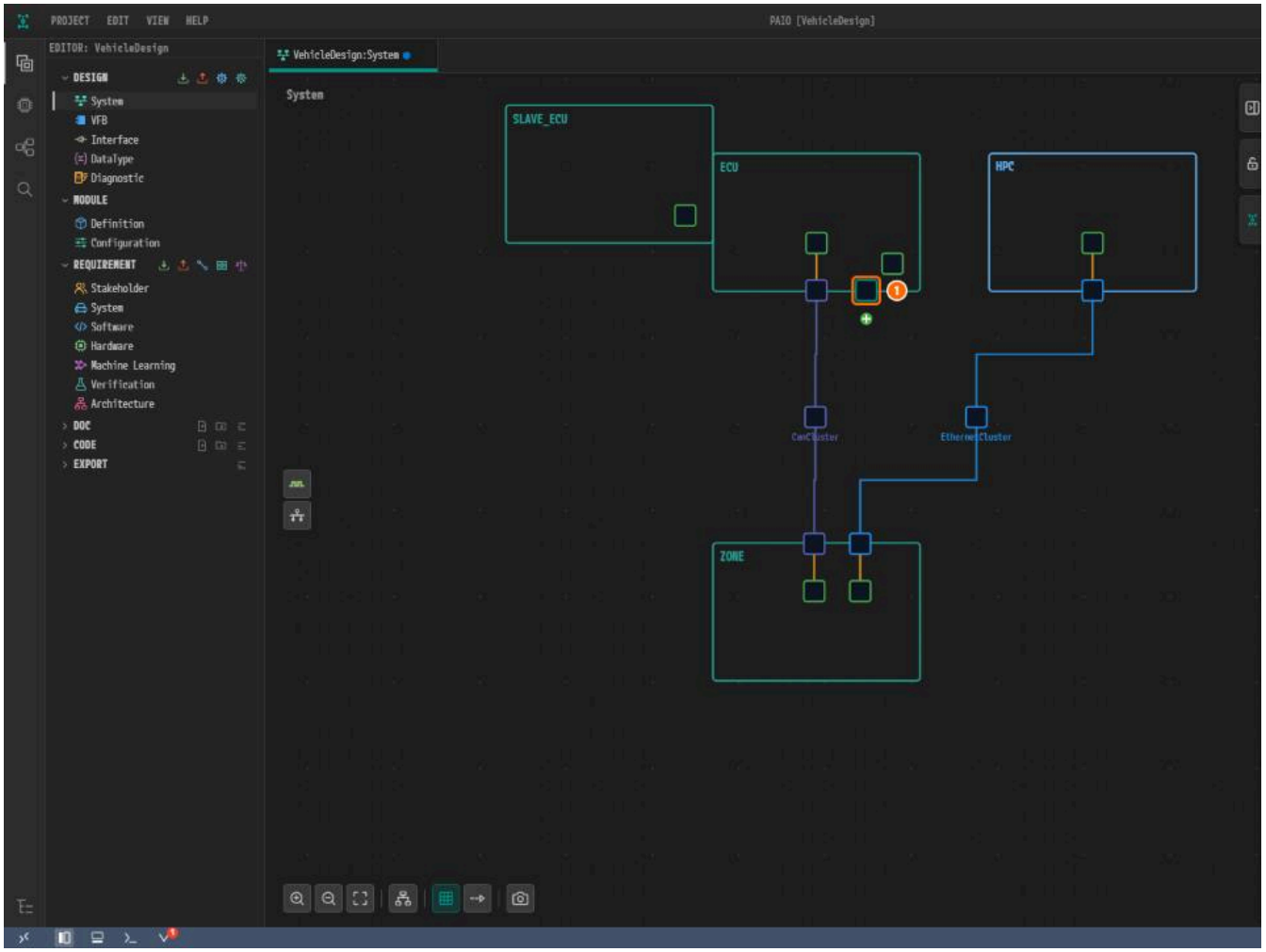


*그림 5-7. ① 커넥터, ② **LIN CONNECTOR**, ③ **NAME**, ④ **CONTROLLER**, ⑤ **LIN CONNECTOR SETTINGS**.*

이름과 컨트롤러 지정하기

1. **NAME** 에 `LinConnector_Master` 를 입력한다.
2. **CONTROLLER** 를 클릭하고 목록의 유일한 항목 `LinController_Master` 를 클릭한다.

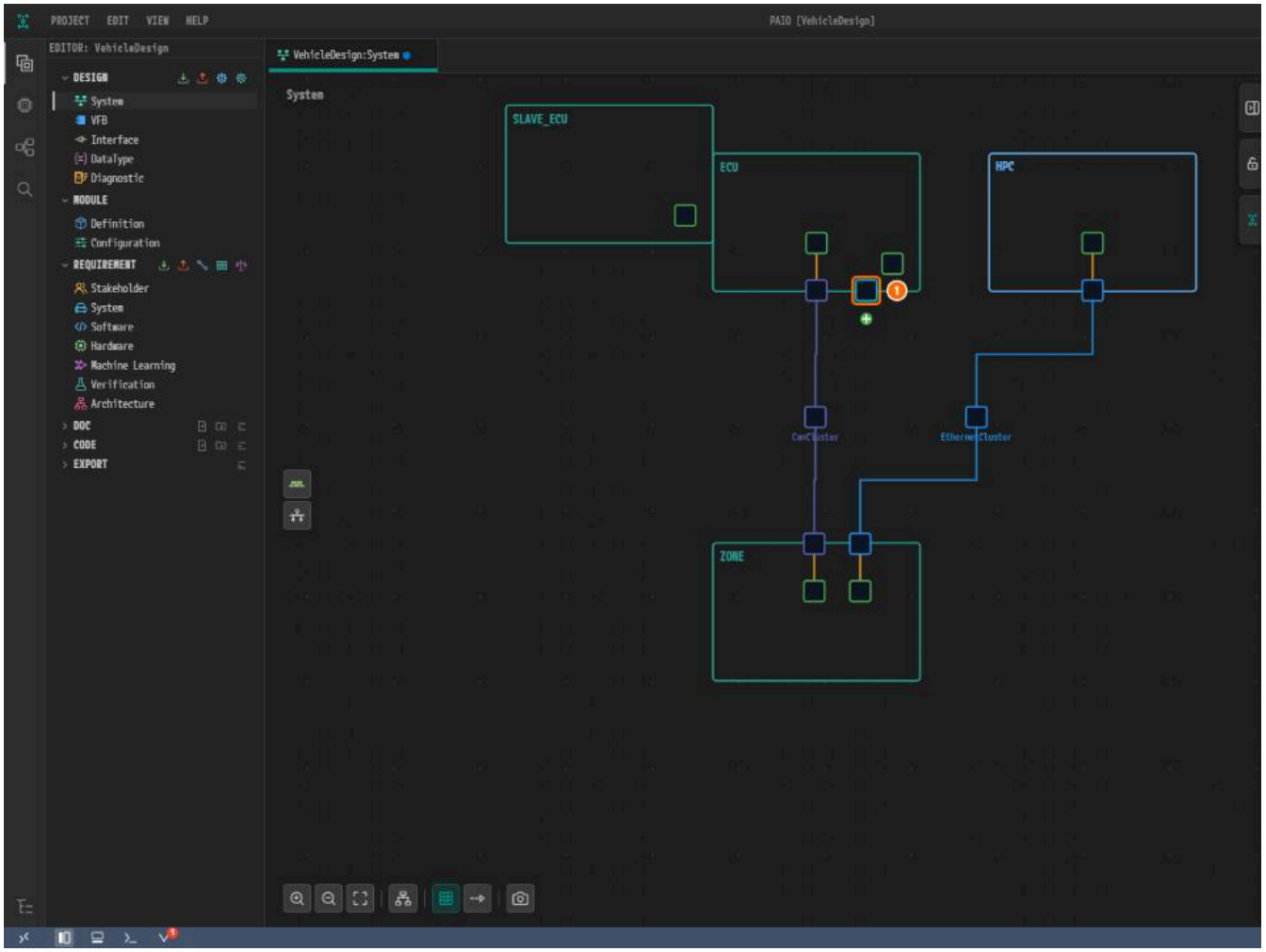


*그림 5-8. ① 커넥터, ② **NAME**, ③ **CONTROLLER**, ④ `LinController_Master`.*

> **참고.** 커넥터 메뉴의 **LIN** 항목은 하나뿐이다. 커넥터가 마스터 쪽인지 슬레이브 쪽인지는 고른 컨트롤러가 정한다.

SLAVE_ECU 에도 커넥터 만들기

1. `SLAVE_ECU` 안쪽의 빈 곳을 오른쪽 클릭해 **Add Connector > LIN** 을 클릭하고, 새 커넥터를 클릭한다.
2. **NAME** 에 `LinConnector` 를 입력한다.
3. **CONTROLLER** 에서 `LinSlaveController` 를 고른다.

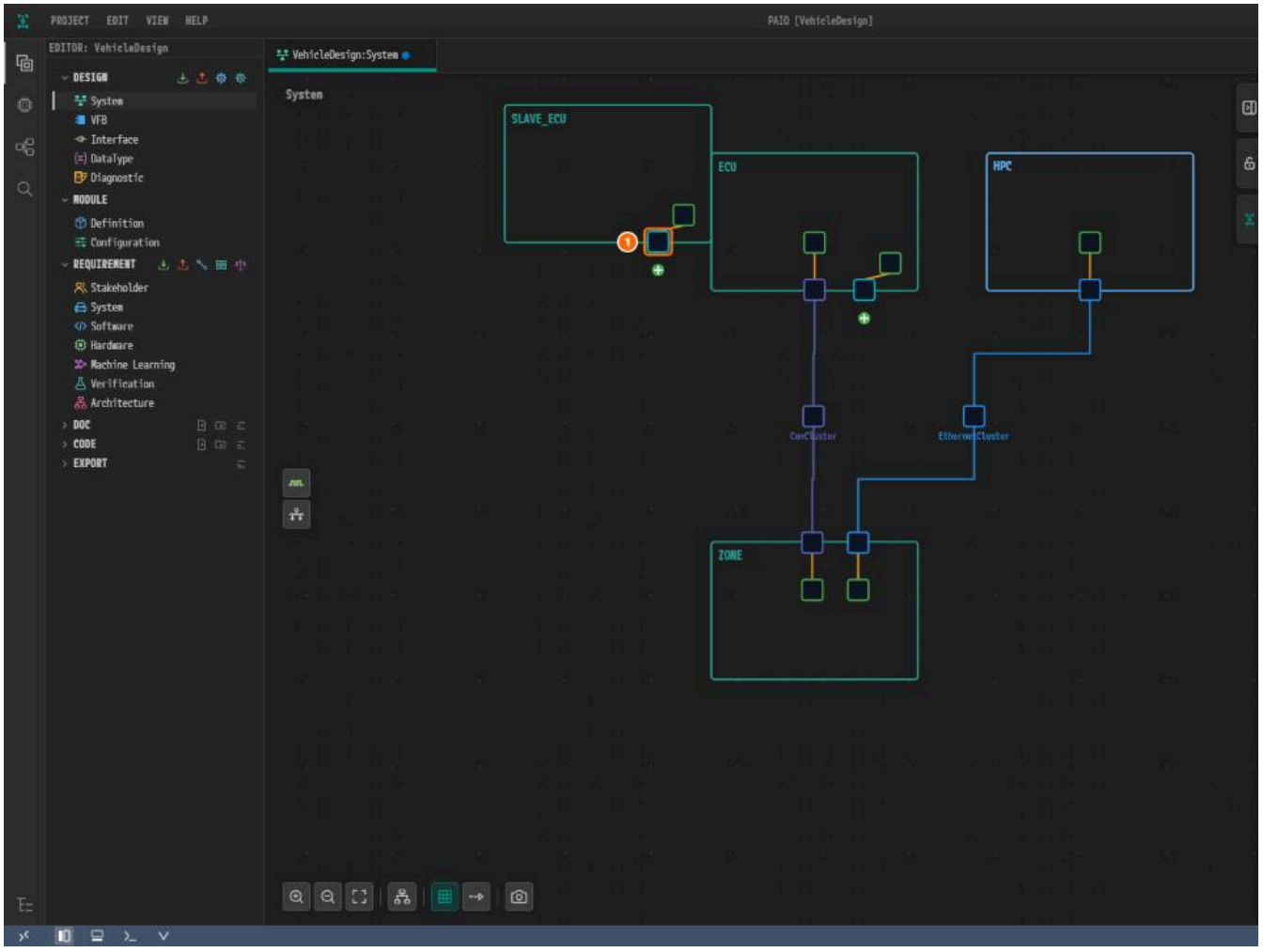


*그림 5-9. ① `SLAVE_ECU` 의 커넥터, ② **NAME**, ③ **CONTROLLER**.*

LIN 클러스터 만들기

SLAVE_ECU 커넥터에서 클러스터 만들기

참조 설계처럼 채널에서 `SLAVE_ECU` 의 커넥터가 먼저 오도록 `SLAVE_ECU` 에서 클러스터를 만든다.

1. `SLAVE_ECU` 의 `LinConnector` 아래 초록 **+** 배지를 클릭한다.
2. **Create CommunicationCluster [Lin]** 을 클릭한다.

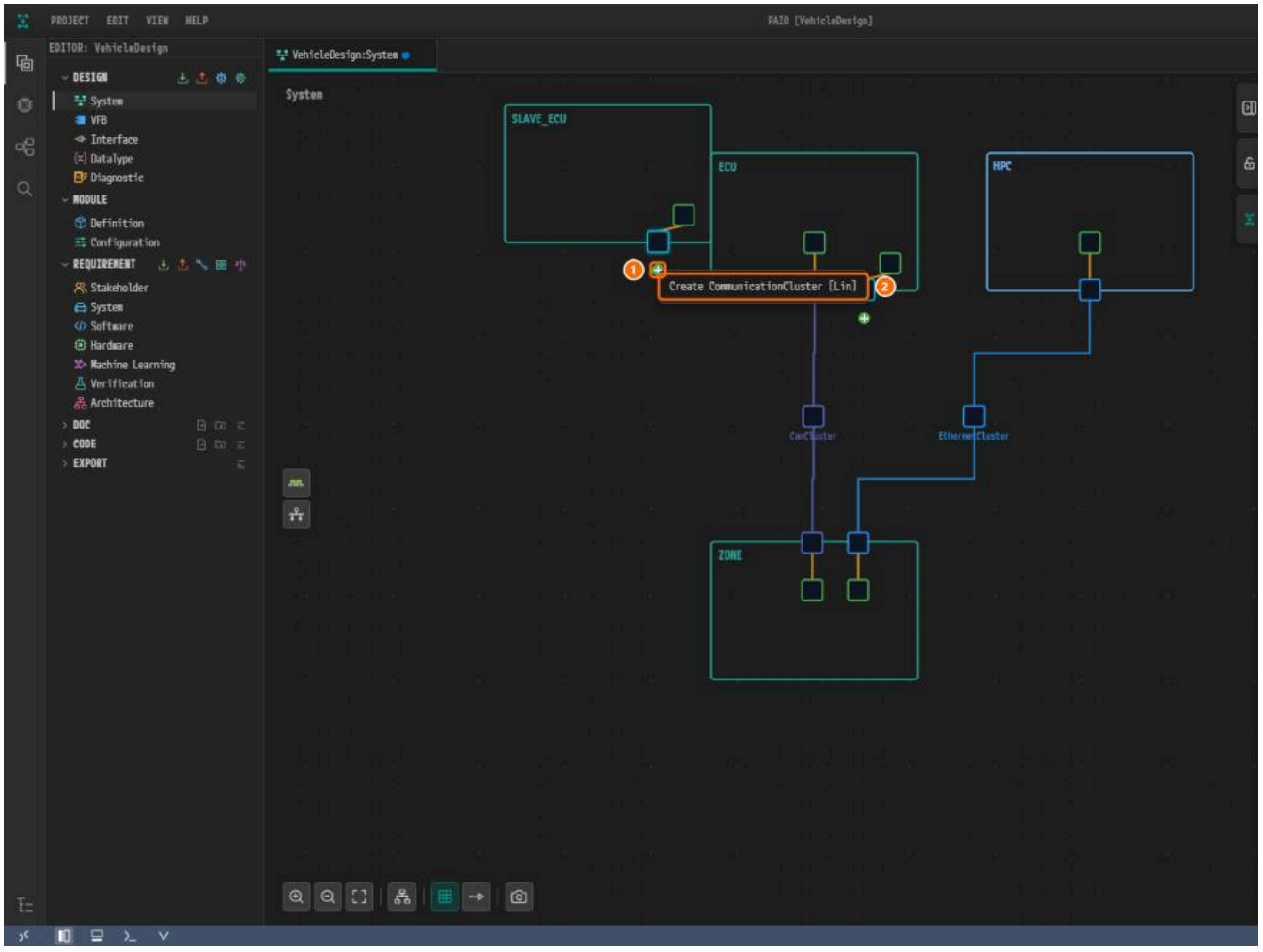


*그림 5-10. ① + 배지, ② **Create CommunicationCluster [Lin]**.*

3. 새 클러스터 사각형이 보이지 않으면 **Fit to View**(필요하면 **Zoom Out**)를 클릭한다.

4. 클러스터 사각형을 클릭한다. **LIN CLUSTER** 패널이 열린다.

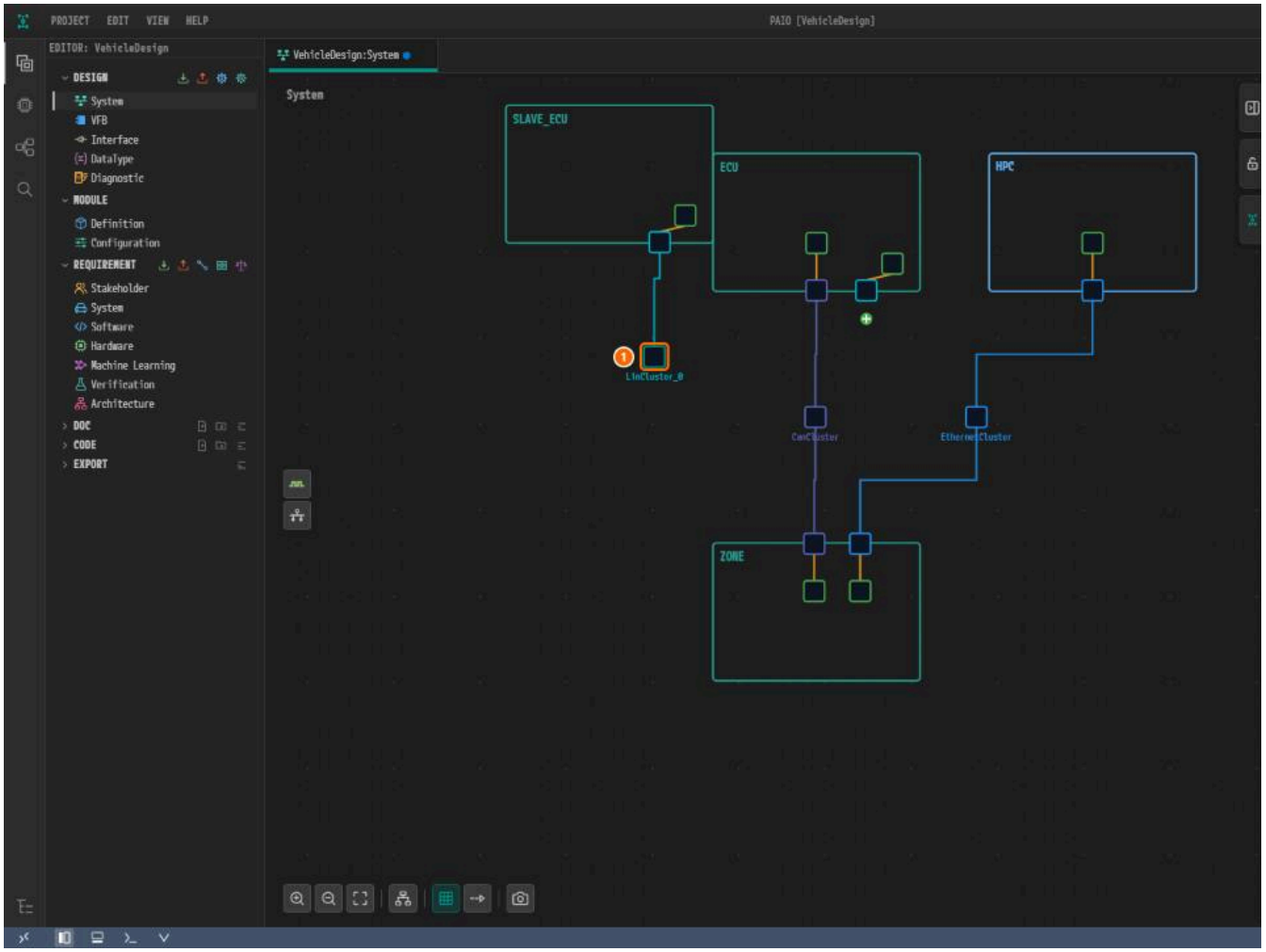


*그림 5-11. ① 클러스터, ② **NAME**, ③ **BAUDRATE (BPS)**, ④ **CHANNEL NAME**, ⑤ **IDLE TIMEOUT**.*

5. **NAME** 에 `LinCluster` 를 입력한다.

6. **IDLE TIMEOUT** 의 단위를 `ms` 로 바꾸고 `10` 을 입력한다. 칸이 빨개지고 `Must be at least 0.1` 이 보이며 값은 저장되지 않는다. 나머지 칸은 그대로 둔다.

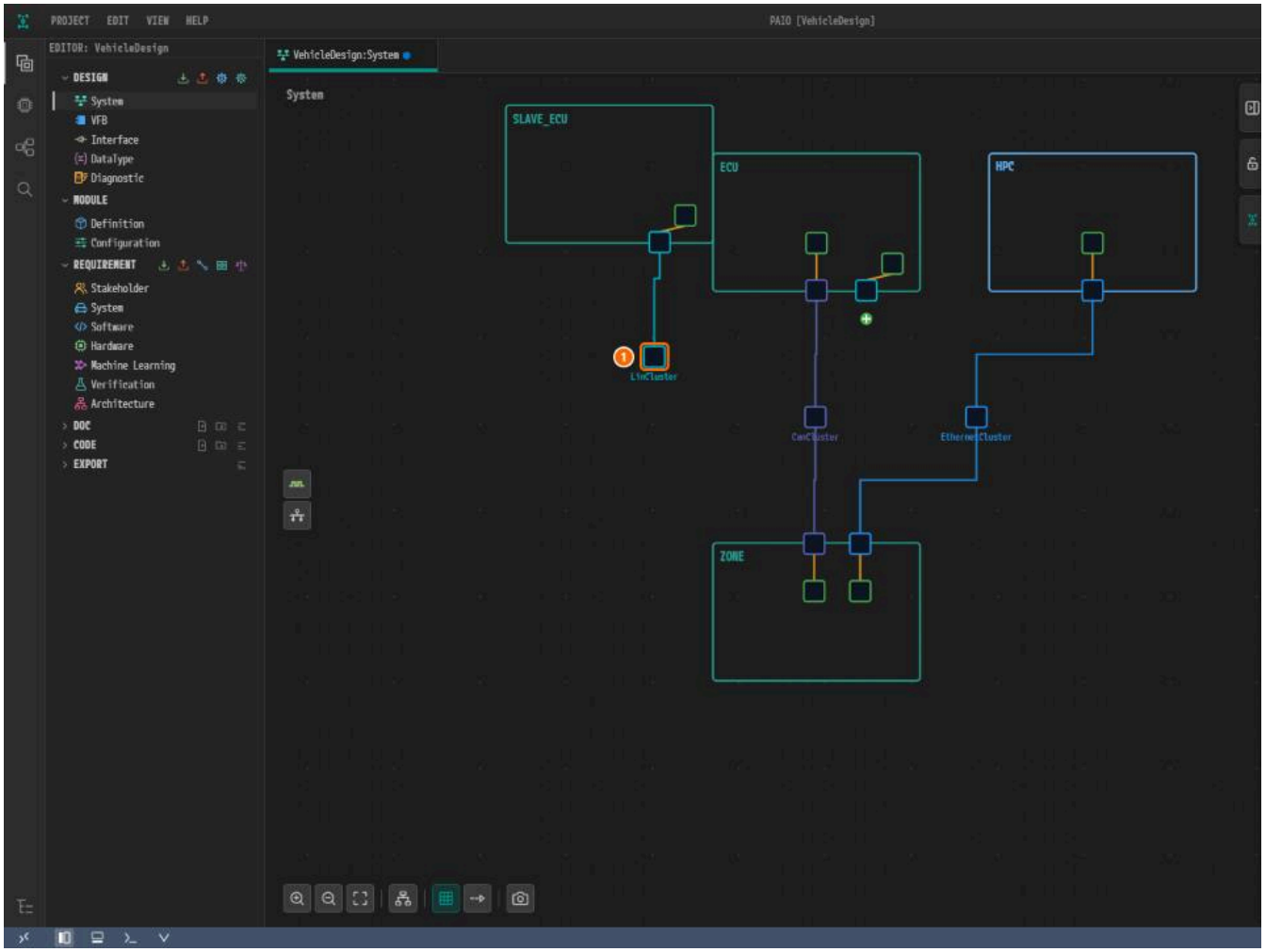


*그림 5-12. ① 클러스터, ② **NAME** = `LinCluster`, ③ 거부된 **IDLE TIMEOUT**.*

칸이 빨개지지 않고 `10` 을 받았다면 단위가 아직 `s` 였던 것이다. 숫자를 지우고 `Tab` 을 눌러 빈 칸으로 둔다.

> **참고. BAUDRATE (BPS)** 버튼은 하나도 켜져 있지 않지만 저장된 클러스터에는 `19200` 이 들어간다. ARXML 익스포트를 본다.

ECU 커넥터를 채널에 연결하기

1. `ECU` 의 `LinConnector_Master` 아래 **+** 배지를 클릭한다.
2. **Connect to Channel... > LinCluster > PhysicalChannel** 을 클릭한다.

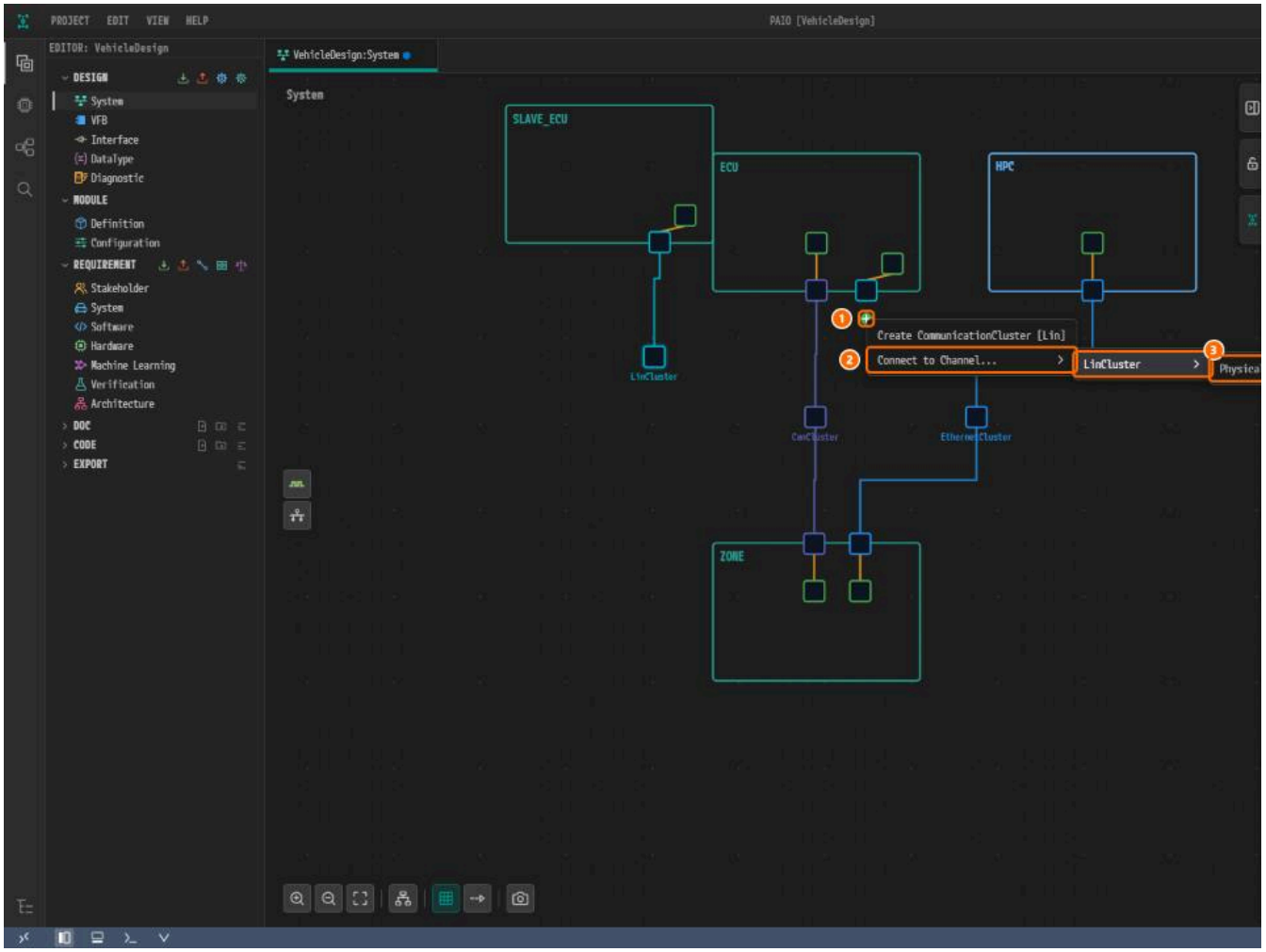


*그림 5-13. ① + 배지, ② **Connect to Channel...**, ③ **LinCluster**, ④ **PhysicalChannel**.*

> **주의.** 여기서 **Create CommunicationCluster [Lin]** 은 누르지 않는다. 클러스터를 하나 더 만든다.

배치 정리하고 저장하기

1. **Auto Layout** 을 클릭하고 이어서 **Fit to View**(필요하면 **Zoom Out**)를 클릭한다.
2. LIN 클러스터를 클릭해 **IDLE TIMEOUT** 이 비어 있고 **Connected Connectors (2)** 에 `SLAVE_ECU/LinConnector`, `ECU/LinConnector_Master` 가 있는지 확인한다.

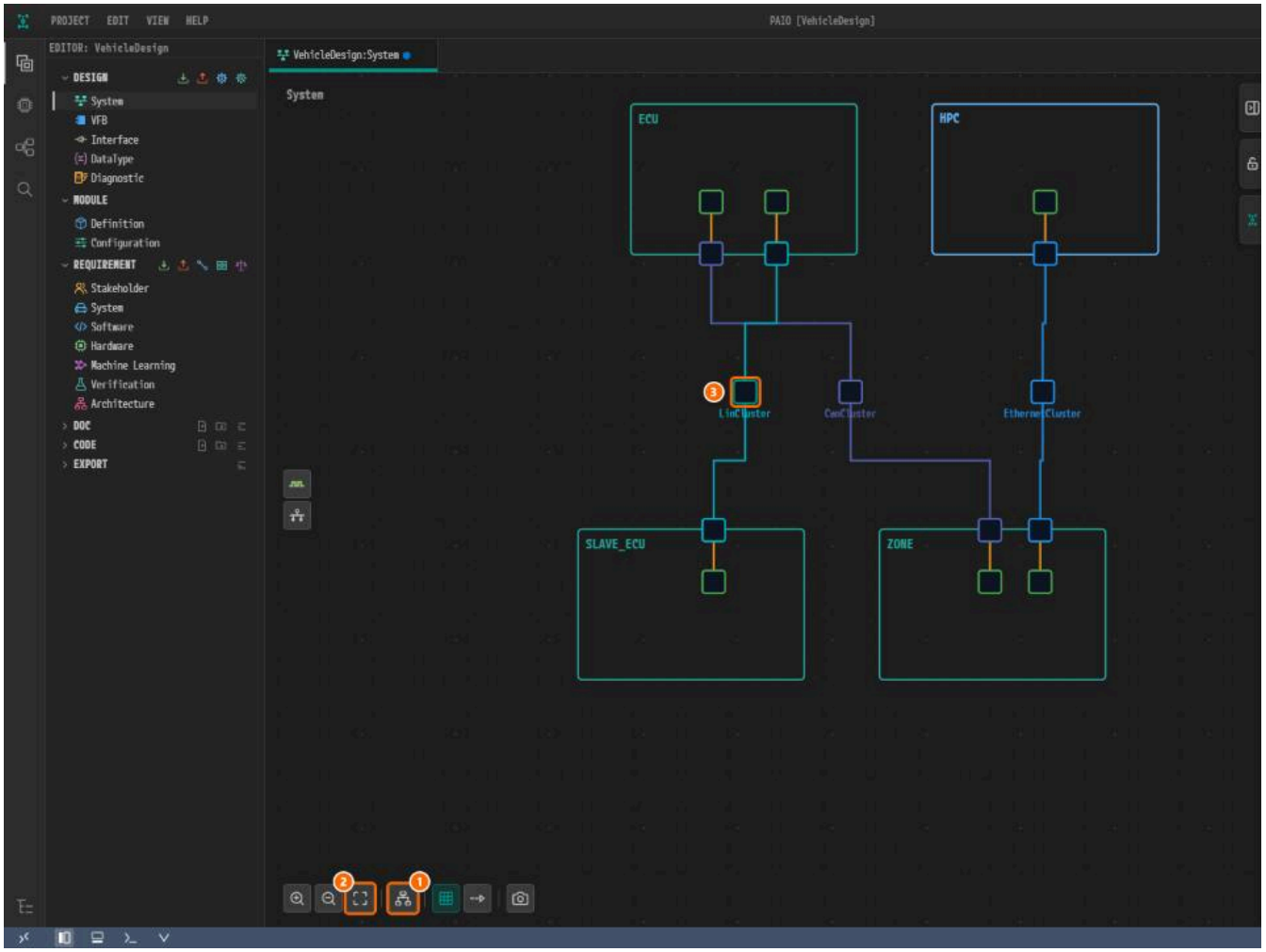


*그림 5-14. ① **Auto Layout**, ② **Fit to View**, ③ 클러스터, ④ **CHANNEL NAME**, **IDLE TIMEOUT** 과 두 커넥터.*

3. 빈 캔버스를 클릭하고 `Ctrl+S` 를 누른다.

검증 결과 보기

1. **V** 아이콘을 클릭한다. **VALIDATION** 패널에 `No validation issues.` 가 보인다.

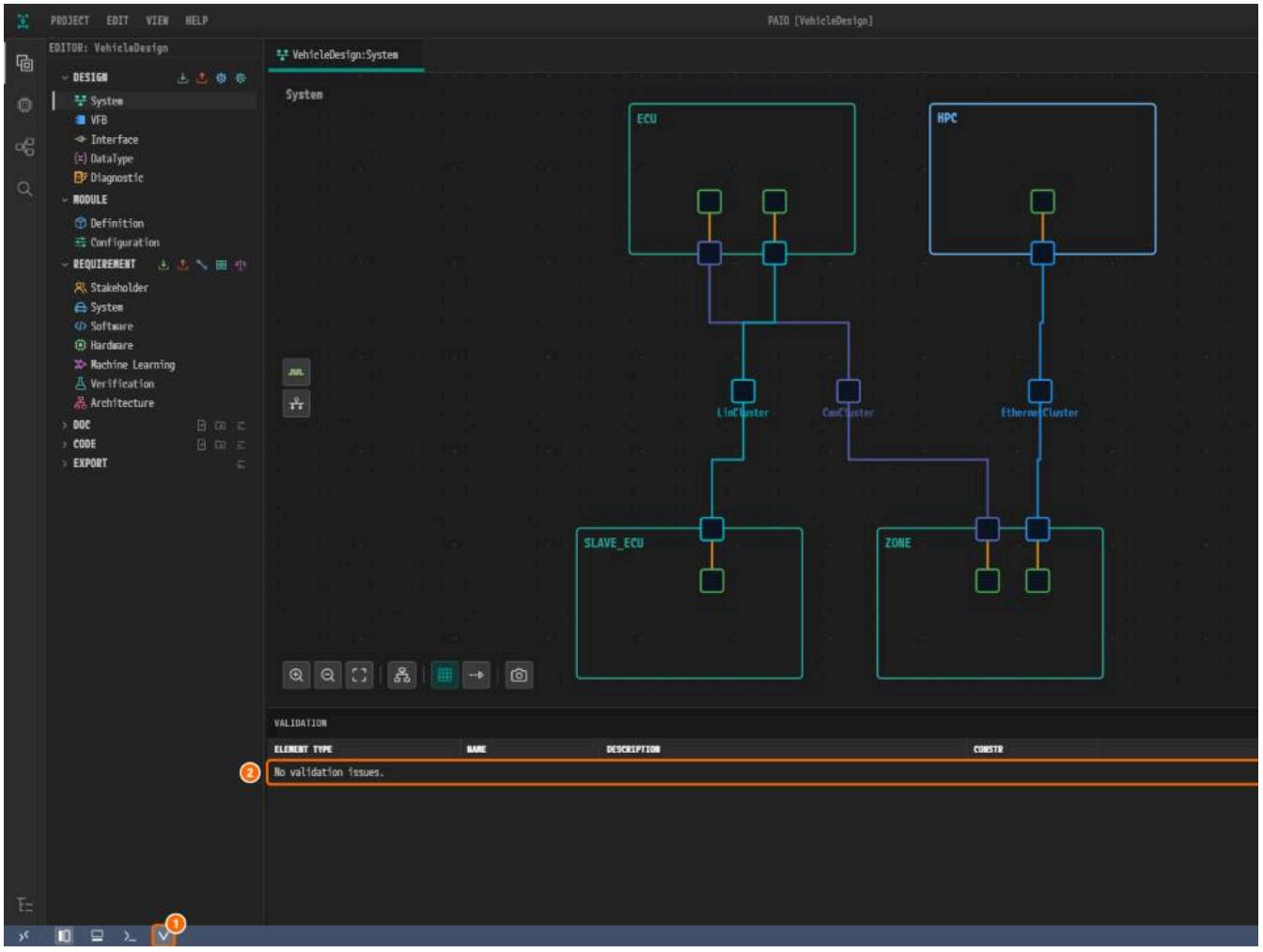


*그림 5-15. ① **V** 아이콘, ② `No validation issues.`.*

LIN 클러스터에는 NM 이 없고 TP 는 꺼져 있으므로, 클러스터 설정 창에서 확인할 것이 없다.

이 장에서 넣지 않는 값

익스포트 결과의 차이는 ARXML 익스포트에서 다룬다.

참조 설계의 값	위치	System 편집기
`WAKE-UP-BY-CONTROLLER-SUPPORTED` = `true`	`ECU` 의 `LinController_Master`	**WAKE-UP BY CONTROLLER** 를 누르지 않으므로 값이 저장되지 않는다
`BUS-IDLE-TIMEOUT-PERIOD` = `0.01`	`LinCluster` 의 `PhysicalChannel`	**IDLE TIMEOUT** 은 `0.1` 초 이상만 받는다

왜 이렇게 설계하는가

- **커넥터 순서.** 참조 설계는 `SLAVE_ECU`, `ECU` 순서이므로 클러스터를 `SLAVE_ECU` 에서 만든다.
- **마스터와 슬레이브.** LIN 마스터 하나가 일정을 정하고 슬레이브가 따른다. 참조 설계는 마스터를 `ECU` 에 둔다.
- **타임 베이스는 마스터에만.** `LinController_Master` 에는 둘 다 `0.001` 초가 있고, `LinSlaveController` 에는 프로토콜 버전만 있다.
- **유휴 타임아웃은 비워 둔다. IDLE TIMEOUT** 은 AUTOSAR 가 `LinIfBusIdleTimeoutPeriod` 에 정한 범위(`0.1` 초 이상)를 따른다.
- **NAD, 스케줄 테이블, 프레임은 만들지 않는다.** 참조 설계에 없다.

확인하기

- [ ] `ECU` 에 `LinController_Master`(`LIN 2.0`, `1` `ms`, `1` `ms`)와 `LinConnector_Master`(**CONTROLLER** = `LinController_Master`)가 있다.
- [ ] `SLAVE_ECU` 에 `LinSlaveController`(`LIN 2.0`)와 `LinConnector`(**CONTROLLER** = `LinSlaveController`)가 있다.
- [ ] `LinCluster` 의 **IDLE TIMEOUT** 이 비어 있고, `PhysicalChannel` 에 `SLAVE_ECU/LinConnector`, `ECU/LinConnector_Master` 가 이 순서로 있다.
- [ ] 두 커넥터의 **CONNECTED CHANNEL** 이 `LinCluster / PhysicalChannel` 이다.
- [ ] 수정됨 표시가 없고 `No validation issues.` 가 보인다.

다음 단계

검증에서 완성한 토폴로지를 검증하고, 오류 행에서 해당 요소로 이동한다.

검증

이 장에서 할 일

LIN 설정까지 마친 토폴로지에 결함 세 개를 넣고, **VALIDATION** 패널이 이를 어떻게 보고하는지 읽은 뒤, 패널에 `No validation issues.` 가 보일 때까지 고친다.

시작하기 전에

- LIN 설정을 마친 상태에서 시작한다. **VALIDATION** 패널에는 `No validation issues.` 가 보인다.
- System 편집기(**EDITOR** > **DESIGN** > **System**)에서 아래 결함 세 개를 만들고 `Ctrl+S` 로 저장한다. 그다음 탭 `VehicleDesign:System` 을 × 로 닫고 **VALIDATION** 패널도

요소	칸	원래 값	결함 만들기
`HPC` 의 `EthernetConnector`	**UNICAST IPV4 ADDRESS**	`192.168.0.11`	`224.0.0.1` 을 입력하고 `Tab` 키를 누른다
`SLAVE_ECU` 의 `LinConnector`	**CONTROLLER**	`LinSlaveController`	칸 안의 ×(Clear Controller)를 클릭한다
`SLAVE_ECU` 의 `LinSlaveController`	**PROTOCOL VERSION**	`LIN 2.0`	`LIN 2.0` 을 한 번 더 클릭해 끈다

VALIDATION 패널 읽기

1. PAIO 는 열려 있는 프로젝트를 스스로 검증하므로, 푸터의 **V** 아이콘에는 이미 빨간 배지 `3` 이 붙어 있다. 아이콘을 클릭한다.

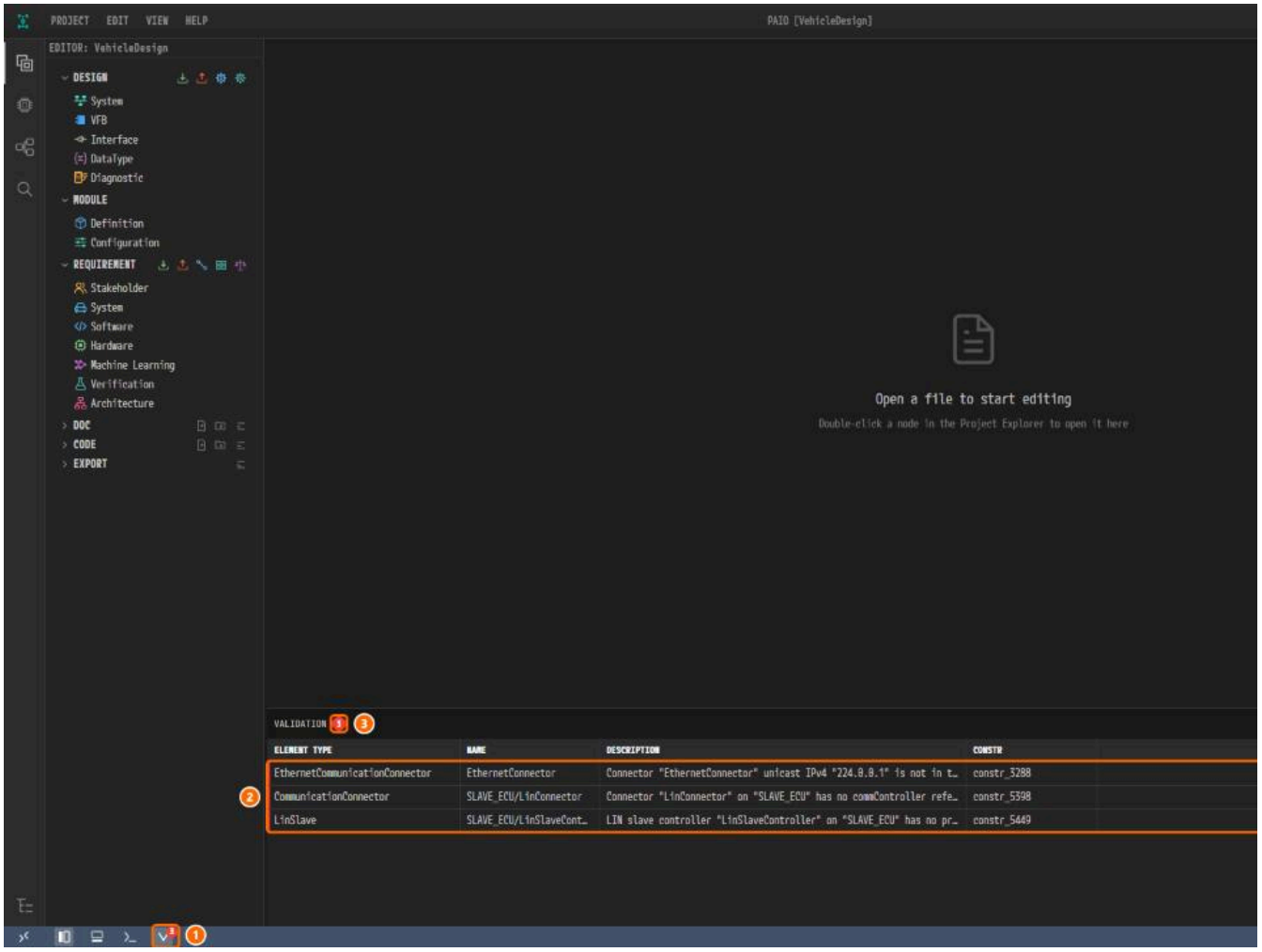


*그림 6-1. ① **V** 아이콘(배지 `3`), ② 오류 세 행, ③ 패널의 배지 `3`.*

2. 글자가 잘려 보이면 **NAME** 과 **DESCRIPTION** 머리글의 오른쪽 경계를 끌어 넓힌다(선택 사항).

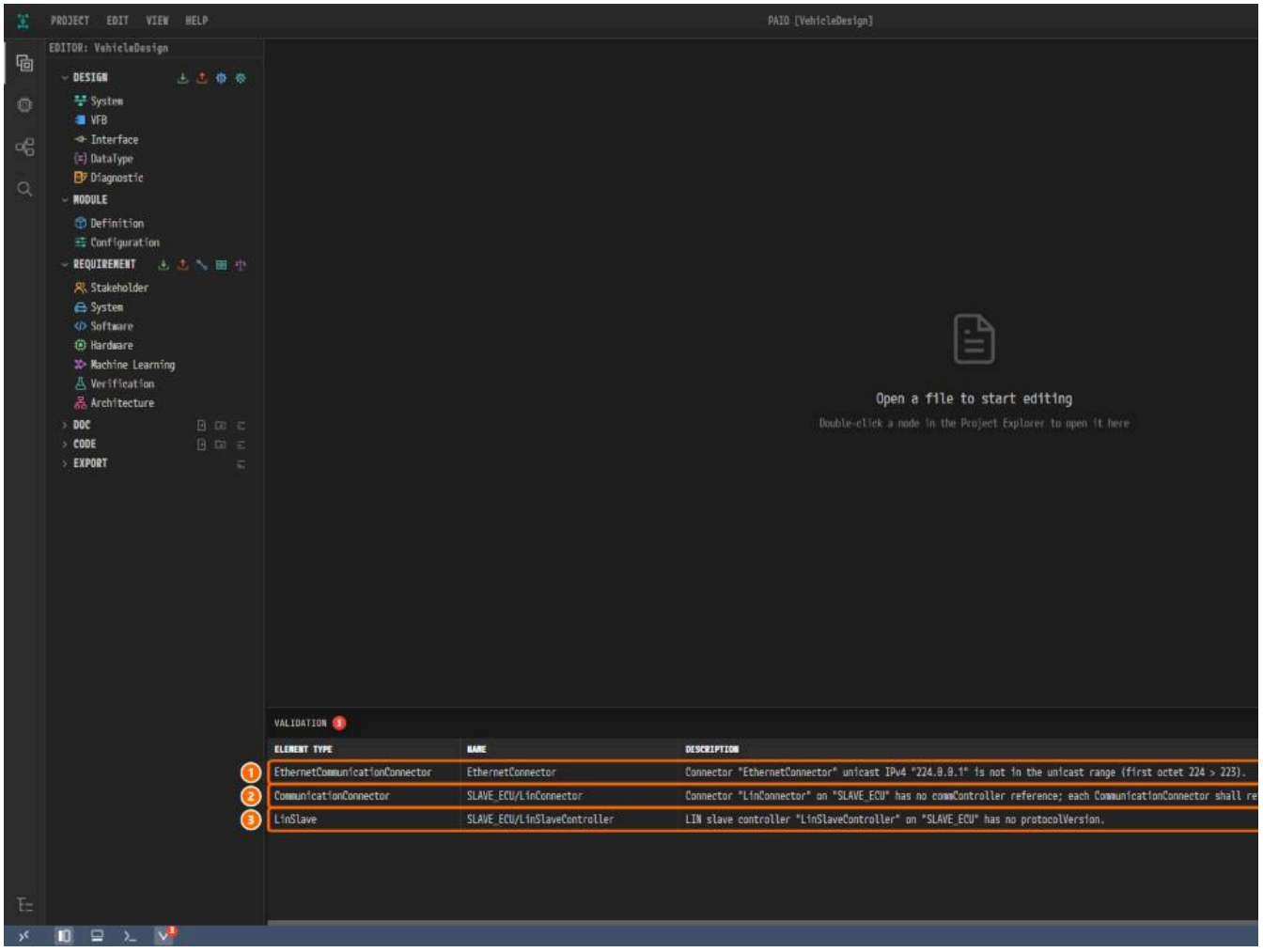


그림 6-2. ① `constr_3288` 행, ② `constr_5398` 행, ③ `constr_5449` 행.

#	ELEMENT TYPE	NAME	DESCRIPTION
1	`EthernetCommunicationConnector`	`EthernetConnector`	`Connector "EthernetConnector" unicast IPv4 "224.0.0.1" is not in the unicast`
2	`CommunicationConnector`	`SLAVE_ECU/LinConnector`	`Connector "LinConnector" on "SLAVE_ECU" has no commController reference; each reference exactly one communication controller.`
3	`LinSlave`	`SLAVE_ECU/LinSlaveController`	`LIN slave controller "LinSlaveController" on "SLAVE_ECU" has no protocolVersi`

행을 한 번 클릭하면 그 요소나 그 근처로 이동한다. 패널은 값을 고치는 즉시, 저장하기 전에도 갱신된다.

HPC 커넥터의 유니캐스트 주소 고치기

1. `constr_3288` 행을 클릭한다. 탭 `VehicleDesign:System` 이 열리고, `HPC` 가 아니라 `ZONE` 의 커넥터가 선택된다. 두 커넥터의 이름이 모두 `EthernetConnector` 이기

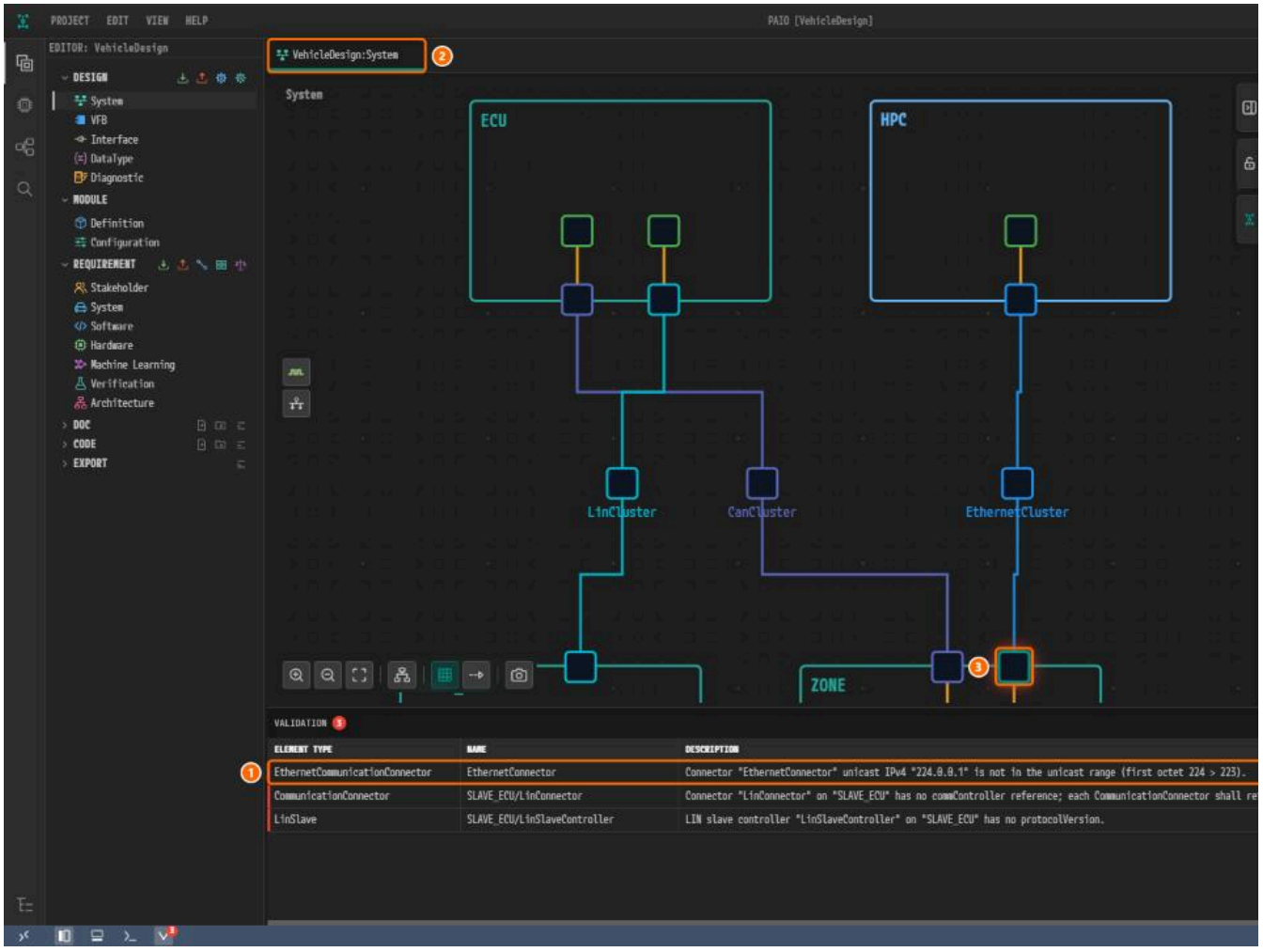


*그림 6-3. ① `constr_3288` 행, ② 탭 `VehicleDesign:System`, ③ `ZONE` 의 커넥터, ④ **UNICAST IPV4 ADDRESS**(`192.168.0.19`).*

2. **Fit to View** 를 클릭한 뒤 `HPC` 아래 가장자리의 커넥터를 클릭한다. **UNICAST IPV4 ADDRESS** 가 `224.0.0.1` 이다.

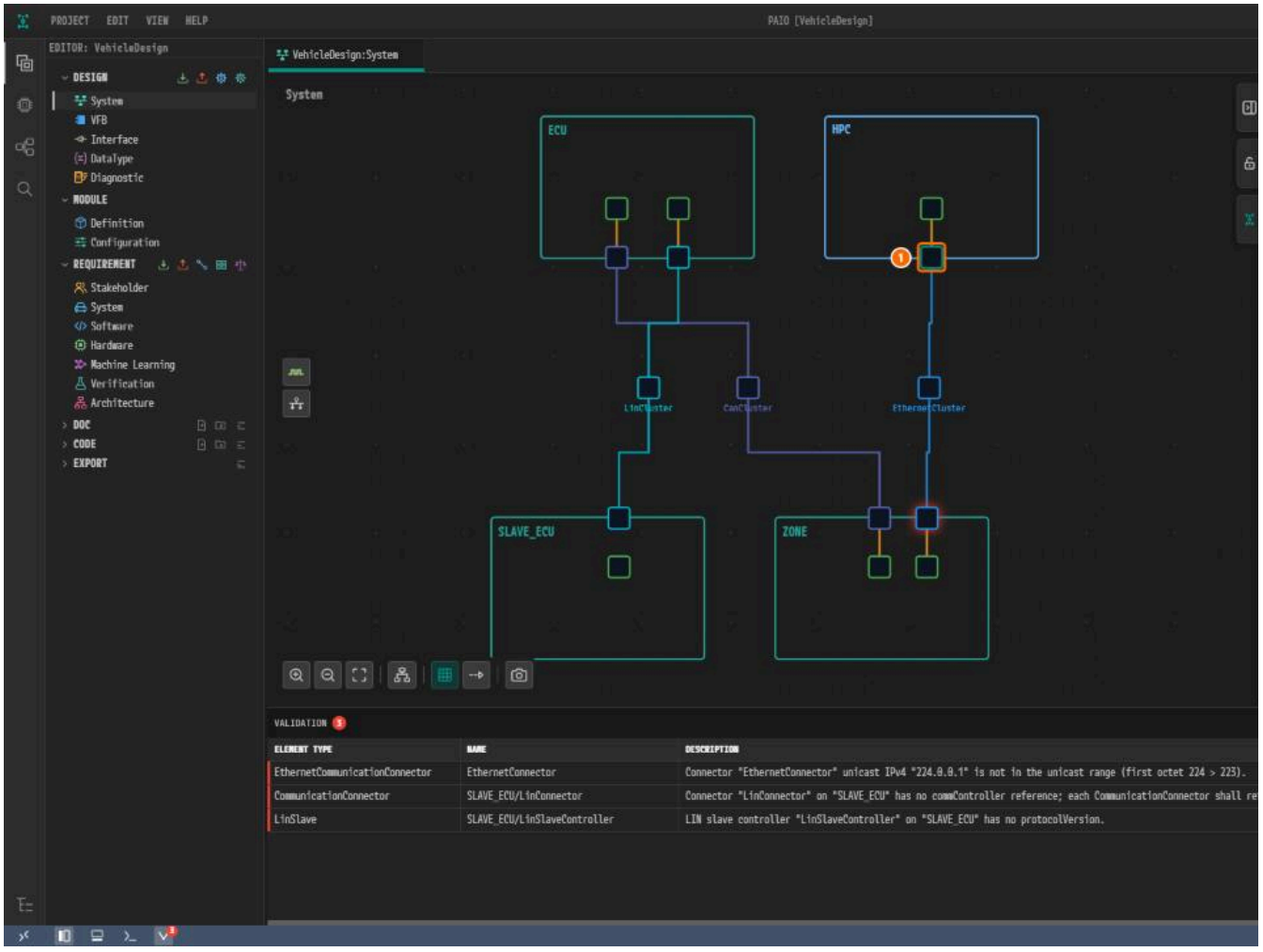


*그림 6-4. ① `HPC` 의 커넥터, ② **UNICAST IPV4 ADDRESS**, ③ **MULTICAST IPV4 ADDRESS**.*

3. **UNICAST IPV4 ADDRESS** 의 값을 `192.168.0.11` 로 바꾸고 `Tab` 키를 누른다. 패널이 두 행으로 줄어든다.

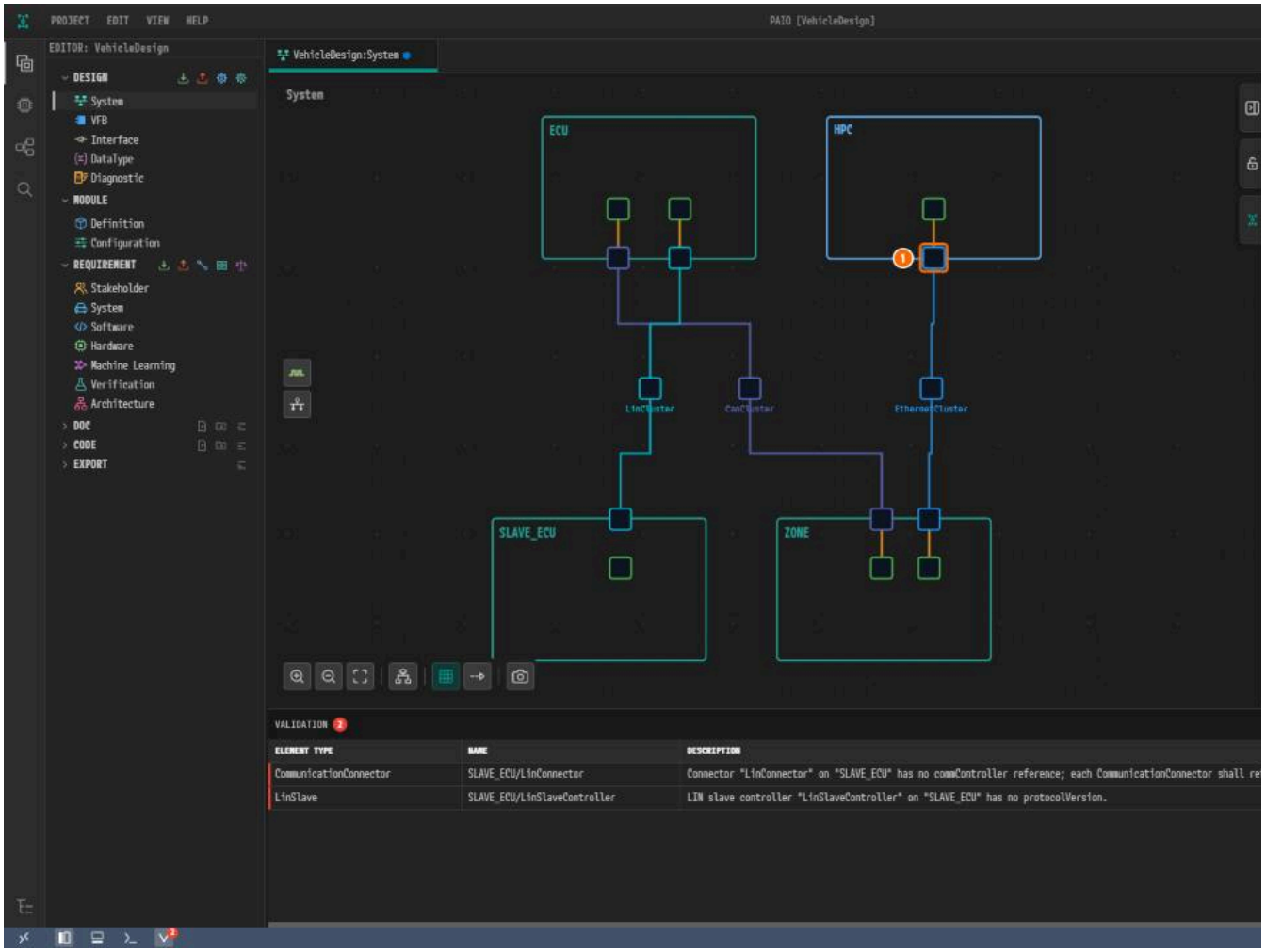


*그림 6-5. ① `HPC` 의 커넥터, ② **UNICAST IPV4 ADDRESS** = `192.168.0.11`.*

SLAVE_ECU 커넥터에 컨트롤러 지정하기

1. `constr_5398` 행을 클릭한다. `SLAVE_ECU` 의 `LinConnector` 가 바로 선택된다. **CONTROLLER** 에는 빨간 `Select controller...` 가 보인다.

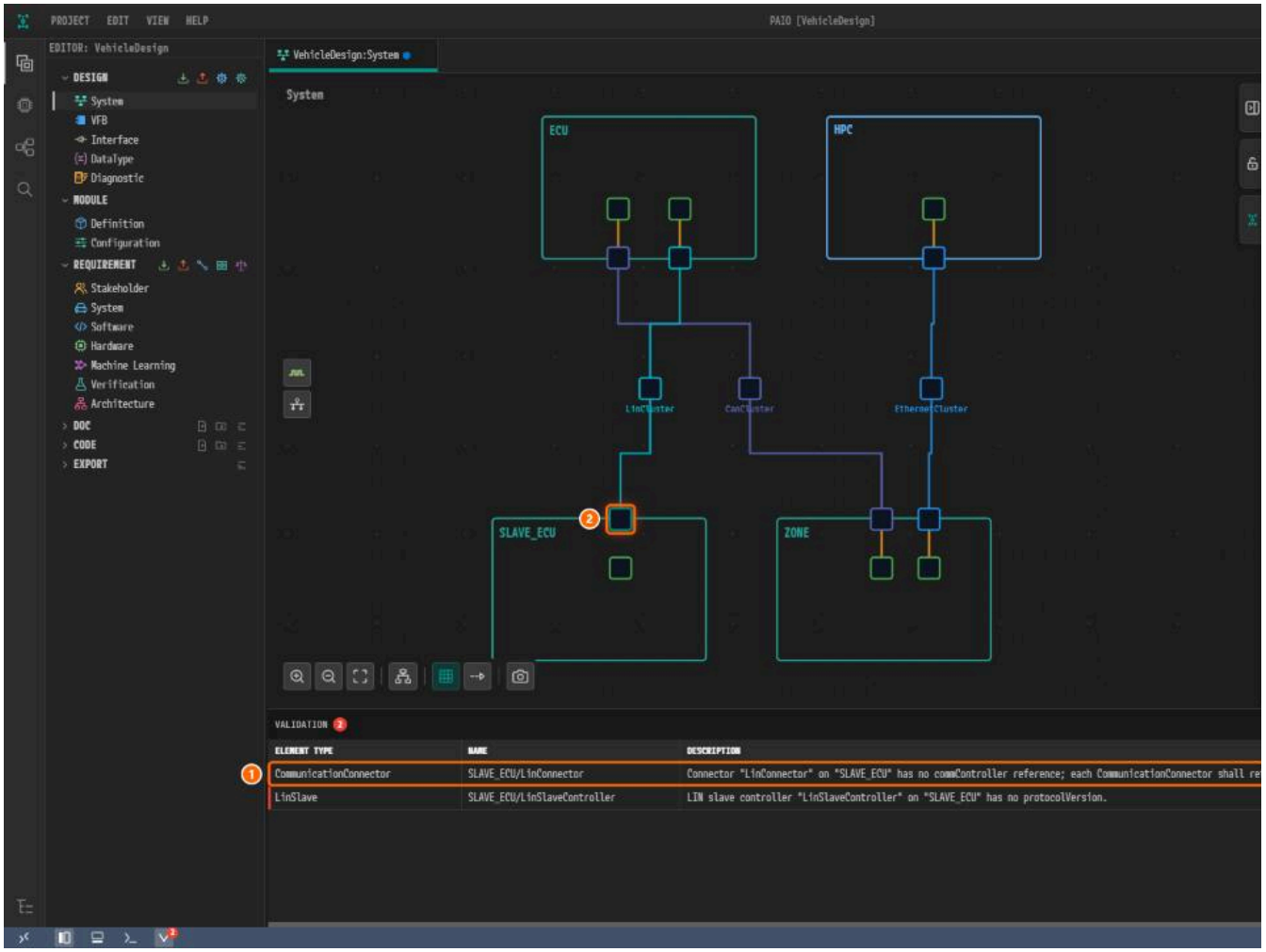


*그림 6-6. ① `constr_5398` 행, ② `SLAVE_ECU` 의 커넥터, ③ **CONTROLLER**.*

2. **CONTROLLER** 칸을 클릭한다.

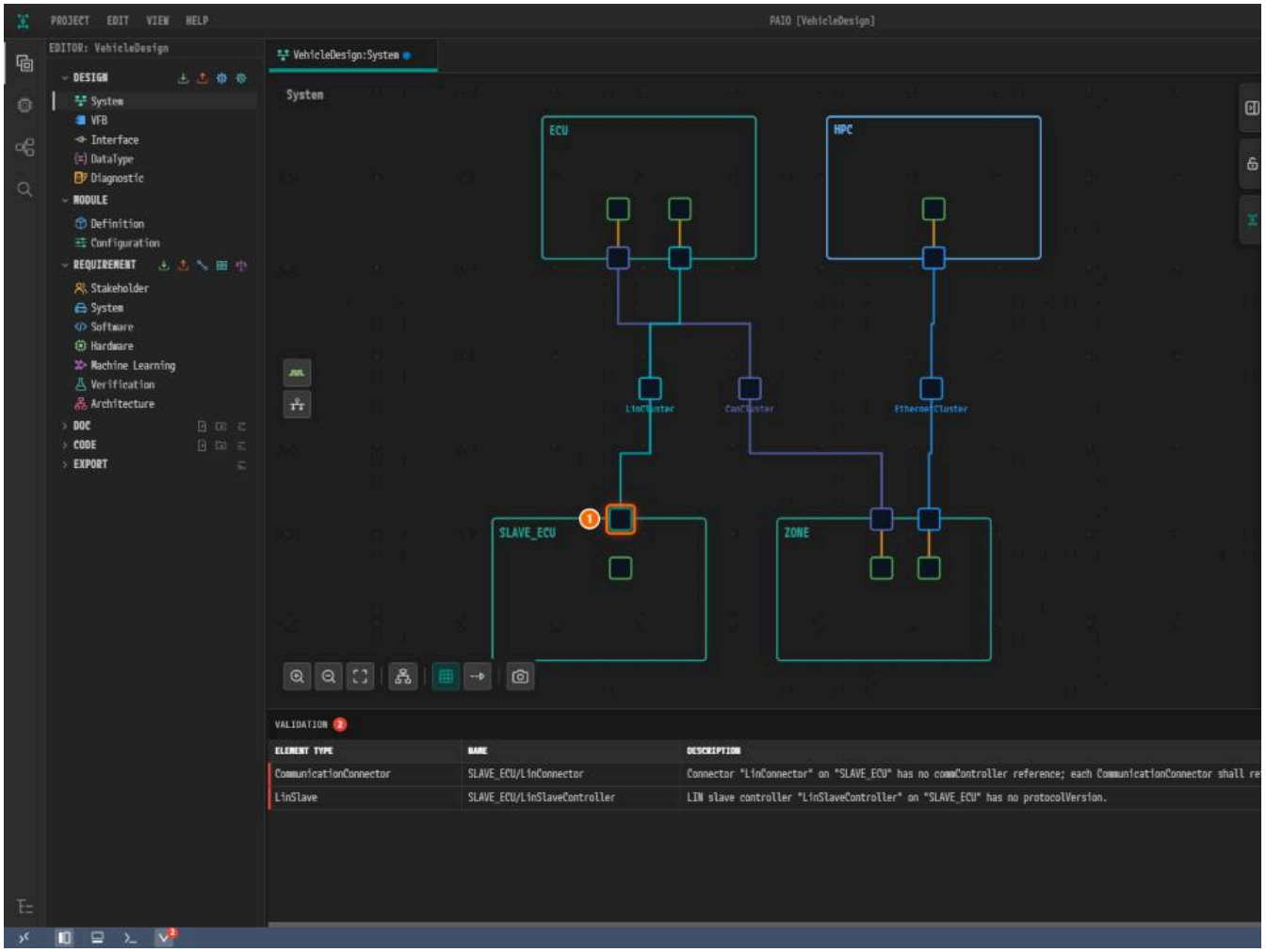


*그림 6-7. ① 커넥터, ② **CONTROLLER**, ③ `LinSlaveController`.*

3. `LinSlaveController` 를 클릭한다. 패널이 한 행으로 줄어든다.

LIN 슬레이브 컨트롤러의 프로토콜 버전 고르기

1. `constr_5449` 행을 클릭한다. 컨트롤러가 아니라 노드 `SLAVE_ECU`(패널 **ECU**)가 선택된다.

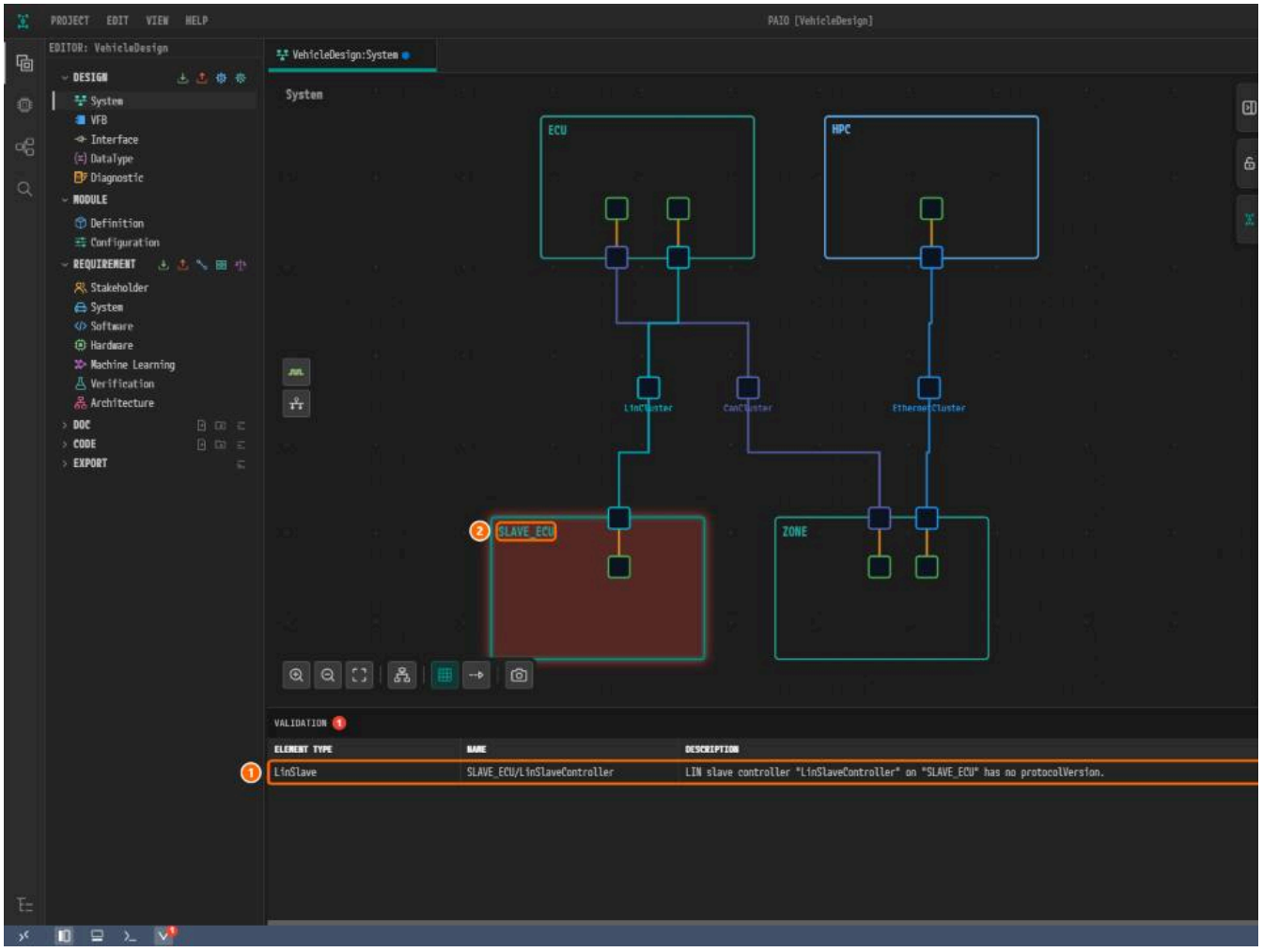


*그림 6-8. ① `constr_5449` 행, ② 노드 `SLAVE_ECU`, ③ 패널 제목 **ECU**.*

2. 노드 안의 컨트롤러 `LinSlaveController` 를 클릭하고 **PROTOCOL VERSION** 의 `LIN 2.0` 을 클릭한다. 패널에 `No validation issues.` 가 보인다.

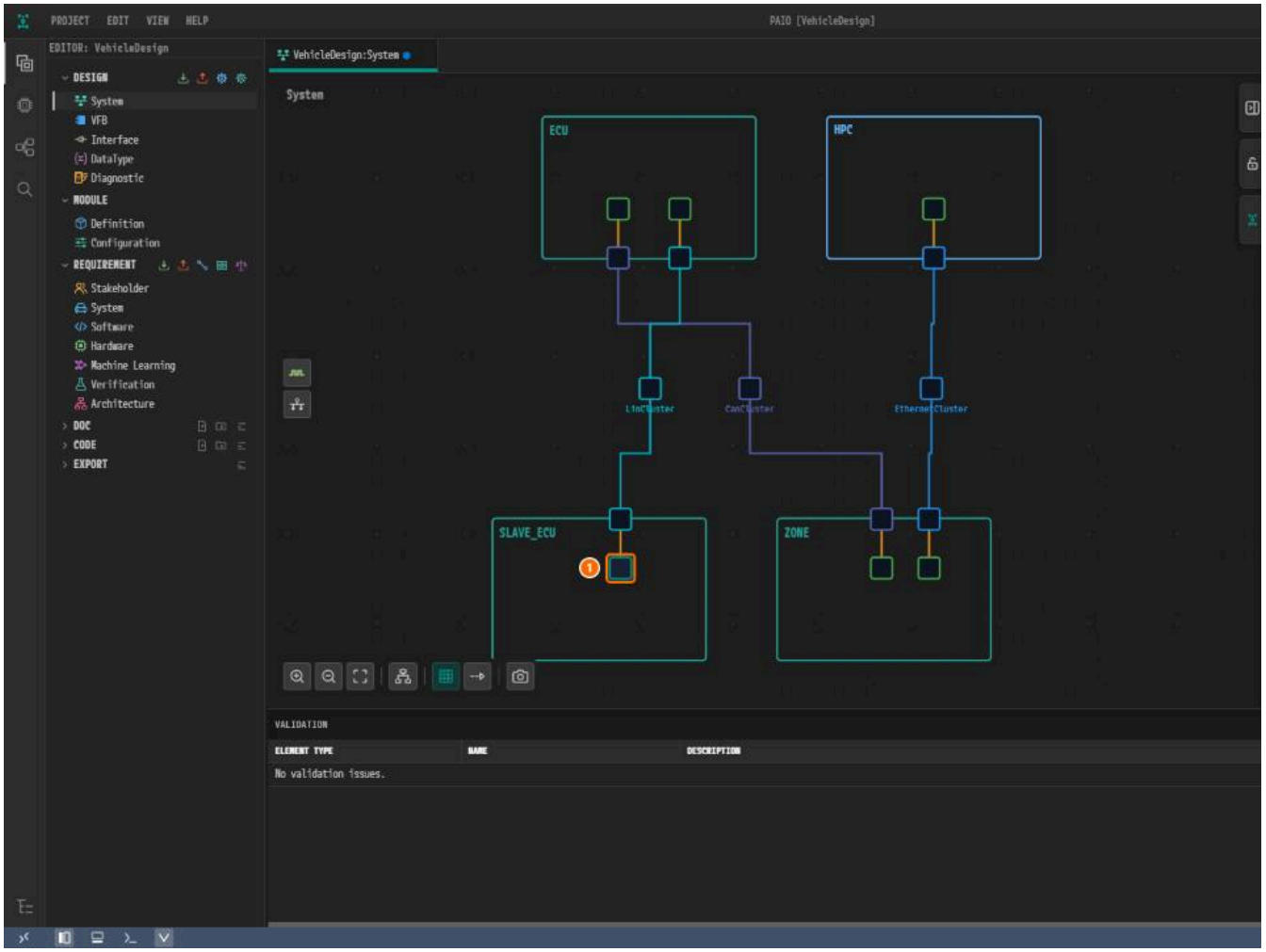


*그림 6-9. ① 컨트롤러, ② **PROTOCOL VERSION** = `LIN 2.0`.*

저장하고 다시 검증하기

1. 캔버스의 빈 곳을 클릭하고 `Ctrl+S` 를 누른다. 패널에는 여전히 `No validation issues.` 가 보이고 배지는 없다.

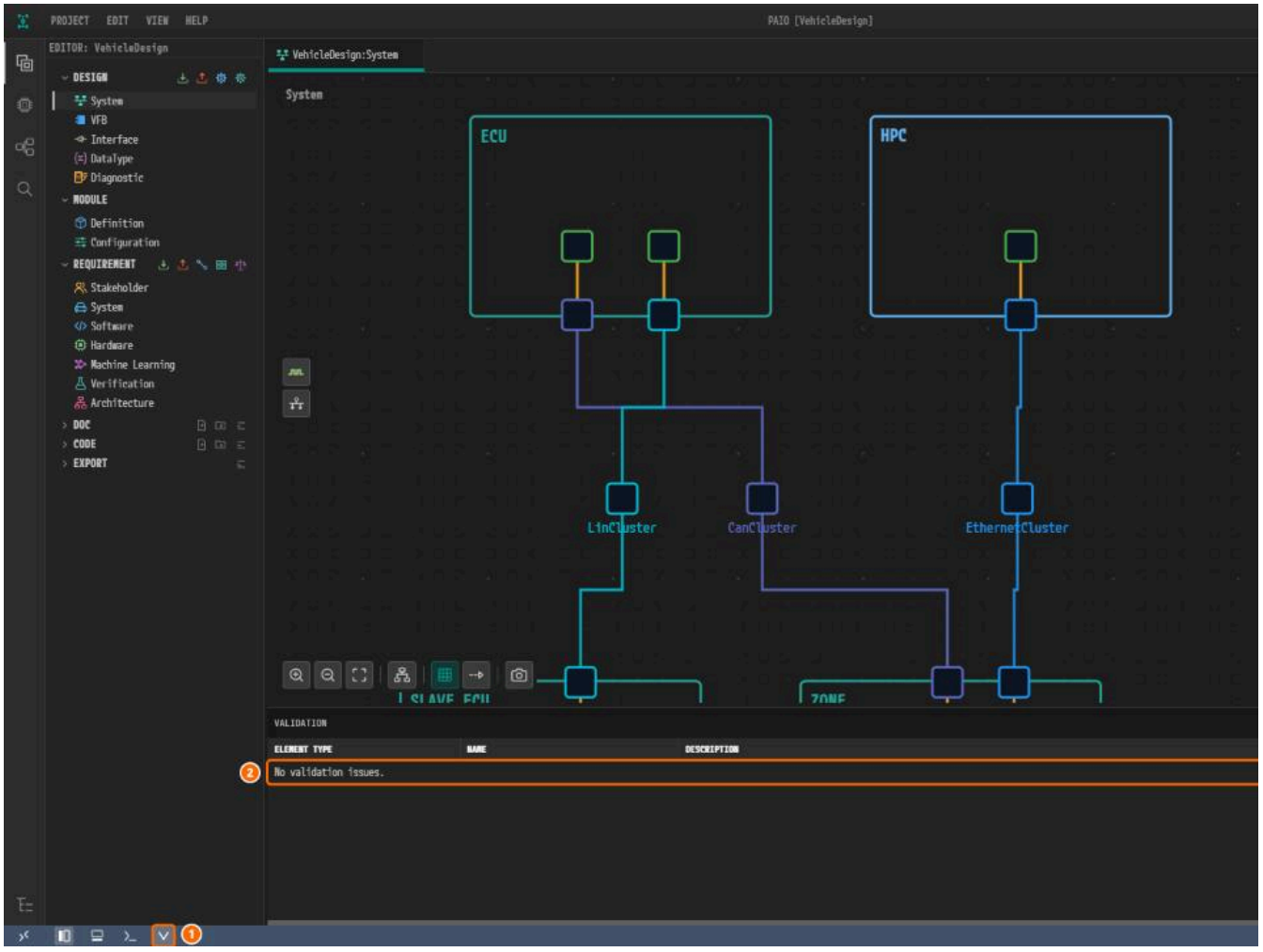


*그림 6-10. ① 배지가 없는 **V** 아이콘, ② `No validation issues.`. 캔버스 배율은 **Fit to View** 전으로 돌아갔지만 설계는 바뀌지 않았다.*

왜 이렇게 설계하는가

- **유니캐스트 주소는 첫 옥텟이 223 이하여야 한다.** `224.0.0.1` 은 ETHERNET 설정에서 넣은 SOME/IP 서비스 디스커버리용 멀티캐스트 주소이고, `HPC` 자신의 주소는 `192.`
- **커넥터는 컨트롤러를 정확히 하나 가리켜야 한다.** 커넥터는 컨트롤러를 통해 버스에 붙는다.
- **LIN 컨트롤러에는 프로토콜 버전이 필요하다.** `LIN 2.0` 은 참조 설계의 `LIN20` 에 해당한다.

확인하기

- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보이고, **V** 아이콘에 배지가 없다.
- [ ] `HPC` 의 `EthernetConnector` 의 **UNICAST IPV4 ADDRESS** 가 `192.168.0.11` 이고, `ZONE` 의 것은 여전히 `192.168.0.19` 이다.
- [ ] `SLAVE_ECU` 의 `LinConnector` 의 **CONTROLLER** 가 `LinSlaveController` 이고, `LinSlaveController` 의 **PROTOCOL VERSION** 이 `LIN 2.0` 이다.
- [ ] 탭 `VehicleDesign:System` 에 수정됨 표시가 없다.

다음 단계

ARXML 익스포트에서 토폴로지를 ARXML 파일로 익스포트하고 참조 설계와 비교한다.

ARXML 익스포트

이 장에서 할 일

토폴로지를 프로젝트 `export` 폴더의 ARXML 파일 네 개로 익스포트한다. 그다음 화면에서 넣은 값이 어떻게 나왔는지 XML 편집기로 확인하고, 익스포트가 참조 설계 `System_De`

시작하기 전에

- 검증을 마친 뒤 시작한다. **VALIDATION** 패널에 `No validation issues.` 가 보인다. 푸터의 **V** 아이콘으로 패널을 닫는다.
- `export` 폴더는 비어 있다. 탭 `VehicleDesign:System` 을 × 로 닫아 편집기가 하나도 열려 있지 않게 한다.

찾을 것	파일	찾기 문자열	찾기 창	줄
HPC 노드 `HPC`	`VehicleDesign_COM.arxml`	`<MACHINE-DESIGN>`	`1 of 1`	109
이더넷 엔드포인트 세 개	`VehicleDesign_COM.arxml`	`<NETWORK-ENDPOINTS>`	`1 of 1`	191
VLAN `VLAN_13`	`VehicleDesign_COM.arxml`	`<VLAN>`	`1 of 1`	237

프로젝트 익스포트하기

1. **EDITOR** 뷰에서 **DESIGN** 머리글 오른쪽의 두 번째 아이콘을 가리킨다. 툴팁은 **Export** 이다.

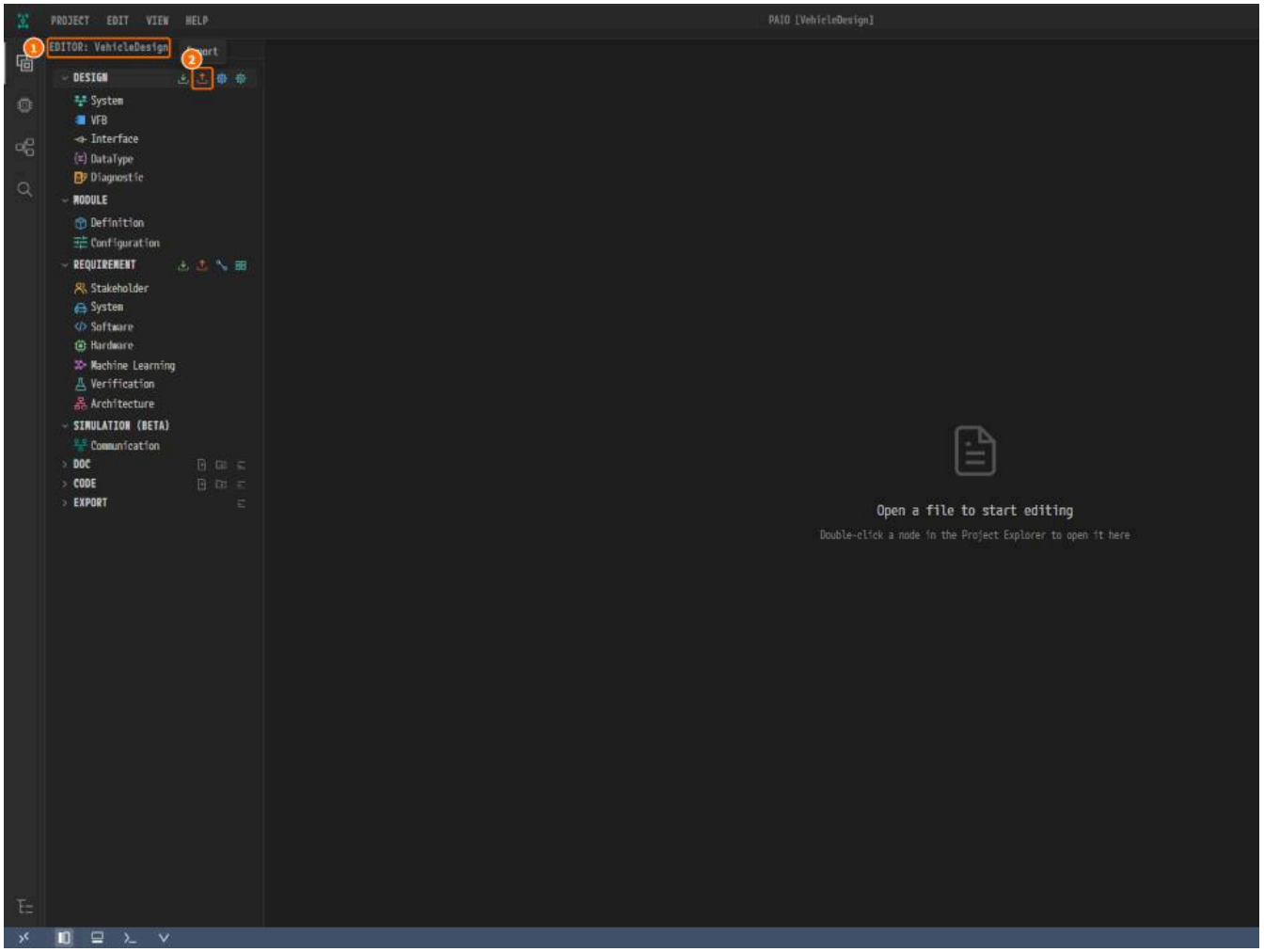


*그림 7-1. ① 뷰 머리글 `EDITOR: VehicleDesign`, ② **Export** 아이콘과 툴팁.*

2. **Export** 를 클릭한다. **Export** 창이 열린다. **FOLDER** 는 `export`, **FORMAT** 은 **ARXML (R25-11)** 그대로 두고 **Export** 를 클릭한다. 창에 `✓ Complete!` 와 **ARXML** `R25-11`

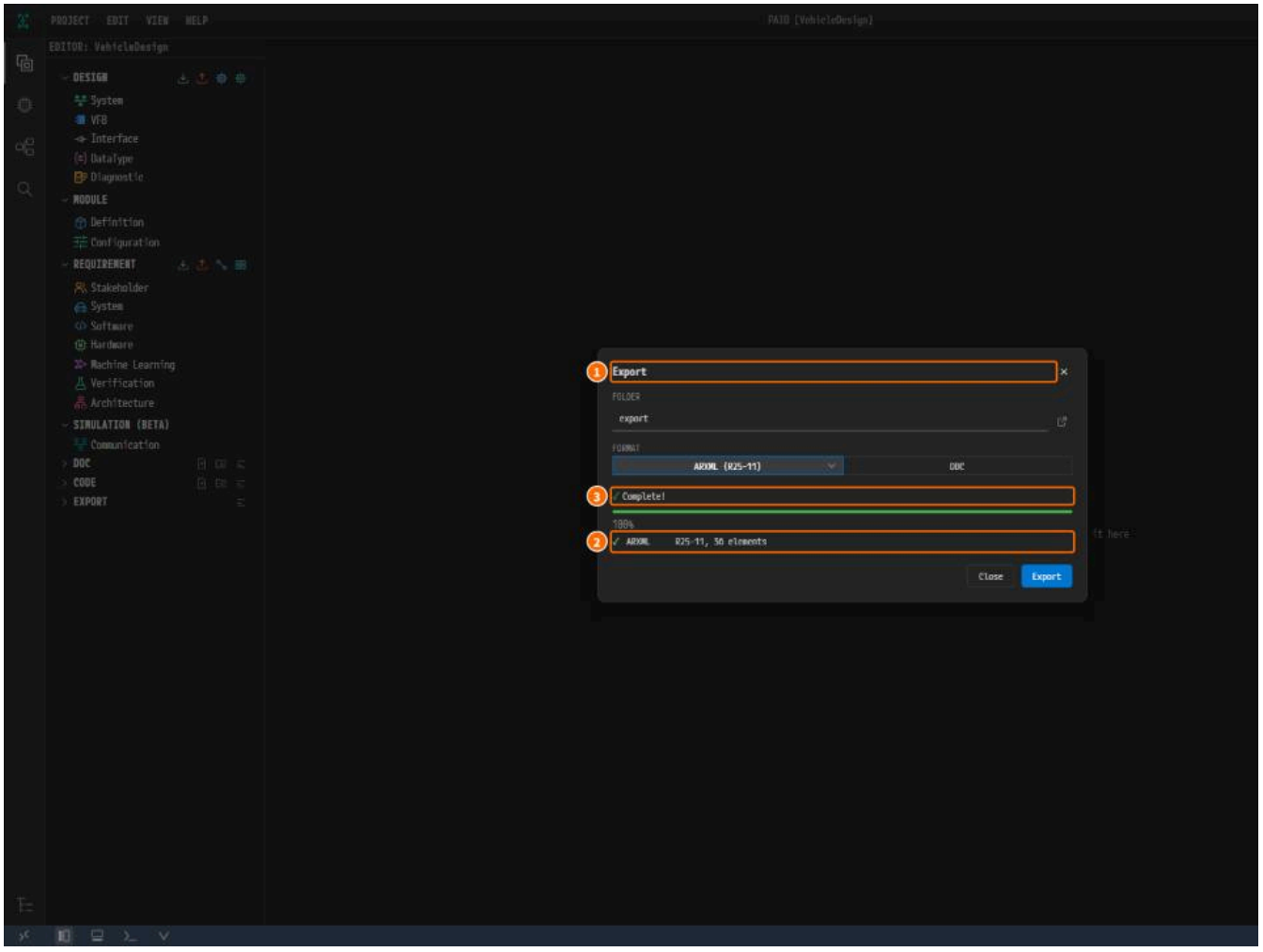


*그림 7-2. ① **Export**, ② 결과 줄, ③ `✓ Complete!`.*

3. **Close** 를 클릭한다.

> **참고.** 익스포트는 **FOLDER** 의 폴더에 `<프로젝트 이름>_*.arxml` 을 쓰며, `design` 의 파일은 바꾸지 않는다.

export 폴더의 파일 네 개 보기

1. **EXPORT** 머리글을 클릭해 펼친다. 파일 네 개가 보인다.

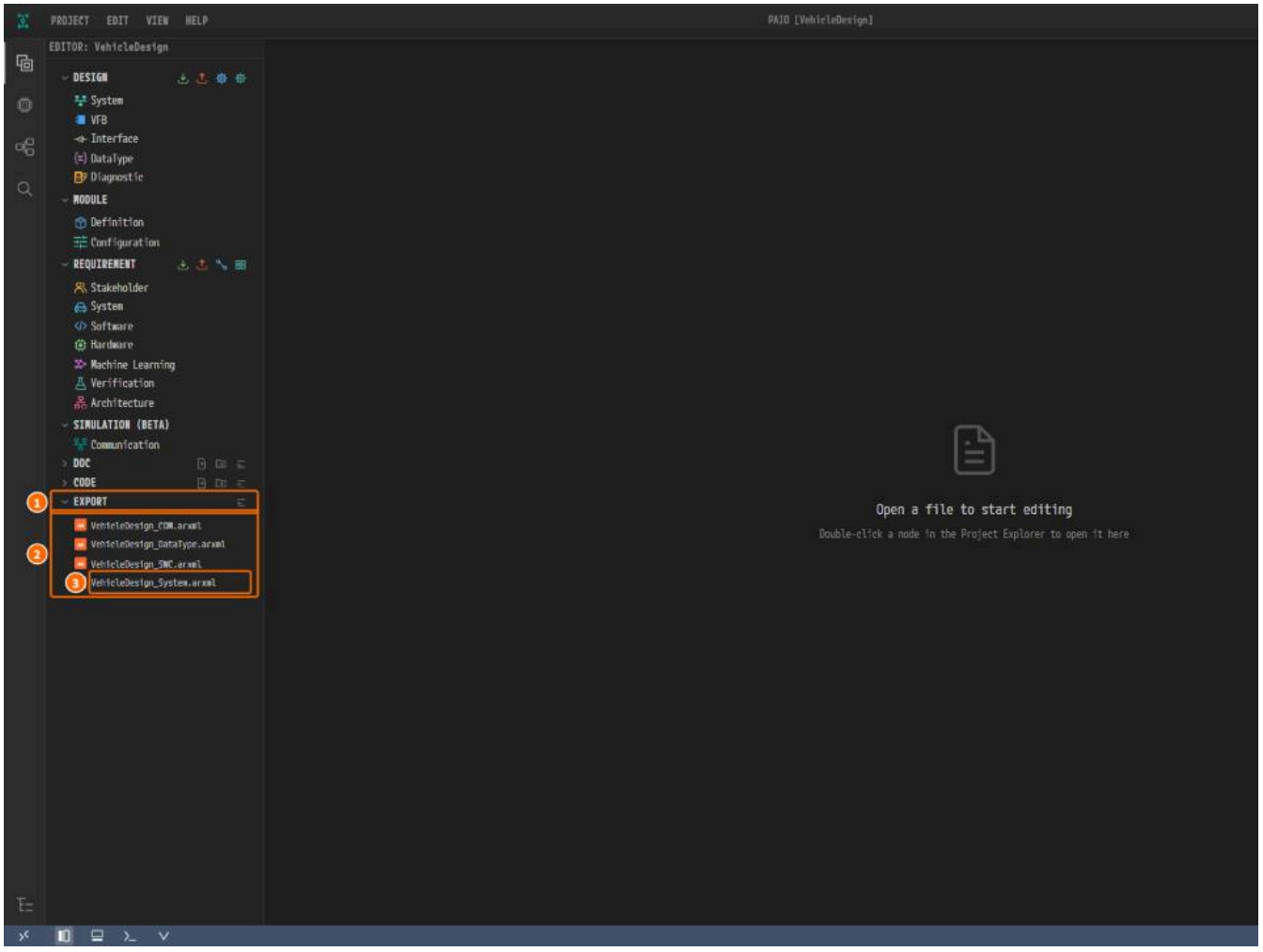


*그림 7-3. ① **EXPORT**, ② 파일 네 개, ③ 다음에 열 `VehicleDesign_System.arxml`.*

파일	담긴 것
`VehicleDesign_COM.arxml`	노드 `ECU`, `ZONE`, `SLAVE_ECU`, `HPC` 와 그 컨트롤러·커넥터, 클러스터 세 개, 엔드포인트, VLAN
`VehicleDesign_DataType.arxml`	PAIO 가 모든 익스포트에 넣는 기본 데이터 타입. 이 설계는 쓰지 않는다
`VehicleDesign_SWC.arxml`	빈 패키지 `SwComponentType` 하나
`VehicleDesign_System.arxml`	시스템 `System`

시스템 System 보기

1. **EXPORT** 에서 `VehicleDesign_System.arxml` 을 더블클릭한다. 파일이 XML 편집기에 읽기 전용으로 열린다.
2. 상태 표시줄에 `Schema: R25-11` 과 `✓ No problems` 가 보이는지 확인한다. 편집기는 파일을 열 때 AUTOSAR 스키마로 검사한다.

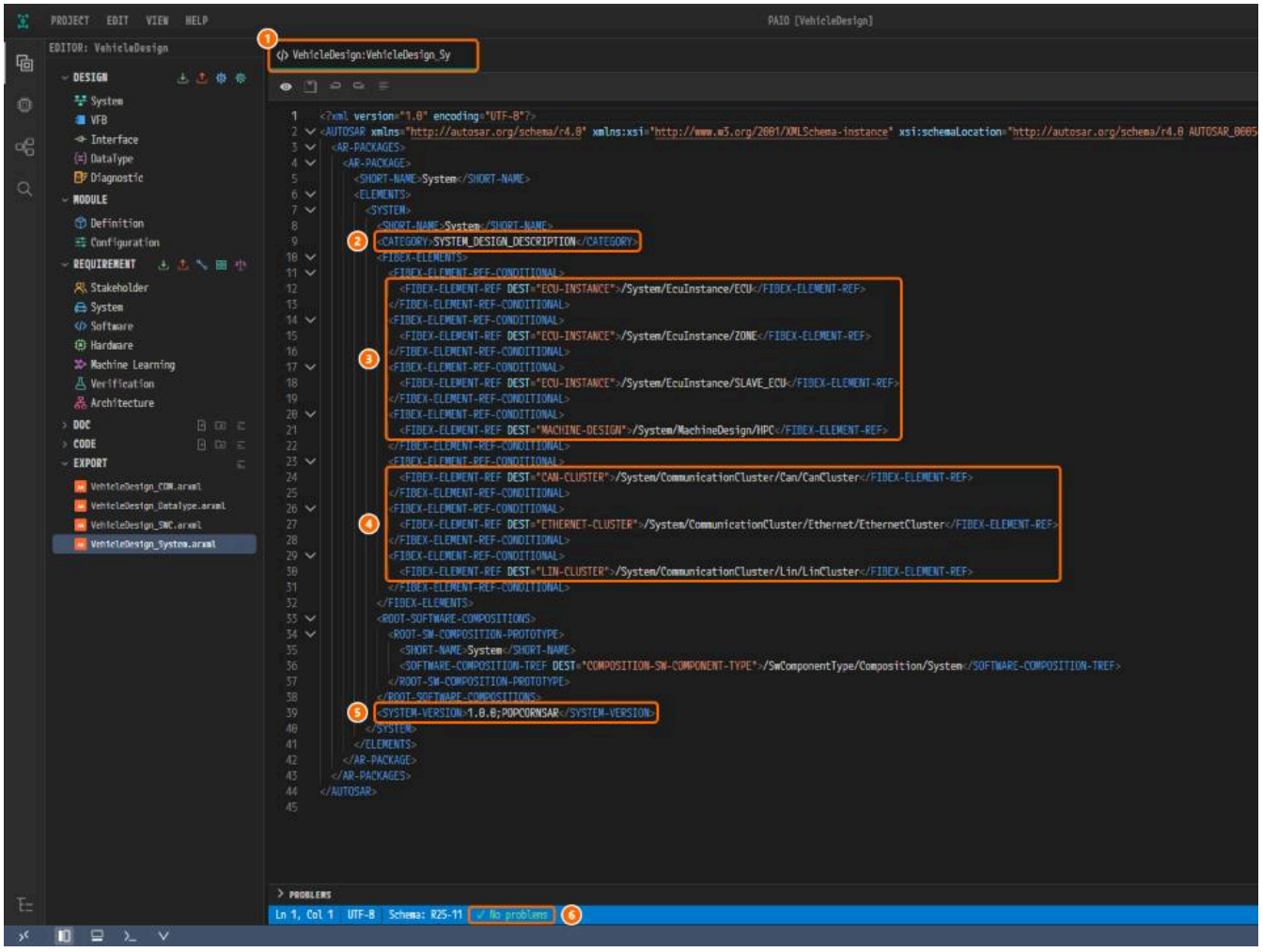


그림 7-4. 1-45행. ① 편집기 탭, ② 9행 `SYSTEM_DESIGN_DESCRIPTION`, ③ 12-21행 노드 네 개, ④ 24-30행 클러스터 세 개, ⑤ 39행 `SYSTEM-VERSION` `1.0.0;POPCU`

줄	내용	나온 곳
9	`CATEGORY` `SYSTEM_DESIGN_DESCRIPTION`	익스포트가 정하는 값
12-21	`ECU`, `ZONE`, `SLAVE_ECU`, 그다음 `HPC` 의 참조	캔버스의 노드. ECU 노드는 추가한 순서이다(프로젝트와 노드 만들기)
24-30	`CanCluster`, `EthernetCluster`, `LinCluster` 의 참조	CAN 설정, ETHERNET 설정, LIN 설정
33-38	루트 컴포지션 `System`	입력 없음. 네 파일 어디에도 없는 컴포지션을 가리킨다. 스키마 검사는 참조를 따라가지 않는다
39	`SYSTEM-VERSION` `1.0.0;POPCORNSAR`	ECU 속성의 **VERSION**

ECU 노드 ECU 보기

1. **EXPORT** 에서 `VehicleDesign_COM.arxml` 을 더블클릭한다. 상태 표시줄은 `✓ No problems` 이다.
2. 10행 `<ECU-INSTANCE>` 가 맨 위에 오도록 아래로 굴린다.

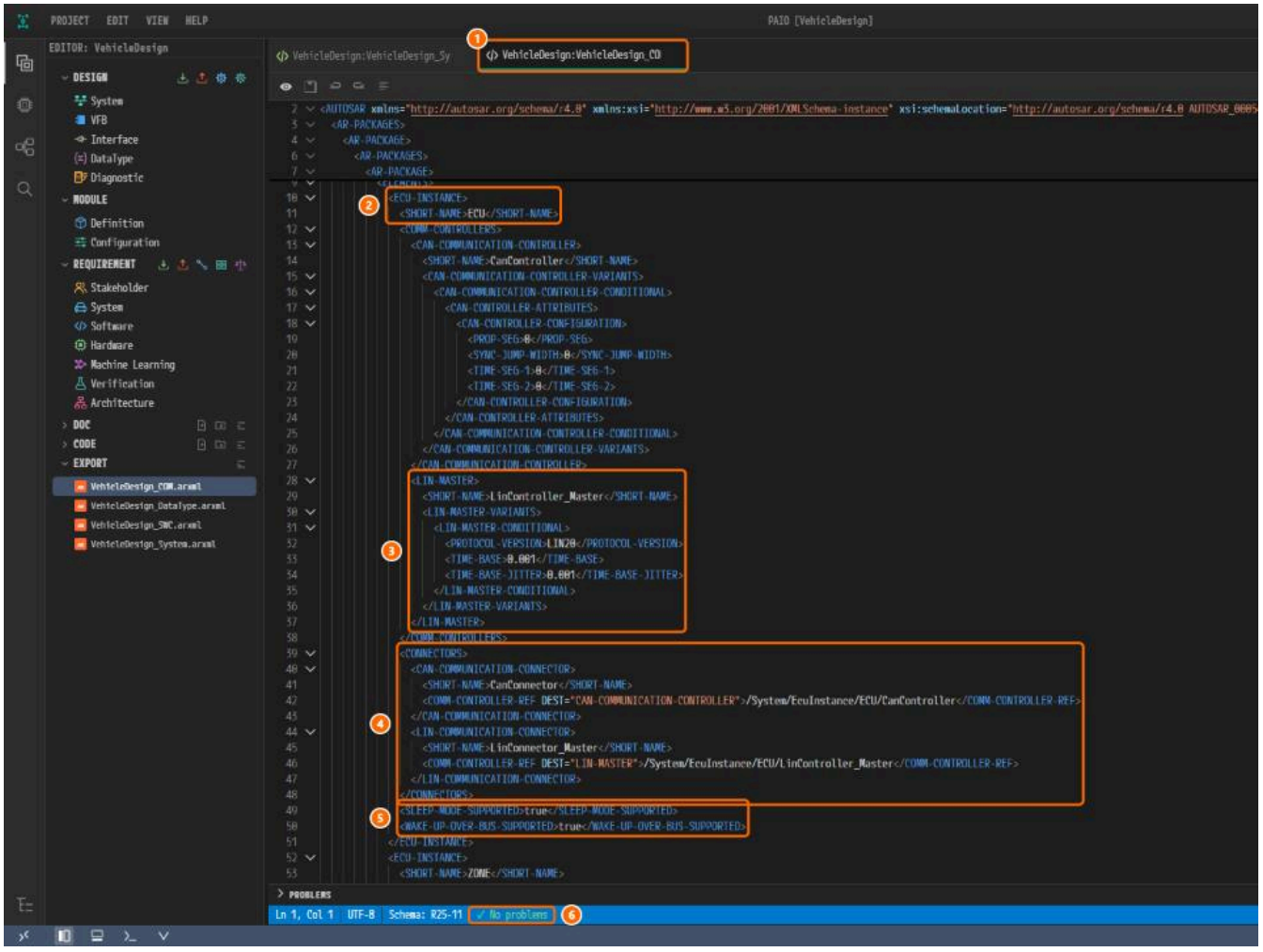


그림 7-5. 10-51행. ① 활성 탭, ② `ECU-INSTANCE` `ECU`, ③ `LinController_Master`, ④ `CONNECTORS`, ⑤ `SLEEP-MODE-SUPPORTED`, `WAKE-UP-OVER-BUS-SUPPO` 상위 요소를 고정해 보여 주는 머리글이다.

줄	내용	나온 곳
13-27	`CanController`, 비트 타이밍 네 값 `0`	CAN 설정
28-37	`LinController_Master`: `LIN20`, `TIME-BASE`·`TIME-BASE-JITTER` `0.001`	LIN 설정의 **PROTOCOL VERSION** `LIN 2.0`, `1` `ms`, `1` `ms`
40-47	`CanConnector` → `CanController`, `LinConnector_Master` → `LinController_Master`	커넥터의 **CONTROLLER**
49-50	`SLEEP-MODE-SUPPORTED`, `WAKE-UP-OVER-BUS-SUPPORTED` `true`	ECU 속성

`ZONE`(52-82행)과 `SLAVE_ECU`(83-103행)도 같은 꼴이다. 참조 설계에 있는 `ECU`의 파티션 두 개는 없다. System 편집기에 넣는 칸이 없다.

HPC 노드 HPC 보기

1. 편집기 안을 클릭하고 `Ctrl+F`를 누른다.
2. `<MACHINE-DESIGN>`을 입력한다. 찾기 창에 `1 of 1`이 보이고 109행이 선택된다.
3. 109행이 맨 위에 오도록 아래로 굴린다.

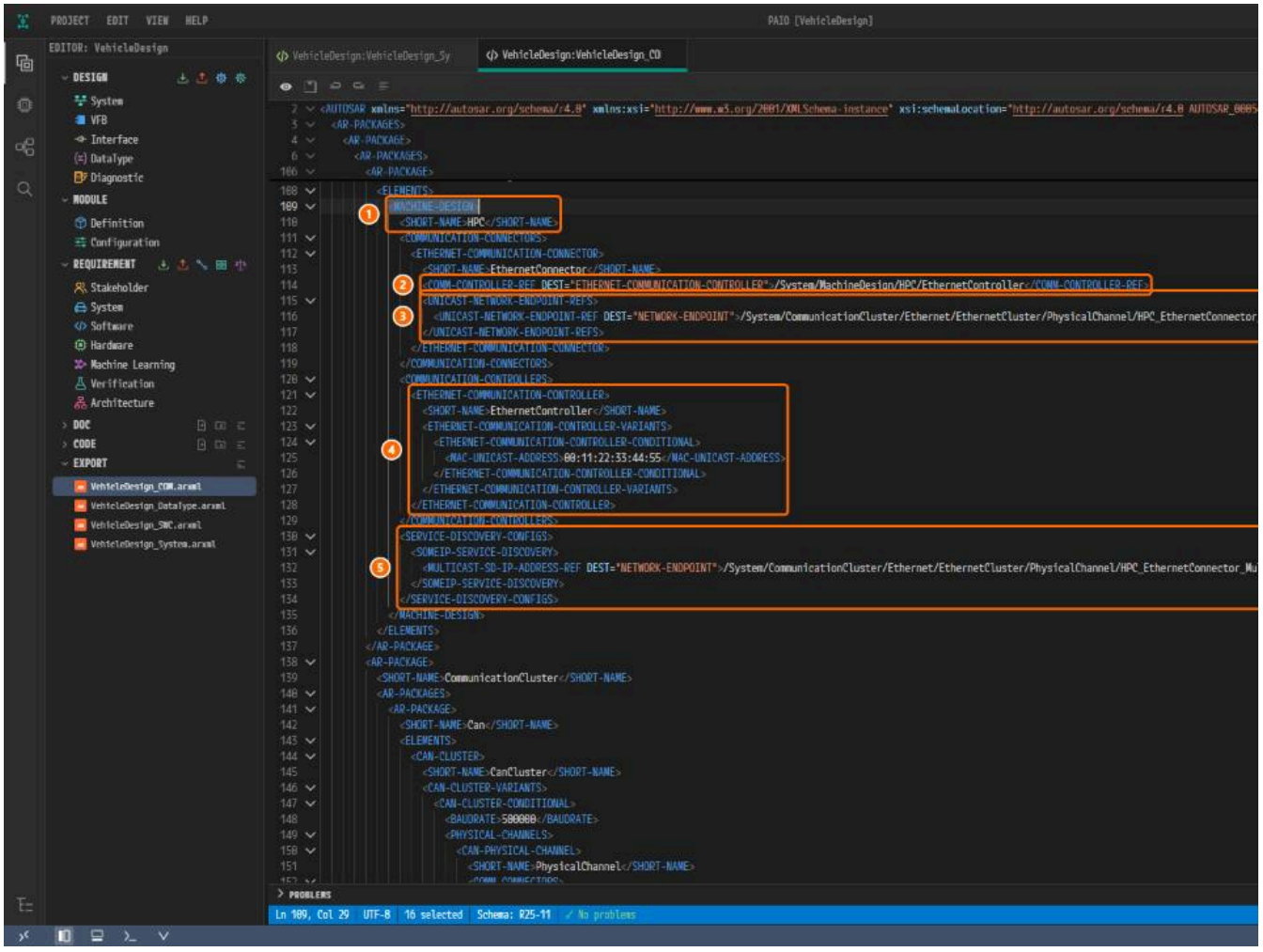


그림 7-6. ① `MACHINE-DESIGN` `HPC`, ② 114행 `COMM-CONTROLLER-REF`, ③ 115-117행 `HPC_EthernetConnector_Endpoint`, ④ `EthernetController` `00:11.` ⑥ 찾기 창.

HPC 노드는 `ECU-INSTANCE` 가 아니라 `MACHINE-DESIGN` 으로 나간다. 114행은 `/System/MachineDesign/HPC/EthernetController` 를, 116행은 `HPC_EthernetCon` 튜토리얼이 참조 설계를 바로잡은 곳이다(아래 "HPC 커넥터를 바로잡은 곳" 참고). 커넥터에 **MULTICAST IPV4 ADDRESS** 가 있으므로 익스포트가 `SERVICE-DISCOVERY-CONFI`

이더넷 엔드포인트 보기

1. 찾기 문자열을 `<NETWORK-ENDPOINTS>` 로 바꾼다. 찾기 창에 `1 of 1` 이 보이고 191행이 선택된다.

2. 191행이 맨 위에 오도록 아래로 굴린다.

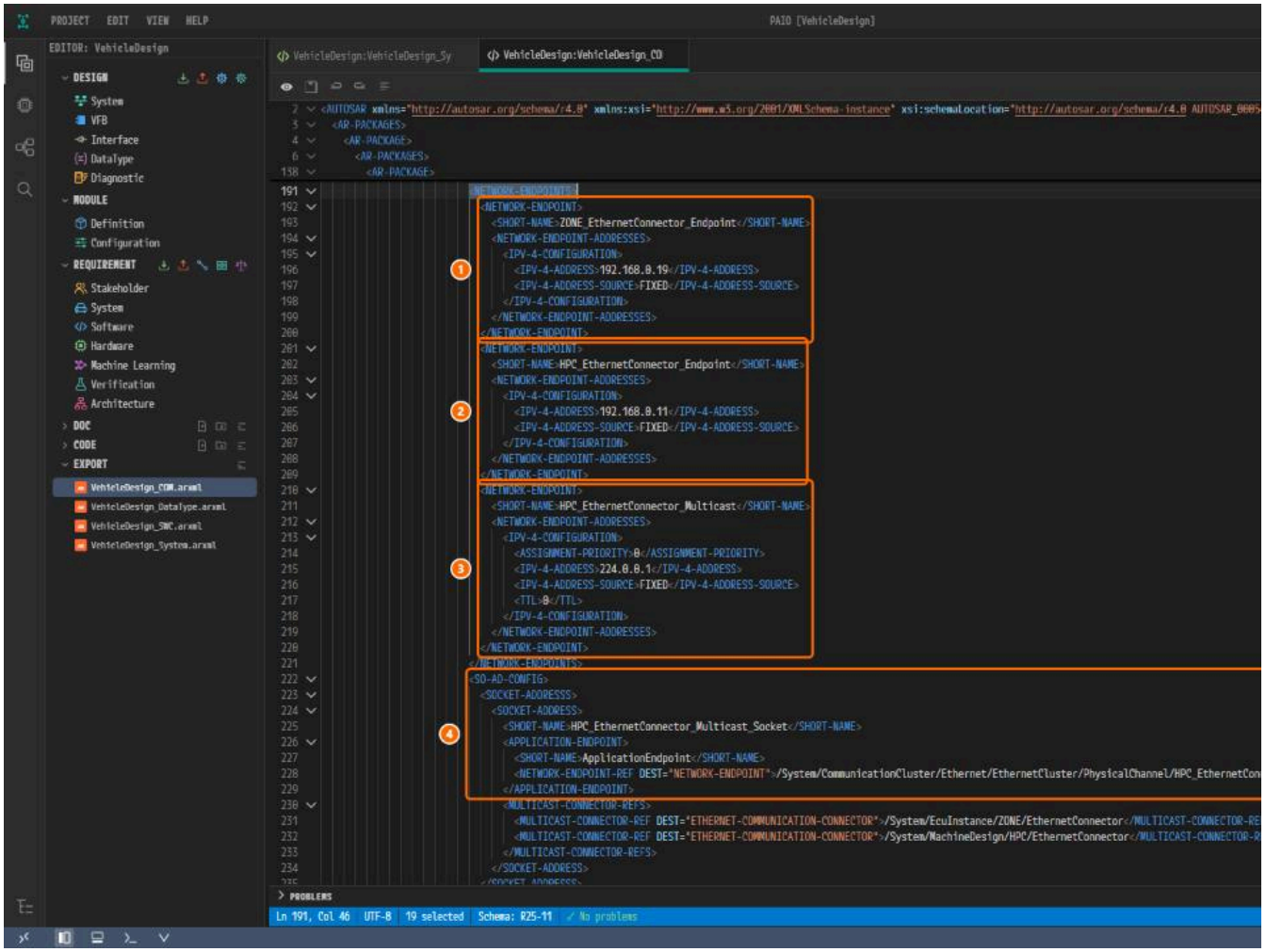


그림 7-7. ① `ZONE_EthernetConnector_Endpoint(192.168.0.19)`, ② `HPC_EthernetConnector_Endpoint(192.168.0.11)`, ③ `HPC_EthernetConnecti` 소켓.

엔드포인트 세 개는 모두 `FIXED` 이다. 이름은 넣지 않는다. 익스포트가 `<노드>_<커넥터>_Endpoint`, `<노드>_<커넥터>_Multicast` 로 이름을 붙이고, 멀티캐스트 주소에 `ASS`

VLAN 과 LIN 클러스터 보기

1. 찾기 문자열을 `<VLAN>` 으로 바꾼다. 찾기 창에 `1 of 1` 이 보이고 237행이 선택된다.

2. 237행이 맨 위에 오도록 아래로 굴린다.

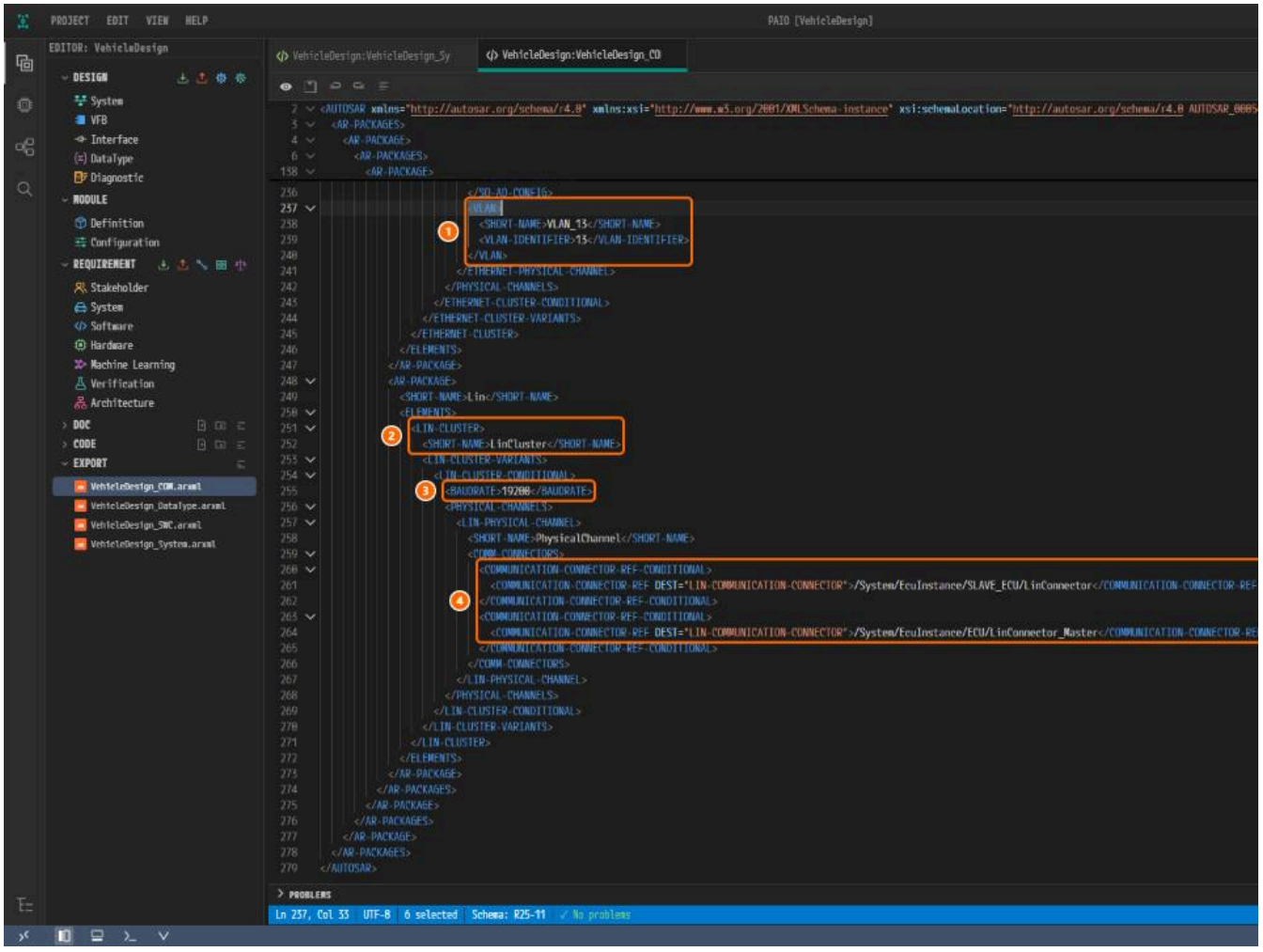


그림 7-8. ① `VLAN_13`, `VLAN-IDENTIFIER` `13`, ② `LIN-CLUSTER` `LinCluster`, ③ `BAUDRATE` `19200`, ④ `SLAVE_ECU/LinConnector` 와 `ECU/LinConnector`

3. `Escape` 키를 눌러 찾기 창을 닫는다.

VLAN 이름은 익스포트가 `VLAN_<ID>` 로 붙인다. `BAUDRATE` `19200` 은 PAIO 가 새 LIN 클러스터에 저장하는 값이고, **IDLE TIMEOUT** 을 비워 두었으므로 LIN 채널에 `BUS-IDLE-` 에는 `BAUDRATE` `500000` 과 CAN 설정의 `BUS-OFF-RECOVERY` 값이 있다.

HPC 커넥터를 바로잡은 곳

항목	참조 설계	내 익스포트
`COMM-CONTROLLER-REF`	없음	`/System/MachineDesign/HPC/EthernetController` (114행)
유니캐스트 엔드포인트	`ZONE_Endpoint` = `192.168.0.19`, `ZONE` 의 주소	`HPC_EthernetConnector_Endpoint` = `192.168.0.11` (116행)

- 참조 설계의 커넥터에는 컨트롤러가 없고, 두 이더넷 커넥터가 `192.168.0.19` 를 함께 쓰며 `HPC_Endpoint` 는 쓰이지 않는다. 이 튜토리얼은 ETHERNET 설정에서 둘 다 바
- 검증은 이 결함을 알리지 않는다. `constr_5398` 은 ECU 노드의 커넥터만 검사한다.

내 익스포트와 참조 설계의 차이

참조 설계는 이름이 있는 요소 29개의 파일 하나이고, 내 익스포트는 79개의 파일 네 개이다. 짝이 있는 요소는 27개이다. 요소째 빠지는 것은 파티션 두 개뿐이다.

차이	
노드는 `/System/EcuInstance`(`HPC` 는 `/System/MachineDesign`), 클러스터는 `/System/CommunicationCluster/Can`, `/Ethernet`, `/Lin` 으로 옮겨짐	익
`CATEGORY` `SYSTEM_DESCRIPTION` → `SYSTEM_DESIGN_DESCRIPTION`, 참조의 순서가 다름	익
`ZONE_Endpoint`, `HPC_Endpoint`, `SOMEIP_SD_Multicast`, `Vlan_13` → `ZONE_EthernetConnector_Endpoint`, `HPC_EthernetConnector_Endpoint`, `HPC_EthernetConnector_Multicast`, `VLAN_13`	익
MAC 주소 `11:22:EE:44:55:66` → `11:22:ee:44:55:66`, `CanCommController` 에 빈 `CAN-COMMUNICATION-CONTROLLER-CONDITIONAL` 이 없음	이
`ECU_Partition0`, `ECU_Partition1` 이 없음	호
`CATEGORY` `WIRED`(이더넷 채널, 컨트롤러, 커넥터), `PROTOCOL-NAME` `IEEE 802.3`, `HPC` 의 `WAKE-UP-BY-CONTROLLER-SUPPORTED` `false` 가 없음	호
LIN 채널의 `BUS-IDLE-TIMEOUT-PERIOD` `0.01` 이 없음	**I**
`MAC-LAYER-TYPE` `X-MII` 가 없음	튜 틀
`LinController_Master` 의 `WAKE-UP-BY-CONTROLLER-SUPPORTED` `true`, `PROTOCOL-NAME` `CAN-FD`, `HPC` 커넥터의 `CREATE-ECU-WAKEUP-SOURCE` `false` 가 없음	튜 없
`CanCluster` 에 `BUS-OFF-RECOVERY` 가 더 있음	튜
`HPC` 커넥터: 컨트롤러 참조가 더 있고 유니캐스트 엔드포인트가 바뀜	참
`BAUDRATE` `500000`·`19200`, `SERVICE-DISCOVERY-CONFIGS`, SD 소켓(2), `ASSIGNMENT-PRIORITY`·`TTL` `0`, 기본 데이터 타입 49개, 루트 컴포지션 `System`	P
`UUID` 속성이 없음, 파일 구성과 요소 순서가 다름	실

내 익스포트에는 NM 이나 TP 요소가 없다. 참조 설계에도 없다.

왜 이렇게 설계하는가

- **XML 편집기는 열 때 스키마를 검사한다.** `✓ No problems` 는 구조가 맞음을 보여 주지만, 참조 대상이 있는지는 보지 않는다.
- **차이는 대부분 익스포트 규칙이 정하는 위치와 이름이다.**
- `HPC` **커넥터는 참조 설계를 따르지 않는다.** 참조 설계가 그 부분에서 잘못되어 있기 때문이다.

확인하기

- [ ] **EXPORT** 에 `VehicleDesign_COM.arxml`, `VehicleDesign_DataType.arxml`, `VehicleDesign_SWC.arxml`, `VehicleDesign_System.arxml` 이 있다.
- [ ] 연 두 파일의 상태 표시줄이 `Schema: R25-11` 과 `✓ No problems` 이다.
- [ ] `VehicleDesign_System.arxml` 의 39행이 `SYSTEM-VERSION` `1.0.0;POPCORNSAR` 이다.
- [ ] `VehicleDesign_COM.arxml` 의 114행이 `/System/MachineDesign/HPC/EthernetController` 를, 116행이 `HPC_EthernetConnector_Endpoint` 를 가리킨다.
- [ ] 엔드포인트 세 개가 `192.168.0.19`, `192.168.0.11`, `224.0.0.1` 이고, `VLAN_13` 의 `VLAN-IDENTIFIER` 가 `13` 이다.
- [ ] 두 탭 어느 것에도 수정됨 표시 `●` 가 없다.

다음 단계

튜토리얼의 본문은 여기까지이다. 이어서 부록을 본다.

부록

이 장에서 할 일

이 부록은 따라 하는 장이 아니라 찾아보는 자료이다. 내 설계를 점검하거나 어느 값이 어느 장에 속하는지 찾을 때 쓴다.

시작하기 전에

- 여기의 값은 1~7장과 같다. 다르게 보이면 해당 장을 따른다.

완성된 설계 점검표

노드와 시스템

- [ ] System 편집기에 `ECU`, `ZONE`, `SLAVE_ECU`(패널 제목 **ECU**)와 `HPC`(패널 제목 **HPC**)가 있다.
- [ ] ECU 노드 세 개는 **SLEEP MODE SUPPORTED** 와 **WAKE-UP OVER BUS** 가 켜져 있다. `HPC` 에는 **POWER MANAGEMENT** 섹션이 없다.
- [ ] SYSTEM 패널의 **VERSION** 이 `1.0.0;POPCORNSAR` 이고, **ECU MAPPING** 에 노드 네 개가 있다.

CAN 버스

- [ ] `ECU` 에 `CanController` 와 `CanConnector`, `ZONE` 에 `CanCommController` 와 `CanConnector` 가 있다.
- [ ] `CanCluster` 의 **Connected Connectors (2)** 가 `ZONE/CanConnector`, `ECU/CanConnector` 이다.
- [ ] `CanCluster` 의 **BUS-OFF RECOVERY** 가 `100` `ms`, `200` `ms`, `2`, `100` `ms` 이다.

이더넷 버스

- [ ] `ZONE`: `EthernetCommController`(`11:22:ee:44:55:66`), `EthernetConnector`(`192.168.0.19`, **FIXED**).
- [ ] `HPC`: `EthernetController`(`00:11:22:33:44:55`), `EthernetConnector`(`192.168.0.11`, **FIXED**, 멀티캐스트 `224.0.0.1`).
- [ ] 두 컨트롤러의 **MAC LAYER TYPE** 이 꺼져 있다.
- [ ] `EthernetCluster` 의 **VLAN ID** 가 `13` 이고, `ZONE/EthernetConnector`, `HPC/EthernetConnector` 가 연결되어 있다.

LIN 버스

- [ ] `ECU`: `LinController_Master`(`LIN 2.0`, `1` `ms`, `1` `ms`), `LinConnector_Master`.
- [ ] `SLAVE_ECU`: `LinSlaveController`(`LIN 2.0`), `LinConnector`.
- [ ] `LinCluster` 의 **IDLE TIMEOUT** 이 빈 칸이고, `SLAVE_ECU/LinConnector`, `ECU/LinConnector_Master` 가 연결되어 있다.

검증과 익스포트

- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.
- [ ] `export` 폴더에 ARXML 파일 네 개가 있고, XML 편집기에서 `✓ No problems` 이다.
- [ ] 어느 탭에도 수정됨 표시가 없다.

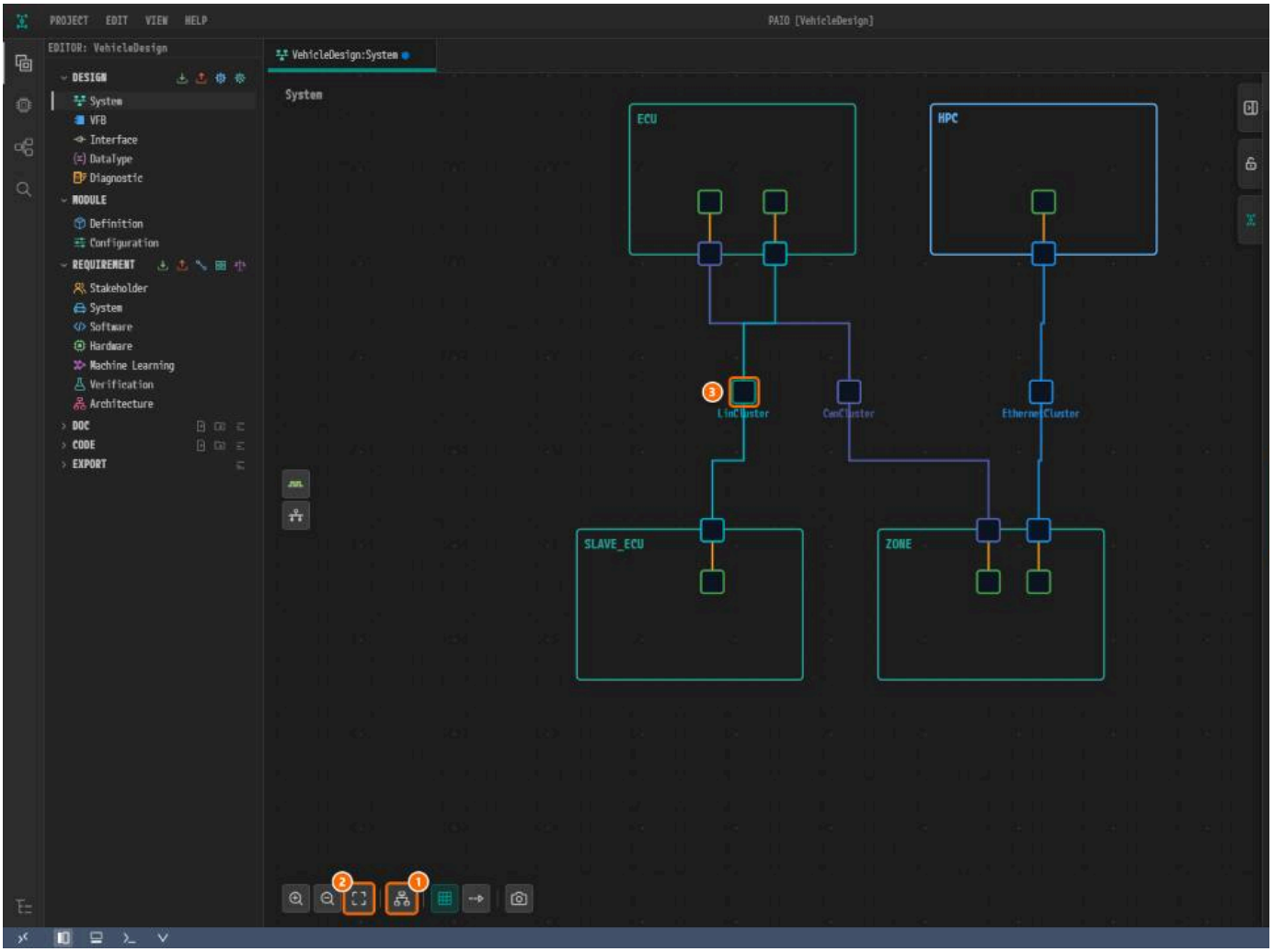


*그림 8-1. 완성된 설계(LIN 설정의 그림 5-14). ① **Auto Layout**, ② **Fit to View**, ③ `LinCluster`, ④ **CHANNEL NAME**, 빈 칸인 **IDLE TIMEOUT**, **Connected Connectors (2)**.*

값 표 모음

프로젝트와 시스템 (1장, 2장)

항목	값	장
프로젝트 이름	`VehicleDesign`	1
시스템 이름	`System` (새 프로젝트의 기본값, 바꾸지 않음)	1
SYSTEM 패널 **VERSION**	`1.0.0;POPCORNSAR` (기본값 `1.0.0`)	2
SYSTEM 패널 **ECU MAPPING**	`ECU`, `ZONE`, `SLAVE_ECU`, `HPC` (모두 `No composition mapped`, 확인만 함)	2

노드와 전원 관리 (1장, 2장)

추가 순서	오른쪽 클릭 위치	메뉴 항목	기본 이름	NAME	패널 제목	SLEEP MODE SUPPORTED / WAKE-UP OVER BUS
1	캔버스 왼쪽 위	**Add ECU**	`ECU_0`	`ECU`	**ECU**	켬 / 켬
2	캔버스 오른쪽 위	**Add ECU**	`ECU_0`	`ZONE`	**ECU**	켬 / 켬
3	캔버스 왼쪽 아래	**Add ECU**	`ECU_0`	`SLAVE_ECU`	**ECU**	켬 / 켬
4	캔버스 오른쪽 아래	**Add HPC**	`HPC_0`	`HPC`	**HPC**	섹션 없음

추가 순서 `ECU` → `ZONE` → `SLAVE_ECU` → `HPC` 를 지킨다. 노드는 오른쪽 클릭한 자리에 생기고, **Auto Layout** 은 CAN 설정부터 쓴다.

참조 설계의 파티션	EXEC-IN-USER-MODE	이 튜토리얼
ECU_Partition0 (ECU)	true	만들지 않는다. System 편집기에 입력란이 없다
ECU_Partition1 (ECU)	false	만들지 않는다. System 편집기에 입력란이 없다

컨트롤러 (3장~5장)

노드	메뉴	패널 제목	기본 이름	NAME	
ECU	Add Controller > CAN	CAN CONTROLLER	ECU_Can_Controller_0	CanController	CAN 켜짐, SYNC JUMP WIDTH, PROPA(2 모두 0
ZONE	Add Controller > CAN	CAN CONTROLLER	ZONE_Can_Controller_0	CanCommController	CAN 켜짐, 비트 타이밍 네 칸은 비워 둠
ZONE	Add Controller > Ethernet	ETHERNET CONTROLLER	ZONE_Ethernet_Controller_0	EthernetCommController	MAC ADDRESS 11:22:EE:44:55:66 TYPE 누르지 않음
HPC	Add Controller > Ethernet	ETHERNET CONTROLLER	HPC_Ethernet_Controller_0	EthernetController	MAC ADDRESS 00:11:22:33:44:55
ECU	Add Controller > LIN Master	LIN MASTER	ECU_LinMaster_Controller_0	LinController_Master	PROTOCOL VERSION LIN 2.0, TIME I CONTROLLER 누르지 않음
SLAVE_ECU	Add Controller > LIN Slave	LIN SLAVE	SLAVE_ECU_LinSlave_Controller_0	LinSlaveController	PROTOCOL VERSION LIN 2.0. 그 아래

TIME BASE 와 **TIME BASE JITTER** 는 단위 칸을 먼저 ms 로 바꾼 뒤 1 을 입력한다. 저장값은 초 단위 0.001 이다.

커넥터 (3장~5장)

노드	메뉴	기본 이름	NAME	CONTROLLER	클러스터 / 채널
ECU	Add Connector > CAN	ECU_Can_Connector_0	CanConnector	CanController	CanCluster / PhysicalChannel
ZONE	Add Connector > CAN	ZONE_Can_Connector_0	CanConnector	CanCommController	CanCluster / PhysicalChannel
ZONE	Add Connector > Ethernet	ZONE_Ethernet_Connector_0	EthernetConnector	EthernetCommController	EthernetCluster / PhysicalChann
HPC	Add Connector > Ethernet	HPC_Ethernet_Connector_0	EthernetConnector	EthernetController	EthernetCluster / PhysicalChann
ECU	Add Connector > LIN	ECU_Lin_Connector_0	LinConnector_Master	LinController_Master	LinCluster / PhysicalChannel
SLAVE_ECU	Add Connector > LIN	SLAVE_ECU_Lin_Connector_0	LinConnector	LinSlaveController	LinCluster / PhysicalChannel

연결한 뒤 이더넷 커넥터의 **CONNECTED CHANNEL** 은 EthernetCluster / PhysicalChannel (VLAN: 13), LIN 커넥터는 LinCluster / PhysicalChannel 이다.

클러스터와 채널 (3장~5장)

클러스터	만든 곳	기본 이름	패널 제목	CHANNEL NAME	채널 값
`CanCluster`	`ZONE` 의 `CanConnector` > **+** > **Create CommunicationCluster [Can]**	`CanCluster_0`	**CAN CLUSTER**	`PhysicalChannel` (기본값)	—
`CanCluster` (버스 오프 복구)	**CAN CLUSTER** 패널 > **BUS-OFF RECOVERY**	—	**CAN CLUSTER**	—	**BOR TIME L1** `100` `ms`, **BOR TIME L2** `200` `ms` **COUNTER L1 TO L2** `2`, **BOR TIME TX ENSURE** `ms` (저장값 `0.1`, `0.2`, `2`, `0.1`), **MAIN FUNC PERIOD** 는 빈 칸
`EthernetCluster`	`ZONE` 의 `EthernetConnector` > **+** > **Create CommunicationCluster [Ethernet]**	`EthernetCluster_0`	**ETHERNET CLUSTER**	`PhysicalChannel` (기본값)	**VLAN ID** `13`
`LinCluster`	`SLAVE_ECU` 의 `LinConnector` > **+** > **Create CommunicationCluster [Lin]**	`LinCluster_0`	**LIN CLUSTER**	`PhysicalChannel` (기본값)	**IDLE TIMEOUT** 빈 칸(`10` `ms` 는 거부됨)

두 번째 커넥터는 **+** > **Connect to Channel...** > 클러스터 이름 > **PhysicalChannel**(이더넷은 **PhysicalChannel (VLAN: 13)**)로 연결한다. 보오레이트는 입력하지 않는다. PAIO 가 을 저장한다.

엔드포인트와 VLAN (4장)

참조 설계의 요소	값	넣는 곳	익스포트 이름
`ZONE_Endpoint`	`192.168.0.19` (`FIXED`)	`ZONE` 커넥터의 **UNICAST IPV4 ADDRESS**, **ADDRESS SOURCE** 의 **FIXED**	`ZONE_EthernetConnector_Endpoint`
`HPC_Endpoint`	`192.168.0.11` (`FIXED`)	`HPC` 커넥터의 **UNICAST IPV4 ADDRESS**, **ADDRESS SOURCE** 의 **FIXED**	`HPC_EthernetConnector_Endpoint`
`SOMEIP_SD_Multicast`	`224.0.0.1`	`HPC` 커넥터의 **MULTICAST IPV4 ADDRESS**. **MULTICAST PORT** 는 비워 둠	`HPC_EthernetConnector_Multicast`
`Vlan_13`	`VLAN-IDENTIFIER` `13`	채널의 **VLAN ID**	`VLAN_13`

NM 클러스터와 TP 설정 (3장~5장)

NM, TP, 타임 도메인은 **Communication Cluster Configuration** 창(클러스터를 더블클릭)에서 **Enable** 로 클러스터마다 켠다. 참조 설계에는 NM 도 TP 도 없으므로 이 튜토리얼

클러스터	창의 탭	꺼 둔 탭	장
`CanCluster`	**CLUSTER**, **NM CLUSTER**, **TRANSPORT PROTOCOL**, **TIME DOMAIN**	**NM CLUSTER**, **TRANSPORT PROTOCOL**, **TIME DOMAIN** (이름 앞에 빈 동그라미)	3
`EthernetCluster`	**CLUSTER**, **NM CLUSTER**, **TIME DOMAIN**	**NM CLUSTER**, **TIME DOMAIN** (이름 앞에 빈 동그라미)	4
`LinCluster`	**CLUSTER**, **TRANSPORT PROTOCOL**	**TRANSPORT PROTOCOL**. 5장에서는 창을 열지 않는다	5

LIN 에는 NM 이 적용되지 않으므로 LIN 클러스터에는 **NM CLUSTER** 탭이 없다.

검증 장의 결함 (6장)

검증은 아래 결함 세 개가 든 설계에서 시작해 LIN 설정의 값으로 되돌린다.

위치	결함 값	고친 값	검증 오류
`HPC` 의 `EthernetConnector` > **UNICAST IPV4 ADDRESS**	`224.0.0.1`	`192.168.0.11`	`constr_3288`
`SLAVE_ECU` 의 `LinConnector` > **CONTROLLER**	빈 값(`Select controller...`)	`LinSlaveController`	`constr_5398`
`SLAVE_ECU` 의 `LinSlaveController` > **PROTOCOL VERSION**	빈 값(켜진 버튼 없음)	`LIN 2.0`	`constr_5449`

프로젝트 파일 구성

```
VehicleDesign/
├── VehicleDesign.io.prj
├── design/
│   ├── IVC.io.model                    1장~6장
│   ├── SYSTEM.io.model                 1장, 2장
│   └── VFB.io.model                    1장
└── export/
    ├── VehicleDesign_COM.arxml         7장
    ├── VehicleDesign_DataType.arxml    7장
    ├── VehicleDesign_SWC.arxml         7장
    └── VehicleDesign_System.arxml      7장
```

- `IVC.io.model` 에는 노드, 컨트롤러, 커넥터, 클러스터(그리고 익스포트되지 않는 보기 상태)가 들어간다. `SYSTEM.io.model` 에는 **VERSION** 이 들어간다. 이 튜토리얼은 `V`
- 익스포트는 `export/` 에만 쓴다. `doc`, `code`, `workflow`, `import` 폴더는 쓰지 않는다.

참조 설계와 다른 곳

내 익스포트는 `System_Design.arxml` 과 설계 내용이 거의 같다. 이름이 있는 요소 29개 가운데 27개에 짝이 있고, 빠지는 것은 파티션 두 개뿐이다. 이유를 포함한 전체 목록은 A
있다.

종류	예
튜토리얼이 일부러 다르게 한 곳	자기 컨트롤러와 `192.168.0.11` 을 쓰는 `HPC` 커넥터, `MAC-LAYER-TYPE` 없음, AUTOSAR CP 튜토리얼을 위한 `CanCluster` 의 `BUS-OFF-RECOVERY`
PAIO 의 이름 규칙	패키지 `/System/EcuInstance`, `/System/MachineDesign`, `/System/CommunicationCluster/…`, 엔드포인트 이름 `<노드>_<커넥터>_Endpoint`, `VLAN_1`
PAIO 가 넣는 것	`BAUDRATE` `500000`·`19200`, SD 설정과 소켓, 기본 데이터 타입 49개, 루트 컴포지션 `System`
화면에 입력칸 없음	파티션, `CATEGORY` `WIRED`, `PROTOCOL-NAME` `IEEE 802.3`, LIN 의 `BUS-IDLE-TIMEOUT-PERIOD` `0.01`

자주 겪는 문제

증상 또는 검증 메시지	원인	해결
`constr_3288`: 유니캐스트 IPv4 `224.0.0.1` 이 유니캐스트 범위가 아님	`HPC` 커넥터에 멀티캐스트 주소가 들어 있다	캔버스에서 `HPC` 커넥터를 클릭해 `192.168.0.11`
`constr_5398`: `SLAVE_ECU` 의 `LinConnector` 에 commController 참조가 없음	**CONTROLLER** 가 비어 있다	`LinSlaveController` 를 고르고 `Ctrl+S`
`constr_5449`: `LinSlaveController` 에 protocolVersion 이 없음	**PROTOCOL VERSION** 에 켜진 버튼이 없다	컨트롤러를 클릭하고 `LIN 2.0` 을 누른 뒤 `Ctrl+S`
TIME BASE 가 1초, **IDLE TIMEOUT** 이 10초가 됨	단위 칸의 기본값이 `s` 이다	입력 전에 단위를 `ms` 로 바꾼다. **IDLE TIMEOUT** 은
클러스터가 하나 더 생김	두 번째 커넥터에서 **Create CommunicationCluster** 를 썼다	두 번째 커넥터는 **Connect to Channel…** 로 연결
Connected Connectors 의 순서가 반대임	다른 쪽 커넥터에서 클러스터를 만들었다	CAN·이더넷 클러스터는 `ZONE` 에서, LIN 클러스터는 든다
익스포트의 `XMII` 에 스키마 오류	**MAC LAYER TYPE** 을 눌렀다	**MAC LAYER TYPE** 은 누르지 않는다
ADDRESS SOURCE 가 모두 꺼져 있음	**FIXED** 를 클릭하지 않았다	두 이더넷 커넥터에서 **FIXED** 를 클릭한다
값이 반영되지 않음	입력란은 빠져나와야 값이 확정된다	입력한 뒤 `Tab` 키를 누른다
ARXML 익스포트의 줄 번호가 맞지 않음	노드를 다른 순서로 추가했다	`ECU` → `ZONE` → `SLAVE_ECU` → `HPC` 순서로 추
`HPC` 커넥터의 + 배지를 누르기 어려움	배지가 작고, 화면 오른쪽 끝에서는 메뉴가 왼쪽으로 펼쳐진다	확대하거나, **Zoom Out** 을 한 번 더 누르고 메뉴를 다

확인하기

- [ ] "완성된 설계 점검표" 의 항목이 모두 맞다.

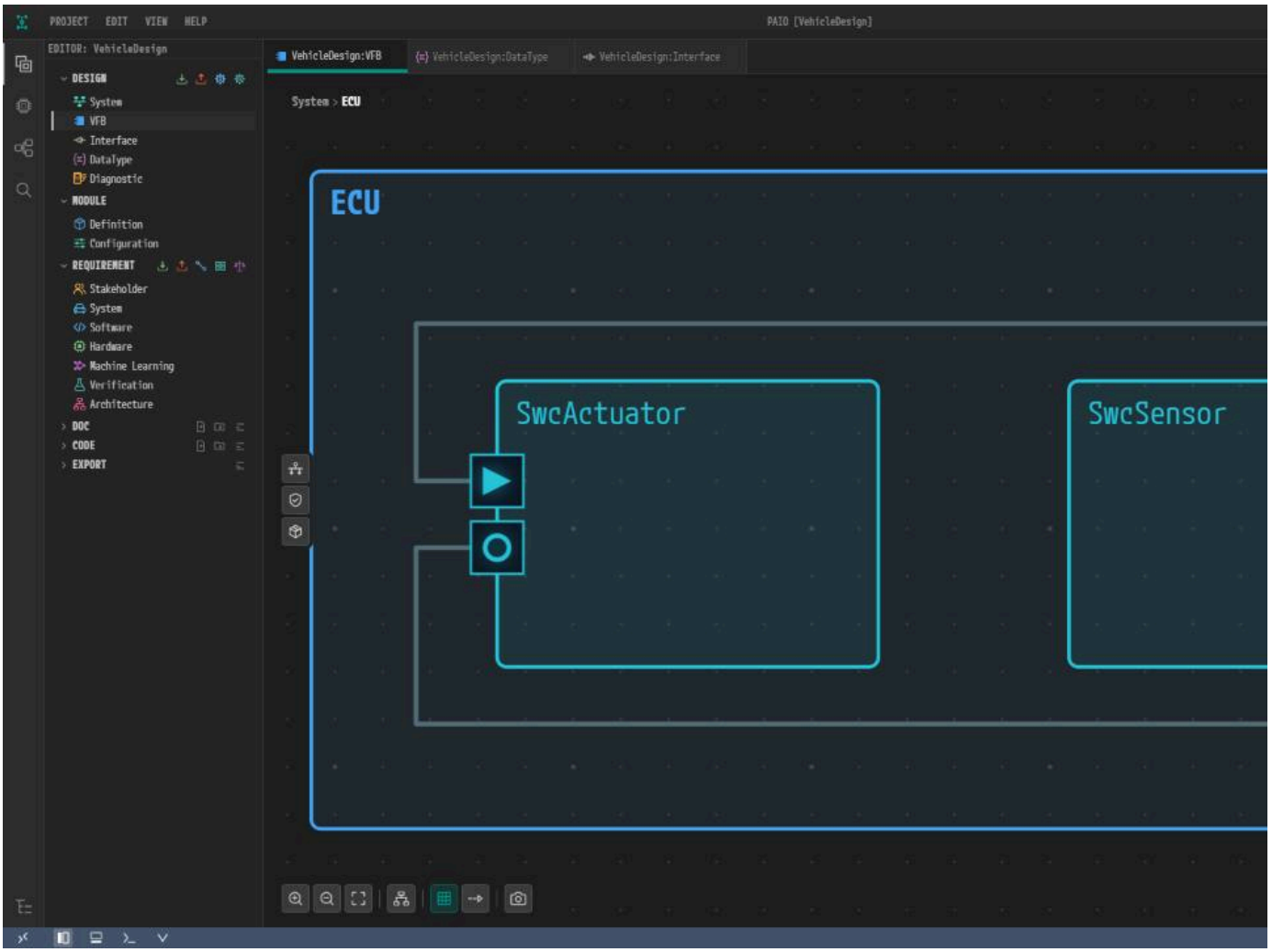
- [ ] 내 값이 "값 표 모음" 과 같다.

다음 단계

AUTOSAR CP 튜토리얼이 이 프로젝트를 이어서 쓴다. 완성된 상태에서 시작하려면 VehicleDesign-system.zip을 내려받는다.

AUTOSAR CP 튜토리얼

이 튜토리얼은 프로젝트 `VehicleDesign` 의 ECU 노드 `ECU` 에 소프트웨어 컴포넌트 두 개를 PAIO 로 올리고, CAN 스택과 네트워크 관리를 포함한 BSW 를 PRTE AUTO CONFIGU Platform 코드 생성기 PRTE 로 ECU 코드를 생성한다. 한 프로젝트를 차례로 만드는 세 편 가운데 둘째 편이다: AUTOSAR System → AUTOSAR CP → AUTOSAR AP.



이 튜토리얼에서 만드는 소프트웨어. 컴포지션 `ECU` 안의 `SwcSensor` 와 `SwcActuator` 를 커넥터 두 개로 잇는다.

무엇을 만드는가

`SwcSensor` 는 20ms 마다 시작되는 러너블에서 속도 값을 쓸 수 있고 `SwcActuator` 에 스케일 값을 요청할 수 있다. `SwcActuator` 는 그 값을 받고 요청에 답한다. 튜토리얼은 코드이다.

SW-C	포트	인터페이스	하는 일
`SwcSensor`	`PpSpeed` (제공)	`IfSpeed` (Sender-Receiver)	20ms 마다 시작되어 `speed` 를 쓸 수 있다
`SwcSensor`	`RpScale` (요구)	`IfScale` (Client-Server)	`GetScale` 을 호출할 수 있다
`SwcActuator`	`RpSpeed` (요구)	`IfSpeed` (Sender-Receiver)	`speed` 가 도착하면 실행된다
`SwcActuator`	`PpScale` (제공)	`IfScale` (Client-Server)	`GetScale` 에 답한다

준비물

- 설치되어 실행할 수 있는 PAIO.
- AUTOSAR System 튜토리얼을 마친 프로젝트 `VehicleDesign`. 그 튜토리얼을 건너뛰려면 VehicleDesign-system.zip을 내려받아 압축을 풀고, **PROJECT** 메뉴의 **Open**
- 6장에서 쓸 Rte 값 파일 VehicleDesign_Rte.arxml. 시작하기 전에 저장해 둔다. 링크가 브라우저 탭에서 열리면 다른 이름으로 저장한다.
- 1920×1080 이상의 화면.

- 포트, 러너블, RTE 를 대략 알면 충분하다. 어느 화면에 무엇을 입력하는지는 장마다 보여 준다.

장 구성

장	하는 일	사용하는 편집기
1. 컴포지션과 ECU 매핑	컴포지션 `ECU` 를 만들어 노드 `ECU` 에 매핑한다	VFB 편집기
2. 데이터 타입	데이터 타입 `SpeedType` 을 만든다	DataType 편집기
3. 인터페이스	인터페이스 `IfSpeed` 와 `IfScale` 을 만든다	Interface 편집기
4. SW-C와 포트	포트를 두 개씩 가진 `SwcSensor` 와 `SwcActuator` 를 만들고 커넥터로 잇는다	VFB 편집기
5. Internal Behavior	러너블 4개와 그 액세스 포인트, RTE 이벤트를 넣는다	SWC Internal Behavior
6. PRTE Setup	BSW 모듈을 가져오고 `EcuM`, `BswM`, `Mcu`, `Det` 를 설정한 뒤 Rte 값 파일을 임포트한다	ECU Configuration, Configuration
7. Os 설정	OS 설정, 태스크, 알람, 카운터를 손으로 넣는다	Configuration
8. Rte 설정	RTE 이벤트를 태스크에 매핑하고 BSW 모듈 인스턴스를 채운다	Configuration
9. Communication 설정	`EcuC`, `Can`, `CanIf`, `ComM`, `CanSM`, `Nm`, `CanNm`, `LSduR`, `PduR`, `Com` 을 손으로 넣는다	Configuration
10. 검증과 코드 생성	설계를 검증하고 PRTE 로 코드를 생성한다	VALIDATION 패널, PRTE Code Generation
부록	모든 값, 자주 겪는 문제	—

이 튜토리얼을 읽는 법

- **장은 순서대로 따라간다.** 각 장은 앞 장의 결과에서 시작한다.
- **반복 작업은 한 번만 보여 준다.** 하나를 절차로 보여 주고, 나머지는 값 표로 주며 같은 방법으로 입력한다.
- UI 라벨은 화면에 보이는 대로 **굵게**, 모델 요소 이름과 값은 `code` 로 적는다.

다음 단계

컴포지션과 ECU 매핑부터 시작한다.

컴포지션과 ECU 매핑

이 장에서 할 일

이 장에서는 이 ECU 의 소프트웨어를 담을 컴포지션 `ECU` 를 만들어 노드 `ECU` 에 매핑한다. 매핑하면 컴포지션의 ECU 아이콘이 **ECU Configuration** 창을 연다. 6장에서 이 창으로

시작하기 전에

- AUTOSAR System 튜토리얼을 마친 프로젝트 `VehicleDesign` 에서 시작한다. 노드 `ECU` 는 그 튜토리얼에서 만들었고 CAN 버스와 LIN 버스에 연결되어 있다.
- 그 튜토리얼을 건너뛰었다면 완성된 프로젝트 VehicleDesign-system.zip을 내려받아 압축을 풀고, **PROJECT** 메뉴의 **Open Project** 로 폴더 `VehicleDesign` 을 연다.

위치	항목	값
컴포지션 패널	**NAME, AS PROTOTYPE**	`ECU`
컴포지션 메뉴	**ECU/HPC Mapping**	**ECU [ECU]**

컴포지션 만들기

1. **DESIGN** 아래의 **VFB** 를 누른다. 탭 `VehicleDesign:VFB` 가 열린다.

2. 캔버스의 빈 곳을 오른쪽 클릭하고 **Add Composition** 을 누른다.

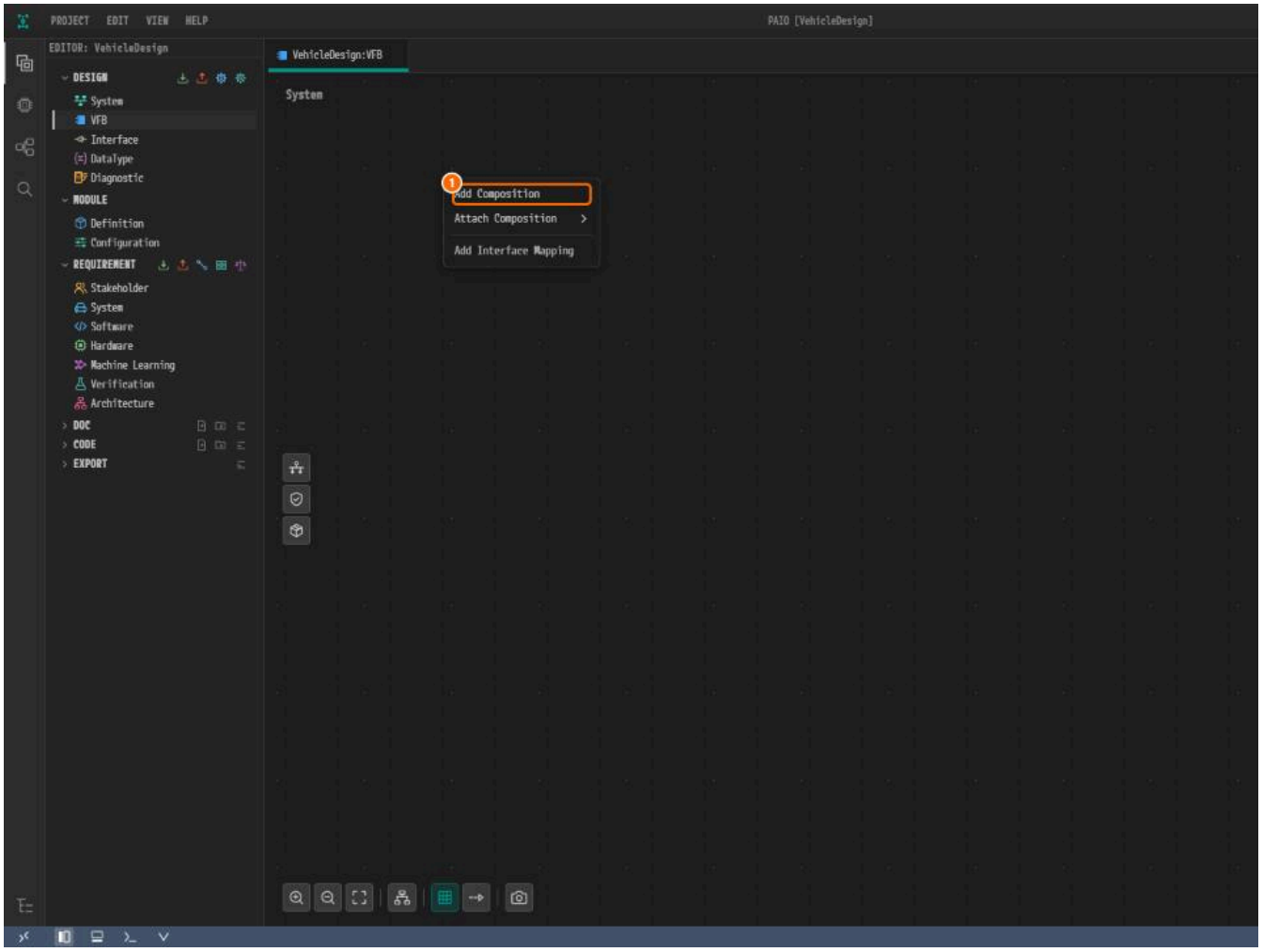


*그림 1-1. ① 캔버스 메뉴의 **Add Composition**.*

3. 새 컴포지션을 누른다. 패널의 **NAME** 과 **AS PROTOTYPE** 에 모두 `ECU` 를 입력하고 각각 `Enter` 를 누른 뒤 `Ctrl+S` 로 저장한다.

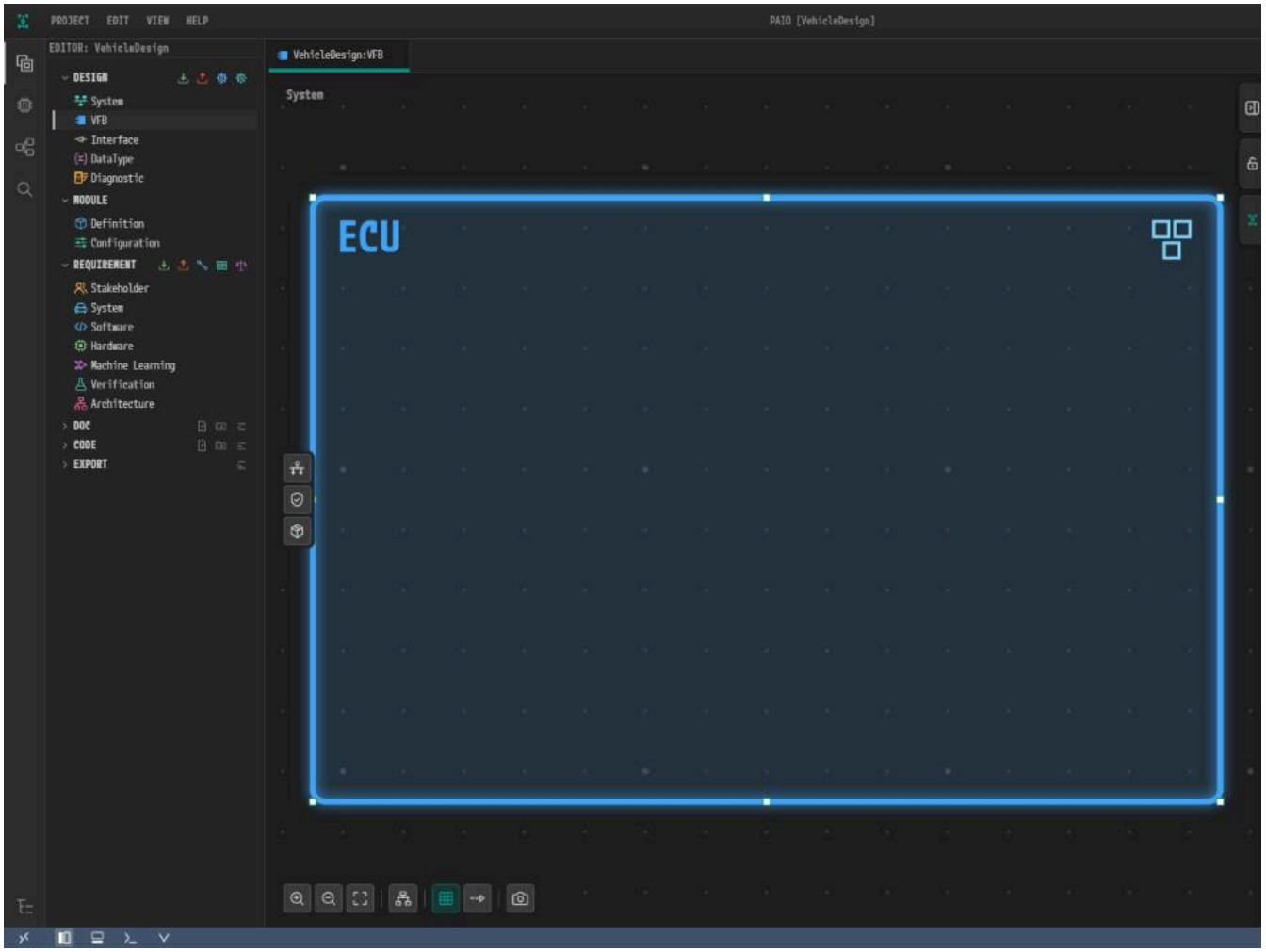


*그림 1-2. ① **NAME**, ② **AS PROTOTYPE**.*

ECU 노드에 매핑하기

1. 컴포지션 안쪽을 오른쪽 클릭하고 **ECU/HPC Mapping** 위에 포인터를 올린다. 하위 메뉴에 System 편집기의 노드 네 개가 보인다. **ECU [ECU]** 를 누른다. `Ctrl+S` 로 저장한다

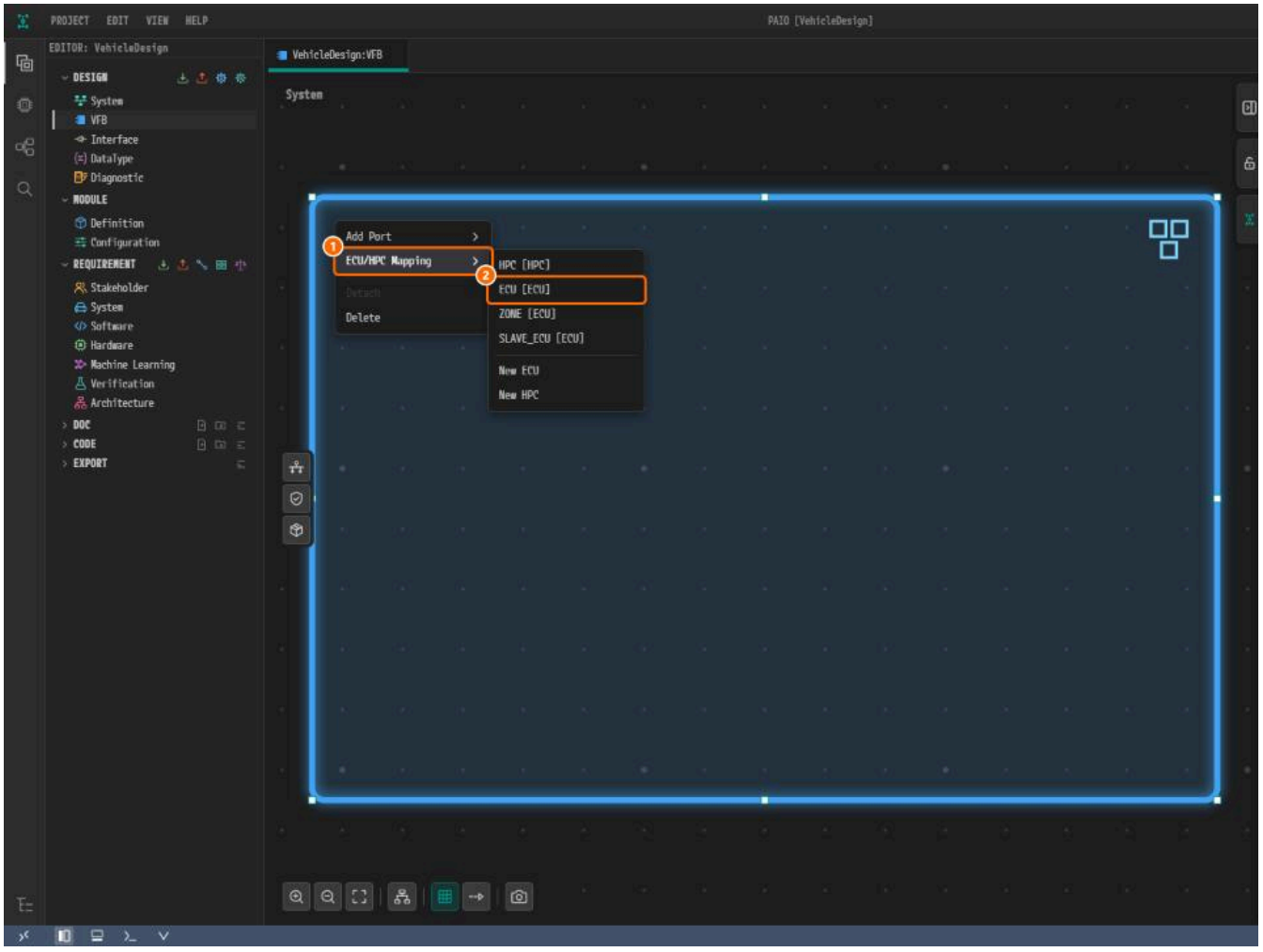


*그림 1-3. ① **ECU/HPC Mapping**, ② **ECU [ECU]**.*

2. 컴포지션을 더블클릭해 안으로 들어간다. 브레드크럼이 **System > ECU** 가 되고, 프레임 오른쪽 위에 ECU 아이콘이 나타난다.

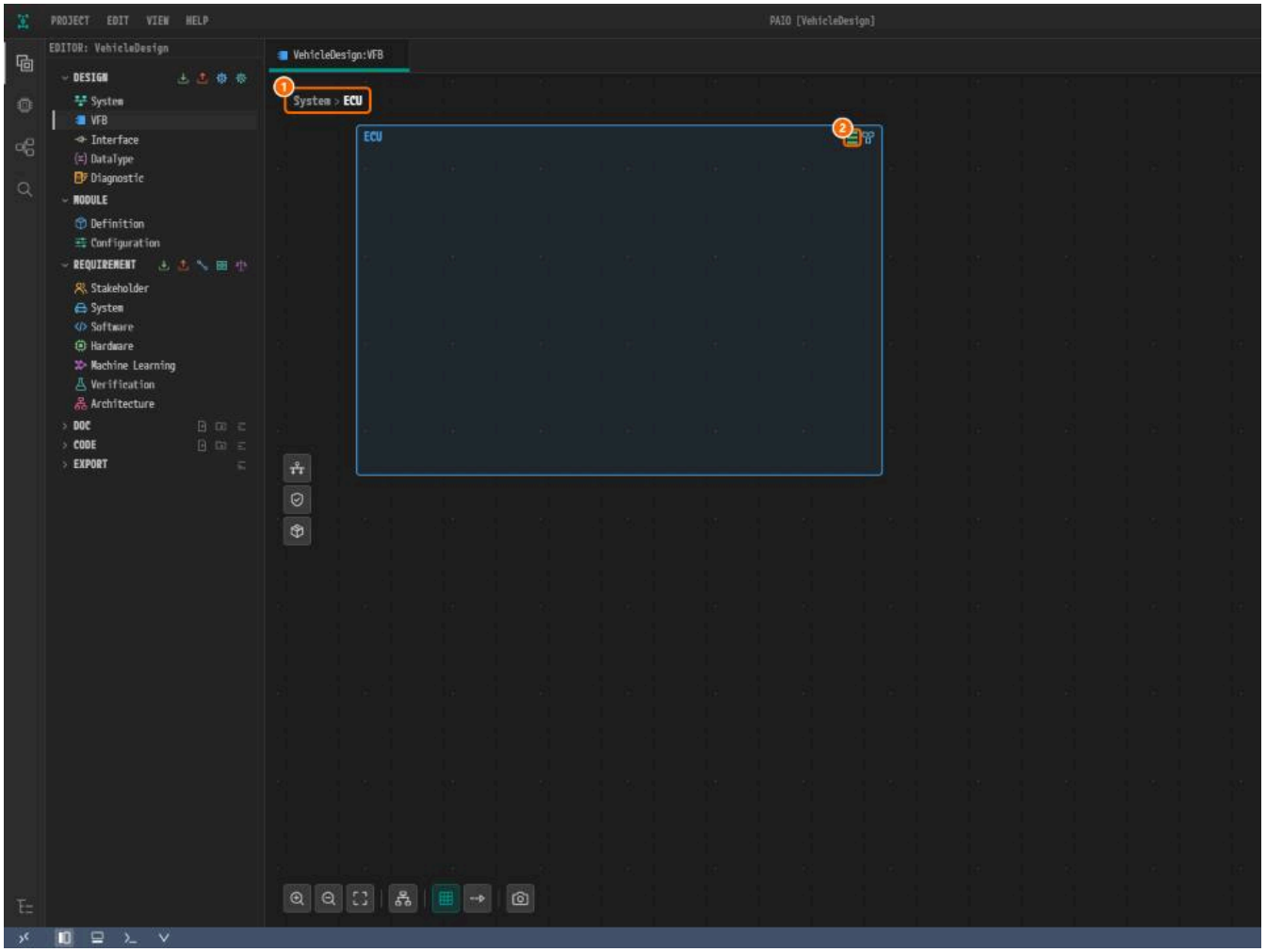


그림 1-4. ① 브레드크럼, ② ECU 아이콘. 컴포지션이 ECU 노드에 매핑되어 있어 아이콘이 나타난다.

왜 이렇게 설계하는가

- **BSW 는 나중에.** BSW 모듈과 그 설정은 6장에서 이 ECU 의 **ECU Configuration** 창으로 한 번에 가져온다. 2장부터 5장까지는 애플리케이션 쪽만 만든다.

확인하기

- [ ] 컴포지션 `ECU` 안에서 브레드크럼이 **System > ECU** 이고 ECU 아이콘이 보인다.

다음 단계

데이터 타입에서 두 SW-C 가 주고받을 데이터 타입을 만든다.

데이터 타입

이 장에서 할 일

이 장에서는 `uint16` 의 별칭인 데이터 타입 `SpeedType` 을 만든다. 다음 장의 두 인터페이스가 이 타입을 쓴다.

시작하기 전에

- 컴포지션과 ECU 매핑를 마친 뒤 시작한다.

위치	항목	값
DataType 편집기	**NAME**	`SpeedType`
DataType 편집기	**CATEGORY**	**ALIAS**
DataType 편집기	**BASETYPE**	`uint16`

SpeedType 만들기

1. **DESIGN** 아래의 **DataType** 을 누른다. 목록에 기본 타입이 있다. 목록 위의 **+** 를 누른다.

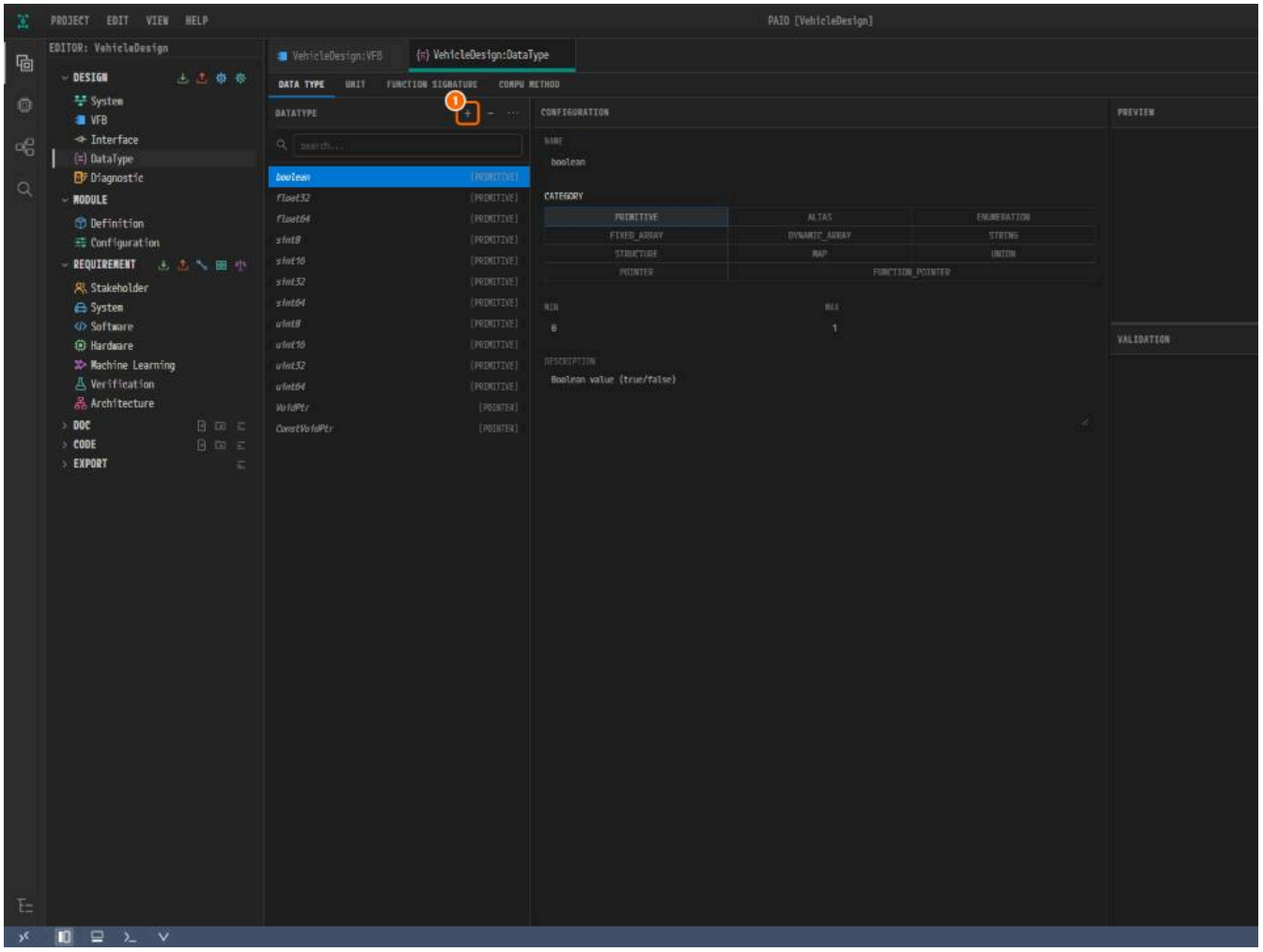


그림 2-1. ① +(Add DataType).

2. **NAME** 에 `SpeedType` 을 입력하고 `Tab` 을 누른 뒤 **CATEGORY** 의 **ALIAS** 를 누른다. **BASETYPE** 을 열어 `uint16` 을 입력하고 목록의 `uint16` 을 누른다.

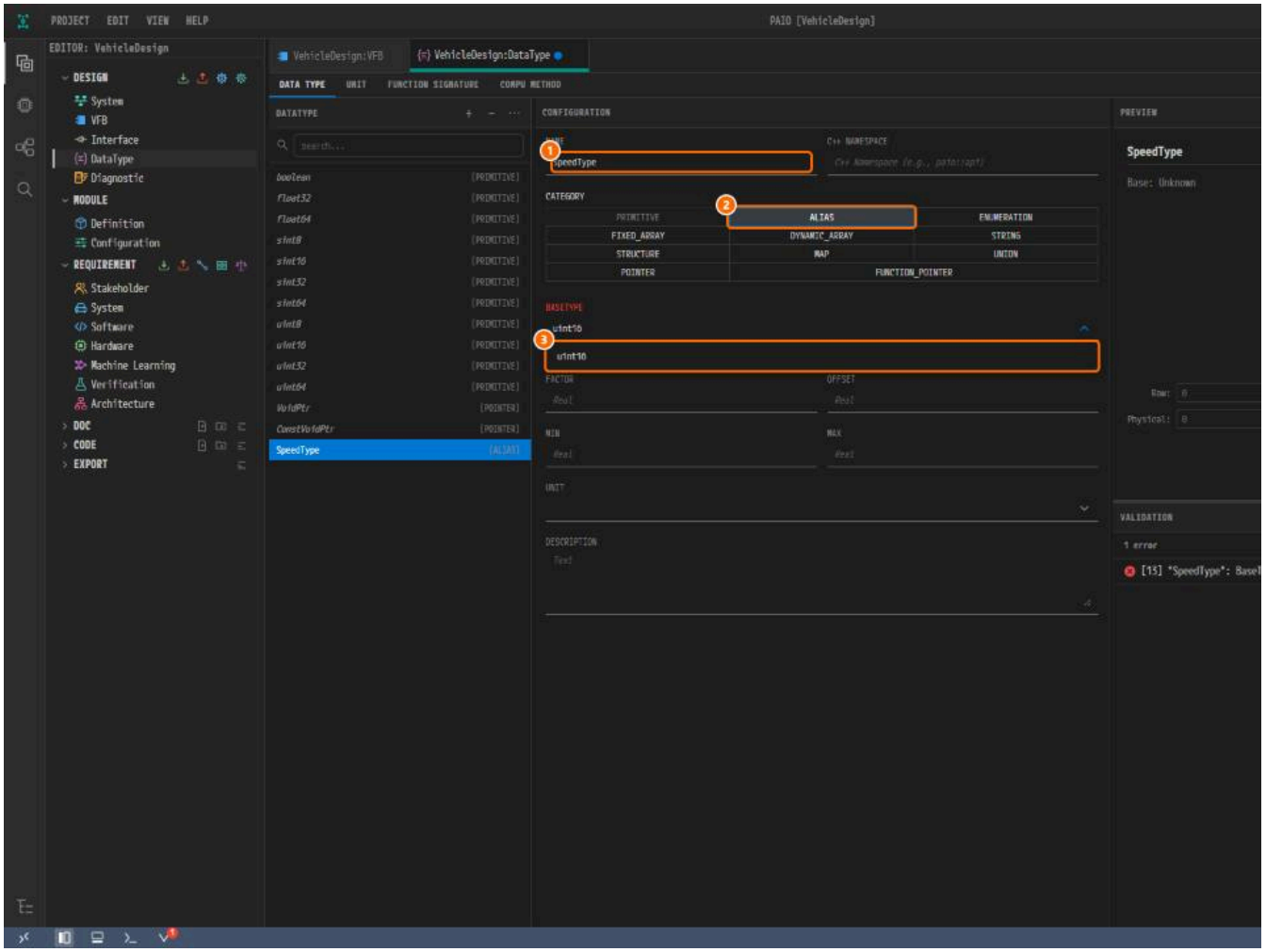


*그림 2-2. ① **NAME**, ② **ALIAS**, ③ **BASETYPE** 목록의 `uint16`.*

3. `Ctrl+S` 로 저장한다.

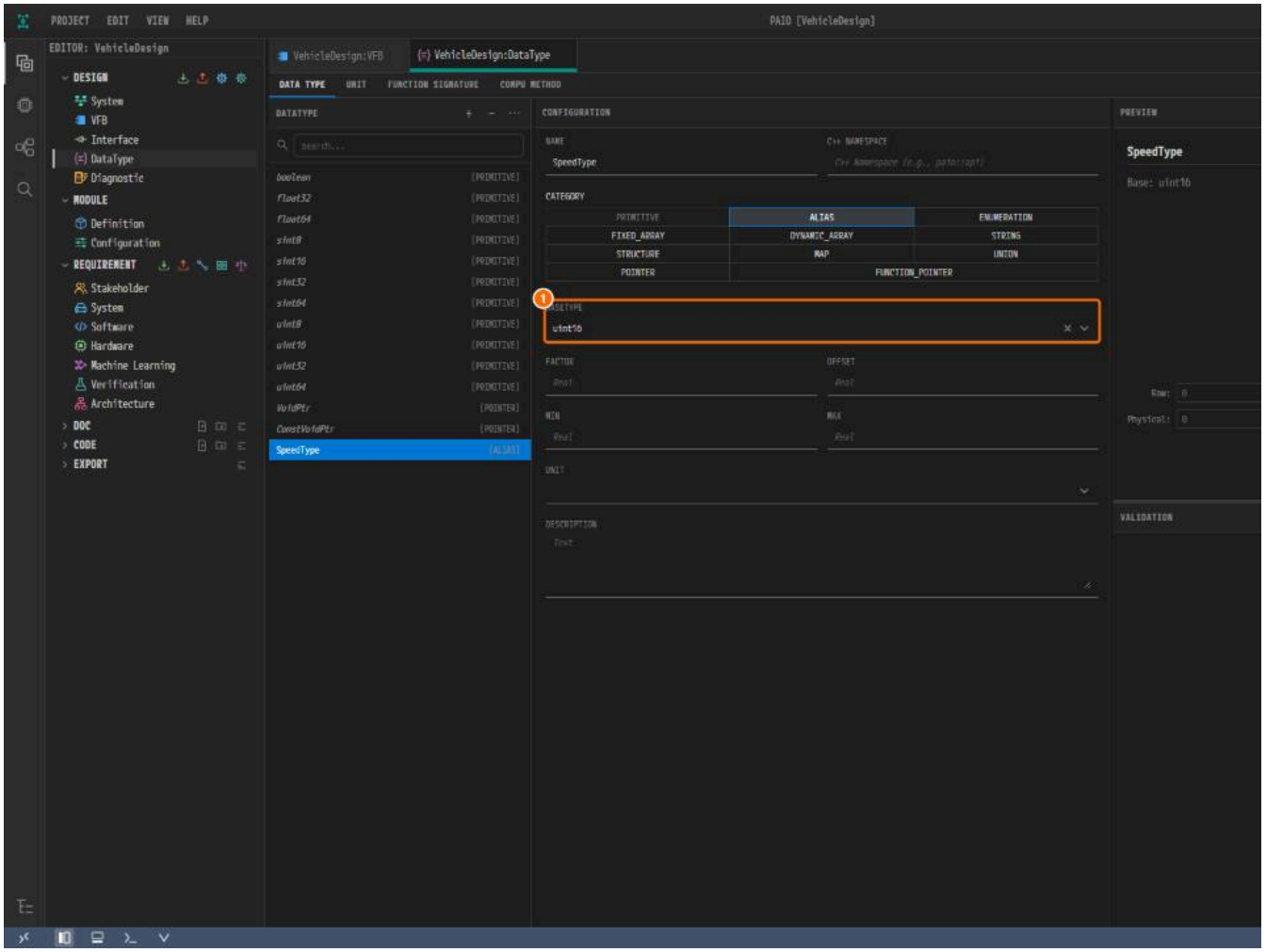


*그림 2-3. ① **BASETYPE** = `uint16`.*

왜 이렇게 설계하는가

- `uint16` **의 별칭.** 이름 붙은 타입은 값의 의미를 말해 주고, 기본 타입이 바뀌면 쓰는 곳이 함께 바뀐다.

확인하기

- [ ] 목록에 `SpeedType` 이 있고, 선택하면 **ALIAS** 와 **BASETYPE** `uint16` 이 보인다.
- [ ] 탭 `VehicleDesign:DataType` 에 수정 표시가 없다.

다음 단계

인터페이스에서 `SpeedType` 을 쓰는 인터페이스 두 개를 만든다.

인터페이스

이 장에서 할 일

이 장에서는 SW-C 들이 통신할 인터페이스 두 개를 만든다. 속도 값을 위한 Sender-Receiver 인터페이스 `IfSpeed` 와 배율 요청을 위한 Client-Server 인터페이스 `IfScale` 이

시작하기 전에

- 데이터 타입을 마친 뒤 시작한다.

인터페이스	탭	요소	이름	타입 또는 방향
`IfSpeed`	**SENDER-RECEIVER**	데이터 요소	`speed`	**TYPE** `SpeedType`
`IfScale`	**CLIENT-SERVER**	오퍼레이션	`GetScale`	
`IfScale`	**CLIENT-SERVER**	`GetScale` 의 인자	`scale`	**DIRECTION** `OUT`, **TYPE** `SpeedType`

IfSpeed 만들기

1. **DESIGN** 아래의 **Interface** 를 누르고 **SENDER-RECEIVER** 탭을 누른 뒤 **INTERFACES** 목록 위의 **+** 를 누른다.

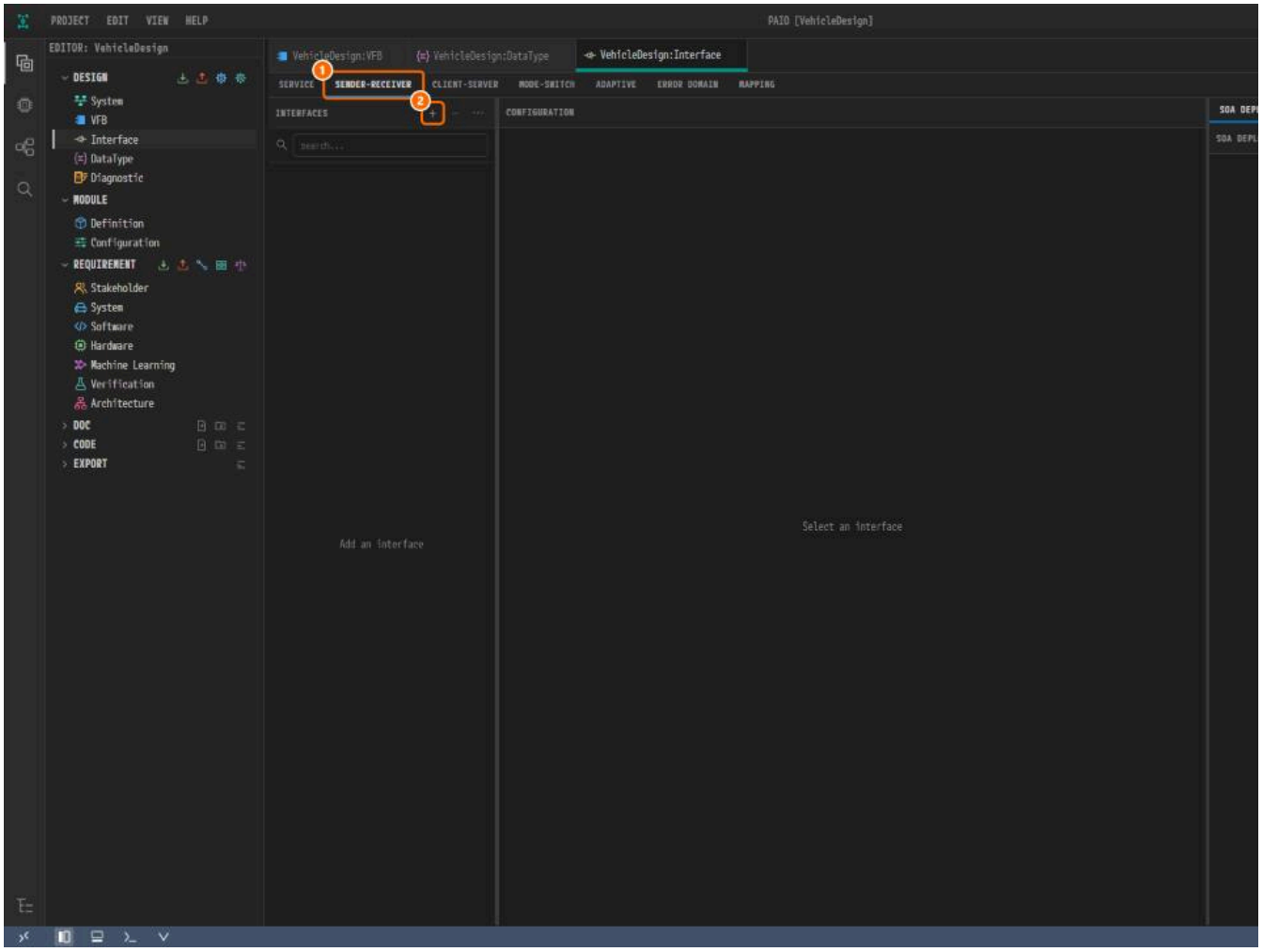


*그림 3-1. ① **SENDER-RECEIVER**, ② **+**(Add Interface).*

2. **NAME** 에 `IfSpeed` 를 입력하고 `Tab` 을 누른다.

3. **DATA ELEMENTS** 머리글의 **+** 를 누른다. 새 행의 **NAME** 에 `speed` 를 입력하고, **TYPE** 을 열어 `SpeedType` 을 입력한 뒤 목록에서 누른다.

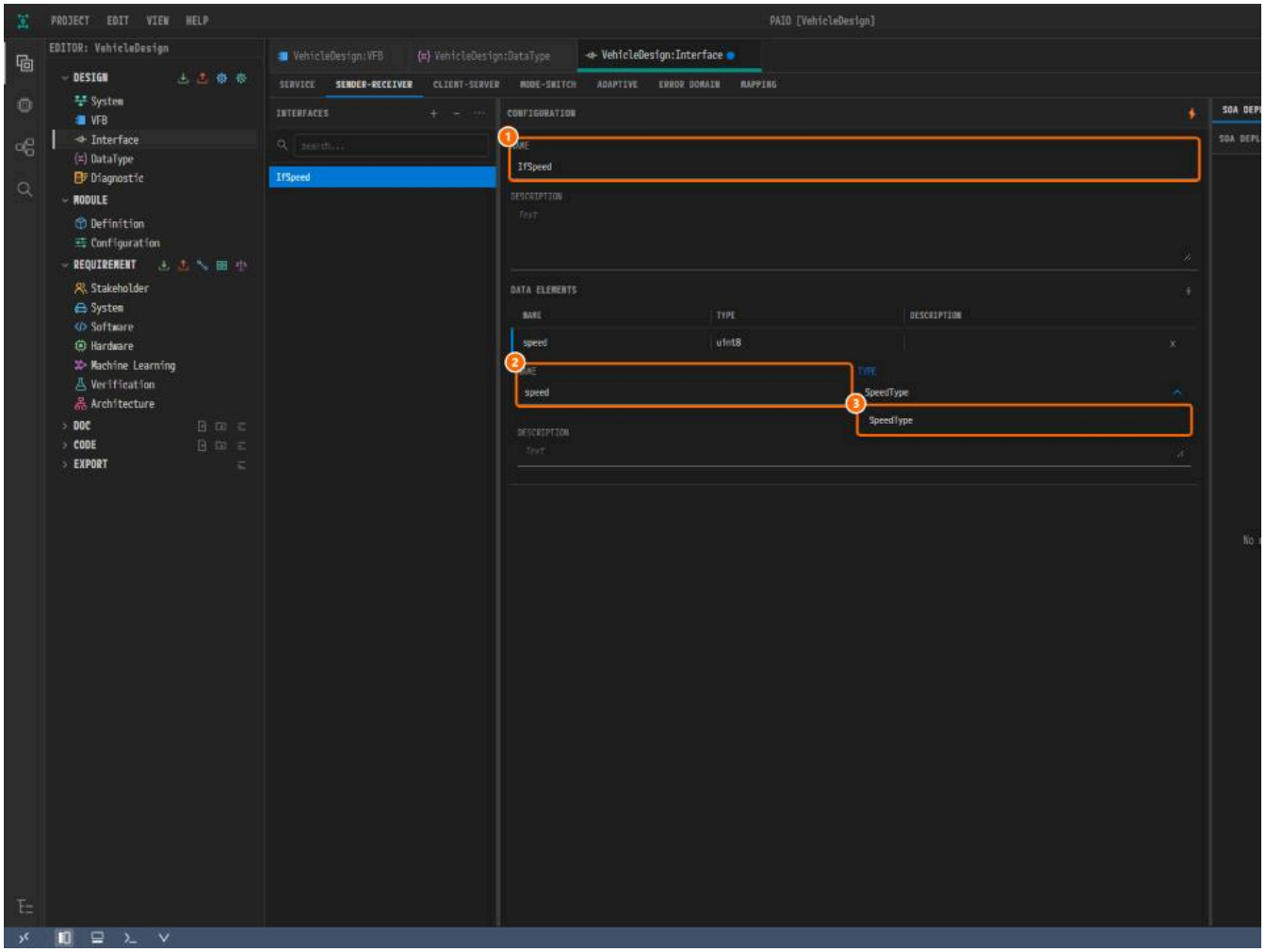


*그림 3-2. ① 인터페이스 **NAME**, ② 데이터 요소 **NAME**, ③ **TYPE** 목록의 `SpeedType`.*

IfScale 만들기

1. **CLIENT-SERVER** 탭을 누르고 **INTERFACES** 목록 위의 **+** 를 누른다. **NAME** 에 `IfScale` 을 입력하고 `Tab` 을 누른다.
2. **OPERATIONS** 머리글의 **+** 를 누르고 오퍼레이션 **NAME** 에 `GetScale` 을 입력한다.
3. **ARGUMENTS** 머리글의 **+** 를 누른다. 인자 **NAME** 에 `scale` 을 입력하고, **DIRECTION** 에서 `OUT`, **TYPE** 에서 `SpeedType` 을 고른다.
4. `Ctrl+S` 로 저장한다.

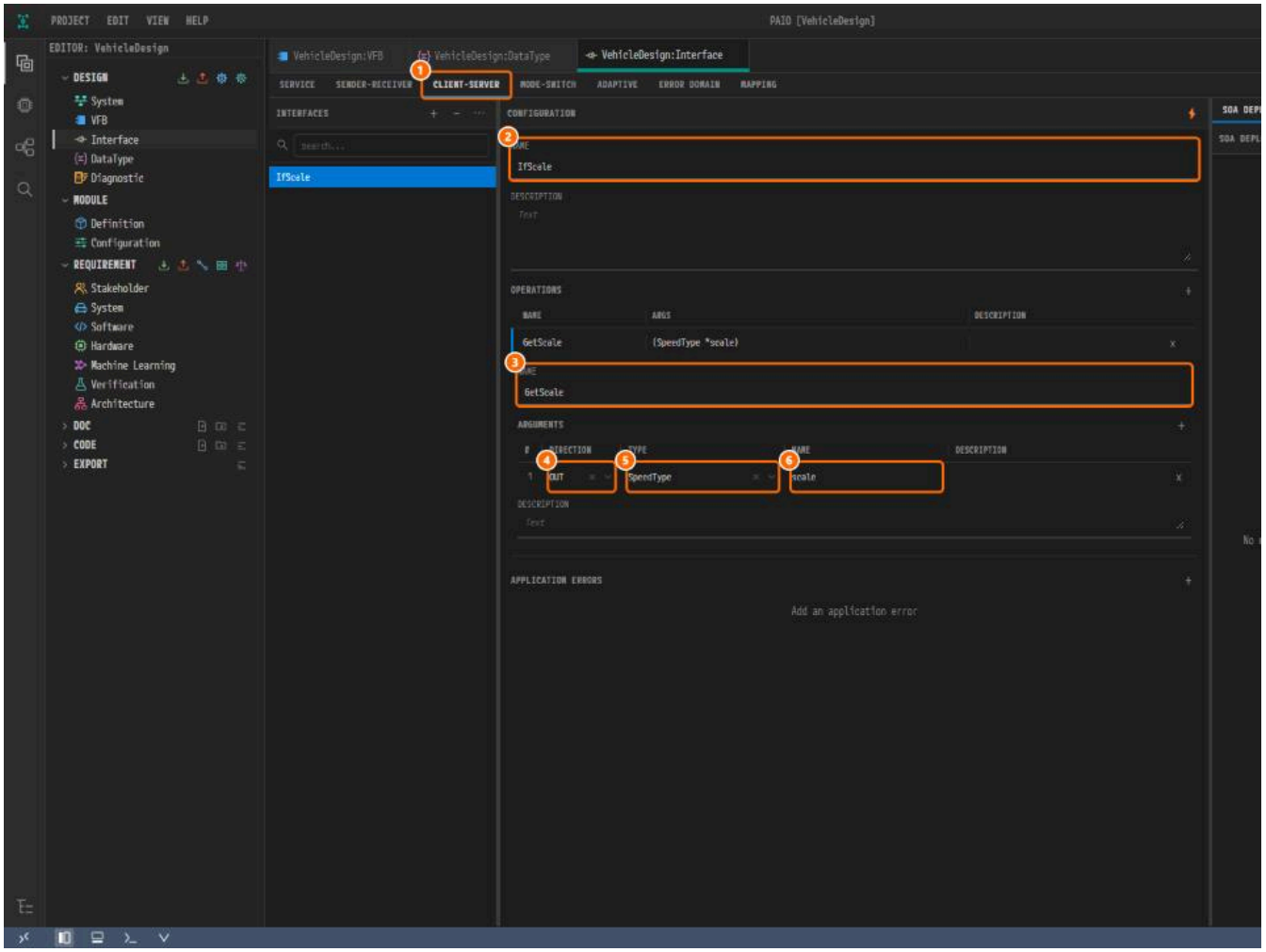


*그림 3-3. ① **CLIENT-SERVER**, ② 인터페이스 **NAME**, ③ 오퍼레이션 **NAME**, ④ **DIRECTION**, ⑤ **TYPE**, ⑥ 인자 **NAME**. 오퍼레이션 행의 **ARGS** 열은* `(SpeedType *scale)`

왜 이렇게 설계하는가

- **두 종류의 인터페이스.** Sender-Receiver 는 값을 보내고 기다리지 않는다. Client-Server 는 묻고 답을 받는다. 이 설계는 각각 하나씩 써서 작은 예제 하나로 둘 다 다룬다.

확인하기

- [ ] **SENDER-RECEIVER** 탭에 `IfSpeed` 와 `SpeedType` 타입의 데이터 요소 `speed` 가 있다.
- [ ] **CLIENT-SERVER** 탭에 `IfScale` 과 오퍼레이션 `GetScale`, 그 인자 `scale` 이 있다.
- [ ] 탭 `VehicleDesign:Interface` 에 수정 표시가 없다.

다음 단계

SW-C와 포트에서 SW-C 두 개를 만들고 이 인터페이스로 포트의 타입을 정한다.

SW-C와 포트

이 장에서 할 일

이 장에서는 컴포지션 `ECU` 안에 애플리케이션 소프트웨어 컴포넌트 `SwcSensor` 와 `SwcActuator` 를 만들고 각각 포트 두 개를 준 뒤, 인터페이스마다 커넥터 하나씩 두 커넥터로

시작하기 전에

- 인터페이스를 마친 뒤 시작한다.
- 탭 `VehicleDesign:VFB` 를 연다. 컴포지션 `ECU` 안이 아니면 더블클릭해 들어간다.

SW-C	포트	Add Port 항목	PORT INTERFACE	설정
`SwcSensor`	`PpSpeed`	**PPort**	`IfSpeed`	**NON-QUEUED**
`SwcSensor`	`RpScale`	**RPort**	`IfScale`	**CLIENT**
`SwcActuator`	`RpSpeed`	**RPort**	`IfSpeed`	**NON-QUEUED**
`SwcActuator`	`PpScale`	**PPort**	`IfScale`	**SERVER**

SW-C 만들기

1. 프레임 안의 빈 곳을 오른쪽 클릭한다. **Add SWC** 위에 포인터를 올리고 **Application** 을 누른다.

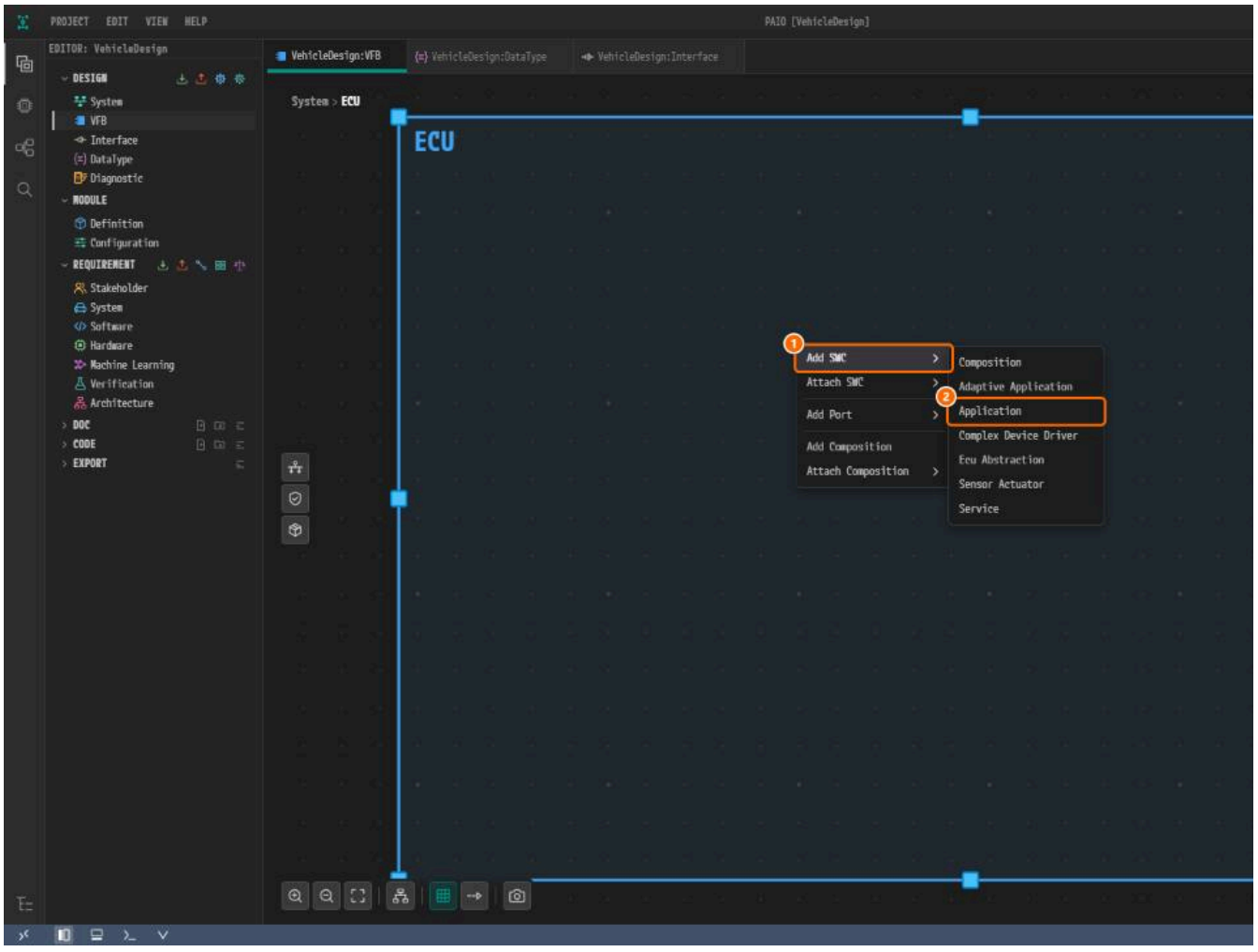


*그림 4-1. ① **Add SWC**, ② **Application**.*

2. 새 SW-C 를 누른다. 패널의 **NAME** 과 **AS PROTOTYPE** 에 모두 `SwcSensor` 를 입력한다.

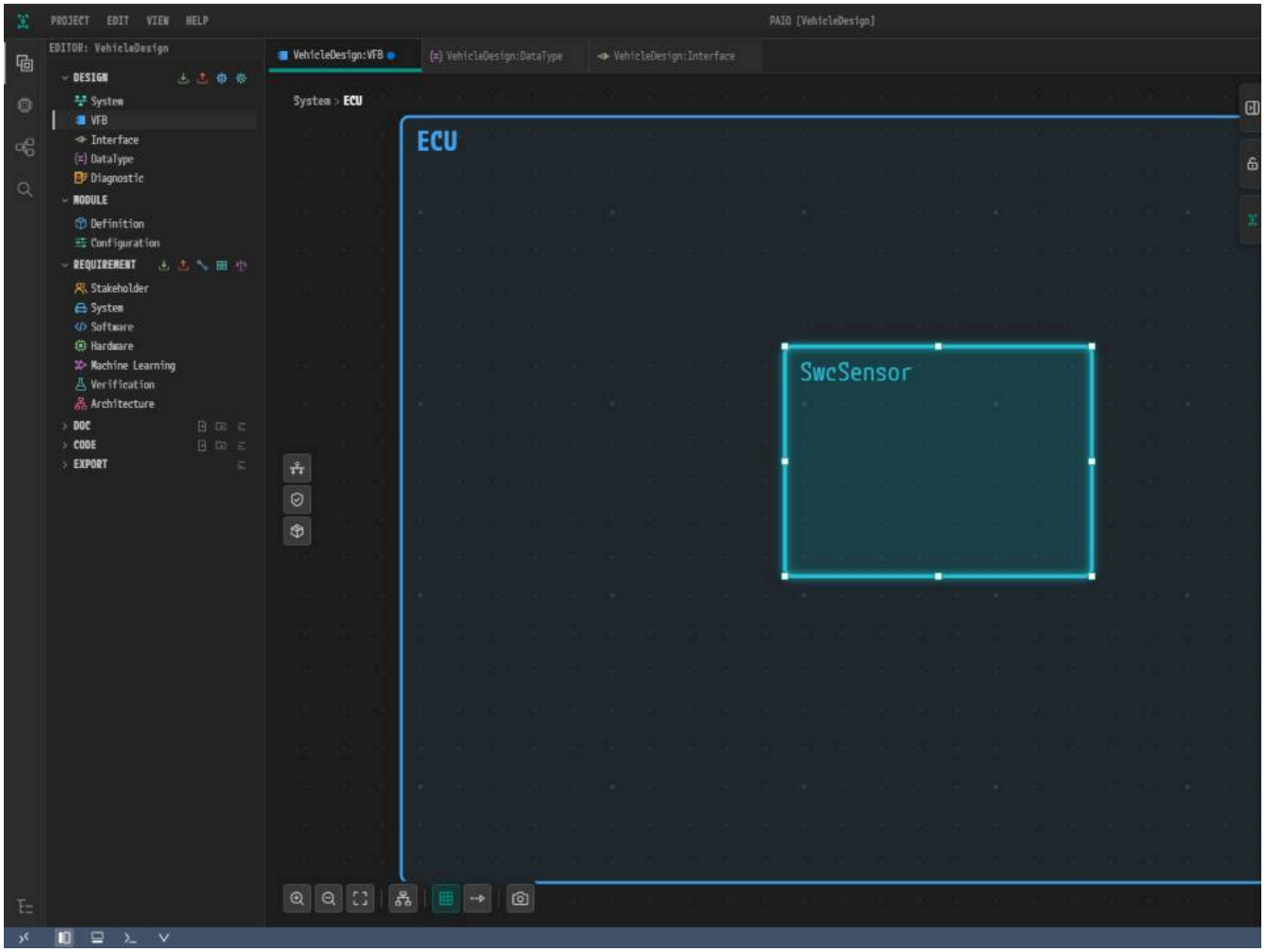


*그림 4-2. ① **NAME**, ② **AS PROTOTYPE**.*

3. `SwcSensor` 아래에 같은 방법으로 `SwcActuator` 를 만든다.

포트 만들기

먼저 포트 `PpSpeed` 를 따라간다.

1. `SwcSensor` 의 오른쪽 가장자리 근처를 오른쪽 클릭한다. **Add Port** 위에 포인터를 올리고 **PPort** 를 누른다. 가장자리에 포트가 나타난다.

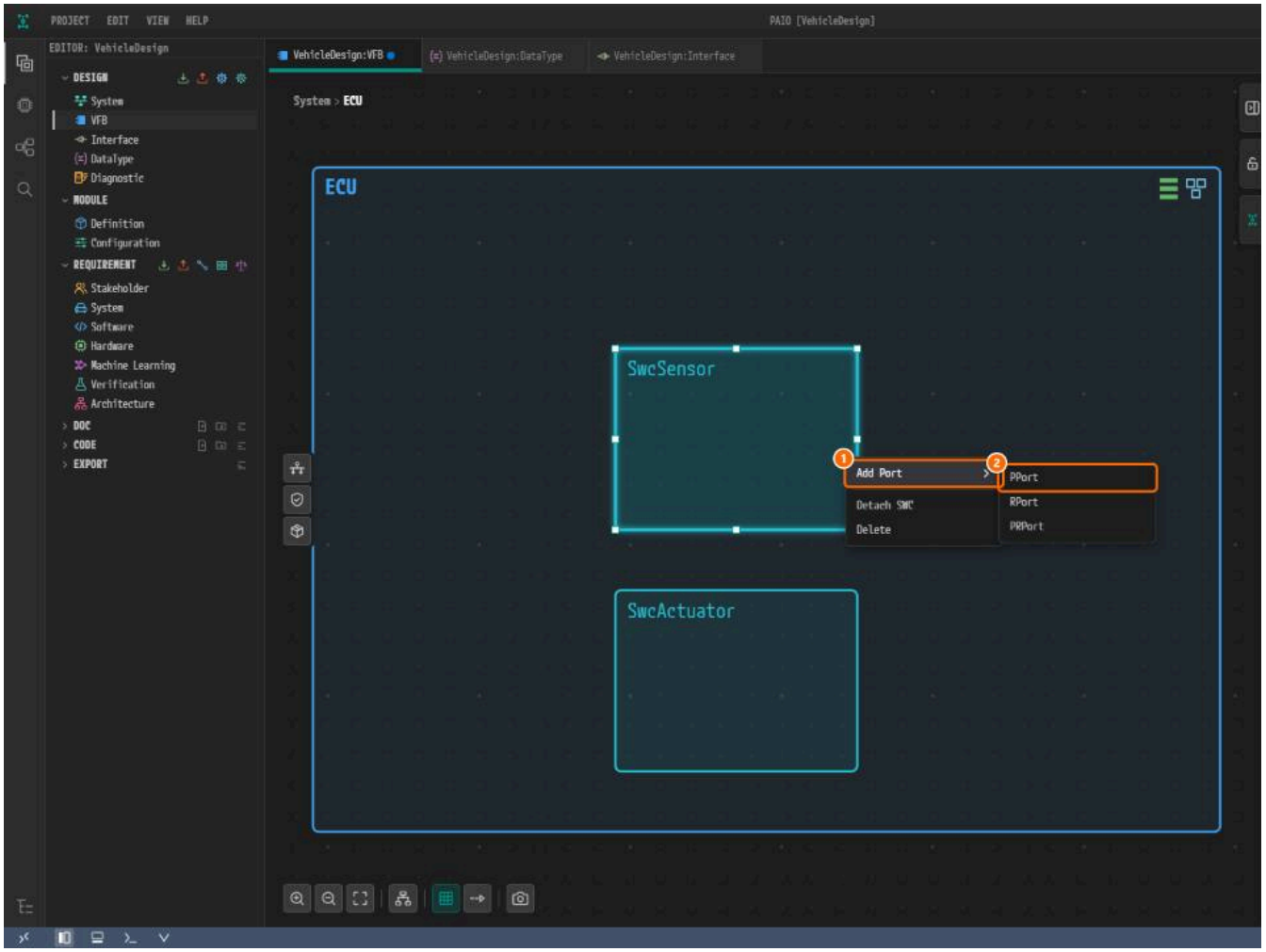


*그림 4-3. ① **Add Port**, ② **PPort**.*

2. 새 포트를 누른다. **NAME** 에 `PpSpeed` 를 입력하고 **PORT INTERFACE** 를 열어 `IfSpeed` 를 누른다.

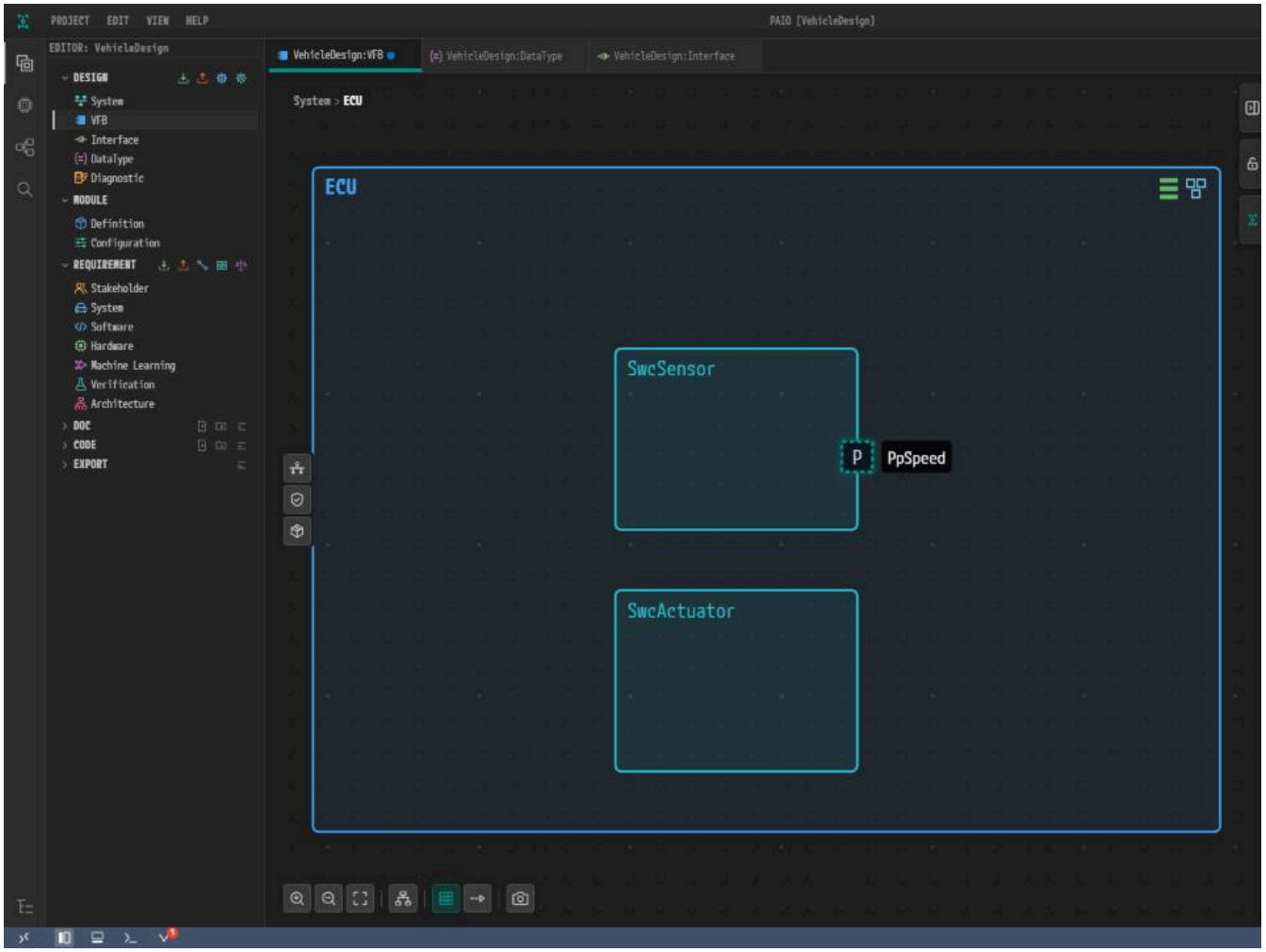


*그림 4-4. ① **NAME**, ② **PORT INTERFACE** 목록의 `IfSpeed`.*

3. 패널에 데이터 요소 `speed` 의 행이 나타난다. **COMSPEC** 열의 **NON-QUEUED** 를 누른다.

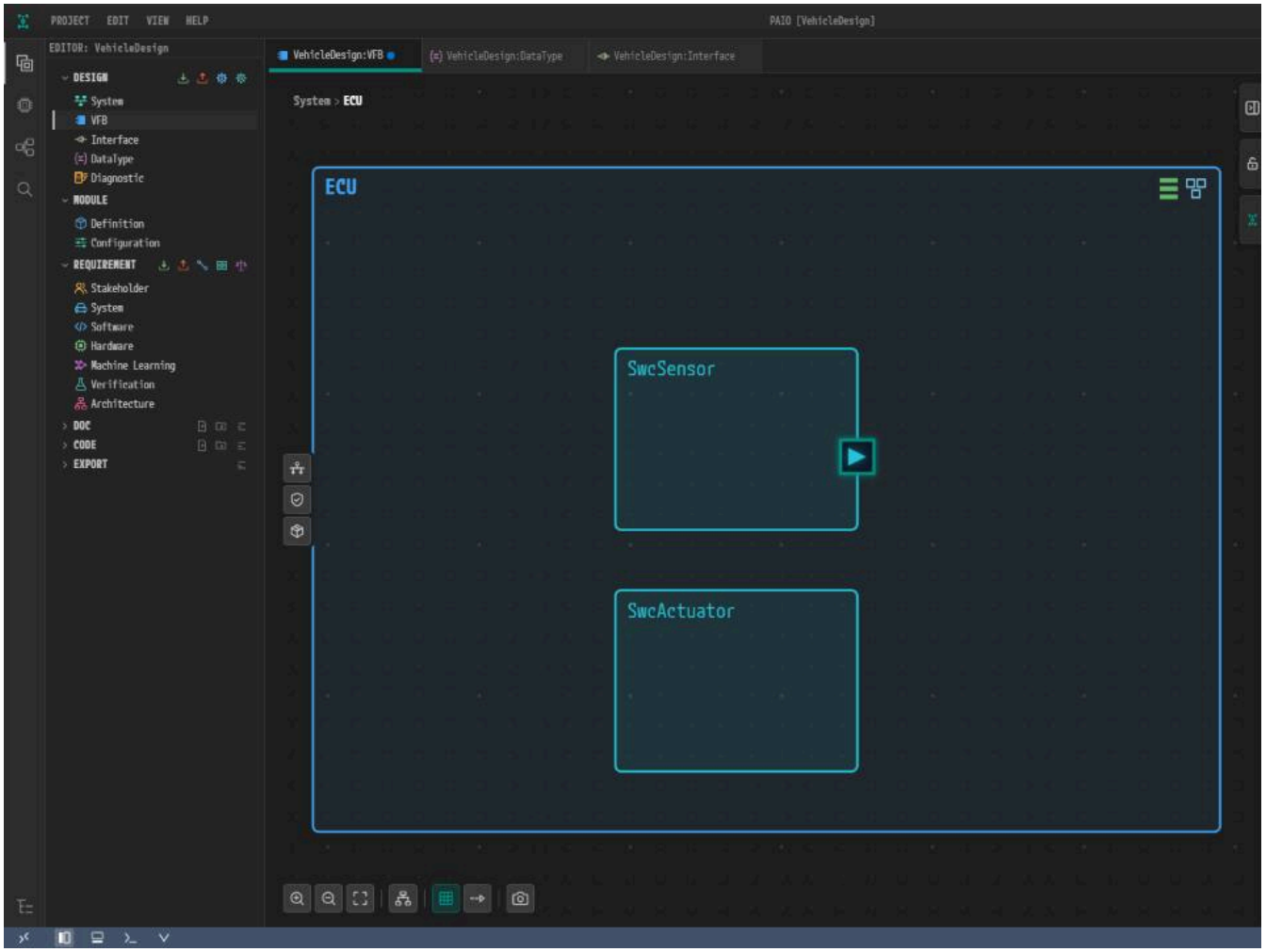


*그림 4-5. ① **NON-QUEUED**.*

4. 왼쪽 가장자리 근처에서 **RPort** 로 `RpScale` 을 같은 방법으로 만든다. Client-Server 포트의 행에는 오퍼레이션 `GetScale` 이 있다. **CLIENT** 를 누른다.

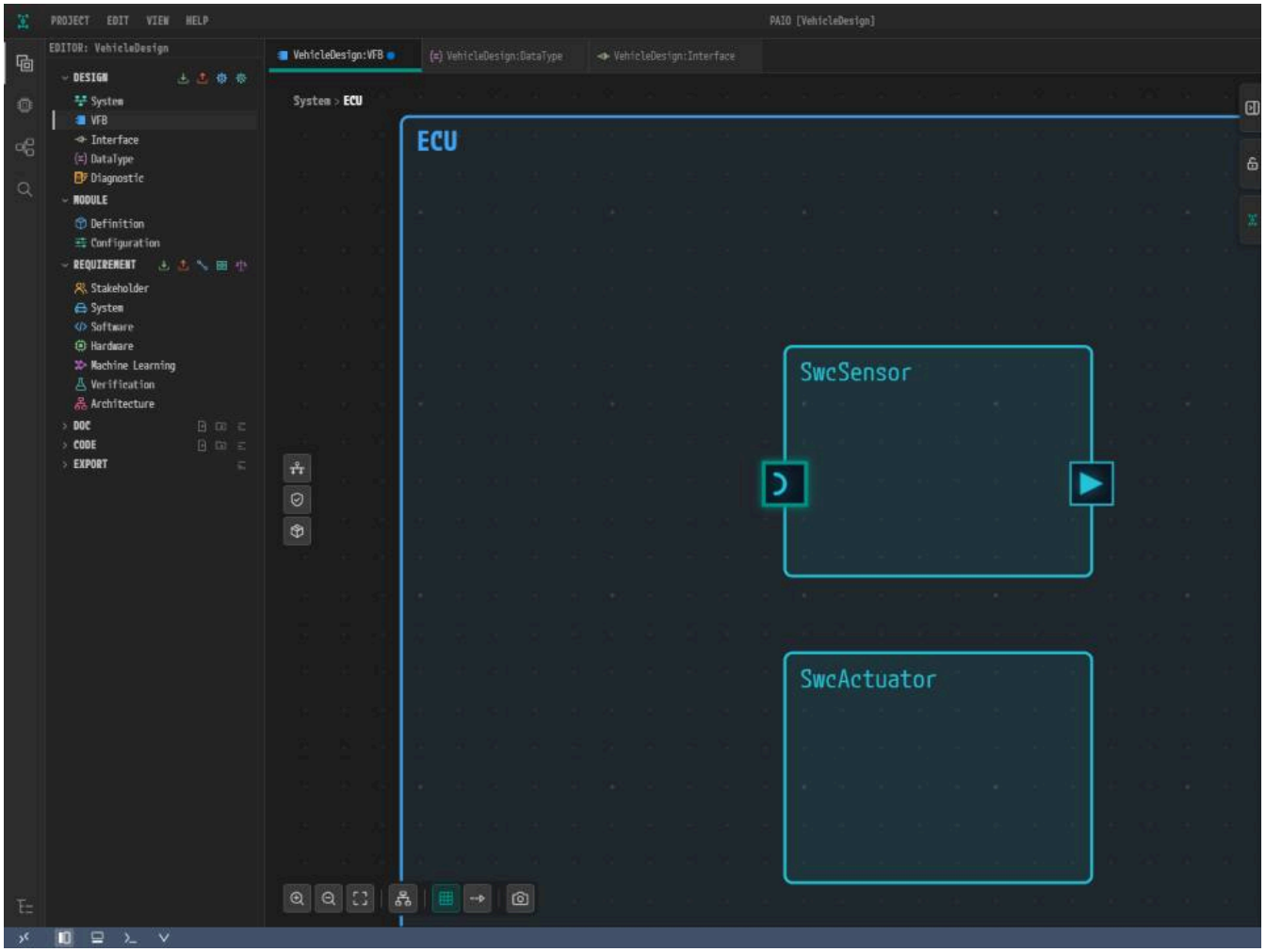


*그림 4-6. ① **CLIENT**.*

5. 위 표대로 `SwcActuator` 에 `RpSpeed` 와 `PpScale` 을 만든다.

6. `Esc` 를 누르고 `Ctrl+S` 로 저장한다. 캔버스 아래 도구 모음의 **Auto Layout** 으로 프레임을 정리하고 다시 저장한다.

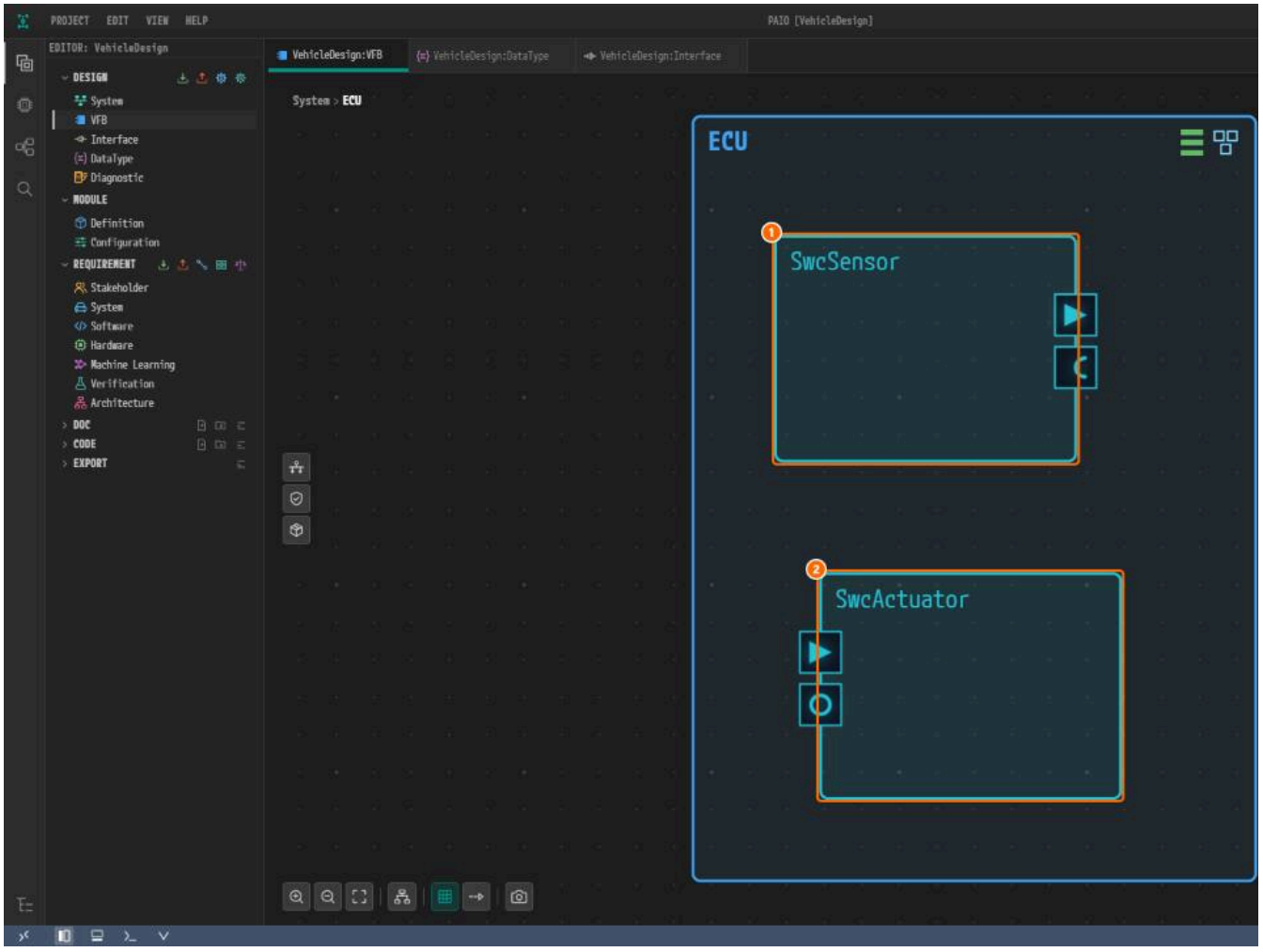


그림 4-7. ① `SwcSensor`, ② `SwcActuator`. 위치는 Auto Layout 이 정하므로 독자의 화면과 순서가 다를 수 있다.

포트 잇기

인터페이스마다 커넥터 하나씩, 두 개의 커넥터로 두 SW-C 를 잇는다.

시작	끝	커넥터 이름(자동으로 채워진다)
`SwcSensor` / `PpSpeed`	`SwcActuator` / `RpSpeed`	`Assembly_SwcSensor_PpSpeed_SwcActuator_RpSpeed`
`SwcActuator` / `PpScale`	`SwcSensor` / `RpScale`	`Assembly_SwcActuator_PpScale_SwcSensor_RpScale`

1. `Shift` 를 누른 채 포트 `PpSpeed` 에서 포트 `RpSpeed` 로 끈다.

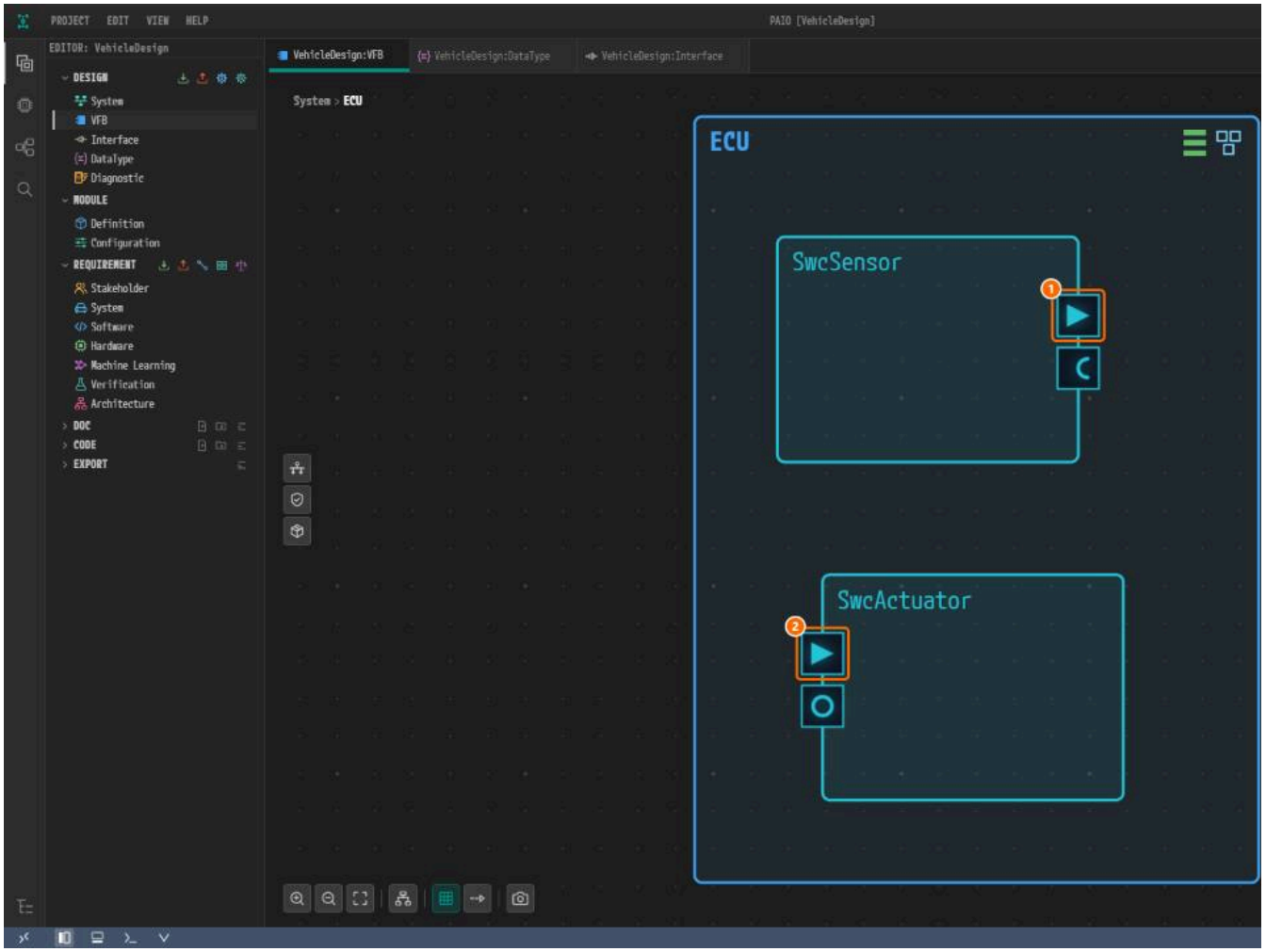


그림 4-8. ① `PpSpeed`, ② `RpSpeed`.

2. `Shift` 를 누른 채 포트 `PpScale` 에서 포트 `RpScale` 로 끈다.

3. 캔버스 아래 도구 모음의 **Auto Layout** 을 누르고 `Ctrl+S` 로 저장한다.

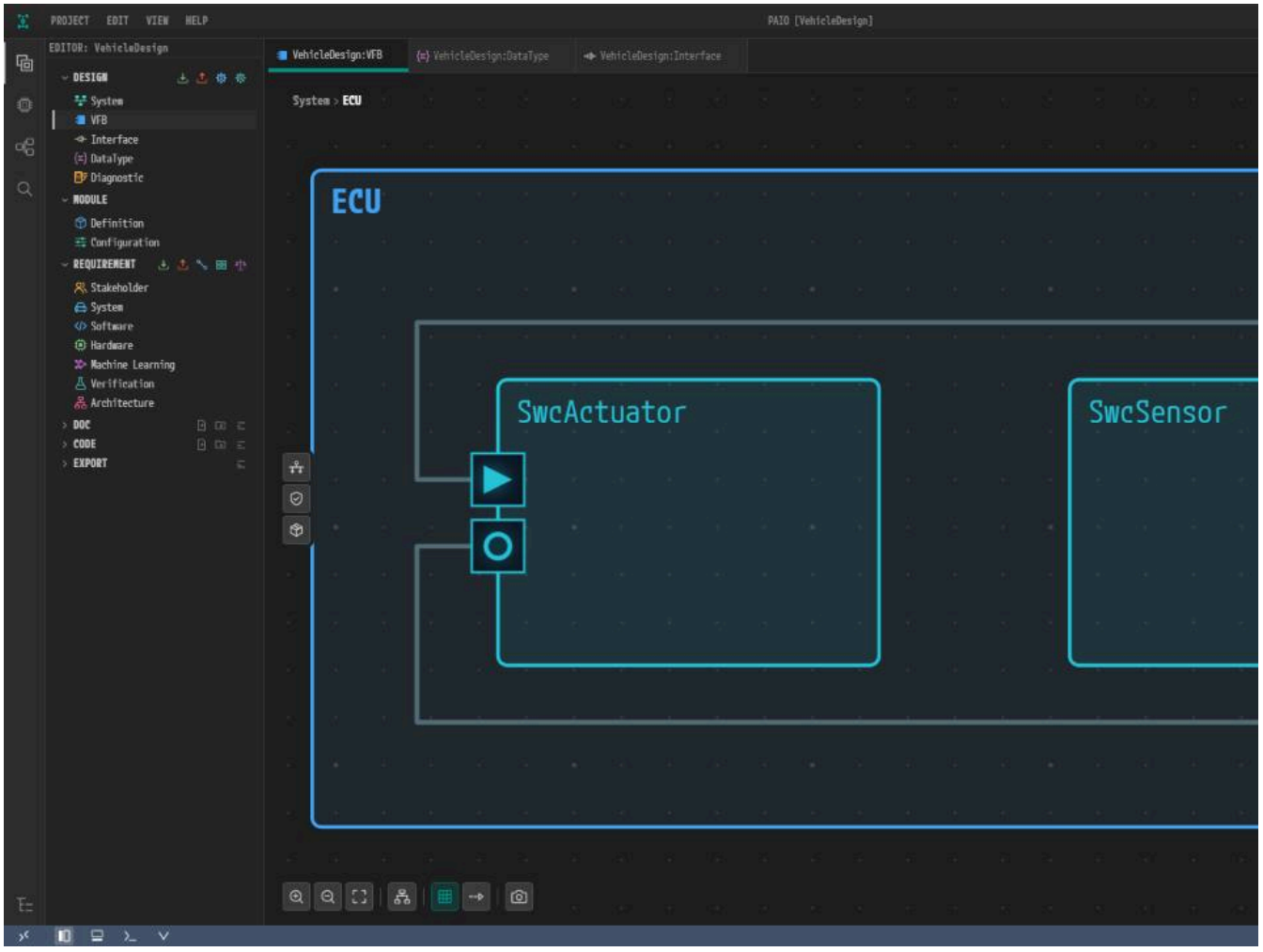


그림 4-9. 두 커넥터. 위치는 독자의 화면과 다를 수 있다.

왜 이렇게 설계하는가

- **이름 규칙.** `Pp` 는 제공 포트, `Rp` 는 요구 포트를 뜻하므로 이름만 보고 방향을 안다.
- **NON-QUEUED.** 수신 측에 필요한 것은 모든 속도 값이 아니라 최신 속도 값이다.
- **언제나 제공 포트에서 요구 포트로.** 커넥터는 같은 인터페이스의 제공자와 요구자를 잇는다. 두 번째 끌기가 `SwcActuator` 에서 `SwcSensor` 로 가는 것은 여기서 액추에이터

확인하기

- [ ] `SwcSensor` 에 포트 `PpSpeed`, `RpScale` 이 있다.
- [ ] `SwcActuator` 에 포트 `RpSpeed`, `PpScale` 이 있다.
- [ ] 포트를 선택하면 표의 **PORT INTERFACE** 와 설정이 보인다.
- [ ] 커넥터 두 개가 SW-C 를 잇고 있다.
- [ ] 탭 `VehicleDesign:VFB` 에 수정 표시가 없다.

다음 단계

Internal Behavior에서 이 포트들을 쓰는 러너블을 추가한다.

Internal Behavior

이 장에서 할 일

이 장에서는 두 SW-C 의 러너블을 추가한다. 러너블은 RTE 가 호출하는 C 함수다. 액세스 포인트는 러너블이 어느 포트를 쓰는지를, RTE 이벤트는 언제 실행되는지를 정한다.

시작하기 전에

- SW-C와 포트를 마친 뒤, 탭 `VehicleDesign:VFB` 의 컴포지션 `ECU` 안에서 시작한다.

SW-C	러너블(NAME, SYMBOL)	액세스 포인트	RTE 이벤트	
`SwcSensor`	`Sensor_Cyclic`	`PpSpeed` 에서 **DataSendPoint**, `RpScale` 에서 **SyncServerCallPoint**	**TimingEvent** `SensorCyclicTiming`	**PERIOD** 0
`SwcActuator`	`Actuator_Init`		**InitEvent** `ActuatorInitEvent`	
`SwcActuator`	`Actuator_OnData`	`RpSpeed` 에서 **DataReceivedPoint**	**DataReceivedEvent** `ActuatorOnDataEvent`	**R-PORT**
`SwcActuator`	`Actuator_GetScale`		**OperationInvokedEvent** `ActuatorGetScaleEvent`	**P-PORT**

`Actuator_GetScale` 은 **CAN BE INVOKED** = **TRUE** 로도 설정한다.

SwcSensor

1. `SwcSensor` 를 더블클릭한다. **SWC Internal Behavior [SwcSensor]** 창이 열린다.
2. SW-C 본체를 오른쪽 클릭하고 **Add Runnable** 을 누른다. 러너블이 포트와 겹치면 창 도구 모음의 **Auto Layout** 을 누른다.

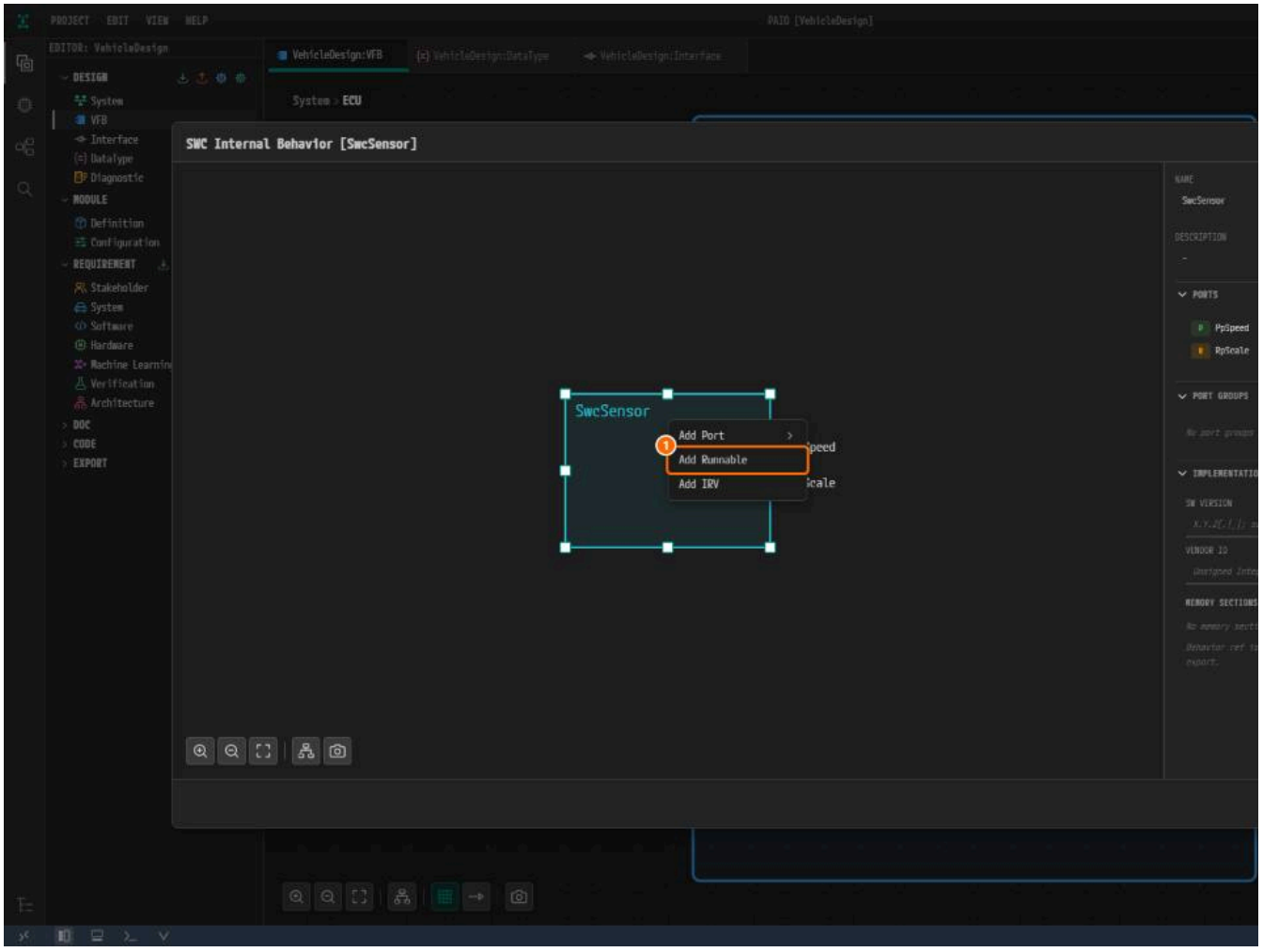


*그림 5-1. ① **Add Runnable.***

3. 러너블을 누르고 **NAME** 과 **SYMBOL** 에 `Sensor_Cyclic` 을 입력한다.

4. `Shift` 를 누른 채 포트 `PpSpeed` 에서 러너블로 끈다. 열린 메뉴에서 **DataSendPoint** 를 누른다.

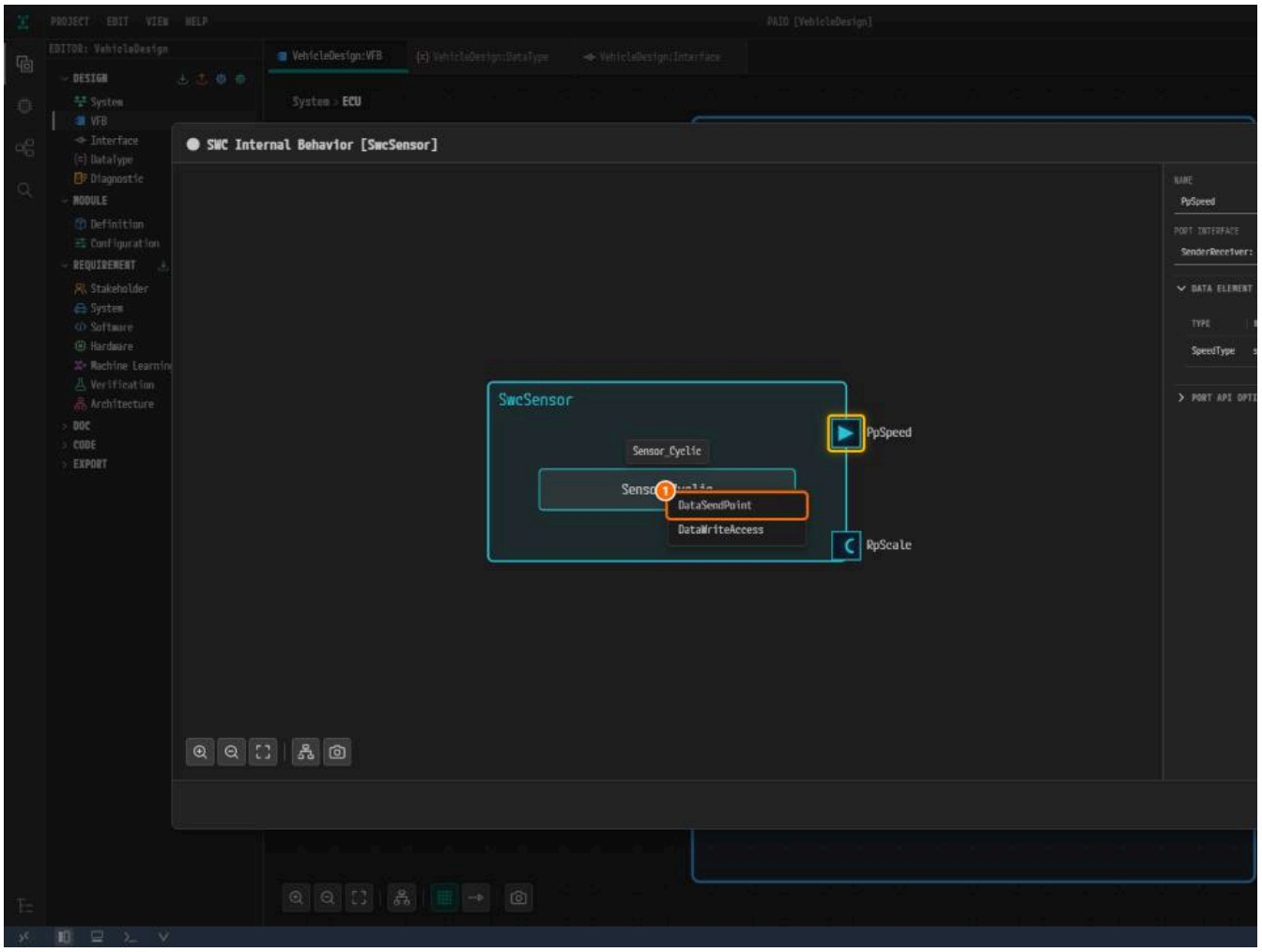


*그림 5-2. ① **DataSendPoint**.*

5. `Shift` 를 누른 채 포트 `RpScale` 에서 러너블로 끌고 **SyncServerCallPoint** 를 누른다.

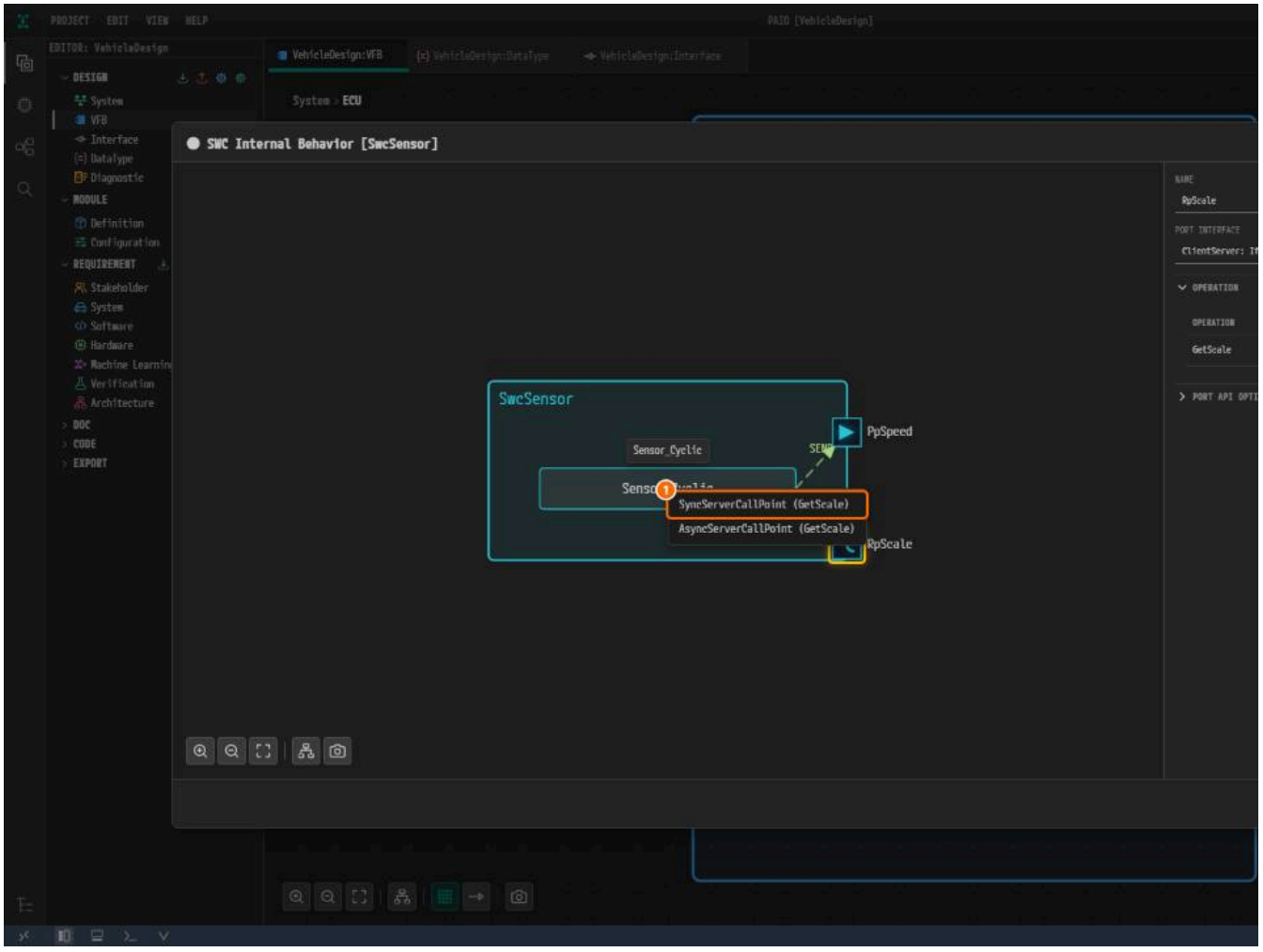


*그림 5-3. ① **SyncServerCallPoint.***

이제 **ACCESS POINTS** 에 `Rte_Write_PpSpeed_speed` 와 `Rte_Call_RpScale_GetScale` 이 있다. 러너블이 호출할 수 있는 RTE 함수이며, 호출 코드는 직접 작성한다

6. 러너블을 다시 누르고 **RTE EVENTS** 머리글의 **+** 를 누른 뒤 **TimingEvent** 를 누른다.

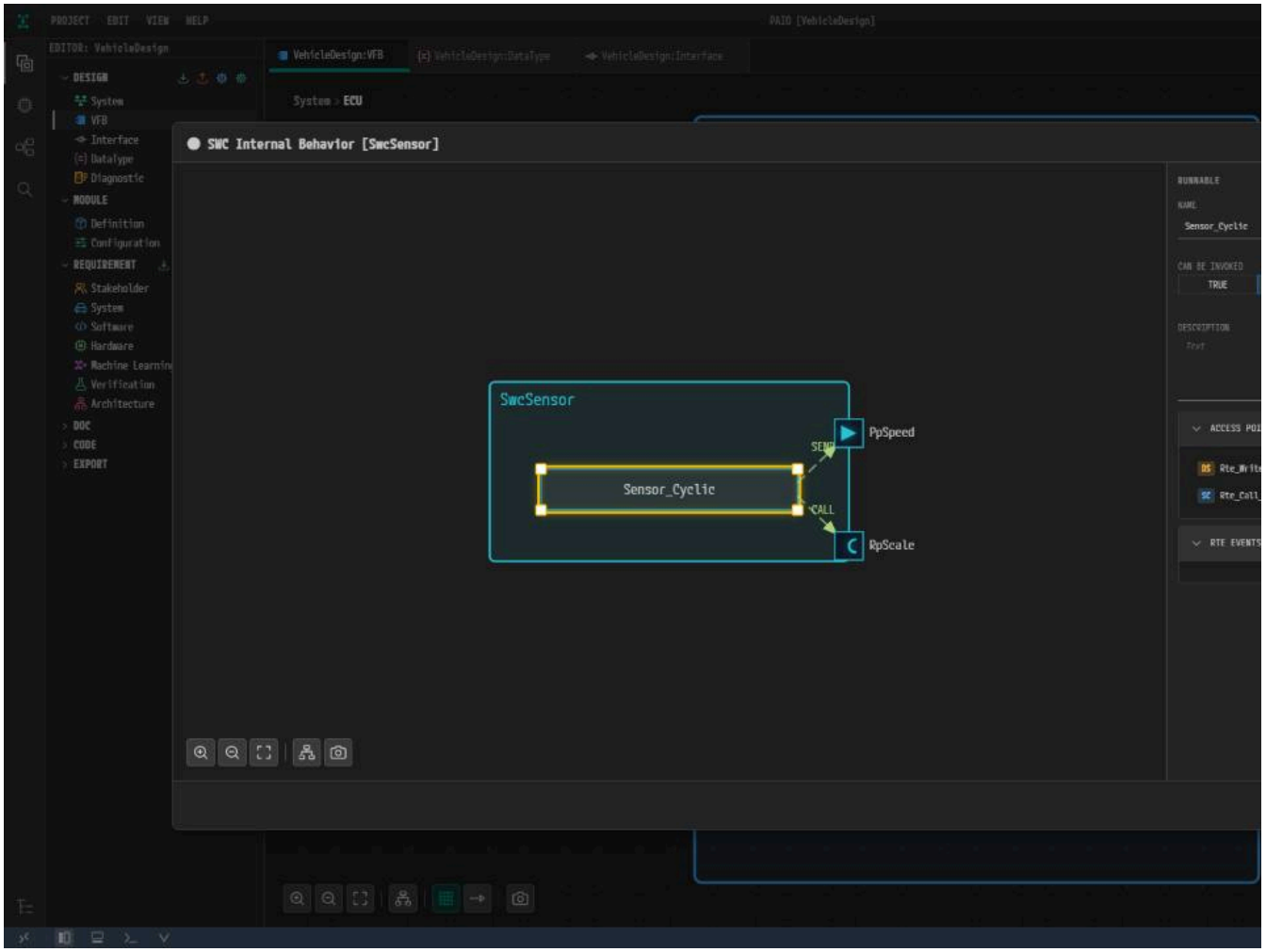


*그림 5-4. ① **+**(Add RTE Event), ② **TimingEvent**.*

7. **NAME** 에 `SensorCyclicTiming`, **PERIOD** 에 `0.02` 를 입력한다. 단위는 `s` 그대로 둔다.

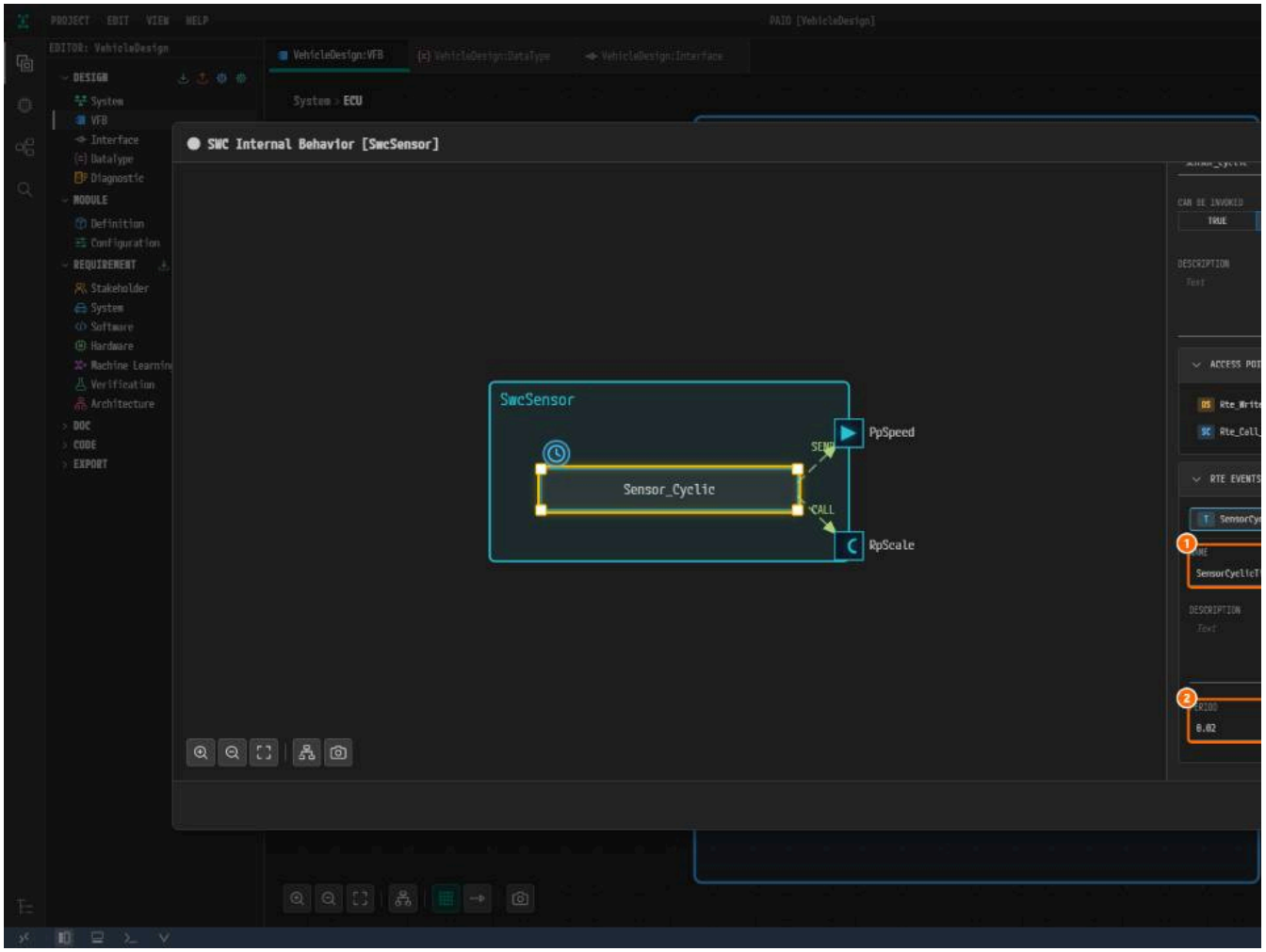


*그림 5-5. ① **NAME**, ② **PERIOD**.*

8. **Auto Layout** 을 누르고 캔버스의 빈 곳을 누른다. `Ctrl+S` 로 저장하고 **Close** 를 누른다.

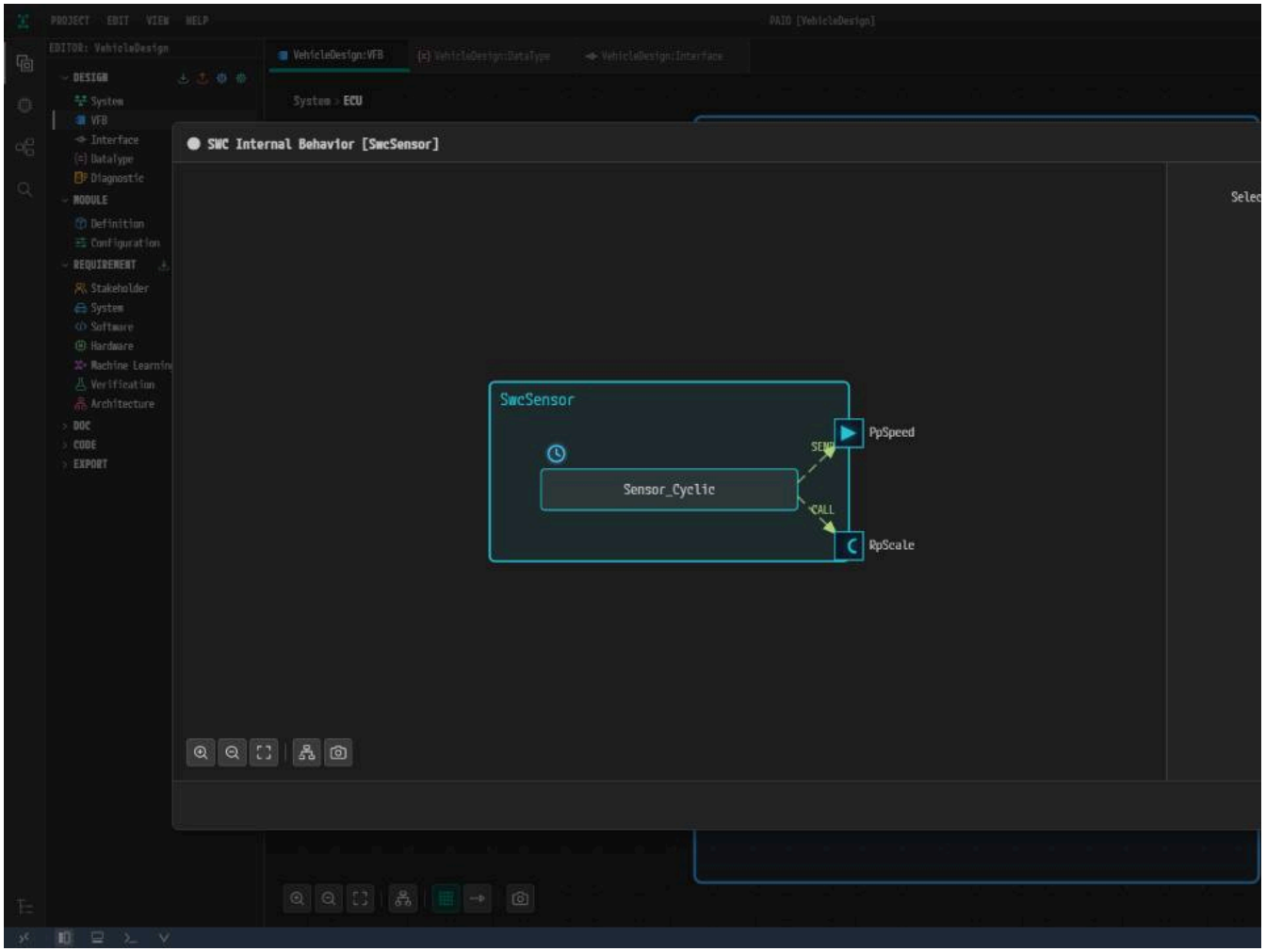


그림 5-6. 러너블과 액세스 포인트를 갖춘 `SwcSensor`.

SwcActuator

`SwcActuator` 를 더블클릭하고 표대로 러너블 세 개를 같은 방법으로 추가한다.

1. `Actuator_OnData` 의 **DataReceivedEvent** 는 포트와 데이터 요소를 묻는다. 이 둘을 먼저 고른 뒤 이벤트의 **NAME** 을 입력한다.

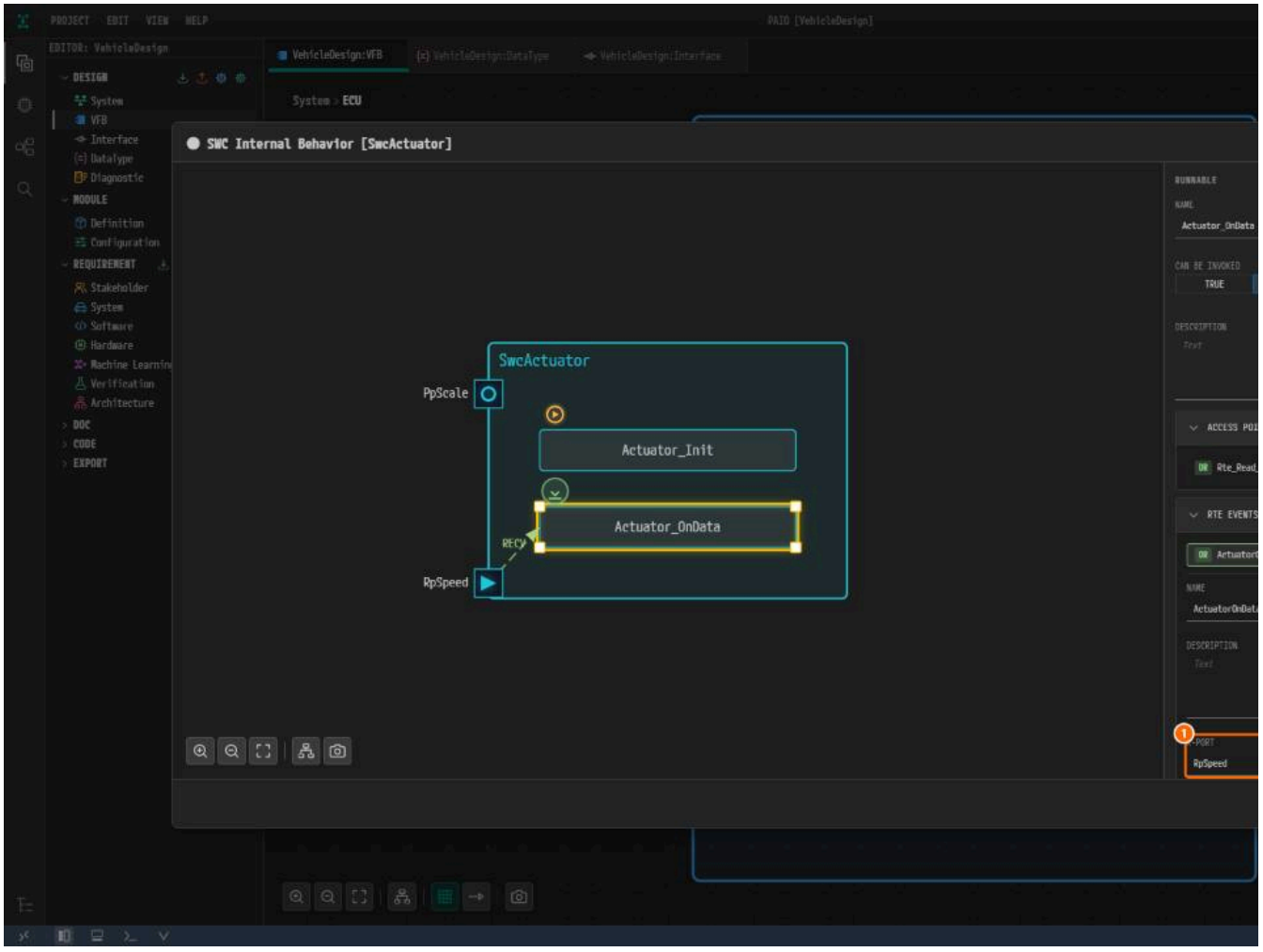


*그림 5-7. ① **R-PORT**, ② **DATA ELEMENT**.*

2. `Actuator_GetScale` 에서는 **CAN BE INVOKED** 의 **TRUE** 를 누른다.

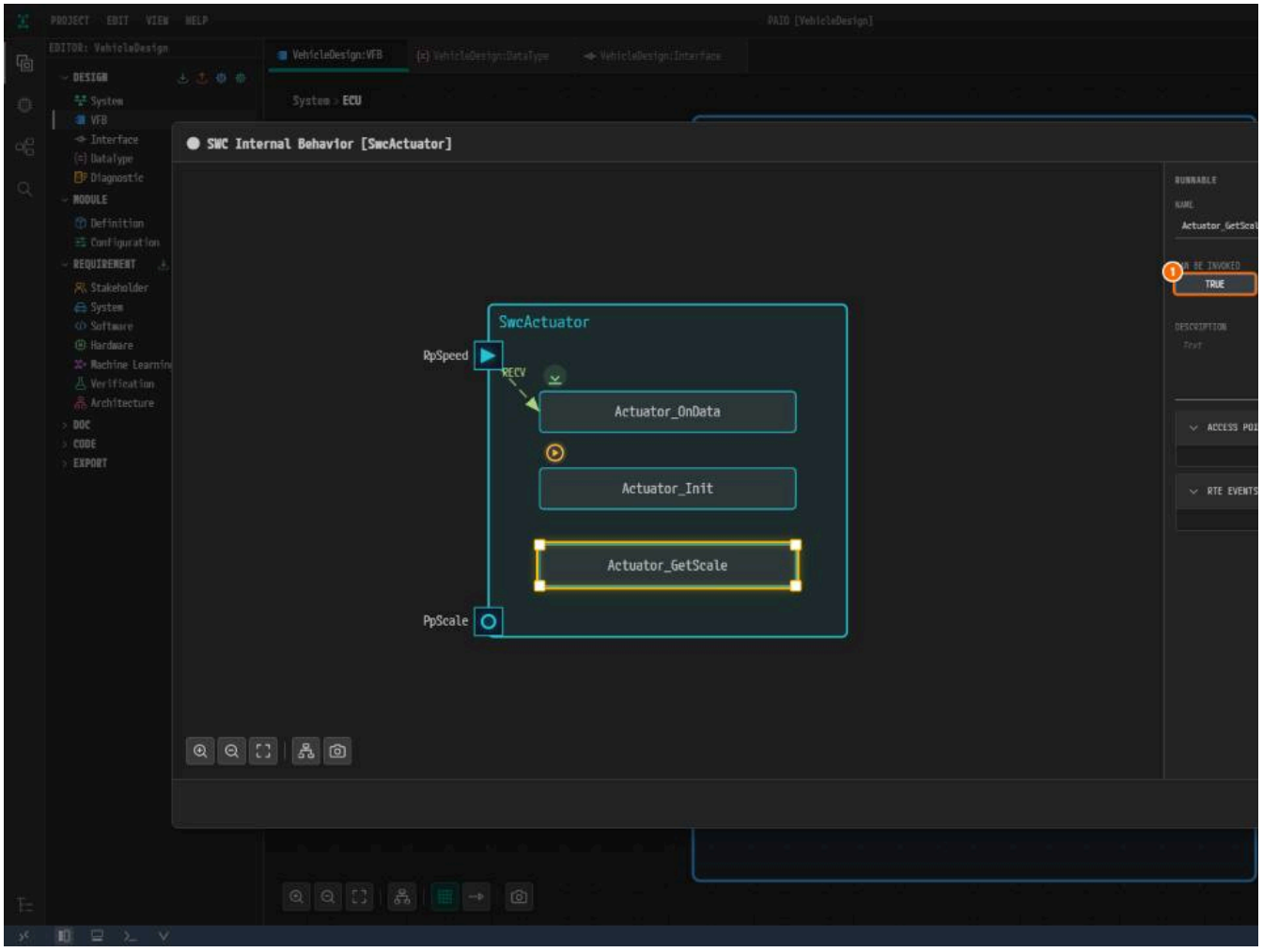


*그림 5-8. ① **TRUE**.*

3. **OperationInvokedEvent** 는 포트와 오퍼레이션을 묻는다.

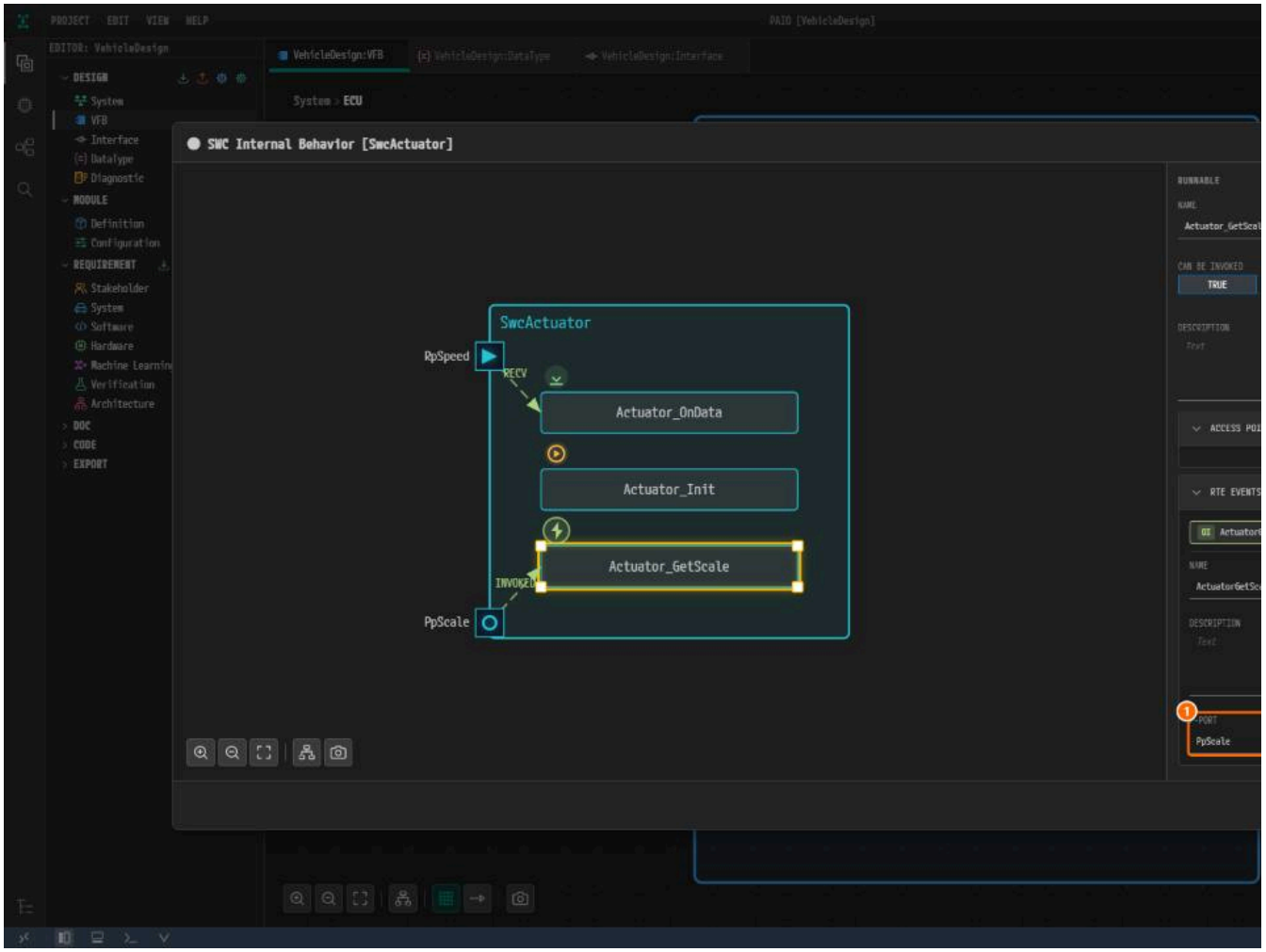


*그림 5-9. ① **P-PORT**, ② **OPERATION**.*

4. **Auto Layout** 을 누르고 캔버스의 빈 곳을 누른 뒤 `Ctrl+S` 로 저장하고 **Close** 를 누른다. 탭도 `Ctrl+S` 로 저장한다.

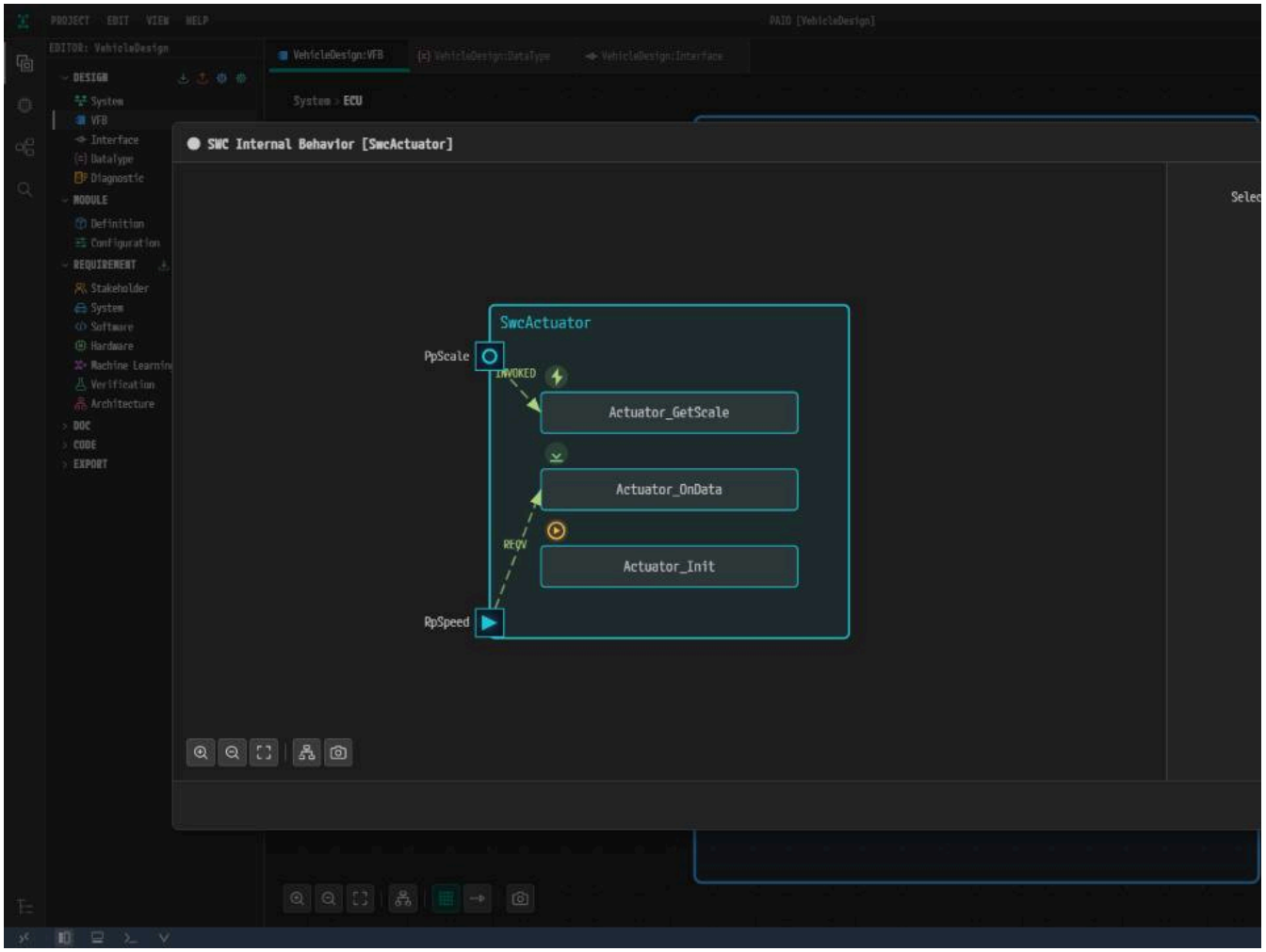


그림 5-10. 러너블 세 개를 갖춘 `SwcActuator`.

> **주의.** `Actuator_GetScale` 의 **TRUE** 를 빠뜨리지 않는다. 8장에서 이 이벤트는 어느 태스크에도 매핑되지 않는데, 코드 생성기는 동시 호출할 수 있는 서버 러너블에만 이를 허
> `RTE_CS_SERVER_UNMAPPED_NOT_CONCURRENT` 로 멈춘다.

왜 이렇게 설계하는가

- `Sensor_Cyclic` **은 타이머로 실행된다.** RTE 가 20ms 마다 시작하고, 액세스 포인트가 속도 쓰기와 배율 요청을 허용한다. 호출 자체는 나중에 직접 작성하는 코드다.
- `Actuator_OnData` **는 도착할 때 실행된다.** 폴링하지 않고, `speed` 가 도착하면 이벤트가 시작시킨다.
- `Actuator_Init` **은 시작할 때 한 번 실행된다.** 한 번만 하는 준비 작업을 두는 곳이다.
- `Actuator_GetScale` **은 호출될 때 실행된다.** 자기 태스크 없이 호출자의 호출 안에서 바로 실행된다.

확인하기

- [ ] `SwcSensor` 에 러너블 `Sensor_Cyclic` 과 액세스 포인트 두 개, 이벤트 `SensorCyclicTiming` 이 있다.
- [ ] `SwcActuator` 에 러너블 `Actuator_Init`, `Actuator_OnData`, `Actuator_GetScale` 과 각각의 이벤트가 있다.
- [ ] `Actuator_GetScale` 의 **CAN BE INVOKED** 가 **TRUE** 다.

다음 단계

PRTE Setup에서 BSW 모듈을 가져오고 시동 모듈 `EcuM`, `BswM`, `Mcu`, `Det` 를 설정한다.

PRTE Setup

이 장에서 할 일

이 장에서는 ECU 의 BSW 설정(ECUC)을 세 단계로 준비한다.

1. VFB 의 **ECU Configuration** 창에서 **PRTE Platform Import** 로 네트워크 관리 모듈 `CanNm`, `Nm` 을 포함한 BSW 모듈을 가져온다. 모듈 정의와 모듈마다 빈 설정 하나가 생긴
2. **ECU Configuration** 에디터에서 `EcuM`, `BswM`, `Mcu`, `Det` 의 시동·하드웨어 값을 손으로 설정한다. 설계가 정하지 않는 이 ECU 고유의 값이다.
3. Rte 값 파일을 임포트한다. 속성 패널이 아직 받지 못하는 `Rte` 참조 여섯 개를 담고 있다.

> **주의.** 이 튜토리얼은 **PRTE AUTO CONFIGURATION** 을 쓰지 않는다. 모든 값을 직접 보고 정하도록 ECU Configuration 에디터에서 BSW 를 모듈별로 손으로 설정한다. **PRTE Al**
> **PRTE Auto Configuration** 아이콘, **Auto-Generate** 를 누르지 않고 **ECU Configuration** 창을 적용하지 않는다. 이것들은 설계에서 BSW 값을 써서 직접 넣은 값을 덮어쓴다.

시작하기 전에

- Internal Behavior를 마친 뒤 시작한다.
- 파일 VehicleDesign_Rte.arxml을 컴퓨터에 받아 둔다.

> **주의.** 이 파일은 이름이 `ECU` 인 ECU 의 값을 담고 있다. AUTOSAR System 튜토리얼에서 만든 노드 `ECU` 의 이름이다. 노드 이름을 바꾸지 않는다.

위치	항목	값
PRTE Platform Import 창	모듈 지도	제안된 선택에 `CanNm`, `Nm` 추가
ECU Configuration [ECU] 창		`Esc` 로 닫기
ECU Configuration 에디터	`EcuM`, `BswM`, `Mcu`, `Det`	아래 표
Conflict Resolution 창	**Module Definition**	**All → Use Imported**

BSW 모듈 가져오기

1. 탭 `VehicleDesign:VFB` 에서 컴포지션 `ECU` 안으로 들어간다. 프레임 오른쪽 위의 ECU 아이콘을 누르면 **ECU Configuration [ECU]** 창이 열린다. 창 오른쪽 위의 **PRTE Pl**
   CONFIGURATION 은 누르지 않는다.

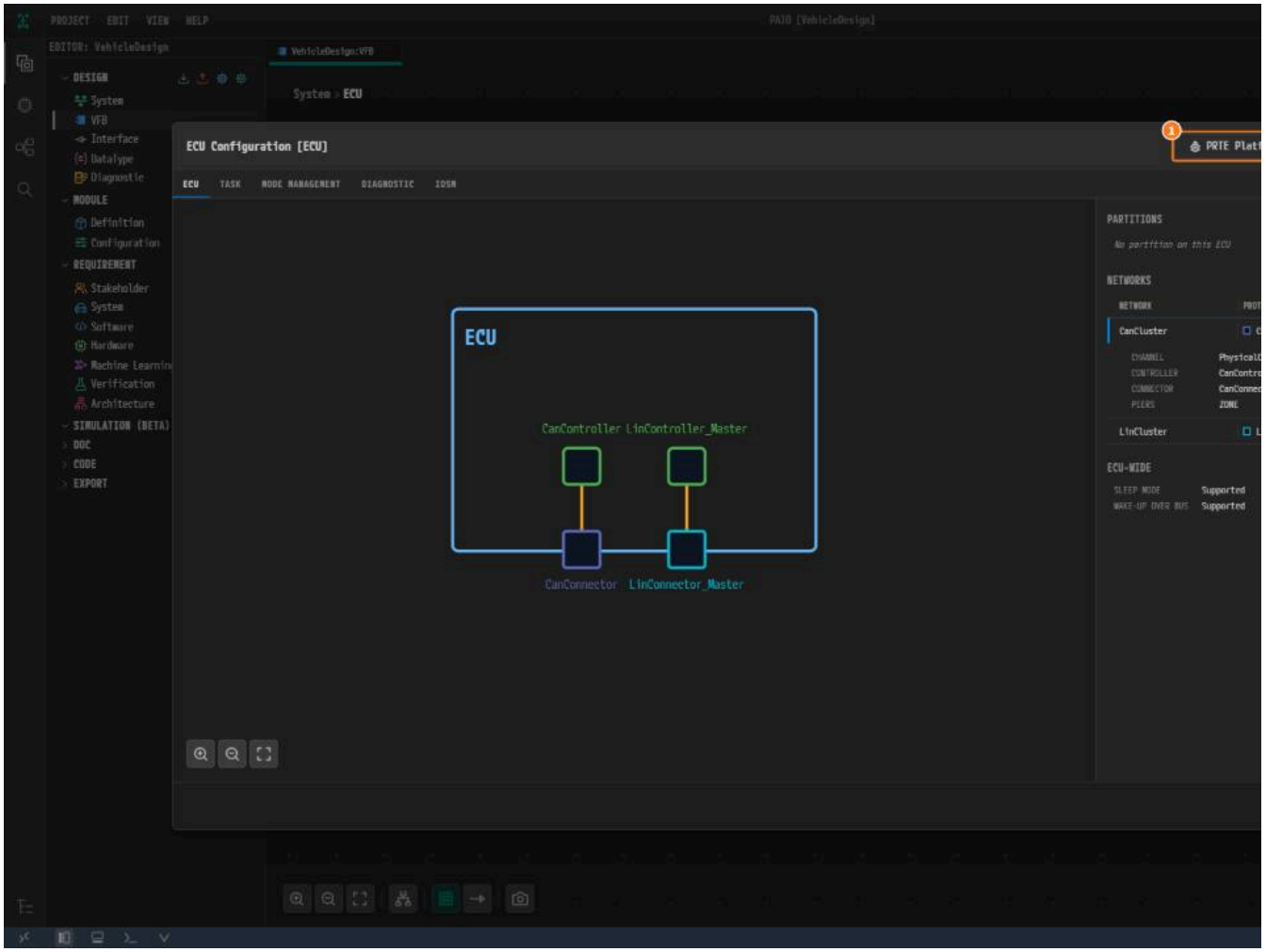


*그림 6-1. ① **PRTE Platform Import**.*

2. **PRTE Platform Import** 창에 BSW 모듈의 지도가 보인다. 밝은 것은 설계에서 이미 선택된 모듈이다. CAN 스택이 선택되어 있다: `Can`, `CanIf`, `LSduR`, `CanSM`, `ComM`, `Co` 있지 않다. `CanNm` 을 누르면 `Nm` 도 함께 선택되고, 아래쪽 줄에 `Selected CanNm, Nm.` 이 보인다. 나머지는 그대로 둔 채 **Import** 를 누른다. `Nm` 은 AUTOSAR 가 Netwo PRTE 에서는 이름이 `Nm` 이다.

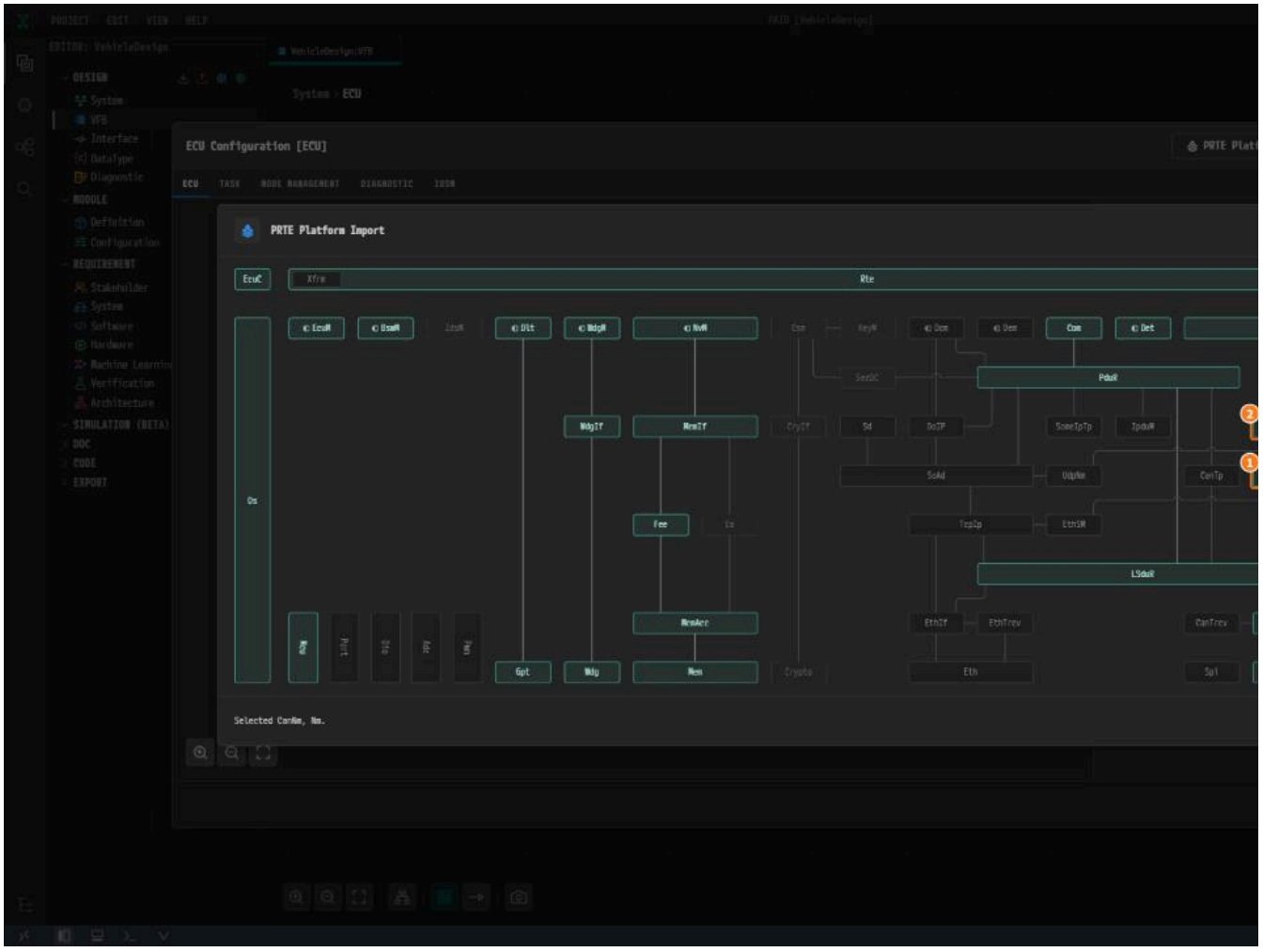


*그림 6-2. ① `CanNm`, ② `Nm`, ③ **Import**.*

진행 표시가 `Complete` 에 이를 때까지 기다린다.

3. `Esc` 로 **ECU Configuration** 창을 닫는다. 이 창에서는 `Ctrl+S` 를 누르지 않는다. 이 창의 `Ctrl+S` 는 창을 적용한다. `Esc` 로 닫으면 아무것도 쓰지 않는다.

> **참고.** 임포트는 BSW 서비스 컴포넌트를 담은 컴포지션 `PRTE_ECU` 를 프레임에 추가하고, BSW 모듈의 인터페이스와 데이터 타입도 추가한다. 그래서 이제 Interface 편집기와 I
> 다. `CanNm` 과 `Nm` 을 선택하면 그 정의와 빈 설정만 추가되고 서비스 컴포넌트는 추가되지 않는다.

ECU Configuration 에디터 열기

MODULE 아래의 **Configuration** 을 누른다. ECU Configuration 에디터가 열린다. 왼쪽의 **ECU** 목록에 ECU 가 나열되고, 가져온 모듈은 `ECU` 아래에 있다. 설정은 비어 있으며, `Rt`

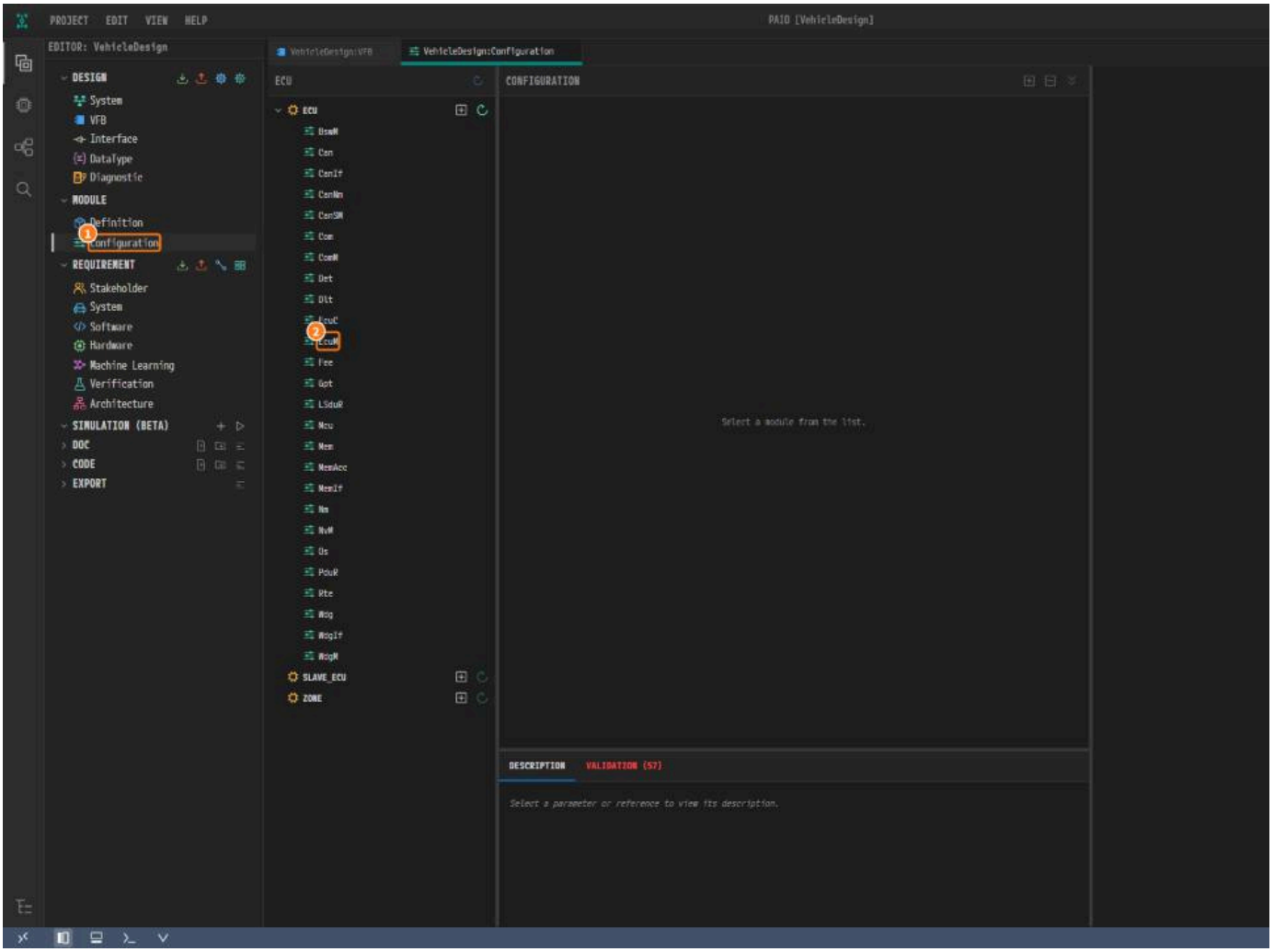


*그림 6-3. ① **Configuration**, ② `ECU` 의 모듈 가운데 하나인 `EcuM`.*

각 ECU 행의 오른쪽 끝에 **Add Module** 과 **PRTE Auto Configuration**(둥근 화살표) 아이콘이 있다. **PRTE Auto Configuration** 은 누르지 않는다.

에디터는 이 장과 뒤의 장의 모든 모듈에서 같은 방식으로 다룬다.

- **CONFIGURATION** 트리에서 행 오른쪽 끝의 **+** 는 그 정의의 컨테이너를 기본값과 함께 추가하고 선택한다. 표에 있는 값만 바꾼다. 숫자는 입력하고 `Tab` 을 누르고, 참조는 목록
- **Container** 열의 굵은 이름은 **+** 가 주는 이름이 아니다. 패널의 **Name** 에 입력하고 `Tab` 을 누른다.
- 대상이 아직 없는 참조는 목록에 나오지 않는다. 표의 경로를 입력하고 `Tab` 을 누른다. 대상이 생길 때까지 경로가 노란색으로 보이고, 참조를 여러 개 받는 필드에는 `(unresol`

EcuM 설정하기

1. 목록의 `EcuM` 을 누른다. **CONFIGURATION** 트리에 `EcuM` 의 컨테이너 정의가 모두 빈 채로 나열된다. `EcuMGeneral` 행의 **+** 를 누른다. 에디터가 `EcuMGeneral` 을 추가하고

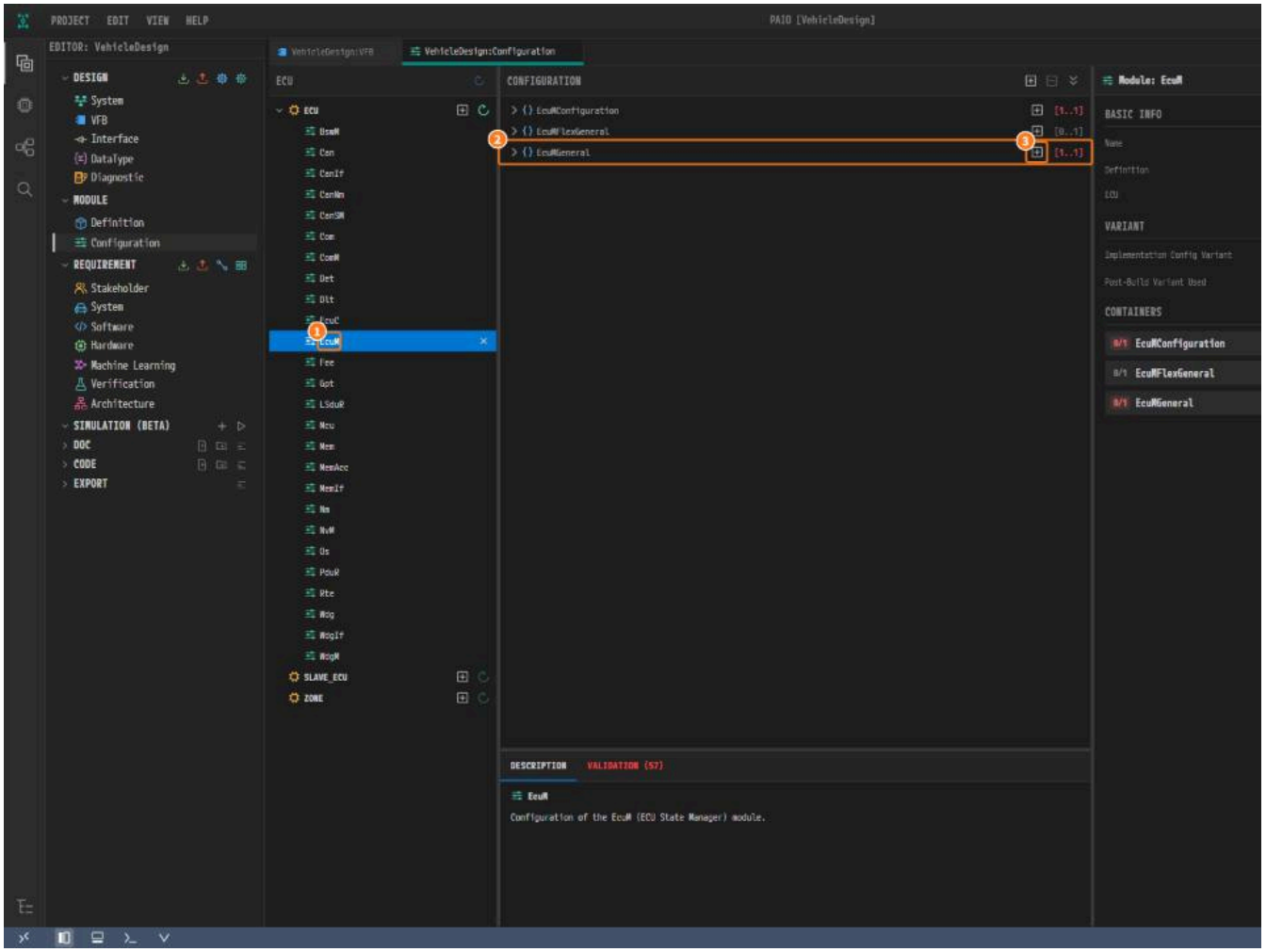


그림 6-4. ① `EcuM`, ② `EcuMGeneral` 행, ③ +.

2. 오른쪽 패널에서 **EcuMDevErrorDetect** 의 **TRUE** 를 누르고 **EcuMMainFunctionPeriod** 에 `0.01` 을 입력한다. **EcuMVersionInfoApi** 는 **FALSE** 로 둔다.

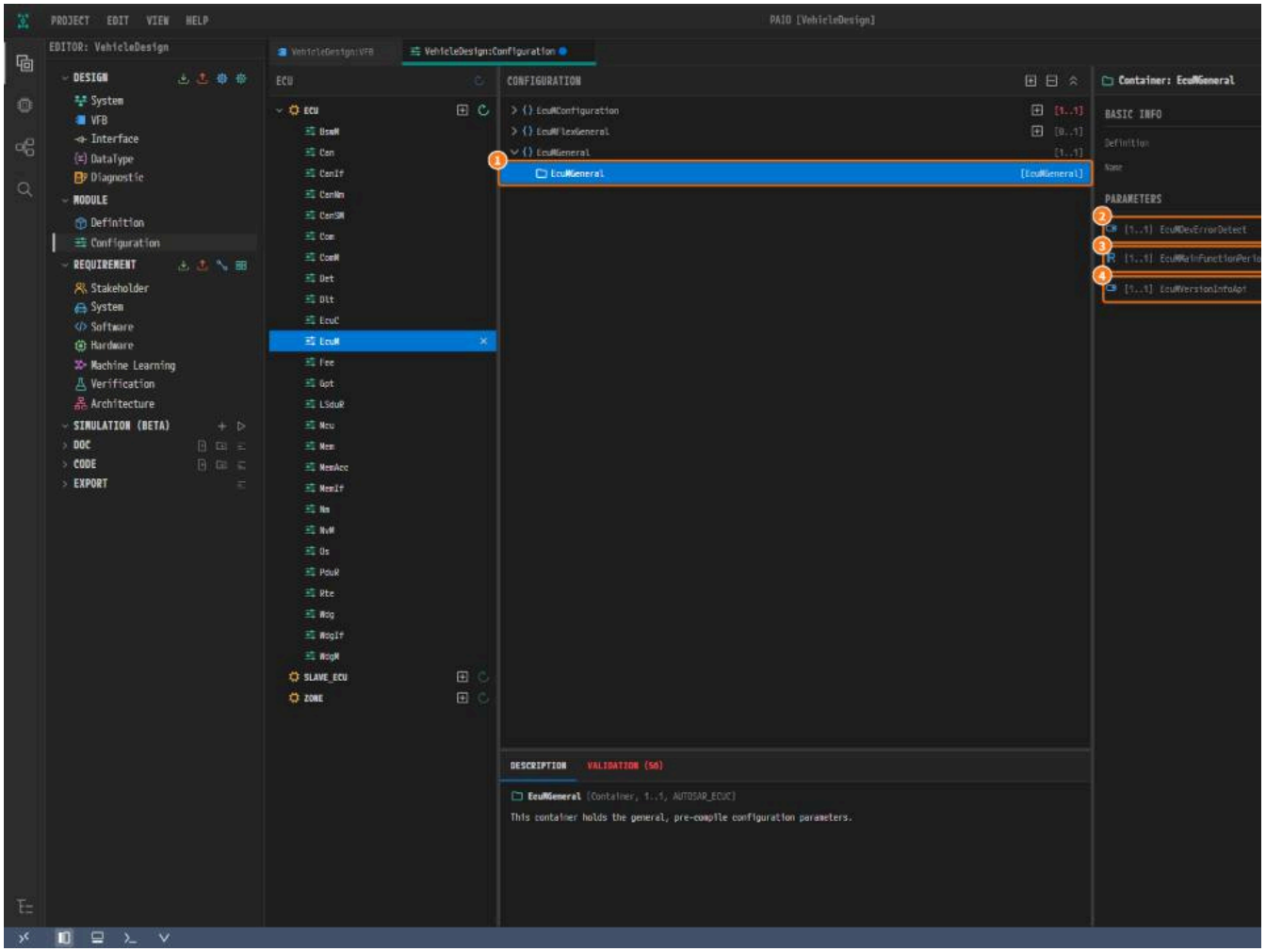


*그림 6-5. ① `EcuMGeneral`, ② **EcuMDevErrorDetect**, ③ **EcuMMainFunctionPeriod**, ④ **EcuMVersionInfoApi**.*

3. 같은 방법으로 `EcuMFlexGeneral`, `EcuMConfiguration`, 그 아래 `EcuMCommonConfiguration` 을 추가하고 각각 표의 값을 넣는다. `EcuMCommonConfiguration` `Os/OSDEFAULTAPPMODE`, **EcuMOSResource** 에 `Os/RES_AUTOSAR_ECUM` 을 입력한다. 둘 다 7장에서 만들므로 그때까지 노란색으로 보이고, `Os/RES_AUTOSAR_ECUM`

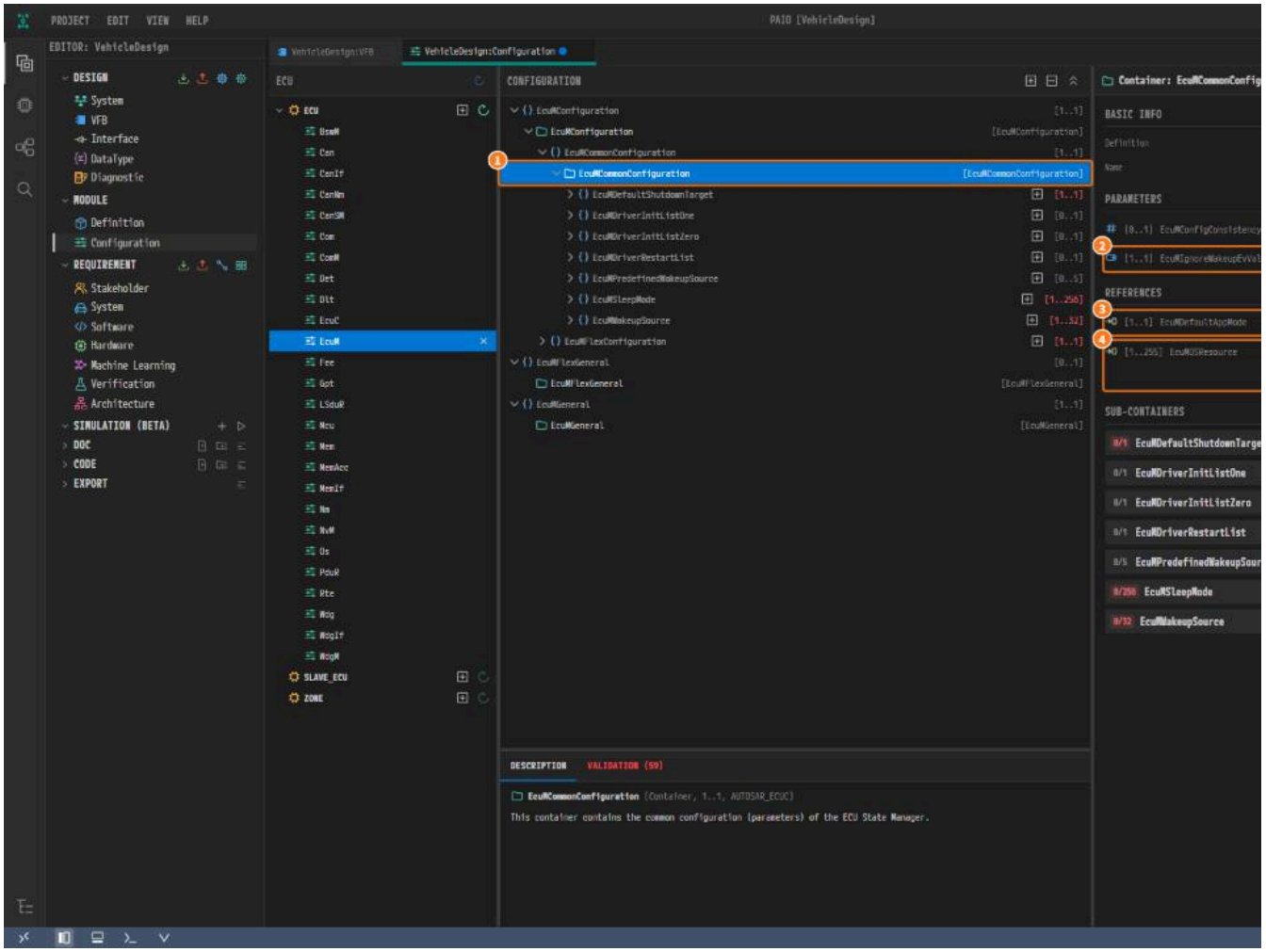


*그림 6-6. ① `EcuMCommonConfiguration`, ② **EcuMIgnoreWakeupEvValOffPreOS**, ③ **EcuMDefaultAppMode**, ④ **EcuMOSResource**.*

4. `EcuMCommonConfiguration` 아래에 `EcuMDriverInitListOne` 을 추가한다. 그 아래 `EcuMDriverInitItem` 행의 **+** 를 누른다.

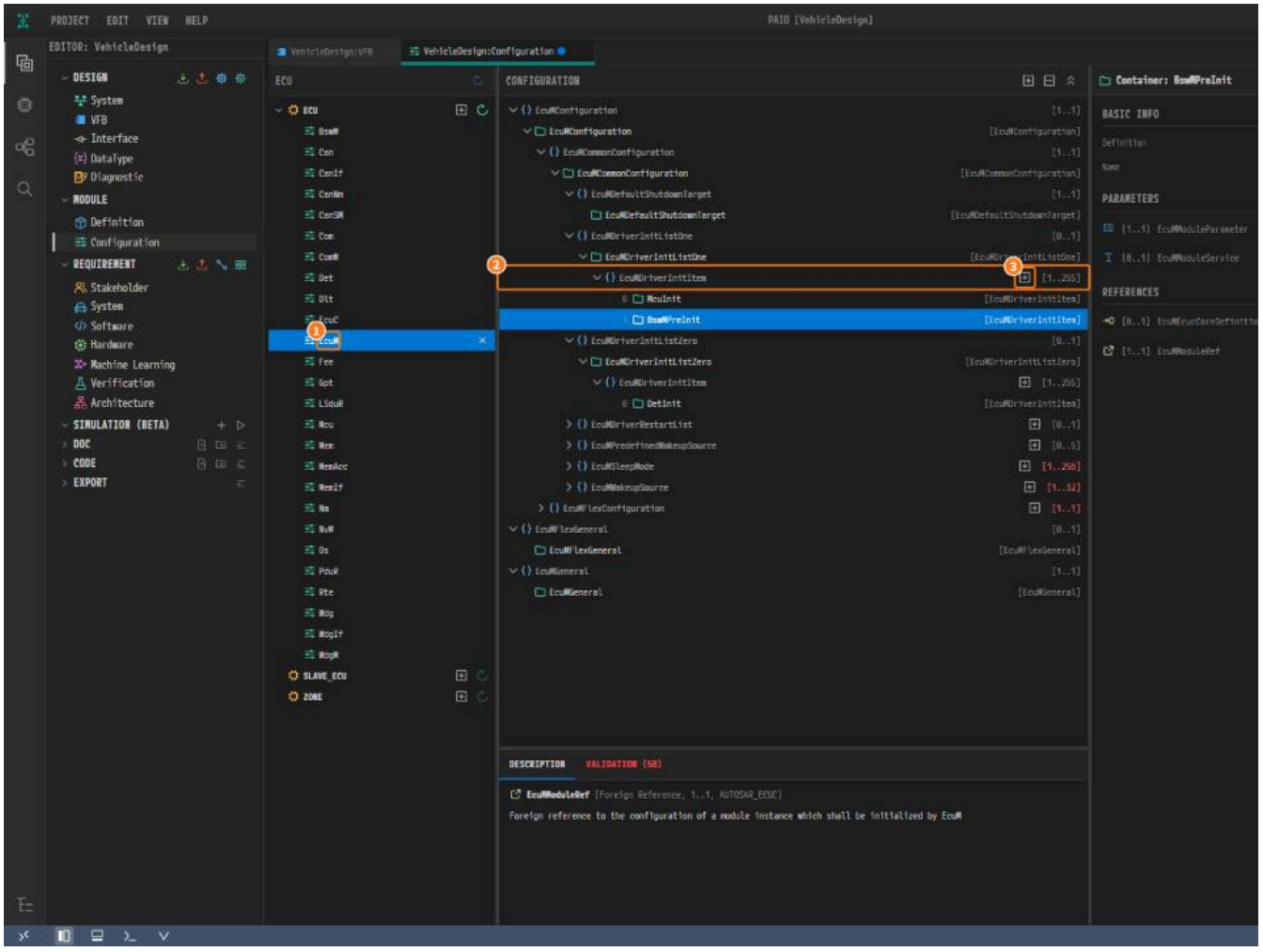


그림 6-7. ① `EcuM`, ② `EcuMDriverInitListOne` 아래의 `EcuMDriverInitItem` 행, ③ +.

5. 에디터가 `EcuMDriverInitItem_0` 을 추가한다. **Name** 을 표의 이름으로 바꾸고, **EcuMModuleParameter** 는 표의 값(`POSTBUILD_PTR`, `NmInit` 만 `NULL_PTR`)을 고 순서대로 추가한다. `EcuM` 이 드라이버를 초기화하는 순서다.

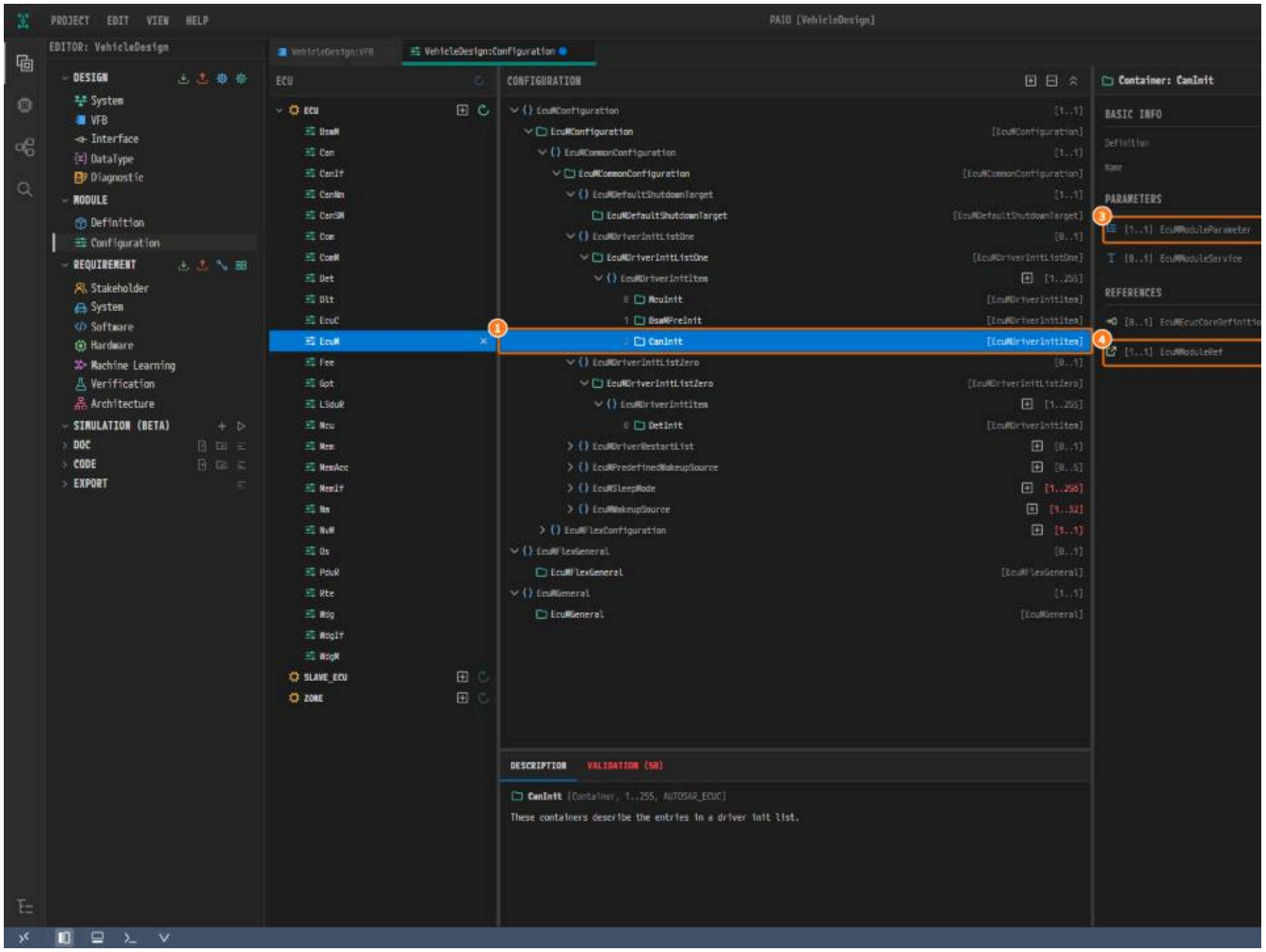


*그림 6-8. ① `CanInit`, ② **Name**, ③ **EcuMModuleParameter**, ④ **EcuMModuleRef**.*

6. 표의 나머지 컨테이너를 추가하고 `Ctrl+S` 로 저장한다.

Container	+ 를 누를 행	
`EcuMGeneral`	`EcuMGeneral`	**EcuMDevErrorDetect TRUE**, **EcuMMainFunction**
`EcuMFlexGeneral`	`EcuMFlexGeneral`	**EcuMAlarmClockPresent FALSE**, **EcuMModeHar** **EcuMSetProgrammableInterrupts FALSE**
`EcuMConfiguration`	`EcuMConfiguration`	없음
`EcuMConfiguration/EcuMCommonConfiguration`	`EcuMCommonConfiguration`	**EcuMDefaultAppMode** `Os/OSDEFAULTAPPMODE`
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDefaultShutdownTarget`	`EcuMDefaultShutdownTarget`	**EcuMDefaultShutdownTarget** `EcuMShutdownTa` `EcuM/EcuMConfiguration/EcuMFlexConfigur` `EcuM/EcuMConfiguration/EcuMCommonConfig`
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListZero`	`EcuMDriverInitListZero`	없음
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListZero/DetInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne`	`EcuMDriverInitListOne`	없음
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/McuInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/BswMPreInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/CanInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/CanIfInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/LSduRInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/PduRInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/ComInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/CanNmInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/NmInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `NULL_PTR`, **EcuMModu**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/ComMInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverInitListOne/CanSMInit`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverRestartList`	`EcuMDriverRestartList`	없음
`EcuMConfiguration/EcuMCommonConfiguration/EcuMDriverRestartList/DetRestart`	`EcuMDriverInitItem`	**EcuMModuleParameter** `POSTBUILD_PTR`, **EcuM**
`EcuMConfiguration/EcuMCommonConfiguration/Power`	`EcuMPredefinedWakeupSource`	**EcuMPredefinedWakeupSource** `POWER`, **EcuMR** `Mcu/McuPublishedInformation/PowerOn`
`EcuMConfiguration/EcuMCommonConfiguration/Reset`	`EcuMPredefinedWakeupSource`	**EcuMPredefinedWakeupSource** `RESET`, **EcuMR** `Mcu/McuPublishedInformation/Software`
`EcuMConfiguration/EcuMCommonConfiguration/WakeupTimer`	`EcuMWakeupSource`	**EcuMWakeupSourceId** `5`, **EcuMValidationTime** **EcuMWakeupSourcePolling FALSE**
`EcuMConfiguration/EcuMCommonConfiguration/Halt`	`EcuMSleepMode`	**EcuMSleepModeId** `0`, **EcuMSleepModeSuspen** `Mcu/McuModuleConfiguration/Sleep`, **EcuMW** `EcuM/EcuMConfiguration/EcuMCommonConfig`
`EcuMConfiguration/EcuMCommonConfiguration/Poll`	`EcuMSleepMode`	**EcuMSleepModeId** `1`, **EcuMSleepModeSuspen** `Mcu/McuModuleConfiguration/Normal`, **EcuM** `EcuM/EcuMConfiguration/EcuMCommonConfig`
`EcuMConfiguration/EcuMFlexConfiguration`	`EcuMFlexConfiguration`	**EcuMNormalMcuModeRef** `Mcu/McuModuleCon`

Container	+ 를 누를 행	
`EcuMConfiguration/EcuMFlexConfiguration/McuReset`	`EcuMResetMode`	**EcuMResetModeId** `0`
`EcuMConfiguration/EcuMFlexConfiguration/VecuMaster`	`EcuMFlexUserConfig`	**EcuMFlexUser** `0`
`EcuMConfiguration/EcuMFlexConfiguration/GoDownUsers`	`EcuMGoDownAllowedUsers`	**EcuMGoDownAllowedUserRef** `EcuM/EcuMConfi`
`EcuMConfiguration/EcuMFlexConfiguration/SupervisionFault`	`EcuMShutdownCause`	**EcuMShutdownCauseId** `4`

BswM 설정하기

`BswM` 은 `EcuM` 이 RUN 을 알리면 RTE 를 시작한다. 모드 요청 포트가 `EcuM` 의 상태를 보고, 조건이 그 상태를 `ECUM_STATE_RUN` 과 비교하고, 규칙이 RTE 를 시작하는 액션 목록

1. 목록의 `BswM` 을 누른다. `BswMGeneral` 을 추가하고 표의 값을 넣는다.

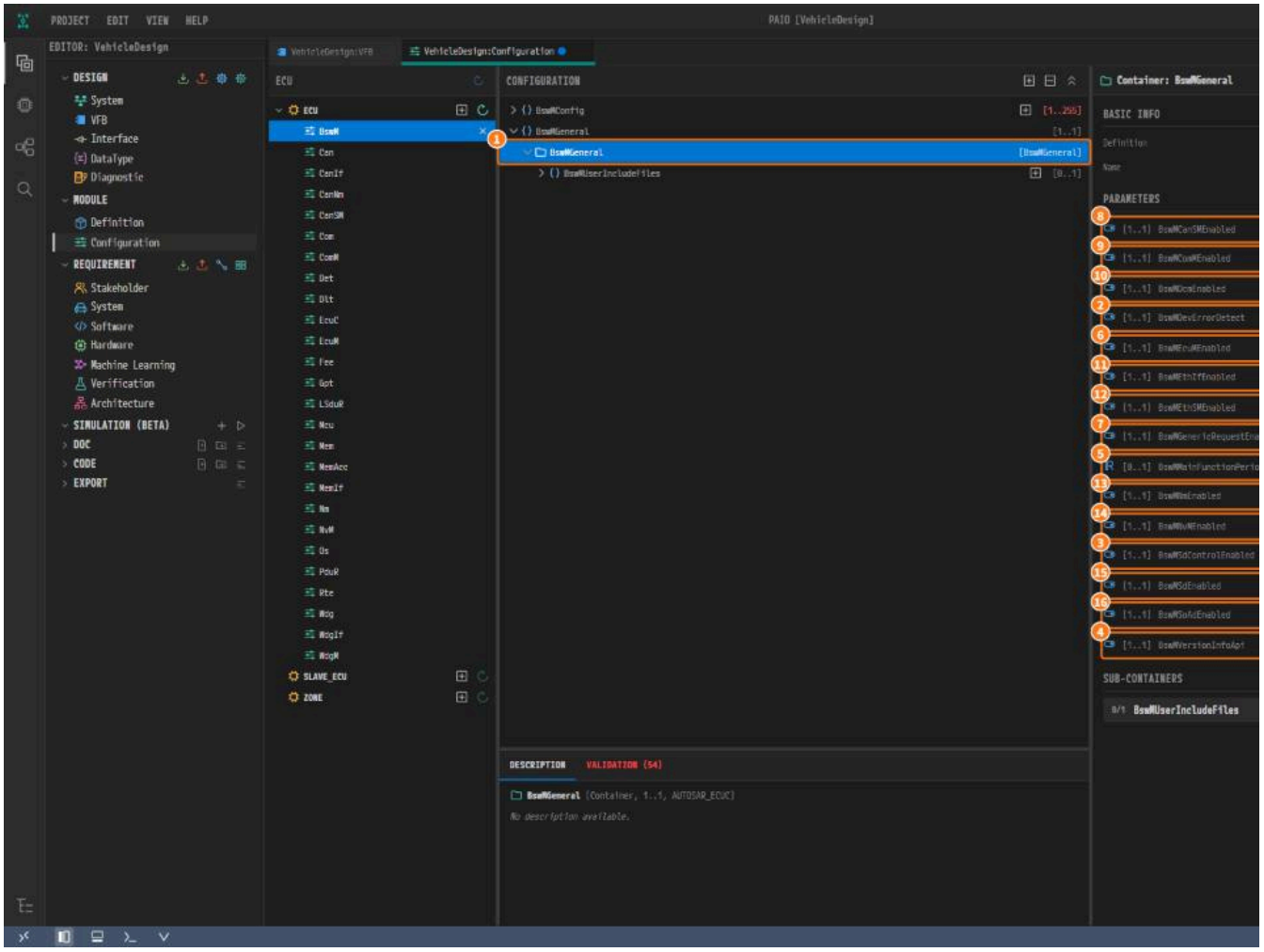


*그림 6-9. ① `BswMGeneral`, ②~⑯ 그 값. 표에서 **TRUE** 인 것은 **TRUE** 를 누르고, **BswMMainFunctionPeriod** 에 `0.01` 을 입력하고, 나머지는 그대로 둔다.*

2. `BswMConfig` 의 컨테이너를 표의 순서대로 추가한다. 규칙은 식과 액션 목록을 가리킨다. 액션 목록 `AlRteStart` 는 표에서 규칙보다 뒤에 있으므로 경로를 입력한다.

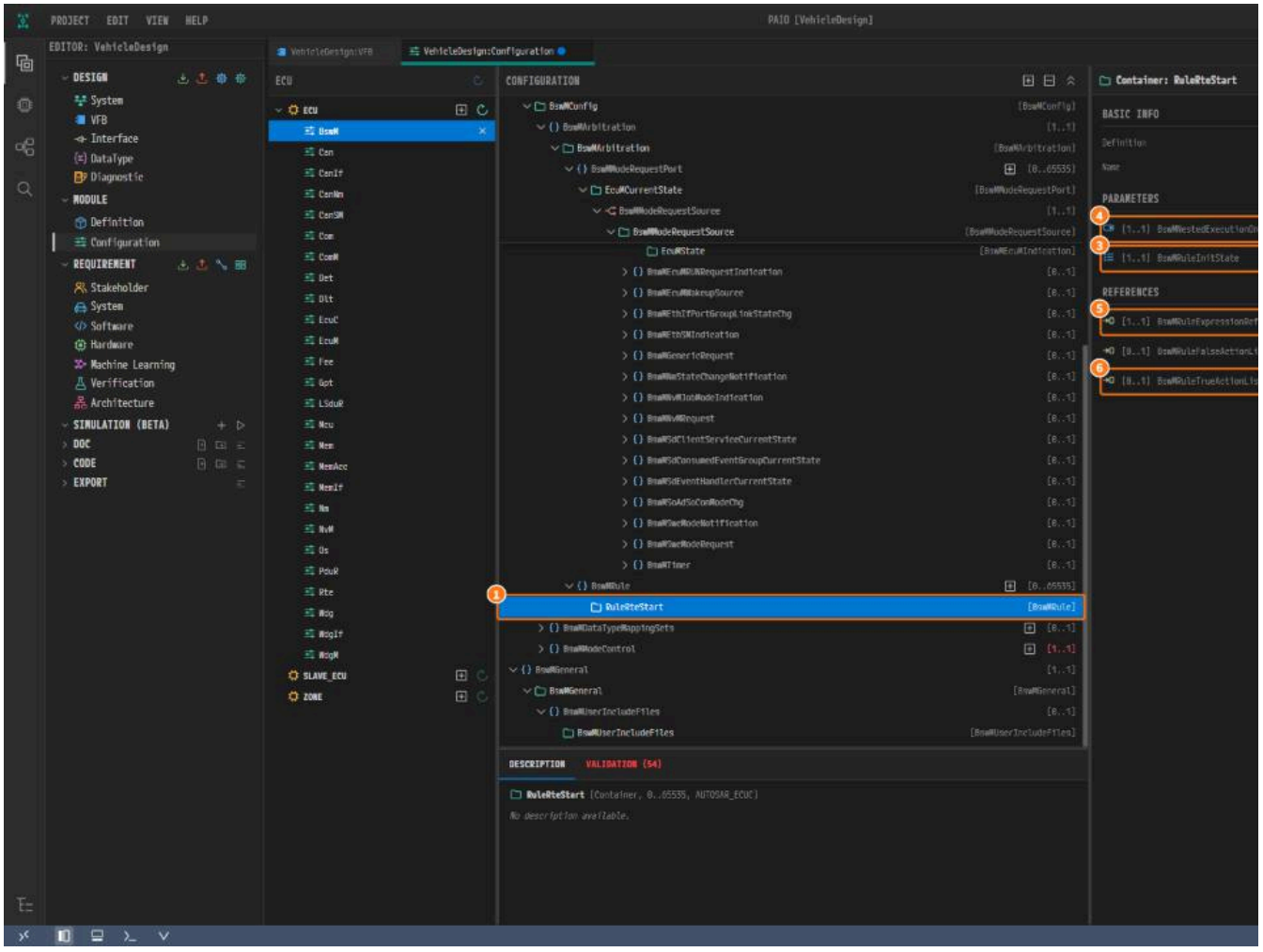


*그림 6-10. ① `RuleRteStart`, ② **Name**, ③ **BswMRuleInitState**, ④ **BswMNestedExecutionOnly**, ⑤ **BswMRuleExpressionRef**, ⑥ **BswMRuleTrueActionList**.*

3. `Ctrl+S` 로 저장한다.

Container	+ 를 누를 행	
`BswMGeneral`	`BswMGeneral`	**BswMDevErrorDetect TRUE**, **BswMVersionInfoApi TRU** TRUE, **BswMGenericRequestEnabled TRUE**, **BswMCan** **BswMNmEnabled TRUE**
`BswMGeneral/BswMUserIncludeFiles`	`BswMUserIncludeFiles`	**BswMUserIncludeFile** `EcuM.h`
`BswMConfig`	`BswMConfig`	없음
`BswMConfig/BswMArbitration`	`BswMArbitration`	없음
`BswMConfig/BswMArbitration/EcuMCurrentState`	`BswMModeRequestPort`	**BswMRequestProcessing** `BSWM_IMMEDIATE`
`BswMConfig/BswMArbitration/EcuMCurrentState/BswMModeRequestSource`	`BswMModeRequestSource`	없음
`BswMConfig/BswMArbitration/EcuMCurrentState/BswMModeRequestSource/EcuMState`	`BswMEcuMIndication`	없음
`BswMConfig/BswMArbitration/CondEcuMStateRun`	`BswMModeCondition`	**BswMConditionType** `BSWM_EQUALS`, **BswMConditionI** `BswM/BswMConfig/BswMArbitration/EcuMCurrentS`
`BswMConfig/BswMArbitration/CondEcuMStateRun/BswMConditionValue`	`BswMConditionValue`	없음
`BswMConfig/BswMArbitration/CondEcuMStateRun/BswMConditionValue/Value`	`BswMBswMode`	**BswMBswRequestedMode** `ECUM_STATE_RUN`
`BswMConfig/BswMArbitration/ExprEcuMRun`	`BswMLogicalExpression`	**BswMArgumentRef** `BswM/BswMConfig/BswMArbitra`
`BswMConfig/BswMArbitration/RuleRteStart`	`BswMRule`	**BswMRuleInitState** `BSWM_UNDEFINED`, **BswMRuleExp** `BswM/BswMConfig/BswMArbitration/ExprEcuMRun`, `BswM/BswMConfig/BswMModeControl/AlRteStart`
`BswMConfig/BswMModeControl`	`BswMModeControl`	없음
`BswMConfig/BswMModeControl/RteStart`	`BswMAction`	없음
`BswMConfig/BswMModeControl/RteStart/BswMAvailableActions`	`BswMAvailableActions`	없음
`BswMConfig/BswMModeControl/RteStart/BswMAvailableActions/Kind`	`BswMRteStart`	없음
`BswMConfig/BswMModeControl/AlRteStart`	`BswMActionList`	**BswMActionListExecution** `BSWM_TRIGGER`
`BswMConfig/BswMModeControl/AlRteStart/Item0`	`BswMActionListItem`	**BswMActionListItemIndex** `0`, **BswMAbortOnFail TRUI** `BswM/BswMConfig/BswMModeControl/RteStart`

Mcu 와 Det 설정하기

1. 목록의 `Mcu` 를 누르고 표의 컨테이너를 추가한다. `CoreClock` 은 9장에서 CAN 드라이버가 가리키는 클록이다.

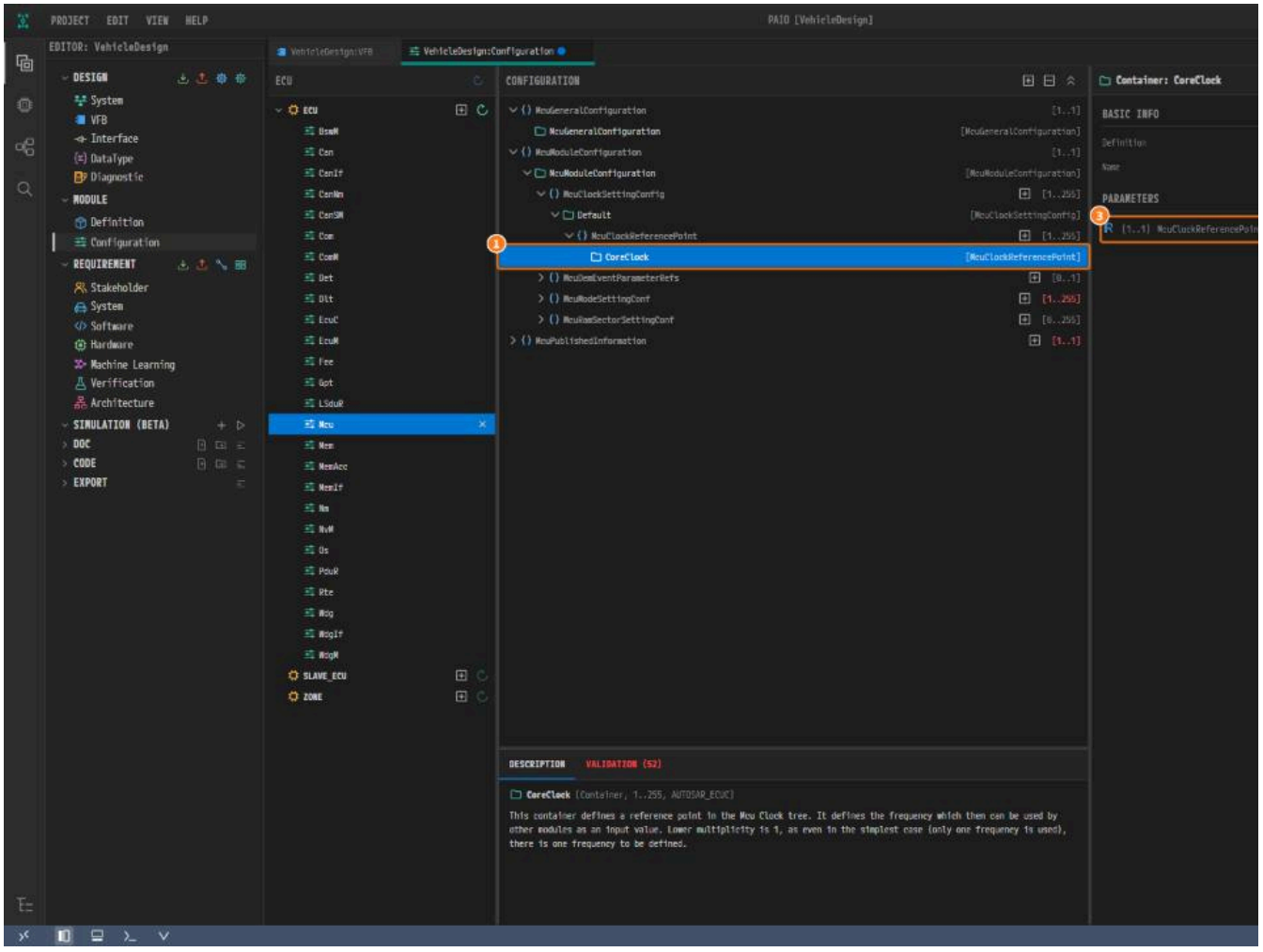


*그림 6-11. ① `CoreClock`, ② **Name**, ③ **McuClockReferencePointFrequency**.*

2. 목록의 `Det` 를 누르고 `DetGeneral` 을 추가한다.

3. `Ctrl+S` 로 저장한다.

Container	+ 를 누를 행	값
`McuGeneralConfiguration`	`McuGeneralConfiguration`	McuDevErrorDetect TRUE, McuGetRamStateApi TRUE, McuNoPll FALSE, McuPerformResetApi
`McuModuleConfiguration`	`McuModuleConfiguration`	McuClockSrcFailureNotification `DISABLED`, McuNumberOfMcuModes `2`, McuRamSectors `1`
`McuModuleConfiguration/Default`	`McuClockSettingConfig`	McuClockSettingId `0`
`McuModuleConfiguration/Default/CoreClock`	`McuClockReferencePoint`	McuClockReferencePointFrequency `100000000`
`McuModuleConfiguration/Normal`	`McuModeSettingConf`	McuMode `0`, McuLowPowerMode FALSE
`McuModuleConfiguration/Sleep`	`McuModeSettingConf`	McuMode `1`, McuLowPowerMode TRUE
`McuModuleConfiguration/Sector0`	`McuRamSectorSettingConf`	McuRamDefaultValue `0`, McuRamSectionBaseAddress `0`, McuRamSectionSize `256`
`McuPublishedInformation`	`McuPublishedInformation`	없음
`McuPublishedInformation/PowerOn`	`McuResetReasonConf`	McuResetReason `0`
`McuPublishedInformation/Software`	`McuResetReasonConf`	McuResetReason `2`

Container	+ 를 누를 행	값
`DetGeneral`	`DetGeneral`	**DetVersionInfoApi TRUE**

Rte 값 파일 가져오기

`Rte` 의 참조 여섯 개는 아직 속성 패널에서 넣을 수 없다. 목록에 대상이 나오지 않고, 경로를 입력해도 값이 남지 않는다.

컨테이너	참조	대상
`BswMInstance/RteEventToTaskMapping_0`	**RteEventRef**	BswM 폴링 이벤트 `BswMPollTimi`
`RteBswModuleInstance_0`, `_3`, `_4`, `_5`, `_6`(`BswM`, `CanIf`, `CanSM`, `ComM`, `Com`)의 `RteBswExclusiveAreaImpl_0`	**RteBswExclusiveAreaRef**	그 모듈의 BSW 배타 영역

Rte 값 파일은 이 컨테이너를 참조와 함께 가져온다. `Rte` 를 임포트하면 설정 전체가 바뀌므로, 파일은 PRTE Platform Import 가 만든 SW-C 타입·인스턴스 쌍도 담는다. 그렇지 않

1. **EDITOR** 뷰의 **DESIGN** 머리글에 있는 **Import** 아이콘을 누른다.

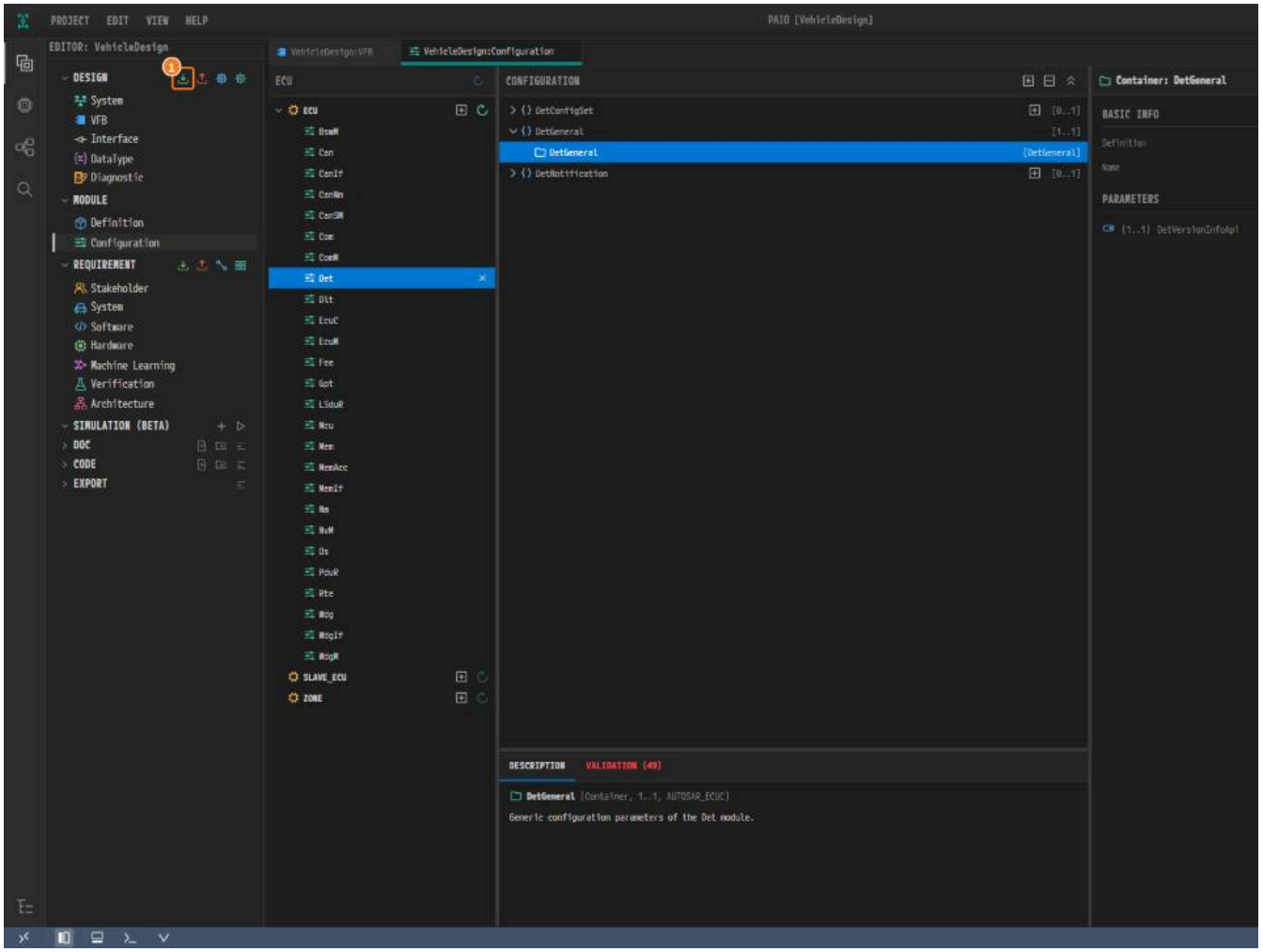


*그림 6-12. ① **Import**.*

2. 파일 대화상자에서 받아 둔 파일 `VehicleDesign_Rte.arxml` 을 고른다. 파일의 유일한 모듈 `Rte` 로 **Conflict Resolution** 창이 열린다. 오른쪽 아래 버튼은 `Resolve 0` **Use Imported** 를 누른다. 버튼이 `Resolve 1/1` 로 바뀌면 누른다.

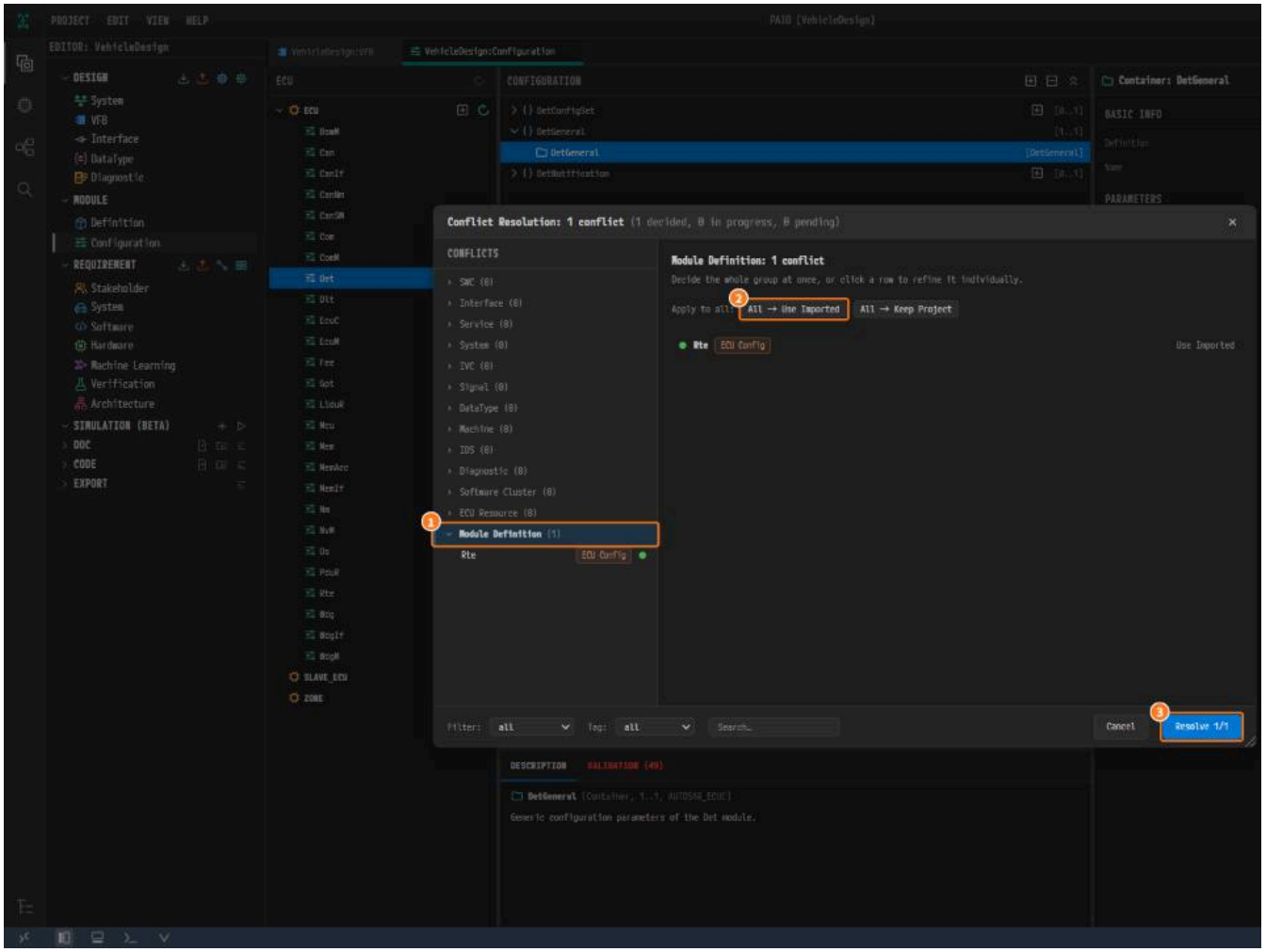


*그림 6-13. ① **Module Definition**, ② **All → Use Imported**, ③ `Resolve 1/1`.*

오른쪽 아래에 `Import complete` 메시지와 그 아래 `1 replaced` 가 보인다. `Rte` 컨테이너는 8장에서 채운다.

왜 이렇게 설계하는가

- **AUTO CONFIGURATION 을 쓰지 않는다.** 모든 BSW 값을 독자가 넣으므로 하나하나가 에디터에 보인다. 빈 프로젝트에서 이렇게 설정하면 검증은 문제를 보고하지 않고 PRTE
- **Platform Import 는 설정이 아니다.** 모듈 정의, 빈 설정, BSW 서비스 컴포넌트를 만들 뿐 BSW 값은 하나도 정하지 않는다.
- **PRTE 가 요구하는 다섯 모듈.** PRTE 가 코드를 생성하려면 `Os`, `Rte`, `EcuM`, `BswM`, `Mcu` 의 설정이 있어야 한다. 하나라도 비어 있으면 생성이 `MANDATORY_MODULE_MISSI`
- **Rte 값 파일은 임시다.** 속성 패널이 아직 받지 못하는 참조 여섯 개와, 임포트가 지워 버릴 SW-C 쌍만 담는다. 패널이 이 대상을 보여 주게 되면 이 참조도 다른 값처럼 8장에서 입
- **시동 시의 네트워크 관리. BswMNmEnabled** 와 항목 `CanNmInit`, `NmInit` 은 실행 시 네트워크 관리의 시동에 쓰이며, PRTE 가 코드를 생성하는 데는 필요하지 않다. `CanN` 구하는 BSW 배타 영역이 없다.

확인하기

- [ ] 프레임 안에 컴포지션 `PRTE_ECU` 가 있다.
- [ ] `EcuM` 에 `EcuMConfiguration`, `EcuMFlexGeneral`, `EcuMGeneral` 이 있고, `EcuMDriverInitListOne` 에 `McuInit` 부터 `CanSMInit` 까지 항목 열한 개가 있다.
- [ ] `BswM` 에 `RuleRteStart` 와 `AlRteStart` 가 있다.
- [ ] `Import complete` 메시지와 `1 replaced` 가 보였다.
- [ ] 탭 `VehicleDesign:Configuration` 에 수정 표시가 없다.

다음 단계

Os 설정에서 `Os` 의 태스크, 알람, 카운터를 손으로 만든다.

Os 설정

이 장에서 할 일

`Os` 는 무엇을 언제 실행할지 정한다. 이 장에서는 ECU Configuration 에디터에서 그 컨테이너를 손으로 만든다. 운영체제 전체의 설정, `EcuM` 이 이름으로 가리키는 애플리케이션 !
성화하는 알람 세 개다.

시작하기 전에

- PRTE Setup을 마친 뒤 시작한다.
- **MODULE** 아래의 **Configuration** 을 눌러 ECU Configuration 에디터를 연다.
- **CONFIGURATION** 트리에서 행 오른쪽 끝의 **+** 는 그 정의의 컨테이너를 기본값과 함께 추가하고 선택한다. 표에 있는 값만 바꾼다. 숫자는 입력하고 `Tab` 을 누르고, 참조는 목록
- **Container** 열의 굵은 이름은 **+** 가 주는 이름이 아니다. 패널의 **Name** 에 입력하고 `Tab` 을 누른다.

Container	+ 를 누를 행	값
`OsOS`	`OsOS`	**OsStatus** `EXTENDED`, **OsStackMonitoring** TRUE, **OsUseGetServiceId** TRUE, **OsUseParameterAcc
`OsOS/OsHooks`	`OsHooks`	**OsErrorHook**, **OsPostTaskHook**, **OsPreTaskHook**, **OsShutdownHook**, **OsStartupHook** 모두 **FALSE**
`OSDEFAULTAPPMODE`	`OsAppMode`	없음
`RES_AUTOSAR_ECUM`	`OsResource`	**OsResourceProperty** `STANDARD`
`OsCounter_0`	`OsCounter`	**OsCounterMaxAllowedValue** `1000000`, **OsCounterMinCycle** `1`, **OsCounterTicksPerBase** `1`, Os
`OsTask_0`	`OsTask`	**OsTaskActivation** `1`, **OsTaskPeriod** `0.01`, **OsTaskPriority** `2`, **OsTaskSchedule** `FULL`, **OsTaskSt**
`OsTask_EcuMStartup`	`OsTask`	**OsTaskActivation** `1`, **OsTaskPriority** `3`, **OsTaskSchedule** `FULL`, **OsTaskStackSize** `32768`
`OsTask_EcuMStartup/OsTaskAutostart`	`OsTaskAutostart`	**OsTaskAppModeRef** `Os/OSDEFAULTAPPMODE`
`OsTask_2`	`OsTask`	**OsTaskActivation** `1`, **OsTaskPeriod** `0.02`, **OsTaskPriority** `1`, **OsTaskSchedule** `FULL`, **OsTaskSt**
`OsTask_3`	`OsTask`	**OsTaskActivation** `1`, **OsTaskPeriod** `0.01`, **OsTaskPriority** `1`, **OsTaskSchedule** `FULL`, **OsTaskSt**
`OsAlarm_0`	`OsAlarm`	**OsAlarmCounterRef** `Os/OsCounter_0`
`OsAlarm_0/OsAlarmAction`	`OsAlarmAction`	없음
`OsAlarm_0/OsAlarmAction/OsAlarmActivateTask`	`OsAlarmActivateTask`	**OsAlarmActivateTaskRef** `Os/OsTask_0`
`OsAlarm_1`	`OsAlarm`	**OsAlarmCounterRef** `Os/OsCounter_0`
`OsAlarm_1/OsAlarmAction`	`OsAlarmAction`	없음
`OsAlarm_1/OsAlarmAction/OsAlarmActivateTask`	`OsAlarmActivateTask`	**OsAlarmActivateTaskRef** `Os/OsTask_2`
`OsAlarm_2`	`OsAlarm`	**OsAlarmCounterRef** `Os/OsCounter_0`
`OsAlarm_2/OsAlarmAction`	`OsAlarmAction`	없음
`OsAlarm_2/OsAlarmAction/OsAlarmActivateTask`	`OsAlarmActivateTask`	**OsAlarmActivateTaskRef** `Os/OsTask_3`

운영체제 설정하기

1. 목록의 `Os` 를 누른다. **CONFIGURATION** 트리에 `Os` 의 컨테이너 정의가 모두 빈 채로 나열된다. `OsOS` 행의 **+** 를 누른다. 에디터가 `OsOS` 를 추가하고 선택한다.

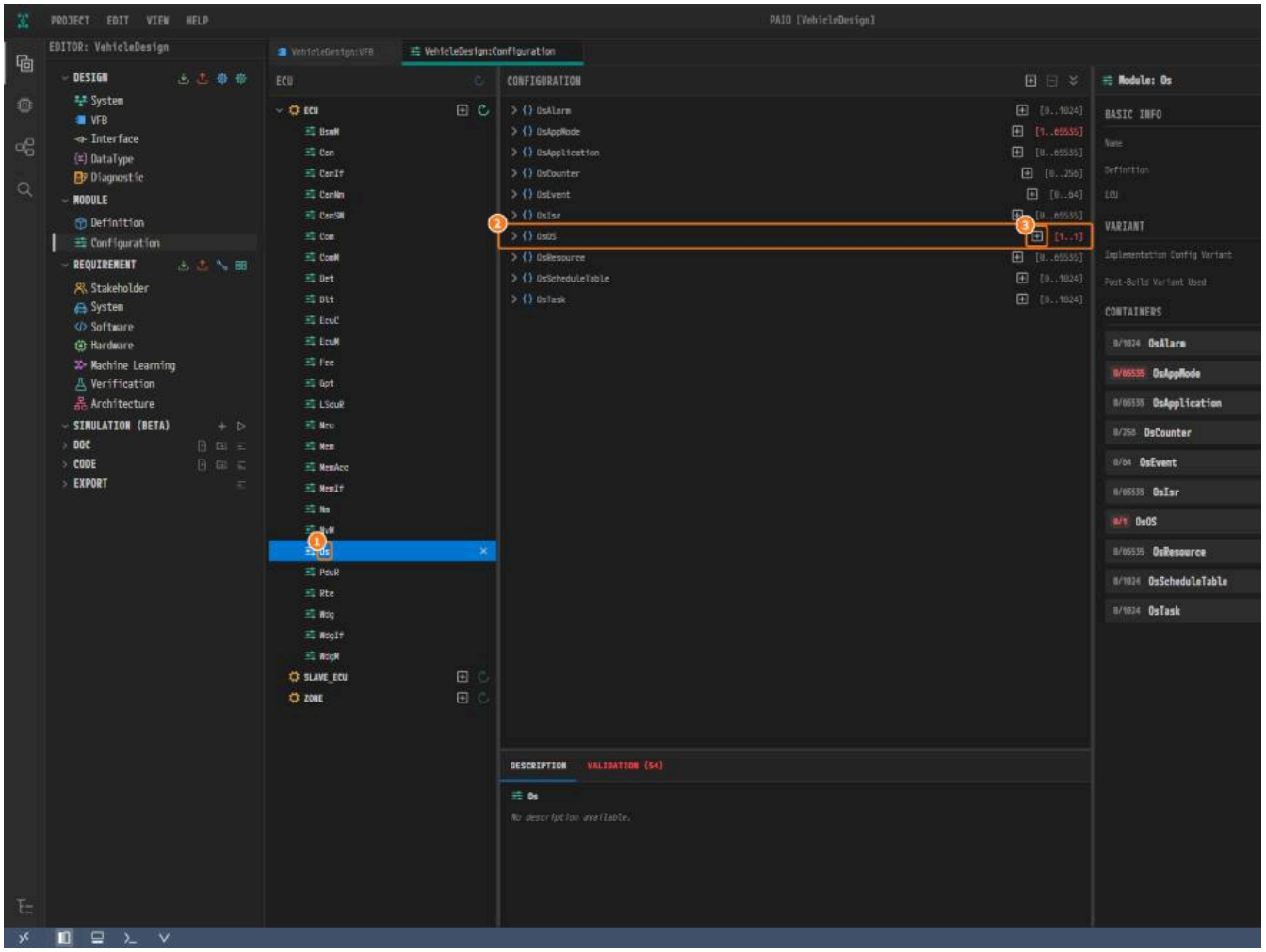


그림 7-1. ① `Os`, ② `OsOS` 행, ③ +.

2. 오른쪽 패널에서 **OsStatus** 를 `EXTENDED` 로 고른다. **OsStackMonitoring**, **OsUseGetServiceId**, **OsUseParameterAccess** 의 **TRUE** 를 누른다.

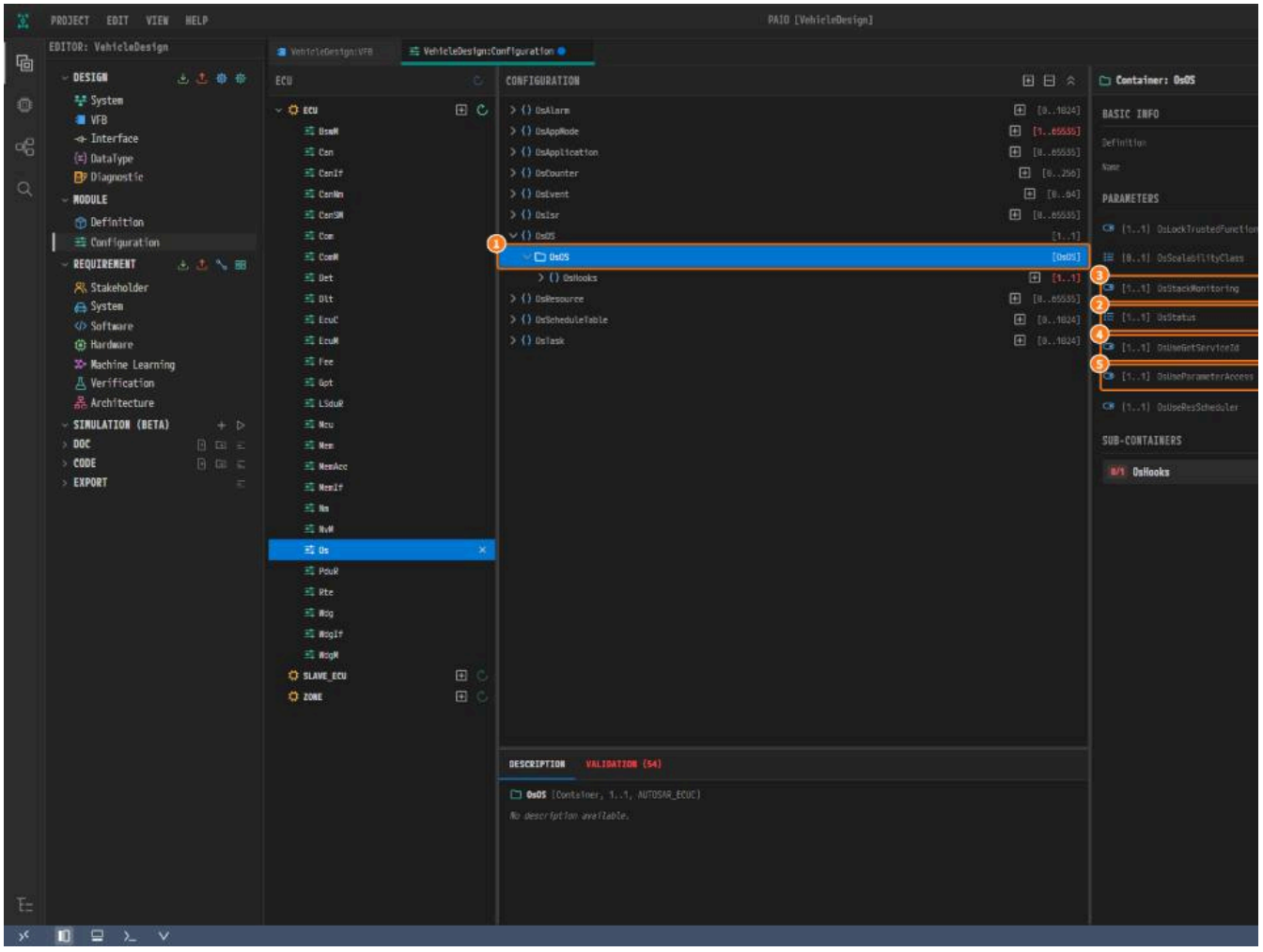


*그림 7-2. ① `OsOS`, ② **OsStatus**, ③ **OsStackMonitoring**, ④ **OsUseGetServiceId**, ⑤ **OsUseParameterAccess**.*

3. `OsOS` 아래의 `OsHooks` 행에서 **+** 를 누르고 다섯 훅의 **FALSE** 를 누른다.
4. `OsAppMode` 행의 **+** 를 누른다. **Name** 의 이름을 `OSDEFAULTAPPMODE` 로 바꾸고 `Tab` 을 누른다.
5. `OsResource` 행의 **+** 를 누른다. 같은 방법으로 이름을 `RES_AUTOSAR_ECUM` 으로 바꾸고 **OsResourceProperty** 를 `STANDARD` 로 고른다.

   두 이름은 정확해야 한다. PRTE Setup에서 넣은 `EcuM` 설정이 이 이름을 가리킨다.
6. `OsCounter` 행의 **+** 를 누르고 표의 `OsCounter_0` 값을 입력한다.

태스크 만들기

1. `OsTask` 행의 **+** 를 누른다. 에디터가 `OsTask_0` 을 추가하고 선택한다.

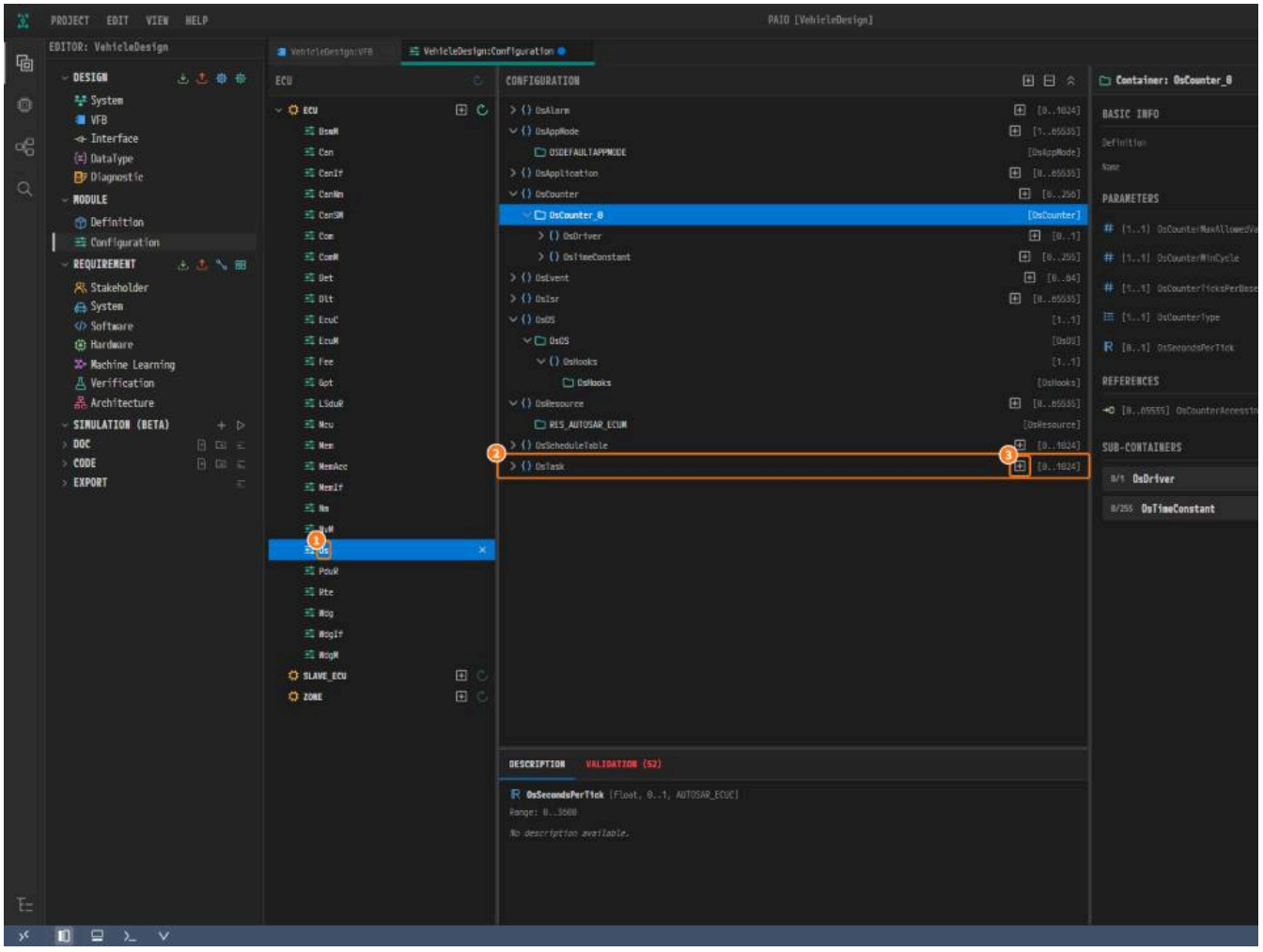


그림 7-3. ① `Os`, ② `OsTask` 행, ③ +.

2. 표의 `OsTask_0` 값을 입력한다.

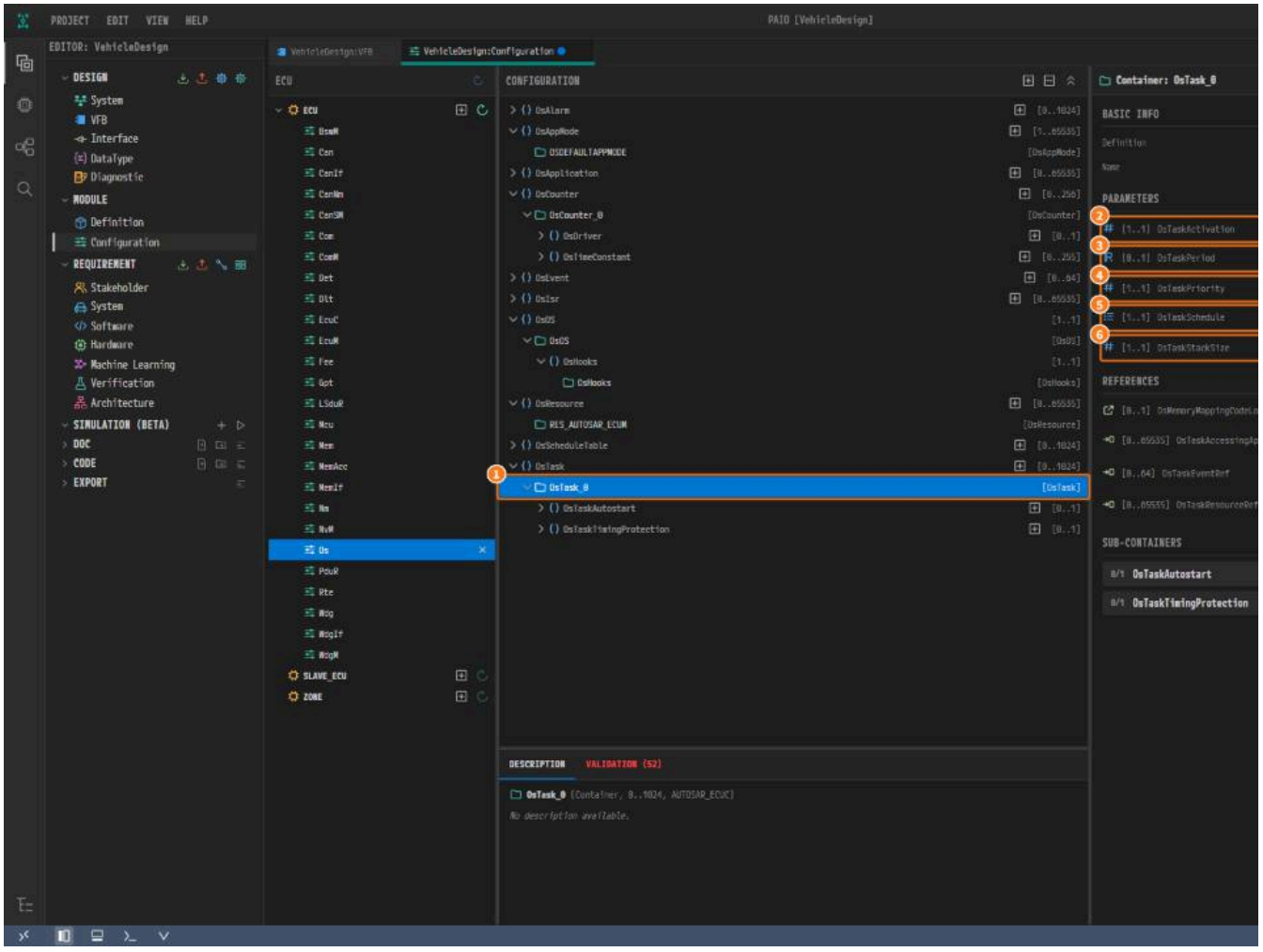


*그림 7-4. ① `OsTask_0`, ② **OsTaskActivation**, ③ **OsTaskPeriod**, ④ **OsTaskPriority**, ⑤ **OsTaskSchedule**, ⑥ **OsTaskStackSize**.*

3. `OsTask` 행의 **+** 를 다시 누른다. 에디터가 `OsTask_1` 을 추가한다. **Name** 의 이름을 `OsTask_EcuMStartup` 으로 바꾸고 `Tab` 을 누른다. 표의 값을 입력하고 **OsTaskPeri**

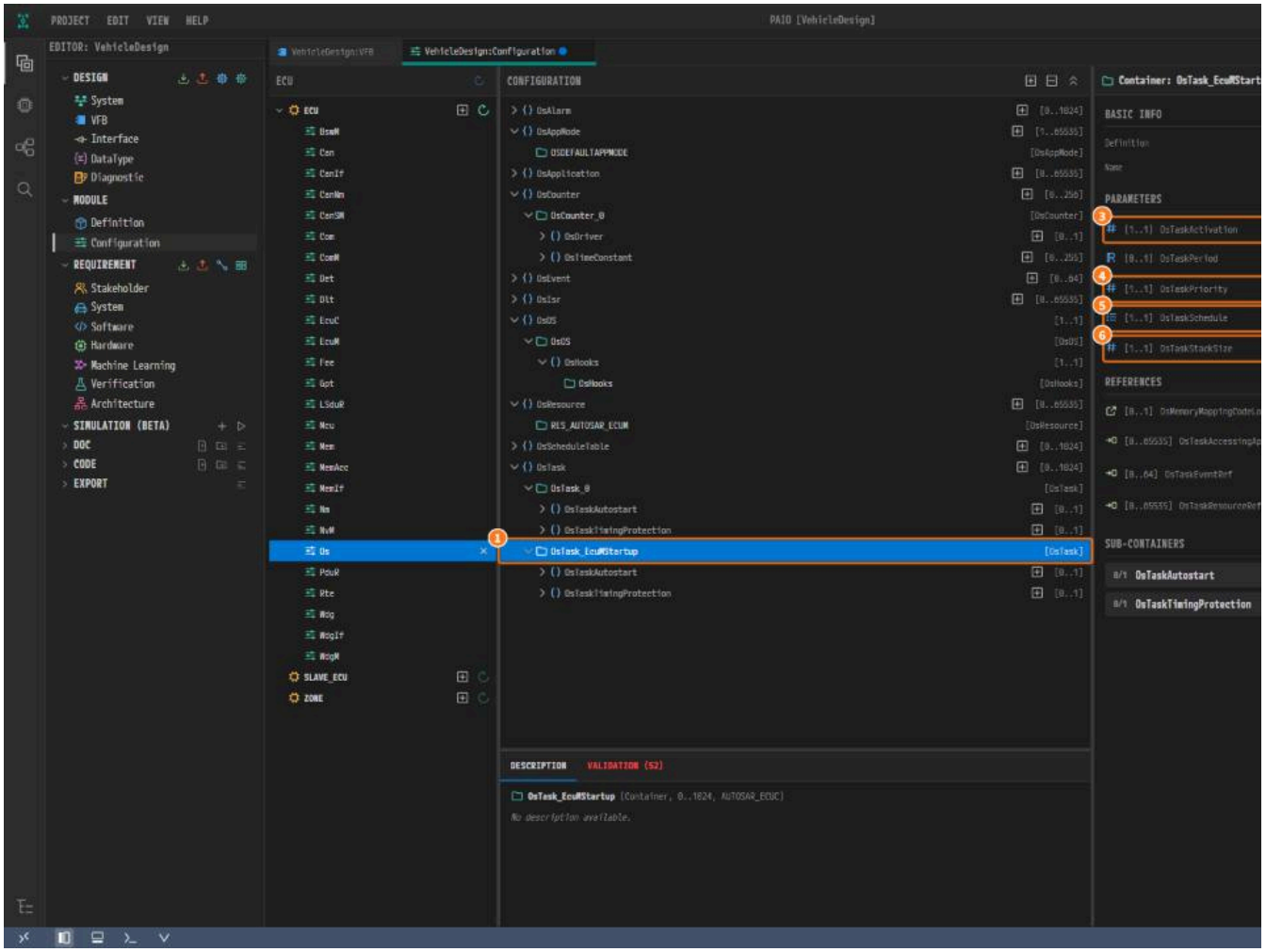


*그림 7-5. ① `OsTask_EcuMStartup`, ② **Name**, ③ **OsTaskActivation**, ④ **OsTaskPriority**, ⑤ **OsTaskSchedule**, ⑥ **OsTaskStackSize**.*

4. `OsTask_EcuMStartup` 아래의 `OsTaskAutostart` 행에서 **+** 를 누른다. 새 `OsTaskAutostart` 의 **OsTaskAppModeRef** 를 `Os/OSDEFAULTAPPMODE` 로 고른다.

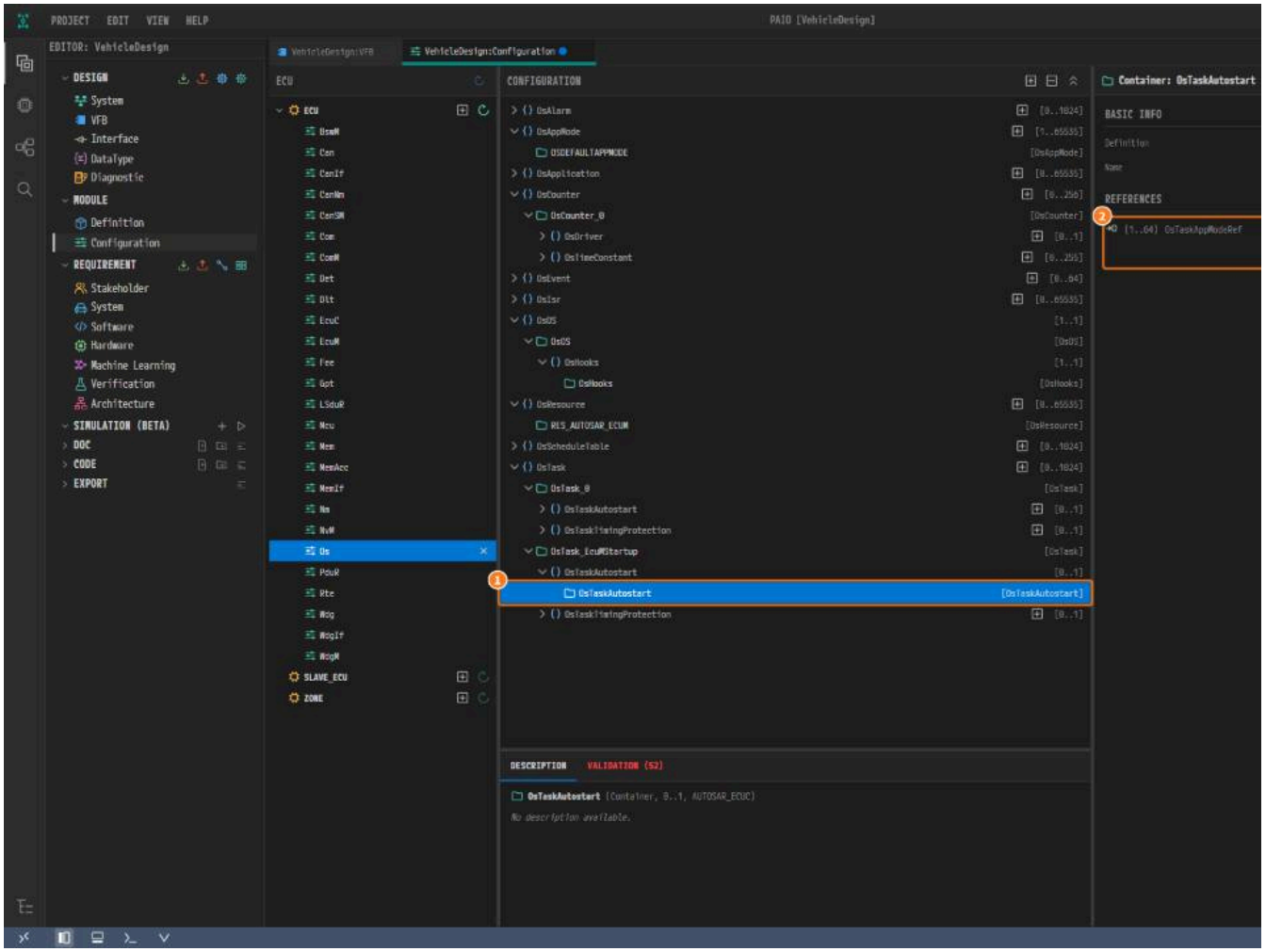


*그림 7-6. ① `OsTaskAutostart`, ② **OsTaskAppModeRef**.*

5. `OsTask` 행의 **+** 를 다시 누른다. 번호 1이 비어 있으므로 에디터는 새 태스크를 다시 `OsTask_1` 로 이름 짓는다. 이름을 `OsTask_2` 로 바꾸고 표의 값을 입력한다.

6. 같은 방법으로 `OsTask_3` 을 만든다. **+** 가 주는 `OsTask_1` 의 이름을 `OsTask_3` 으로 바꾼다.

알람 만들기

1. `OsAlarm` 행의 **+** 를 누른다. 새 `OsAlarm_0` 의 **OsAlarmCounterRef** 를 `Os/OsCounter_0` 으로 고른다.

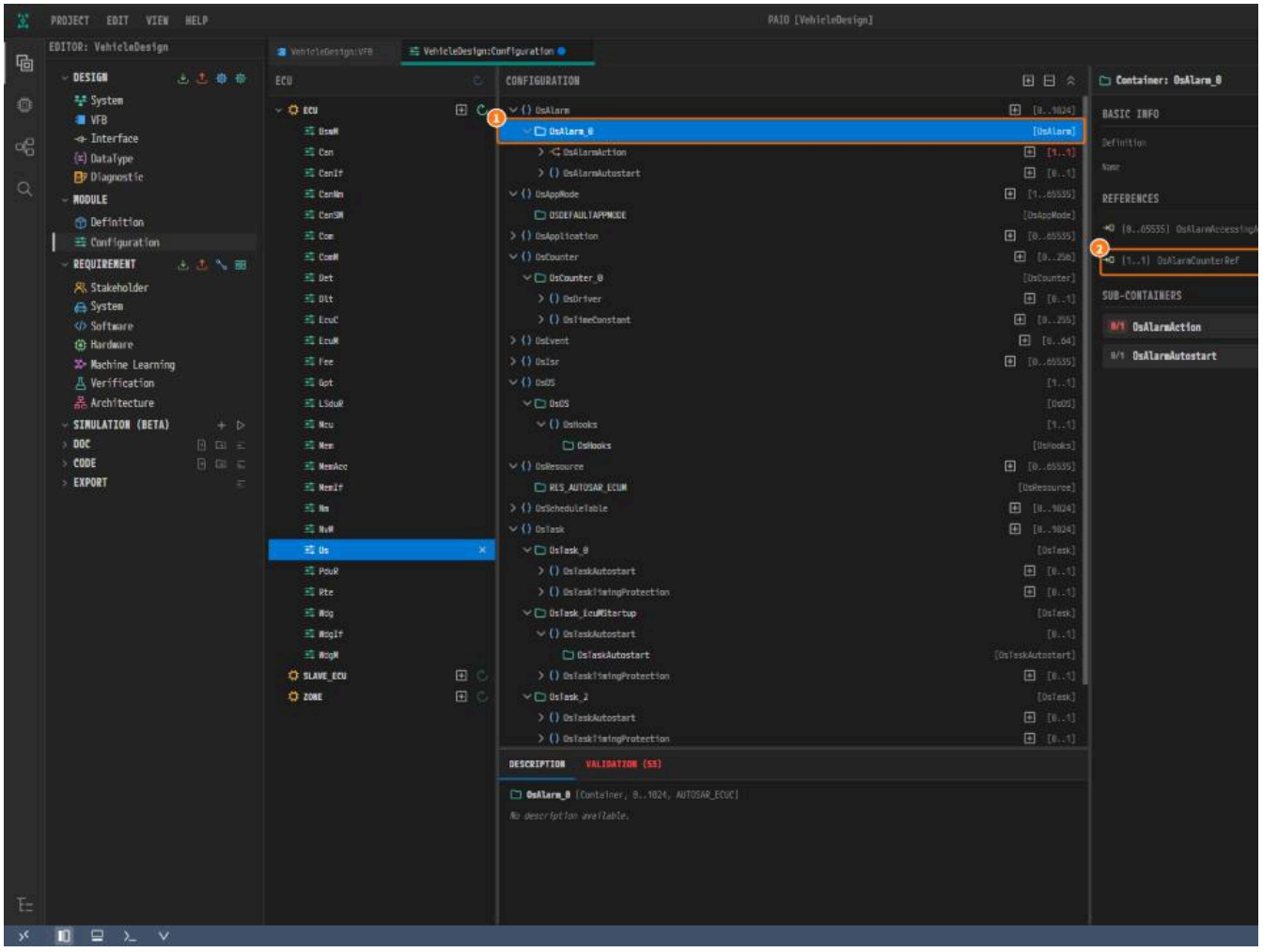


*그림 7-7. ① `OsAlarm_0`, ② **OsAlarmCounterRef.***

2. `OsAlarm_0` 아래의 `OsAlarmAction` 행에서 **+** 를 누른다. 새 `OsAlarmAction` 아래의 `OsAlarmActivateTask` 행에서 **+** 를 누른다.

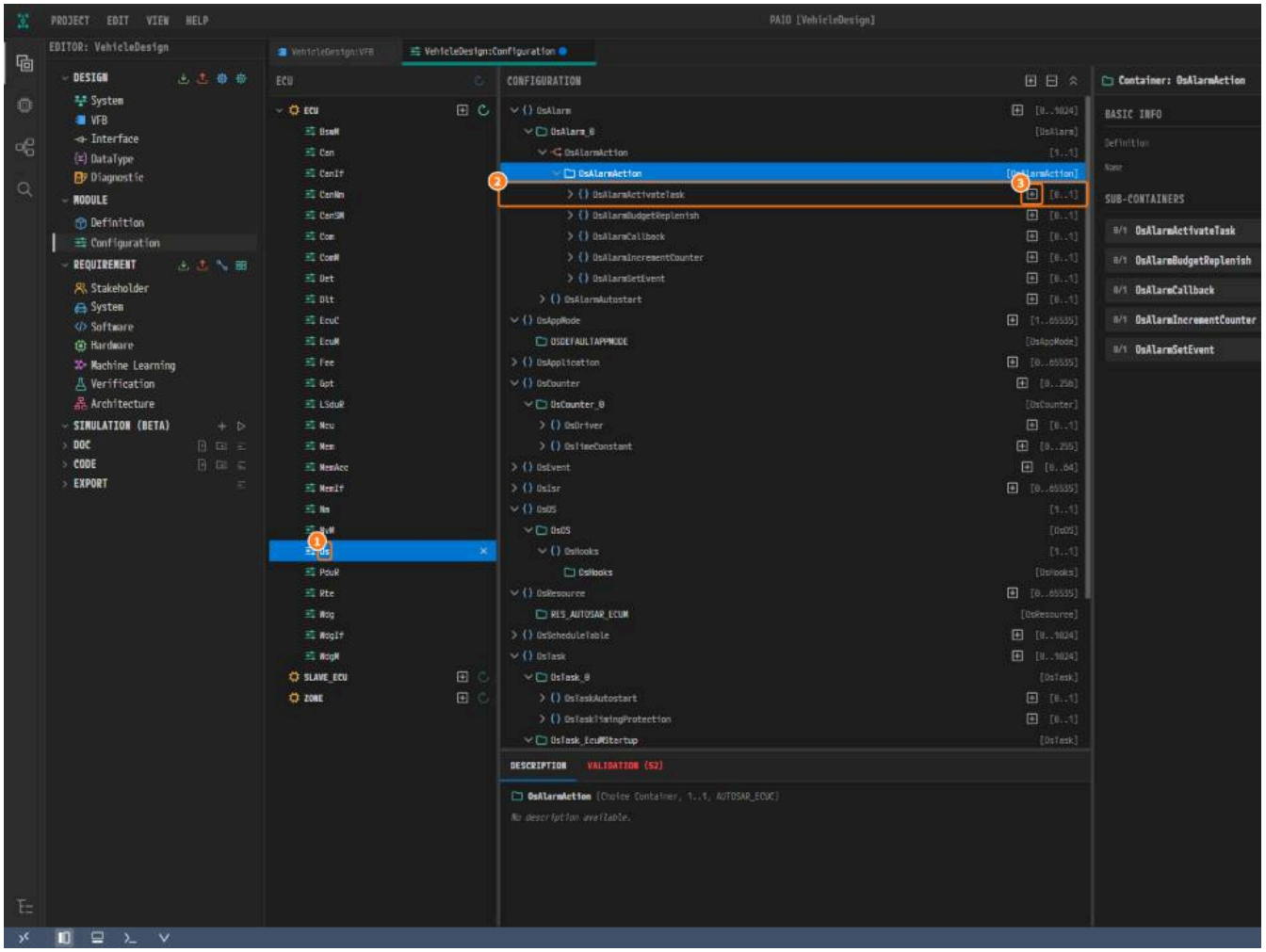


*그림 7-8. ① `Os`, ② `OsAlarm_0/OsAlarmAction` 아래의 `OsAlarmActivateTask` 행, ③ **+**.*

3. 새 `OsAlarmActivateTask` 의 **OsAlarmActivateTaskRef** 를 `Os/OsTask_0` 으로 고른다.

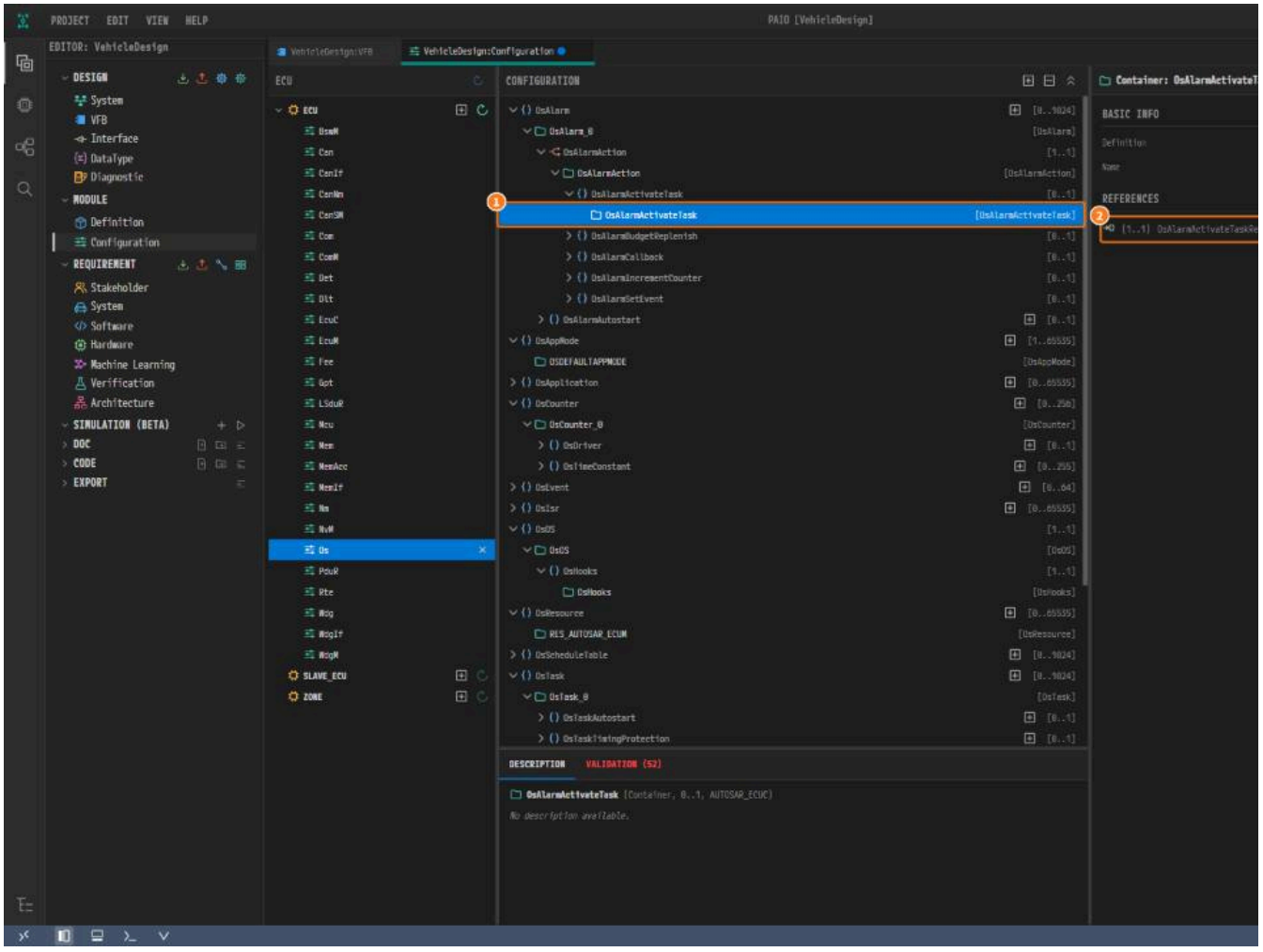


*그림 7-9. ① `OsAlarmActivateTask`, ② **OsAlarmActivateTaskRef**.*

4. 표대로 `OsAlarm_1` 과 `OsAlarm_2` 를 같은 방법으로 만든다. `OsAlarm_1` 은 `OsTask_2` 를, `OsAlarm_2` 는 `OsTask_3` 을 활성화한다.

5. `Ctrl+S` 로 저장한다.

왜 이렇게 설계하는가

- **고정된 이름.** PRTE Setup 의 `EcuM` 이 `OSDEFAULTAPPMODE` 와 `RES_AUTOSAR_ECUM` 을 이름으로 가리킨다. PAIO 는 이름이 `OsTask_EcuMStartup` 인 태스크에 대해서 다.
- **타이밍마다 태스크 하나.** `OsTask_0` 은 10 ms 마다 BSW 메인 함수를, `OsTask_2` 는 20 ms 마다 센서 러너블을, `OsTask_3` 은 액추에이터 러너블과 BswM 폴링을 실행한다.
- **시동 태스크.** `OsTask_EcuMStartup` 에는 주기가 없다. **OsTaskAutostart** 가 `OSDEFAULTAPPMODE` 에서 한 번 시작하고, 우선순위 `3` 이 가장 높다.
- **카운터 하나.** `OsCounter_0` 은 0.01 s 마다 틱을 세고 알람 세 개를 모두 구동한다.
- **EXTENDED.** 확장 상태에서는 OS 서비스가 오류를 보고하므로 개발 중에 도움이 된다.

확인하기

- [ ] `Os` 의 **CONFIGURATION** 트리에 `OsOS`, `OSDEFAULTAPPMODE`, `RES_AUTOSAR_ECUM`, `OsCounter_0` 이 있다.
- [ ] 태스크 네 개 `OsTask_0`, `OsTask_EcuMStartup`, `OsTask_2`, `OsTask_3` 이 있다.
- [ ] `OsAlarm_0`, `OsAlarm_1`, `OsAlarm_2` 가 각각 `OsTask_0`, `OsTask_2`, `OsTask_3` 을 활성화한다.
- [ ] 탭 `VehicleDesign:Configuration` 에 수정 표시가 없다.

다음 단계

Rte 설정에서 러너블과 BSW 메인 함수를 이 태스크에 매핑한다.

Rte 설정

이 장에서 할 일

이 장에서는 모듈 `Rte` 를 손으로 설정한다. Rte 는 SW-C 의 러너블 이벤트와 `CanNm` 의 메인 함수를 포함한 BSW 의 메인 함수를 각각 Os 태스크와 그 태스크를 깨우는 알람에 매핑

시작하기 전에

- Os 설정을 마친 뒤 시작한다. Rte 참조는 거기서 만든 Os 태스크와 알람을 고른다.
- 탭 `VehicleDesign:Configuration` 을 열고 **MODULE** > **Configuration** > `Rte` 를 누른다.
- "+" 는 **CONFIGURATION** 트리의 행 오른쪽 끝에 있는 **+** 를 뜻한다. 컨테이너를 추가하고 선택한다. 여기 적힌 값만 바꾼다.
- 참조는 드롭다운에서 고른다. 단 `RteBswImplementationRef` 와 `RteBswEventRef` 는 드롭다운에 아무것도 없으므로 경로를 입력하고 `Tab` 을 누른다.
- 패널에는 긴 경로의 끝부분만 보일 수 있다.

일부 컨테이너는 PRTE Setup의 Rte 값 파일이 이미 만들었다. 그것은 다시 추가하지 말고 채워 넣기만 한다.

출처	컨테이너
Rte 값 파일	SW-C 인스턴스 (`EcuMInstance`, `BswMInstance`, …, `SwcSensorInstance`, `SwcActuatorInstance`)
Rte 값 파일	`BswMInstance/RteEventToTaskMapping_0`
Rte 값 파일	`RteBswModuleInstance_0`, `_3`, `_4`, `_5`, `_6` 과 각각의 `RteBswExclusiveAreaImpl_0`
"+" (26개)	`RteGeneration`, `RteBswGeneral`, `RteInitializationBehavior_0`, `RteEventToTaskMapping` 3개, `RteModeMachineInstanceConfig_0`, `RteOsInteraction`, `RteUs` `RteBswModuleInstance_1`, `_2`, `_7`, `RteBswEventToTaskMapping` 12개

Rte 값 파일이 가져온 것 확인하기

1. `Rte` 를 누른다. 트리에는 이미 SW-C 인스턴스와 BSW 모듈 인스턴스 다섯 개가 있다.

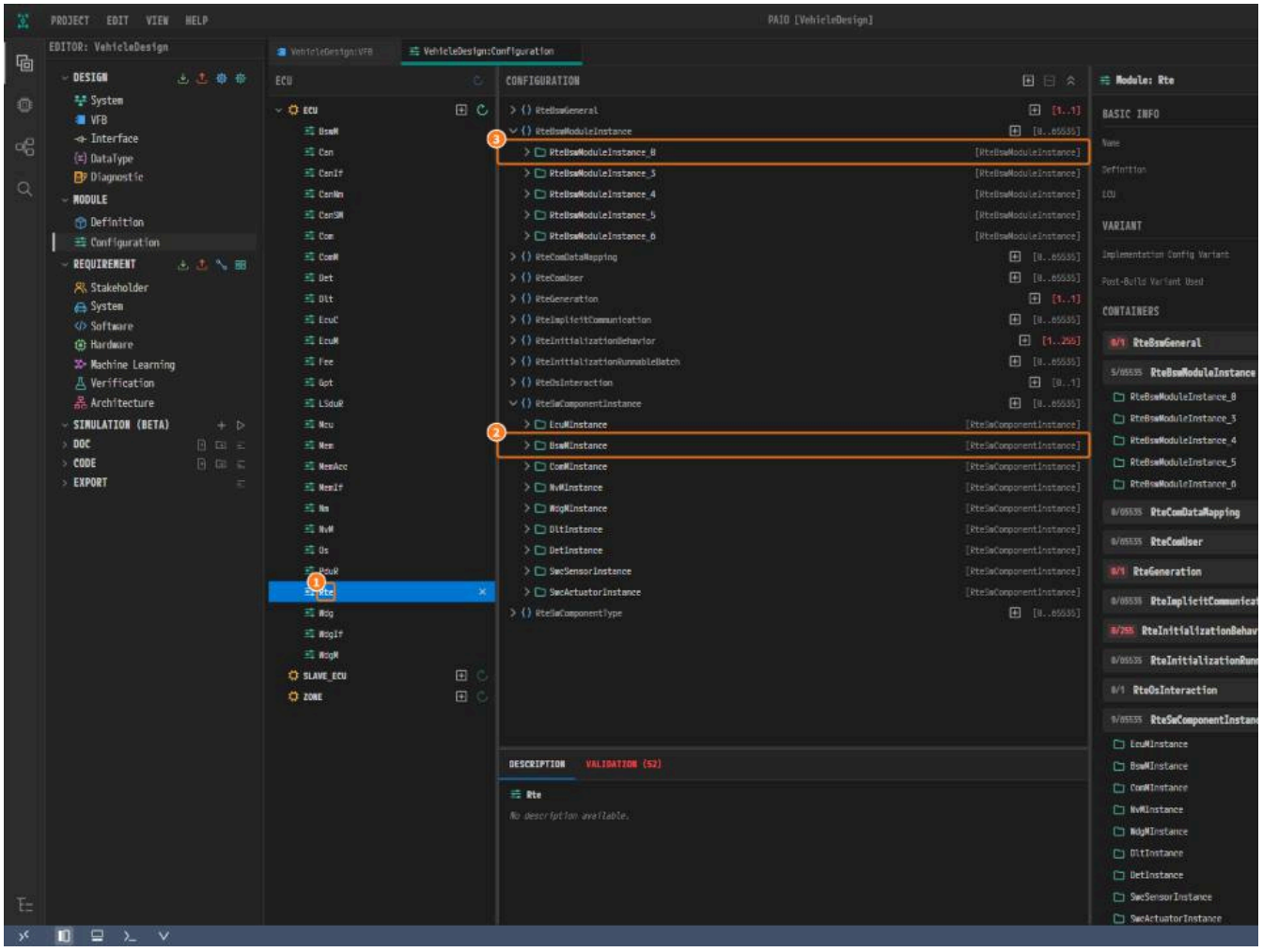


그림 8-1. ① `Rte`, ② `BswMInstance` (Rte 값 파일에서 옴), ③ `RteBswModuleInstance_0` (Rte 값 파일에서 옴).

생성과 초기화

1. `RteGeneration` 행에서 "+" 를 누른다.

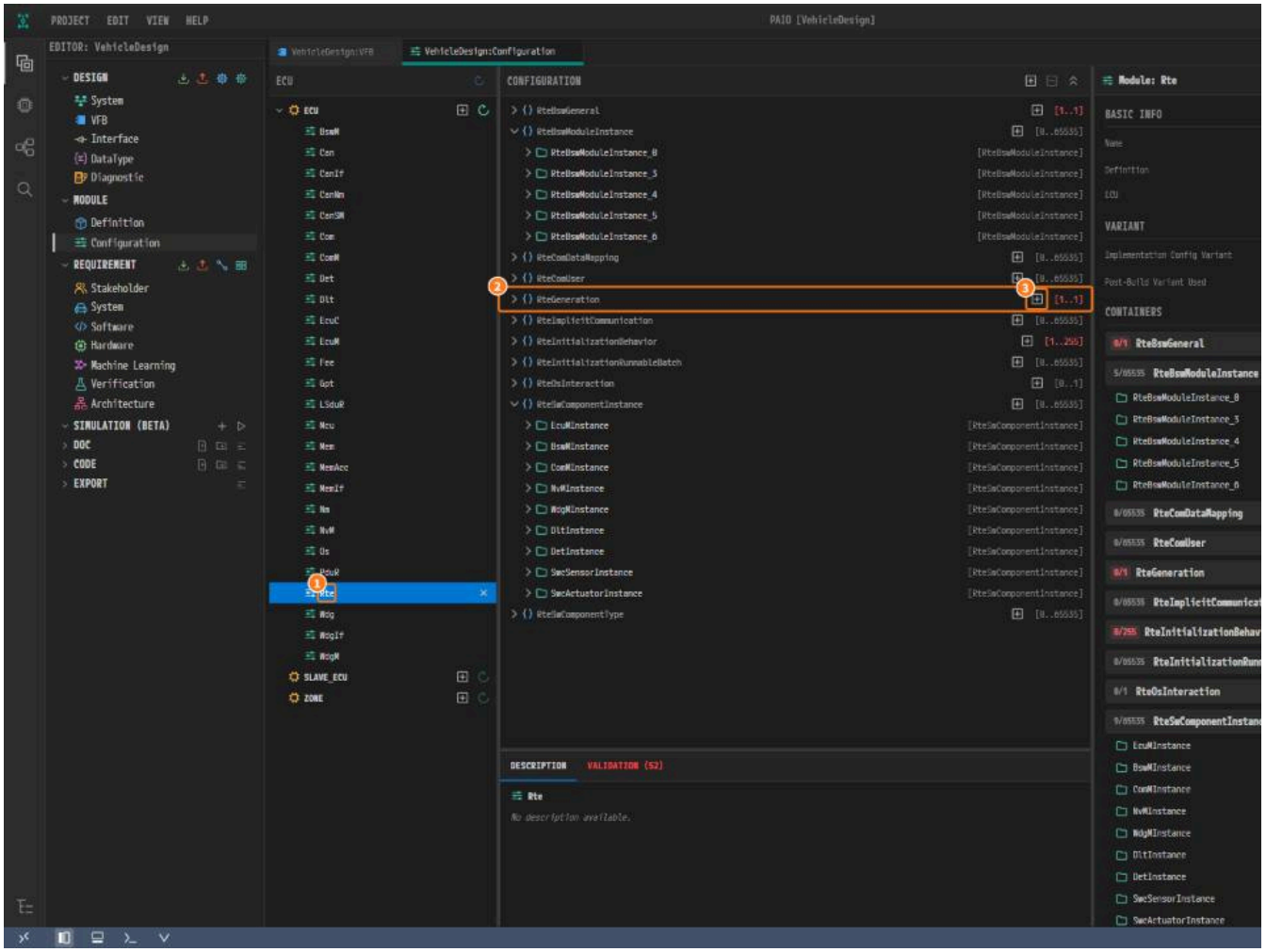


그림 8-2. ① `Rte`, ② `RteGeneration`, ③ +.

2. `RteCodeVendorId` 에 `79` 를 입력하고 `Tab` 을 누른다.

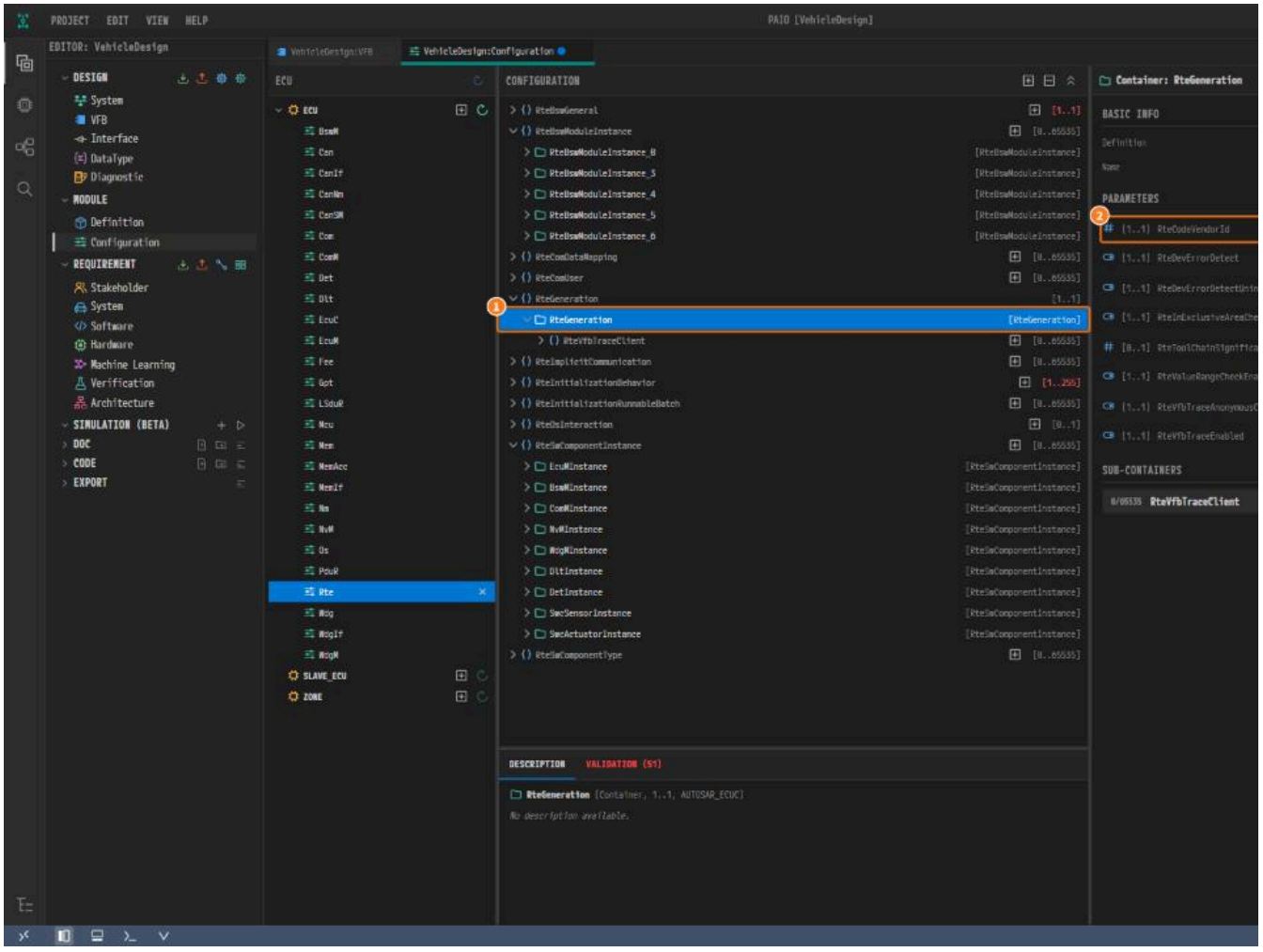


그림 8-3. ① `RteGeneration`, ② `RteCodeVendorId`.

3. `RteBswGeneral` 에서 "+" 를 누른다. 값은 넣지 않는다.
4. `RteInitializationBehavior` 에서 "+" 를 누른다. `RteInitializationStrategy` 를 `RTE_INITIALIZATION_STRATEGY_AT_DATA_DECLARATION` 으로 하고 `R` 력한다.

SW-C 이벤트 매핑

여기서의 함정은 `RteEventIsMappedToTask` 다. "+" 는 `FALSE` 를 쓰지만, 태스크를 지정하는 매핑은 모두 `TRUE` 여야 한다.

1. `SwcSensorInstance` 아래의 `RteEventToTaskMapping` 에서 "+" 를 누른다.

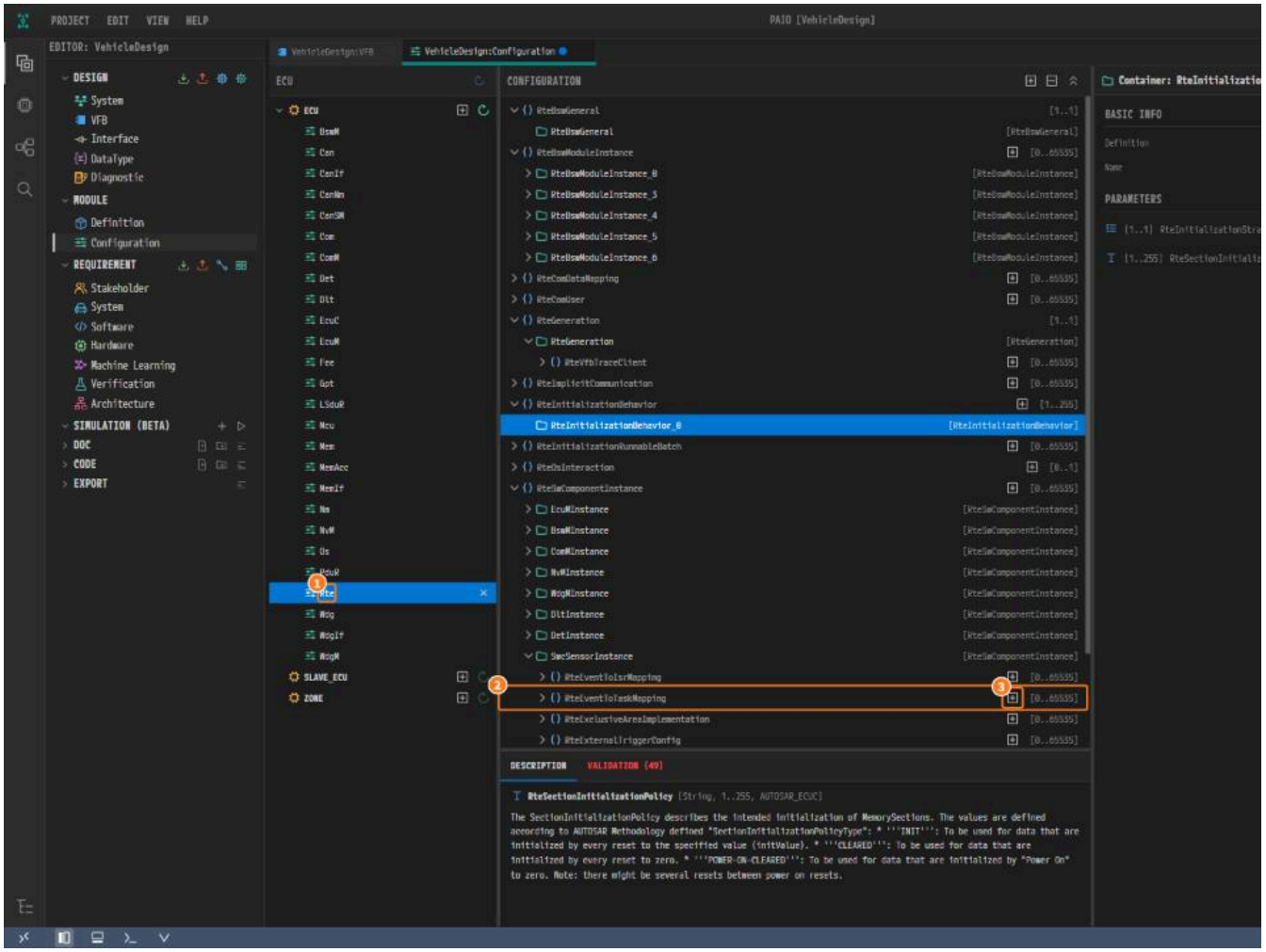


*그림 8-4. ① `Rte`, ② `SwcSensorInstance` 아래의 `RteEventToTaskMapping`, ③ **+**.*

2. `RteEventIsMappedToTask` 의 **TRUE** 를 누르고, 표를 보고 참조를 고른다.

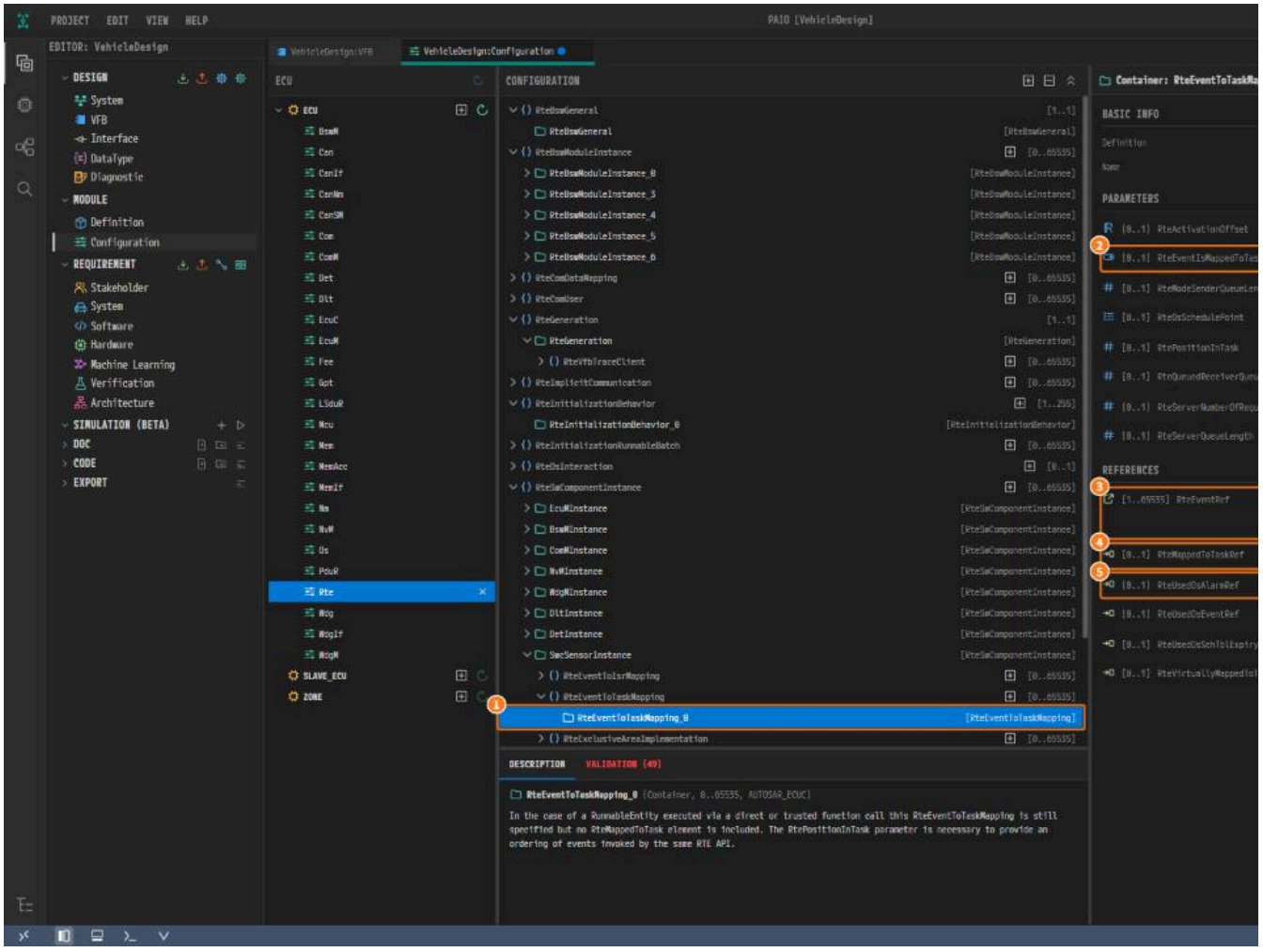


그림 8-5. ① `RteEventToTaskMapping_0`, ② `RteEventIsMappedToTask`, ③ `RteEventRef`, ④ `RteMappedToTaskRef`, ⑤ `RteUsedOsAlarmRef`.

3. `SwcActuatorInstance` 의 매핑 두 개를 같은 방법으로 추가한다.

4. `BswMInstance/RteEventToTaskMapping_0` (Rte 값 파일에서 옴)을 선택한다. `RteEventRef` 는 이미 들어 있으니 나머지를 채운다.

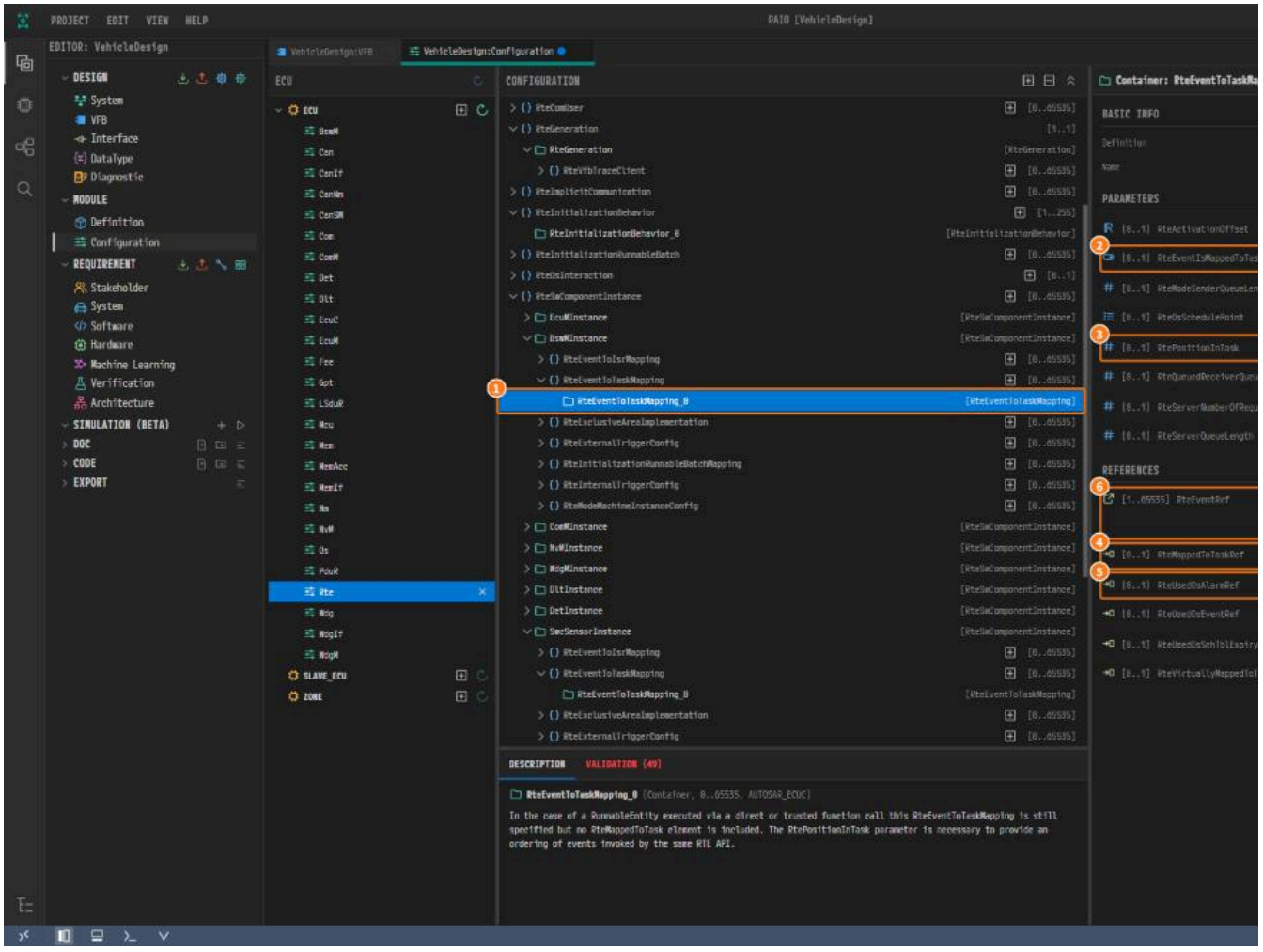


그림 8-6. ① `RteEventToTaskMapping_0`, ② `RteEventIsMappedToTask`, ③ `RtePositionInTask`, ④ `RteMappedToTaskRef`, ⑤ `RteUsedOsAlarmRef`, ⑥

모든 행의 `RteEventIsMappedToTask` 는 `TRUE` 다.

컨테이너	`RtePositionInTask`	`RteEventRef`
`SwcSensorInstance/RteEventToTaskMapping_0` ("+")	-	`SwComponentType/Atomic/Application/SwcSensor/InternalBehavior/SensorCyclicTiming`
`SwcActuatorInstance/RteEventToTaskMapping_0` ("+")	`2`	`SwComponentType/Atomic/Application/SwcActuator/InternalBehavior/ActuatorInitEven`
`SwcActuatorInstance/RteEventToTaskMapping_1` ("+")	`1`	`SwComponentType/Atomic/Application/SwcActuator/InternalBehavior/ActuatorOnDataEv`
`BswMInstance/RteEventToTaskMapping_0` (Rte 값 파일)	`0`	(Rte 값 파일에서 옴)

`Actuator_GetScale` 의 작업 호출 이벤트 `ActuatorGetScaleEvent` 는 매핑하지 않는다. 코드 생성기는 동시 호출할 수 있는 서버 러너블에만 매핑되지 않은 이벤트를 허용하 **TRUE** 로 정해 두었다.

EcuM 모드 머신

1. `EcuMInstance` 아래의 `RteModeMachineInstanceConfig` 에서 "+" 를 누른다.

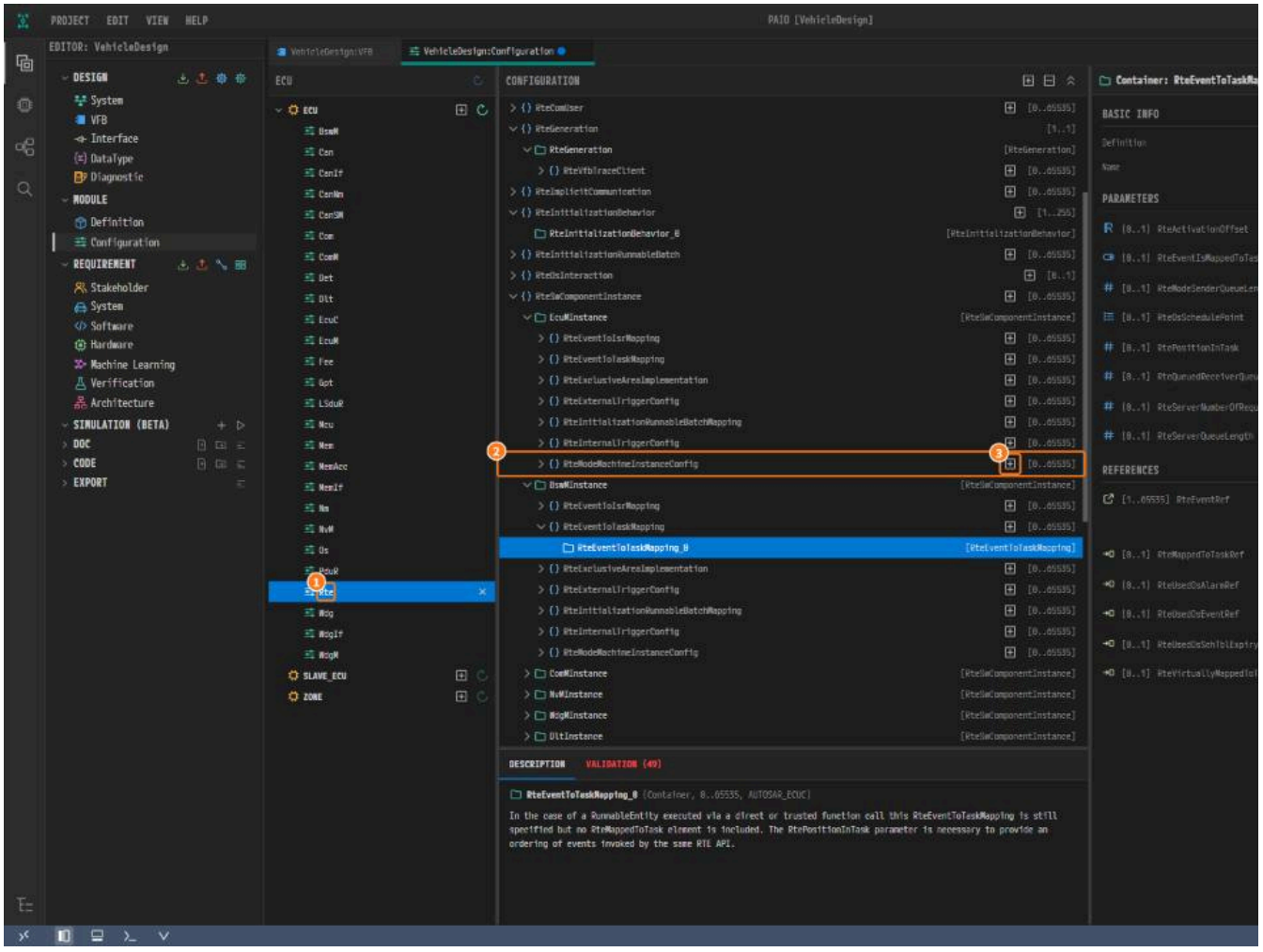


*그림 8-7. ① `Rte`, ② `EcuMInstance` 아래의 `RteModeMachineInstanceConfig`, ③ **+**.*

2. `RteModeMachineQueueLength` 에 `1` 을 입력하고 `Tab` 을 누른다.

3. `RteSwcModeManagerRef` 에서 `PortInterface/ModeSwitch/EcuM_CurrentMode/currentMode` 를 고른다.

4. 트리의 다른 노드를 선택했다가 `RteModeMachineInstanceConfig_0` 을 다시 선택한다. 그제야 **Context Refs** 행이 나타난다. `SwComponentType/Atomic/Service, PROTOTYPE`)를 고른다.

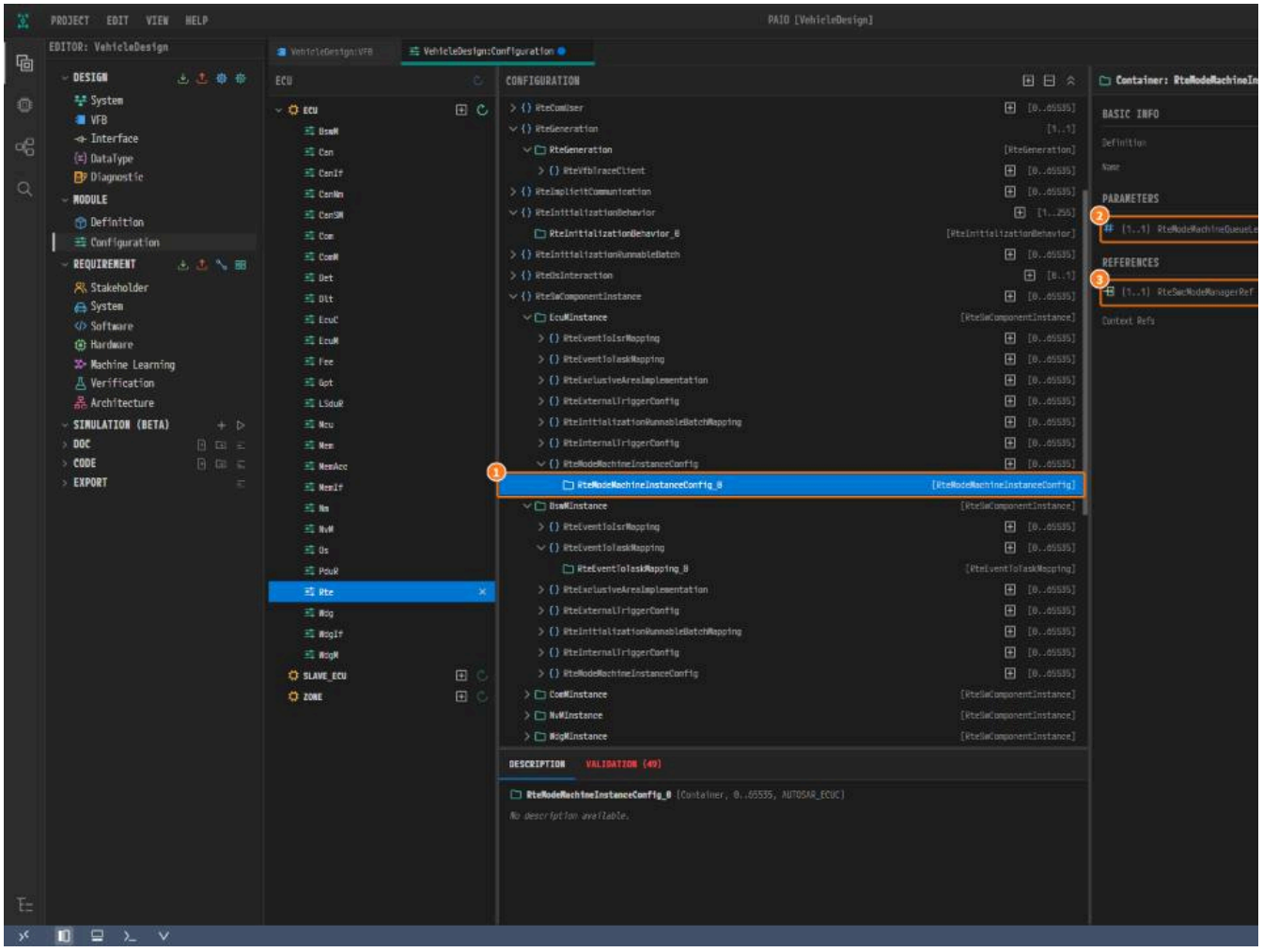


*그림 8-8. ① `RteModeMachineInstanceConfig_0`, ② `RteModeMachineQueueLength`, ③ `RteSwcModeManagerRef`, ④ **Context Refs**.*

Os 활성화

1. `RteOsInteraction` 에서 "+" 를 누른다. 값은 넣지 않는다.
2. 그 아래의 `RteUsedOsActivation` 에서 "+" 를 누른다. 새 `RteUsedOsActivation_0` 에 표의 값을 넣는다.

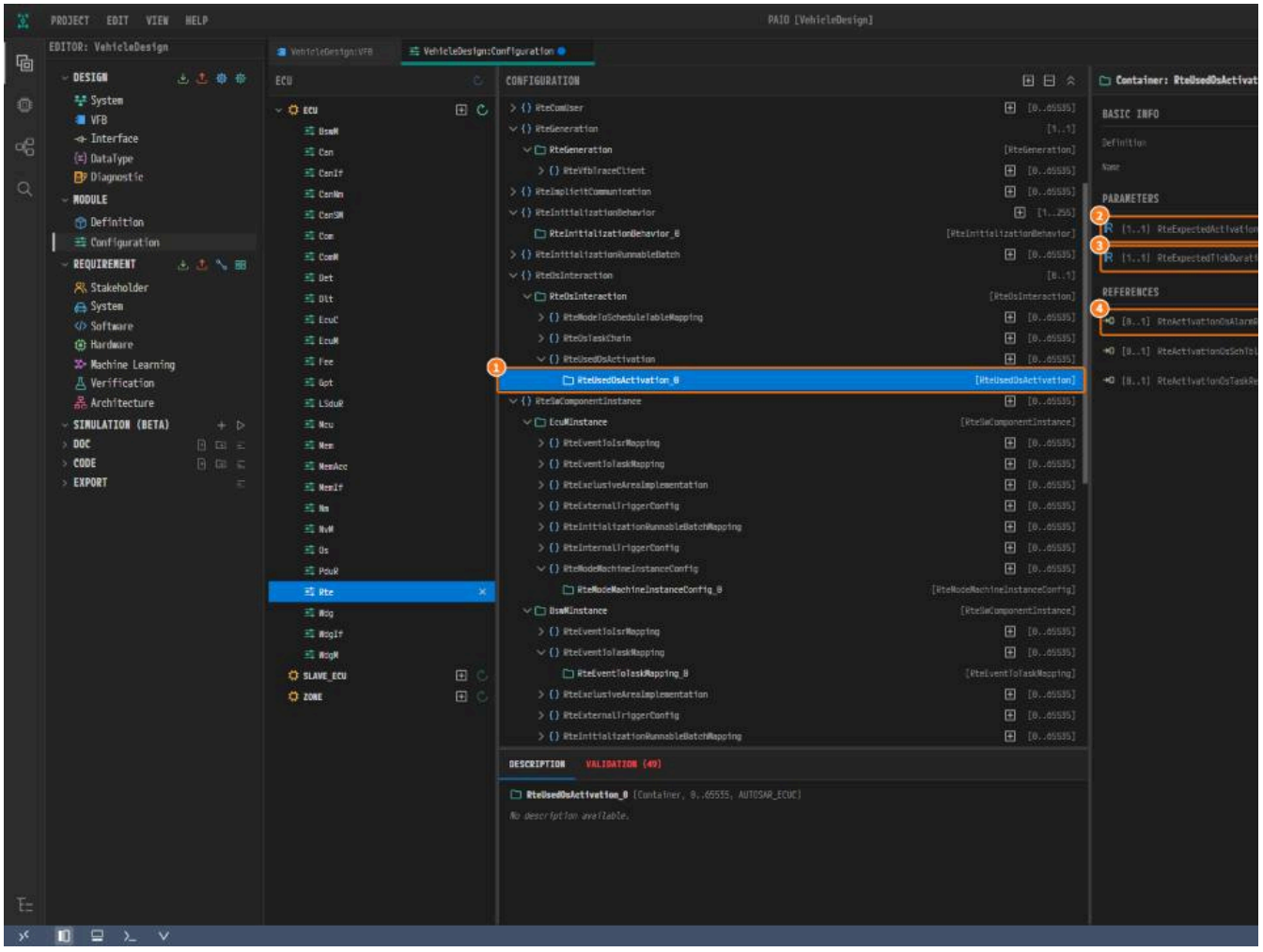


그림 8-9. ① `RteUsedOsActivation_0`, ② `RteExpectedActivationOffset`, ③ `RteExpectedTickDuration`, ④ `RteActivationOsAlarmRef`.

3. `_1` 과 `_2` 도 표를 보고 같은 방법으로 추가한다.

컨테이너	`RteExpectedActivationOffset`	`RteExpectedTickDuration`	`RteActivationOsAlarmRef`
`RteUsedOsActivation_0`	`0`	`0.01`	`Os/OsAlarm_0`
`RteUsedOsActivation_1`	`0`	`0.01`	`Os/OsAlarm_1`
`RteUsedOsActivation_2`	`0`	`0.01`	`Os/OsAlarm_2`

BSW 모듈 인스턴스

Rte 값 파일은 `_0`, `_3`, `_4`, `_5`, `_6` 을 가져왔다. "+" 는 비어 있는 가장 작은 번호를 주므로 EcuM 의 인스턴스(`_1`), Can 의 인스턴스(`_2`), CanNm 의 인스턴스(`_7`) 순서로 추

1. `RteBswModuleInstance_0` (Rte 값 파일에서 옴)을 선택한다. `RteBswImplementationRef` 에 `BswModuleDescription/BswM/BswMImplementation` 을 입력하
2. 그 아래의 `RteBswExclusiveAreaImpl_0` (Rte 값 파일에서 옴)을 선택한다. `RteExclusiveAreaImplMechanism` 을 `ALL_INTERRUPT_BLOCKING` 으로 한다. `RteB

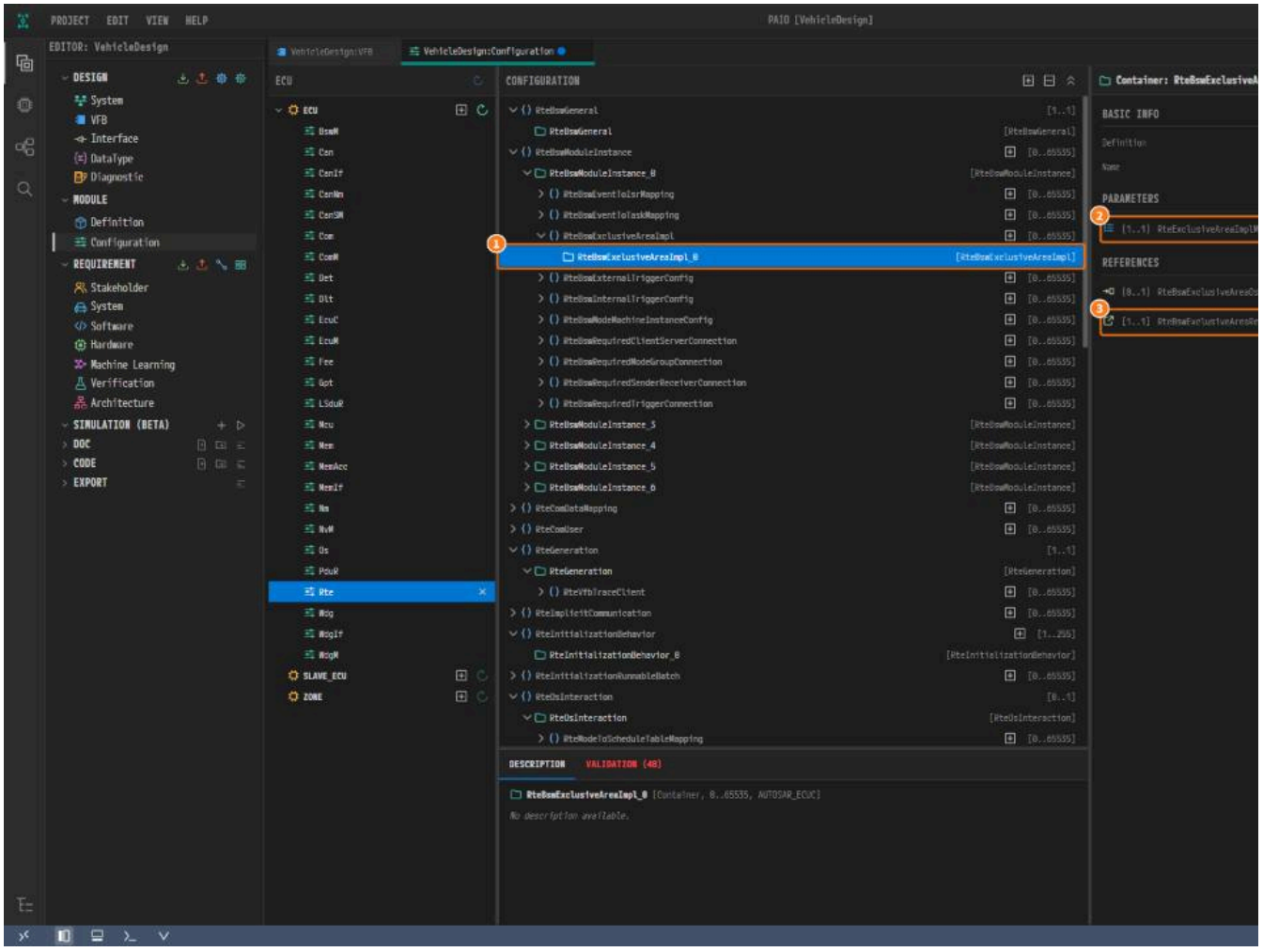


그림 8-10. ① `RteBswExclusiveAreaImpl_0`, ② `RteExclusiveAreaImplMechanism`, ③ `RteBswExclusiveAreaRef` (Rte 값 파일에서 옴).

3. `_3`, `_4`, `_5`, `_6` 도 표를 보고 같은 방법으로 채운다.

4. `RteBswModuleInstance` 에서 "+" 를 누른다. `RteBswModuleInstance_1` 의 `RteBswImplementationRef` 에 표의 EcuM 경로를 입력하고 `Tab` 을 누른다.

5. `RteBswModuleInstance` 에서 "+" 를 한 번 더 누른다.

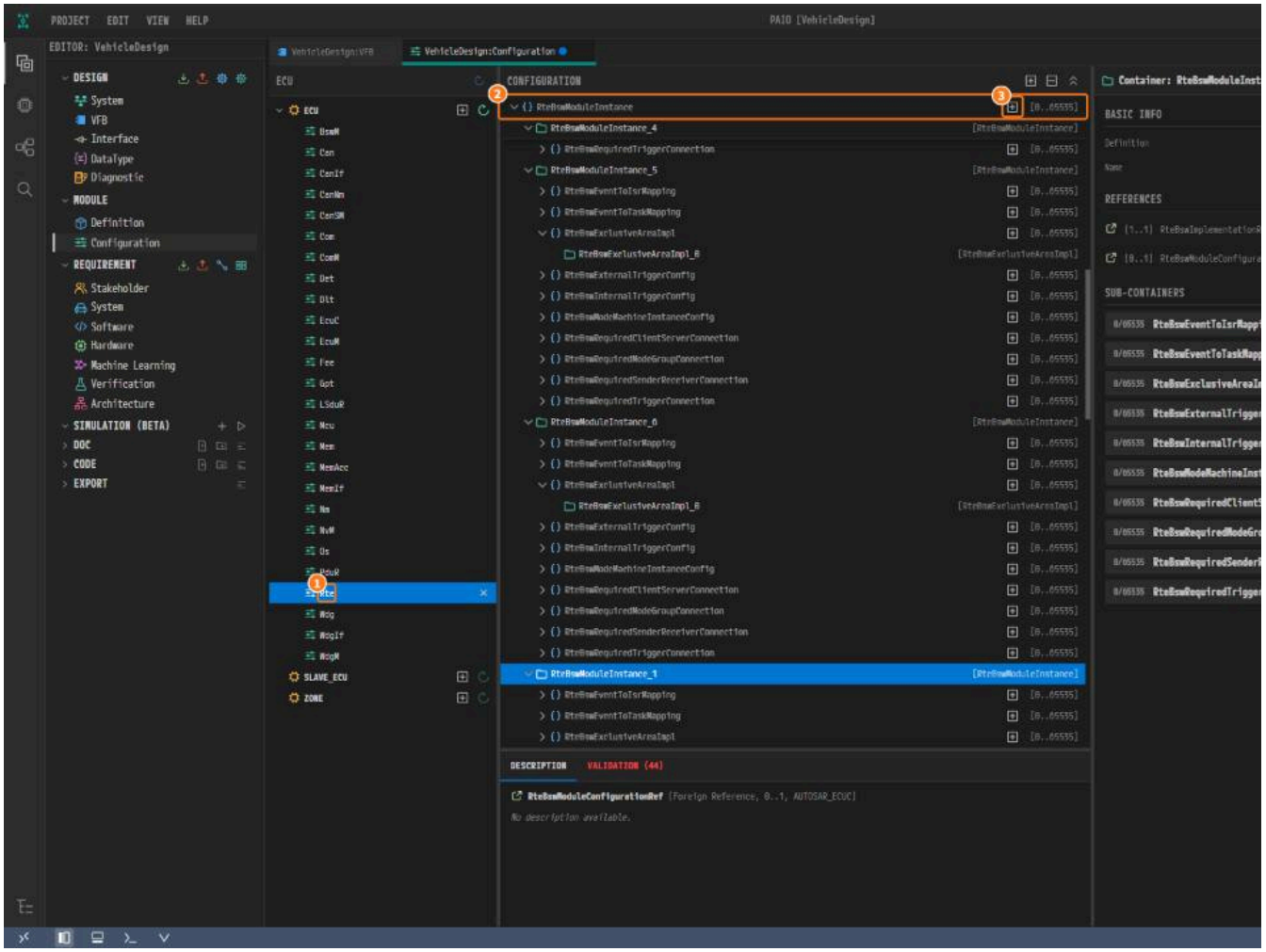


그림 8-11. ① `Rte`, ② `RteBswModuleInstance`, ③ +.

6. `RteBswModuleInstance_2` 의 `RteBswImplementationRef` 에 `BswModuleDescription/Can/CanImplementation` 을 입력하고 `Tab` 을 누른다.

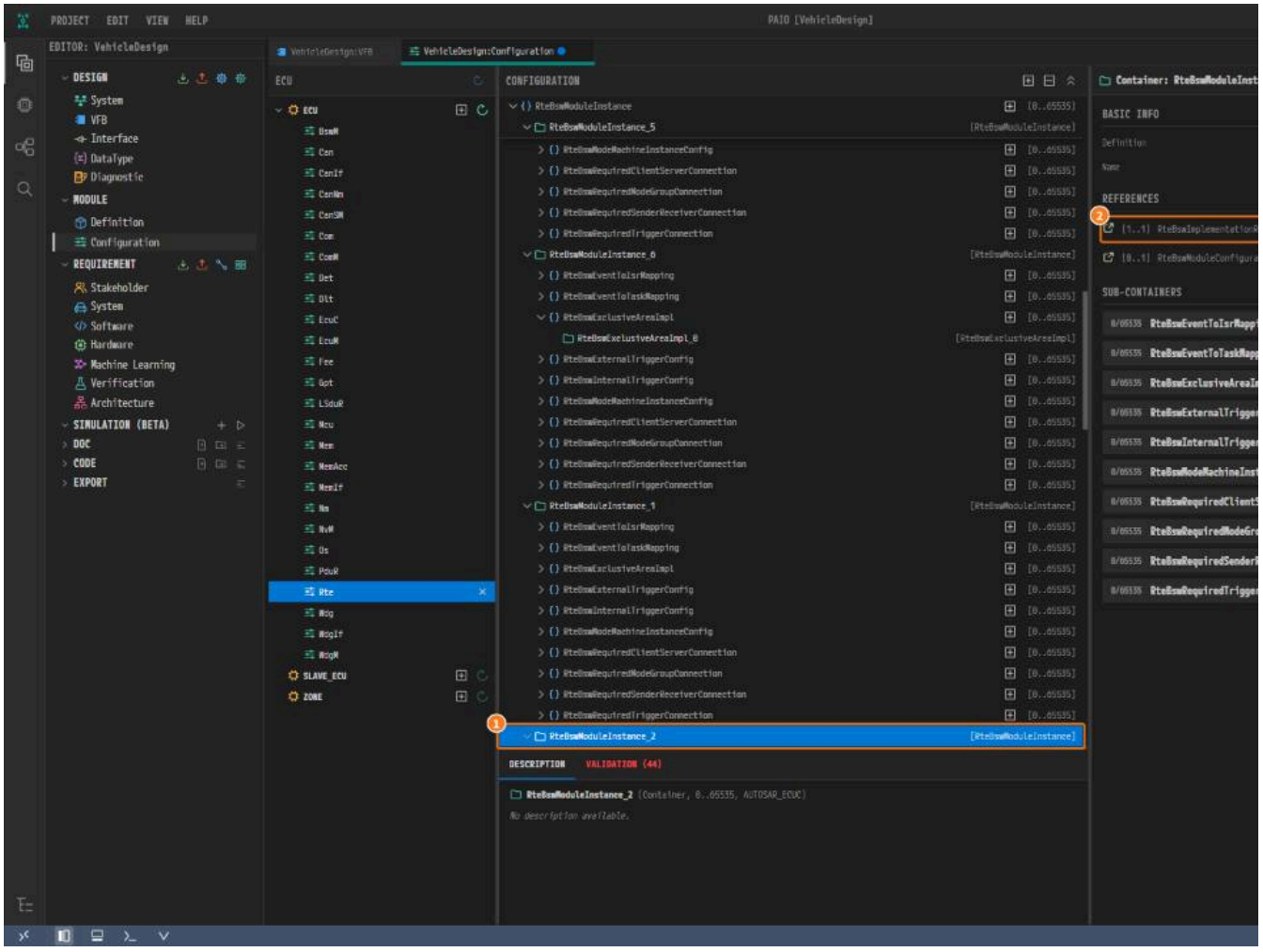


그림 8-12. ① `RteBswModuleInstance_2`, ② `RteBswImplementationRef`.

7. `RteBswModuleInstance` 에서 "+" 를 한 번 더 누른다. `_3` 부터 `_6` 까지가 이미 있으므로 에디터는 `RteBswModuleInstance_7` 이라는 이름을 준다. 그 `RteBswImple` 을 누른다.

인스턴스	출처	`RteBswImplementationRef` (입력)	`RteBswExclusiveAreaImpl_0` 의 `RteExclusiveAreaImplMechanism`
`RteBswModuleInstance_0`	Rte 값 파일	`BswModuleDescription/BswM/BswMImplementation`	`ALL_INTERRUPT_BLOCKING`
`RteBswModuleInstance_1`	"+"	`BswModuleDescription/EcuM/EcuMImplementation`	-
`RteBswModuleInstance_2`	"+"	`BswModuleDescription/Can/CanImplementation`	-
`RteBswModuleInstance_3`	Rte 값 파일	`BswModuleDescription/CanIf/CanIfImplementation`	`ALL_INTERRUPT_BLOCKING`
`RteBswModuleInstance_4`	Rte 값 파일	`BswModuleDescription/CanSM/CanSMImplementation`	`ALL_INTERRUPT_BLOCKING`
`RteBswModuleInstance_5`	Rte 값 파일	`BswModuleDescription/ComM/ComMImplementation`	`ALL_INTERRUPT_BLOCKING`
`RteBswModuleInstance_6`	Rte 값 파일	`BswModuleDescription/Com/ComImplementation`	`ALL_INTERRUPT_BLOCKING`
`RteBswModuleInstance_7`	"+"	`BswModuleDescription/CanNm/CanNmImplementation`	-

BSW 이벤트 매핑

`RteBswEventIsMappedToTask` 에도 같은 함정이 있다. "+" 는 `FALSE` 를 쓰므로 **TRUE** 를 누른다.

1. `RteBswModuleInstance_2` 아래의 `RteBswEventToTaskMapping` 에서 "+" 를 누른다.
2. `RteBswEventIsMappedToTask` 의 **TRUE** 를 누르고 `RteBswPositionInTask` 에 `4` 를 입력한다.

3. `RteBswEventRef` 에 `BswModuleDescription/Can/Can/CanInternalBehavior/CanMainWriteTimingEvent` 를 입력하고 `Tab` 을 누른다.

4. `RteBswMappedToTaskRef` 에서 `Os/OsTask_0` 을, `RteBswUsedOsAlarmRef` 에서 `Os/OsAlarm_0` 을 고른다.

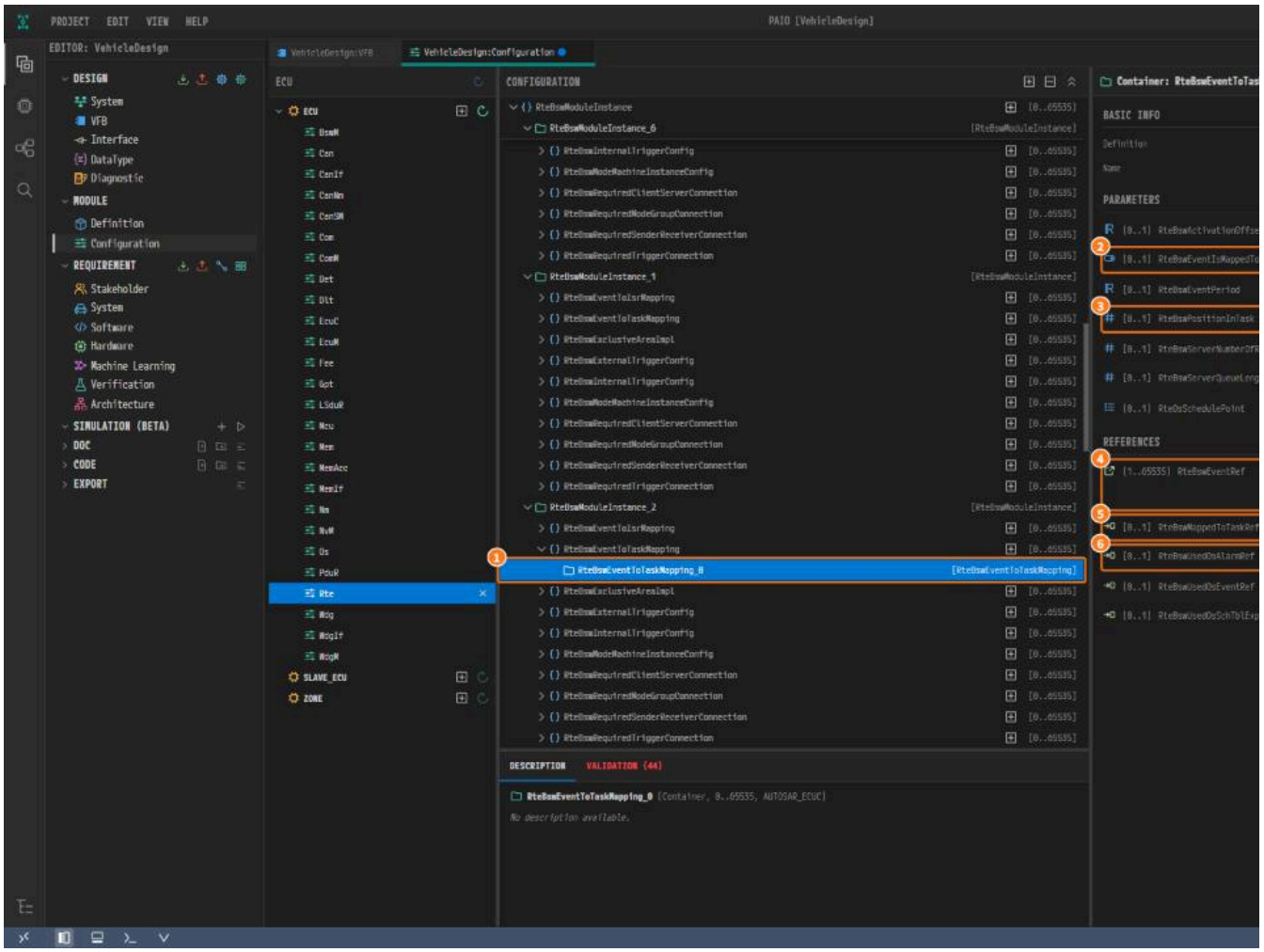


그림 8-13. ① `RteBswEventToTaskMapping_0`, ② `RteBswEventIsMappedToTask`, ③ `RteBswPositionInTask`, ④ `RteBswEventRef`, ⑤ `RteBswMappedTo`

5. 나머지 매핑도 표를 보고 같은 방법으로 추가한다. 모두 "+" 로 추가하며 `RteBswEventIsMappedToTask` 는 `TRUE`, `RteBswMappedToTaskRef` 는 `Os/OsTask_0`, `Rte`

6. 마지막은 `CanNm` 의 메인 함수다. `RteBswModuleInstance_7` 아래의 `RteBswEventToTaskMapping` 에서 "+" 를 누르고 **TRUE** 를 누른 뒤, `RteBswPositionInTask` 입력하고 태스크와 알람을 고른다.

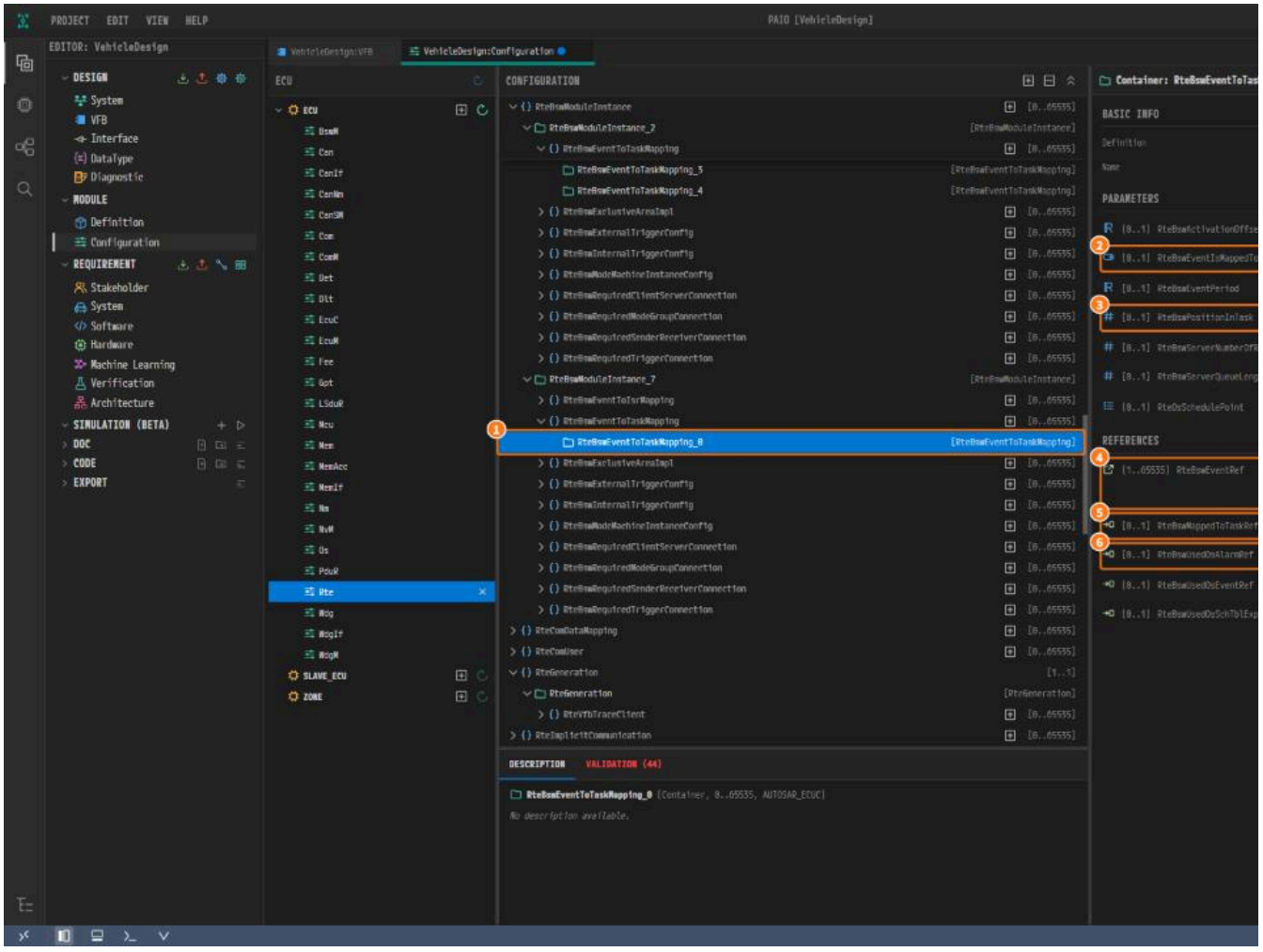


그림 8-14. ① `RteBswEventToTaskMapping_0`, ② `RteBswEventIsMappedToTask`, ③ `RteBswPositionInTask`, ④ `RteBswEventRef`, ⑤ `RteBswMappedTo`

7. `Ctrl+S` 로 모듈을 저장한다.

매핑	`RteBswPositionInTask`	`RteBswEventRef` (입력)
`RteBswModuleInstance_0/RteBswEventToTaskMapping_0`	2	`BswModuleDescription/BswM/BswM/BswMInternalBehavior/BswMMainTimingEvent`
`RteBswModuleInstance_1/RteBswEventToTaskMapping_0`	3	`BswModuleDescription/EcuM/EcuM/EcuMInternalBehavior/EcuMMainTimingEvent`
`RteBswModuleInstance_2/RteBswEventToTaskMapping_0`	4	`BswModuleDescription/Can/Can/CanInternalBehavior/CanMainWriteTimingEven`
`RteBswModuleInstance_2/RteBswEventToTaskMapping_1`	5	`BswModuleDescription/Can/Can/CanInternalBehavior/CanMainReadTimingEvent`
`RteBswModuleInstance_2/RteBswEventToTaskMapping_2`	6	`BswModuleDescription/Can/Can/CanInternalBehavior/CanMainModeTimingEvent`
`RteBswModuleInstance_2/RteBswEventToTaskMapping_3`	7	`BswModuleDescription/Can/Can/CanInternalBehavior/CanMainBusOffTimingEve`
`RteBswModuleInstance_2/RteBswEventToTaskMapping_4`	8	`BswModuleDescription/Can/Can/CanInternalBehavior/CanMainWakeupTimingEve`
`RteBswModuleInstance_4/RteBswEventToTaskMapping_0`	9	`BswModuleDescription/CanSM/CanSM/CanSMInternalBehavior/CanSMMainTimingE`
`RteBswModuleInstance_5/RteBswEventToTaskMapping_0`	10	`BswModuleDescription/ComM/ComM/ComMInternalBehavior/ComMMainFunctionTim`
`RteBswModuleInstance_6/RteBswEventToTaskMapping_0`	11	`BswModuleDescription/Com/Com/ComInternalBehavior/ComMainFunctionRxTimin`
`RteBswModuleInstance_6/RteBswEventToTaskMapping_1`	12	`BswModuleDescription/Com/Com/ComInternalBehavior/ComMainFunctionTxTimin`
`RteBswModuleInstance_7/RteBswEventToTaskMapping_0`	13	`BswModuleDescription/CanNm/CanNm/CanNmInternalBehavior/CanNmMainTimingE`

왜 이렇게 설계하는가

- **매핑.** 러너블이나 BSW 메인 함수는 Rte 매핑이 Os 태스크에 넣어 줄 때만 실행된다.
- **모든 매핑에 TRUE.** 태스크를 지정했는데 플래그가 `FALSE` 인 매핑은 PRTE 가 거부한다 (`RTE_EVENT_MAPPING_TASK_FLAG_MISMATCH`, `RTE_BSW_EVENT_MAPPING_T`
- **Com 이벤트 이름은 컨테이너 이름을 따른다.** `Com` 은 `ComMainFunctionRx`, `ComMainFunctionTx` 컨테이너마다 메인 함수 이벤트를 하나씩 가지므로, 이벤트 이름 끝에 이 붙는다.
- **CanNm 의 인스턴스.** `CanNm` 은 메인 함수가 있으므로 PRTE 는 그 BSW 모듈 인스턴스를 요구한다 (없으면 `RTE_BSW_INSTANCE_MISSING`). `Nm` 은 메인 함수가 없어 인스
- **Rte 값 파일.** 패널이 받지 못하는 값(BswM 폴링 이벤트와 BSW 배타 영역)과, 임포트가 지워 버릴 SW-C 인스턴스만 담는다.
- **위치.** `RteBswPositionInTask` 는 `OsTask_0` 안에서 BSW 메인 함수가 불리는 순서를 정한다.

확인하기

- [ ] 모든 `RteEventToTaskMapping` 과 `RteBswEventToTaskMapping` 의 태스크 매핑 플래그가 **TRUE** 다.
- [ ] `RteBswModuleInstance_0` 부터 `_7` 까지 모두 `RteBswImplementationRef` 가 있다.
- [ ] `RteModeMachineInstanceConfig_0` 의 **Context Refs** 행이 채워져 있다.
- [ ] 탭 `VehicleDesign:Configuration` 에 수정 표시가 없다.

다음 단계

Communication 설정에서 `EcuC` 와 CAN 드라이버부터 `LSduR`, `PduR`, `Com` 까지 CAN 통신 스택을 설정한다.

Communication 설정

이 장에서 할 일

AUTOSAR System 튜토리얼에서 노드 `ECU` 를 CAN 클러스터 `CanCluster` 에 연결했으므로, 6장의 PRTE Platform Import 는 CAN 통신 스택 모듈을 빈 문서로 가져왔다. 이 장

- `EcuC` 는 여러 모듈이 함께 쓰는 PDU 를 담는다. 여기서 네트워크 관리 PDU `PduNmTx` 와 `PduNmRx` 를 만든다.
- `Can` 은 CAN 컨트롤러와 그 하드웨어 오브젝트 두 개(송신용 하나, 수신용 하나)를 다루는 드라이버다.
- `CanIf` 는 드라이버와 위쪽 모듈 사이의 인터페이스다. 각 PDU 에 CAN id 를 붙인다.
- `ComM` 은 ECU 의 통신 채널을 관리한다.
- `CanSM` 은 CAN 네트워크의 상태와 버스 오프에서의 회복을 관리한다.
- `Nm` 은 버스 종류에 매이지 않는 네트워크 관리 계층이다. 채널을 `CanNm` 에 넘긴다.
- `CanNm` 은 CAN 의 네트워크 관리다. `EcuC` 의 NM PDU `PduNmTx` 와 `PduNmRx` 를 보내고 받는다.
- `LSduR` 은 NM PDU 를 `CanIf` 와 `CanNm` 사이에서 넘긴다. 함께 일하는 두 모듈을 선언하고 NM PDU 마다 경로를 하나씩 둔다.
- `PduR` 은 통신 모듈 사이에서 애플리케이션 PDU 를 라우팅한다. 이 설계에는 그런 PDU 가 없으므로 필수 컨테이너 두 개만 둔다.
- `Com` 은 시그널을 PDU 에 담고 PDU 에서 꺼낸다. 이 모듈도 필수 컨테이너 두 개만 둔다.

시작하기 전에

- Rte 설정을 마친 뒤 시작한다.
- **MODULE** 아래의 **Configuration** 을 눌러 ECU Configuration 에디터를 연다.
- 모듈은 다음 순서로 설정한다. 아래쪽 모듈은 위쪽 모듈의 컨테이너를 목록에서 고른다.

순서	모듈	추가하는 컨테이너	참조하는 모듈
1	`EcuC`	6	—
2	`Can`	6	`Mcu` (PRTE Setup 에서 넣었다)
3	`CanIf`	12	`Can`, `EcuC`
4	`ComM`	4	—
5	`CanSM`	4	`ComM`, `CanIf`
6	`Nm`	7	`ComM`
7	`CanNm`	4	`ComM`, `EcuC`
8	`LSduR`	10	`CanIf`, `CanNm`, `EcuC`
9	`PduR`	2	—
10	`Com`	2	—

각 절의 표에 넣을 값이 모두 있다. 시간 값의 단위는 초다.

> **참고.** **CONFIGURATION** 트리에서 행 오른쪽 끝의 **+** 를 누르면 그 정의의 컨테이너가 추가되고 선택된다. 새 컨테이너에는 정의의 기본값이 이미 들어 있으므로 표에 있는 값만 바꾸
> 는 목록에서 고른다. 컨테이너는 하나씩 만든다. **+** 를 누르고, 표의 이름이 다르면 이름을 바꾸고, 값을 모두 넣은 뒤 다음 컨테이너로 넘어간다. 모듈을 마치면 `Ctrl+S` 로 한 번 저

> **주의.** 모듈 목록에서 `ECU` 옆의 **PRTE Auto Configuration** 아이콘은 누르지 않는다. 이 튜토리얼은 BSW 를 직접 설정하며, 이 아이콘은 값을 스스로 써 넣는다.

> **주의.** `Nm` 과 `CanNm` 의 **TRUE/FALSE** 항목 가운데 상당수는 기본값이 없으므로, **+** 는 이를 비워 두고 PRTE 는 빈 항목마다 멈춘다. 그 항목마다 **FALSE**(표에서 **TRUE** 인 것은 **TRU**
> 로, 그때는 한 번 더 누른다.

EcuC 설정

`EcuC` 를 가장 먼저 설정한다. `CanIf`, `CanNm`, `LSduR` 이 그 PDU 를 목록에서 고른다. `EcuC` 자체는 생성 파일을 만들지 않는다.

컨테이너	추가하는 방법	항목	값
`EcucConfigSet`	`EcucConfigSet` 의 **+**		추가만 한다
`EcucPduCollection`	`EcucConfigSet` 아래 `EcucPduCollection` 의 **+**	**PduIdTypeEnum**	`UINT16`
		PduLengthTypeEnum	`UINT32`
`PduNmTx`	`EcucPduCollection` 아래 `Pdu` 의 **+**	**Name**	`PduNmTx`
		PduLength	`8`
`PduNmRx`	`EcucPduCollection` 아래 `Pdu` 의 **+**	**Name**	`PduNmRx`
		PduLength	`8`
`EcucHardware`	`EcucHardware` 의 **+**		추가만 한다
`Core0`	`EcucHardware` 아래 `EcucCoreDefinition` 의 **+**	**Name**	`Core0`
		EcucCoreId	`0`

1. 모듈 목록에서 `EcuC` 를 누른다. `EcucConfigSet` 을 **+** 로 추가하고, 그 아래에 `EcucPduCollection` 을 추가한 뒤 표에 따라 값 두 개를 고른다.

2. `EcucPduCollection` 아래 `Pdu` 행의 **+** 를 누른다.

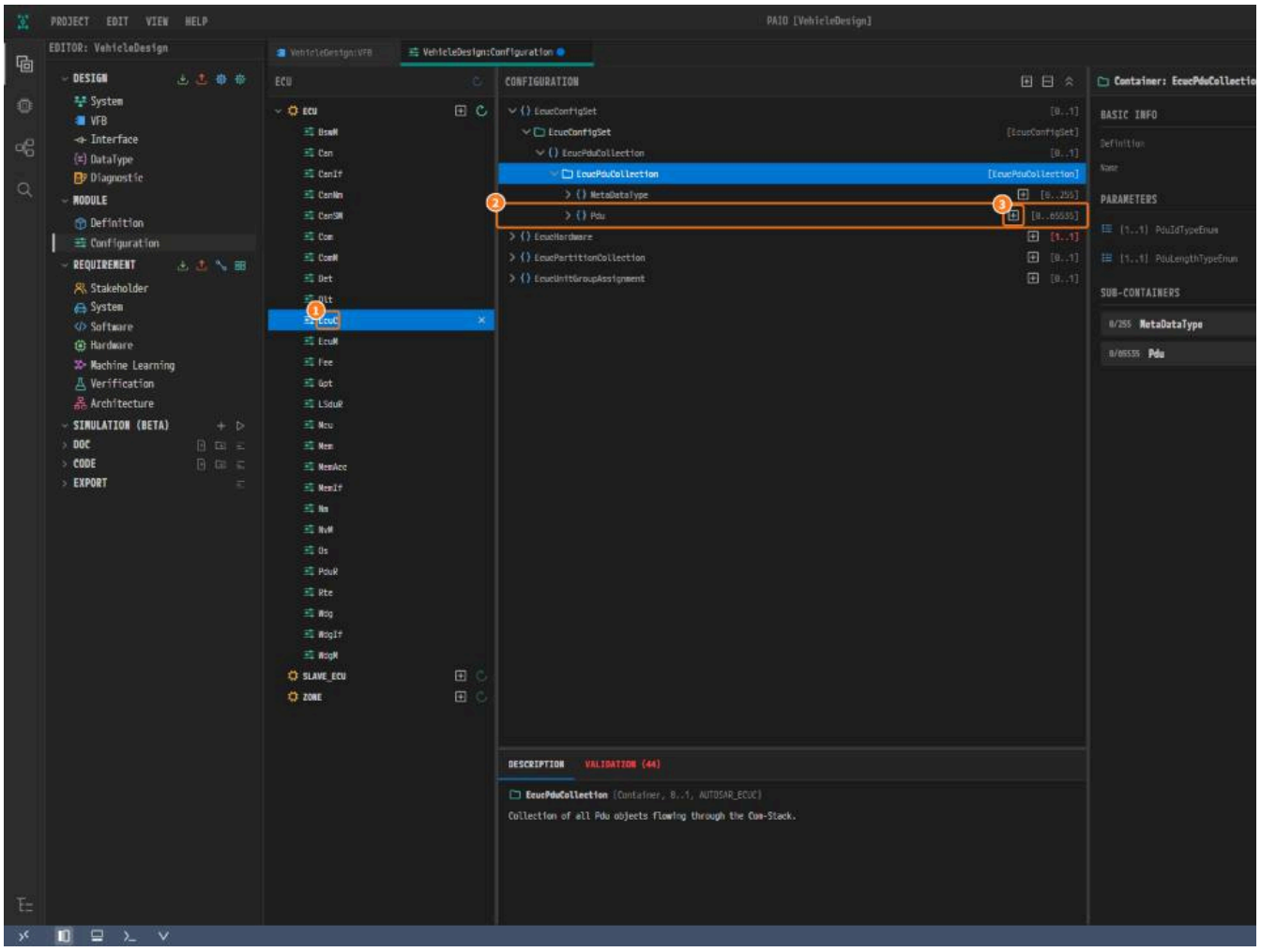


*그림 9-1. ① `EcuC`, ② `Pdu`, ③ **+**.*

3. 에디터가 `Pdu_0` 을 추가하고 선택한다. **Name** 에 `PduNmTx` 를 입력하고 `Tab` 키를 누른다. **PduLength** 에 `8` 을 입력하고 `Tab` 키를 누른다.

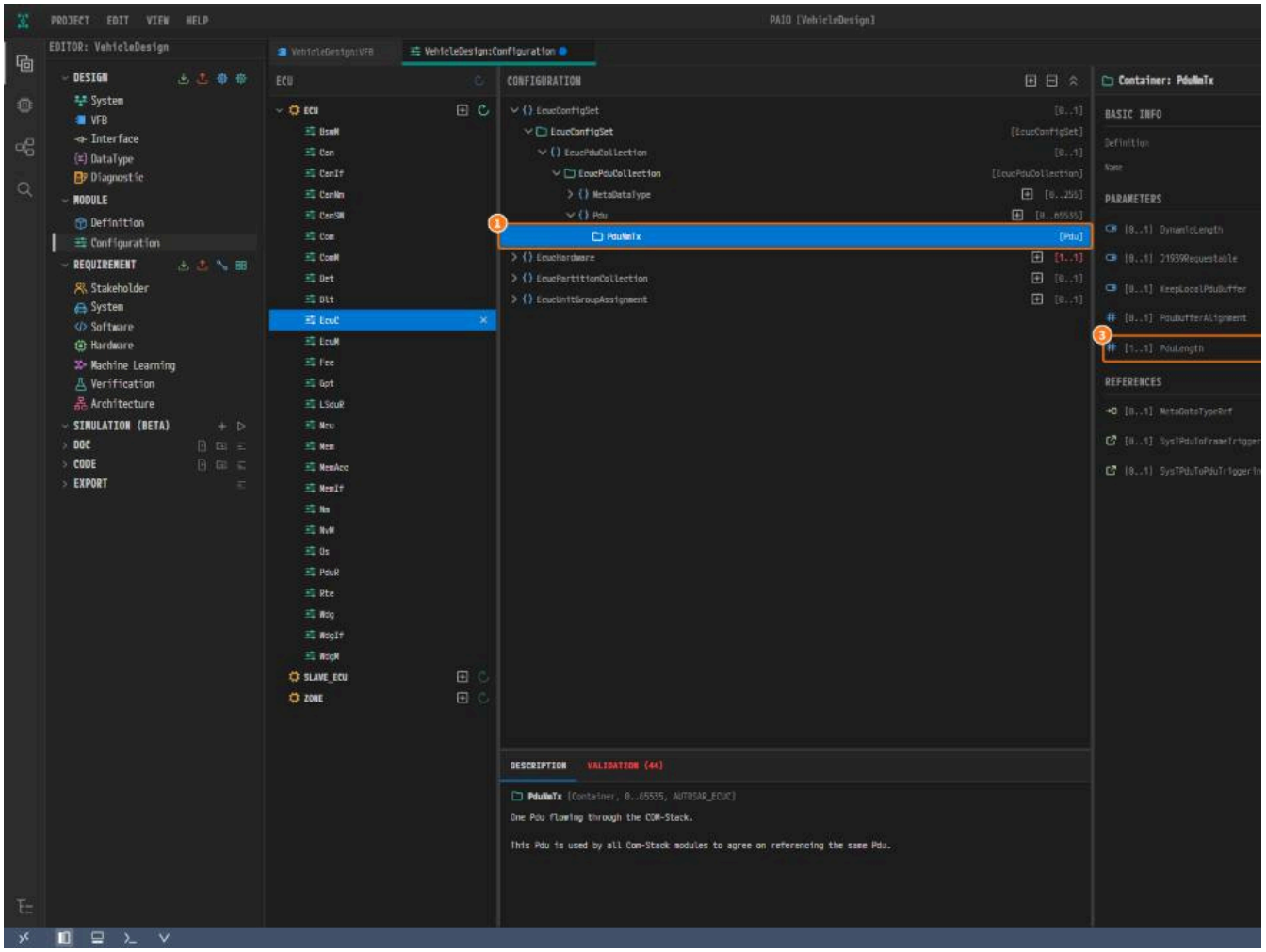


*그림 9-2. ① `PduNmTx`, ② **Name**, ③ **PduLength**.*

4. `PduNmRx` 도 같은 방법으로 만든다. 그 이름이 비어 있으므로 **+** 는 다시 `Pdu_0` 을 주며, 이를 `PduNmRx` 로 바꾼다.

5. `EcucHardware` 를 **+** 로 추가한다. 그 아래 `EcucCoreDefinition` 행의 **+** 를 누르고, 새 `EcucCoreDefinition_0` 의 이름을 `Core0` 으로 바꾼 뒤 **EcucCoreId** 에 `0` 을 다. `Ctrl+S` 로 저장한다.

Can 설정

컨테이너	추가하는 방법	항목	
`CanGeneral`	`CanGeneral` 의 +	**CanGlobalTimeSupport**	**FALSE**
		CanIndex	`0`
		CanMainFunctionModePeriod	`0.01`
		CanMultiplexedTransmission	**FALSE**
		CanTimeoutDuration	`0.001`
`CanConfigSet`	`CanConfigSet` 의 +		추가만 한다
`CanController_0`	`CanConfigSet` 아래 `CanController` 의 +	**CanBusoffProcessing**	`POLLING`
		CanControllerActivation	**TRUE**
		CanControllerBaseAddress	`0`
		CanControllerId	`0`
		CanRxProcessing	`POLLING`
		CanTxProcessing	`POLLING`
		CanWakeupProcessing	`POLLING`
		CanWakeupSupport	**FALSE**
		CanControllerDefaultBaudrate	`Can/CanConfigSe`
		CanCpuClockRef	`Mcu/McuModuleCo`
`CanControllerBaudrateConfig_0`	`CanController_0` 아래 `CanControllerBaudrateConfig` 의 +	**CanControllerBaudRate**	`500`
		CanControllerPropSeg, **CanControllerSeg1**, **CanControllerSeg2**, **CanControllerSyncJumpWidth**	`0`
`CanHardwareObject_0`	`CanConfigSet` 아래 `CanHardwareObject` 의 +	**CanHandleType**	`BASIC`
		CanIdType	`STANDARD`
		CanObjectId	`0`
		CanObjectType	`TRANSMIT`
		CanControllerRef	`Can/CanConfigSe`
`CanHardwareObject_1`	`CanConfigSet` 아래 `CanHardwareObject` 의 +	**CanHandleType**	`BASIC`
		CanIdType	`STANDARD`
		CanObjectId	`1`
		CanObjectType	`RECEIVE`
		CanControllerRef	`Can/CanConfigSe`

1. 모듈 목록에서 `Can` 을 누른다. **CONFIGURATION** 트리에서 `CanGeneral` 행의 + 를 누른다.

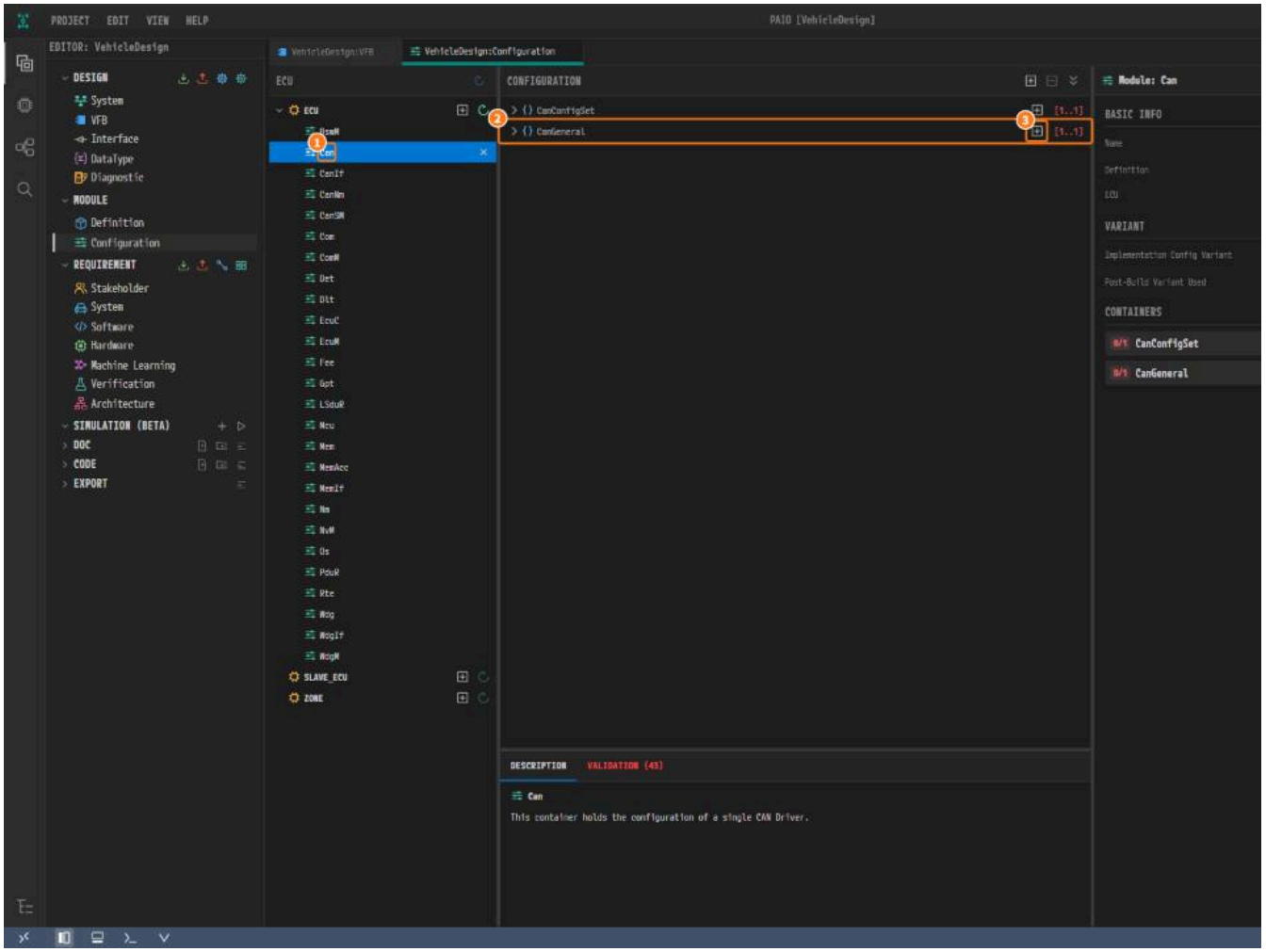


그림 9-3. ① `Can`, ② `CanGeneral`, ③ +.

2. 표에 따라 `CanGeneral` 의 값 다섯 개를 넣는다.

> **주의.** 이미 선택된 **TRUE** 나 **FALSE** 버튼을 누르면 값이 지워지고 행에 빨간 테두리가 생긴다. 그때는 같은 버튼을 한 번 더 누른다.

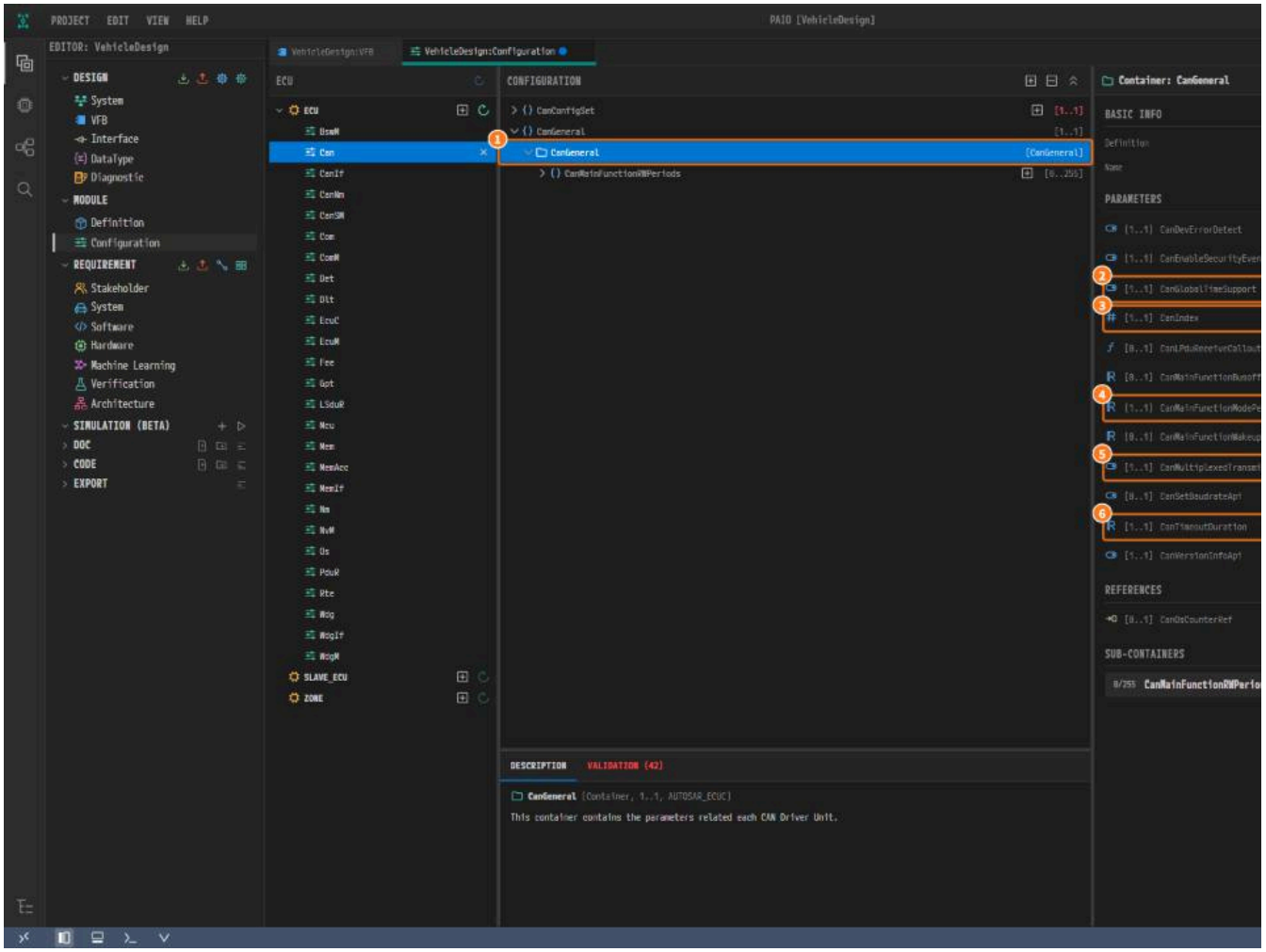


*그림 9-4. ① `CanGeneral`, ② **CanGlobalTimeSupport**, ③ **CanIndex**, ④ **CanMainFunctionModePeriod**, ⑤ **CanMultiplexedTransmission**, ⑥ **CanTimeoutDur***

3. `CanConfigSet` 행의 **+** 를 누른다. 그 아래 `CanController` 행의 **+** 를 누르고, 새로 생긴 `CanController_0` 아래 `CanControllerBaudrateConfig` 행의 **+** 를 누른다
어야 한다.

4. `CanController_0` 을 누르고 표에 따라 값 열 개를 넣는다. **CanControllerDefaultBaudrate** 에는 3단계의 `CanControllerBaudrateConfig_0` 을 고른다.

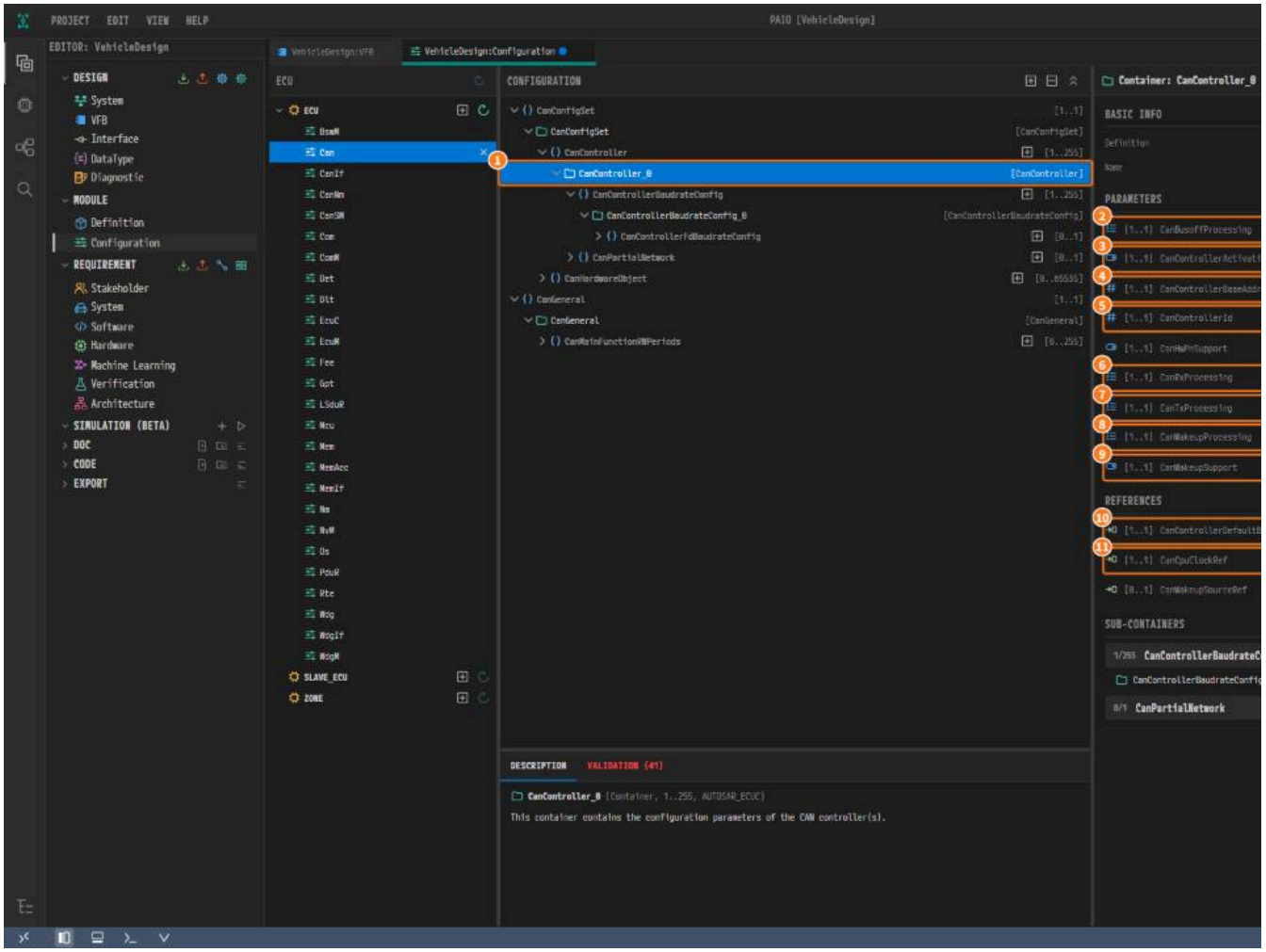


*그림 9-5. ① `CanController_0`, ② **CanBusoffProcessing**, ③ **CanControllerActivation**, ④ **CanControllerBaseAddress**, ⑤ **CanControllerId**, ⑥ **CanRxProce**
CanWakeupProcessing, ⑨ **CanWakeupSupport**, ⑩ **CanControllerDefaultBaudrate**, ⑪ **CanCpuClockRef**.*

5. `CanControllerBaudrateConfig_0` 을 누르고 값 다섯 개를 넣는다.

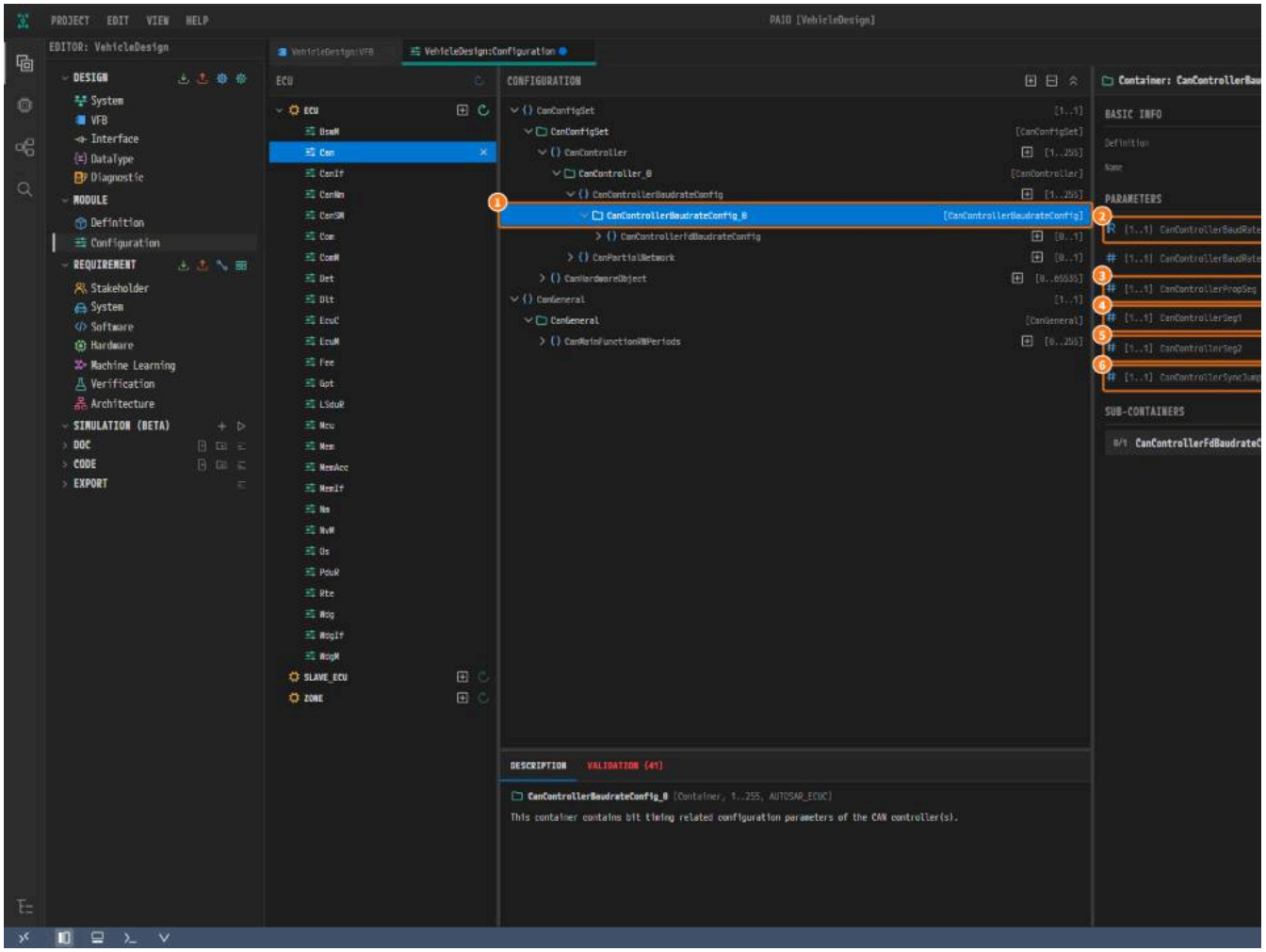


*그림 9-6. ① `CanControllerBaudrateConfig_0`, ② **CanControllerBaudRate**, ③ **CanControllerPropSeg**, ④ **CanControllerSeg1**, ⑤ **CanControllerSeg2**, ⑥ (*

6. `CanConfigSet` 아래 `CanHardwareObject` 행의 **+** 를 누른다.

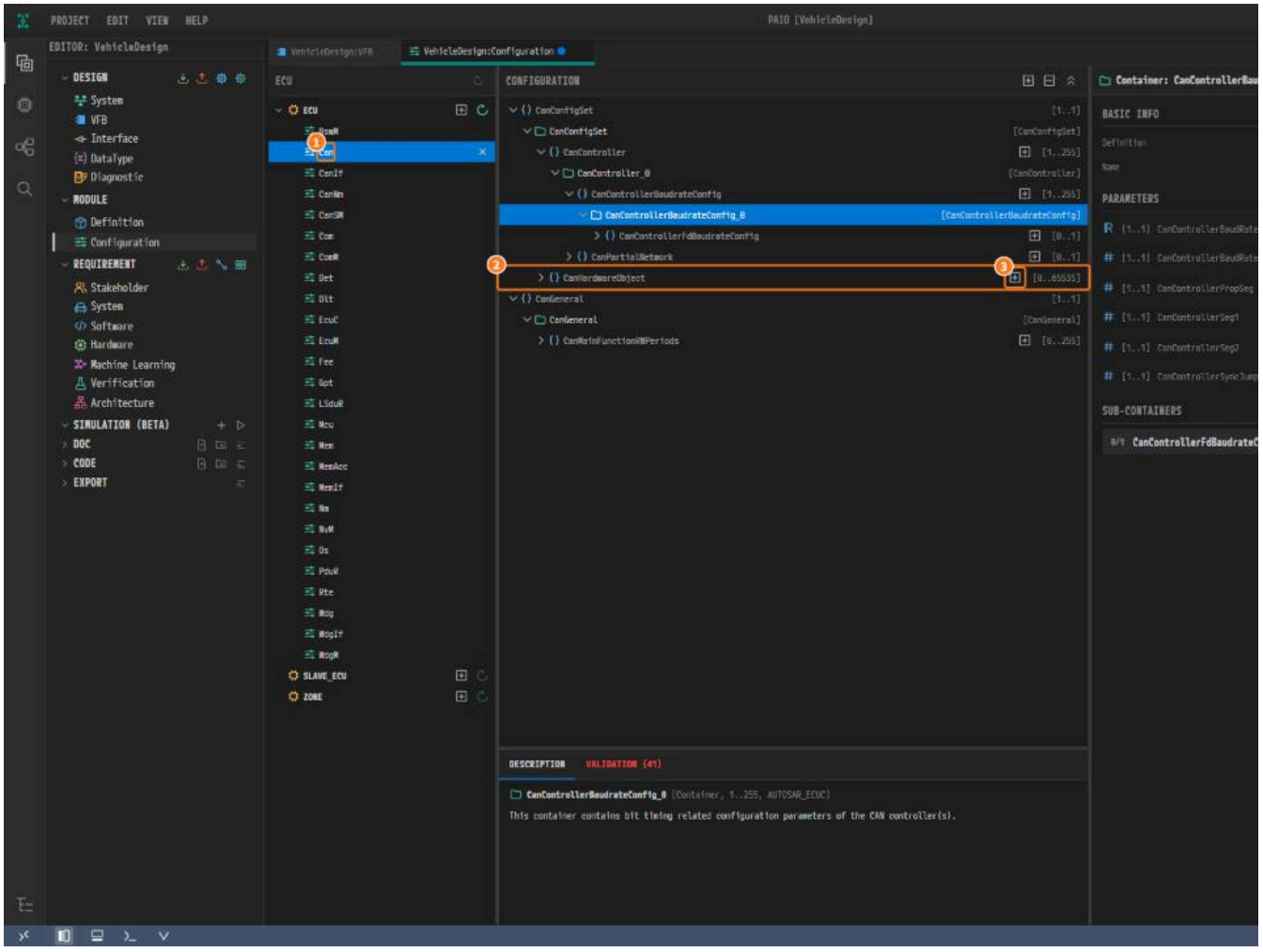


그림 9-7. ① `Can`, ② `CanHardwareObject`, ③ +.

7. 표에 따라 `CanHardwareObject_0` 의 값 다섯 개를 넣는다. 이것이 송신 오브젝트다.

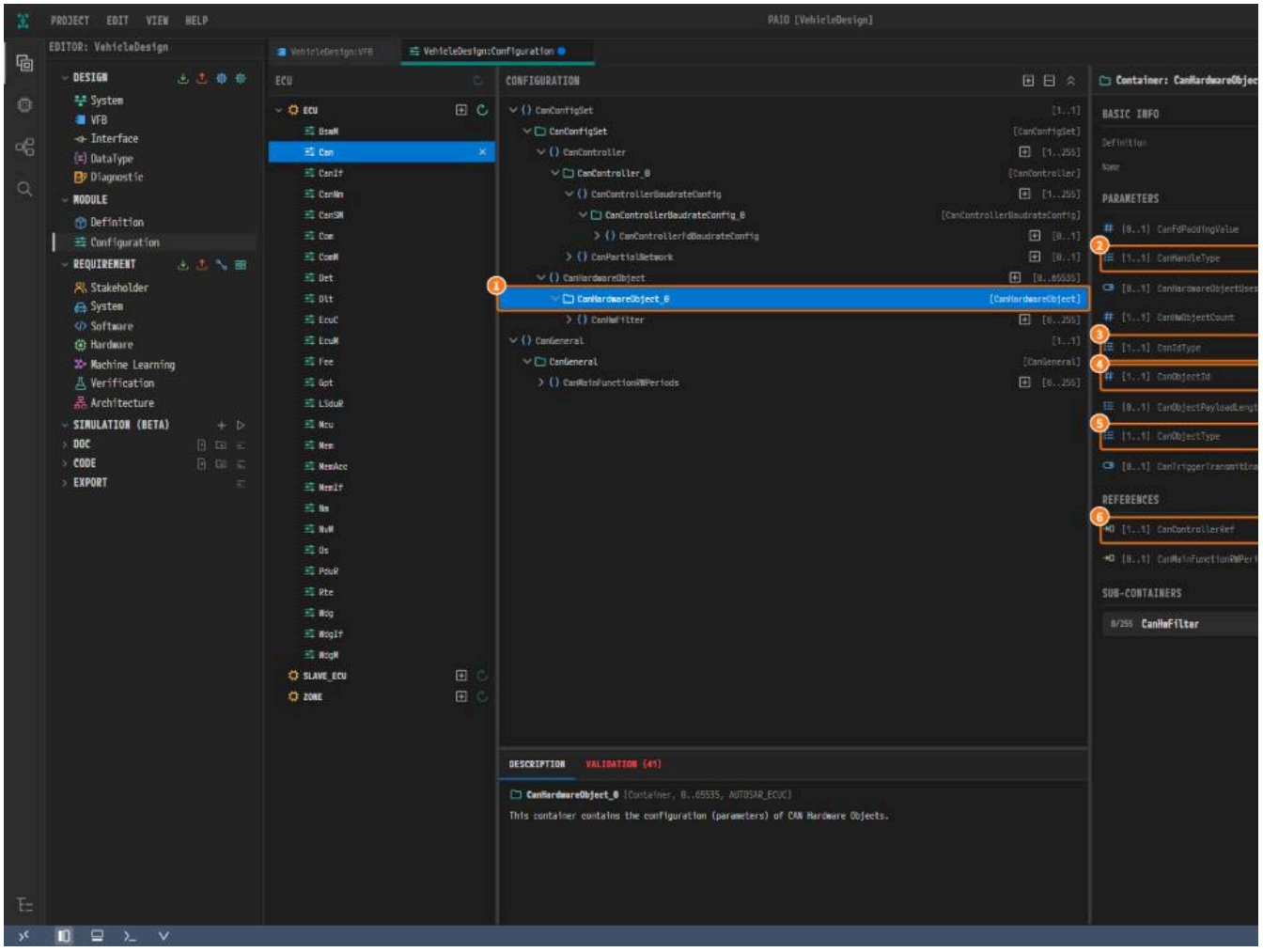


*그림 9-8. ① `CanHardwareObject_0`, ② **CanHandleType**, ③ **CanIdType**, ④ **CanObjectId**, ⑤ **CanObjectType**, ⑥ **CanControllerRef**.*

8. 수신 오브젝트 `CanHardwareObject_1` 도 표에 따라 같은 방법으로 만든다. `Ctrl+S` 로 저장한다.

CanIf 설정

컨테이너	추가하는 방법	항목	값
`CanIfPublicCfg`	`CanIfPublicCfg` 의 +	**CanIfGlobalTimeSupport**	**FALSE**
		CanIfPublicHandleTypeEnum	`UINT16`
		CanIfPublicTxConfirmPollingSupport	**TRUE**
		CanIfPublicTxBuffering	**TRUE**
`CanIfPrivateCfg`	`CanIfPrivateCfg` 의 +	**CanIfPrivateSoftwareFilterType**	`LINEAR`
`CanIfDispatchCfg`	`CanIfDispatchCfg` 의 +	**CanIfDispatchUserCtrlBusOffUL**	`CAN_SM`
		CanIfDispatchUserCtrlModeIndicationUL	`CAN_SM`
`CanIfInitCfg`	`CanIfInitCfg` 의 +	**CanIfInitCfgSet**	`CanIf_Config`
`CanIfInitHohCfg_0`	`CanIfInitCfg` 아래 `CanIfInitHohCfg` 의 +		추가만 한다
`CanIfCtrlDrvCfg_0`	`CanIfCtrlDrvCfg` 의 +	**CanIfCtrlDrvNameRef**	`Can/CanGeneral`
		CanIfCtrlDrvInitHohConfigRef	`CanIf/CanIfInitCfg/CanIfInitHohCfg_0`
`CanIfCtrlCfg_0`	`CanIfCtrlDrvCfg_0` 아래 `CanIfCtrlCfg` 의 +	**CanIfCtrlId**	`0`
		CanIfCtrlCanCtrlRef	`Can/CanConfigSet/CanController_0`
`CanIfHthCfg_0`	`CanIfInitHohCfg_0` 아래 `CanIfHthCfg` 의 +	**CanIfHthCanCtrlIdRef**	`CanIf/CanIfCtrlDrvCfg_0/CanIfCtrlCfg_0`
		CanIfHthIdSymRef	`Can/CanConfigSet/CanHardwareObject_0`
`CanIfHrhCfg_0`	`CanIfInitHohCfg_0` 아래 `CanIfHrhCfg` 의 +	**CanIfHrhSoftwareFilter**	**TRUE**
		CanIfHrhCanCtrlIdRef	`CanIf/CanIfCtrlDrvCfg_0/CanIfCtrlCfg_0`
		CanIfHrhIdSymRef	`Can/CanConfigSet/CanHardwareObject_1`
`CanIfBufferCfg_0`	`CanIfInitCfg` 아래 `CanIfBufferCfg` 의 +	**CanIfBufferSize**	`1`
		CanIfBufferHthRef	`CanIf/CanIfInitCfg/CanIfInitHohCfg_0/CanIfHthCf`
`CanIfTxPduCfg_0`	`CanIfInitCfg` 아래 `CanIfTxPduCfg` 의 +	**CanIfTxPduCanId**	`1281`
		CanIfTxPduId	`0`
		CanIfTxPduType	`STATIC`
		CanIfTxPduBufferRef	`CanIf/CanIfInitCfg/CanIfBufferCfg_0`
		CanIfTxPduRef	`EcuC/EcucConfigSet/EcucPduCollection/PduNmTx`
`CanIfRxPduCfg_0`	`CanIfInitCfg` 아래 `CanIfRxPduCfg` 의 +	**CanIfRxPduCanId**	`1282`
		CanIfRxPduDataLength	`8`
		CanIfRxPduId	`0`
		CanIfRxPduHrhIdRef	`CanIf/CanIfInitCfg/CanIfInitHohCfg_0/CanIfHrhCf`
		CanIfRxPduRef	`EcuC/EcucConfigSet/EcucPduCollection/PduNmRx`

1. 모듈 목록에서 `CanIf` 를 누른다. `CanIfPublicCfg` 를 **+** 로 추가하고 값 네 개를 넣는다. **CanIfPublicTxBuffering** 은 반드시 **TRUE** 여야 한다. 송신 PDU 가 버퍼 `CanIfBu` `CANIF_BUFFERING_GATE_INCONSISTENT` 로 멈춘다.

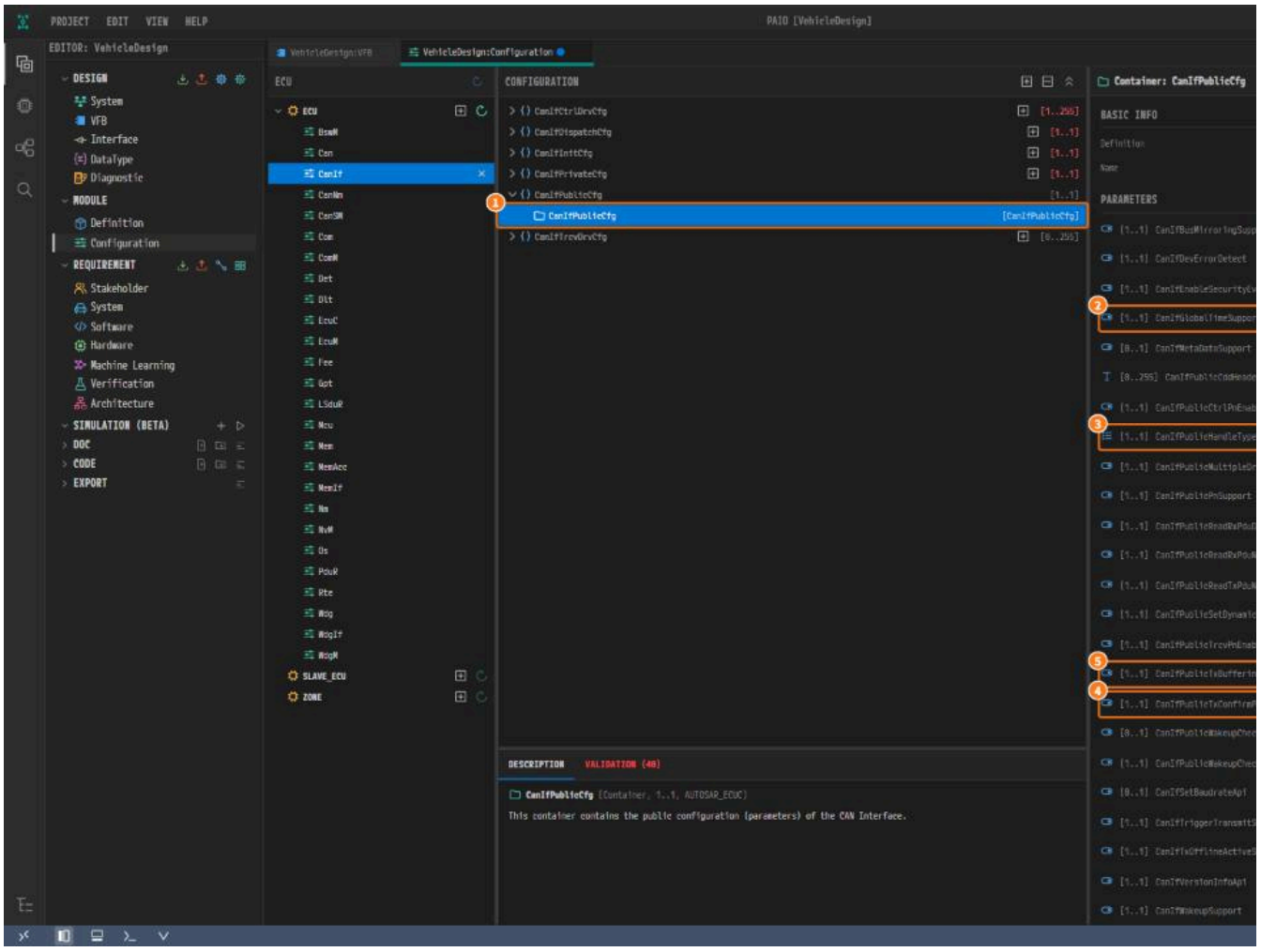


*그림 9-9. ① `CanIfPublicCfg`, ② **CanIfGlobalTimeSupport**, ③ **CanIfPublicHandleTypeEnum**, ④ **CanIfPublicTxConfirmPollingSupport**, ⑤ **CanIfPublicTxB***

2. `CanIfPrivateCfg` 를 추가하고 `LINEAR` 를 고른다. 이어서 `CanIfDispatchCfg` 행의 **+** 를 누른다.

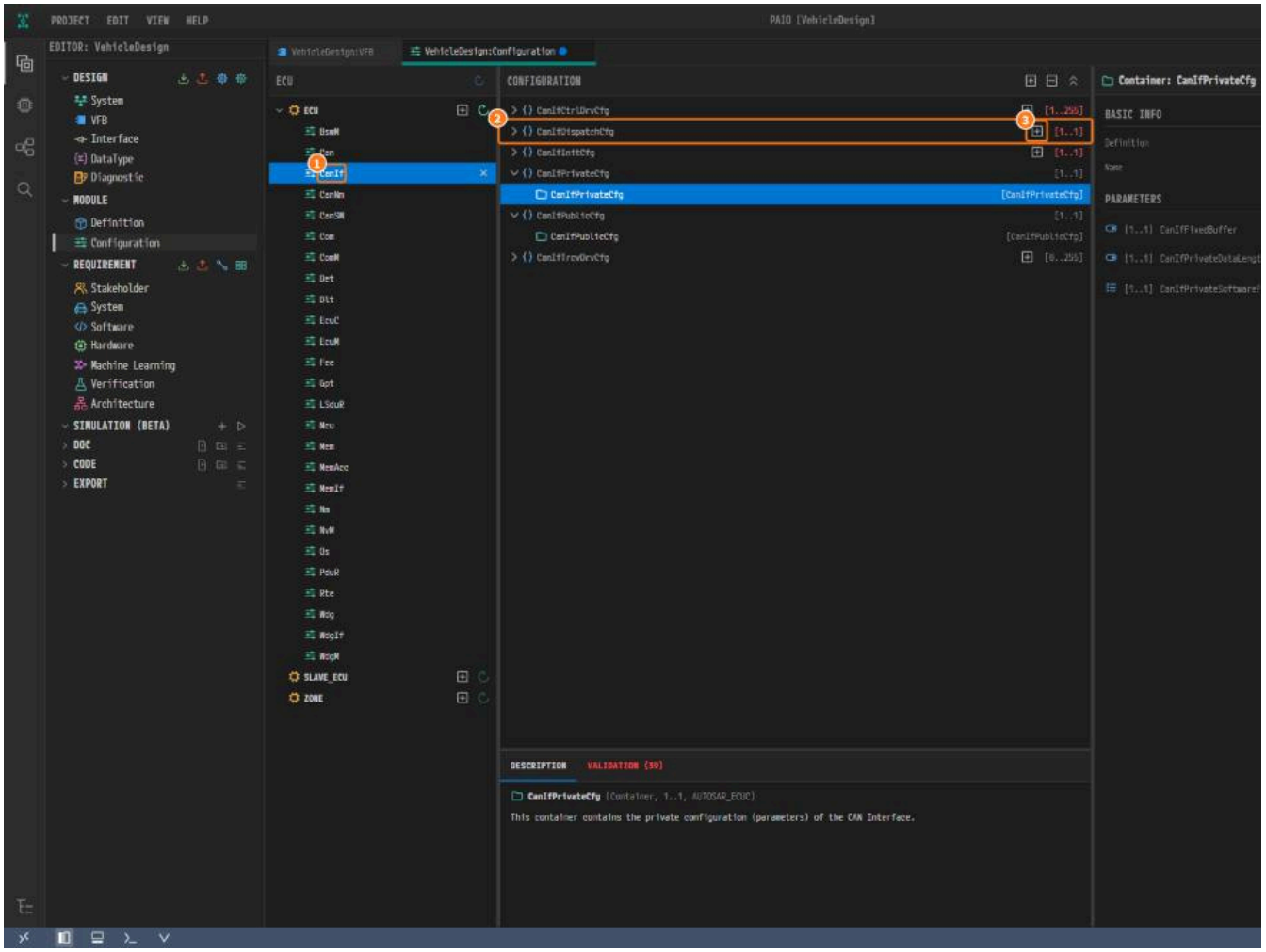


그림 9-10. ① `CanIf`, ② `CanIfDispatchCfg`, ③ +.

3. 두 목록에서 모두 `CAN_SM` 을 고른다. 그러면 컨트롤러의 버스 오프와 모드 보고를 `CanSM` 이 받는다.

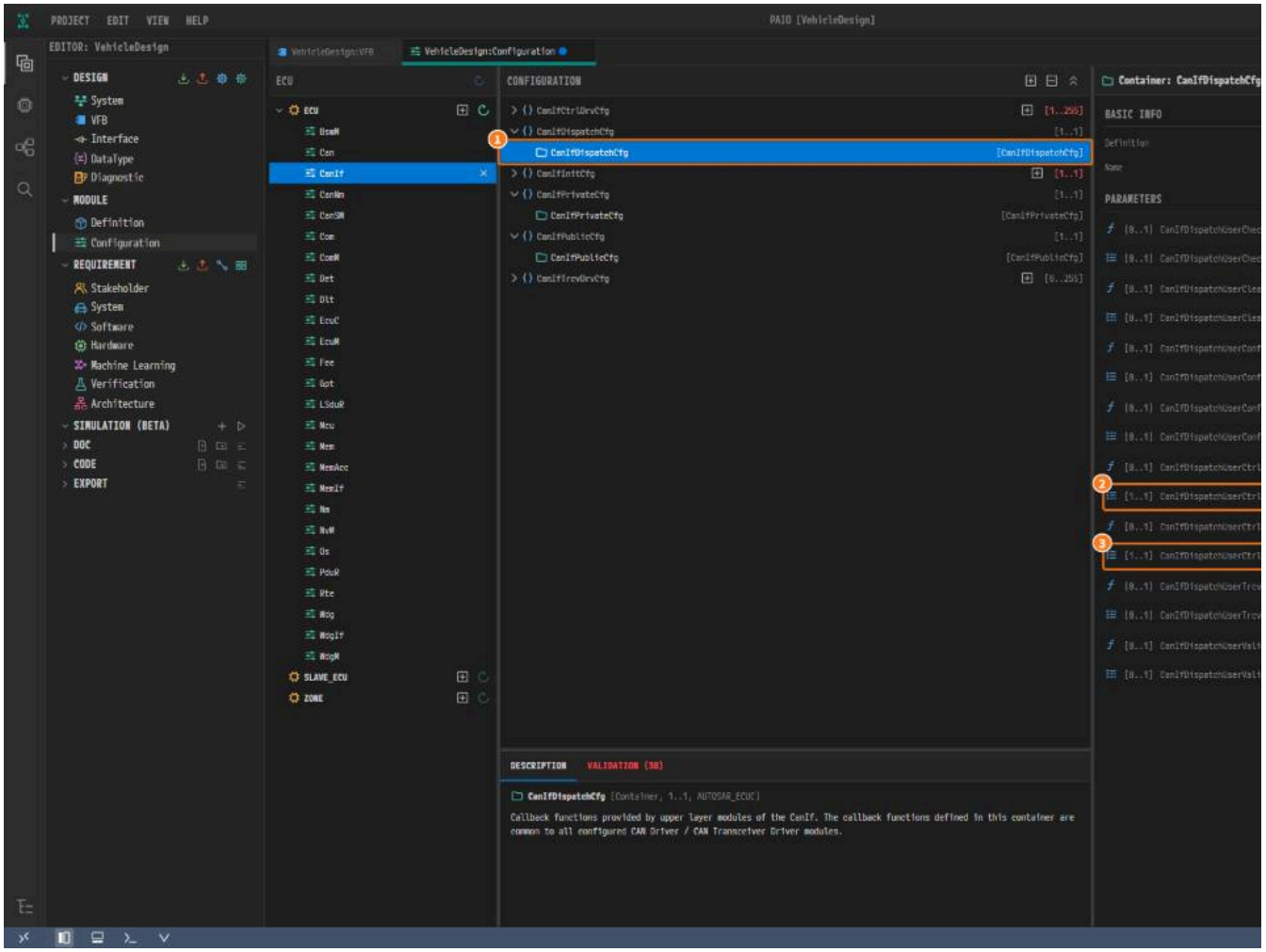


*그림 9-11. ① `CanIfDispatchCfg`, ② **CanIfDispatchUserCtrlBusOffUL**, ③ **CanIfDispatchUserCtrlModeIndicationUL**.*

4. `CanIfInitCfg` 를 추가하고 **CanIfInitCfgSet** 에 `CanIf_Config` 를 입력한 뒤, 그 아래에 `CanIfInitHohCfg_0` 을 추가한다.

5. `CanIfCtrlDrvCfg_0` 을 추가하고 표에 따라 참조 두 개를 고른다.

6. `CanIfCtrlDrvCfg_0` 아래에 `CanIfCtrlCfg_0` 을 추가하고, **CanIfCtrlId** 에 `0` 을 입력한 뒤 **CanIfCtrlCanCtrlRef** 에서 `Can/CanConfigSet/CanController_0` 을

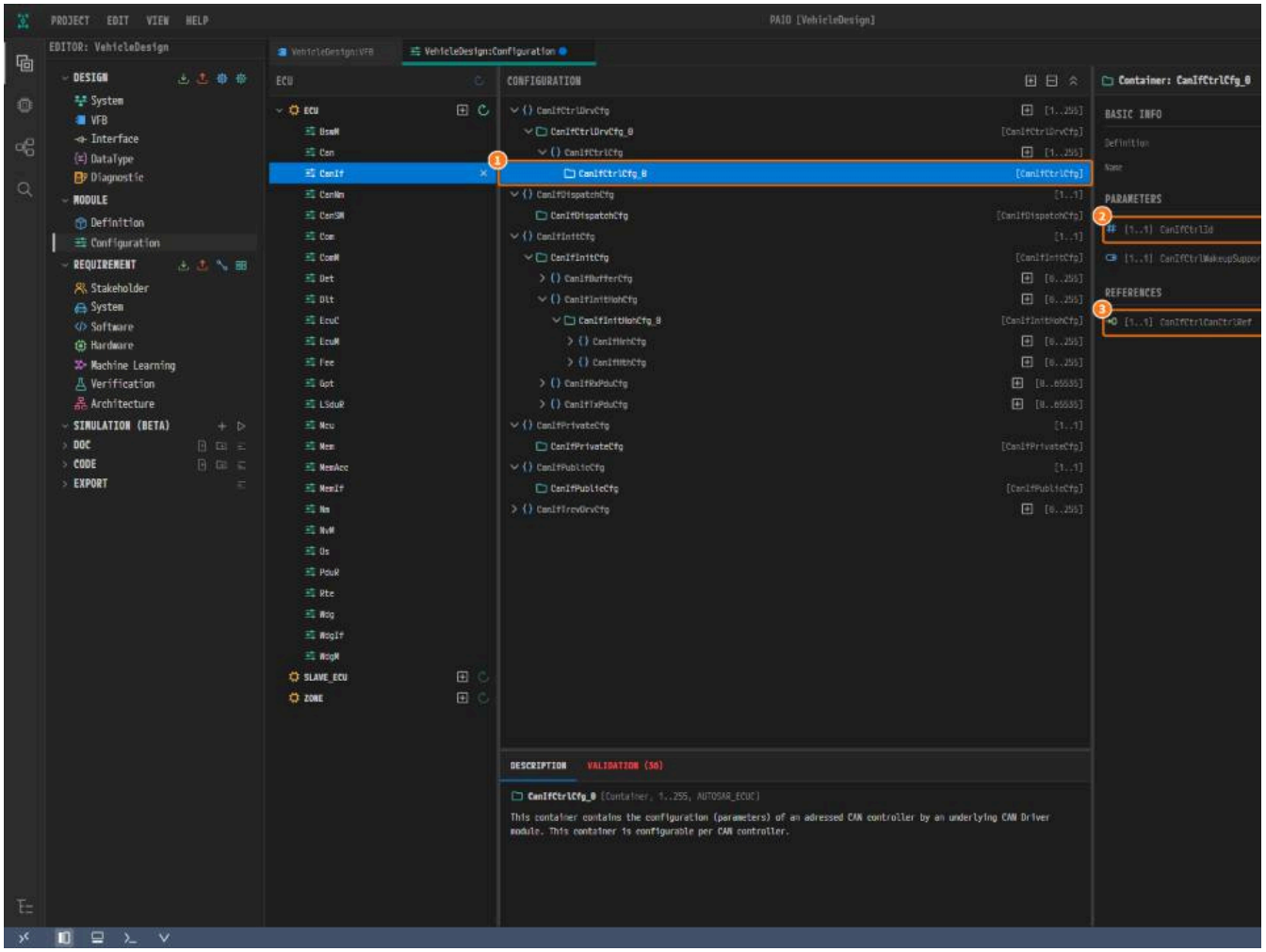


*그림 9-12. ① `CanIfCtrlCfg_0`, ② **CanIfCtrlId**, ③ **CanIfCtrlCanCtrlRef**.*

7. `CanIfHthCfg_0`, `CanIfHrhCfg_0`, `CanIfBufferCfg_0` 을 표에 따라 같은 방법으로 만든다. 송신 핸들은 `CanHardwareObject_0` 을, 수신 핸들은 `CanHardwareOb`

8. `CanIfInitCfg` 아래 `CanIfTxPduCfg` 행의 **+** 를 누른다.

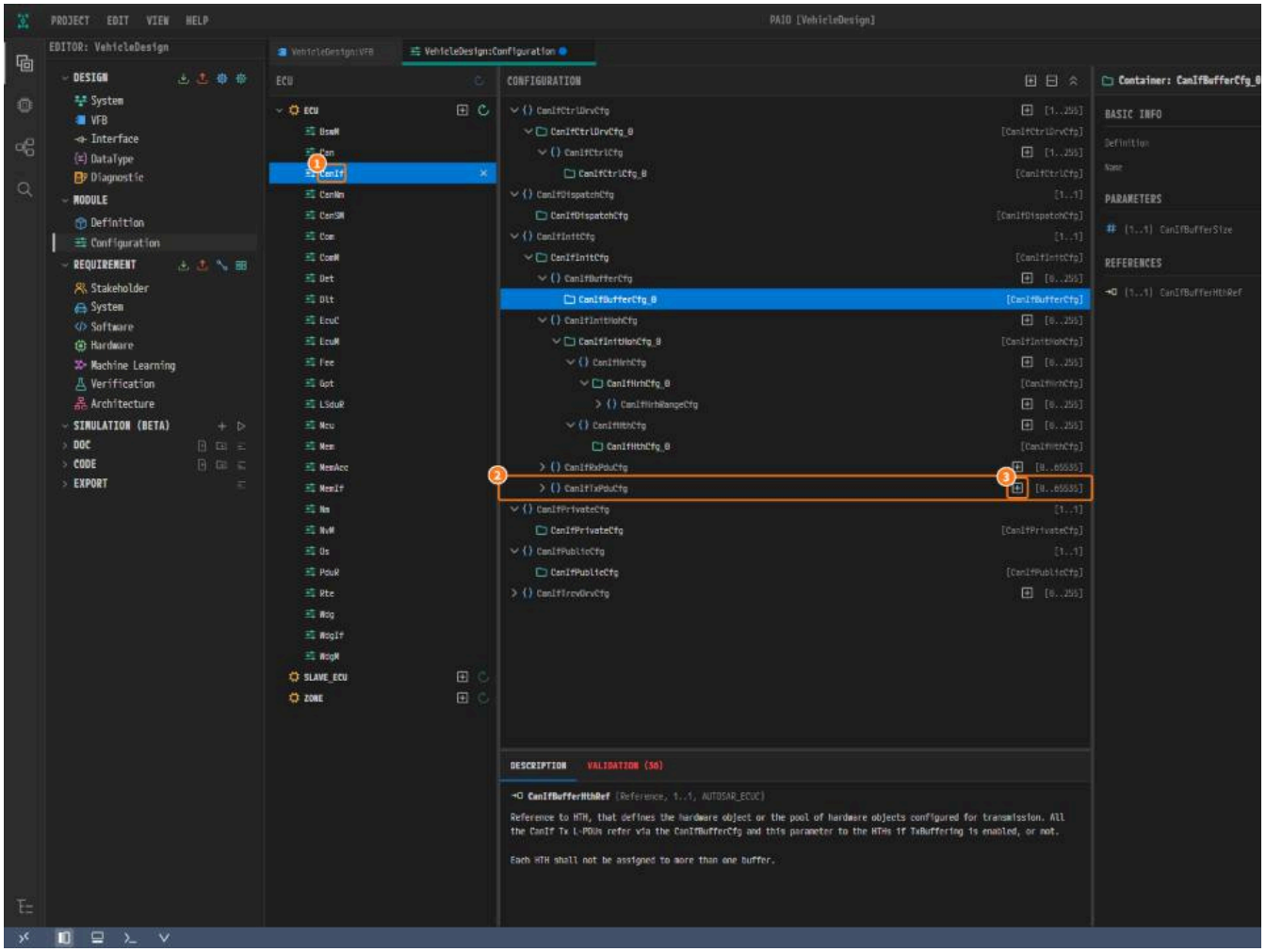


*그림 9-13. ① `CanIf`, ② `CanIfTxPduCfg`, ③ **+**.*

9. 표에 따라 `CanIfTxPduCfg_0` 의 값 다섯 개를 넣는다. `1281` 은 CAN id `0x501` 이다. **CanIfTxPduRef** 에는 `EcuC` 에서 만든 `PduNmTx` 를 고른다.

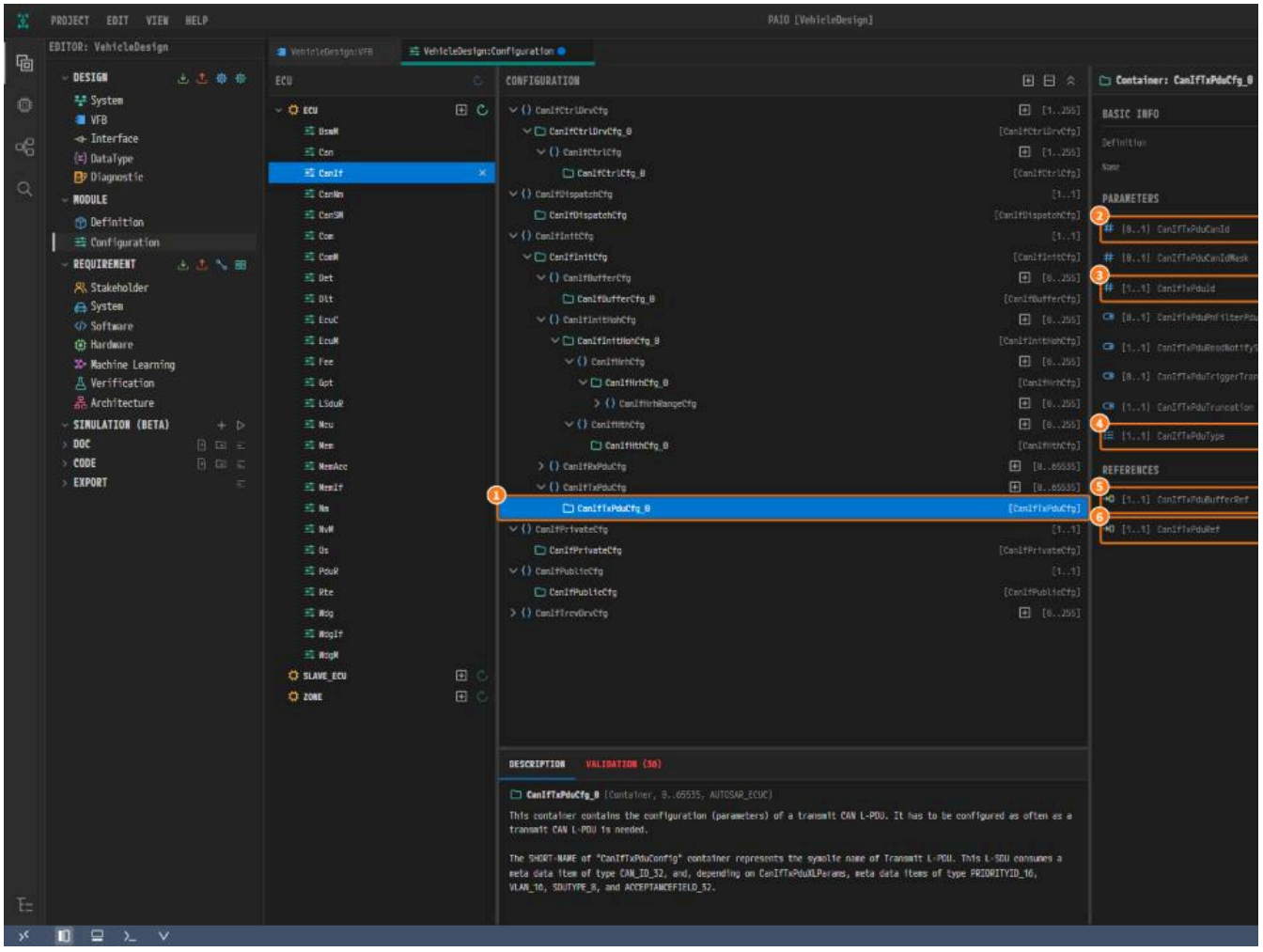


*그림 9-14. ① `CanIfTxPduCfg_0`, ② **CanIfTxPduCanId**, ③ **CanIfTxPduId**, ④ **CanIfTxPduType**, ⑤ **CanIfTxPduBufferRef**, ⑥ **CanIfTxPduRef**.*

10. `CanIfRxPduCfg_0` 도 표에 따라 같은 방법으로 만든다. `1282` 는 CAN id `0x502` 이며, **CanIfRxPduRef** 에는 `PduNmRx` 를 고른다. `Ctrl+S` 로 저장한다.

ComM 설정

컨테이너	추가하는 방법	항목	값
`ComMGeneral`	`ComMGeneral` 의 **+**		추가만 한다
`ComMConfigSet`	`ComMConfigSet` 의 **+**		추가만 한다
`ComMChannel_0`	`ComMConfigSet` 아래 `ComMChannel` 의 **+**	**ComMBusType**	`COMM_BUS_TYPE_CAN`
		ComMChannelId	`0`
		ComMNoWakeUpInhibitionNvmStorage	**FALSE**
`ComMNetworkManagement`	`ComMChannel_0` 아래 `ComMNetworkManagement` 의 **+**	**ComMNmVariant**	`FULL`

1. 모듈 목록에서 `ComM` 을 누른다. `ComMGeneral` 과 `ComMConfigSet` 을 **+** 로 추가한다. `ComMConfigSet` 아래 `ComMChannel` 행의 **+** 를 누른다.

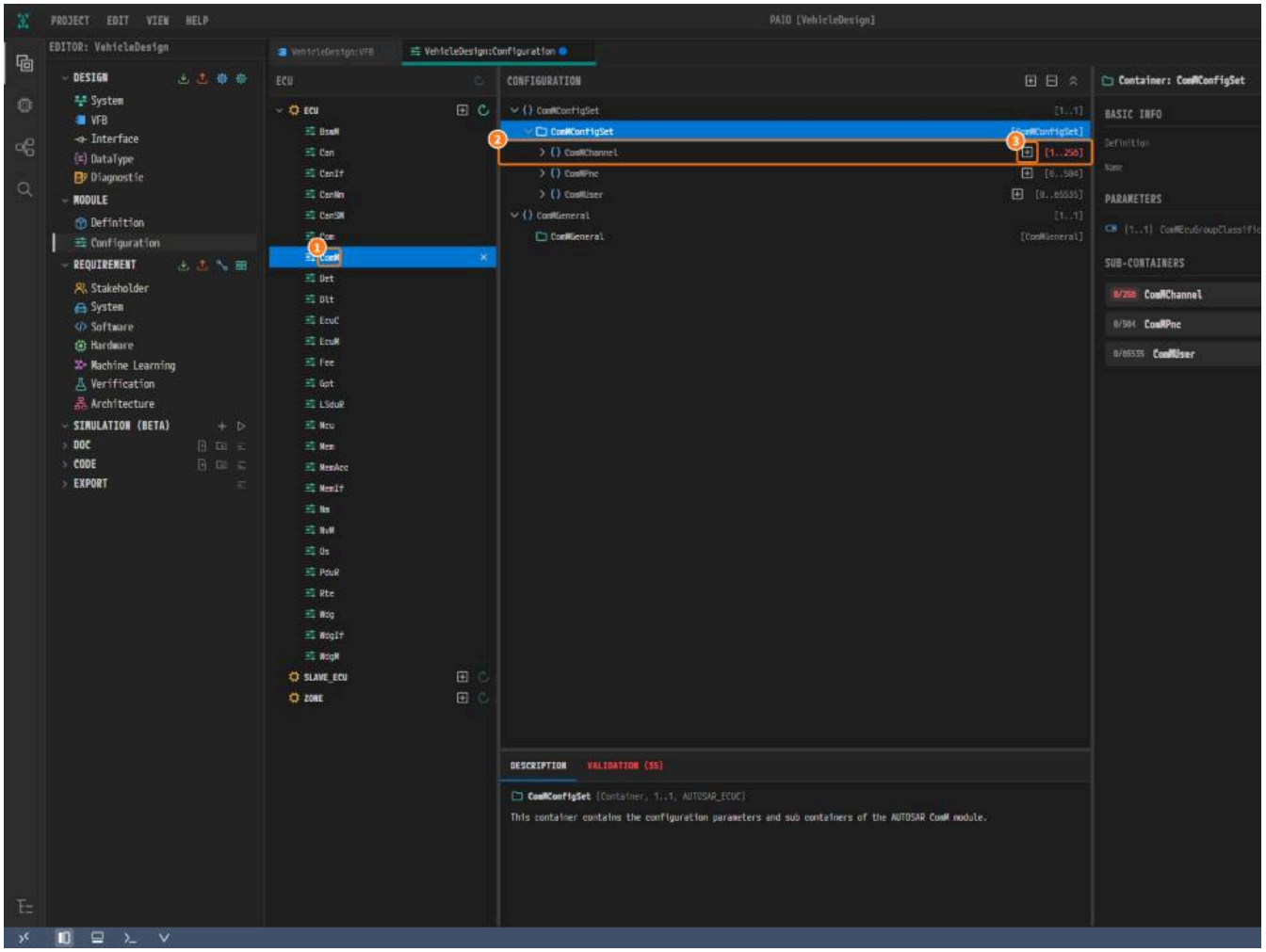


그림 9-15. ① `ComM`, ② `ComMChannel`, ③ +.

2. 표에 따라 `ComMChannel_0` 의 값 세 개를 넣는다.

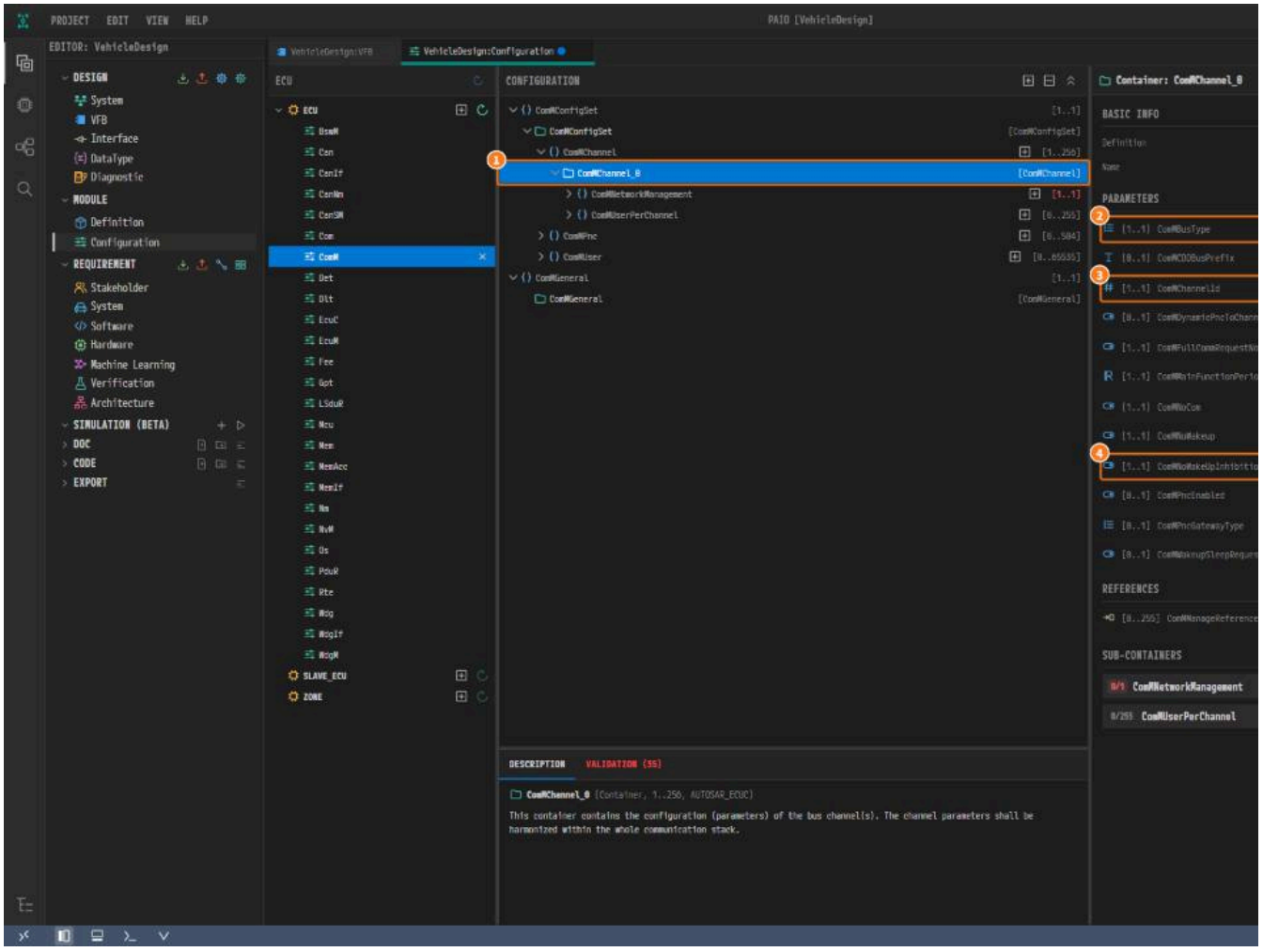


*그림 9-16. ① `ComMChannel_0`, ② **ComMBusType**, ③ **ComMChannelId**, ④ **ComMNoWakeUpInhibitionNvmStorage**.*

3. `ComMChannel_0` 아래에 `ComMNetworkManagement` 를 추가한다. 이 ECU 는 `CanNm` 을 돌리므로 **ComMNmVariant** 는 반드시 `FULL` 이어야 한다. + 가 이미 `FULL` 을 스

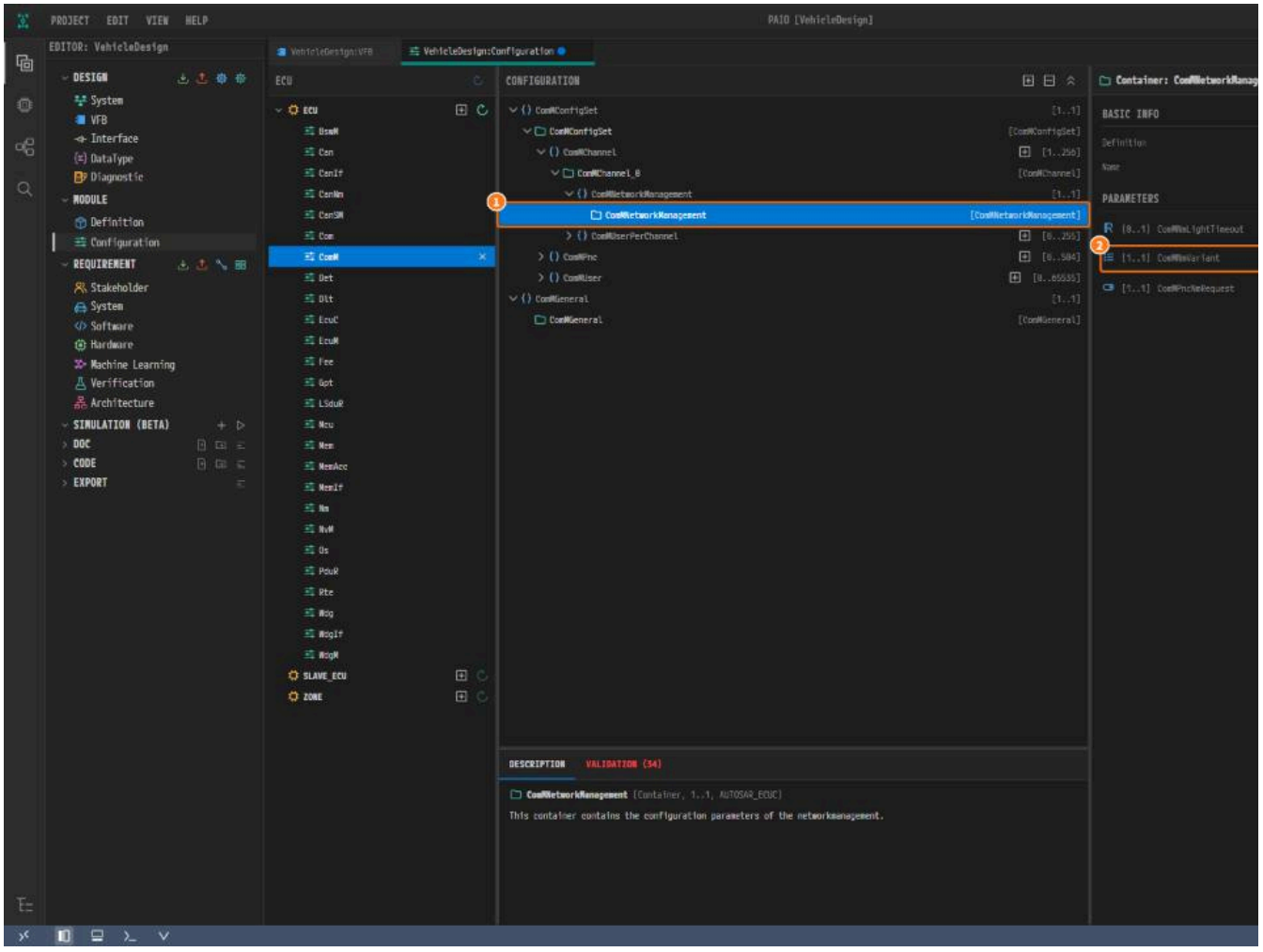


*그림 9-17. ① `ComMNetworkManagement`, ② **ComMNmVariant**.*

CanSM 설정

`CanSMManagerNetwork_0` 의 버스 오프 회복 값 네 개는 System 튜토리얼에서 `CanCluster` 에 넣은 값(L1 `100` ms, L2 `200` ms, 횟수 `2`, Tx ensured `100` ms)이다. 여기서

컨테이너	추가하는 방법	항목	값
`CanSMGeneral`	`CanSMGeneral` 의 +	**CanSMMainFunctionTimePeriod**	`0.01`
`CanSMConfiguration`	`CanSMConfiguration` 의 +	**CanSMModeRequestRepetitionMax**	`3`
		CanSMModeRequestRepetitionTime	`0.02`
`CanSMManagerNetwork_0`	`CanSMConfiguration` 아래 `CanSMManagerNetwork` 의 +	**CanSMBorCounterL1ToL2**	`2`
		CanSMBorTimeL1	`0.1`
		CanSMBorTimeL2	`0.2`
		CanSMBorTimeTxEnsured	`0.1`
		CanSMBorTxConfirmationPolling	**TRUE**
		CanSMComMNetworkHandleRef	`ComM/ComMConfigSet/ComMChannel_0`
`CanSMController_0`	`CanSMManagerNetwork_0` 아래 `CanSMController` 의 +	**CanSMControllerId**	`CanIf/CanIfCtrlDrvCfg_0/CanIfCtrlCfg_0`

1. 모듈 목록에서 `CanSM` 을 누른다. `CanSMGeneral` 과 `CanSMConfiguration` 을 + 로 추가하고 표에 따라 값을 넣는다. `CanSMConfiguration` 아래 `CanSMManagerNe`

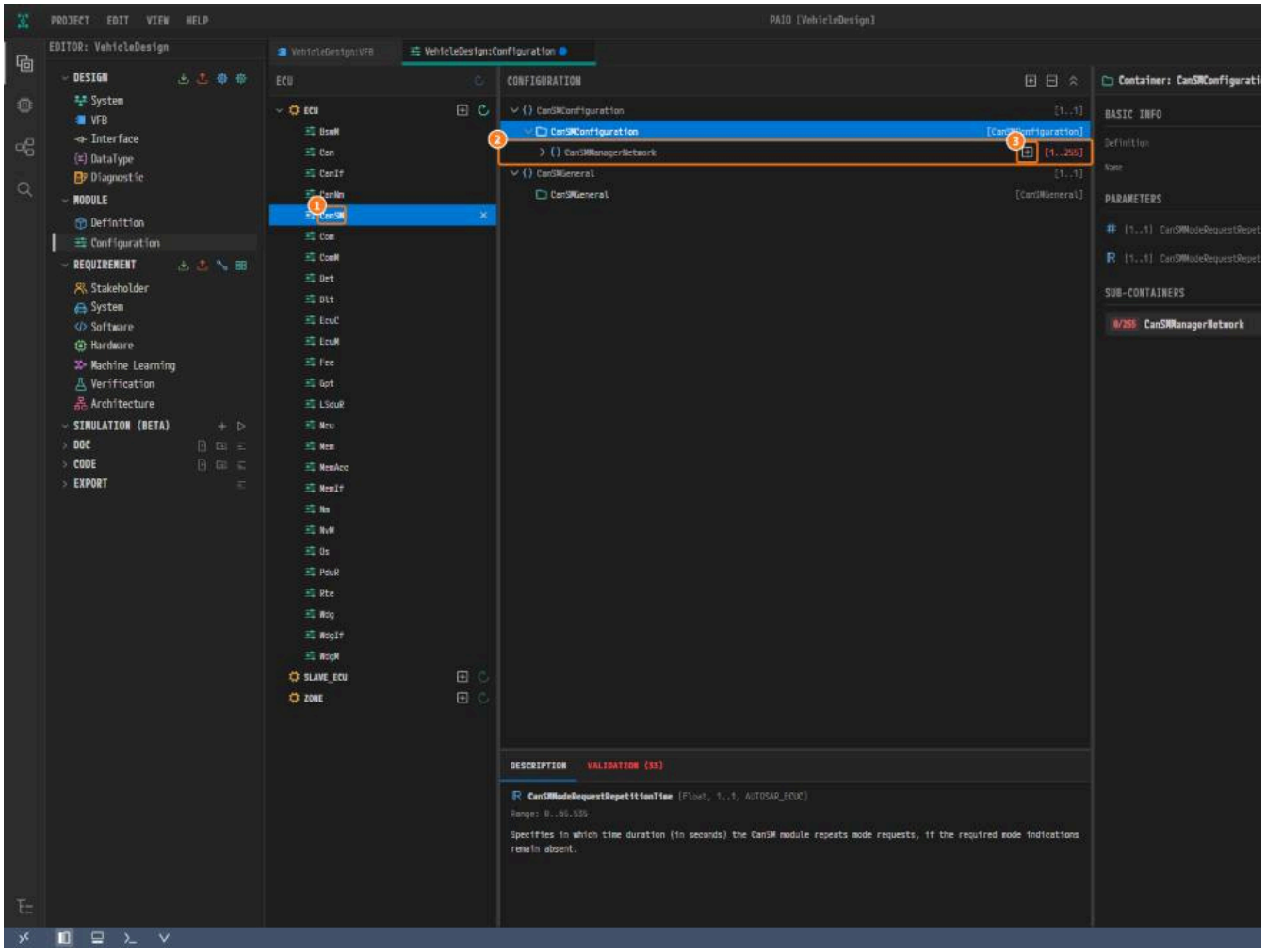


그림 9-18. ① `CanSM`, ② `CanSMManagerNetwork`, ③ +.

2. 표에 따라 `CanSMManagerNetwork_0` 의 값 여섯 개를 넣는다. **CanSMComMNetworkHandleRef** 에는 `ComM` 에서 만든 채널을 고른다.

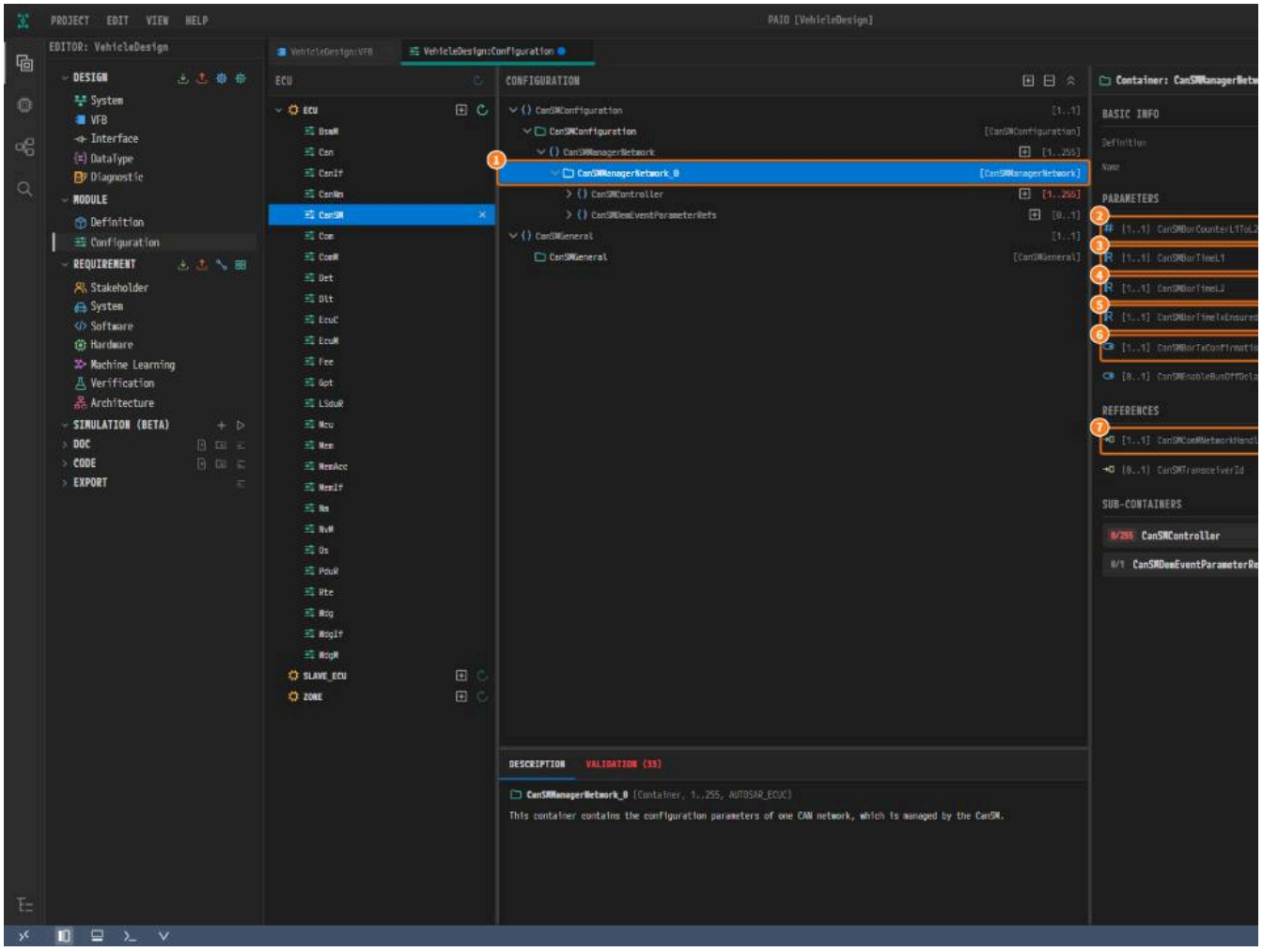


*그림 9-19. ① `CanSMManagerNetwork_0`, ② **CanSMBorCounterL1ToL2**, ③ **CanSMBorTimeL1**, ④ **CanSMBorTimeL2**, ⑤ **CanSMBorTimeTxEnsured**, ⑥ **CanSM**
CanSMComMNetworkHandleRef.*

3. `CanSMManagerNetwork_0` 아래에 `CanSMController_0` 을 추가하고 **CanSMControllerId** 에서 `CanIf/CanIfCtrlDrvCfg_0/CanIfCtrlCfg_0` 을 고른다. `Ctrl`

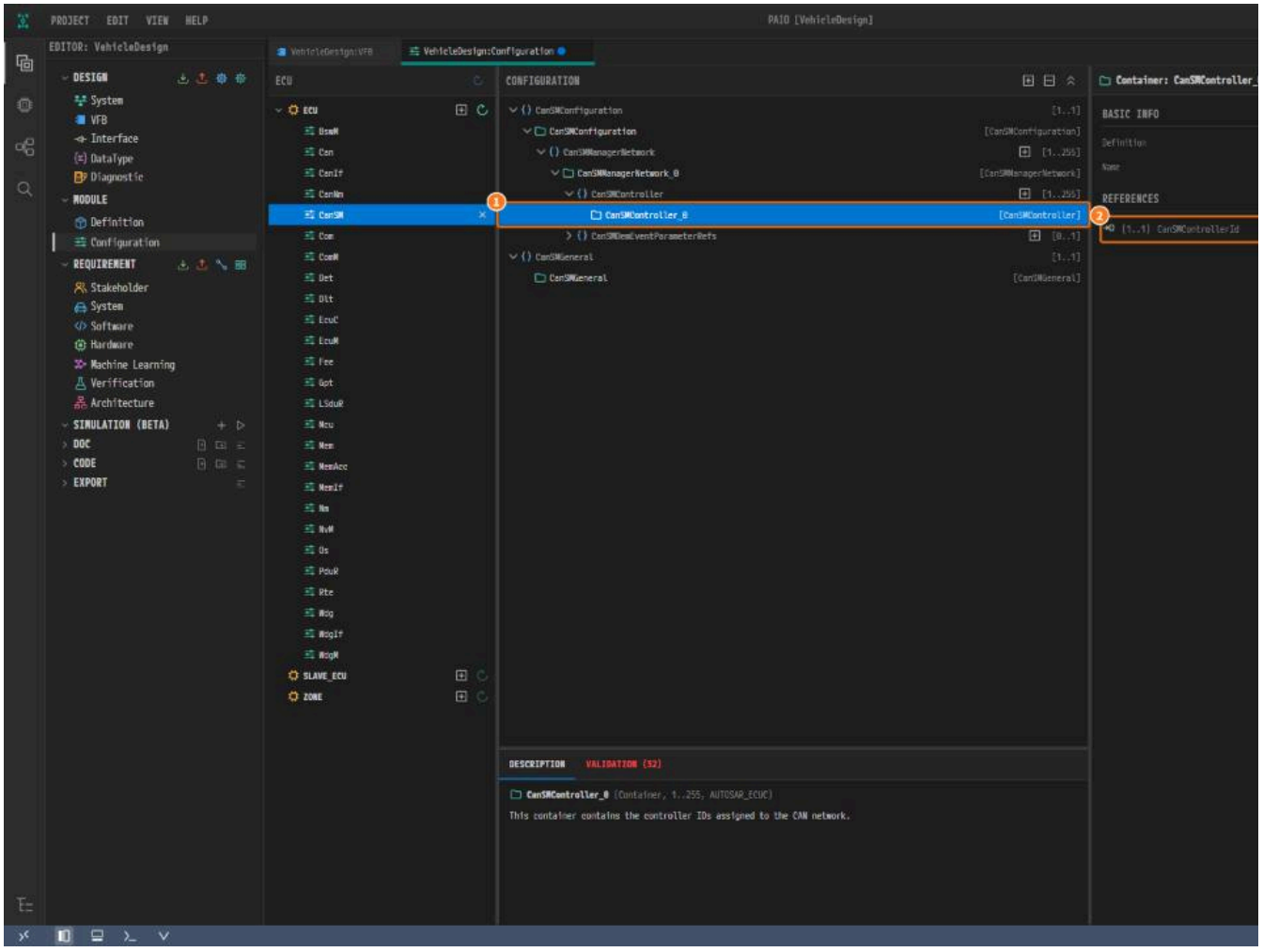


*그림 9-20. ① `CanSMController_0`, ② **CanSMControllerId**.*

Nm 설정

컨테이너	추가하는 방법	항목
`NmGlobalConfig`	`NmGlobalConfig` 의 +	
`NmGlobalConstants`	`NmGlobalConfig` 아래 `NmGlobalConstants` 의 +	**NmNumberOfChannels**
`NmGlobalFeatures`	`NmGlobalConfig` 아래 `NmGlobalFeatures` 의 +	**NmBusSynchronizationEnabled**, **NmComControlEnabled**, **NmCoordinatorSyncSupport**, **NmDynamicPncToChannelMappingSupport**, **NmPduRxIndicationEnabled**, **NmRemoteSleepIndEnabled**, **NmStateChangeIndEnabled**, **NmUserDataEnabled**
`NmGlobalProperties`	`NmGlobalConfig` 아래 `NmGlobalProperties` 의 +	**NmCycletimeMainFunction**
`NmChannelConfig_0`	`NmChannelConfig` 의 +	**NmChannelSleepMaster**, **NmPassiveModeEnabled**, **NmStateReportEnabled**, **NmSynchronizingNetwork**
		NmComMChannelRef
`NmBusType`	`NmChannelConfig_0` 아래 `NmBusType` 의 +	
`NmStandardBusNmConfig`	`NmBusType` 아래 `NmStandardBusNmConfig` 의 +	**NmStandardBusType**

1. 모듈 목록에서 `Nm` 을 누른다. `NmGlobalConfig` 를 + 로 추가한다. 그 아래에 `NmGlobalConstants` 를 추가하고 **NmNumberOfChannels** 에 `1` 을 입력한다.

2. `NmGlobalConfig` 아래에 `NmGlobalFeatures` 를 추가하고 표의 항목 여덟 개마다 **FALSE** 를 누른다. 이 항목들에는 기본값이 없다.

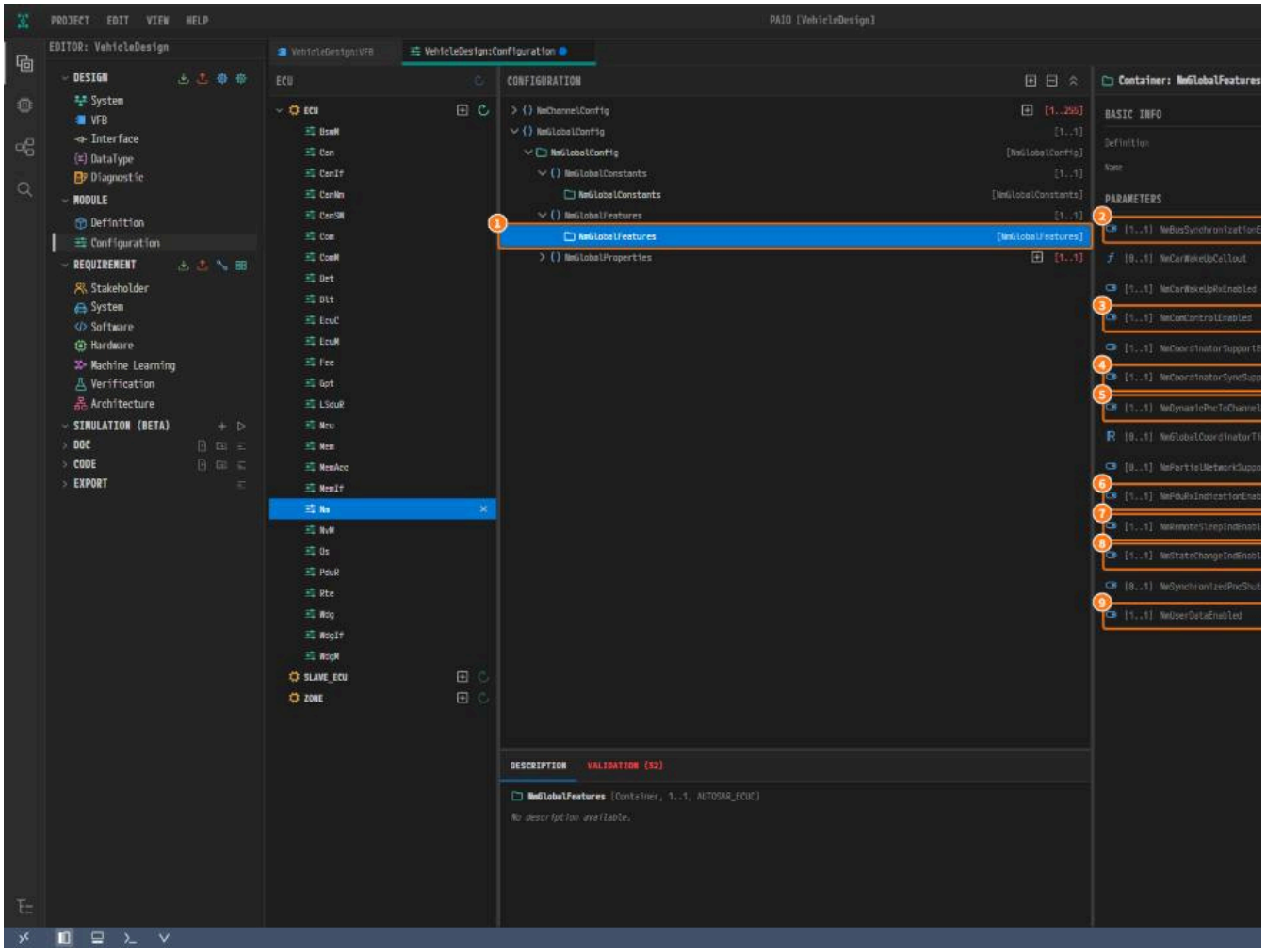


*그림 9-21. ① `NmGlobalFeatures`, ② **NmBusSynchronizationEnabled**, ③ **NmComControlEnabled**, ④ **NmCoordinatorSyncSupport**, ⑤ **NmDynamicPncToC** **NmPduRxIndicationEnabled**, ⑦ **NmRemoteSleepIndEnabled**, ⑧ **NmStateChangeIndEnabled**, ⑨ **NmUserDataEnabled**.*

3. `NmGlobalConfig` 아래에 `NmGlobalProperties` 를 추가하고 **NmCycletimeMainFunction** 에 `0.01` 을 입력한다. 이어서 트리 맨 위 `NmChannelConfig` 행의 **+** 를 누

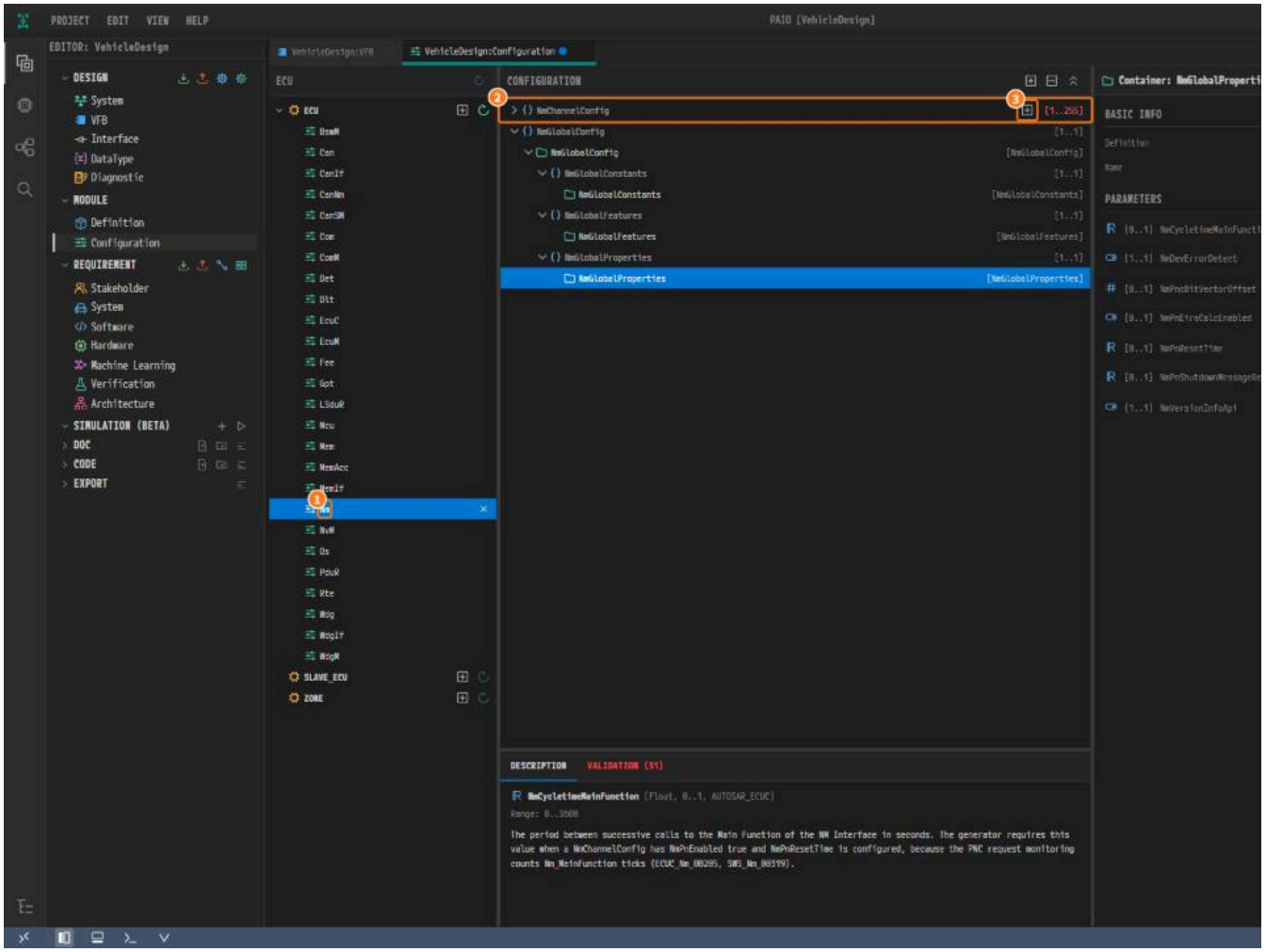


*그림 9-22. ① `Nm`, ② `NmChannelConfig`, ③ **+**.*

4. `NmChannelConfig_0` 의 항목 네 개마다 **FALSE** 를 누르고 **NmComMChannelRef** 에서 `ComM/ComMConfigSet/ComMChannel_0` 을 고른다.

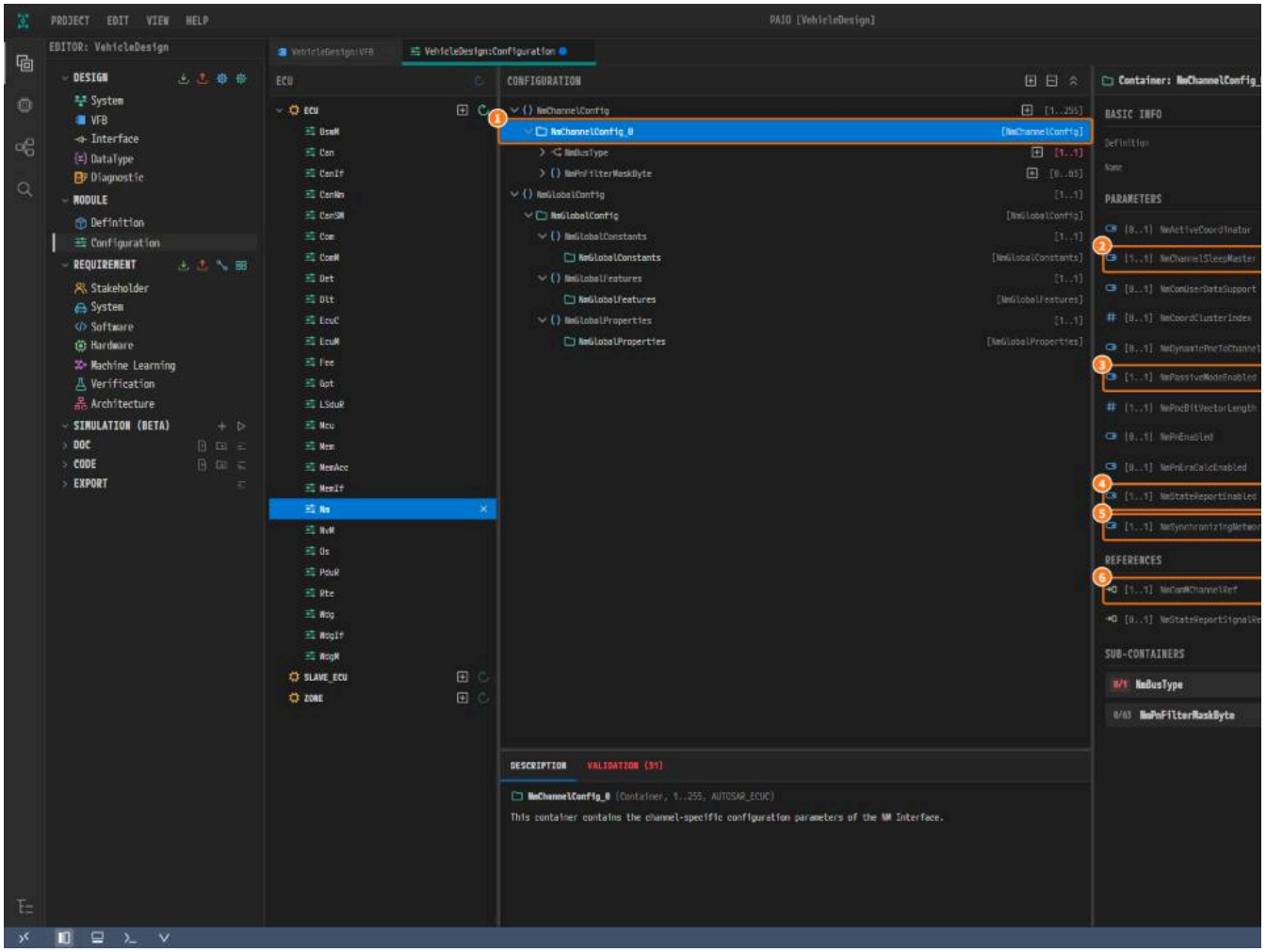


*그림 9-23. ① `NmChannelConfig_0`, ② **NmChannelSleepMaster**, ③ **NmPassiveModeEnabled**, ④ **NmStateReportEnabled**, ⑤ **NmSynchronizingNetwork**, ⑥*

5. `NmChannelConfig_0` 아래에 `NmBusType` 을, 그 아래에 `NmStandardBusNmConfig` 를 추가한다. **NmStandardBusType** 에서 `NM_BUSNM_CANNM` 을 고른다. `Ctrl+S`

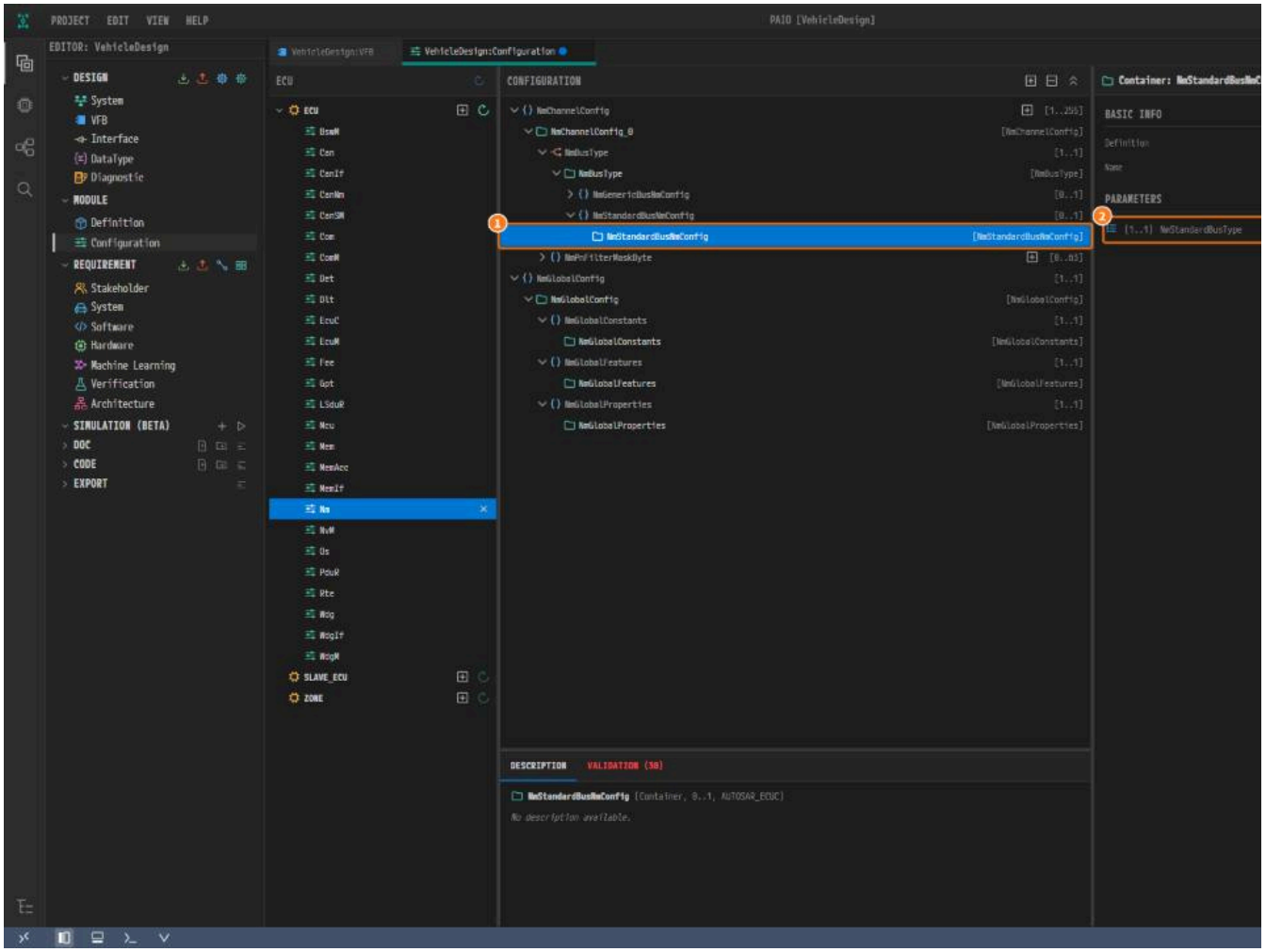


*그림 9-24. ① `NmStandardBusNmConfig`, ② **NmStandardBusType**.*

CanNm 설정

컨테이너	추가하는 방법	항목
`CanNmGlobalConfig`	`CanNmGlobalConfig` 의 +	**CanNmBusLoadReductionEnabled, CanNmBusSynchronizationEnabled, CanNmComControlEnabled, CanNmComUserDataSupport, CanNmCoordinatorSyncSupport, CanNmDynamicPncToChannelMappingSupport, CanNmImmediateRestartEnabled, CanNmImmediateTxconfEnabled, CanNmPassiveModeEnabled, CanNmPduRxIndicationEnabled, CanNmRemoteSleepIndEnabled, CanNmStateChangeIndEnabled, CanNmUserDataEnabled**
		CanNmMainFunctionPeriod
`CanNmChannelConfig_0`	`CanNmGlobalConfig` 아래 `CanNmChannelConfig` 의 +	**CanNmMsgCycleTime**
		CanNmTimeoutTime
		CanNmRepeatMessageTime
		CanNmWaitBusSleepTime
		CanNmRemoteSleepIndTime
		CanNmImmediateNmCycleTime
		CanNmImmediateNmTransmissions
		CanNmMsgCycleOffset
		CanNmMsgReducedTime
		CanNmNodeId
		CanNmNodeIdEnabled
		CanNmPduCbvPosition
		CanNmPduNidPosition
		CanNmBusLoadReductionActive, CanNmNodeDetectionEnabled, CanNmRepeatMsgIndEnabled
		CanNmCarWakeUpFilterEnabled
		CanNmComMNetworkHandleRef
`CanNmRxPdu_0`	`CanNmChannelConfig_0` 아래 `CanNmRxPdu` 의 +	**CanNmRxPduId**
		CanNmRxPduRef
`CanNmTxPdu`	`CanNmChannelConfig_0` 아래 `CanNmTxPdu` 의 +	**CanNmTxConfirmationPduId**
		CanNmTxPduRef

1. 모듈 목록에서 `CanNm` 을 누른다. `CanNmGlobalConfig` 를 + 로 추가한다. 기본값이 없는 표의 항목 열세 개마다 **FALSE** 를 누르고, **CanNmMainFunctionPeriod** 에 `0.01`

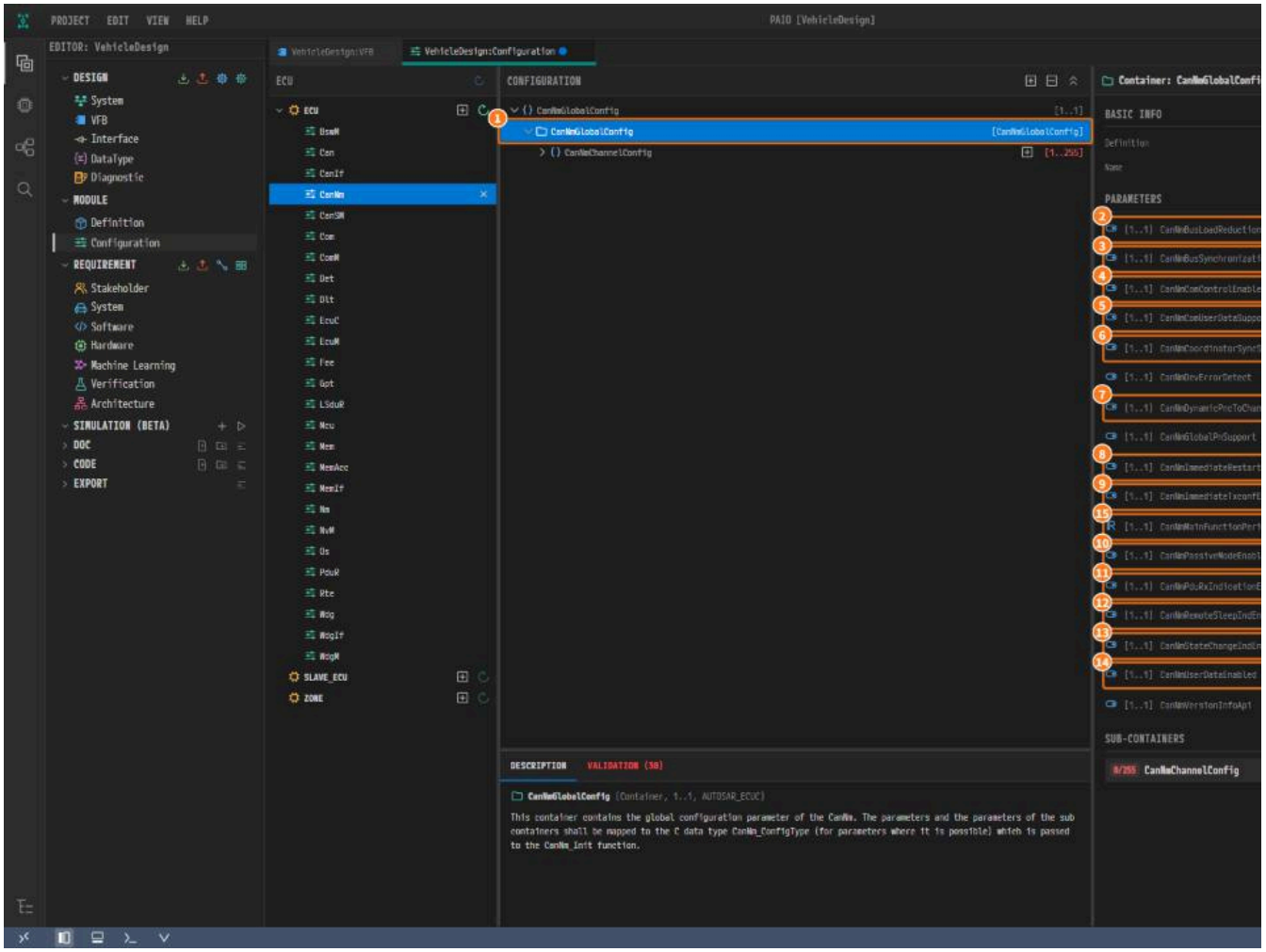


그림 9-25. ① `CanNmGlobalConfig`, ② CanNmBusLoadReductionEnabled, ③ CanNmBusSynchronizationEnabled, ④ CanNmComControlEnabled, ⑤ Cani CanNmCoordinatorSyncSupport, ⑦ CanNmDynamicPncToChannelMappingSupport, ⑧ CanNmImmediateRestartEnabled, ⑨ CanNmImmediateTxconfE CanNmPduRxIndicationEnabled, ⑫ CanNmRemoteSleepIndEnabled, ⑬ CanNmStateChangeIndEnabled, ⑭ CanNmUserDataEnabled, ⑮ CanNmMainFun

2. `CanNmGlobalConfig` 아래 `CanNmChannelConfig` 행의 **+** 를 누른다.

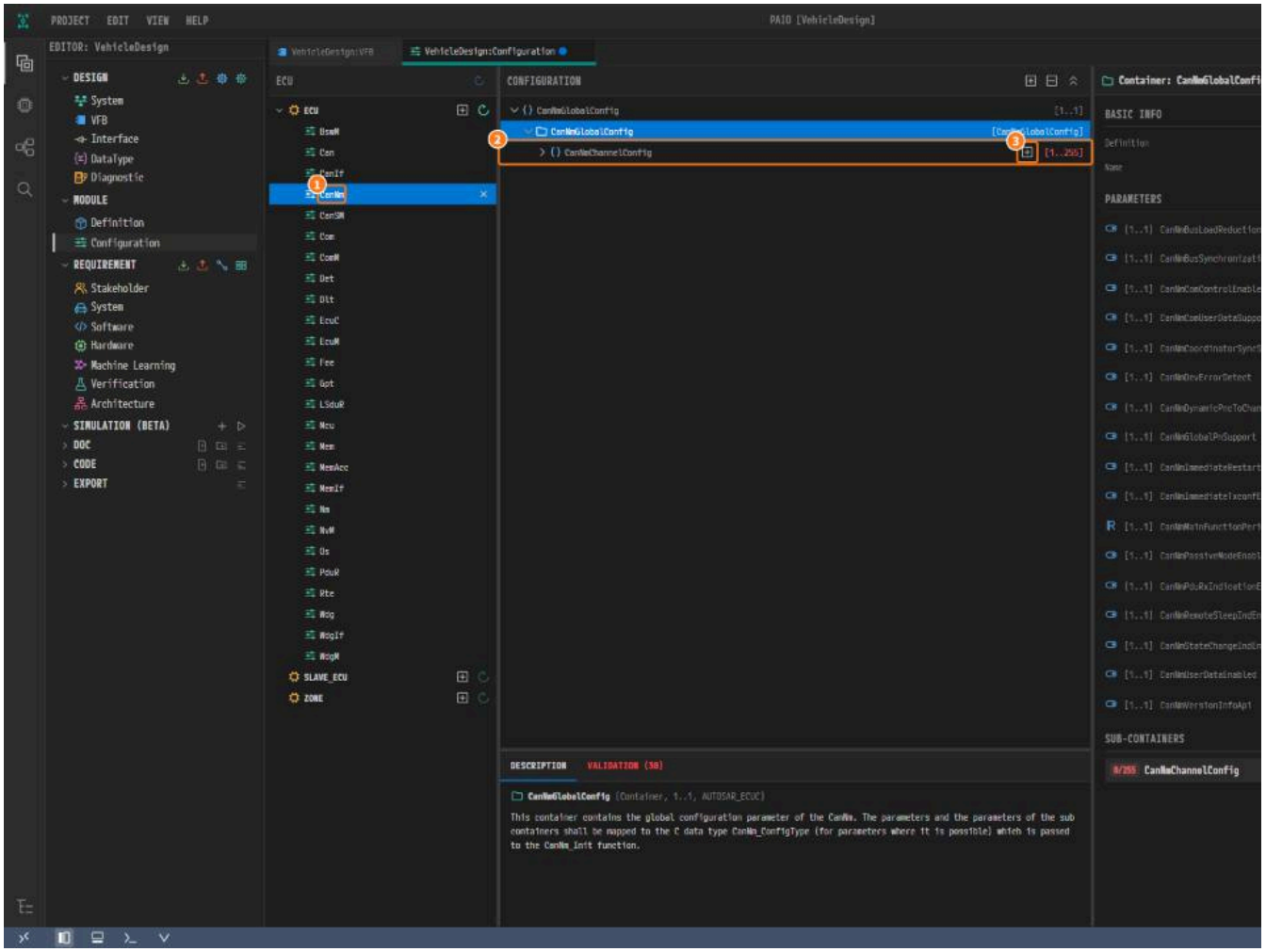


그림 9-26. ① `CanNm`, ② `CanNmChannelConfig`, ③ +.

3. 표에 따라 `CanNmChannelConfig_0` 의 값을 넣는다. 시간과 숫자를 입력하고, **CanNmNodeIdEnabled** 에 **TRUE** 를, 바이트 위치 두 개를 고르고, 기본값이 없는 항목 세 개에서 `ComM` 의 채널을 고른다.

4. **CanNmCarWakeUpFilterEnabled** 는 비어 있어야 한다. + 가 여기에 값을 쓰므로, 선택된 버튼을 한 번 더 눌러 지운다. 값이 남아 있으면 PRTE 가 `CANNM_CARWAKEUP_LAY`

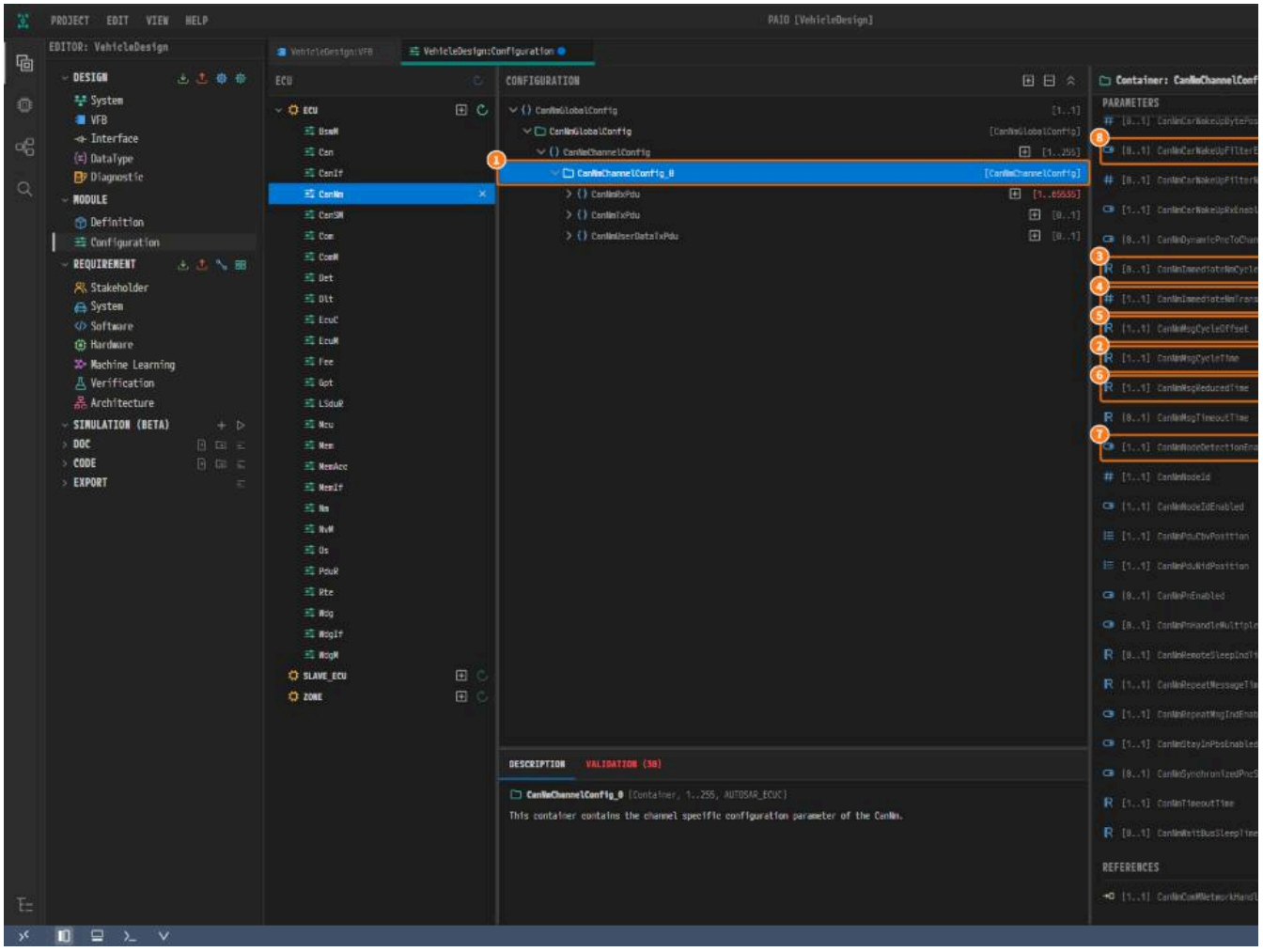


*그림 9-27. ① `CanNmChannelConfig_0`, ② **CanNmMsgCycleTime**, ③ **CanNmImmediateNmCycleTime**, ④ **CanNmImmediateNmTransmissions**, ⑤ **CanNmMs**
CanNmNodeDetectionEnabled, ⑧ 두 버튼 모두 선택되지 않은 **CanNmCarWakeUpFilterEnabled**.*

5. `CanNmChannelConfig_0` 아래에 `CanNmRxPdu_0` 을 추가하고, **CanNmRxPduId** 에 `0` 을 입력한 뒤 **CanNmRxPduRef** 에서 `PduNmRx` 를 고른다.

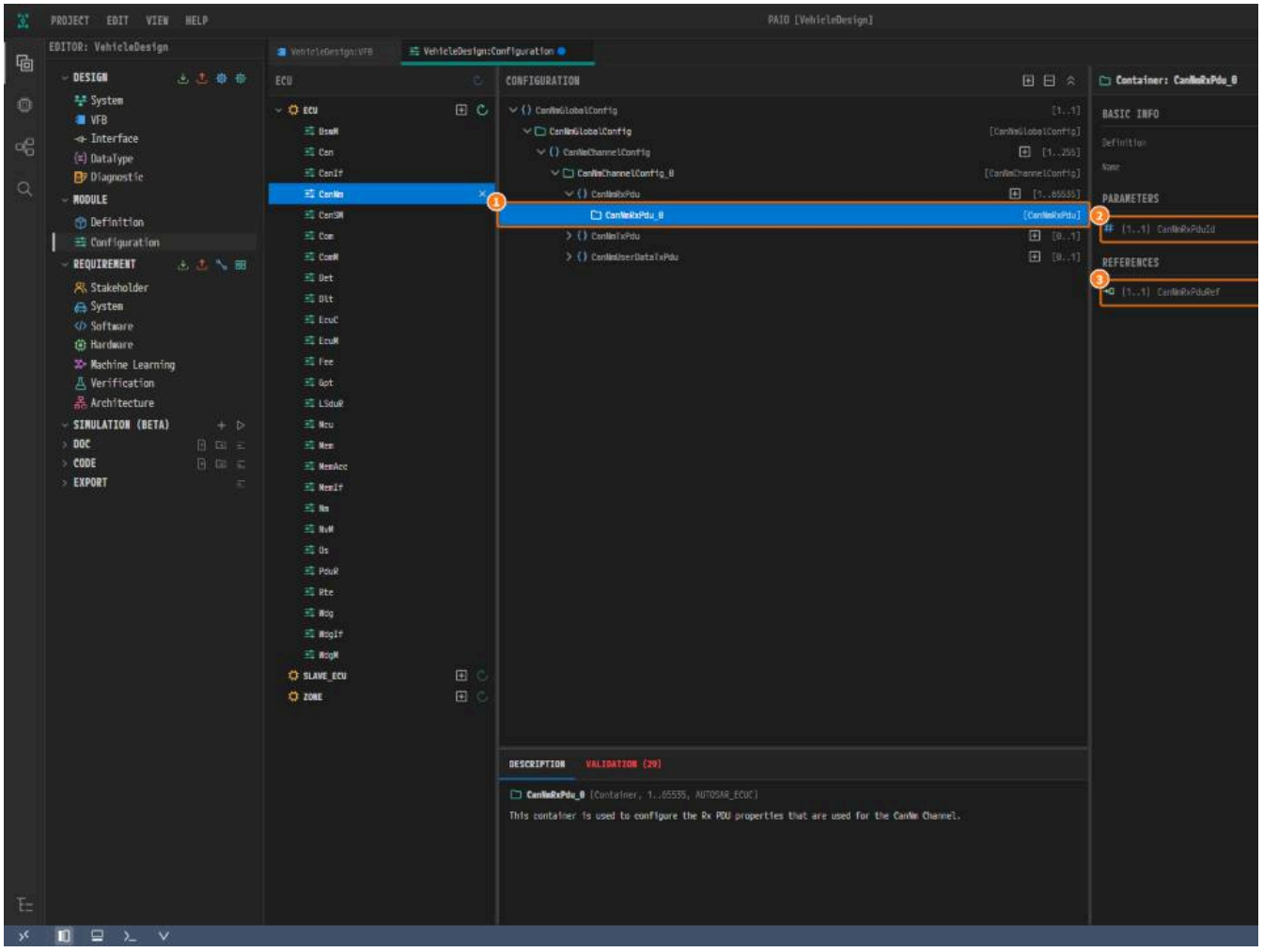


*그림 9-28. ① `CanNmRxPdu_0`, ② **CanNmRxPduId**, ③ **CanNmRxPduRef**.*

6. `CanNmTxPdu` 도 표에 따라 같은 방법으로 만들고, **CanNmTxPduRef** 에서 `PduNmTx` 를 고른다. `Ctrl+S` 로 저장한다.

LSduR 설정

`LSduR`, `PduR`, `Com` 에 넣는 값은 다음 표와 같다.

모듈	컨테이너	추가하는 방법	항목	값
LSduR	LSduRGeneral	LSduRGeneral 의 +	**LSduRMetaDataSupport**	**FALSE**
			LSduRZeroCostOperation	**FALSE**
LSduR	LSduRConfig	LSduRConfig 의 +		추가만 한다
LSduR	LSduRBswModules_0	LSduRBswModules 의 +	**LSduRRxIndication**	**TRUE**
			LSduRTransmit	**TRUE**
			LSduRTxConfirmation	**TRUE**
			LSduRProvideTxBuffer	**FALSE**
			LSduRReleaseRxBuffer	**FALSE**
			LSduRLowerModule	**TRUE**
			LSduRUpperModule	**FALSE**
			LSduRBswModuleRef	ECU/CanIf (목록에서 선택)
LSduR	LSduRBswModules_1	LSduRBswModules 의 +	**LSduRRxIndication**	**TRUE**
			LSduRTransmit	**TRUE**
			LSduRTxConfirmation	**TRUE**
			LSduRProvideTxBuffer	**FALSE**
			LSduRReleaseRxBuffer	**FALSE**
			LSduRLowerModule	**FALSE**
			LSduRUpperModule	**TRUE**
			LSduRBswModuleRef	ECU/CanNm (목록에서 선택)
LSduR	LSduRConfig/LSduRPath_0	LSduRPath 의 +		추가만 한다
LSduR	LSduRPath_0/LSduRPathSourcePdu	LSduRPathSourcePdu 의 +	**LSduRPathSourcePduId**	0
			LSduRPathSourcePduRef	EcuC/EcucConfigSet/EcucPduCollection/P
LSduR	LSduRPath_0/LSduRPathDestinationPdu_0	LSduRPathDestinationPdu 의 +	**LSduRPathDestinationPduRef**	EcuC/EcucConfigSet/EcucPduCollection/P
LSduR	LSduRConfig/LSduRPath_1	LSduRPath 의 +		추가만 한다
LSduR	LSduRPath_1/LSduRPathSourcePdu	LSduRPathSourcePdu 의 +	**LSduRPathSourcePduId**	1
			LSduRPathSourcePduRef	EcuC/EcucConfigSet/EcucPduCollection/P
LSduR	LSduRPath_1/LSduRPathDestinationPdu_0	LSduRPathDestinationPdu 의 +	**LSduRPathDestinationPduRef**	EcuC/EcucConfigSet/EcucPduCollection/P
PduR	PduRGeneral	PduRGeneral 의 +	**PduRZeroCostOperation**	**FALSE**
PduR	PduRRoutingPaths	PduRRoutingPaths 의 +	**PduRConfigurationId**	0
Com	ComGeneral	ComGeneral 의 +	**ComSupportedIPduGroups**	0
Com	ComConfig	ComConfig 의 +		추가만 한다
Com	ComConfig/ComMainFunctionRx	ComMainFunctionRx 의 +	**ComMainRxTimeBase**	0.01(채워져 있음)

모듈	컨테이너	추가하는 방법	항목	값
`Com`	`ComConfig/ComMainFunctionTx`	`ComMainFunctionTx` 의 **+**	**ComMainTxTimeBase**	`0.01`(채워져 있음)

> **주의.** 이미 선택된 **TRUE** 나 **FALSE** 버튼을 누르면 값이 지워진다. 그때는 같은 버튼을 한 번 더 누른다.

1. 모듈 목록에서 `LSduR` 을 누른다. **CONFIGURATION** 트리에서 `LSduRGeneral` 행의 **+** 를 누른다.

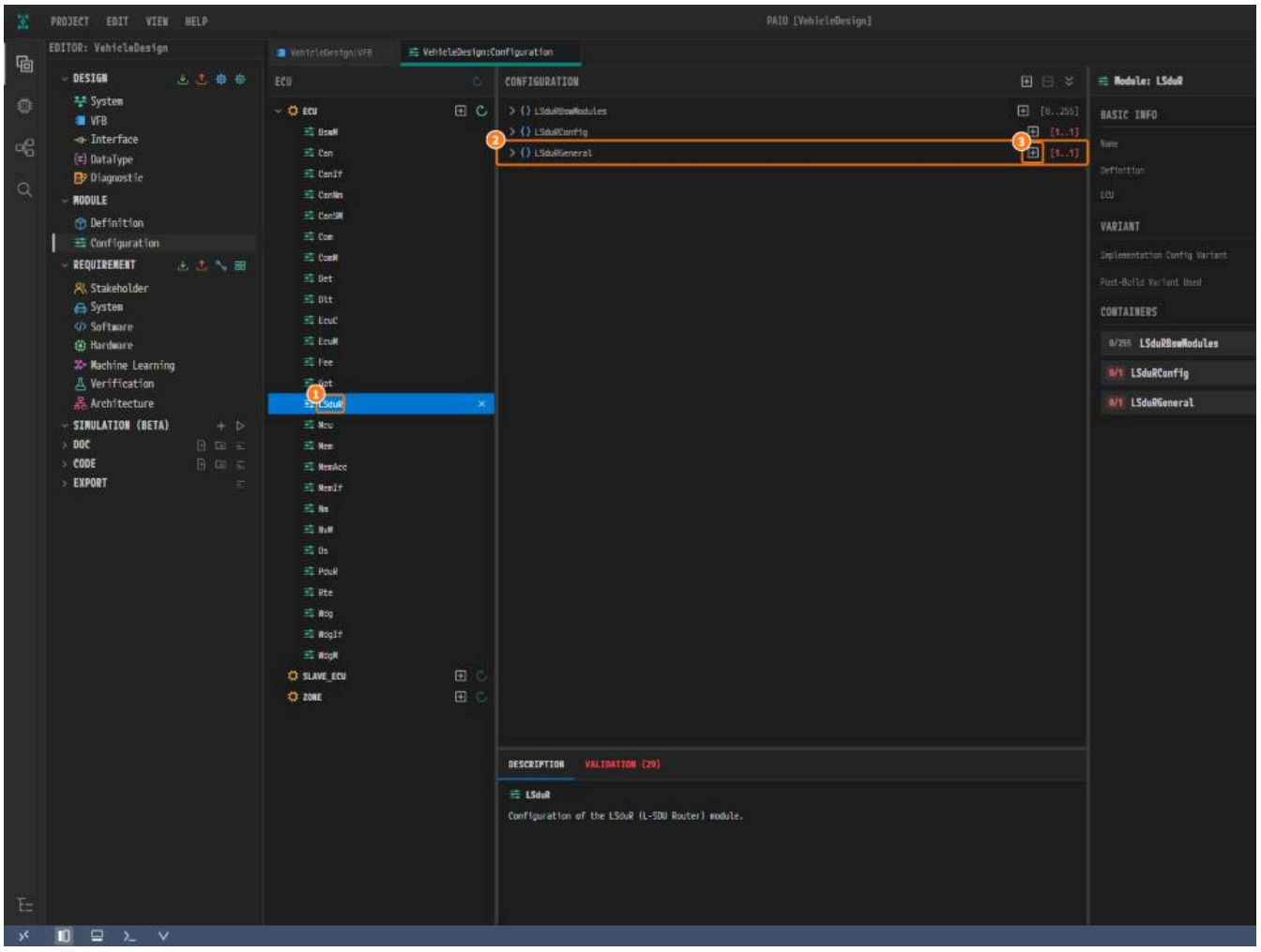


*그림 9-29. ① `LSduR`, ② `LSduRGeneral`, ③ **+**.*

2. `LSduRConfig` 행의 **+** 를 누른다.

3. `LSduRGeneral` 을 누른다. **LSduRMetaDataSupport** 와 **LSduRZeroCostOperation** 의 **FALSE** 를 누른다.

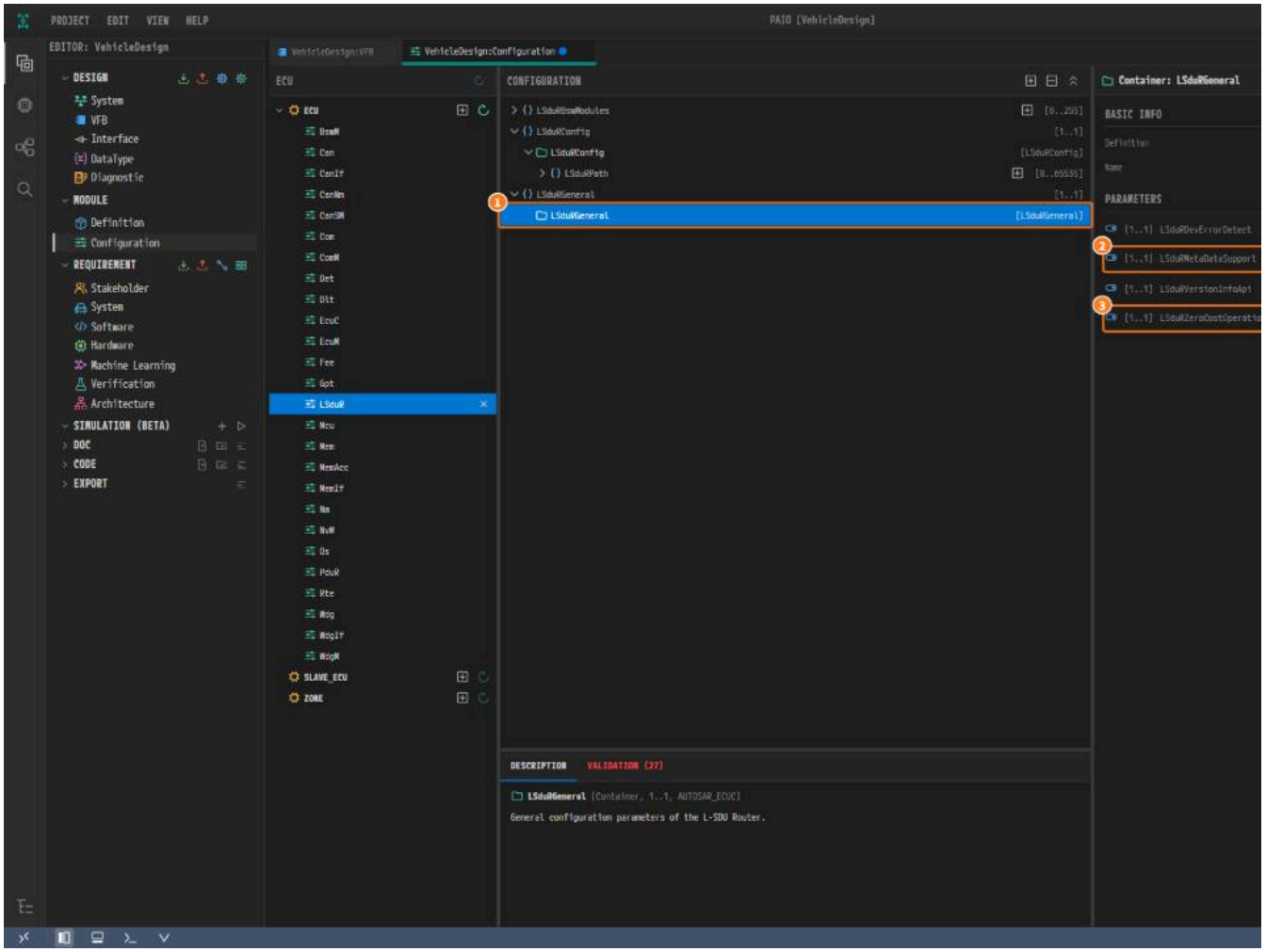


*그림 9-30. ① `LSduRGeneral`, ② **LSduRMetaDataSupport**, ③ **LSduRZeroCostOperation**.*

4. `LSduRBswModules` 행의 **+** 를 누른다. 에디터가 `LSduRBswModules_0` 을 추가하고 선택한다.

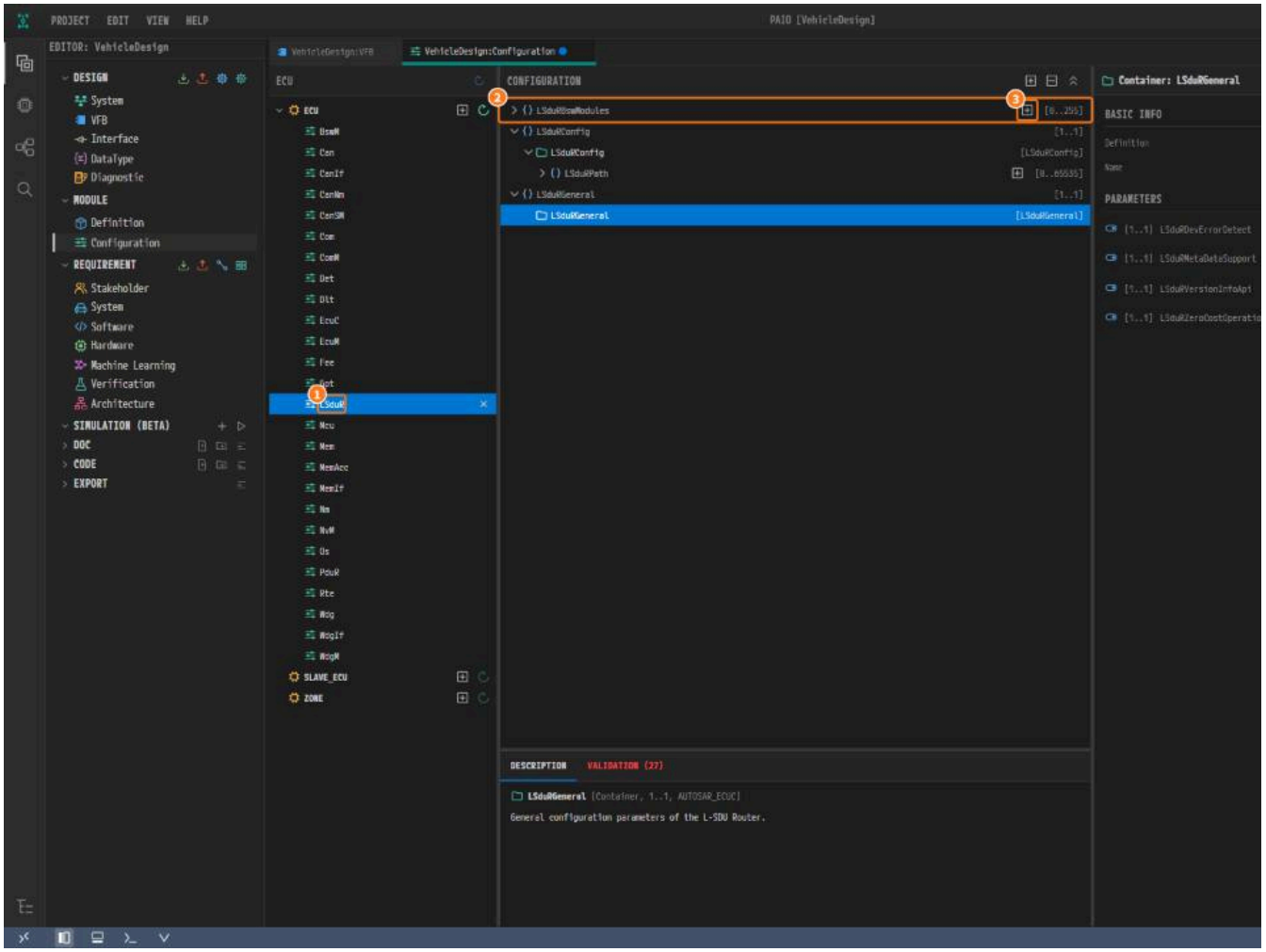


그림 9-31. ① `LSduR`, ② `LSduRBswModules`, ③ +.

5. `LSduRBswModules_0` 에서 **LSduRRxIndication**, **LSduRTransmit**, **LSduRTxConfirmation** 의 **TRUE**, **LSduRProvideTxBuffer** 와 **LSduRReleaseRxBuffer** 의 **FALS**
   LSduRUpperModule 의 **FALSE** 를 누른다.

6. **LSduRBswModuleRef** 드롭다운을 열고 `ECU/CanIf` 를 고른다. 반드시 목록에서 고른다. `CanIf` 를 입력하면 참조가 풀리지 않아 PRTE 가 `LSDUR_PATH_SOURCE_UNRES`

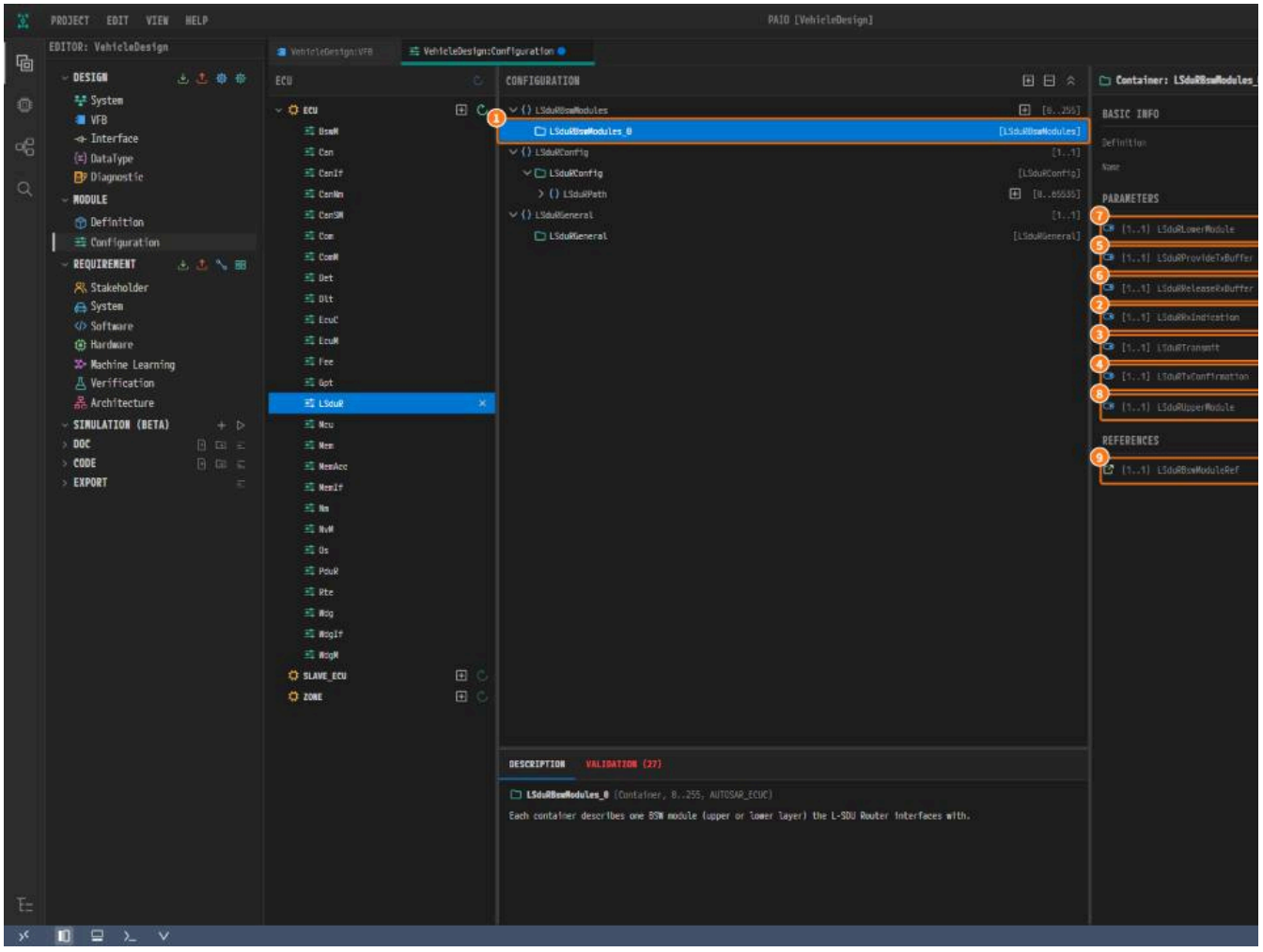


*그림 9-32. ① `LSduRBswModules_0`, ② **LSduRRxIndication**, ③ **LSduRTransmit**, ④ **LSduRTxConfirmation**, ⑤ **LSduRProvideTxBuffer**, ⑥ **LSduRReleaseRxBu**
LSduRUpperModule, ⑨ **LSduRBswModuleRef**.*

7. `LSduRBswModules` 행의 **+** 를 다시 누른다. `LSduRBswModules_1` 에 같은 값을 넣되 **LSduRLowerModule** 은 **FALSE**, **LSduRUpperModule** 은 **TRUE** 를 누르고, **LSdul**

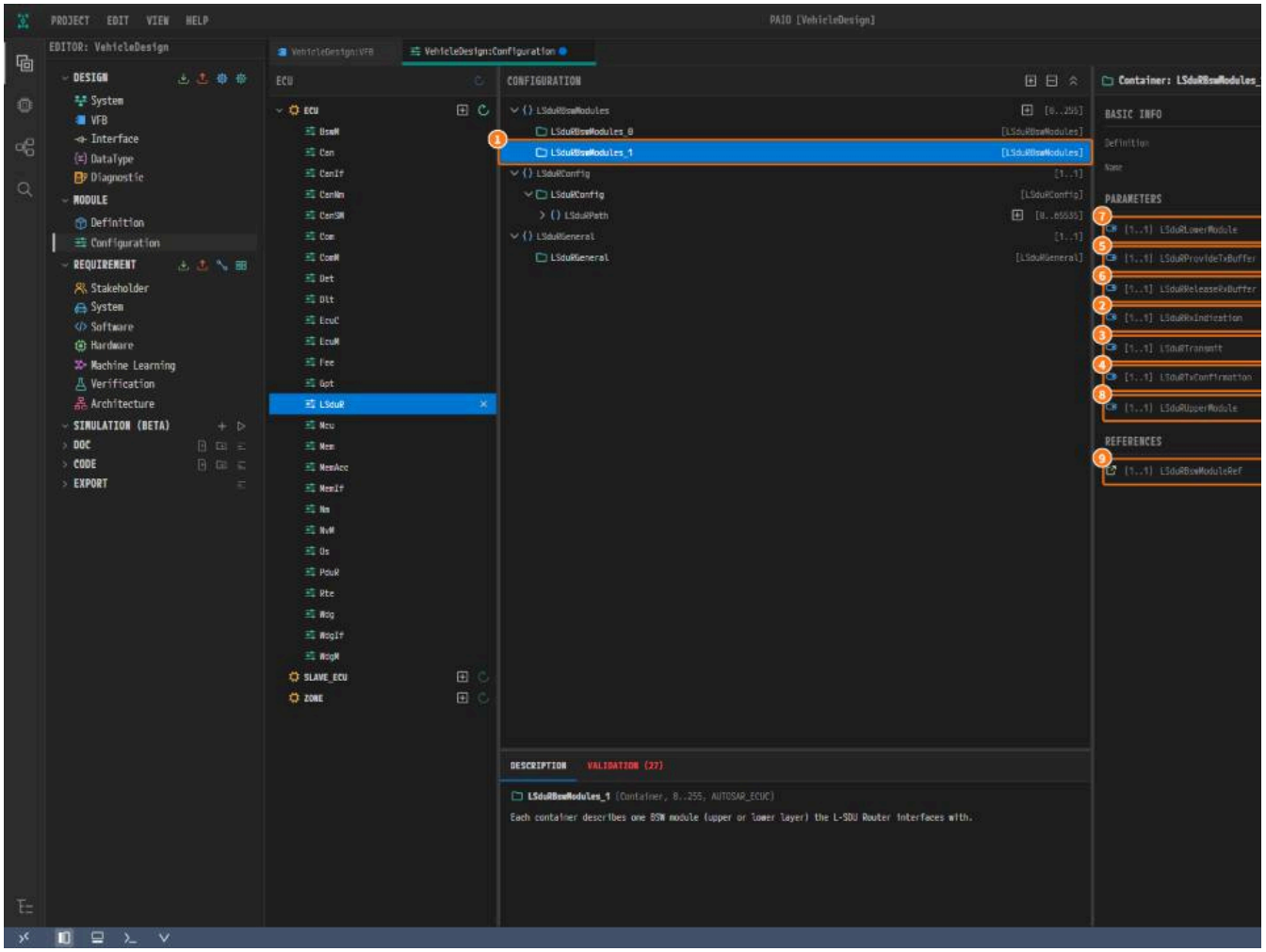


*그림 9-33. ① `LSduRBswModules_1`, ② **LSduRRxIndication**, ③ **LSduRTransmit**, ④ **LSduRTxConfirmation**, ⑤ **LSduRProvideTxBuffer**, ⑥ **LSduRReleaseRxBu**
LSduRUpperModule, ⑨ **LSduRBswModuleRef**.*

8. `LSduRConfig` 아래 `LSduRPath` 행의 **+** 를 누른다. 에디터가 `LSduRPath_0` 을 추가한다.

9. `LSduRPath_0` 아래 `LSduRPathSourcePdu` 행의 **+** 를 누른다. **LSduRPathSourcePduId** 에 `0` 을 입력하고 `Tab` 키를 누른다. **LSduRPathSourcePduRef** 에서 `EcuC/` 고른다.

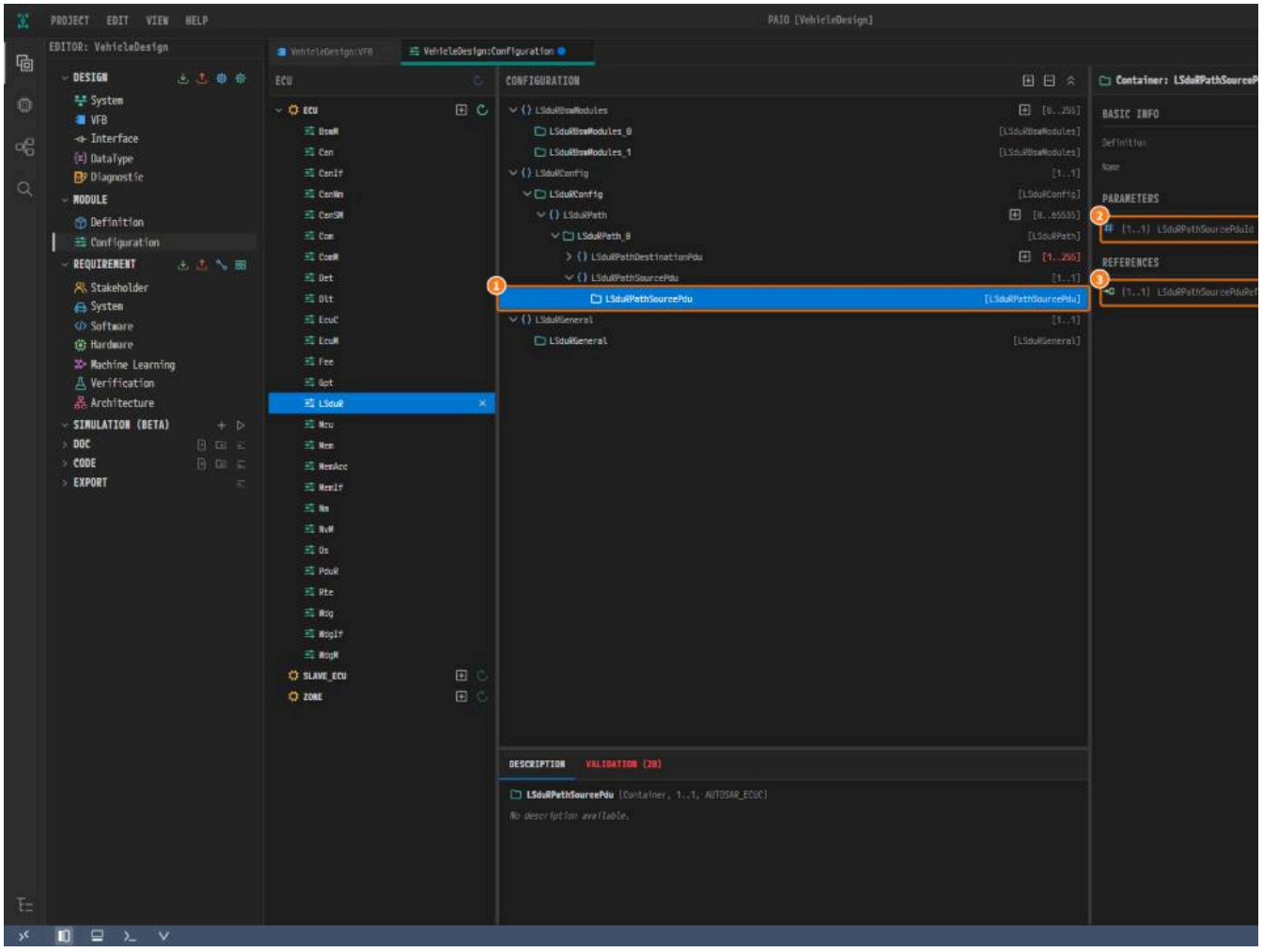


*그림 9-34. ① `LSduRPathSourcePdu`, ② **LSduRPathSourcePduId**, ③ **LSduRPathSourcePduRef**.*

10. `LSduRPath_0` 아래 `LSduRPathDestinationPdu` 행의 **+** 를 누른다. `LSduRPathDestinationPdu_0` 의 **LSduRPathDestinationPduRef** 에서 `EcuC/EcucConfig`

11. `LSduRPath_1` 도 표를 보고 같은 방법으로 만든다. 값은 `1` 과 `PduNmTx` 다. `Ctrl+S` 로 저장한다.

PduR 설정

1. 모듈 목록에서 `PduR` 을 누른다. `PduRGeneral` 행의 **+** 를 누른다.

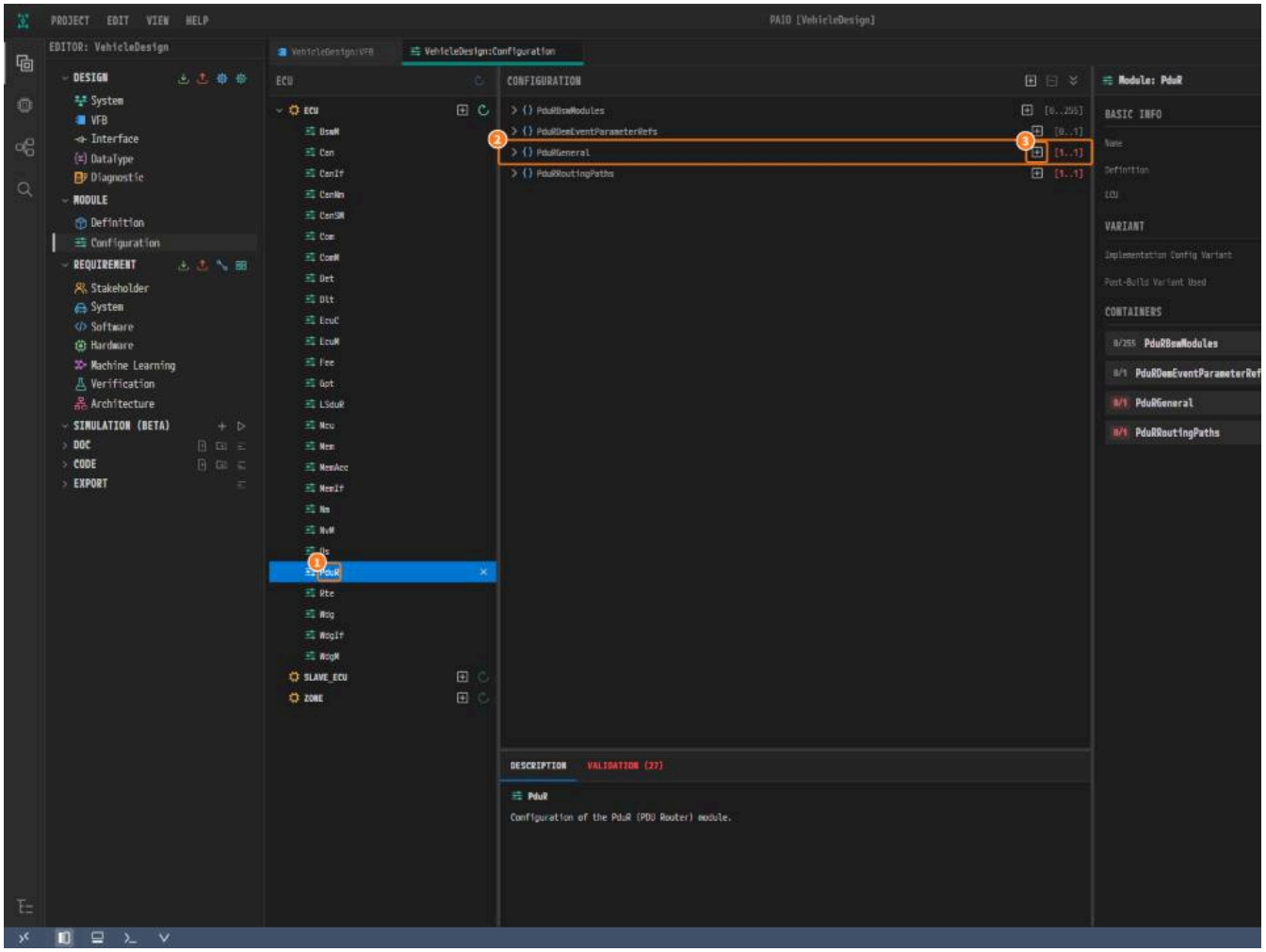


*그림 9-35. ① `PduR`, ② `PduRGeneral`, ③ **+**.*

2. `PduRRoutingPaths` 행의 **+** 를 누른다.

3. `PduRGeneral` 을 누르고 **PduRZeroCostOperation** 의 **FALSE** 를 누른다.

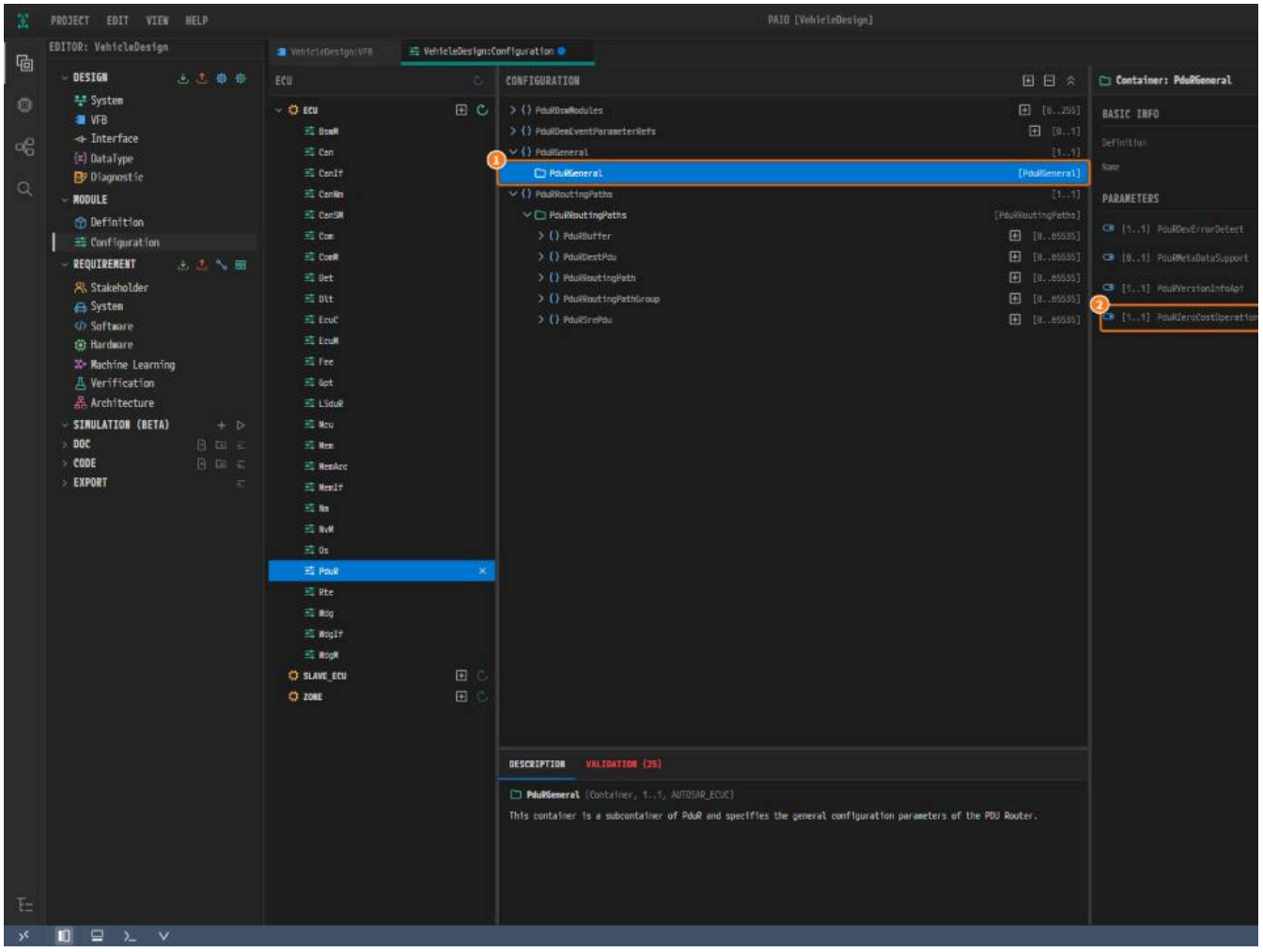


*그림 9-36. ① `PduRGeneral`, ② **PduRZeroCostOperation**.*

4. `PduRRoutingPaths` 를 누르고 **PduRConfigurationId** 에 `0` 을 입력한 뒤 `Tab` 키를 누른다. `Ctrl+S` 로 저장한다.

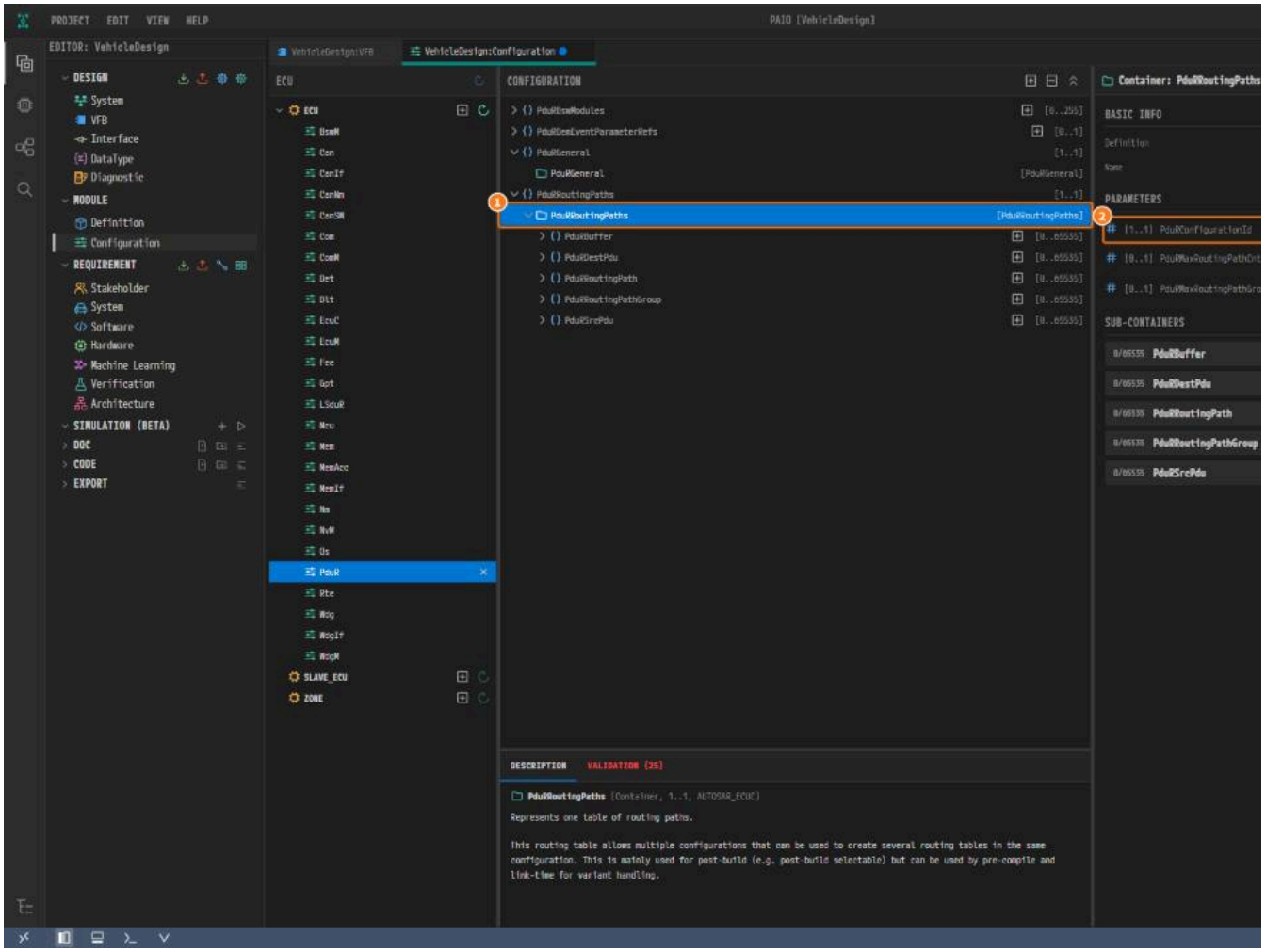


*그림 9-37. ① `PduRRoutingPaths`, ② **PduRConfigurationId.***

Com 설정

1. 모듈 목록에서 `Com` 을 누른다. `ComGeneral` 행의 **+** 를 누른다.

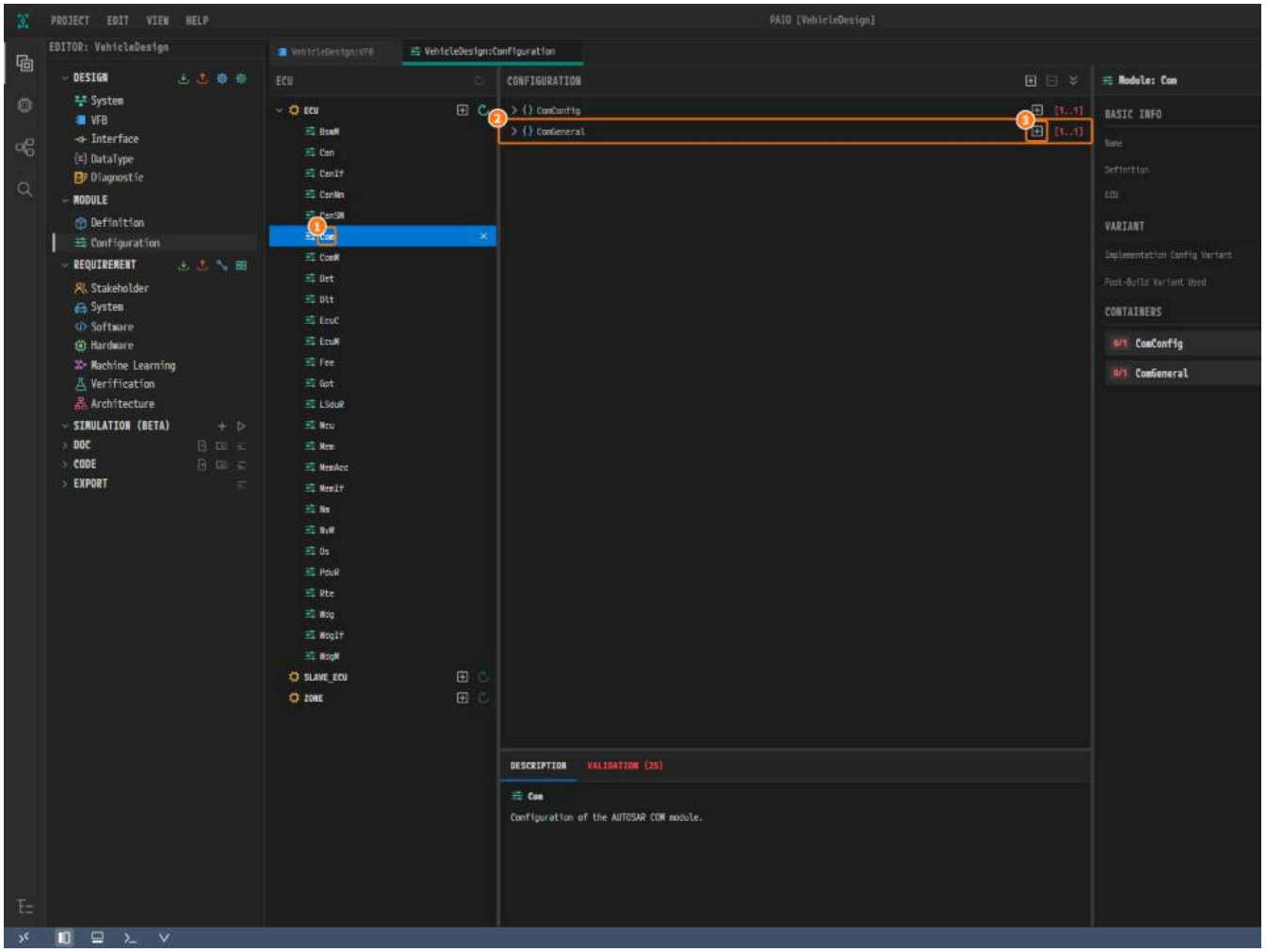


*그림 9-38. ① `Com`, ② `ComGeneral`, ③ **+**.*

2. `ComConfig` 행의 **+** 를 누른다.

3. `ComGeneral` 을 누르고 **ComSupportedIPduGroups** 에 `0` 을 입력한 뒤 `Tab` 키를 누른다.

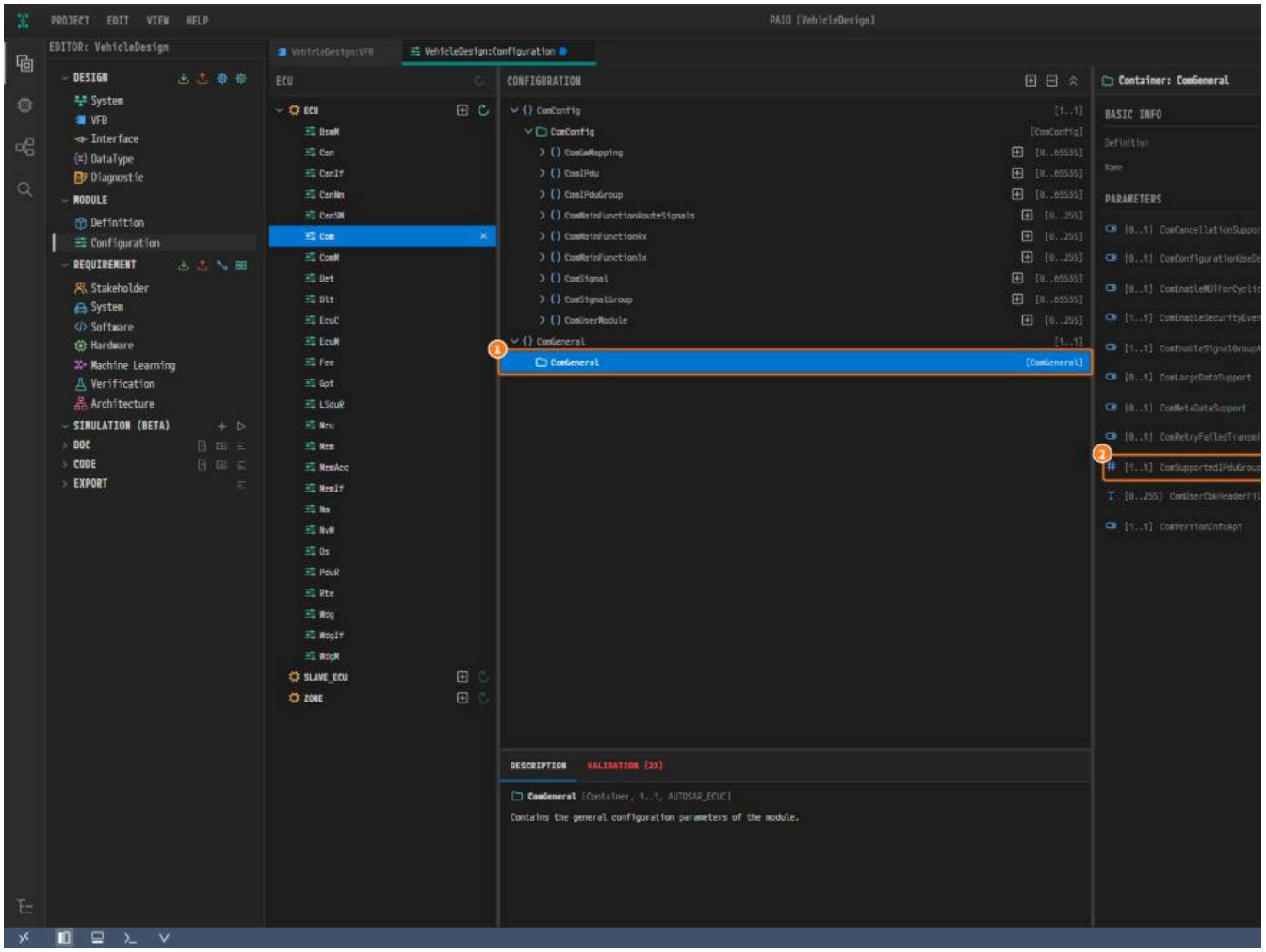


*그림 9-39. ① `ComGeneral`, ② **ComSupportedIPduGroups.***

4. `ComConfig` 아래에서 `ComMainFunctionRx` 행의 **+** 를 누른다.

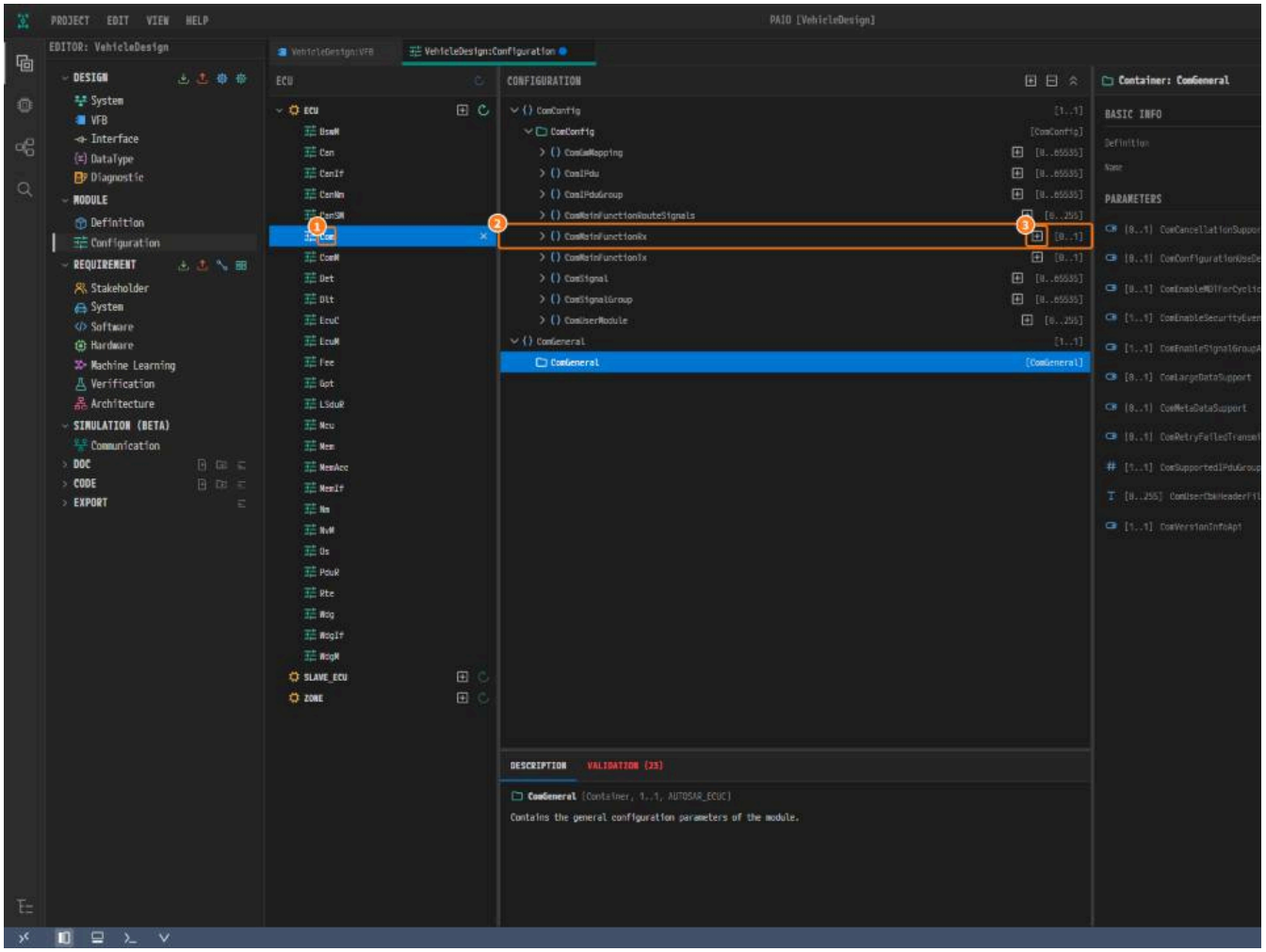


그림 9-40. ① `Com`, ② `ComMainFunctionRx`, ③ +.

5. **ComMainRxTimeBase** 가 `0.01` 인지 확인한다. 이름 `ComMainFunctionRx` 는 그대로 둔다. 8장의 Rte 매핑이 이 이름을 가리킨다.

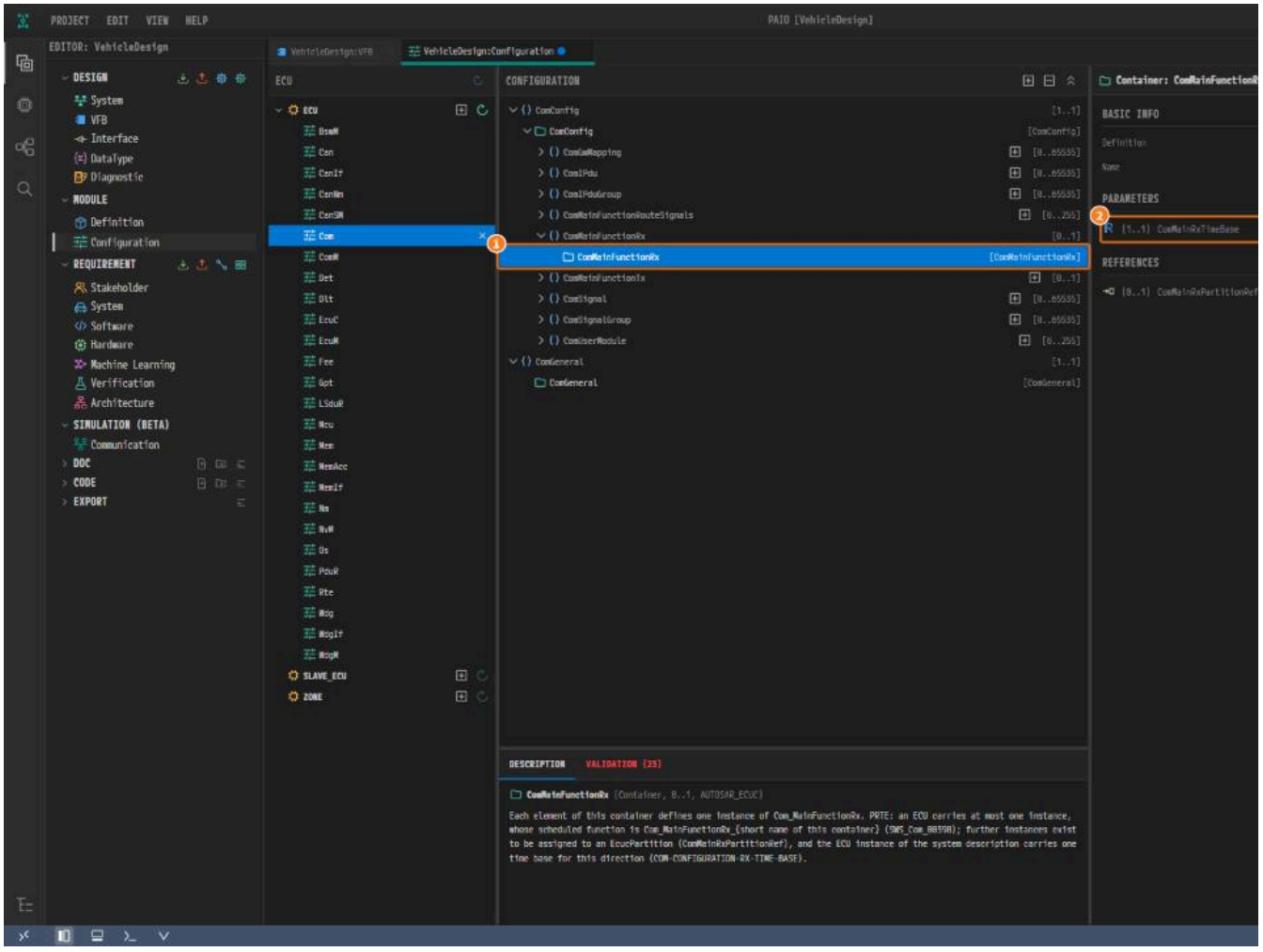


*그림 9-41. ① `ComMainFunctionRx`, ② **ComMainRxTimeBase.***

6. 같은 방법으로 `ComMainFunctionTx` 행의 **+** 를 누르고 **ComMainTxTimeBase** 가 `0.01` 인지 확인한다. `Ctrl+S` 로 저장한다.

왜 이렇게 설계하는가

- **PDU 는 EcuC 에 둔다.** 한 PDU 를 여러 모듈이 쓰므로 `EcuC` 에 한 번 정의하고 모든 모듈이 그것을 참조한다. 그래서 `EcuC` 가 `CanIf` 보다 먼저다.
- **방향마다 오브젝트 하나.** `CanHardwareObject_0` 은 송신하고 `CanHardwareObject_1` 은 수신하며, `CanIf` 는 `CanIfHthCfg_0` 과 `CanIfHrhCfg_0` 으로 이들에 닿
- **CAN id 는 CanIf 에 있다.** 두 PDU 의 CAN id `0x501`, `0x502` 는 `CanIf` 에만 입력한다.
- **버스 오프 보고는 CanSM 으로 간다.** `CanIfDispatchCfg` 의 `CAN_SM` 은 `CanSM` 이 컨트롤러를 다루도록 준비한다.
- **ComM 이 먼저다.** `CanSM`, `Nm`, `CanNm` 이 모두 `ComMChannel_0` 을 고르므로 그 채널이 먼저 있어야 한다.
- **NM 모듈이 둘이다.** `Nm` 은 버스 위의 계층이며, `NM_BUSNM_CANNM` 으로 채널을 NM PDU 를 실어 나르는 `CanNm` 에 넘긴다.
- **버스 오프 값은 직접 입력한다.** 이 튜토리얼에서는 클러스터 값을 `CanSM` 으로 옮겨 주는 것이 없으므로 같은 숫자를 직접 입력한다.
- **NM 은 ECU 설정에만 있다.** System 튜토리얼은 `CanCluster` 에서 NM 을 끄고 NM 프레임도 두지 않으므로, NM PDU 와 그 CAN id `0x501`/`0x502` 는 `EcuC` 와 `CanIf`
  하며, 실제 버스에서 NM 이 어떻게 동작하는지는 측정하지 않았다.
- **NM PDU 는 PduR 을 거치지 않는다.** NM PDU 는 `CanIf` ↔ `LSduR` ↔ `CanNm` 으로 오가므로 NM 경로는 `LSduR` 에 있고 `PduR` 은 아무것도 라우팅하지 않는다.
- **이웃 모듈마다 `LSduRBswModules` 하나.** `CanIf` 가 하위 모듈이고 `CanNm` 이 상위 모듈이다. `LSduR` 은 각 모듈을 **LSduRBswModuleRef** 로 찾으며, 이 참조는 드롭다운에
- **NM PDU 마다 경로 하나.** `LSduRPath_0` 은 `PduNmRx` 를, `LSduRPath_1` 은 `PduNmTx` 를 나른다. 출발지와 목적지는 같은 `EcuC` PDU 이며, `CanIf` 와 `CanNm` 도 이 PDU
- **Com 의 메인 함수는 둘이다.** `ComMainFunctionRx` 와 `ComMainFunctionTx` 는 각각 `Com` 의 메인 함수 하나를 만들고, 8장의 Rte 매핑이 이를 부른다. 없으면 PRTE 가 `R`
- **PduR 과 Com 에는 PDU 가 없다.** 두 SW-C 는 ECU 안에서만 통신하므로 애플리케이션 PDU 와 시그널을 설정하지 않는다. 그래도 이 모듈들은 ECU 의 CAN 스택에 속하며 함께

확인하기

- [ ] `EcuC` 에 **PduLength** `8` 인 `PduNmTx`, `PduNmRx` 와 `Core0` 이 있다.
- [ ] `Can` 에 `CanGeneral`, `CanControllerBaudrateConfig_0` 을 품은 `CanController_0`, 하드웨어 오브젝트 두 개가 있다.
- [ ] **CanIfPublicTxBuffering** 이 **TRUE** 다.
- [ ] `CanIfTxPduCfg_0` 은 `1281` 로 `PduNmTx` 를, `CanIfRxPduCfg_0` 은 `1282` 로 `PduNmRx` 를 가리킨다.
- [ ] `ComMNetworkManagement` 의 **ComMNmVariant** 가 `FULL` 이다.

- [ ] `CanSMManagerNetwork_0` 에 `2`, `0.1`, `0.2`, `0.1` 이 있고 `ComMChannel_0` 을 가리킨다.
- [ ] 표에 있는 `NmGlobalFeatures`, `NmChannelConfig_0`, `CanNmGlobalConfig`, `CanNmChannelConfig_0` 의 **TRUE/FALSE** 항목에 **CanNmCarWakeUpFilterEnable**
- [ ] **CanNmCarWakeUpFilterEnabled** 에 값이 없다.
- [ ] `CanNmRxPdu_0` 은 `PduNmRx` 를, `CanNmTxPdu` 는 `PduNmTx` 를 가리킨다.
- [ ] `LSduR` 에 `LSduRGeneral`, `LSduRConfig`, `LSduRBswModules_0`(`ECU/CanIf`), `LSduRBswModules_1`(`ECU/CanNm`)이 있다.
- [ ] `LSduRPath_0` 의 출발지와 목적지가 `PduNmRx` 를, `LSduRPath_1` 의 출발지와 목적지가 `PduNmTx` 를 가리킨다.
- [ ] `PduR` 에 `PduRGeneral` 과 `PduRRoutingPaths` 가 있다.
- [ ] `Com` 에 `ComGeneral` 과, `ComMainFunctionRx`·`ComMainFunctionTx` 를 둔 `ComConfig` 가 있다.
- [ ] 탭 `VehicleDesign:Configuration` 에 수정 표시가 없다.

다음 단계

검증과 코드 생성에서 설계를 검증하고 ECU 의 코드를 생성한다.

검증과 코드 생성

이 장에서 할 일

이 장에서는 **VALIDATION** 패널에서 `No validation issues.` 를 확인하고, 직접 넣은 값으로 ECU 의 코드를 생성한 뒤 결과를 살펴본다.

시작하기 전에

- Communication 설정을 마친 뒤 시작한다.
- 모든 모듈이 저장되어 있다. 탭 `VehicleDesign:Configuration` 에 수정 표시가 없다.

위치	항목	값
PRTE Code Generation 창	TARGET ECU	`ECU`

검증

1. 푸터의 **V** 아이콘을 눌러 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

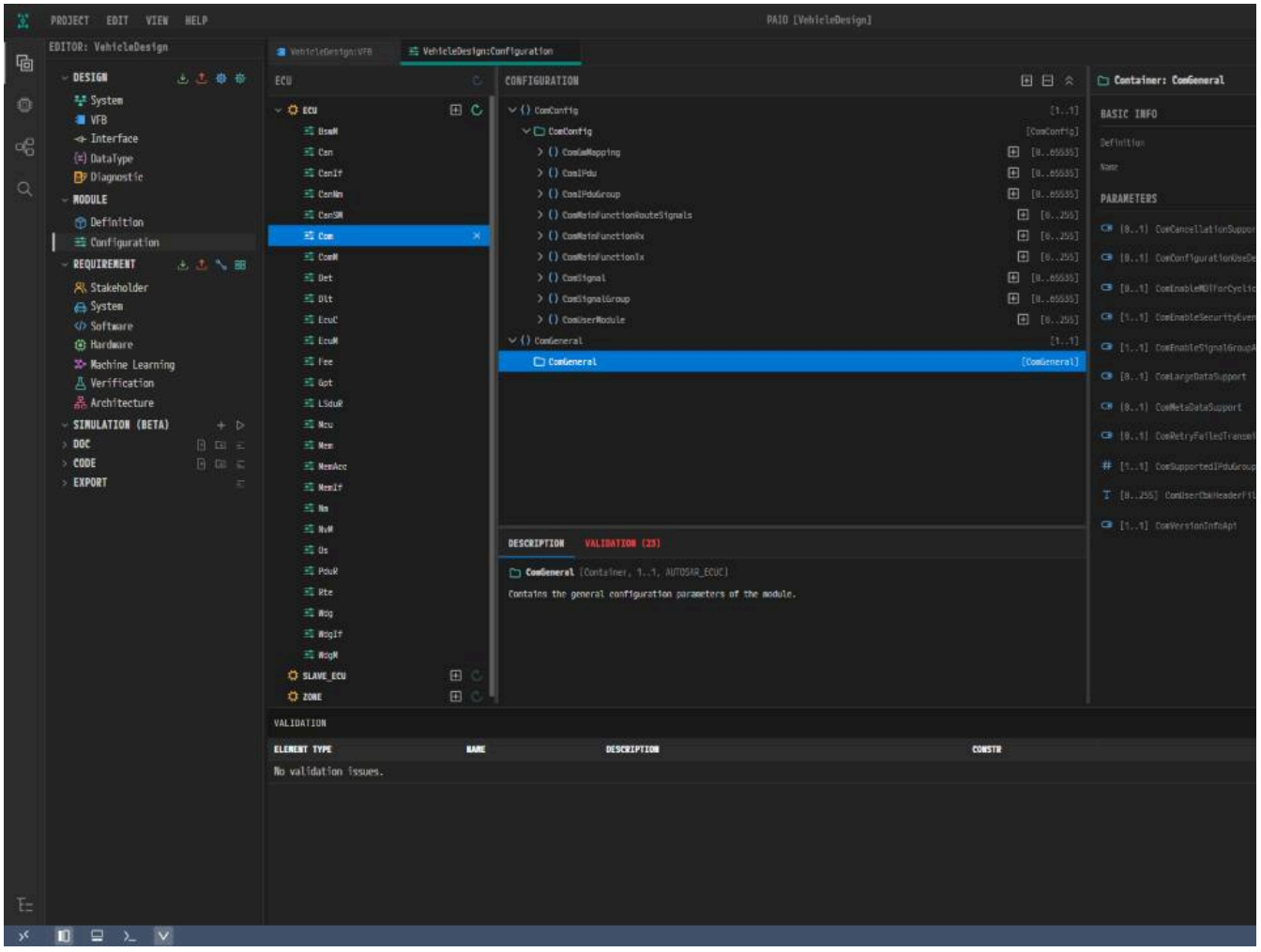


*그림 10-1. `No validation issues.` ECU Configuration 에디터 안의 **VALIDATION** 탭에는 여전히 숫자가 보인다. 비워 둔 선택 값 같은 항목을 세는 것이며, 코드 생성 여*

코드 생성

1. 패널을 닫는다. **DESIGN** 머리글의 **PRTE Code Generation** 아이콘을 누른다.

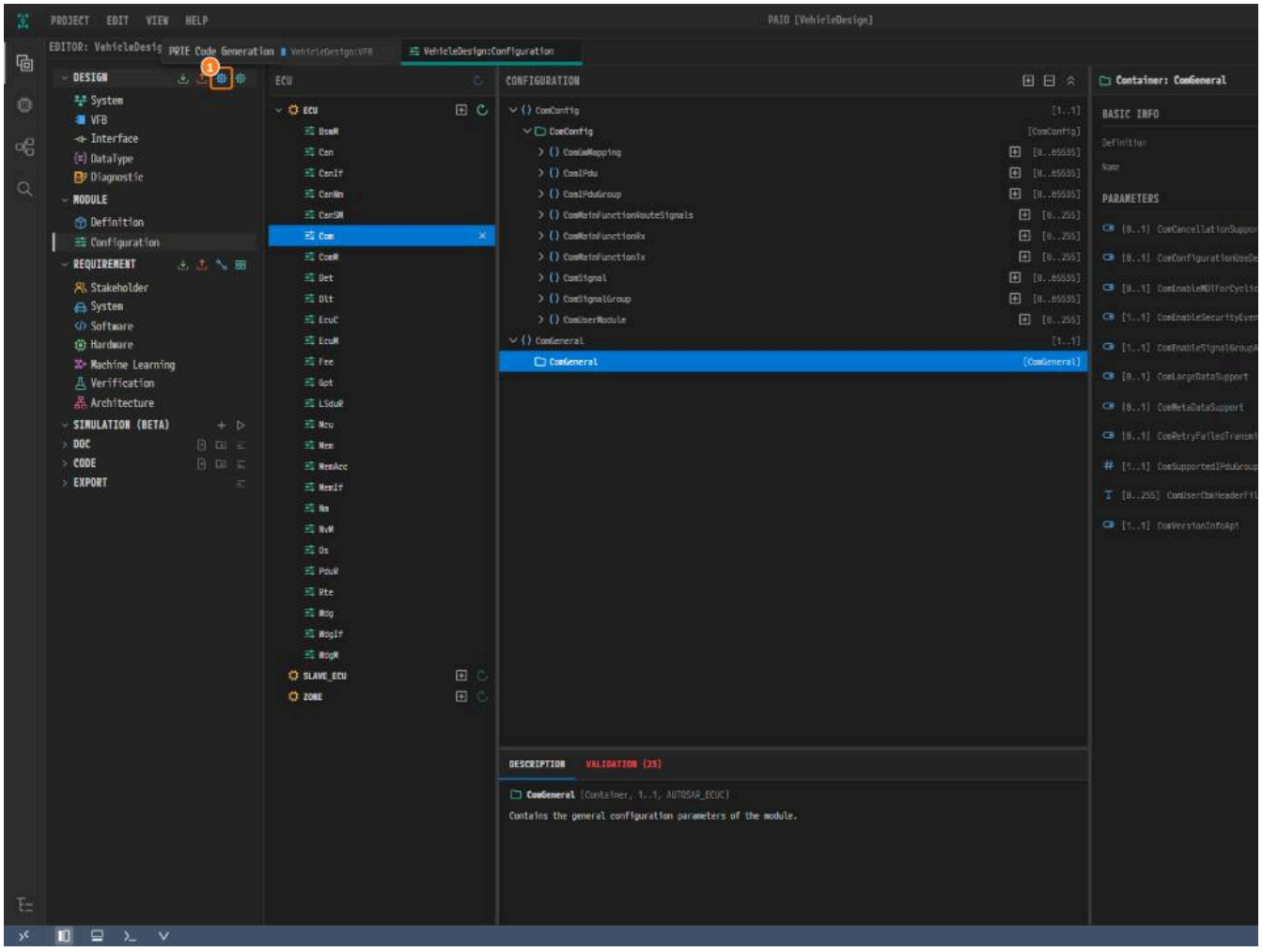


*그림 10-2. ① **PRTE Code Generation**.*

2. **TARGET ECU** 가 `ECU` 인지 확인하고 **Generate** 를 누른다.

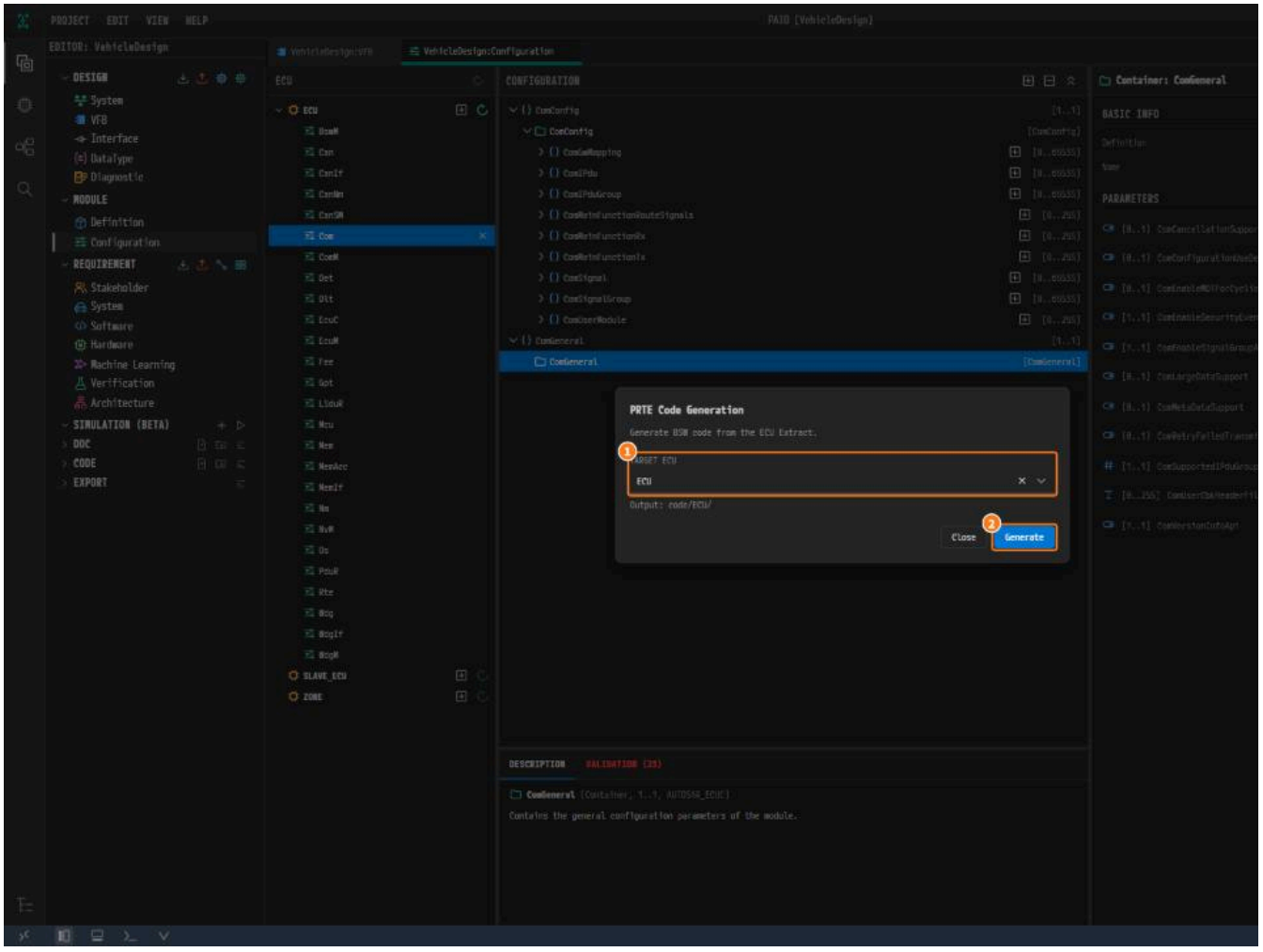


*그림 10-3. ① **TARGET ECU**, ② **Generate**.*

3. 끝나면 창에 `COMPLETE` 와 `99 file(s) generated under code/ECU/` 가 보이고, `INFO` 행 네 개가 있으며 경고와 오류는 없다. `MODULE_SET` 행에 생성된 모듈 15개 `Com`, `ComM`, `CanSM`, `CanNm`, `Nm`, `LSduR`, `Det` 가 나열된다. **Close** 를 누른다.

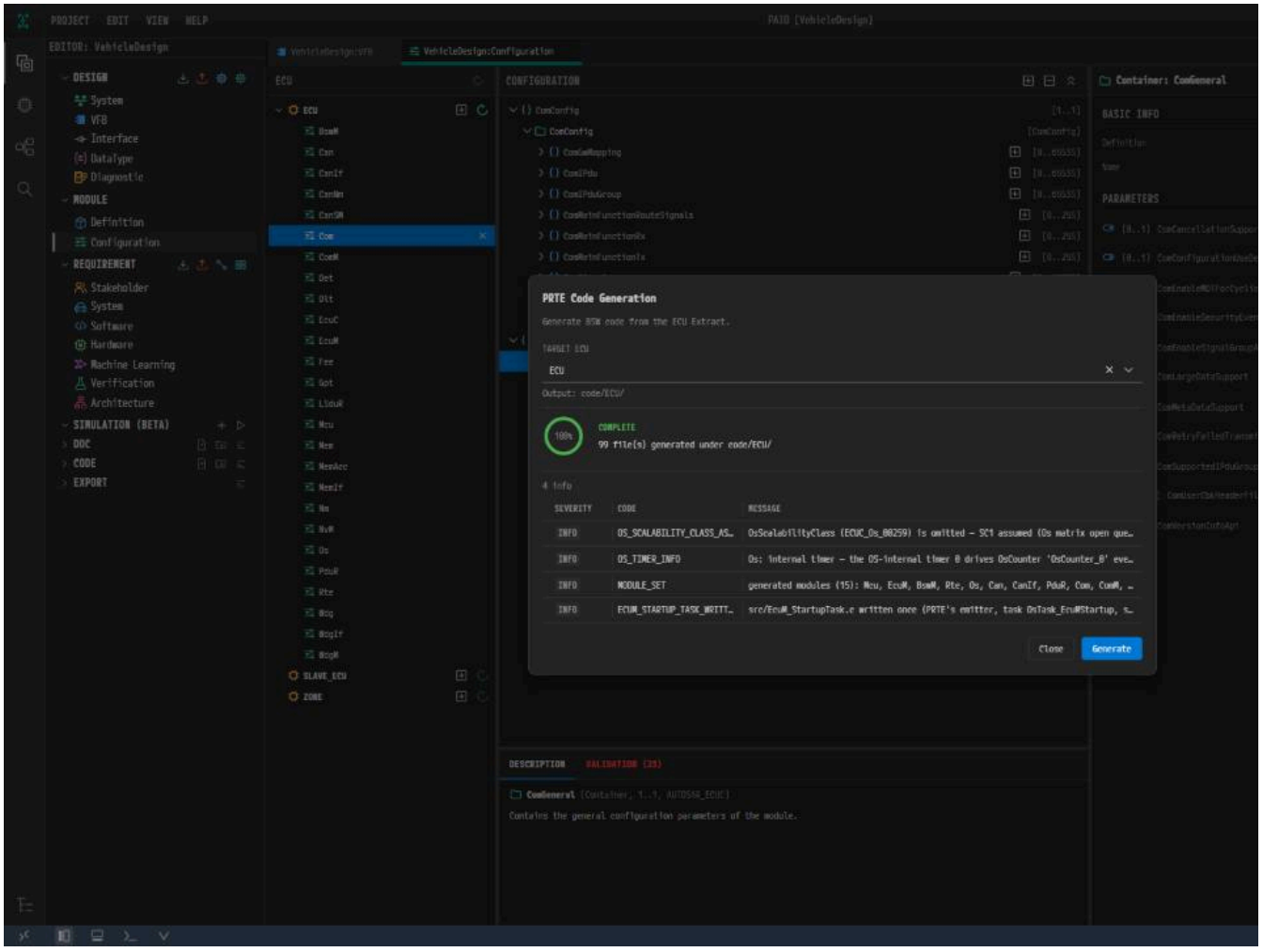


그림 10-4. 생성 결과.

결과 살펴보기

1. **EDITOR** 뷰의 **CODE** 를 열고 `ECU` 를 연다.

폴더 `code/ECU/` 의 내용은 다음과 같다.

경로	내용	소유자
`gen/`	생성된 RTE 와 BSW 설정	생성기. 실행할 때마다 다시 쓴다
`src/`	`SwcSensor.c`, `SwcActuator.c`, `EcuM` 시동 파일 `EcuM_StartupTask.c` 와 `EcuM_Callout_Stubs.c`	개발자. 한 번 만들어지고 덮어쓰지 않는다
`include/`	`SwcSensor.h`, `SwcActuator.h`	개발자
`CMakeLists.txt`	빌드 설명	개발자. 한 번 만들어진다

> **참고.** 이 창에는 **PROBLEMS** 숫자와 코드의 빨간 표시가 보일 수 있다. `Rte.h` 같은 SDK 헤더는 컴퓨터가 아니라 PRTE vECU 컨테이너에 있어서 컴퓨터의 창에서는 찾지 못했다. 른 종류의 표시는 따로 확인한다. 이 튜토리얼은 **VALIDATION** 패널과 코드 생성으로 설계를 판정한다. 코드 자체의 판정은 빌드로 하며 vECU 의 일이다.

왜 이렇게 설계하는가

- **BSW 의 값은 모두 직접 넣은 것이다.** 자동 설정은 한 번도 돌지 않았다. 생성된 BSW 코드의 값은 6~9장에서 직접 넣었거나 "+" 로 컨테이너를 만들 때 들어간 기본값이다. 6장 타입·인스턴스 쌍만 예외다.
- **시동 태스크.** `ECUM_STARTUP_TASK_WRITTEN` 행은 태스크 `OsTask_EcuMStartup` 에 대해 `src/EcuM_StartupTask.c` 를 썼다는 뜻이며, 그래서 이 태스크 이름은 정
- **튜토리얼은 여기서 끝난다.** 검증을 통과하고 코드가 생성되는 설계가 목표다. 러너블 본문 작성과 vECU 에서의 빌드는 그다음이며 각자의 프로젝트에서 하는 일이다.

확인하기

- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.
- [ ] **PRTE Code Generation** 창에 `COMPLETE` 와 `99 file(s) generated under code/ECU/` 가 보였다.
- [ ] `MODULE_SET` 행에 모듈 15개가 있었고, 경고나 오류 행은 없었다.
- [ ] **CODE** 에 `ECU` 와 그 아래 `gen`, `include`, `src`, `CMakeLists.txt` 가 있다.

다음 단계

부록에 이 설계의 모든 값과 자주 생기는 문제를 모았다.

AUTOSAR AP 튜토리얼이 이 프로젝트를 이어서 쓴다. 노드 `HPC` 에 Adaptive Application 을 올린다. 이 튜토리얼을 건너뛰고 다음 편부터 시작하려면 완성된 프로젝트 VehicleD

부록

이 장에서 할 일

이 부록은 따라 하는 장이 아니라 참고 자료다. 설계를 다시 점검하거나 어느 값이 어느 장의 것인지 찾을 때 쓴다.

모든 값

1장에서 3장

장	요소	값
System	프로젝트, ECU 노드	`VehicleDesign`, `ECU` (AUTOSAR System 튜토리얼에서 만든 것)
1	컴포지션(**NAME**, **AS PROTOTYPE**)	`ECU`, **ECU [ECU]** 에 매핑
2	데이터 타입	`SpeedType`, **ALIAS**, **BASETYPE** `uint16`
3	Sender-Receiver 인터페이스	`IfSpeed`, `SpeedType` 타입의 데이터 요소 `speed`
3	Client-Server 인터페이스	`IfScale`, 오퍼레이션 `GetScale`, 인자 `scale`, `OUT`, `SpeedType`

4장과 5장

SW-C	포트	종류	인터페이스	설정
`SwcSensor`	`PpSpeed`	**PPort**	`IfSpeed`	**NON-QUEUED**
`SwcSensor`	`RpScale`	**RPort**	`IfScale`	**CLIENT**
`SwcActuator`	`RpSpeed`	**RPort**	`IfSpeed`	**NON-QUEUED**
`SwcActuator`	`PpScale`	**PPort**	`IfScale`	**SERVER**

러너블	액세스 포인트	RTE 이벤트
`Sensor_Cyclic`	`Rte_Write_PpSpeed_speed`, `Rte_Call_RpScale_GetScale`	**TimingEvent** `SensorCyclicTiming`, `0.02` `s`
`Actuator_Init`		**InitEvent** `ActuatorInitEvent`
`Actuator_OnData`	`Rte_Read_RpSpeed_speed`	**DataReceivedEvent** `ActuatorOnDataEvent`, `RpSpeed` / `speed`
`Actuator_GetScale` (**CAN BE INVOKED** = **TRUE**)		**OperationInvokedEvent** `ActuatorGetScaleEvent`, `PpScale` /

커넥터: `Assembly_SwcSensor_PpSpeed_SwcActuator_RpSpeed`, `Assembly_SwcActuator_PpScale_SwcSensor_RpScale`.

6장

요소	값
BSW 모듈	**PRTE Platform Import** 의 제안된 선택. CAN 스택 `Can`, `CanIf`, `CanSM`, `ComM`, `Com`, `PduR`, `LSduR` 를 포함하며, 여기에 `CanNm` 과 `Nm` 을 직접 선택한다
ECU Configuration [ECU] 창	적용하지 않고 `Esc` 로 닫는다
`EcuM`, `BswM`, `Mcu`, `Det`	PRTE Setup의 표대로 손으로 넣는다. `BswMGeneral` 의 **BswMNmEnabled** `TRUE`, `McuInit`, `BswMPreInit`, `CanInit`, `CanIfInit`, `LSduRInit`, `PduRInit`, `Co` 담은 `EcuMDriverInitListOne` 이 여기에 있다
Rte 값 파일	VehicleDesign_Rte.arxml, `Rte` 에 **All → Use Imported** (`1 replaced`)
Rte 값 파일에서 온 `Rte`	SW-C 타입·인스턴스 쌍, `BswMInstance/RteEventToTaskMapping_0` (`BswMPollTiming` 을 가리키는 **RteEventRef**), `RteBswModuleInstance_0`, `_3`, `_4`, `_5`,

7장

`Os` 컨테이너	값
`OsOS`	**OsStatus** `EXTENDED`, **OsStackMonitoring**, **OsUseGetServiceId**, **OsUseParameterAccess** TRUE
`OsOS/OsHooks`	훅 다섯 개 **FALSE**
`OSDEFAULTAPPMODE`(`OsAppMode`)	값 없음
`RES_AUTOSAR_ECUM`(`OsResource`)	**OsResourceProperty** `STANDARD`
`OsCounter_0`	**OsCounterMaxAllowedValue** `1000000`, **OsCounterMinCycle** `1`, **OsCounterTicksPerBase** `1`, **OsCounterType** `HARDWARE`, **OsSecondsPerTick** `0`

모든 태스크는 **OsTaskActivation** `1`, **OsTaskSchedule** `FULL`, **OsTaskStackSize** `32768` 이다. 모든 알람은 **OsAlarmCounterRef** `Os/OsCounter_0` 이다.

태스크	OsTaskPeriod	OsTaskPriority	활성화하는 것
`OsTask_0`	`0.01`	`2`	`OsAlarm_0`
`OsTask_EcuMStartup`	비움	`3`	`Os/OSDEFAULTAPPMODE` 의 **OsTaskAutostart**
`OsTask_2`	`0.02`	`1`	`OsAlarm_1`
`OsTask_3`	`0.01`	`1`	`OsAlarm_2`

8장

`Rte` 컨테이너	값
`RteGeneration`	**RteCodeVendorId** `79`
`RteBswGeneral`	추가만, 값 없음
`RteInitializationBehavior_0`	**RteInitializationStrategy** `RTE_INITIALIZATION_STRATEGY_AT_DATA_DECLARATION`, **RteSectionInitializationPolicy** `INIT`
`EcuMInstance/RteModeMachineInstanceConfig_0`	**RteModeMachineQueueLength** `1`, **RteSwcModeManagerRef** `PortInterface/ModeSwitch/EcuM_CurrentMode/current` `SwComponentType/Atomic/Service/EcuM/currentMode`
`RteOsInteraction/RteUsedOsActivation_0`, `_1`, `_2`	**RteExpectedActivationOffset** `0`, **RteExpectedTickDuration** `0.01`, **RteActivationOsAlarmRef** `Os/OsAlarm_0`, `Os/OsAlar`
`RteBswModuleInstance_0`~`_7`	차례로 `BswM`, `EcuM`, `Can`, `CanIf`, `CanSM`, `ComM`, `Com`, `CanNm` 의 **RteBswImplementationRef**. `_1`, `_2`, `_7` 은 "+" 로 추가하고 **RteExclusiveAreaImplMechanism** 이 `ALL_INTERRUPT_BLOCKING` 이다. `Nm` 에는 인스턴스가 없다

모든 매핑은 태스크 매핑 플래그가 **TRUE** 다. `ActuatorGetScaleEvent` 는 매핑하지 않는다.

이벤트	태스크	알람	위치
`SensorCyclicTiming`	`Os/OsTask_2`	`Os/OsAlarm_1`	-
`ActuatorInitEvent`	`Os/OsTask_3`	-	`2`
`ActuatorOnDataEvent`	`Os/OsTask_3`	-	`1`
`BswMPollTiming`(Rte 값 파일)	`Os/OsTask_3`	`Os/OsAlarm_2`	`0`
`BswM`, `EcuM`, `Can`(다섯 개), `CanSM`, `ComM`, `Com`(두 개), `CanNm` 의 BSW 메인 함수 이벤트 열두 개	`Os/OsTask_0`	`Os/OsAlarm_0`	`2`~`13`

9장

모듈	값
EcuC	PduNmTx 와 PduNmRx (**PduLength** 8)를 가진 EcucConfigSet/EcucPduCollection (**PduIdTypeEnum** UINT16 , **PduLengthTypeEnum** UINT32), EcucHardware/Core0 (**Ecu**
Can	CanGeneral (**CanIndex** 0 , **CanMainFunctionModePeriod** 0.01 , **CanTimeoutDuration** 0.001), CanConfigSet/CanController_0 (**CanControllerId** 0 , 처리 방식 POLLING , Mcu/McuModuleConfiguration/Default/CoreClock), CanControllerBaudrateConfig_0 (**CanControllerBaudRate** 500), CanHardwareObject_0 (TRANSMIT)과 CanHa
CanIf	CanIfPublicCfg (**CanIfPublicHandleTypeEnum** UINT16 , **CanIfPublicTxConfirmPollingSupport** 와 **CanIfPublicTxBuffering TRUE**), CanIfPrivateCfg (LINEAR), CanIfDis CanIfInitCfg (CanIf_Config), CanIfCtrlCfg_0 (→ CanController_0)을 가진 CanIfCtrlDrvCfg_0
CanIf	CanIfHthCfg_0 (→ CanHardwareObject_0), CanIfHrhCfg_0 (→ CanHardwareObject_1), CanIfBufferCfg_0 (**CanIfBufferSize** 1 , → CanIfHthCfg_0), CanIfTxPduCf CanIfRxPduCfg_0 (1282 = 0x502 , → PduNmRx)
ComM	ComMGeneral , ComMConfigSet/ComMChannel_0 (COMM_BUS_TYPE_CAN , **ComMChannelId** 0), ComMNetworkManagement (**ComMNmVariant** FULL)
CanSM	CanSMGeneral (0.01), CanSMConfiguration (3 , 0.02), CanSMManagerNetwork_0 (2 , 0.1 , 0.2 , 0.1 , **TRUE**, → ComMChannel_0), CanSMController_0 (→ CanIfCtrl
Nm	NmGlobalConstants (**NmNumberOfChannels** 1), NmGlobalFeatures (항목 여덟 개 **FALSE**), NmGlobalProperties (**NmCycletimeMainFunction** 0.01)를 가진 NmGlobalCon NmBusType/NmStandardBusNmConfig (NM_BUSNM_CANNM)를 가진 NmChannelConfig_0 (항목 네 개 **FALSE**, → ComMChannel_0)
CanNm	CanNmGlobalConfig (항목 열세 개 **FALSE**, **CanNmMainFunctionPeriod** 0.01), CanNmChannelConfig_0 (**CanNmMsgCycleTime** 0.1 , **CanNmTimeoutTime** 1.0 , **CanNmNode** **CanNmCarWakeUpFilterEnabled** 값 없음, → ComMChannel_0), CanNmRxPdu_0 (→ PduNmRx), CanNmTxPdu (→ PduNmTx)
LSduR	LSduRGeneral (**LSduRMetaDataSupport**, **LSduRZeroCostOperation FALSE**), LSduRConfig
LSduR	LSduRBswModules_0 (ECU/CanIf , **LSduRLowerModule TRUE**), LSduRBswModules_1 (ECU/CanNm , **LSduRUpperModule TRUE**). 둘 다 **LSduRRxIndication**, **LSduRTransmit**, L
LSduR	LSduRPath_0 (0 , 출발지와 목적지 모두 PduNmRx), LSduRPath_1 (1 , 출발지와 목적지 모두 PduNmTx)
PduR	PduRGeneral (**PduRZeroCostOperation FALSE**), PduRRoutingPaths (**PduRConfigurationId** 0)
Com	ComGeneral (**ComSupportedIPduGroups** 0), ComConfig 와 그 아래 ComMainFunctionRx (**ComMainRxTimeBase** 0.01), ComMainFunctionTx (**ComMainTxTimeBase** 0.01

항목의 전체 목록은 각 장의 표에 있다. 네트워크 관리는 ECU 설정에만 있다. System 튜토리얼은 CanCluster 에서 NM 을 끄고 NM 프레임도 두지 않으므로, NM PDU 와 그 CAN 정).

생성된 코드

검증과 코드 생성은 COMPLETE 와 99 file(s) generated under code/ECU/ 로 끝나며, INFO 행 네 개가 있고 경고와 오류는 없다. MODULE_SET 행에 모듈 15개 Mcu , E ComM , CanSM , CanNm , Nm , LSduR , Det 가 나열된다.

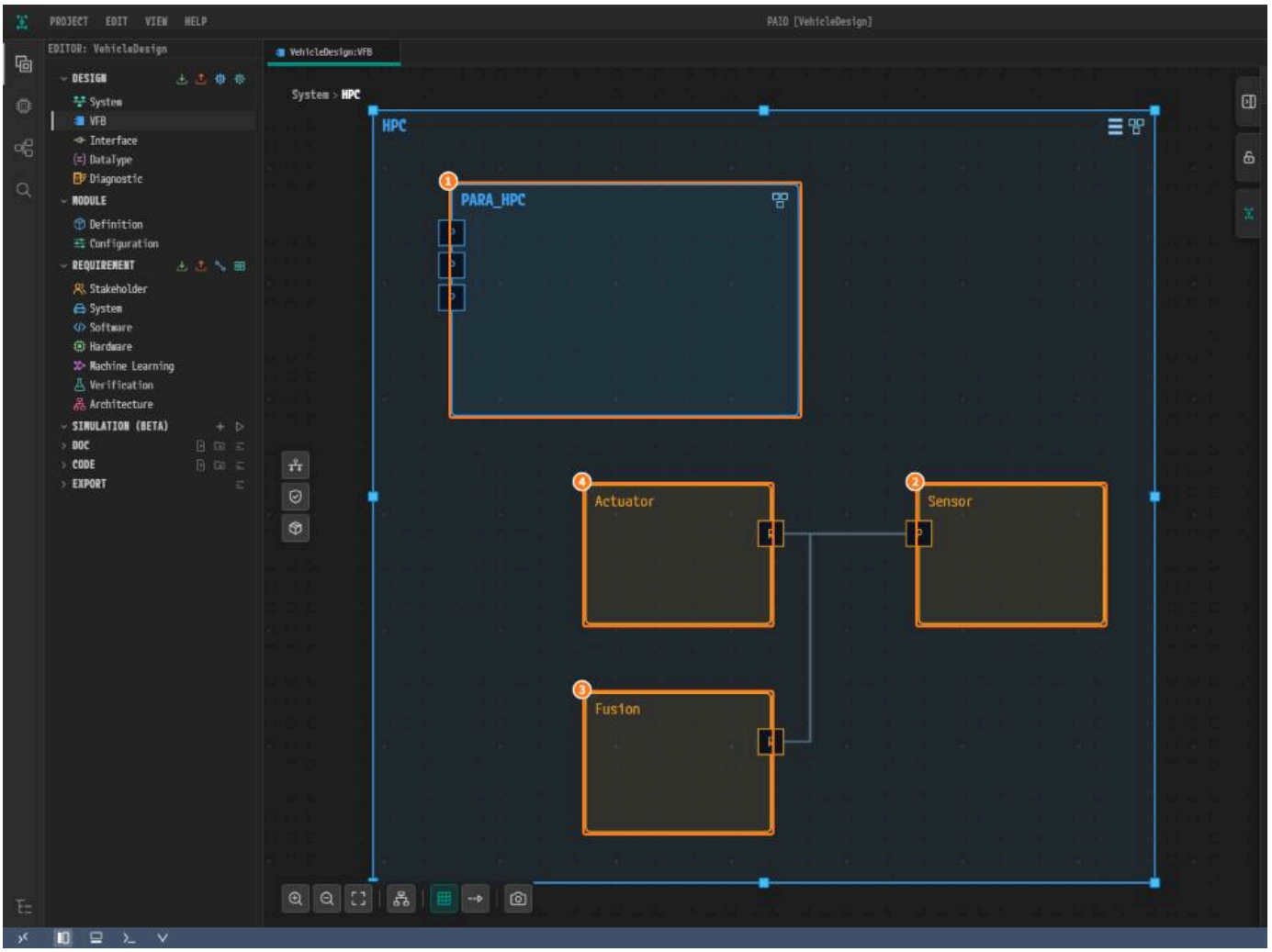
code/ECU/ 아래 경로	내용	소유자
gen/	생성된 RTE 와 BSW 설정	생성기. 실행할 때마다 다시 쓴다
src/	SwcSensor.c , SwcActuator.c , EcuM 시동 파일 EcuM_StartupTask.c 와 EcuM_Callout_Stubs.c	개발자. 한 번 만들어지고 덮어쓰지 않는다
include/	SwcSensor.h , SwcActuator.h	개발자
CMakeLists.txt	빌드 설명	개발자. 한 번 만들어진다

자주 생기는 문제

증상	원인
컴포지션 안에 ECU 아이콘이 없다	컴포지션이 노드에 매핑되지 않았다
직접 넣은 BSW 값이 덮어써졌다	**PRTE AUTO CONFIGURATION**, **PRTE Auto Configuration** 아이콘, **Auto-Generate** 를 눌렀거나 **E Configuration** 창을 `Ctrl+S` 로 적용했다
생성이 `MANDATORY_MODULE_MISSING` 으로 멈춘다	`Os`, `Rte`, `EcuM`, `BswM`, `Mcu` 가운데 설정이 없는 모듈이 있다
PRTE Setup 의 `EcuM` 이 애플리케이션 모드나 리소스를 찾지 못한다	이름이 정확히 `OSDEFAULTAPPMODE`, `RES_AUTOSAR_ECUM` 이 아니다
`src/EcuM_StartupTask.c` 가 쓰이지 않는다	시동 태스크 이름이 정확히 `OsTask_EcuMStartup` 이 아니다
생성이 `RTE_EVENT_MAPPING_TASK_FLAG_MISMATCH` 나 `RTE_BSW_EVENT_MAPPING_TASK_FLAG_MISMATCH` 로 멈춘다	**RteEventIsMappedToTask** 나 **RteBswEventIsMappedToTask** 가 "+" 가 쓴 `FALSE` 그대로다
생성이 `RTE_BSW_INSTANCE_MISSING` 으로 멈춘다	`CanNm` 에는 메인 함수가 있는데 `RteBswModuleInstance_7` 이 없다
생성이 `RTE_CS_SERVER_UNMAPPED_NOT_CONCURRENT` 로 멈춘다	`Actuator_GetScale` 이 **TRUE** 가 아니다
`EcuC` 에 PDU 는 있는데 `Core0` 이 없다	`EcuC` 에 무엇이든 들어가면 코어 `Core0` (**EcucCoreId** `0`)이 필요하다
생성이 `CANIF_BUFFERING_GATE_INCONSISTENT` 로 멈춘다	송신 PDU 가 `CanIfBufferCfg_0` 을 쓰는데 **CanIfPublicTxBuffering** 이 **TRUE** 가 아니다
TRUE 나 **FALSE** 값이 비고 행에 빨간 테두리가 생겼다	이미 선택된 버튼을 다시 눌러 값이 지워졌다
ComMNmVariant 가 `FULL` 이 아니다	이 ECU 는 `CanNm` 을 돌리므로 "+" 가 쓴 `FULL` 그대로 두어야 한다
`CanSM`, `Nm`, `CanNm` 에서 `ComMChannel_0` 을 가리키는 참조의 목록이 비어 있다	`ComM` 을 먼저 설정하지 않았다
생성이 `Nm` 이나 `CanNm` 의 빈 **TRUE/FALSE** 항목에서 멈춘다	이 항목들 가운데 상당수는 기본값이 없어서 "+" 가 비워 둔다
생성이 `CANNM_CARWAKEUP_LAYOUT` 으로 멈춘다	**CanNmCarWakeUpFilterEnabled** 에 "+" 가 쓴 값이 남아 있다
생성이 `LSDUR_PATH_SOURCE_UNRESOLVED` 로 멈춘다	**LSduRBswModuleRef** 를 목록에서 고르지 않고 입력했다
생성 코드에 `Rte.h` 같은 헤더를 찾지 못했다는 표시가 있다	SDK 헤더가 컴퓨터가 아니라 vECU 에 있어서 컴퓨터의 창에서는 찾지 못한다

AUTOSAR AP 튜토리얼

이 튜토리얼은 프로젝트 `VehicleDesign` 의 HPC 노드 `HPC` 에 Adaptive Application 세 개를 PAIO 로 올리고, SOME/IP 서비스 `Sensor` 로 잇고, PARA 의 플랫폼을 가져오고, 젝트를 차례로 만드는 세 편 가운데 마지막 편이다: AUTOSAR System → AUTOSAR CP → AUTOSAR AP.



이 튜토리얼에서 만드는 소프트웨어. 컴포지션 `HPC` 안의 플랫폼 `PARA_HPC` 와 `Sensor`, `Fusion`, `Actuator` 이고, 서비스 `Sensor` 가 `Sensor` 에서 `Fusion` 과 `Actuator`

무엇을 만드는가

`Sensor` 는 서비스 `Sensor` 를 SOME/IP(서비스 ID `11`)로 제공하고, `Fusion` 과 `Actuator` 가 그 서비스를 소비한다. 세 프로세스는 플랫폼 프로세스 `CM` 이 실행된 뒤 함수 그룹 `CM` 을 비롯한 플랫폼 프로세스 열한 개는 PARA 에서 온다.

Adaptive Application	역할	DLT 애플리케이션 ID
`Sensor`	서비스 `Sensor` 제공	`SENS`
`Fusion`	서비스 `Sensor` 소비	`FUSN`
`Actuator`	서비스 `Sensor` 소비	`ACTA`

준비물

- 설치되어 실행할 수 있는 PAIO.
- AUTOSAR CP 튜토리얼을 마친 프로젝트 `VehicleDesign`. 앞 튜토리얼을 건너뛰려면 VehicleDesign-cp.zip을 내려받아 압축을 풀고, **PROJECT** 메뉴의 **Open Project** 로
- 1920×1080 이상의 화면. 다이얼로그가 넓다.
- Adaptive Application, 서비스 인터페이스, SOME/IP 를 대략 알면 충분하다. 어느 화면에 무엇을 입력하는지는 장마다 보여 준다.

장 구성

장	하는 일	사용하는 편집기
1. 컴포지션 HPC 와 머신	컴포지션 `HPC` 를 만들어 노드 `HPC` 에 매핑하고 머신을 저장한다	VFB, Machine Configuration
2. 머신 설정	OS, 코어, DLT ECU ID, State Management 값을 정한다	Machine Configuration
3. 토폴로지 — SOME/IP 디스커버리 포트와 UDP 엔드포인트	`EthernetConnector` 에 SD 포트와 UDP 엔드포인트 3개를 더한다	System 편집기
4. 데이터 타입	`ByteArray` 와 `String` 을 만든다	DataType
5. 서비스 인터페이스	서비스 인터페이스 `Sensor` 를 만든다	Interface
6. SOME/IP 배포와 서비스 인스턴스	`Sensor` 를 SOME/IP 로 배포하고 제공자 1개와 소비자 2개를 만든다	Interface
7. 커넥터에 서비스 인스턴스 바인딩	인스턴스 3개를 UDP 포트와 함께 `EthernetConnector` 에 바인딩한다	System 편집기
8. Adaptive Application 과 포트	Adaptive Application 3개와 포트, 커넥터 2개를 만든다	VFB
9. PARA 설정	PARA Platform Import 로 플랫폼 애플리케이션을 가져오고 Network Management 를 끈다	Machine Configuration, VFB
10. Executable 과 Process	애플리케이션마다 Executable 과 시작 상태를 가진 Process 를 만든다	AA Internal Behavior, Machine Conf
11. 서비스 인스턴스 매핑	포트 3개를 서비스 인스턴스에 매핑한다	AA Internal Behavior
12. 로그와 DLT	프로세스마다 DLT 애플리케이션을 정한다	AA Internal Behavior, Machine Conf
13. 검증과 코드 생성	설계를 검증하고 PARA Code Generation 으로 `HPC` 의 코드를 생성한다	VALIDATION 패널, PARA Code Genera
부록	설계 체크리스트, 값 표, 파일 구성, 자주 겪는 문제	—

이 튜토리얼을 읽는 법

- **장은 순서대로 따라간다.** 각 장은 앞 장의 결과에서 시작한다.
- **반복 작업은 한 번만 보여 준다.** 하나를 절차로 보여 주고, 나머지는 값 표로 주며 같은 방법으로 입력한다.
- UI 라벨은 화면에 보이는 대로 **굵게**, 모델 요소 이름과 값은 `code` 로 적는다.

다음 단계

컴포지션 HPC 와 머신부터 시작한다.

컴포지션 HPC 와 머신

이 장에서 할 일

이 장에서는 VFB 편집기에 컴포지션 `HPC` 를 만들어 HPC 노드 `HPC` 에 매핑한다. 그다음 **Machine Configuration** 다이얼로그를 한 번 열어 기본값을 저장한다.

시작하기 전에

- AUTOSAR CP 튜토리얼을 마친 프로젝트 `VehicleDesign` 에서 시작한다. 건너뛰었다면 VehicleDesign-cp.zip을 내려받아 압축을 풀고, **PROJECT** 메뉴의 **Open Project**

위치	항목	값
컴포지션 패널	**NAME**, **AS PROTOTYPE**	`HPC`
컴포지션 메뉴	**ECU/HPC Mapping**	**HPC [HPC]**

컴포지션 HPC 만들기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 클릭한다. 컴포지션 `ECU [ECU]` 가 있는 탭 `VehicleDesign:VFB` 가 열린다.
2. `ECU [ECU]` 바깥의 빈 곳을 오른쪽 클릭하고 **Add Composition** 을 클릭한다.

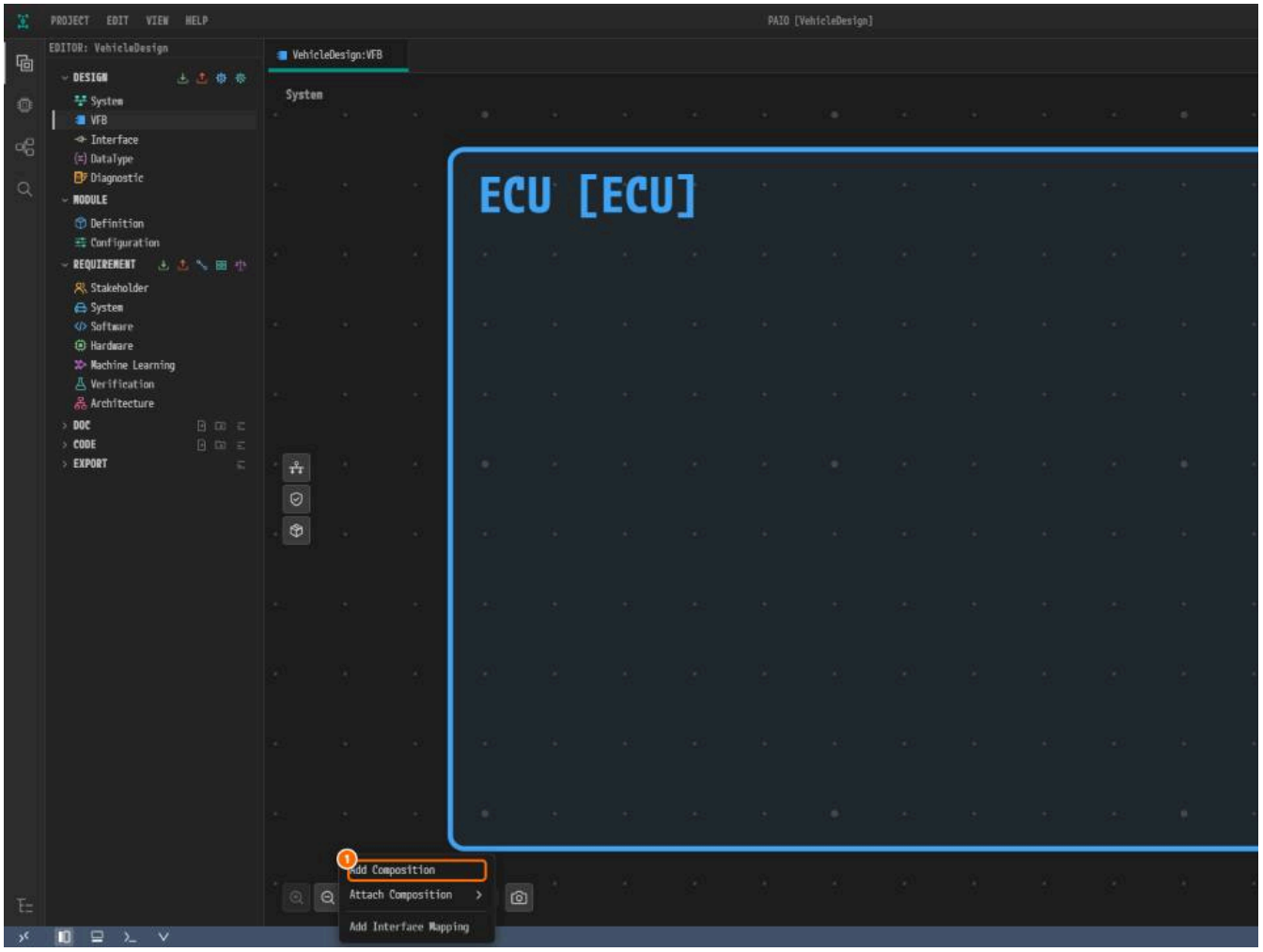


*그림 1-1. ① **Add Composition**.*

3. 새 컴포지션을 클릭한다. 패널의 **NAME** 과 **AS PROTOTYPE** 에 모두 `HPC` 를 입력하고 각각 `Tab` 을 누른다. **AS PROTOTYPE** 은 **NAME** 을 따라 바뀌지 않으므로 직접 입력한다
4. `Ctrl+S` 를 누른다.

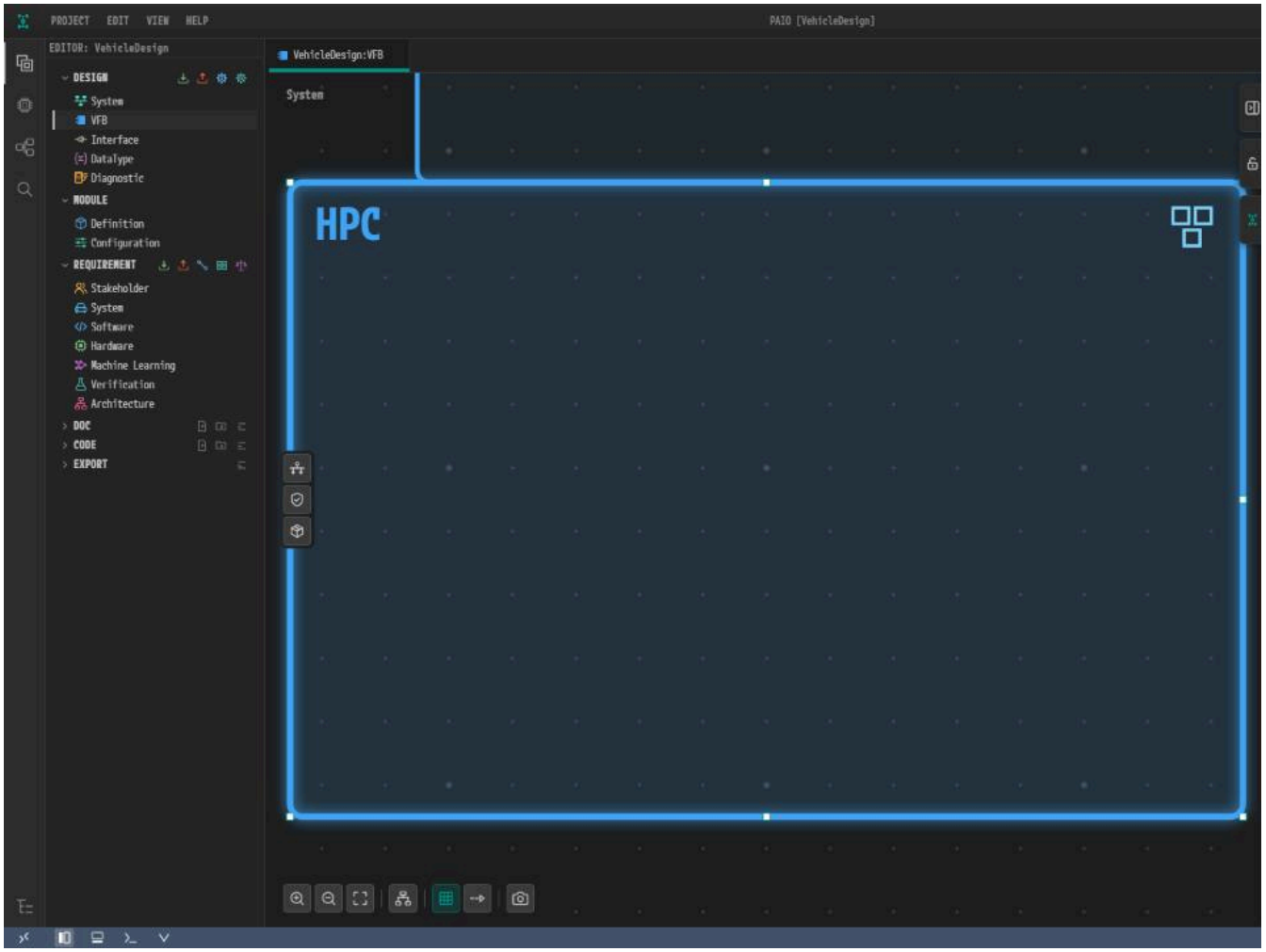


*그림 1-2. ① **NAME**, ② **AS PROTOTYPE**. 새 컴포지션이 지금은 `ECU [ECU]` 를 덮고 있다.*

컴포지션을 노드 HPC 에 매핑하기

1. 컴포지션 `HPC` 안쪽을 오른쪽 클릭한다. **ECU/HPC Mapping** 위에 포인터를 올리고 **HPC [HPC]** 를 클릭한다. 매핑은 바로 저장된다.

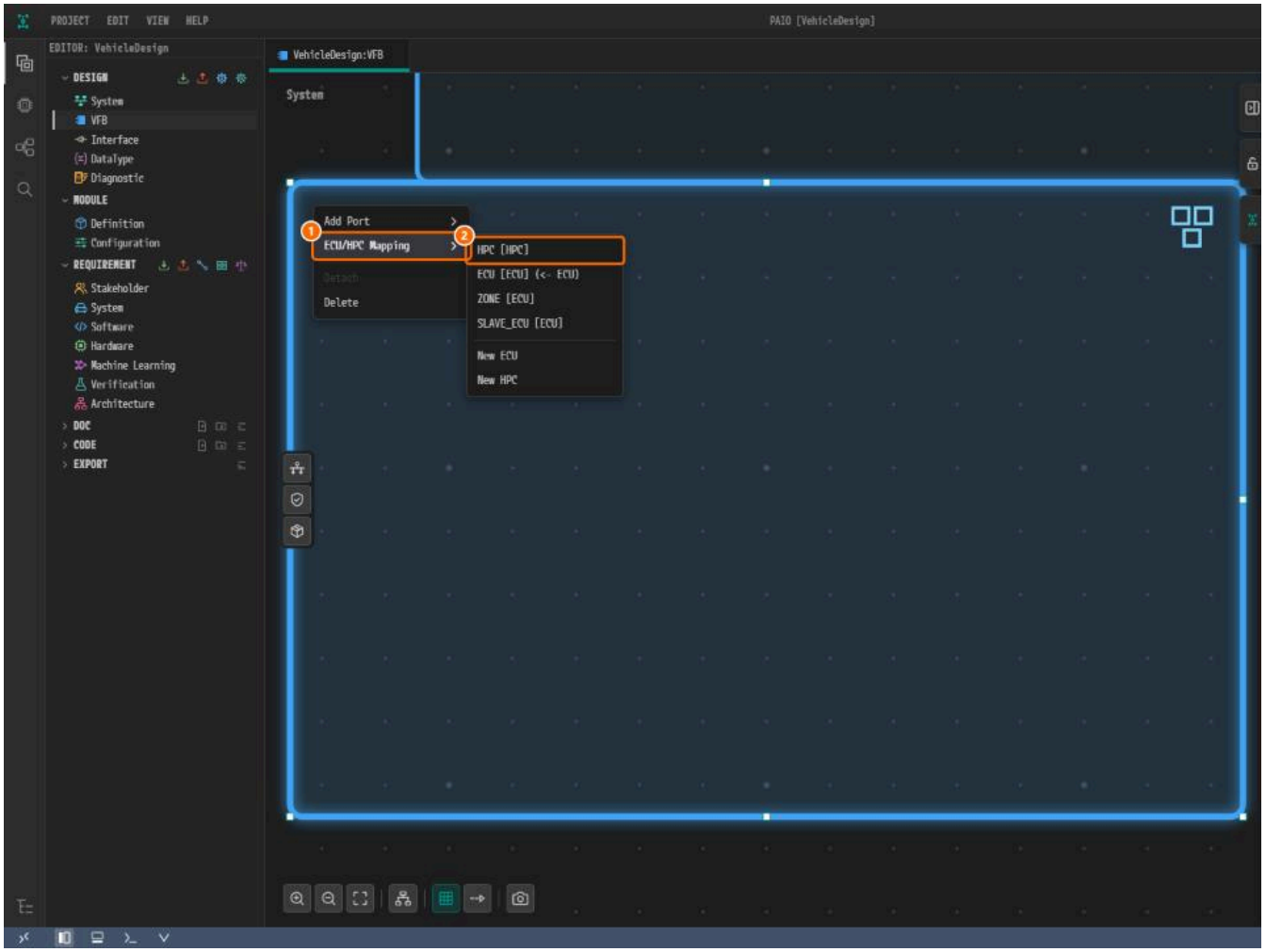


*그림 1-3. ① **ECU/HPC Mapping**, ② **HPC [HPC]**.*

2. 캔버스 아래 도구 막대의 세 번째 버튼(맞춤)을 클릭한다. 컴포지션 이름표가 `HPC [HPC]` 로 바뀌고 헤더 오른쪽에 파란 머신 아이콘이 붙는다.

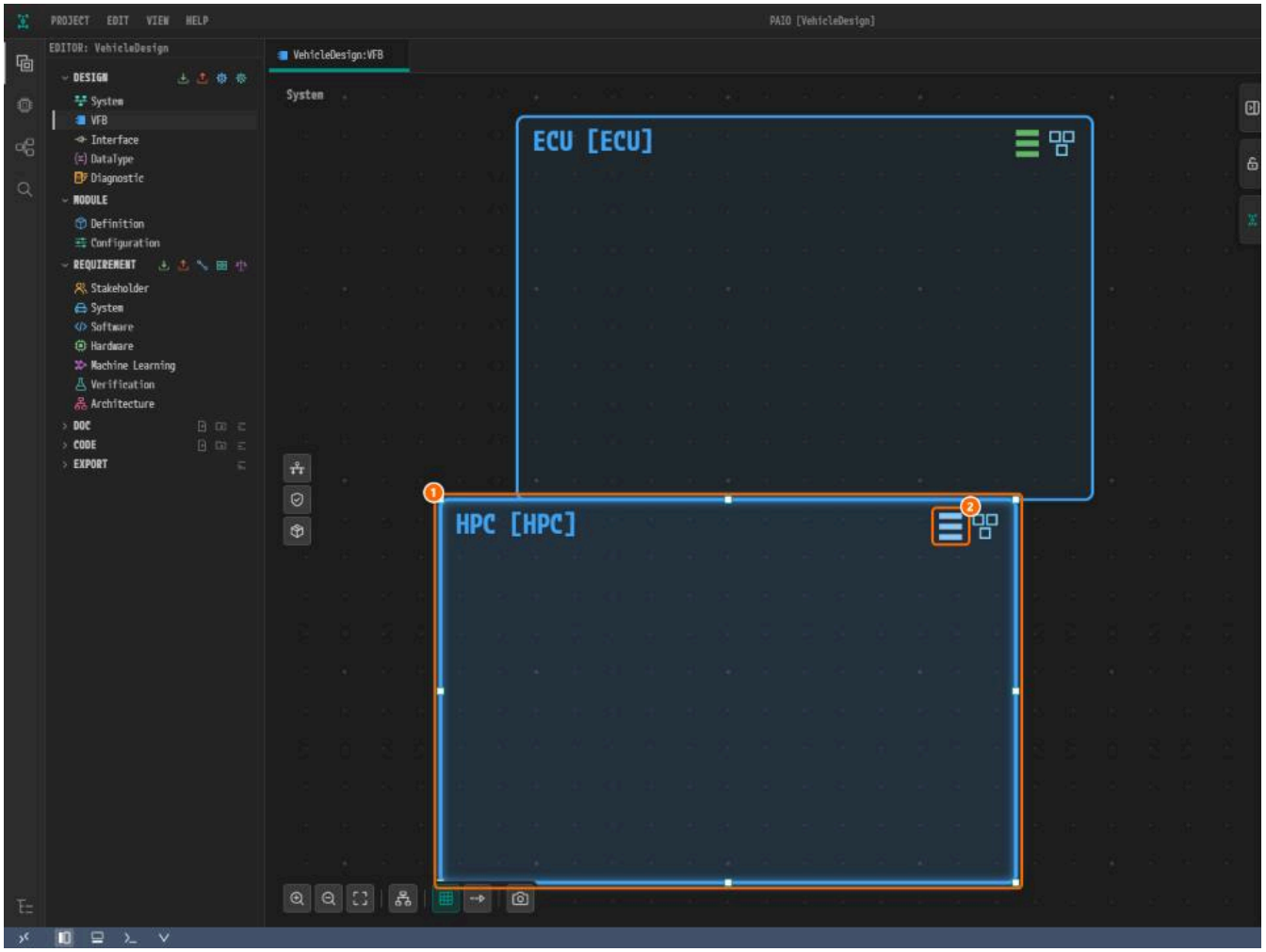


그림 1-4. ① `HPC [HPC]`, ② 머신 아이콘.

Machine Configuration 다이얼로그 열어 보기

다이얼로그에서는 값을 바꾸지 않는다. 한 번 저장하면 기본값(함수 그룹 `MachineFG`, 프로세서 `CPU`, 모듈 11개)이 머신 파일에 기록된다.

다이얼로그 열기

1. 탭 `VehicleDesign:VFB` 의 **×** 로 탭을 닫는다.
2. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 다시 클릭한다.
3. `HPC [HPC]` 의 머신 아이콘을 클릭한다. 다이얼로그 **Machine Configuration [HPC]** 가 열린다.

> **주의.** 매핑할 때 열려 있던 VFB 탭에서는 머신 아이콘으로 다이얼로그가 열리지 않는다. 1~2번처럼 탭을 닫았다가 다시 연다.

MACHINE 탭

다이얼로그는 **MACHINE** 탭으로 열린다. **OS** 는 비어 있고 `Name is required` 가 보인다. 다음 장에서 채운다.

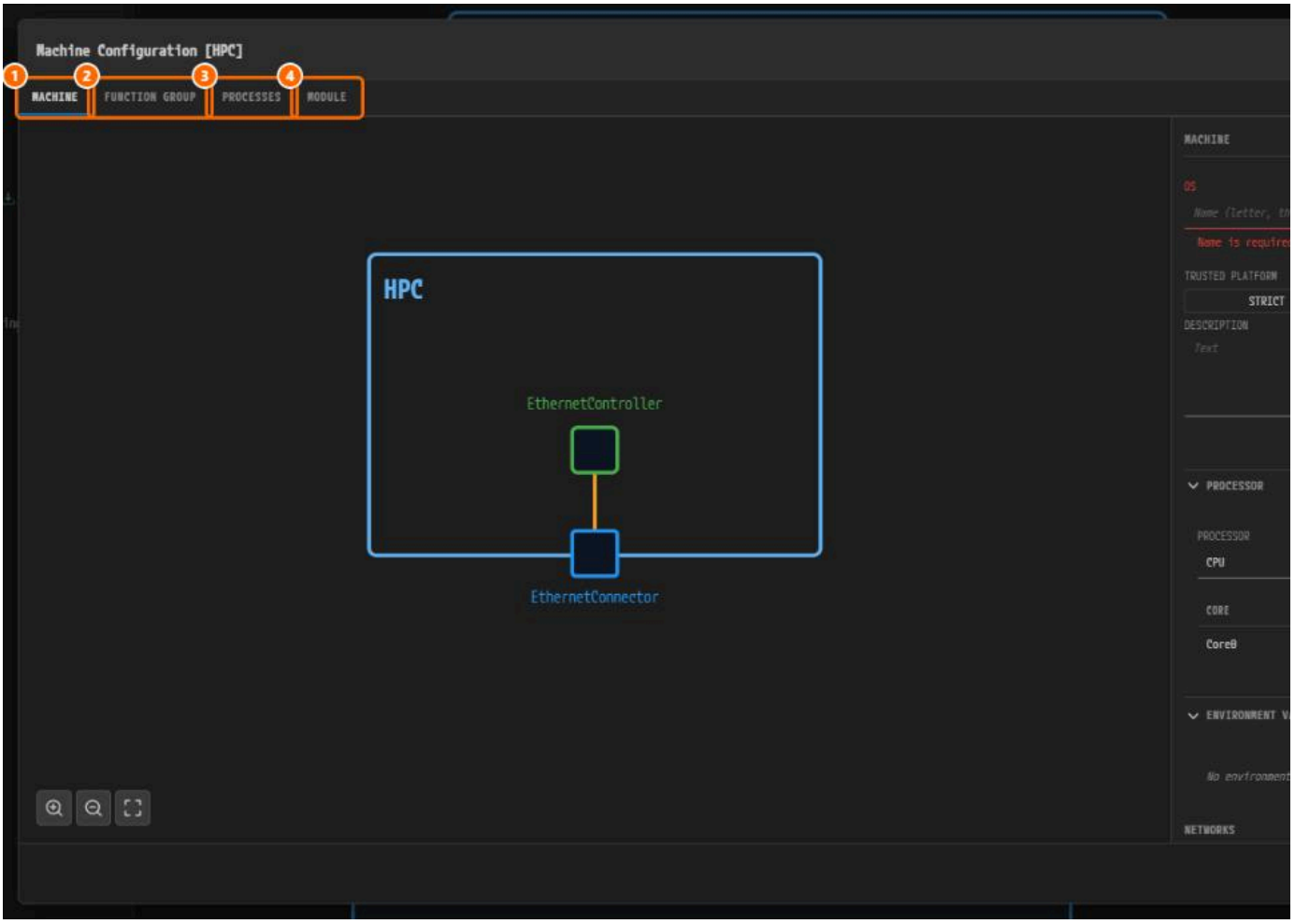


*그림 1-5. ① **MACHINE**, ② **FUNCTION GROUP**, ③ **PROCESSES**, ④ **MODULE** 탭.*

FUNCTION GROUP 탭

1. **FUNCTION GROUP** 탭을 클릭한다. 목록에 상태 `Off`, `Startup`, `Shutdown`, `Restart`, `Verify` 를 가진 `MachineFG` 가 있다.

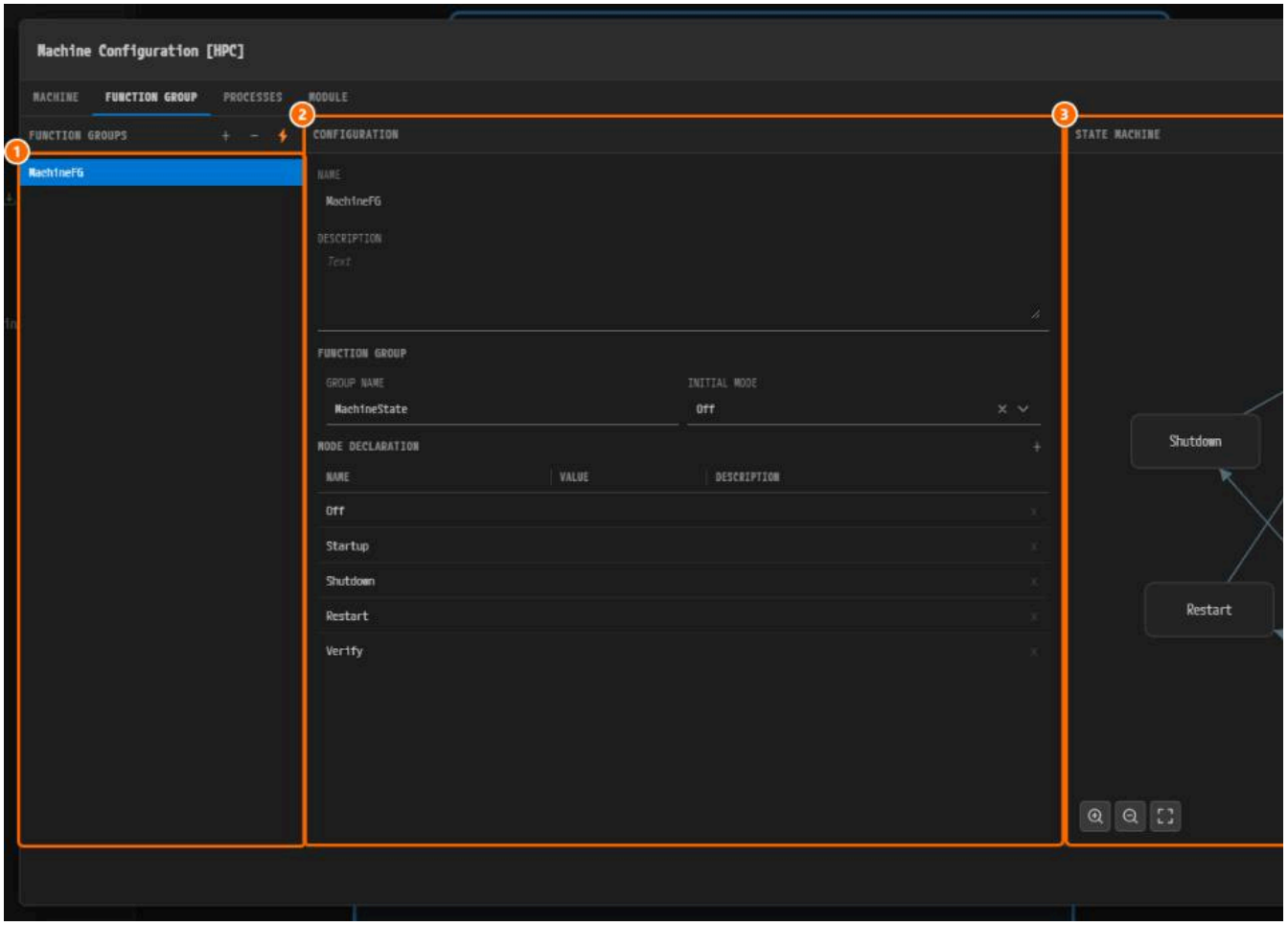


*그림 1-6. ① **FUNCTION GROUPS** 목록, ② **CONFIGURATION**, ③ **STATE MACHINE**.*

PROCESSES 탭

1. **PROCESSES** 탭을 클릭한다. `No Adaptive Application processes found in this Composition` 이 보인다.

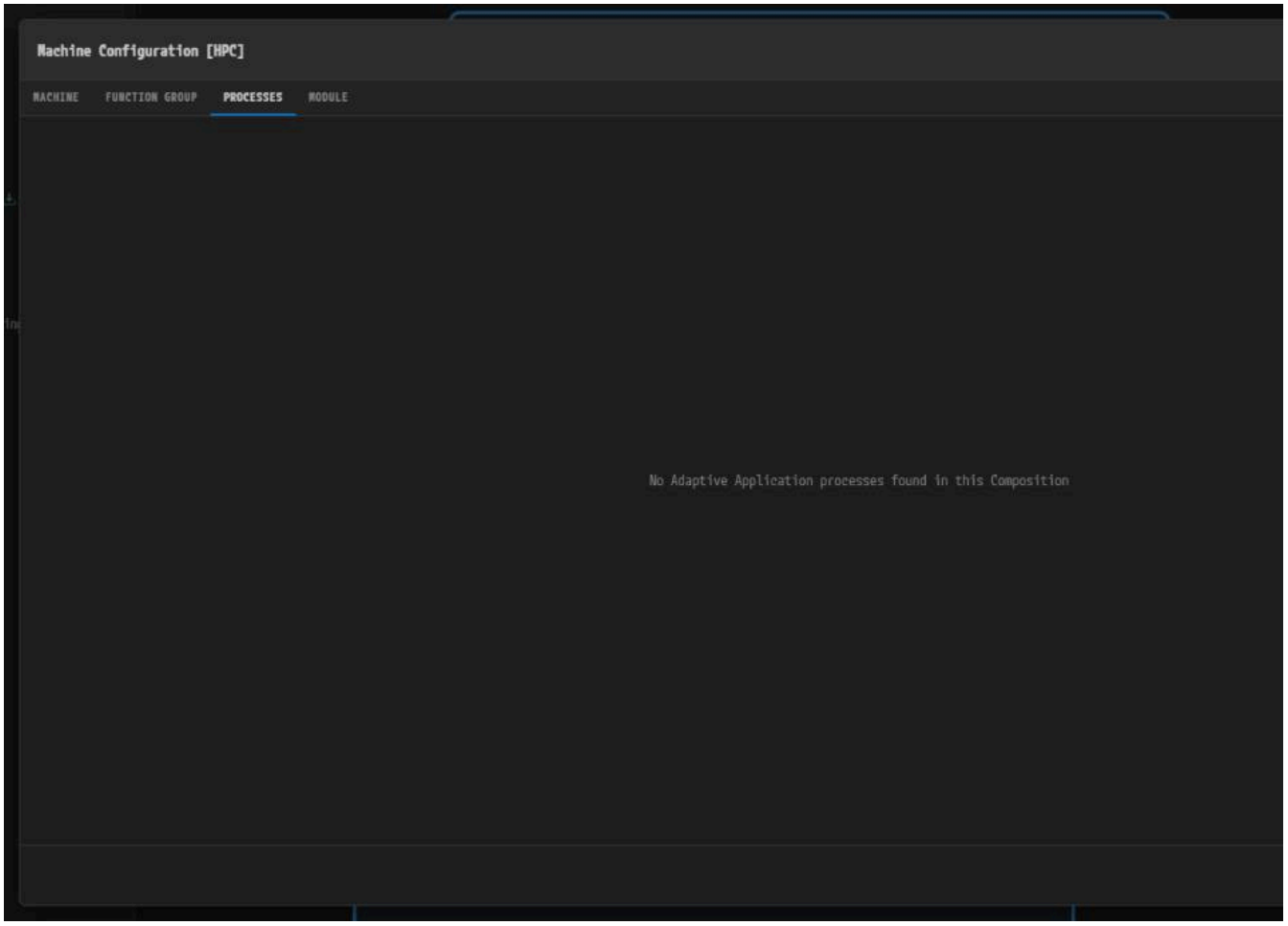


*그림 1-7. 비어 있는 **PROCESSES** 탭. 프로세스는 Executable 과 Process에서 생긴다.*

MODULE 탭

1. **MODULE** 탭을 클릭한다. **MODULES** 목록에 **Log and Trace** 부터 모듈 11개가 있다.

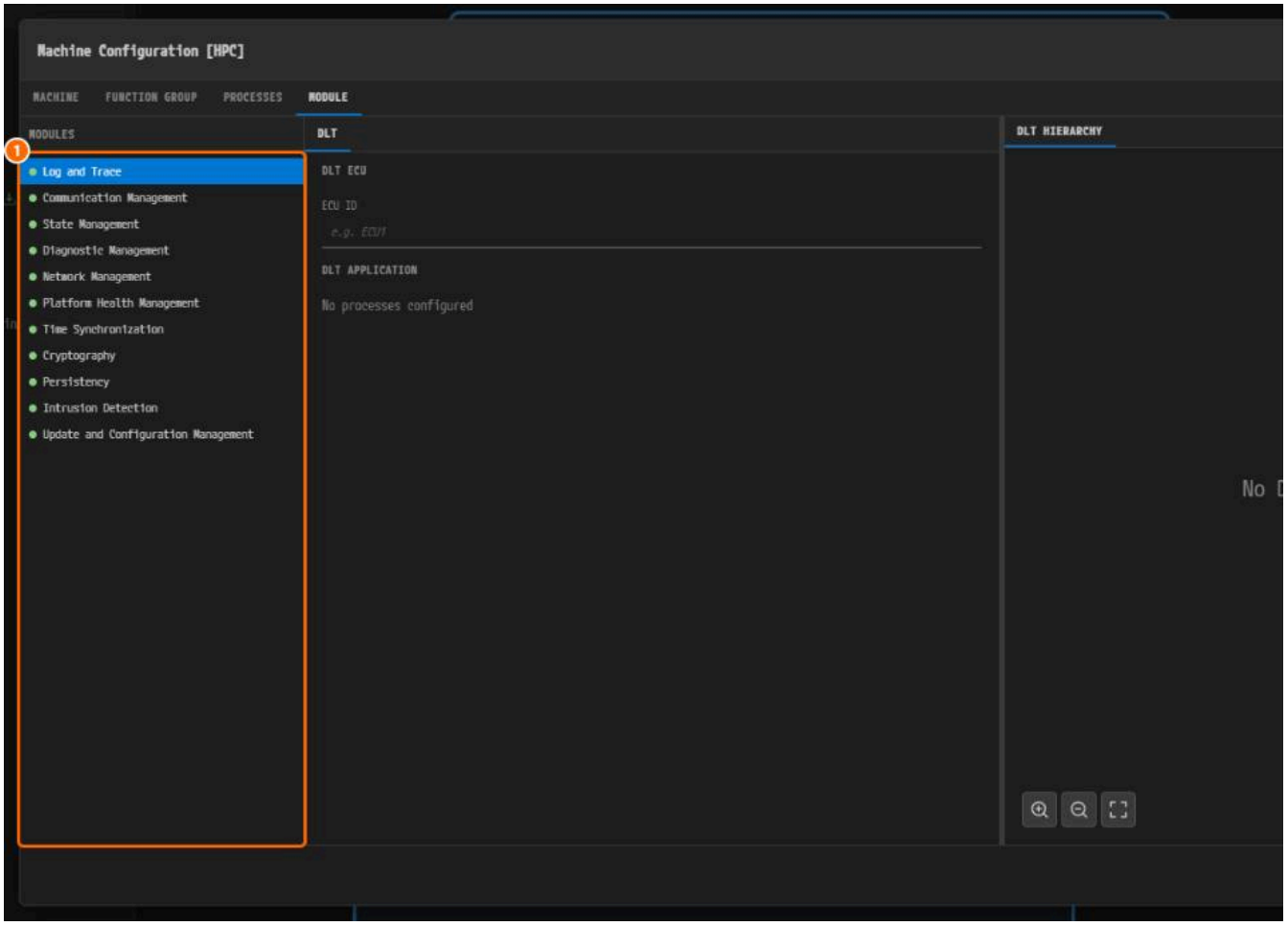


*그림 1-8. ① **MODULES** 목록. 초록 점은 모듈이 켜져 있다는 뜻이다.*

저장하고 닫기

1. 다이얼로그 안에서 `Ctrl+S` 를 누른다.
2. **Close** 를 클릭하고 **PROJECT** > **Save All** 을 클릭한다.

> **주의.** **Close** 만 누르면 머신 파일이 만들어지지 않고 저장 확인도 뜨지 않는다. 반드시 다이얼로그 안에서 `Ctrl+S` 를 누른다.

검증 결과 보기

1. 푸터의 **V** 아이콘(왼쪽에서 다섯 번째)을 클릭해 **VALIDATION** 패널을 연다. 오류 1건이 보인다.

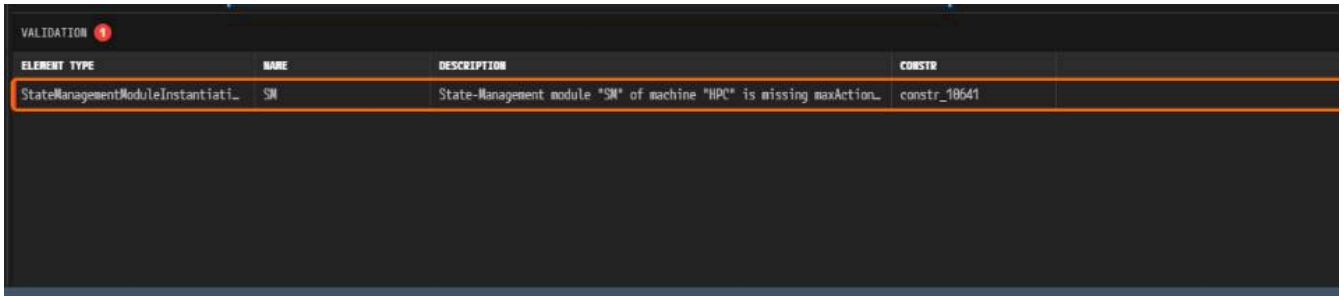


*그림 1-9. 행이 하나 있는 **VALIDATION** 패널.*

ELEMENT TYPE	NAME	CONSTR
`StateManagementModuleInstantiation`	`SM`	`constr_10641`

이 오류는 정상이다. 머신 설정에서 **Max Action List Duration** 을 채우면 사라진다.

왜 이렇게 설계하는가

- **VFB 에서 매핑한다. ECU/HPC Mapping** 에 모든 노드가 보이므로 같은 메뉴로 `ECU` 는 ECU 노드에, `HPC` 는 HPC 노드에 매핑한다.
- **바꾸지 않아도 다이얼로그를 저장한다.** 기본값이 기록되어 다음 장부터 그 위에 값을 채운다.

확인하기

- [ ] VFB 루트 캔버스에 `ECU [ECU]` 와 `HPC [HPC]` 가 있고 `HPC [HPC]` 에 머신 아이콘이 붙어 있다.
- [ ] `HPC` 의 **NAME** 과 **AS PROTOTYPE** 이 모두 `HPC` 이다.
- [ ] 프로젝트 폴더에 `design/Module/Machine/HPC.machine.io.model` 이 있다.
- [ ] **VALIDATION** 패널에 `constr_10641` 한 건만 보인다.

다음 단계

머신 설정에서 머신 값을 채워 이 오류를 없앤다.

머신 설정

이 장에서 할 일

이 장에서는 머신 `HPC` 에 OS, 코어 4개, DLT **ECU ID**, **Max Action List Duration** 을 채운다. 1장에 남은 검증 오류 `constr_10641` 이 사라진다.

시작하기 전에

- 컴포지션 HPC 와 머신을 마친 뒤 시작한다. **VALIDATION** 패널에 `constr_10641` 한 건이 보인다.

탭	항목	입력하는 값
MACHINE	**OS**	`POSIX`
MACHINE	**PROCESSOR**	`CPU` (기본값, 그대로 둔다)
MACHINE	**CORE** 표	+ 버튼 3회 → `Core1` / `1`, `Core2` / `2`, `Core3` / `3` 자동 추가
FUNCTION GROUP	`MachineFG`	바꾸지 않는다 (확인만)
MODULE > Log and Trace	**ECU ID**	`ECU1`
MODULE > State Management > ACTION ITEMS	**MAX ACTION LIST DURATION**	`1000` (단위 `s`)

입력란은 칸을 벗어나야 값이 확정되므로 입력한 뒤 `Tab` 을 누른다.

시작 상태 확인하기

1. 푸터의 **V** 아이콘을 클릭해 **VALIDATION** 패널을 연다. 행 하나가 보인다.

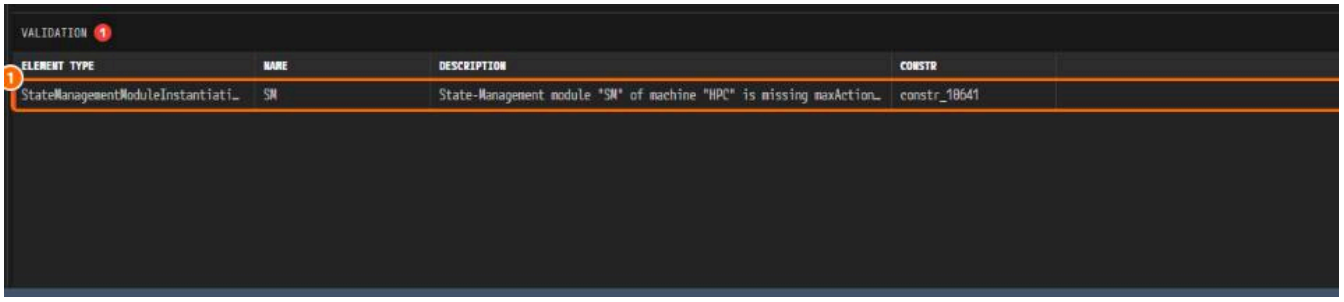


그림 2-1. ① `constr_10641` 행.

다이얼로그 열기

1. **V** 아이콘으로 **VALIDATION** 패널을 닫는다. 탭 `VehicleDesign:VFB` 가 열려 있으면 닫는다.
2. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 클릭한다.
3. `HPC [HPC]` 의 파란 머신 아이콘을 클릭한다. `ECU [ECU]` 의 초록 아이콘은 클릭하지 않는다.

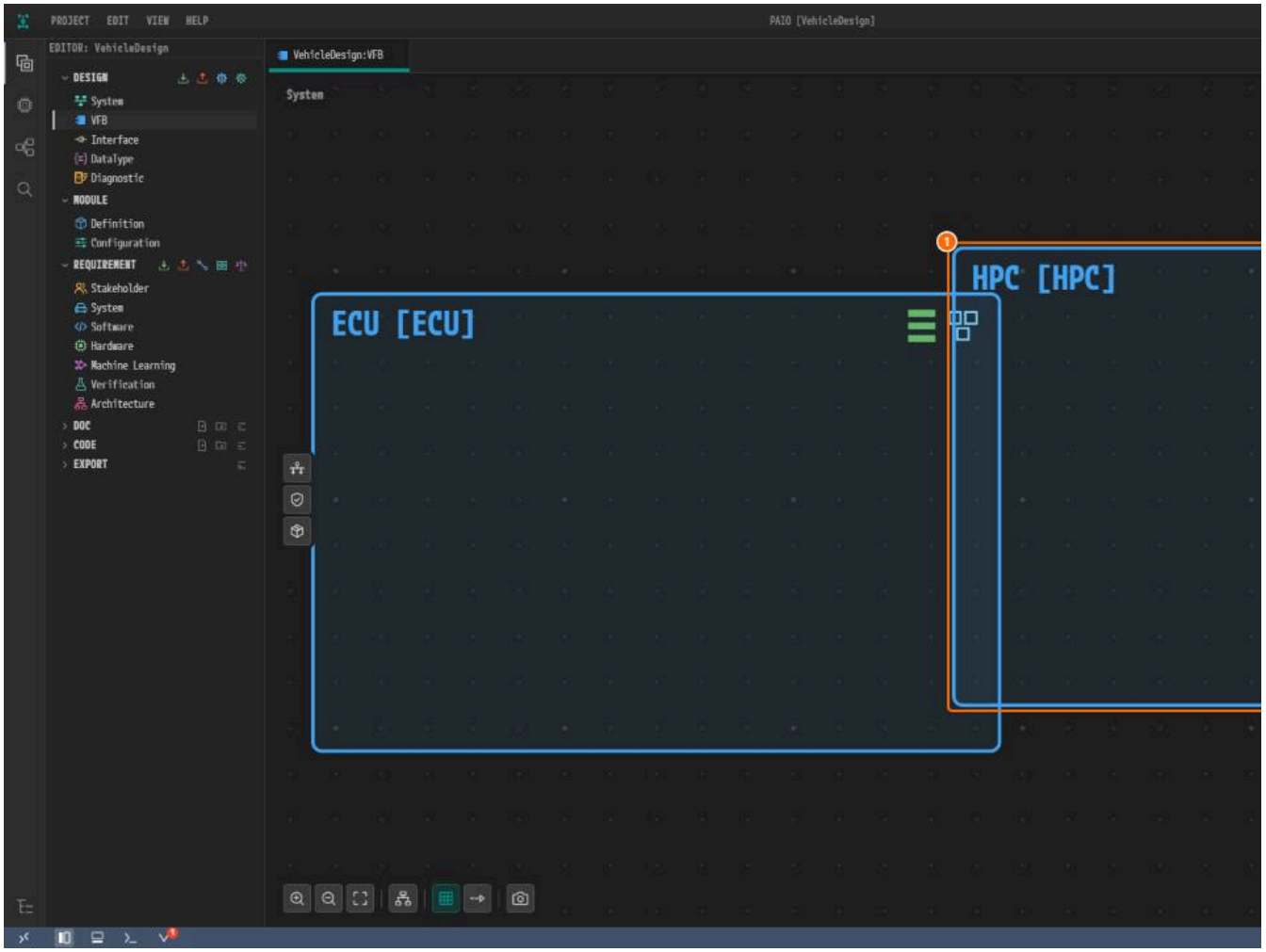


그림 2-2. ① `HPC [HPC]`, ② 머신 아이콘.

4. 다이얼로그 **Machine Configuration [HPC]** 가 **MACHINE** 탭으로 열린다.

MACHINE 탭 — OS 와 코어

OS 를 POSIX 로 입력하기

1. **OS** 에 `POSIX` 를 입력하고 `Tab` 을 누른다. 빨간 `Name is required` 가 사라진다.
2. **PROCESSOR** 가 `CPU` 인 것을 확인한다.

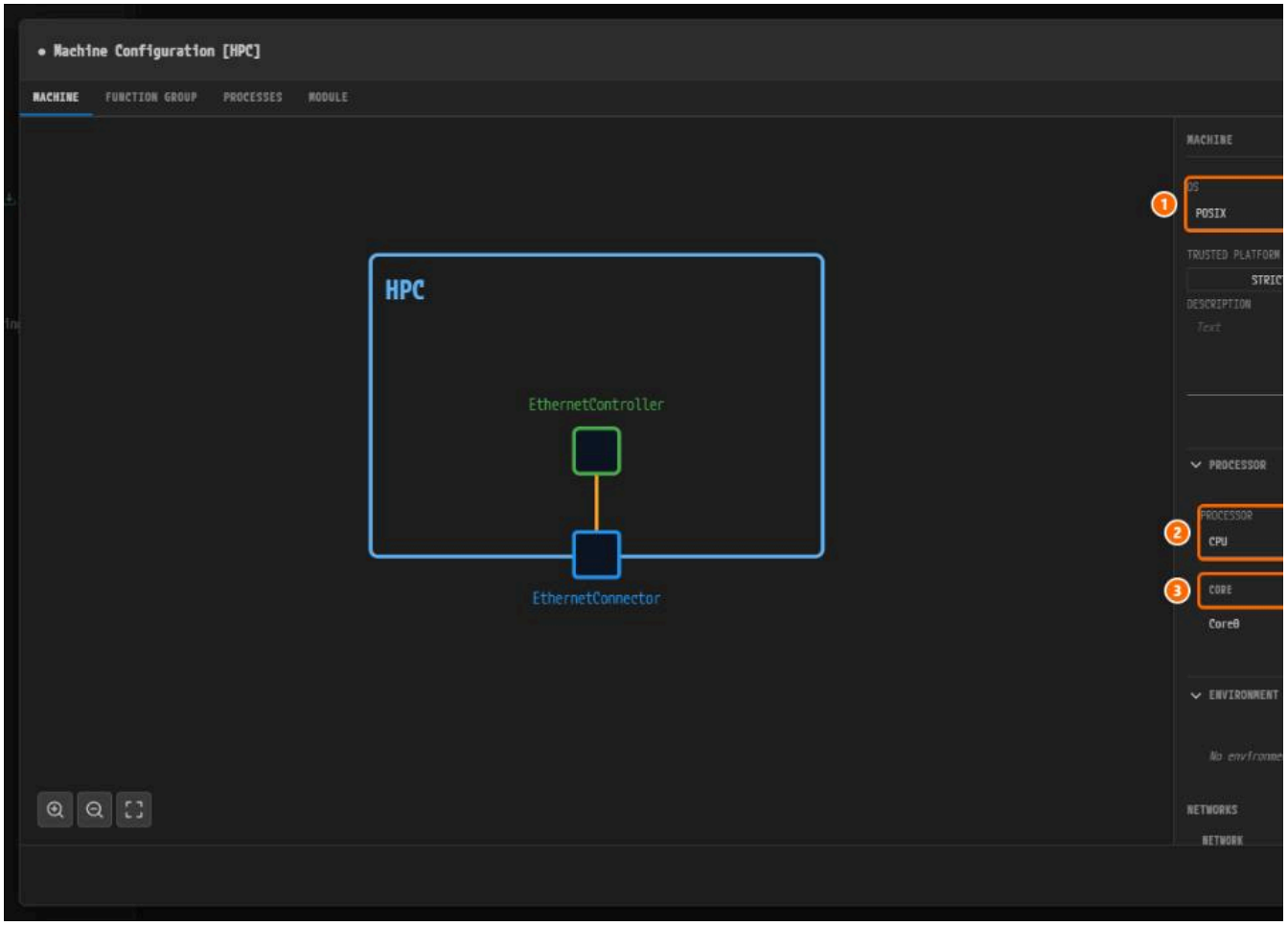


*그림 2-3. ① **OS**, ② **PROCESSOR**, ③ **+** 버튼이 있는 **CORE** 표.*

> **주의.** `Tab` 을 누르면 포커스가 **TRUSTED PLATFORM** 의 **STRICT** 버튼에 간다. Space 나 Enter 를 누르지 않는다. 값은 **NO SUPPORT** 로 남아야 한다.

코어 추가하기

1. **CORE** 표의 **+** 버튼(`Add Core`)을 클릭한다. `Core1` / `1` 행이 추가된다.

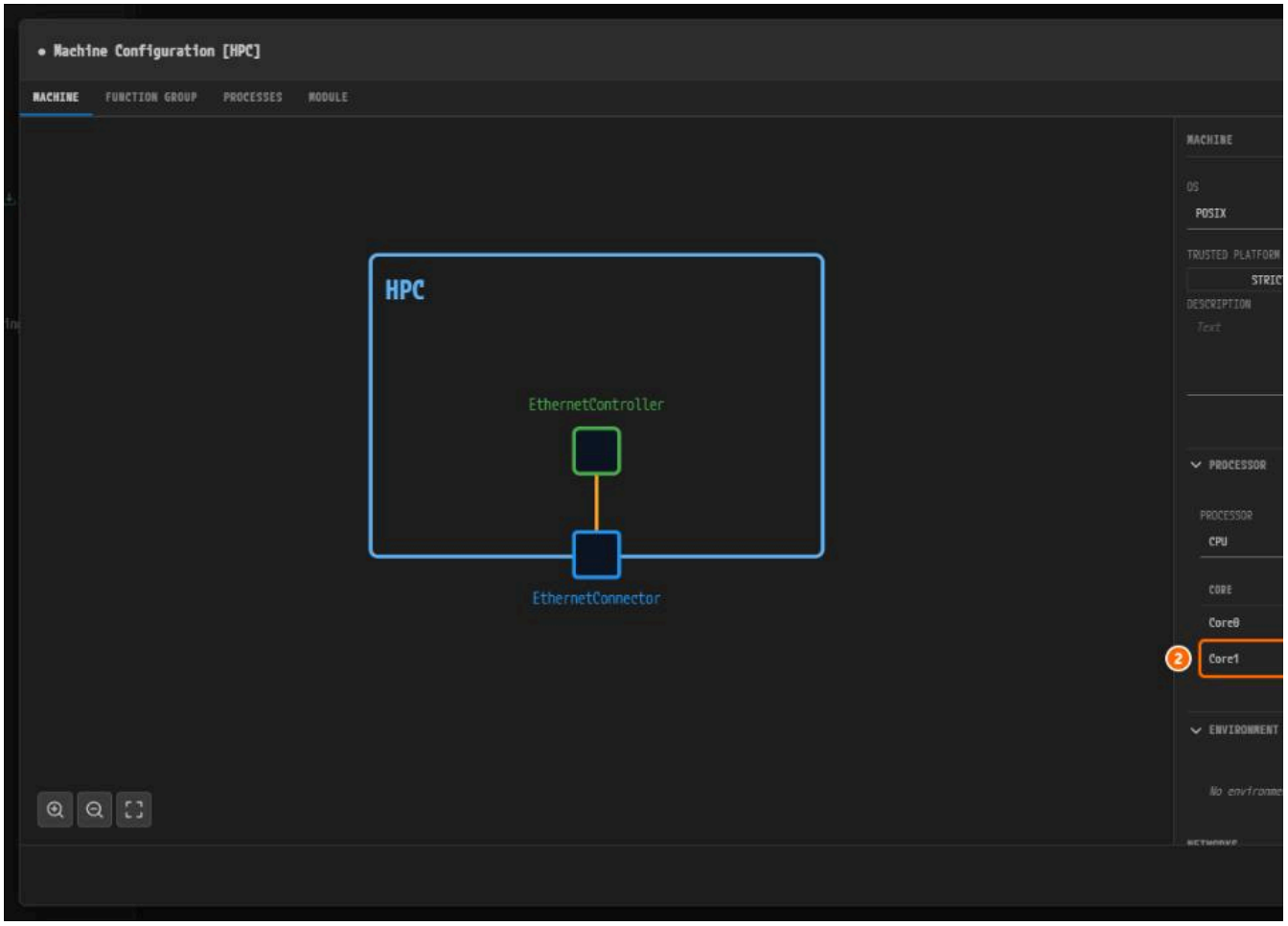


그림 2-4. ① +, ② 새 행 `Core1` / `1`.

2. + 를 두 번 더 클릭한다. `Core2` / `2`, `Core3` / `3` 이 추가된다.

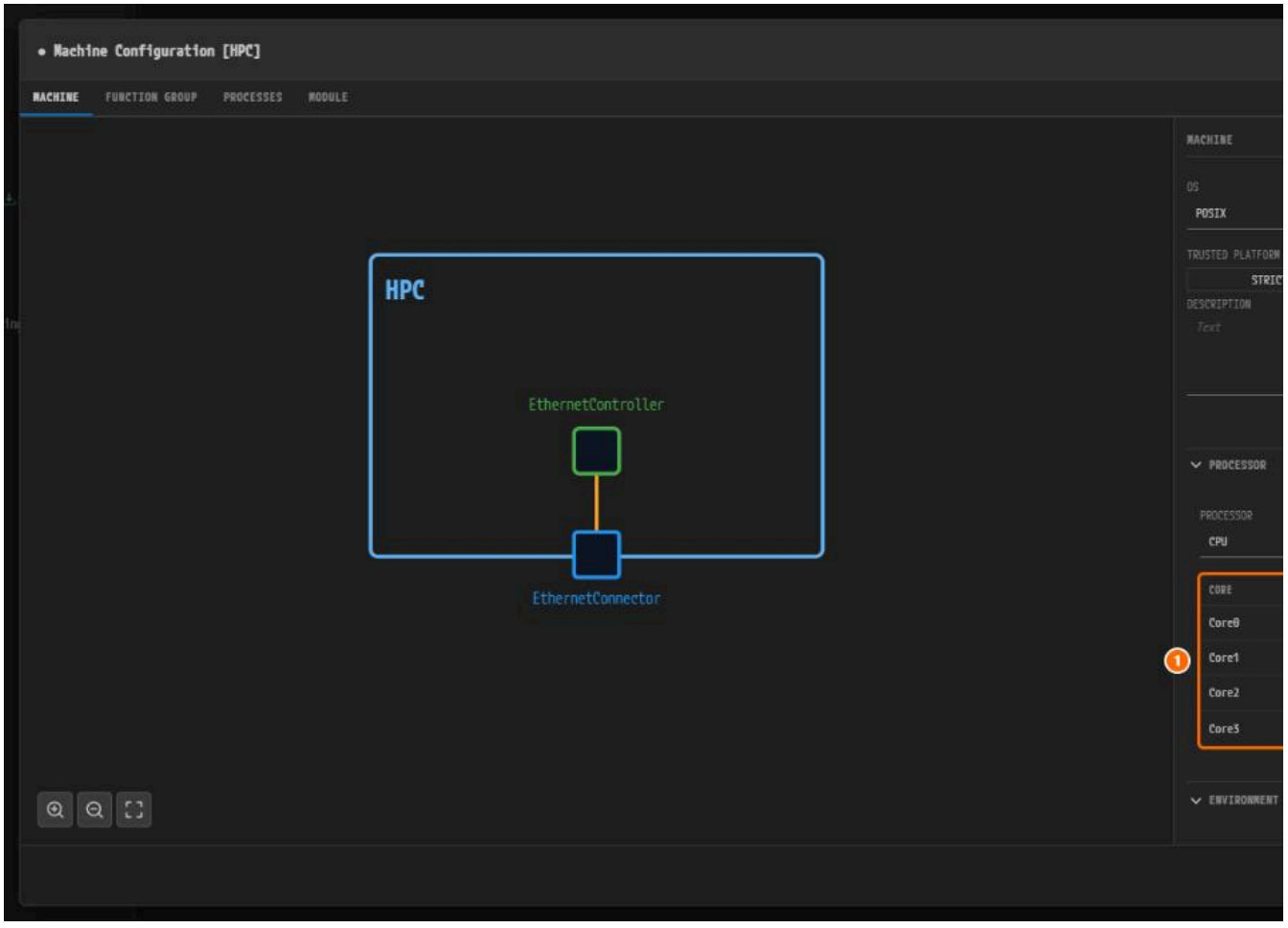


그림 2-5. ① 코어 4개 `Core0` ~ `Core3`.

FUNCTION GROUP 탭 — MachineFG 확인하기

1. **FUNCTION GROUP** 탭을 클릭한다. 항목이 `MachineFG` 하나이고 **GROUP NAME** 이 `MachineState`, **INITIAL MODE** 가 `Off` 인 것을 확인한다. 아무것도 바꾸지 않는다.

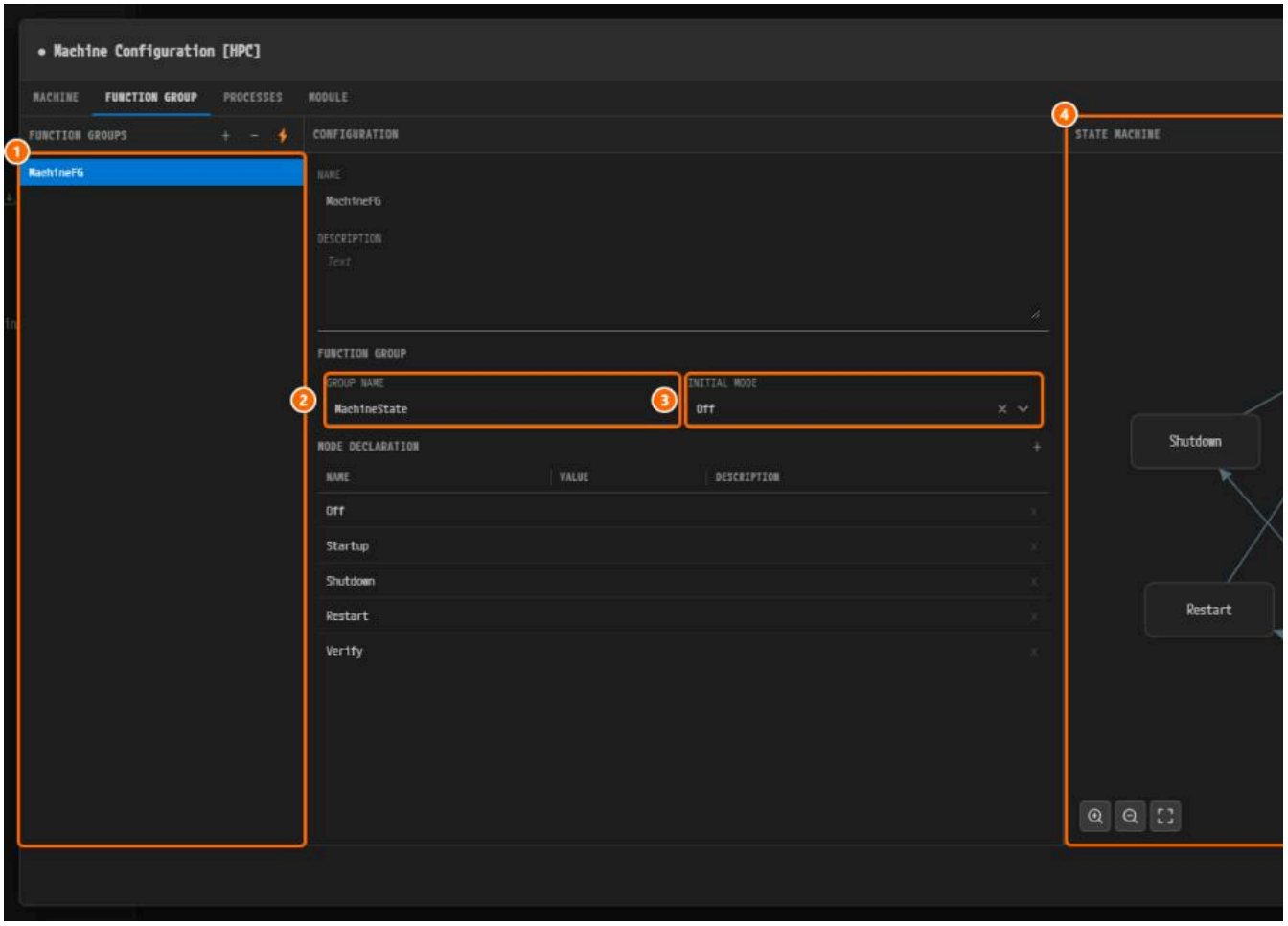


*그림 2-6. ① `MachineFG`, ② **GROUP NAME**, ③ **INITIAL MODE**, ④ **STATE MACHINE**.*

MODULE 탭 — Log and Trace 의 ECU ID

1. **MODULE** 탭을 클릭한다. **Log and Trace** 가 선택되어 있다.
2. **DLT ECU** 섹션의 **ECU ID** 에 `ECU1` 을 입력하고 `Tab` 을 누른다.

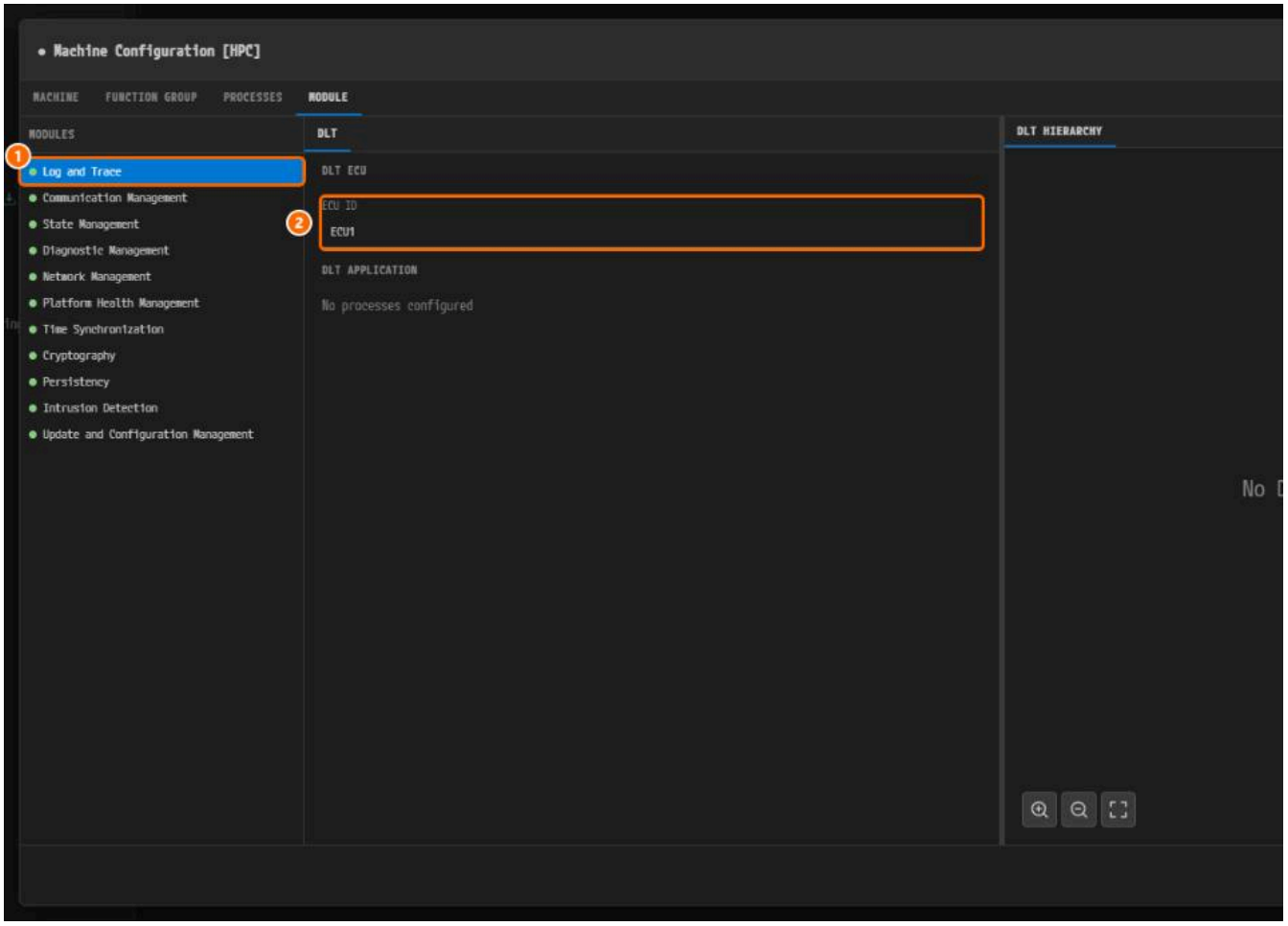


*그림 2-7. ① **Log and Trace**, ② **ECU ID**.*

MODULE 탭 — State Management 의 Max Action List Duration

1. **MODULES** 목록에서 **State Management** 를 클릭하고 서브탭 **ACTION ITEMS** 를 클릭한다.

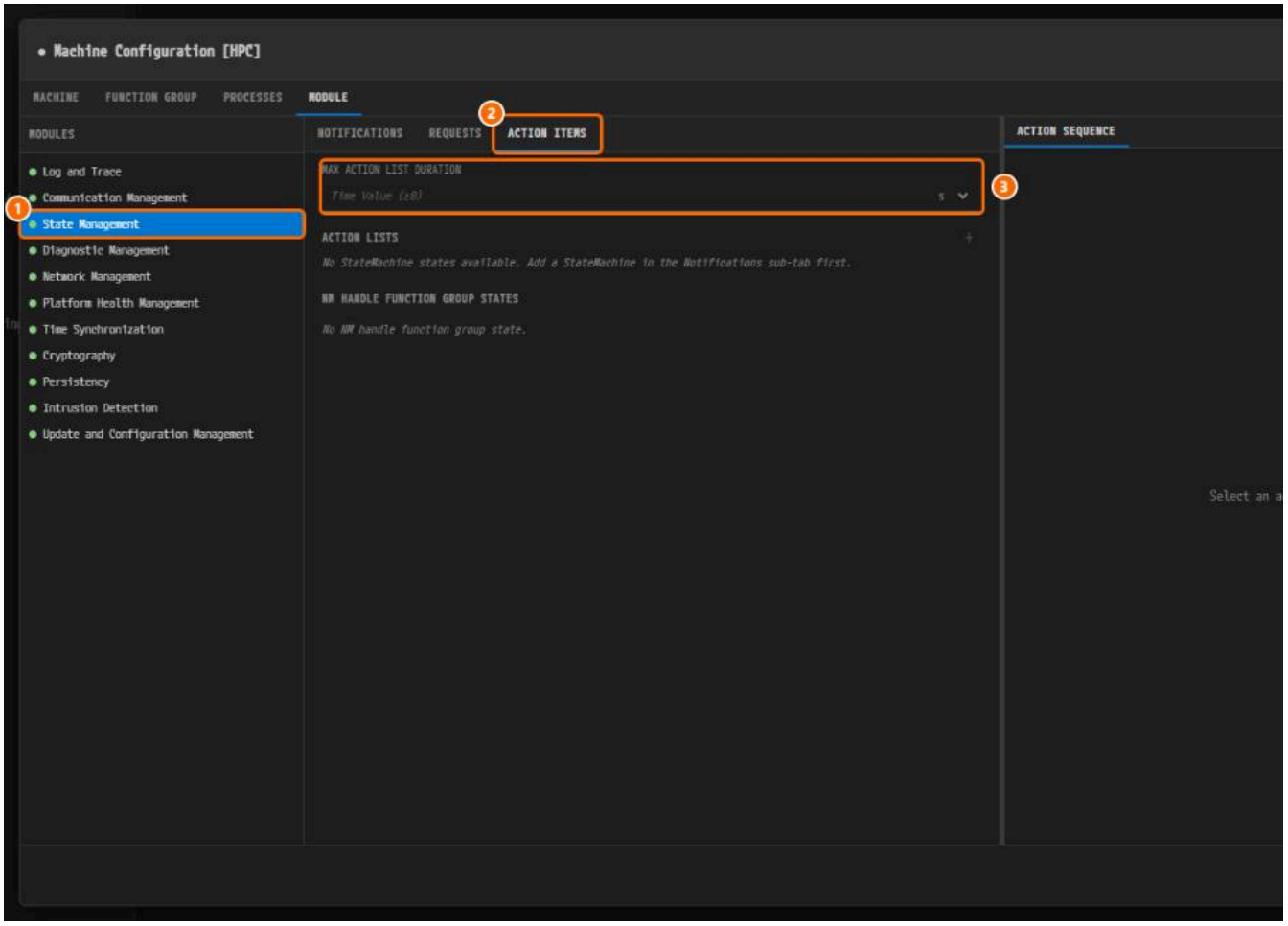


*그림 2-8. ① **State Management**, ② **ACTION ITEMS**, ③ **MAX ACTION LIST DURATION**.*

2. **MAX ACTION LIST DURATION** 에 `1000` 을 입력하고 `Tab` 을 누른다. 단위는 `s` 그대로 둔다.

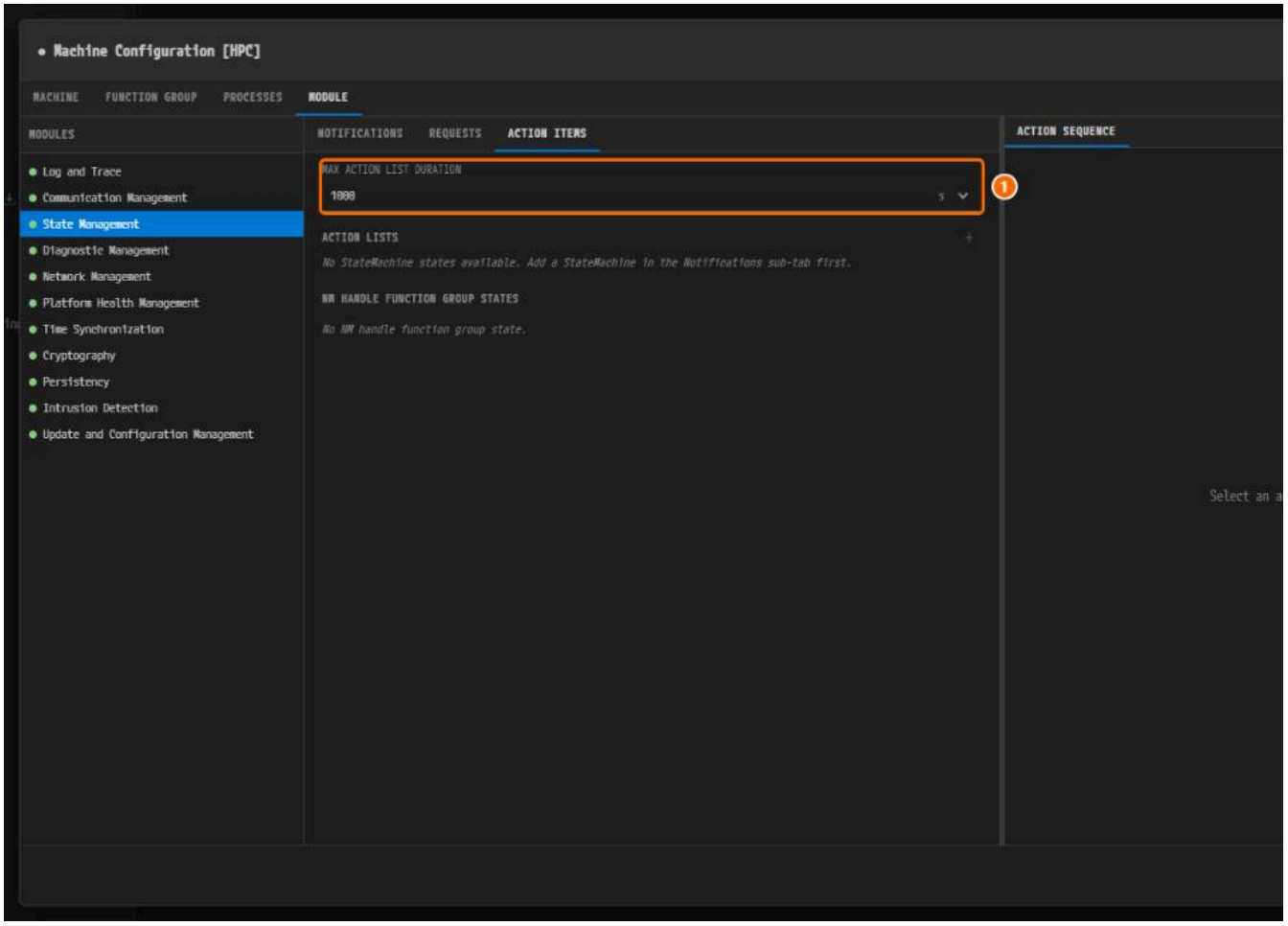


*그림 2-9. ① **MAX ACTION LIST DURATION** = `1000`, 단위 `s`.*

저장하고 닫기

1. 다이얼로그 안에서 `Ctrl+S` 를 누른다.
2. **Close** 를 클릭하고 **PROJECT** > **Save All** 을 클릭한다.

> **주의.** **Close** 만으로는 머신 파일이 저장되지 않는다. 저장 확인이 뜨면 **Save** 를 클릭한다.

검증 결과 보기

1. **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

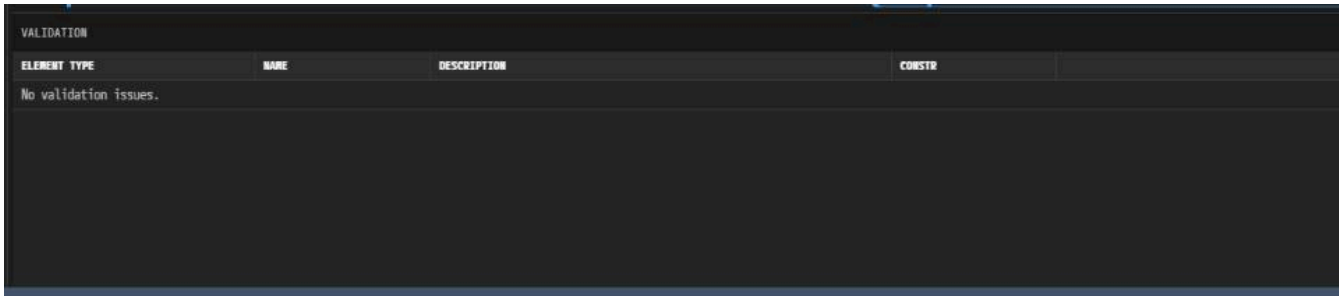


*그림 2-10. 저장 뒤의 **VALIDATION** 패널.*

> **참고.** V 아이콘의 툴팁에는 마지막 건수가 남는다. 패널을 기준으로 본다.

왜 이렇게 설계하는가

- `POSIX` 는 익스포트되는 ARXML 의 `OS-MODULE-INSTANTIATION` 이름이 된다.
- **코어** 의 이름과 ID 는 **+** 버튼이 매기므로 손으로 맞출 필요가 없다.
- `ECU1` 은 로그와 DLT에서 DLT 애플리케이션이 모이는 DLT ECU 이다.

- `Max Action List Duration` 은 PAIO 검증(`constr_10641`)이 요구한다. 달리 쓰이지 않으므로 `1000` 이면 된다.
- `MachineFG` 는 기본값 그대로 쓴다. Executable 과 Process에서 프로세스를 그 상태에 묶는다.

확인하기

- [ ] **MACHINE** 탭에서 **OS** 가 `POSIX`, **TRUSTED PLATFORM** 이 **NO SUPPORT** 이고 **CORE** 표에 `Core0` / `0` ~ `Core3` / `3` 이 있다.
- [ ] **MODULE > Log and Trace** 의 **ECU ID** 가 `ECU1` 이다.
- [ ] **MODULE > State Management > ACTION ITEMS** 의 **MAX ACTION LIST DURATION** 이 `1000`, 단위 `s` 이다.
- [ ] 수정됨 표시가 남은 탭이 없다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

토폴로지 — SOME/IP 디스커버리 포트와 UDP 엔드포인트에서 커넥터 `EthernetConnector` 에 SOME/IP 디스커버리 포트와 UDP 엔드포인트를 추가한다.

토폴로지 — SOME/IP 디스커버리 포트와 UDP 엔드포인트

이 장에서 할 일

이 장에서는 노드 `HPC` 의 커넥터 `EthernetConnector` 에 SOME/IP 에 필요한 서비스 디스커버리 포트 `30490` 과 UDP 엔드포인트 세 개를 더한다.

시작하기 전에

- 머신 설정을 마친 뒤 시작한다. **VALIDATION** 패널에 `No validation issues.` 가 보인다.
- 커넥터에는 AUTOSAR System 튜토리얼에서 넣은 유니캐스트 주소 `192.168.0.11` 과 멀티캐스트 주소 `224.0.0.1` 이 이미 있다.

위치	항목	값
NETWORK ENDPOINT > MULTICAST CONFIG	**MULTICAST PORT**	`30490`
APPLICATION ENDPOINTS	카드 3개, **TYPE** = **UDP**, **PORT NUMBER**	`20001`, `20002`, `20003`

입력란은 칸을 벗어나야 값이 확정되므로 입력한 뒤 `Tab` 을 누른다.

System 편집기 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System** 을 클릭한다. 탭 `VehicleDesign:System` 이 열린다.
2. 캔버스 아래 도구 막대의 세 번째 버튼(맞춤)을 클릭한다.

Network Endpoint — SOME/IP 디스커버리 포트

1. 노드 `HPC` 의 아래 테두리에 있는 파란 사각형, 커넥터 `EthernetConnector` 를 클릭한다. **ETHERNET CONNECTOR** 패널이 열린다.
2. **NETWORK ENDPOINT > MULTICAST CONFIG** 의 **MULTICAST PORT** 에 `30490` 을 입력하고 `Tab` 을 누른다. 다른 칸은 그대로 둔다.

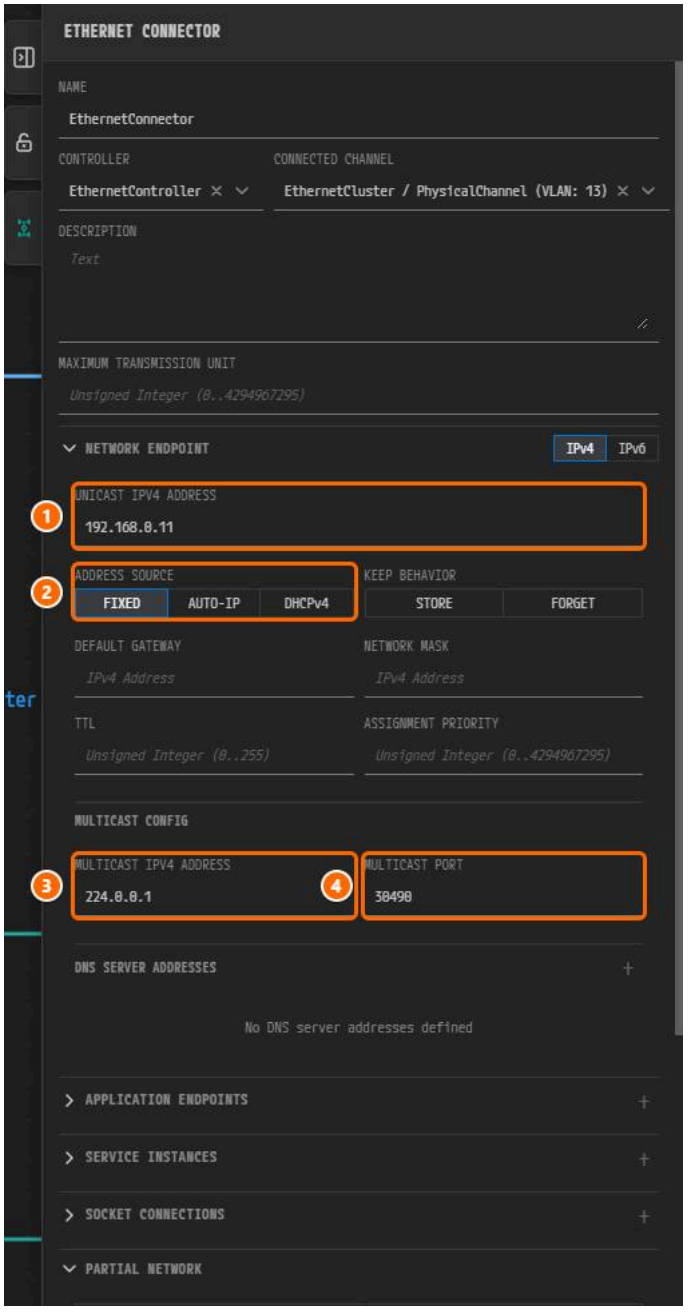


*그림 3-1. ① **UNICAST IPV4 ADDRESS**, ② **ADDRESS SOURCE**, ③ **MULTICAST IPV4 ADDRESS**, ④ **MULTICAST PORT**.*

Application Endpoints — UDP 포트 3개

먼저 포트 `20001` 을 따라간다.

첫 엔드포인트 추가하기

1. 머리글을 클릭해 **NETWORK ENDPOINT** 를 접고 **APPLICATION ENDPOINTS** 를 펼친다.
2. 머리글 오른쪽 **+**(`Add Application Endpoint`)를 클릭한다. 카드 `TCP EthernetConnector, 192.168.0.11:0` 이 생긴다.
3. 카드를 클릭해 펼치고 **TYPE** 에서 **UDP** 를 클릭한다.
4. **PORT NUMBER** 에 `20001` 을 입력하고 `Tab` 을 누른다. 카드가 `UDP EthernetConnector, 192.168.0.11:20001` 로 바뀐다.

나머지 엔드포인트 추가하기

나머지 두 개도 표대로 같은 방법으로 만든다. 이름은 자동으로 붙는다. TCP 엔드포인트는 만들지 않는다.

순서	TYPE	PORT NUMBER	요약 줄	자동으로 붙는 이름
1	**UDP**	`20001`	`UDP EthernetConnector, 192.168.0.11:20001`	`HPC_EthernetConnector_UDP_20001`
2	**UDP**	`20002`	`UDP EthernetConnector, 192.168.0.11:20002`	`HPC_EthernetConnector_UDP_20002`
3	**UDP**	`20003`	`UDP EthernetConnector, 192.168.0.11:20003`	`HPC_EthernetConnector_UDP_20003`

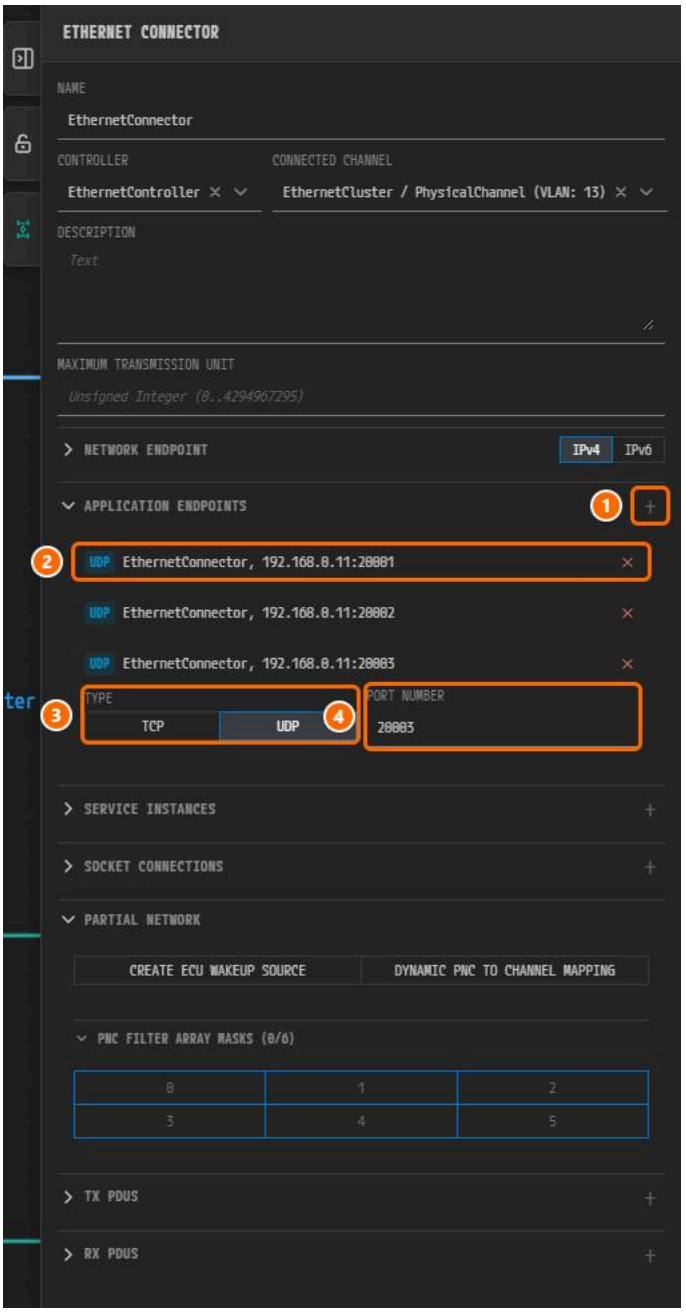


*그림 3-2. ① **+**, ② 첫 카드, ③ **TYPE** = **UDP**, ④ **PORT NUMBER** = `20003`.*

저장하기

1. 캔버스의 빈 곳을 클릭하고 `Ctrl+S` 를 누른다. 탭의 수정됨 표시 ● 가 사라진다.

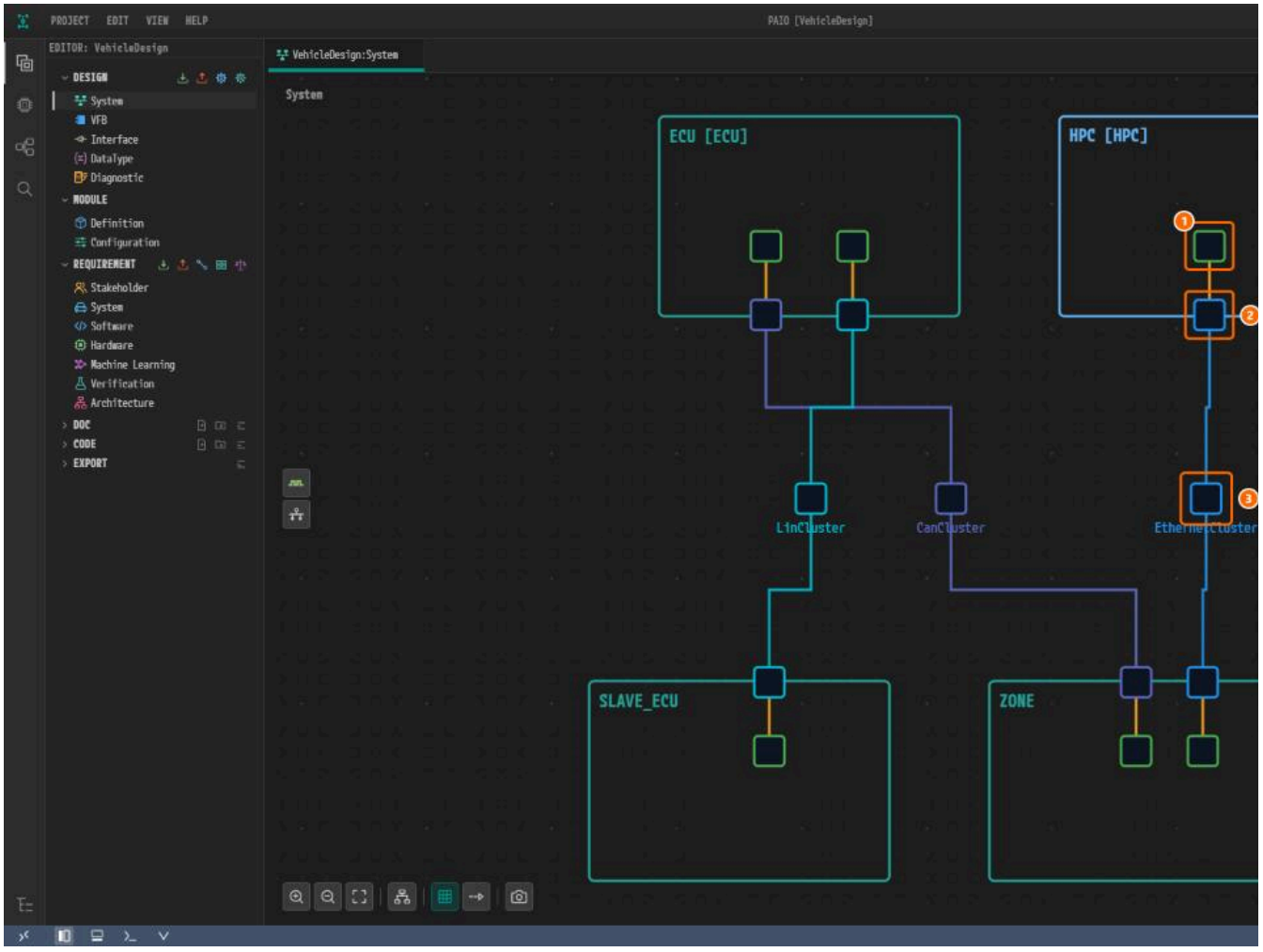


그림 3-3. ① `EthernetController`, ② `EthernetConnector`, ③ `EthernetCluster`.

검증 결과와 NM CLUSTER 탭 확인하기

검증 결과 확인하기

1. 푸터의 **V** 아이콘으로 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

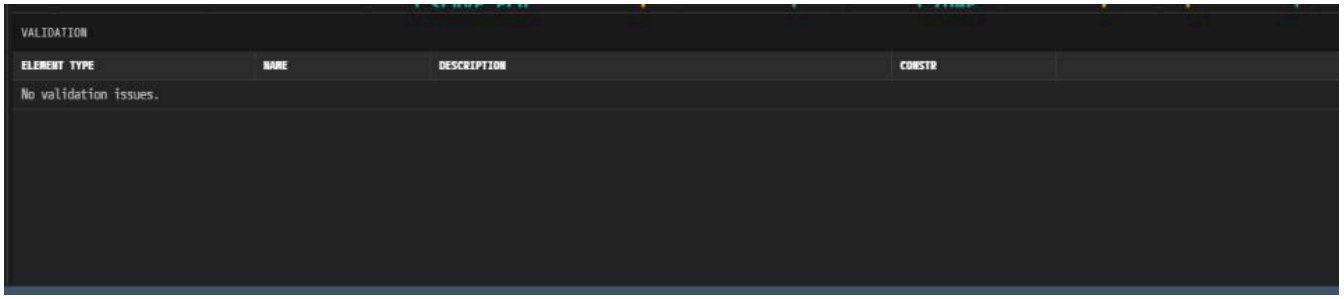


*그림 3-4. 저장한 뒤의 **VALIDATION** 패널.*

다이얼로그의 NM CLUSTER 탭 확인하기

1. **VALIDATION** 패널을 닫고 클러스터 `EthernetCluster` 를 더블클릭한다. 다이얼로그 **Communication Cluster Configuration [EthernetCluster]** 가 열린다.
2. **NM CLUSTER** 탭을 클릭한다. 이름 앞의 빈 점은 NM 이 꺼져 있다는 뜻이며, 탭은 흐린 미리보기다.

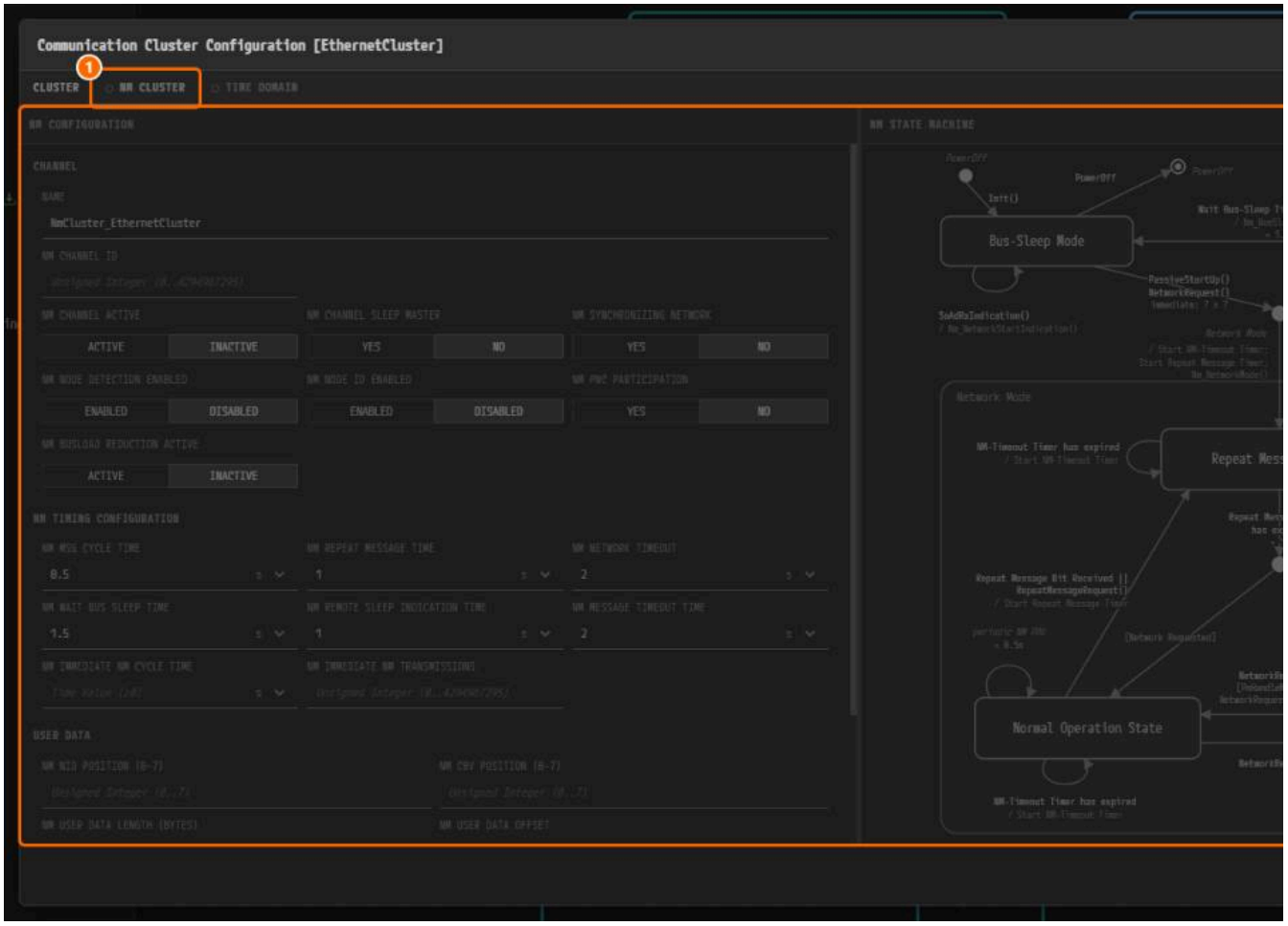


*그림 3-5. ① 빈 점이 붙은 **NM CLUSTER**, ② 흐린 미리보기.*

3. **Close** 를 클릭한다.

> **주의.** 탭의 오른쪽 클릭 메뉴에서 **Enable** 을 고르지 않는다. 이 튜토리얼은 NM 을 꺼 둔다.

왜 이렇게 설계하는가

- **SD 포트는 커넥터에서 나온다.** 익스포트의 `SOMEIP-SERVICE-DISCOVERY-PORT 30490` 은 **MULTICAST IPV4 ADDRESS** `224.0.0.1` 과 **MULTICAST PORT** `30490` 으
- **UDP 포트 세 개, TCP 없음.** 커넥터에 서비스 인스턴스 바인딩에서 `Actuator_Sensor` 는 `20001`, `Fusion_Sensor` 는 `20002`, `Provider_Sensor` 는 `20003` 을 쓴다.
- **NM 은 꺼 둔다.** 이 튜토리얼은 Network Management 를 쓰지 않는다.

확인하기

- [ ] `EthernetConnector` 의 **MULTICAST PORT** 가 `30490` 이다.
- [ ] **APPLICATION ENDPOINTS** 에 UDP 카드 `20001`, `20002`, `20003` 세 개가 있고 TCP 카드는 없다.
- [ ] 탭 `VehicleDesign:System` 에 수정됨 표시 ● 가 없다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

데이터 타입에서 C++ 구현 데이터 타입 `ByteArray` 와 `String` 을 만든다.

데이터 타입

이 장에서 할 일

이 장에서는 DataType 편집기에서 데이터 타입 `ByteArray` 와 `String` 을 만든다. 서비스 인터페이스에서 이벤트 `SensorData` 가 `ByteArray` 를 쓴다.

시작하기 전에

- 토폴로지 — SOME/IP 디스커버리 포트와 UDP 엔드포인트를 마친 뒤 시작한다. **VALIDATION** 패널은 `No validation issues.` 이다.
- 탭 `VehicleDesign:System` 은 닫아도 된다.

타입	CATEGORY	ELEMENT TYPE	MAX SIZE	C++ NAMESPACE
`ByteArray`	**DYNAMIC_ARRAY**	`uint8`	`1`	`aa::type`
`String`	**STRING**	(칸 없음)	`256`	`aa::type`

DESCRIPTION 은 비워 둔다. 입력란은 칸에서 빠져나와야 값이 확정되므로 입력한 뒤 `Tab` 을 누른다.

DataType 편집기 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **DataType** 을 누른다. 탭 `VehicleDesign:DataType` 이 열린다.
2. **DATATYPE** 목록 맨 위에는 기울임체 내장 타입 13개(`[PRIMITIVE]` 11개, `[POINTER]` 2개)가 있고, 그 아래에 AUTOSAR CP 편의 타입 53개가 있다. 모두 그대로 둔다. `uin` 칠 수 없다.

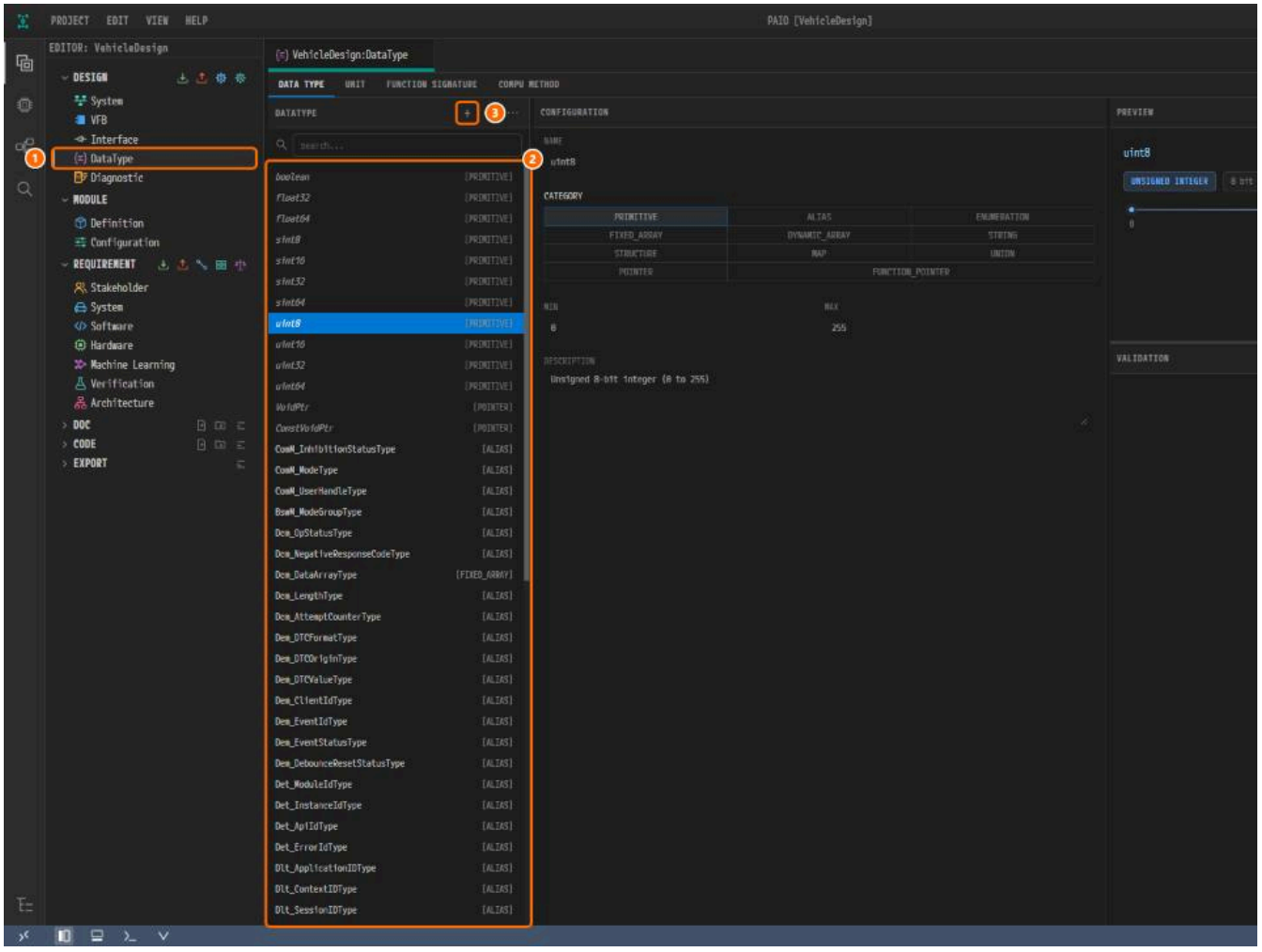


*그림 4-1. ① **DESIGN** 의 **DataType**, ② **DATATYPE** 목록(`uint8` 선택), ③ **+** 버튼.*

ByteArray 만들기 — DYNAMIC_ARRAY

1. 목록 머리글의 **+** 를 누른다. 목록 맨 아래에 새 항목 `DataType66` 이 생기고 선택된다. 보이지 않으면 아래로 스크롤한다.

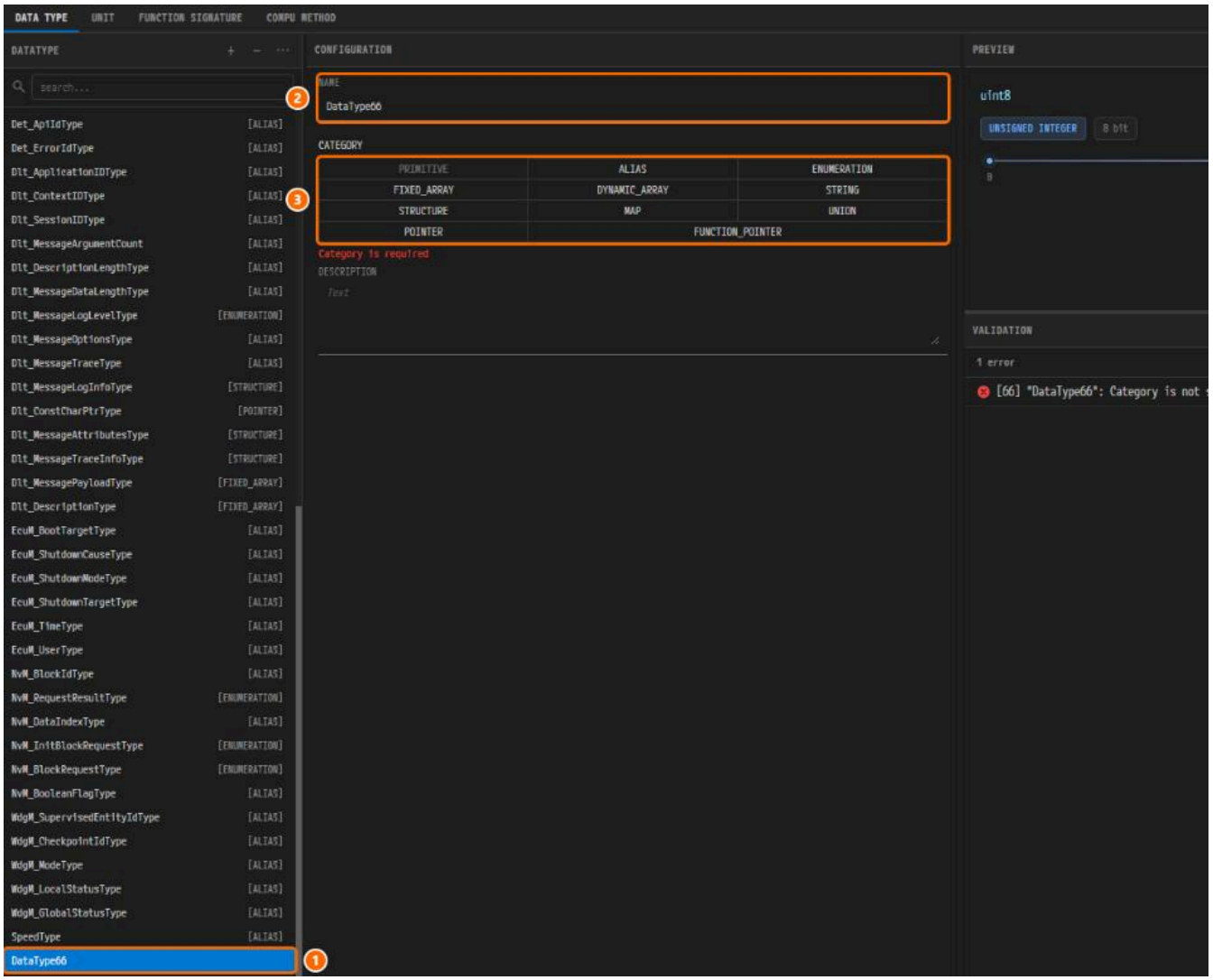


*그림 4-2. ① `DataType66`, ② **NAME**, ③ **CATEGORY**.*

2. **NAME** 을 `ByteArray` 로 바꾸고 `Tab` 을 누른다.

3. **CATEGORY** 에서 **DYNAMIC_ARRAY** 를 누른다. **C++ NAMESPACE**, **ELEMENT TYPE**, **MAX SIZE** 칸이 나타난다. 요소 타입을 고를 때까지 편집기에 오류가 보인다.

4. **ELEMENT TYPE** 칸을 누르고 목록을 스크롤해 `uint8` 을 누른다.

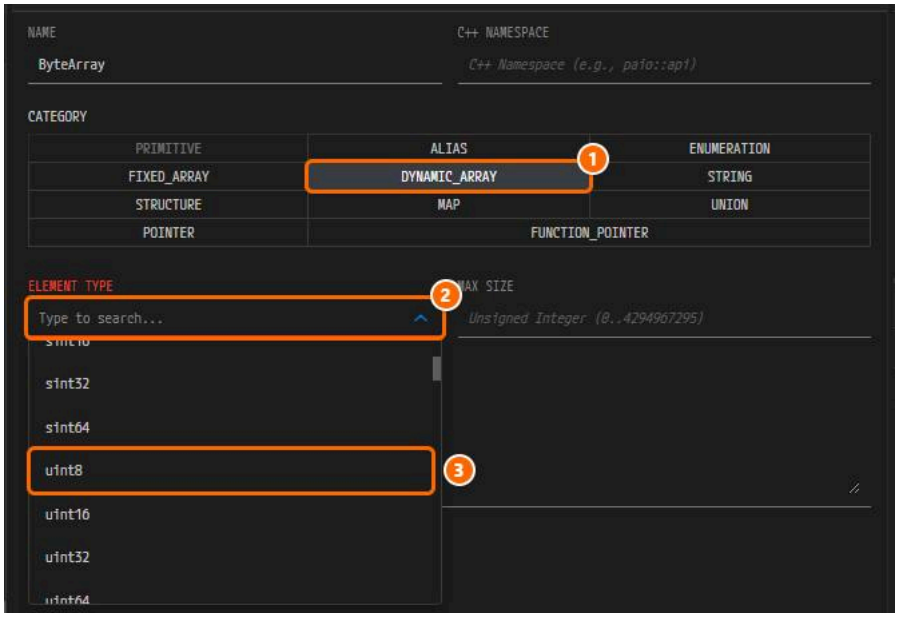


*그림 4-3. ① **DYNAMIC_ARRAY**, ② **ELEMENT TYPE**, ③ `uint8`.*

5. **MAX SIZE** 에 `1`, **C++ NAMESPACE** 에 `aa::type` 을 입력하고 각각 `Tab` 을 누른다. 행이 `ByteArray [DYNAMIC_ARRAY]` 로 바뀐다.

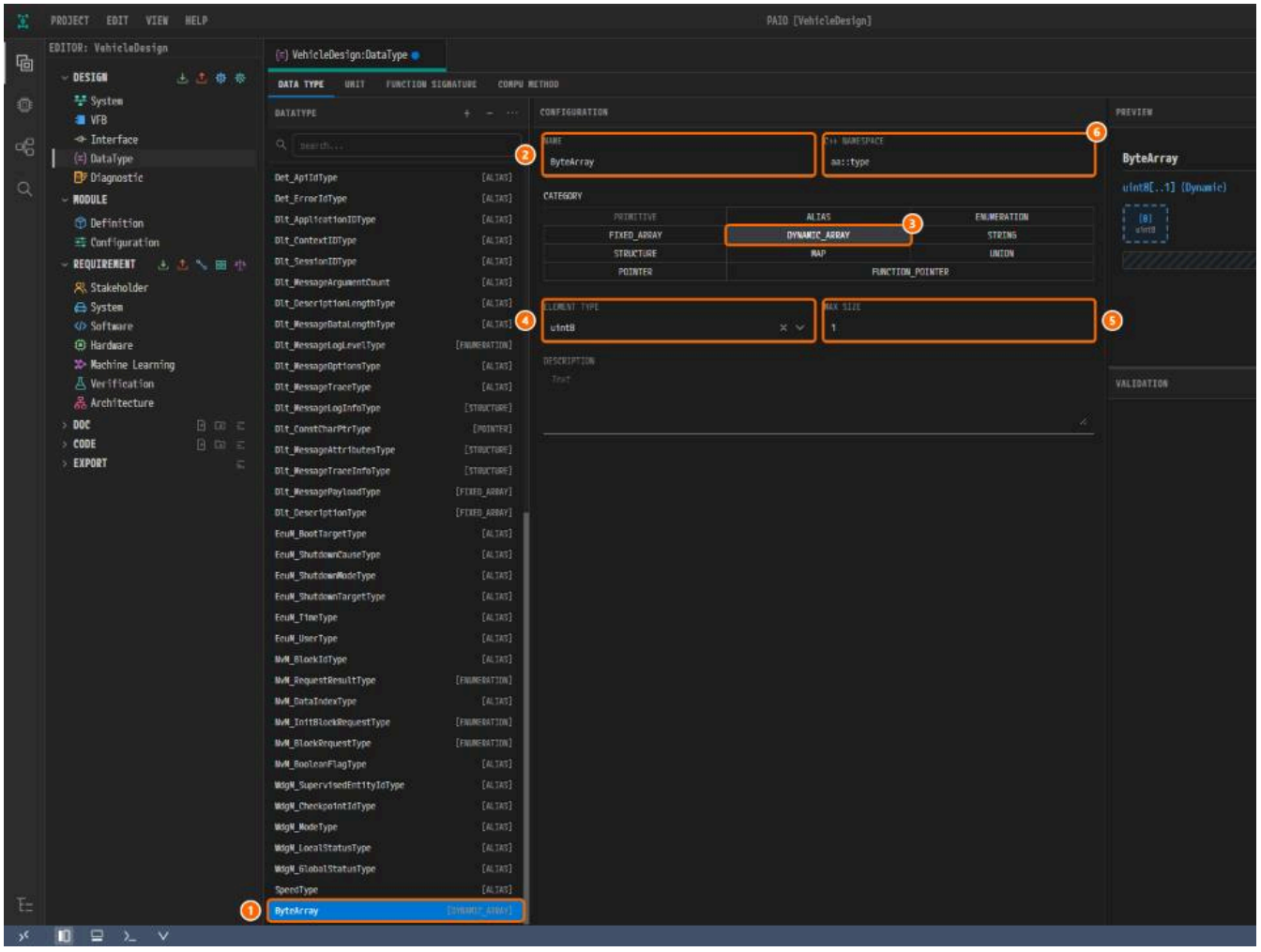


*그림 4-4. ① `ByteArray [DYNAMIC_ARRAY]`, ② **NAME**, ③ **CATEGORY**, ④ **ELEMENT TYPE**, ⑤ **MAX SIZE**, ⑥ **C++ NAMESPACE**.*

String 만들기 — STRING

1. **+** 를 다시 누른다. 새 항목 `DataType67` 의 **NAME** 을 `String` 으로 바꾸고 `Tab` 을 누른다.

2. **CATEGORY** 에서 **STRING** 을 누른다. 이 카테고리에는 **ELEMENT TYPE** 칸이 없다.

3. **MAX SIZE** 에 `256`, **C++ NAMESPACE** 에 `aa::type` 을 입력하고 각각 `Tab` 을 누른다. **MAX SIZE** 를 비워 두면 대신 기본값 `255` 가 적용된다.

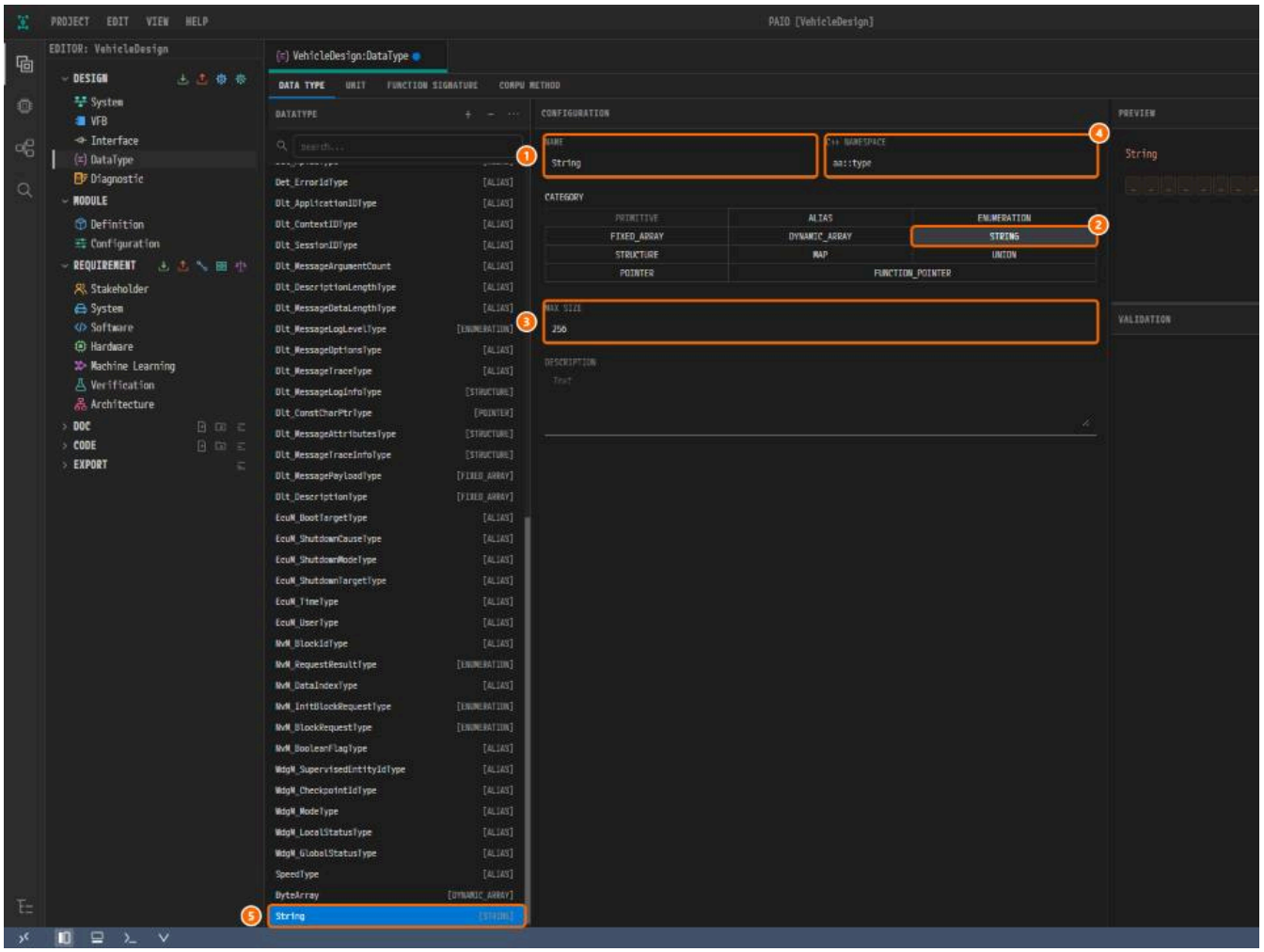


*그림 4-5. ① **NAME**, ② **CATEGORY**, ③ **MAX SIZE**, ④ **C++ NAMESPACE**, ⑤ `String [STRING]`.*

저장하기

1. `Ctrl+S` 를 누른다. 탭 `VehicleDesign:DataType` 의 수정 표시 ● 가 사라진다.

2. 목록에서 `ByteArray` 를 눌러 값이 남아 있는지 확인한다.

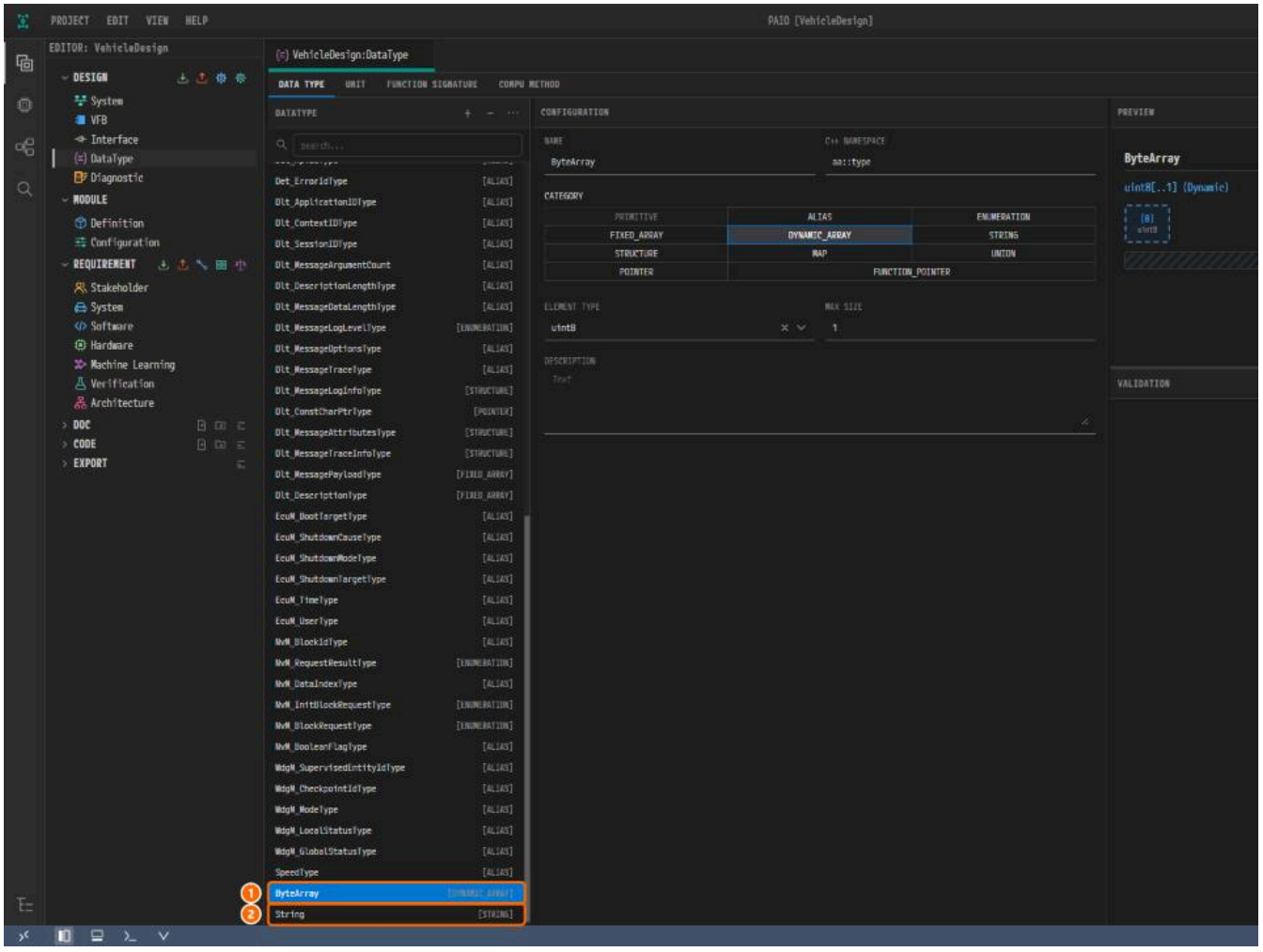


그림 4-6. ① `ByteArray [DYNAMIC_ARRAY]`, ② `String [STRING]`.

검증 결과 보기

1. 푸터의 **V** 아이콘을 눌러 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

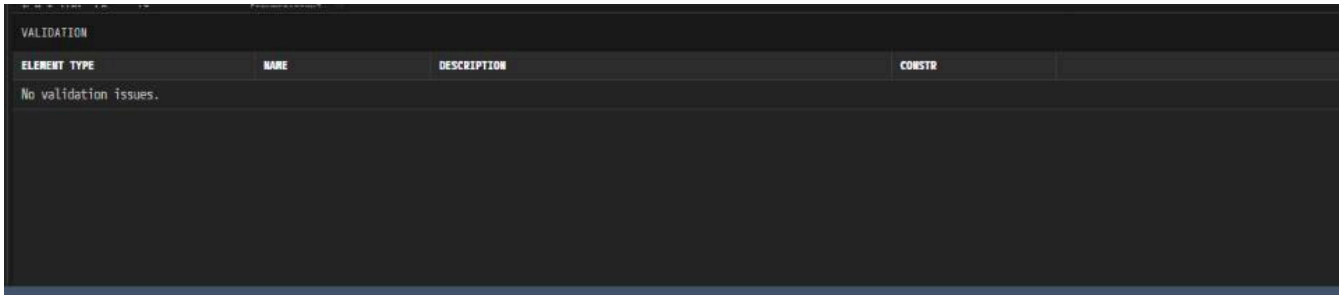


*그림 4-7. 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- `ByteArray` **를 먼저 만든다.** Interface 편집기의 타입 목록에는 DataType 편집기에 있는 타입만 나온다.
- **내장 타입은 입력하지 않는다.** 검증과 코드 생성의 Machine Extract 에서 익스포터가 `uint8_t` 같은 C++ 이름으로 써 준다.
- `String` **은 어떤 인터페이스도 쓰지 않는다.** **STRING** 타입의 입력 방법을 보이며, 이때 **MAX SIZE** 는 최대 텍스트 길이이다.
- **네임스페이스** `aa::type`**.** 익스포트의 `NAMESPACES` 가 된다. 비워 두면 네임스페이스가 쓰이지 않는다.

확인하기

- [ ] 목록이 `ByteArray [DYNAMIC_ARRAY]`, `String [STRING]` 으로 끝나고 항목이 모두 68개이다.
- [ ] `ByteArray` 의 **ELEMENT TYPE** 은 `uint8`, **MAX SIZE** 는 `1`, **C++ NAMESPACE** 는 `aa::type` 이다.

- [ ] `String` 의 **MAX SIZE** 는 `256`, **C++ NAMESPACE** 는 `aa::type` 이다.
- [ ] 탭에 수정 표시가 없고 **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

서비스 인터페이스에서 서비스 인터페이스 `Sensor` 를 만들고 그 이벤트에 타입 `ByteArray` 를 준다.

서비스 인터페이스

이 장에서 할 일

이 장에서는 이벤트 하나와 메서드 하나를 가진 서비스 인터페이스 `Sensor` 를 만든다. 뒤의 장에서 여기에 SOME/IP 배포를 붙이고 세 AA 의 포트에서 이 인터페이스를 쓴다.

시작하기 전에

- 데이터 타입을 마친 뒤 시작한다.
- 탭 `VehicleDesign:DataType` 은 닫아도 된다.

요소	NAME	그 밖의 값
서비스 인터페이스	`Sensor`	**C++ NAMESPACE** `sensor::service`, **MAJOR VERSION** `1`, **MINOR VERSION** `0` (기본값)
이벤트	`SensorData`	**TYPE** `ByteArray`
메서드	`Calculation`	**METHOD TYPE REQUEST AND RESPONSE** (기본값)

`Calculation` 의 인자 다섯 개는 다음 순서로 만든다.

#	DIRECTION	TYPE	NAME
1	**IN** (기본값)	`float32`	`front`
2	**IN** (기본값)	`float32`	`rear`
3	**IN** (기본값)	`float32`	`left`
4	**IN** (기본값)	`float32`	`right`
5	**OUT**	`float32`	`result`

DESCRIPTION 은 모두 비워 둔다. 이름을 입력한 뒤에는 `Enter`, 네임스페이스를 입력한 뒤에는 `Tab` 을 누른다.

Interface 편집기 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **Interface** 를 누른다. 탭 `VehicleDesign:Interface` 가 **SERVICE** 탭이 활성인 채로 열린다. **INTERFACES** 목록은 비어 있다.

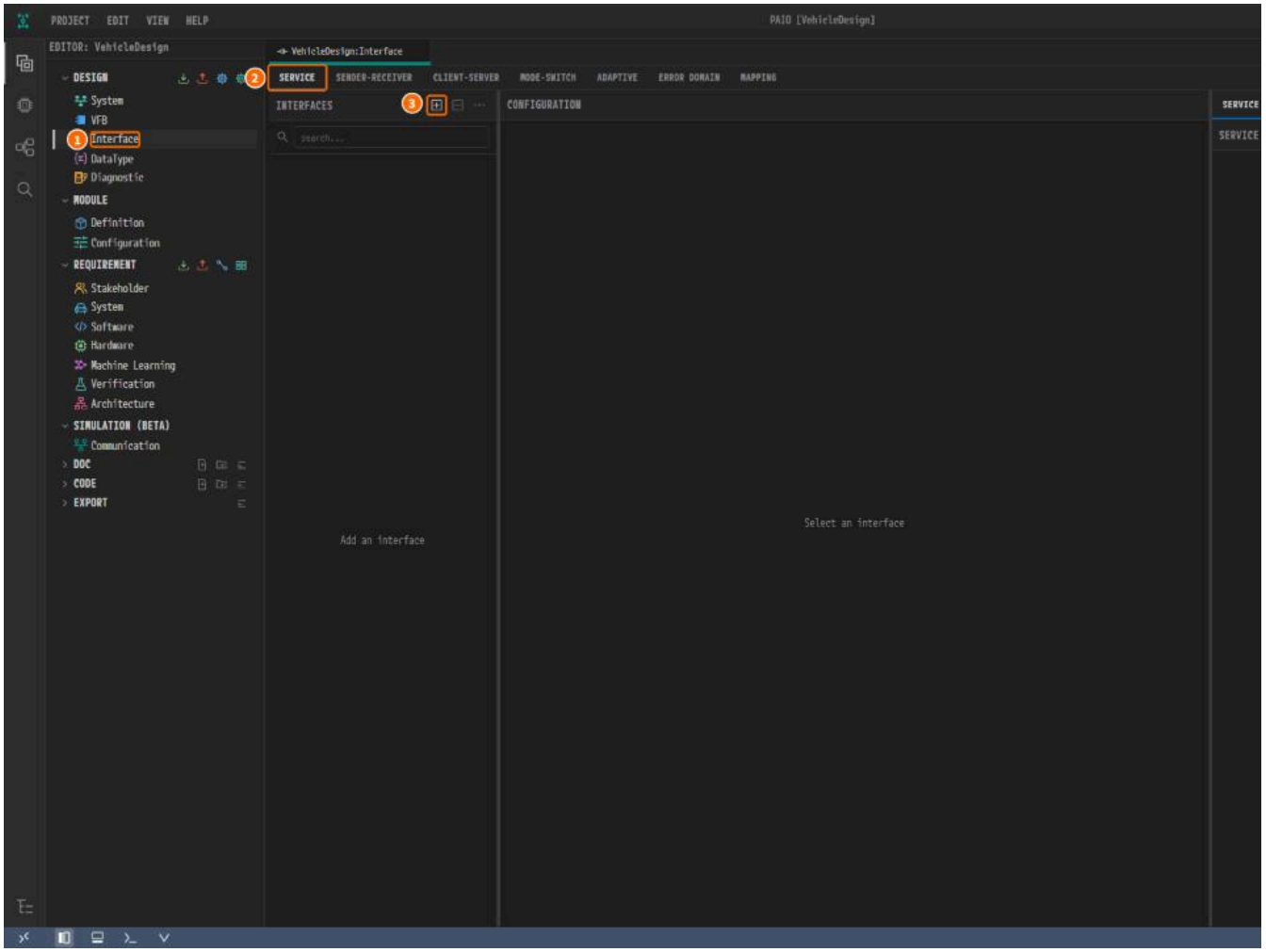


*그림 5-1. ① **DESIGN** 의 **Interface**, ② **SERVICE**, ③ **+** 버튼.*

서비스 인터페이스 Sensor 만들기

1. **INTERFACES** 머리글의 **+** 를 누른다. 새 인터페이스 `ServiceInterface_0` 이 생기고 선택된다.

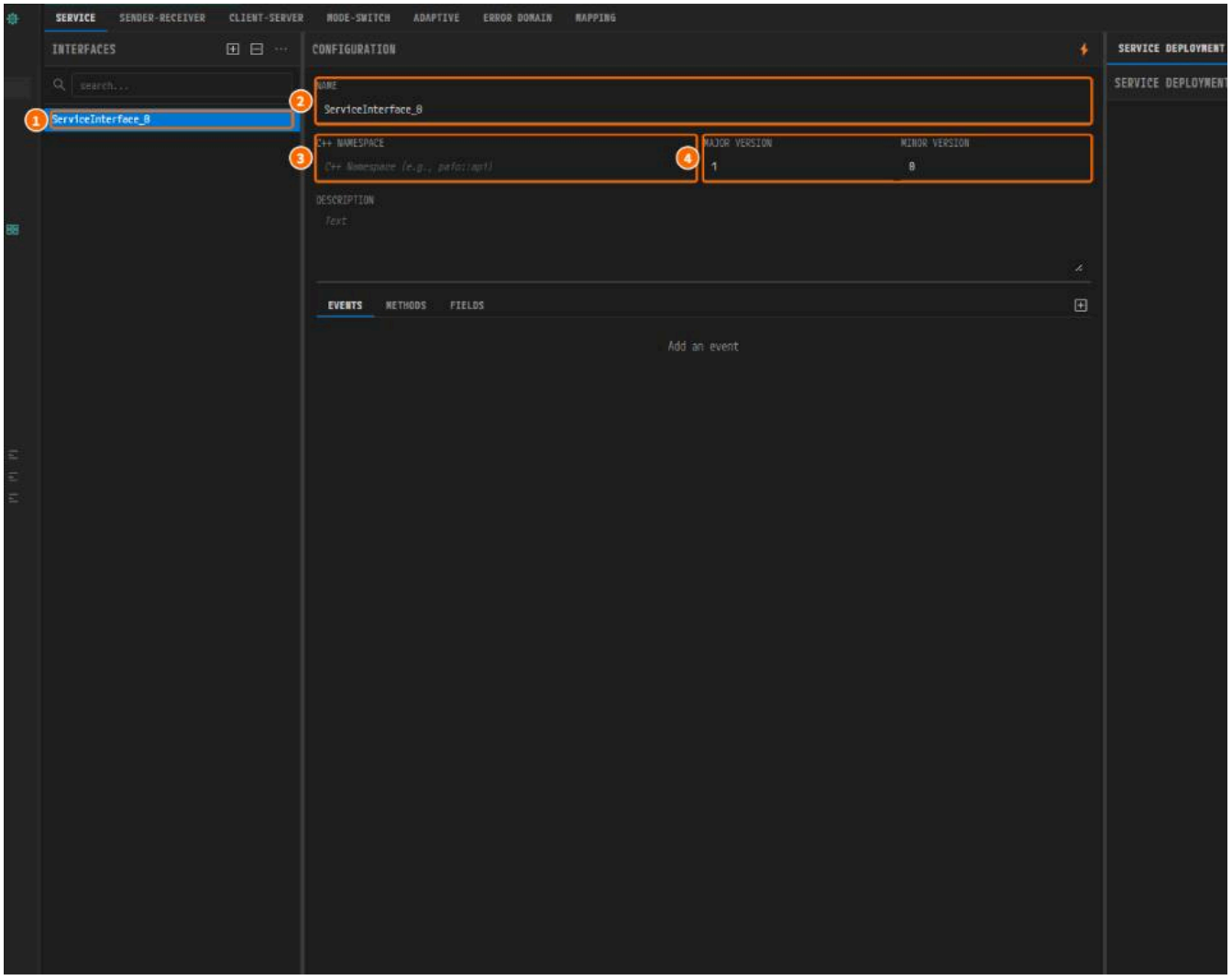


*그림 5-2. ① `ServiceInterface_0`, ② **NAME**, ③ **C++ NAMESPACE**, ④ **MAJOR VERSION** 과 **MINOR VERSION**.*

2. **NAME** 을 `Sensor` 로 바꾸고 `Enter` 를 누른다.

3. **C++ NAMESPACE** 에 `sensor::service` 를 입력하고 `Tab` 을 누른다. 버전 `1` 과 `0` 은 그대로 둔다.

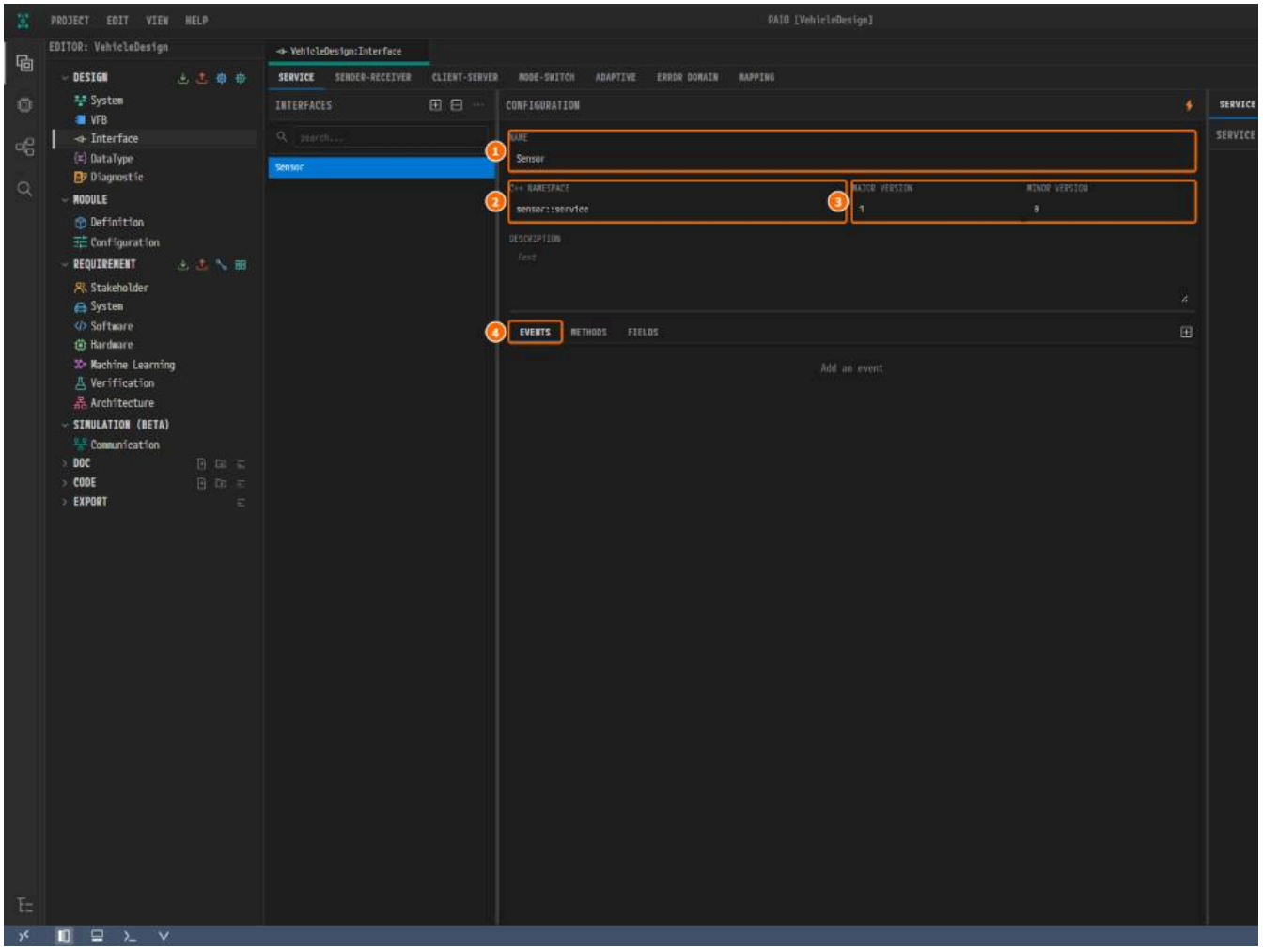


*그림 5-3. ① **NAME**, ② **C++ NAMESPACE**, ③ **MAJOR VERSION** 과 **MINOR VERSION**, ④ **EVENTS**.*

이벤트 SensorData 추가하기

1. **EVENTS** 서브탭이 활성인 상태에서 서브탭 줄 오른쪽 끝의 **+** 를 누른다. 타입이 `uint8` 인 행 `Event_0` 이 펼쳐진 채 생긴다.
2. **NAME** 을 `SensorData` 로 바꾸고 `Enter` 를 누른다.
3. **TYPE** 의 ⌄ 를 누른다. `uint64`, `BswM_ModeGroupType` 아래의 `ByteArray` 까지 스크롤해 누른다.

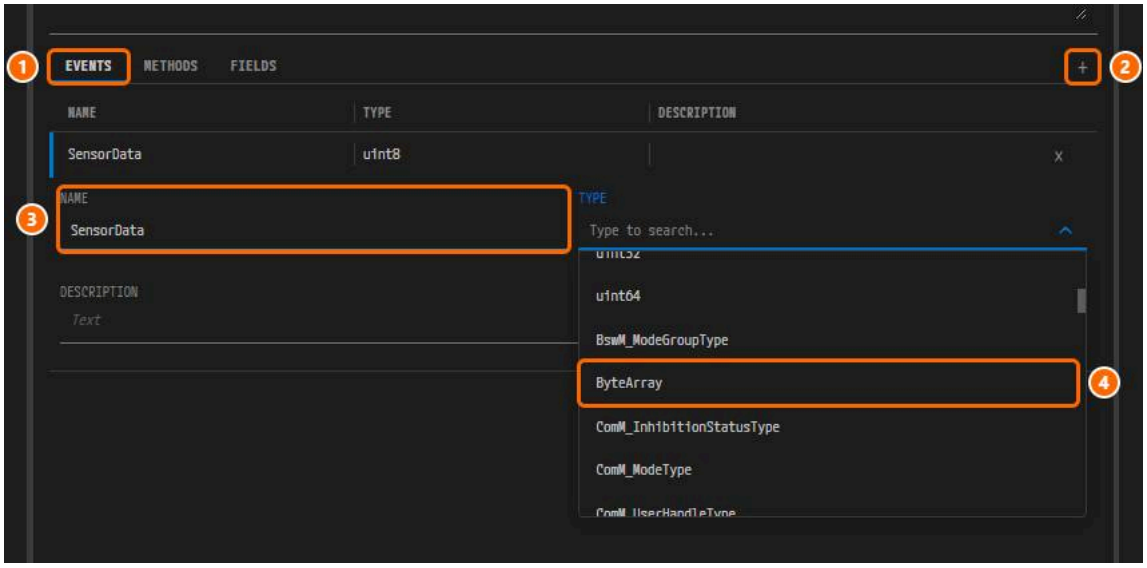


*그림 5-4. ① **EVENTS**, ② **+**, ③ **NAME**, ④ `ByteArray`.*

4. 행이 `SensorData` `ByteArray` 가 된다. 오른쪽 패널이 **DATA TYPE** 으로 바뀌어 `ByteArray` 를 보인다. 거기서는 아무것도 바꾸지 않는다.

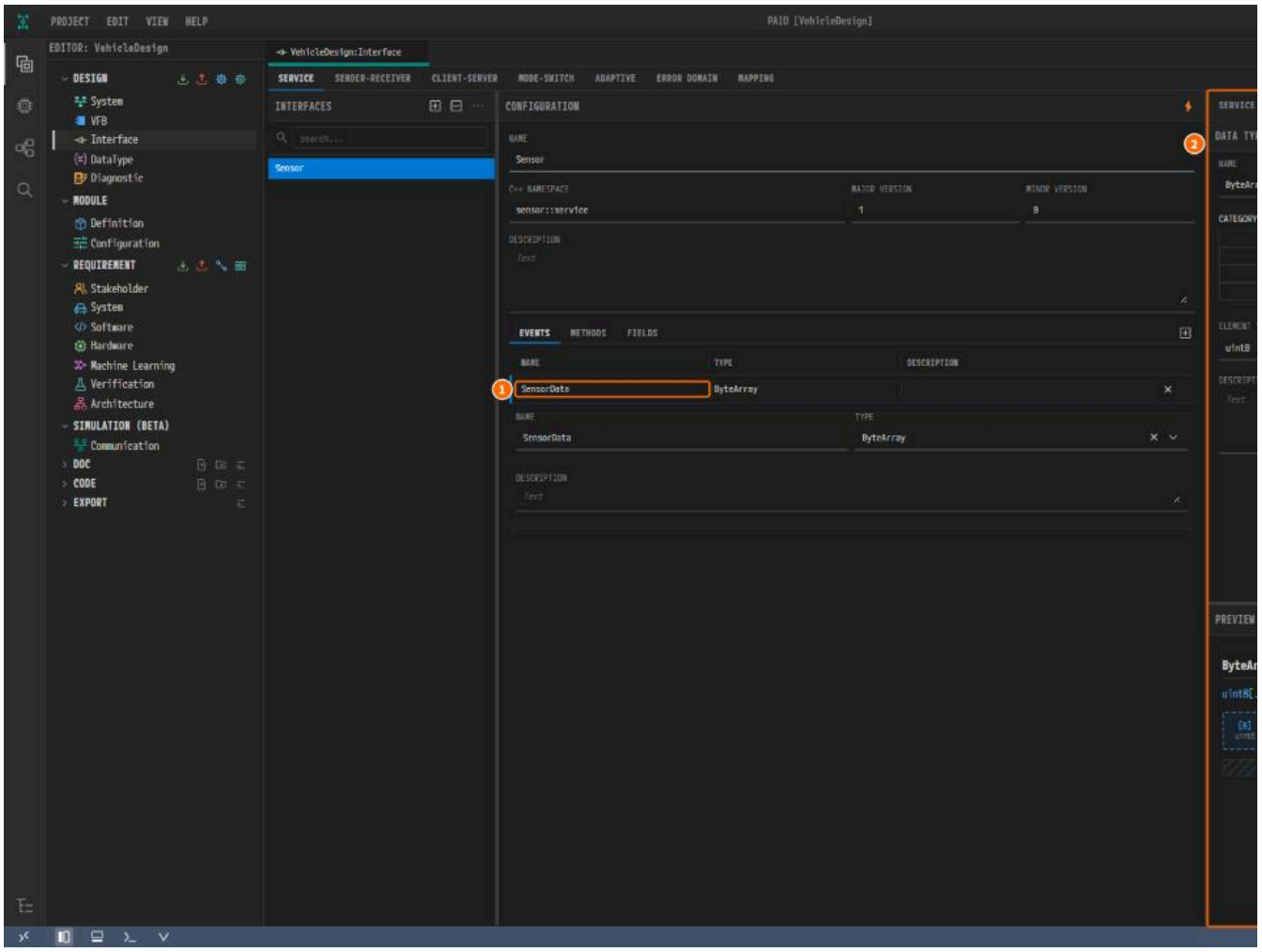


*그림 5-5. ① `SensorData` 행, ② **DATA TYPE** 패널.*

메서드 Calculation 추가하기

1. **METHODS** 서브탭을 누르고 서브탭 줄 오른쪽 끝의 **+** 를 누른다. 행 `Method_0` 이 펼쳐진 채 생긴다.
2. **NAME** 을 `Calculation` 으로 바꾸고 `Enter` 를 누른다. **METHOD TYPE** 은 **REQUEST AND RESPONSE** 그대로 둔다.

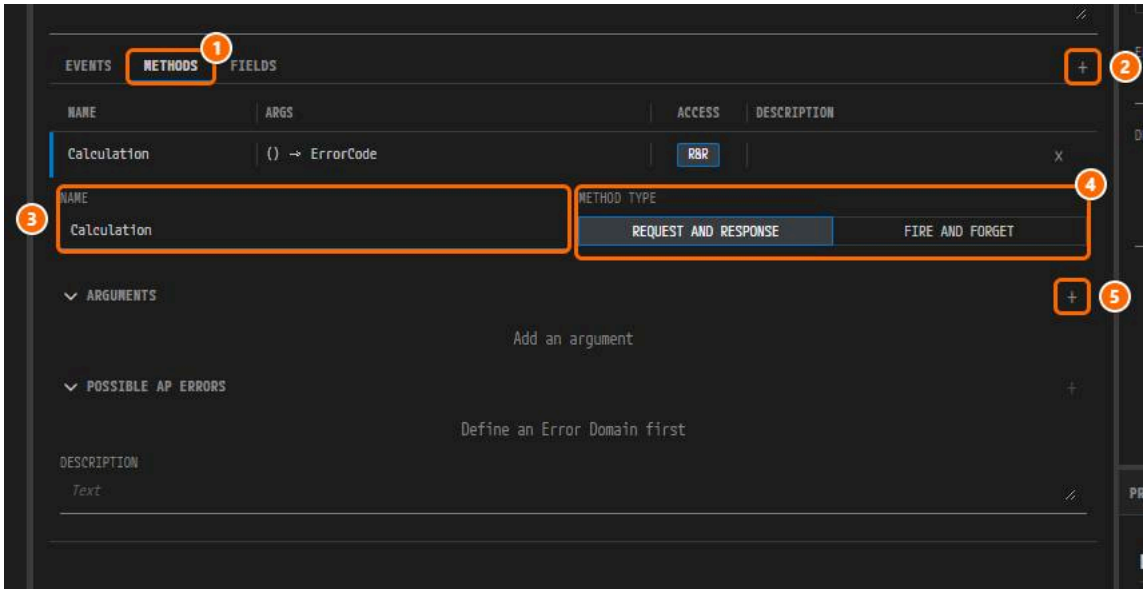


*그림 5-6. ① **METHODS**, ② **+**, ③ **NAME**, ④ **METHOD TYPE**, ⑤ **ARGUMENTS** 의 **+**.*

3. **ARGUMENTS** 의 **+** 를 누른다. `IN`, `uint8`, `argument_0` 인 1행이 생긴다.

4. 그 **NAME** 을 `front` 로 바꾸고 `Enter` 를 누른다. **TYPE** 의 ⌄ 를 누르고 `float32` 를 누른다.

5. 같은 방법으로 `rear`, `left`, `right` 를 추가한다.

6. 5행을 추가한다. 그 **DIRECTION** 의 ⌄ 를 누르고 `OUT` 을 누른다. 목록이 열려 있는 동안 칸에는 안내 `Type` 이 보인다.

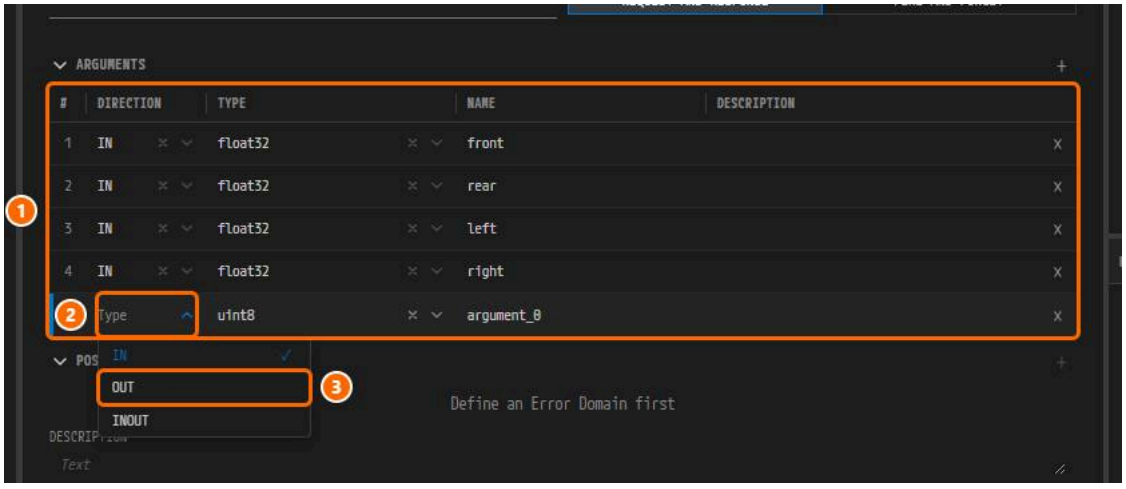


*그림 5-7. ① 1~4행, ② 5행의 **DIRECTION**, ③ `OUT`.*

7. 5행의 **NAME** 을 `result`, **TYPE** 을 `float32` 로 정한다. 메서드 행의 **ARGS** 가 `(float32 front, float32 rear, float32 left, float32 right) → Calcul`

저장하기

1. 탭 줄의 빈 곳을 누른 뒤 `Ctrl+S` 를 누른다. 탭의 수정 표시 ● 가 사라진다.

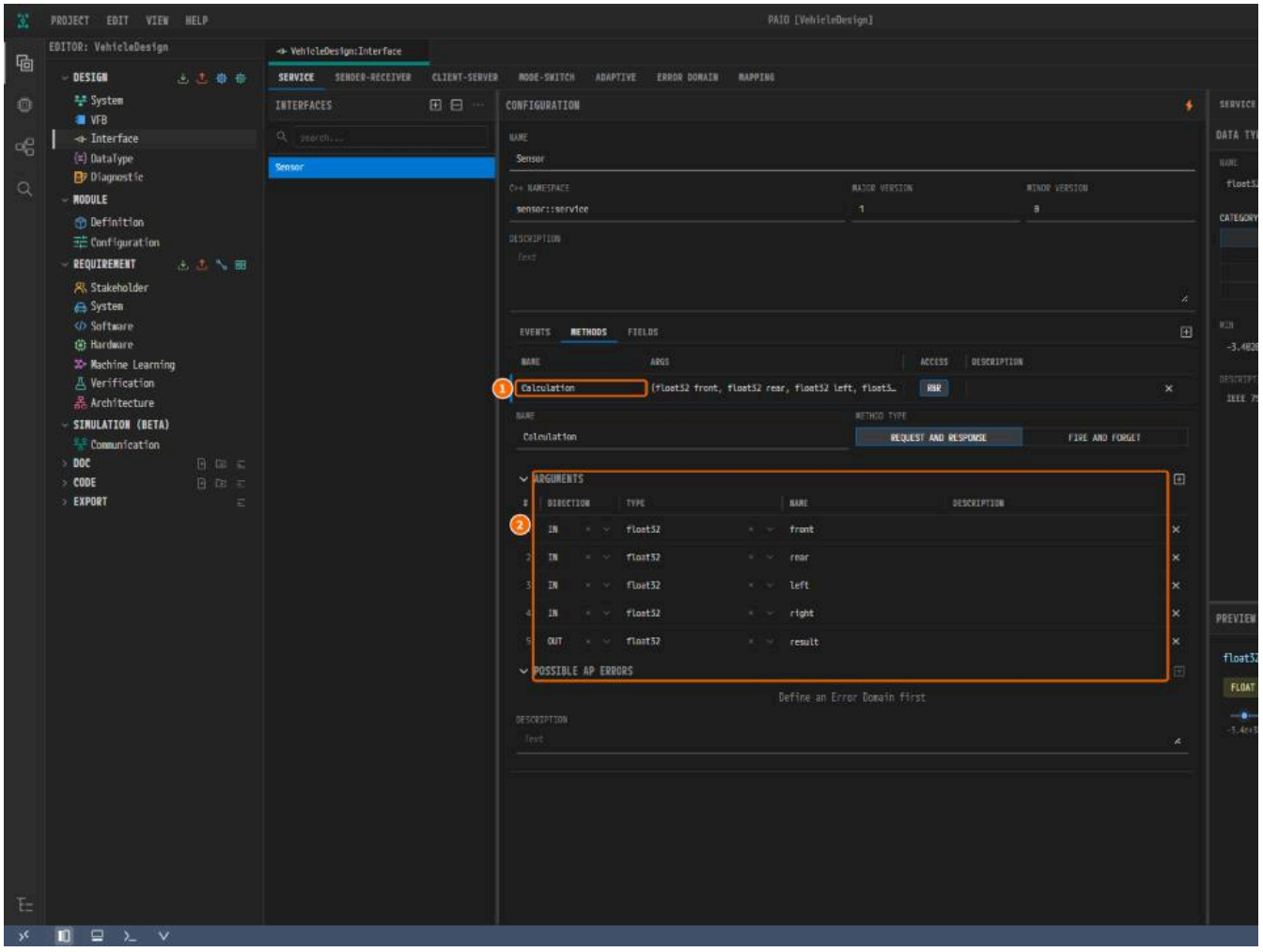


그림 5-8. ① `Calculation` 행, ② 인자 다섯 개.

검증 결과 보기

1. 푸터의 **V** 아이콘을 눌러 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

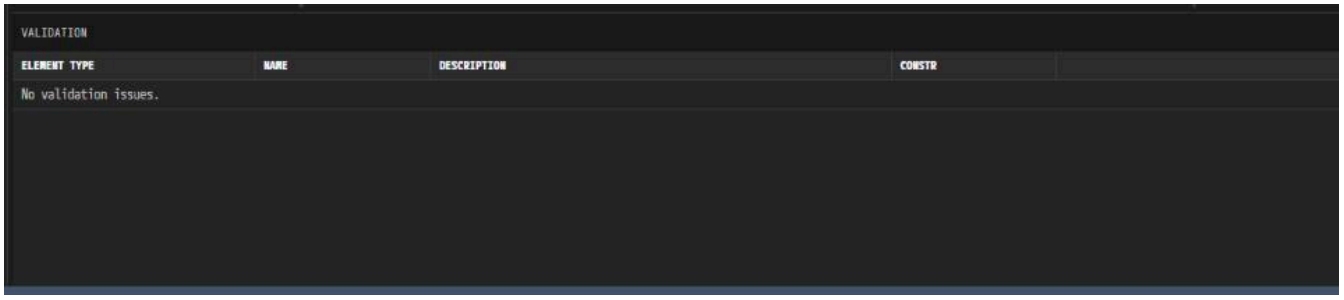


*그림 5-9. 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- **서비스 인터페이스는 하나이다.** 검증과 코드 생성의 Machine Extract 에서 `SERVICE-INTERFACE` 는 `Sensor` 하나뿐이다.
- **네임스페이스** `sensor::service`**.** 익스포트의 `NAMESPACES` 가 된다.
- **기본값을 그대로 둔다.** 버전 `1` / `0` 과 **REQUEST AND RESPONSE** 는 이미 설계에 필요한 값이다.
- **편집기의** `float32`**.** C++ 이름 `float` 로 나온다. 인자 순서도 표와 같아야 한다.
- **배포는 아직 없다. SERVICE DEPLOYMENT** 패널은 SOME/IP 배포와 서비스 인스턴스까지 비어 있다.

확인하기

- [ ] **INTERFACES** 목록에는 **C++ NAMESPACE** 가 `sensor::service` 인 `Sensor` 하나만 있다.

- [ ] **EVENTS** 에 `SensorData` `ByteArray` 가 있다.
- [ ] **METHODS** 에 표의 인자 다섯 개를 가진 `Calculation` 이 있다.
- [ ] 탭에 수정 표시가 없고 **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

SOME/IP 배포와 서비스 인스턴스에서 `Sensor` 의 SOME/IP 배포(서비스 ID `11`)와 서비스 인스턴스를 만든다.

SOME/IP 배포와 서비스 인스턴스

이 장에서 할 일

이 장에서는 인터페이스 `Sensor` 에 SOME/IP 배포 하나와 서비스 인스턴스 세 개, 곧 제공자 `Provider_Sensor` 와 소비자 `Actuator_Sensor`, `Fusion_Sensor` 를 준다.

시작하기 전에

- 서비스 인터페이스를 마친 뒤 시작한다.
- 탭 `VehicleDesign:Interface` 를 열어 둔다. 닫았다면 **EDITOR** 뷰의 **DESIGN** 아래에서 **Interface** 를 누른다.

요소	기본 이름 → 입력할 NAME	그 밖의 값
SOME/IP 배포	`Sensor` (바꾸지 않음)	**SERVICE ID** `11`, **MAJOR VERSION** `1` / **MINOR VERSION** `0` (기본값)
이벤트 배포	`SensorData` (자동)	**EVENT ID** `1`, **TRANSPORT UDP** (모두 기본값)
메서드 배포	`Calculation` (자동)	**METHOD ID** `1`, **TRANSPORT UDP** (모두 기본값)
이벤트 그룹	`EG_SensorData` → `Eventgroup_SensorData`	**EVENTGROUP ID** `1`, **EVENTS** `SensorData` (모두 기본값)
제공자	`Provider_Sensor_1` → `Provider_Sensor`	**INSTANCE ID** `1` (기본값)
소비자 1	`Consumer_Sensor_1` → `Actuator_Sensor`	**REQUIRED INSTANCE ID** `ANY` (기본값)
소비자 2	`Consumer_Sensor_2` → `Fusion_Sensor`	**REQUIRED INSTANCE ID** `ANY` (기본값)

SD(Service Discovery) 타이밍은 제공자의 **SD SERVER CONFIG** 와 소비자의 **SD CLIENT CONFIG** 에 넣는다. 그 밖의 SD 칸은 기본값 그대로 둔다.

칸	기본값	입력값	제공자	소비자
INITIAL DELAY MIN	`10 ms`	`10 us`	입력	입력
INITIAL DELAY MAX	`100 ms`	`100 us`	입력	입력
REPETITION BASE DELAY	`200 ms`	`200 us`	입력	입력
OFFER CYCLIC DELAY	`500 ms`	`1 ms`	입력	칸 없음

시간 칸은 값 오른쪽의 단위를 먼저 고른 뒤 숫자를 입력한다. 단위를 바꾸면 보이는 숫자만 환산된다.

> **참고.** SD 값을 하나 넣을 때마다 `Tab` 을 누르고 **INTERFACES** 목록의 `Sensor` 를 한 번 눌러 패널을 새로 그린다. 예전 PAIO 는 연달아 고친 SD 값을 잃었다.

SOME/IP 배포 만들기

1. **SERVICE** 탭의 **INTERFACES** 목록에서 `Sensor` 를 누른다. 오른쪽 패널이 **DATA TYPE** 이면 **SERVICE DEPLOYMENT** 를 누른다.

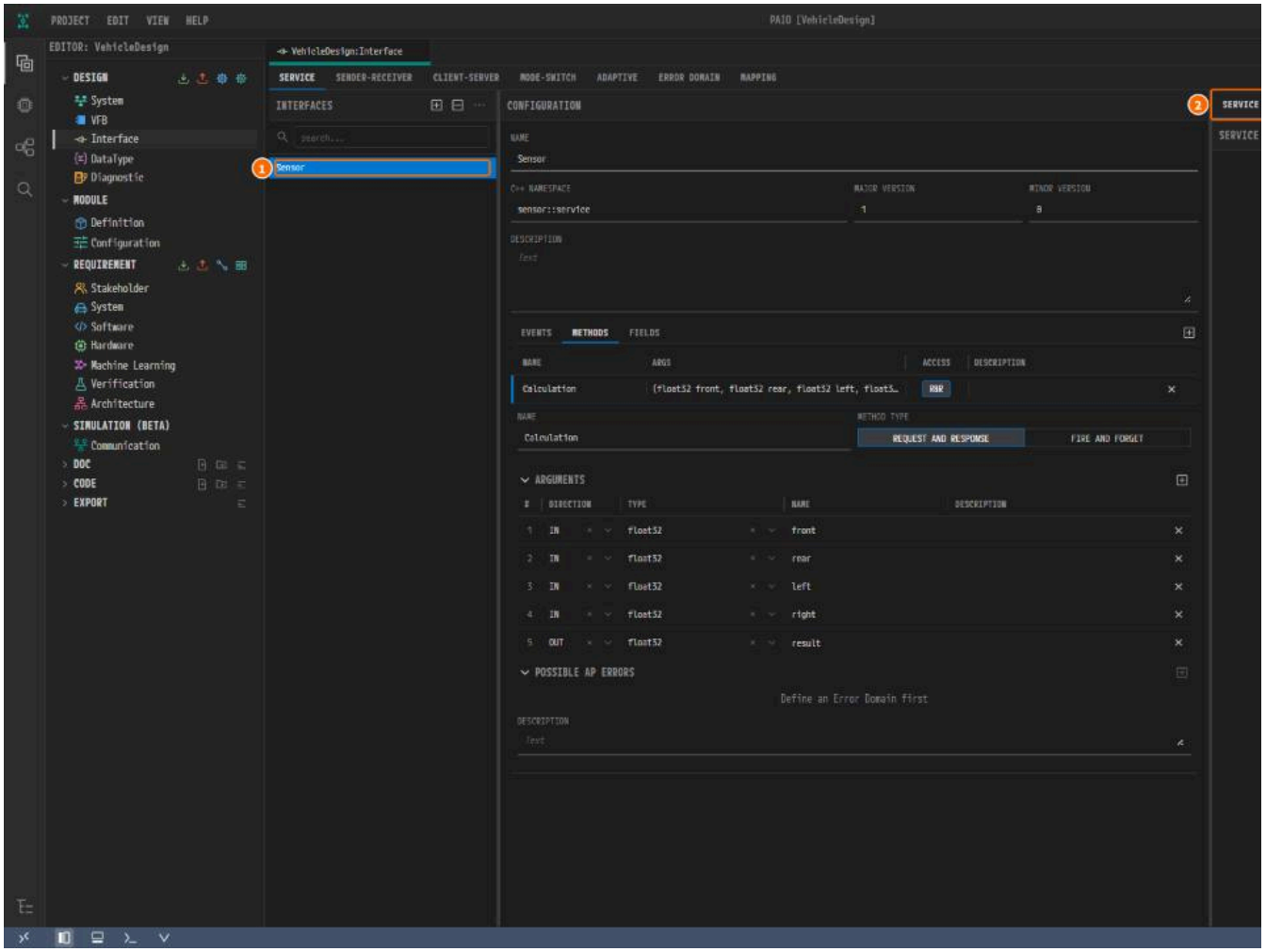


*그림 6-1. ① `Sensor`, ② **SERVICE DEPLOYMENT**, ③ **+** 버튼.*

2. **SERVICE DEPLOYMENT** 제목줄의 **+** 를 누른다.

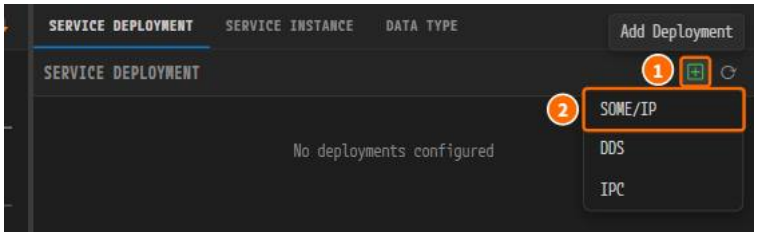


*그림 6-2. ① **+**, ② **SOME/IP**.*

3. **SOME/IP** 를 누른다. 접힌 카드 `SOME/IP` `Sensor` 가 생긴다. `SensorData` 와 `Calculation` 의 배포도 함께 만들어진다.



그림 6-3. ① 카드 `SOME/IP` `Sensor`. 서비스 ID 는 아직 기본값 `0x0001` 이다.

4. 카드 머리글을 눌러 펼친다. **DEC** 가 활성인 상태에서 **SERVICE ID** 를 `11` 로 바꾸고 `Tab` 을 누른다.

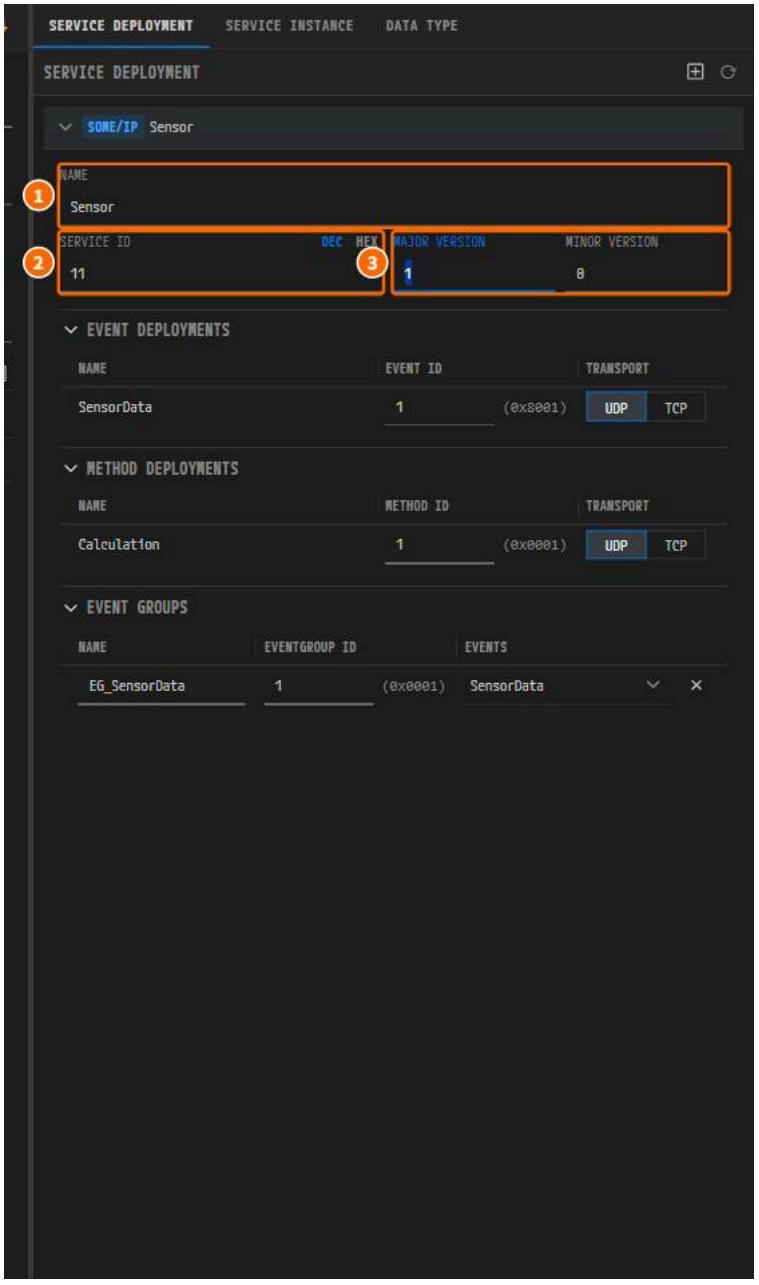


*그림 6-4. ① **NAME**, ② **SERVICE ID**, ③ **MAJOR VERSION** 과 **MINOR VERSION**.*

5. **EVENT DEPLOYMENTS** 에 `SensorData`, **METHOD DEPLOYMENTS** 에 `Calculation` 이 있고 둘 다 ID `1`, **UDP** 인지 확인한다. 그대로 둔다.

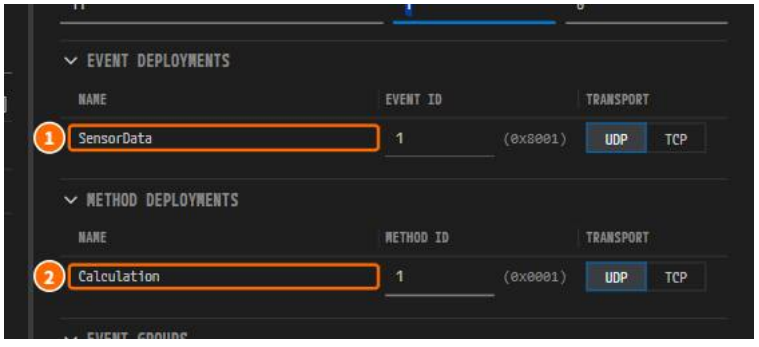


그림 6-5. ① `SensorData`, ② `Calculation`. `(0x8001)` 은 SOME/IP 가 이벤트를 보내는 형태를 보일 뿐이며 이벤트 ID 는 `1` 이다.

6. **EVENT GROUPS** 에서 `EG_SensorData` 의 **NAME** 을 `Eventgroup_SensorData` 로 바꾸고 `Tab` 을 누른다.



*그림 6-6. ① **NAME**, ② **EVENTGROUP ID**, ③ **EVENTS**. 칸에는 이름의 마지막 글자가 잘려 보인다.*

서비스 인스턴스 추가하기

1. 카드 머리글에 마우스를 올리고 오른쪽 끝에 나타나는 **+** 를 누른다.



*그림 6-7. ① **+**, ② **Provider** 와 **Consumer**.*

2. **Provider** 를 누른다. 이어서 **+** 와 **Consumer** 를 두 번 눌러 소비자 둘을 추가한다.

3. 카드 머리글을 눌러 접는다. 요약에 `Provider_Sensor_1`, `Consumer_Sensor_1`, `Consumer_Sensor_2` 가 보인다.



그림 6-8. ① `Provider_Sensor_1`, ② `Consumer_Sensor_1`, ③ `Consumer_Sensor_2`.

제공자 Provider_Sensor 설정하기

1. 요약에서 `Provider_Sensor_1` 을 누른다. 오른쪽 패널이 **SERVICE INSTANCE** 로 바뀌고 제공자 카드가 펼쳐진다.

2. **NAME** 을 `Provider_Sensor` 로 바꾸고 `Tab` 을 누른다. **INSTANCE ID** `1` 은 그대로 둔다.

3. **SD SERVER CONFIG** 를 펼치고 표의 값 네 개를 넣는다. `Sensor` 를 누른 뒤 카드가 접히면 카드와 **SD SERVER CONFIG** 를 다시 펼친다.

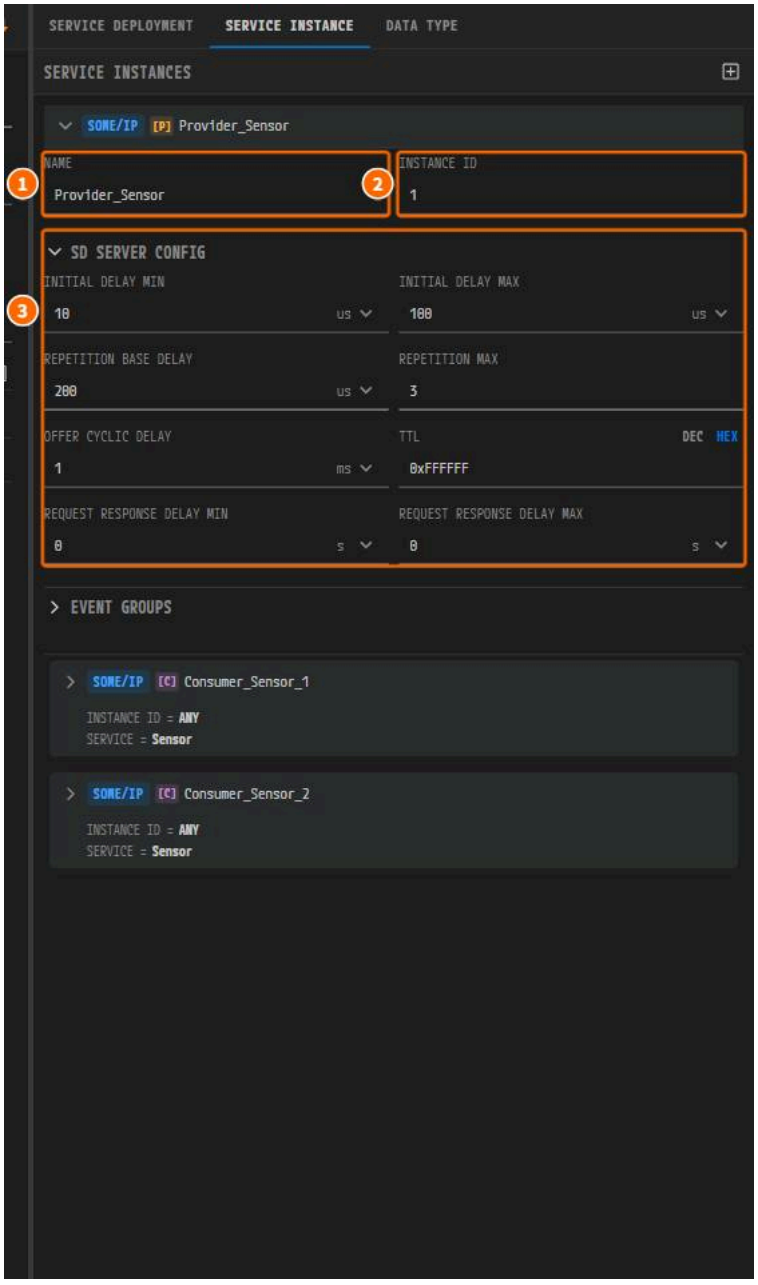


*그림 6-9. ① **NAME**, ② **INSTANCE ID**, ③ **SD SERVER CONFIG**.*

4. **EVENT GROUPS** 와 행 `Eventgroup_SensorData` 를 펼친다. 모두 기본값 **UNICAST** 그대로 둔다.

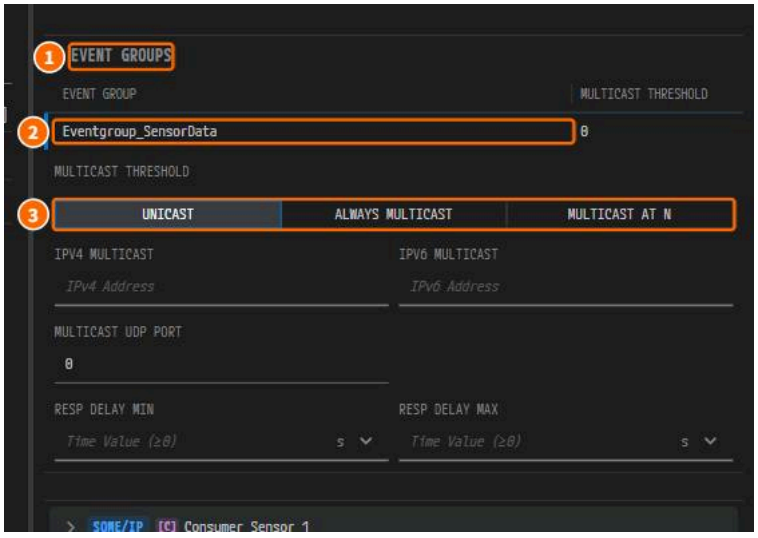


*그림 6-10. ① **EVENT GROUPS**, ② `Eventgroup_SensorData`, ③ **MULTICAST THRESHOLD**.*

소비자 Actuator_Sensor 와 Fusion_Sensor 설정하기

1. 카드 `Consumer_Sensor_1` 을 펼친다. **NAME** 을 `Actuator_Sensor` 로 바꾸고 `Tab` 을 누른다. **REQUIRED INSTANCE ID** `ANY` 는 그대로 둔다.
2. **SD CLIENT CONFIG** 를 펼치고 같은 방법으로 표의 값 세 개를 넣는다.

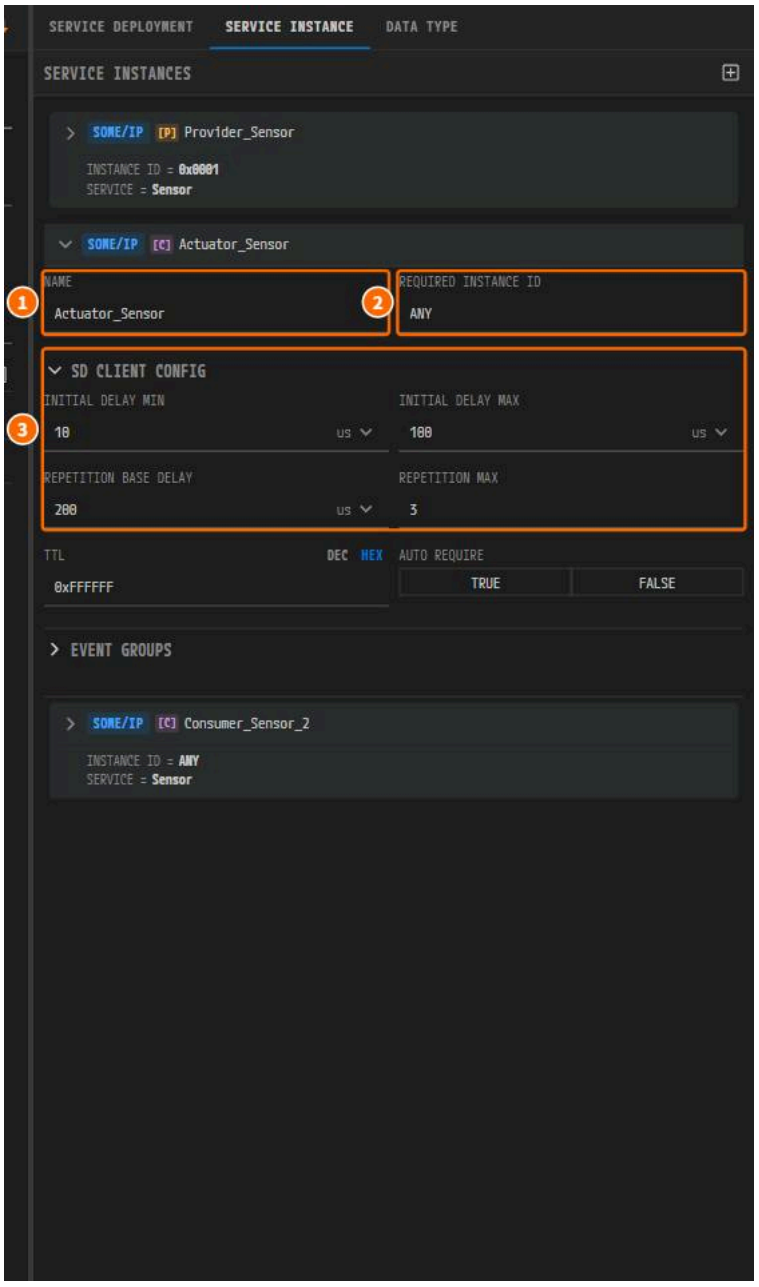


*그림 6-11. ① **NAME**, ② **REQUIRED INSTANCE ID**, ③ **SD CLIENT CONFIG**.*

3. `Consumer_Sensor_2` 도 같은 방법으로 이름 `Fusion_Sensor` 로 설정한다.

저장하기

1. 탭 줄의 빈 곳을 누르고 `Ctrl+S` 를 누른다. 수정 표시 ● 가 사라지고 `design/SERVICE.io.model` 이 만들어진다.
2. 펼쳐진 카드를 접는다.

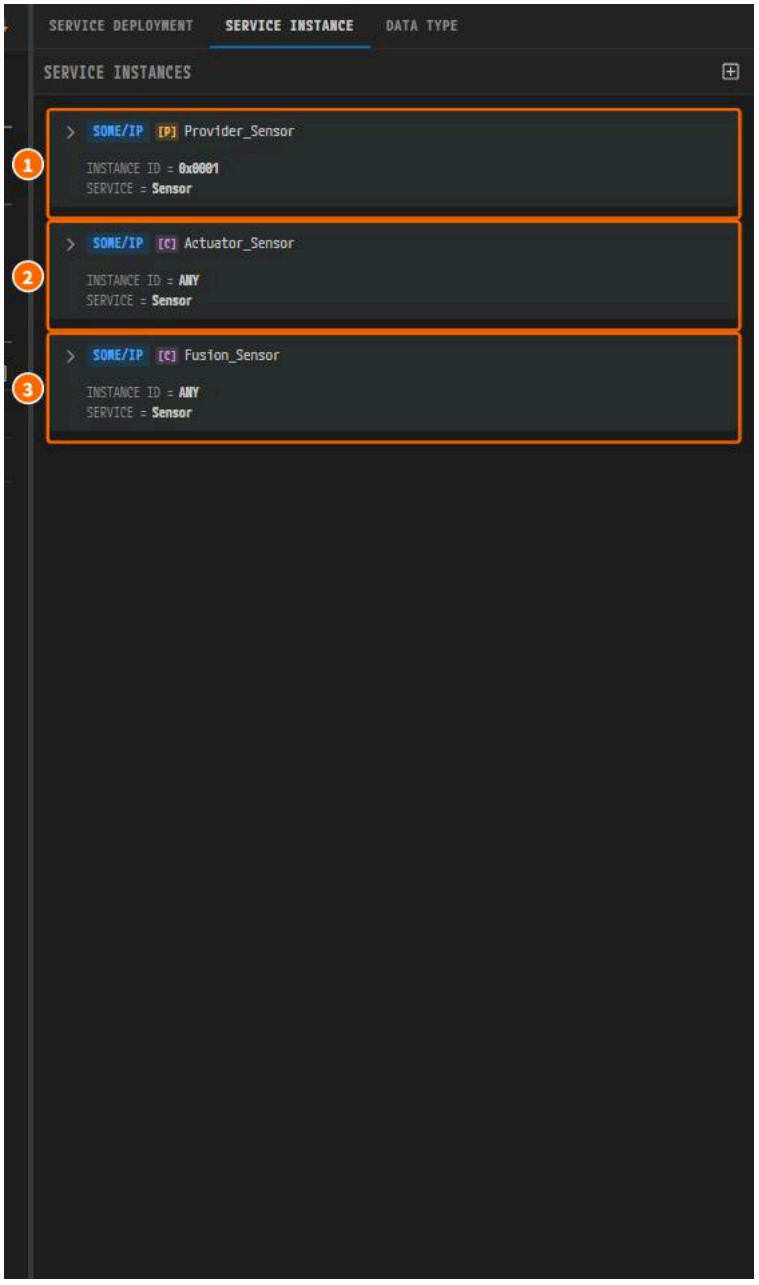


그림 6-12. ① `Provider_Sensor`, ② `Actuator_Sensor`, ③ `Fusion_Sensor`.

저장하면 `SIGNAL.io.model` 의 Classic Platform 시그널과 인터페이스 `Sensor_SR`, `Sensor_CS` 도 자동으로 파생된다. 그대로 둔다.

검증 결과 보기

1. 푸터의 **V** 아이콘을 눌러 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

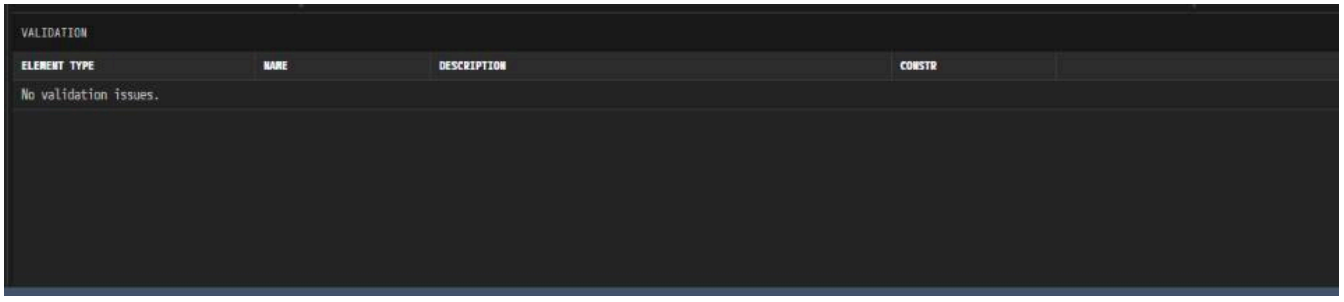


*그림 6-13. 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- **서비스 ID 만 입력한다.** 편집기가 인터페이스 이름으로 배포 이름을 정하고, 이벤트·메서드 배포를 필요한 ID 로 만든다.
- **이벤트 그룹 이름.** 검증과 코드 생성의 익스포터는 `Eventgroup_{이벤트}` 규칙으로 그룹의 이벤트를 찾는다.
- **인스턴스 이름.** 익스포트에서 인스턴스의 `SHORT-NAME` 이 된다.
- **짧은 SD 타이밍.** 입력값은 기본값보다 훨씬 짧다.
- **머신 바인딩은 아직 없다.** 커넥터에 서비스 인스턴스 바인딩에서 만든다.

확인하기

- [ ] 카드 `SOME/IP` `Sensor` 의 **SERVICE ID** 는 `11` 이고 이벤트 그룹은 `Eventgroup_SensorData` 이다.
- [ ] **SERVICE INSTANCE** 에 `[P]` `Provider_Sensor`, `[C]` `Actuator_Sensor`, `[C]` `Fusion_Sensor` 가 있다.
- [ ] 카드마다 SD 값이 표와 같다.
- [ ] 탭에 수정 표시가 없고 **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

커넥터에 서비스 인스턴스 바인딩에서 인스턴스 세 개를 Ethernet 커넥터에 바인딩하고 UDP 포트를 정한다.

커넥터에 서비스 인스턴스 바인딩

이 장에서 할 일

이 장에서는 SOME/IP 배포와 서비스 인스턴스의 서비스 인스턴스 세 개를 Ethernet 커넥터 `EthernetConnector` 에 바인딩하고, 각각에 토폴로지 — SOME/IP 디스커버리 포트

시작하기 전에

- SOME/IP 배포와 서비스 인스턴스를 마친 뒤 시작한다.
- **VALIDATION** 패널은 `No validation issues.` 이다.

인스턴스	역할	UDP PORT	TCP PORT
`Provider_Sensor`	제공자(배지 **P**)	`20003`	비워 둠
`Actuator_Sensor`	소비자(배지 **C**)	`20001`	비워 둠
`Fusion_Sensor`	소비자(배지 **C**)	`20002`	비워 둠

커넥터 EthernetConnector 의 SERVICE INSTANCES 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System** 을 누른다. 탭 `VehicleDesign:System` 이 열린다.
2. 노드 `HPC [HPC]` 경계의 커넥터 `EthernetConnector` 를 누른다. 오른쪽에 **ETHERNET CONNECTOR** 패널이 열린다.
3. **SERVICE INSTANCES** 머리글을 눌러 펼친다. `No service instances defined` 가 보인다.

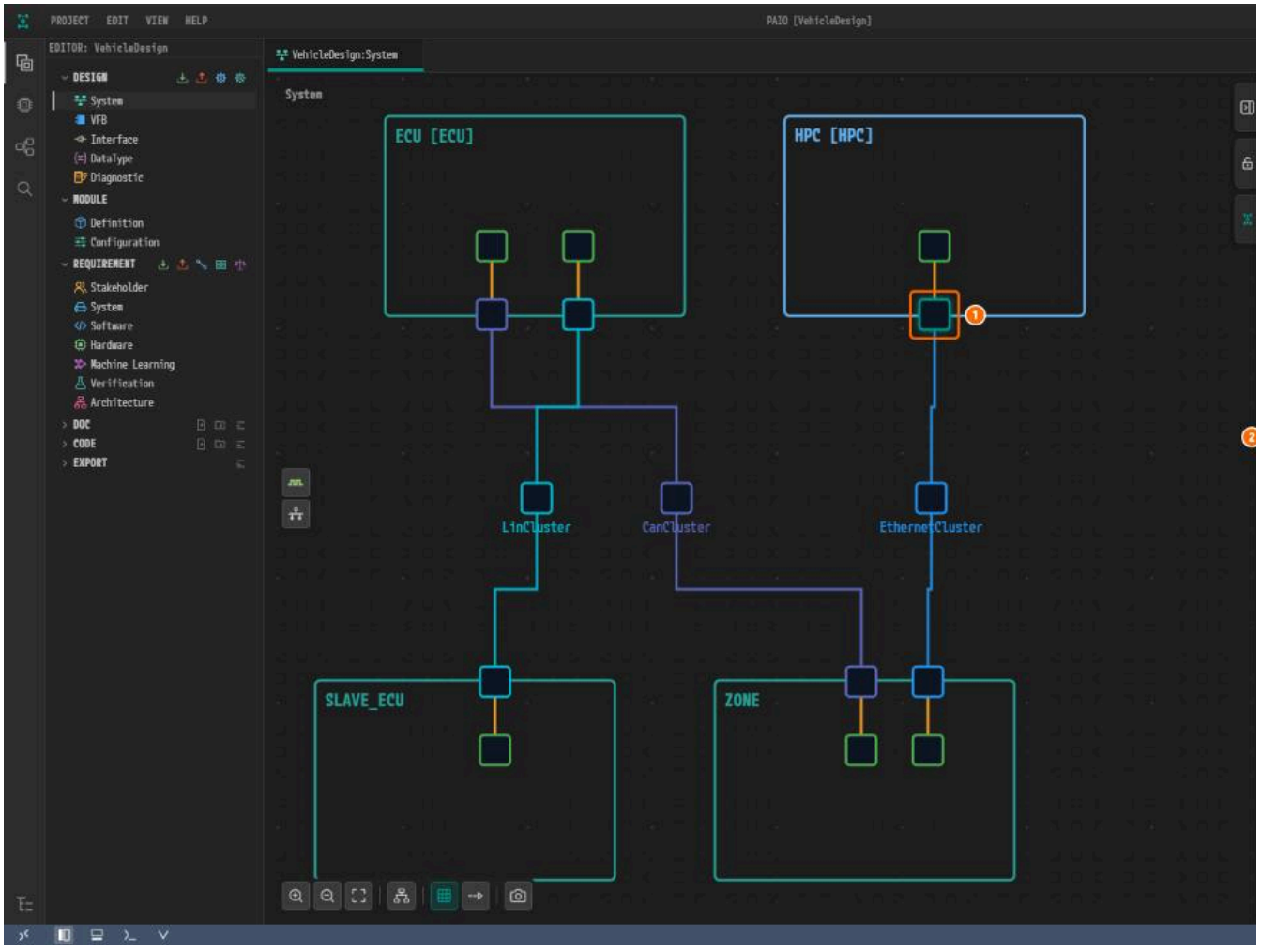


*그림 7-1. ① 커넥터 `EthernetConnector`, ② **SERVICE INSTANCES**, ③ **+** 버튼.*

서비스 인스턴스 세 개 바인딩하기

1. **SERVICE INSTANCES** 머리글의 **+** 버튼을 누른다. 행 세 개가 있는 모달 **Select Service Instance** 가 열린다.

2. 세 행을 모두 체크하고 **Bind** 를 누른다.

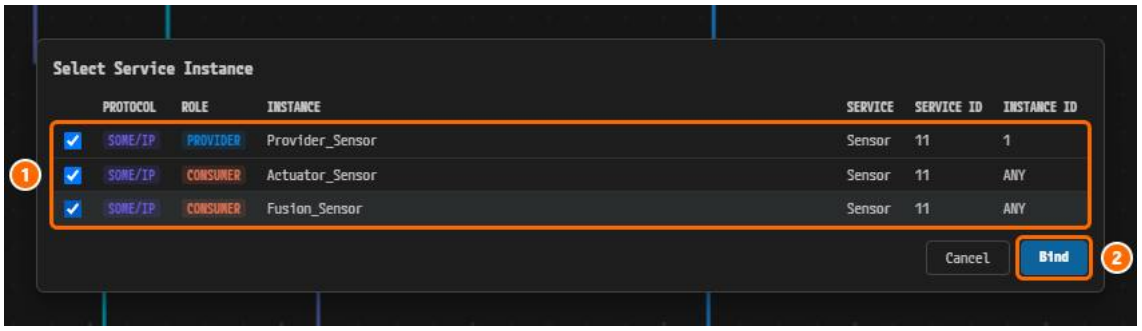


*그림 7-2. ① 모두 체크한 행 세 개, ② **Bind**.*

3. 카드 세 장이 생긴다. 푸터의 **V** 아이콘에 빨간 배지 `3` 이 붙는다. UDP 포트가 없는 바인딩마다 오류가 하나씩 있고, 포트를 하나 고를 때마다 하나씩 사라진다.

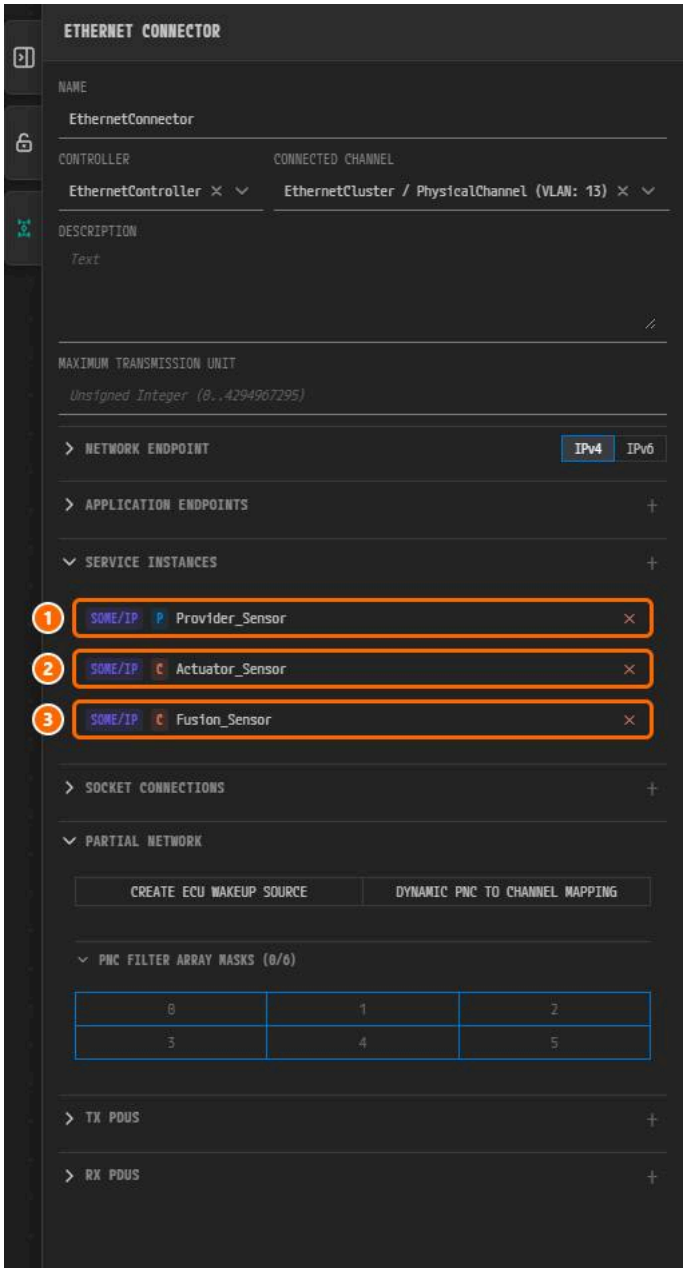


그림 7-3. ① `Provider_Sensor`, ② `Actuator_Sensor`, ③ `Fusion_Sensor`. 아직 어느 카드에도 포트가 없다.

UDP 포트 정하기

먼저 `Provider_Sensor` 를 따라간다.

1. 카드 `Provider_Sensor` 를 눌러 펼친다. **UDP PORT** 오른쪽의 ▾ 버튼을 누른다.

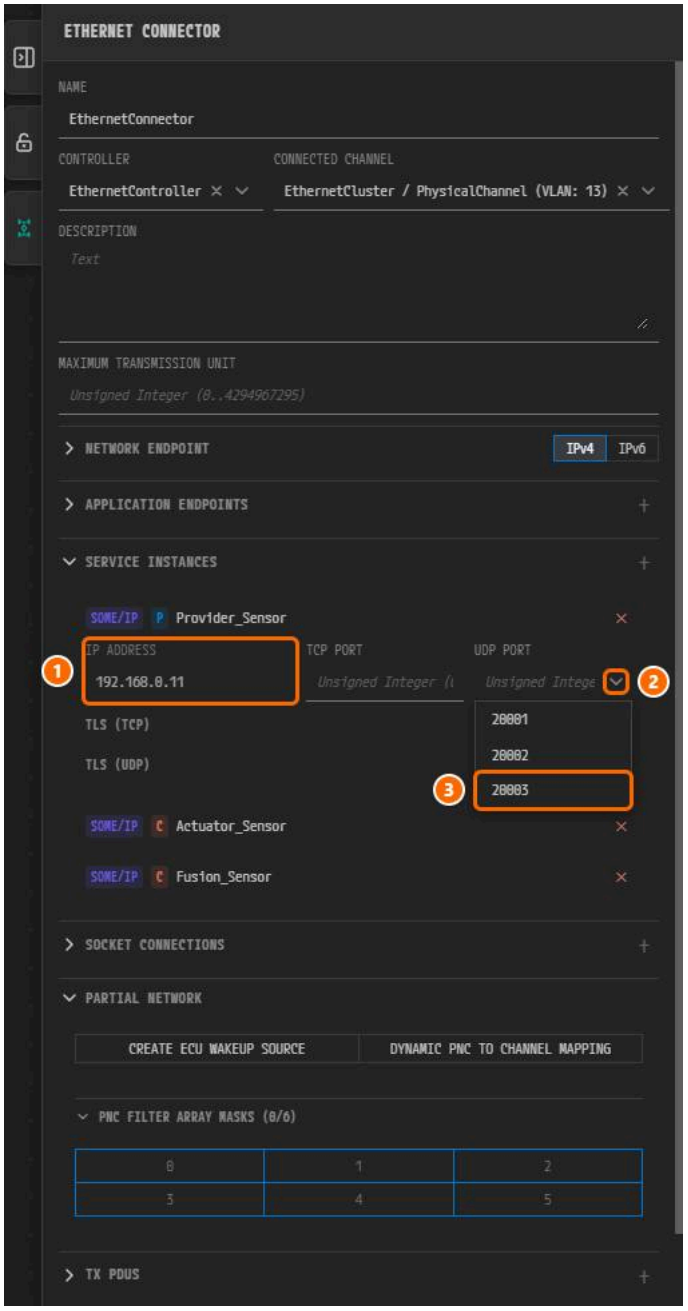


*그림 7-4. ① **IP ADDRESS**(읽기 전용), ② ▾ 버튼, ③ `20003`.*

2. `20003` 을 누른다. **TCP PORT** 와 **TLS (TCP)**, **TLS (UDP)** 행은 비워 둔다.
3. 커넥터 `EthernetConnector` 를 다시 누르고 카드를 펼친다. 요약 줄에 `UDP:20003` 이 보인다.

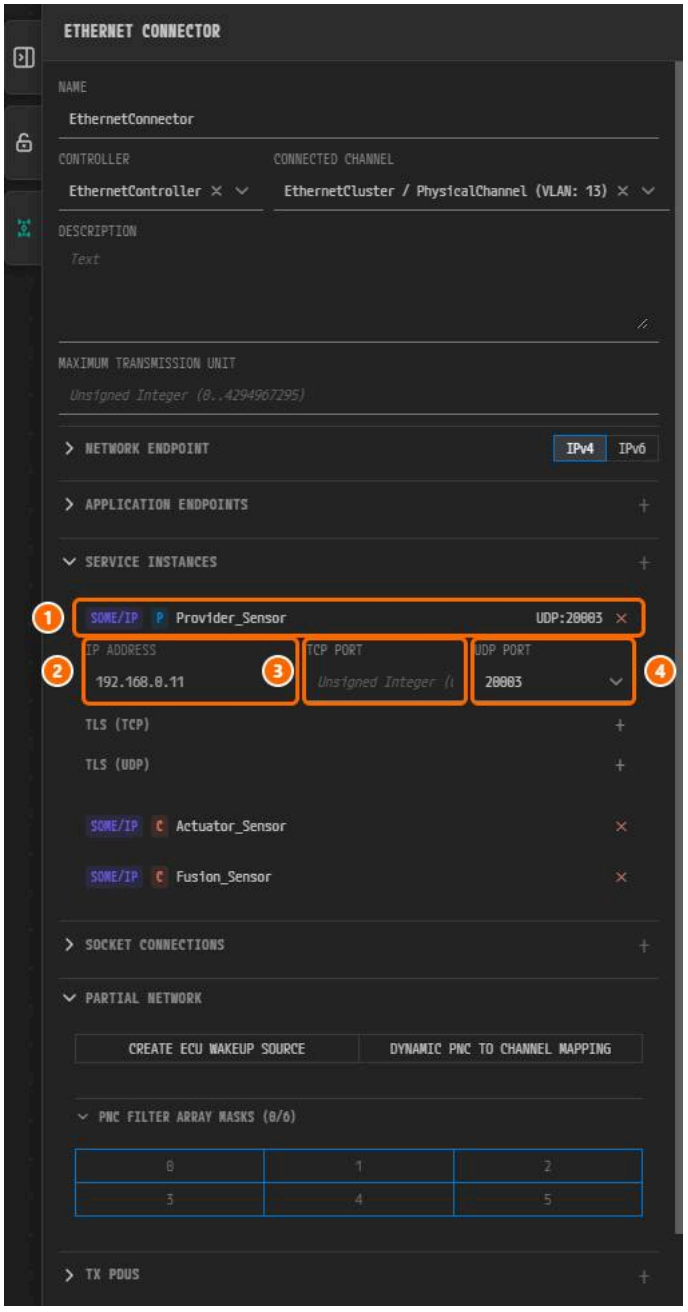


*그림 7-5. ① `UDP:20003`, ② **IP ADDRESS**, ③ **TCP PORT**(비어 있음), ④ **UDP PORT** = `20003`.*

4. 같은 방법으로 `Actuator_Sensor` 에 `20001`, `Fusion_Sensor` 에 `20002` 를 정한다. **V** 아이콘의 배지가 사라진다.

5. 커넥터를 다시 누르고 **APPLICATION ENDPOINTS** 를 펼친다. 3장의 엔드포인트 세 개만 그대로 있다.

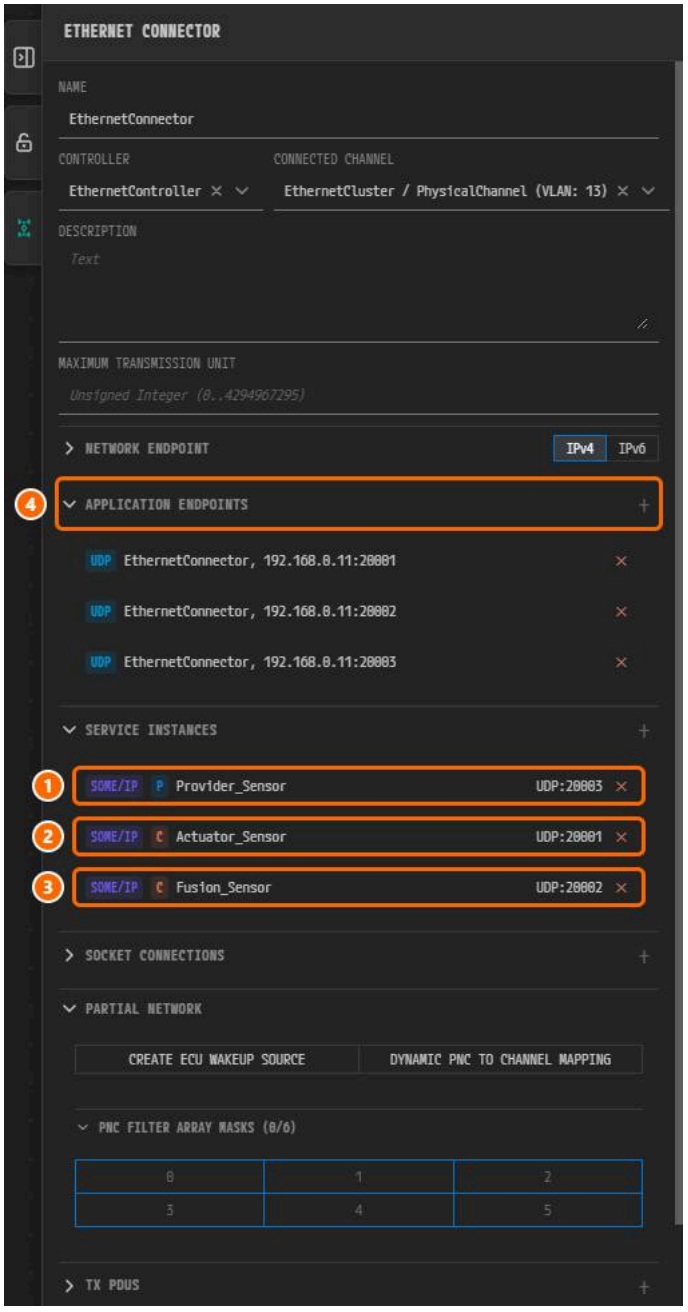


*그림 7-6. ① `Provider_Sensor` `UDP:20003`, ② `Actuator_Sensor` `UDP:20001`, ③ `Fusion_Sensor` `UDP:20002`, ④ 바뀌지 않은 **APPLICATION ENDPOINTS**.*

저장하기

1. 캔버스의 빈 곳을 누르고 `Ctrl+S` 를 누른다. 탭 `VehicleDesign:System` 의 수정 표시 ● 가 사라진다.
2. 커넥터 `EthernetConnector` 를 다시 눌러 카드 세 장을 확인한다.

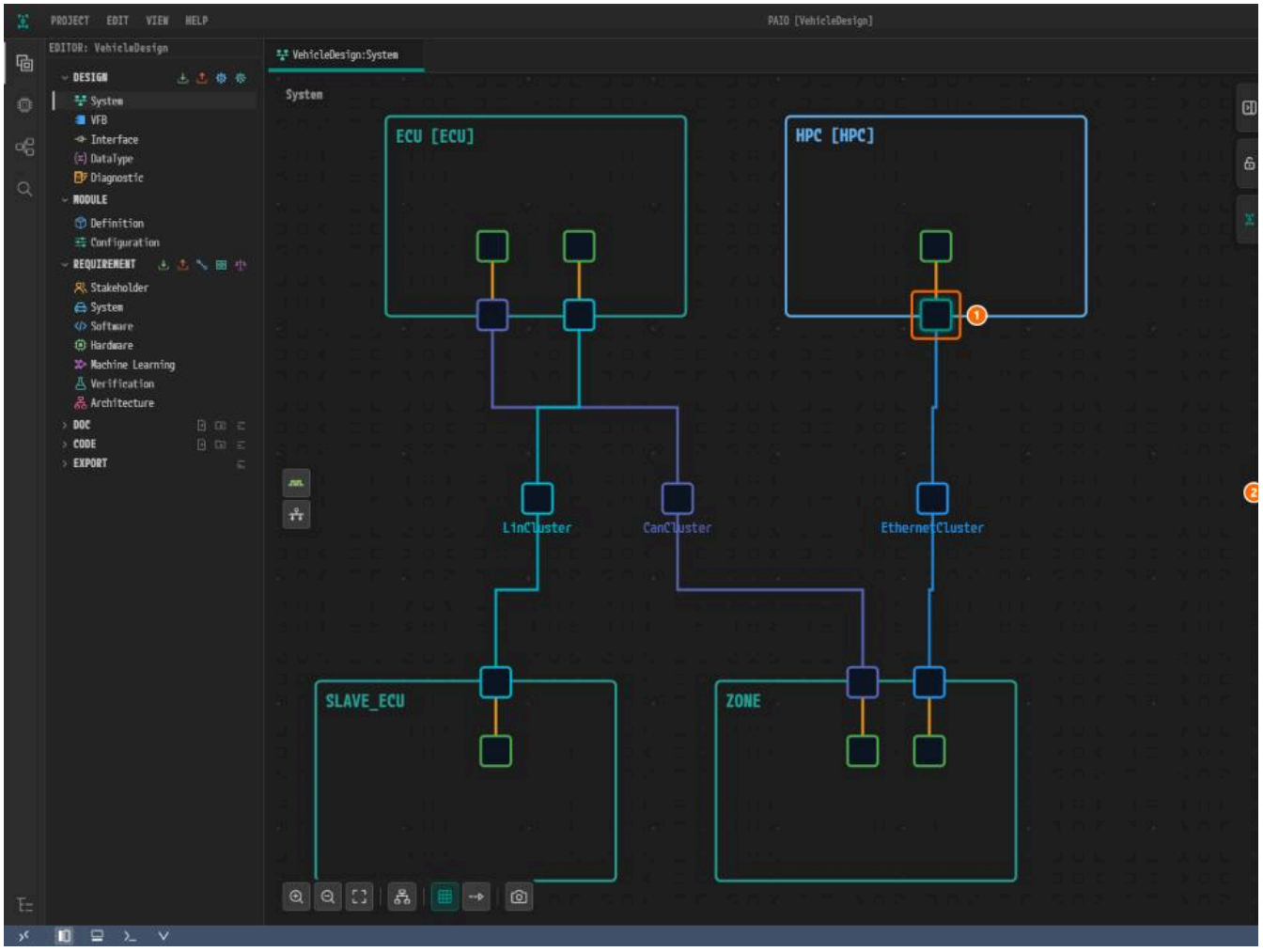


그림 7-7. ① `EthernetConnector`, ② 포트를 갖춘 카드 세 장.

검증 결과 보기

1. **VALIDATION** 패널을 본다. 닫혀 있으면 푸터의 **V** 아이콘을 누른다. `No validation issues.` 가 보인다.

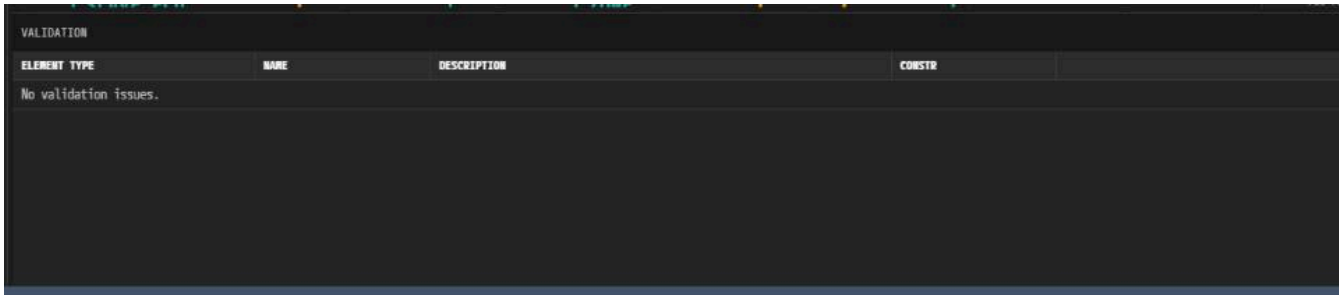


*그림 7-8. 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- **바인딩 하나가 매핑 하나.** 검증과 코드 생성의 Machine Extract 에서 바인딩마다 `SOMEIP-SERVICE-INSTANCE-TO-MACHINE-MAPPING` 이 하나씩 생긴다. 매핑 이름은 우
- **목록에서 고르는 포트.** 목록에서 고르면 기존 엔드포인트를 참조하므로 엔드포인트가 늘지 않는다.
- **TCP 포트 없음.** 이벤트 `SensorData` 와 메서드 `Calculation` 이 **UDP** 를 쓰므로 UDP 포트만 있으면 된다.

확인하기

- [ ] `EthernetConnector` 의 **SERVICE INSTANCES** 에 `Provider_Sensor` `UDP:20003`, `Actuator_Sensor` `UDP:20001`, `Fusion_Sensor` `UDP:20002` 가 보인다.
- [ ] **APPLICATION ENDPOINTS** 에는 3장의 엔드포인트 세 개만 있다.
- [ ] 탭 `VehicleDesign:System` 에 수정 표시 ● 가 없다.

- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

Adaptive Application 과 포트에서 Adaptive Application 세 개와 그 포트를 만든다.

Adaptive Application 과 포트

이 장에서 할 일

이 장에서는 컴포지션 `HPC` 안에 Adaptive Application 세 개를 만들고, `Sensor` 에 P 포트, `Fusion` 과 `Actuator` 에 R 포트를 서비스 인터페이스의 `Sensor` 로 붙인 뒤 P 포트

시작하기 전에

- 커넥터에 서비스 인스턴스 바인딩을 마친 뒤 시작한다.
- 컴포지션 HPC 와 머신에서 만든 컴포지션 `HPC` 는 아직 비어 있다.

Adaptive Application	NAME = AS PROTOTYPE	포트	포트 NAME	PORT INTERFACE
센서	`Sensor`	**PPort** 1개	`Sensor`	`Service: Sensor`
융합	`Fusion`	**RPort** 1개	`Sensor`	`Service: Sensor`
액추에이터	`Actuator`	**RPort** 1개	`Sensor`	`Service: Sensor`

커넥터 두 개의 이름은 `Assembly_Sensor_Fusion` 과 `Assembly_Sensor_Actuator` 이다.

컴포지션 HPC 안으로 들어가기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 누른다. 탭 `VehicleDesign:VFB` 가 열린다.
2. 노드 `HPC [HPC]` 를 더블클릭한다. 빵부스러기가 `System > HPC` 로 바뀌고 빈 프레임 `HPC` 가 보인다.
3. 프레임 제목 `HPC` 를 누른다. 노드 세 개가 들어가도록 오른쪽 아래 핸들을 끌어 프레임을 약 900×520 으로 넓힌다.

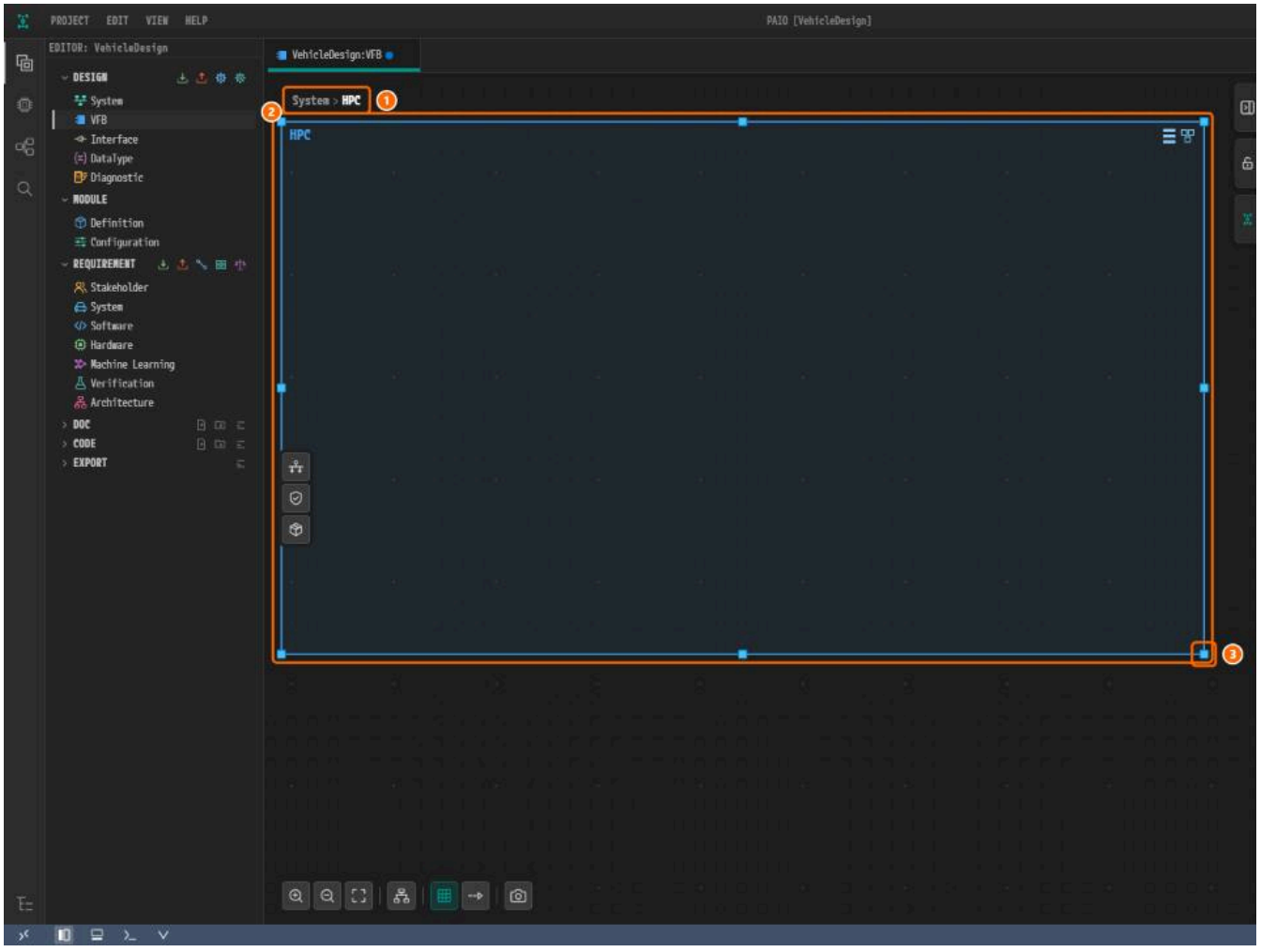


그림 8-1. ① 빵부스러기 `System > HPC`, ② 프레임 `HPC`, ③ 크기 조절 핸들.

Adaptive Application Sensor 만들기

1. 프레임 안의 빈 곳을 오른쪽 클릭한다. **Add SWC** 위에 포인터를 올리고 **Adaptive Application** 을 누른다.

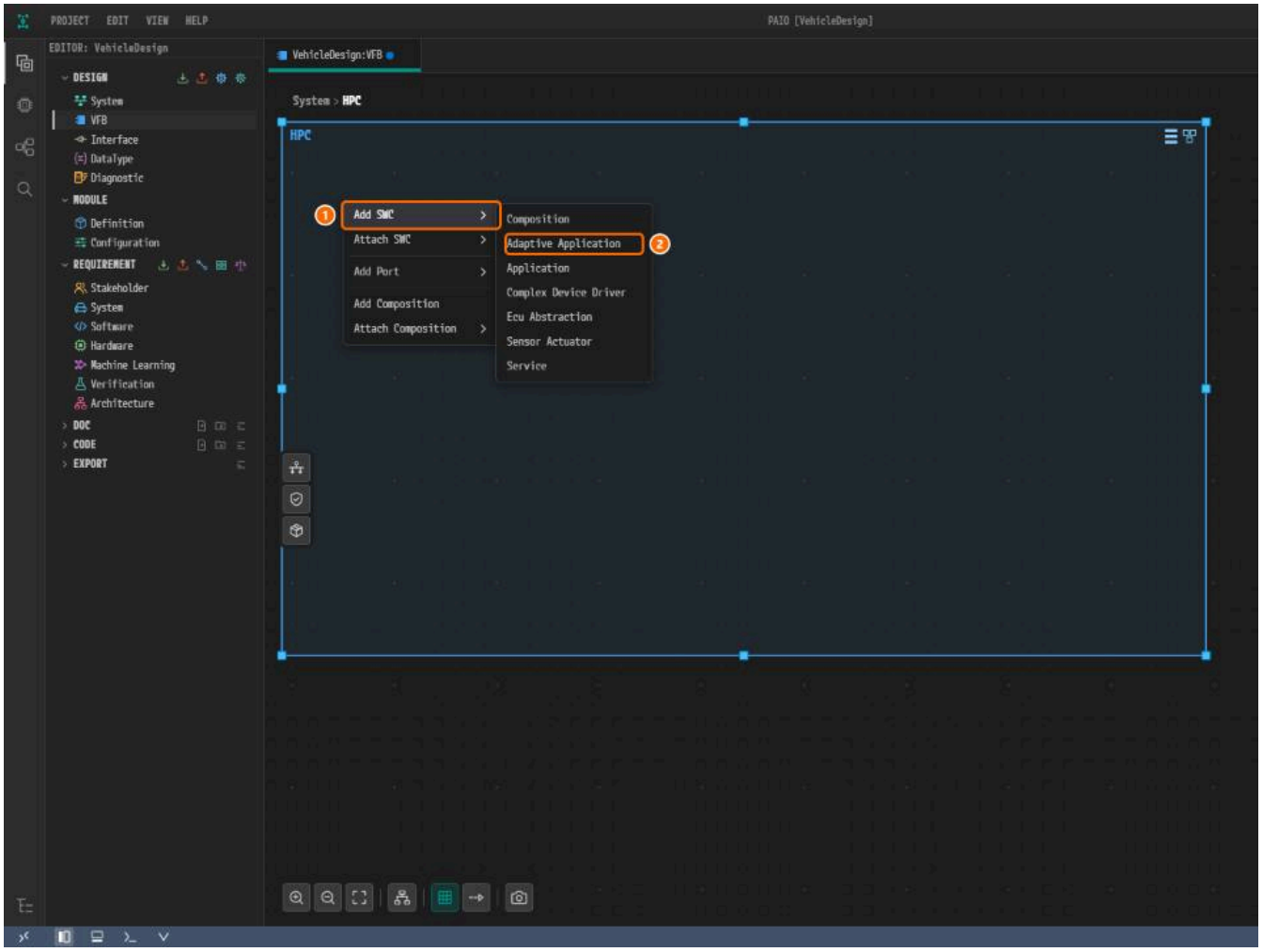


*그림 8-2. ① **Add SWC**, ② **Adaptive Application**.*

2. 새 노드 `AdaptiveApp_0` 의 머리글을 누른다. **ADAPTIVE APPLICATION** 패널이 열린다.

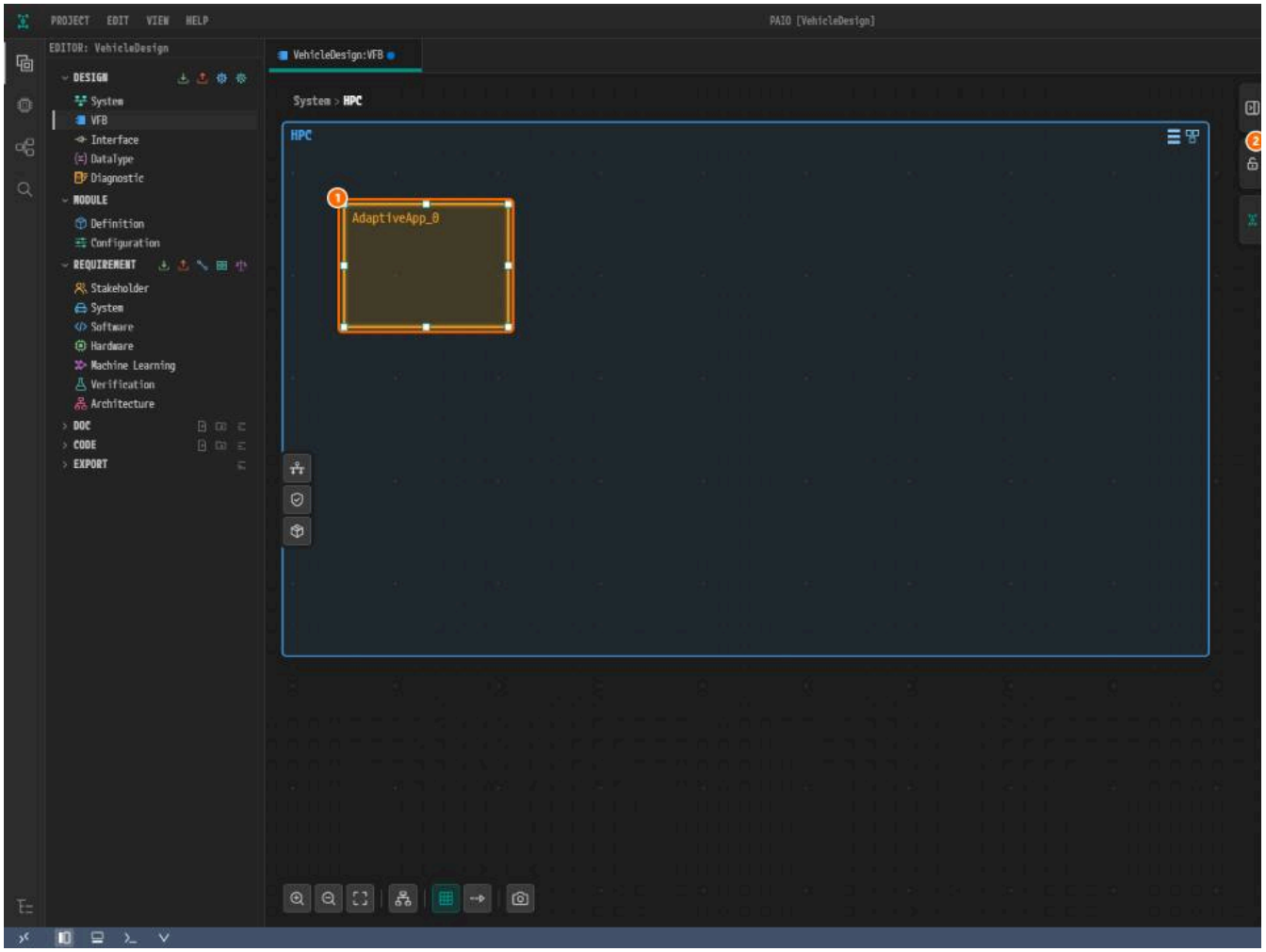


*그림 8-3. ① 노드 `AdaptiveApp_0`, ② **NAME**, ③ **AS PROTOTYPE**.*

3. **NAME** 에 `Sensor` 를 입력하고 `Tab` 을 누른다. **AS PROTOTYPE** 에도 `Sensor` 를 입력하고 `Tab` 을 누른다. **AS PROTOTYPE** 은 **NAME** 을 따라 바뀌지 않는다.

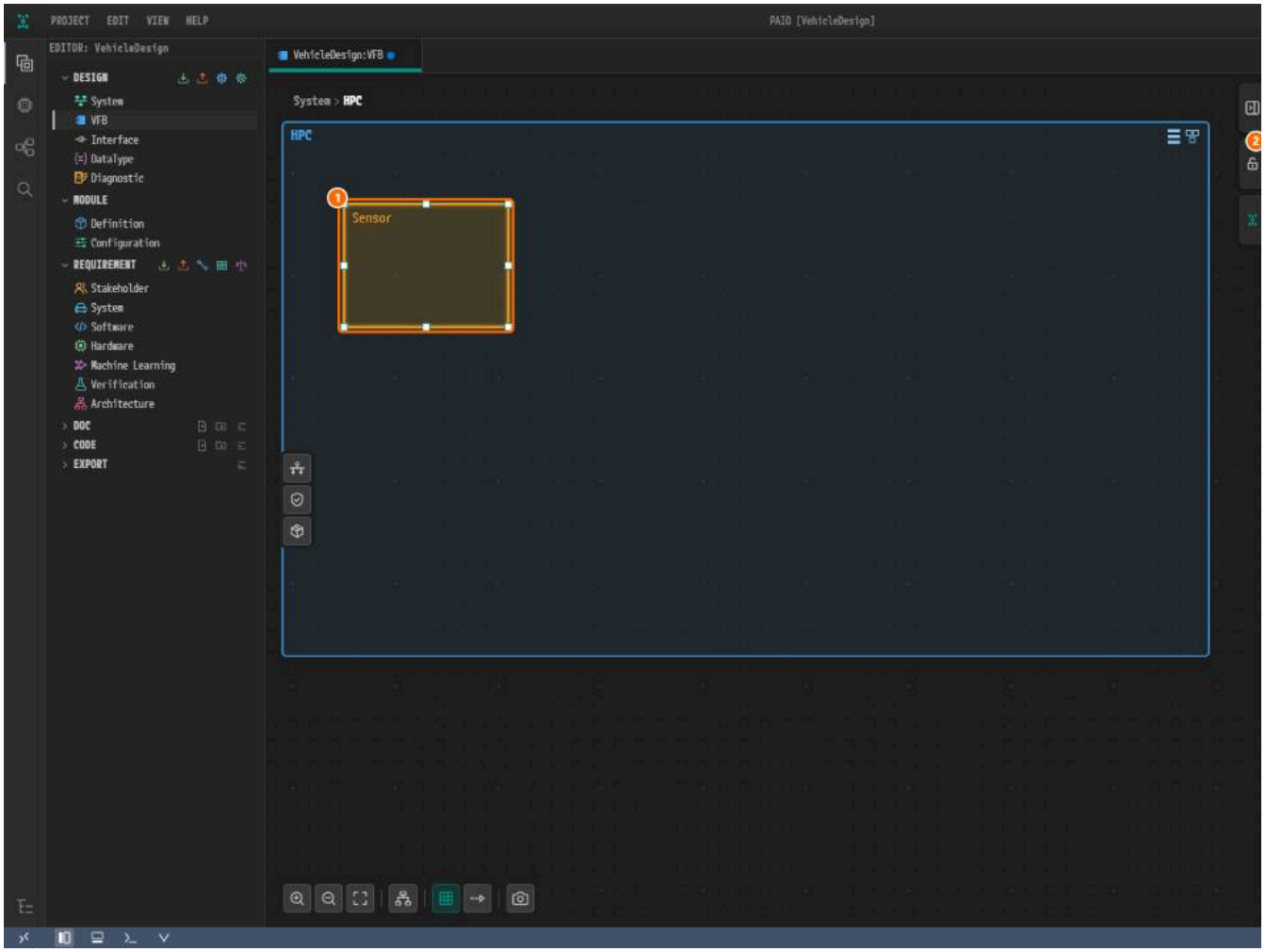


*그림 8-4. ① 노드 `Sensor`, ② **NAME**, ③ **AS PROTOTYPE**.*

Fusion 과 Actuator 만들기

같은 방법으로 표대로 `Fusion` 과 `Actuator` 를 만들어 `Sensor` 오른쪽에 둔다.

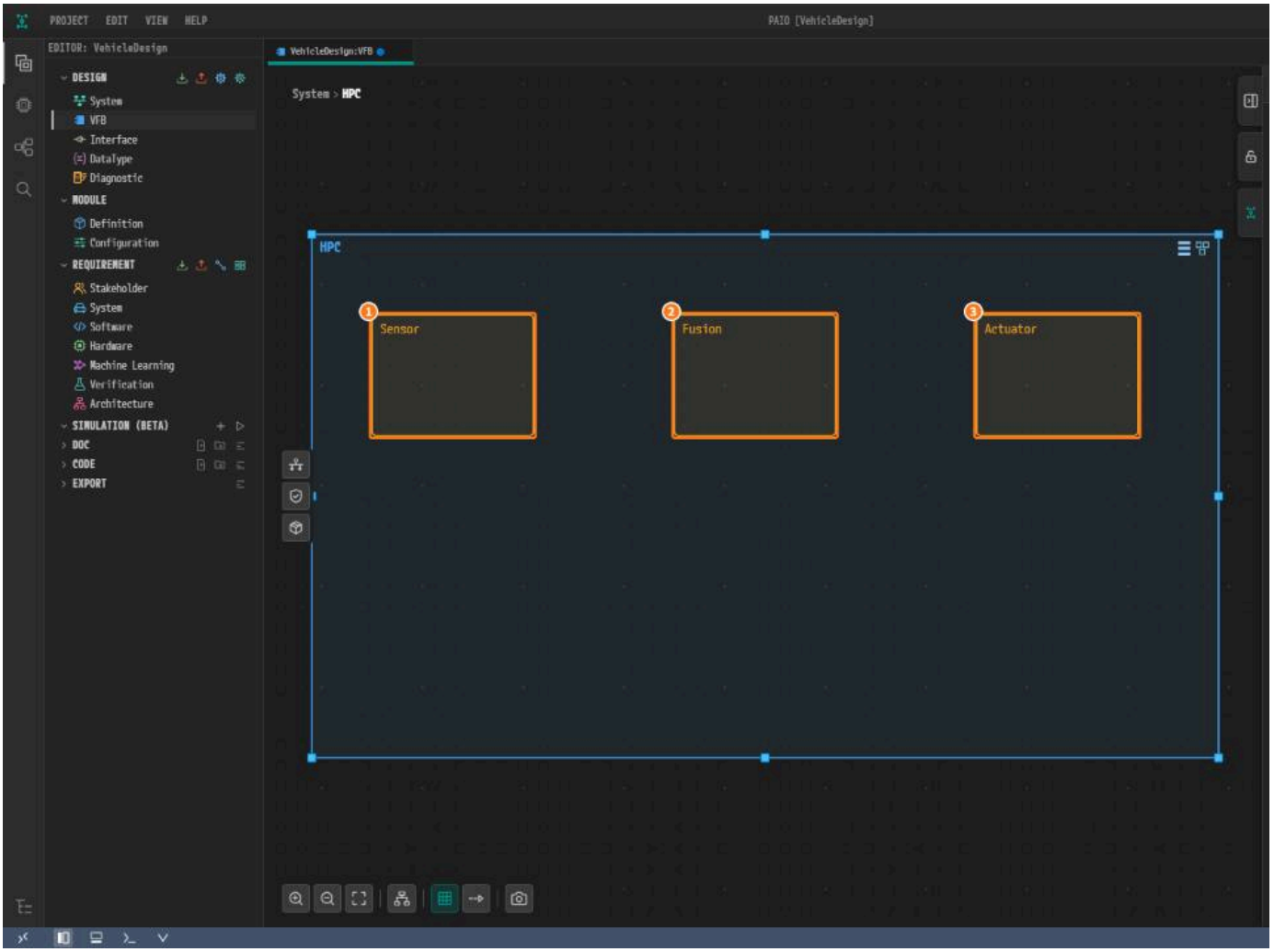


그림 8-5. ① `Sensor`, ② `Fusion`, ③ `Actuator`.

Sensor 에 P 포트 붙이기

1. `Sensor` 의 오른쪽 가장자리 근처를 오른쪽 클릭한다. **Add Port** 위에 포인터를 올리고 **PPort** 를 누른다.

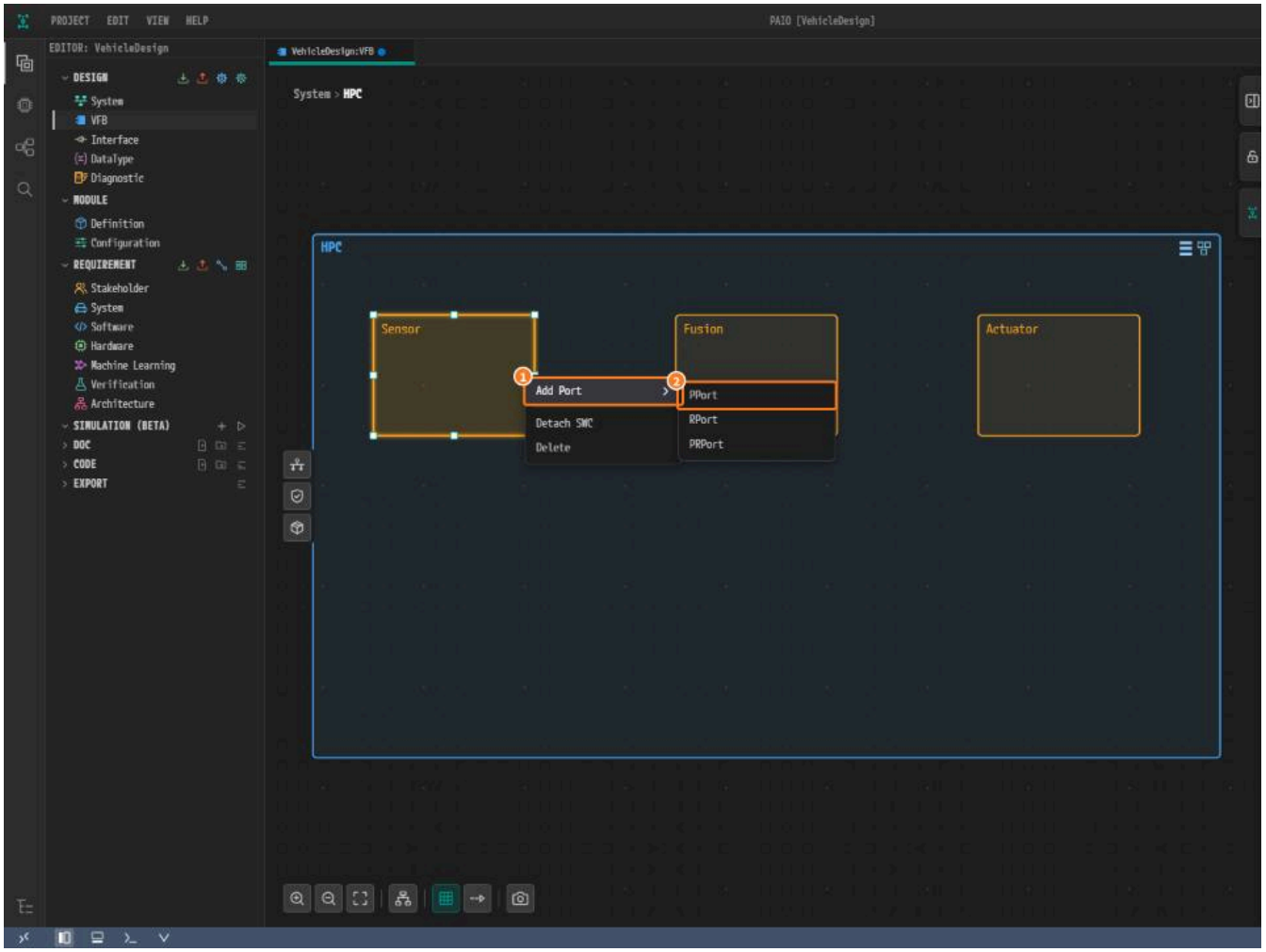


*그림 8-6. ① **Add Port**, ② **PPort**.*

2. 새 `P` 포트를 누른다. **NAME** 에 `Sensor` 를 입력하고 `Tab` 을 누른다.

3. **PORT INTERFACE** 를 누르고 유일한 항목 `Service: Sensor` 를 누른다. 고르기 전까지는 푸터의 **V** 아이콘에 빨간 배지 `1` 이 붙어 있다.

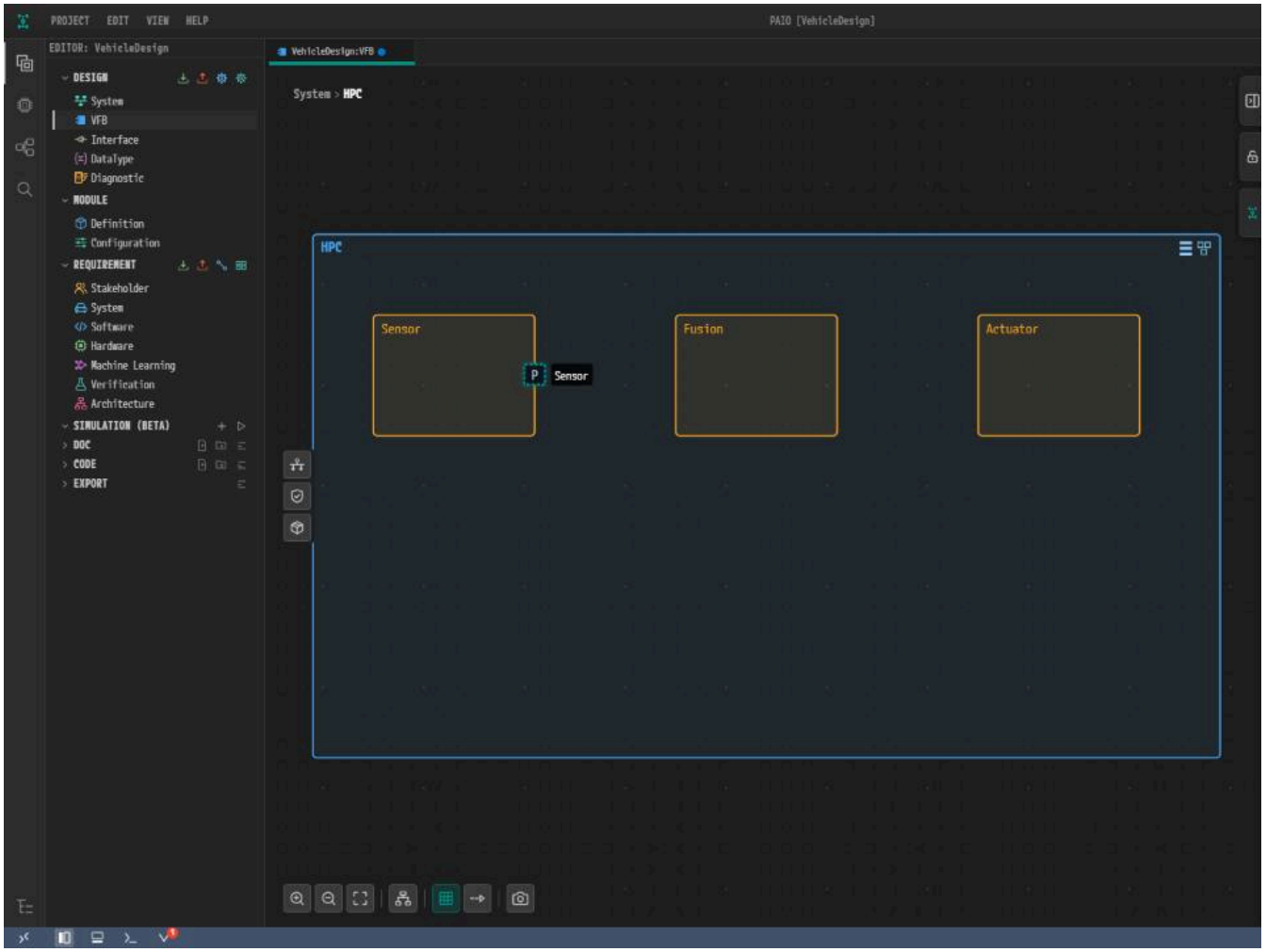


*그림 8-7. ① **NAME**, ② **PORT INTERFACE**, ③ `Service: Sensor`.*

4. **SERVICE COMSPEC** 섹션이 나타난다. 그대로 둔다.

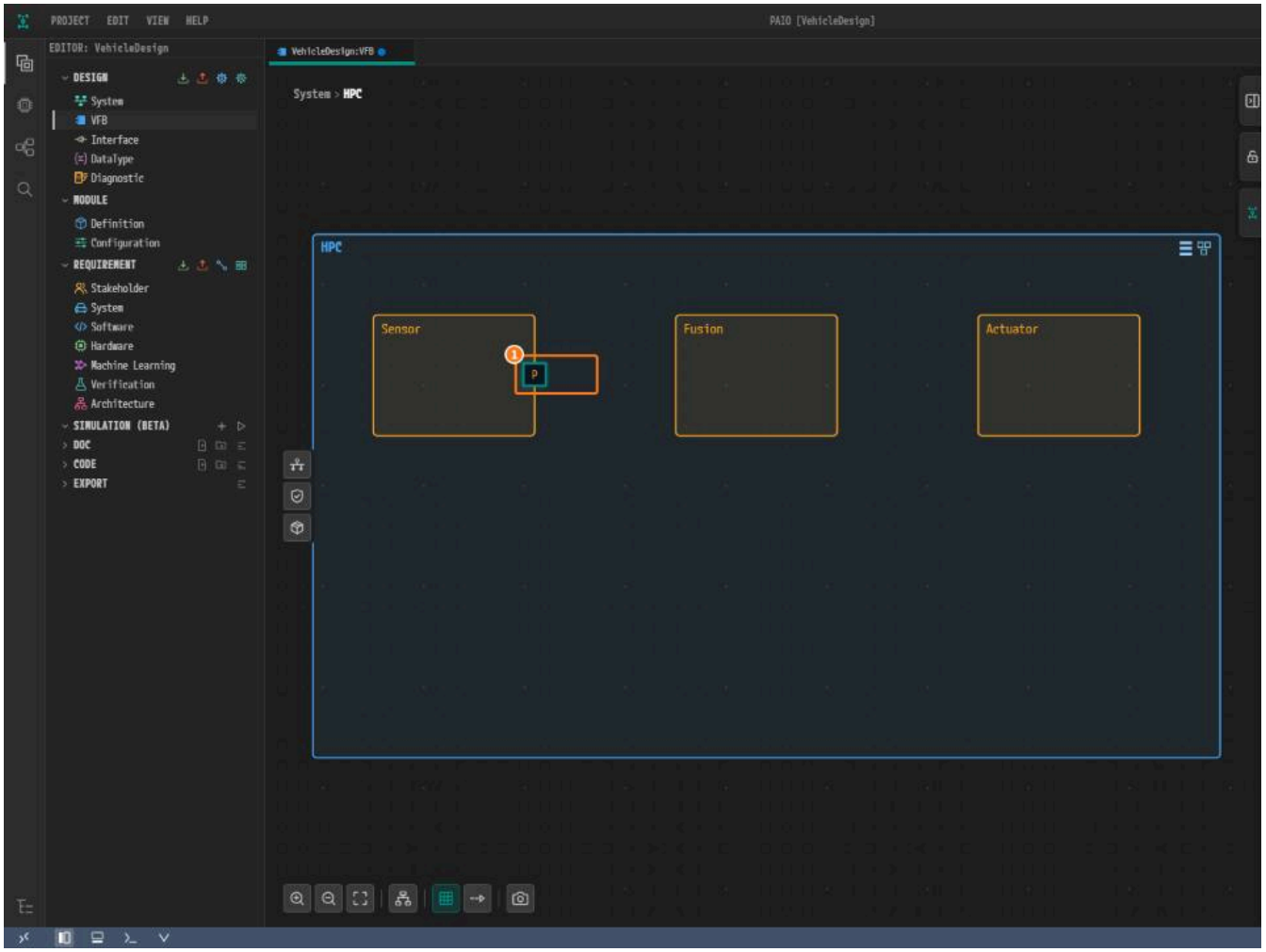


*그림 8-8. ① `P` 포트, ② **PORT INTERFACE** = `Service: Sensor`.*

Fusion 과 Actuator 에 R 포트 붙이기

1. `Fusion` 의 왼쪽 가장자리 근처를 오른쪽 클릭한다. **Add Port** > **RPort** 를 누른다. **NAME** 과 **PORT INTERFACE** 는 `Sensor` 와 같이 정한다.
2. `Actuator` 에도 똑같이 한다.

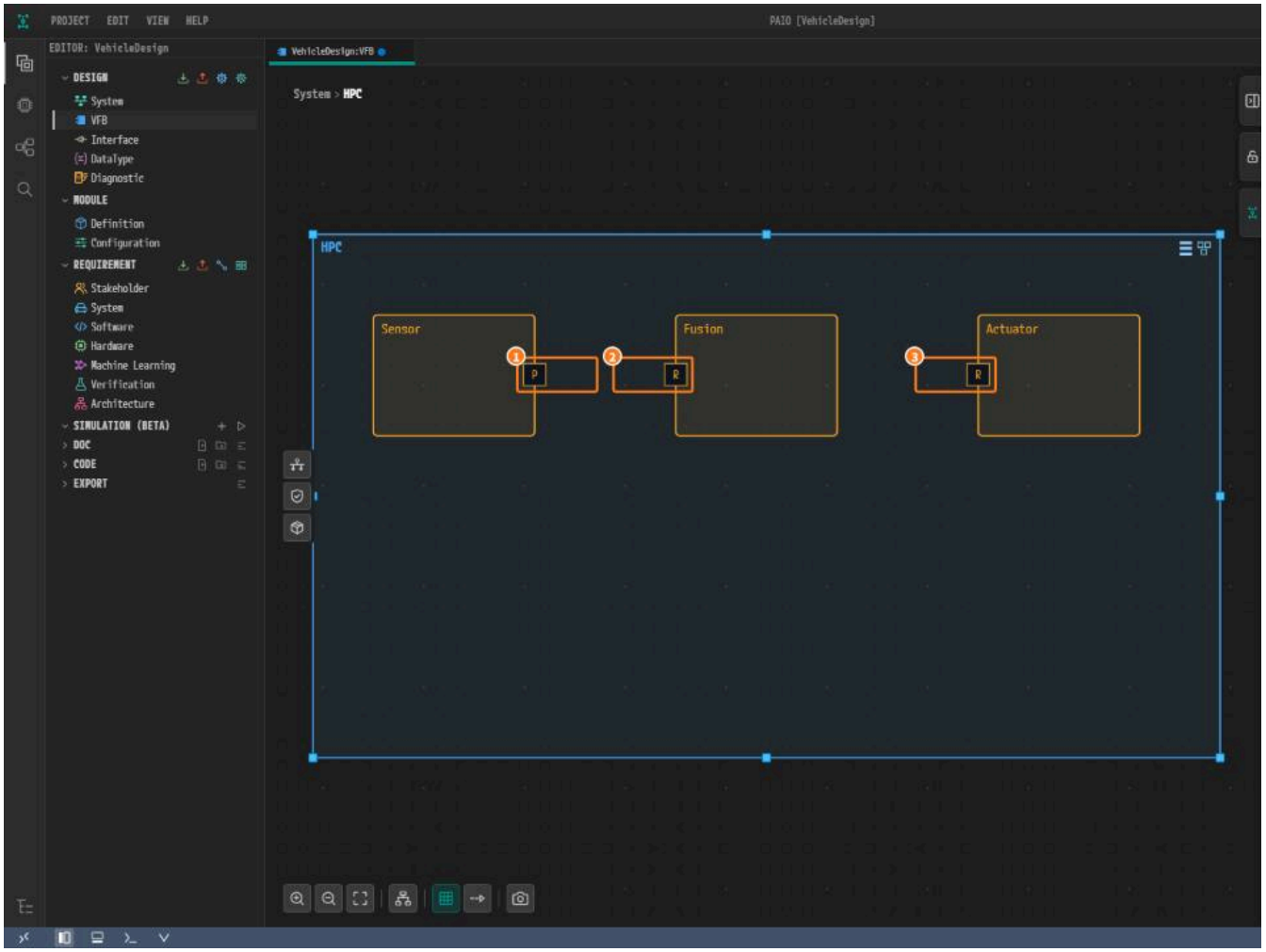


그림 8-9. ① `Sensor` 의 `P`, ② `Fusion` 의 `R`, ③ `Actuator` 의 `R`.

포트를 커넥터로 잇기

1. `Sensor` 의 `P` 포트를 오른쪽 클릭한다. **Connect to** 위에 포인터를 올리고 `[A] Fusion / Sensor` 를 누른다.

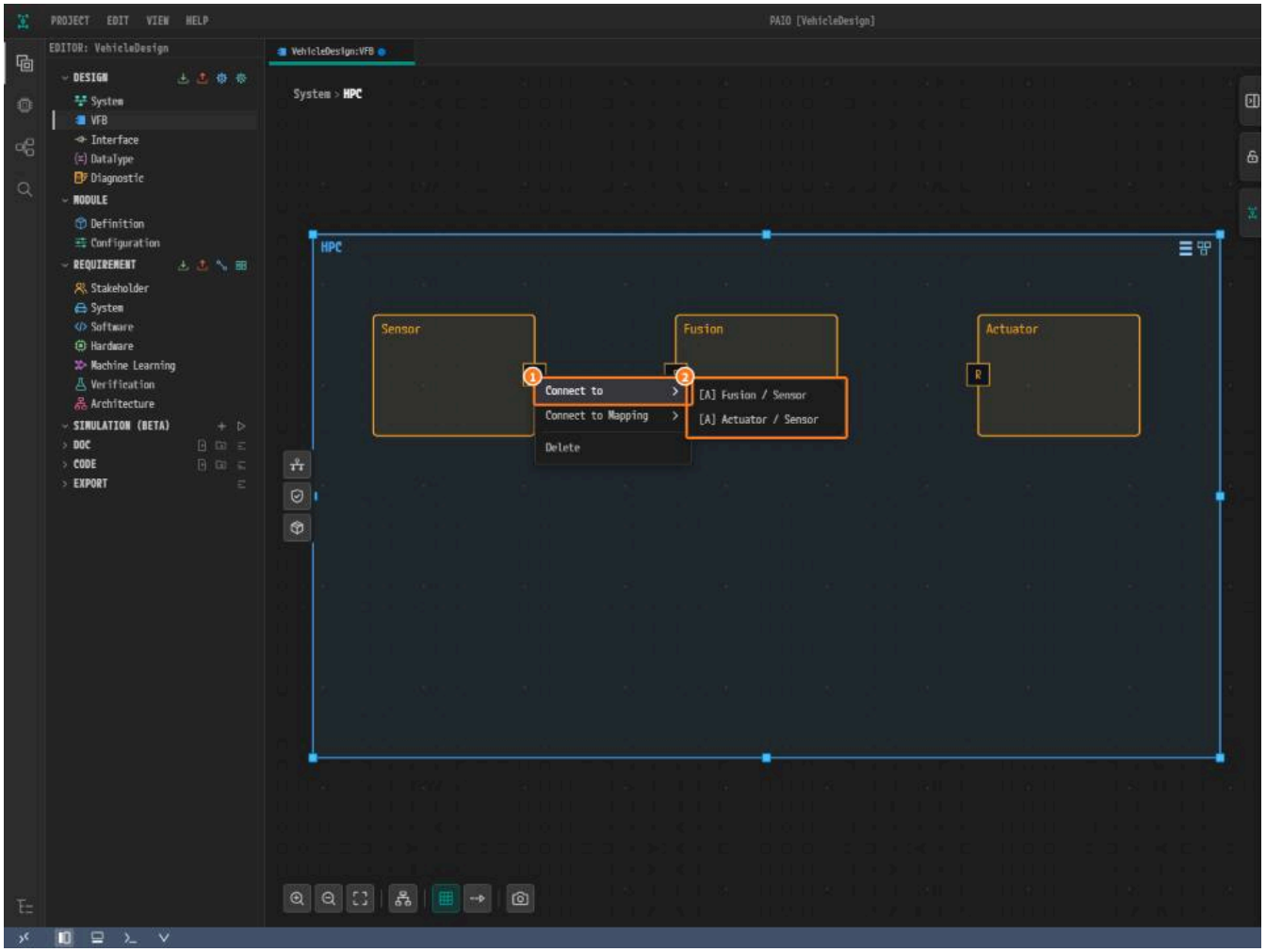


*그림 8-10. ① **Connect to**, ② 후보 목록.*

2. 같은 메뉴를 다시 열고 `[A] Actuator / Sensor` 를 누른다.

3. `Fusion` 으로 가는 선을 `Fusion` 의 R 포트 근처에서 누른다. **ASSEMBLY** 패널의 **NAME** 을 `Assembly_Sensor_Sensor_Fusion_Sensor` 에서 `Assembly_Sensor_Fu`

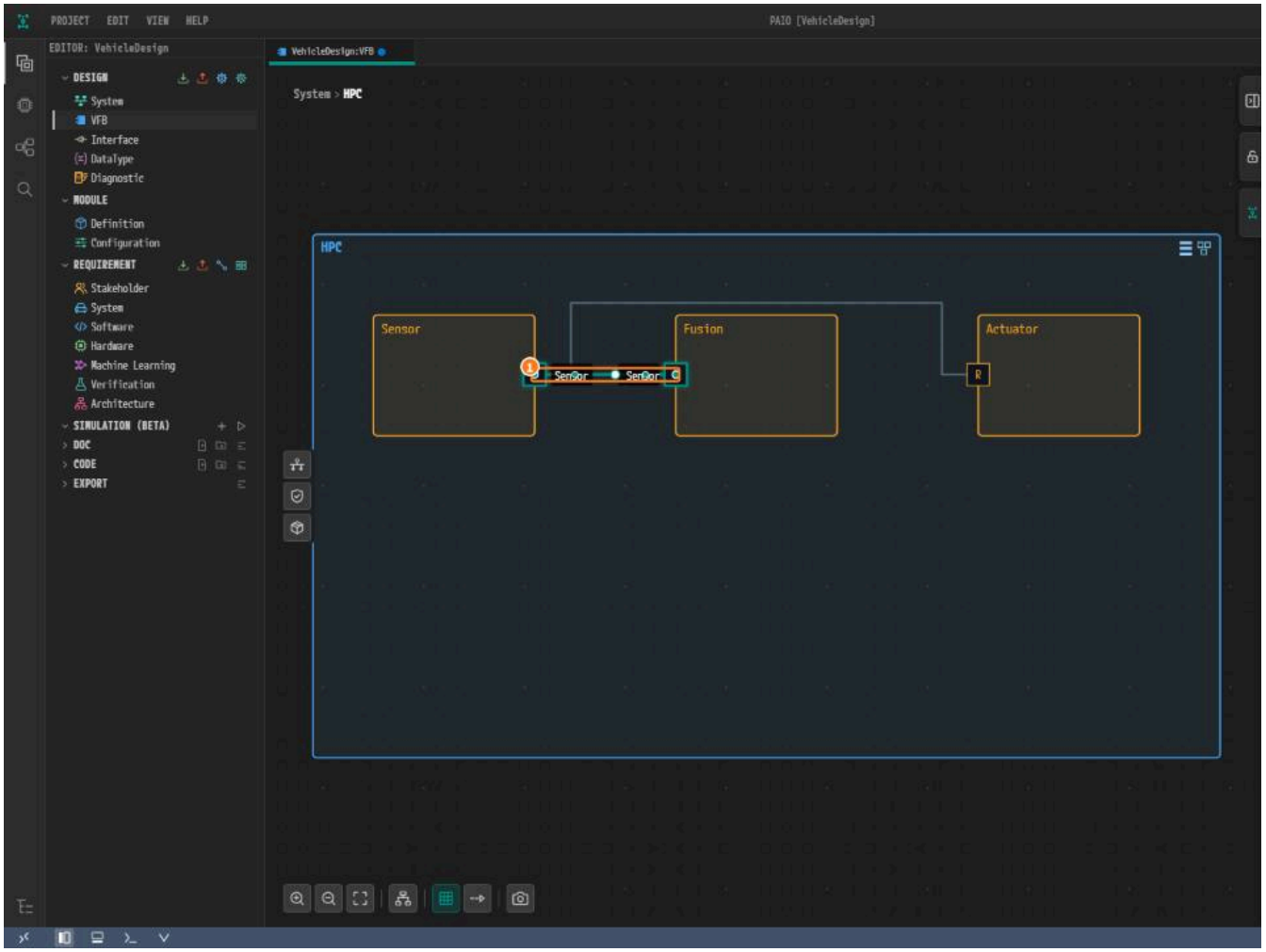


*그림 8-11. ① 누른 선, ② **NAME** = `Assembly_Sensor_Fusion`.*

4. 같은 방법으로 `Actuator` 로 가는 선의 이름을 `Assembly_Sensor_Sensor_Actuator_Sensor` 에서 `Assembly_Sensor_Actuator` 로 바꾼다.

Auto Layout 으로 정돈하고 저장하기

1. 캔버스 아래 도구 모음의 **Auto Layout** 을 누른다. 포트가 반대쪽으로 옮겨질 수 있다.
2. 캔버스의 빈 곳을 누르고 `Ctrl+S` 를 누른다.

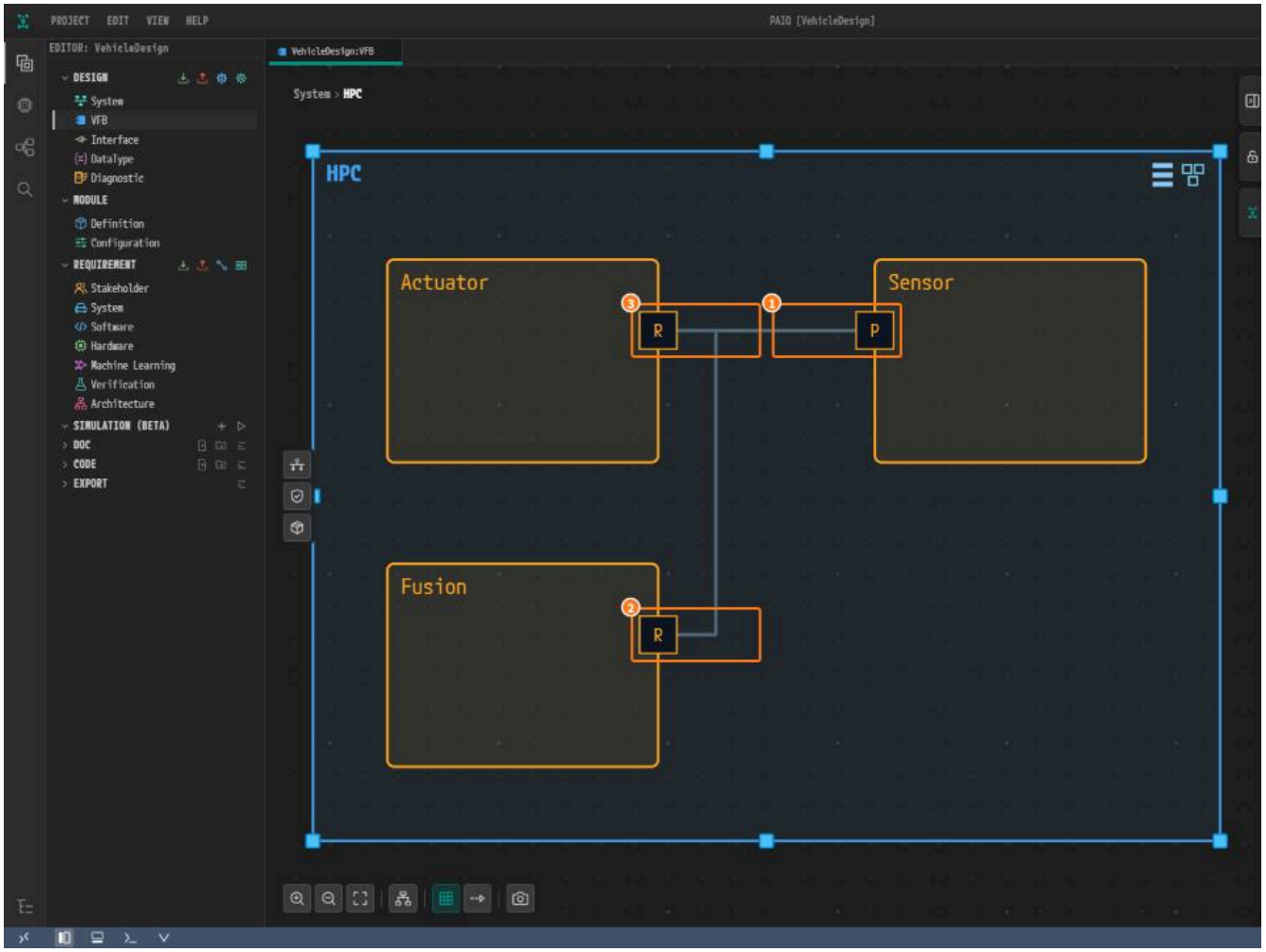


그림 8-12. ① `Sensor` 의 P 포트, ② `Fusion` 의 R 포트, ③ `Actuator` 의 R 포트. 위치는 Auto Layout 이 정하므로 독자의 화면과 다를 수 있다.

검증 결과 보기

1. **VALIDATION** 패널을 본다. 닫혀 있으면 푸터의 **V** 아이콘을 누른다. `No validation issues.` 가 보인다.

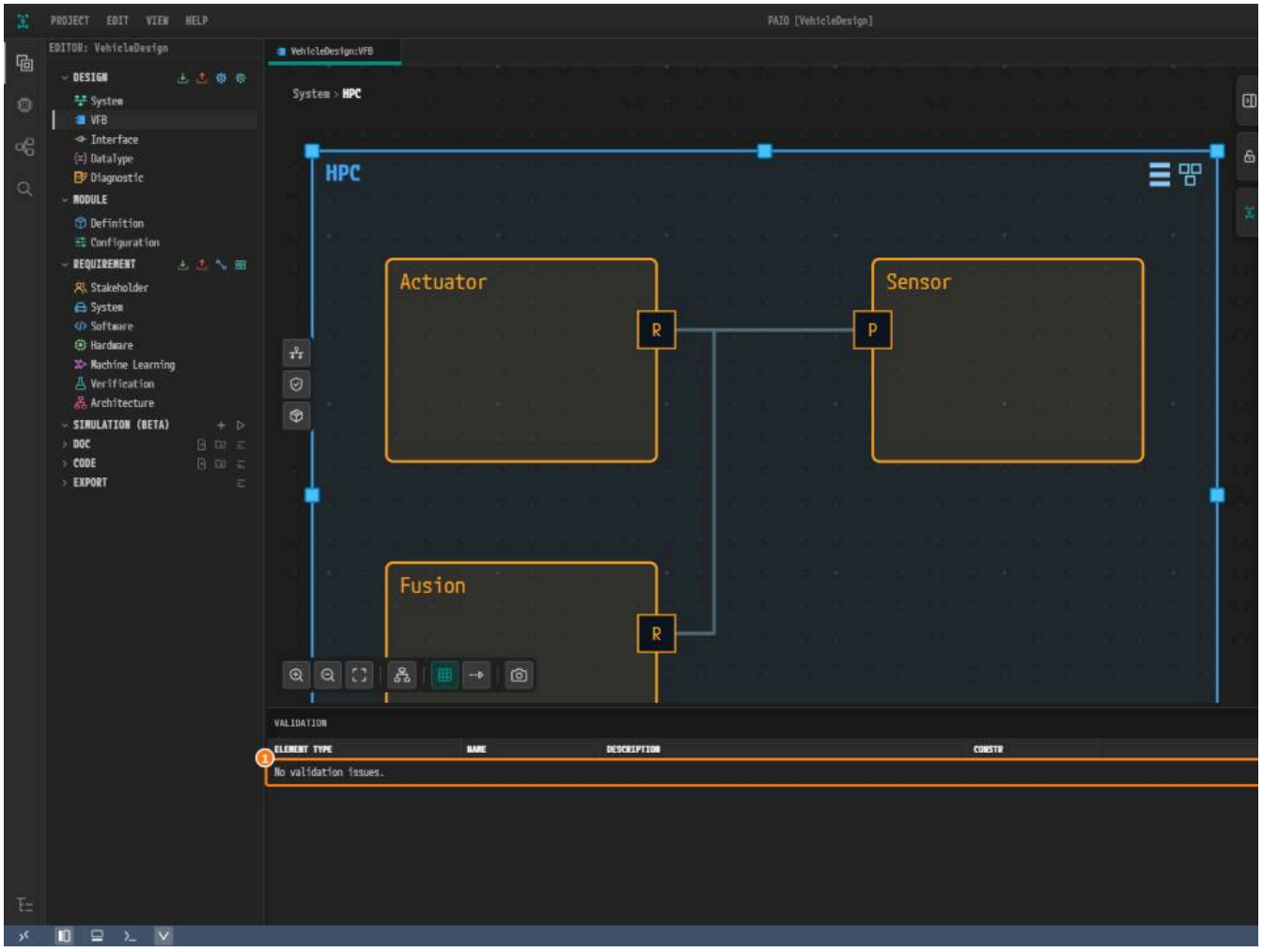


*그림 8-13. 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- **프로토타입 이름 = 타입 이름.** `HPC` 안의 프로토타입은 타입과 같은 이름을 가지므로 **AS PROTOTYPE** 에 **NAME** 과 같은 값을 넣는다.
- **포트 세 개의 이름은 모두** `Sensor` 이다. 인터페이스 이름을 따른다.
- **제공자 하나, 소비자 둘.** `Sensor` 가 `Fusion` 과 `Actuator` 로 갈라져 나간다. `Fusion` 을 거치는 사슬이 아니다.
- **짧은 커넥터 이름.** 기본 이름에는 포트 이름도 들어 있다. 이 설계는 커넥터를 두 프로토타입 이름으로만 부른다.
- **내 애플리케이션만 만든다.** `EM`, `CM` 같은 플랫폼 애플리케이션은 다음 장에서 PARA 가 가져온다. 직접 그리지 않는다.
- **레이아웃은 설계가 아니다.** 위치는 `HPC.vfb.layout` 에만 저장되고 익스포트되지 않는다.

확인하기

- [ ] 프레임 `HPC` 에 `Sensor`, `Fusion`, `Actuator` 가 있고, 각각 **NAME** 과 **AS PROTOTYPE** 이 같다.
- [ ] `Sensor` 에 `P` 포트 하나, `Fusion` 과 `Actuator` 에 `R` 포트 하나씩 있고, 모두 이름이 `Sensor`, **PORT INTERFACE** 가 `Service: Sensor` 이다.
- [ ] 선 두 개의 이름이 `Assembly_Sensor_Fusion` 과 `Assembly_Sensor_Actuator` 이다.
- [ ] 탭 `VehicleDesign:VFB` 에 수정 표시 ● 가 없다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

PARA 설정에서 `EM` 과 `CM` 을 비롯한 PARA 의 플랫폼 애플리케이션을 머신에 가져온다.

PARA 설정

이 장에서 할 일

이 장에서는 **PARA Platform Import** 로 PopcornSAR 의 AUTOSAR Adaptive Platform 인 PARA 의 플랫폼을 머신 `HPC` 에 가져온다. 내 애플리케이션과 함께 도는 플랫폼 애플리 `CM`(통신 관리)이 있다. 그다음 이 설계가 쓰지 않는 **Network Management** 모듈을 끈다.

시작하기 전에

- Adaptive Application 과 포트를 마친 뒤 시작한다. 탭 `VehicleDesign:VFB` 는 저장되어 있다.
- PARA 는 PAIO 에 들어 있다. 따로 준비할 파일은 없다.

위치	항목	값
Conflict Resolution 창	**ECU ID** (`ECU1` / `HPC`)	**Keep Project**
Conflict Resolution 창	**Max Action List Duration** (`1000` / `5`)	**Keep Project**
Machine Configuration [HPC] 의 **MODULE**	**Network Management**	**Disable**

PARA 플랫폼 가져오기

1. 탭 `VehicleDesign:VFB` 에서 컴포지션 `HPC` 안(`System > HPC`)으로 들어간다. 프레임 오른쪽 위의 머신 아이콘을 누른다. **Machine Configuration [HPC]** 가 열린다.

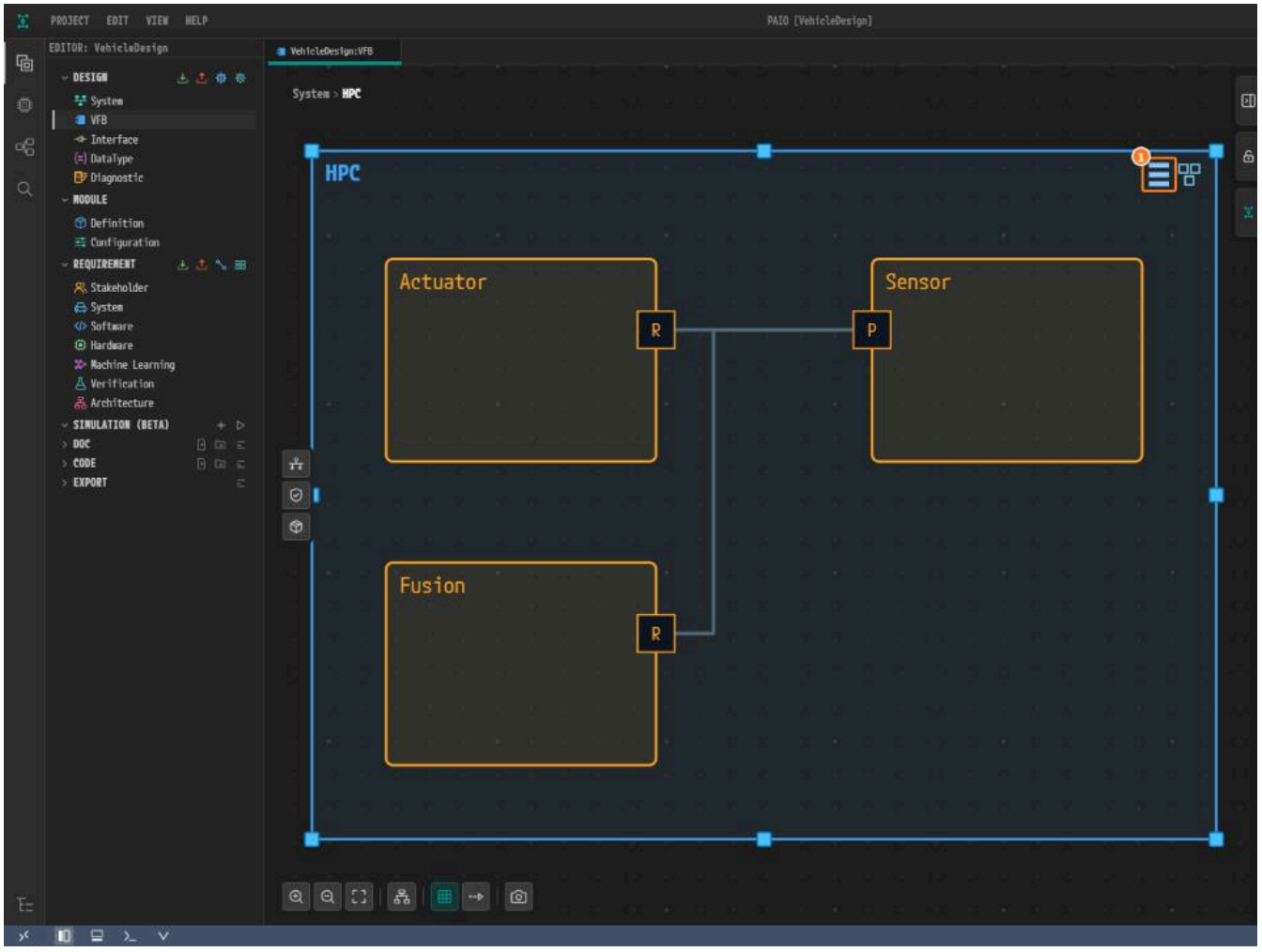


그림 9-1. ① 머신 아이콘.

2. 창 오른쪽 위의 **PARA Platform Import** 를 누른다.

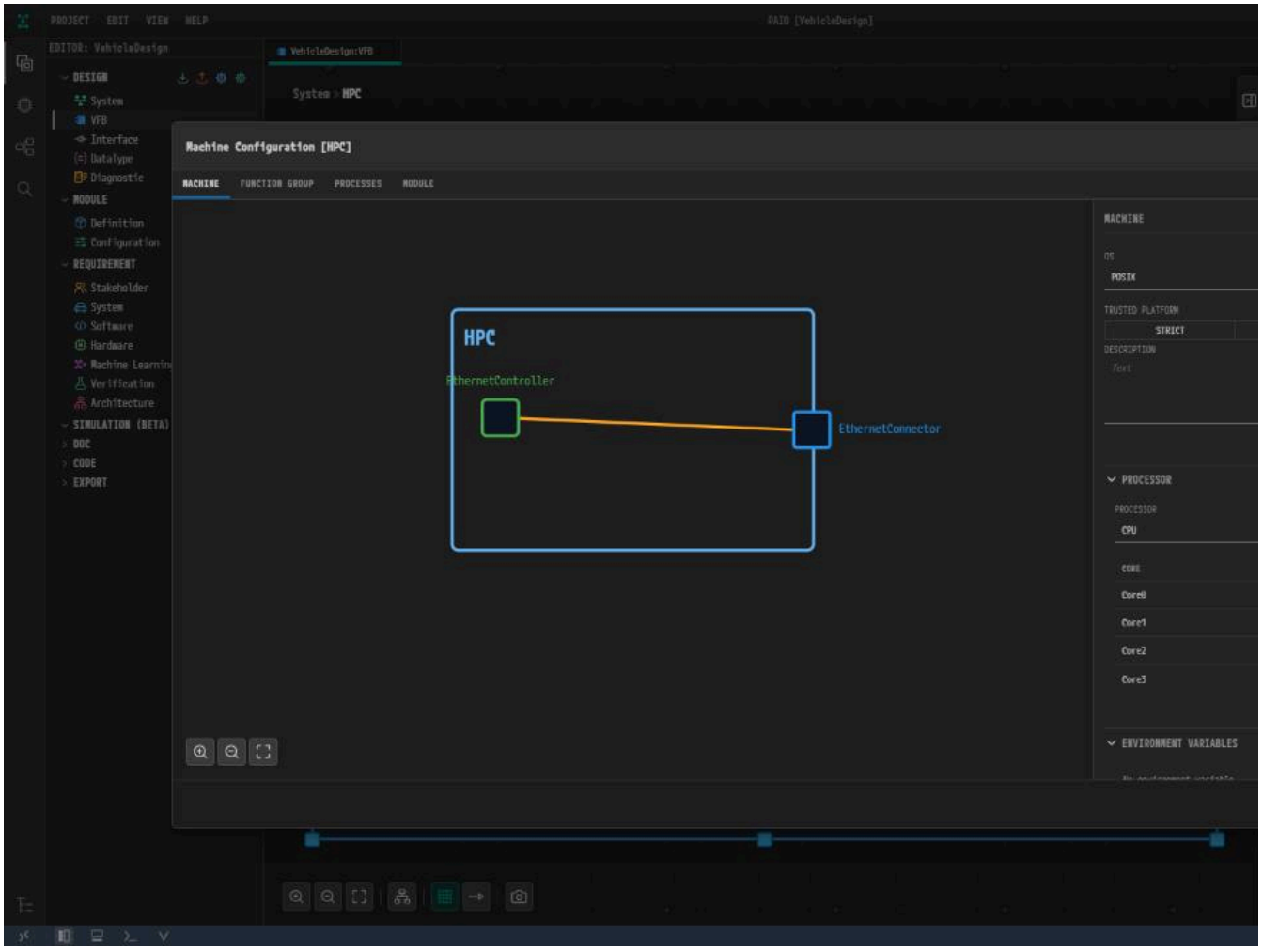


*그림 9-2. ① **PARA Platform Import.***

3. 확인 창에서 **Import** 를 누른다.

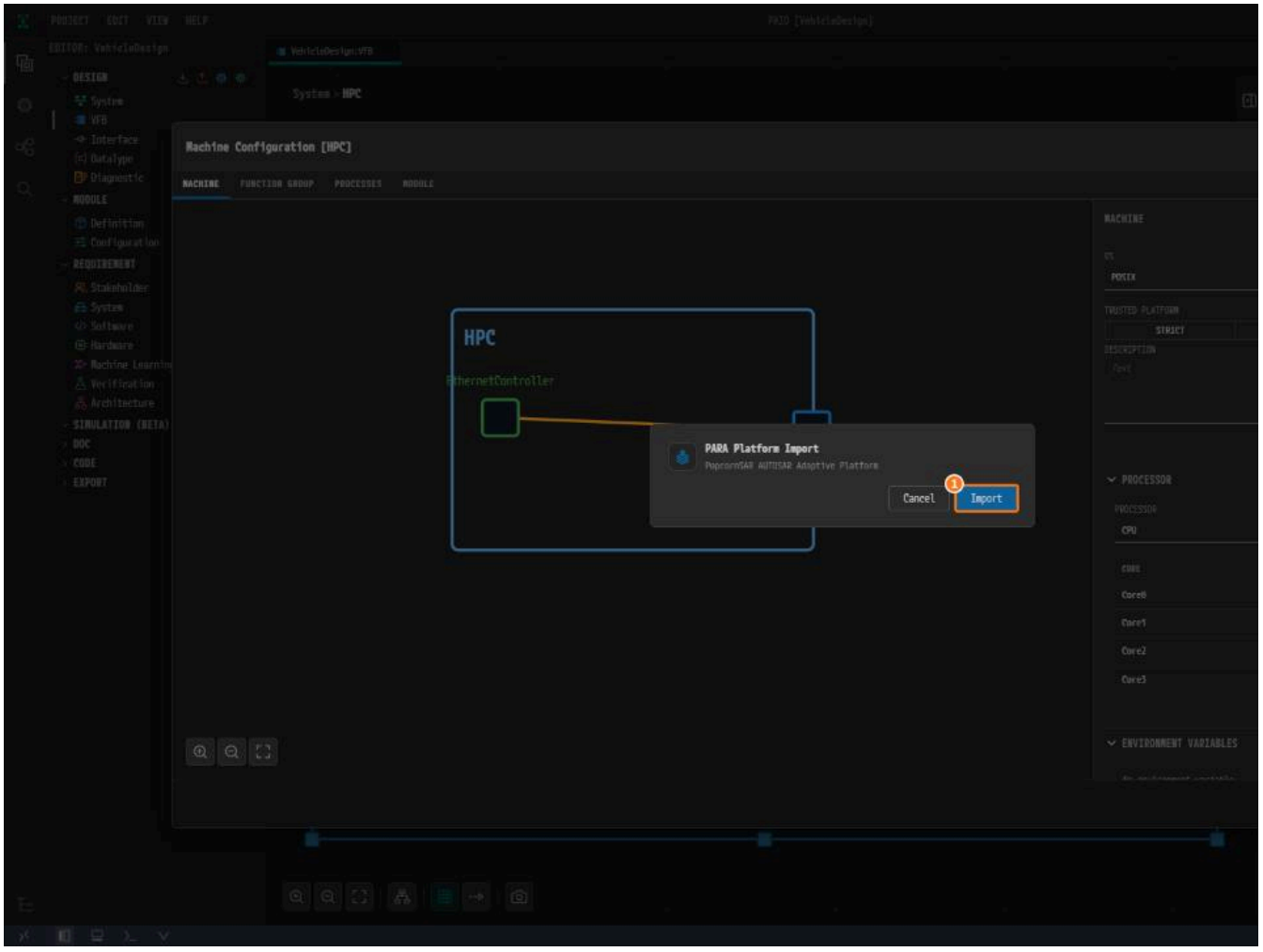


*그림 9-3. ① **Import**.*

4. **Conflict Resolution** 창이 열린다. 머신 설정에서 입력한 값 두 개가 PARA 의 값과 다르기 때문이다. **ECU ID** 와 **Max Action List Duration** 아래의 **Keep Project** 를 각각 누

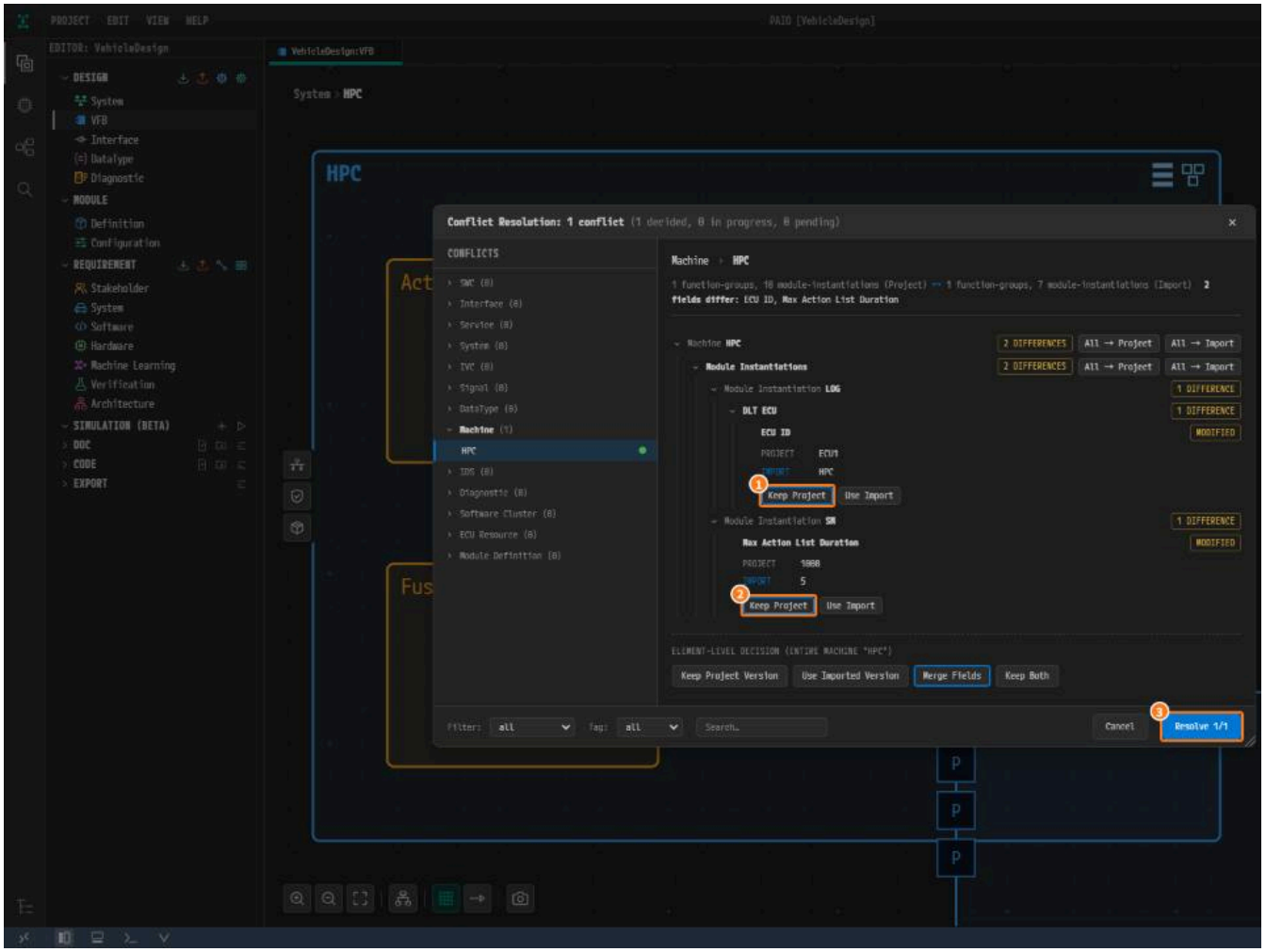


*그림 9-4. ① **ECU ID** 의 **Keep Project**, ② **Max Action List Duration** 의 **Keep Project**, ③ `Resolve 1/1`.*

5. **Machine Configuration [HPC]** 창으로 돌아오고 오른쪽 아래에 `Import complete` 가 보인다. 탭 **PROCESSES** 를 누른다. 목록에 플랫폼 프로세스 열한 개 `EM`, `CM`, `PHM`

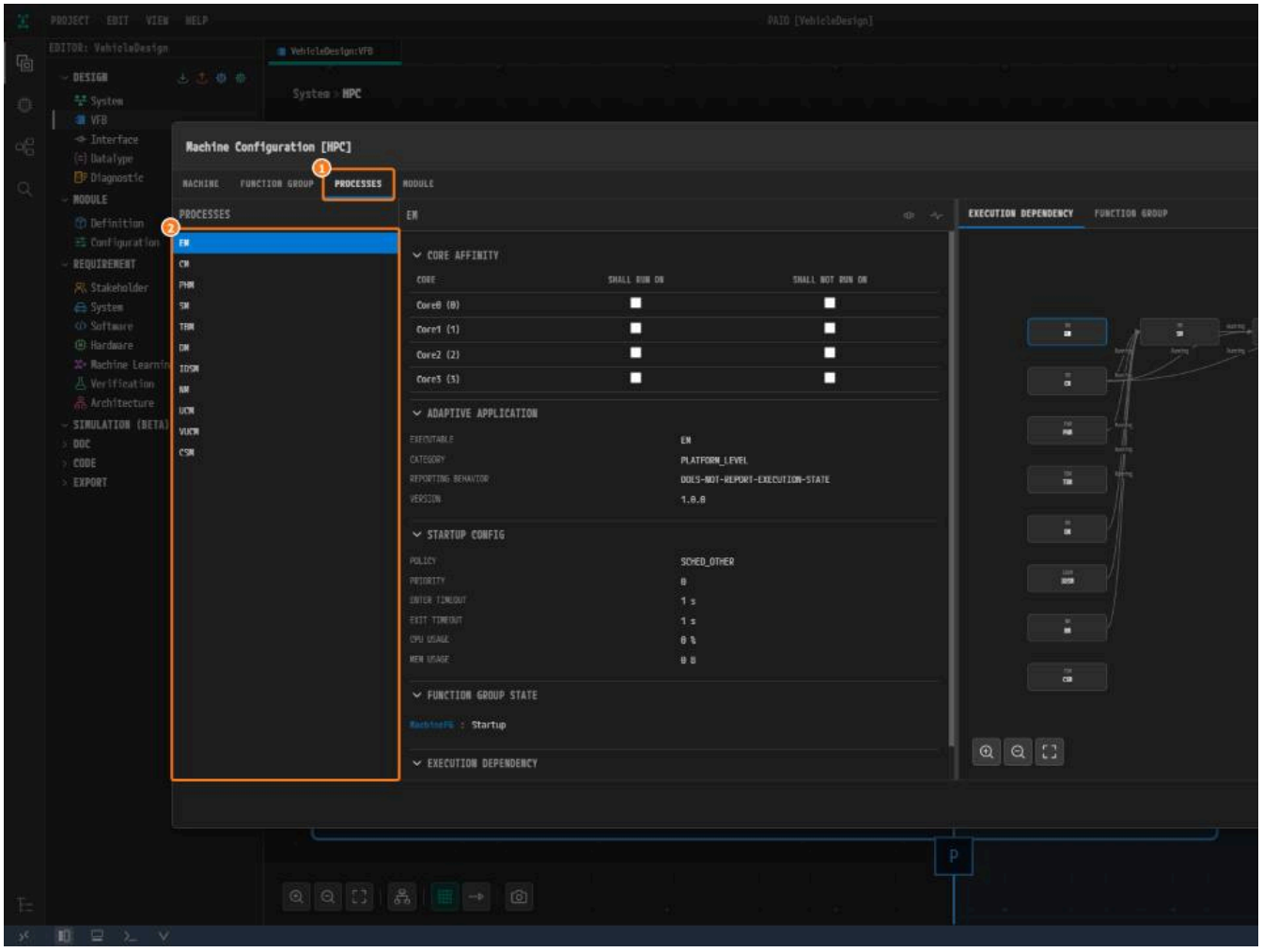


*그림 9-5. ① **PROCESSES**, ② 플랫폼 프로세스 열한 개.*

Network Management 끄기

1. 탭 **MODULE** 을 누른다. **Network Management** 를 오른쪽 클릭하고 **Disable** 을 누른다.

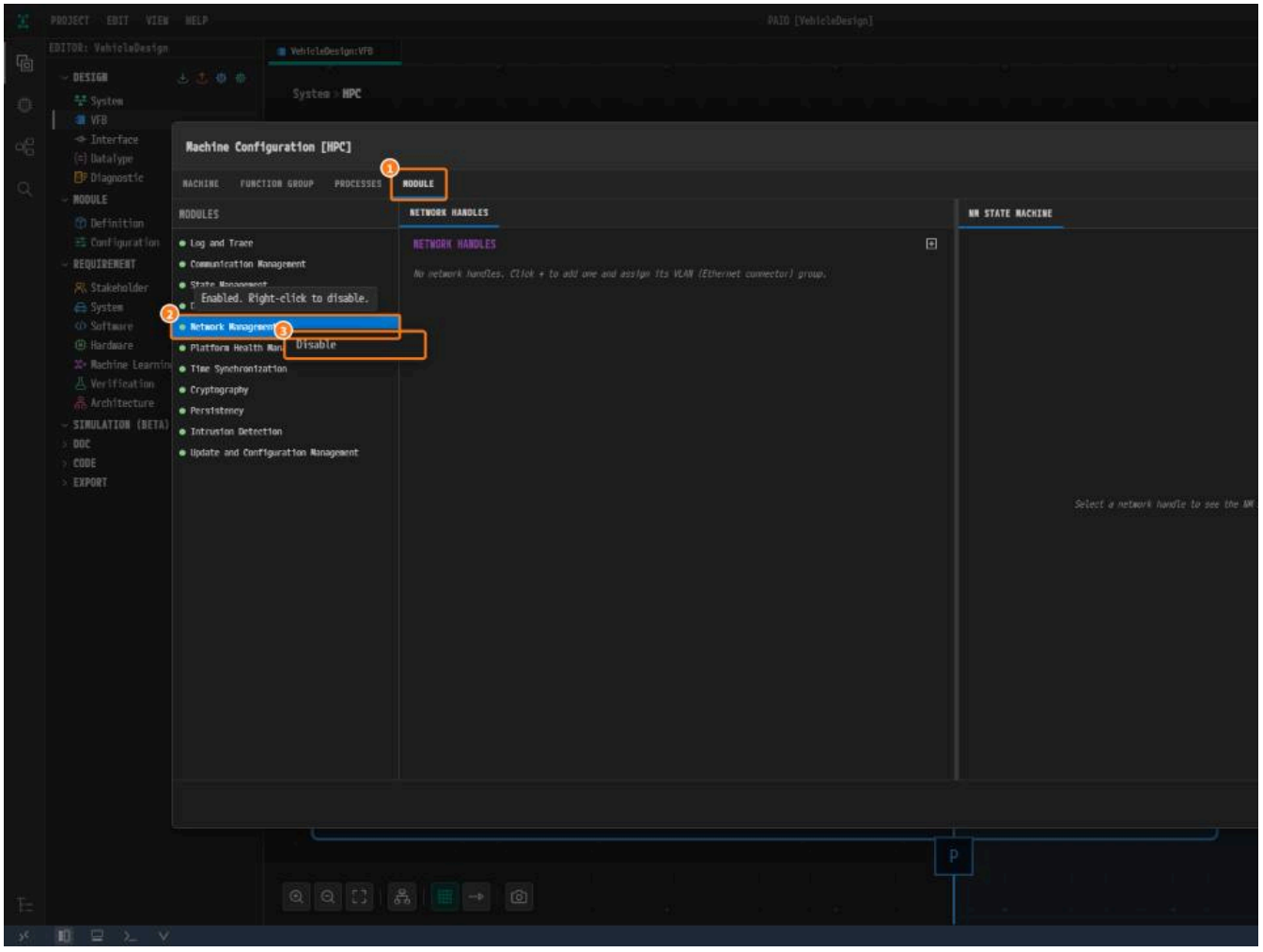


*그림 9-6. ① **MODULE**, ② **Network Management**, ③ **Disable**.*

2. **Network Management** 앞의 점이 비고 패널이 흐려진다. 창에서 `Ctrl+S` 를 누르고 **Close** 를 누른다.

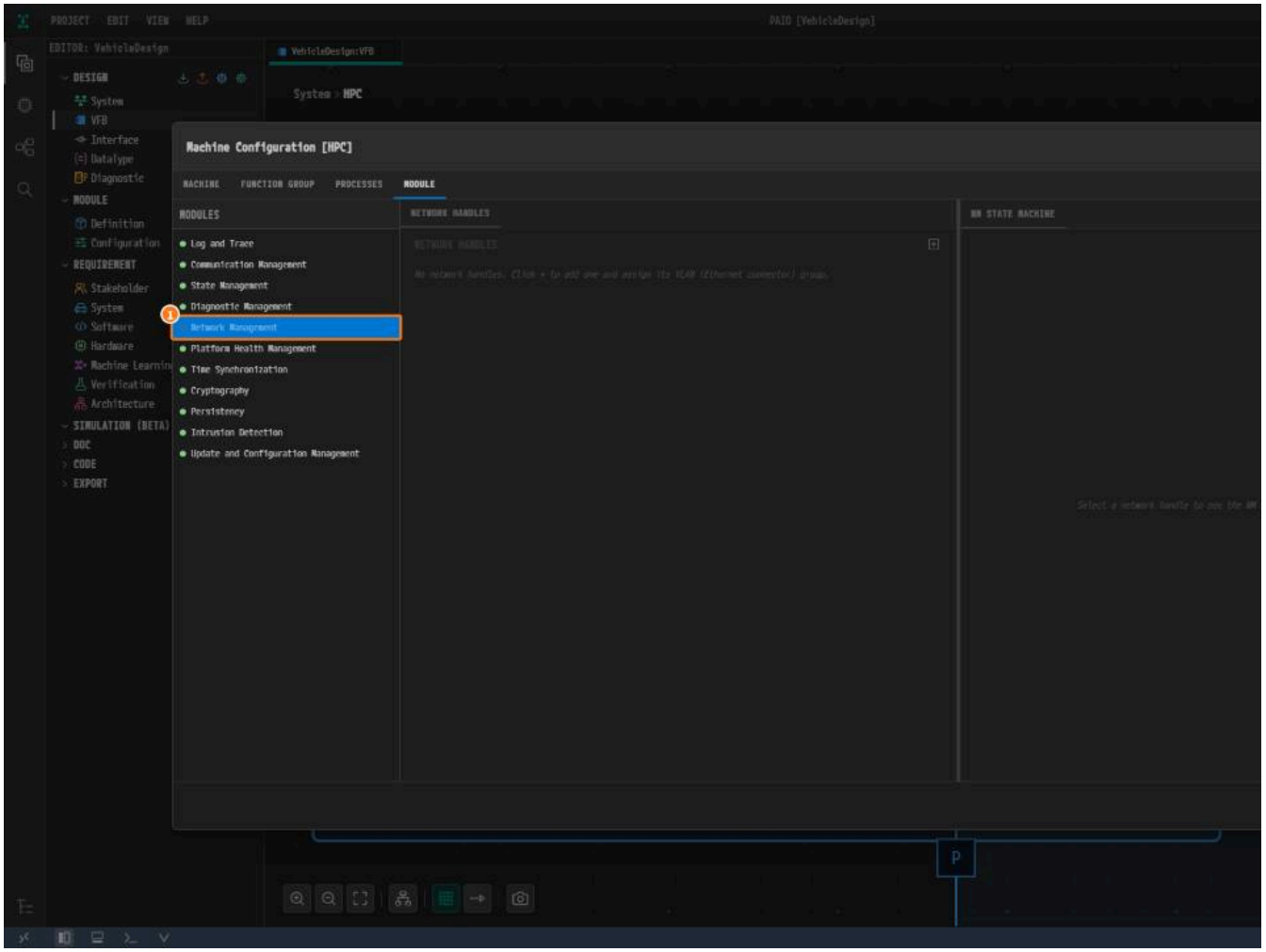


*그림 9-7. ① 꺼진 **Network Management**.*

Auto Layout 으로 정돈하고 저장하기

1. 가져오기는 컴포지션 `PARA_HPC` 도 캔버스에 놓는데, 일부가 프레임 밖에 있다. 캔버스 아래 도구 모음의 **Auto Layout** 을 누른다.

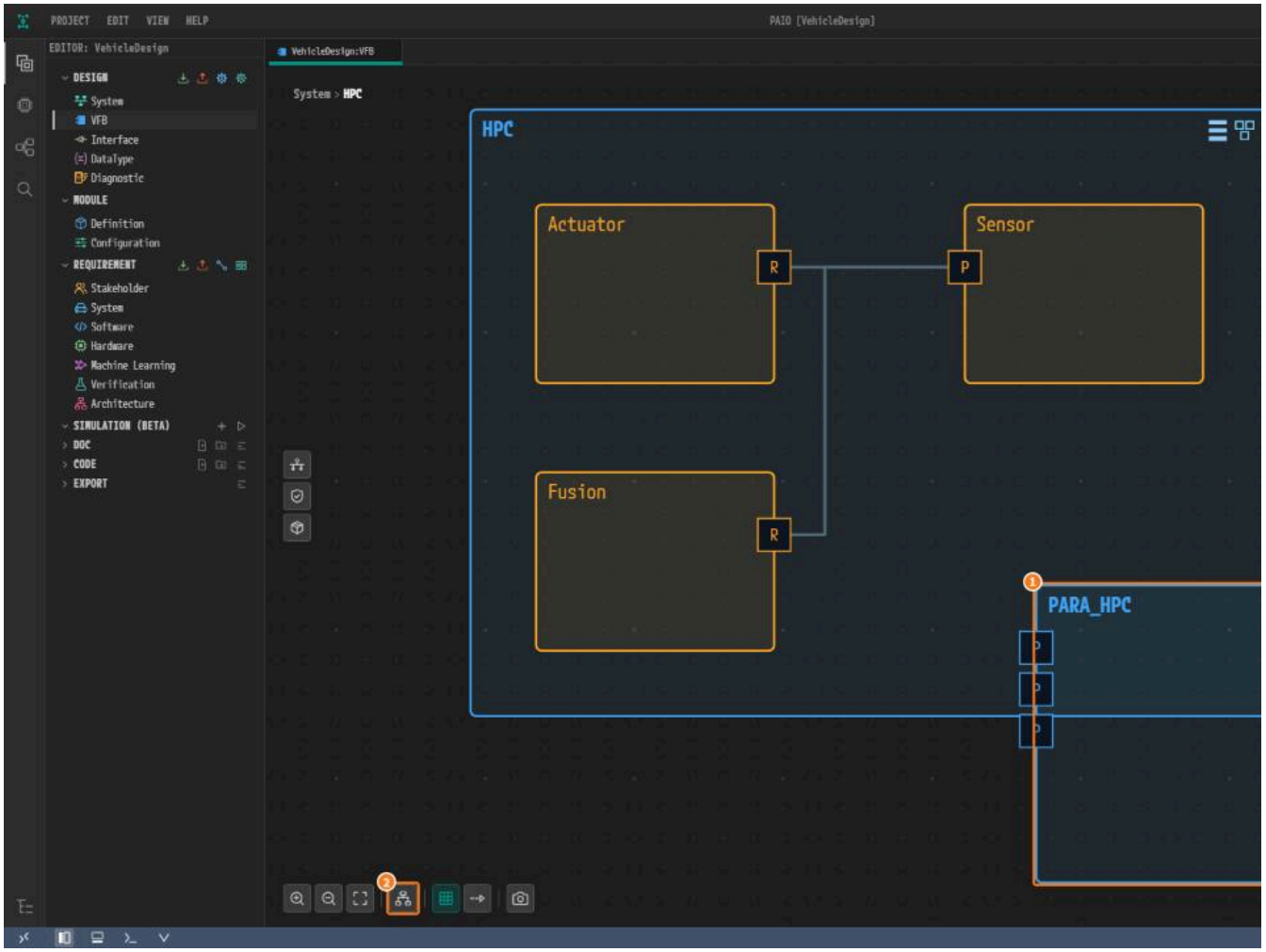


*그림 9-8. ① `PARA_HPC`, ② **Auto Layout**.*

2. 캔버스의 빈 곳을 누르고 `Ctrl+S` 를 누른다.

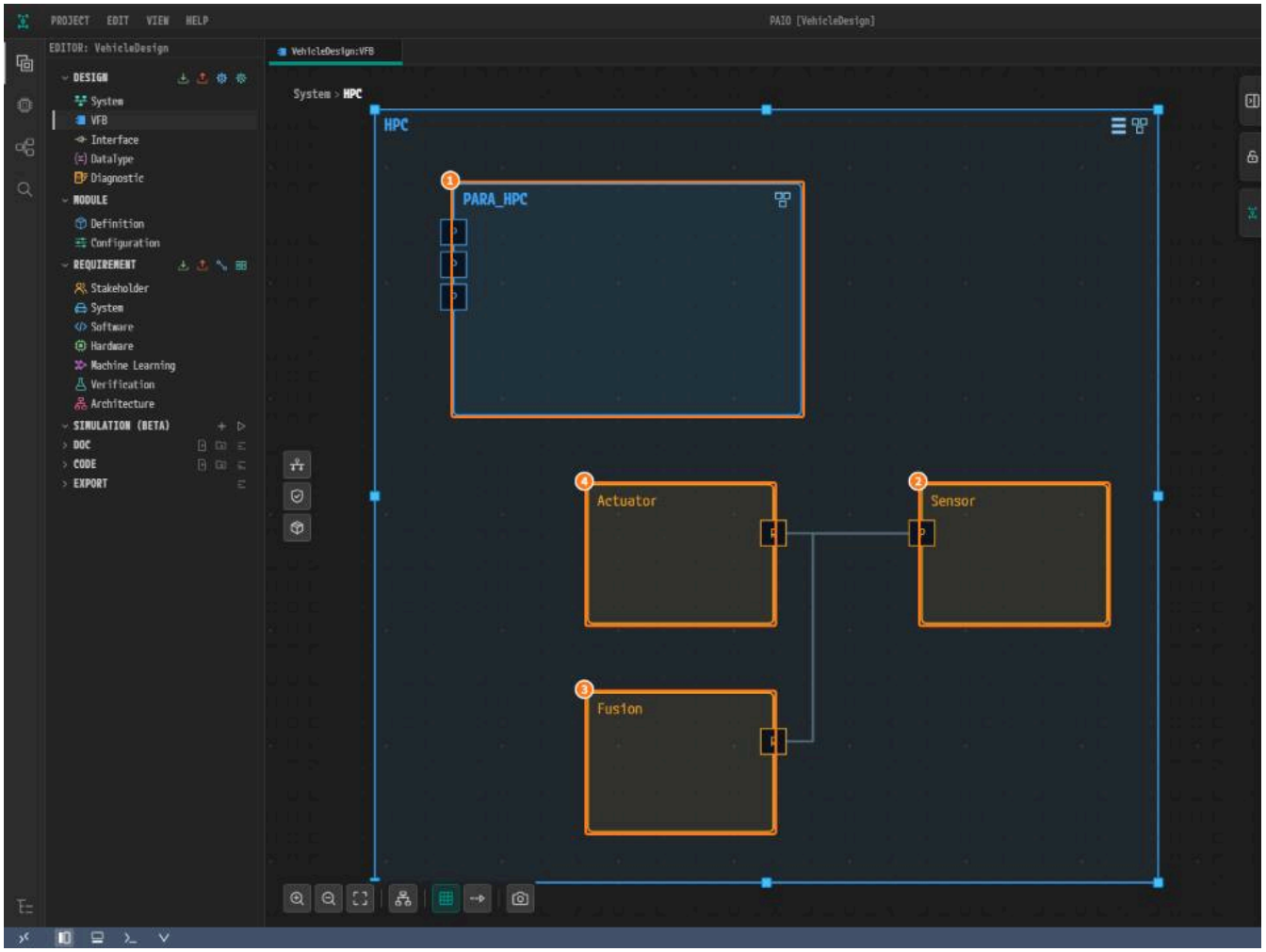


그림 9-9. ① `PARA_HPC`, ② `Sensor`, ③ `Fusion`, ④ `Actuator`. 위치는 Auto Layout 이 정하므로 독자의 화면과 다를 수 있다.

> **참고.** `PARA_HPC` 에는 플랫폼 애플리케이션 열한 개가 들어 있다. 안의 내용은 그대로 둔다. 플랫폼의 인터페이스와 데이터 타입도 들어오므로, Interface 편집기와 DataType 편

검증 결과 보기

1. **VALIDATION** 패널을 본다. 닫혀 있으면 푸터의 **V** 아이콘을 누른다. `No validation issues.` 가 보인다.

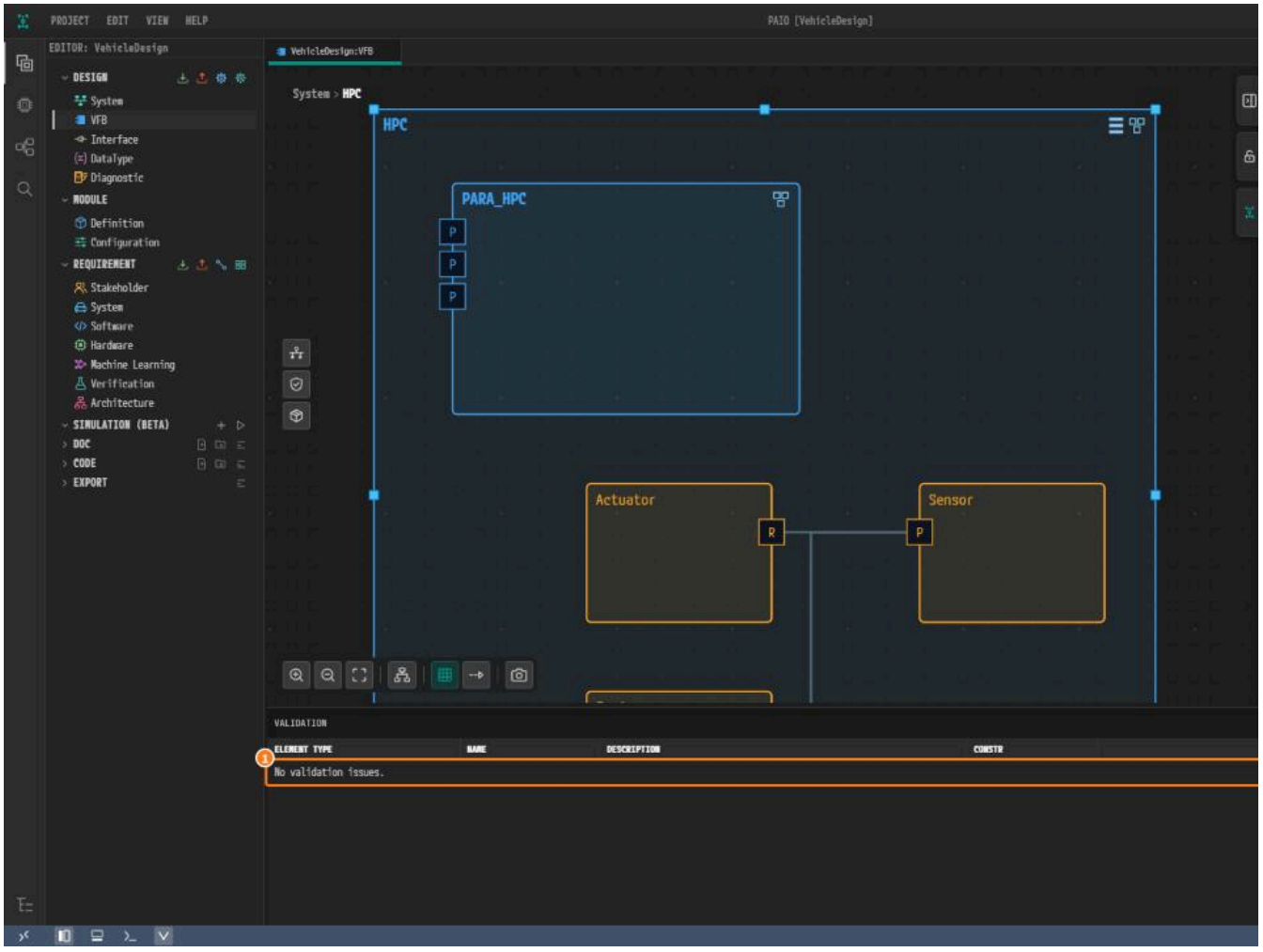


*그림 9-10. 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- **더 앞이 아니라 여기.** 8장에서 포트의 인터페이스 목록에 항목이 하나만 나오는 것은 PARA 의 인터페이스가 아직 프로젝트에 없기 때문이다.
- **더 뒤가 아니라 여기.** 다음 장에서 `Sensor`, `Fusion`, `Actuator` 가 기다리는 프로세스 `CM` 이 이 가져오기로 들어온다.
- **플랫폼은 그리지 않고 가져온다.** `EM` 이나 `CM` 을 직접 만들면 PARA 의 것과 나란히 놓이고, 검증이 중복을 보고한다.
- **두 값 모두 Keep Project.** `ECU1` 과 `1000` 은 2장에서 정한 이 머신의 값이다.
- **Network Management 는 끈다.** PARA 는 이 모듈을 켠 채로 가져오고, 그러면 코드 생성이 네트워크 핸들을 요구하는데 이 설계에는 없다. 끈 모듈은 값을 간직하지만 익스포!

확인하기

- [ ] **Machine Configuration [HPC]** 의 **PROCESSES** 탭에 플랫폼 프로세스 열한 개가 있다.
- [ ] **MODULE** 탭의 **Network Management** 가 꺼져 있고 나머지 모듈은 켜져 있다.
- [ ] **MODULE > Log and Trace** 의 **ECU ID** 가 그대로 `ECU1` 이다.
- [ ] 프레임 `HPC` 에 `PARA_HPC`, `Sensor`, `Fusion`, `Actuator` 가 있고 탭에 수정 표시 ● 가 없다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

Executable 과 Process에서 내 Adaptive Application 세 개 각각에 Process 를 만든다.

Executable 과 Process

이 장에서 할 일

이 장에서는 Adaptive Application 세 개 `Sensor`, `Fusion`, `Actuator` 각각에 Process 를 추가한다. 각 Process 에 시작 타임아웃, `MachineFG` 의 상태, `CM` 프로세스에 대한

시작하기 전에

- PARA 설정을 마친 뒤 시작한다. 프로세스 `CM` 은 거기서 들어왔다.
- 머신 `HPC` 에는 머신 설정의 Function Group `MachineFG` 가 있다.

Adaptive Application	Process NAME	ENTER TIMEOUT / EXIT TIMEOUT	FUNCTION GROUP STATE	EXECUTION DEPENDENCY
`Sensor`	`Sensor`	`5` / `5`	`MachineFG`, `Startup`	`CM`, `Running`
`Fusion`	`Fusion`	`5` / `5`	`MachineFG`, `Startup`	`CM`, `Running`
`Actuator`	`Actuator`	`5` / `5`	`MachineFG`, `Startup`	`CM`, `Running`

세 개 모두 **EXECUTABLE** 섹션, 타임아웃 단위 `s`, **SCHEDULING POLICY** `SCHED_OTHER`, **SCHEDULING PRIORITY** `0` 은 그대로 둔다.

Sensor 의 Process 만들기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 누르고 노드 `HPC [HPC]` 를 더블클릭한다. 빵부스러기가 이미 `System > HPC` 이면 건너뛴다.
2. 노드 `Sensor` 를 더블클릭한다.

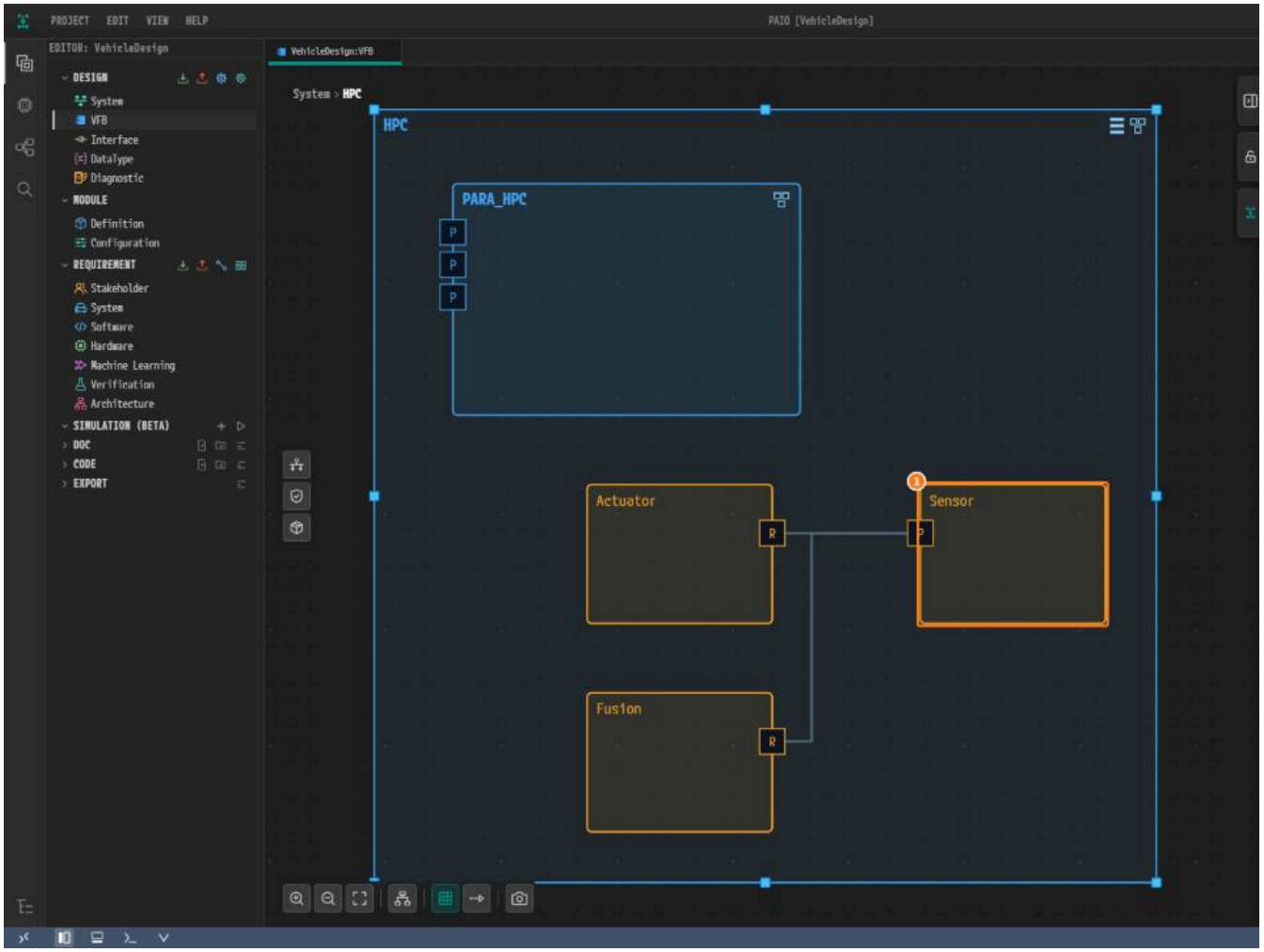


그림 10-1. ① 노드 `Sensor`.

3. 다이얼로그 **AA Internal Behavior [Sensor]** 가 열린다. 탭 **EXECUTION** 을 누른다.

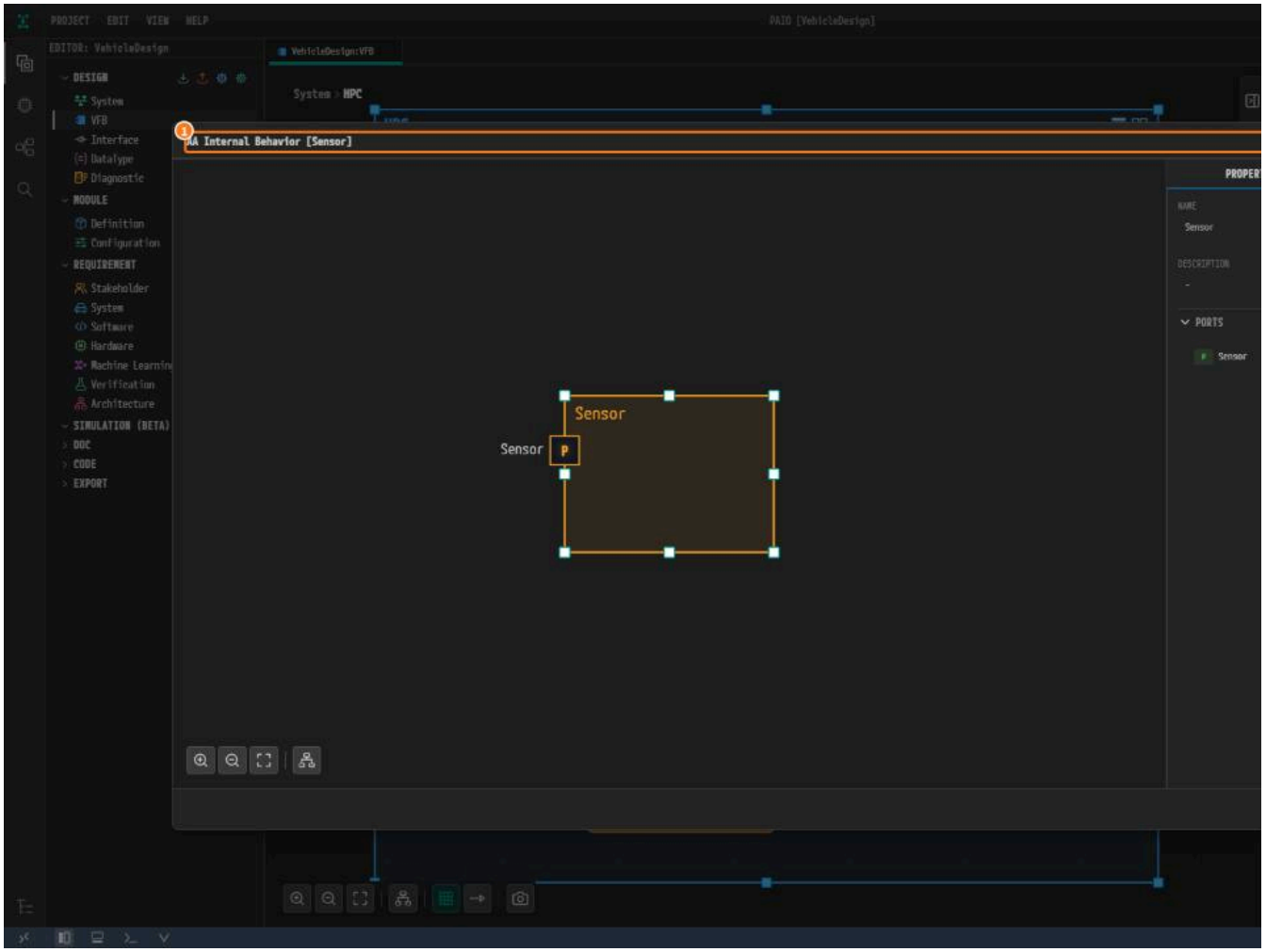


*그림 10-2. ① 다이얼로그 제목, ② **EXECUTION**.*

4. **EXECUTABLE** 섹션은 이미 채워져 있다. 그대로 두고 **PROCESS** 의 **+** 버튼을 누른다.

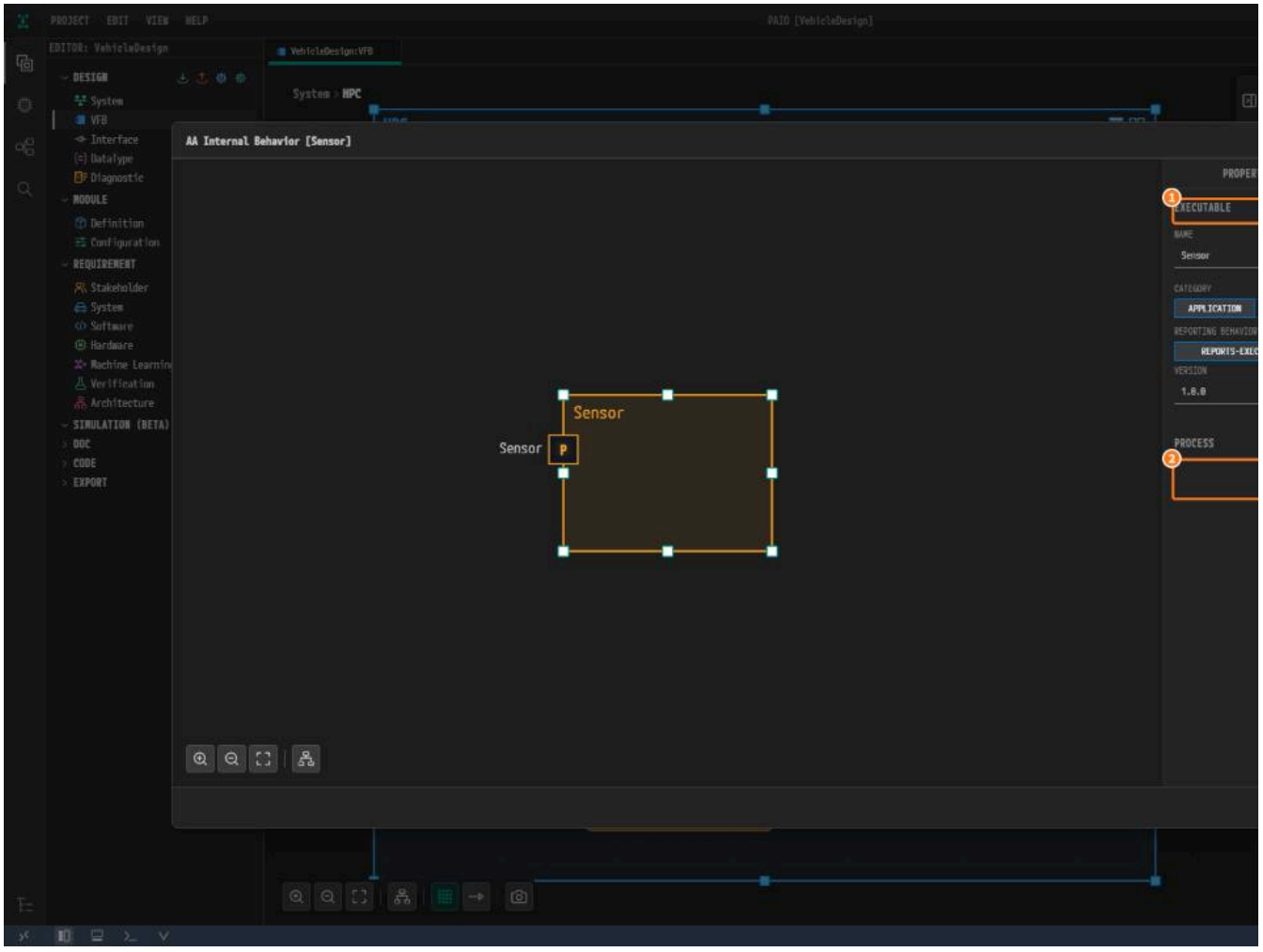


*그림 10-3. ① **EXECUTABLE**, ② `No processes defined`, ③ **+** 버튼.*

5. 새 카드 `Sensor_0` 을 눌러 펼친다. **NAME** 에 `Sensor`, **ENTER TIMEOUT** 에 `5`, **EXIT TIMEOUT** 에 `5` 를 입력하고 입력할 때마다 `Tab` 을 누른다.

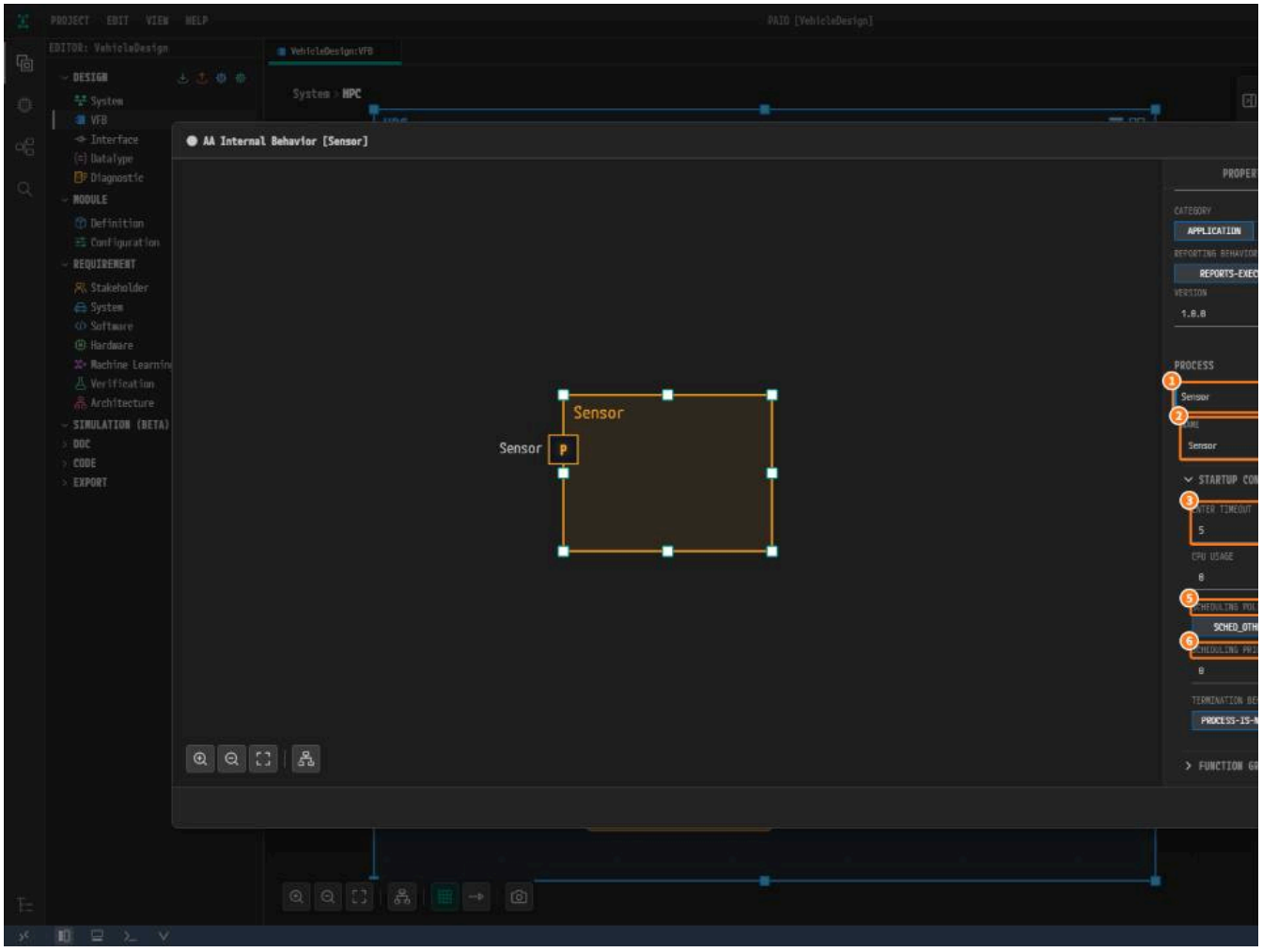


*그림 10-4. ① 카드 머리글, ② **NAME**, ③ **ENTER TIMEOUT** = `5`, ④ **EXIT TIMEOUT** = `5`, ⑤ **SCHEDULING POLICY**, ⑥ **SCHEDULING PRIORITY**.*

6. **FUNCTION GROUP STATE** 의 **+** 버튼을 누른다. 목록이 비어 있는 동안 이 버튼은 빨갛다. 행 `MachineFG`, `Off` 가 생긴다. **FUNCTION GROUP STATE** 열의 `Off` 칸을 누르고

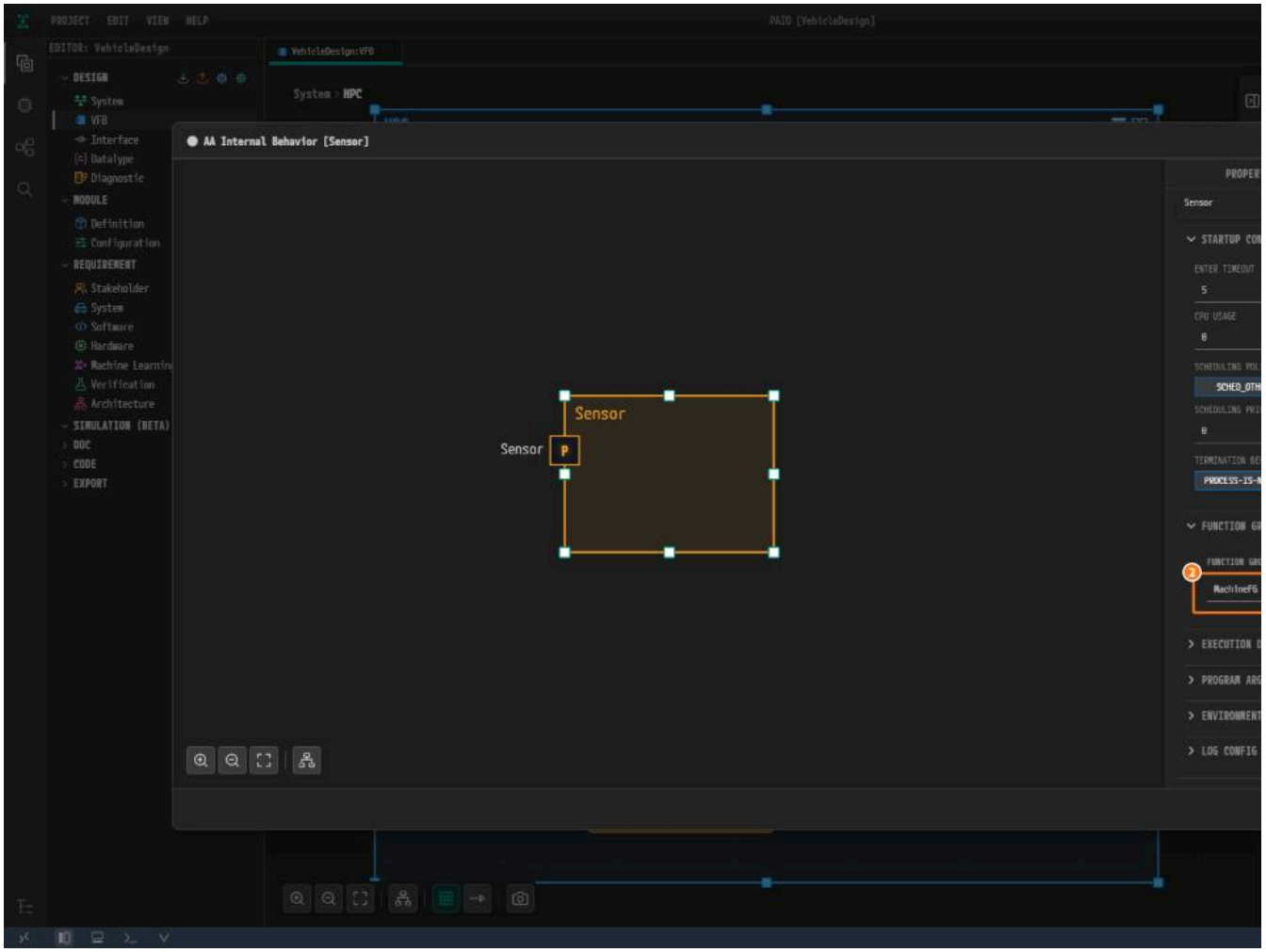


그림 10-5. ① + 버튼, ② 새 행, ③ `Startup`.

7. **EXECUTION DEPENDENCY** 의 **+** 버튼을 누른다. 행 `EM`, `Running` 이 생긴다. **PROCESS** 열의 칸을 누르고 `CM` 을 누른다. **PROCESS STATE** 는 `Running` 그대로 둔다.

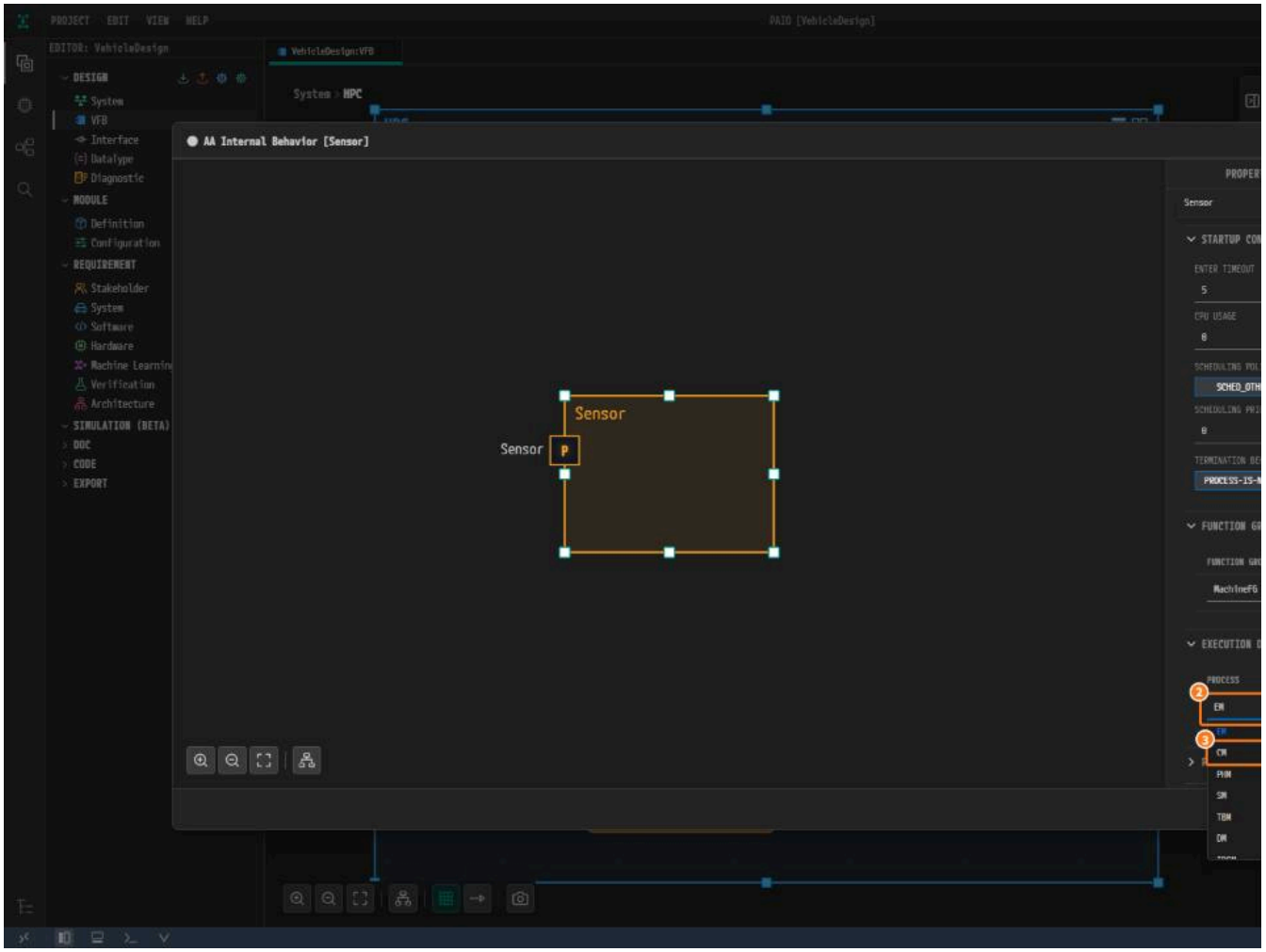


*그림 10-6. ① + 버튼, ② **PROCESS**, ③ `CM`, ④ **PROCESS STATE** = `Running`.*

8. 다이얼로그에서 `Ctrl+S` 를 누르고 **Close** 를 누른다.

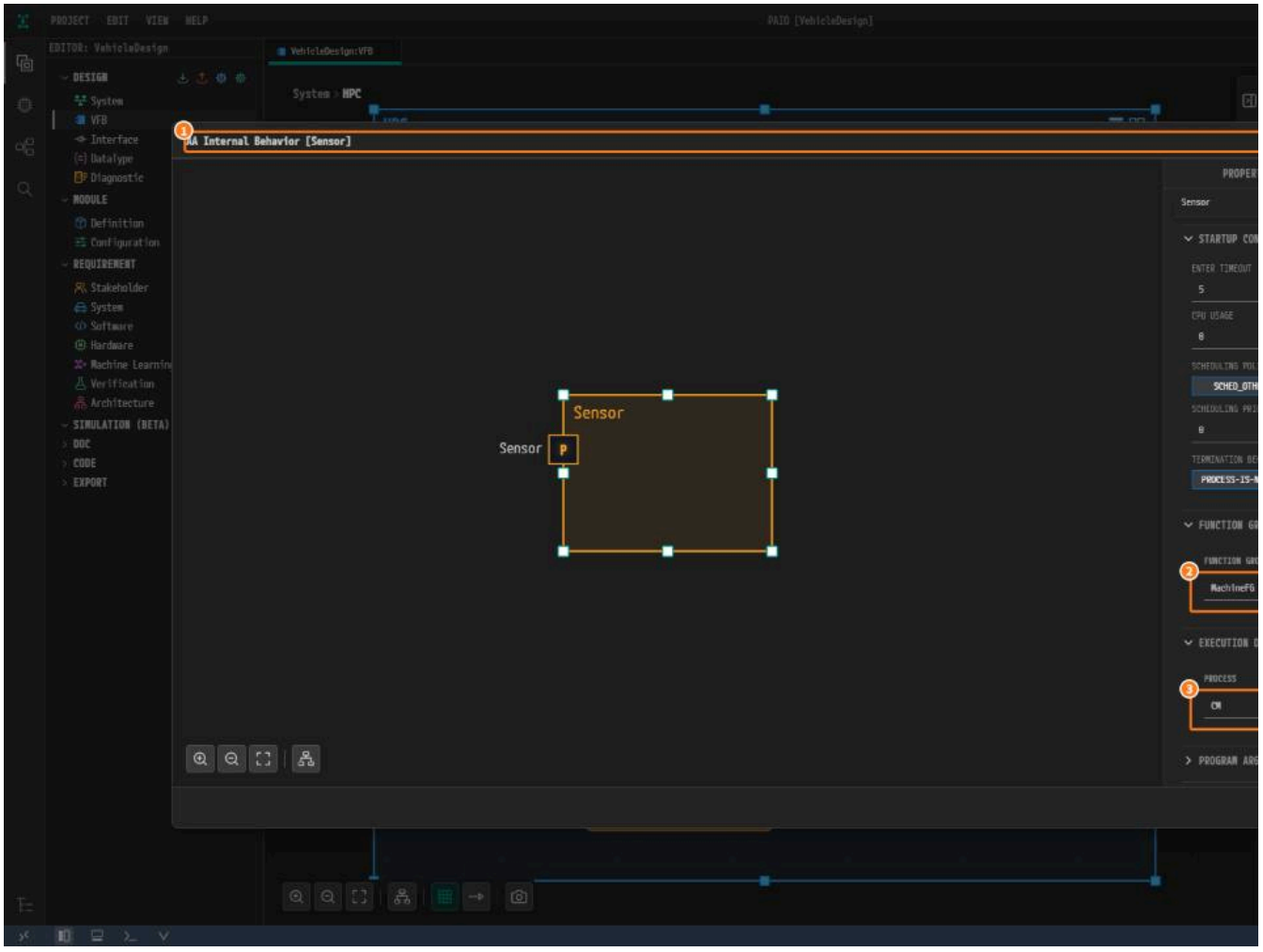


그림 10-7. ① ● 가 없는 제목, ② `MachineFG`, `Startup`, ③ `CM`, `Running`.

Fusion 과 Actuator 의 Process 만들기

같은 방법으로 표대로 나머지 둘을 만들고, 각각 `Ctrl+S` 와 **Close** 로 마친다.

> **주의.** `Fusion` 과 `Actuator` 에서는 새 **EXECUTION DEPENDENCY** 행이 프로세스 `Sensor` 로 시작한다. 저장하기 전에 **PROCESS** 를 `CM` 으로 바꾼다.

머신의 PROCESSES 탭에서 확인하기

1. 프레임 `HPC` 오른쪽 위의 머신 아이콘을 누른다. **Machine Configuration [HPC]** 가 열린다.
2. 탭 **PROCESSES** 를 누른다. 목록에 `Sensor`, `Fusion`, `Actuator` 가 있고 그 뒤에 플랫폼 프로세스 열한 개가 있다.
3. `Sensor` 를 누른다. 오른쪽 **EXECUTION DEPENDENCY** 그래프에 `CM` 에서 `Sensor`, `Fusion`, `Actuator` 로 가는 `Running` 화살표가 보인다.

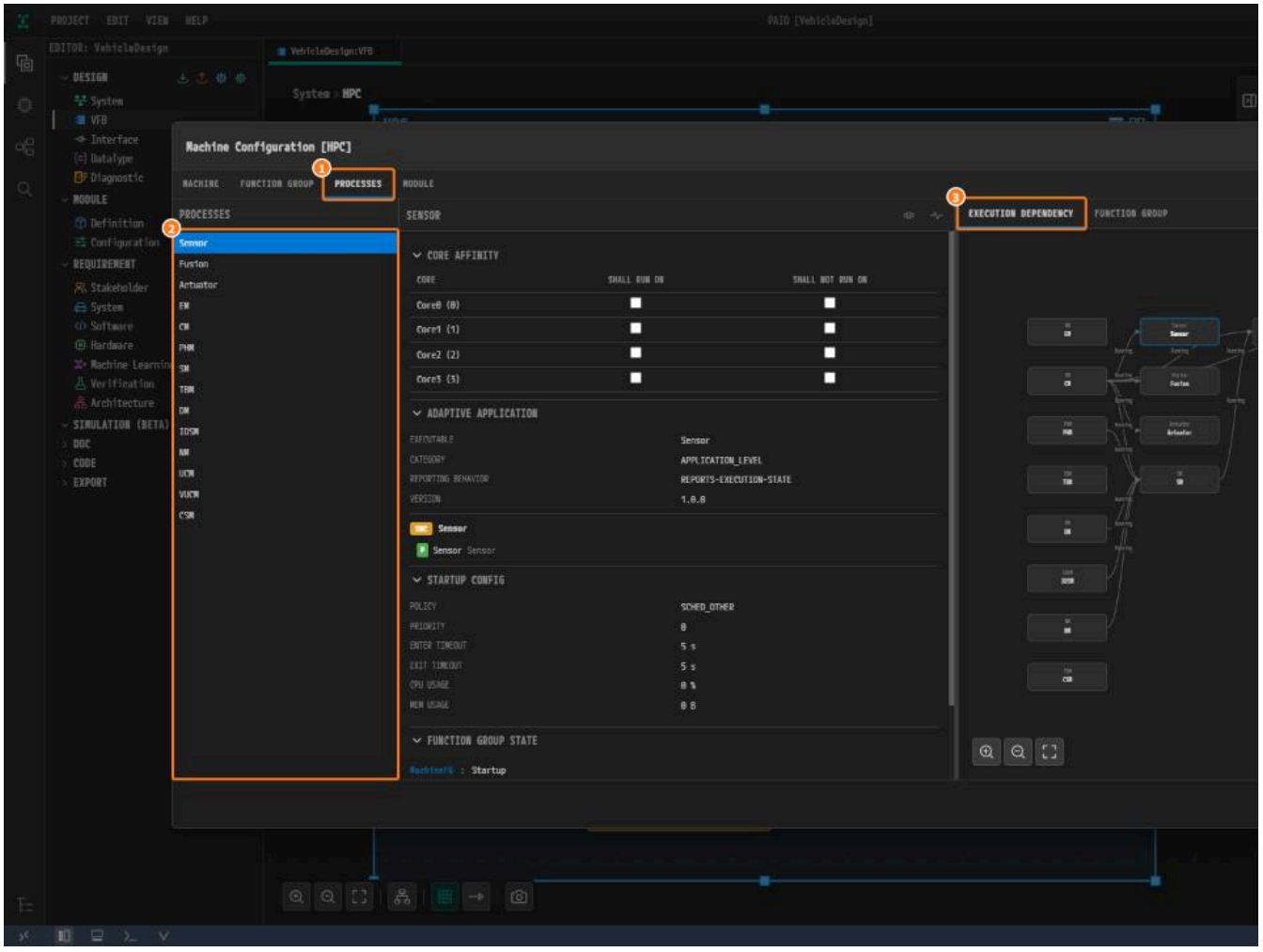


*그림 10-8. ① **PROCESSES**, ② 프로세스 열네 개, ③ **EXECUTION DEPENDENCY** 그래프.*

4. **Close** 를 누른다.

검증 결과 보기

1. **VALIDATION** 패널을 본다. 닫혀 있으면 푸터의 **V** 아이콘을 누른다. `No validation issues.` 가 보인다.

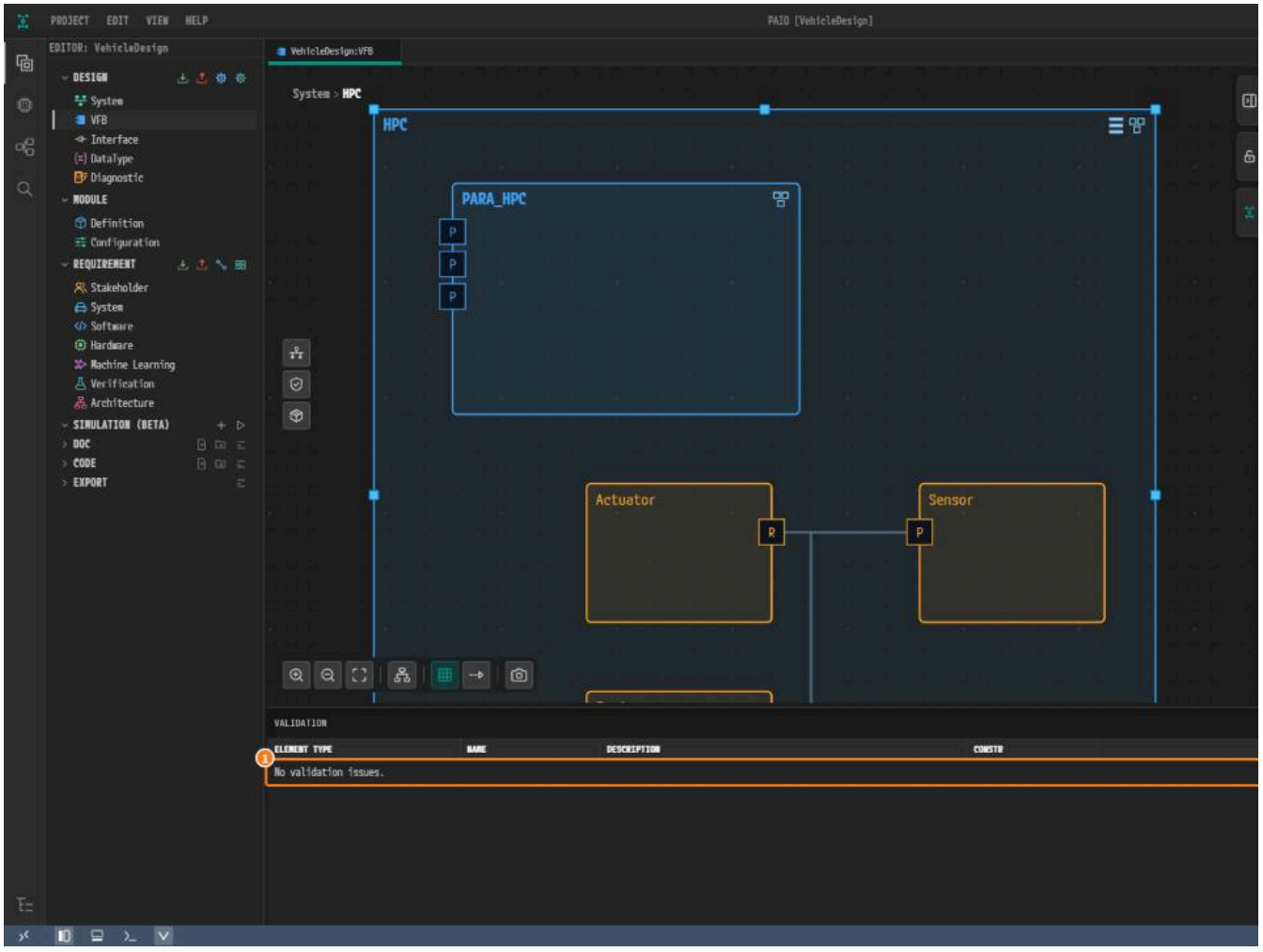


*그림 10-9. 세 다이얼로그를 모두 저장한 뒤의 **VALIDATION** 패널.*

왜 이렇게 설계하는가

- **Executable 은 기본값 그대로.** 기본값이 이미 이 설계가 익스포트하는 값이다.
- `CM` **이 먼저.** `Sensor`, `Fusion`, `Actuator` 는 `CM` 이 `Running` 일 때만 시작한다.
- **모든 프로세스에 Function Group State 가 필요하다.** 검증이 Function Group State 없는 프로세스를 거부한다.
- **플랫폼 프로세스는 PARA 가 준 그대로 둔다.** 타임아웃, 상태, 의존성이 가져오기와 함께 들어온다.

확인하기

- [ ] Adaptive Application 세 개 각각에 자기 이름의 프로세스 카드가 하나 있고 타임아웃은 `5` / `5` 이다.
- [ ] 각각 **FUNCTION GROUP STATE** 행 `MachineFG`, `Startup` 이 하나 있다. `Off` 행은 없다.
- [ ] 각각 의존성 `CM`, `Running` 이 있다.
- [ ] **Machine Configuration [HPC]** 의 **PROCESSES** 탭에 프로세스 열네 개가 있다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

서비스 인스턴스 매핑에서 `Sensor`, `Fusion`, `Actuator` 의 포트를 SOME/IP 서비스 인스턴스에 매핑한다.

서비스 인스턴스 매핑

이 장에서 할 일

이 장에서는 `Sensor` 의 포트를 제공자 인스턴스 `Provider_Sensor` 에, `Fusion` 과 `Actuator` 의 포트를 소비자 인스턴스 `Fusion_Sensor` 와 `Actuator_Sensor` 에 매핑

시작하기 전에

- Executable 과 Process를 마친 뒤 시작한다. SOME/IP 배포와 서비스 인스턴스의 인스턴스는 커넥터에 서비스 인스턴스 바인딩대로 커넥터에 바인딩되어 있다.
- 매핑하는 것은 내 애플리케이션 세 개의 포트뿐이다. `PARA_HPC` 안의 애플리케이션은 그대로 둔다.

	`Sensor`	`Fusion`	`Actuator`
PORT	`Sensor` (`P`)	`Sensor` (`R`)	`Sensor` (`R`)
SERVICE INSTANCE	`Provider_Sensor (SOME/IP)`	`Fusion_Sensor (SOME/IP)`	`Actuator_Sensor (SOME/IP)`
CLIENT ID	칸 없음	`2` → `0x2`	`1` → `0x1`
매핑 박스	`Sensor/AA/Sensor`	`Fusion/AA/Sensor`	`Actuator/AA/Sensor`
되읽기 **ENDPOINT**	`192.168.0.11:20003`	`192.168.0.11:20002`	`192.168.0.11:20001`

Sensor 의 포트를 Provider_Sensor 에 매핑하기

AA Internal Behavior 다이얼로그 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 클릭하고 노드 `HPC [HPC]` 를 더블클릭한다.

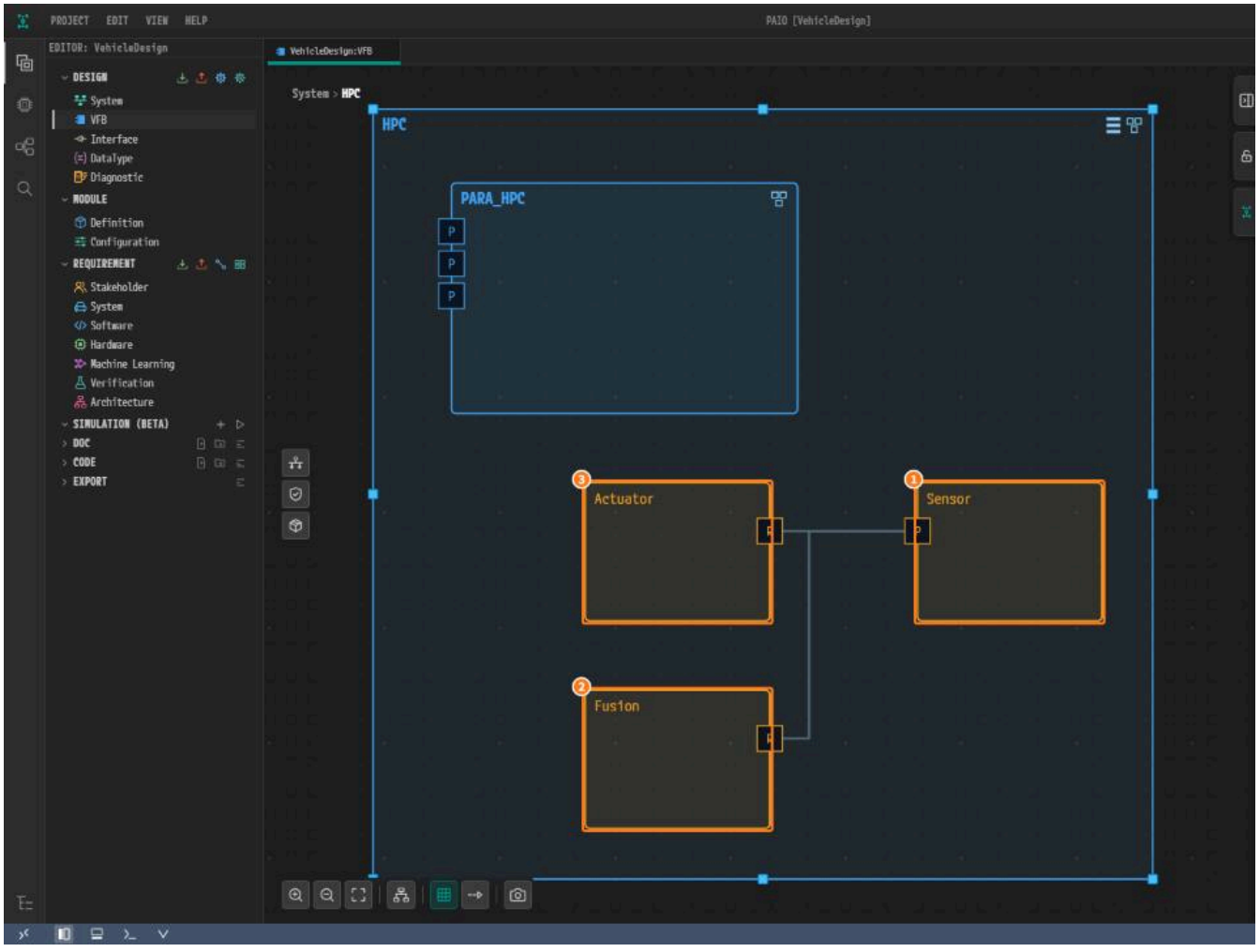


그림 11-1. ① `Sensor`, ② `Fusion`, ③ `Actuator`. 노드 배치는 독자의 화면과 다를 수 있다.

2. 노드 `Sensor` 를 더블클릭한다. 다이얼로그 **AA Internal Behavior [Sensor]** 가 열린다.

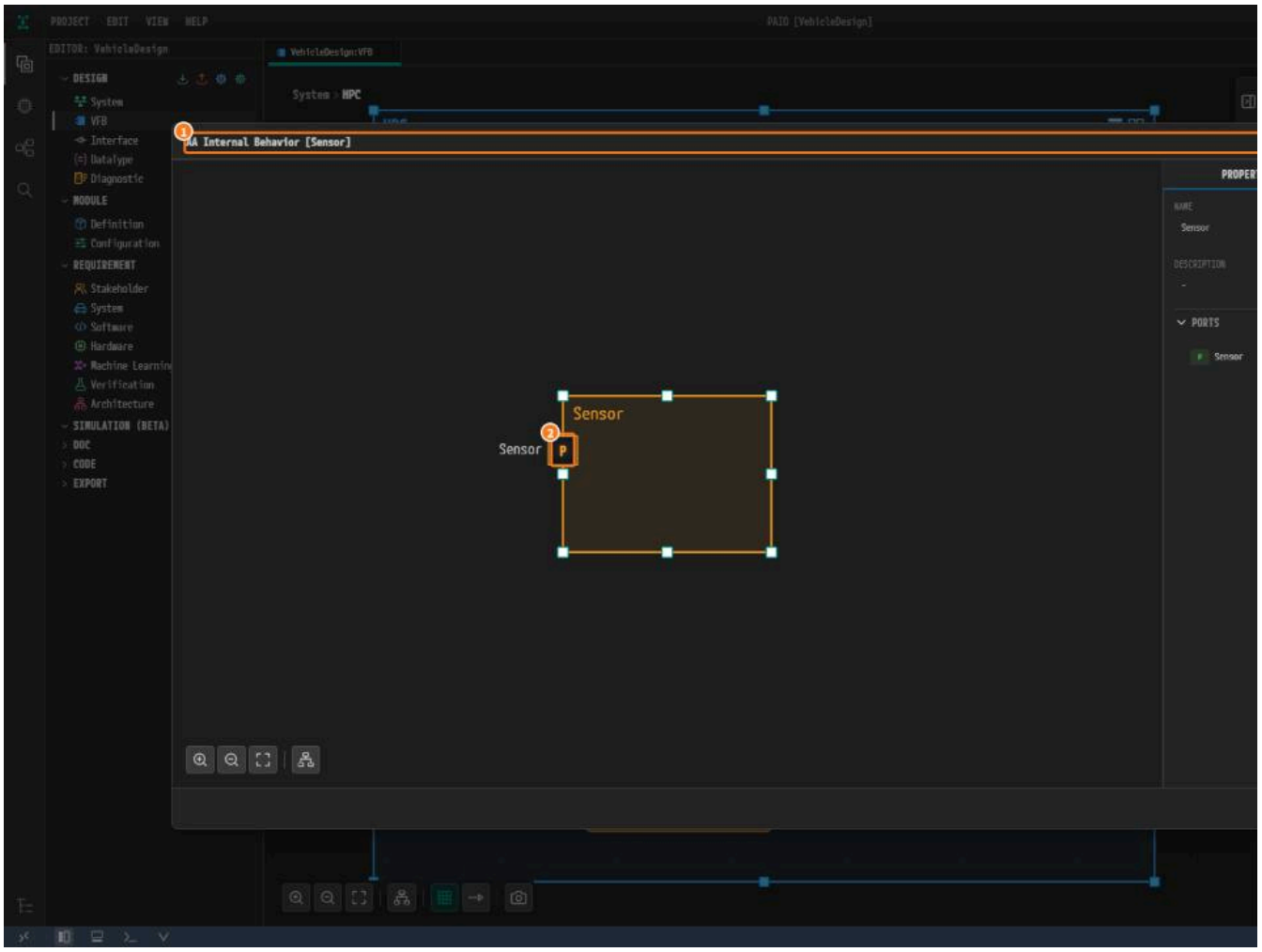


그림 11-2. ① 다이얼로그 제목, ② `P` 포트 `Sensor`.

Add Instance 메뉴로 매핑 박스 만들기

1. `Sensor` 본체의 빈 곳을 오른쪽 클릭한다.
2. **Add Instance** 위에 포인터를 올린다.
3. **ara::com** 위에 포인터를 올린다.

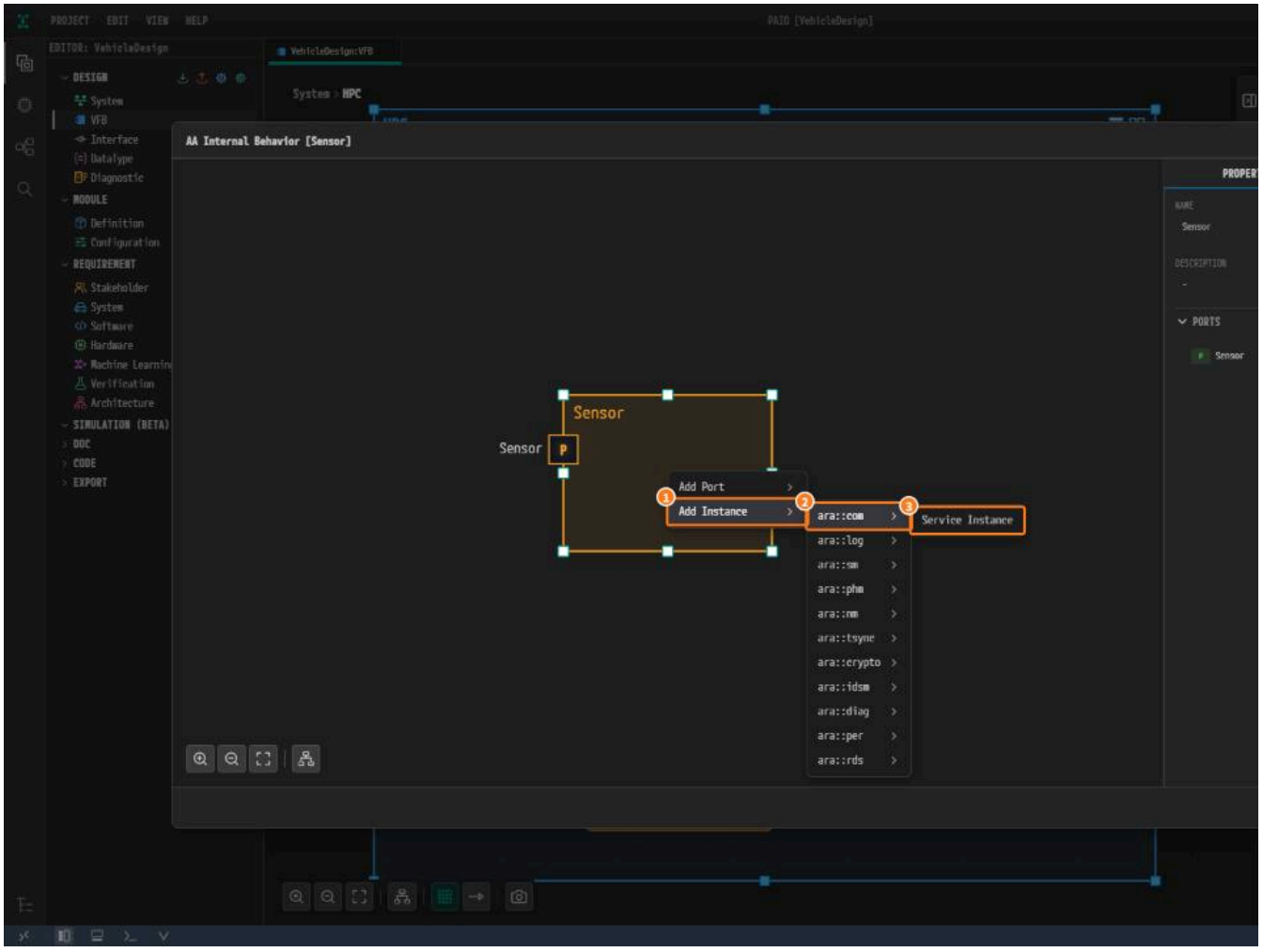


*그림 11-3. ① **Add Instance**, ② **ara::com**, ③ **Service Instance**.*

4. **Service Instance** 를 누른다. 매핑 박스 `Sensor/AA/` 가 생기고 패널이 **SERVICE INSTANCE MAPPING** 으로 바뀐다. **PORT** 와 **SERVICE INSTANCE** 는 채울 때까지 빨갛다.

PORT 고르기

1. **PORT** 칸을 누른다.

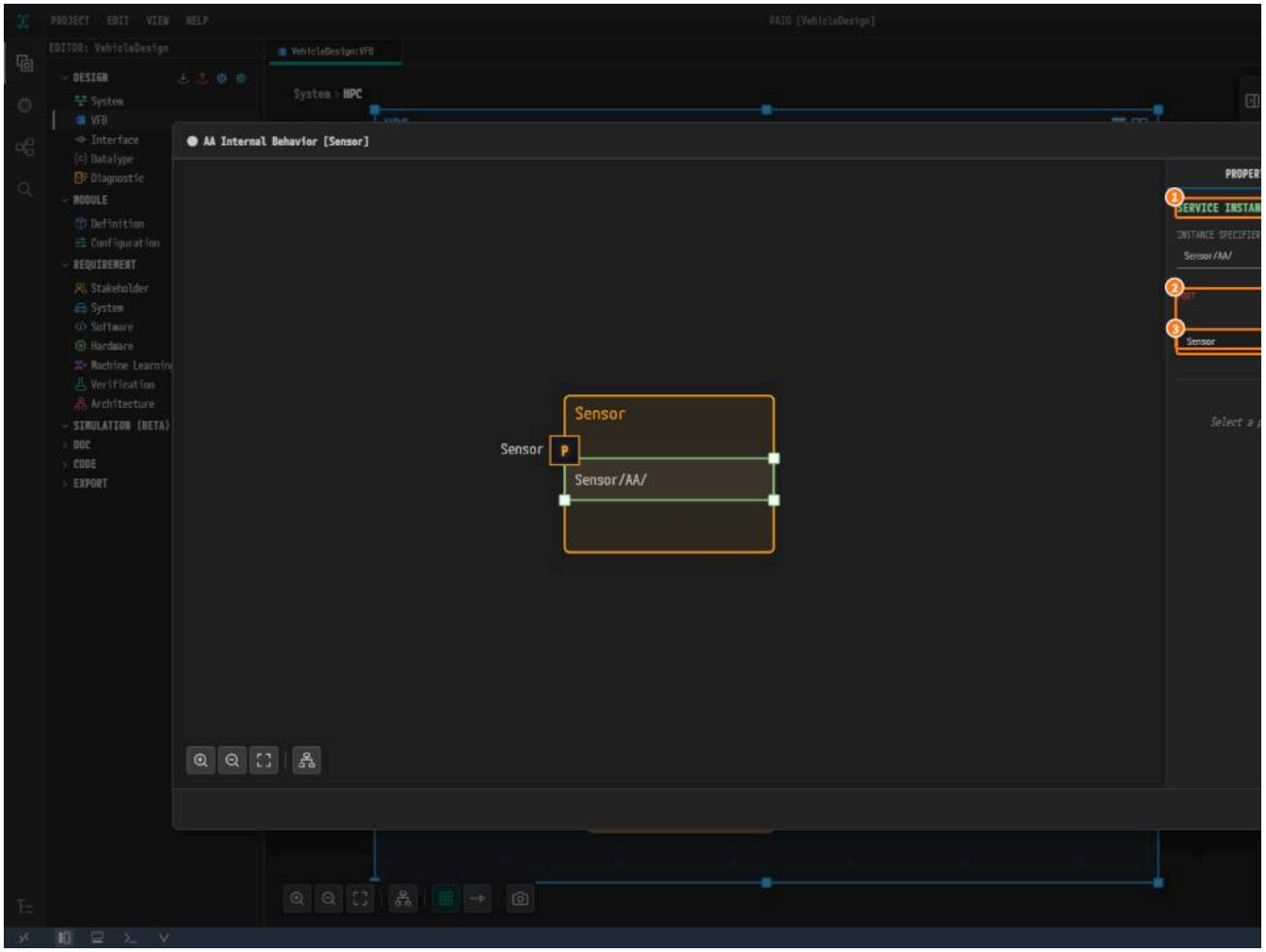


*그림 11-4. ① **SERVICE INSTANCE MAPPING**, ② **PORT**, ③ `Sensor`.*

2. `Sensor` 를 누른다. 박스가 `Sensor/AA/Sensor` 가 된다.

> **주의.** **PORT** 를 먼저 고른다. 그러면 **SERVICE INSTANCE** 목록에 맞는 인스턴스만 나온다. `P` 포트에는 제공자, `R` 포트에는 소비자가 나온다.

SERVICE INSTANCE 고르기

1. **SERVICE INSTANCE** 칸을 누른다.

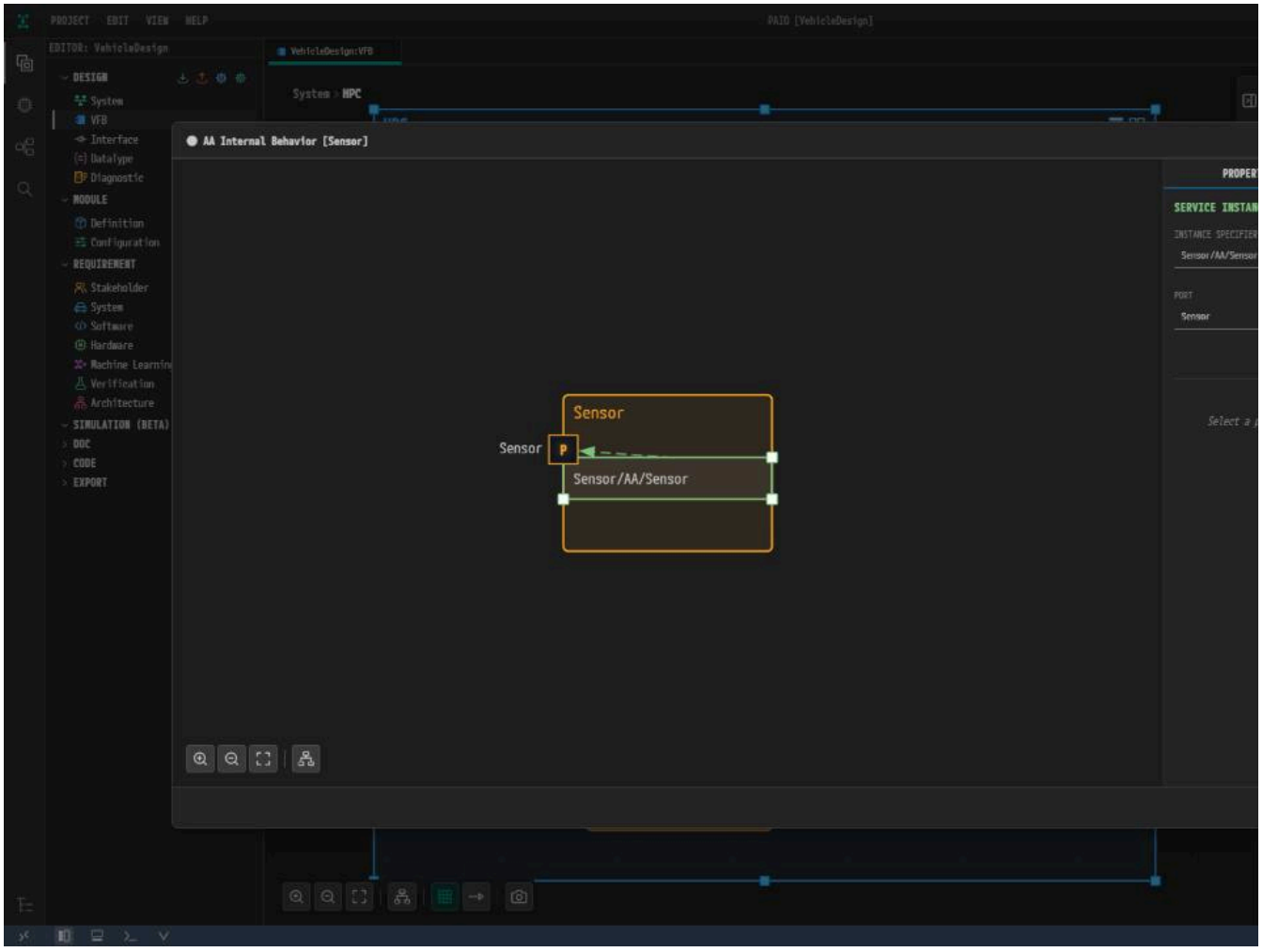


*그림 11-5. ① **SERVICE INSTANCE**, ② `Provider_Sensor (SOME/IP)`.*

2. `Provider_Sensor (SOME/IP)` 를 누른다. 읽기 전용 블록 **REFERENCED INSTANCE** 가 나타난다. 제공자에는 **CLIENT ID** 칸이 없다.

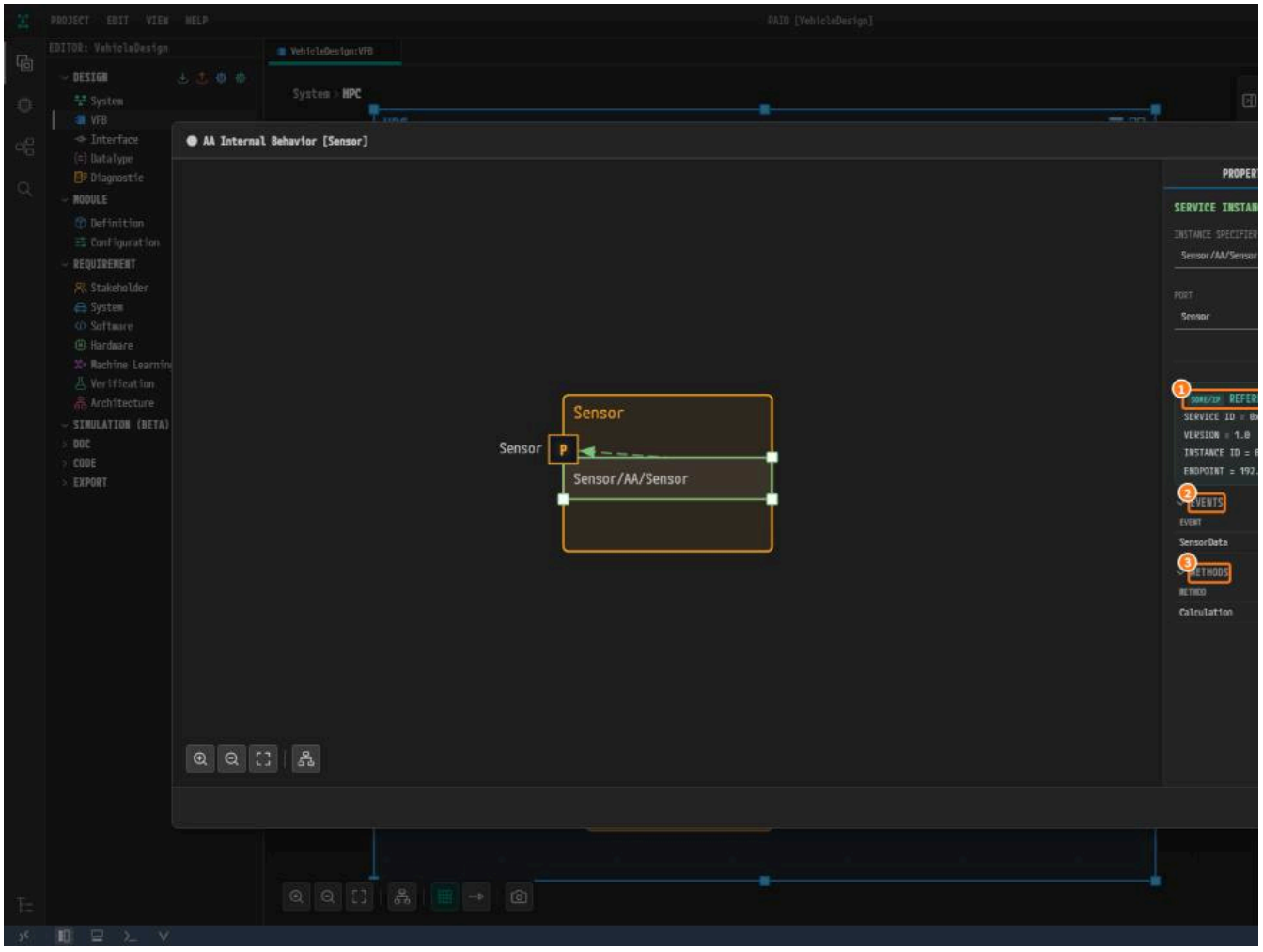


*그림 11-6. ① **REFERENCED INSTANCE**(`ENDPOINT = 192.168.0.11:20003`), ② **EVENTS**, ③ **METHODS**.*

저장하고 닫기

1. 다이얼로그 안에서 `Ctrl+S` 를 누른다.
2. **Close** 를 누른다.

Fusion 의 포트를 Fusion_Sensor 에 매핑하기

`Fusion` 은 소비자이다. 절차는 `Sensor` 와 같고 **CLIENT ID** 가 더해진다.

1. 노드 `Fusion` 을 더블클릭한다.
2. 본체를 오른쪽 클릭하고 **Add Instance** > **ara::com** > **Service Instance** 를 누른다.
3. **PORT** 에서 `Sensor` 를 고른다.
4. **SERVICE INSTANCE** 에서 `Fusion_Sensor (SOME/IP)` 를 고른다.
5. **ENDPOINT** 가 `192.168.0.11:20002` 인지 확인한다.
6. **CLIENT ID** 에 `2` 를 입력하고 `Tab` 을 누른다. 칸에 `0x2` 가 보인다.

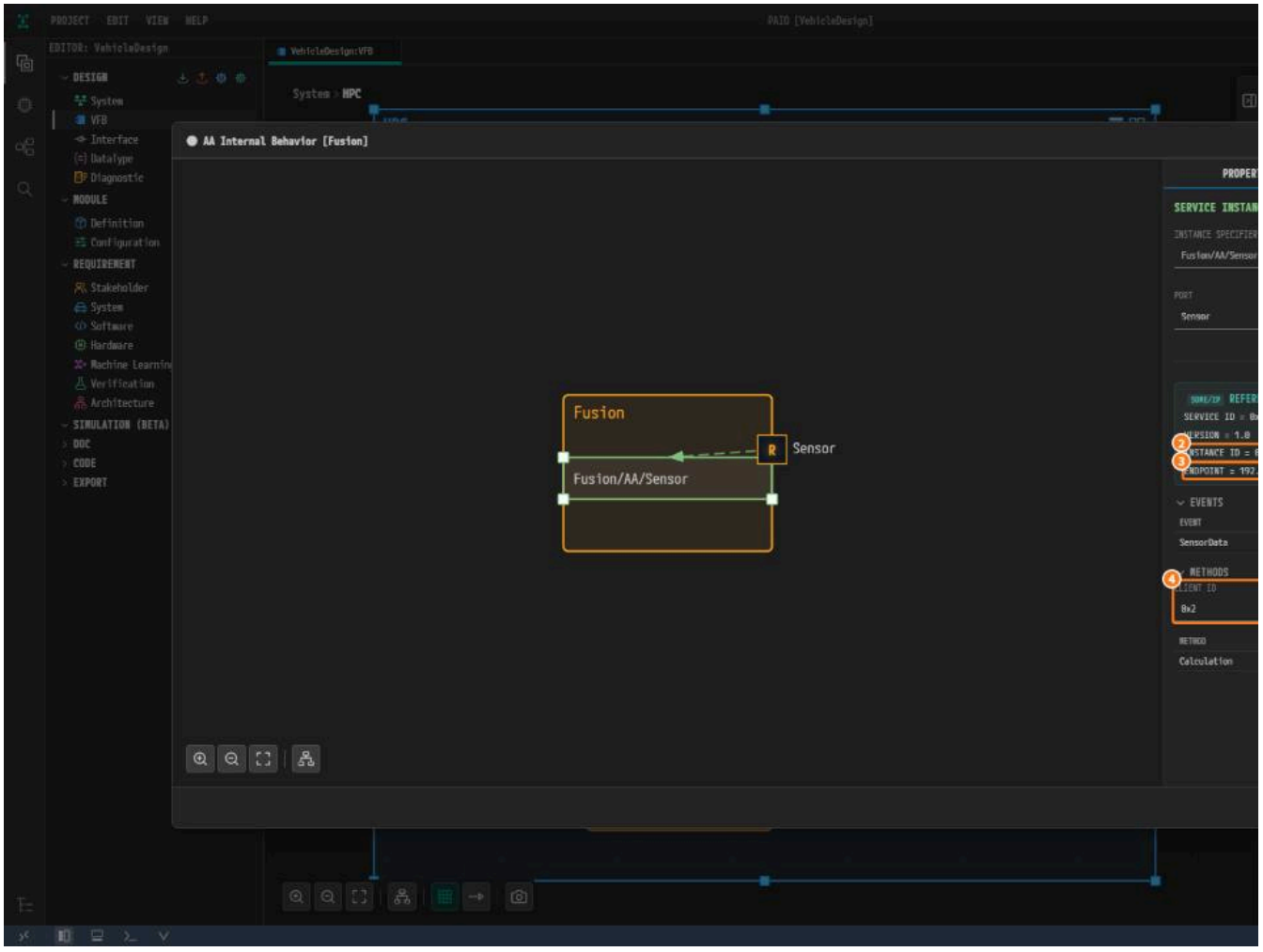


*그림 11-7. ① **SERVICE INSTANCE**, ② `INSTANCE ID = 0xFFFF`, ③ `ENDPOINT = 192.168.0.11:20002`, ④ **CLIENT ID** `0x2`.*

7. `Ctrl+S` 를 누른 뒤 **Close** 를 누른다.

Actuator 의 포트를 Actuator_Sensor 에 매핑하기

같은 방법으로 `Actuator` 를 매핑한다. **SERVICE INSTANCE** 는 `Actuator_Sensor (SOME/IP)`, **CLIENT ID** 는 `1` 이다.

> **주의.** `Fusion` 과 `Actuator` 에는 같은 두 항목이 나온다. **ENDPOINT** 를 확인한다. `Actuator` 는 `192.168.0.11:20001` 이어야 한다.

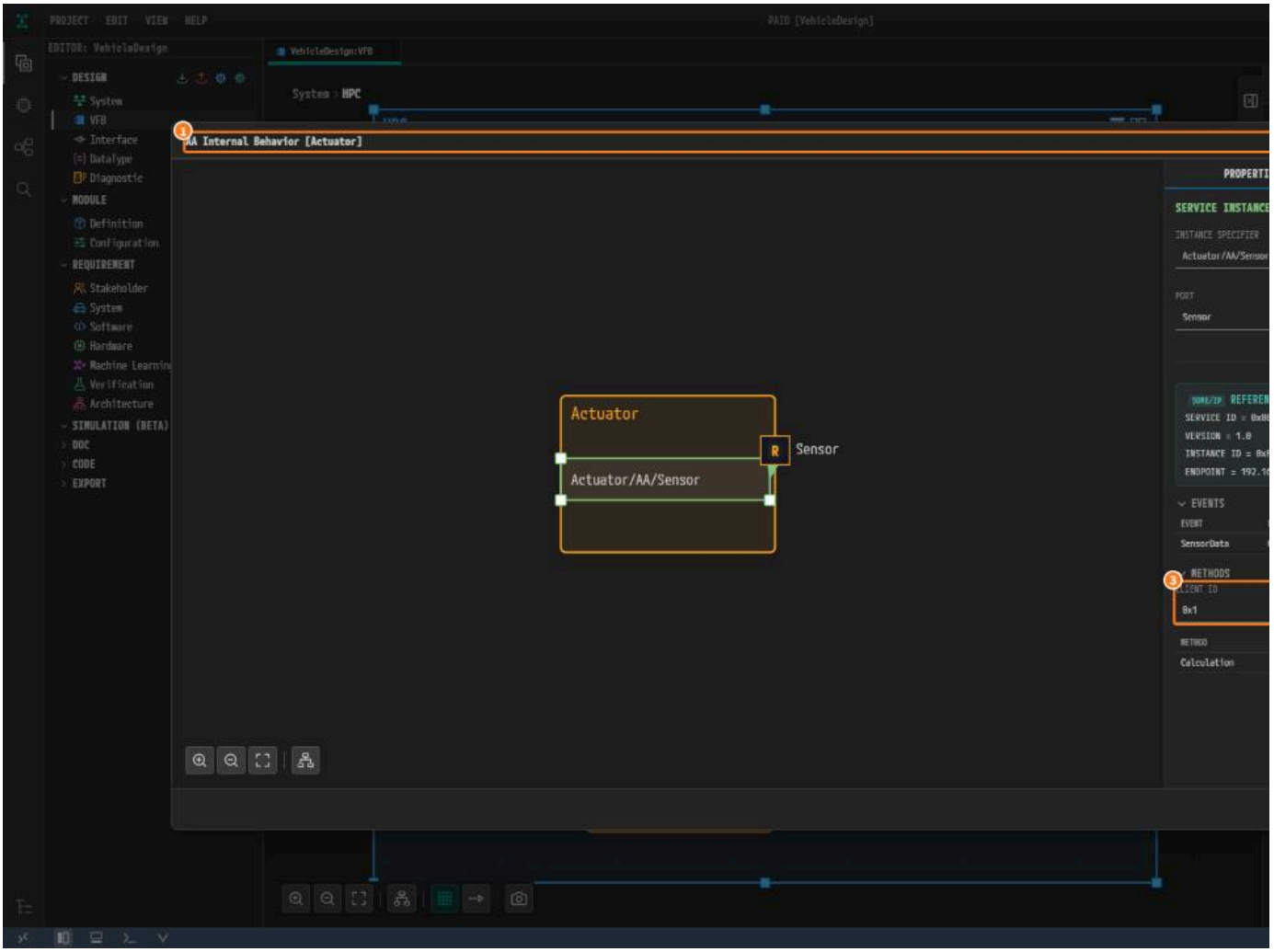


*그림 11-8. ① 수정 표시가 없는 제목, ② **SERVICE INSTANCE**, ③ **CLIENT ID** `0x1`.*

매핑 박스는 다이얼로그 안에서만 보이므로 `HPC` 캔버스는 겉보기에 그대로이다.

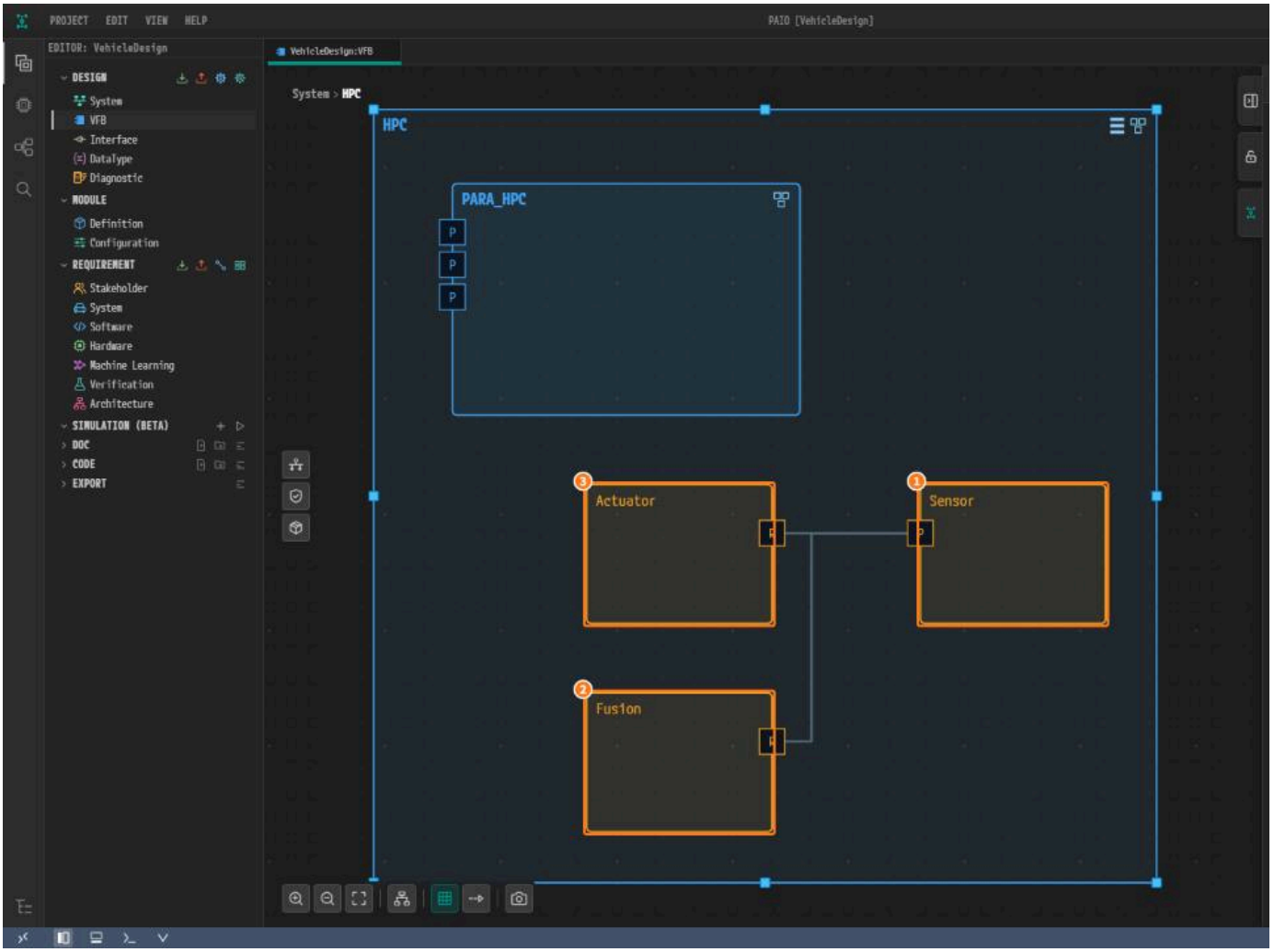


그림 11-9. ① `Sensor`, ② `Fusion`, ③ `Actuator`.

검증 결과 보기

1. **VALIDATION** 패널을 본다. 닫혀 있으면 푸터의 **V** 아이콘을 누른다. `No validation issues.` 가 보인다.

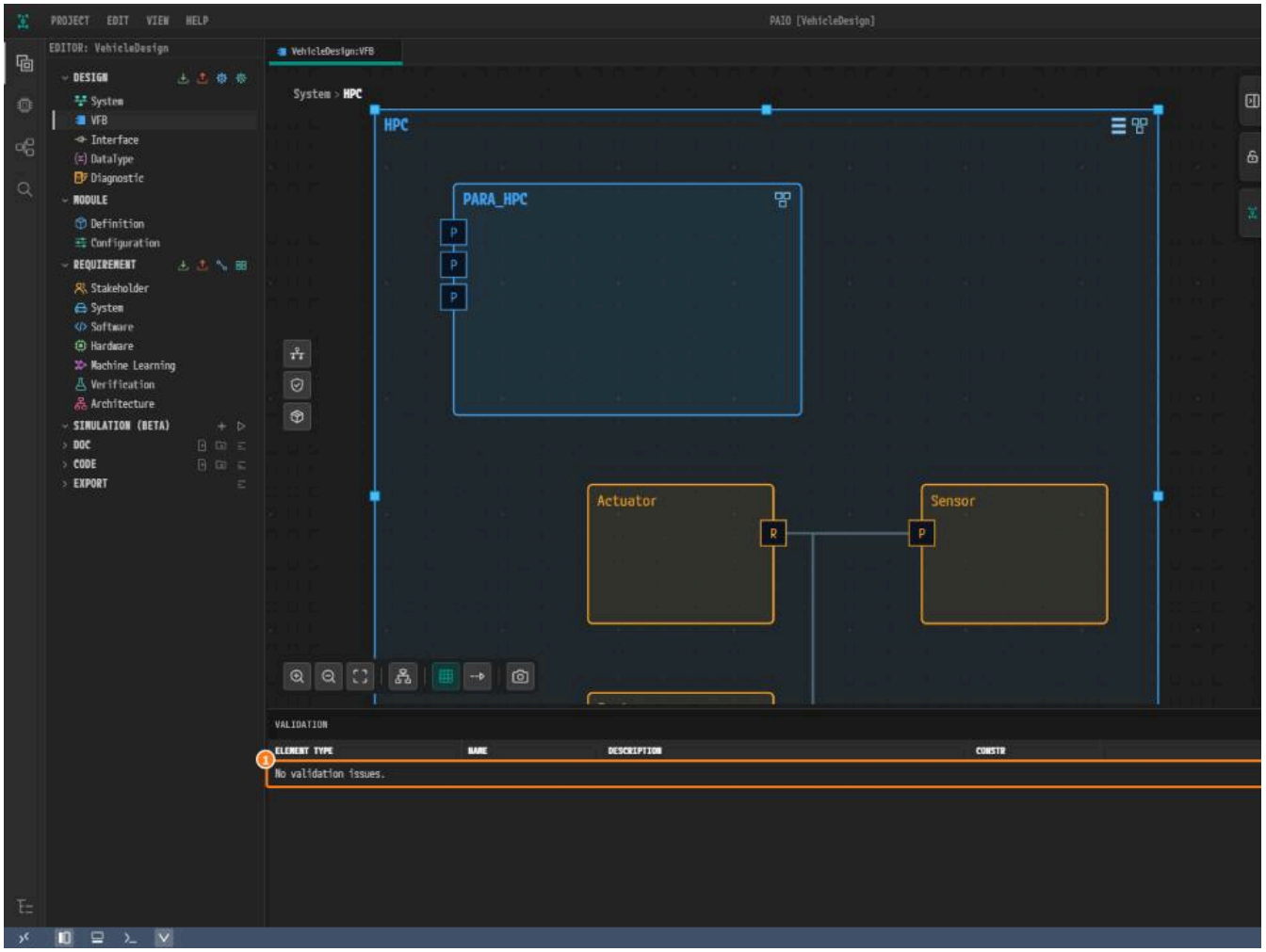


그림 11-10. `No validation issues.`.

왜 이렇게 설계하는가

- **내 포트만 매핑한다.** 이 설계는 PARA 의 플랫폼 서비스를 네트워크에 내놓지 않으므로 그 포트는 매핑하지 않는다.
- **CLIENT ID 는 소비자에만 있다.** 제공자 매핑에는 **CLIENT ID** 칸이 없다.
- **되읽기 블록은 앞에서 정한 값을 보여 줄 뿐이다.** Service ID, Instance ID, 엔드포인트는 그 값을 정한 장에서 바꾸고, 여기서는 바꾸지 않는다.

확인하기

- [ ] 세 다이얼로그에 박스 `Sensor/AA/Sensor`, `Fusion/AA/Sensor`, `Actuator/AA/Sensor` 가 있다.
- [ ] **ENDPOINT** 가 `Sensor` 는 `…:20003`, `Fusion` 은 `…:20002`, `Actuator` 는 `…:20001` 이다.
- [ ] **CLIENT ID** 가 `Fusion` 은 `0x2`, `Actuator` 는 `0x1` 이다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

로그와 DLT에서 세 프로세스의 로그 설정을 입력한다.

로그와 DLT

이 장에서 할 일

이 장에서는 `Sensor`, `Fusion`, `Actuator` 프로세스의 **LOG CONFIG** 를 채운다. 각각 DLT(Diagnostic Log and Trace) 애플리케이션 ID 와 기본 로그 레벨을 정하고 DLT 컨텍스트에서 결과를 확인한다.

시작하기 전에

- 서비스 인스턴스 매핑을 마친 뒤 시작한다.
- 플랫폼 프로세스의 로그 설정은 PARA 가 가져온 그대로 둔다.
- **LOG MODE** 는 모두 `CONSOLE`, 컨텍스트의 **LOG LEVEL** 은 모두 `INFO` 그대로 둔다.

프로세스	APPLICATION ID	APPLICATION DESCRIPTION	DEFAULT LOG LEVEL	CONTEXT ID	CONTEXT DESCRIPTION
`Sensor`	`SENS`	`Sensor application`	`VERBOSE`	`SENS`	`Sensor context`
`Fusion`	`FUSN`	`Fusion application`	`VERBOSE`	`FUSN`	`Fusion context`
`Actuator`	`ACTA`	`Actuator application`	`VERBOSE`	`ACTA`	`Actuator context`

Sensor 의 LOG CONFIG 와 DLT 컨텍스트 입력하기

LOG CONFIG 펼치기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB** 를 클릭하고 노드 `HPC [HPC]` 를 더블클릭한다.

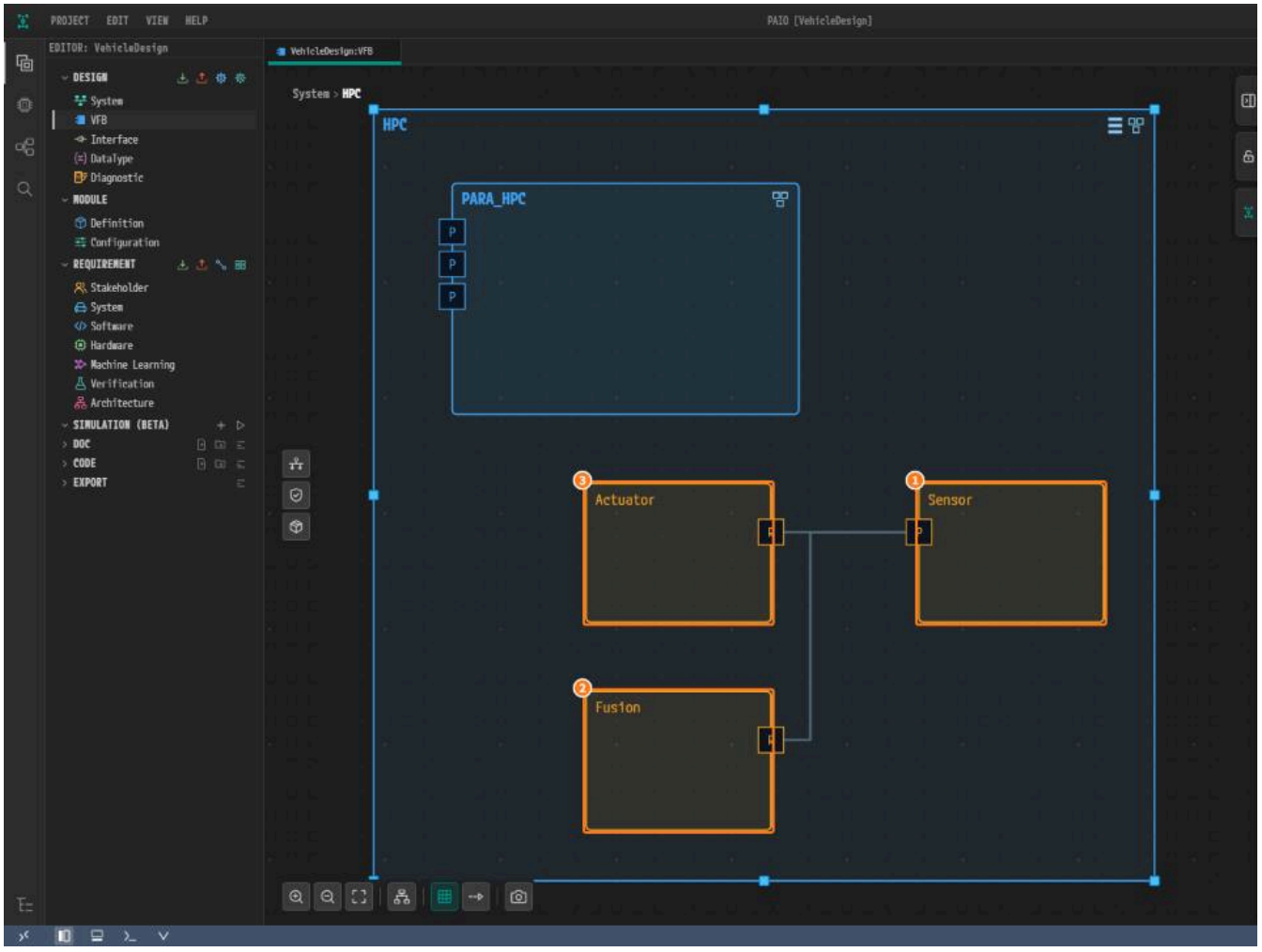


그림 12-1. ① `Sensor`, ② `Fusion`, ③ `Actuator`. 노드 배치는 독자의 화면과 다를 수 있다.

2. 노드 `Sensor` 를 더블클릭한다.

3. **EXECUTION** 탭을 누른다. 프로세스 카드 `Sensor` 가 접혀 있으면 머리글을 누른다.

4. **LOG CONFIG** 머리글을 눌러 펼친다.

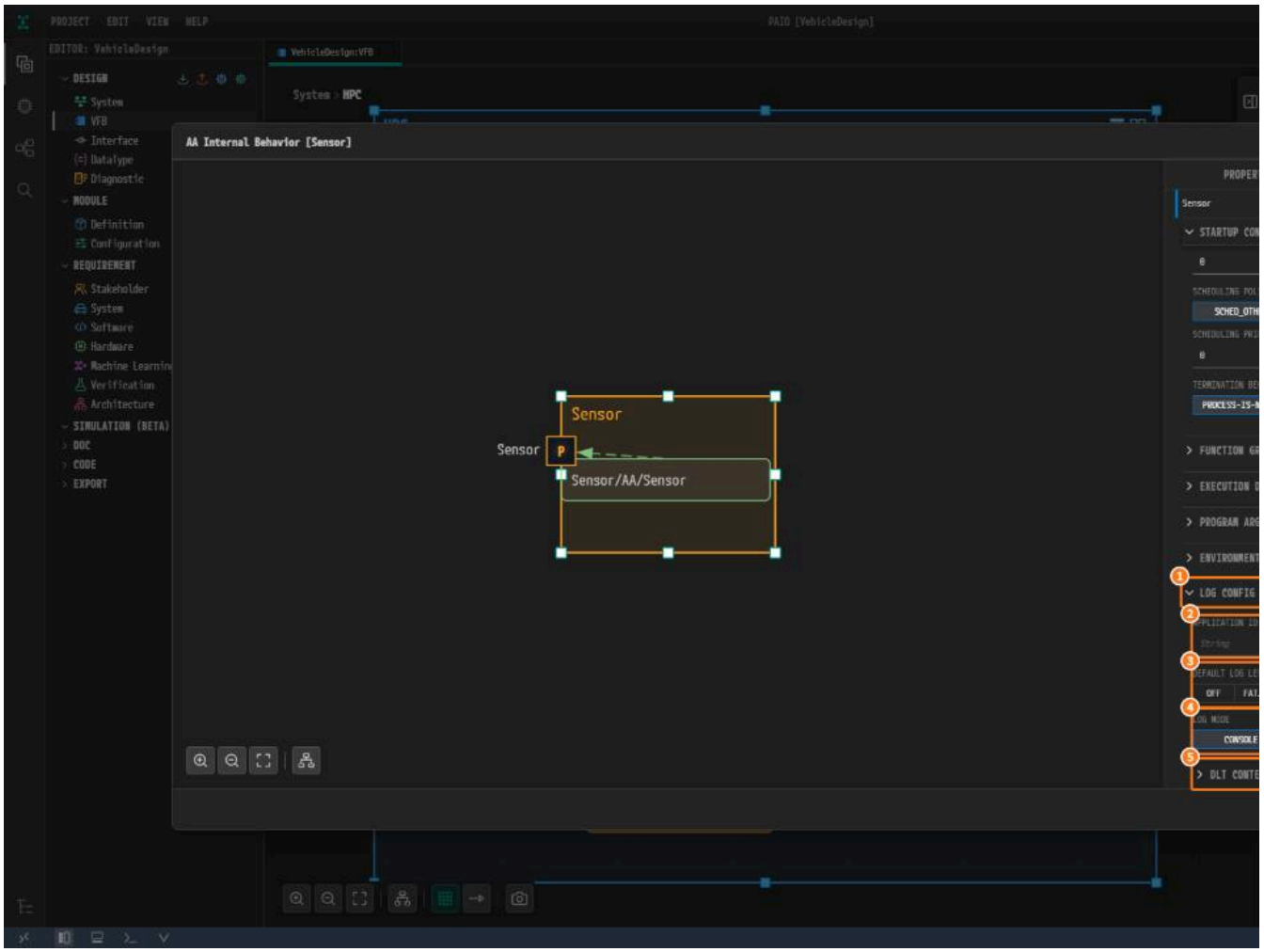


*그림 12-2. ① **LOG CONFIG**, ② **APPLICATION ID**, ③ **DEFAULT LOG LEVEL**, ④ **LOG MODE**, ⑤ **DLT CONTEXTS**.*

> **참고.** 레벨 버튼은 폭이 좁아 `VERBOSE` 는 `VER…` 로 보인다.

APPLICATION ID 와 DEFAULT LOG LEVEL 입력하기

1. **APPLICATION ID** 에 `SENS` 를 입력하고 `Tab` 을 누른다.

2. **APPLICATION DESCRIPTION** 에 `Sensor application` 을 입력하고 `Tab` 을 누른다.

3. **DEFAULT LOG LEVEL** 에서 `VERBOSE` 를 누른다.

4. **LOG MODE** 는 `CONSOLE` 그대로 둔다.

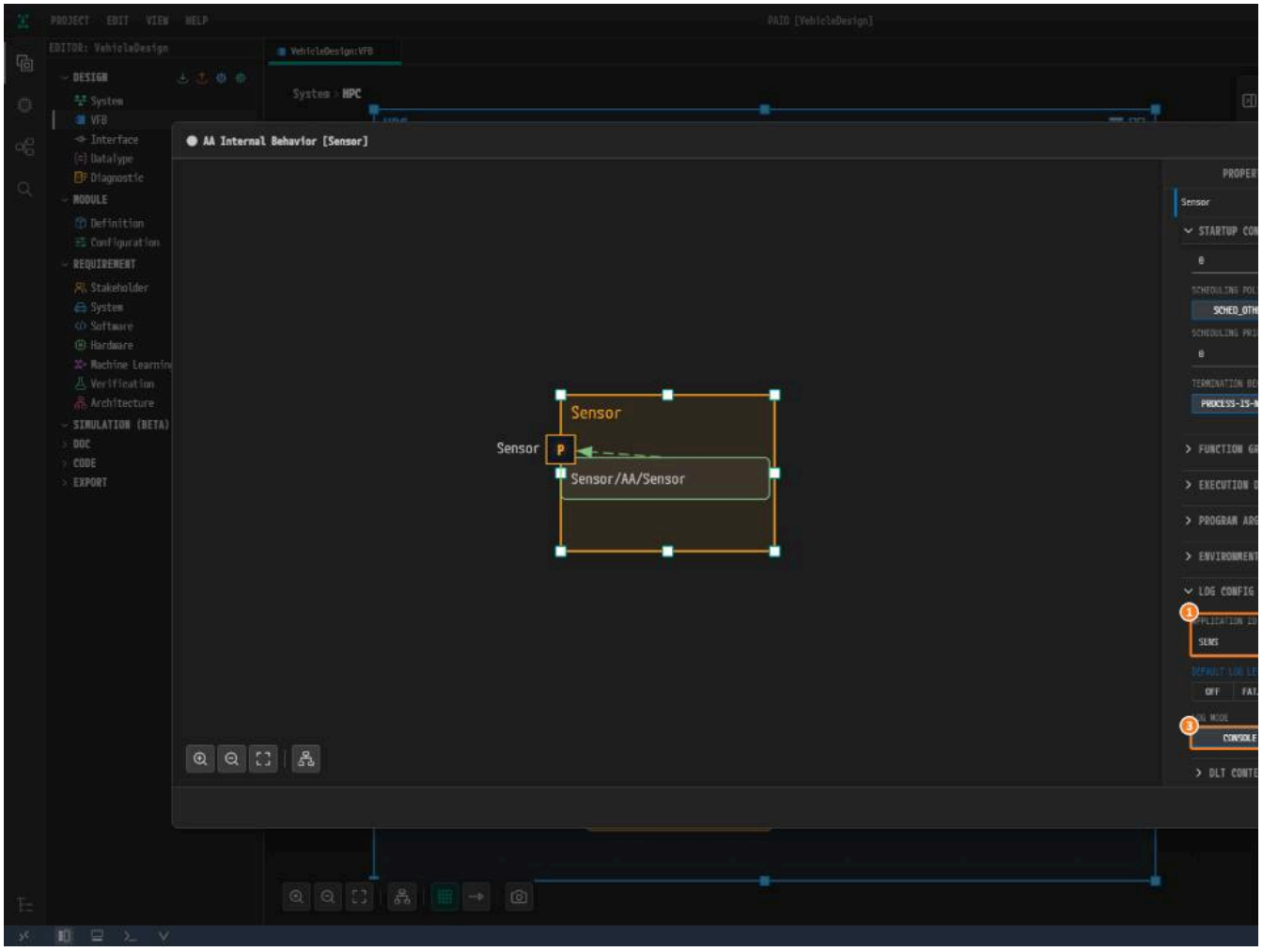


*그림 12-3. ① **APPLICATION ID** `SENS`, ② `VERBOSE`, ③ `CONSOLE`.*

DLT 컨텍스트 추가하기

1. **DLT CONTEXTS** 머리글을 눌러 펼친다.
2. 오른쪽의 **+** 버튼을 누른다. 접힌 행 `Context_0 : INFO` 가 생긴다.

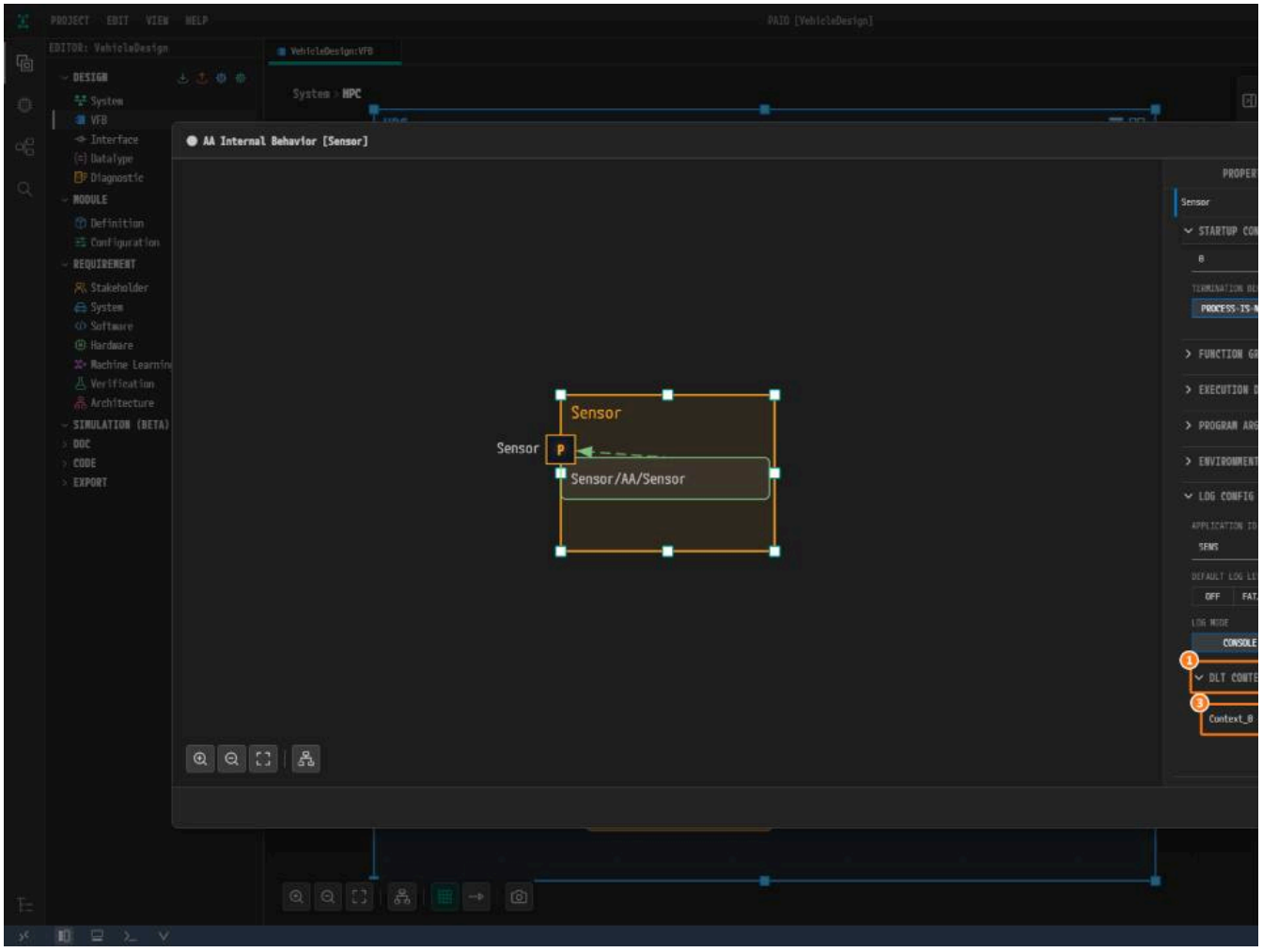


*그림 12-4. ① **DLT CONTEXTS**, ② **+**, ③ `Context_0 : INFO`.*

3. 행 머리글 `Context_0 : INFO` 를 눌러 펼친다.

4. **CONTEXT ID** 의 `Context_0` 을 `SENS` 로 바꾸고 `Tab` 을 누른다. 행 머리글이 `SENS : INFO` 가 된다.

5. **CONTEXT DESCRIPTION** 에 `Sensor context` 를 입력하고 `Tab` 을 누른다.

6. **LOG LEVEL** 은 `INFO` 그대로 둔다.

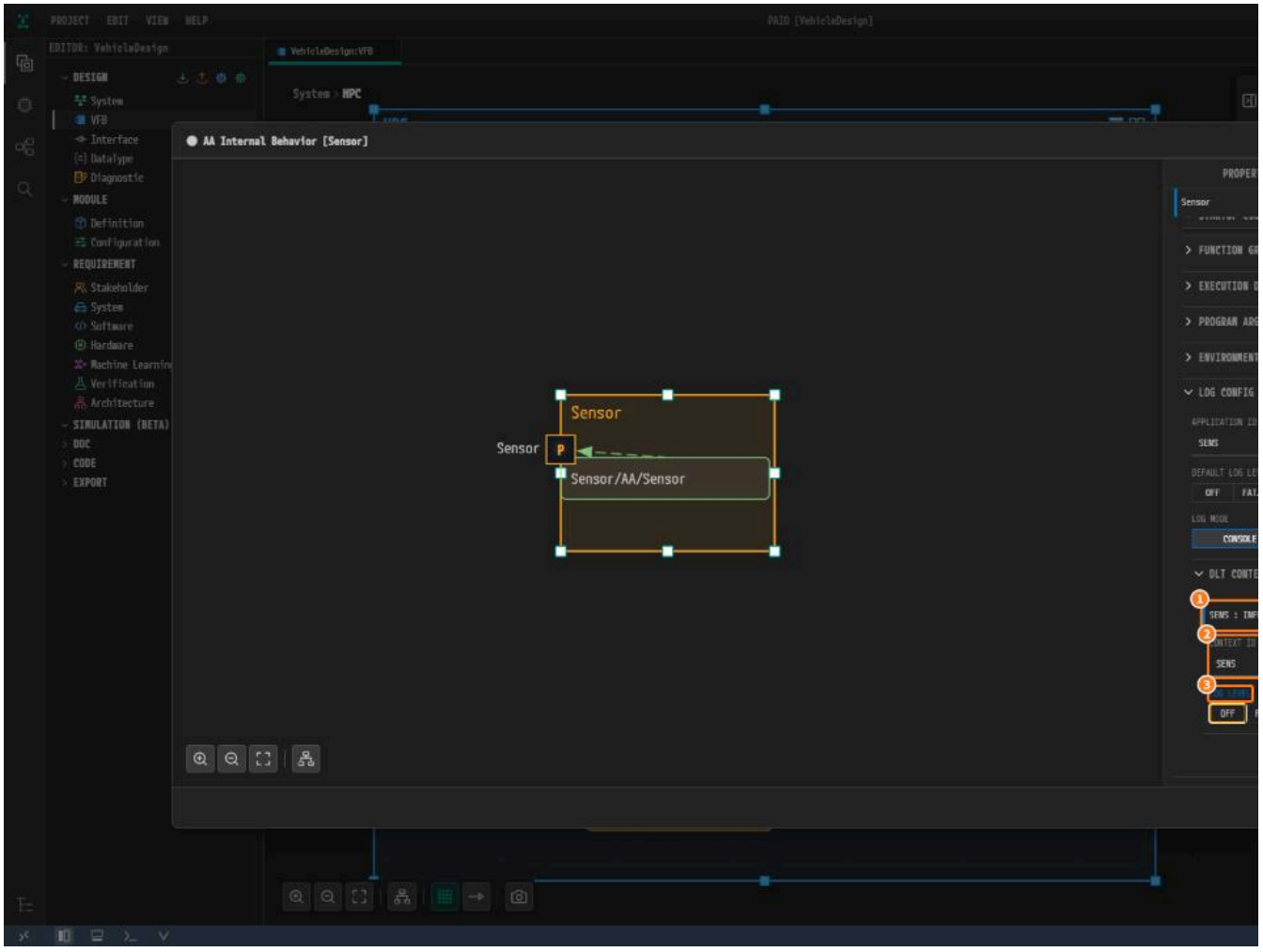


*그림 12-5. ① `SENS : INFO`, ② **CONTEXT ID** `SENS`, ③ **LOG LEVEL** `INFO`.*

저장하고 닫기

1. 다이얼로그 안에서 `Ctrl+S` 를 누른다.

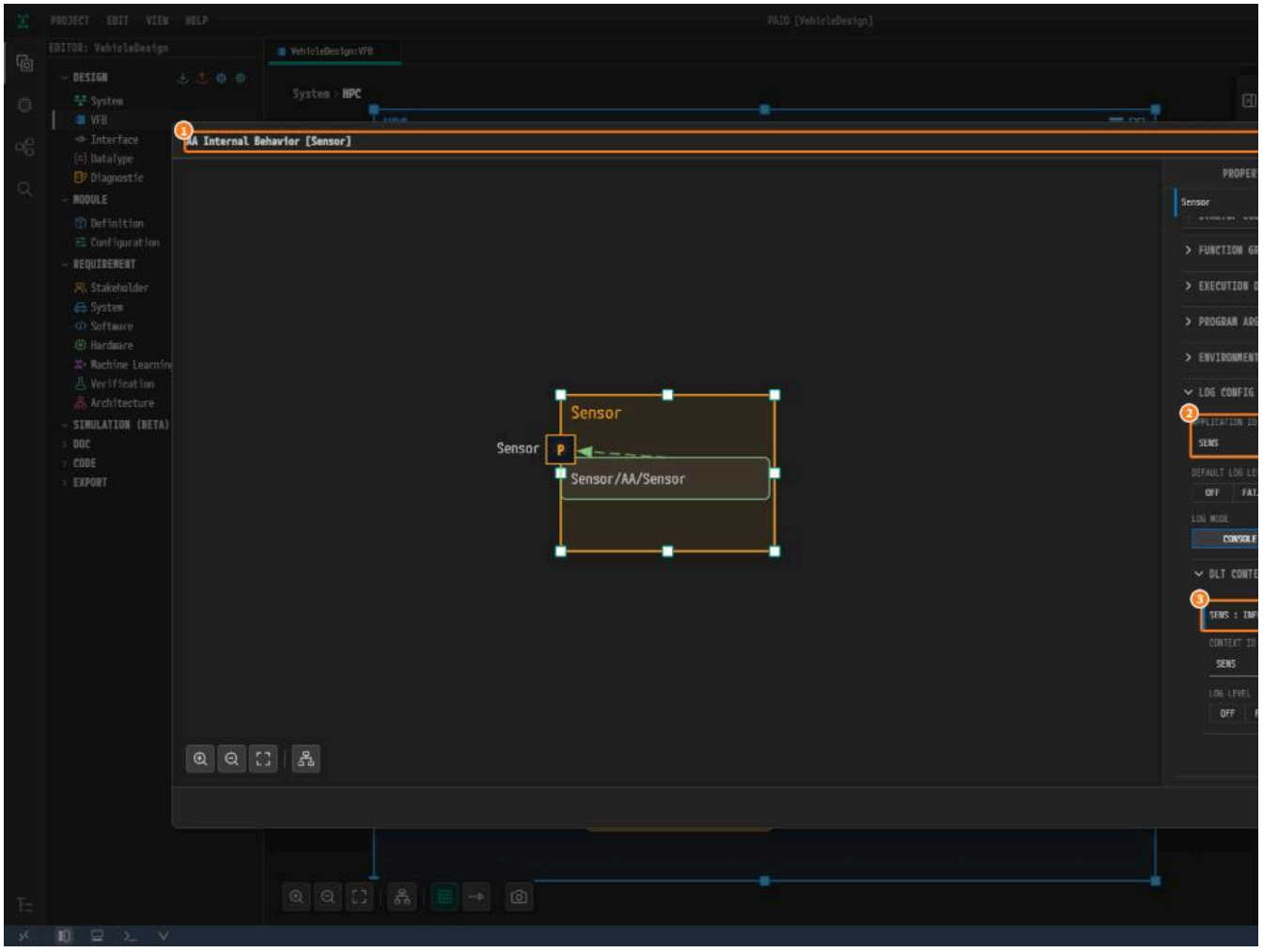


그림 12-6. ① 수정 표시가 없는 제목, ② `SENS`, ③ `SENS : INFO`.

2. **Close** 를 누른다.

Fusion 과 Actuator 에 같은 방법으로 입력하기

표대로 `Fusion` 과 `Actuator` 를 같은 방법으로 입력하고 컨텍스트를 하나씩 추가한다. 각각 `Ctrl+S` 와 **Close** 로 마친다.

머신의 Log and Trace 표 확인하기

1. 프레임 `HPC` 오른쪽 위의 머신 아이콘을 누른다. **Machine Configuration [HPC]** 가 열린다.

2. **MODULE** 탭을 누르고 **Log and Trace** 를 누른다.

3. **ECU ID** 가 `ECU1` 이고 **DLT APPLICATION** 표에 열네 행이 있는지 확인한다. `SENS`, `FUSN`, `ACTA` 는 `VERBOSE` 이고, 플랫폼 프로세스 열한 개는 프로세스 이름이 ID 이며 `ERF`

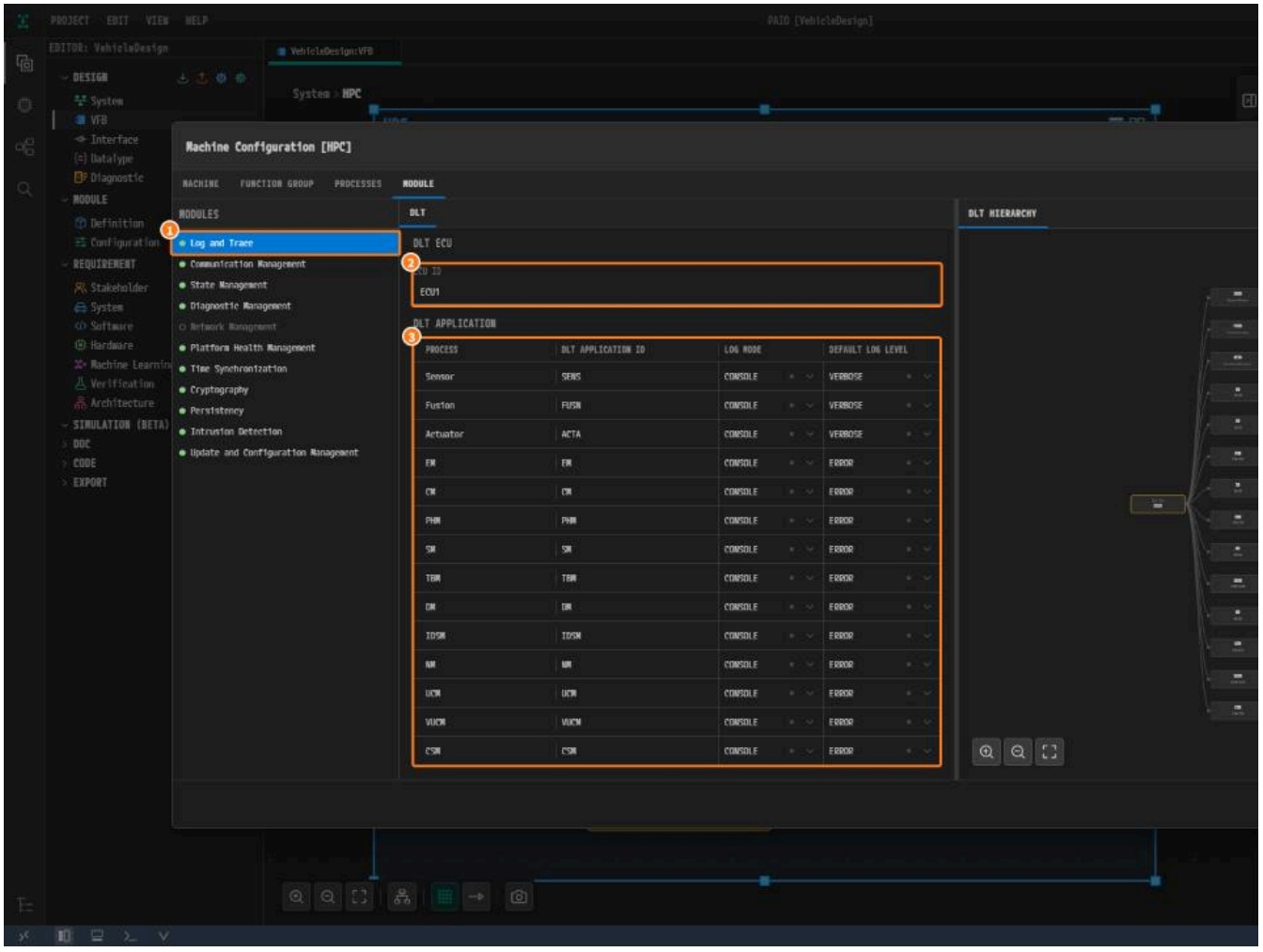


*그림 12-7. ① **Log and Trace**, ② **ECU ID** `ECU1`, ③ 열네 행.*

4. `Sensor` 행을 누른다. `No LogAndTrace ports` 가 보이며 이것이 정상이다.

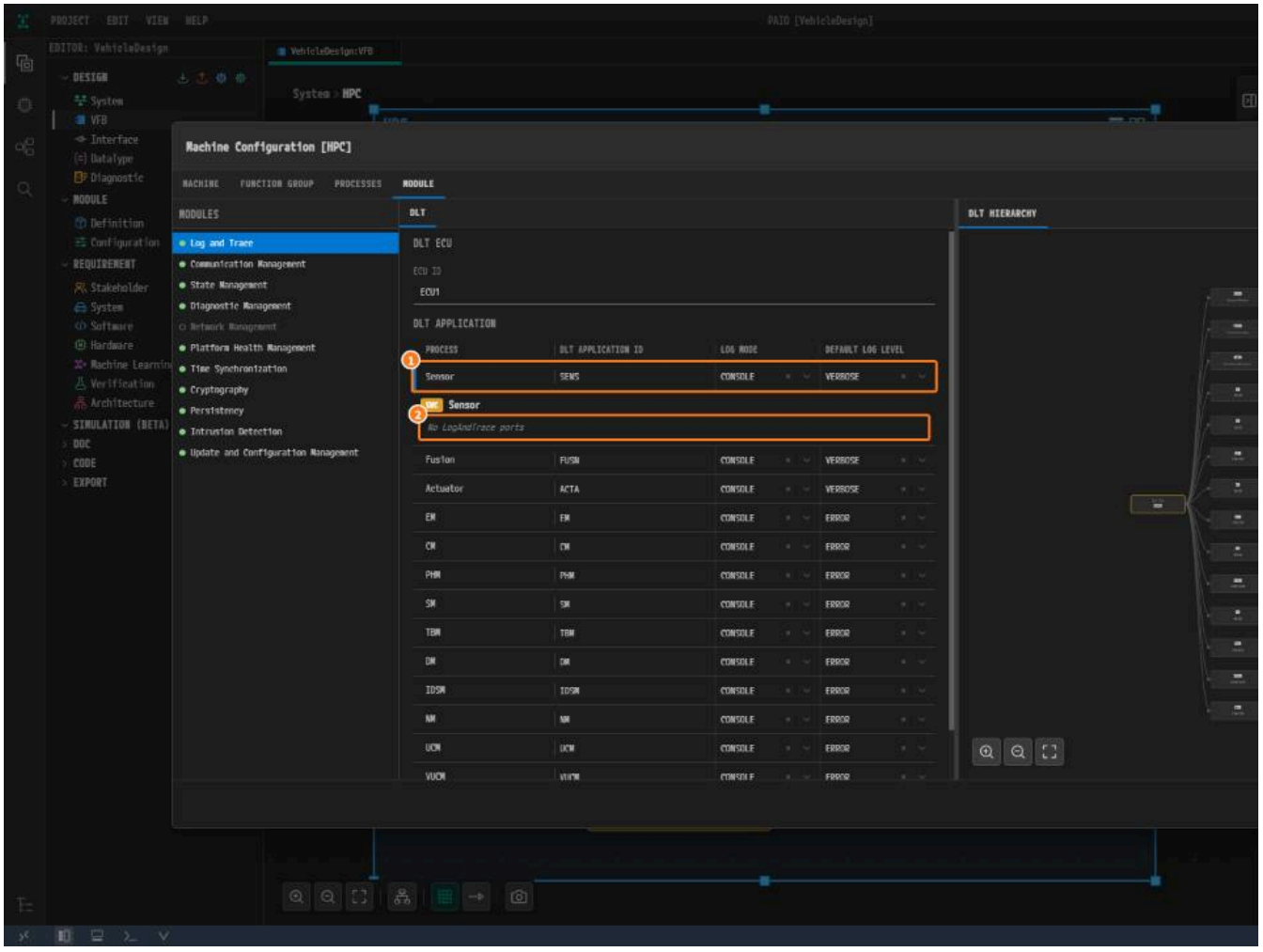


그림 12-8. ① `Sensor` 행, ② `No LogAndTrace ports`.

5. **Close** 를 누른다.

검증 결과 보기

1. **VALIDATION** 패널을 본다. 닫혀 있으면 푸터의 **V** 아이콘을 누른다. `No validation issues.` 가 보인다.

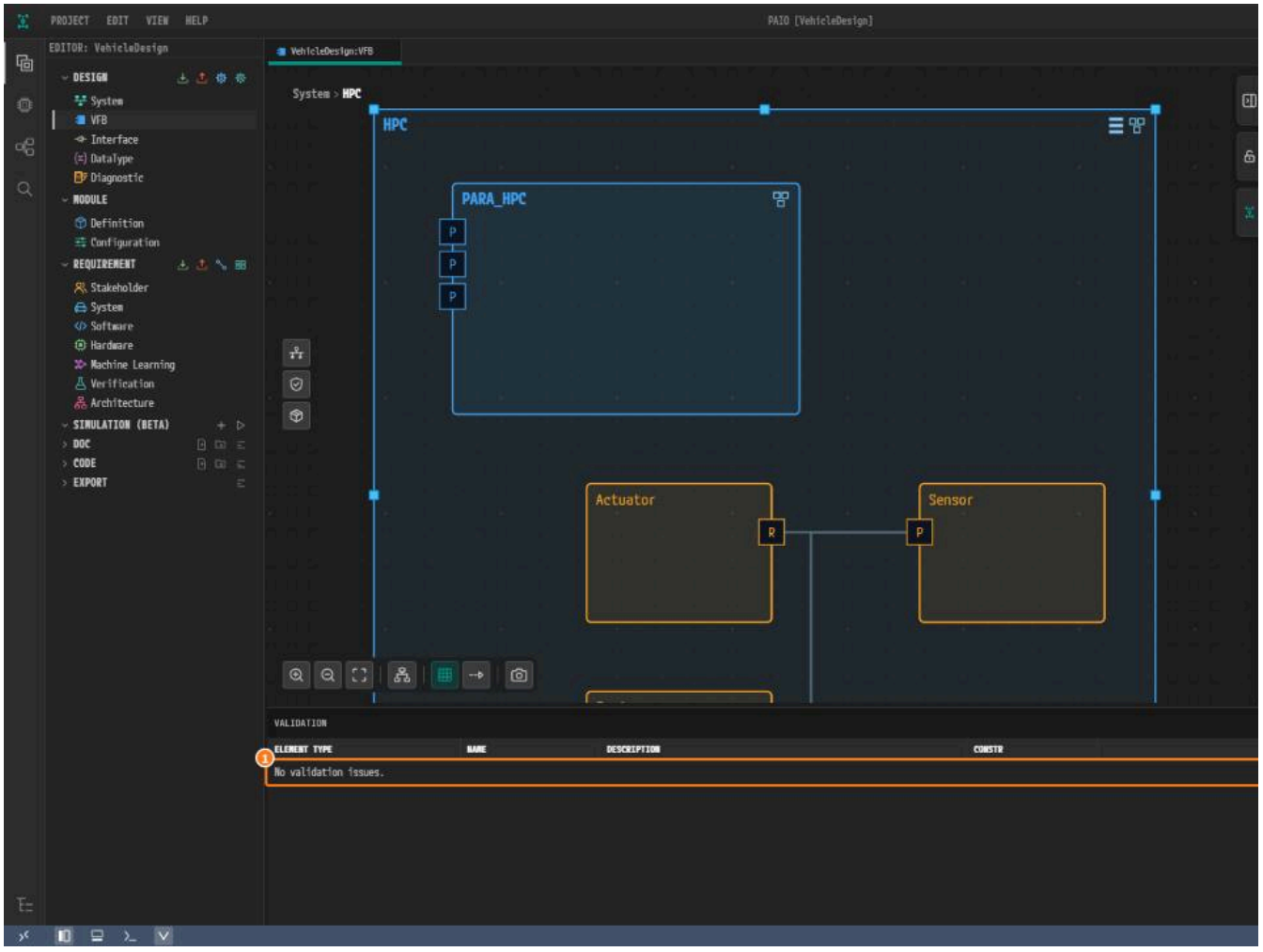


그림 12-9. `No validation issues.`.

왜 이렇게 설계하는가

- **설명은 필수이다.** PAIO 는 모든 DLT 애플리케이션과 컨텍스트에 설명을 요구한다(`constr_5297`, `constr_5299`).
- **ID 는 서로 달라야 한다.** 애플리케이션 ID 는 머신 안에서 겹치면 안 된다. 겹치면 **VALIDATION** 패널이 두 행을 보고하고, 검증과 코드 생성은 생성 전에 멈춘다.
- **로그용 포트는 없다.** 이 설계에는 **ara::log** 포트가 없으므로 `No LogAndTrace ports` 가 정상이다.

확인하기

- [ ] `Sensor`, `Fusion`, `Actuator` 의 **APPLICATION ID** 가 `SENS`, `FUSN`, `ACTA` 이고 `VERBOSE` 이며 컨텍스트가 하나씩 있다.
- [ ] **Log and Trace** 표에 열네 행이 있다.
- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.

다음 단계

검증과 코드 생성에서 설계를 검증하고 `HPC` 의 코드를 생성한다.

검증과 코드 생성

이 장에서 할 일

이 장에서는 **VALIDATION** 패널에서 `No validation issues.` 를 확인하고, **PARA Code Generation** 으로 머신 `HPC` 의 Adaptive Platform 코드를 생성한 뒤 결과를 살펴본다.

시작하기 전에

- 로그와 DLT를 마친 뒤 시작한다.
- 모든 다이얼로그를 저장하고 닫았으며, 어느 탭에도 수정 표시 `●` 가 없다.

위치	항목	값
PARA Code Generation 창	TARGET HPC	`HPC`

검증

1. 푸터의 **V** 아이콘을 눌러 **VALIDATION** 패널을 연다. `No validation issues.` 가 보인다.

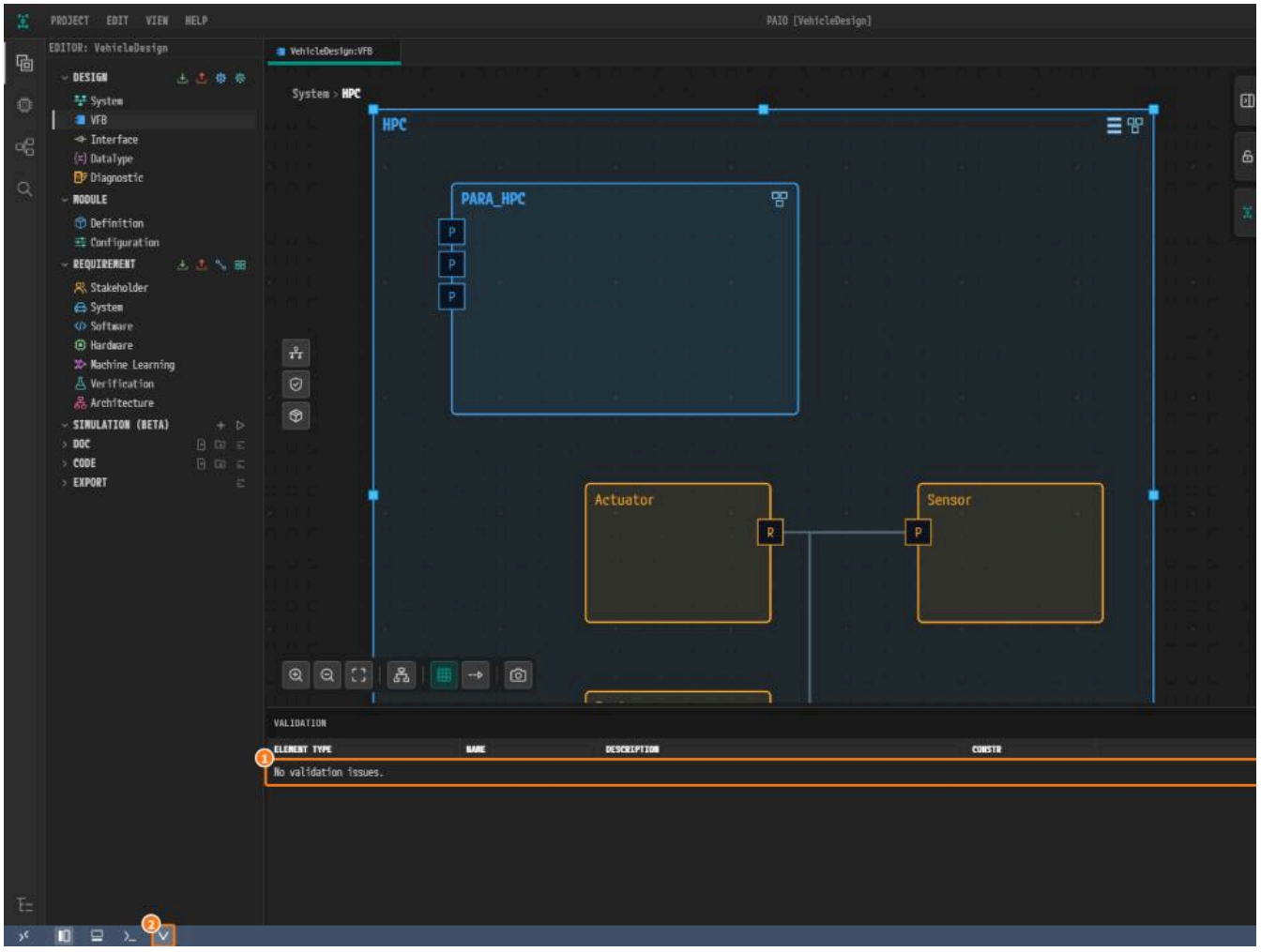


*그림 13-1. ① `No validation issues.`, ② 배지가 없는 **V** 아이콘.*

> **참고.** 패널에 행이 보이면 그 행을 누른다. PAIO 가 값이 있는 에디터나 다이얼로그를 열고 필드를 표시한다. 값을 고치고 `Ctrl+S` 를 누른 뒤 패널을 다시 본다. 자주 나오는 메시지

코드 생성

1. **V** 아이콘으로 패널을 닫는다. **DESIGN** 머리글에서 **PARA Code Generation** 아이콘을 누른다. 아이콘 네 개 중 마지막이며 **PRTE Code Generation** 바로 뒤에 있다.

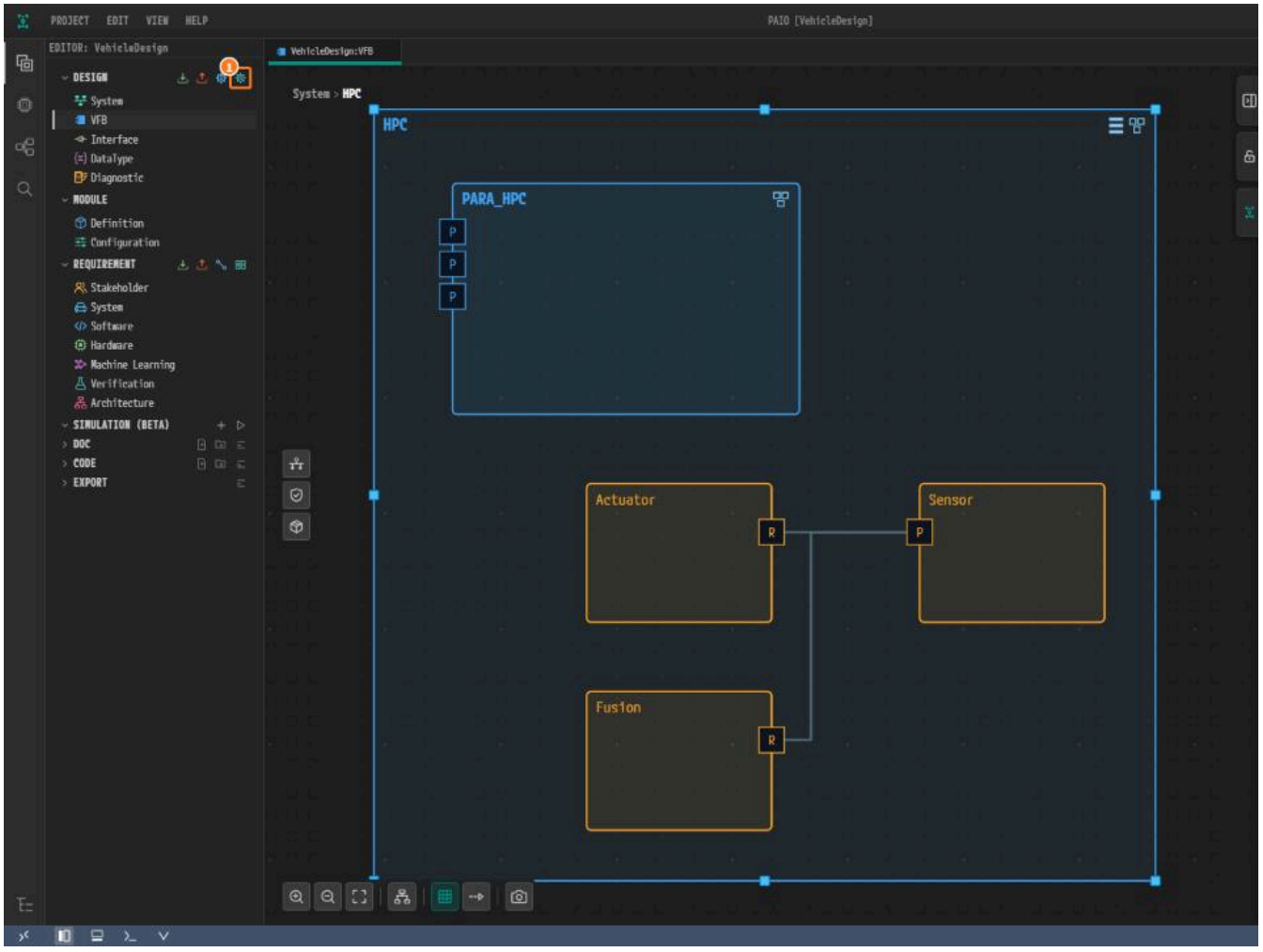


*그림 13-2. ① **PARA Code Generation.***

2. **TARGET HPC** 가 `HPC` 인지 확인하고 **Generate** 를 누른다. 필드 아래 줄에는 `Output: code/<Machine>/` 가 보인다.

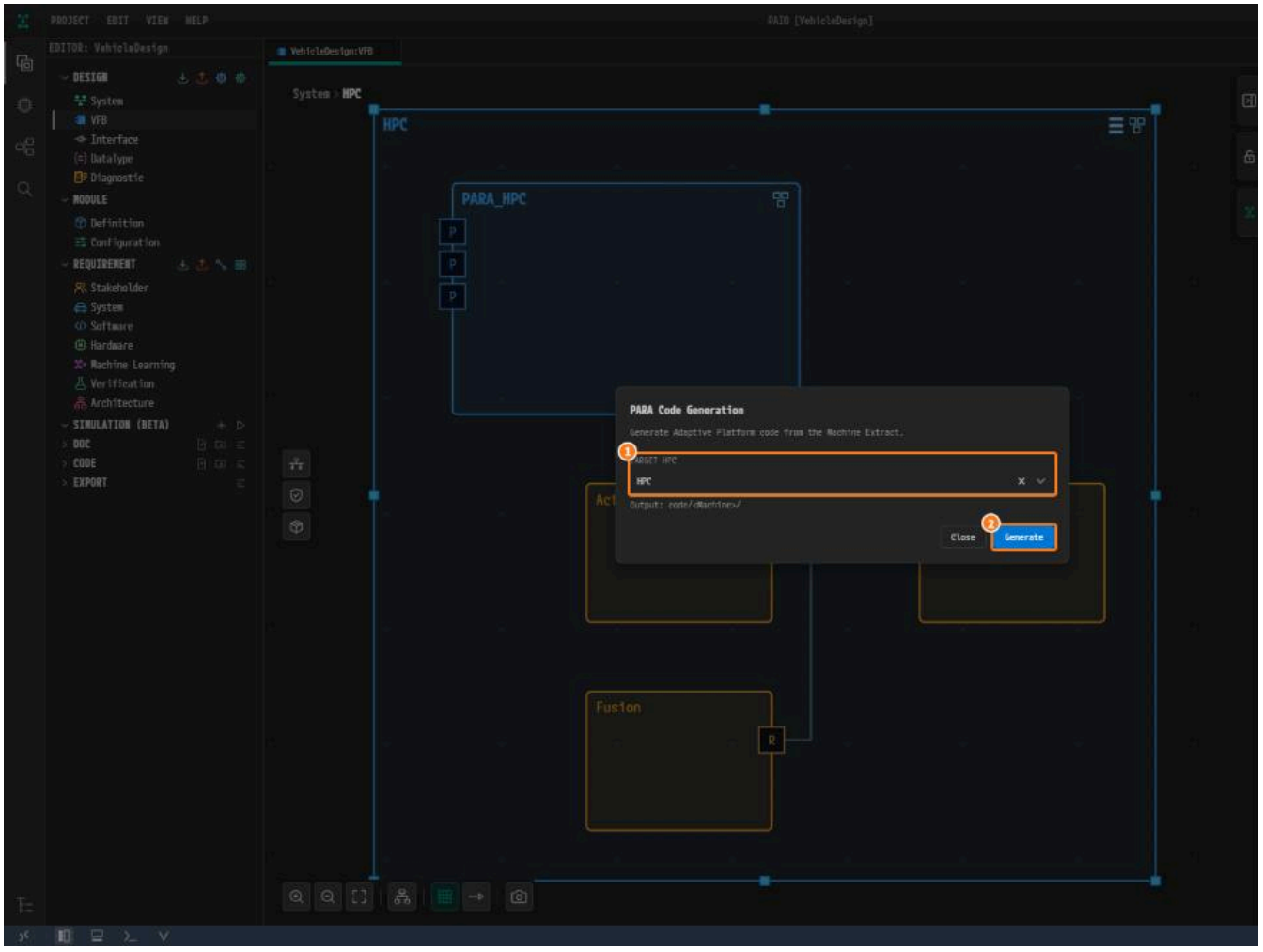


*그림 13-3. ① **TARGET HPC**, ② **Generate**.*

3. 끝나면 창에 `COMPLETE` 와 `Generated 85 file(s) for HPC` 가 보인다. 아래 표에는 `7 warning(s), 1 info` 가 있고 오류는 없다. **Close** 를 누른다.

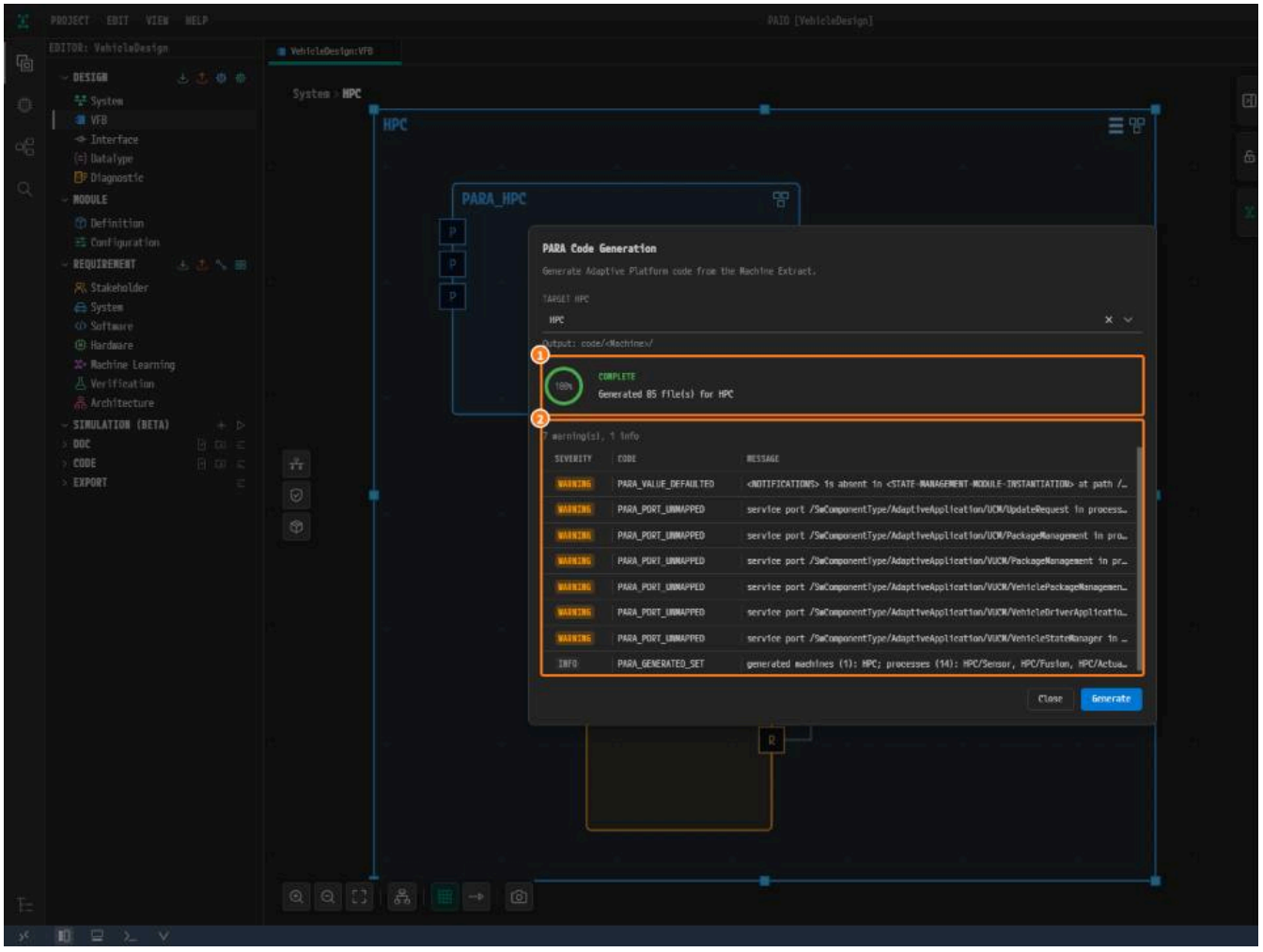


그림 13-4. ① 결과, ② 진단 표.

CODE	뜻
`PARA_VALUE_DEFAULTED`	`HPC` 의 State Management 모듈에 **NOTIFICATIONS** 가 없어서 상태 머신이 정의되지 않고 SM 매니페스트를 쓰지 않는다
`PARA_PORT_UNMAPPED` (여섯 행)	`UCM` 의 서비스 포트 두 개와 `VUCM` 의 네 개에 서비스 인스턴스 매핑이 없어서 그 포트의 코드를 생성하지 않는다
`PARA_GENERATED_SET` (info)	머신 `HPC` 하나와 프로세스 열네 개를 생성했다. 내 프로세스 세 개와 플랫폼 프로세스 열한 개다

경고는 이 설계가 쓰지 않는 플랫폼 기능을 알려 줄 뿐이며 생성을 막지 않는다.

결과 살펴보기

1. **EDITOR** 뷰에서 **CODE** 를 열고 `HPC` 를 연다.
2. **EXPORT** 를 열고 `HPC` 를 연다.

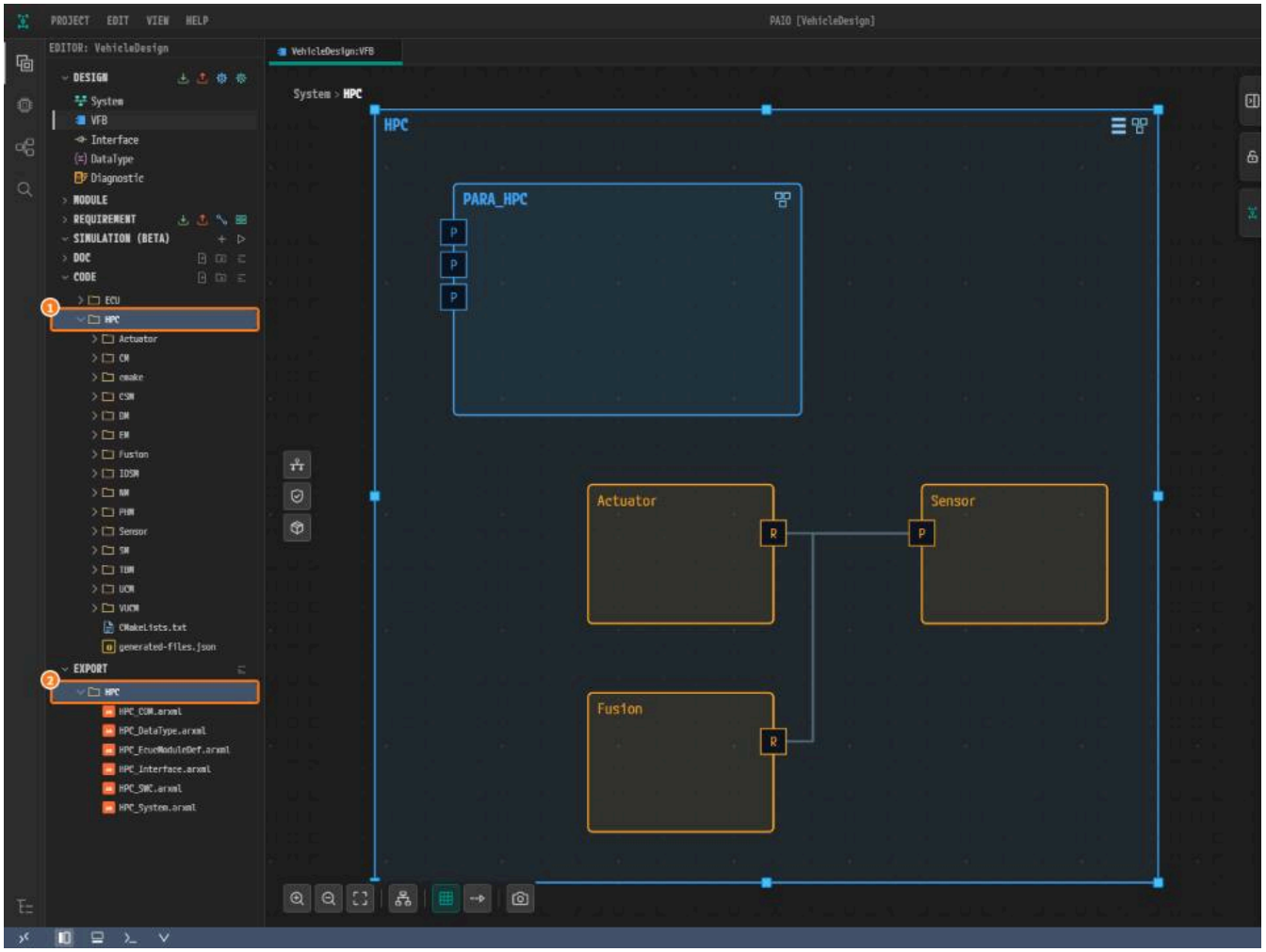


그림 13-5. ① `code/HPC`, ② `export/HPC`.

생성은 폴더 두 개를 쓴다.

폴더	내용
`export/HPC/`	노드 `HPC` 의 Machine Extract: `HPC_COM.arxml`, `HPC_DataType.arxml`, `HPC_EcucModuleDef.arxml`, `HPC_Interface.arxml`, `HPC_SWC.arxml`, `HPC_System.arxml`
`code/HPC/`	이 Extract 로 생성한 머신 `HPC` 의 코드

폴더 `code/HPC/` 의 내용은 다음과 같다.

경로	내용
`CMakeLists.txt`, `cmake/`	머신 전체의 빌드
`Sensor/`, `Fusion/`, `Actuator/`	애플리케이션마다 폴더 하나: `CMakeLists.txt`, `src/main.cpp` 와 애플리케이션 클래스, `include/` 아래 서비스 `Sensor` 의 헤
`EM/`	플랫폼 프로세스 `EM`: 실행 매니페스트와, 머신 `HPC` 와 펑션 그룹 `MachineFG` 의 매니페스트
`CM/`	플랫폼 프로세스 `CM`: 실행 매니페스트와, 서비스 인스턴스 `Provider_Sensor`, `Fusion_Sensor`, `Actuator_Sensor` 세 개의 SO
`PHM/`, `SM/`, `TBM/`, `DM/`, `IDSM/`, `NM/`, `UCM/`, `VUCM/`, `CSM/`	나머지 플랫폼 프로세스: 저마다 실행 매니페스트. `IDSM/` 에는 IdsM 매니페스트, `UCM/` 과 `VUCM/` 에는 애플리케이션 소스도 있다
`generated-files.json`	생성기가 쓴 나머지 파일의 목록

> **참고. Generate** 를 다시 누르면 `generated-files.json` 에 있는 파일을 다시 쓰고, 직접 추가한 파일은 그대로 둔다. **CODE** 아래의 `ECU` 폴더는 AUTOSAR CP 튜토리얼의 코

왜 이렇게 설계하는가

- **검증이 관문이다.** **Generate** 는 먼저 설계를 검증하고, 오류가 있으면 아무것도 쓰기 전에 생성을 멈춘다.
- **Extract 가 입력이다.** 생성기는 설계 전체가 아니라 같은 실행에서 쓴 Machine Extract `export/HPC/` 만 읽는다.
- **머신 하나에 폴더 하나.** 출력 폴더는 머신 이름을 따라 `code/HPC/` 이며, 그 안에 머신의 프로세스마다 폴더가 있다.
- **튜토리얼은 여기서 끝난다.** 검증을 통과하고 코드가 생성되는 설계가 목표다. 애플리케이션 로직 작성과 PARA vECU 에서의 빌드와 실행은 그다음이며 각자의 프로젝트에서 하

확인하기

- [ ] **VALIDATION** 패널에 `No validation issues.` 가 보인다.
- [ ] **PARA Code Generation** 창에 `COMPLETE` 와 `Generated 85 file(s) for HPC` 가 보였고, 오류 행은 없었다.
- [ ] **CODE** 에 `HPC` 와 그 아래 프로세스 폴더 열네 개, `cmake`, `CMakeLists.txt`, `generated-files.json` 이 있다.
- [ ] **EXPORT** 에 `HPC` 와 그 아래 `HPC_*.arxml` 파일 여섯 개가 있다.

다음 단계

튜토리얼의 본편은 여기서 끝난다. 부록에 이 설계의 모든 값과 자주 생기는 문제를 모았다.

부록

이 장에서 할 일

이 부록은 따라 하는 장이 아니라 찾아보는 자료다. 설계를 점검하거나 어느 값이 어느 장에 속하는지 찾을 때 쓴다.

시작하기 전에

- 여기의 값이 본문과 다르면 본문을 따른다.
- 완성된 설계는 검증과 코드 생성을 마친 상태다.

완성된 설계 점검표

머신 `HPC`

- [ ] VFB 루트 캔버스에 컴포지션 `HPC [HPC]` 가 있고, 헤더 오른쪽의 머신 아이콘을 클릭하면 **Machine Configuration [HPC]** 다이얼로그가 열린다.
- [ ] **MACHINE** 탭의 **OS** 가 `POSIX`, **PROCESSOR** 가 `CPU`, **CORE** 표가 `Core0` / `0` 부터 `Core3` / `3` 까지 네 줄, **TRUSTED PLATFORM** 이 **NO SUPPORT** 이다.
- [ ] **FUNCTION GROUP** 탭에 `MachineFG` 하나가 있고 모드는 `Off`, `Startup`, `Shutdown`, `Restart`, `Verify` 이다.
- [ ] **MODULE > Log and Trace** 의 **ECU ID** 가 `ECU1`, **MODULE > State Management > ACTION ITEMS** 의 **MAX ACTION LIST DURATION** 이 `1000`, 단위 `s` 이다.
- [ ] **PROCESSES** 탭 목록에 `Sensor`, `Fusion`, `Actuator` 와 플랫폼 프로세스 열한 개가 있고, **EXECUTION DEPENDENCY** 그래프에 `CM` 에서 `Sensor`, `Fusion`, `Actuator` 로
- [ ] **MODULE** 탭의 **Network Management** 가 꺼져 있다.

토폴로지

- [ ] 커넥터 `EthernetConnector` 의 **CONTROLLER** 가 `EthernetController`, **CONNECTED CHANNEL** 이 `EthernetCluster / PhysicalChannel (VLAN: 13)` 이다
- [ ] **NETWORK ENDPOINT** 가 `192.168.0.11` / **FIXED**, 멀티캐스트 `224.0.0.1` / `30490` 이다.
- [ ] **APPLICATION ENDPOINTS** 에 `UDP EthernetConnector, 192.168.0.11:20001`, `:20002`, `:20003` 세 장만 있다.
- [ ] **SERVICE INSTANCES** 에 `Provider_Sensor` `UDP:20003`, `Actuator_Sensor` `UDP:20001`, `Fusion_Sensor` `UDP:20002` 가 있다.
- [ ] **Communication Cluster Configuration [EthernetCluster]** 의 **NM CLUSTER**, **TIME DOMAIN** 탭 이름 앞에 빈 동그라미가 있다.

데이터 타입, 인터페이스, SOME/IP

- [ ] DataType 편집기의 목록에 기본 제공 타입 13개와 CP 튜토리얼, PRTE Base Set, PARA 플랫폼의 타입에 더해 `ByteArray [DYNAMIC_ARRAY]`, `String [STRING]` 이 있
- [ ] Interface 편집기 **SERVICE** 탭의 `Sensor` 에 이벤트 `SensorData` `ByteArray` 와 메서드 `Calculation`(인자 다섯 개)이 있다.
- [ ] **SERVICE DEPLOYMENT** 의 카드 `SOME/IP` `Sensor` 요약이 `SERVICE ID = 0x000B` `VERSION 1.0` `EVENTS = 1` `METHODS = 1` 이고, 제공자 `Provider_Sensor` `
`Fusion_Sensor` `0xFFFF (ANY)` 가 있다.

Adaptive Application

- [ ] 컴포지션 `HPC` 안에 `PARA_HPC`, `Sensor`, `Fusion`, `Actuator` 가 있고, 세 애플리케이션 각각의 **NAME** 과 **AS PROTOTYPE** 가 같다.
- [ ] 포트 셋이 모두 이름 `Sensor`, **PORT INTERFACE** `Service: Sensor` 이고, 커넥터 `Assembly_Sensor_Fusion`, `Assembly_Sensor_Actuator` 가 있다.
- [ ] 세 애플리케이션 각각의 **AA Internal Behavior** 다이얼로그의 **EXECUTION** 탭에 프로세스 카드가 하나씩 있고, 상태가 `Off` 인 행은 어디에도 없다.
- [ ] `Sensor`, `Fusion`, `Actuator` 의 매핑 박스가 `Sensor/AA/Sensor`, `Fusion/AA/Sensor`, `Actuator/AA/Sensor` 이고, **CLIENT ID** 가 `Fusion` 은 `0x2`, `Actuator`
- [ ] **LOG CONFIG** 의 **APPLICATION ID** 가 `SENS`, `FUSN`, `ACTA` 이고 DLT 컨텍스트 `SENS : INFO`, `FUSN : INFO`, `ACTA : INFO` 가 있다.

검증과 코드 생성

- [ ] **VALIDATION** 패널이 `No validation issues.` 이고, 패널 제목 옆과 푸터 **V** 아이콘에 배지가 없다.
- [ ] `HPC` 의 **PARA Code Generation** 이 오류 없이 `COMPLETE` 로 끝나고, `export/HPC` 에 `HPC_*.arxml` 여섯 개, `code/HPC` 에 생성된 코드가 있다.
- [ ] 수정됨 표시 `•` 가 붙은 탭이나 다이얼로그가 없다.

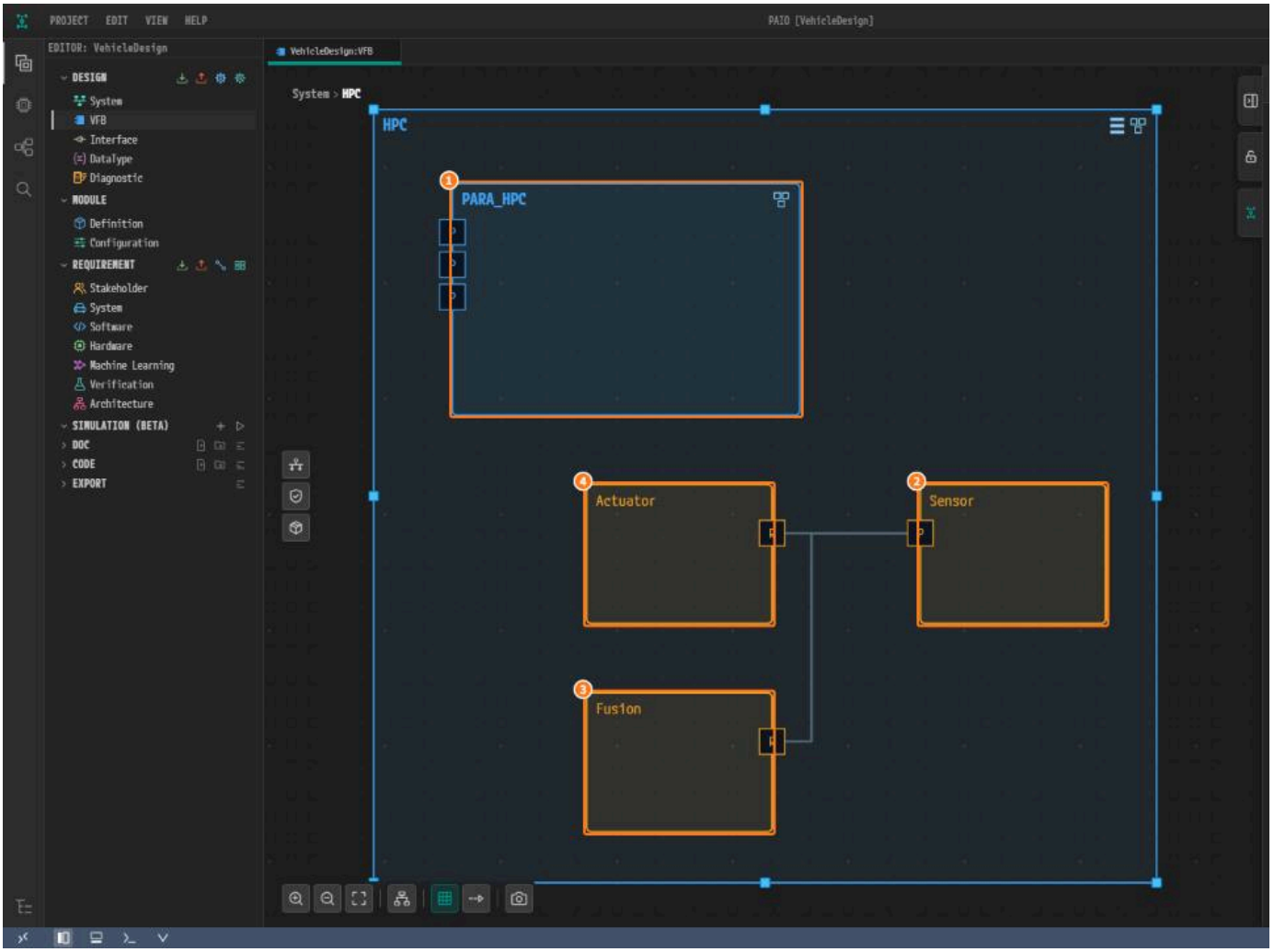


*그림 14-1. 완성된 컴포지션 `HPC` (PARA 설정의 그림 9-9). ① `PARA_HPC`, ② `Sensor`, ③ `Fusion`, ④ `Actuator`. 위치는 **Auto Layout** 에 따라 다르다.*

값 표 모음

프로젝트와 머신 (1장, 2장)

항목	값	장
프로젝트 이름	`VehicleDesign`	AUTOSAR System 튜토리얼
System 편집기의 HPC 노드 이름	`HPC`	AUTOSAR System 튜토리얼
컴포지션 **NAME** / **AS PROTOTYPE**	`HPC` / `HPC`	1
매핑	컴포지션 `HPC` > **ECU/HPC Mapping** > **HPC [HPC]**	1
OS	`POSIX`	2
TRUSTED PLATFORM	**NO SUPPORT** (기본값)	1
PROCESSOR	`CPU` (기본값)	1
CORE / **CORE ID**	`Core0` / `0`, `Core1` / `1`, `Core2` / `2`, `Core3` / `3`	2
함수 그룹	`MachineFG` (기본값, 바꾸지 않음)	1
GROUP NAME / **INITIAL MODE**	`MachineState` / `Off`	1
모드 선언	`Off`, `Startup`, `Shutdown`, `Restart`, `Verify`	1
Log and Trace > **ECU ID**	`ECU1`	2
State Management > **MAX ACTION LIST DURATION**	`1000` `s`	2

토폴로지 (3장)

System 표시가 붙은 값은 AUTOSAR System 튜토리얼에서 입력했다. 3장은 멀티캐스트 포트와 애플리케이션 엔드포인트만 더한다.

요소	항목	값
Ethernet 컨트롤러	**NAME**	`EthernetController` (*System*)
Ethernet 커넥터	**NAME** / **CONTROLLER**	`EthernetConnector` / `EthernetController` (*System*)
통신 클러스터	**NAME**	`EthernetCluster` (*System*)
물리 채널	**CHANNEL NAME** / **VLAN ID**	`PhysicalChannel` / `13` (*System*)
네트워크 엔드포인트	**UNICAST IPV4 ADDRESS** / **ADDRESS SOURCE**	`192.168.0.11` / **FIXED** (*System*)
멀티캐스트	**MULTICAST IPV4 ADDRESS** / **MULTICAST PORT**	`224.0.0.1` (*System*) / `30490`
애플리케이션 엔드포인트	**TYPE UDP**, **PORT NUMBER**	`20001`, `20002`, `20003`
엔드포인트 이름 (자동)	—	`HPC_EthernetConnector_UDP_20001`, `HPC_EthernetConnector_UDP`
Communication Cluster Configuration [EthernetCluster]	**NM CLUSTER**, **TIME DOMAIN** 탭	꺼짐 (기본값, 그대로 둠)

데이터 타입 (4장)

타입	CATEGORY	ELEMENT TYPE	MAX SIZE	C++ NAMESPACE
`ByteArray`	**DYNAMIC_ARRAY**	`uint8`	`1`	`aa::type`
`String`	**STRING**	—	`256`	`aa::type`

프리미티브 11개와 포인터 타입 `VoidPtr`, `ConstVoidPtr` 는 내장 타입이라 입력하지 않는다.

서비스 인터페이스 (5장)

요소	값
서비스 인터페이스	`Sensor`, C++ NAMESPACE `sensor::service`, MAJOR VERSION `1`, MINOR VERSION `0`
이벤트	`SensorData`, TYPE `ByteArray`
메서드	`Calculation`, METHOD TYPE REQUEST AND RESPONSE
인자 1~4	IN `float32` `front`, `rear`, `left`, `right`
인자 5	OUT `float32` `result`
필드	없음

SOME/IP 배포와 서비스 디스커버리 (6장)

요소	값
배포	`Sensor`, SERVICE ID `11` (`0x000B`), MAJOR VERSION `1` / MINOR VERSION `0`
이벤트 배포	`SensorData`, EVENT ID `1`, TRANSPORT UDP
메서드 배포	`Calculation`, METHOD ID `1`, TRANSPORT UDP
이벤트 그룹	`Eventgroup_SensorData`, EVENTGROUP ID `1`, EVENTS `SensorData`
제공자	`Provider_Sensor`, INSTANCE ID `1`
소비자	`Actuator_Sensor`, `Fusion_Sensor`, REQUIRED INSTANCE ID `ANY` (모델 값 `65535`)
제공자 이벤트 그룹	MULTICAST THRESHOLD UNICAST (`0`)

SD 칸	기본값	입력값	제공자 (SD SERVER CONFIG)	소비자 (SD CLIENT CONFIG)
INITIAL DELAY MIN	`10 ms`	`10 us`	입력	입력
INITIAL DELAY MAX	`100 ms`	`100 us`	입력	입력
REPETITION BASE DELAY	`200 ms`	`200 us`	입력	입력
REPETITION MAX	`3`	그대로	기본값	기본값
OFFER CYCLIC DELAY	`500 ms`	`1 ms`	입력	칸 없음
TTL	`0xFFFFFF`	그대로	기본값	기본값
REQUEST RESPONSE DELAY MIN / MAX	`0 s` / `0 s`	그대로	기본값	칸 없음

커넥터 바인딩 (7장)

인스턴스	역할	UDP PORT	TCP PORT
`Provider_Sensor`	제공자(P)	`20003`	비워 둠
`Actuator_Sensor`	소비자(C)	`20001`	비워 둠
`Fusion_Sensor`	소비자(C)	`20002`	비워 둠

세 인스턴스 모두 커넥터 `EthernetConnector` 에 바인딩하고, IP ADDRESS 는 `192.168.0.11` 이 자동으로 들어온다.

Adaptive Application, 포트, 커넥터 (8장)

Adaptive Application	NAME = AS PROTOTYPE	포트	포트 NAME	PORT INTERFACE
센서	`Sensor`	**PPort**	`Sensor`	`Service: Sensor`
융합	`Fusion`	**RPort**	`Sensor`	`Service: Sensor`
액추에이터	`Actuator`	**RPort**	`Sensor`	`Service: Sensor`

커넥터	제공 쪽	요구 쪽
`Assembly_Sensor_Fusion`	`Sensor` / `Sensor`	`Fusion` / `Sensor`
`Assembly_Sensor_Actuator`	`Sensor` / `Sensor`	`Actuator` / `Sensor`

PARA 설정 (9장)

항목	값
가져오기	**Machine Configuration [HPC]** 의 **PARA Platform Import**
충돌 **ECU ID, Max Action List Duration**	**Keep Project** (`ECU1`, `1000`)
들어온 컴포지션	`PARA_HPC`
플랫폼 프로세스	`EM`, `CM`, `PHM`, `SM`, `TBM`, `DM`, `IDSM`, `NM`, `UCM`, `VUCM`, `CSM`
MODULE > Network Management	꺼짐

Executable 과 Process (10장)

Executable 은 셋 모두 미리 채워진 값을 그대로 쓴다.

라벨	값
NAME	Adaptive Application 이름
ROOT SWC PROTOTYPE	`AA`
CATEGORY	`APPLICATION` (파일에는 `APPLICATION_LEVEL`)
BUILD TYPE	비어 있음
REPORTING BEHAVIOR	`REPORTS-EXECUTION-STATE`
VERSION	`1.0.0`
MINIMUM TIMER GRANULARITY	비어 있음

Process NAME	ENTER TIMEOUT / EXIT TIMEOUT	FUNCTION GROUP STATE	EXECUTION DEPENDENCY
`Sensor`	`5` / `5` `s`	`MachineFG`, `Startup`	`CM`, `Running`
`Fusion`	`5` / `5` `s`	`MachineFG`, `Startup`	`CM`, `Running`
`Actuator`	`5` / `5` `s`	`MachineFG`, `Startup`	`CM`, `Running`

프로세스의 나머지 값은 기본값을 그대로 둔다. 플랫폼 프로세스의 값은 PARA 가 준 그대로다.

서비스 인스턴스 매핑 (11장)

	`Sensor`	`Fusion`	`Actuator`
PORT	`Sensor` (`P`)	`Sensor` (`R`)	`Sensor` (`R`)
SERVICE INSTANCE	`Provider_Sensor (SOME/IP)`	`Fusion_Sensor (SOME/IP)`	`Actuator_Sensor (SOME/IP)`
CLIENT ID	칸 없음	`2` → `0x2`	`1` → `0x1`
INSTANCE SPECIFIER	`Sensor/AA/Sensor`	`Fusion/AA/Sensor`	`Actuator/AA/Sensor`
PROCESS (자동)	`Sensor`	`Fusion`	`Actuator`
되읽기 **INSTANCE ID**	`0x0001`	`0xFFFF`	`0xFFFF`
되읽기 **ENDPOINT**	`192.168.0.11:20003`	`192.168.0.11:20002`	`192.168.0.11:20001`
저장 이름 (자동)	`Sensor_AA_Sensor`	`Fusion_AA_Sensor`	`Actuator_AA_Sensor`

플랫폼 애플리케이션의 포트는 매핑하지 않는다. 되읽기의 **SERVICE ID** 와 **VERSION** 은 셋 다 `0x000B` 와 `1.0` 이다.

로그와 DLT (12장)

프로세스	APPLICATION ID	DEFAULT LOG LEVEL	LOG MODE	DLT CONTEXTS (CONTEXT ID : LOG LEVEL)
`Sensor`	`SENS`	`VERBOSE`	`CONSOLE`	`SENS : INFO`
`Fusion`	`FUSN`	`VERBOSE`	`CONSOLE`	`FUSN : INFO`
`Actuator`	`ACTA`	`VERBOSE`	`CONSOLE`	`ACTA : INFO`

APPLICATION DESCRIPTION 은 `Sensor application`, `Fusion application`, `Actuator application`, **CONTEXT DESCRIPTION** 은 `Sensor context`, `Fusion c`
의 로그 설정은 PARA 가 준 그대로다.

프로젝트 파일 구성

이 튜토리얼이 만들거나 고치는 파일과 그 파일을 쓰는 장이다. System, CP 튜토리얼의 파일은 뺐다.

```
VehicleDesign/
├── VehicleDesign.io.prj
├── design/
│   ├── DATATYPE.io.model                    4, 9장
│   ├── INTERFACE.io.model                   5, 6, 9장
│   ├── IVC.io.model                         3, 7장
│   ├── SERVICE.io.model                     6장
│   ├── SIGNAL.io.model                      6장
│   ├── SYSTEM.io.model                      1장
│   ├── VFB.io.model                         1, 8, 9장
│   ├── Module/
│   │   └── Machine/
│   │       └── HPC.machine.io.model         1, 2, 9장
│   └── SWC/
│       ├── AdaptiveApplication/
│       │   ├── Actuator.swc.io.model        8, 10~12장
│       │   ├── Fusion.swc.io.model          8, 10~12장
│       │   ├── Sensor.swc.io.model          8, 10~12장
│       │   └── HPC/                         9장  플랫폼 애플리케이션 11개
│       ├── Composition/
│       │   ├── HPC.swc.io.model             1, 8, 9장
│       │   ├── PARA_HPC.swc.io.model        9장
│       │   └── System.swc.io.model          1장
│       └── Layout/
│           ├── HPC.vfb.layout               8, 9장
│           ├── PARA_HPC.vfb.layout          9장
│           └── System.vfb.layout            1장
├── export/
│   └── HPC/                                 13장
│       ├── HPC_COM.arxml
│       ├── HPC_DataType.arxml
│       ├── HPC_EcucModuleDef.arxml
│       ├── HPC_Interface.arxml
│       ├── HPC_SWC.arxml
│       └── HPC_System.arxml
└── code/
    └── HPC/                                 13장
        ├── CMakeLists.txt
        ├── generated-files.json
        ├── cmake/
        ├── Actuator/  Fusion/  Sensor/
        ├── CM/  CSM/  DM/  EM/  IDSM/  NM/  PHM/  SM/  TBM/  UCM/  VUCM/
        └── ...
```

- `HPC.machine.io.model` 의 이름은 System 편집기의 HPC 노드 이름 `HPC` 를 따른다.
- `*.vfb.layout` 은 노드와 포트의 위치만 담는다. 익스포트되지 않는다.
- `doc`, `workflow`, `import` 폴더는 쓰지 않는다. `code/ECU` 는 CP 튜토리얼의 코드다.
- `SIGNAL.io.model`, `Sensor_SR`, `Sensor_CS` 는 SOME/IP 배포와 서비스 인스턴스의 SOME/IP 배포에서 자동으로 파생된다.

자주 겪는 문제

증상 또는 검증 메시지	원인	해결
`State-Management module "SM" of machine "HPC" is missing maxActionListDuration. (constr_10641)`	1장이 끝나면 정상적으로 남는다	**MAX ACTION LIST DURATION** 에 `1000` 을 넣고 `Ctrl+S`
`The DLT application-id "SENS" truncates to "SENS" and collides with "SENS" (Sensor).` 와 짝을 이루는 `… (Fusion).` 두 행	두 프로세스의 **APPLICATION ID** 가 같다	**APPLICATION ID** 를 서로 다르게(`FUSN` 등) 고치고
`The process "Sensor" declares no functionGroupState; at least one is required.`	**FUNCTION GROUP STATE** 행이 없다	**+** 로 행을 만들고 `Startup` 을 고른 뒤 `Ctrl+S`
`"Sensor" of "Sensor": referenced port interface "SensorIf" not found.`	포트가 없는 인터페이스를 가리킨다	**PORT INTERFACE** 를 `Service: Sensor` 로 고쳐 VFB 탭을 다시 연다
머신 아이콘을 클릭해도 **Machine Configuration** 다이얼로그가 열리지 않음	매핑할 때 열려 있던 VFB 탭이다	VFB 탭을 닫고 **VFB** 를 다시 연다
다이얼로그에서 넣은 값이 저장되지 않음	**Close** 만 눌렀다	다이얼로그에서 `Ctrl+S` 를 누른 뒤 **Close**
SD 칸을 여러 개 고쳤는데 마지막 칸만 남음	연달아 고치면 앞의 칸이 기본값으로 돌아간다	값마다 `Tab` 을 누르고 **INTERFACES** 의 `Sensor`
`Fusion` 이나 `Actuator` 의 실행 의존성이 `Sensor`, `Running` 으로 들어감	새 행의 기본값이 첫 프로세스다	**PROCESS** 를 `CM` 으로 고른다
EXECUTION DEPENDENCY 의 **PROCESS** 목록에 `CM` 이 없음	PARA 플랫폼을 가져오지 않았다	**PARA Platform Import** 를 하고 저장한다
코드 생성이 `PARA_REQUIRED_VALUE_MISSING` (`<NM-NETWORK-HANDLE>`) 오류로 멈춤	**Network Management** 모듈이 켜져 있다	**MODULE** 탭에서 **Network Management** 를 끄고
프로토타입 이름이 `Composition_0` 이나 `AdaptiveApp_0_0` 으로 남음	**AS PROTOTYPE** 는 **NAME** 을 따라 바뀌지 않는다	두 칸에 이름을 각각 입력하고 커넥터 이름을 확인한
커넥터 이름이 `Assembly_Sensor_Sensor_Fusion_Sensor`, `Assembly_Sensor_Sensor_Actuator_Sensor` 로 남음	커넥터의 기본 이름이다	**ASSEMBLY** 에서 `Assembly_Sensor_Fusion`, `Assembly_Sensor_Actuator` 로 바꾸고 `Ctrl`

확인하기

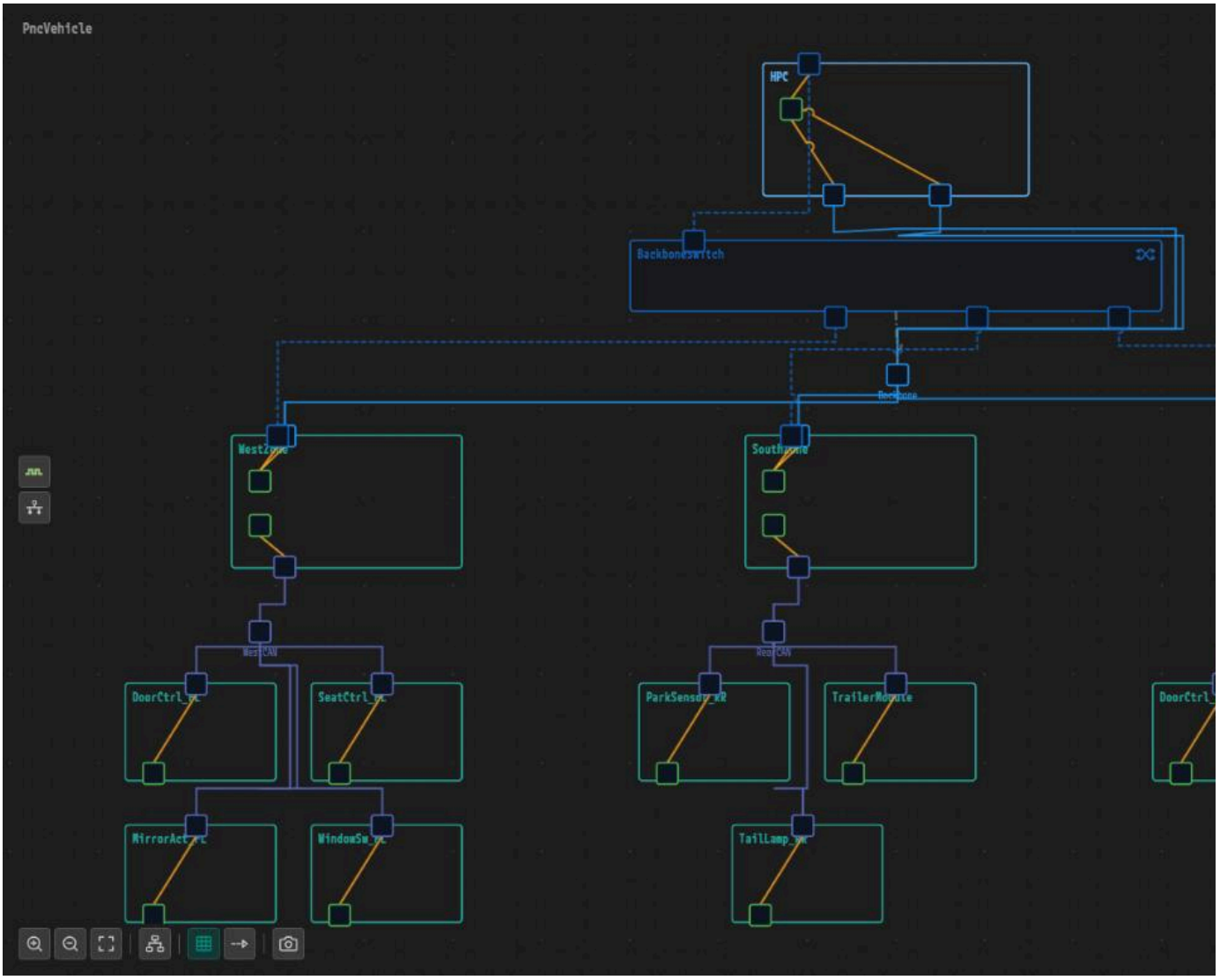
- [ ] "완성된 설계 점검표" 의 항목이 모두 맞다.
- [ ] 값 표 모음의 값이 내 설계의 화면 값과 같다.
- [ ] `design/` 아래 파일이 "프로젝트 파일 구성" 과 같다.

다음 단계

설계를 바꾸려면 해당 장에서 값을 고치고 검증과 코드 생성을 다시 따라가며, 처음부터 하려면 AUTOSAR AP 튜토리얼로 돌아간다.

PNC 설계 튜토리얼

이 튜토리얼은 조널(zonal) 차량에 PNC(Partial Network Cluster) 3개를 PAIO로 설계한다. 토폴로지부터 ARXML 익스포트까지 다룬다.



이 튜토리얼에서 만드는 차량. 맨 위는 백본 스위치를 내장한 HPC(AUTOSAR Adaptive Platform), 가운데는 Zone ECU 3개, 맨 아래는 CAN FD 버스 3개에 매달린 말단 ECU 10개

무엇을 만드는가

"문이 열리면 Comfort 망만 깨우고, 후진 기어에서만 주차 보조 망이 살아나며, 트레일러가 붙었을 때만 트레일러 망이 동작한다"는 차량 요구를 PNC 3개로 표현한다.

PNC	식별자	웨이크업 트리거
`PNC_Comfort`	`17`	도어 언락, 도어 개방
`PNC_ParkAssist`	`18`	후진 기어. HPC가 요청한다.
`PNC_Trailer`	`19`	트레일러 체결

준비물

- PAIO **0.6.0**(AUTOSAR R25-11).
- 미리 만든 프로젝트는 필요 없다. 2장은 프로젝트가 열려 있지 않은 PAIO 창에서 시작한다.
- 1920×1080 이상의 화면. PNC 다이얼로그와 Machine Configuration 다이얼로그가 넓다.
- 활성 라이선스와 서버 연결. 12장(ARXML 익스포트)에서만 필요하다.
- PNC 개념을 대략 알면 충분하다. 어느 화면에 무엇을 입력하는지는 장마다 보여 준다.

장 구성

장	하는 일	사용
1. 예제와 설계 목표	차량 시나리오에서 PNC 3개를 도출하고, PNC가 걸친 층을 확인한다	—
2. 토폴로지 설계	프로젝트를 만들고 노드, 클러스터, 백본 스위치와 커플링 포트를 배치한다	System 편집기
3. PNC 정의와 커넥터 PNC 속성	PNC 3개를 정의하고, 커넥터에 게이트웨이, 웨이크업, 필터 마스크를 입력하고 NM 클러스터와 NM 노드 값을 채운다	System 편집기, PNC 다
4. 신호·PDU·프레임과 I-PDU Group	프레임, I-PDU, 신호, NM-PDU를 만들어 매핑하고, PNC별 Rx/Tx I-PDU Group 8개를 만든다	Signal, PNC 다이얼로그
5. 인터페이스와 SOME/IP 서비스	서비스 인터페이스와 SOME/IP 배포를 만들고 인스턴스를 커넥터에 바인딩한다	Interface, System 편집
6. VFB와 VFC(PortGroup)	노드마다 컴포지션 1개와 원자 SWC를 만들어 잇고, 기능별 포트를 PortGroup(= VFC)으로 묶는다	VFB
7. PNC 다이얼로그 — PNC별 바인딩	PNC마다 VFC, I-PDU Group, 물리 채널, 서비스 인스턴스, 웨이크업 프레임을 바인딩한다	PNC 다이얼로그
8. PDU GROUPS·MATRIX·NM VECTOR 검토	뷰어 세 탭으로 설계를 교차 검토한다	PNC 다이얼로그
9. AP 머신 — NM 핸들과 FunctionGroup	HPC에 FunctionGroup 3개와 State Management 값을 넣고, NM 네트워크 핸들 3개를 FunctionGroup 상태와 매핑한다	VFB, Machine Configu
10. PARA 플랫폼 임포트와 PNC 요청 포트	PARA를 임포트하고, 응용 AA에 `ara::nm` 포트와 핸들 매핑을 단다	VFB, Interface, Machir
11. System 에디터와 검증	SYSTEM 패널을 확인하고 VALIDATION 패널이 보고한 항목을 고친다	System 편집기, VFB, In
12. ARXML 익스포트	설계를 내보내고 결과에서 PNC 매핑을 확인한다	EDITOR 뷰, XML 편집기
부록	설계 체크리스트, 프로젝트 파일 구조, 모델 조각, 자주 겪는 문제	—

이 튜토리얼을 읽는 법

- **장은 순서대로 따라간다.** 각 장은 앞 장의 결과에서 시작한다. 되돌아가는 곳은 한 곳뿐이다. 4장을 마치고 3장으로 돌아와 NM-PDU 참조를 채운다. 두 장 모두 해당 단계에 안내
- **반복 작업은 한 번만 보여 준다.** 하나(신호 42개, 컴포지션 14개, PortGroup 33개 등 가운데)를 절차로 보여 주고, 나머지는 값 표로 주며 같은 방법으로 입력한다.
- **"시작하기 전에"** 에는 시작할 때 모델에 있어야 할 것과, 항목 수가 독자의 프로젝트와 다른 그림이 적혀 있다.
- UI 라벨은 화면에 보이는 대로 **굵게**, 모델 요소 이름과 값은 `code`로 적는다.

다음 단계

예제와 설계 목표부터 시작한다.

예제와 설계 목표

이 장에서 할 일

이 장은 화면 조작 없이 읽는 장이다. 차량 요구에서 예제의 PNC(Partial Network Cluster) 3개를 도출하고, PNC의 각 부분을 PAIO의 어느 편집기가 기록하는지 보여 준다.

시작하기 전에

- 준비할 것은 없다. 모델 작업은 다음 장인 토폴로지 설계에서 시작한다.

설계 목표

이 튜토리얼은 다음 차량 수준 요구를 AUTOSAR PNC로 모델링한다.

> 문이 열리면 Comfort 망만 깨우고, 후진 기어에서만 주차 보조 망이 살아나며, 트레일러가 붙었을 때만 트레일러 망이 동작한다.

예제 차량은 조널 아키텍처이다. HPC 하나가 AUTOSAR Adaptive Platform으로 동작하고, 그 아래 Zone ECU 3개가 Classic Platform으로 동작하며, 각 Zone 아래 CAN FD 버스 Ethernet으로 이어진다.

완성된 모델은 **VALIDATION** 패널의 보고 항목이 0건이다(System 에디터와 검증). 익스포트한 ARXML은 `PNC_COM.arxml`의 익스포트 자체에서 생기는 오류를 빼면 AUTOSAR R

PNC란 무엇인가

PNC는 **함께 깨어나고 함께 잠드는 기능 집합**이다. 차량이 켜져 있어도 필요 없는 ECU와 통신은 잠들어 있게 해 전력과 버스 부하를 줄인다. PNC 하나는 다음 네 가지로 정해진다.

구성 요소	뜻
식별자(PNC identifier)	NM PDU 안의 PNC 비트 벡터에서 이 PNC가 차지하는 비트 번호
참여 물리 채널	이 PNC가 켜지면 함께 깨어나는 CAN 채널과 Ethernet VLAN
통신 단위	이 PNC가 켜고 끄는 I-PDU Group(Rx/Tx 쌍)
기능 단위	이 PNC가 제어하는 SWC 포트의 묶음. AUTOSAR에서는 PortGroup, 이 튜토리얼에서는 VFC(Virtual Function Cluster)라고 부른다.

기능을 **차량 상황(웨이크업 트리거)** 별로 묶으면, 거기 참여하는 ECU와 버스에서 PNC와 그 채널이 정해진다.

차량 시나리오에서 PNC 3개 도출하기

요구에는 차량 상황이 셋 있다. 다음 표가 이 튜토리얼 전체의 설계 명세이다.

PNC	식별자	차량 상황(웨이크업 트리거)	참여 ECU	관련 버스
`PNC_Comfort`	17	도어 언락, 도어 개방(`DoorStatus` 프레임)	`DoorCtrl_FL`, `DoorCtrl_FR`, `SeatCtrl_FL`, `SeatCtrl_FR`, `MirrorAct_FL`, `MirrorAct_FR`, `WindowSw_FL`, `WestZone`, `EastZone`, HPC의 `ComfortApp`	`WestCAN`, `EastCAN`, `VLAN_Body`
`PNC_ParkAssist`	18	후진 기어. HPC의 `ParkAssistFG` 상태가 `Active`가 될 때	`ParkSensor_RR`, `SouthZone`, HPC의 `ParkAssistApp`	`RearCAN`, `VLAN_Rear`
`PNC_Trailer`	19	트레일러 체결(`TrailerStatus` 프레임)	`TrailerModule`, `TailLamp_RR`, `SouthZone`, HPC의 `VehicleStateApp`	`RearCAN`, `VLAN_Rear`

`PNC_Comfort` 같은 프레임 트리거는 커넥터의 웨이크업 속성이 된다(PNC 정의와 커넥터 PNC 속성). `PNC_ParkAssist`의 FunctionGroup 트리거는 AP 머신의 NM 네트워크 한



*그림 1-1. 완성 예제에서 PNC 다이얼로그의 **MATRIX** 탭. 행은 PNC, 열은 물리 채널이며, 표시는 "관련 버스" 열과 같다. 이 화면은 PDU GROUPS·MATRIX·NM VECTOR 검토에서 다시*

PNC 식별자를 17, 18, 19로 잡은 이유

PNC 식별자는 **NM PDU 안의 비트 번호**이므로 PNC 비트 벡터 안에 들어가야 한다. 이 예제는 CanNm 표준 배치를 따른다. Byte 0은 CBV(Control Bit Vector), Byte 1은 NID(Nod

그래서 **PNC VECTOR OFFSET**은 `2`, **PNC VECTOR LENGTH**는 `6`이고, 쓸 수 있는 식별자는 비트 `16`(2 × 8)부터 `63`((2 + 6) × 8 − 1)까지이다. AUTOSAR는 이 범위를 `constr_3`서 오류로 보고된다.

초안에서는 `8`, `9`, `10`을 썼고, 이는 `constr_3040` 위반이었다. PNC 정의와 커넥터 PNC 속성에서 이 오류를 내고 고쳐 본다. 오프셋과 길이는 PDU GROUPS·MATRIX·NM VECTOR

PNC가 걸친 5개 층

AUTOSAR는 PNC를 여러 요소에 나눠 기록하고, PAIO는 그 요소마다 담당 편집기가 다르다.

AUTOSAR 요소	역할	PA
`PncMapping`(SystemMapping)	PNC 식별자, 참여 물리 채널, I-PDU Group, VFC, 서비스 인스턴스, 웨이크업 프레임	System 편집기의 PNC 다이얼로그
`CommunicationConnector`의 PNC 속성	`pncGatewayType`, 웨이크업 CAN ID와 마스크, PNC 필터 배열 마스크	System 편집기 속성 패널
`ISignalIPduGroup`	PNC가 켜고 끄는 통신 단위(Rx/Tx, 컨테이너와 리프)	Signal 편집기와 PNC 다이얼로그
`PortGroup`(= VFC)	PNC가 제어하는 기능 단위. SWC 포트의 묶음	원자 SWC는 VFB 편집기, 컴포지션과 **HIERARCHY**
`NmNetworkHandle`과 `NmHandleToFunctionGroupStateMapping`	AP 머신(HPC)이 PNC를 요청하고 해제하는 창구, FunctionGroup 상태 연동	Machine Configuration 다이얼로그
`CanNmCluster`, `UdpNmCluster`	NM PDU 안의 PNC 비트 벡터(오프셋과 길이), NM 노드	System 편집기의 NM 설정과 PNC 다

PNC 매핑이 참조하는 서비스 인스턴스는 5장에서 만든다. "NM" 두 행은 서로 다르다. `NmNetworkHandle`은 PNC를 요청하는 소프트웨어 창구이고, `CanNmCluster`와 `UdpNmCl`

확인하기

- [ ] PNC 3개의 식별자, 트리거, 관련 버스를 말할 수 있다.
- [ ] `17`, `18`, `19`는 유효하고 `8`, `9`, `10`은 아닌 이유를 보일 수 있다.
- [ ] PNC를 기록하는 편집기를 안다. System 편집기(PNC 다이얼로그 포함), Signal, Interface, VFB, Machine Configuration이다.

다음 단계

토폴로지 설계에서 프로젝트 `PNC`를 만들고, 이 장의 차량을 System 편집기에 그린다.

토폴로지 설계

이 장에서 할 일

이 장에서는 프로젝트 `PNC`를 만들고 System 편집기에 차량 토폴로지를 그린다. HPC, Zone ECU 3개, 말단 ECU 10개, CAN FD 클러스터 3개, VLAN 두 개의 Ethernet 백본, HPC는 표를 보고 추가한다.

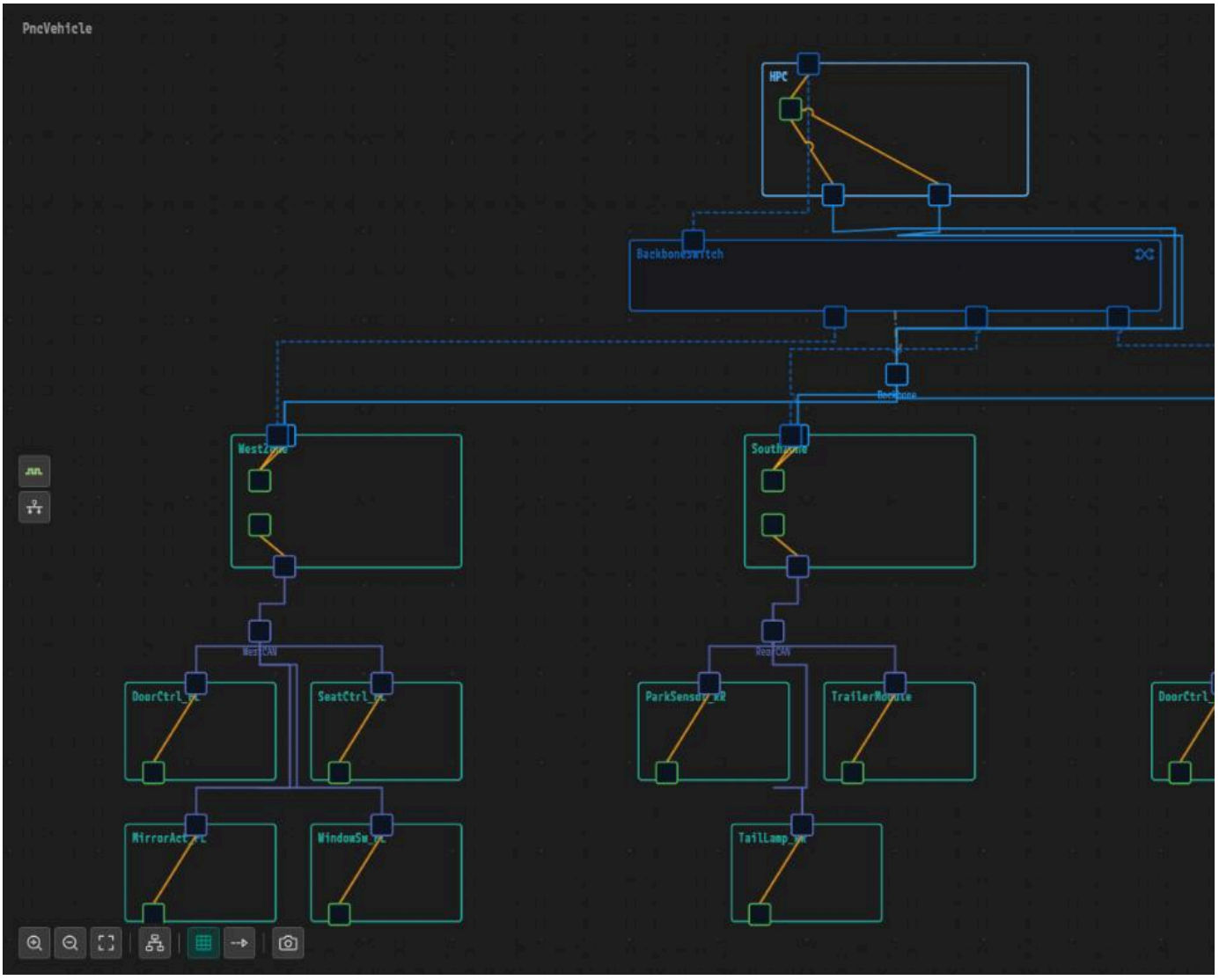


그림 2-1. 완성된 토폴로지. 왼쪽 위의 `PncVehicle`은 완성 예제 프로젝트의 루트 이름이다.

시작하기 전에

- 프로젝트가 열려 있지 않은 창에서 PAIO를 시작한다. **EDITOR** 뷰에 `No project open`이 보이면 된다. 전에 작업한 프로젝트가 열리면 **PROJECT** > **New Window**를 고른다.
- System 편집기(이전 이름 IVC 편집기)의 전체 기능은 IVC Editor에 있다.

다음 요소는 절차를 따라 만든다. 나머지는 나머지 노드와 채널 추가하기의 표를 따른다.

요소	이름	비고
HPC	`HPC`	Ethernet 컨트롤러 `HPC_Eth`, 커넥터 `HPC_Eth_Body_Conn`
Zone ECU	`WestZone`	CAN FD 컨트롤러 `WestZone_CanFd`, Ethernet 컨트롤러 `WestZone_Eth`
말단 ECU	`DoorCtrl_FL`	CAN FD 컨트롤러 `DoorCtrl_FL_CanFd`
CAN 클러스터	`WestCAN`	500 kbps, 채널 `WestCAN_Channel`
Ethernet 클러스터	`Backbone`	채널 `VLAN_Body`(VLAN ID 100), `VLAN_Rear`(VLAN ID 200)
이더넷 스위치	`BackboneSwitch`	HPC 내장, 클러스터 `Backbone`
커플링 포트	`HPC_Eth_Port`, `WestZone_Eth_Port`, `Port_0`, `Port_1`	스위치 포트와 컨트롤러 포트의 물리 연결

새 프로젝트 만들기

1. **PROJECT** > **New Project...**를 누른다.

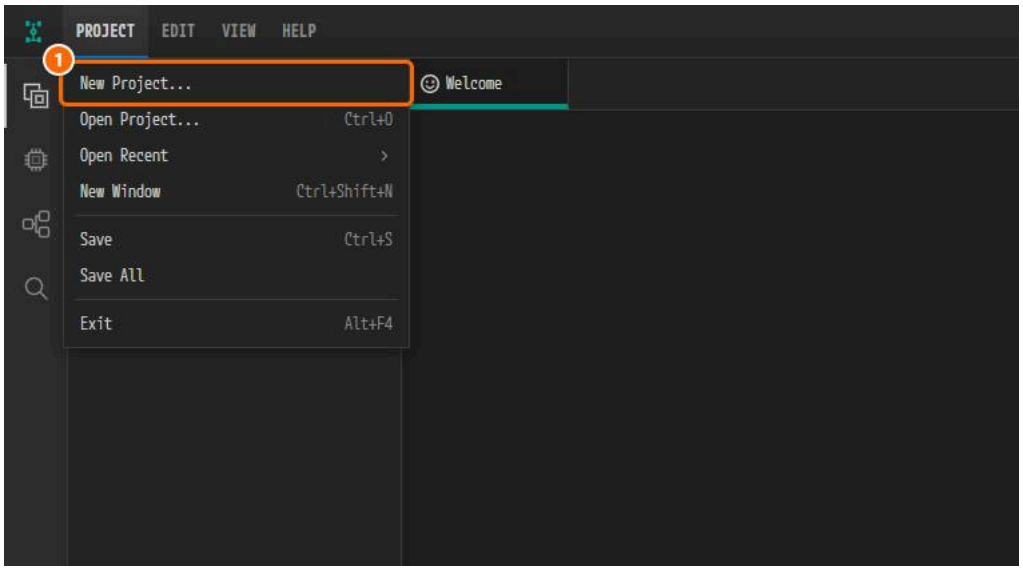


*그림 2-2. ① **New Project...**.*

2. **Name**에 `PNC`를 입력한다. **Location**에 직접 고른 폴더를 입력하거나 **Browse...**로 고른다. **Create**를 누른다.

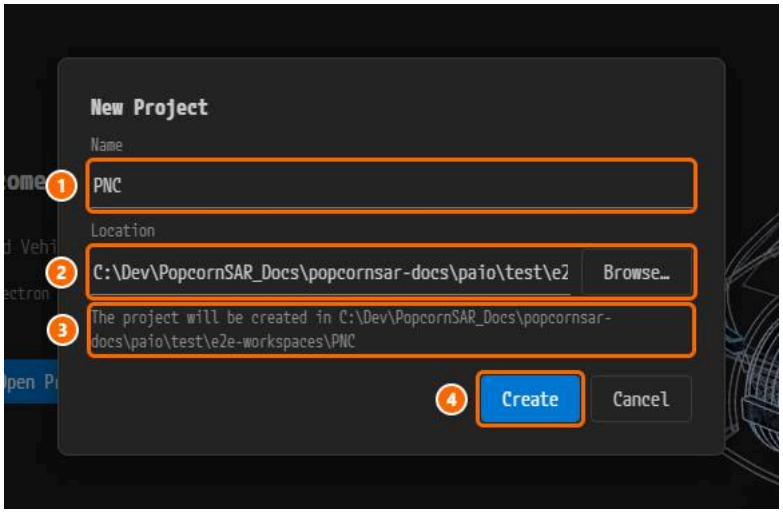


*그림 2-3. ① **Name**, ② **Location**과 **Browse...**, ③ 새 프로젝트 폴더, ④ **Create**. 폴더는 예시일 뿐이다.*

3. 창 제목이 `PAIO [PNC]`인지 확인한다.

System 편집기 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System**을 누른다.

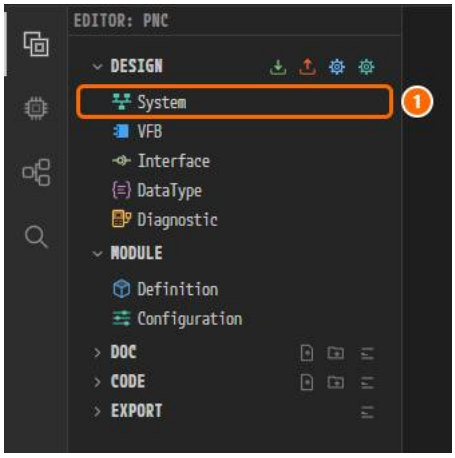


*그림 2-4. ① **System**.*

2. 탭 **PNC:System**이 빈 캔버스로 열린다.

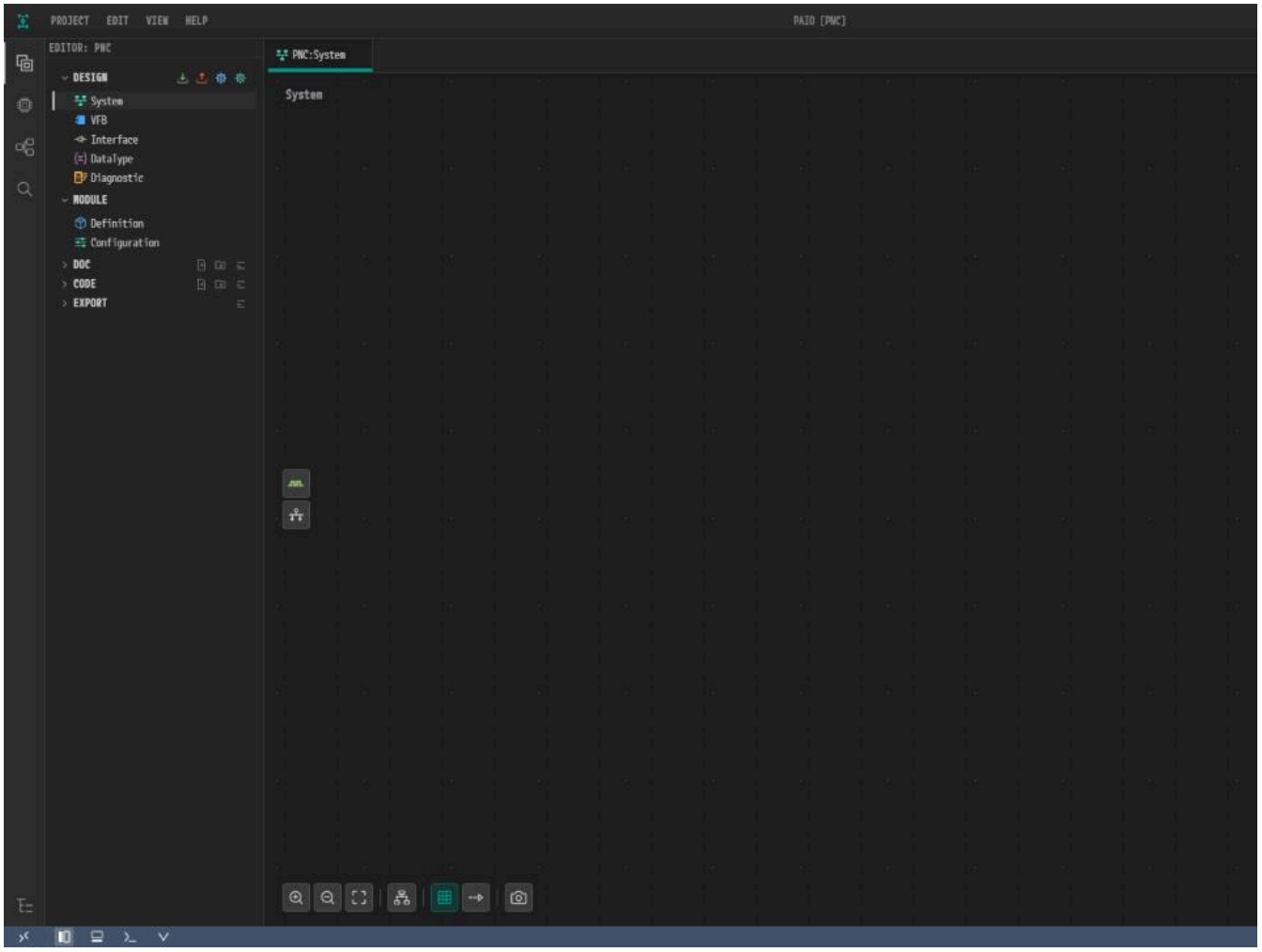


*그림 2-5. 빈 System 편집기. **Auto Layout**이 있는 툴바는 캔버스 왼쪽 아래에 있다. 새 프로젝트의 루트 이름은 `System`이다.*

HPC 추가하기

1. 캔버스의 빈 곳을 우클릭하고 **Add HPC**를 누른다.



*그림 2-6. ① **Add HPC**.*

2. 새 노드 `HPC_0`을 누른다. **NAME**에 `HPC`를 입력하고 `Tab`을 누른다.

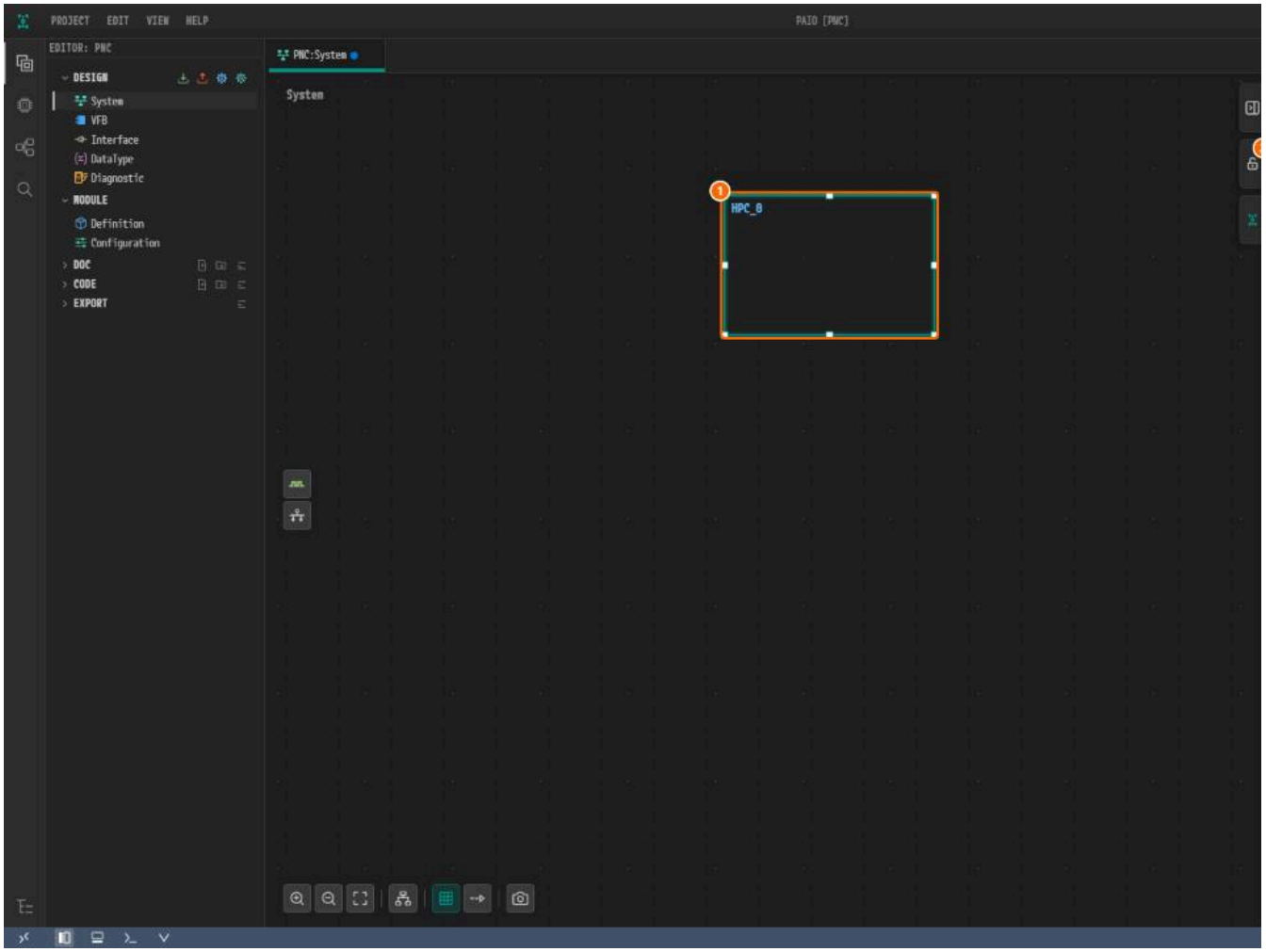


*그림 2-7. ① HPC 노드, ② **NAME**.*

Zone ECU와 말단 ECU 추가하기

1. 캔버스의 빈 곳을 우클릭하고 **Add ECU**를 누른다. 새 노드의 이름을 `WestZone`으로 바꾼다.

2. 같은 방법으로 ECU를 하나 더 만들고 이름을 `DoorCtrl_FL`로 바꾼다.

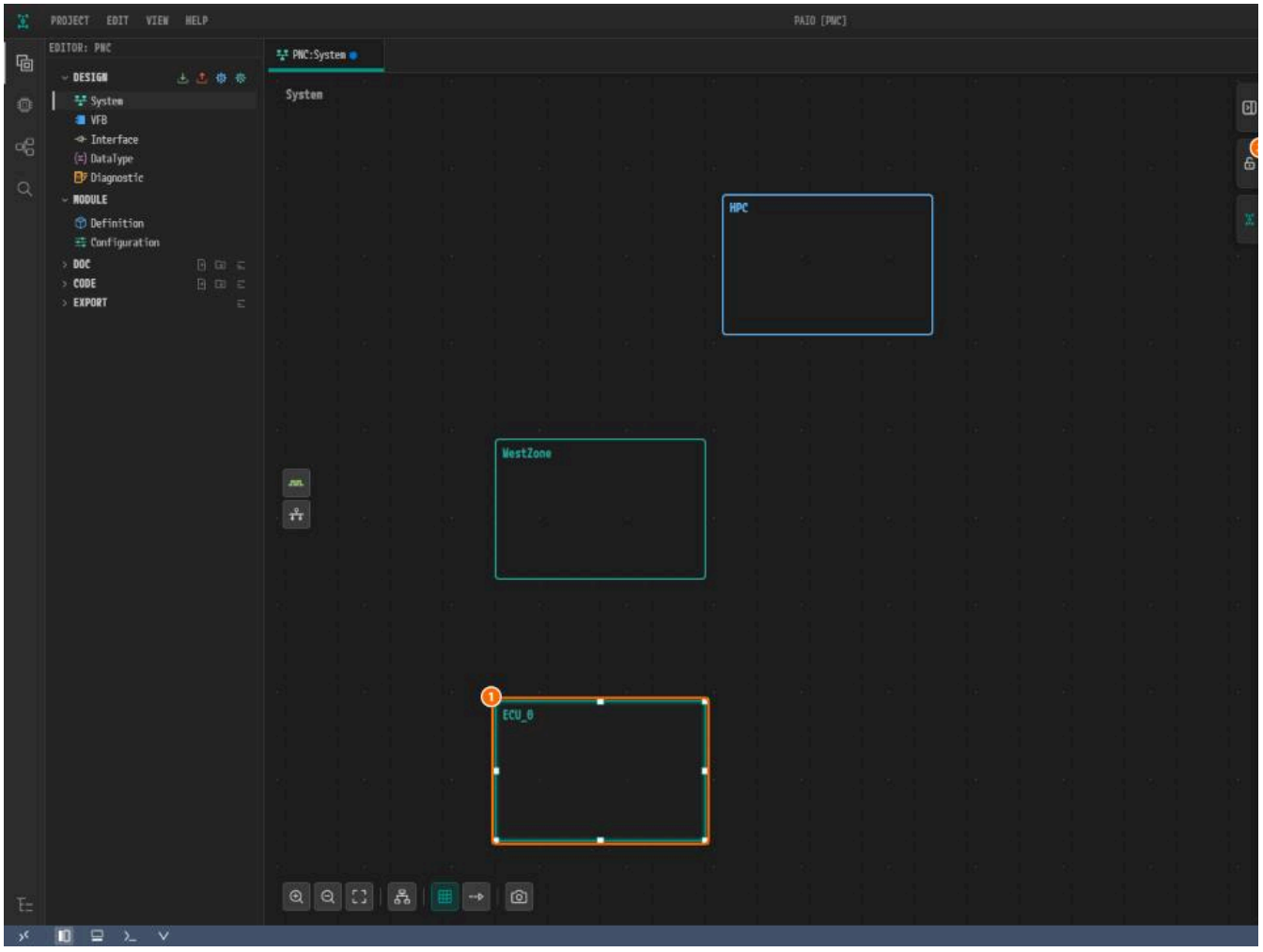


*그림 2-8. ① 새 ECU, ② **NAME**.*

CAN FD 컨트롤러와 커넥터 추가하기

1. `WestZone` 안의 빈 곳을 우클릭하고 **Add Controller** ▸ **CAN**을 고른다.

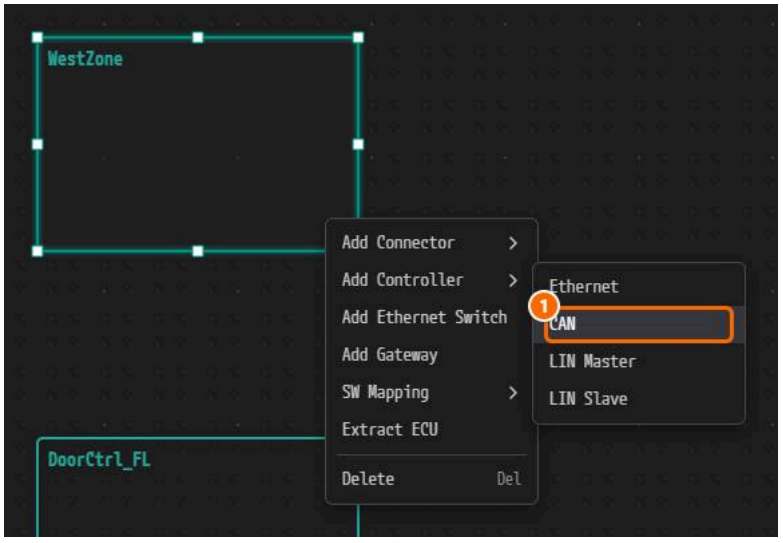


*그림 2-9. ① **CAN**.*

> **주의.** 노드의 머리글과 가장자리의 커넥터를 피해 노드 안쪽을 우클릭한다. 그렇지 않으면 그 요소의 메뉴가 열린다.

2. 새 컨트롤러(초록 사각형)를 누른다. **NAME**을 `WestZone_CanFd`로 바꾸고 **CAN CONTROLLER CONFIGURATION** 머리글의 **CAN FD**를 누른다. 비트 타이밍은 비워 둔다.

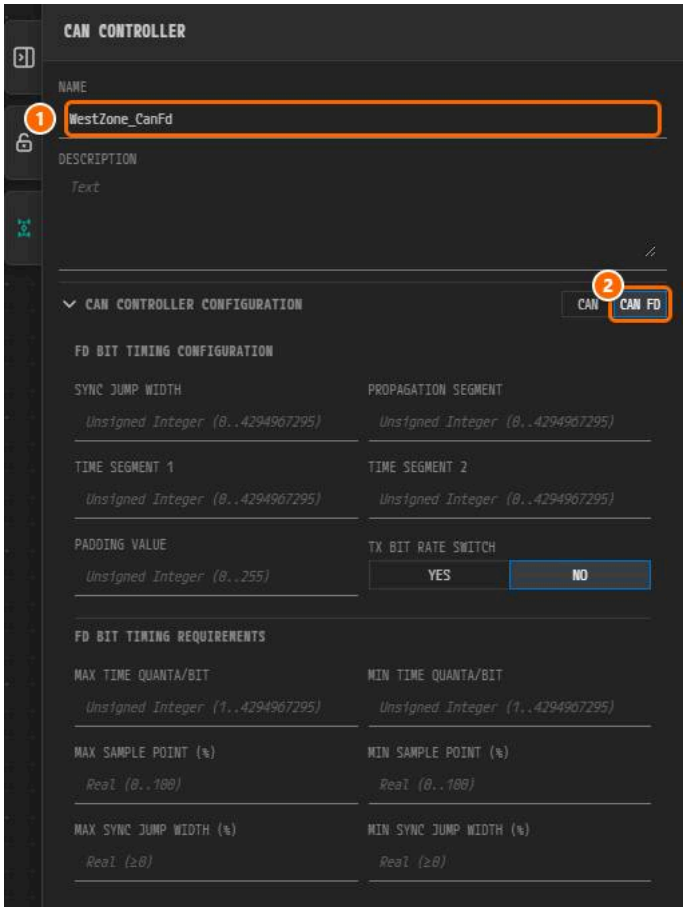


*그림 2-10. ① **NAME**, ② **CAN FD**.*

3. `WestZone` 안을 우클릭하고 **Add Connector ▸ CAN**을 고른다. 노드 가장자리의 새 커넥터를 누른다.

4. **NAME**을 `WestZone_CanFd_Conn`으로 바꾸고 **CONTROLLER**에서 `WestZone_CanFd`를 고른다. 컨트롤러와 커넥터 사이에 주황 선이 생긴다.

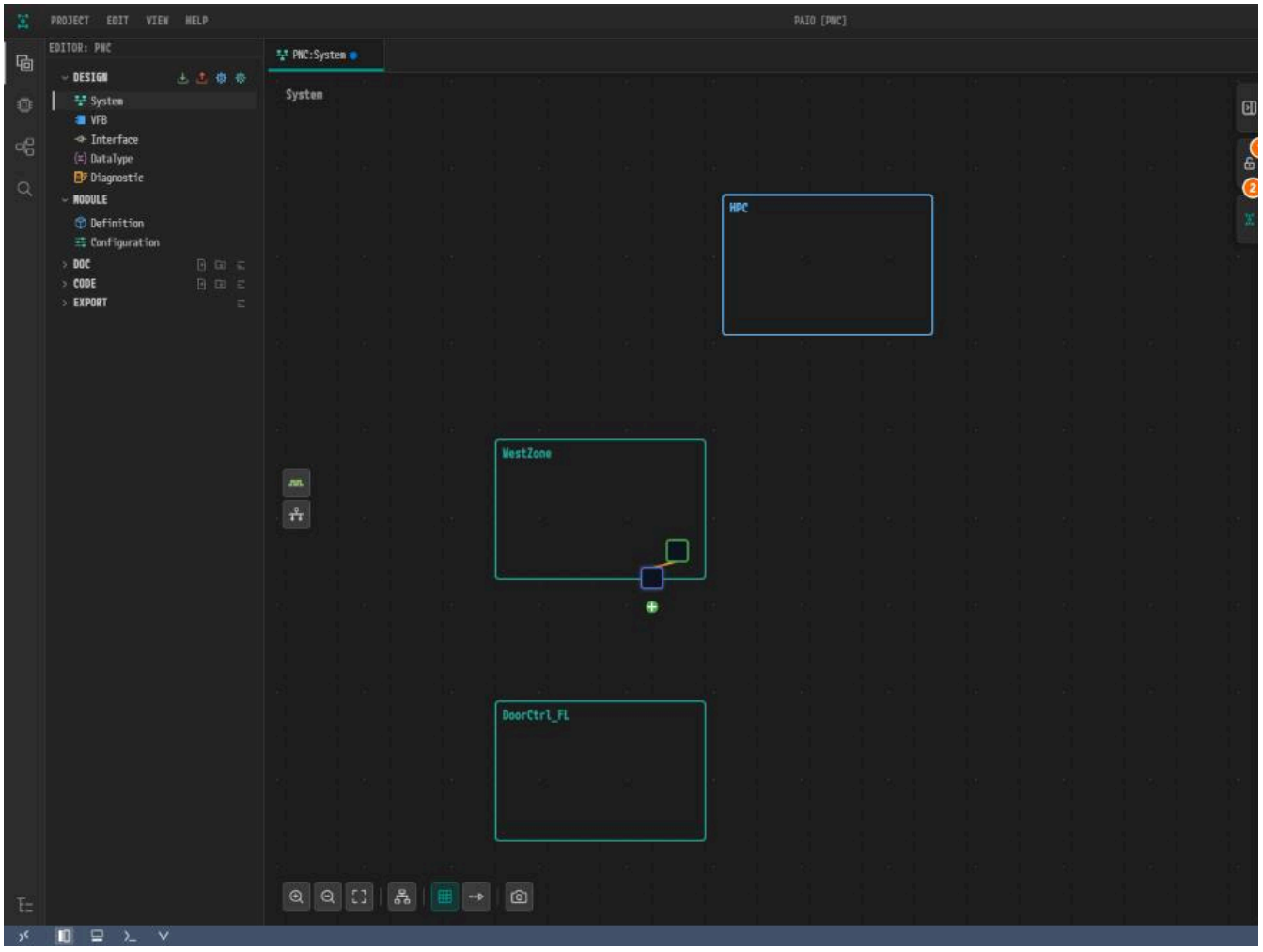


*그림 2-11. ① **NAME**, ② **CONTROLLER**.*

5. 같은 방법으로 `DoorCtrl_FL`에 컨트롤러 `DoorCtrl_FL_CanFd`(**CAN FD**)와 커넥터 `DoorCtrl_FL_CAN_Conn`을 만든다.

CAN 클러스터 WestCAN 만들기

1. `WestZone_CanFd_Conn` 아래의 초록 **[+]** 배지를 누르고 **Create CommunicationCluster [Can]**을 누른다.

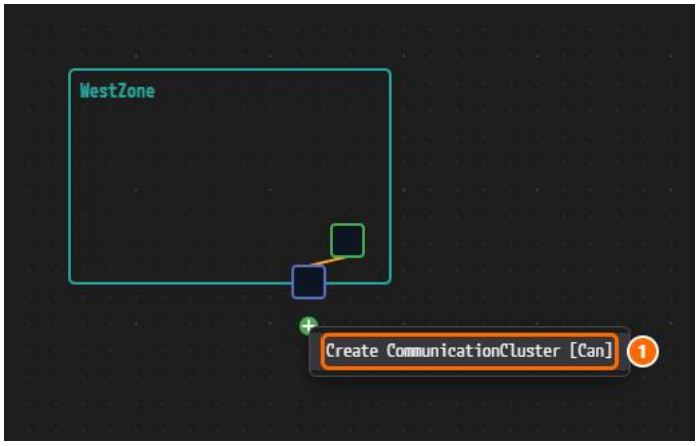


*그림 2-12. ① **Create CommunicationCluster [Can]**.*

2. 새 클러스터(보라 사각형)를 누른다. **NAME**을 `WestCAN`으로 바꾸고, **BAUDRATE (ARBITRATION, KBPS)**에서 `500`을 누르고, **CHANNEL NAME**을 `WestCAN_Channel`

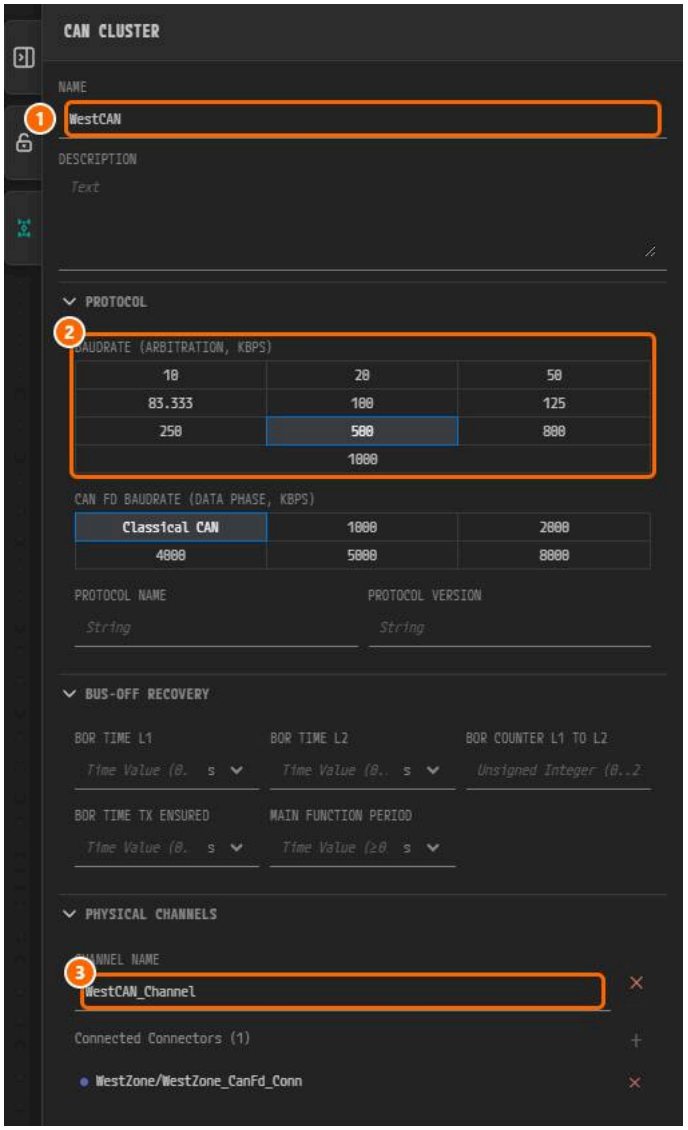


*그림 2-13. ① **NAME**, ② `500`, ③ **CHANNEL NAME**. **CAN FD BAUDRATE (DATA PHASE, KBPS)*는 ***Classical CAN*** *그대로 둔다.*

3. `DoorCtrl_FL_CAN_Conn` 아래의 **[+]** 배지를 누르고 **Connect to Channel...** ▸ **WestCAN** ▸ **WestCAN_Channel**을 고른다.

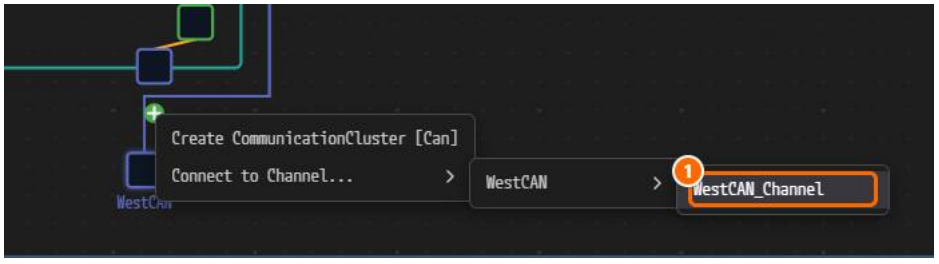


그림 2-14. ① `WestCAN_Channel`.

Ethernet 클러스터 Backbone과 VLAN 채널 만들기

1. `HPC` 안에 **Add Controller** ▸ **Ethernet**으로 Ethernet 컨트롤러를 추가하고 이름을 `HPC_Eth`로 짓는다.
2. `HPC` 안에 **Add Connector** ▸ **Ethernet**으로 커넥터를 추가한다. 이름을 `HPC_Eth_Body_Conn`으로 짓고 **CONTROLLER**를 `HPC_Eth`로 지정한다.
3. `HPC_Eth_Body_Conn`의 **[+]** 배지를 누르고 **Create CommunicationCluster [Ethernet]**을 고른다.
4. 새 클러스터를 누르고 **NAME**을 `Backbone`으로 바꾼다. **PHYSICAL CHANNELS**의 첫 채널에서 **CHANNEL NAME**을 `VLAN_Body`로, **VLAN ID**를 `100`으로 정한다.
5. **PHYSICAL CHANNELS** 오른쪽의 **+**를 누른다. 둘째 채널을 `VLAN_Rear`, **VLAN ID** `200`으로 정한다.

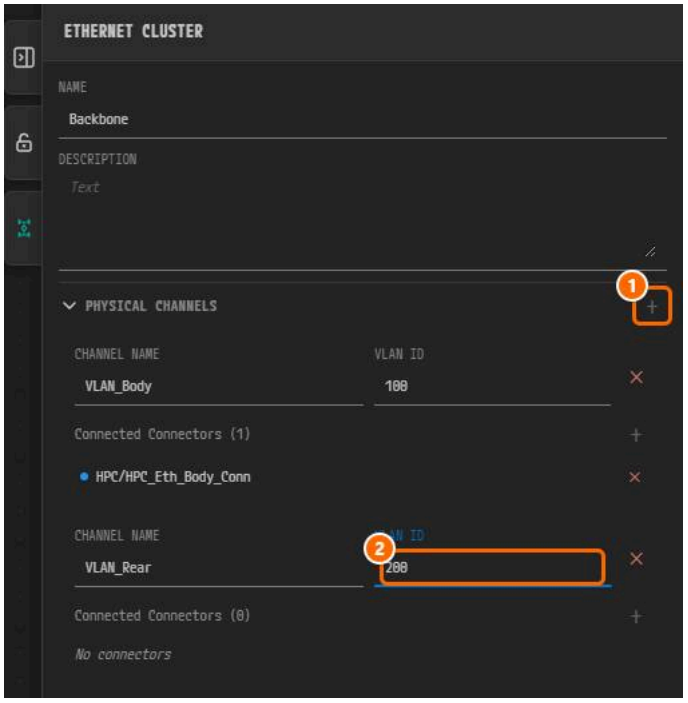


*그림 2-15. ① +(Add Channel), ② 둘째 채널의 **VLAN ID**.*

6. `WestZone` 안에 Ethernet 컨트롤러 `WestZone_Eth`와, **CONTROLLER**가 `WestZone_Eth`인 Ethernet 커넥터 `WestZone_Eth_Conn`을 추가한다.

7. `WestZone_Eth_Conn`의 **[+]** 배지를 누르고 **Connect to Channel... ▸ Backbone ▸ VLAN_Body (VLAN: 100)**을 고른다.

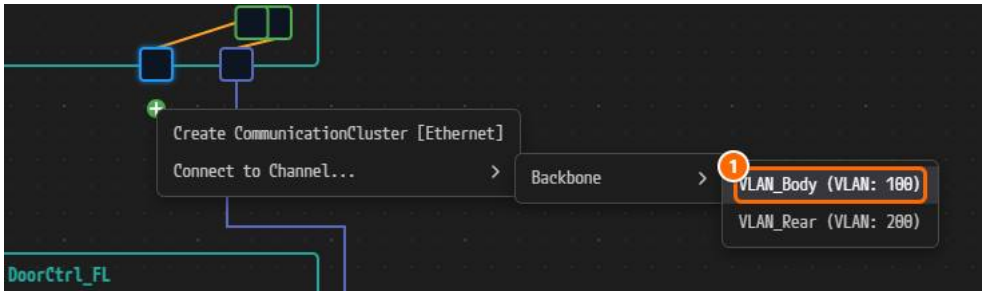


*그림 2-16. ① **VLAN_Body (VLAN: 100)**.*

이더넷 스위치 추가하기

1. `HPC` 안의 빈 곳을 우클릭하고 **Add Ethernet Switch**를 누른다.

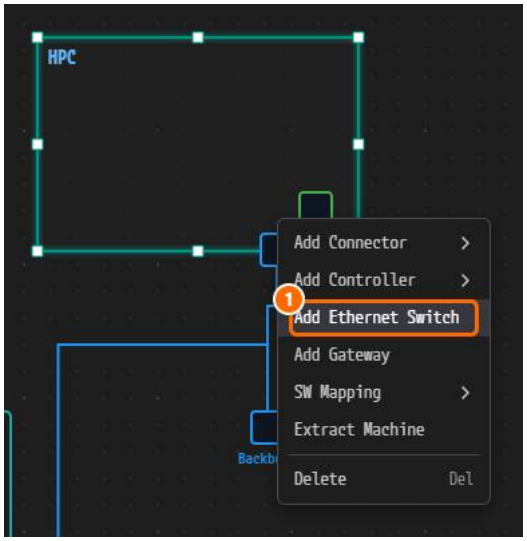


*그림 2-17. ① **Add Ethernet Switch**.*

2. 새 스위치 `EthSwitch_0`을 누른다. **NAME**을 `BackboneSwitch`로 바꾸고 **COMMUNICATION CLUSTER**에서 `Backbone`을 고른다. **COUPLING TYPE**은 **SWITCH** 그대로 둔

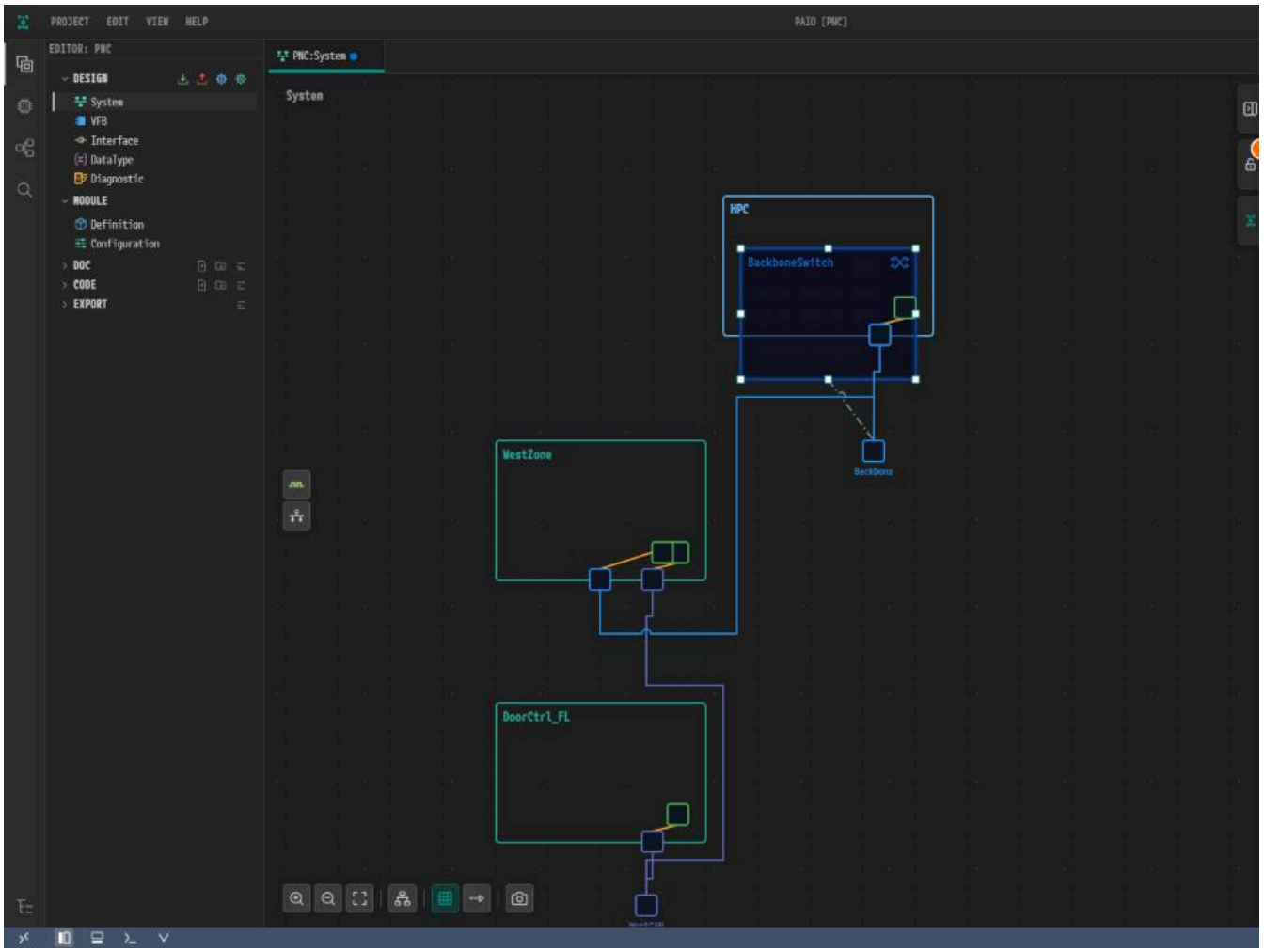


*그림 2-18. ① **NAME**, ② **COMMUNICATION CLUSTER**.*

새 스위치는 **Auto Layout**을 실행하기 전까지 HPC 경계에 겹쳐 그려진다.

자동 배치와 저장

1. 캔버스 왼쪽 아래 툴바의 **Auto Layout**을 누른다.

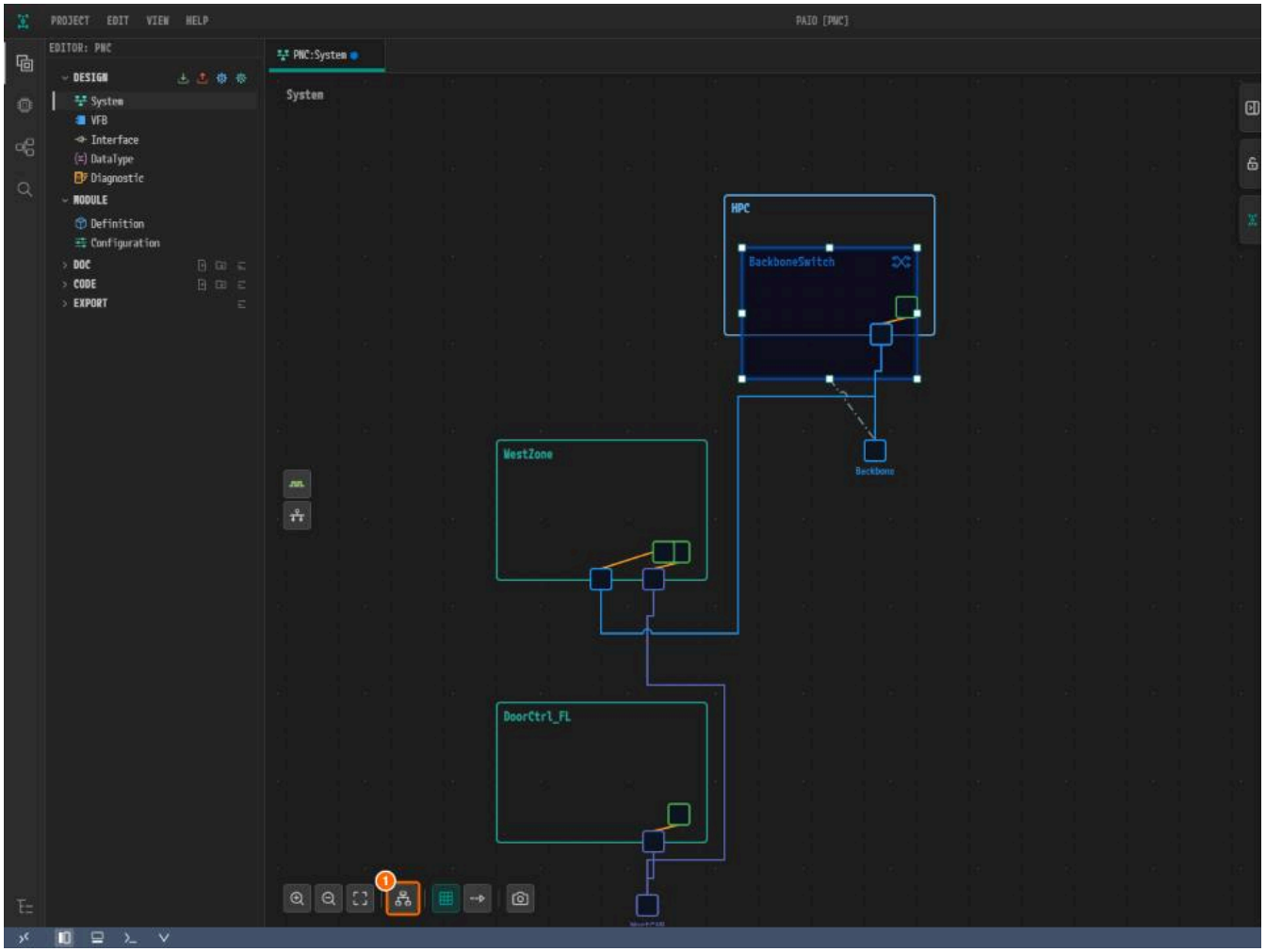


*그림 2-19. ① **Auto Layout**.*

2. `Ctrl+S`로 저장한다.

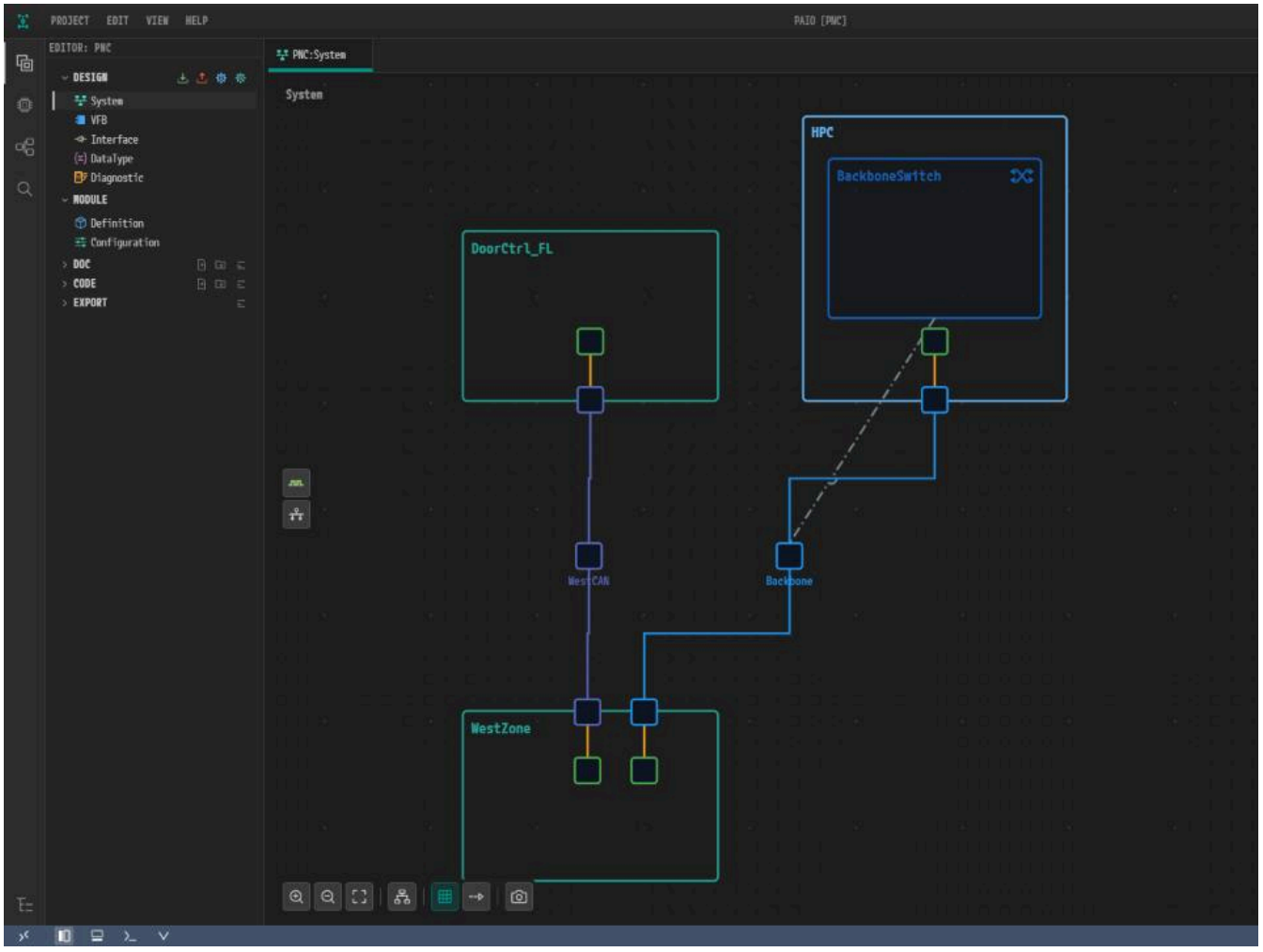


*그림 2-20. **Auto Layout** 후. `BackboneSwitch`가 HPC 안에 들어갔다.*

Auto Layout은 노드를 위아래 계층으로 정렬하지 않으므로 `DoorCtrl_FL`이 `WestZone`보다 위에 놓일 수 있다. 원하면 노드를 끌어 옮긴다.

스위치 포트 만들고 연결하기

커플링 포트는 스위치 포트와 Ethernet 컨트롤러를 잇는다. **Connect to Port...**에는 같은 Ethernet 클러스터에서 아직 연결되지 않은 포트만 나오므로, 커넥터를 먼저 채널에

1. `HPC` 안의 컨트롤러 `HPC_Eth`를 우클릭하고 **Add Coupling Port**를 누른다.

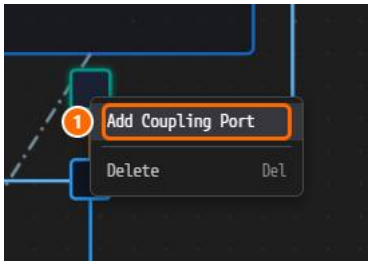


*그림 2-21. ① **Add Coupling Port**.*

2. 노드 가장자리의 새 포트를 누르고 **NAME**을 `HPC_Eth_Port`로 바꾼다.
3. 같은 방법으로 `WestZone_Eth`에 포트 `WestZone_Eth_Port`를 만든다.
4. `BackboneSwitch`를 우클릭하고 **Add CouplingPort**를 누른다.

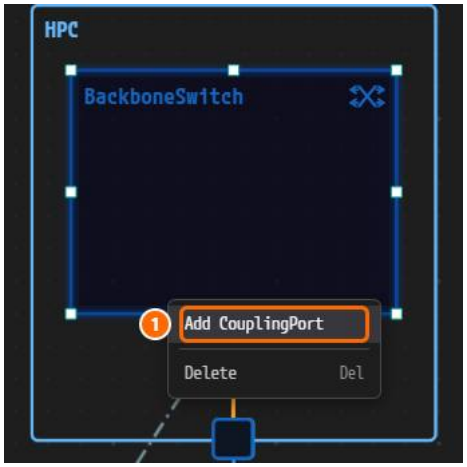


*그림 2-22. ① **Add CouplingPort**.*

5. 새 포트를 누른다. **NAME**을 `Port_0`으로 바꾸고, **PORT ROLE**은 **HOST-PORT**, **PHYSICAL LAYER TYPE**은 **1000BASE-T1**, **NEGOTIATION**은 **AUTO**로 정한다. **NEGOTIATION**

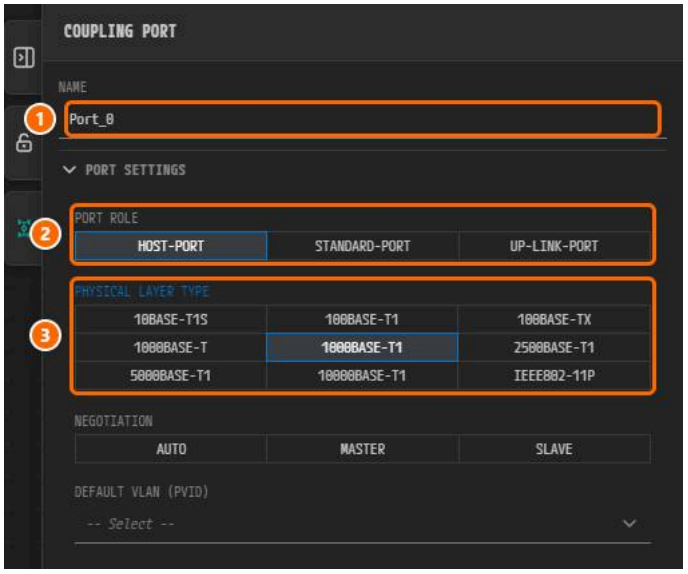


*그림 2-23. ① **NAME**, ② **PORT ROLE**, ③ **PHYSICAL LAYER TYPE**. **NEGOTIATION**은 ③ 아래에 있다.*

6. `BackboneSwitch`의 다른 지점을 우클릭해 같은 방법으로 `Port_1`을 만들고, **PORT ROLE**은 **STANDARD-PORT**로 정한다.

> **주의.** 새 포트는 우클릭한 지점에 가장 가까운 경계에 붙는다. 같은 지점에 만든 두 포트는 겹쳐서 먼저 만든 포트를 누를 수 없게 된다.

7. `Port_0`을 우클릭하고 **Connect to Port...** ▸ **HPC_Eth / HPC_Eth_Port**를 고른다. 같은 방법으로 `Port_1`을 **WestZone_Eth / WestZone_Eth_Port**에 연결한다. 연결



*그림 2-24. ① **WestZone_Eth / WestZone_Eth_Port**.*

8. `Ctrl+S`로 저장한다.

나머지 노드와 채널 추가하기

같은 방법으로 나머지 요소를 추가한다. 이름은 표대로 쓴다.

- 노드: 캔버스에서 **Add ECU**, 이어서 **NAME**.

- 컨트롤러와 커넥터: 노드 안에서 **Add Controller**와 **Add Connector**, 이어서 **NAME**과 커넥터의 **CONTROLLER**. CAN 컨트롤러는 모두 **CAN FD**로 켠다. `HPC_Eth_Rear_Co` `HPC_Eth`이다.
- 클러스터: 첫 커넥터의 **[+]** → **Create CommunicationCluster [Can]**, 이어서 **NAME**, `500`, **CHANNEL NAME**.
- 채널 연결: 커넥터의 **[+]** → **Connect to Channel...** ▸ **클러스터** ▸ **채널**.

노드 표

노드	종류	컨트롤러	커넥터	연결할 채널
`HPC`	HPC	`HPC_Eth` (Ethernet)	`HPC_Eth_Body_Conn`, `HPC_Eth_Rear_Conn`	`VLAN_Body`, `VLAN_Rear`
`WestZone`	ECU (Zone)	`WestZone_Eth` (Ethernet), `WestZone_CanFd` (CAN FD)	`WestZone_Eth_Conn`, `WestZone_CanFd_Conn`	`VLAN_Body`, `WestCAN_Chann`
`SouthZone`	ECU (Zone)	`SouthZone_Eth` (Ethernet), `SouthZone_CanFd` (CAN FD)	`SouthZone_Eth_Conn`, `SouthZone_CanFd_Conn`	`VLAN_Rear`, `RearCAN_Chann`
`EastZone`	ECU (Zone)	`EastZone_Eth` (Ethernet), `EastZone_CanFd` (CAN FD)	`EastZone_Eth_Conn`, `EastZone_CanFd_Conn`	`VLAN_Body`, `EastCAN_Chann`
`DoorCtrl_FL`	ECU	`DoorCtrl_FL_CanFd` (CAN FD)	`DoorCtrl_FL_CAN_Conn`	`WestCAN_Channel`
`SeatCtrl_FL`	ECU	`SeatCtrl_FL_CanFd` (CAN FD)	`SeatCtrl_FL_CAN_Conn`	`WestCAN_Channel`
`MirrorAct_FL`	ECU	`MirrorAct_FL_CanFd` (CAN FD)	`MirrorAct_FL_CAN_Conn`	`WestCAN_Channel`
`WindowSw_FL`	ECU	`WindowSw_FL_CanFd` (CAN FD)	`WindowSw_FL_CAN_Conn`	`WestCAN_Channel`
`ParkSensor_RR`	ECU	`ParkSensor_RR_CanFd` (CAN FD)	`ParkSensor_RR_CAN_Conn`	`RearCAN_Channel`
`TrailerModule`	ECU	`TrailerModule_CanFd` (CAN FD)	`TrailerModule_CAN_Conn`	`RearCAN_Channel`
`TailLamp_RR`	ECU	`TailLamp_RR_CanFd` (CAN FD)	`TailLamp_RR_CAN_Conn`	`RearCAN_Channel`
`DoorCtrl_FR`	ECU	`DoorCtrl_FR_CanFd` (CAN FD)	`DoorCtrl_FR_CAN_Conn`	`EastCAN_Channel`
`SeatCtrl_FR`	ECU	`SeatCtrl_FR_CanFd` (CAN FD)	`SeatCtrl_FR_CAN_Conn`	`EastCAN_Channel`
`MirrorAct_FR`	ECU	`MirrorAct_FR_CanFd` (CAN FD)	`MirrorAct_FR_CAN_Conn`	`EastCAN_Channel`

클러스터 표

클러스터	종류	BAUDRATE (ARBITRATION, KBPS)	채널 (VLAN ID)	연결되는 커넥터
`Backbone`	Ethernet	—	`VLAN_Body` (100)	`HPC_Eth_Body_Conn`, `WestZone_Eth_Conn`, `EastZone_Eth_Conn`
			`VLAN_Rear` (200)	`HPC_Eth_Rear_Conn`, `SouthZone_Eth_Conn`
`WestCAN`	CAN	500	`WestCAN_Channel`	`WestZone_CanFd_Conn`, `DoorCtrl_FL_CAN_Conn`, `SeatCtrl_FL_CAN_Conn`, `MirrorAct_FL`
`RearCAN`	CAN	500	`RearCAN_Channel`	`SouthZone_CanFd_Conn`, `ParkSensor_RR_CAN_Conn`, `TrailerModule_CAN_Conn`, `TailLam`
`EastCAN`	CAN	500	`EastCAN_Channel`	`EastZone_CanFd_Conn`, `DoorCtrl_FR_CAN_Conn`, `SeatCtrl_FR_CAN_Conn`, `MirrorAct_FR`

스위치 포트 표

모든 포트의 **PHYSICAL LAYER TYPE**은 **1000BASE-T1**, **NEGOTIATION**은 **AUTO**이다.

포트	역할	연결 대상	VLAN
`Port_0`	HOST	`HPC/HPC_Eth_Port`	`VLAN_Body`, `VLAN_Rear`
`Port_1`	STD	`WestZone/WestZone_Eth_Port`	`VLAN_Body`
`Port_2`	STD	`SouthZone/SouthZone_Eth_Port`	`VLAN_Rear`
`Port_3`	STD	`EastZone/EastZone_Eth_Port`	`VLAN_Body`

3장에서 이 포트에 PNC를 연결하므로 모두 만들어 둔다. 자세한 설명은 IVC Editor의 6.4.3절, 6.4.4절, 6.5.4절에 있다.

1. 스위치 포트 만들고 연결하기와 같은 방법으로 `SouthZone_Eth`, `EastZone_Eth`에 `SouthZone_Eth_Port`, `EastZone_Eth_Port`를 만든다.
2. `BackboneSwitch`에 `Port_2`, `Port_3`을 **STANDARD-PORT**로 추가한다. 포트마다 경계의 다른 지점을 쓴다.
3. **Connect to Port...**로 스위치 포트마다 표의 연결 대상에 연결한다.
4. `BackboneSwitch`를 더블클릭해 워크벤치를 연다. 포트마다 **DEFAULT VLAN (PVID)**과 **VLAN MEMBERSHIP**을 표대로 정한다. 기본 VLAN은 `Port_2`만 `VLAN_Rear`

Auto Layout을 실행하고 `Ctrl+S`로 저장한다.

완성 예제 살펴보기

배치와 루트 이름은 그림과 다를 수 있다. 노드, 클러스터, 연결이 같으면 된다.

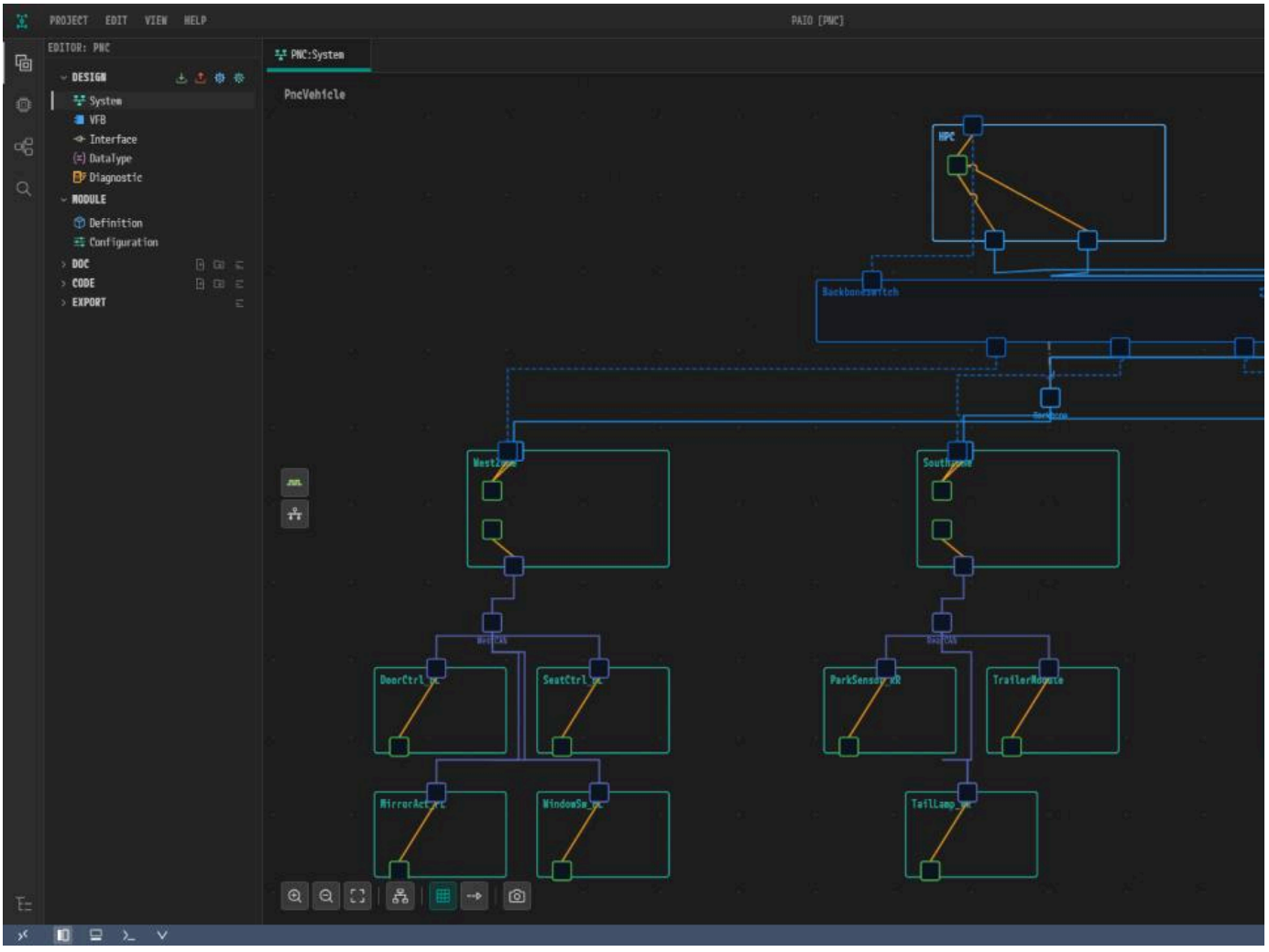


*그림 2-25. **Fit to View** 후의 완성 예제. 가운데 파란 사각형이 `Backbone`, 보라 사각형이 `WestCAN`, `RearCAN`, `EastCAN`이다. 파란 점선은 스위치 포트 연결이다.*

클러스터에 마우스를 올리면 이름, 종류, 전송 속도가 보인다.

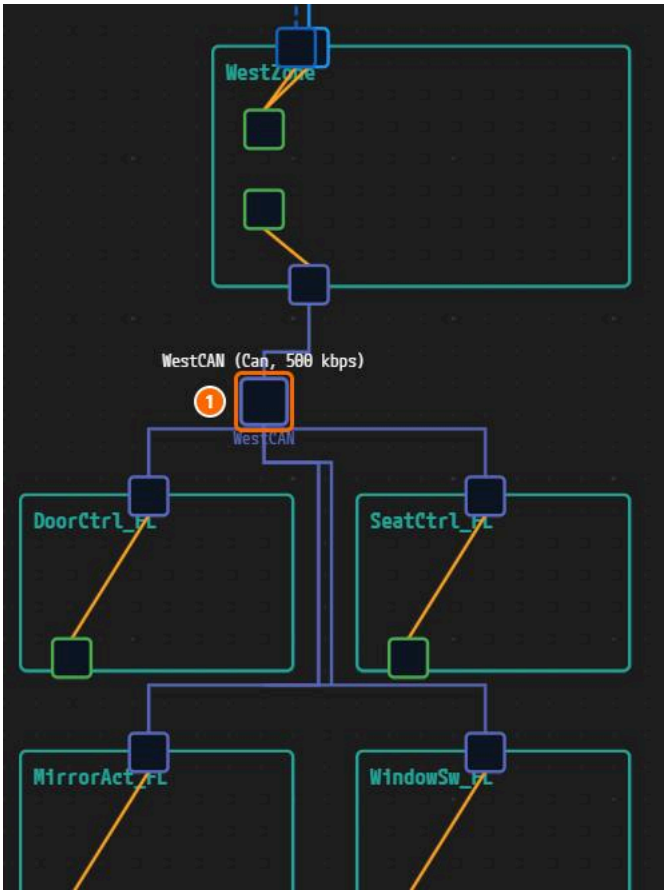


그림 2-26. ① `WestCAN (Can, 500 kbps)`로 표시되는 `WestCAN` 클러스터.

`BackboneSwitch`를 더블클릭해 워크벤치를 확인하고 **Close**를 누른다.

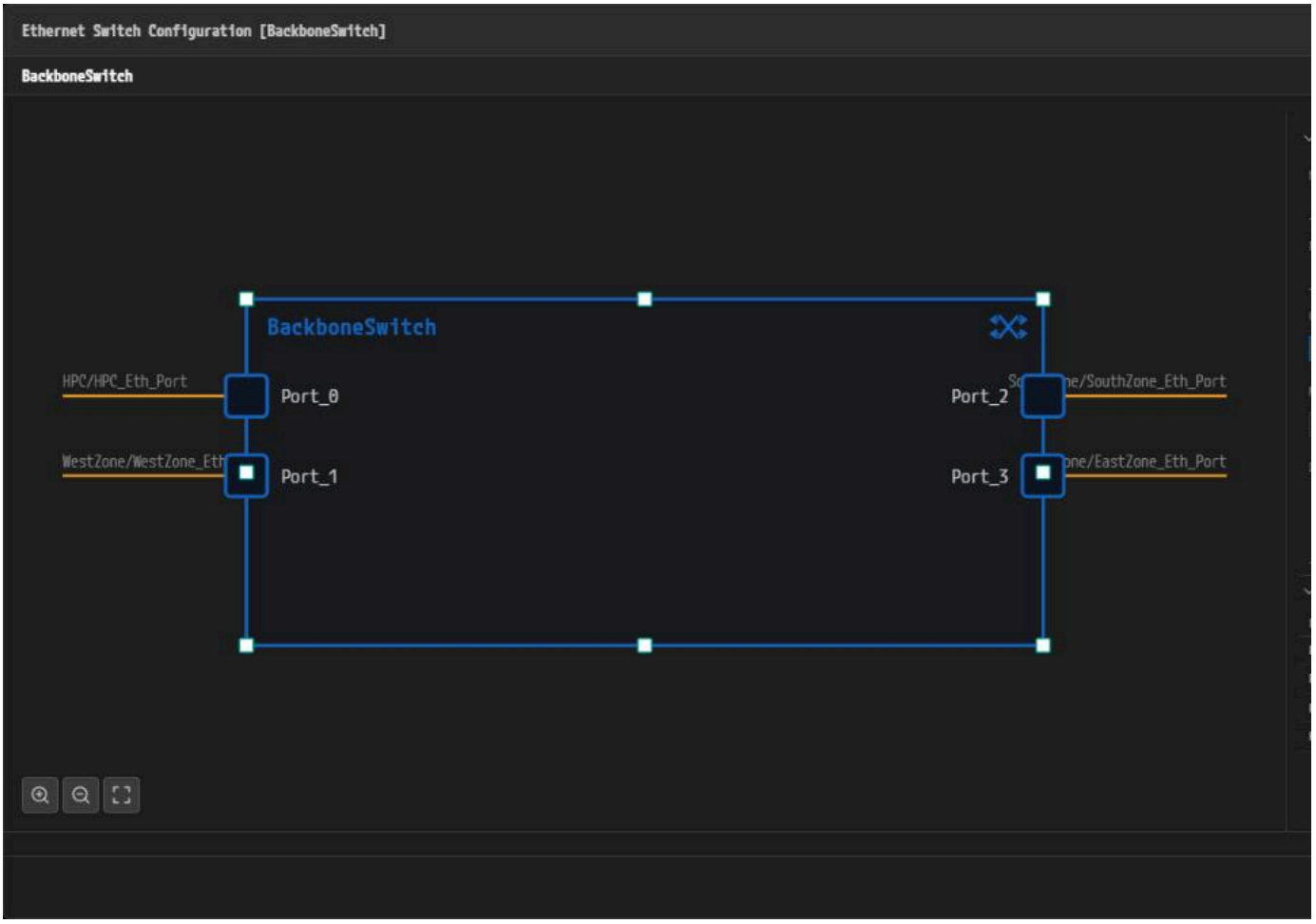


*그림 2-27. **Ethernet Switch Configuration [BackboneSwitch]** 워크벤치. **PORTS (4)* *에서 `Port_0`만 HOST이고 `Port_2`만 VLAN 200이다.*

왜 백본을 VLAN 둘로 나누는가

- `constr_5093`. 한 물리 채널에 PNC 게이트웨이 유형이 active인 커넥터는 최대 1개이다.
- `constr_3781`. PNC마다 최상위 코디네이터가 필요하다. 관련 채널 2개 이상에 연결되고 그 연결이 모두 active인 ECU이다.
- **HPC는 AP 머신**이라 PNC 게이트웨이가 될 수 없다. 백본이 채널 하나라면 세 Zone 중 한 곳만 active가 될 수 있다.
- **VLAN 둘.** `VLAN_Body`(West, East)와 `VLAN_Rear`(South)로 나누면 WestZone이 `PNC_Comfort`를, SouthZone이 `PNC_ParkAssist`와 `PNC_Trailer`를 조정하고 하위 코디네이터가 된다. 게이트웨이는 PNC 정의와 커넥터 PNC 속성에서 입력한다.

확인하기

- [ ] 캔버스에 `HPC`, `WestZone`, `SouthZone`, `EastZone`과 말단 ECU 10개가 있다.
- [ ] 모든 CAN 컨트롤러가 **CAN FD**이고, 모든 CAN 클러스터가 `500`이며 클러스터 표의 커넥터가 연결되어 있다.
- [ ] `Backbone`에 `VLAN_Body`(100)와 `VLAN_Rear`(200)가 있다.
- [ ] `<노드>_Eth_Port`마다 스위치 포트에 연결되어 있고, 모든 스위치 포트의 **NEGOTIATION**이 **AUTO**이다.
- [ ] 워크벤치의 **PORTS (4)** 표가 스위치 포트 표와 같다.
- [ ] 탭에 수정 표시가 없다.

다음 단계

PNC 정의와 커넥터 PNC 속성에서 PNC 3개를 정의하고 이 커넥터들에 PNC 속성을 입력한다.

PNC 정의와 커넥터 PNC 속성, NM 설정

이 장에서 할 일

이 장에서는 PNC 대화상자에서 PNC(Partial Network Cluster) 3개를 정의하고, 잘못된 식별자가 일으키는 검증 오류를 고친다. 이어서 PNC 비트 벡터를 정하고, 커넥터와 노드, 백나르는 NM 값을 채운다.

시작하기 전에

- 토폴로지 설계를 마쳐 모든 노드, 클러스터, 커넥터, 스위치 포트가 있는 상태에서 시작한다.
- **EDITOR** 뷰에서 **DESIGN** 아래의 **System**을 눌러 System 편집기(탭 **PNC:System**)를 연다.
- NM 노드의 **TX-NM-PDU-REFS**와 **RX-NM-PDU-REFS**에는 시그널, PDU, 프레임, I-PDU 그룹의 NM-PDU 18개가 필요하다. 그 장을 마친 뒤 채우고, 나머지는 지금 입력한다.
- 스크린샷은 완성 예제에서 찍었으므로 캔버스의 루트 이름이 `PncVehicle`로 보인다. 독자의 프로젝트에서는 `System`으로 보일 수 있다.

PNC IDENTIFIER	NAME	필터 마스크에서의 위치
`17`	`PNC_Comfort`	Byte 0, bit 1 → `0x02`
`18`	`PNC_ParkAssist`	Byte 0, bit 2 → `0x04`
`19`	`PNC_Trailer`	Byte 0, bit 3 → `0x08`

PNC 대화상자 열기

1. 캔버스 왼쪽 가장자리의 아이콘 줄에서 아래쪽 아이콘 **Partial Network (PNC)**를 누른다.

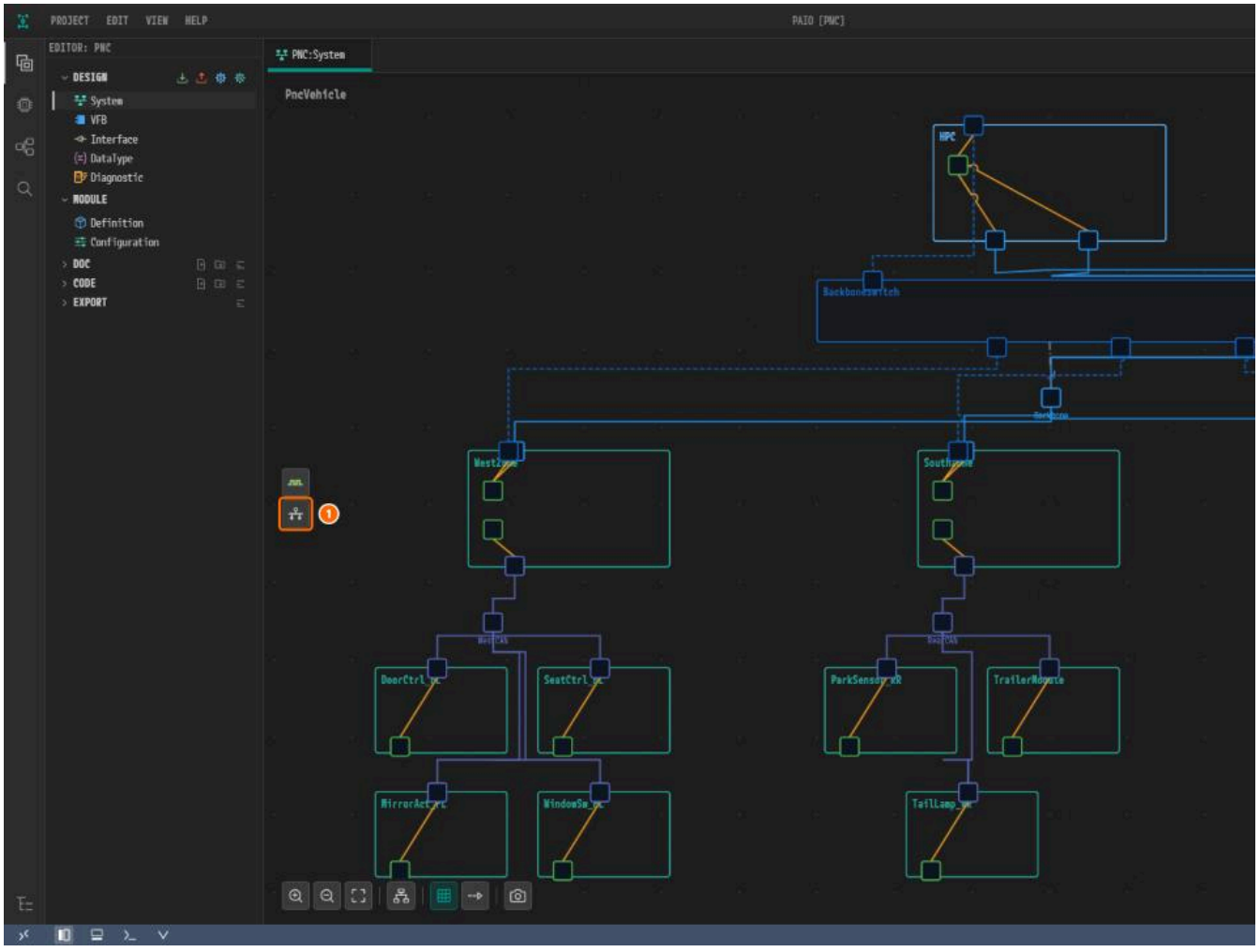


*그림 3-1. ① PNC 버튼. 그 위의 아이콘은 **Signal**이다.*



*그림 3-2. ① **Partial Network (PNC)** 툴팁.*

2. **Partial Network (PNC) Configuration** 대화상자가 빈 **PARTIAL NETWORKS** 목록과 함께 열린다.

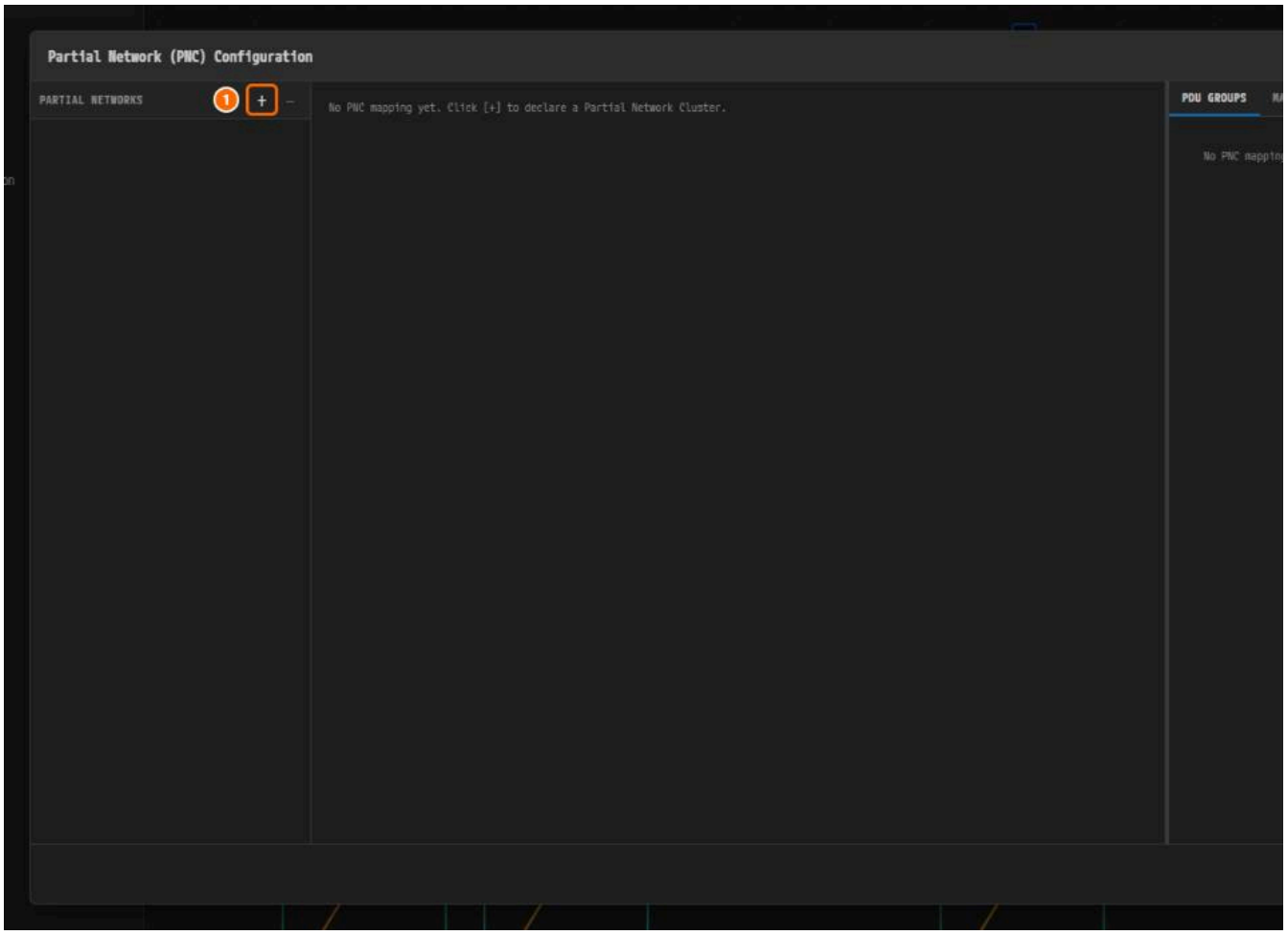


*그림 3-3. ① **[+]** 버튼.*

첫 PNC 추가하기

1. **PARTIAL NETWORKS** 옆의 **[+]**를 누른다. 항목 **PNC 0 Pnc_0**이 나타난다.

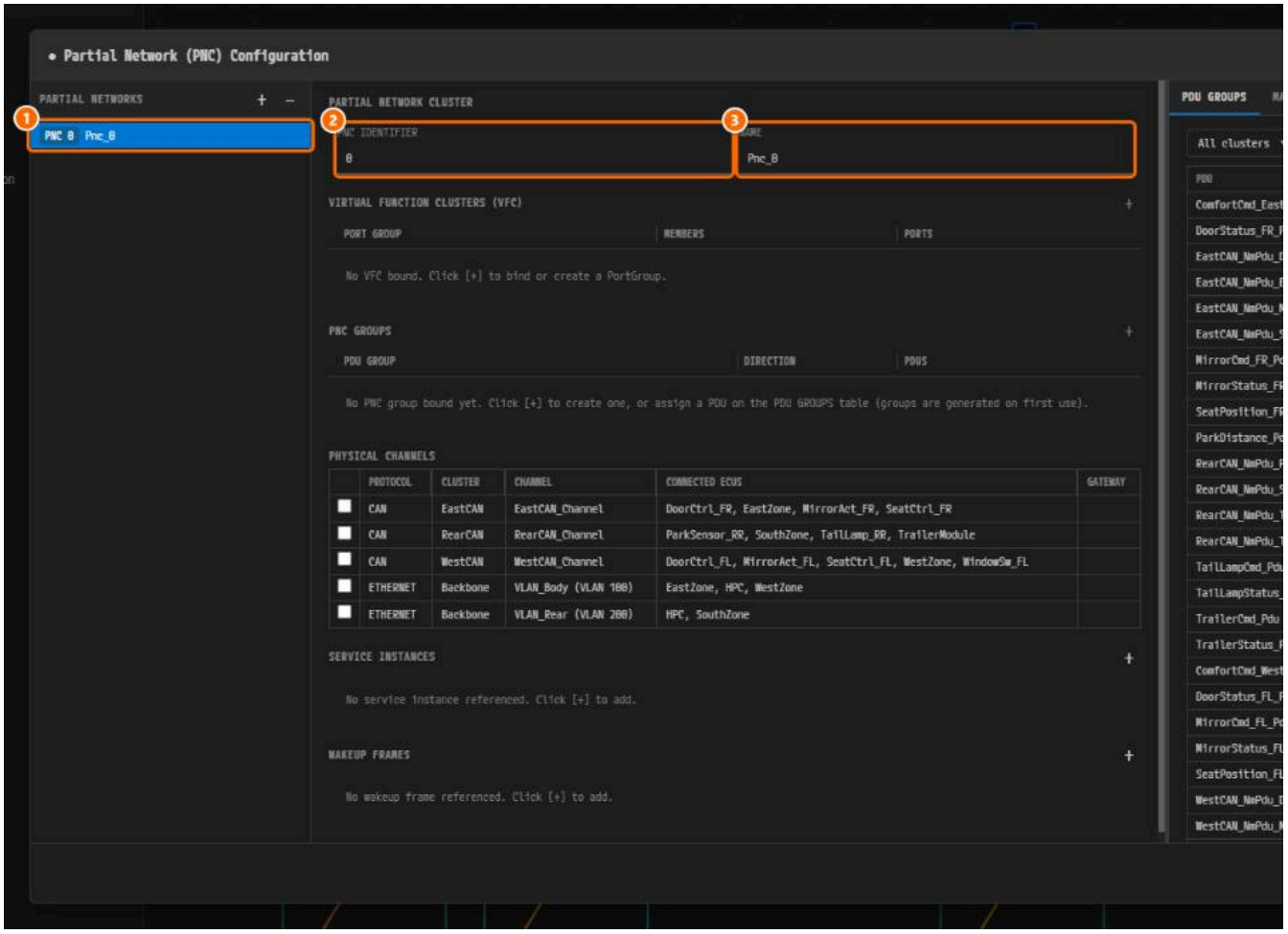


*그림 3-4. ① **PNC 0 Pnc_0**, ② **PNC IDENTIFIER**, ③ **NAME**. 그 아래의 목록은 PNC 다이얼로그 — PNC별 바인딩에서 채운다.*

2. **NAME**에 `PNC_Comfort`를 입력하고 `Enter`를 누른다. **PNC IDENTIFIER**는 일부러 `0` 그대로 둔다.

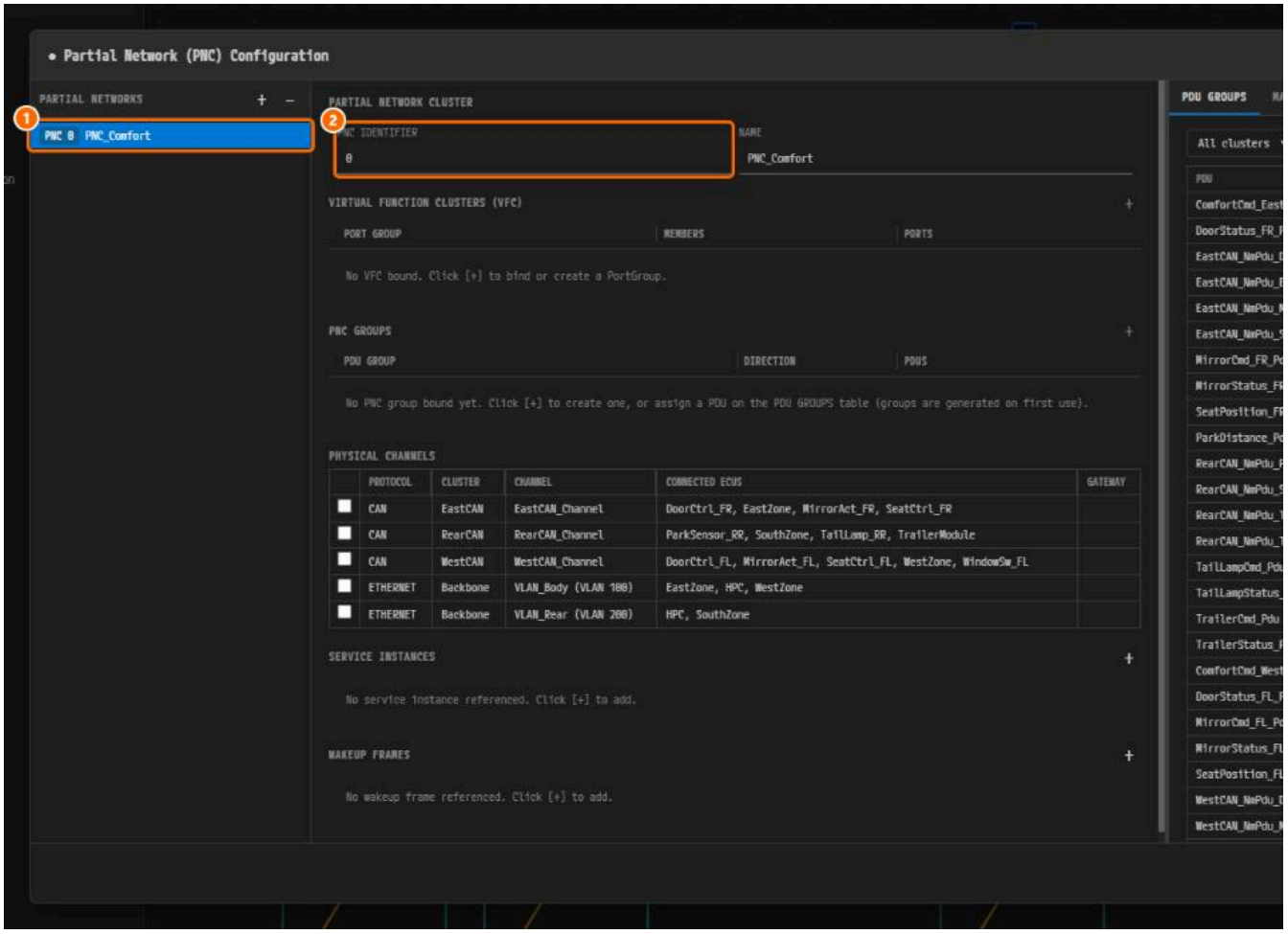


*그림 3-5. ① **PNC 0 PNC_Comfort**, ② `0` 그대로 둔 **PNC IDENTIFIER**.*

3. `Ctrl+S`를 누르고 **Close**를 누른다.

검증 오류 확인하고 고치기

저장하면 프로젝트 전체에 검증이 돈다. 식별자 `0`은 잘못된 값이므로 오류 2건이 나온다.

1. 상태 표시줄의 검증 아이콘(**V**)을 누른다. 아이콘에 빨간 배지 `2`가 붙어 있다. **VALIDATION** 패널이 열린다.

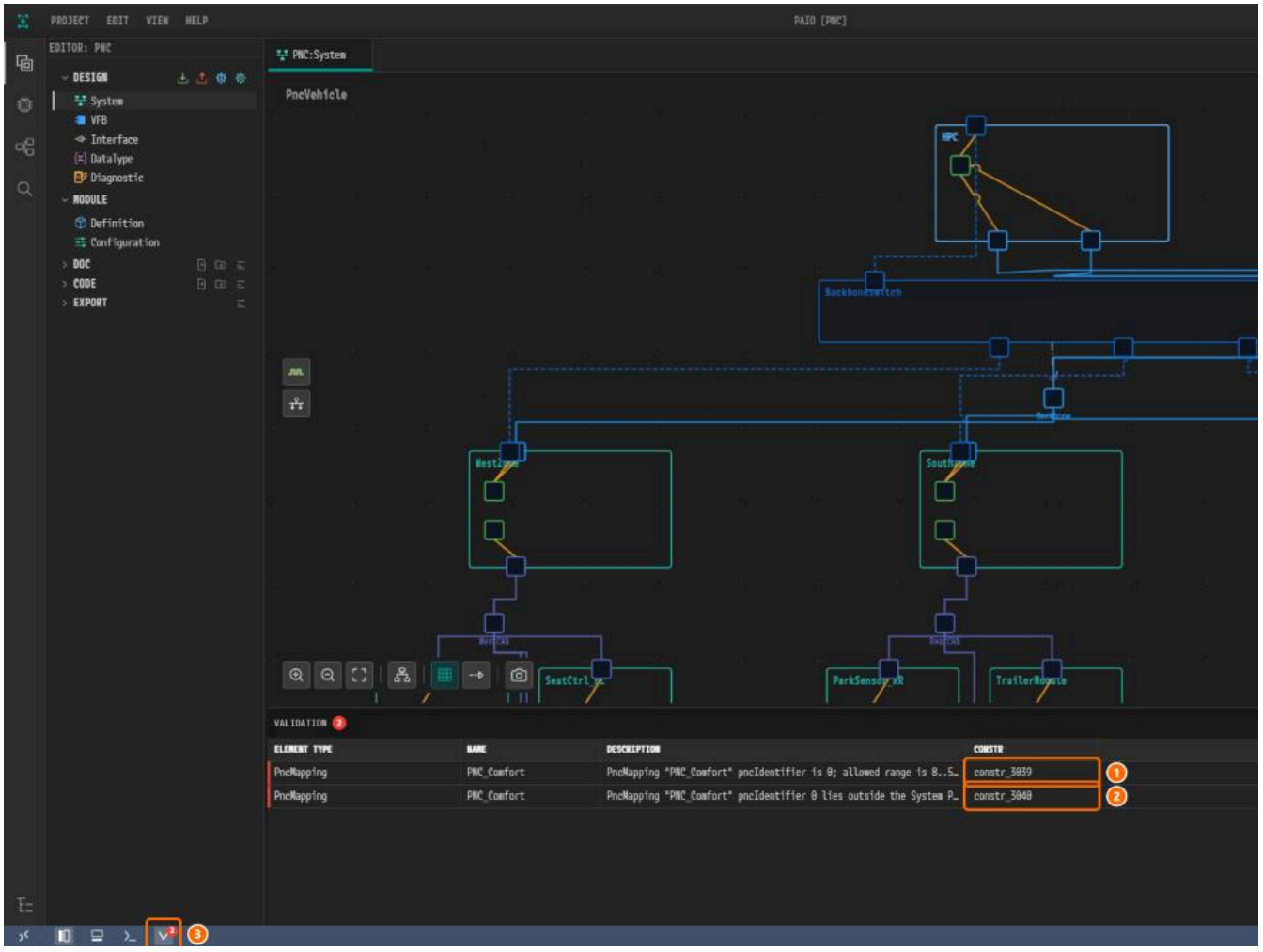


그림 3-6. ① `constr_3039`, ② `constr_3040`, ③ 검증 아이콘.



그림 3-7. ① `constr_3039`: 식별자는 `8`..`511` 안이어야 한다. ② `constr_3040`: PNC 벡터 비트 `16`..`63` 안이어야 한다.

2. `constr_3040` 행을 누른다. 문제 항목이 빨갛게 표시된 채 PNC 대화상자가 열린다.

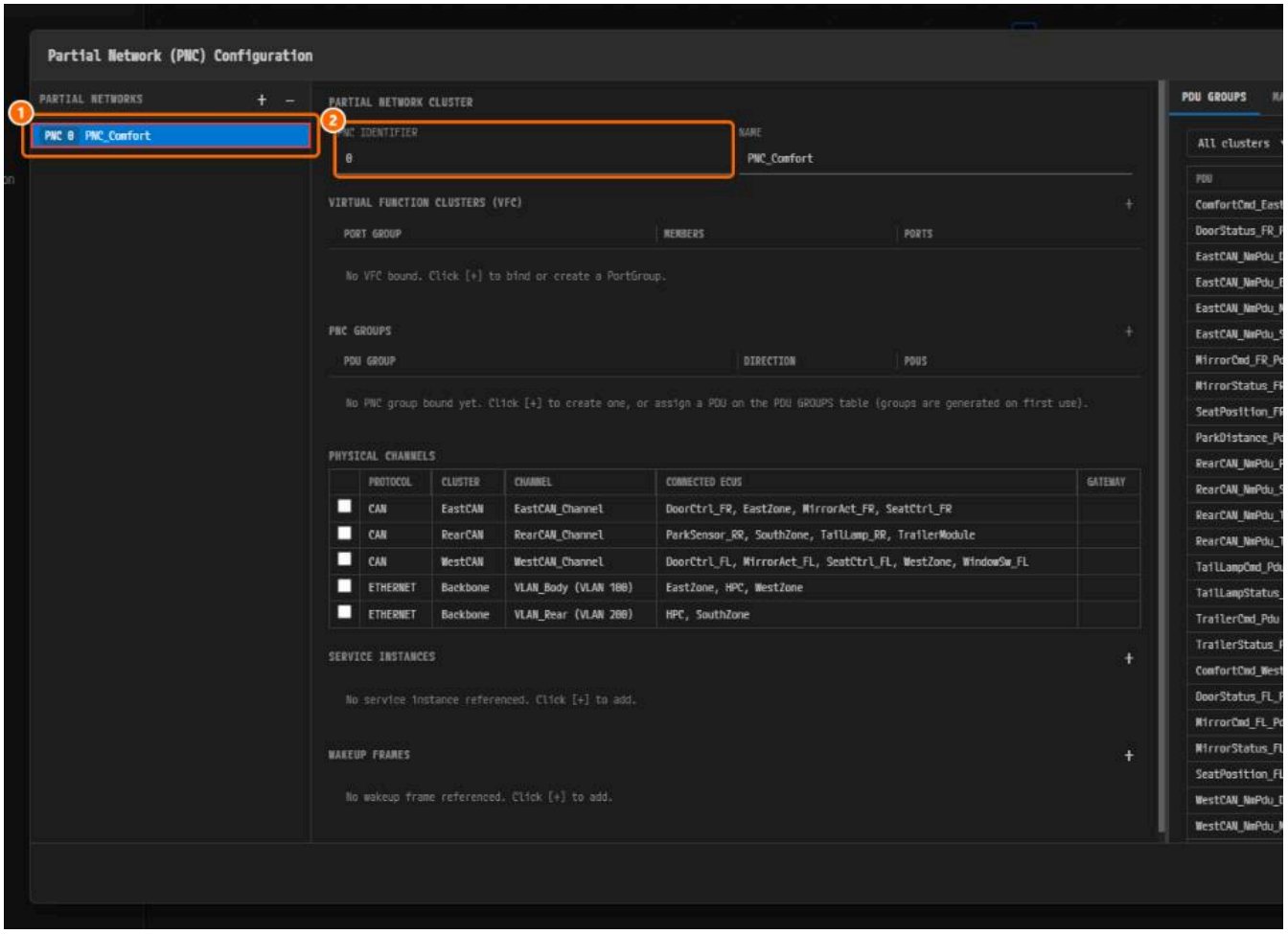


*그림 3-8. ① 문제 PNC 항목, ② **PNC IDENTIFIER**.*

3. **PNC IDENTIFIER**에 `17`을 입력하고 `Enter`를 누른다.

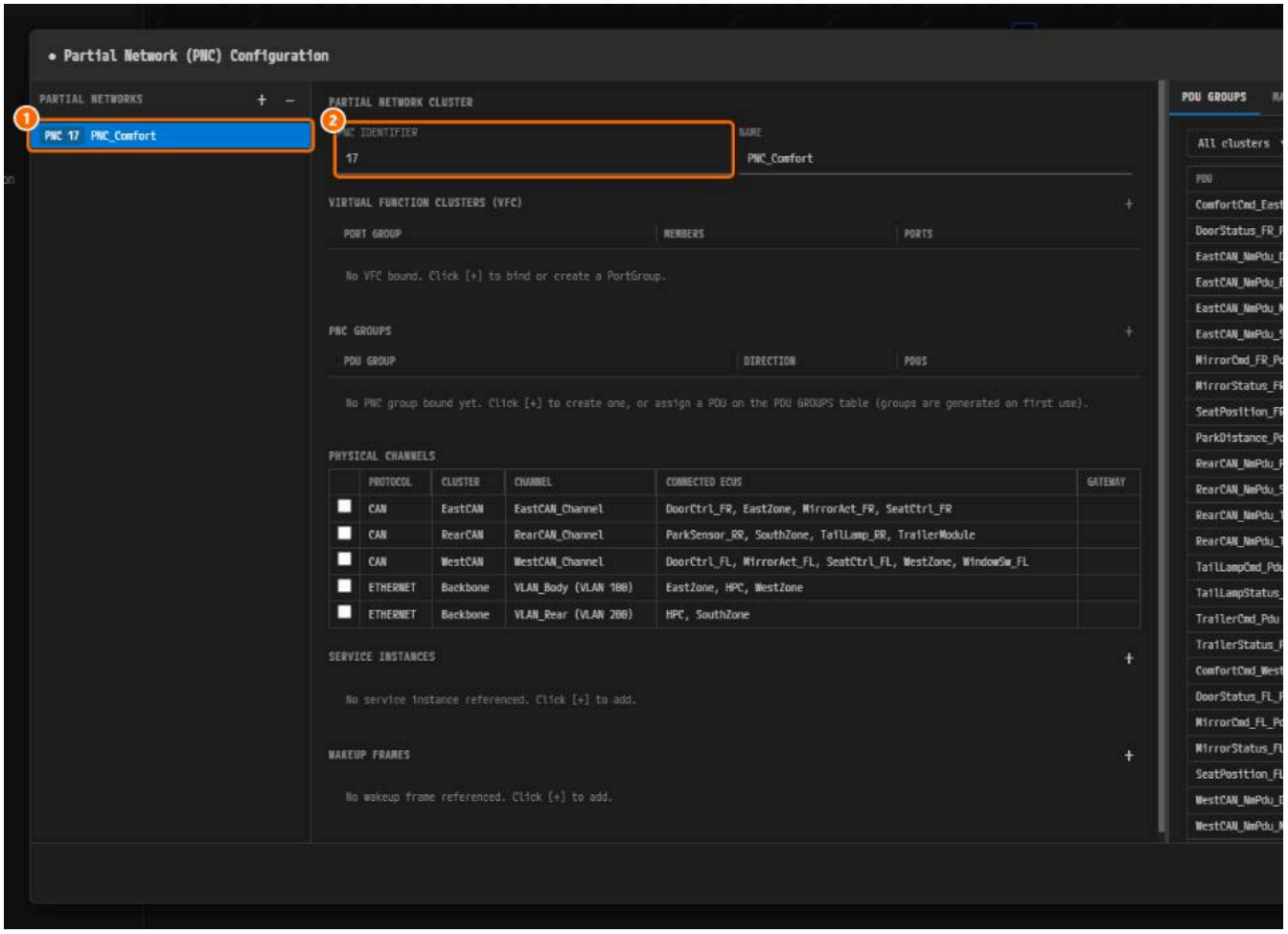


*그림 3-9. ① **PNC 17 PNC_Comfort**, ② **PNC IDENTIFIER** = `17`.*

4. `Ctrl+S`를 누르고 **Close**를 누른다. **VALIDATION** 패널에 "No validation issues."가 보인다.

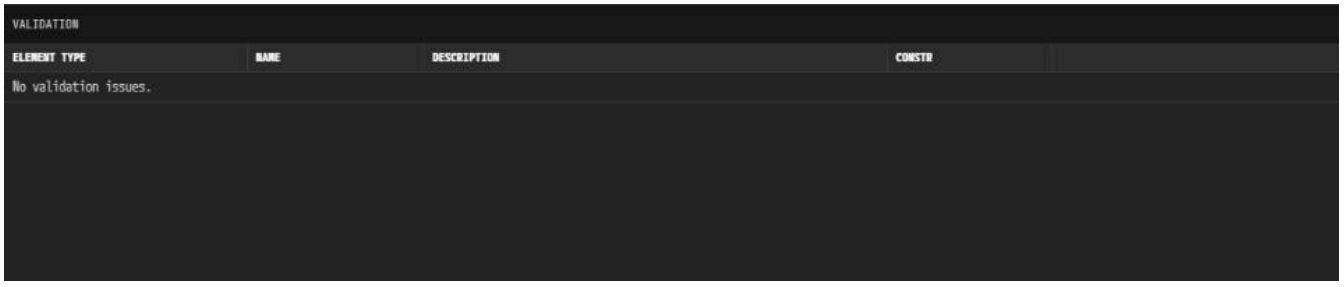


그림 3-10. "No validation issues."

나머지 PNC 추가하기

1. PNC 버튼(그림 3-1)을 눌러 대화상자를 다시 연다.
2. **[+]**를 누른다. **PNC IDENTIFIER**에 `18`, **NAME**에 `PNC_ParkAssist`를 입력하고 각각 `Enter`를 누른다.
3. 같은 방법으로 `19` `PNC_Trailer`를 추가하고 `Ctrl+S`를 누른다.

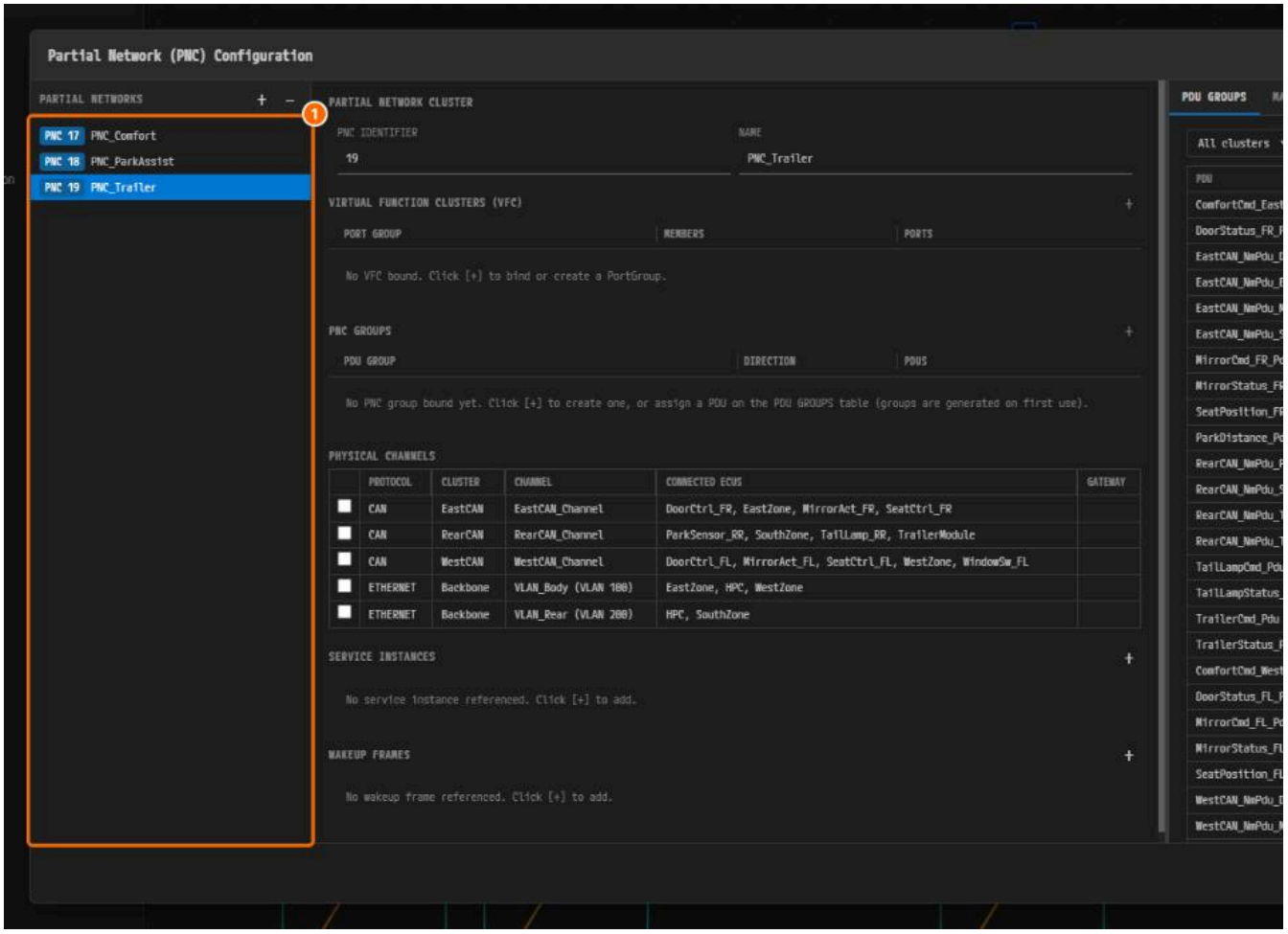


*그림 3-11. ① **PNC 17 PNC_Comfort, PNC 18 PNC_ParkAssist, PNC 19 PNC_Trailer.***

4. **Close**를 누른다. 검증 오류는 여전히 없다.

PNC 비트 벡터 설정하기

1. PNC 대화상자를 다시 열고 **NM VECTOR** 탭을 누른다.

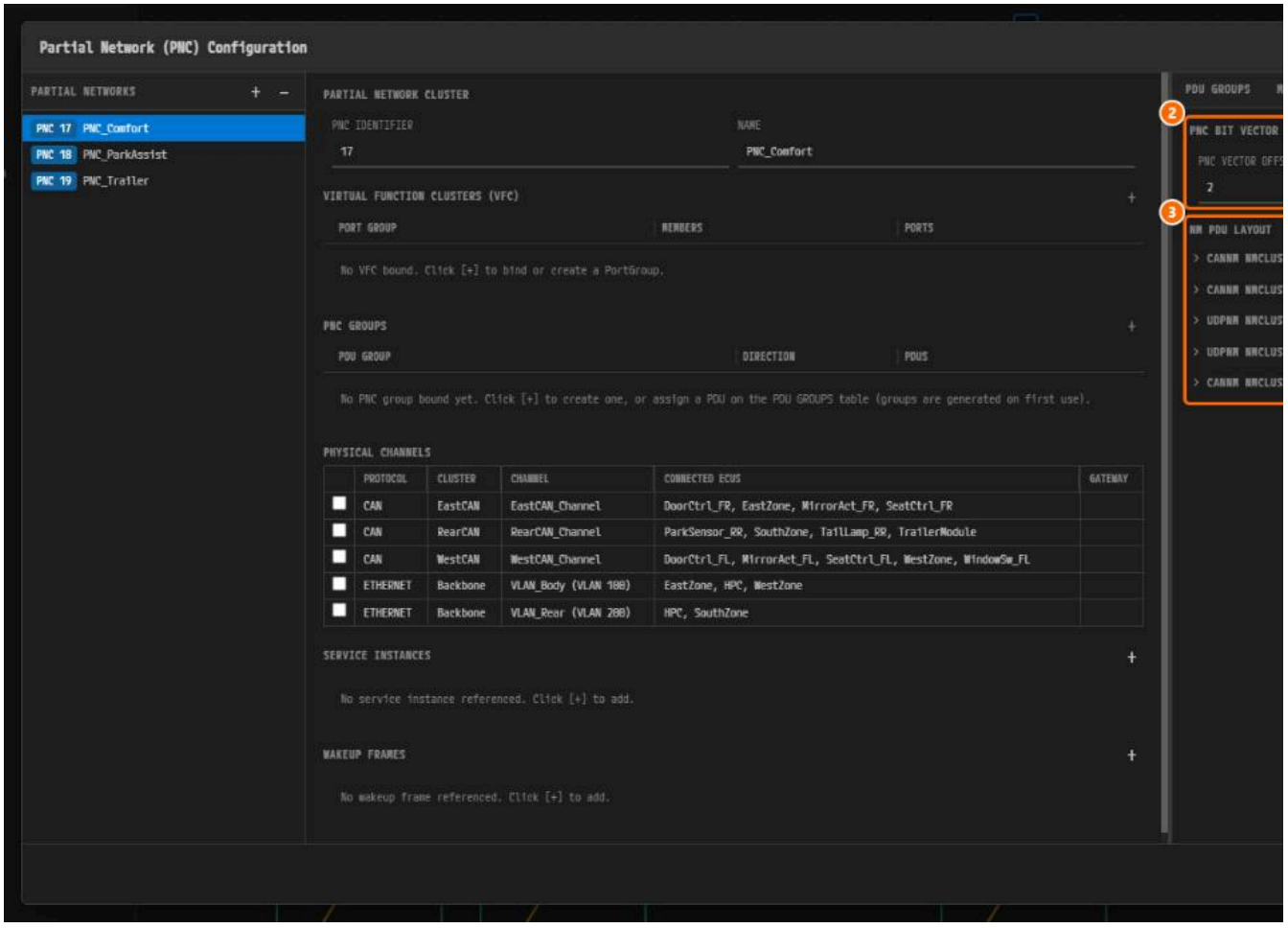


*그림 3-12. ① **NM VECTOR**, ② **PNC BIT VECTOR**, ③ **NM PDU LAYOUT**.*

2. **PNC VECTOR OFFSET**이 `2`, **PNC VECTOR LENGTH**가 `6`인지 확인한다. 다르거나 비어 있으면 입력한다.



*그림 3-13. ① **PNC VECTOR OFFSET** = `2`, ② **PNC VECTOR LENGTH** = `6`.*

3. **NM PDU LAYOUT**에서 `CANNM NMCLUSTER_WESTCAN (WESTCAN)`을 펼친다. 독자의 프로젝트에서는 NM 설정에서 NM을 켠 뒤에야 클러스터 머리글이 보인다.

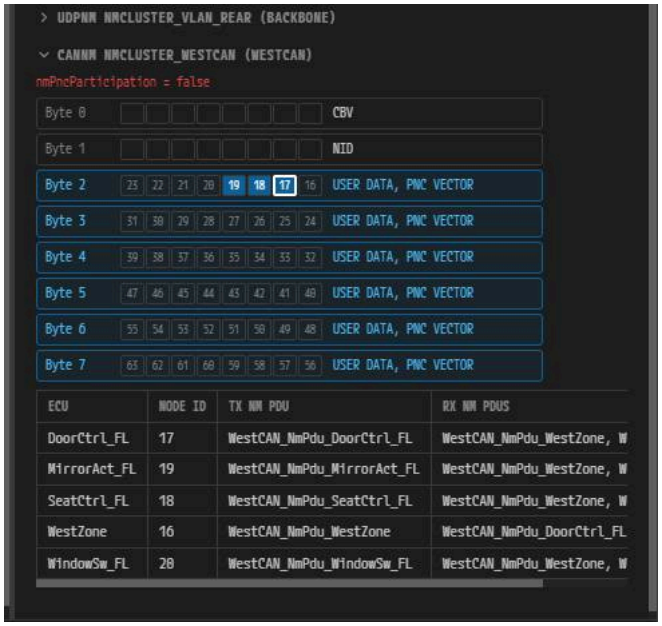


*그림 3-14. Byte 0 = **CBV**, Byte 1 = **NID**, Byte 2~7 = **USER DATA, PNC VECTOR**. Byte 2의 비트 `17`, `18`, `19`가 채워져 있다.*

4. (선택) **PNC VECTOR LENGTH**를 `4`로 바꾼다. **PNC VECTOR**가 Byte 2~5로 줄어든다.

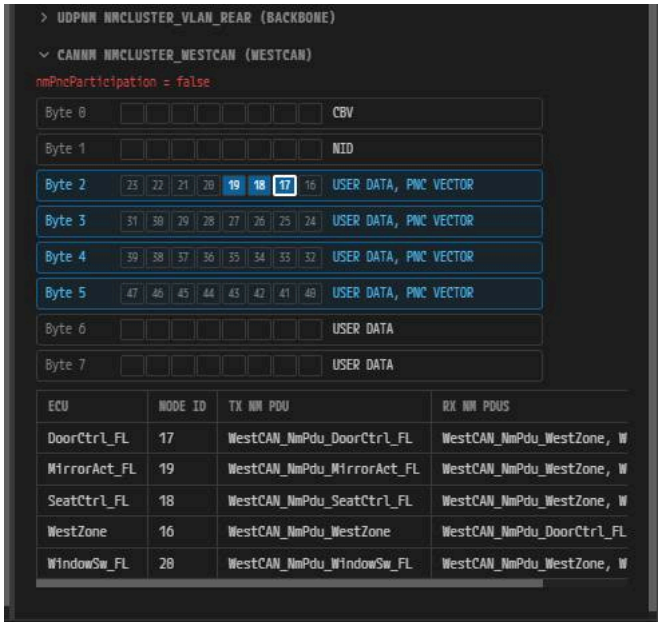


*그림 3-15. **PNC VECTOR LENGTH** = `4`.*

5. **PNC VECTOR LENGTH**를 `6`으로 되돌리고 `Ctrl+S`를 누른 뒤 **Close**를 누른다.

커넥터에 PNC 속성 입력하기

게이트웨이 역할, 웨이크업, 필터 마스크는 ECU가 아니라 커넥터에 속한다. `WestZone_CanFd_Conn`을 따라 한 뒤 나머지는 전체 커넥터 값대로 입력한다.

WestZone_CanFd_Conn 입력하기

1. `WestZone` 아래 가장자리의 커넥터 `WestZone_CanFd_Conn`을 누른다. 속성 패널이 접혀 있으면 캔버스 오른쪽 가장자리의 맨 위 버튼을 누른다.

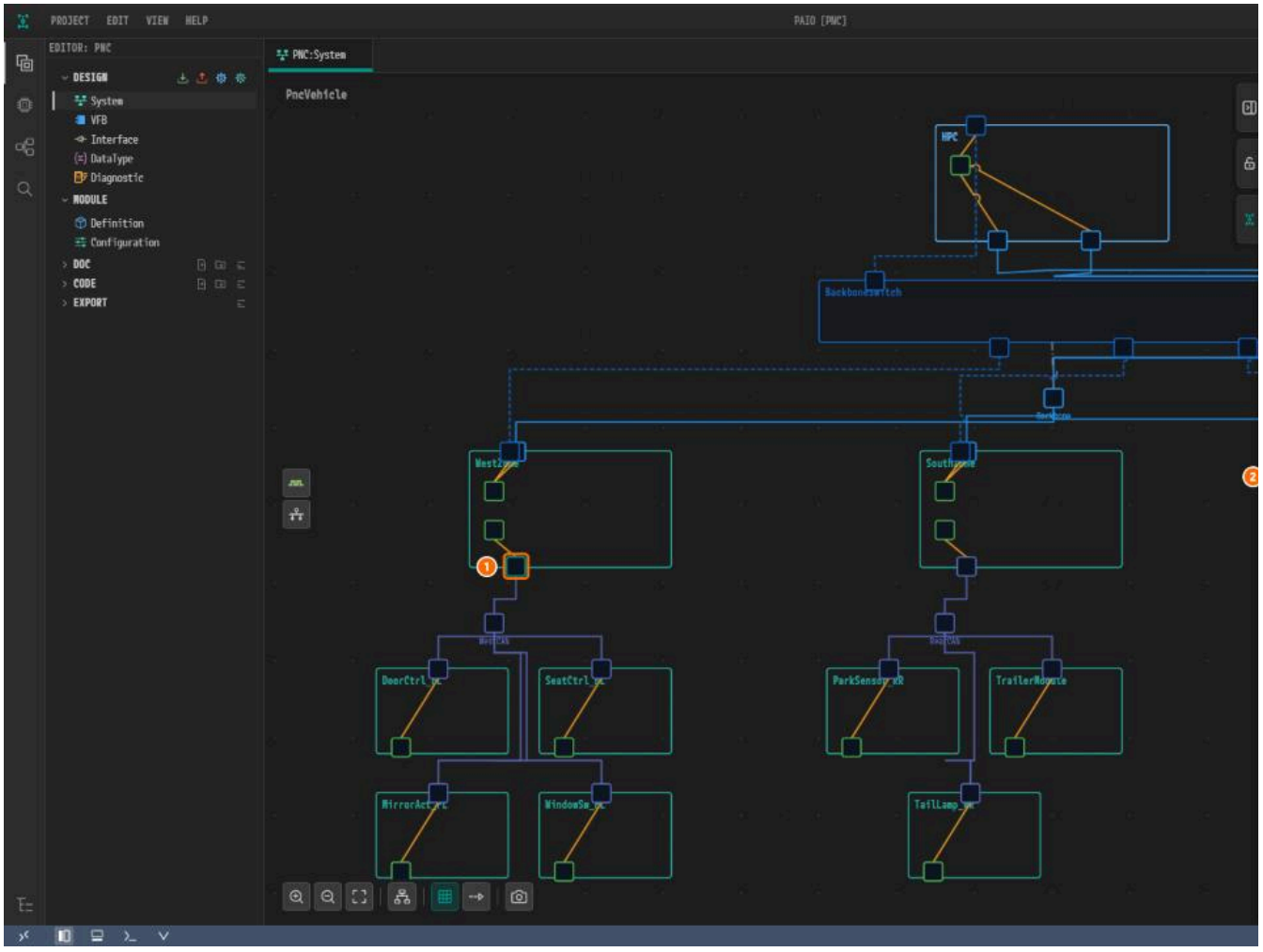


*그림 3-16. ① `WestZone_CanFd_Conn`, ② **PARTIAL NETWORK** 섹션.*

2. **PNC GATEWAY TYPE**에서 **ACTIVE**를 누른다.

3. **CREATE ECU WAKEUP SOURCE**를 켠다.

4. **PNC WAKEUP CAN ID**에 `500`을 입력한다. 칸이 **HEX**이므로 `0x500`으로 보인다.

5. **PNC WAKEUP CAN ID MASK**에 `700`, **PNC WAKEUP DLC**에 `8`을 입력한다.

6. **PNC FILTER ARRAY MASKS**의 Byte 0을 누르고 `02`를 입력한 뒤 `Enter`를 누른다. 빈 칸은 `0x00`으로 저장된다.



*그림 3-17. ① **ACTIVE**, ② **CREATE ECU WAKEUP SOURCE**, ③ `0x500`, ④ `0x700`, ⑤ `8`, ⑥ Byte 0 = `0x02`.*

7. 나머지 칸은 비우거나 끈 채로 두고 `Ctrl+S`를 누른다.

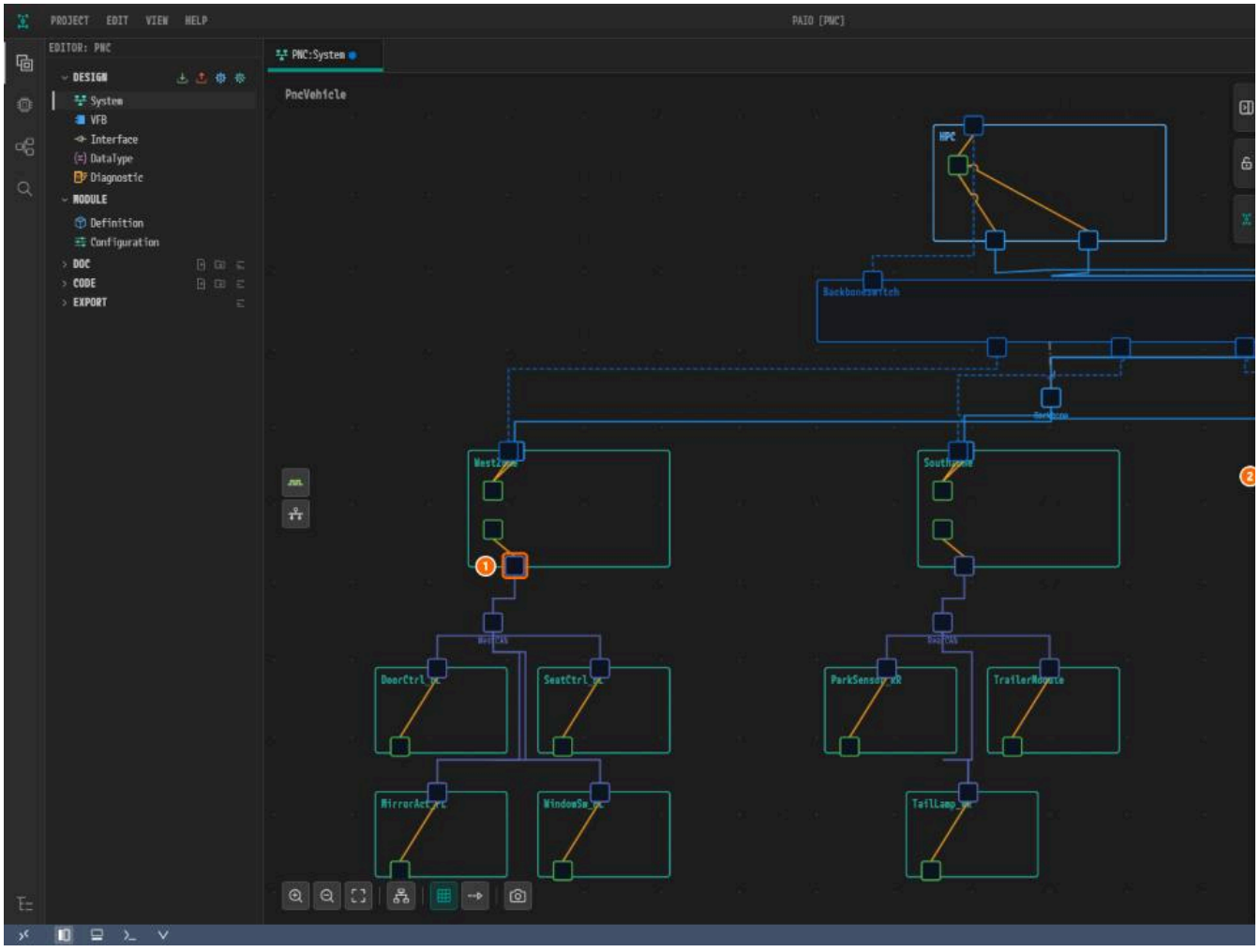


*그림 3-18. ① 커넥터, ② **PARTIAL NETWORK**, 저장 전.*

필터 마스크 계산

필터 마스크는 이 커넥터를 지나는 PNC 비트를 표시한다.

```
byteIndex = pncId / 8 의 몫 - pncVectorOffset    # 17 / 8 = 2, 2 - 2 = 0 → Byte 0
bit       = pncId / 8 의 나머지                  # 17 mod 8 = 1
value     = 2 ^ bit                              # 2 ^ 1 = 0x02
```

PNC가 여럿이면 같은 바이트 안에서 OR 한다: `0x04` | `0x08` = `0x0C`. 배열은 `pncVectorLength`(6) 바이트이다(`constr_3686`).

Ethernet 커넥터와 HPC 커넥터

Ethernet 커넥터에는 웨이크업 CAN ID 칸이 없다. `HPC` 커넥터에는 **PNC GATEWAY TYPE**도 없다. CP ECU에만 있기 때문이다. 필터 마스크만 입력한다.

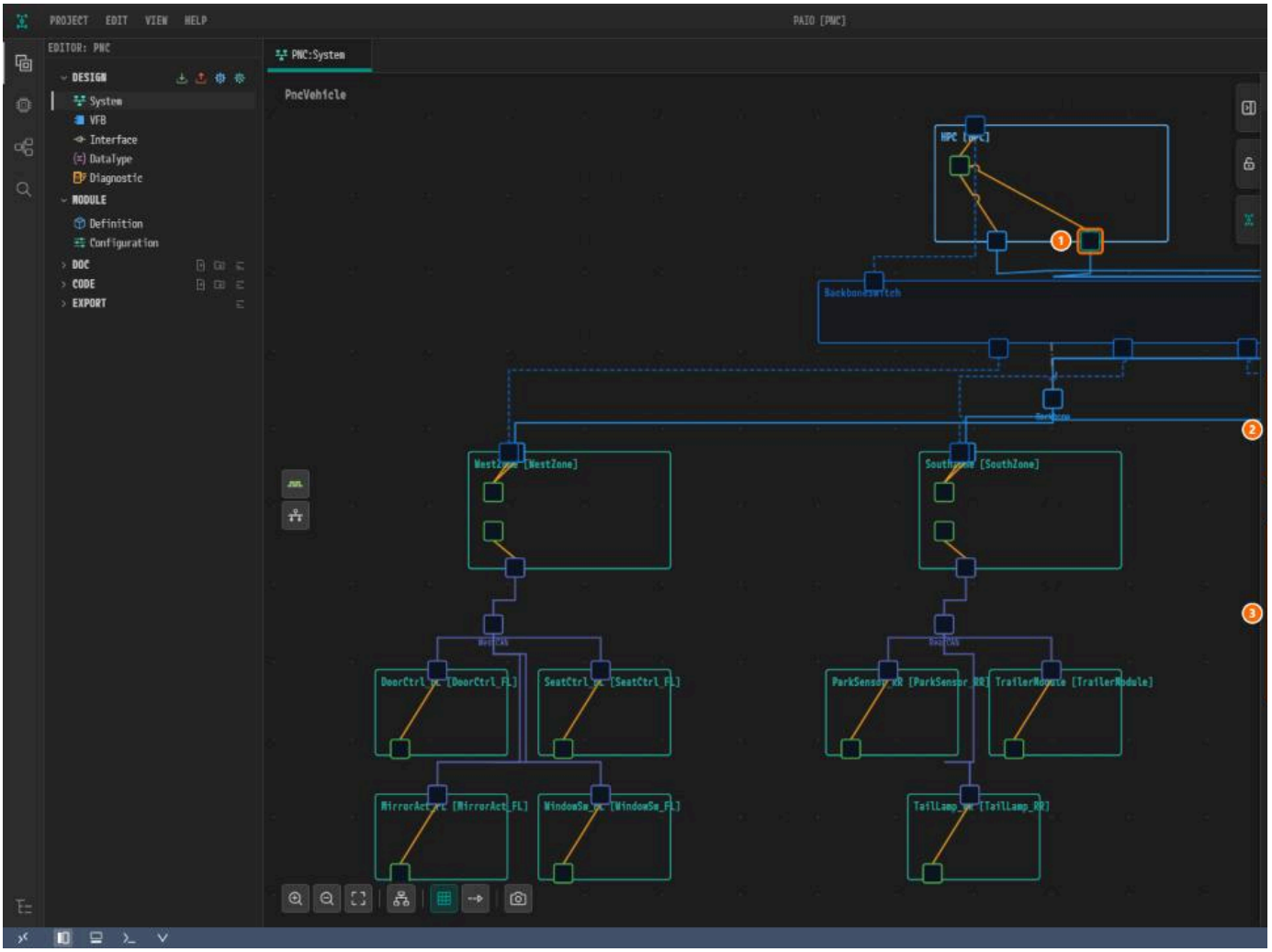


*그림 3-19. 완성 예제의 `HPC_Eth_Rear_Conn`. ① 커넥터, ② **SERVICE INSTANCES**, ③ **PARTIAL NETWORK**.*

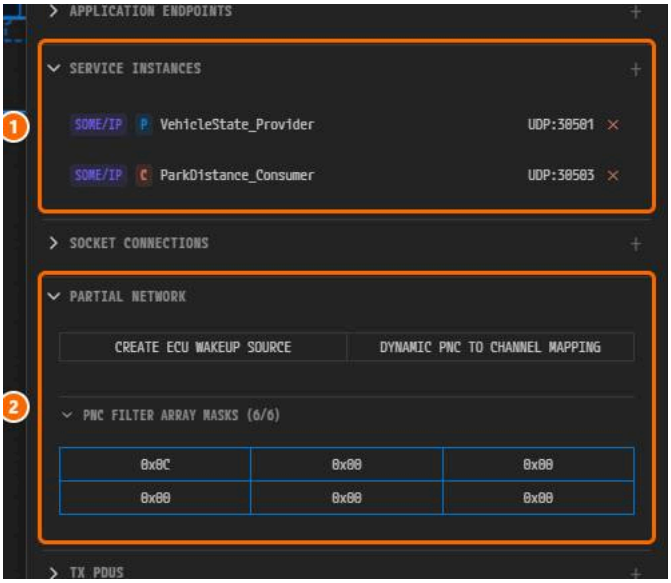


*그림 3-20. ① 인터페이스와 SOME/IP 서비스에서 만드는 **SERVICE INSTANCES**. ② **PARTIAL NETWORK**: 마스크 `0x0C 0x00 0x00 0x00 0x00 0x00`, 게이트웨이 유형 없음.*

VLAN_Body의 활성 게이트웨이는 하나뿐이므로 `EastZone_Eth_Conn`은 **PASSIVE**이다.

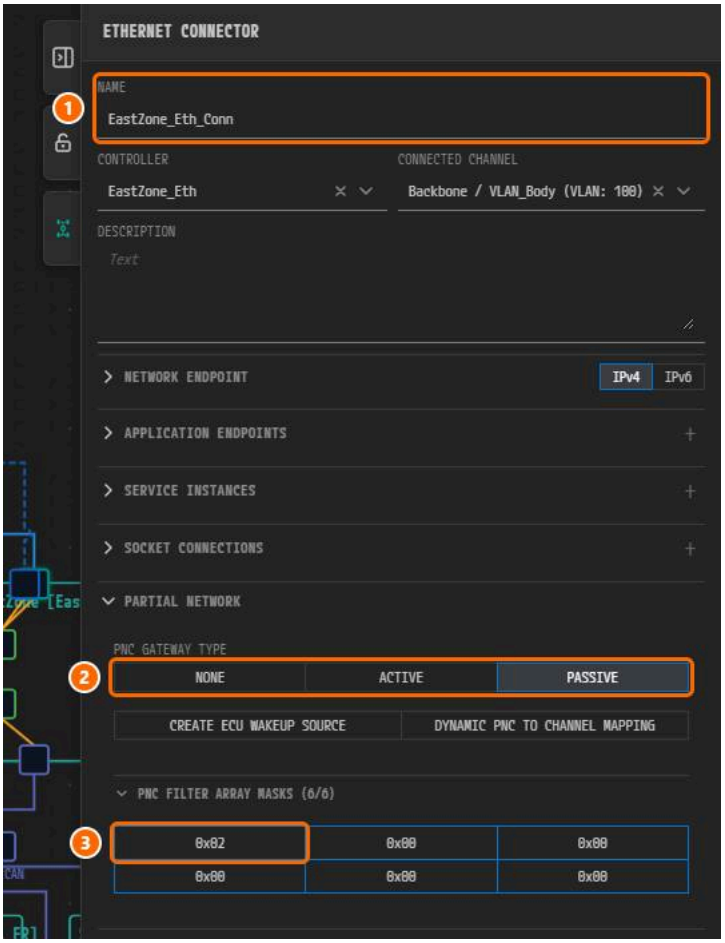


*그림 3-21. ① `EastZone_Eth_Conn`, ② **PASSIVE**, ③ Byte 0 = `0x02`.*

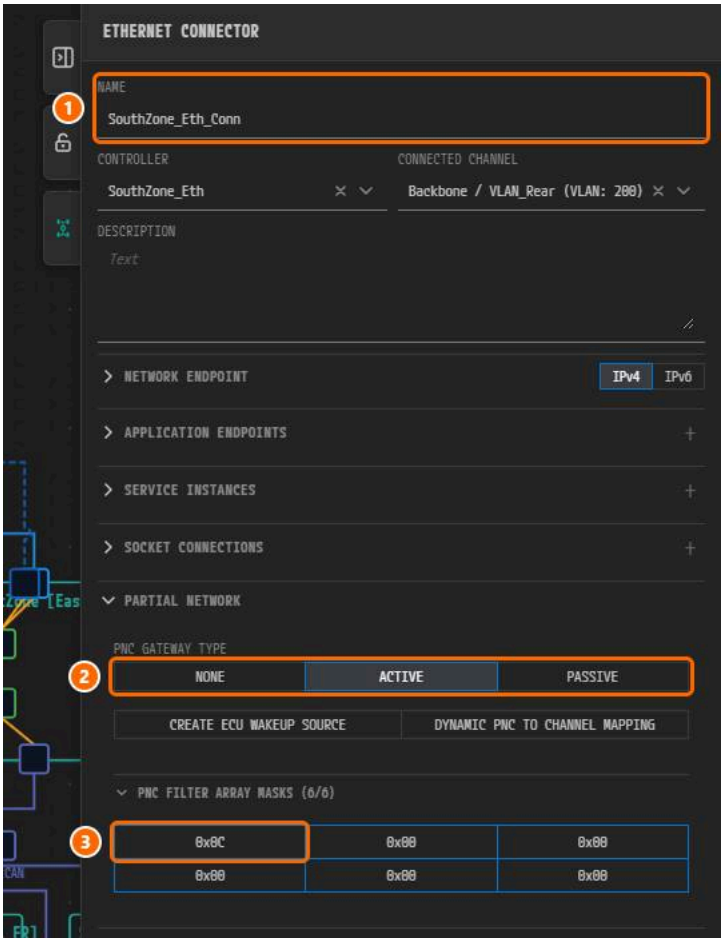


*그림 3-22. ① `SouthZone_Eth_Conn`, ② **ACTIVE**, ③ Byte 0 = `0x0C`.*

> **팁.** 펼친 속성 패널이 캔버스의 `EastZone` 쪽을 가린다. 그쪽 커넥터를 누르기 전에 패널을 접거나 캔버스를 끌어 옮긴다.

전체 커넥터 값

Byte 0만 적었다. Byte 1~5는 `0x00`이다. "—"는 비우거나 끈다는 뜻이다.

커넥터	채널	PNC GATEWAY TYPE	CREATE ECU WAKEUP SOURCE	WAKEUP CAN ID / MASK / DLC	Byte 0
`WestZone_CanFd_Conn`	WestCAN	**ACTIVE**	켬	`0x500` / `0x700` / `8`	`0x02`
`EastZone_CanFd_Conn`	EastCAN	**ACTIVE**	켬	`0x500` / `0x700` / `8`	`0x02`
`SouthZone_CanFd_Conn`	RearCAN	**ACTIVE**	켬	`0x500` / `0x700` / `8`	`0x0C`
`WestZone_Eth_Conn`	VLAN_Body	**ACTIVE**	—	(칸 없음)	`0x02`
`EastZone_Eth_Conn`	VLAN_Body	**PASSIVE**	—	(칸 없음)	`0x02`
`SouthZone_Eth_Conn`	VLAN_Rear	**ACTIVE**	—	(칸 없음)	`0x0C`
`DoorCtrl_FL`, `SeatCtrl_FL`, `MirrorAct_FL`, `WindowSw_FL`의 `_CAN_Conn`	WestCAN	**NONE**	켬	—	`0x02`
`DoorCtrl_FR`, `SeatCtrl_FR`, `MirrorAct_FR`의 `_CAN_Conn`	EastCAN	**NONE**	켬	—	`0x02`
`ParkSensor_RR_CAN_Conn`	RearCAN	**NONE**	켬	—	`0x04`
`TrailerModule_CAN_Conn`, `TailLamp_RR_CAN_Conn`	RearCAN	**NONE**	켬	—	`0x08`
`HPC_Eth_Body_Conn`	VLAN_Body	(칸 없음)	—	(칸 없음)	`0x02`
`HPC_Eth_Rear_Conn`	VLAN_Rear	(칸 없음)	—	(칸 없음)	`0x0C`

모든 커넥터를 마치면 `Ctrl+S`를 누른다.

노드의 PARTIAL NETWORK 값 입력하기

노드에도 **PARTIAL NETWORK** 섹션이 있다. 여기에는 NM 요청과 슬립 타이머가 있다.

1. `WestZone` 노드의 머리글을 누른다(커넥터가 아니다).
2. **PARTIAL NETWORK**에서 **PNC NM REQUEST**를 켠다. **PNC SYNCHRONOUS WAKEUP**과 **CHANNEL SYNCHRONOUS WAKEUP**은 끈 채로 둔다.
3. **PNC PREPARE SLEEP TIMER**에 `0.5`, **PN RESET TIME**에 `2`를 입력한다. 기본 단위는 **s**이다.

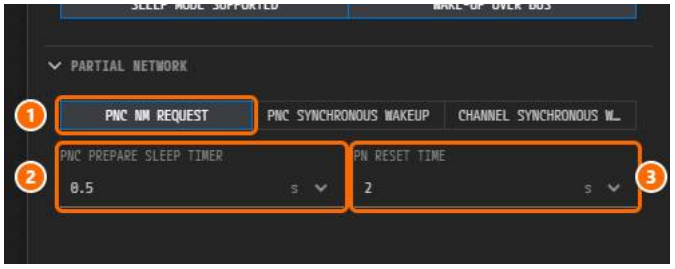


*그림 3-23. ① **PNC NM REQUEST** 켬, ② `0.5` s, ③ `2` s.*

4. 나머지 노드를 표대로 입력하고 `Ctrl+S`를 누른다.

노드	PNC NM REQUEST	PNC SYNCHRONOUS WAKEUP	PNC PREPARE SLEEP TIMER	PN RESET TIME / PN RESET TIMER
`WestZone`, `SouthZone`, `EastZone`	켬	끔	`0.5` s	`2` s(**PN RESET TIME**)
말단 ECU 10개	켬	끔	`0.5` s	`2` s(**PN RESET TIME**)
`HPC`	(칸 없음)	(칸 없음)	`0.5` s	`2` s(**PN RESET TIMER**)

`HPC`는 AP 머신이므로 타이머 칸 두 개만 있다.



*그림 3-24. ① **PNC PREPARE SLEEP TIMER** = `0.5` s, ② **PN RESET TIMER** = `2` s.*

스위치 포트에 PNC 지정하기

스위치 포트의 **PNC MAPPINGS**는 그 포트가 쓰이는 PNC를 정한다.

1. `BackboneSwitch` 노드를 더블클릭한다. **Ethernet Switch Configuration [BackboneSwitch]** 워크벤치가 열린다.
2. 스위치 다이어그램에서 `Port_1`을 누른다.
3. **PNC MAPPINGS**를 열어 `PNC_Comfort (PNC 17)`을 체크하고 `Esc`를 누른다. 목록이 비어 있으면 워크벤치를 닫았다가 다시 연다.

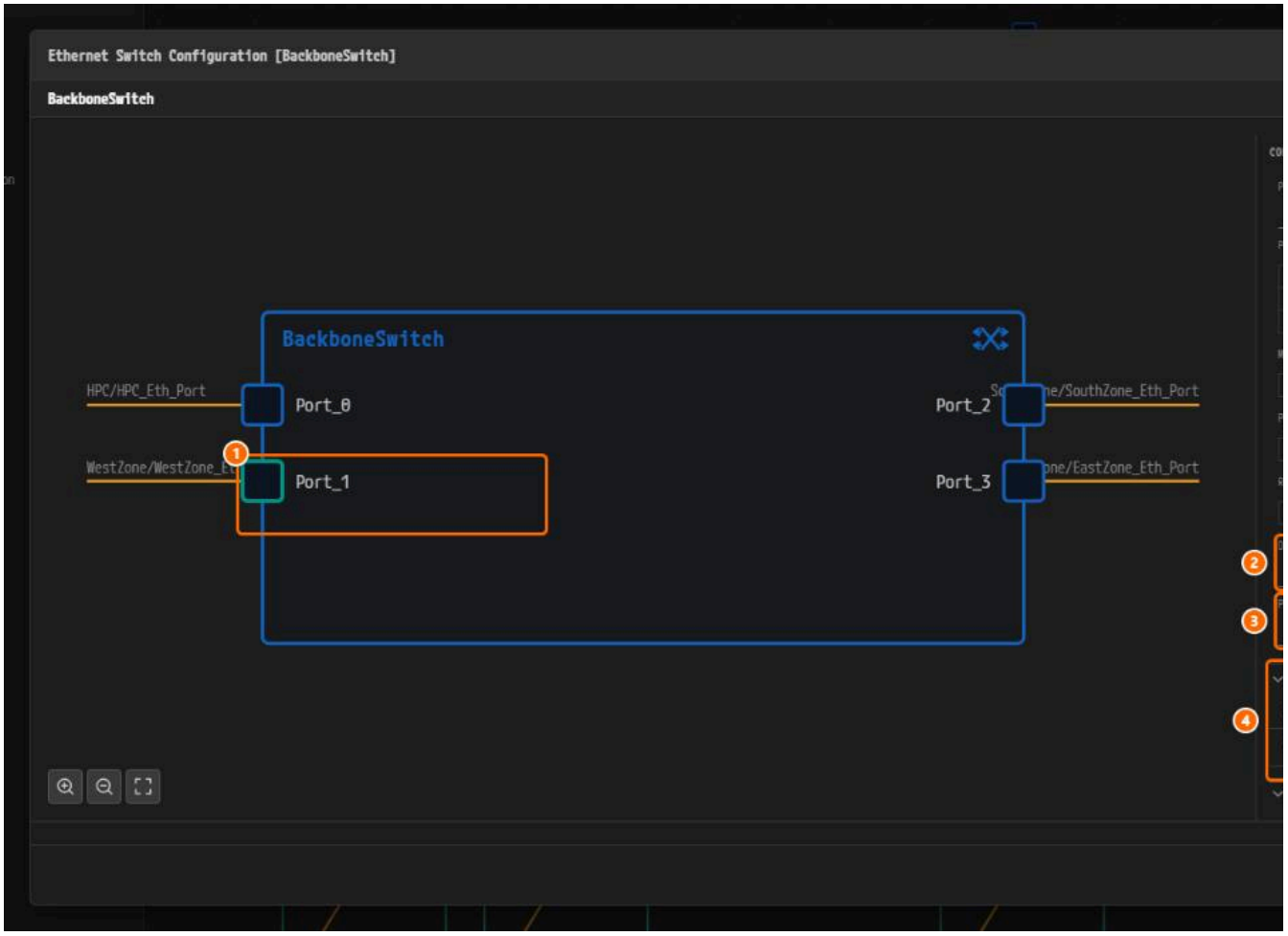


*그림 3-25. ① `Port_1`, ② **DEFAULT VLAN (PVID)**, ③ **PNC MAPPINGS**, ④ **VLAN MEMBERSHIP**.*



*그림 3-26. ① **DEFAULT VLAN (PVID)**, ② **PNC MAPPINGS**, ③ **VLAN MEMBERSHIP**.*

4. 나머지 포트를 표대로 지정한다.

포트	연결 대상	VLAN	PNC MAPPINGS
`Port_0`	HPC	`VLAN_Body`, `VLAN_Rear`	`PNC_Comfort`, `PNC_ParkAssist`, `PNC_Trailer`
`Port_1`	WestZone	`VLAN_Body`	`PNC_Comfort`
`Port_2`	SouthZone	`VLAN_Rear`	`PNC_ParkAssist`, `PNC_Trailer`
`Port_3`	EastZone	`VLAN_Body`	`PNC_Comfort`

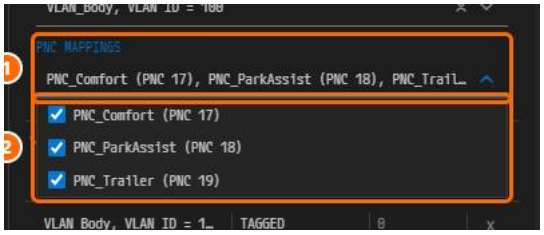


그림 3-27. ① 닫힌 칸, ② `Port_0`에 체크한 PNC 세 개.

5. 워크벤치에서 `Ctrl+S`를 누르고 **Close**를 누른다.

6. System 편집기에서 `Ctrl+S`를 누른다.

> **참고.** 완성 예제는 Ethernet 컨트롤러 포트에도 PNC를 지정한다. `HPC_Eth_Port`에는 세 개 모두, `WestZone_Eth_Port`와 `EastZone_Eth_Port`에는 `PNC_Comfort`, `S` `PNC_Trailer`이다.

NM 설정

PNC 비트는 NM PDU에 실려 간다. 그래서 각 NM 클러스터에는 **NM PNC PARTICIPATION** = **YES**와 필수 타이밍 값이, 각 NM 노드에는 **NM NODE ID**와 NM-PDU가 있어야 한다. NM 자는 클러스터마다 NM을 켜고 값을 채운다. `NmCluster_WestCAN`과 `WestZone`을 따라 한 뒤 전체 NM 값 표를 따른다.

빠진 NM 필수 속성 확인하기

NM 클러스터의 필수 속성이 비어 있으면 검증이 오류 4건을 보고한다.

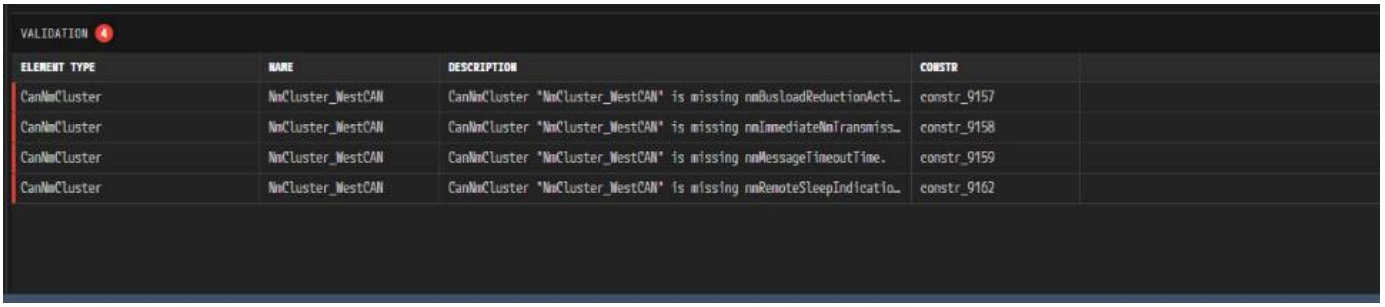


그림 3-28. `NmCluster_WestCAN`의 `constr_9157`, `constr_9158`, `constr_9159`, `constr_9162`.

차례로 **NM BUSLOAD REDUCTION ACTIVE**, **NM IMMEDIATE NM TRANSMISSIONS**, **NM MESSAGE TIMEOUT TIME**, **NM REMOTE SLEEP INDICATION TIME**이다. 독자의 프로 지 않을 수 있다.

NmCluster_WestCAN 값 채우기

1. `WestZone` 아래의 `WestCAN` 클러스터 노드를 더블클릭한다.

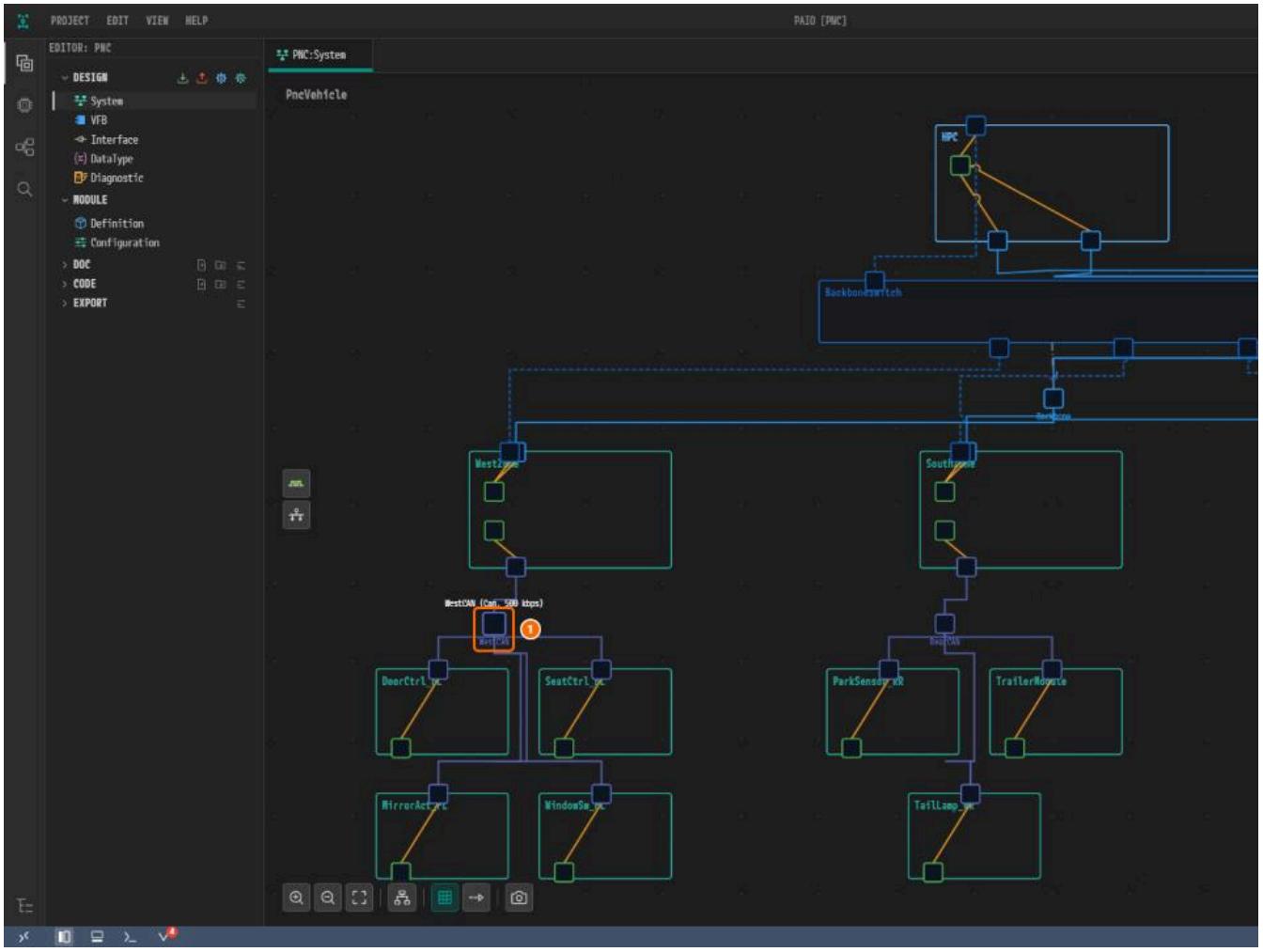


그림 3-29. ① `WestCAN` 클러스터 노드.

2. **Communication Cluster Configuration [WestCAN]**에서 **CAN NM CLUSTER** 탭을 누른다. 탭이 빈 동그라미와 함께 **NM CLUSTER**로 보이면 오른쪽 클릭하고 **Enabl**

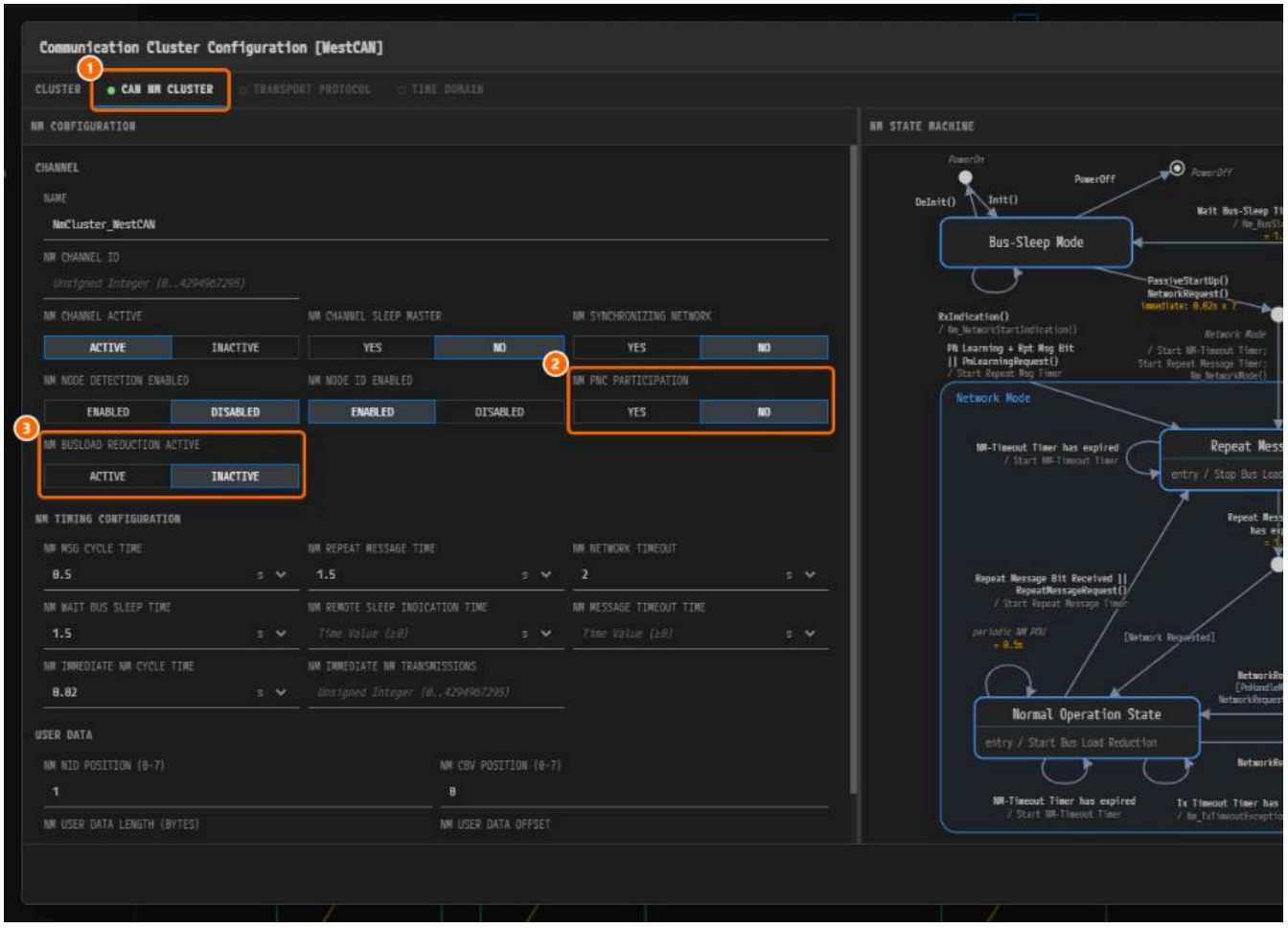


*그림 3-30. ① **CAN NM CLUSTER**, ② **NM PNC PARTICIPATION**, ③ **NM BUSLOAD REDUCTION ACTIVE**.*

3. **NM PNC PARTICIPATION**을 **YES**로 바꾼다.

4. **NM BUSLOAD REDUCTION ACTIVE**를 **ACTIVE**로 눌렀다가 다시 **INACTIVE**로 누른다.

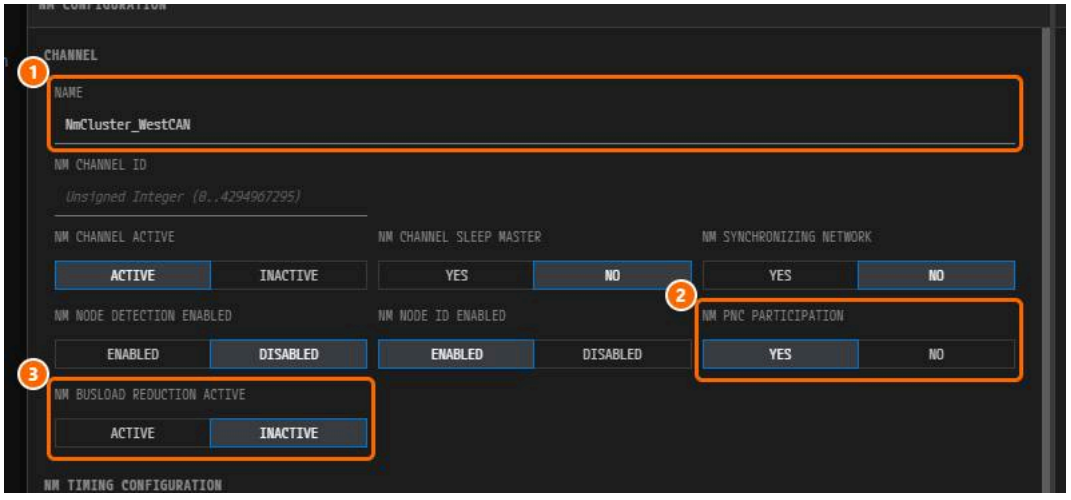


*그림 3-31. ① `NmCluster_WestCAN`, ② **YES**, ③ **INACTIVE**. **NM CHANNEL ID**는 비워 둔다.*

> **주의.** 값이 없는 두 상태 버튼은 한쪽이 눌린 것처럼 보인다. 반대쪽을 한 번 눌렀다가 되돌려 눌러야 값이 저장된다.

5. **NM TIMING CONFIGURATION**에 아래 값을 입력한다. 기본 단위는 **s**이다.

칸	값
NM MSG CYCLE TIME	`0.5`
NM REPEAT MESSAGE TIME	`1.5`
NM WAIT BUS SLEEP TIME	`1.5`
NM NETWORK TIMEOUT	`2`
NM MESSAGE TIMEOUT TIME	`0.1`
NM REMOTE SLEEP INDICATION TIME	`1`
NM IMMEDIATE NM CYCLE TIME	`0.02`
NM IMMEDIATE NM TRANSMISSIONS	`5`

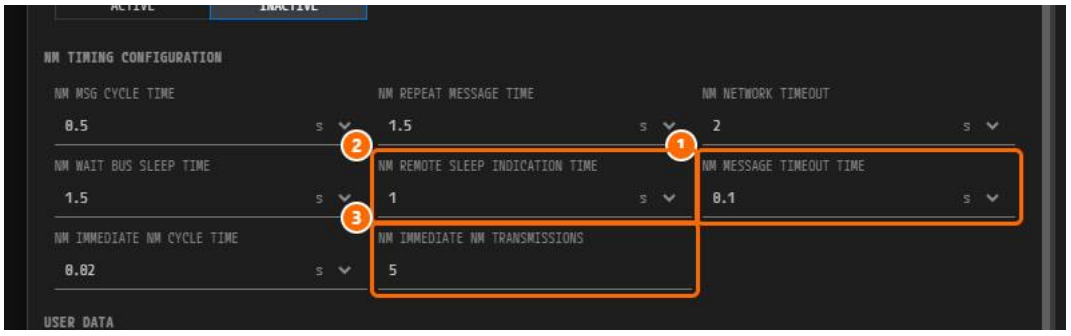


*그림 3-32. ① **NM MESSAGE TIMEOUT TIME** = `0.1` s, ② **NM REMOTE SLEEP INDICATION TIME** = `1` s, ③ **NM IMMEDIATE NM TRANSMISSIONS** = `5`.*

> **주의.** NM REPEAT MESSAGE TIME(`1.0`), NM MESSAGE TIMEOUT TIME(`2`), NM IMMEDIATE NM TRANSMISSIONS(`0`)의 기본값은 표와 다르고 NM IMMEDIATE N

6. 폼 끝의 **USER DATA**에 **NM NID POSITION (0-7)** `1`, **NM CBV POSITION (0-7)** `0`, **NM USER DATA OFFSET** `2`, **NM USER DATA LENGTH (BYTES)** `6`을 입력한다.

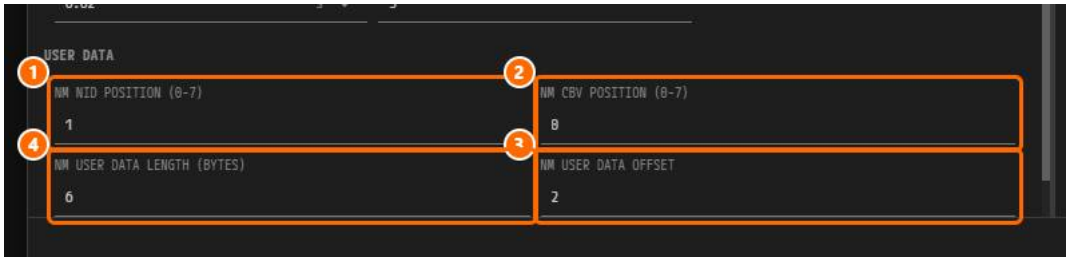


그림 3-33. ① `1`, ② `0`, ③ `2`, ④ `6`. 뒤의 두 값은 PNC 비트 벡터와 같다.

7. `Ctrl+S`를 누르고 **Close**를 누른다.

WestZone의 NM 노드 값 채우기

1. `WestZone` 노드를 더블클릭한다. **ECU Configuration [WestZone]**에서 **NETWORK MANAGEMENT** 탭을 누른다.

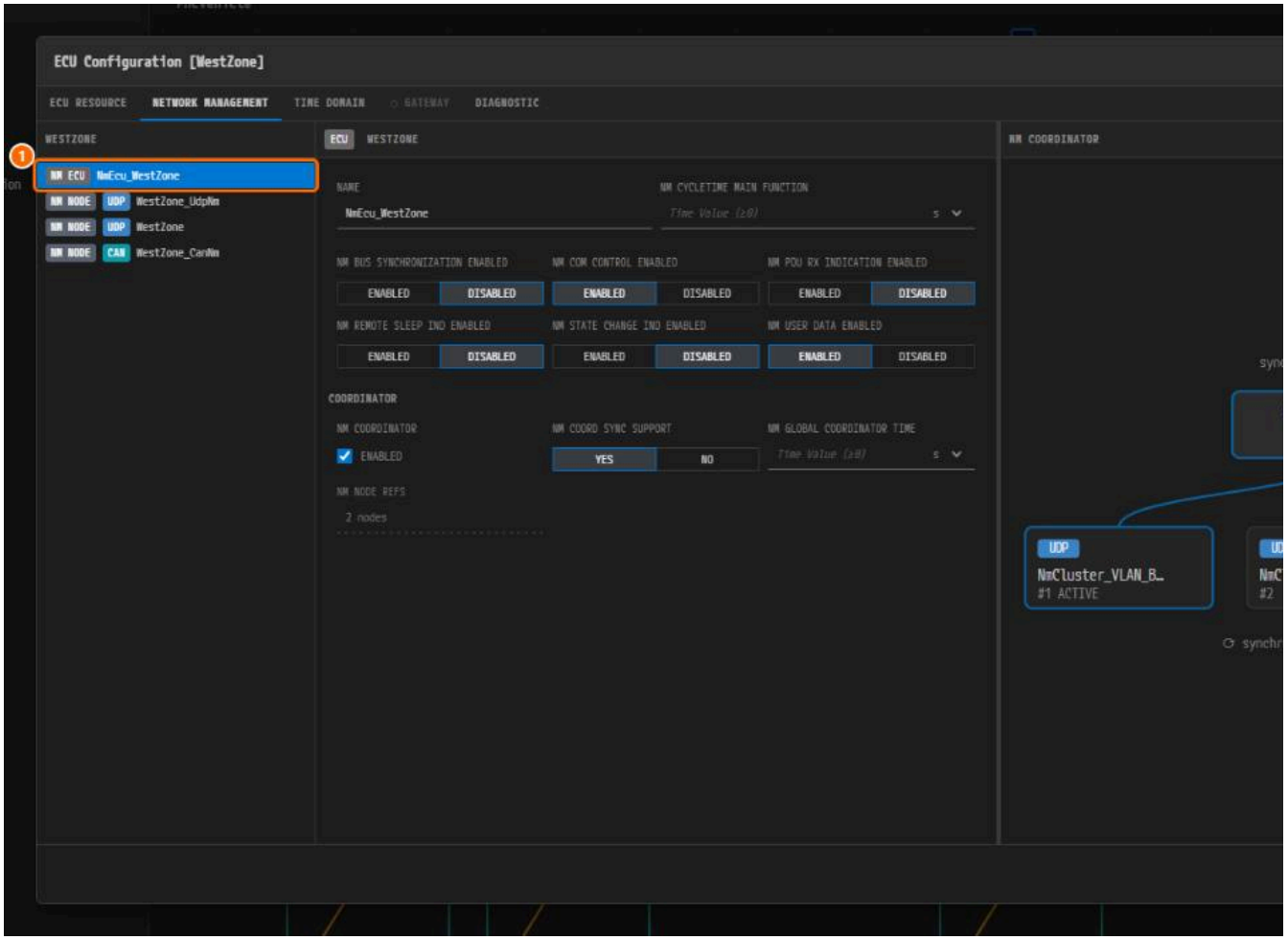


*그림 3-34. ① **NM ECU** 1개와 **CAN**, **UDP** 배지가 붙은 **NM NODE** 항목의 목록.*

2. **CAN** 배지가 붙은 NM 노드를 고른다. **NAME**에 `WestZone_CanNm`, **NM NODE ID**에 `16`을 입력하고 **NM COORDINATOR ROLE**에서 **ACTIVE**를 누른다.

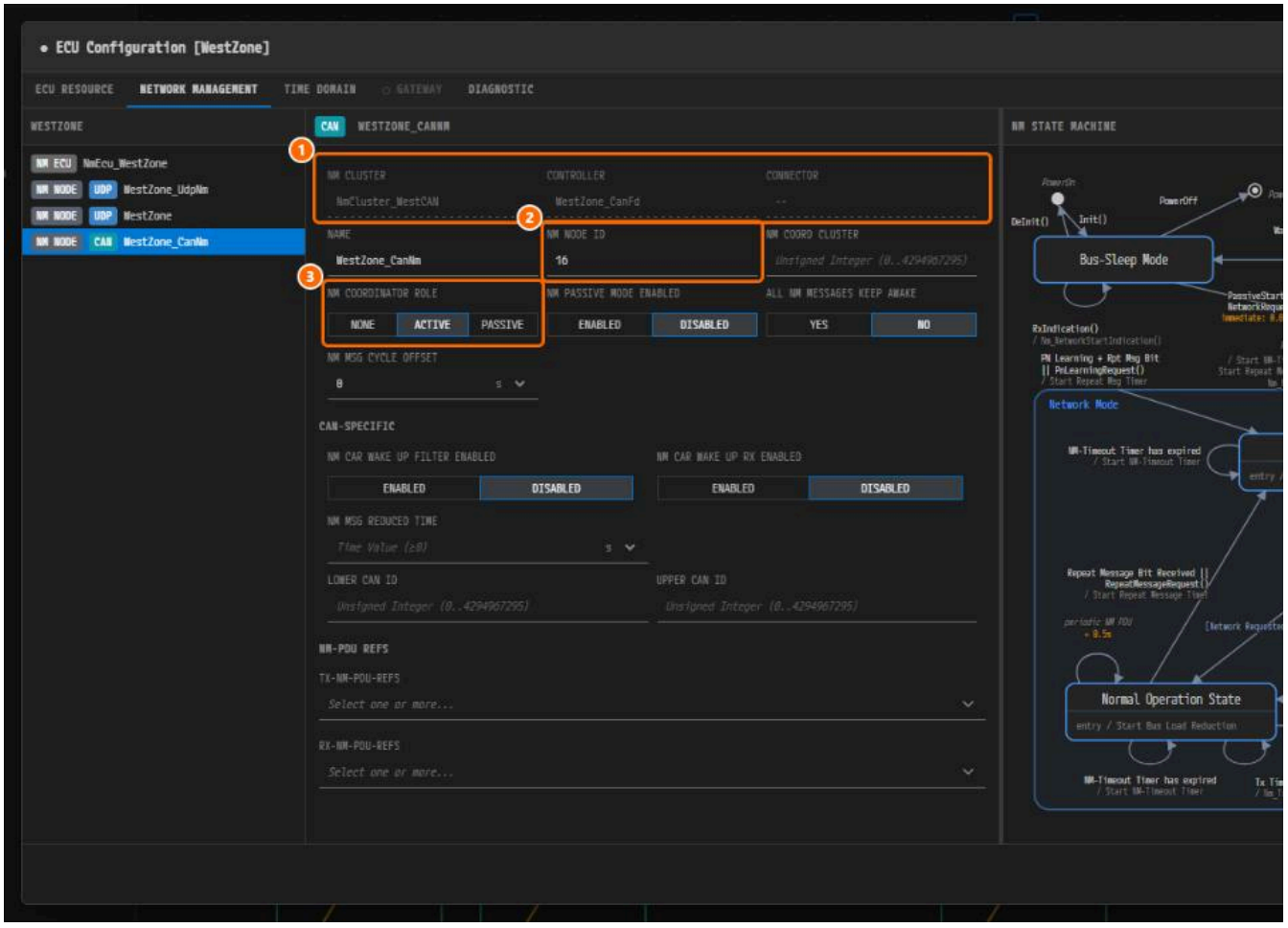


*그림 3-35. ① 읽기 전용 줄, ② **NM NODE ID** = `16`, ③ **NM COORDINATOR ROLE** = **ACTIVE**.*

> **주의.** 3, 4단계에는 NM-PDU가 필요하다. 지금은 건너뛰고 NM-PDU 만들기를 마친 뒤 돌아온다.

3. **TX-NM-PDU-REFS**를 열어 `WestCAN_NmPdu_WestZone`만 체크하고 `Esc`를 누른다.

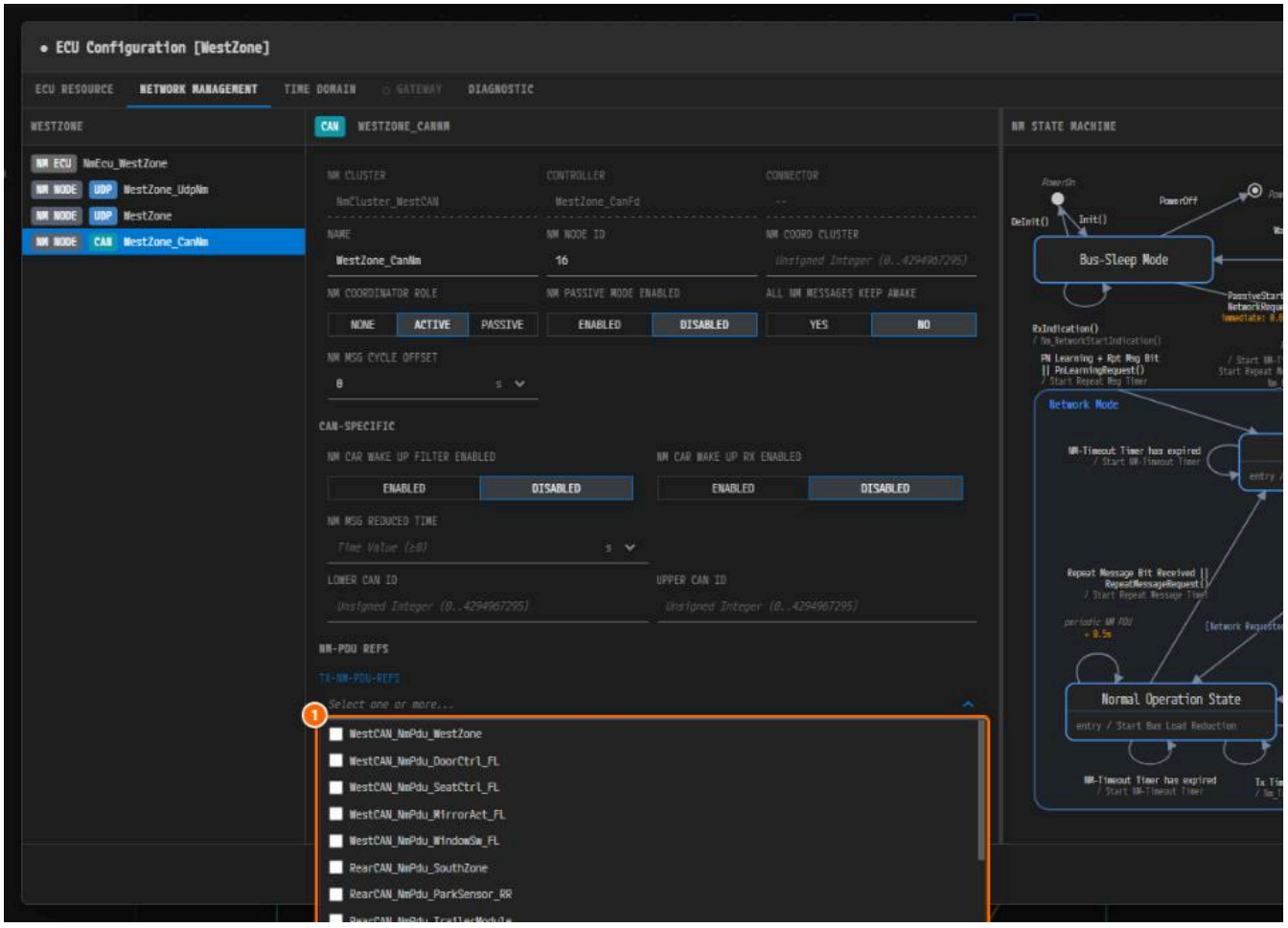


*그림 3-36. ① **TX-NM-PDU-REFS** 목록. 버스 접두어 `WestCAN_`을 보고 고른다.*

4. **RX-NM-PDU-REFS**에서 `WestCAN_NmPdu_DoorCtrl_FL`, `WestCAN_NmPdu_SeatCtrl_FL`, `WestCAN_NmPdu_MirrorAct_FL`, `WestCAN_NmPdu_WindowSw_FL`을

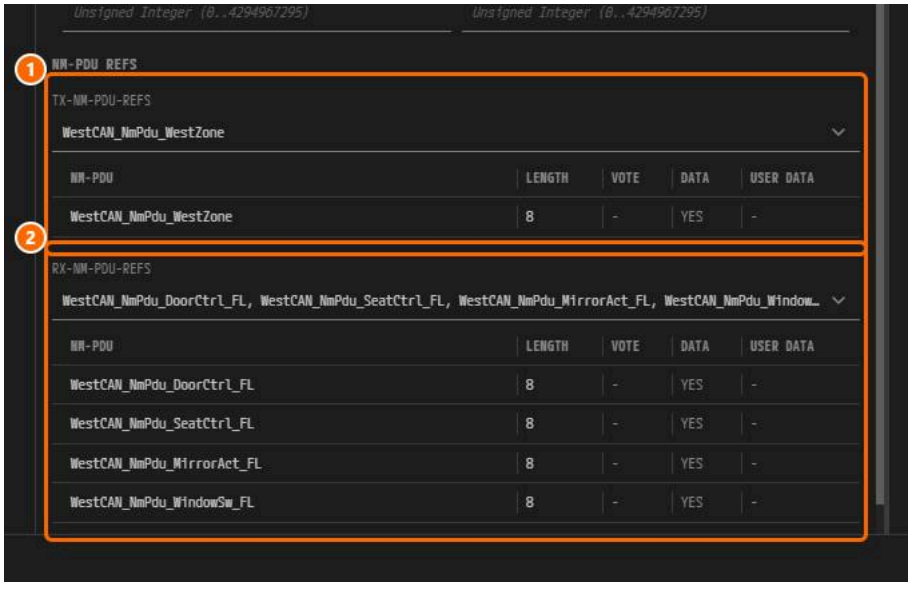


*그림 3-37. ① **TX-NM-PDU-REFS** 1개(최대 1개, `constr_9388`), ② **RX-NM-PDU-REFS** 4개.*

5. `Ctrl+S`를 누르고 **Close**를 누른다.

6. **VALIDATION** 패널에 `constr_9157`, `constr_9158`, `constr_9159`, `constr_9162`가 더 이상 없다.

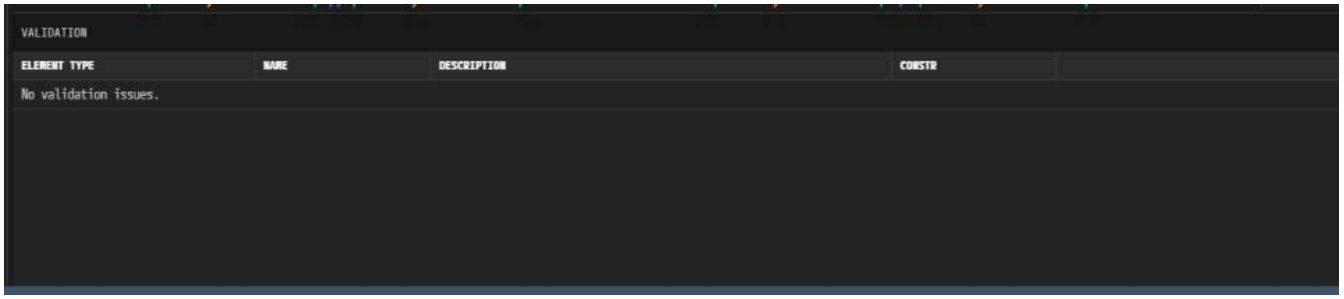


그림 3-38. "No validation issues."

> **주의. Communication Cluster Configuration** 대화상자에서 `Ctrl+S`를 누르면 `IVC.io.model`을 다시 쓴다. `Backbone`의 VLAN 구성이 사라지고 Ethernet NM 노
> 성 예제의 차이를 본다.

Ethernet(UdpNm) 쪽 차이점

1. `Backbone` 클러스터 노드를 더블클릭하고 **UDP NM CLUSTER**를 누른다(필요하면 **Enable**). CAN과 같은 값을 입력하되 **NM IMMEDIATE NM CYCLE TIME**과 **NM IMMEDIA

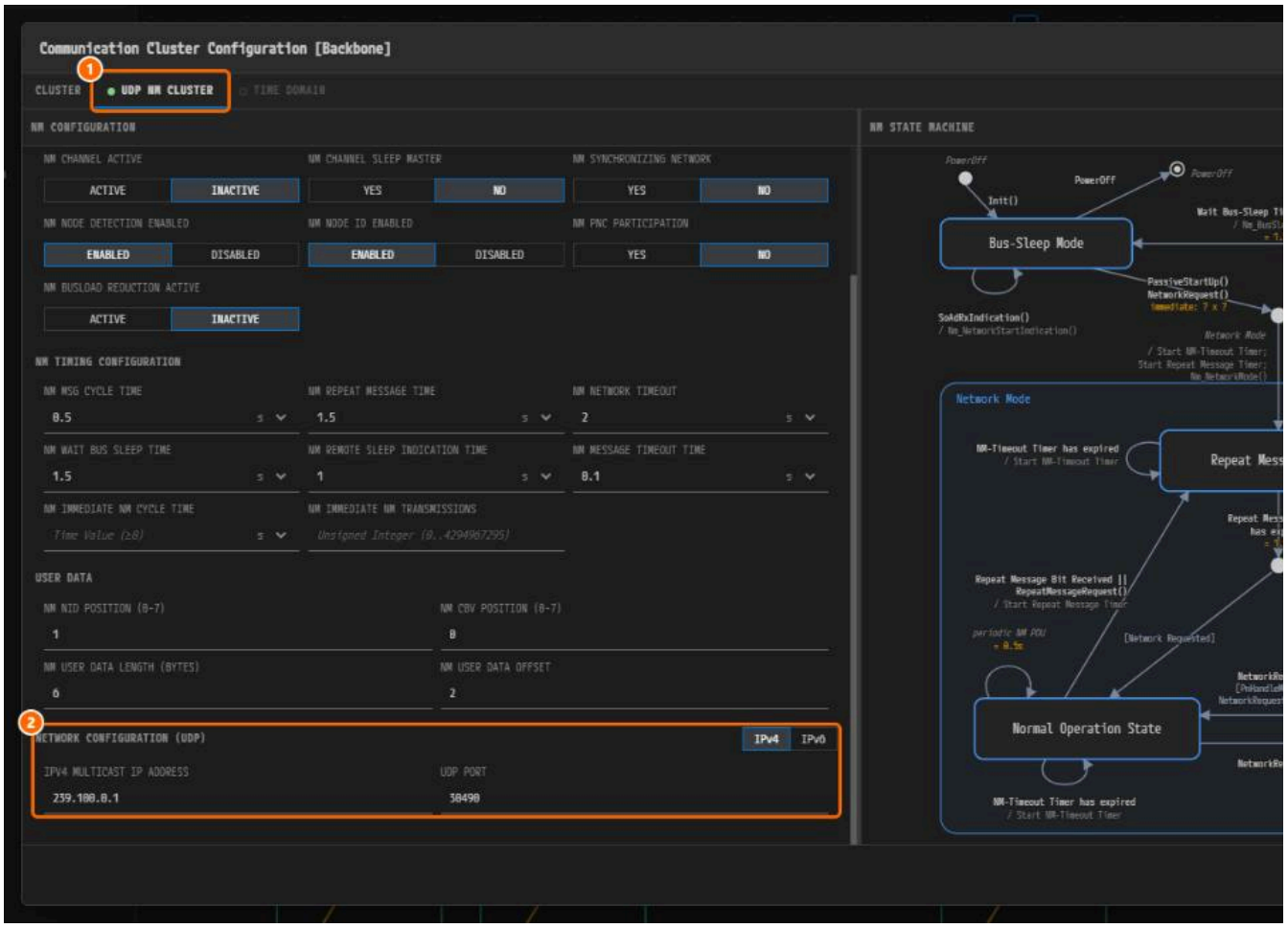


*그림 3-39. ① **UDP NM CLUSTER**, ② 폼 끝의 **NETWORK CONFIGURATION (UDP)**.*

2. **NETWORK CONFIGURATION (UDP)**에 **IPV4 MULTICAST IP ADDRESS** `239.100.0.1`, **UDP PORT** `30490`을 입력한다.

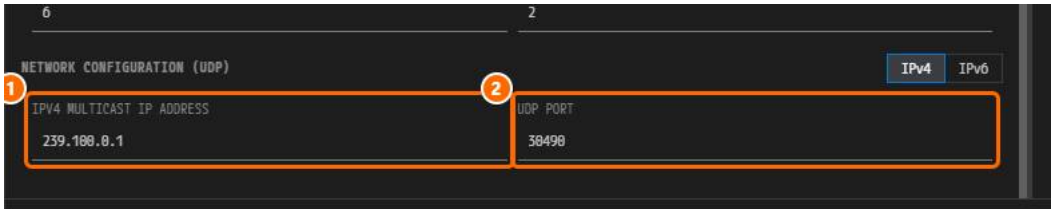


그림 3-40. ① `239.100.0.1`, ② `30490`.

3. UDP NM 노드를 ECU 대화상자에서 채운다(`HPC`는 **HPC Configuration [HPC]**로 열린다). **NM MSG CYCLE OFFSET**은 `0`이라도 입력한다(`constr_5223`).

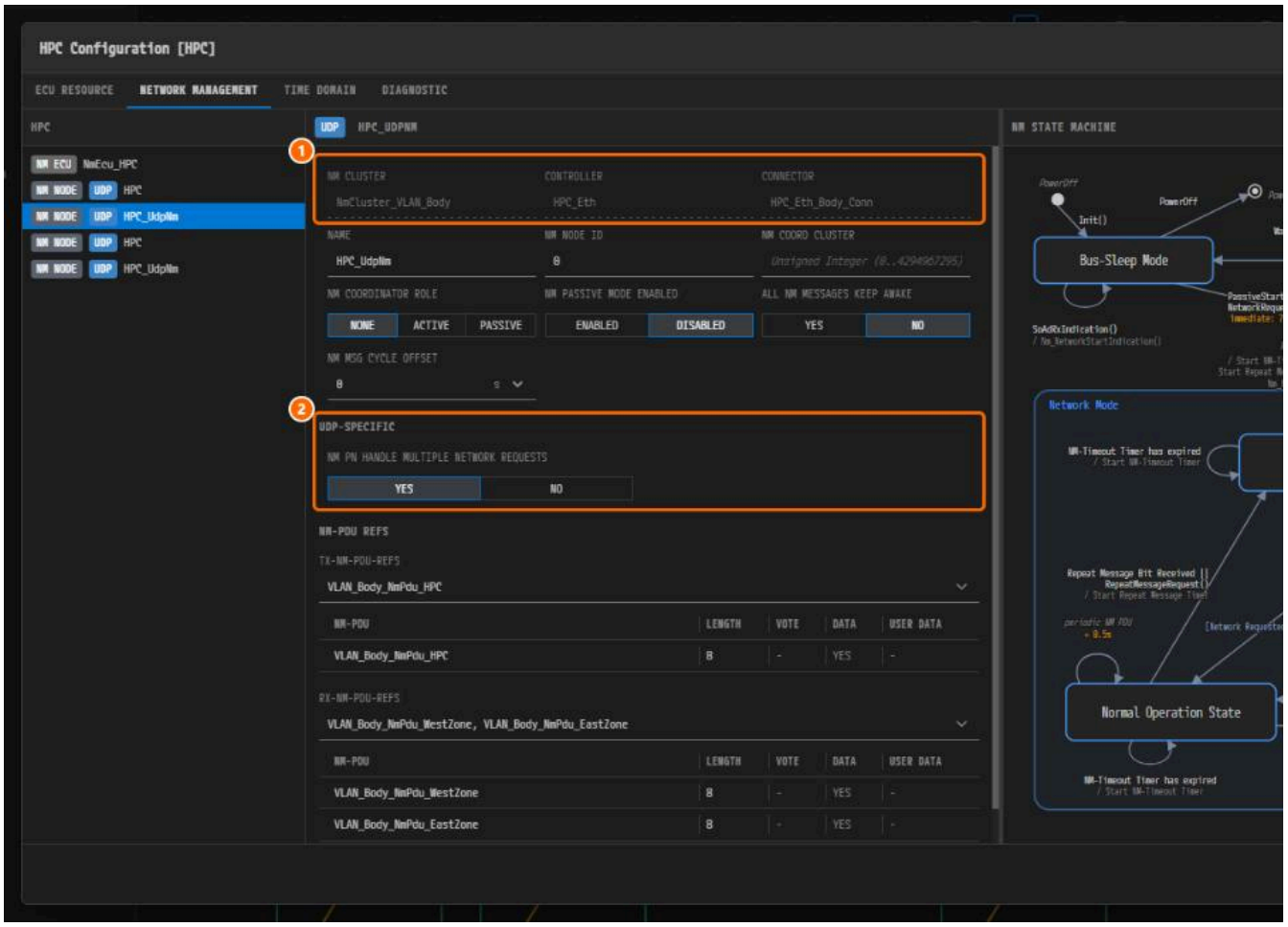


*그림 3-41. ① 읽기 전용 줄. **CONNECTOR**는 `HPC_Eth_Body_Conn`이다. ② **UDP-SPECIFIC: NM PN HANDLE MULTIPLE NETWORK REQUESTS = YES.***

> **주의.** `Backbone`을 더블클릭하면 언제나 `NmCluster_VLAN_Body`가 열린다. `NmCluster_VLAN_Rear`는 화면에서 고칠 수 없다.

전체 NM 값

시간 값은 모두 초(**s**)이다.

NM 클러스터	종류	통신 클러스터	NM PNC PARTICIPATION	MSG CYCLE / REPEAT MESSAGE / WAIT BUS SLEEP / NETWORK TIMEOUT	REMOTE SLEEP IND / MESSAGE TIMEOUT	NID POS / CBV POS / USER DATA OFFSET / LENGTH
`NmCluster_WestCAN`	CanNm	`WestCAN`	**YES**	`0.5` / `1.5` / `1.5` / `2`	`1` / `0.1`	`1` / `0` / `2` / `6`
`NmCluster_RearCAN`	CanNm	`RearCAN`	**YES**	같음	같음	같음
`NmCluster_EastCAN`	CanNm	`EastCAN`	**YES**	같음	같음	같음
`NmCluster_VLAN_Body`	UdpNm	`Backbone`	**YES**	같음	같음	같음
`NmCluster_VLAN_Rear`	UdpNm	`Backbone`	**YES**	같음	같음	같음

`NmCluster_VLAN_Rear`는 화면에서 열 수 없으므로 건너뛴다.

모든 클러스터: **NM NODE ID ENABLED** = **ENABLED**, **NM CHANNEL ID**는 비움. CanNm: **NM CHANNEL ACTIVE** = **ACTIVE**, **NM CHANNEL SLEEP MASTER** = **NO**, **NM NODE DE DETECTION ENABLED** = **ENABLED**.

NM 노드 이름은 표대로 바꾼다. 게이트웨이 노드가 첫 **NM NODE ID**를 갖는다. CAN은 `16`, Ethernet은 `0`부터이다.

NM 클러스터	NM 노드	NM NODE ID	NM COORDINATOR ROLE	NM MSG CYCLE OFFSET	TX-NM-PDU-REFS
NmCluster_WestCAN	WestZone_CanNm	16	ACTIVE	0	WestCAN_NmPdu_WestZone
NmCluster_WestCAN	DoorCtrl_FL_CanNm	17	NONE	0.05	WestCAN_NmPdu_DoorCtrl_FL
NmCluster_WestCAN	SeatCtrl_FL_CanNm	18	NONE	0.10	WestCAN_NmPdu_SeatCtrl_FL
NmCluster_WestCAN	MirrorAct_FL_CanNm	19	NONE	0.15	WestCAN_NmPdu_MirrorAct_FL
NmCluster_WestCAN	WindowSw_FL_CanNm	20	NONE	0.20	WestCAN_NmPdu_WindowSw_FL
NmCluster_RearCAN	SouthZone_CanNm	16	ACTIVE	0	RearCAN_NmPdu_SouthZone
NmCluster_RearCAN	ParkSensor_RR_CanNm	17	NONE	0.05	RearCAN_NmPdu_ParkSensor_RR
NmCluster_RearCAN	TrailerModule_CanNm	18	NONE	0.10	RearCAN_NmPdu_TrailerModule
NmCluster_RearCAN	TailLamp_RR_CanNm	19	NONE	0.15	RearCAN_NmPdu_TailLamp_RR
NmCluster_EastCAN	EastZone_CanNm	16	ACTIVE	0	EastCAN_NmPdu_EastZone
NmCluster_EastCAN	DoorCtrl_FR_CanNm	17	NONE	0.05	EastCAN_NmPdu_DoorCtrl_FR
NmCluster_EastCAN	SeatCtrl_FR_CanNm	18	NONE	0.10	EastCAN_NmPdu_SeatCtrl_FR
NmCluster_EastCAN	MirrorAct_FR_CanNm	19	NONE	0.15	EastCAN_NmPdu_MirrorAct_FR
NmCluster_VLAN_Body	HPC_UdpNm	0	NONE	0	VLAN_Body_NmPdu_HPC
NmCluster_VLAN_Body	WestZone_UdpNm	1	ACTIVE	0	VLAN_Body_NmPdu_WestZone
NmCluster_VLAN_Body	EastZone_UdpNm	2	PASSIVE	0	VLAN_Body_NmPdu_EastZone
NmCluster_VLAN_Rear	HPC_UdpNm	0	NONE	0	VLAN_Rear_NmPdu_HPC
NmCluster_VLAN_Rear	SouthZone_UdpNm	1	ACTIVE	0	VLAN_Rear_NmPdu_SouthZone

NM NODE ID, **NM COORDINATOR ROLE**, **NM MSG CYCLE OFFSET**은 지금 입력한다. 4장을 마친 뒤 **TX-NM-PDU-REFS**는 표대로, **RX-NM-PDU-REFS**에는 같은 클러스터의 나머

왜 이렇게 설계하는가

- **PNC 17, 18, 19.** 식별자는 NM PDU 안의 절대 비트 번호이다. 오프셋 `2`, 길이 `6`이면 비트 `16`..`63`이고, 17~19는 모두 한 바이트에 들어간다.
- **웨이크업 ID와 마스크.** (ID AND `0x700`) = `0x500`인 프레임이 버스를 깨운다. NM 프레임(`0x510` + 노드 ID)은 이 범위에 든다.
- **게이트웨이는 커넥터에.** 같은 존 ECU라도 CAN 커넥터와 Ethernet 커넥터에서 역할이 다를 수 있다.
- **VLAN마다 활성 게이트웨이 하나.** 토폴로지 설계를 본다.
- **두 곳에 기록.** 커넥터는 ECU가 PNC를 어떻게 다루는지를, PNC 다이얼로그 — PNC별 바인딩의 **PHYSICAL CHANNELS**와 **MATRIX**는 PNC가 어느 채널에 걸치는지를 말한다.

확인하기

- [ ] PNC 대화상자에 **PNC 17 PNC_Comfort**, **PNC 18 PNC_ParkAssist**, **PNC 19 PNC_Trailer**가 있다.
- [ ] **PNC VECTOR OFFSET**이 `2`, **PNC VECTOR LENGTH**가 `6`이다.
- [ ] `WestZone_CanFd_Conn`이 **ACTIVE**이고 `0x500` / `0x700` / `8`, Byte 0 = `0x02`이다. **PASSIVE**는 `EastZone_Eth_Conn`뿐이다.
- [ ] `BackboneSwitch`의 `Port_0`에 PNC 세 개가 모두 있다.
- [ ] `NmCluster_WestCAN`의 **NM PNC PARTICIPATION**이 **YES**이고, `WestZone_CanNm`의 **NM NODE ID**가 `16`이다.
- [ ] (4장을 마친 뒤) `WestZone_CanNm`에 TX NM-PDU 참조 1개와 RX 4개가 있다.
- [ ] 탭 **PNC:System**에 수정 표시가 없다.

다음 단계

시그널, PDU, 프레임, I-PDU 그룹에서 시그널, PDU, I-PDU 그룹을 만든 뒤 NM 설정으로 돌아와 NM-PDU 참조를 채운다.

신호·PDU·프레임과 I-PDU Group

이 장에서 할 일

PNC가 켜고 끄는 대상은 I-Signal I-PDU를 묶은 I-PDU Group이다. 이 장에서는 Signal 편집기에서 CAN 프레임, I-Signal I-PDU, 신호, NM-PDU를 만든다. 이어서 PNC 대화상자에

시작하기 전에

- PNC 정의와 커넥터 PNC 속성을 마친 뒤 시작한다. PNC 17, 18, 19가 있어야 한다.
- 한 세트를 단계별로 만든다. 프레임 `WestCAN_DoorStatus_FL`, 그 PDU `DoorStatus_FL_Pdu`, 신호 3개, NM-PDU `WestCAN_NmPdu_WestZone`, 그룹 `WestCAN_PNC`
- 절차 그림은 다른 프레임이 이미 있는 프로젝트의 화면이라 항목 수가 독자의 화면과 다르다. 라벨과 값은 같다.
- Signal 편집기는 NM 프레임을 만들 수 없다. 직접 만든 프로젝트는 프레임 29개가 아니라 응용 프레임 16개로 끝난다(NM-PDU 만들기 참조).

항목	완성 예제
CAN 프레임	29개(응용 16 + NM 13)
I-Signal I-PDU	16개
NM-PDU	18개
신호	42개
I-PDU Group	`PNC_Comfort` 4개, `PNC_ParkAssist` 2개, `PNC_Trailer` 2개

Signal 편집기 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System**을 클릭한다. 캔버스 왼쪽 가장자리 아이콘 줄에서 위쪽 아이콘 **Signal**을 클릭한다.

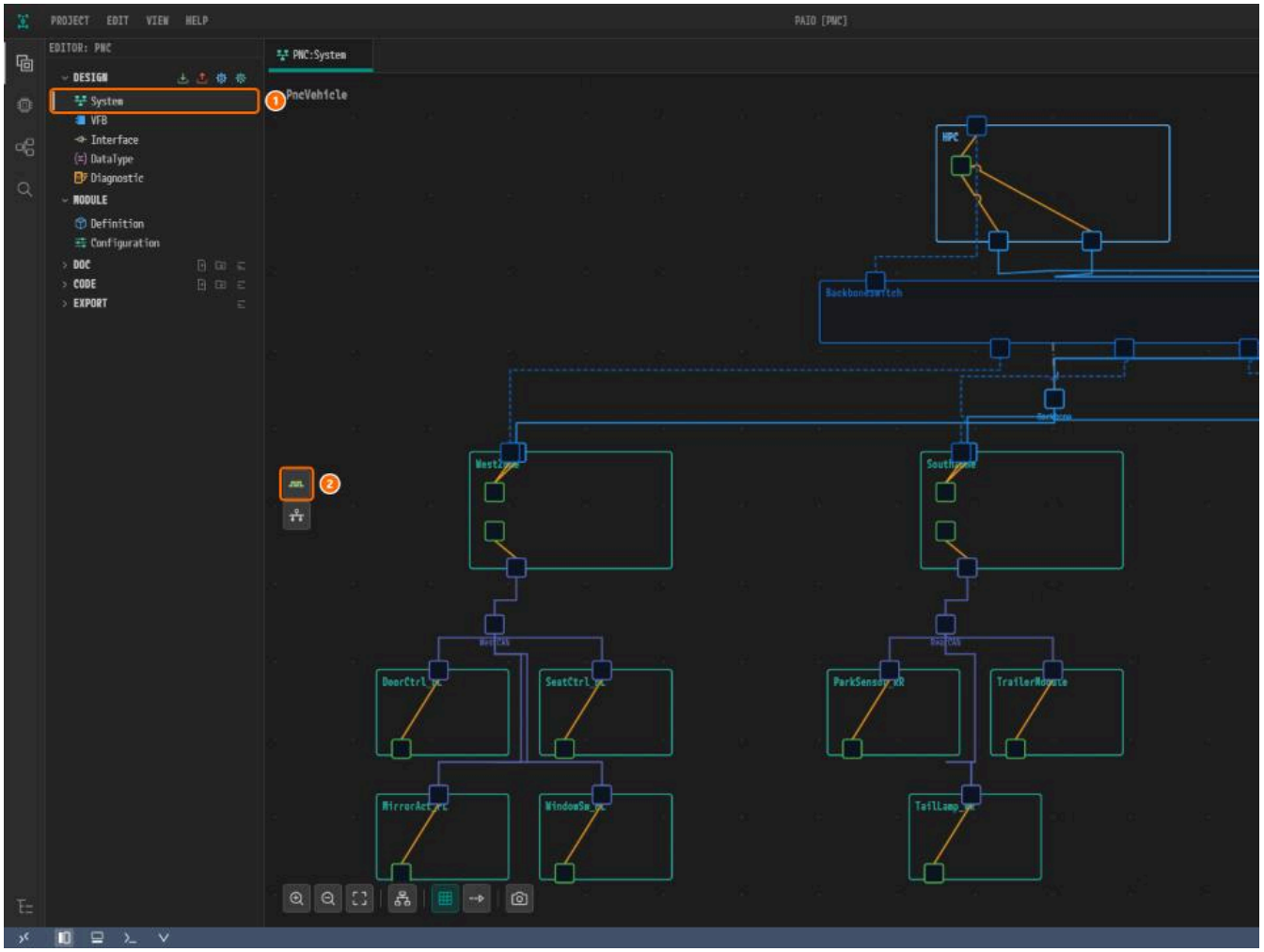


*그림 4-1. ① **System**, ② **Signal**. 그 아래 아이콘은 **Partial Network (PNC)*이다.*

2. 탭 **PNC:Signal**이 열리고 위쪽에 **CAN**, **LIN**, **PDU**, **SIGNAL** 탭이 있다. **CAN**이 선택되어 있다.

왼쪽에 목록, 가운데에 **CONFIGURATION**과 **SIGNALS** 표, 오른쪽에 **PREVIEW**와 **PDU REFERENCES** 탭이 있다.

CAN 프레임과 PDU 만들기

1. **CAN** 탭에서 **CAN FRAMES** 머리글의 **+**(**Add**)를 클릭한다.

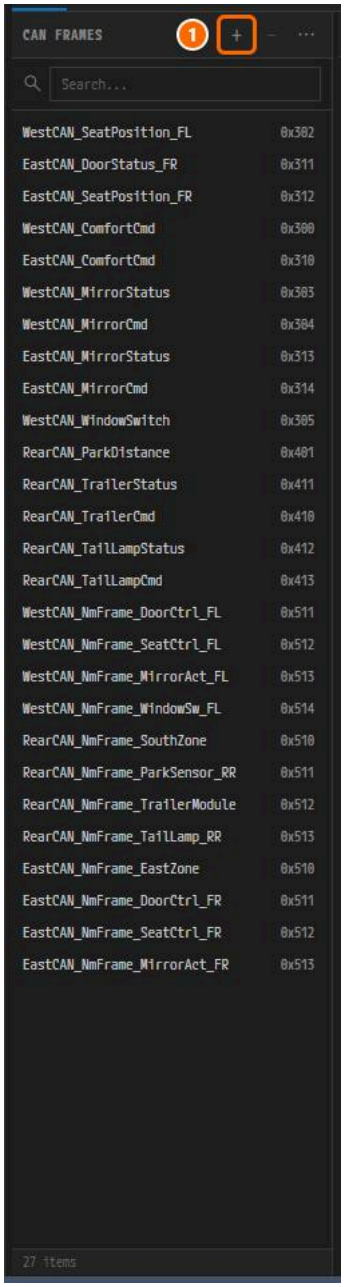


*그림 4-2. ① **+**.*

2. `NewFrame`이 생기고 선택된다. I-Signal I-PDU `NewFrame_Pdu`가 함께 만들어진다. **PDU REFERENCES** 탭을 클릭하면 보인다.

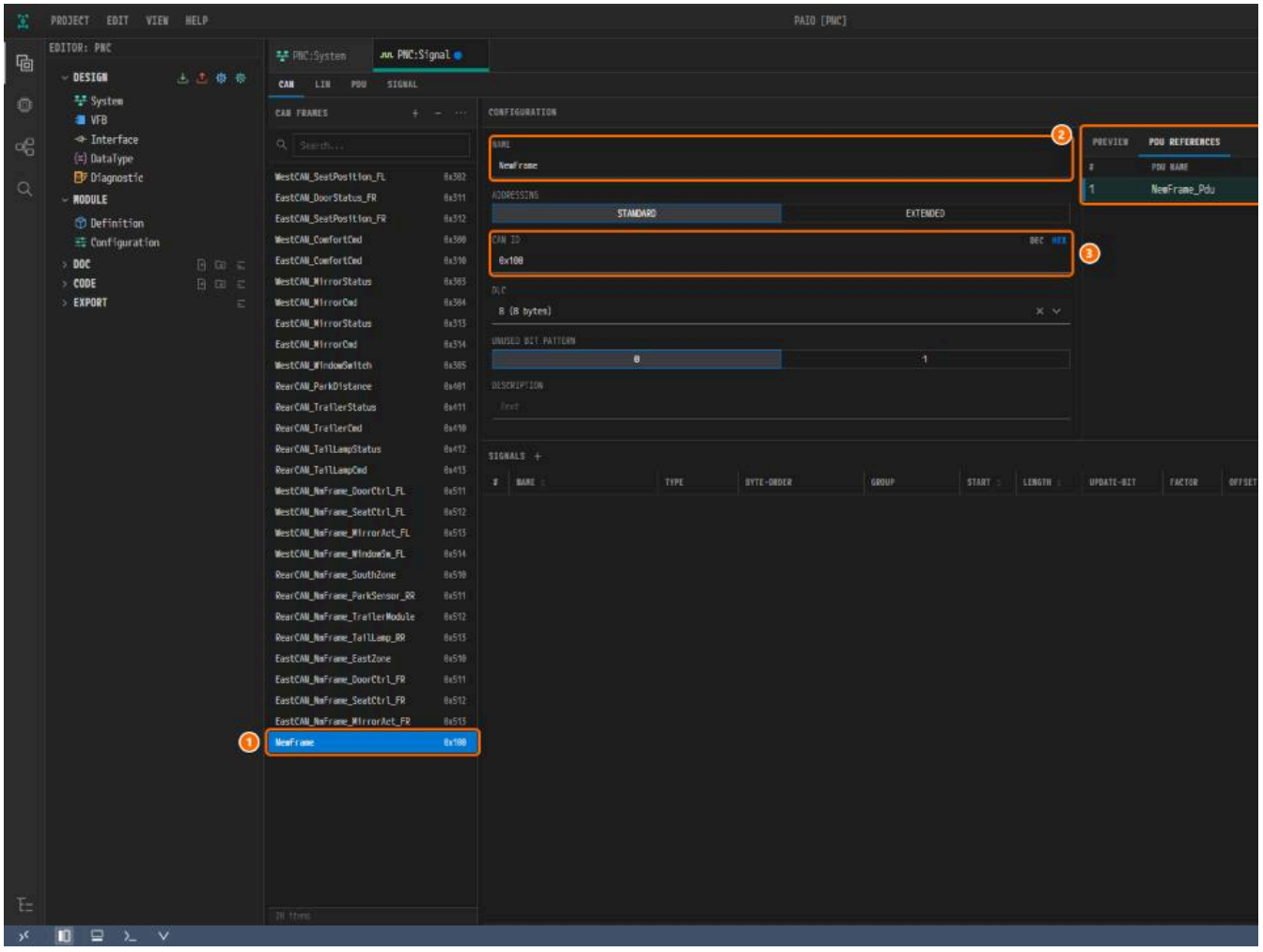


*그림 4-3. ① `NewFrame`, ② **NAME**, ③ **CAN ID**, ④ `NewFrame_Pdu`.*

3. **NAME**에 `WestCAN_DoorStatus_FL`, **CAN ID**에 `0x301`을 입력하고 각각 `Enter`를 누른다. **ADDRESSING** **STANDARD**, **DLC** `8 (8 bytes)`, **UNUSED BIT PATTERN** `0`

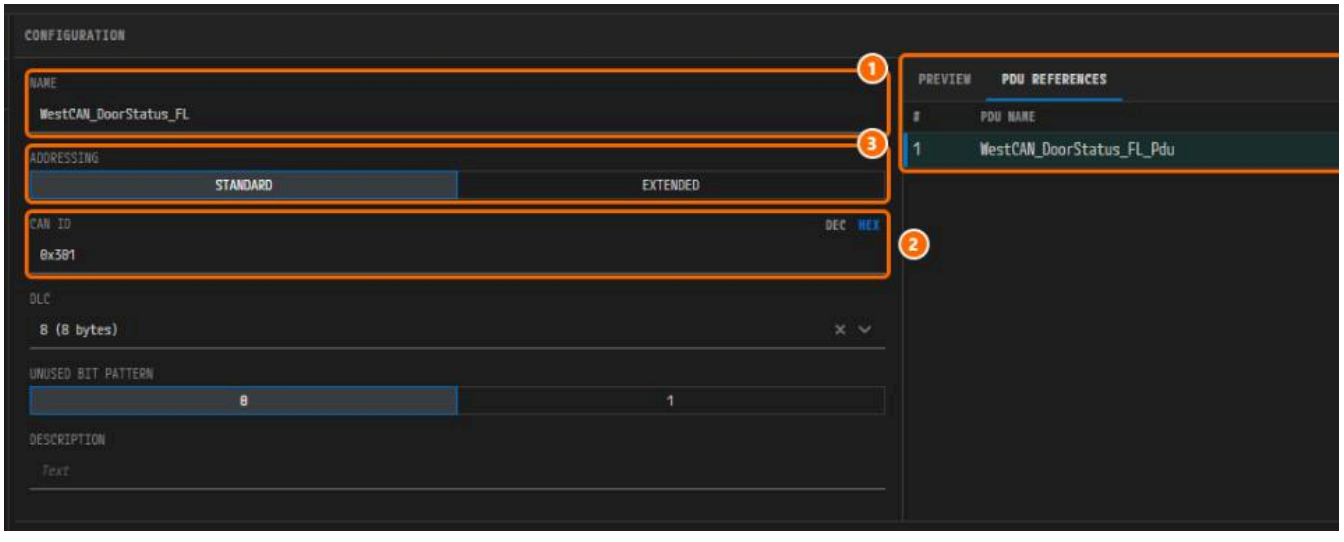


*그림 4-4. ① **NAME**, ② **CAN ID**, ③ **ADDRESSING**, ④ `WestCAN_DoorStatus_FL_Pdu`가 있는 **PDU REFERENCES**.*

CAN ID 라벨 옆의 **DEC** | **HEX**는 표기만 바꾼다.

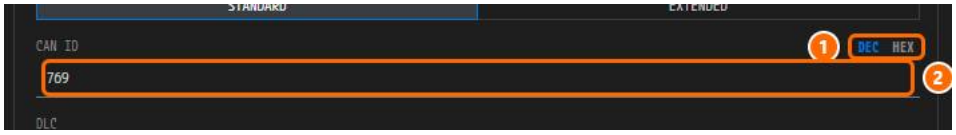


*그림 4-5. ① **DEC | HEX**, ② **DEC**로 본 `769`.*

4. **PDU** 탭을 클릭하고 `WestCAN_DoorStatus_FL_Pdu`(**IS** 태그)를 클릭한다.

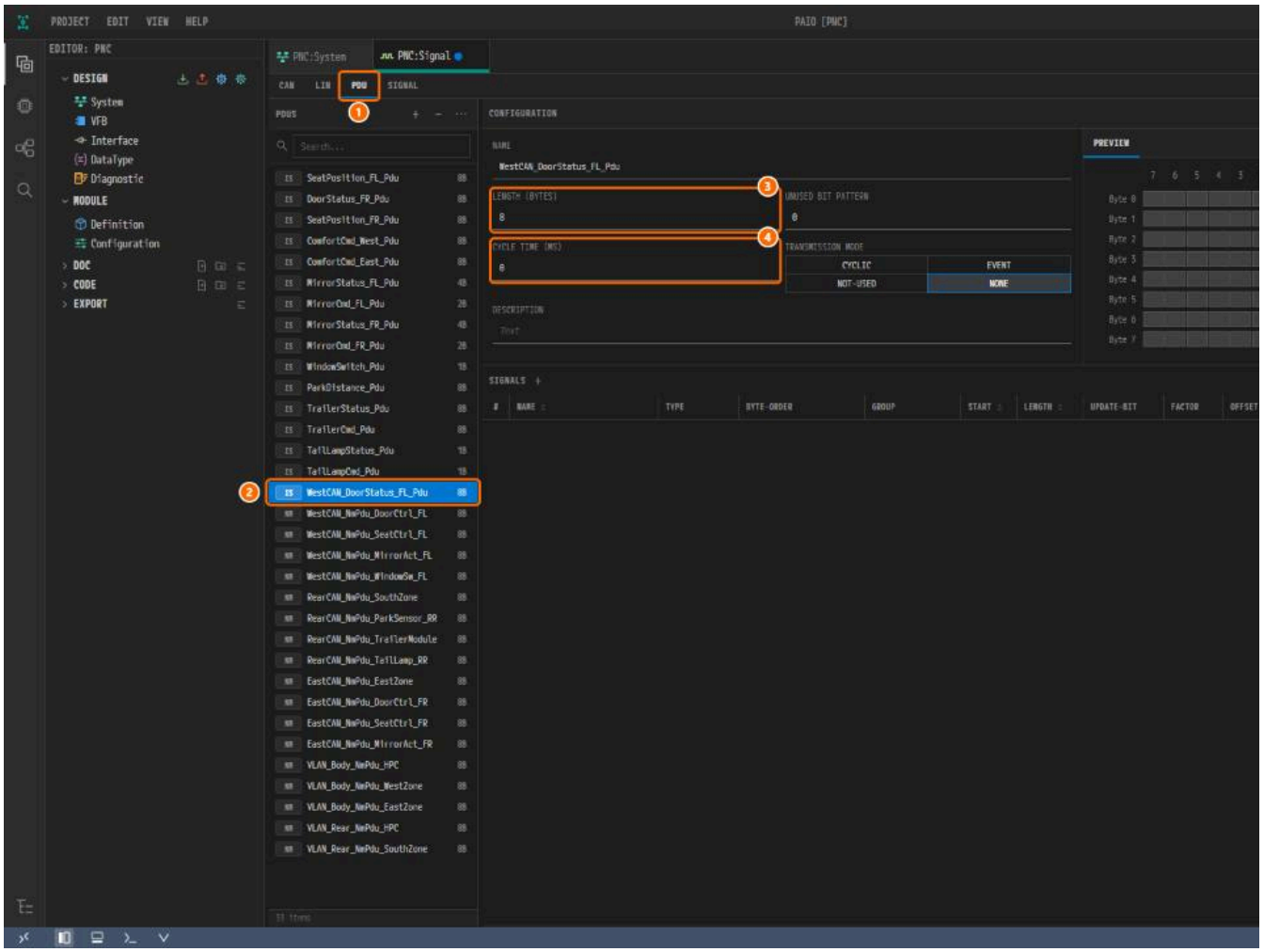


*그림 4-6. ① **PDU**, ② `WestCAN_DoorStatus_FL_Pdu`, ③ **LENGTH (BYTES)**, ④ **CYCLE TIME (MS)**.*

5. **NAME**에 `DoorStatus_FL_Pdu`, **CYCLE TIME (MS)**에 `50`을 입력한다. **LENGTH (BYTES)**는 `8` 그대로 둔다. **TRANSMISSION MODE**에서 **CYCLIC**을 선택한다.

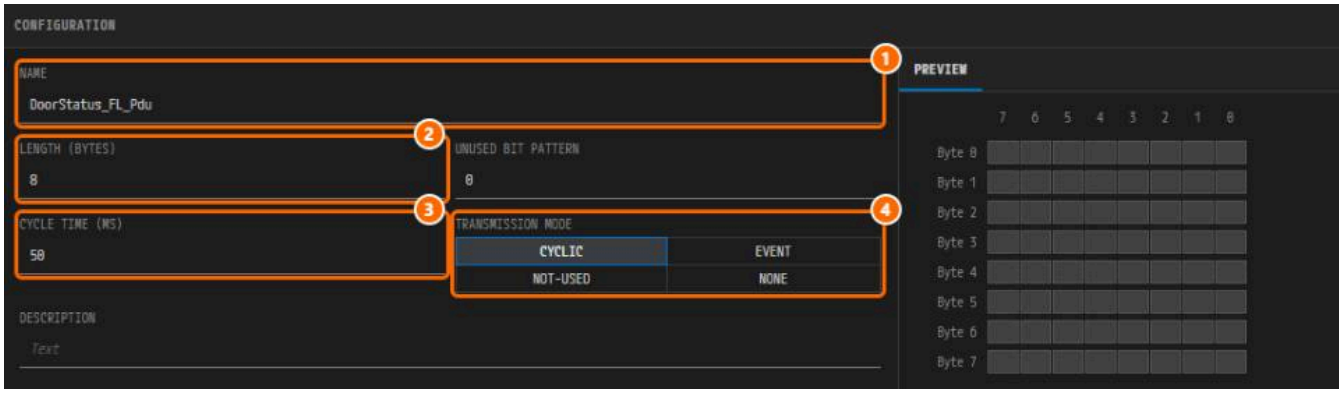


*그림 4-7. ① **NAME**, ② **LENGTH (BYTES)**, ③ **CYCLE TIME (MS)**, ④ **TRANSMISSION MODE**.*

6. `Ctrl+S`로 저장한다.

신호 만들어 PDU에 매핑하기

1. **CAN** 탭에서 `WestCAN_DoorStatus_FL`을 클릭한다.

2. **SIGNALS** 제목 옆의 **+(Add Signal)**를 클릭한다. 신호 `WestCAN_DoorStatus_FL_Sig`가 만들어져 프레임의 PDU에 매핑된다.



그림 4-8. ① +, ② 새 행.

3. 칸을 클릭해 값을 입력하고 `Enter`를 누른다. **NAME** `DoorOpen_FL`, **LENGTH** `1`, **MAX** `0`으로 고친다. **LENGTH**를 **START**보다 먼저 고쳐야 비트가 겹치지 않는다.

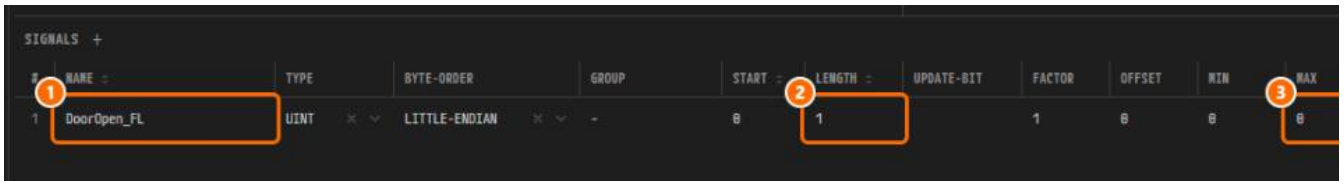


*그림 4-9. ① **NAME**, ② **LENGTH**, ③ **MAX**.*

4. 나머지 신호 2개도 같은 방법으로 **+**를 누르고 새 행을 고친다. **PREVIEW** 탭을 클릭해 배치를 확인한다.

NAME	LENGTH	START	MAX
`DoorOpen_FL`	`1`	`0` (자동값)	`0`
`DoorLock_FL`	`2`	`1` (자동값)	`0`
`WindowPos_FL`	`8`	`8` (직접 입력)	`0`

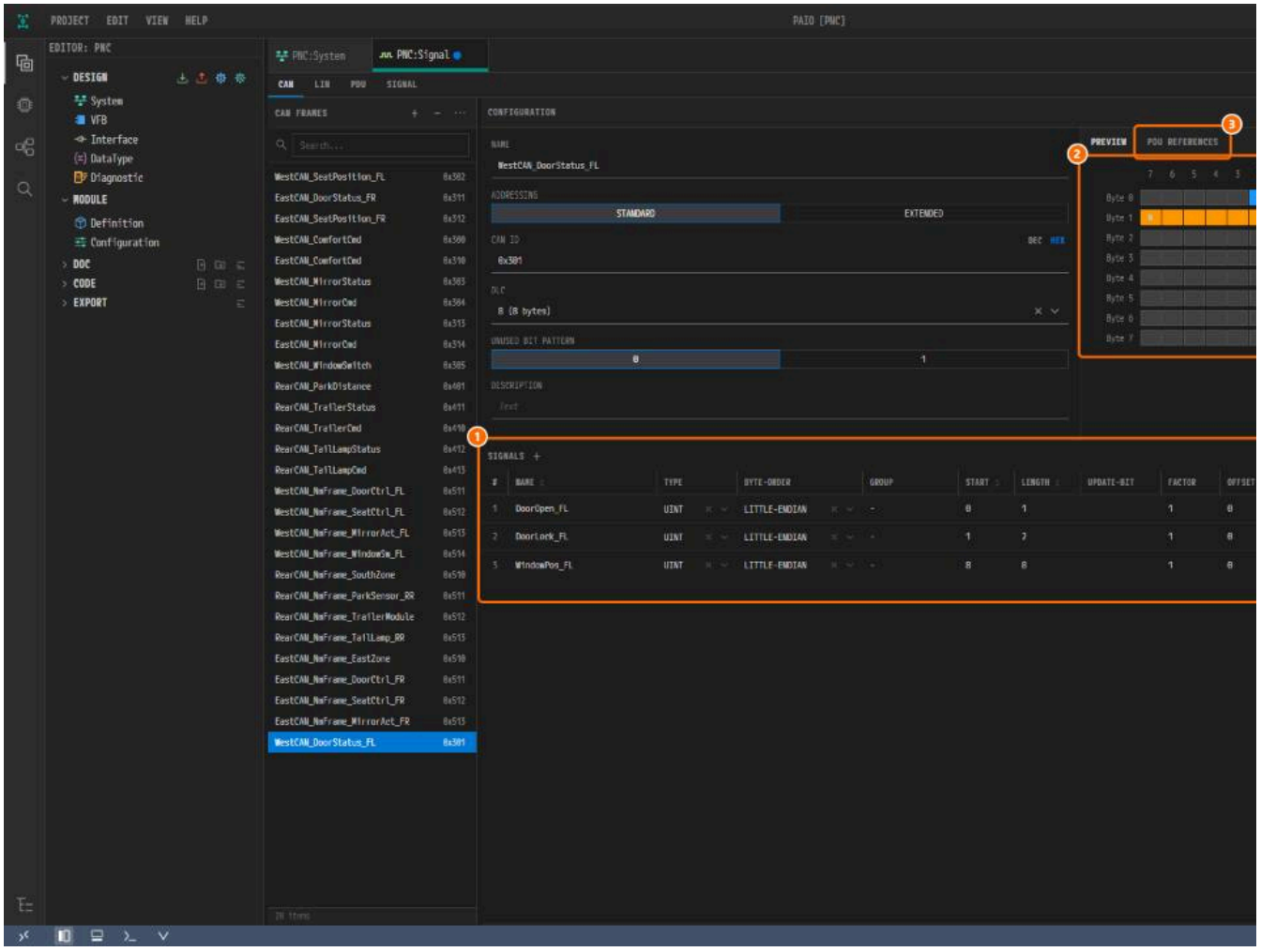


*그림 4-10. ① 3개 행, ② **PREVIEW**, ③ **SIGNALS** `3`인 **PDU REFERENCES**.*

5. `Ctrl+S`로 저장한다.

> **주의.** 새 신호는 **MAX** `255`이고, **START**는 비어 있는 첫 비트가 아니라 마지막 신호 바로 다음이다. **MAX**는 직접 `0`으로 고치고, `WindowPos_FL`의 **START**는 `8`로 입력한다.

NM-PDU 만들기

1. **PDU** 탭에서 **PDUS** 머리글의 **+(Add)**를 클릭한다. **Network Mgmt**에 포인터를 올리고 **NM-PDU**를 클릭한다.

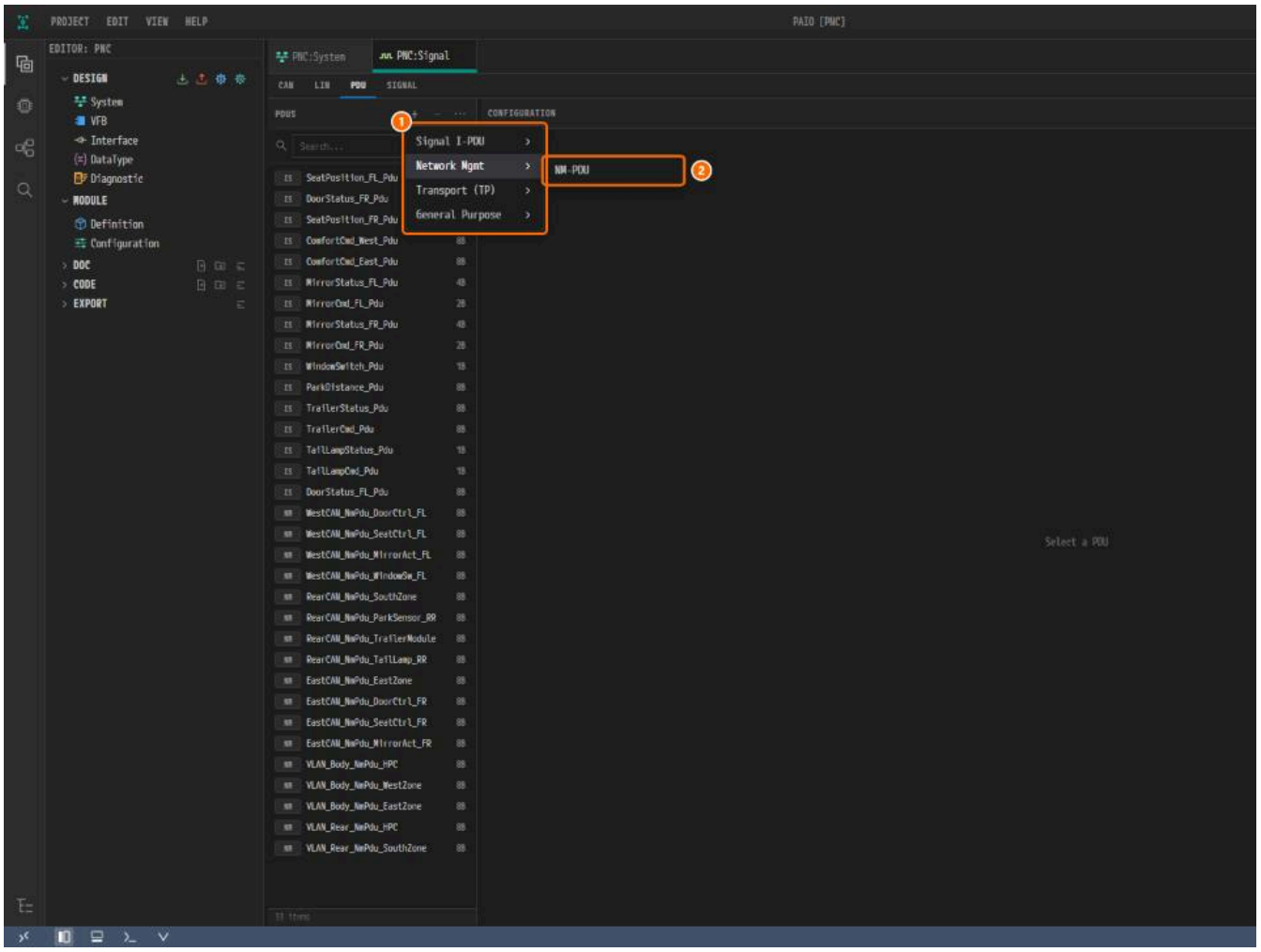


*그림 4-11. ① 메뉴 묶음, ② **NM-PDU**.*

2. `NewNmPdu`가 만들어지고 선택된다.

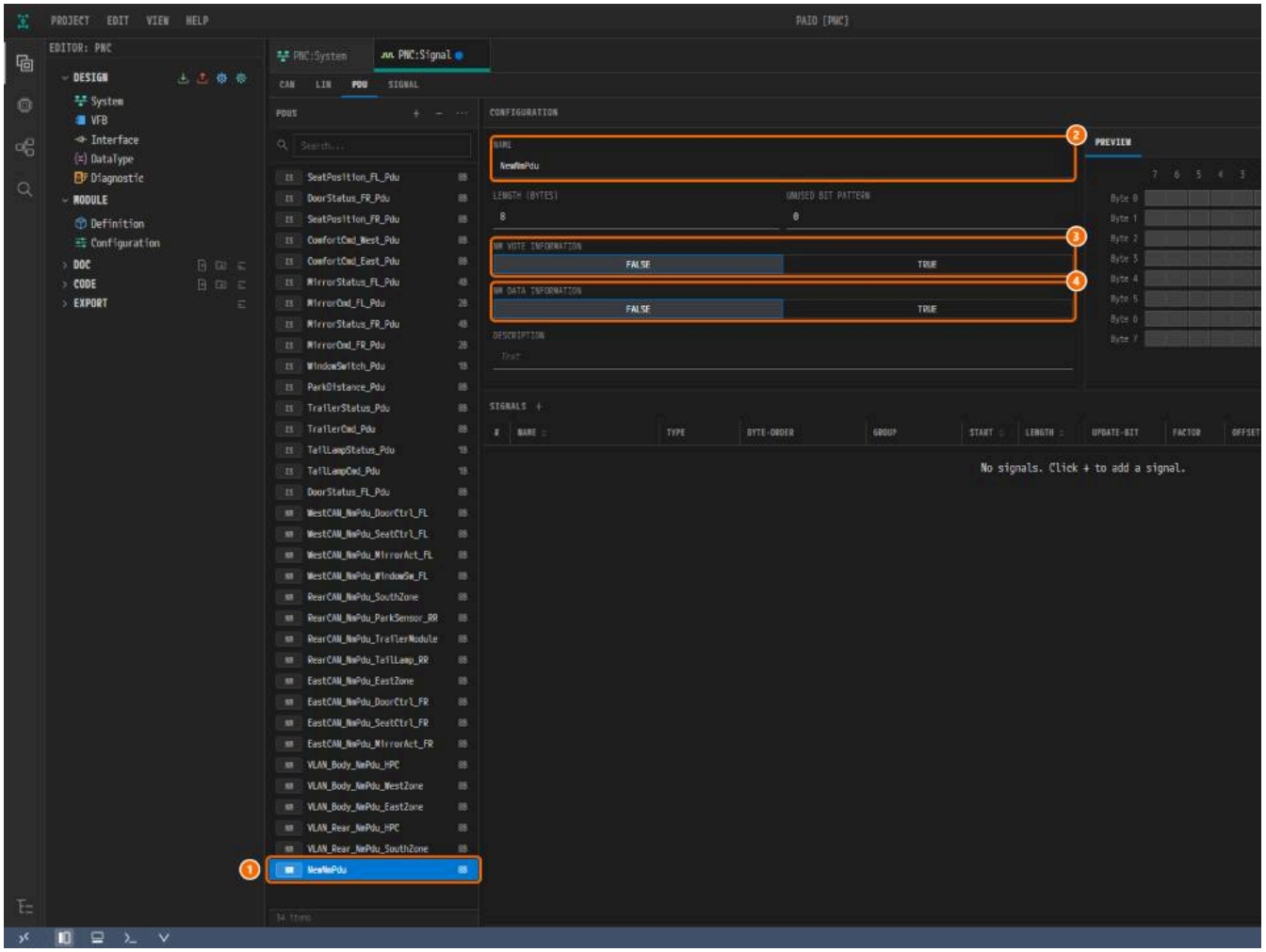


*그림 4-12. ① `NewNmPdu`, ② **NAME**, ③ **NM VOTE INFORMATION**, ④ **NM DATA INFORMATION**.*

3. **NAME**에 `WestCAN_NmPdu_WestZone`을 입력하고 **NM DATA INFORMATION**을 **TRUE**로 바꾼다. **LENGTH (BYTES)** `8`, **NM VOTE INFORMATION FALSE**는 그대로 둔다.



*그림 4-13. ① **NAME**, ② **NM DATA INFORMATION**, ③ **LENGTH (BYTES)**.*

4. 나머지 NM-PDU 17개를 표를 보고 같은 설정으로 만들고 `Ctrl+S`로 저장한다.

> **참고.** 이 18개를 저장하면 NM 설정에서 미뤄 둔 단계를 할 수 있다. 이 장을 마친 뒤 돌아가 전체 NM 값을 보고 NM 노드마다 **TX-NM-PDU-REFS**와 **RX-NM-PDU-REFS**를 채

클러스터	NM-PDU
WestCAN	`WestCAN_NmPdu_WestZone`, `WestCAN_NmPdu_DoorCtrl_FL`, `WestCAN_NmPdu_SeatCtrl_FL`, `WestCAN_NmPdu_MirrorAct_FL`, `WestCAN_NmPdu_WindowSw_FL`
RearCAN	`RearCAN_NmPdu_SouthZone`, `RearCAN_NmPdu_ParkSensor_RR`, `RearCAN_NmPdu_TrailerModule`, `RearCAN_NmPdu_TailLamp_RR`
EastCAN	`EastCAN_NmPdu_EastZone`, `EastCAN_NmPdu_DoorCtrl_FR`, `EastCAN_NmPdu_SeatCtrl_FR`, `EastCAN_NmPdu_MirrorAct_FR`
VLAN_Body	`VLAN_Body_NmPdu_HPC`, `VLAN_Body_NmPdu_WestZone`, `VLAN_Body_NmPdu_EastZone`
VLAN_Rear	`VLAN_Rear_NmPdu_HPC`, `VLAN_Rear_NmPdu_SouthZone`

> **주의.** Signal 편집기는 NM 프레임을 만들 수 없다. CAN 프레임의 **PDU REFERENCES** 목록에 NM-PDU가 나오지 않기 때문이다. 예제의 NM 프레임 13개는 예제 프로젝트에 들어 필요 없으므로 NM 프레임 없이 진행한다.

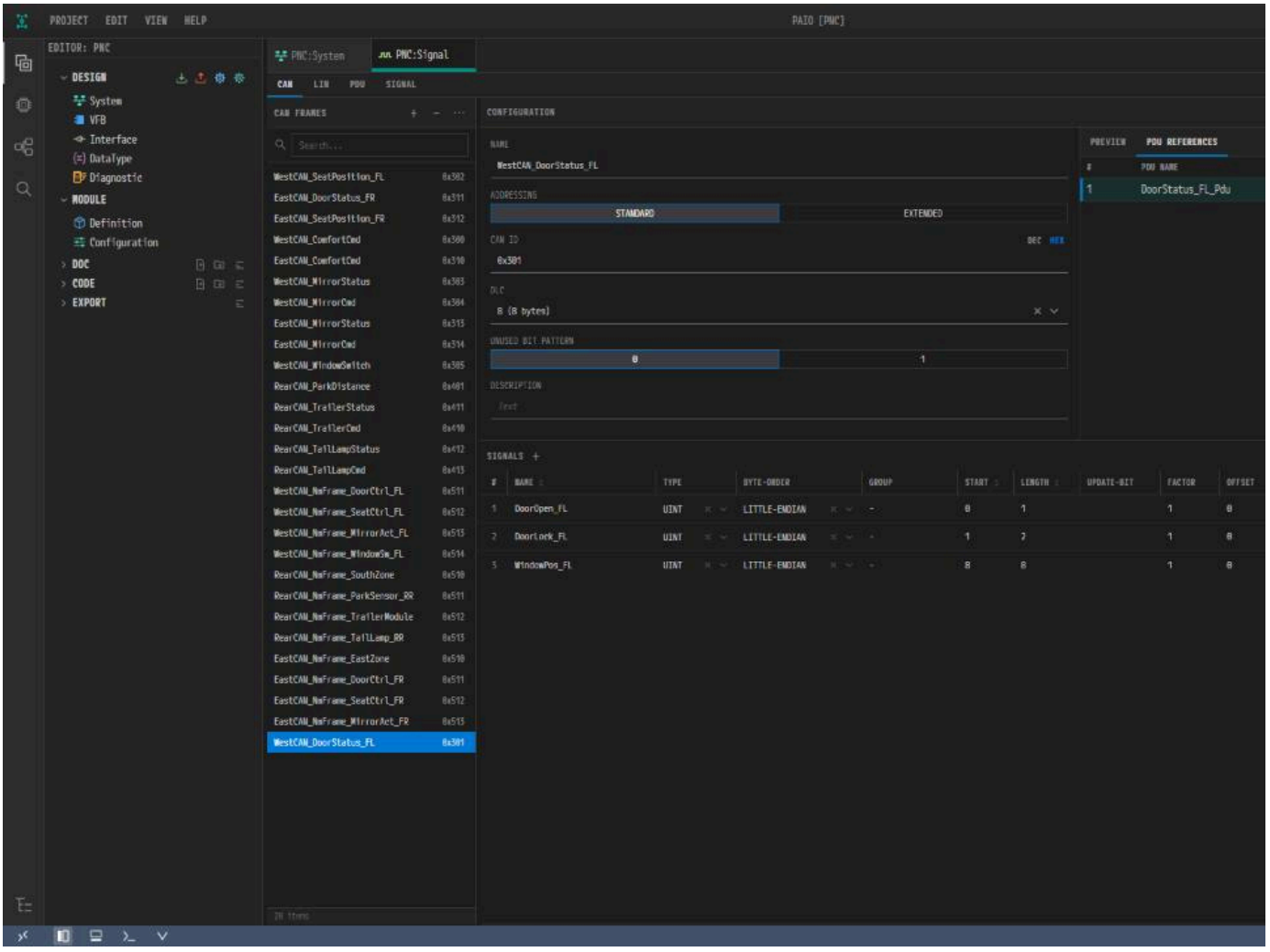


*그림 4-14. ① **PDU REFERENCES** 탭의 선택 상자, ② 후보 목록. I-Signal I-PDU만 있고 NM-PDU는 없다.*

나머지 프레임, PDU, 신호

나머지 프레임 15개를 표를 보고 같은 방법으로 만든다. 프레임 이름과 **CAN ID**, PDU 이름·길이·주기, 신호 순서이다.

- 프레임: **ADDRESSING** **STANDARD**, **DLC** `8 (8 bytes)`, **UNUSED BIT PATTERN** `0`이다. PDU가 더 짧아도 같다.
- PDU: **TRANSMISSION MODE** **CYCLIC**이다. 기본값이 **NONE**이므로 PDU마다 고친다.
- 신호: **MAX** `0`이다. 나머지 필드는 기본값 그대로 둔다.

프레임	CAN ID	PDU	PDU 길이(B)	주기(ms)	신호 (START, LENGTH)
WestCAN_ComfortCmd	0x300	ComfortCmd_West_Pdu	8	20	LockCmd_FL (0, 2), WindowCmd_FL (2, 4), SeatMoveCmd_FL (8, 4), SeatHeatCmd
WestCAN_DoorStatus_FL	0x301	DoorStatus_FL_Pdu	8	50	DoorOpen_FL (0, 1), DoorLock_FL (1, 2), WindowPos_FL (8, 8)
WestCAN_SeatPosition_FL	0x302	SeatPosition_FL_Pdu	8	100	SeatPos_FL (0, 8), SeatHeat_FL (8, 2)
WestCAN_MirrorStatus	0x303	MirrorStatus_FL_Pdu	4	100	MirrorFoldSts_FL (0, 2), MirrorTiltX_FL (8, 8), MirrorTiltY_FL (16, 8)
WestCAN_MirrorCmd	0x304	MirrorCmd_FL_Pdu	2	100	MirrorFoldCmd_FL (0, 2), MirrorMoveCmd_FL (2, 4)
WestCAN_WindowSwitch	0x305	WindowSwitch_Pdu	1	100	WindowSwUp (0, 1), WindowSwDown (1, 1), WindowSwAuto (2, 1)
EastCAN_ComfortCmd	0x310	ComfortCmd_East_Pdu	8	20	LockCmd_FR (0, 2), WindowCmd_FR (2, 4), SeatMoveCmd_FR (8, 4), SeatHeatCmd
EastCAN_DoorStatus_FR	0x311	DoorStatus_FR_Pdu	8	50	DoorOpen_FR (0, 1), DoorLock_FR (1, 2), WindowPos_FR (8, 8)
EastCAN_SeatPosition_FR	0x312	SeatPosition_FR_Pdu	8	100	SeatPos_FR (0, 8), SeatHeat_FR (8, 2)
EastCAN_MirrorStatus	0x313	MirrorStatus_FR_Pdu	4	100	MirrorFoldSts_FR (0, 2), MirrorTiltX_FR (8, 8), MirrorTiltY_FR (16, 8)
EastCAN_MirrorCmd	0x314	MirrorCmd_FR_Pdu	2	100	MirrorFoldCmd_FR (0, 2), MirrorMoveCmd_FR (2, 4)
RearCAN_ParkDistance	0x401	ParkDistance_Pdu	8	50	ParkDist_1 (0, 8), ParkDist_2 (8, 8), ParkDist_3 (16, 8), ParkDist_4 (24, 8),
RearCAN_TrailerCmd	0x410	TrailerCmd_Pdu	8	50	TrailerLampCmd (0, 8)
RearCAN_TrailerStatus	0x411	TrailerStatus_Pdu	8	100	TrailerAttached (0, 1), TrailerLampFault (1, 4), TrailerBrakeLamp (5, 1)
RearCAN_TailLampStatus	0x412	TailLampStatus_Pdu	1	100	TailLampSts (0, 4)
RearCAN_TailLampCmd	0x413	TailLampCmd_Pdu	1	100	TailLampCmd (0, 4)

`Ctrl+S`로 저장한다.

CAN 프레임 살펴보기

여기부터의 그림은 완성 예제이다.

1. **CAN** 탭에서 `WestCAN_DoorStatus_FL`을 클릭하고 **PREVIEW** 탭을 클릭한다.

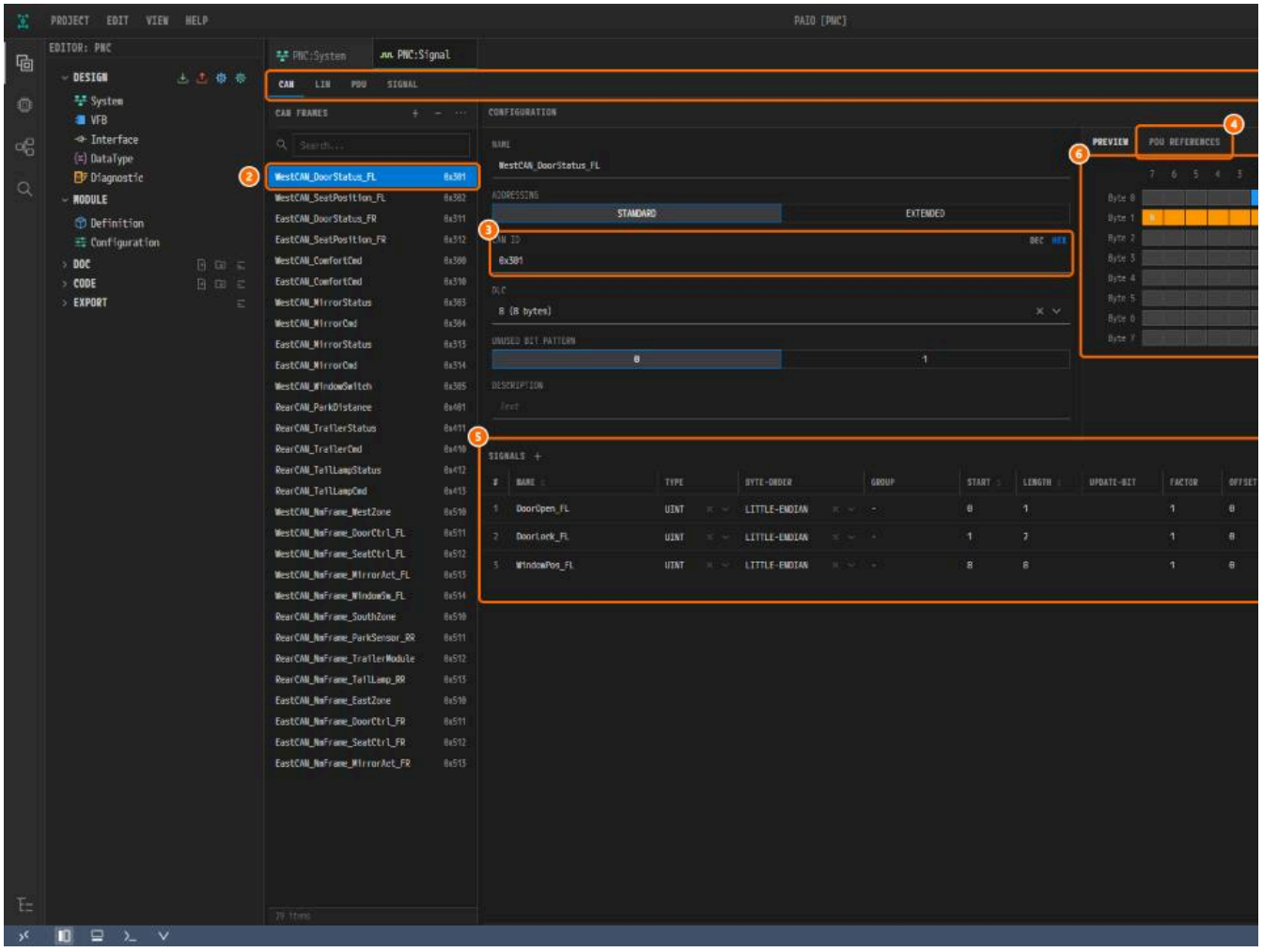


*그림 4-15. ① 프로토콜 탭, ② `WestCAN_DoorStatus_FL`, ③ **CAN ID**, ④ **PDU REFERENCES**, ⑤ **SIGNALS**, ⑥ **PREVIEW**.*

2. **PDU REFERENCES** 탭을 클릭한다. `DoorStatus_FL_Pdu`가 **LENGTH** `8`, **SIGNALS** `3`으로 있다.

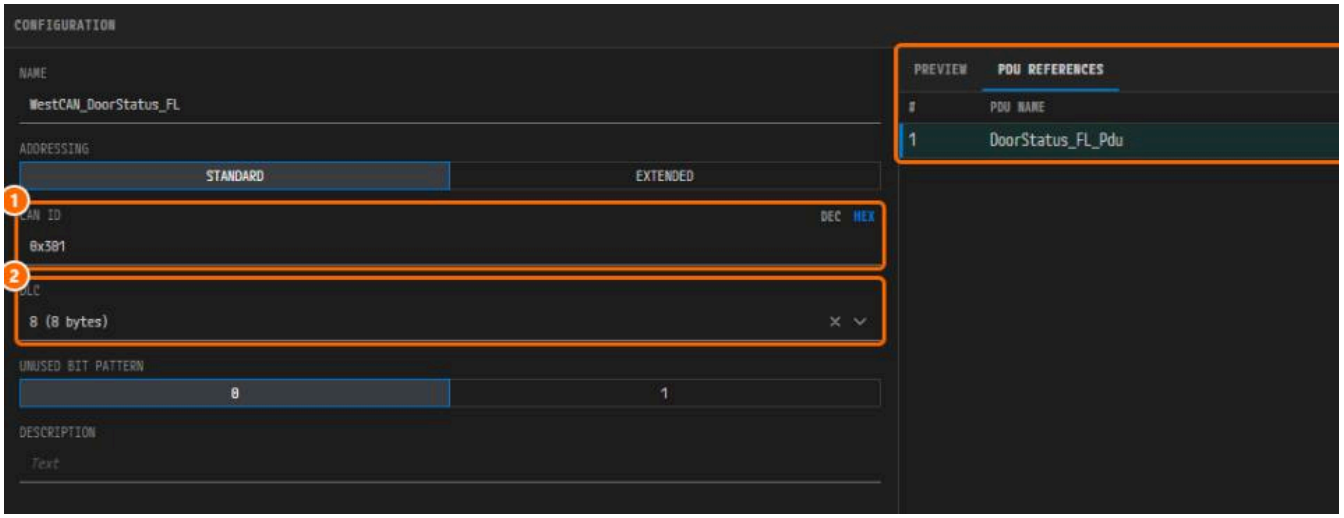


*그림 4-16. ① **CAN ID** `0x301`, ② **DLC**, ③ **PDU REFERENCES**.*

PDU 살펴보기

1. **PDU** 탭을 클릭하고 `DoorStatus_FL_Pdu`를 클릭한다. 목록에는 PDU 34개가 있다. **IS** 태그 16개, **NM** 태그 18개이다.

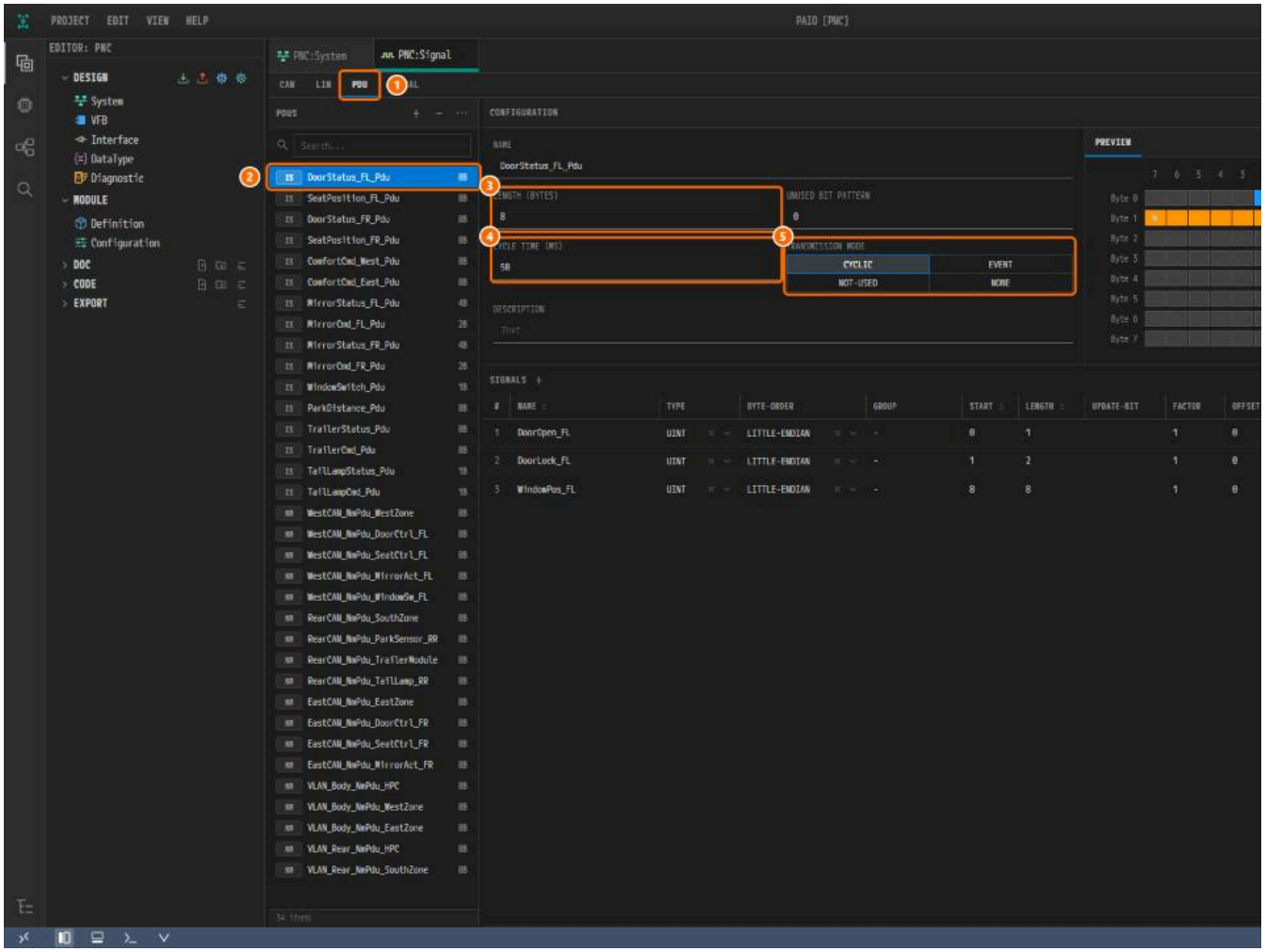


*그림 4-17. ① **PDU**, ② `DoorStatus_FL_Pdu`, ③ **LENGTH (BYTES)**, ④ **CYCLE TIME (MS)**, ⑤ **TRANSMISSION MODE**.*

2. `WestCAN_NmPdu_WestZone`을 클릭한다. NM-PDU에는 신호가 없다.

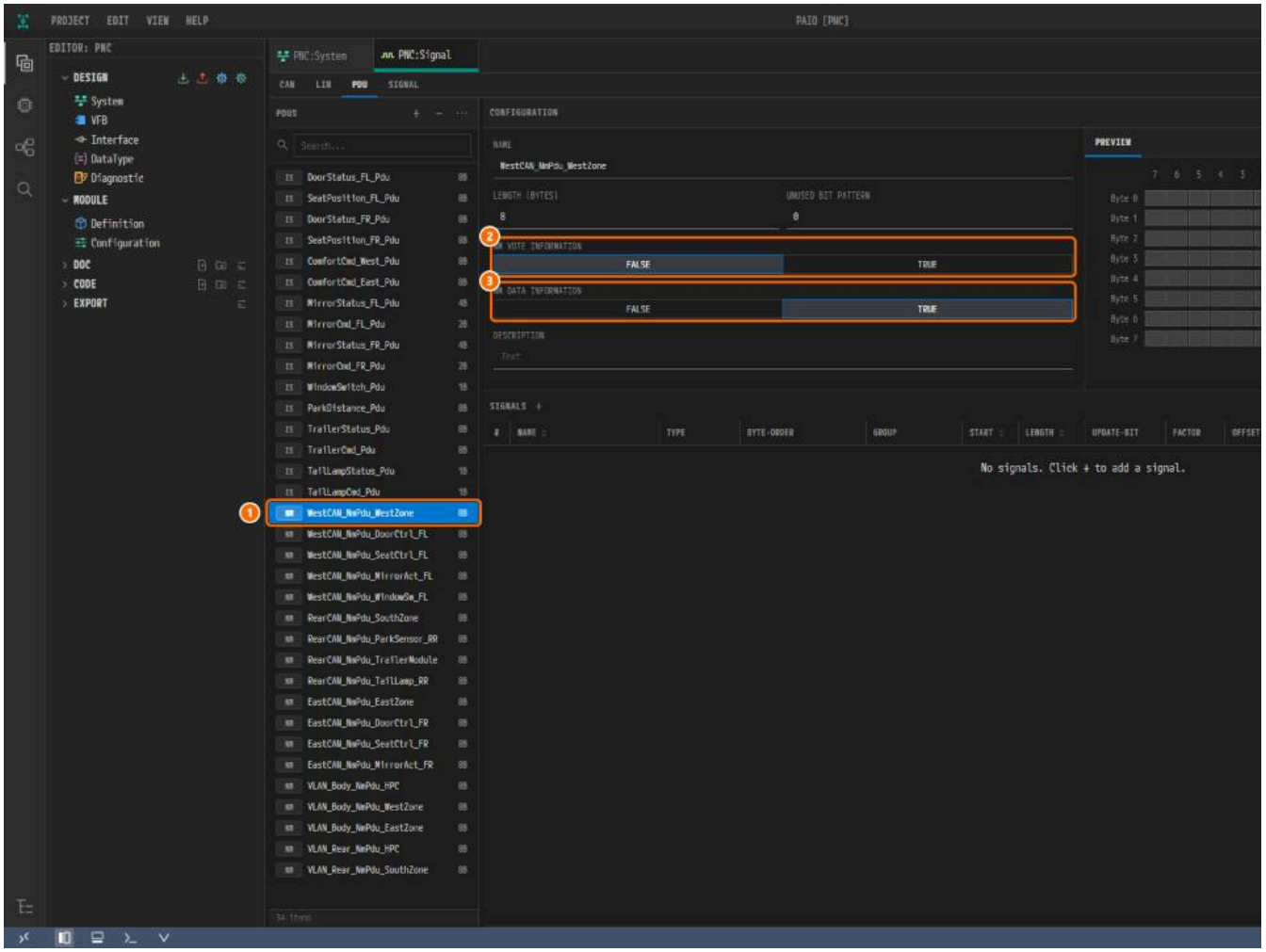


*그림 4-18. ① `WestCAN_NmPdu_WestZone`, ② **NM VOTE INFORMATION**, ③ **NM DATA INFORMATION**.*

신호 살펴보기

1. **SIGNAL** 탭(신호 42개)을 클릭하고 `DoorOpen_FL`을 클릭한다.

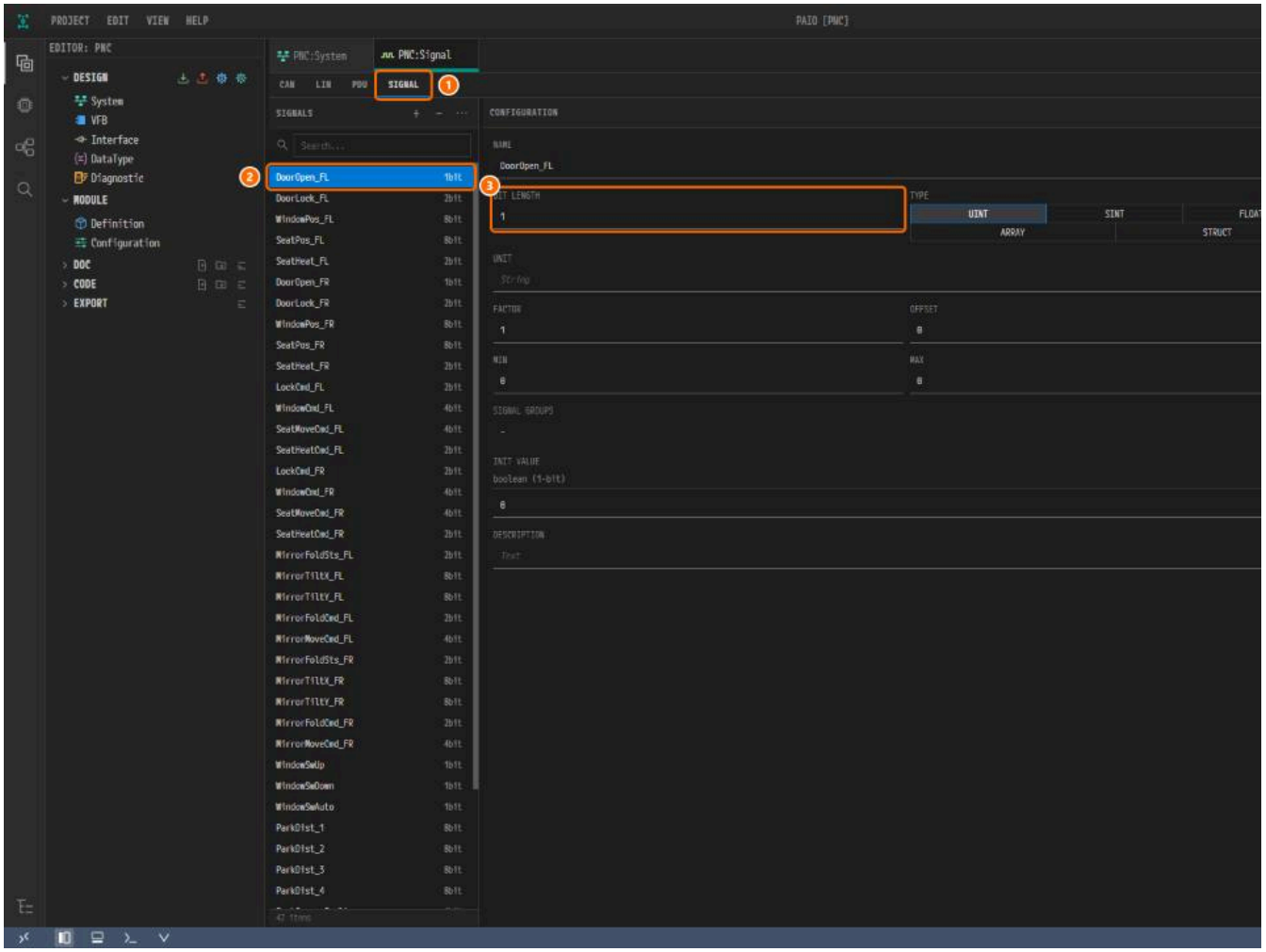


*그림 4-19. ① **SIGNAL**, ② `DoorOpen_FL`, ③ **BIT LENGTH**.*

NM 프레임 살펴보기

NM 프레임은 예제 프로젝트에만 있다.

1. **CAN** 탭에서 `WestCAN_NmFrame_WestZone(0x510)`을 클릭한다.

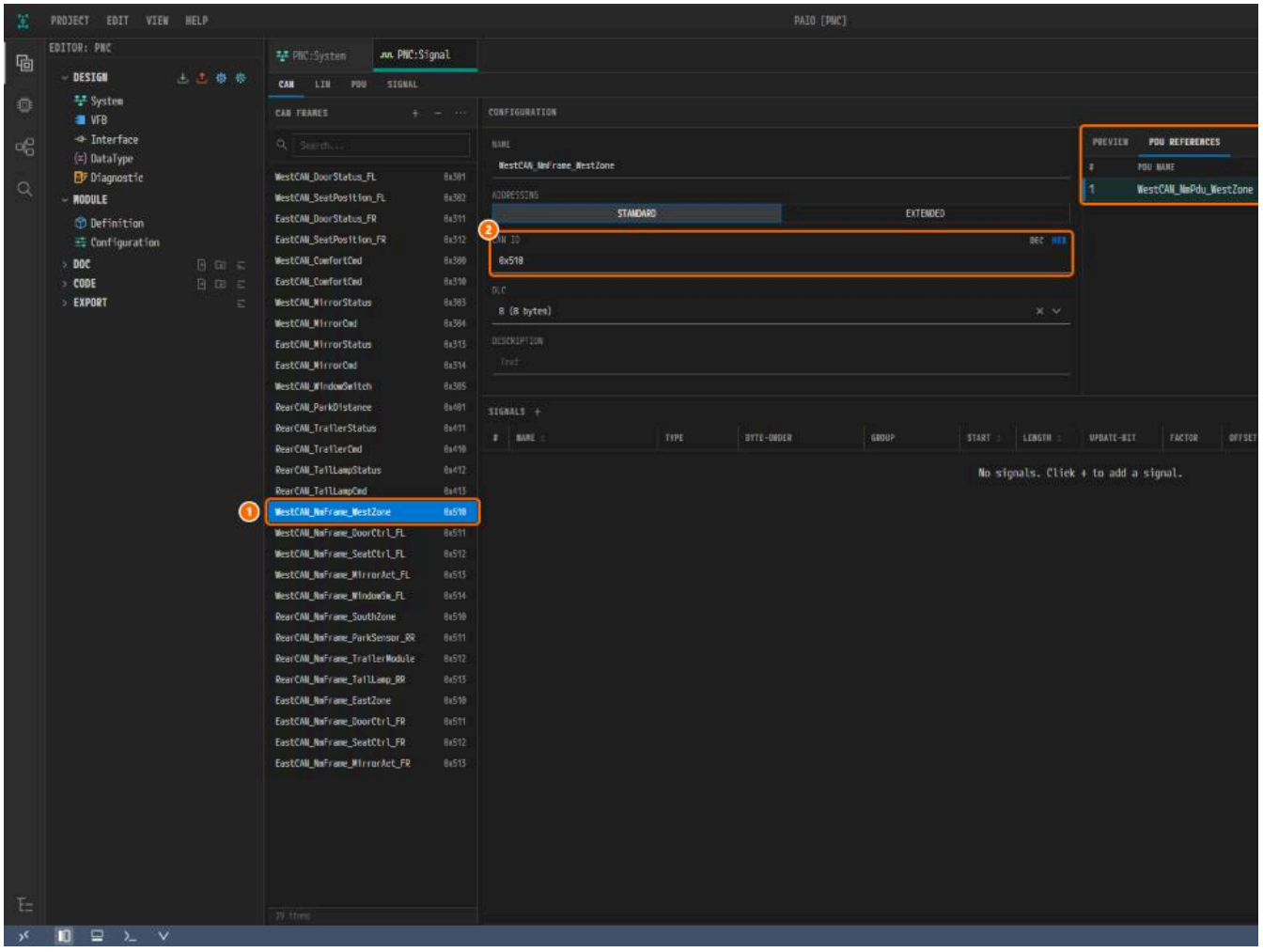


*그림 4-20. ① `WestCAN_NmFrame_WestZone`, ② **CAN ID**, ③ **PDU REFERENCES**의 `WestCAN_NmPdu_WestZone`.*

NM 프레임 `{cluster}_NmFrame_{node}`는 `{cluster}_NmPdu_{node}`를 싣는다. ID는 클러스터마다 `0x510`부터 시작한다.

클러스터	`0x510`	`0x511`	`0x512`	`0x513`	`0x514`
WestCAN	`WestZone`	`DoorCtrl_FL`	`SeatCtrl_FL`	`MirrorAct_FL`	`WindowSw_FL`
RearCAN	`SouthZone`	`ParkSensor_RR`	`TrailerModule`	`TailLamp_RR`	—
EastCAN	`EastZone`	`DoorCtrl_FR`	`SeatCtrl_FR`	`MirrorAct_FR`	—

신호 추가 연습

연습용 신호는 예제에 없는 신호이다. 마지막에 버린다.

1. **CAN** 탭에서 `WestCAN_DoorStatus_FL`을 클릭하고 **PREVIEW** 탭을 클릭한다. **+(Add Signal)**를 클릭한다.



*그림 4-21. ① **+**.*

2. 4행에 `WestCAN_DoorStatus_FL_Sig`가 **START** `16`으로 생긴다.

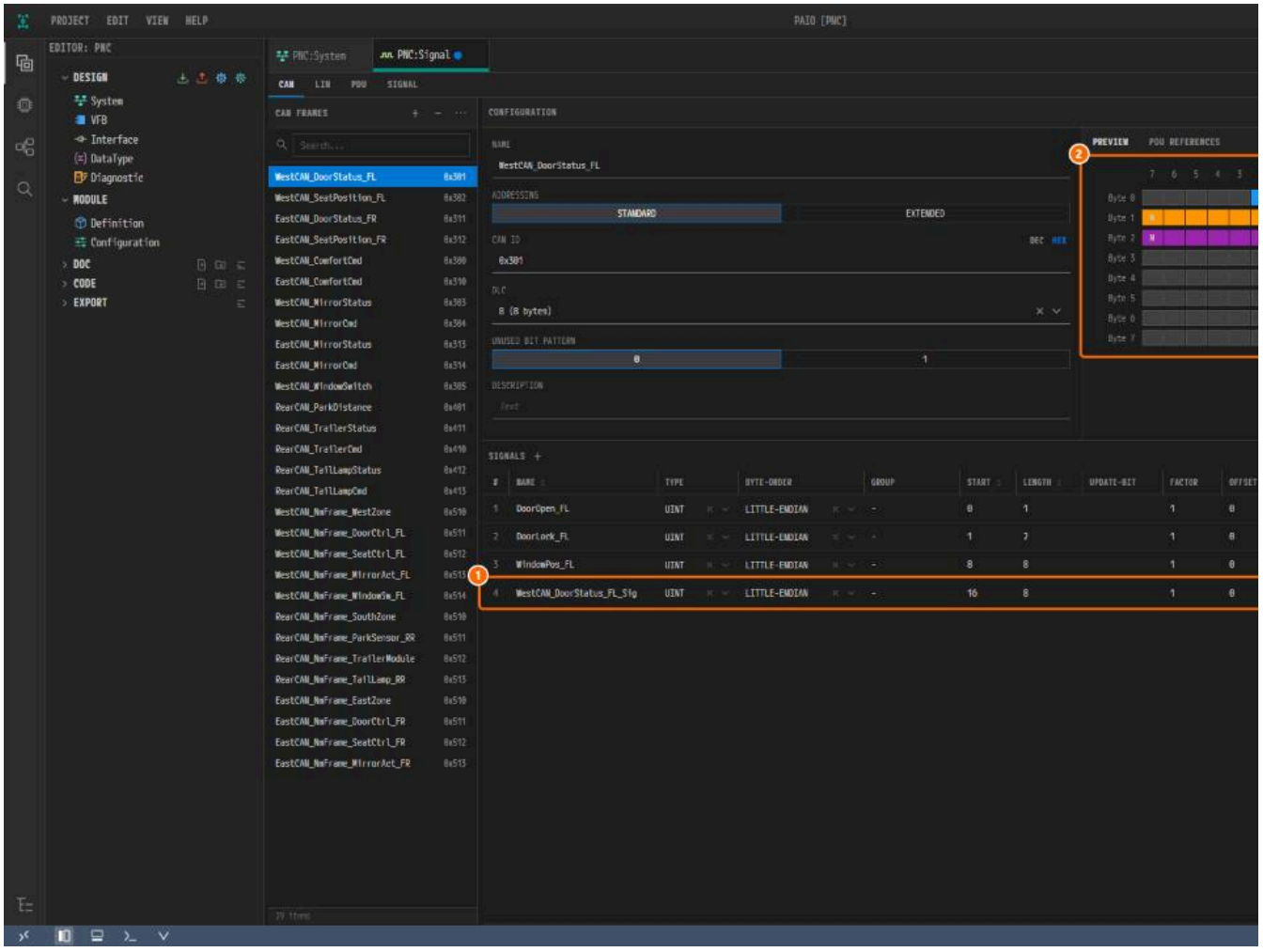


*그림 4-22. ① 새 행, ② **PREVIEW**의 Byte 2.*

3. 이 행의 **NAME** 칸을 클릭하고 `ChildLock_FL`을 입력한 뒤 `Enter`를 누른다.

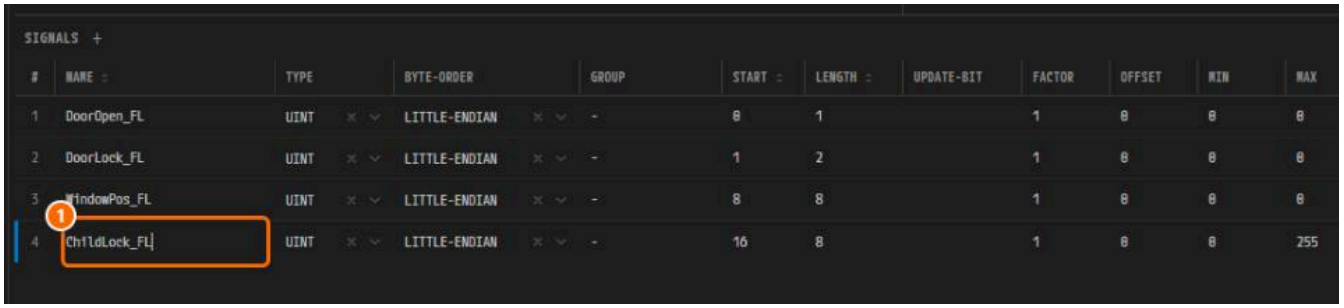


*그림 4-23. ① **NAME**.*

4. **LENGTH**를 `2`로 바꾼다. Byte 2의 비트 1~0만 색이 칠해진다.

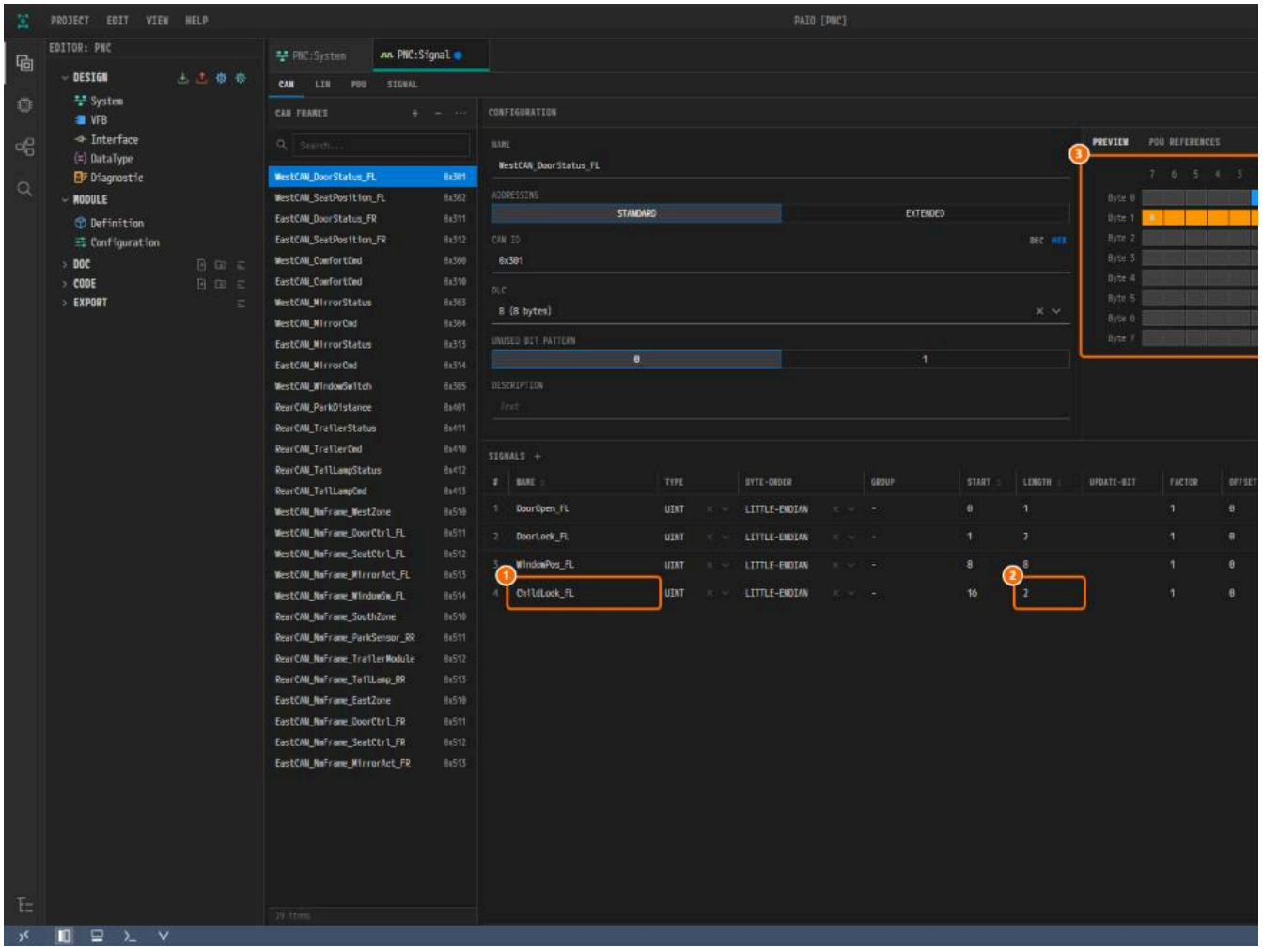


*그림 4-24. ① **NAME**, ② **LENGTH**, ③ Byte 2의 비트 1~0.*

5. 행 오른쪽 끝의 **×(Remove Signal)**를 클릭한다.

6. **PNC:Signal** 탭의 **×**로 탭을 닫고 **Don't Save**를 클릭한다.

> **주의.** PNC 대화상자를 열기 전에 탭을 저장하지 않고 닫는다. PNC 대화상자를 저장하면 열려 있는 수정된 Signal 편집기도 함께 저장된다.

I-PDU Group 만들기

그룹 만들기

1. System 편집기를 열고 캔버스 왼쪽 가장자리의 아래쪽 아이콘 **Partial Network (PNC)**를 클릭한다(그림 3-1).

2. **PARTIAL NETWORKS**에서 **PNC 17 PNC_Comfort**가 선택되어 있는지 확인한다.

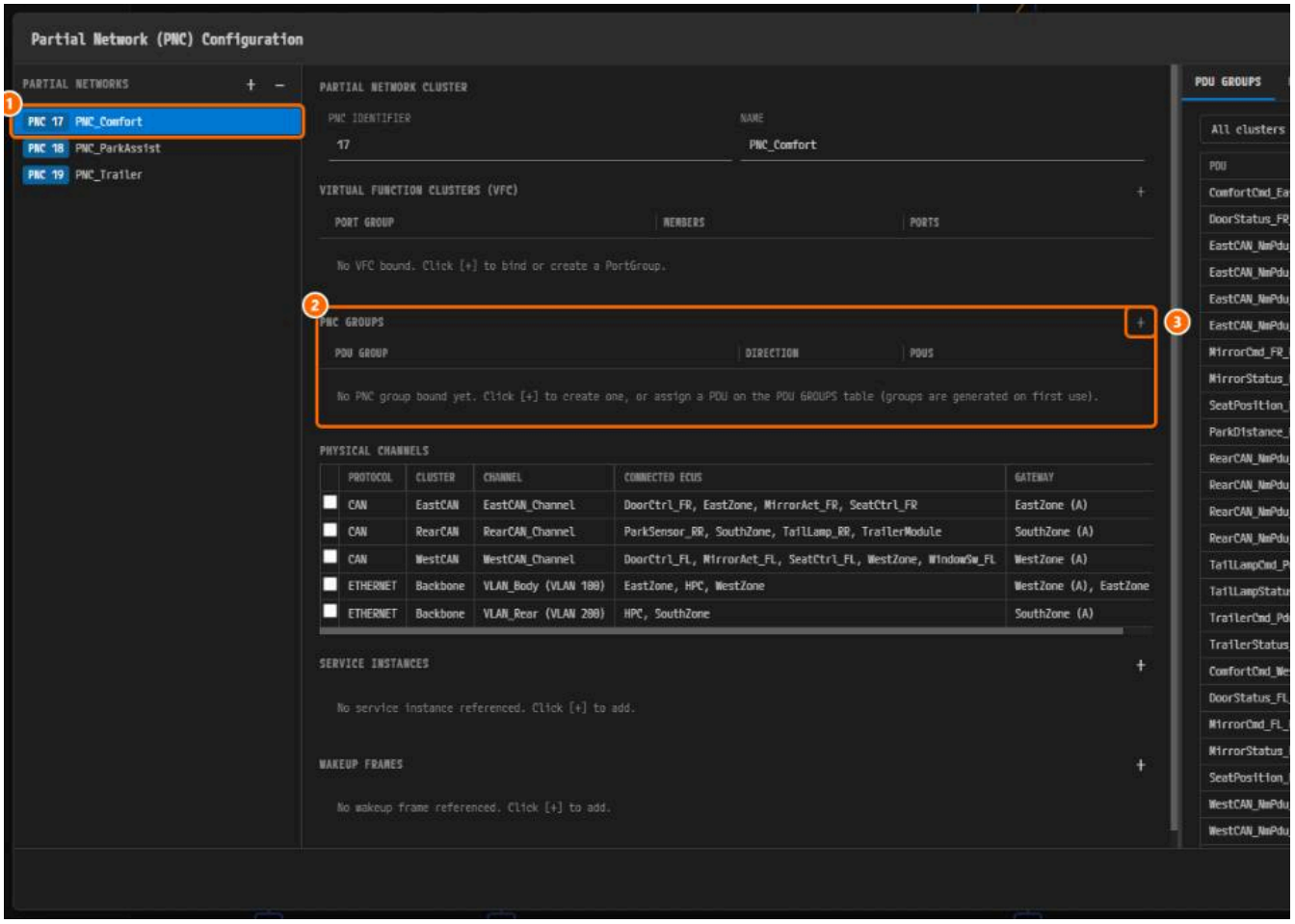


*그림 4-25. ① **PNC 17 PNC_Comfort**, ② **PNC GROUPS**, ③ **+**.*

3. **PNC GROUPS** 머리글의 **+**를 클릭하고 **New Rx group...**을 클릭한다.

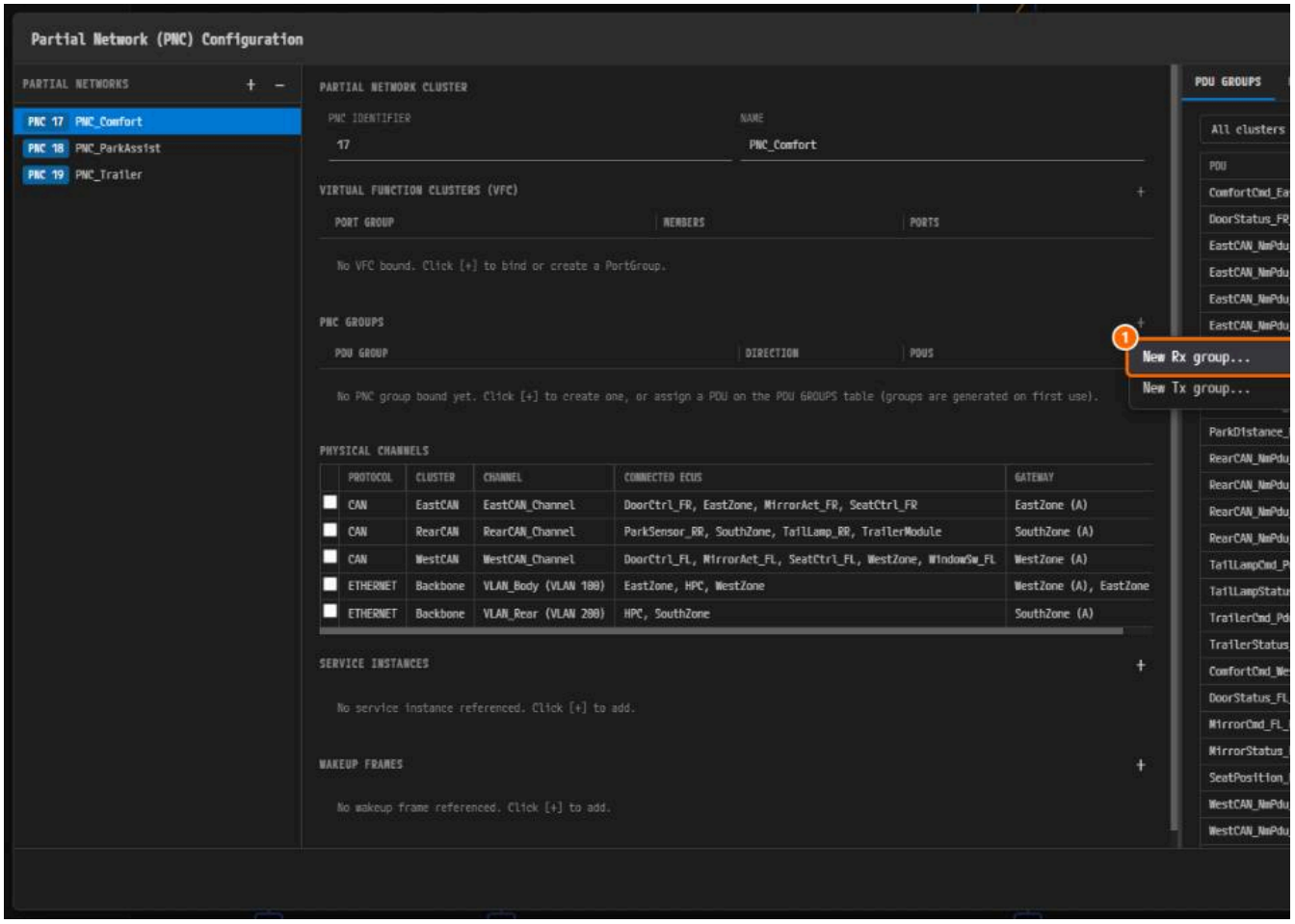


*그림 4-26. ① **New Rx group...**. 송신 그룹은 **New Tx group...*으로 만든다.*

4. `WestCAN_PNC_Comfort_Rx`를 입력하고 **OK**를 클릭한다. PAIO는 명명 규약을 강제하지 않으므로 이름을 그대로 입력한다.

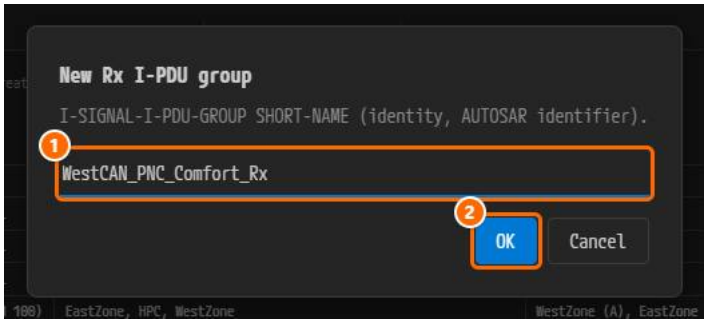


*그림 4-27. ① 이름 입력 칸, ② **OK**.*

5. 그룹 행이 펼쳐진 채로 생긴다. 이 그룹은 **PARTIAL NETWORKS**에서 선택한 PNC에 묶인다.

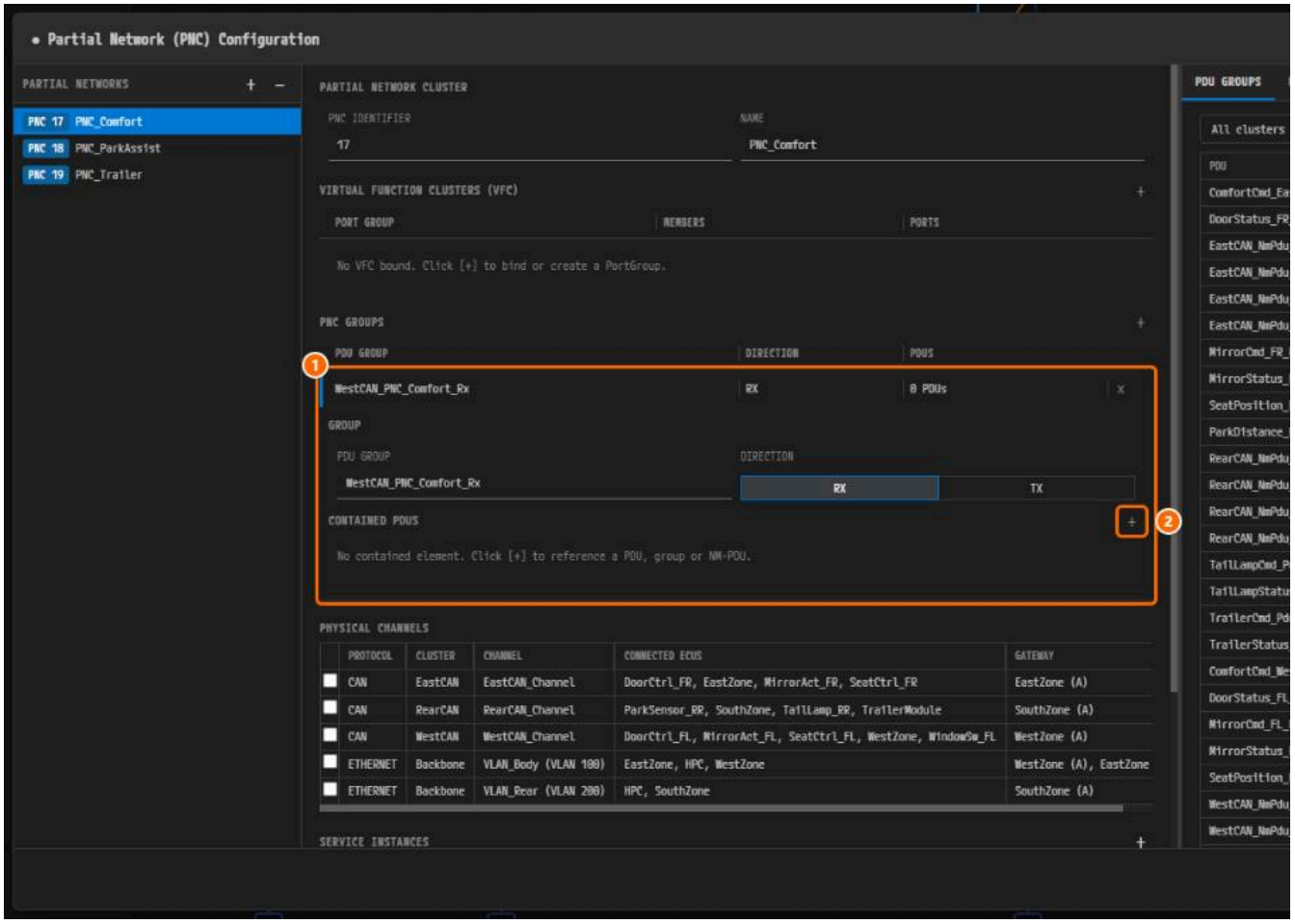


*그림 4-28. ① 새 그룹, ② **CONTAINED PDUS**의 +.*

그룹에 PDU 넣기

1. **CONTAINED PDUS** 머리글의 **+(Add contained element)**를 클릭한다. **KIND**는 **I-SIGNAL-I-PDU**이다.
2. **REFERENCE** 칸을 클릭하고 `DoorStatus`를 입력한다.

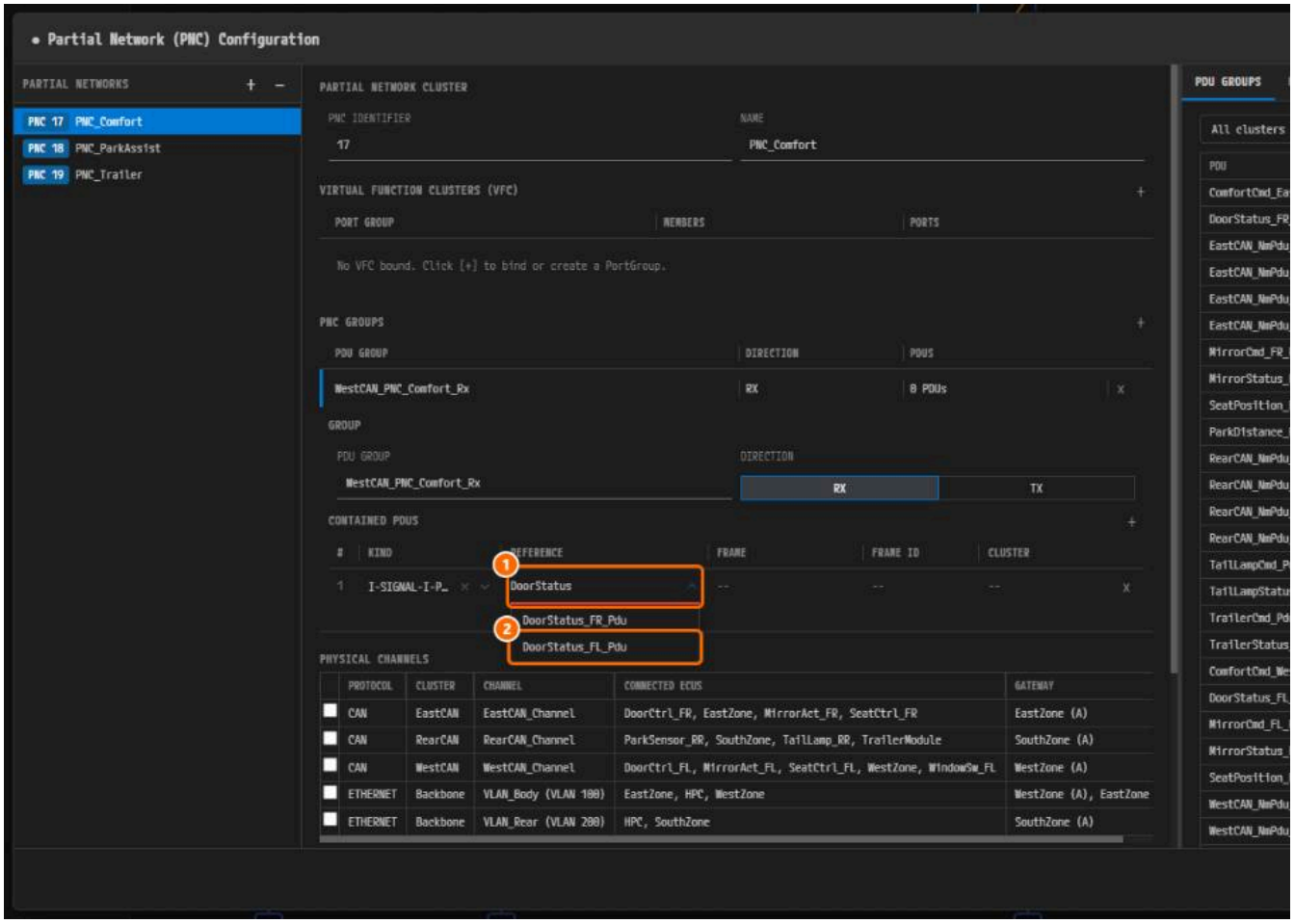


*그림 4-29. ① **REFERENCE**, ② `DoorStatus_FL_Pdu`.*

3. `DoorStatus_FL_Pdu`를 클릭한다. **FRAME**, **FRAME ID**, **CLUSTER**가 채워진다.

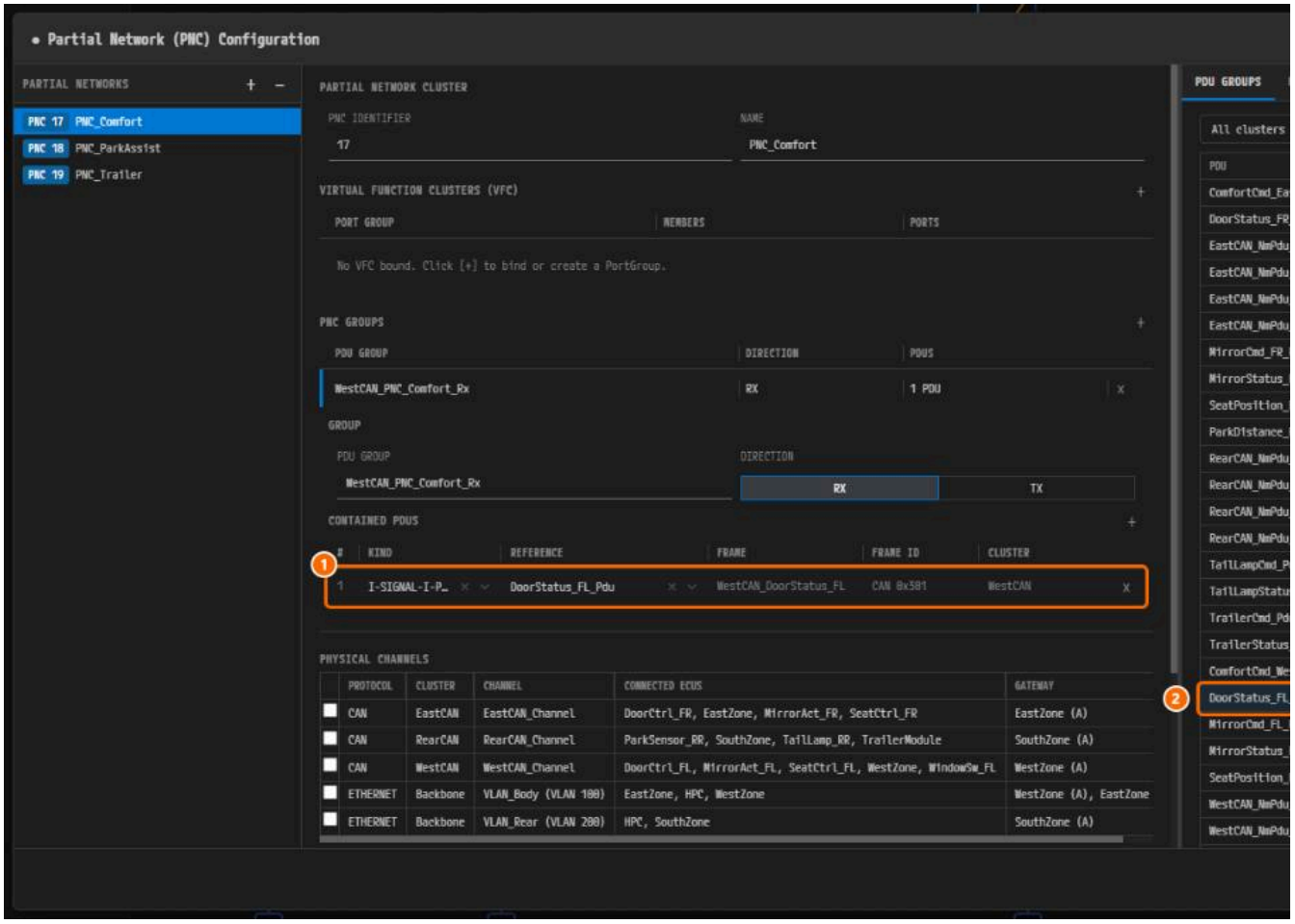


*그림 4-30. ① **CONTAINED PDUS** 행, ② **PDU GROUPS** 표의 같은 PDU.*

4. `Ctrl+S`로 저장한다.

> **주의. REFERENCE** 목록에는 NM-PDU(`*_NmPdu_*`)도 나온다. NM-PDU는 PNC 그룹에 넣지 않는다. 그룹 행의 **×**는 그룹 자체를 삭제한다.

나머지 그룹 만들기

- `WestCAN_PNC_Comfort_Rx`에 `SeatPosition_FL_Pdu`, `MirrorStatus_FL_Pdu`, `WindowSwitch_Pdu`를 더 넣는다.
- 그룹을 만들기 전에 **PARTIAL NETWORKS**에서 해당 PNC를 선택한다.
- `RearCAN_PNC_ParkAssist_Tx`는 비워 둔다.

PNC	그룹	DIRECTION	CONTAINED PDUS
PNC 17 `PNC_Comfort`	`WestCAN_PNC_Comfort_Rx`	**RX**	`DoorStatus_FL_Pdu`, `SeatPosition_FL_Pdu`, `MirrorStatus_FL_Pdu`, `WindowSwitch_Pdu`
PNC 17 `PNC_Comfort`	`WestCAN_PNC_Comfort_Tx`	**TX**	`ComfortCmd_West_Pdu`, `MirrorCmd_FL_Pdu`
PNC 17 `PNC_Comfort`	`EastCAN_PNC_Comfort_Rx`	**RX**	`DoorStatus_FR_Pdu`, `SeatPosition_FR_Pdu`, `MirrorStatus_FR_Pdu`
PNC 17 `PNC_Comfort`	`EastCAN_PNC_Comfort_Tx`	**TX**	`ComfortCmd_East_Pdu`, `MirrorCmd_FR_Pdu`
PNC 18 `PNC_ParkAssist`	`RearCAN_PNC_ParkAssist_Rx`	**RX**	`ParkDistance_Pdu`
PNC 18 `PNC_ParkAssist`	`RearCAN_PNC_ParkAssist_Tx`	**TX**	(없음)
PNC 19 `PNC_Trailer`	`RearCAN_PNC_Trailer_Rx`	**RX**	`TrailerStatus_Pdu`, `TailLampStatus_Pdu`
PNC 19 `PNC_Trailer`	`RearCAN_PNC_Trailer_Tx`	**TX**	`TrailerCmd_Pdu`, `TailLampCmd_Pdu`

`Ctrl+S`로 저장하고 **Close**를 클릭한다.

완성 화면

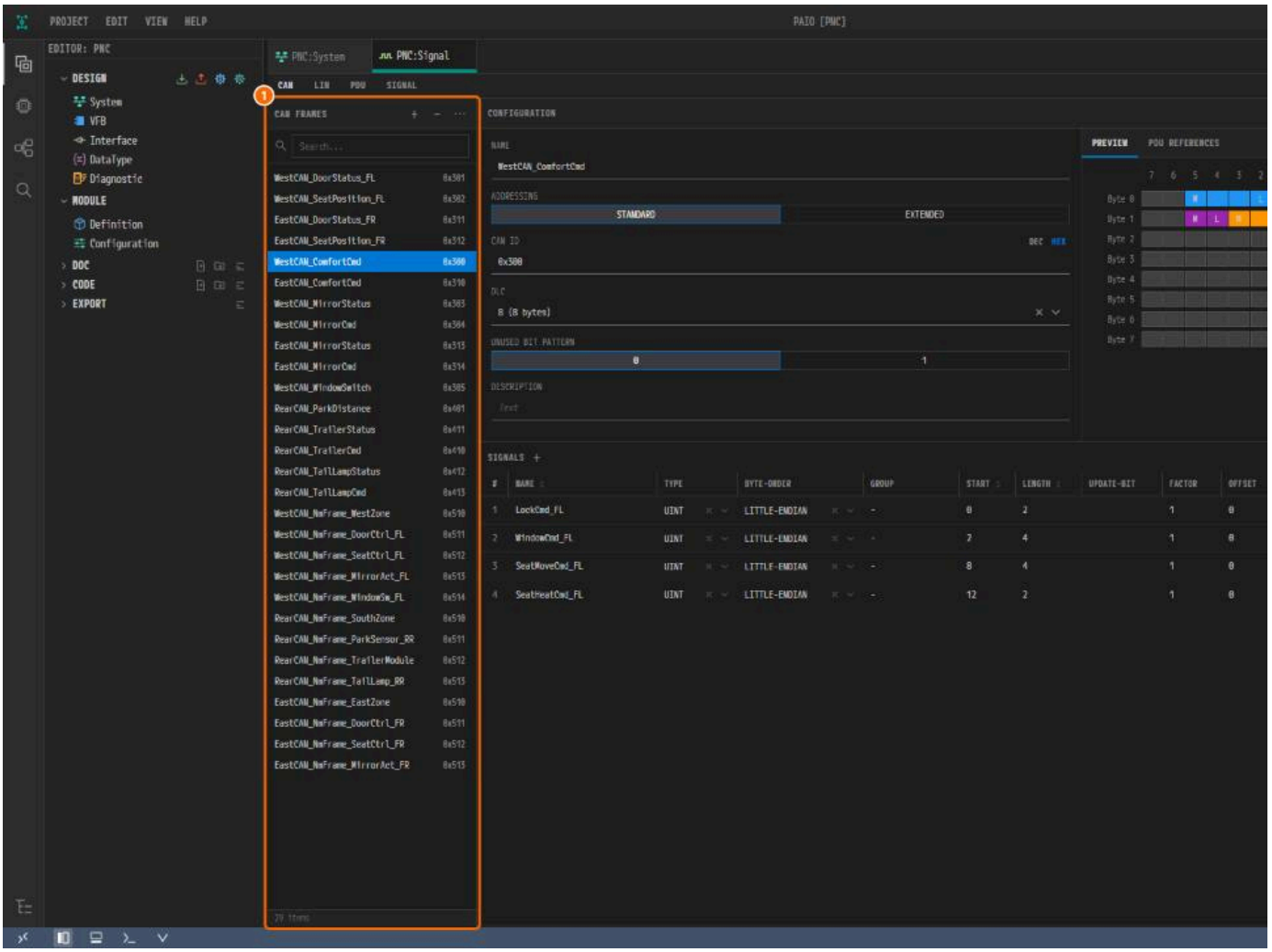


*그림 4-31. ① `WestCAN_ComfortCmd`를 선택한 **CAN FRAMES**(29개).*

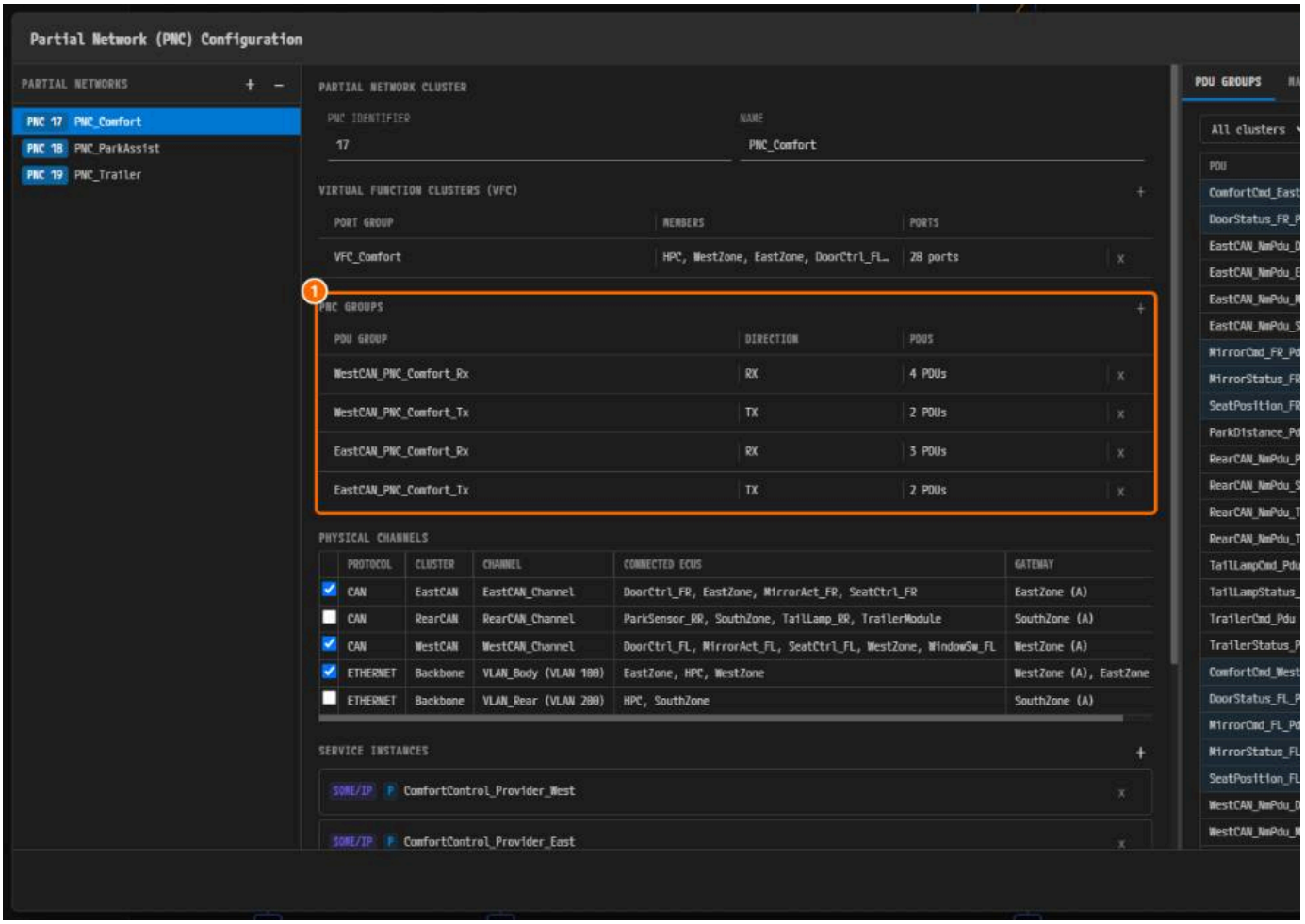


*그림 4-32. ① **PNC 17 PNC_Comfort**의 그룹 4개. 그 아래 섹션은 PNC 다이얼로그 — PNC별 바인딩에서 채운다.*

그룹 행을 클릭하면 **CONTAINED PDUS**가 보인다.

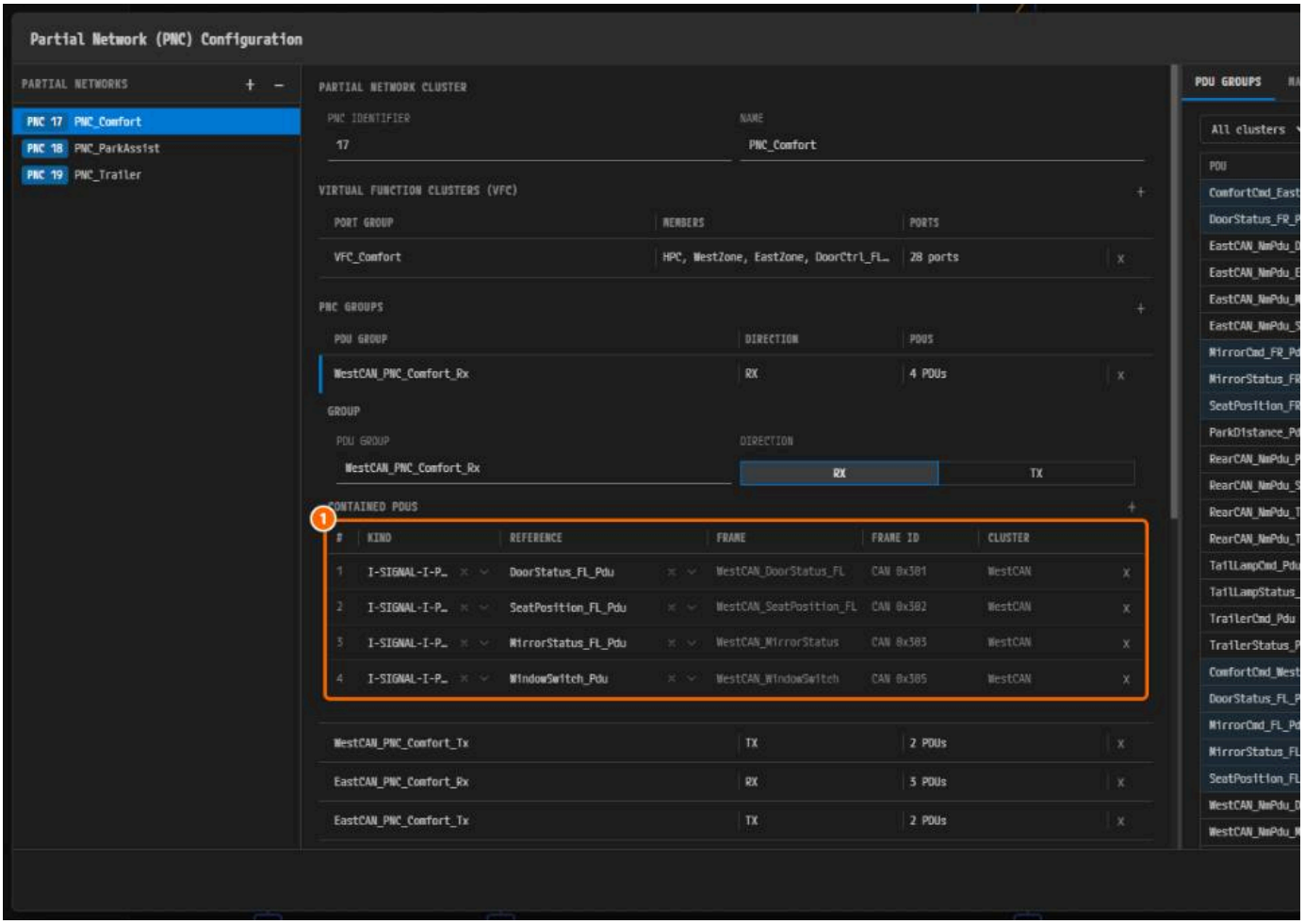


그림 4-33. ① `WestCAN_PNC_Comfort_Rx`의 PDU 4개.

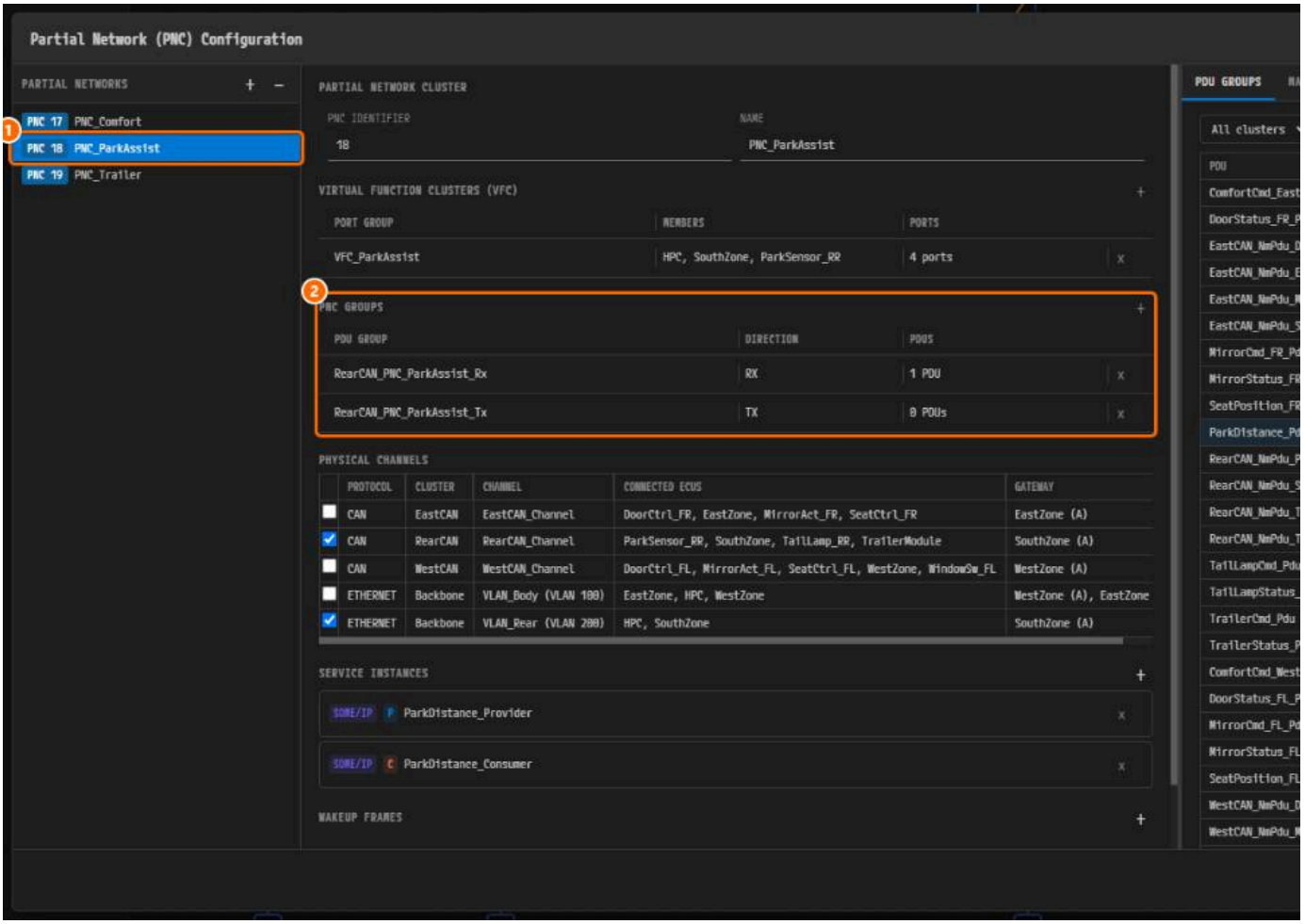


*그림 4-34. ① **PNC 18 PNC_ParkAssist**, ② 그 그룹 2개.*

명명 규약

PAIO는 I-PDU Group 이름에 이 규약을 쓰고, ARXML 임포트와 익스포트도 같은 규약을 쓴다.

형태	이름	예	예제의 내용
컨테이너	`{cluster}_Rx` / `{cluster}_Tx`	`RearCAN_Rx`	그 버스의 리프 그룹
PNC 리프	`{cluster}_{pnc}_Rx` / `{cluster}_{pnc}_Tx`	`WestCAN_PNC_Comfort_Rx`	그 버스에서 그 PNC의 PDU
비참여 리프	`{cluster}_NoPnc_Rx` / `{cluster}_NoPnc_Tx`	`WestCAN_NoPnc_Tx`	PNC에 속하지 않는 PDU. 이 예제에서는 비어 있다.
NM-PDU	`{cluster}_NmPdu_{node}`	`WestCAN_NmPdu_WestZone`	노드마다 1개, 8바이트
NM 프레임	`{cluster}_NmFrame_{node}`	`WestCAN_NmFrame_WestZone`	`0x510`부터 노드 순서대로

- `_Rx` / `_Tx`는 Zone(게이트웨이) 관점이다. 말단 ECU가 보내는 상태 PDU는 `_Rx`, 말단 ECU로 보내는 명령 PDU는 `_Tx`이다.
- PNC에 묶는 것은 PNC 리프 그룹뿐이다.

컨테이너와 NoPnc 그룹은 ARXML 임포트로 들어오며 **PNC GROUPS**에 나타나지 않는다. **PNC GROUPS**의 **+**로 만들지 않는다. 직접 만든 프로젝트는 이 그룹 없이도 된다(부록 있다.

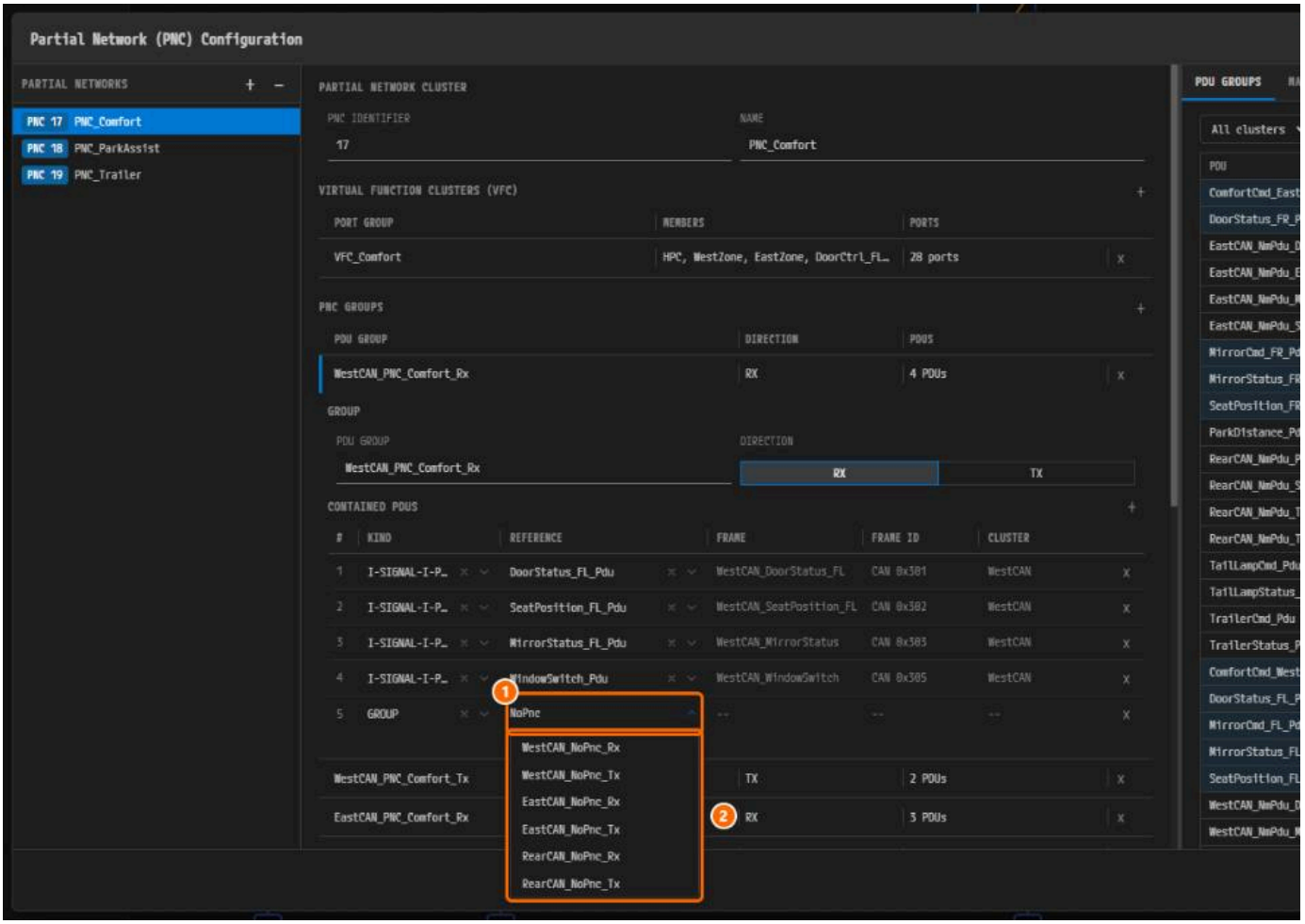


*그림 4-35. ① **REFERENCE**, ② `NoPnc` 후보. 확인용이므로 `Esc`를 누르고 **××*로 행을 지운다.*

왜 이렇게 설계하는가

- **PNC별 그룹.** PNC는 자기 I-PDU Group을 시작하고 멈추므로, 버스마다 PNC별 Rx 그룹과 Tx 그룹을 둔다.
- **PNC 그룹에 NM-PDU를 넣지 않는다.** NM은 PNC가 켜져 있든 꺼져 있든 동작해야 한다.
- **`_FL`과 `_FR` 신호를 따로 둔다.** 같은 I-Signal을 두 PDU에 매핑하면 `constr_3514`를 위반한다.

확인하기

- [ ] Signal 편집기에 PDU 34개(**IS** 16, **NM** 18), 신호 42개, 응용 프레임 16개(완성 예제는 프레임 29개)가 있다.
- [ ] `DoorStatus_FL_Pdu`의 **CYCLE TIME (MS)**가 `50`, **TRANSMISSION MODE**가 **CYCLIC**이다.
- [ ] 연습용 신호 `ChildLock_FL`이 남아 있지 않다.
- [ ] **PNC GROUPS**에 표와 같이 PNC 17은 그룹 4개, PNC 18은 2개, PNC 19는 2개가 있고 NM-PDU는 없다.
- [ ] **PNC:Signal** 탭과 PNC 대화상자 제목 어디에도 수정 표시가 없다.

다음 단계

NM 설정으로 돌아가 NM-PDU 참조를 채운 뒤, 인터페이스와 SOME/IP 서비스로 넘어간다.

인터페이스와 SOME/IP 서비스

이 장에서 할 일

이 장에서는 CP 포트가 쓰는 Sender-Receiver 인터페이스와, HPC와 Zone 사이의 SOME/IP에 쓰는 서비스 인터페이스를 만든다. 서비스마다 SOME/IP 배포와 서비스 인스턴스를 뒤 인스턴스를 커넥터에 바인딩한다.

시작하기 전에

- 신호·PDU·프레임과 I-PDU Group을 마친다.
- Ethernet 커넥터 `HPC_Eth_Body_Conn`, `HPC_Eth_Rear_Conn`, `WestZone_Eth_Conn`, `EastZone_Eth_Conn`, `SouthZone_Eth_Conn`이 있다(토폴로지 설계에서
- 아래 서비스 이름, Service ID, 인스턴스 이름은 PNC 다이얼로그 — PNC별 바인딩에서 그대로 다시 쓴다.

서비스 인터페이스	Service ID	제공자	
`SVC_ParkDistance` (이벤트 DistanceUpdate, SensorFault / 메서드 SetSensitivity)	`0x1001`	SouthZone (`ParkDistance_Provider`)	HPC ParkA (`ParkDis`
`SVC_VehicleState` (이벤트 GearPosition, TrailerAttached)	`0x1002`	HPC VehicleStateApp (`VehicleState_Provider`)	SouthZon
`SVC_ComfortControl` (이벤트 DoorStatus / 메서드 LockAll, FoldMirrors)	`0x1003`	WestZone (`ComfortControl_Provider_West`, 인스턴스 1), EastZone (`ComfortControl_Provider_East`, 인스턴스 2)	HPC Comf (`Comfort`

Sender-Receiver 인터페이스 만들기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **Interface**를 누른다. `PNC:Interface` 탭이 열린다.
2. **SENDER-RECEIVER** 탭을 누른다.

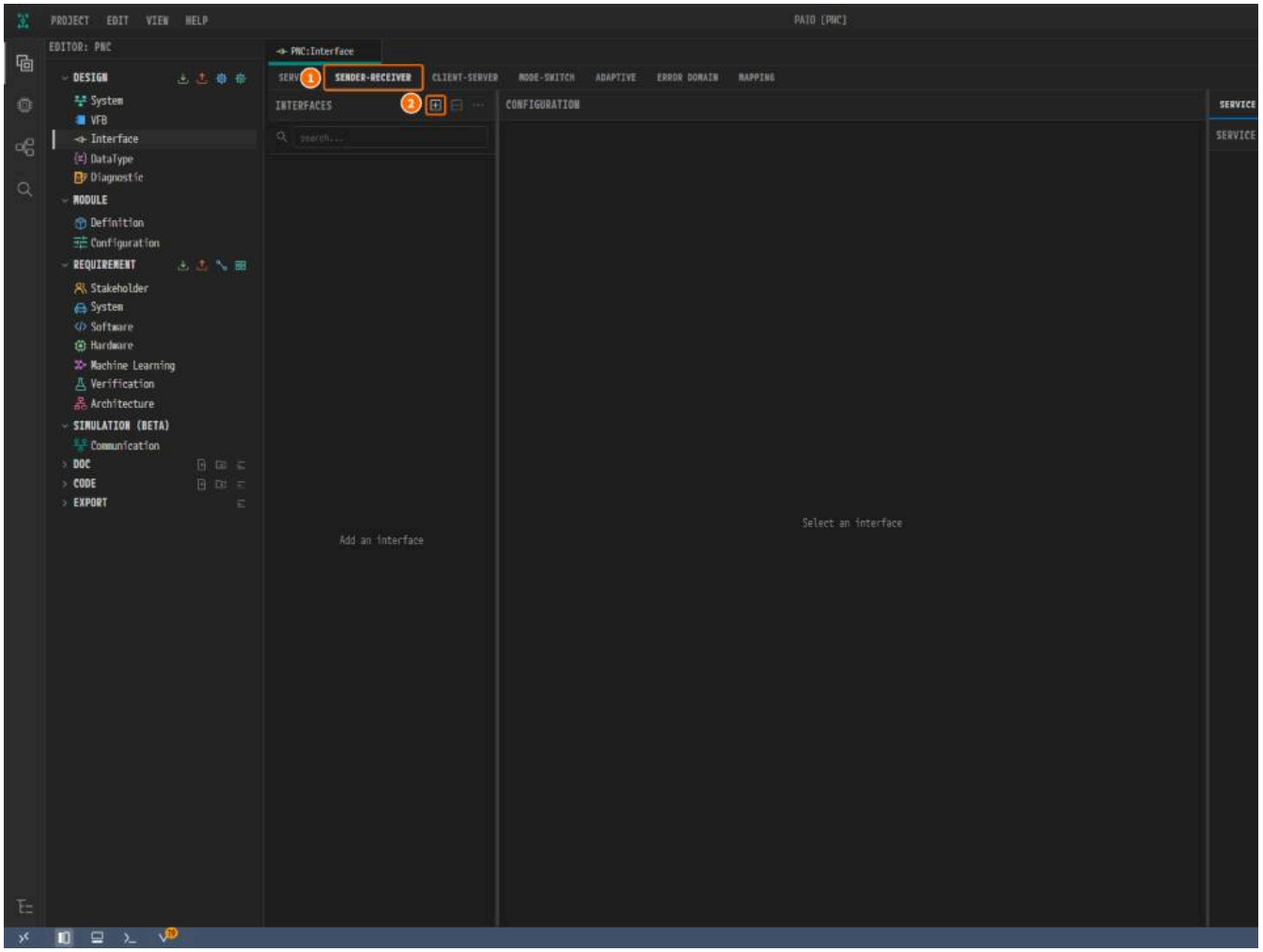


*그림 5-1. ① **SENDER-RECEIVER**, ② **INTERFACES**의 **+**.*

3. **INTERFACES** 오른쪽의 **+**를 누른다. `SenderReceiverInterface_0`이 추가된다.

4. **NAME**에 `SRI_ParkDistance`를 입력하고 `Enter`를 누른다.

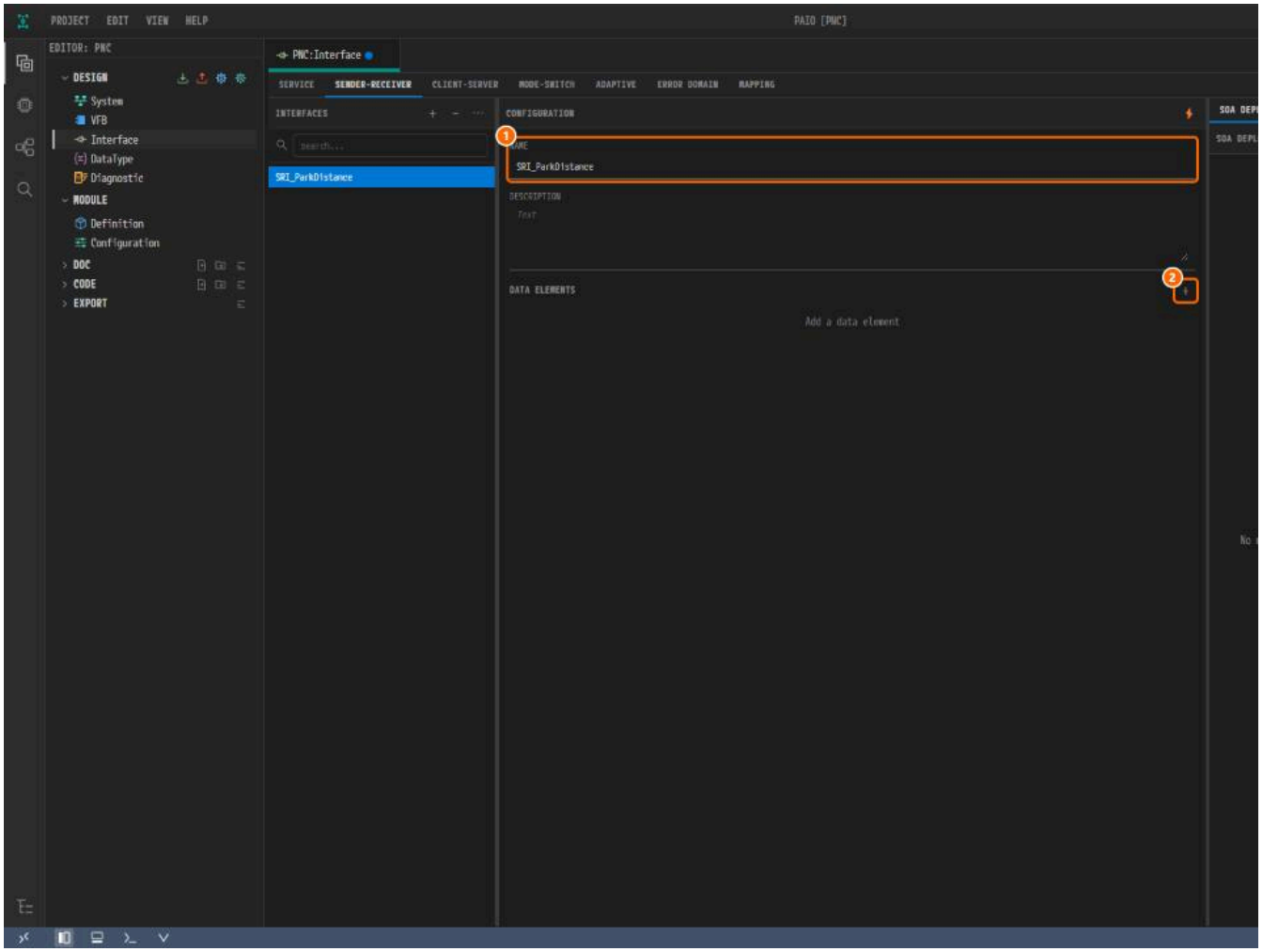


*그림 5-2. ① **NAME**, ② **DATA ELEMENTS**의 **+**.*

5. **DATA ELEMENTS** 오른쪽의 **+**를 누른다. `DataElement_0` 행(`uint8`)이 추가된다.

6. **NAME**에 `Distance1`을 입력한다.

7. **TYPE** 칸을 눌러 타입 목록을 연다.

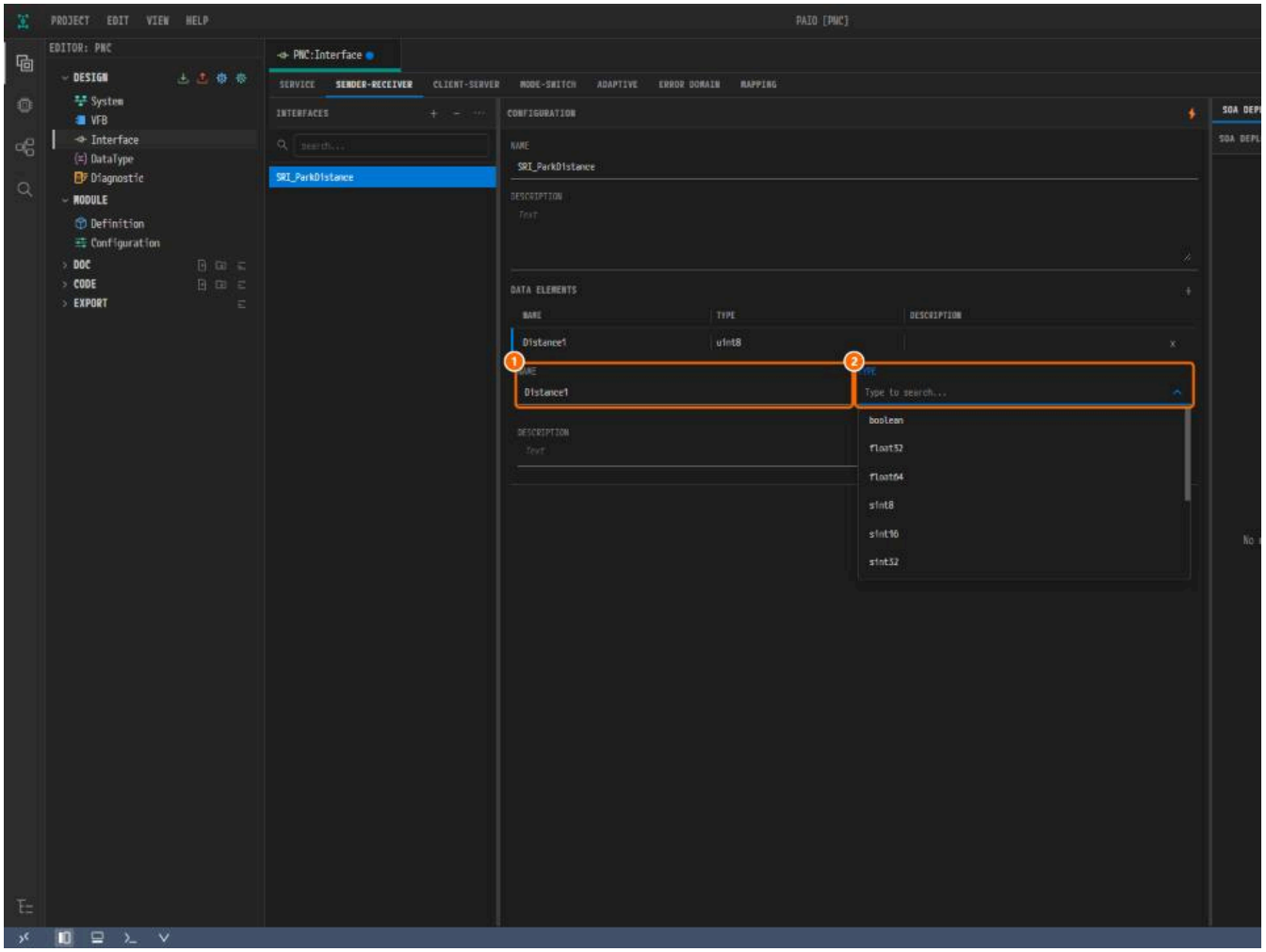


*그림 5-3. ① **NAME**, ② **TYPE** 목록. 타입은 목록을 내리거나 검색 칸에 입력해 찾는다.*

8. `uint8`을 고른다(기본값이므로 목록을 그냥 닫아도 된다).

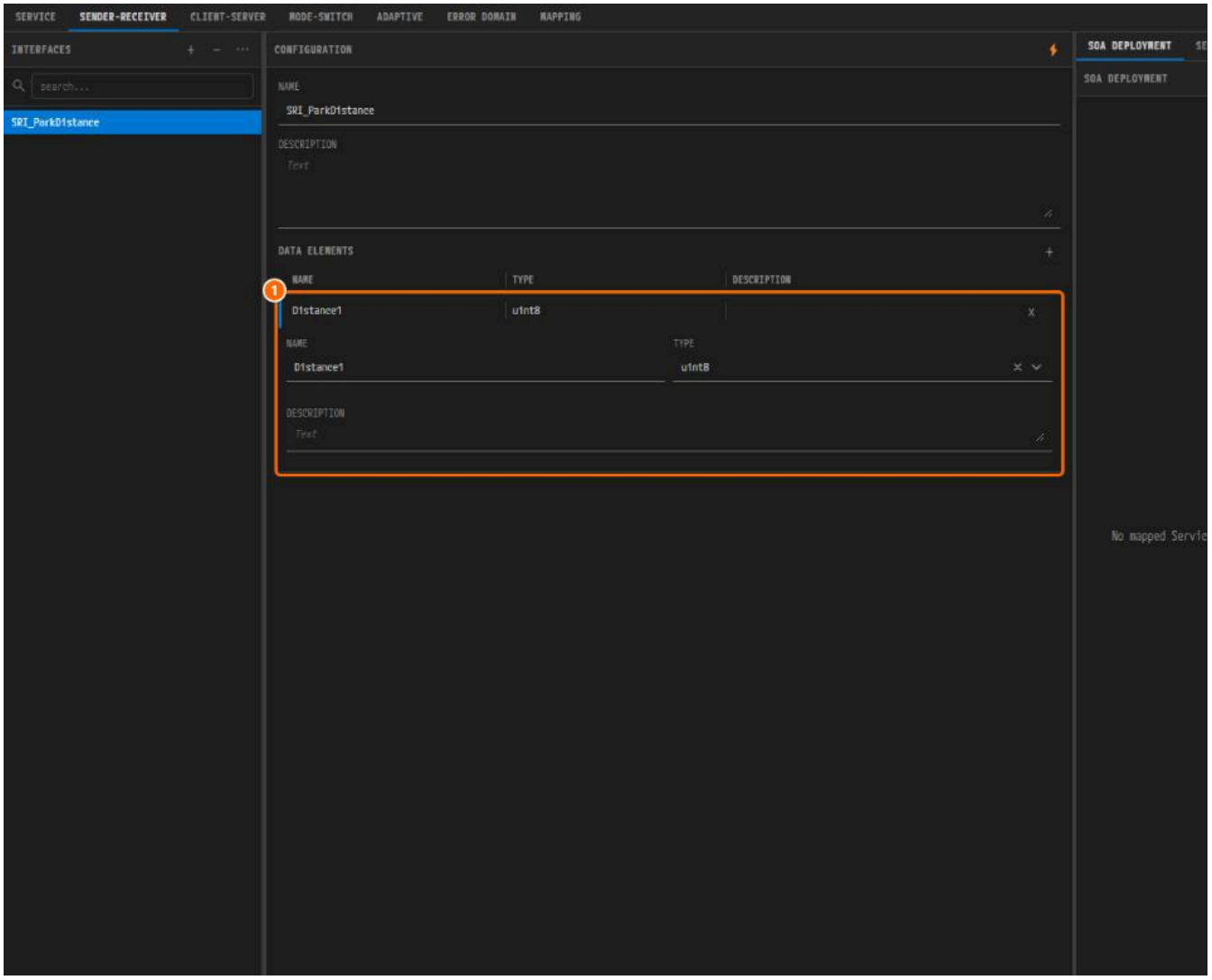


그림 5-4. ① `Distance1`(`uint8`).

9. 같은 방법으로 `Distance2`, `Distance3`, `Distance4`(`uint8`)와 `SensorFault`(`boolean`)를 추가한다.

나머지 Sender-Receiver 인터페이스

나머지도 표를 보고 같은 방법으로 만든다. 타입은 모두 내장 프리미티브이다.

인터페이스	데이터 요소 (타입)
`SRI_DoorStatus`	`DoorOpen` (boolean), `LockState` (uint8), `WindowPos` (uint8)
`SRI_SeatStatus`	`SeatPos` (uint8), `SeatHeat` (uint8)
`SRI_ComfortCmd`	`LockCmd`, `WindowCmd`, `SeatMoveCmd`, `SeatHeatCmd` (모두 uint8)
`SRI_MirrorStatus`	`FoldState`, `TiltX`, `TiltY` (모두 uint8)
`SRI_MirrorCmd`	`FoldCmd`, `MoveCmd` (모두 uint8)
`SRI_ParkDistance`	`Distance1`, `Distance2`, `Distance3`, `Distance4` (uint8), `SensorFault` (boolean)
`SRI_TrailerStatus`	`Attached` (boolean), `LampFault` (uint8), `BrakeLamp` (boolean)
`SRI_TrailerCmd`	`LampCmd` (uint8)
`SRI_WindowSwitch`	`Up`, `Down`, `Auto` (모두 boolean)
`SRI_TailLamp`	`LampState` (uint8)
`SRI_GearPosition`	`Gear` (uint8)

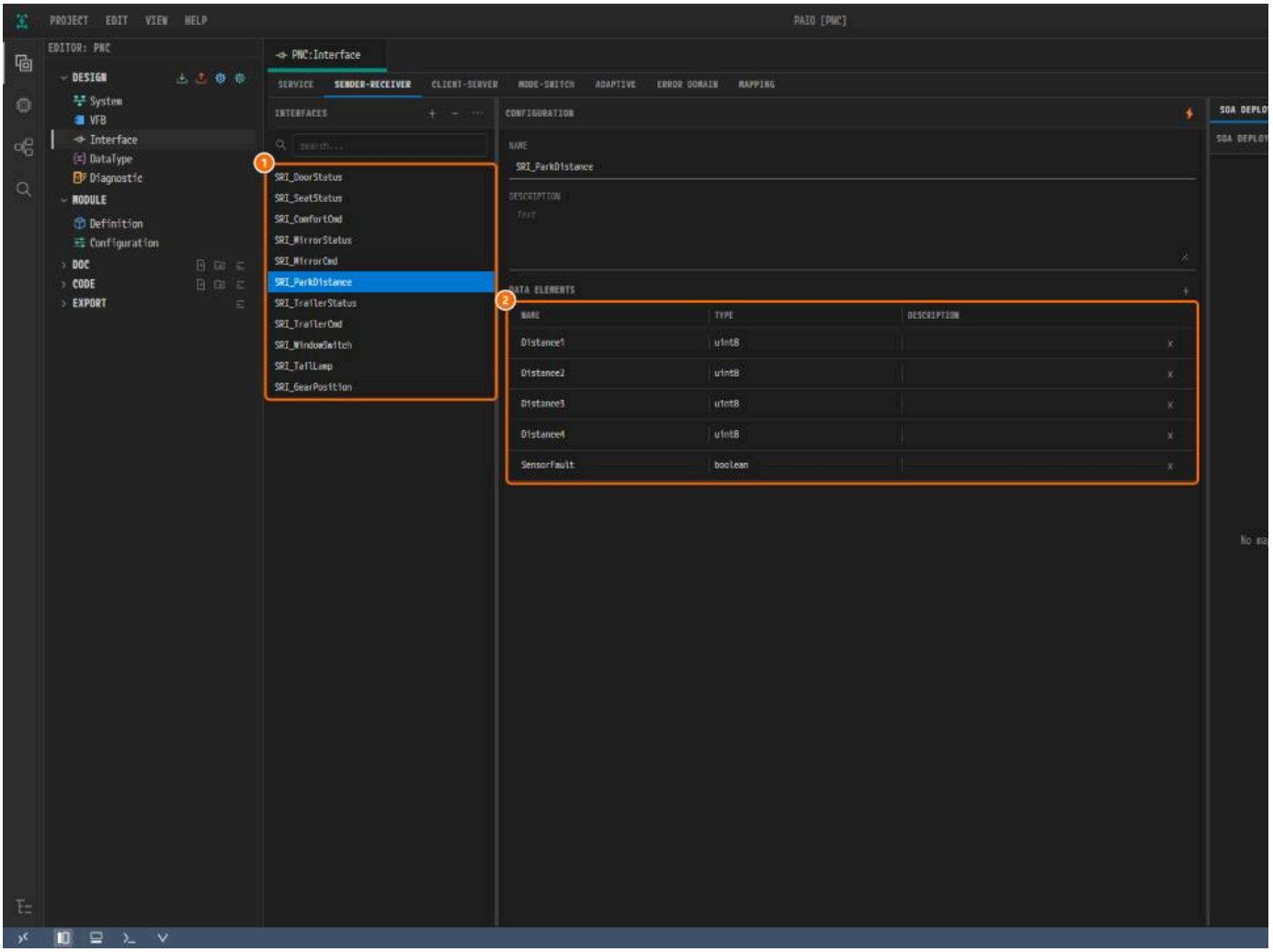


그림 5-5. ① `SRI_*` 인터페이스 11개, ② `SRI_ParkDistance`의 데이터 요소.

서비스 인터페이스 만들기

1. **SERVICE** 탭을 누른다.

2. **INTERFACES** 오른쪽의 **+**를 누른다. `ServiceInterface_0`이 추가된다.

3. **NAME**에 `SVC_ParkDistance`를 입력한다.

4. **C++ NAMESPACE**에 `pnc::park`를 입력하고 `Tab`을 누른다. **MAJOR VERSION** `1`, **MINOR VERSION** `0`은 그대로 둔다.

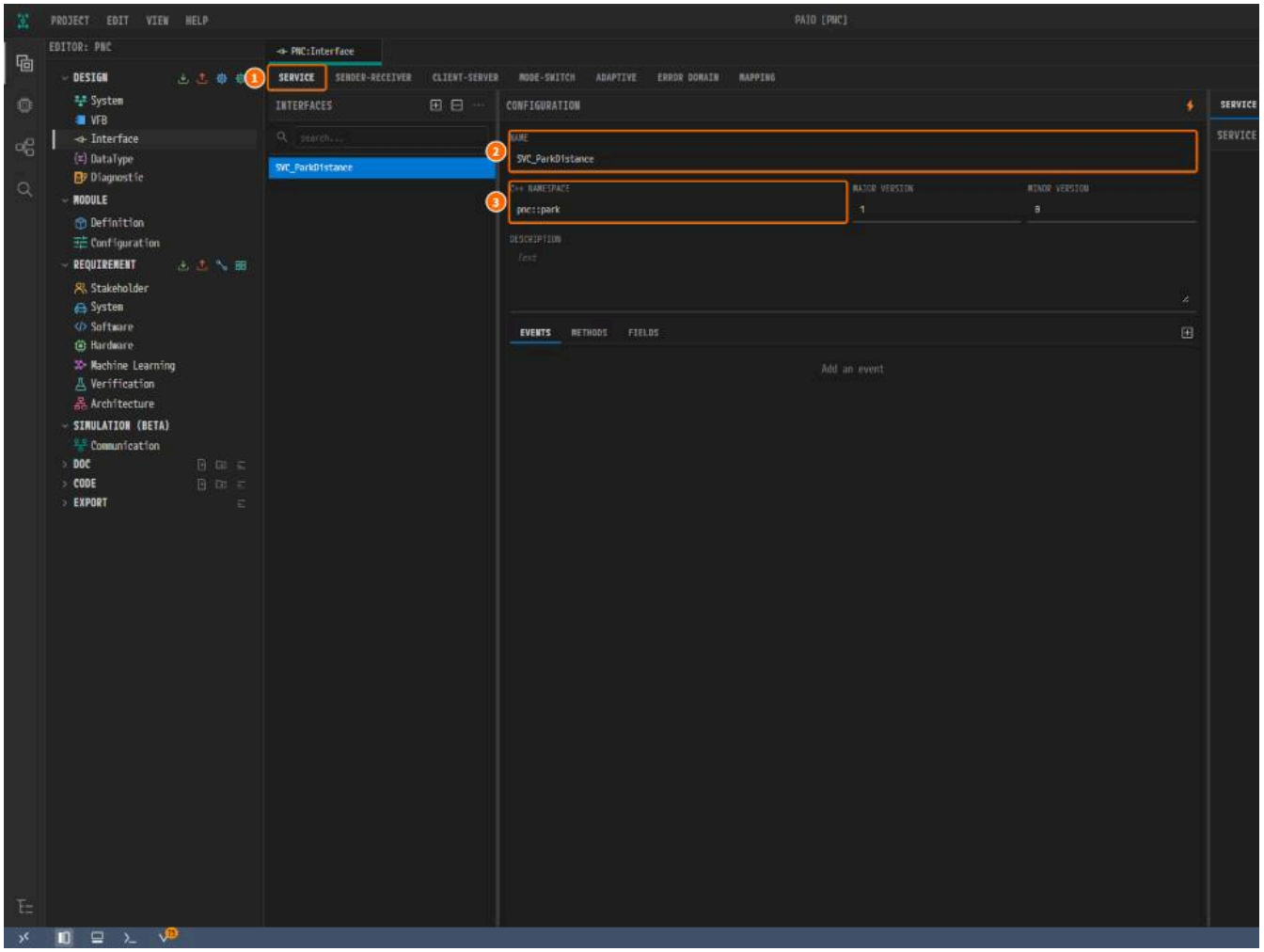


*그림 5-6. ① **SERVICE**, ② **NAME**, ③ **C++ NAMESPACE**.*

이벤트 추가

1. **EVENTS** 서브탭을 누른다.

2. 서브탭 줄 오른쪽 끝의 **+**를 누른다. `Event_0` 행(`uint8`)이 추가된다.

3. **NAME**에 `DistanceUpdate`를 입력한다. 타입은 `uint8` 그대로 둔다.

4. 다시 **+**를 누르고 `SensorFault`를 입력한 뒤 **TYPE**에서 `boolean`을 고른다.

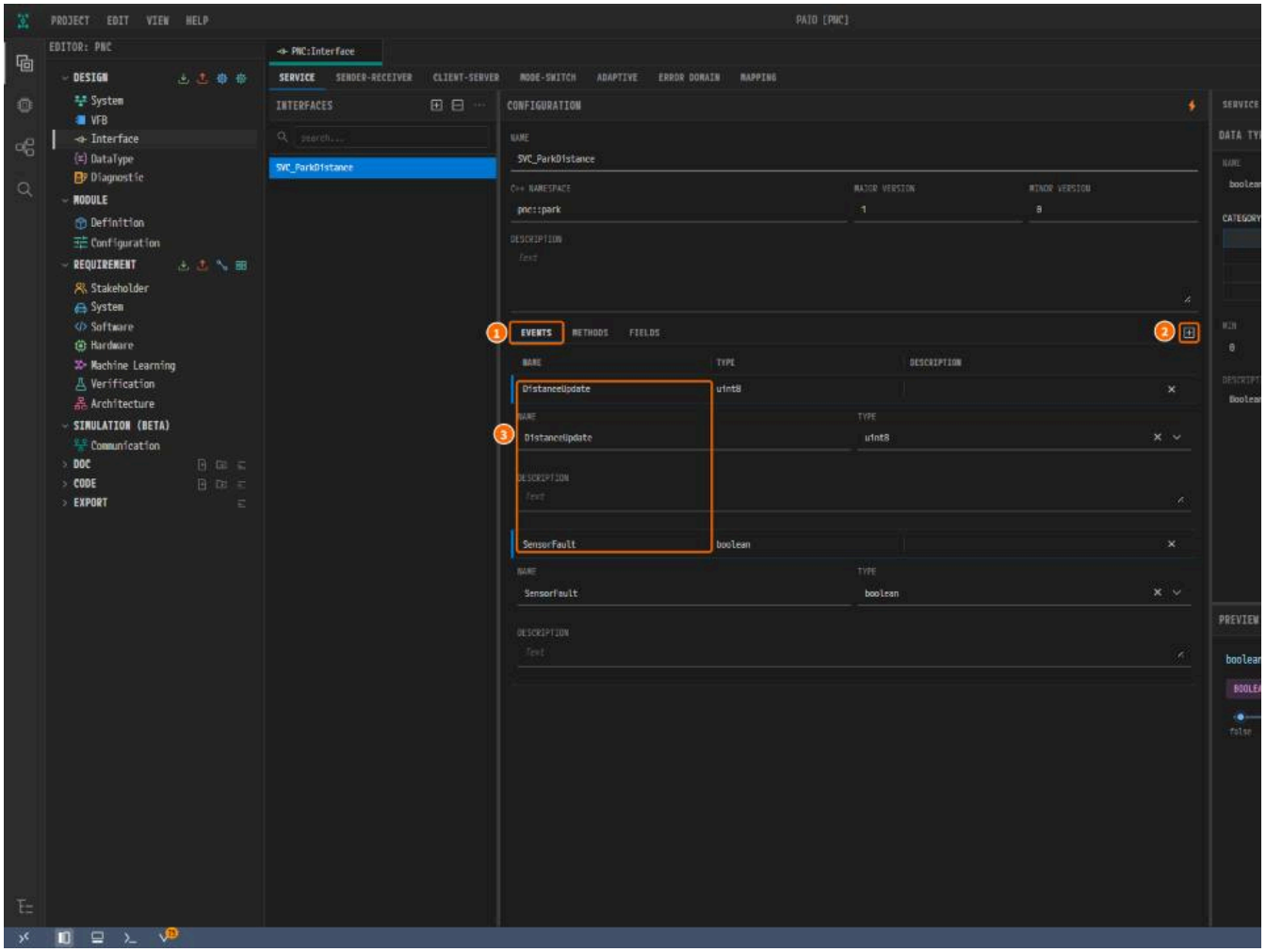


*그림 5-7. ① **EVENTS**, ② **+**, ③ `DistanceUpdate`와 `SensorFault`.*

> **참고.** **TYPE**을 고르면 오른쪽 패널이 **DATA TYPE**으로 바뀐다. **SERVICE DEPLOYMENT**를 눌러 돌아간다.

메서드 추가

1. **METHODS** 서브탭을 누른다.
2. 서브탭 줄 오른쪽 끝의 **+**를 누른다. `Method_0` 행이 추가된다.
3. **NAME**에 `SetSensitivity`를 입력한다.
4. **METHOD TYPE**은 **REQUEST AND RESPONSE** 그대로 둔다.

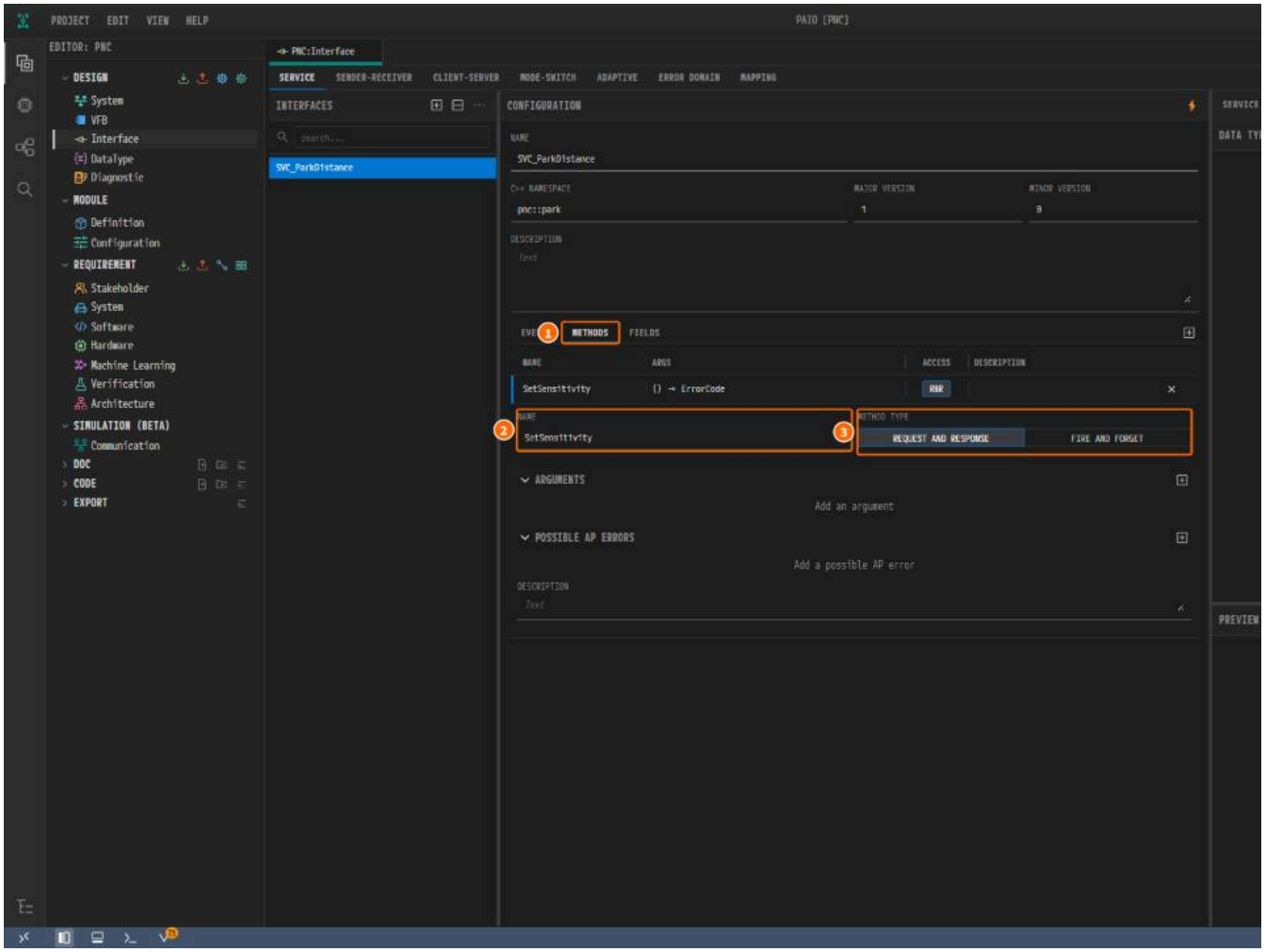


*그림 5-8. ① **METHODS**, ② **NAME**, ③ **METHOD TYPE**. 응답이 없는 메서드는 **FIRE AND FORGET**을 쓴다.*

SOME/IP 배포 추가하기

배포는 추가하는 시점에 있는 이벤트와 메서드로 만들어지므로 맨 마지막에 추가한다.

1. 오른쪽 패널에서 **SERVICE DEPLOYMENT** 서브탭을 누른다.
2. **SERVICE DEPLOYMENT** 오른쪽의 **+**를 누른다.

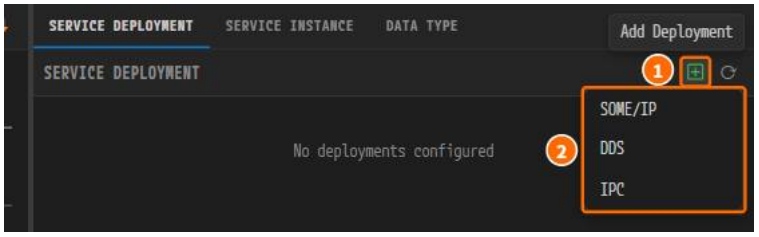


*그림 5-9. ① **+**, ② 프로토콜 메뉴.*

3. **SOME/IP**를 고른다. 접힌 카드 **SOME/IP** `SVC_ParkDistance`가 생긴다.
4. 카드 머리글을 눌러 펼친다.
5. **SERVICE ID** 옆의 **HEX**를 누른다.
6. **SERVICE ID**에 `1001`을 입력하고 `Tab`을 누른다. `0x1001`로 표시된다.
7. **MAJOR VERSION** `1`, **MINOR VERSION** `0`을 확인한다.

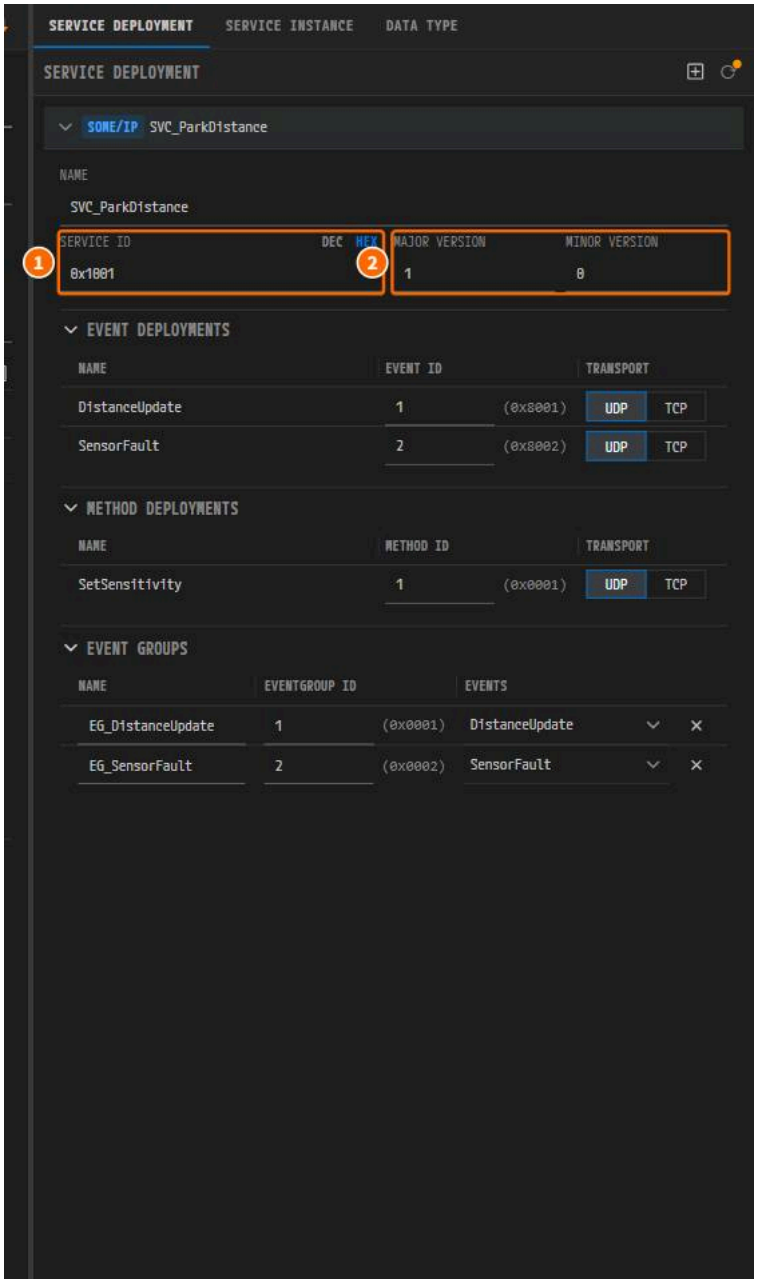


*그림 5-10. ① **SERVICE ID**, ② 버전. 이벤트와 메서드 배포는 자동으로 채워지고, 이벤트마다 이벤트 그룹이 하나씩 생긴다.*

8. 두 이벤트 그룹을 `EG_SVC_ParkDistance`(**EVENTGROUP ID** `1`) 하나로 합친다.

   i. **EVENT GROUPS**에서 `EG_SensorFault` 행의 **x**를 누른다.
   ii. `EG_DistanceUpdate`의 이름을 `EG_SVC_ParkDistance`로 바꾼다. **EVENTGROUP ID**는 `1` 그대로 둔다.
   iii. 그 행의 **EVENTS** 칸을 눌러 `SensorFault`도 체크한다. `DistanceUpdate, SensorFault`로 표시된다.

제공자와 소비자 인스턴스 추가하기

1. 배포 카드 머리글에 마우스를 올린다. 오른쪽에 **+**가 나타난다.

2. **+**를 누르고 **Provider**를 고른다.



*그림 5-11. ① **+**, ② **Provider / Consumer**.*

3. 다시 **+**를 누르고 **Consumer**를 고른다.

4. 카드 머리글을 눌러 접는다.

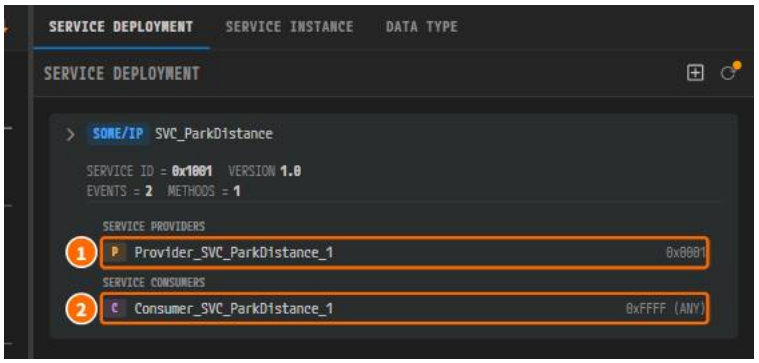


그림 5-12. ① 기본 이름의 제공자, ② 기본 이름의 소비자.

5. **SERVICE PROVIDERS**의 제공자 행을 누른다. **SERVICE INSTANCE** 서브탭이 열린다.

6. **NAME**을 `ParkDistance_Provider`로 바꾼다.

7. **INSTANCE ID**에 `1`을 입력하고 `Tab`을 누른다.

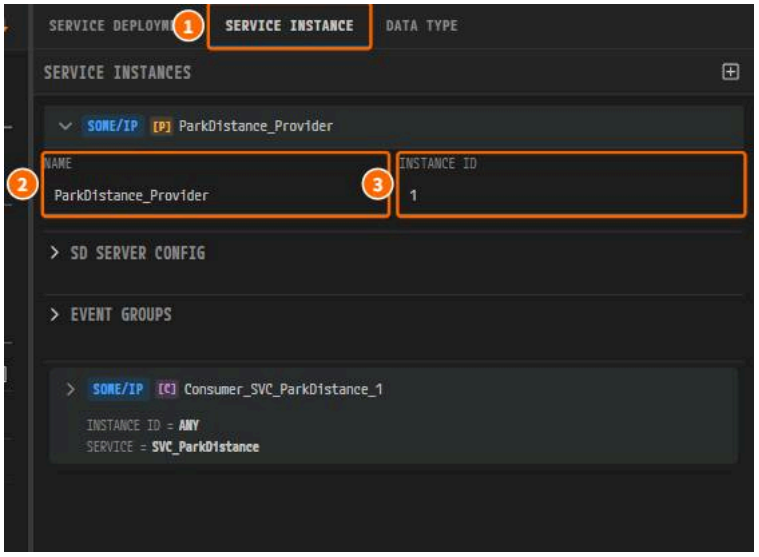


*그림 5-13. ① **SERVICE INSTANCE**, ② **NAME**, ③ **INSTANCE ID**.*

8. 제공자 카드를 접고 소비자 카드를 펼친다.

9. **NAME**을 `ParkDistance_Consumer`로 바꾼다.

10. **REQUIRED INSTANCE ID**에 `1`을 입력한다.

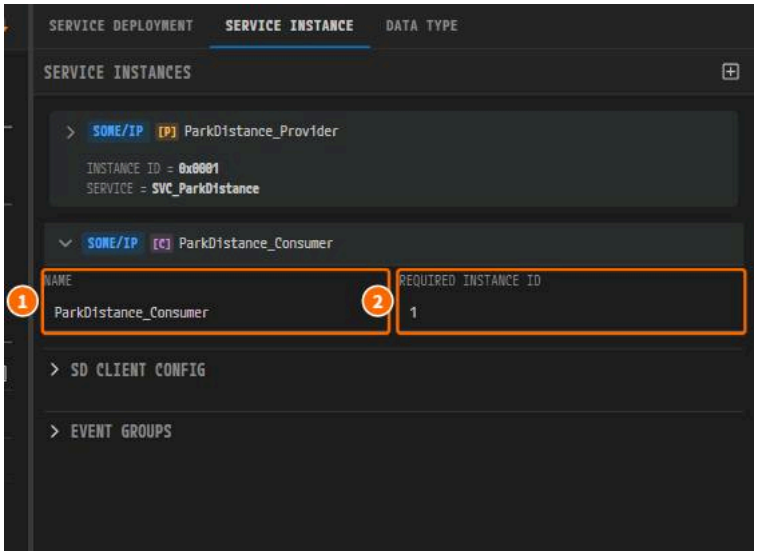


*그림 5-14. ① **NAME**, ② **REQUIRED INSTANCE ID**.*

11. `Ctrl+S`로 저장한다.

> **참고.** PAIO는 `SVC_ParkDistance_SR`과 PDU 4개(`SVC_ParkDistance_*_PDU`)도 만든다. 완성 예제에는 없으며 검증 오류를 내지 않는다.

나머지 서비스 인터페이스와 인스턴스

`SVC_VehicleState`와 `SVC_ComfortControl`도 표를 보고 같은 방법으로 만든다. 버전은 모두 `1.0`, 전송은 모두 UDP이다. 이벤트 그룹은 표의 그룹 하나로 정리하고, `SVC_Co`

서비스	C++ NAMESPACE	이벤트 (타입, EVENT ID)	메서드 (METHOD ID, METHOD TYPE)	
`SVC_ParkDistance`	`pnc::park`	`DistanceUpdate` (uint8, 1), `SensorFault` (boolean, 2)	`SetSensitivity` (1, REQUEST AND RESPONSE)	`0` (4
`SVC_VehicleState`	`pnc::vehicle`	`GearPosition` (uint8, 1), `TrailerAttached` (boolean, 2)	없음	`0` (4
`SVC_ComfortControl`	`pnc::comfort`	`DoorStatus` (uint8, 1)	`LockAll` (1, REQUEST AND RESPONSE), `FoldMirrors` (2, FIRE AND FORGET)	`0` (4

인스턴스	역할	배포	INSTANCE ID / REQUIRED INSTANCE ID
`ParkDistance_Provider`	Provider	`SVC_ParkDistance`	`1` (0x0001)
`ParkDistance_Consumer`	Consumer	`SVC_ParkDistance`	`1`
`VehicleState_Provider`	Provider	`SVC_VehicleState`	`1` (0x0001)
`VehicleState_Consumer`	Consumer	`SVC_VehicleState`	`1`
`ComfortControl_Provider_West`	Provider	`SVC_ComfortControl`	`1` (0x0001)
`ComfortControl_Provider_East`	Provider	`SVC_ComfortControl`	`2` (0x0002)
`ComfortControl_Consumer`	Consumer	`SVC_ComfortControl`	기본값 `0xFFFF (ANY)`

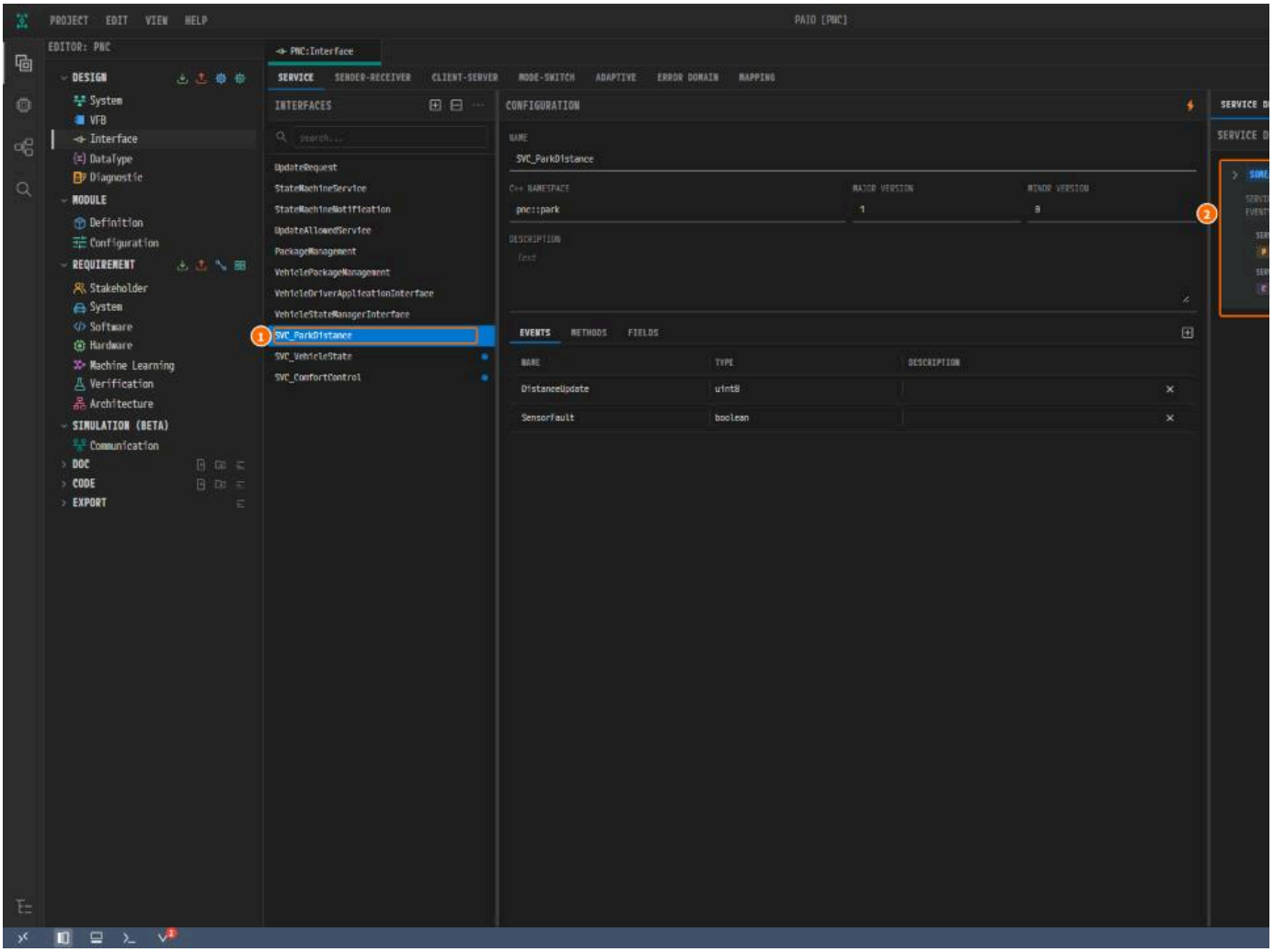


그림 5-15. ① `SVC_ParkDistance`, ② 배포 요약. `UpdateRequest` 등의 서비스는 뒤의 PARA 플랫폼 임포트와 PNC 요청 포트에서 추가된다.

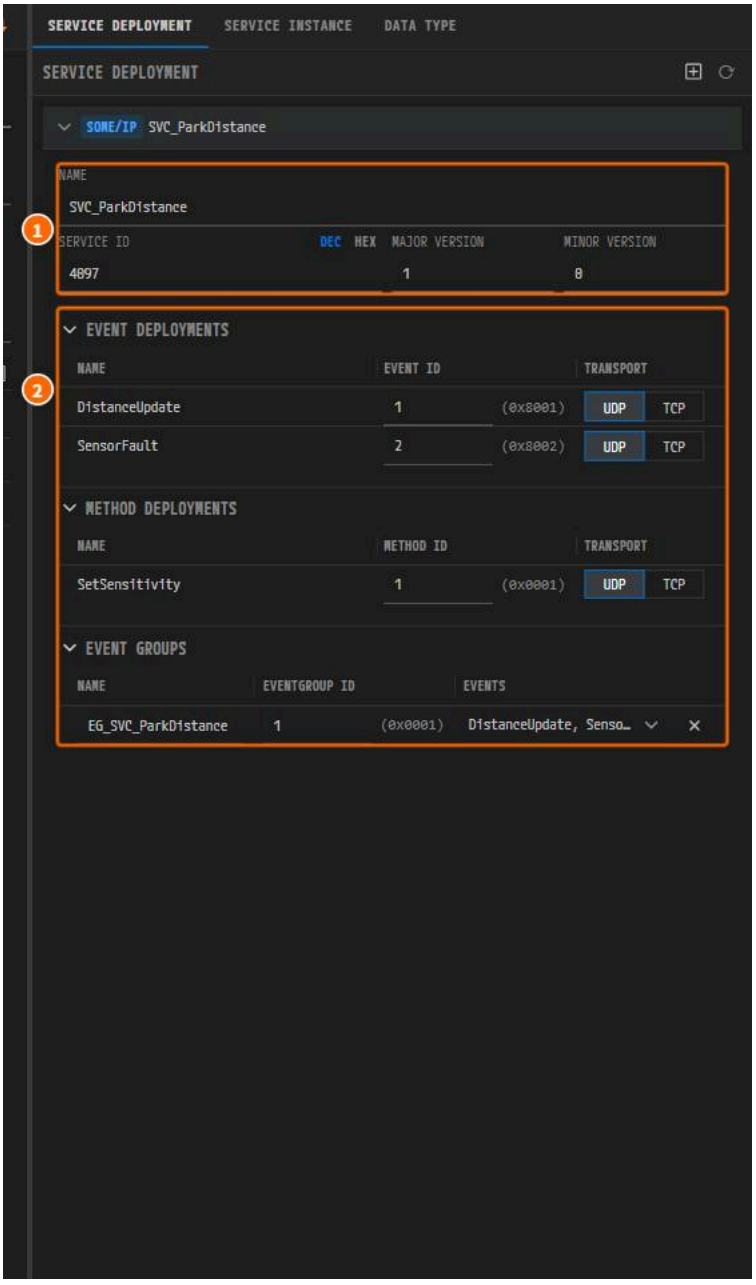


*그림 5-16. ① **NAME**, **SERVICE ID**(**DEC** 모드라 `4097`), 버전, ② 배포와 이벤트 그룹 `EG_SVC_ParkDistance`.*

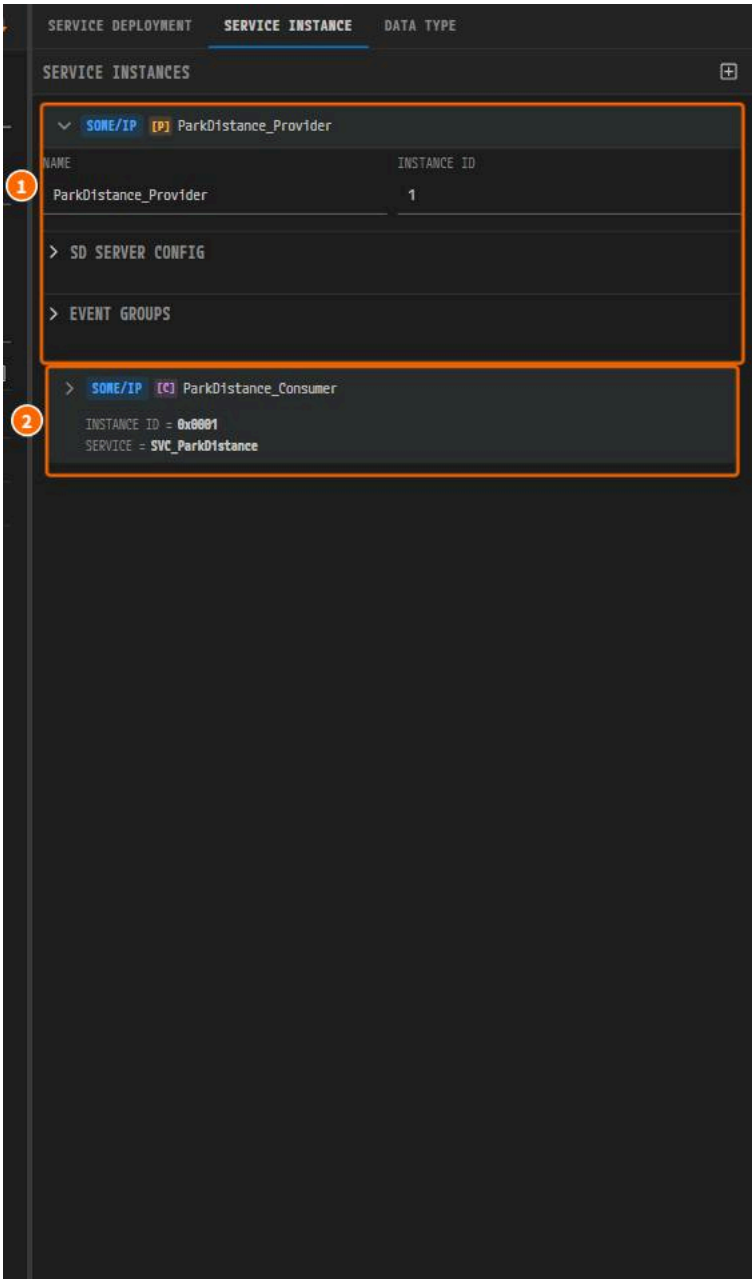


그림 5-17. ① `ParkDistance_Provider`, ② `ParkDistance_Consumer`.

> **참고.** 완성 예제는 다음 SD 설정(제공자는 **SD SERVER CONFIG**, 소비자는 **SD CLIENT CONFIG**)을 쓰며, 기본값과 다르다.

SD 설정	제공자 예제 값	소비자 예제 값
initial-delay min / max	0 / 0.01	0 / 0.01
repetition-base-delay	0.15	0.02
repetition-max	2	5
offer-cyclic-delay	3	—
ttl	10	10

커넥터에 IP 주소와 Application Endpoint 넣기

먼저 `HPC_Eth_Rear_Conn`을 따라간다.

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System**을 누른다. `PNC:System` 탭이 열린다.

2. `HPC` 아래쪽 경계의 커넥터 `HPC_Eth_Rear_Conn`을 누른다(그림 5-22의 ①).

3. **NETWORK ENDPOINT**를 펼치고 **IPv4**를 그대로 둔다.

4. **UNICAST IPV4 ADDRESS**에 `192.168.200.1`, **NETWORK MASK**에 `255.255.255.0`을 입력한다. **ADDRESS SOURCE**는 **FIXED**를 고른다.



*그림 5-18. ① **IPv4**, ② **UNICAST IPV4 ADDRESS**, ③ **ADDRESS SOURCE**, ④ **NETWORK MASK**.*

5. **APPLICATION ENDPOINTS** 머리글의 **+**를 누른다. 행(TCP)이 하나 추가된다.

6. 행을 펼치고 **TYPE**을 **UDP**로 바꾼 뒤 **PORT NUMBER**에 `30501`을 입력한다.

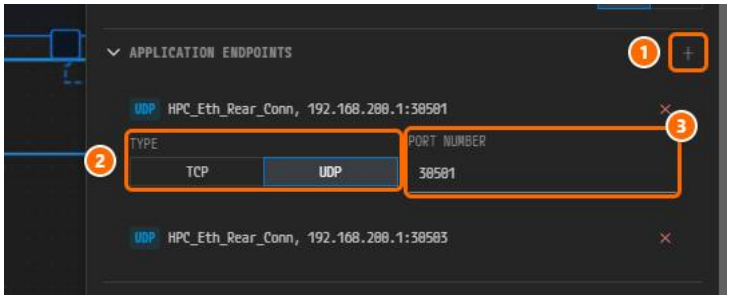


*그림 5-19. ① **+**, ② **TYPE**, ③ **PORT NUMBER**.*

7. 같은 방법으로 두 번째 UDP 엔드포인트 `30503`을 추가한다.

나머지 커넥터도 표를 보고 같은 방법으로 입력한다. **NETWORK MASK**는 모두 `255.255.255.0`, **ADDRESS SOURCE**는 모두 **FIXED**이다.

노드 / 커넥터	UNICAST IPV4 ADDRESS	UDP Application Endpoint (PORT NUMBER)
`HPC` / `HPC_Eth_Body_Conn`	`192.168.100.1`	`30501`
`HPC` / `HPC_Eth_Rear_Conn`	`192.168.200.1`	`30501`, `30503`
`WestZone` / `WestZone_Eth_Conn`	`192.168.100.11`	`30502`
`EastZone` / `EastZone_Eth_Conn`	`192.168.100.13`	`30502`
`SouthZone` / `SouthZone_Eth_Conn`	`192.168.200.12`	`30502`, `30503`

> **참고.** 엔드포인트 이름은 PAIO가 붙이므로 완성 예제와 달라도 된다.

커넥터에 서비스 인스턴스 바인딩하기

1. `HPC_Eth_Rear_Conn`을 선택한 채 **SERVICE INSTANCES** 머리글의 **+**를 누른다. **Select Service Instance** 창이 열린다.
2. `ParkDistance_Consumer`를 체크하고 **Bind**를 누른다.

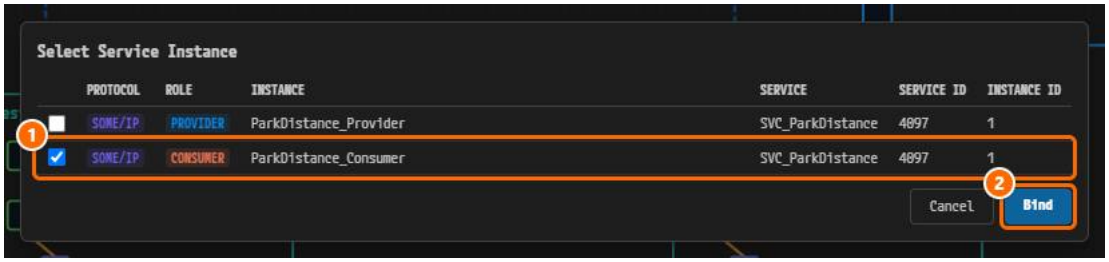


*그림 5-20. ① `ParkDistance_Consumer`, ② **Bind**.*

3. 새 카드 **SOME/IP C** `ParkDistance_Consumer`를 눌러 펼친다.
4. **UDP PORT** 옆의 **▾**를 누른다.
5. `30503`을 고른다.



*그림 5-21. ① **UDP PORT**, ② `30503`. 이 커넥터에서 `VehicleState_Provider`가 `30501`을 쓴다.*

6. 캔버스의 다른 곳을 누른 뒤 `HPC_Eth_Rear_Conn`을 다시 누른다. 요약이 `UDP:30503`으로 표시된다.

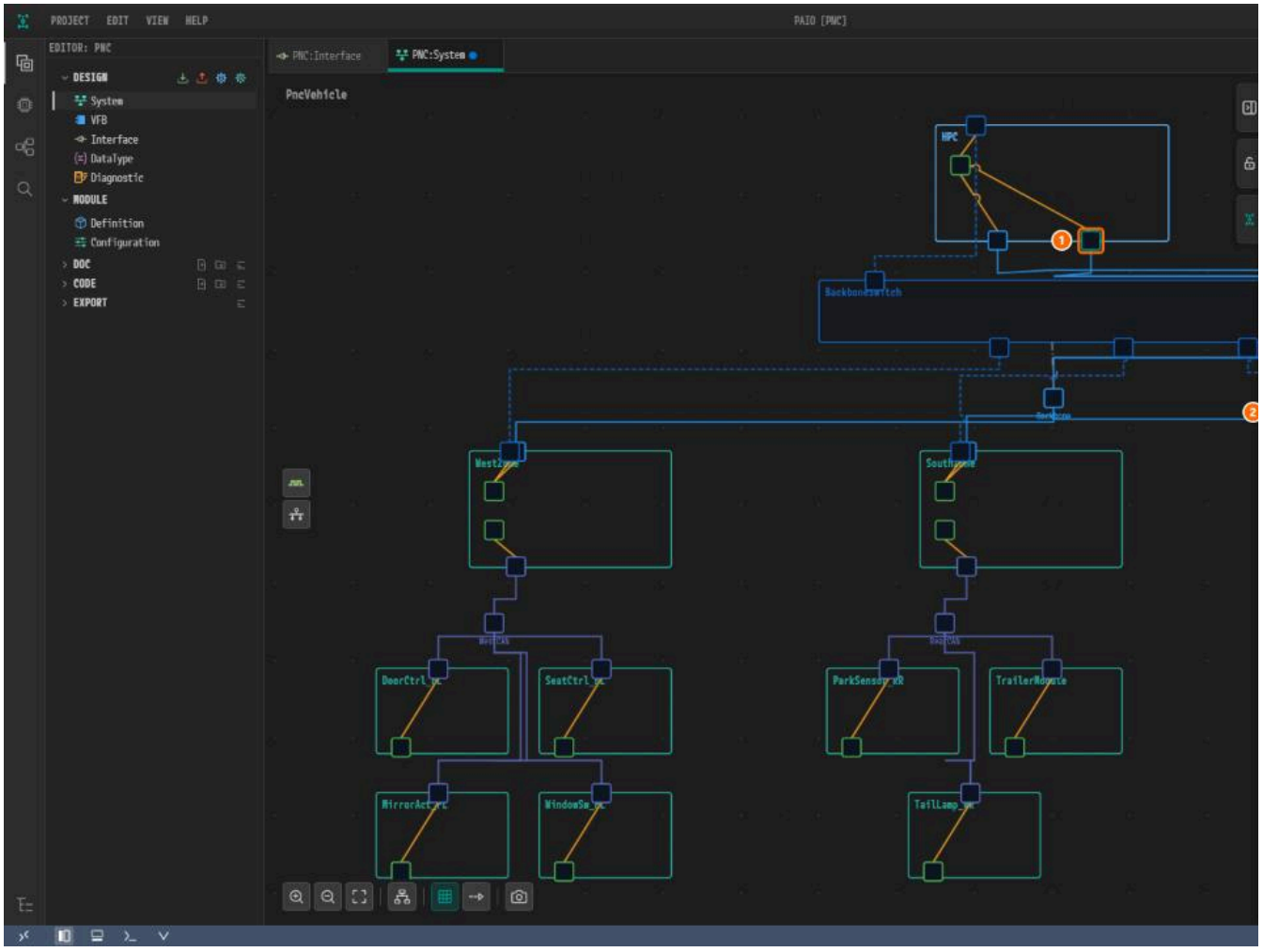


그림 5-22. ① `HPC_Eth_Rear_Conn`, ② `ParkDistance_Consumer` `UDP:30503`.

7. `Ctrl+S`로 저장한다.

> **참고.** UDP 포트를 정하기 전에는 검증이 이 인스턴스의 오류 1건을 보고한다. 5에서 사라진다.

나머지 바인딩

인스턴스 7개를 모두 표를 보고 같은 방법으로 바인딩한다.

노드 / 커넥터	바인딩할 인스턴스 (역할, UDP PORT)
`HPC` / `HPC_Eth_Body_Conn`	`ComfortControl_Consumer` (C, 30501)
`HPC` / `HPC_Eth_Rear_Conn`	`VehicleState_Provider` (P, 30501), `ParkDistance_Consumer` (C, **30503**)
`WestZone` / `WestZone_Eth_Conn`	`ComfortControl_Provider_West` (P, 30502)
`EastZone` / `EastZone_Eth_Conn`	`ComfortControl_Provider_East` (P, 30502)
`SouthZone` / `SouthZone_Eth_Conn`	`ParkDistance_Provider` (P, 30502), `VehicleState_Consumer` (C, **30503**)

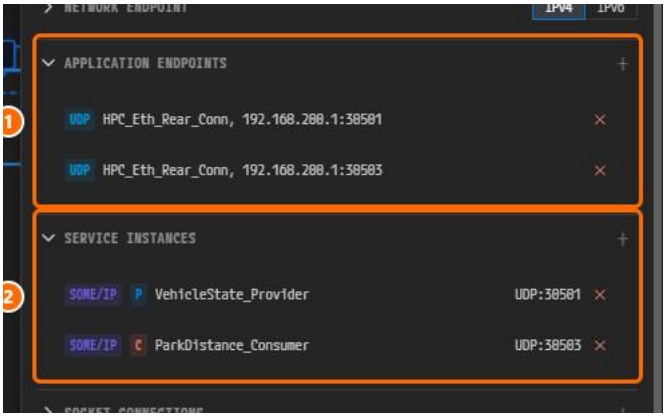


그림 5-23. ① UDP 엔드포인트 두 개, ② 바인딩된 인스턴스 두 개.

> **주의.** 같은 IP 주소와 UDP 포트에 인스턴스가 둘이면 검증 오류 `paio_someip_address_port_unique`가 난다.

왜 이렇게 설계하는가

- **PNC와 서비스.** PNC는 서비스 인스턴스를 참조해 그 PNC가 깨어 있어야 쓸 수 있는 서비스를 기록한다. `PNC_ParkAssist`가 잠들면 `SVC_ParkDistance`도 쓸 수 없다.
- **바인딩.** 바인딩된 인스턴스만 머신과 잇는 `SomeipServiceInstanceToMachineMapping`이 익스포트된다.
- **포트 분리.** 한 커넥터를 함께 쓰는 인스턴스는 UDP 포트가 서로 달라야 한다.

확인하기

- [ ] **SENDER-RECEIVER** 탭에 `SRI_*` 인터페이스 11개가 있다.
- [ ] **SERVICE** 탭에 `SVC_ParkDistance`, `SVC_VehicleState`, `SVC_ComfortControl`이 Service ID `0x1001`, `0x1002`, `0x1003`, 버전 `1.0`으로 있다.
- [ ] 제공자 4개와 소비자 3개가 표의 이름으로 있다.
- [ ] `HPC_Eth_Rear_Conn`에 `VehicleState_Provider` `UDP:30501`과 `ParkDistance_Consumer` `UDP:30503`이 보인다.
- [ ] 두 편집기 탭에 ● 표시가 없다.

다음 단계

VFB와 VFC(PortGroup)에서 기능마다 SWC 포트를 PortGroup으로 묶어 VFC를 만든다.

VFB와 VFC(PortGroup)

이 장에서 할 일

이 장에서는 VFB 편집기에서 노드마다 컴포지션을 하나씩 만든다. 대표 예는 `ParkSensor_RR`과 그 안의 원자 SWC `ParkDistanceSensing`이다. 이어서 AUTOSAR가 PortGro
들고 PNC Overlay로 확인한다.

시작하기 전에

- 인터페이스와 SOME/IP 서비스를 마친다. 포트는 5장에서 만든 `SRI_ParkDistance`를 쓴다.
- VFB 루트 `PncVehicle`은 비어 있다. 컴포지션을 **14개** 만든다. `ParkSensor_RR`은 단계별로, 나머지 13개는 나머지 컴포지션 13개의 표를 보고 만든다.
- `ParkSensor_RR`은 `SouthZone`과 이어지므로 **표의 13개를 먼저 만든다**. 그림도 그 상태에서 찍었다.
- System 편집기에 노드 `ParkSensor_RR [ECU]`가 있다(토폴로지 설계에서 만든 것).

항목	값
컴포지션(**NAME**과 **AS PROTOTYPE**)	`ParkSensor_RR`
원자 SWC(**NAME**과 **AS PROTOTYPE**)	`ParkDistanceSensing`
SWC 포트와 경계 포트	`ParkDistance`, **PPort**, `SenderReceiver: SRI_ParkDistance`
어셈블리 상대	`SouthZone`의 `ParkDistance`(R)
ECU 매핑	`ParkSensor_RR [ECU]`
`ParkDistanceSensing`의 PortGroup	`VFC_ParkAssist`, **PARTIAL_NETWORKING**, OUTER PORTS `ParkDistance`

SWC와 컴포지션 만들기

컴포지션 만들기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB**를 누른다. **PNC:VFB** 탭이 열린다.
2. 캔버스 아래 도구막대의 네 번째 버튼 **Auto Layout**을 누른다. 필요하면 **Fit to View**를 누른다.
3. `SouthZone` 근처의 빈 캔버스를 우클릭하고 **Add Composition**을 누른다.

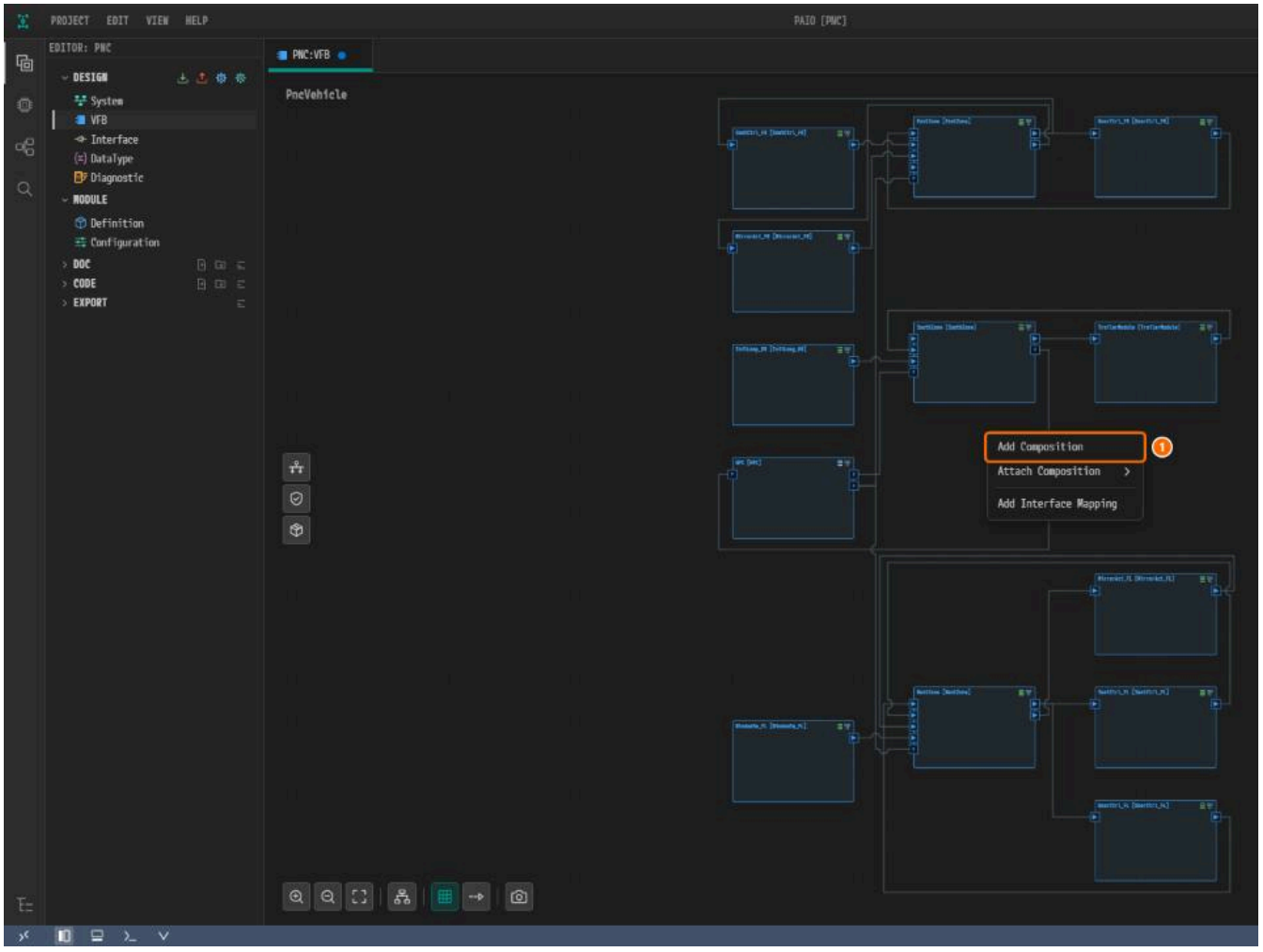


*그림 6-1. ① **Add Composition**.*

4. 새 노드 `Composition_0`을 누른다.

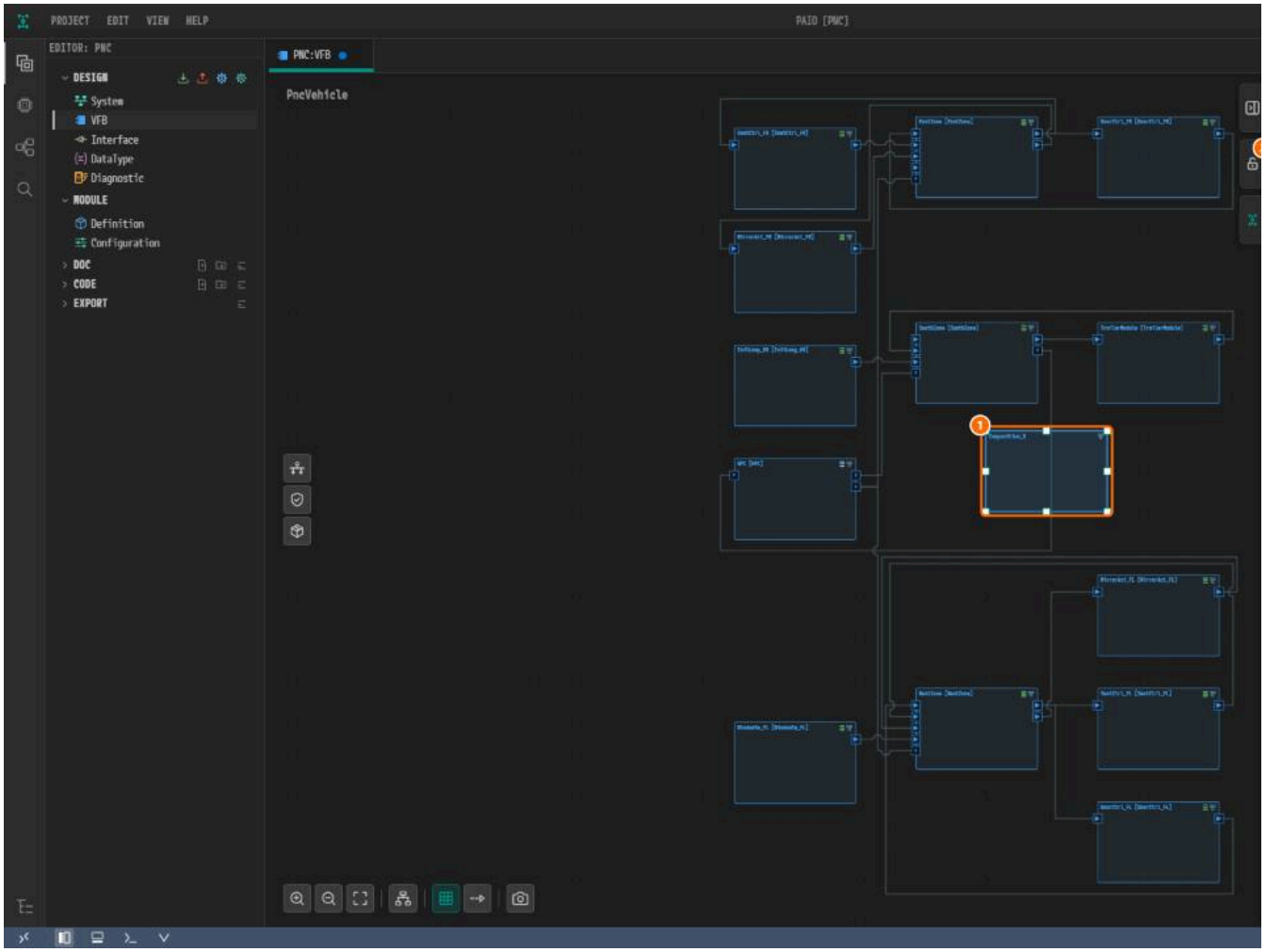


*그림 6-2. ① 노드, ② **NAME**, ③ **AS PROTOTYPE**.*

5. **NAME**과 **AS PROTOTYPE**에 각각 `ParkSensor_RR`을 입력하고 **Tab**을 누른다.

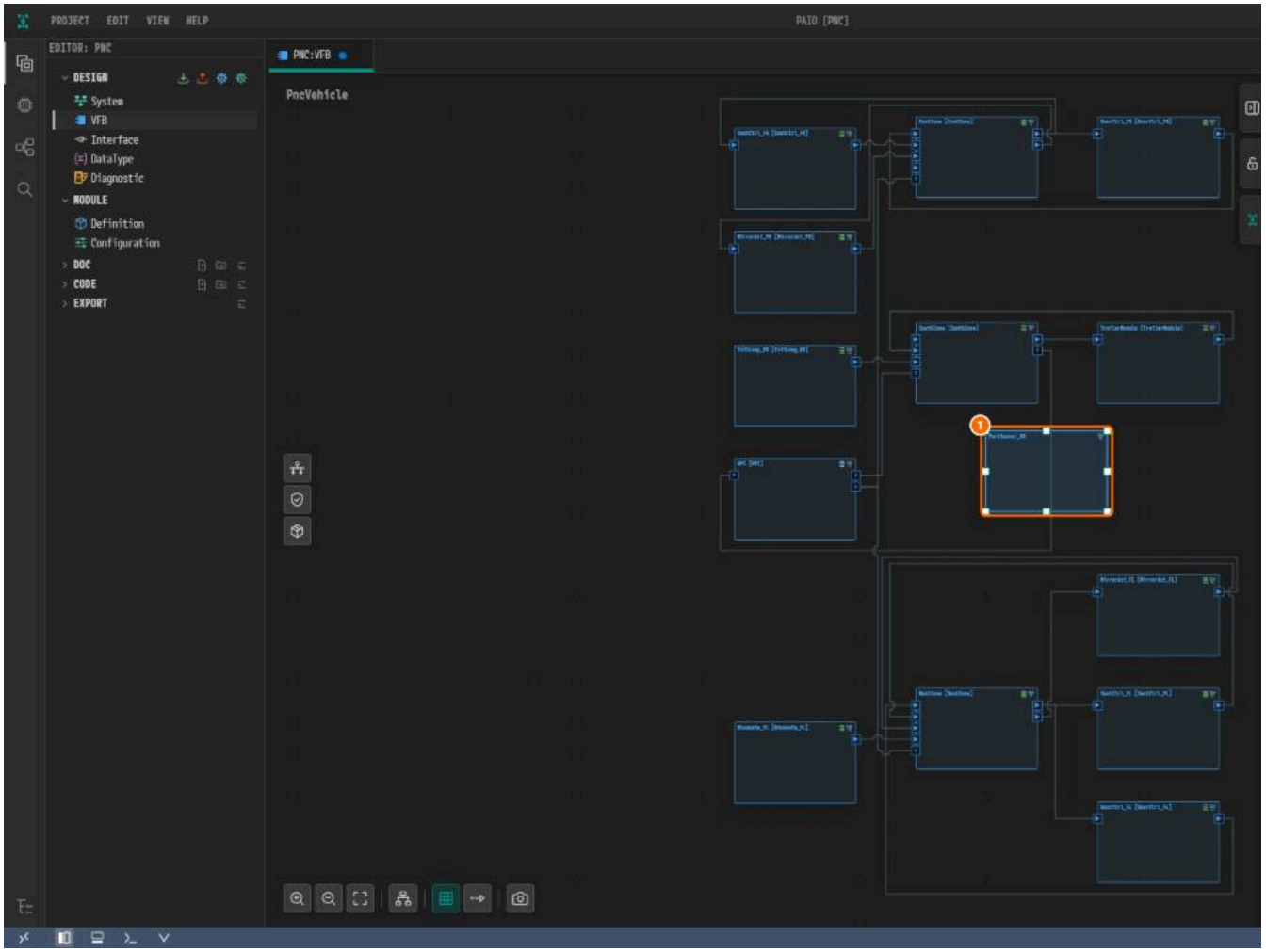


그림 6-3. ① 이름을 바꾼 노드.

> **주의.** **AS PROTOTYPE**은 **NAME**을 따라 바뀌지 않는다. 둘 다 고친다. SWC-ECU 매핑이 **AS PROTOTYPE**을 키로 쓴다.

원자 SWC 만들기

1. `ParkSensor_RR`을 더블클릭한다. 경로 **System > ParkSensor_RR**와 빈 프레임이 보인다. 나중에 경로의 **System**을 누르면 루트로 돌아간다.
2. 프레임 안을 우클릭한다. **Add SWC**에 마우스를 올리고 **Application**을 누른다.

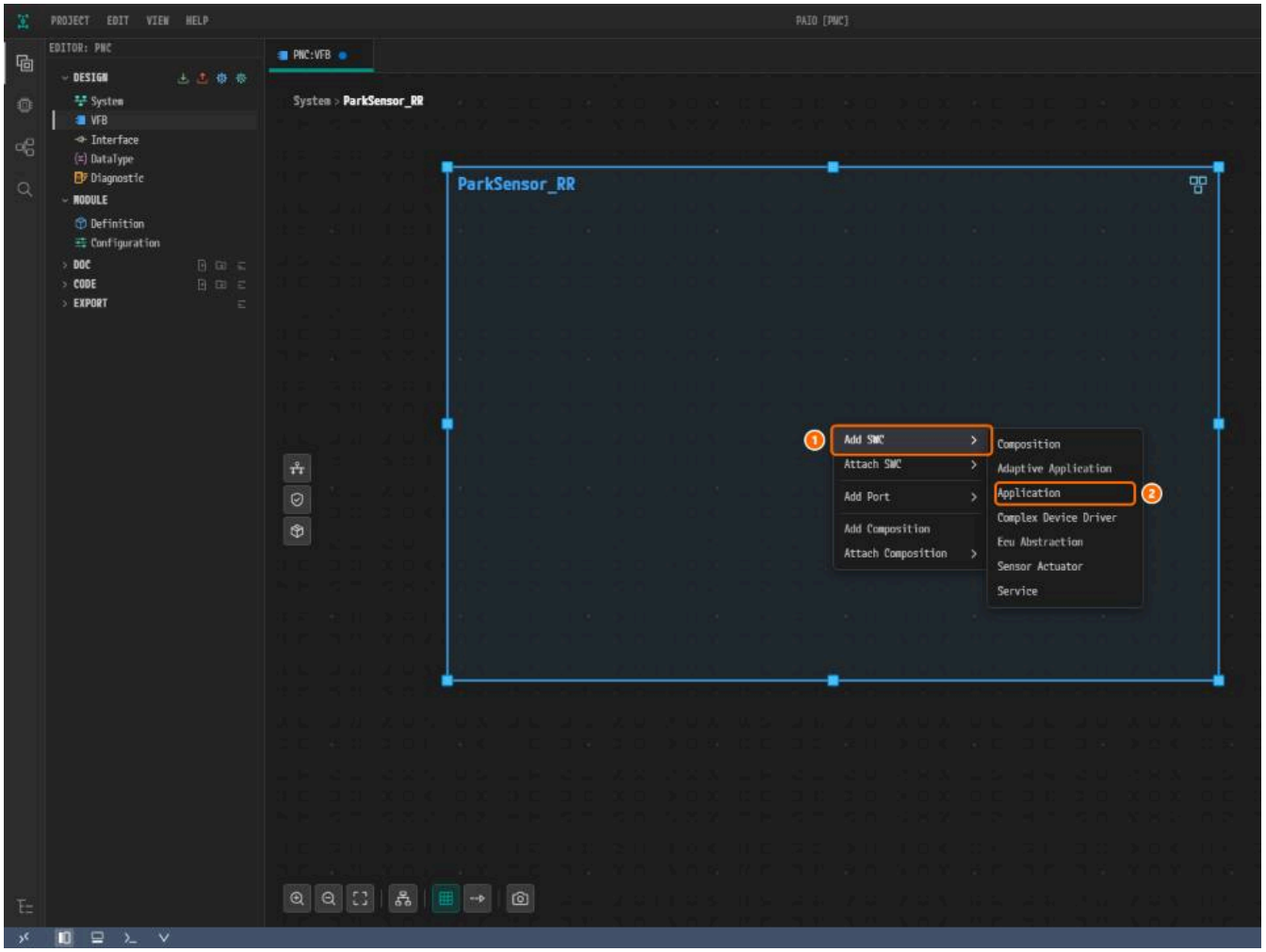


*그림 6-4. ① **Add SWC**, ② **Application**.*

3. 새 노드 `Application_0`을 누른다.

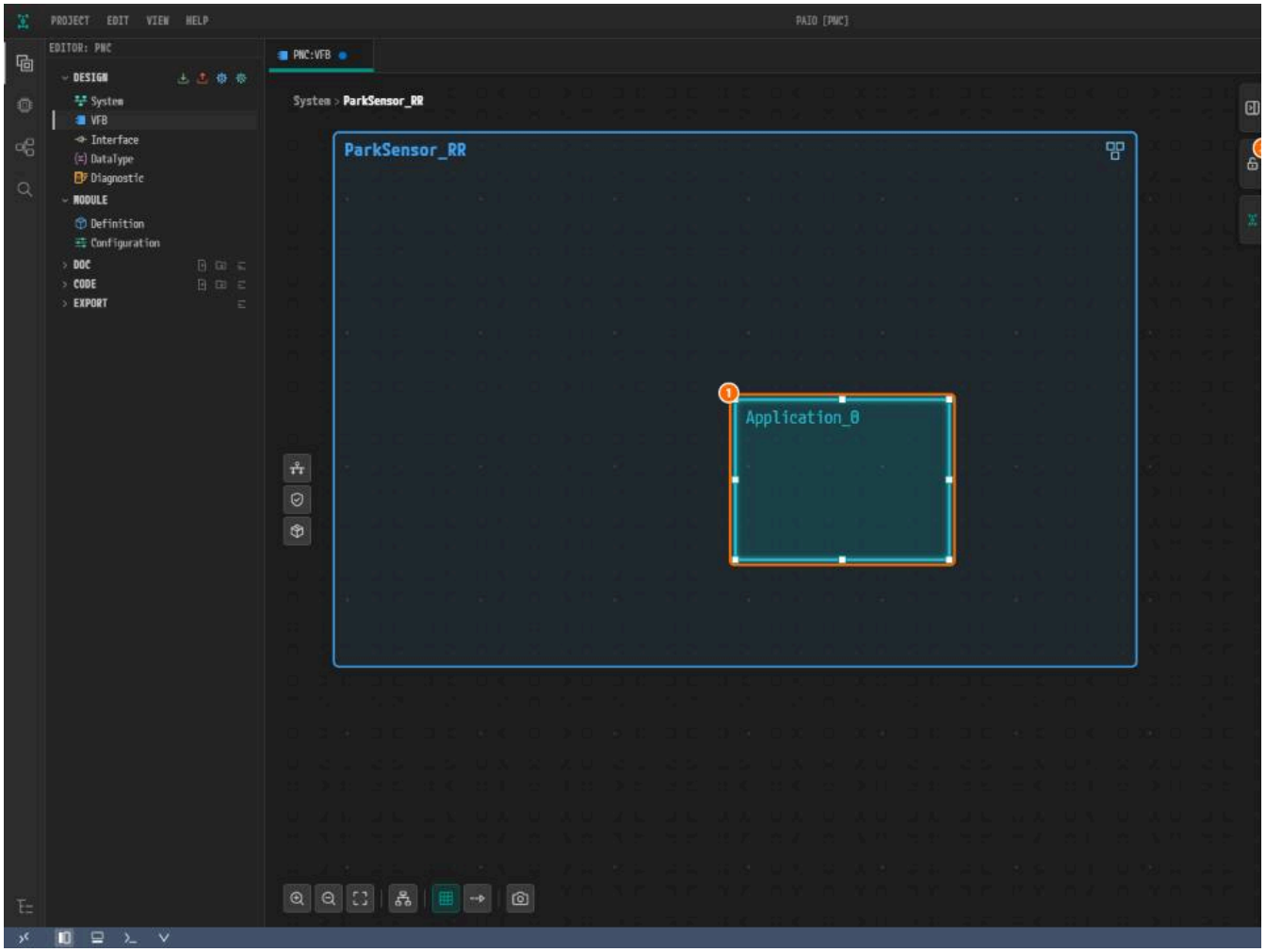


*그림 6-5. ① 노드, ② **NAME**, ③ **AS PROTOTYPE**(기본값이 다르다).*

4. **NAME**과 **AS PROTOTYPE**에 각각 `ParkDistanceSensing`을 입력하고 **Tab**을 누른다.

포트 만들기

1. `ParkDistanceSensing`의 오른쪽 가장자리 근처를 우클릭한다. **Add Port**에 마우스를 올리고 **PPort**를 누른다.

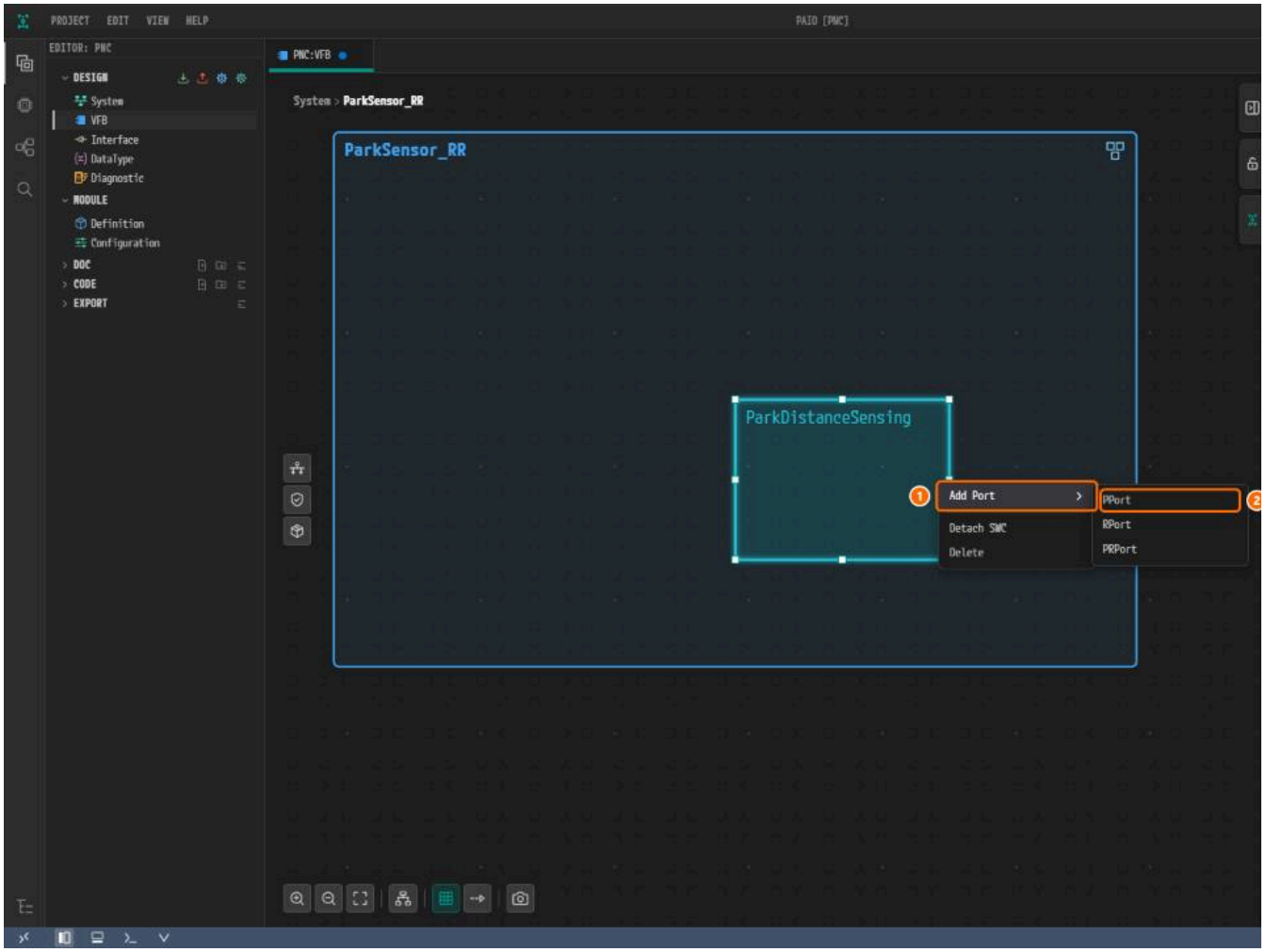


*그림 6-6. ① **Add Port**, ② **PPort**. 포트는 우클릭한 곳에서 가장 가까운 경계에 놓인다.*

2. 새 포트를 누른다. **NAME**에 `ParkDistance`를 입력하고 **PORT INTERFACE**에서 `SenderReceiver: SRI_ParkDistance`를 고른다.

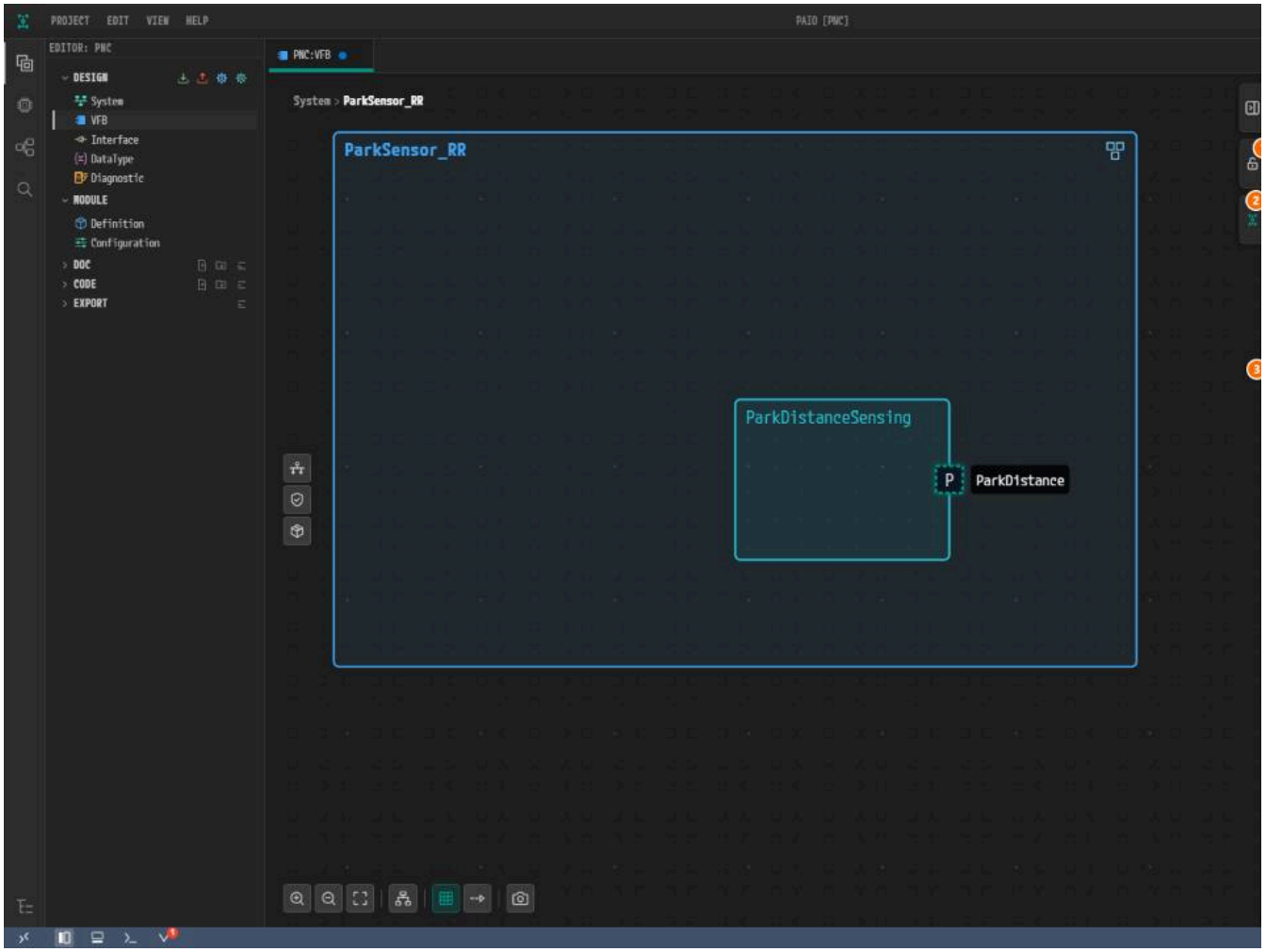


*그림 6-7. ① **NAME**, ② **PORT INTERFACE**, ③ `SenderReceiver: SRI_ParkDistance`.*

> **참고.** 포트를 눌렀는데 노드가 선택되면 빈 캔버스를 먼저 누른 뒤 포트를 누른다. 필요하면 **Ctrl**+휠로 확대한다.

3. 프레임의 오른쪽 가장자리를 우클릭해 같은 **NAME**과 **PORT INTERFACE**로 **PPort**를 추가한다.

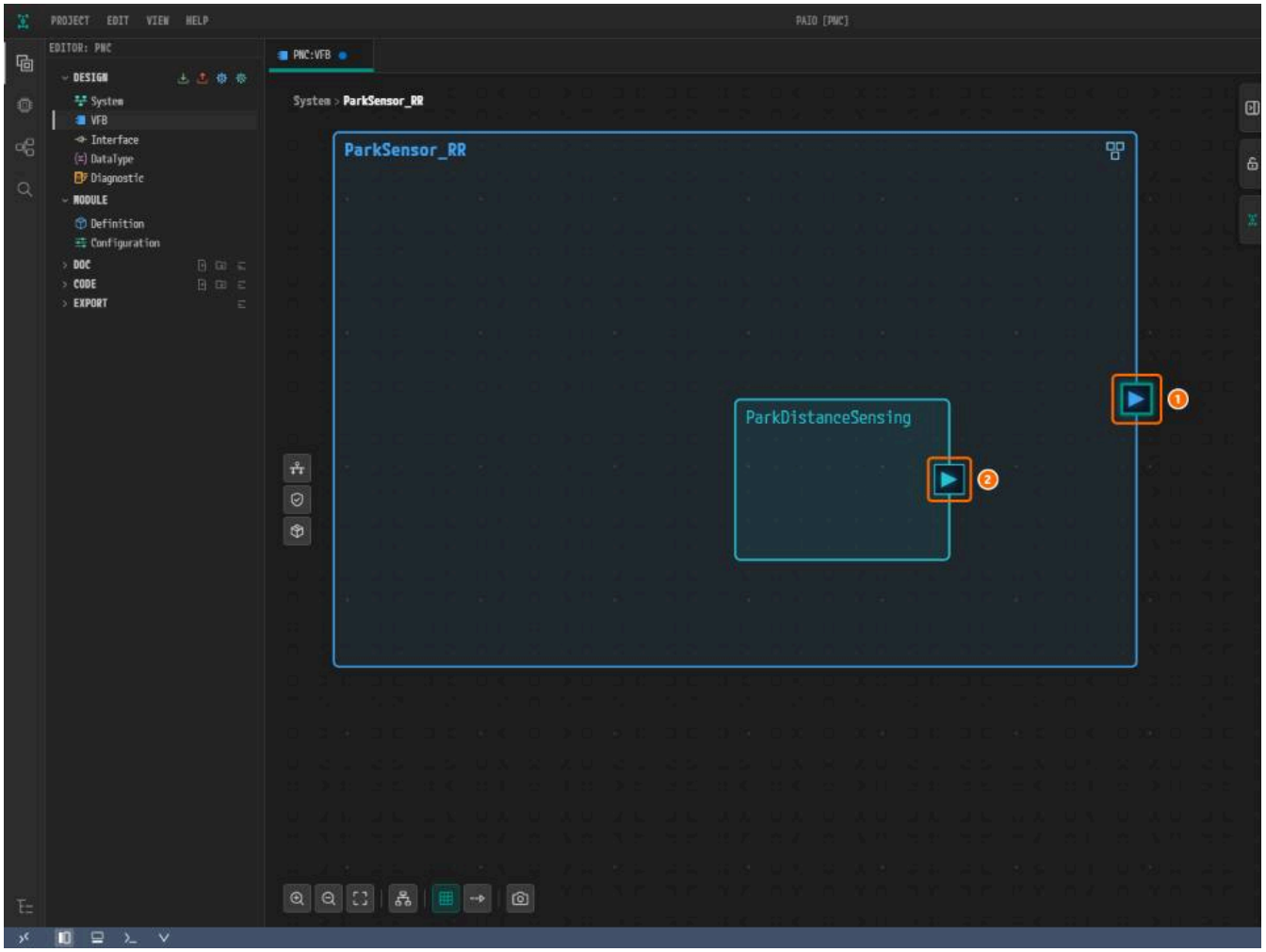


그림 6-8. ① 경계 포트, ② SWC 포트.

위임 커넥터로 잇기

1. 포트를 우클릭하면 **Connect to**에 후보가 나오지 않는다.

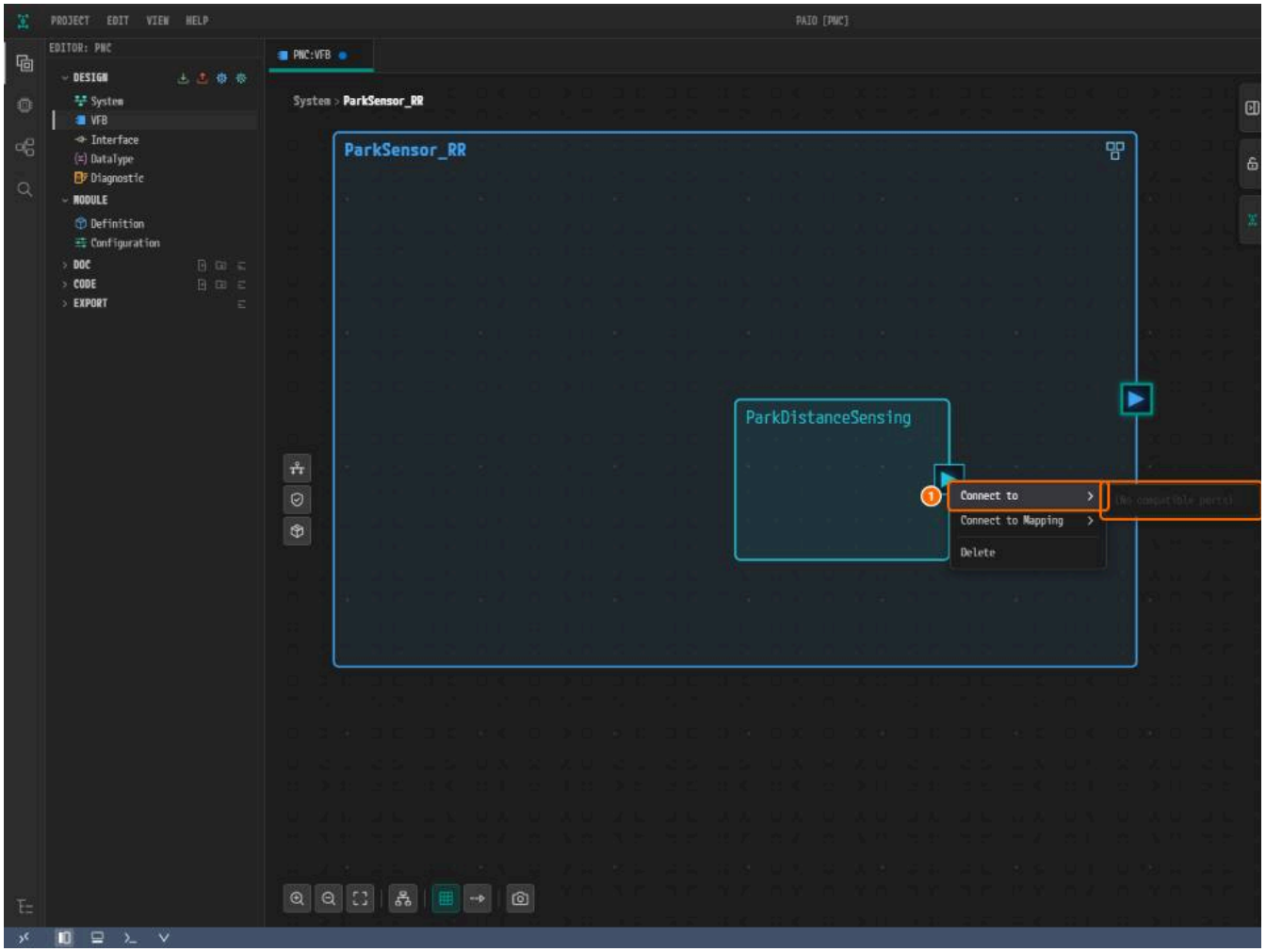


*그림 6-9. ① **Connect to**, ② `(No compatible ports)`.*

2. **Shift**를 누른 채 SWC 포트를 경계 포트까지 끌어다 놓는다. 선이 그려진다.

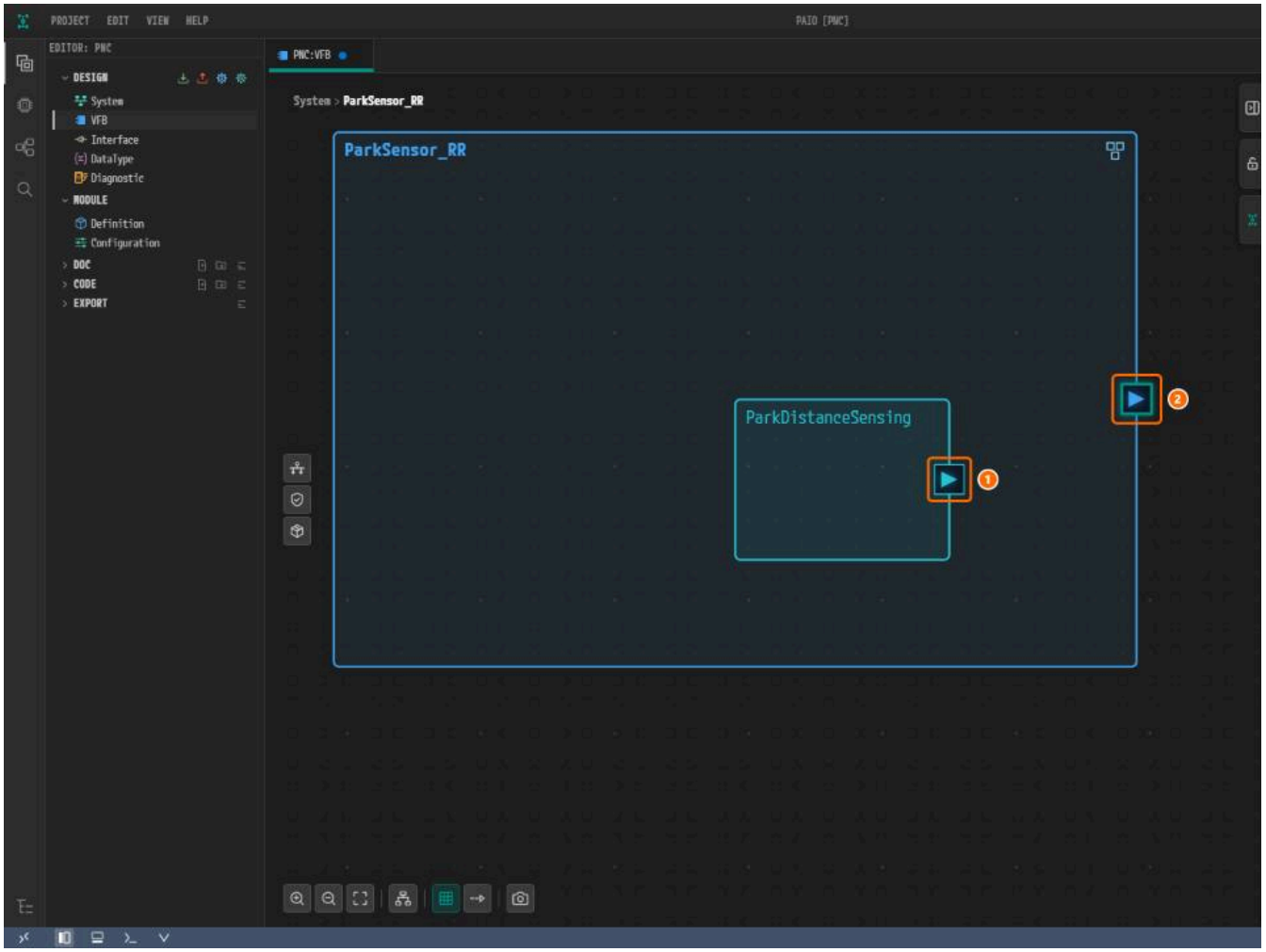


그림 6-10. ① SWC 포트에서 끌기 시작, ② 경계 포트에 놓기.

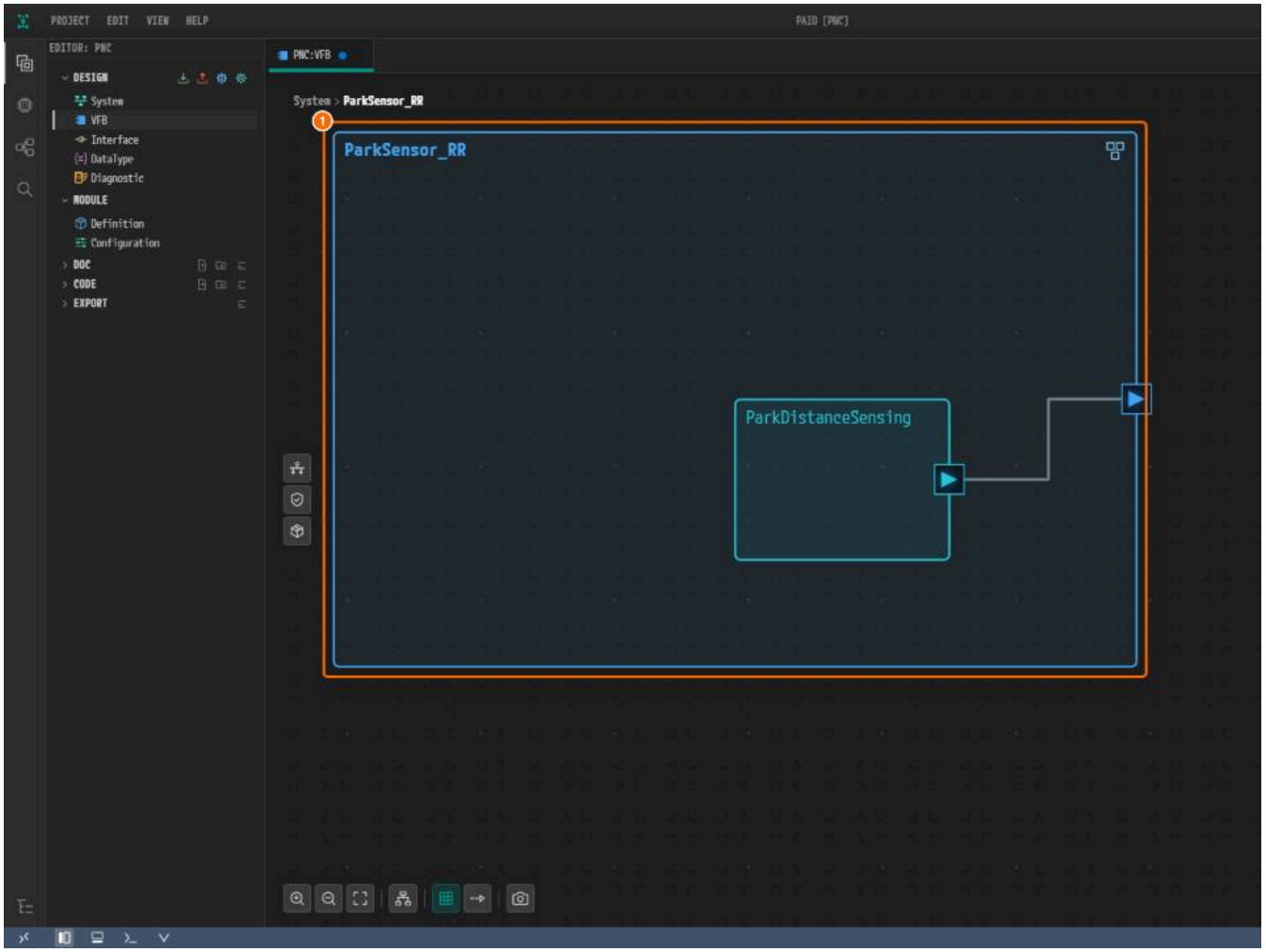


그림 6-11. ① 위임 커넥터가 있는 `ParkSensor_RR`.

어셈블리 커넥터로 잇기

1. 경로의 **System**을 눌러 루트로 돌아간다.
2. 여기서도 **Connect to**는 비어 있다.

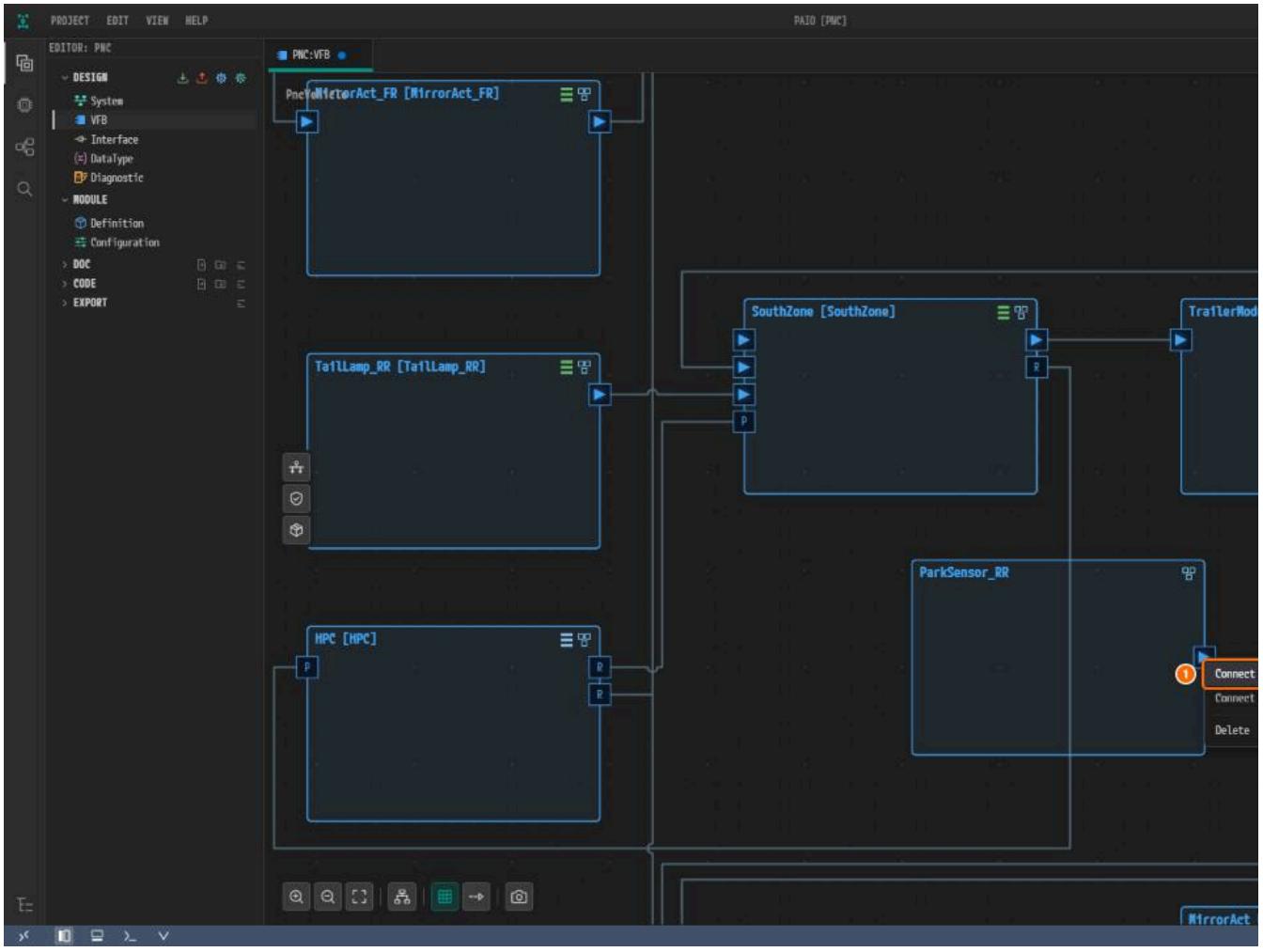


*그림 6-12. ① **Connect to**, ② `(No compatible ports)`.*

3. **Shift**를 누른 채 `ParkSensor_RR`의 포트를 `SouthZone`의 `ParkDistance`(R) 포트까지 끌어다 놓는다.

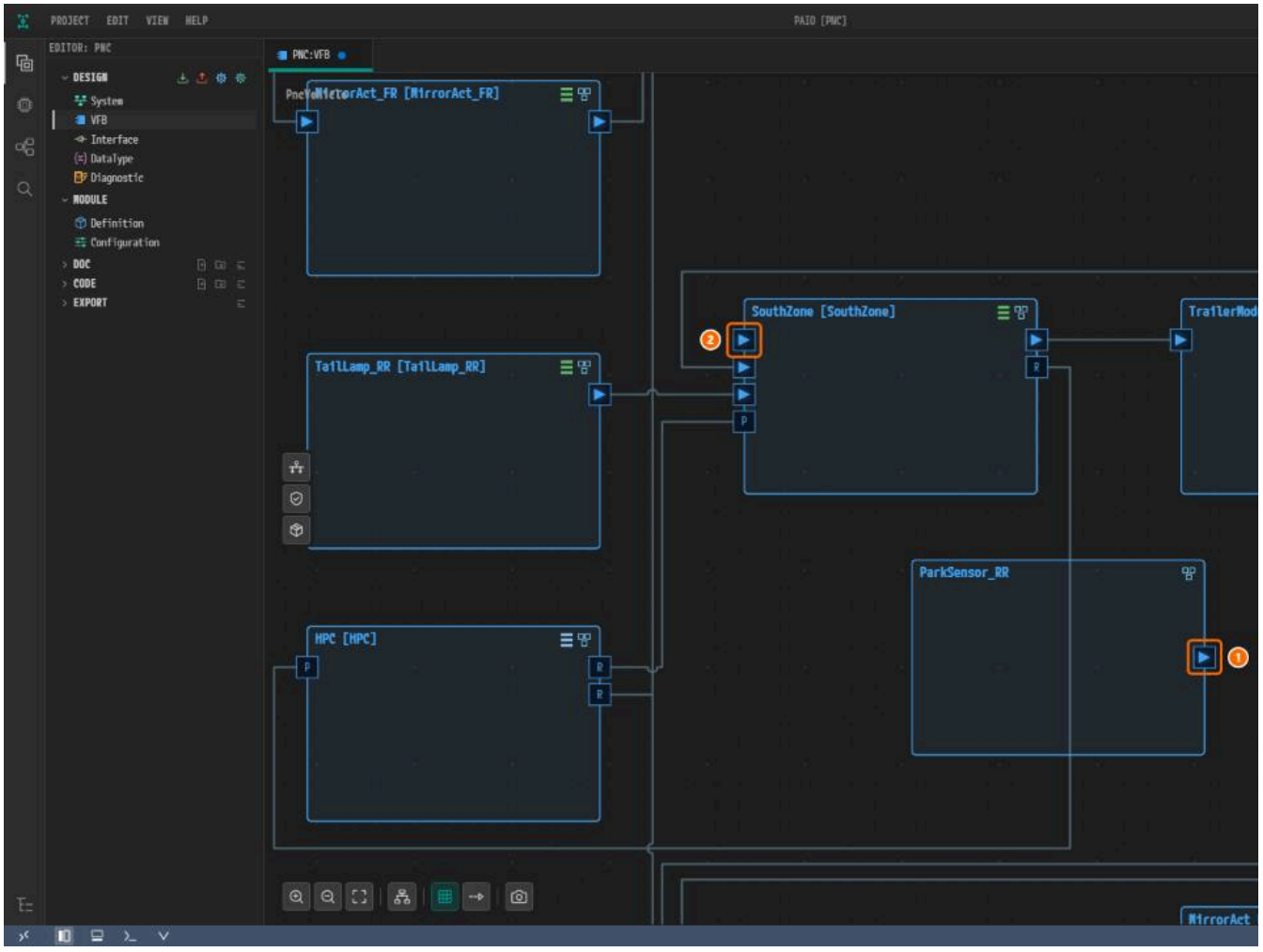


*그림 6-13. ① `ParkSensor_RR`의 `ParkDistance`, ② `SouthZone`의 `ParkDistance`. 확실하지 않으면 포트를 눌러 **NAME**을 확인한다.*

ECU에 매핑하고 저장하기

1. `ParkSensor_RR`을 우클릭한다. **ECU/HPC Mapping**에 마우스를 올리고 `ParkSensor_RR [ECU]`를 누른다.

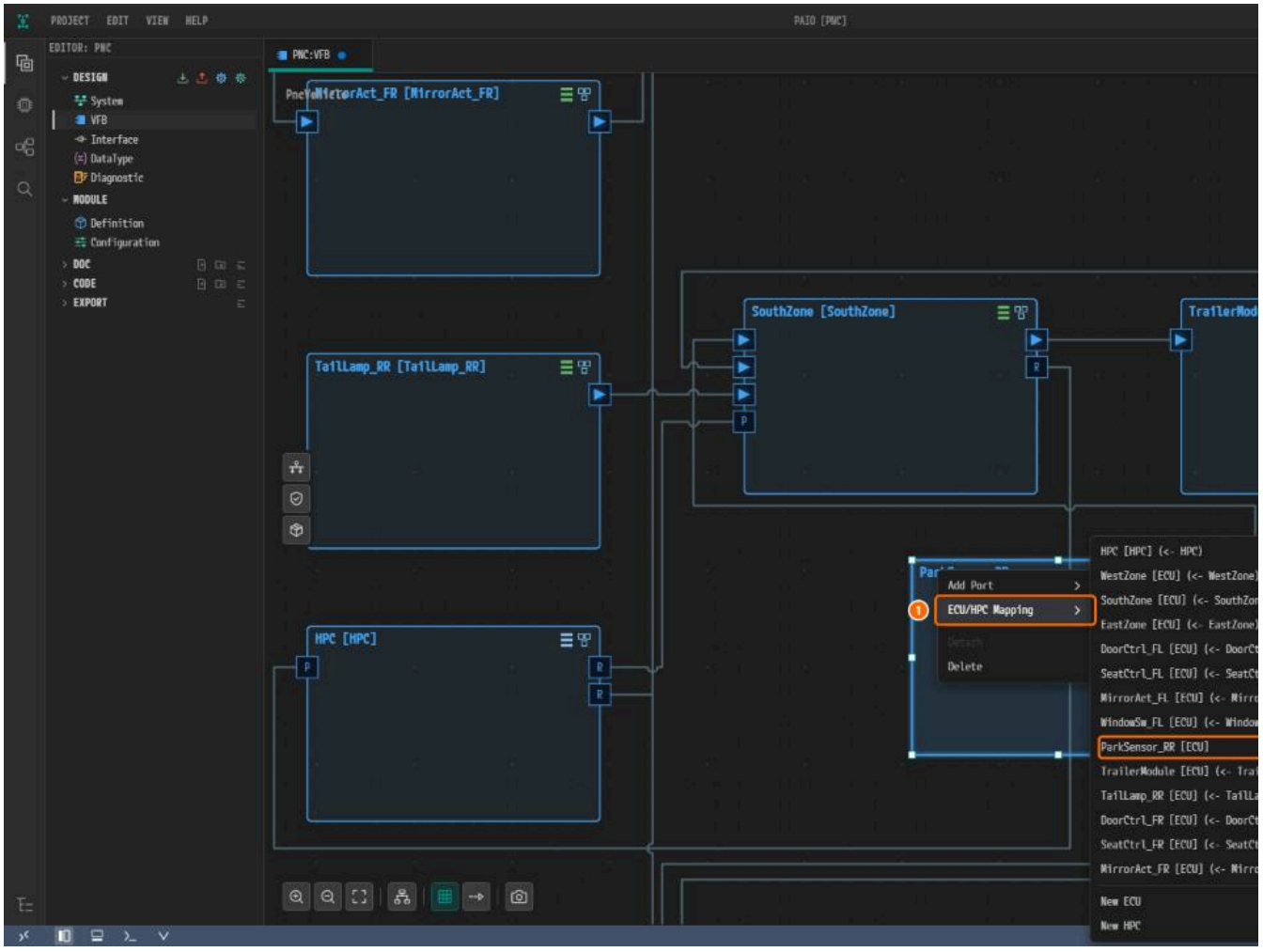


*그림 6-14. ① **ECU/HPC Mapping**, ② `ParkSensor_RR [ECU]`.*

> **참고.** 노드 머리글의 매핑 표시(ECU 아이콘과 `ParkSensor_RR [ParkSensor_RR]`)는 편집기를 다시 열어야 보인다.

2. **Ctrl+S**로 저장한다.

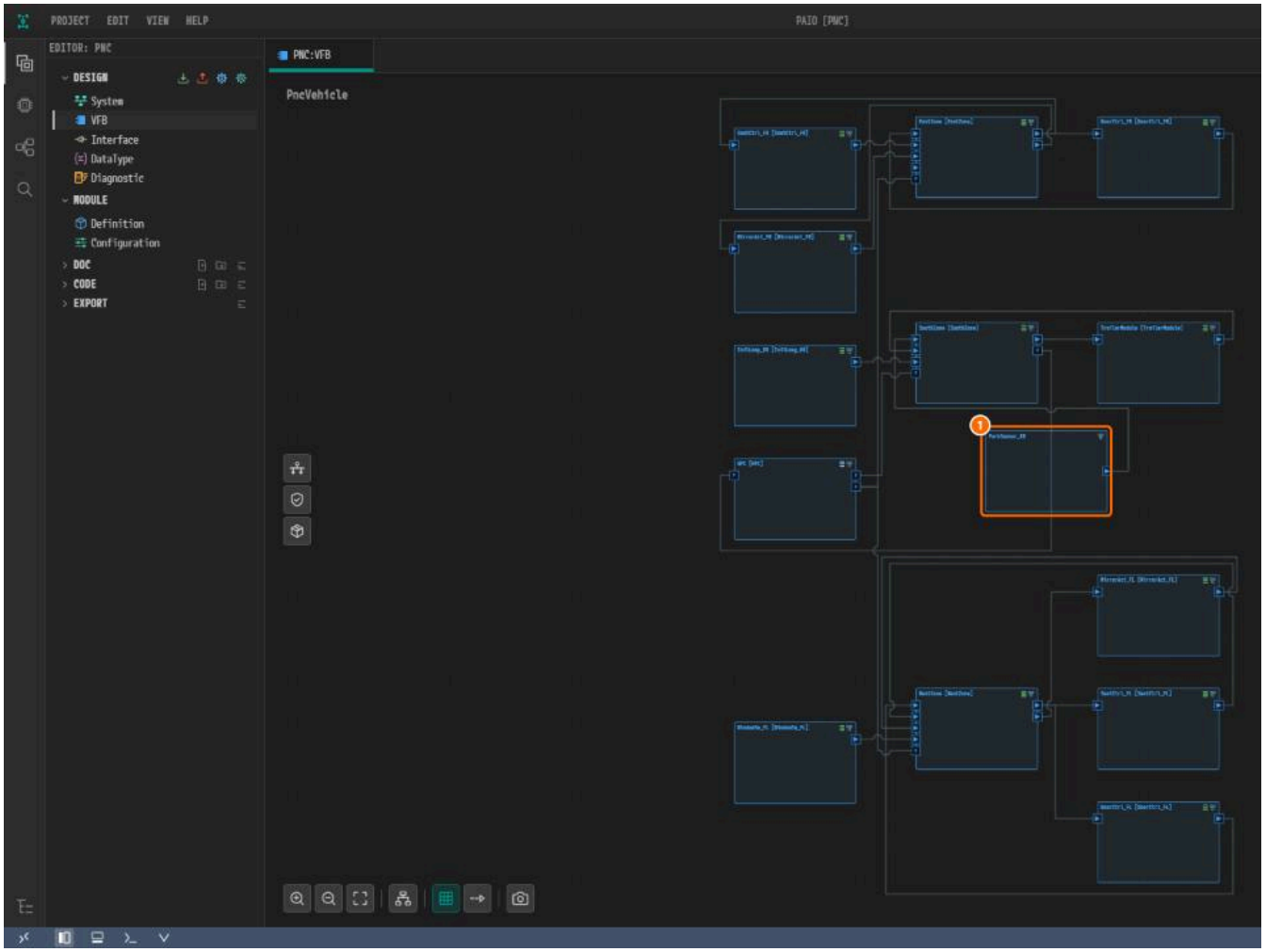


그림 6-15. ① `SouthZone`과 이어진 `ParkSensor_RR`. 루트의 컴포지션은 14개가 된다.

나머지 컴포지션 13개

나머지도 표를 보고 같은 방법으로 만든다. 컴포지션 이름은 노드 이름과 같게 짓고, 경계 포트는 같은 이름의 내부 SWC 포트와 1:1로 위임하며, 그 노드에 매핑한다. **내부 SWC**는 `프로`는 PortInterface이다.

컴포지션(= 매핑 노드)	노드 종류	내부 SWC	컴포지션 경계 포트
`HPC`	HPC	`ParkAssist` = `ParkAssistApp`, `VehicleState` = `VehicleStateApp`, `Comfort` = `ComfortApp`	`ParkDistanceSvc`(R) `SVC_ParkDistance`, `VehicleStateSvc`(P) `SVC_Vehicle`
`WestZone`	ECU	`Gateway` = `ZoneGateway`	`ComfortCmd`(P) `SRI_ComfortCmd`, `MirrorCmd`(P) `SRI_MirrorCmd`, `DoorStatus` `SRI_SeatStatus`, `MirrorStatus`(R) `SRI_MirrorStatus`, `WindowSwitch`(R) `S` `SVC_ComfortControl`
`EastZone`	ECU	`Gateway` = `ZoneGateway`	`WestZone`과 같다
`SouthZone`	ECU	`Gateway` = `RearGateway`	`ParkDistance`(R) `SRI_ParkDistance`, `TrailerStatus`(R) `SRI_TrailerStatu` `SRI_TailLamp`, `ParkDistanceSvc`(P) `SVC_ParkDistance`, `VehicleStateSvc`(
`DoorCtrl_FL`, `DoorCtrl_FR`	ECU	`DoorControl` = `DoorControl`	`DoorStatus`(P) `SRI_DoorStatus`, `ComfortCmd`(R) `SRI_ComfortCmd`
`SeatCtrl_FL`, `SeatCtrl_FR`	ECU	`SeatControl` = `SeatControl`	`SeatStatus`(P) `SRI_SeatStatus`, `ComfortCmd`(R) `SRI_ComfortCmd`
`MirrorAct_FL`, `MirrorAct_FR`	ECU	`MirrorControl` = `MirrorControl`	`MirrorStatus`(P) `SRI_MirrorStatus`, `MirrorCmd`(R) `SRI_MirrorCmd`
`WindowSw_FL`	ECU	`WindowSwitch` = `WindowSwitch`	`WindowSwitch`(P) `SRI_WindowSwitch`
`ParkSensor_RR`	ECU	`ParkDistanceSensing` = `ParkDistanceSensing`	`ParkDistance`(P) `SRI_ParkDistance`(이 장에서 만든 것)
`TrailerModule`	ECU	`TrailerControl` = `TrailerControl`	`TrailerStatus`(P) `SRI_TrailerStatus`, `TrailerCmd`(R) `SRI_TrailerCmd`
`TailLamp_RR`	ECU	`TailLampControl` = `TailLampControl`	`TailLamp`(P) `SRI_TailLamp`

- `HPC`의 SWC 세 개는 **Add SWC > Adaptive Application**으로 만든다. `HPC`의 `PARA` 프로토타입은 뒤의 PARA 플랫폼 임포트와 PNC 요청 포트에서 추가된다.
- 루트의 어셈블리 커넥터는 21개이다. 말단 ECU의 상태 포트(P)를 Zone(R)으로, Zone의 명령 포트(P)를 말단 ECU(R)로, Zone의 서비스 포트를 HPC로 잇는다. `HPC` 안에는 `V`를 잇는 커넥터가 하나 더 있다.
- internal behavior(runnable, RTE 이벤트)는 다루지 않는다. VFB 에디터를 참고한다.

VFB 루트 구성 확인하기

1. 컴포지션 14개가 모두 있으면 **Auto Layout**, **Fit to View**를 차례로 누른다.

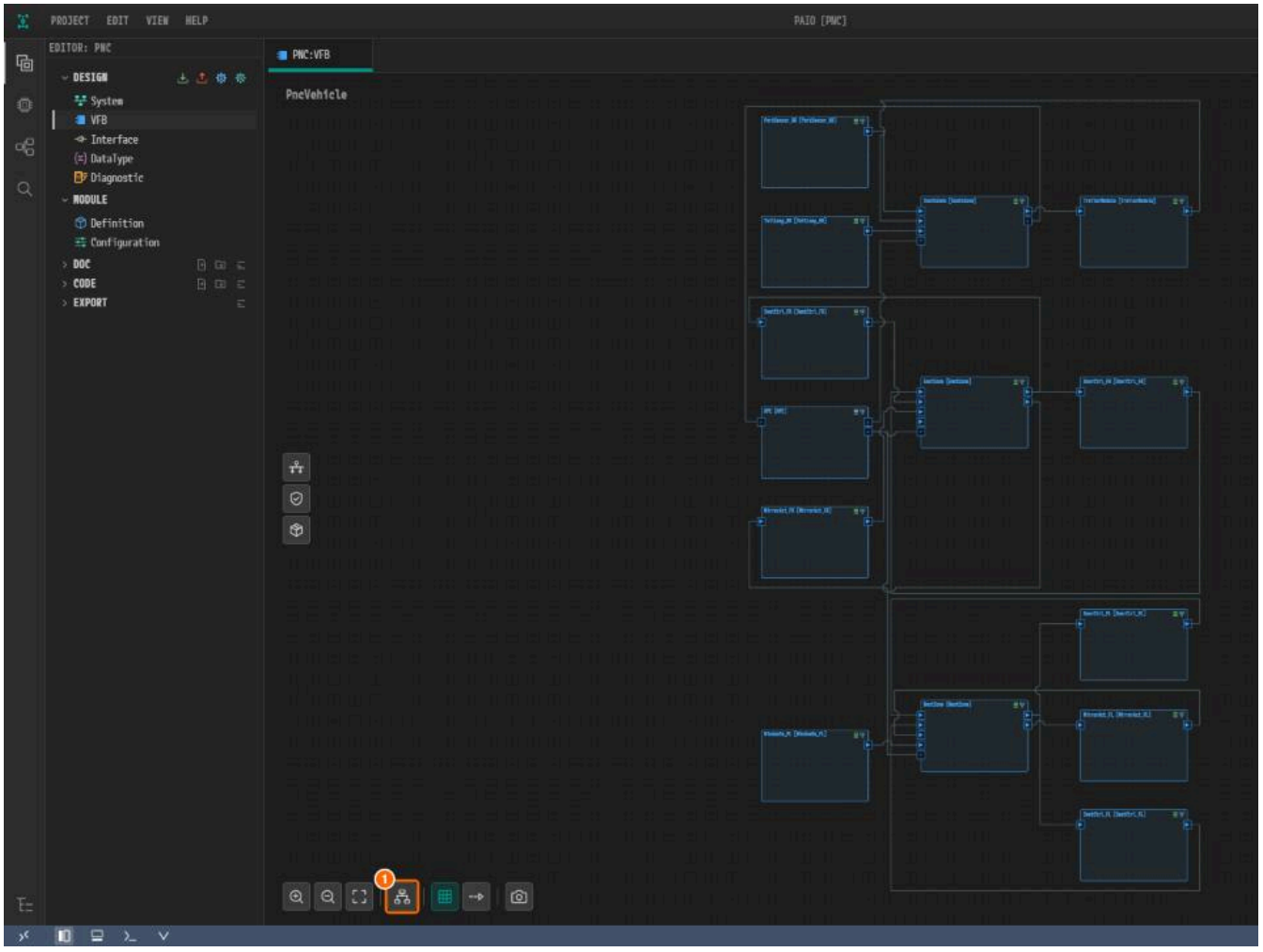


*그림 6-16. ① **Auto Layout.** `ParkSensor_RR`이 왼쪽 맨 위로 온다.*

컴포지션(= 매핑 노드)	역할
`HPC`	HPC(AP). 응용 AA를 담는다
`WestZone`, `EastZone`, `SouthZone`	Zone ECU. 게이트웨이 SWC를 담는다
`DoorCtrl_FL`, `SeatCtrl_FL`, `MirrorAct_FL`, `WindowSw_FL`, `DoorCtrl_FR`, `SeatCtrl_FR`, `MirrorAct_FR`	Comfort 계열 말단 ECU
`ParkSensor_RR`	주차 거리 센서 말단 ECU
`TrailerModule`, `TailLamp_RR`	Trailer 계열 말단 ECU

VFC와 PortGroup

- 기능 하나의 포트를 PortGroup 하나로 묶고 `VFC_<기능>`으로 이름 짓는다: `VFC_Comfort`, `VFC_ParkAssist`, `VFC_Trailer`.
- PortGroup의 CATEGORY는 모두 `PARTIAL_NETWORKING`으로 둔다.

VFC 패밀리 규칙

VFC는 모든 계층에서 **같은 PortGroup 이름**을 쓴다. `VFC_ParkAssist`의 예는 다음과 같다.

계층	예	PortGroup 이름	내용
원자 SWC	`ParkDistanceSensing`	`VFC_ParkAssist`	OUTER PORTS = `ParkDistance`
ECU 컴포지션	`ParkSensor_RR`	`VFC_ParkAssist`	OUTER PORTS = 위임 포트 `ParkDistance`, INNER GROUPS = `ParkDistanceSensing` / `VFC_ParkAssist`
루트 컴포지션	`PncVehicle`	`VFC_ParkAssist`	INNER GROUPS = 참여하는 직계 프로토타입 `HPC`, `SouthZone`, `ParkSensor_RR`의 `VFC_ParkAssist`

- **OUTER PORTS**: 그 SWC나 컴포지션의 포트 중 이 VFC에 참여하는 포트이다.
- **INNER GROUPS**: 컴포지션 안 프로토타입의 같은 이름 PortGroup이다.

> **주의.** 이름이 다른 PortGroup(예: `PG_ParkAssist`)은 VFC 멤버가 아니다. PNC 다이얼로그는 VFC와 같은 이름의 PortGroup을 가진 루트 프로토타입만 센다.

PortGroup 만들기

1. 루트에서 `ParkSensor_RR`을 더블클릭한다.

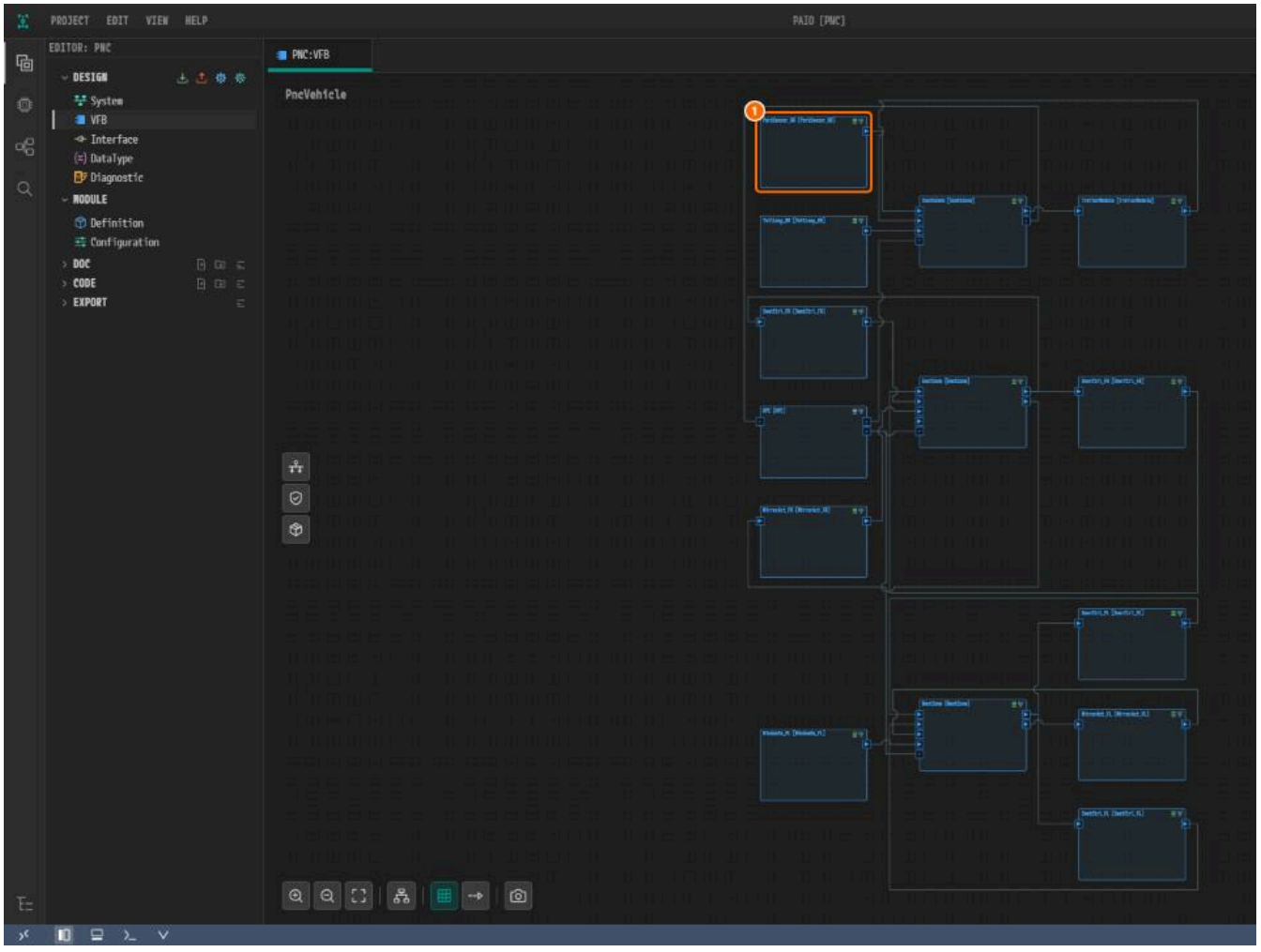


그림 6-17. ① `ParkSensor_RR`.

2. **Fit to View**를 누른다.

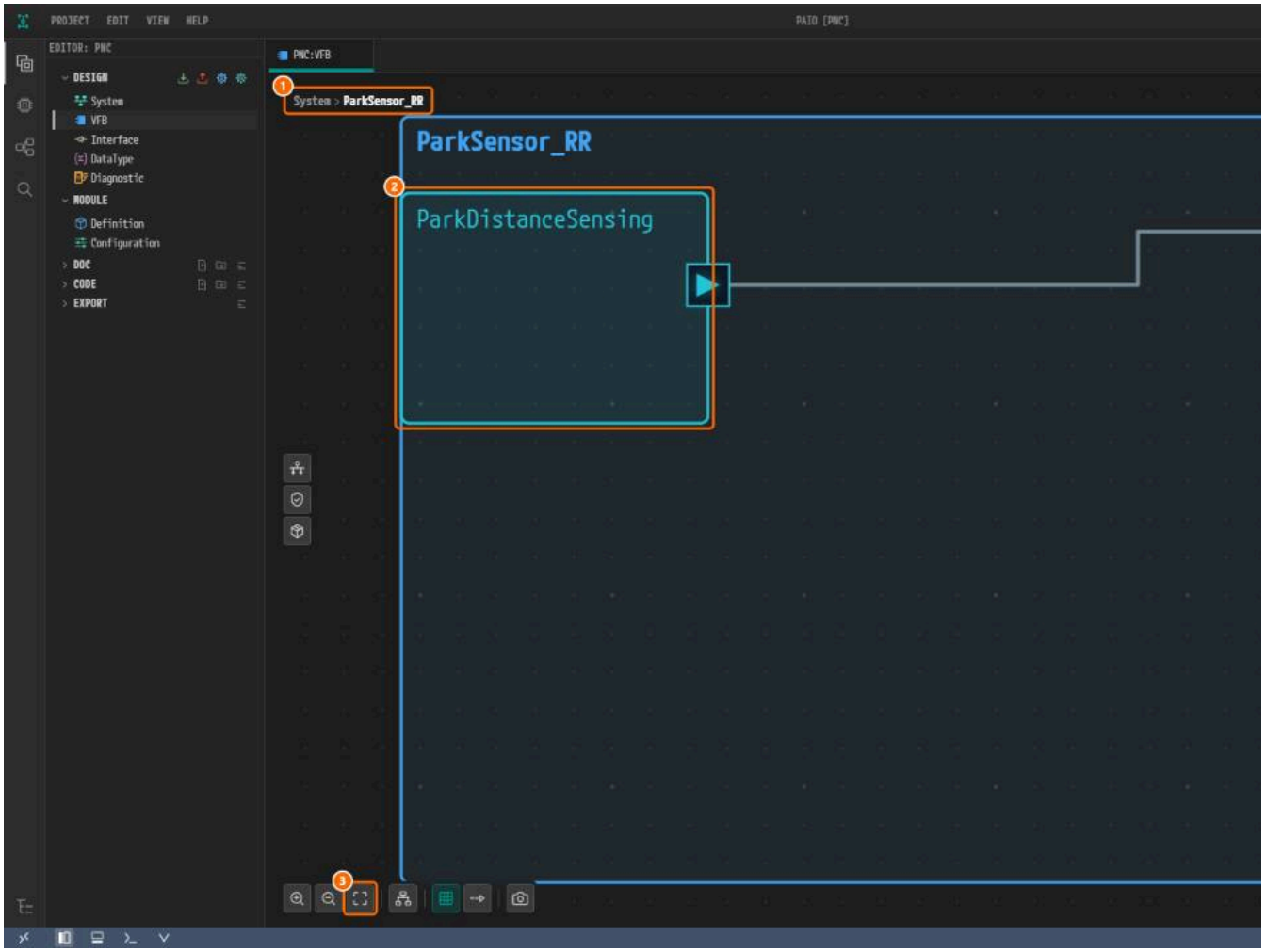


*그림 6-18. ① 경로, ② `ParkDistanceSensing`, ③ **Fit to View**.*

3. `ParkDistanceSensing`을 더블클릭한다. **SWC Internal Behavior [ParkDistanceSensing]** 다이얼로그가 열린다.

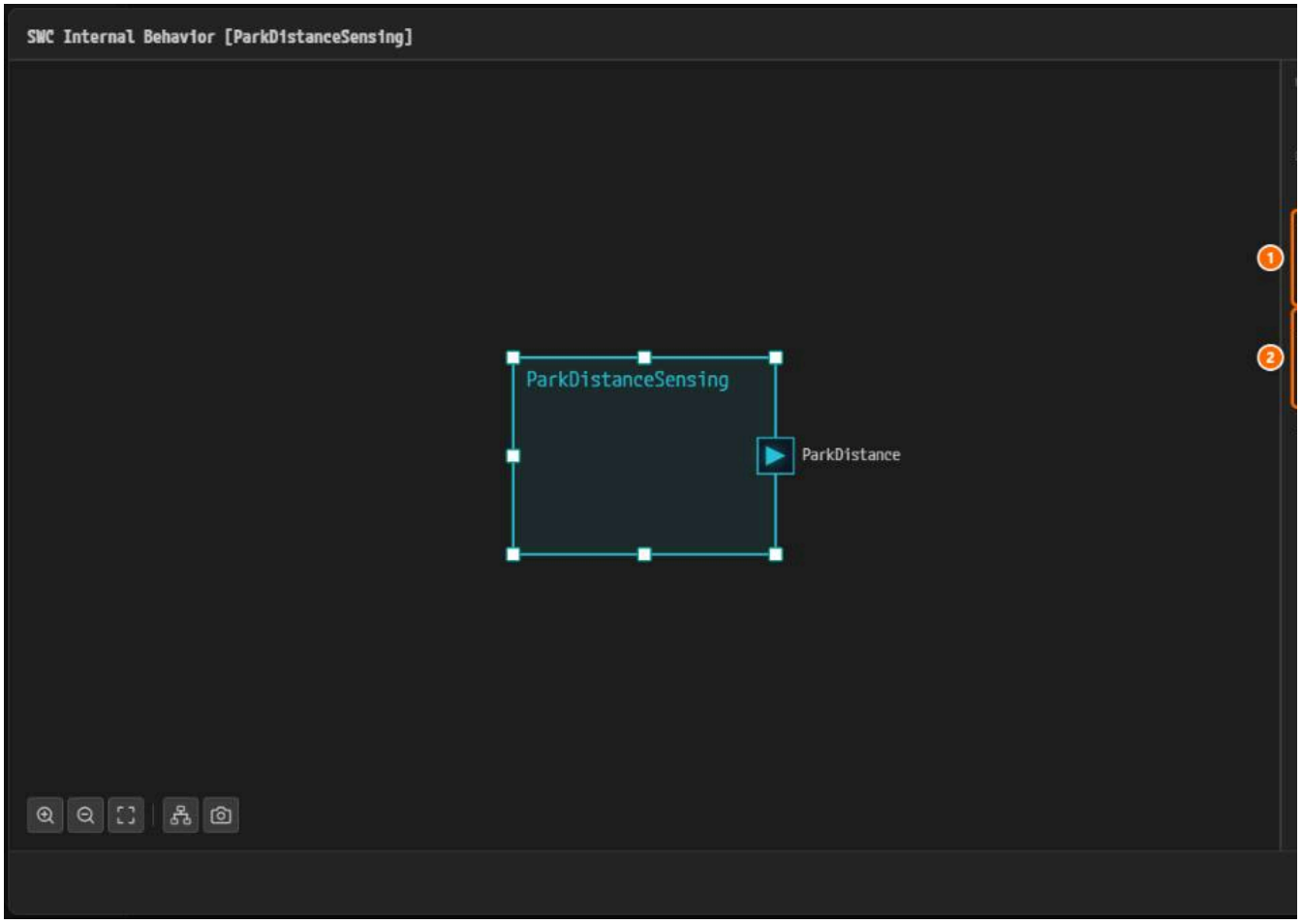


*그림 6-19. ① `ParkDistance`가 있는 **PORTS**, ② 비어 있는 **PORT GROUPS**.*

4. **PORT GROUPS** 오른쪽의 **+**를 누른다. 블록 `PortGroup_0`이 생긴다.

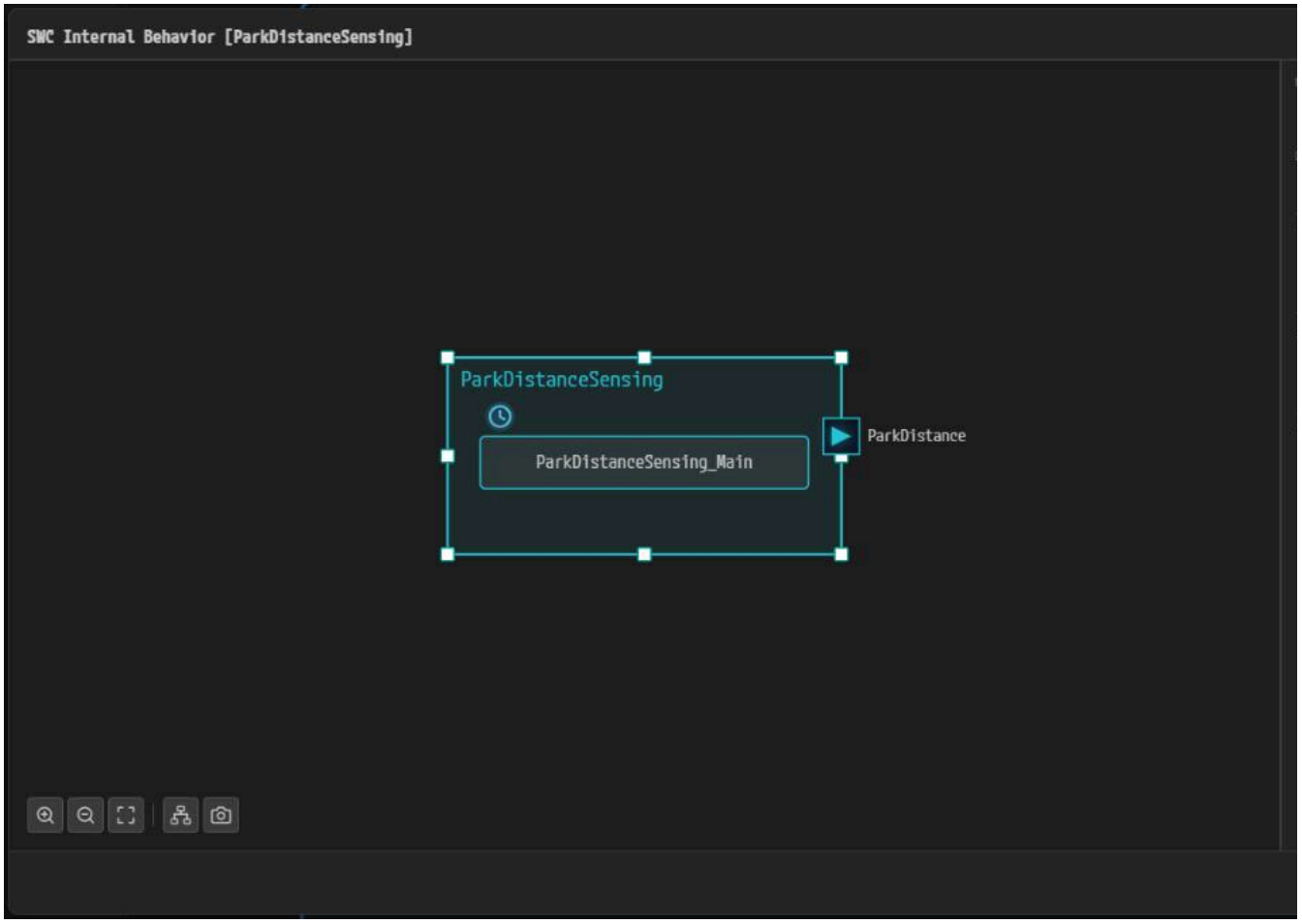


*그림 6-20. ① **PORT GROUPS**의 **+**. 그림 6-20~6-22는 완성 예제의 화면이라 캔버스에 runnable도 보인다.*

5. **NAME**을 `VFC_ParkAssist`로 바꾸고 **Tab**을 누른다.

6. **CATEGORY**에서 **PARTIAL_NETWORKING**을 누른다.

7. **OUTER PORTS** 오른쪽의 **+**를 누른다. 행 `1`이 생긴다.

8. 행의 **▾**를 누르고 `ParkDistance`를 고른다.

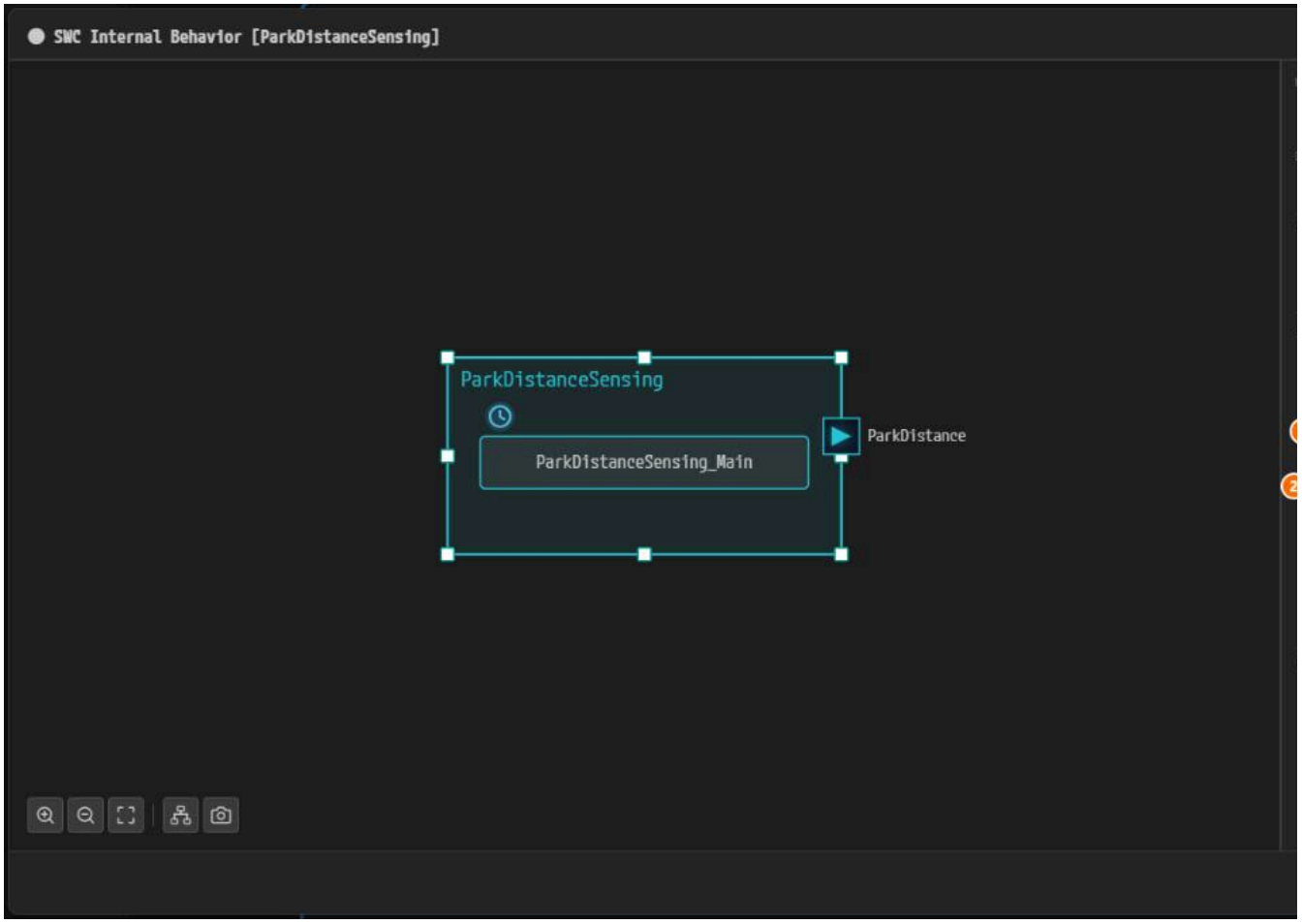


*그림 6-21. ① **NAME**, ② **PARTIAL_NETWORKING**, ③ **OUTER PORTS**의 **+**, ④ `ParkDistance`.*

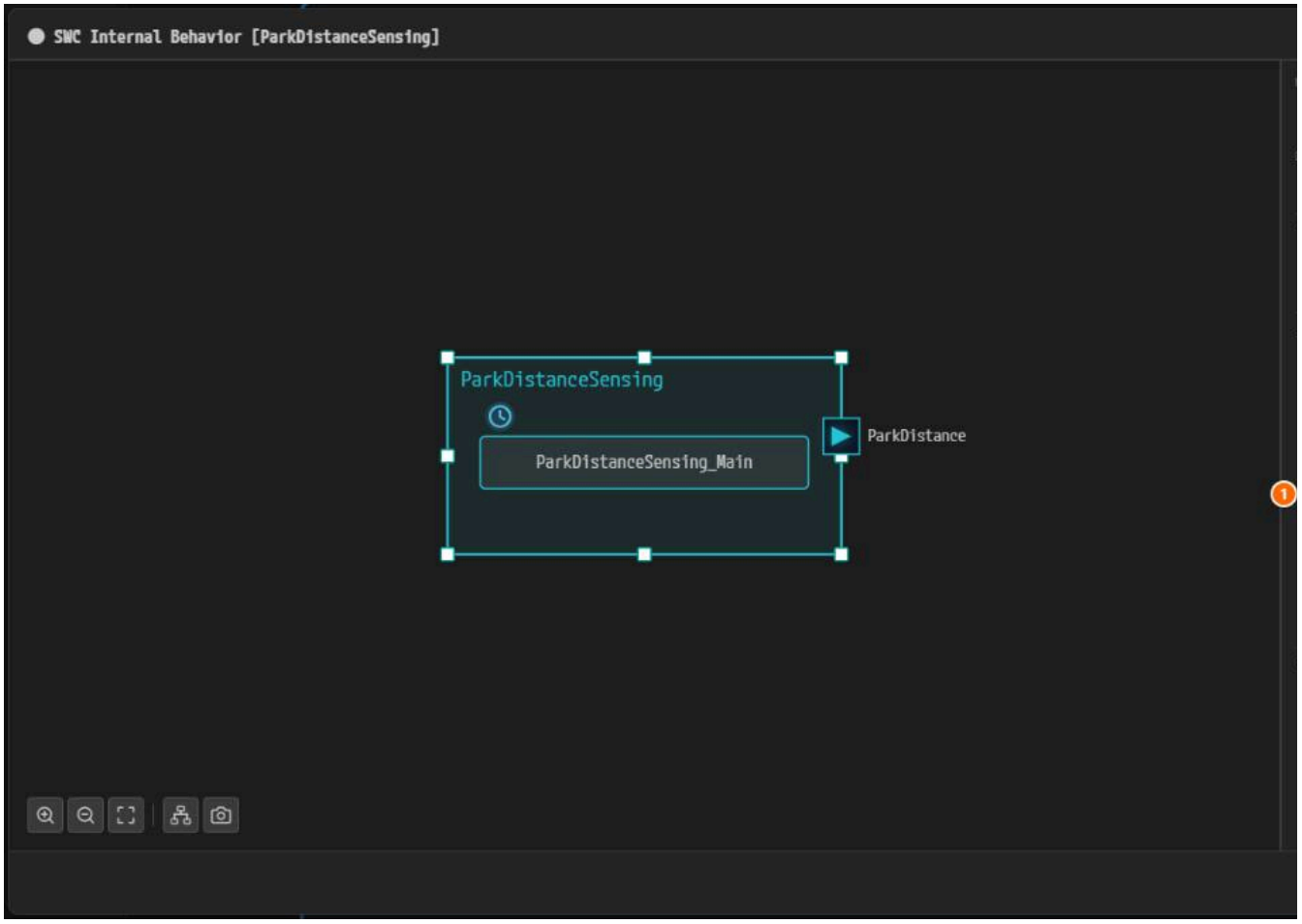


그림 6-22. ① 완성된 PortGroup.

9. **Ctrl+S**로 다이얼로그를 저장한다.
10. **Close**를 누른다.
11. **Ctrl+S**로 VFB 편집기를 저장한다.

저장하면 PAIO가 상위에 같은 이름의 PortGroup을 만든다.

컴포지션	자동으로 생기는 PortGroup
`ParkSensor_RR`	`VFC_ParkAssist`, INNER GROUPS = `ParkDistanceSensing` / `VFC_ParkAssist`
`PncVehicle`(루트)	`VFC_ParkAssist`, INNER GROUPS = `ParkSensor_RR` / `VFC_ParkAssist`

자동으로 생긴 PortGroup 확인하기

1. `ParkSensor_RR` 프레임의 제목 줄을 누른다.
2. **COMPOSITION** 패널의 **PORT GROUPS**를 확인한다.

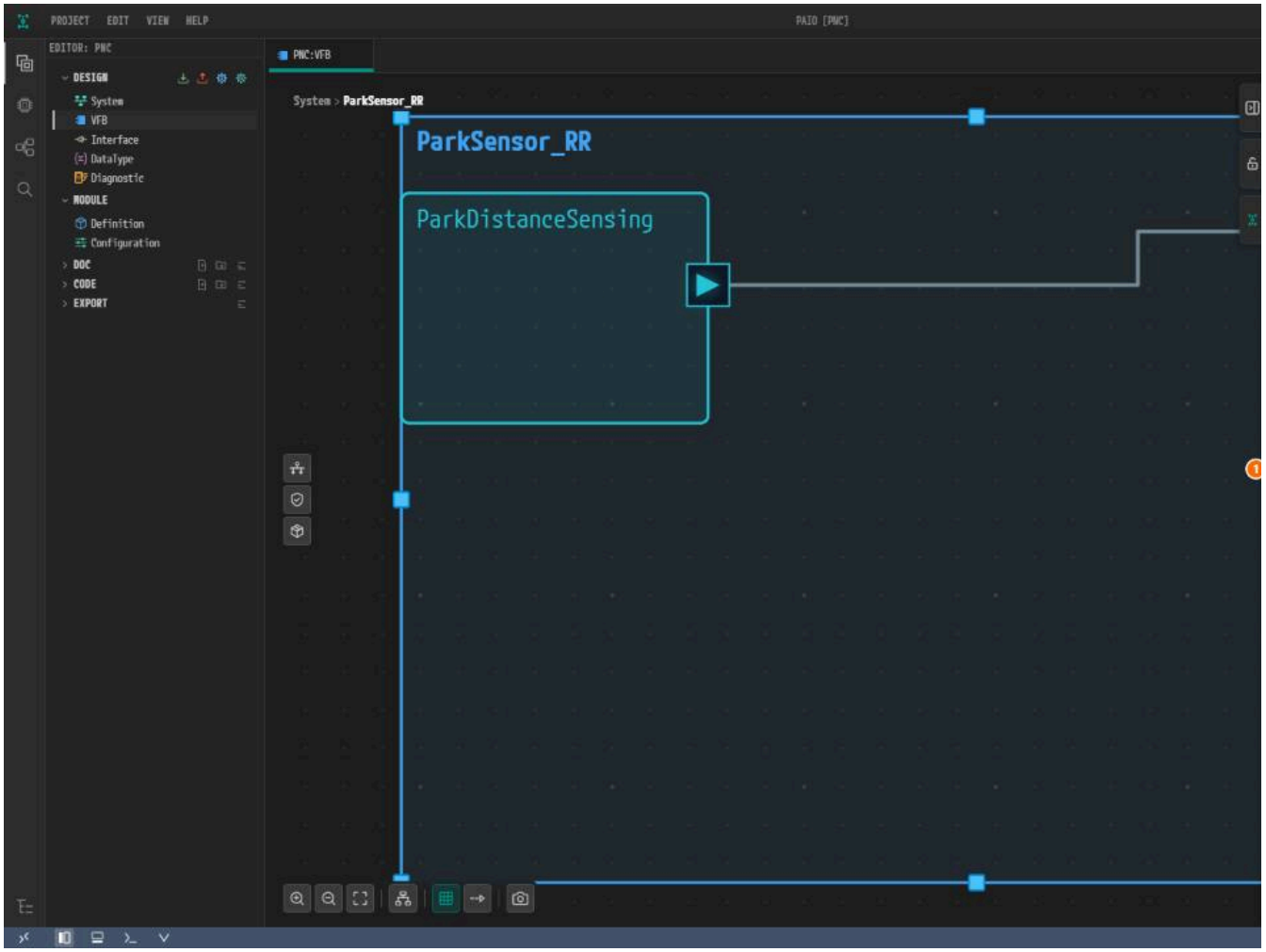


그림 6-23. ① INNER GROUPS `ParkDistanceSensing` / `VFC_ParkAssist`가 있는 `VFC_ParkAssist`.

> **참고.** 자동으로 생긴 컴포지션 PortGroup에는 OUTER PORTS가 없다. PNC 다이얼로그 — PNC별 바인딩에서 채운다.

나머지 PortGroup

PortGroup은 모두 33개이고 모두 `PARTIAL_NETWORKING`이다.

- **원자 SWC:** 위와 같은 방법으로 만든다.
- **ECU 컴포지션과 루트:** 저장할 때 **INNER GROUPS**와 함께 자동으로 생긴다.
- **ECU 컴포지션의 OUTER PORTS와 응용 AA(`ComfortApp`, `ParkAssistApp`, `VehicleStateApp`)의 PortGroup:** VFB 편집기에서 편집할 수 없다. PNC 다이얼로그 — P
래 **OUTER PORTS** 열이 그때 체크할 포트이다.

원자 SWC와 응용 AA (OUTER PORTS)

SWC	PortGroup	OUTER PORTS
`DoorControl`	`VFC_Comfort`	`DoorStatus`, `ComfortCmd`
`SeatControl`	`VFC_Comfort`	`SeatStatus`, `ComfortCmd`
`MirrorControl`	`VFC_Comfort`	`MirrorStatus`, `MirrorCmd`
`WindowSwitch`	`VFC_Comfort`	`WindowSwitch`
`ZoneGateway`	`VFC_Comfort`	`ComfortCmd`, `MirrorCmd`, `DoorStatus`, `SeatStatus`, `MirrorStatus`, `WindowSwitch`, `ComfortControlSvc`
`ParkDistanceSensing`	`VFC_ParkAssist`	`ParkDistance` (이 장에서 만든 것)
`RearGateway`	`VFC_ParkAssist`	`ParkDistance`, `ParkDistanceSvc`
`RearGateway`	`VFC_Trailer`	`TrailerStatus`, `TrailerCmd`, `TailLamp`, `VehicleStateSvc`
`TrailerControl`	`VFC_Trailer`	`TrailerStatus`, `TrailerCmd`
`TailLampControl`	`VFC_Trailer`	`TailLamp`
`ComfortApp` (AA)	`VFC_Comfort`	`ComfortControlSvc`
`ParkAssistApp` (AA)	`VFC_ParkAssist`	`ParkDistanceSvc`, `VehicleStateSvc`
`VehicleStateApp` (AA)	`VFC_Trailer`	`VehicleStateSvc`

ECU 컴포지션 (OUTER PORTS + INNER GROUPS)

컴포지션	PortGroup	OUTER PORTS	INNER G
`DoorCtrl_FL`, `DoorCtrl_FR`	`VFC_Comfort`	`DoorStatus`, `ComfortCmd`	`DoorCo`
`SeatCtrl_FL`, `SeatCtrl_FR`	`VFC_Comfort`	`SeatStatus`, `ComfortCmd`	`SeatCo`
`MirrorAct_FL`, `MirrorAct_FR`	`VFC_Comfort`	`MirrorStatus`, `MirrorCmd`	`Mirror`
`WindowSw_FL`	`VFC_Comfort`	`WindowSwitch`	`Window`
`WestZone`, `EastZone`	`VFC_Comfort`	`ComfortCmd`, `MirrorCmd`, `DoorStatus`, `SeatStatus`, `MirrorStatus`, `WindowSwitch`, `ComfortControlSvc`	`Gatewa`
`ParkSensor_RR`	`VFC_ParkAssist`	`ParkDistance`	`ParkDi`
`SouthZone`	`VFC_ParkAssist`	`ParkDistance`, `ParkDistanceSvc`	`Gatewa`
`SouthZone`	`VFC_Trailer`	`TrailerStatus`, `TrailerCmd`, `TailLamp`, `VehicleStateSvc`	`Gatewa`
`TrailerModule`	`VFC_Trailer`	`TrailerStatus`, `TrailerCmd`	`Traile`
`TailLamp_RR`	`VFC_Trailer`	`TailLamp`	`TailLa`
`HPC`	`VFC_ParkAssist`	`ParkDistanceSvc`	`ParkAs`
`HPC`	`VFC_Comfort`	`ComfortControlSvc`	`Comfor`
`HPC`	`VFC_Trailer`	`VehicleStateSvc`	`Vehicl`

루트 `PncVehicle` (INNER GROUPS만 있음, VFB 저장 시 자동 생성)

PortGroup	참여하는 직계 프로토타입
`VFC_Comfort`	`HPC`, `WestZone`, `EastZone`, `DoorCtrl_FL`, `SeatCtrl_FL`, `MirrorAct_FL`, `WindowSw_FL`, `DoorCtrl_FR`, `SeatCtrl_FR`, `MirrorAct_FR` (10개)
`VFC_ParkAssist`	`HPC`, `SouthZone`, `ParkSensor_RR` (3개)
`VFC_Trailer`	`HPC`, `SouthZone`, `TrailerModule`, `TailLamp_RR` (4개)

PNC 오버레이로 VFC 확인하기

PortGroup은 캔버스에 그려지지 않는다. VFC에 참여하는 컴포지션은 **VFC (PNC Overlay)** 아이콘으로 본다.

1. 루트로 돌아가 **Fit to View**를 누른다.
2. 캔버스 왼쪽 가장자리 아이콘 줄의 첫 번째 아이콘 **VFC (PNC Overlay)**를 누른다. **PNC OVERLAY** 패널이 열리고, 선택한 VFC 밖의 컴포지션은 흐리게 표시된다.

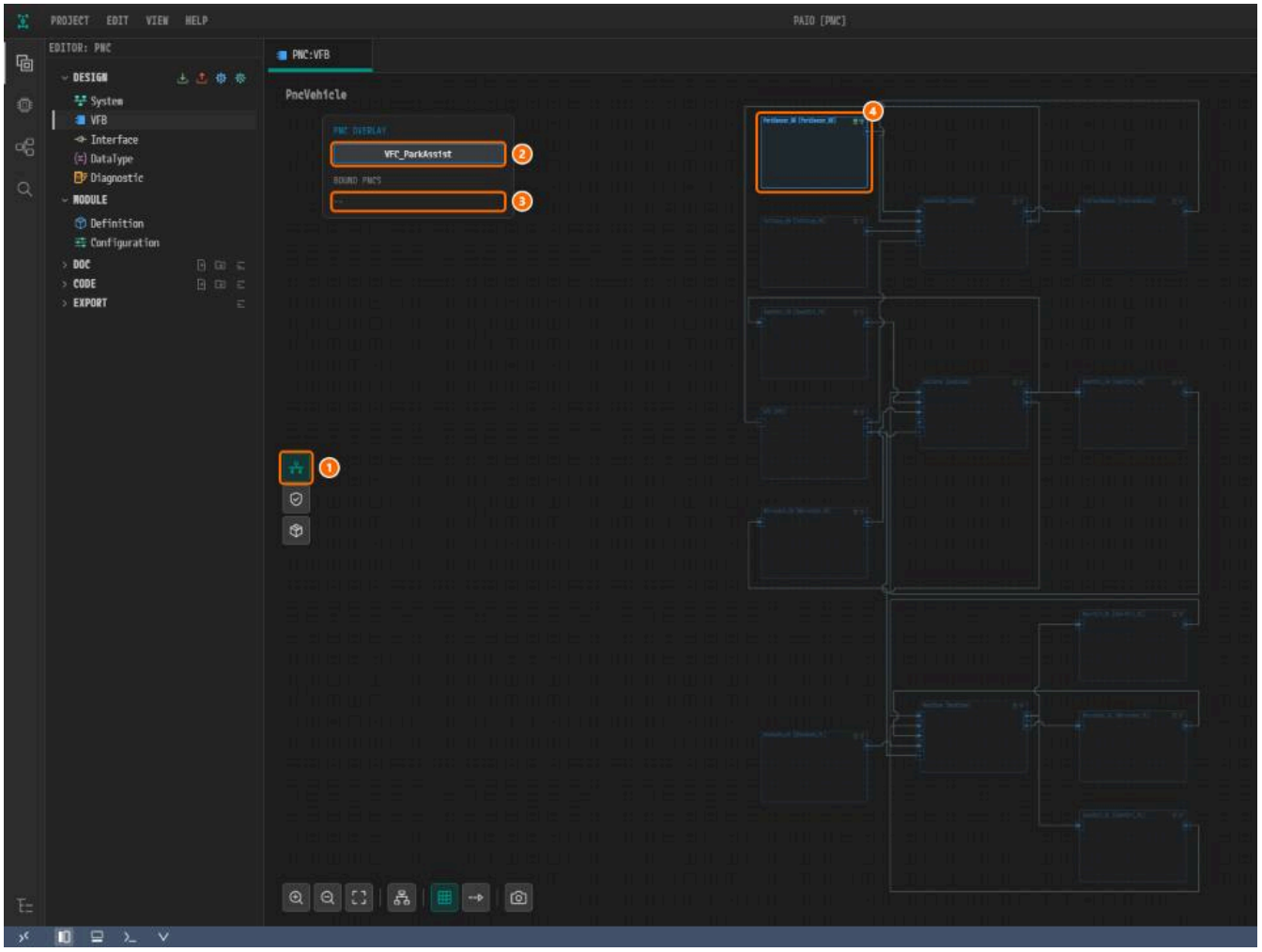


*그림 6-24. `ParkDistanceSensing`의 `VFC_ParkAssist`만 만든 상태. ① **VFC (PNC Overlay)**, ② `VFC_ParkAssist`, ③ **BOUND PNCS** = `--`, ④ `ParkSensor_RR`.*

3. 이름을 읽으려면 **Ctrl**+휠로 확대한다.

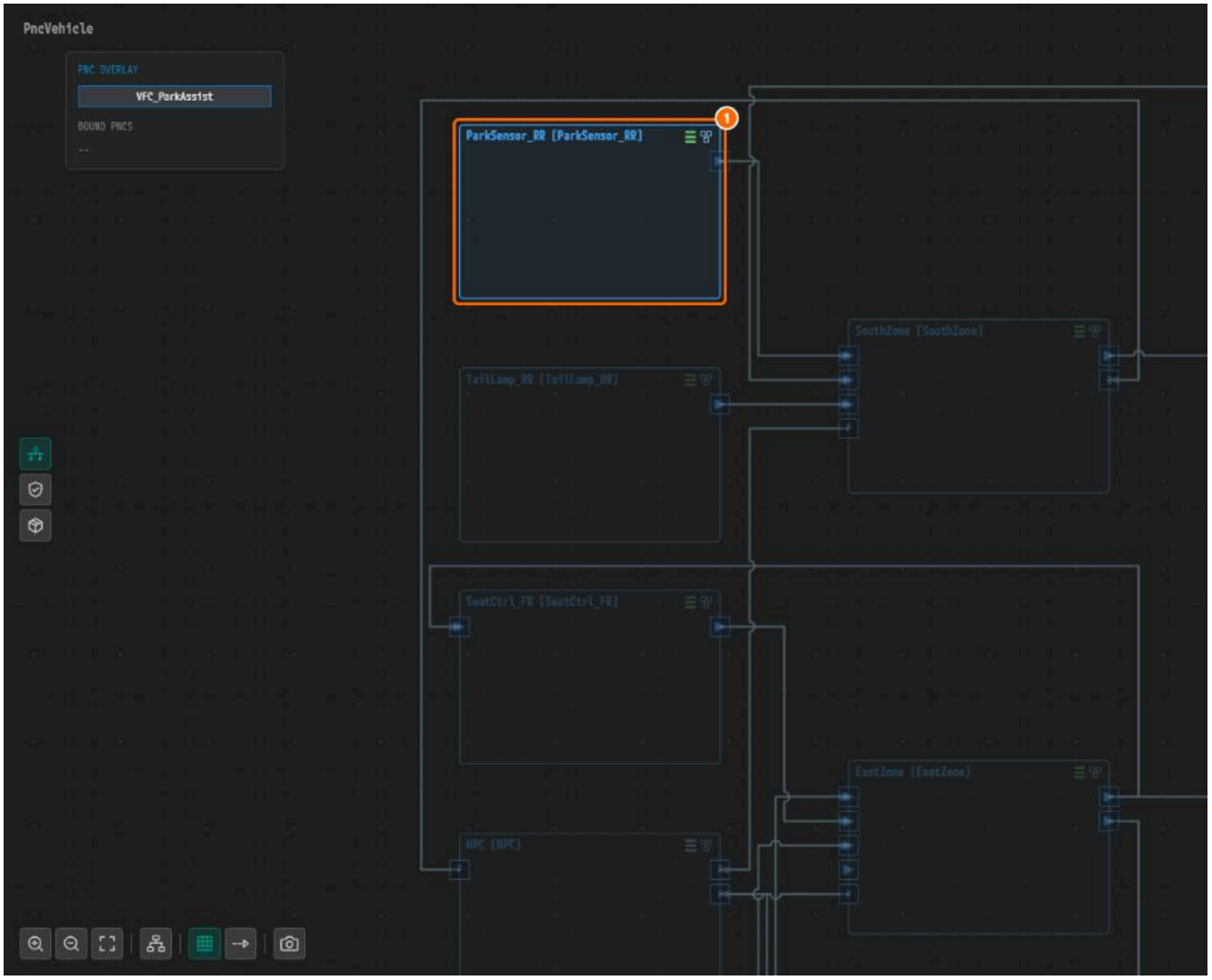


그림 6-25. ① 흐리지 않은 `ParkSensor_RR`.

4. 아이콘을 다시 눌러 오버레이를 끈다.

완성 예제의 오버레이

PortGroup 33개와 PNC 바인딩이 모두 있으면 목록에 `VFC_Comfort`, `VFC_ParkAssist`, `VFC_Trailer`가 나온다.

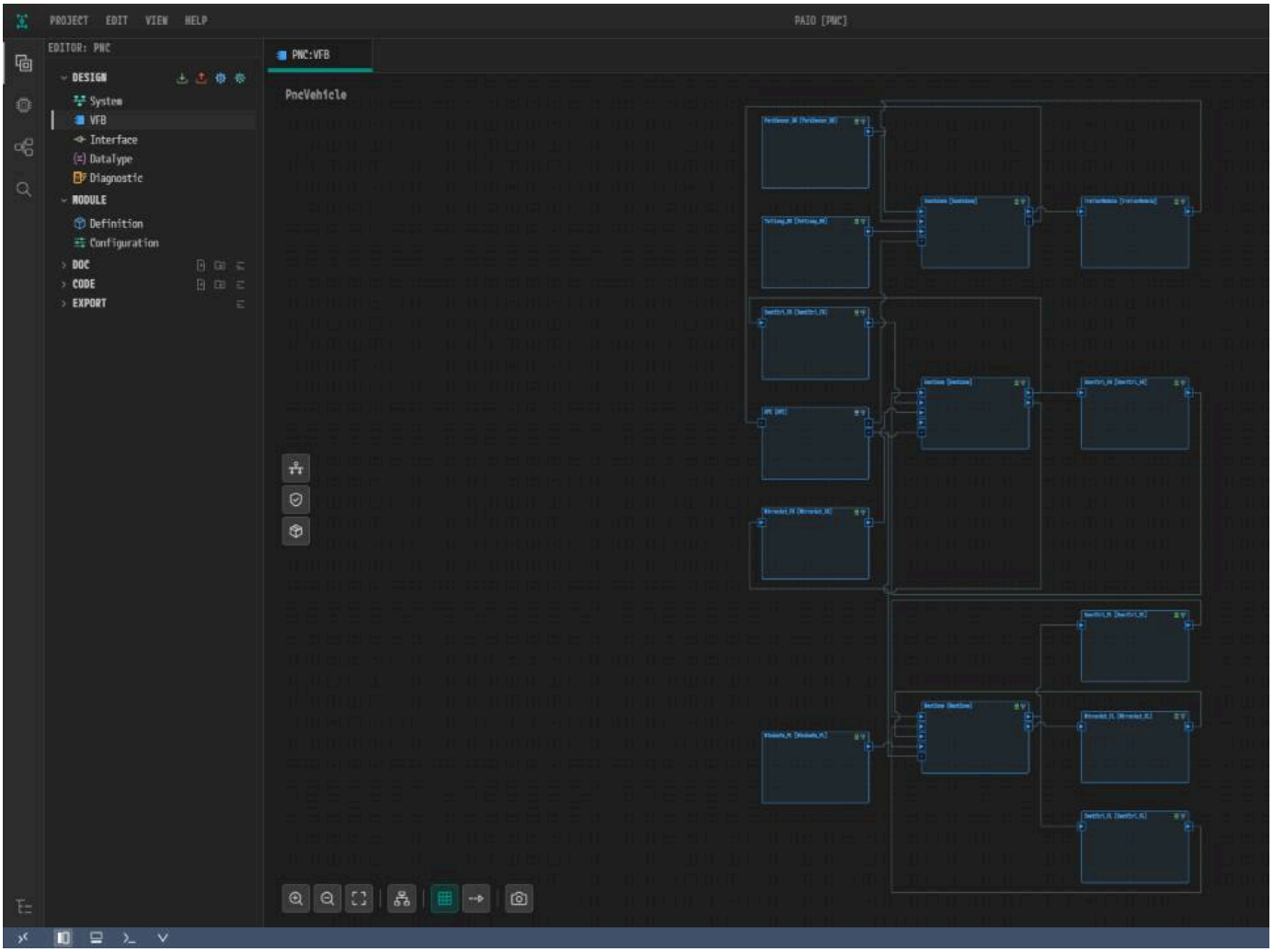


그림 6-26. 완성된 루트. 그림 6-16과 같은 모양이다.

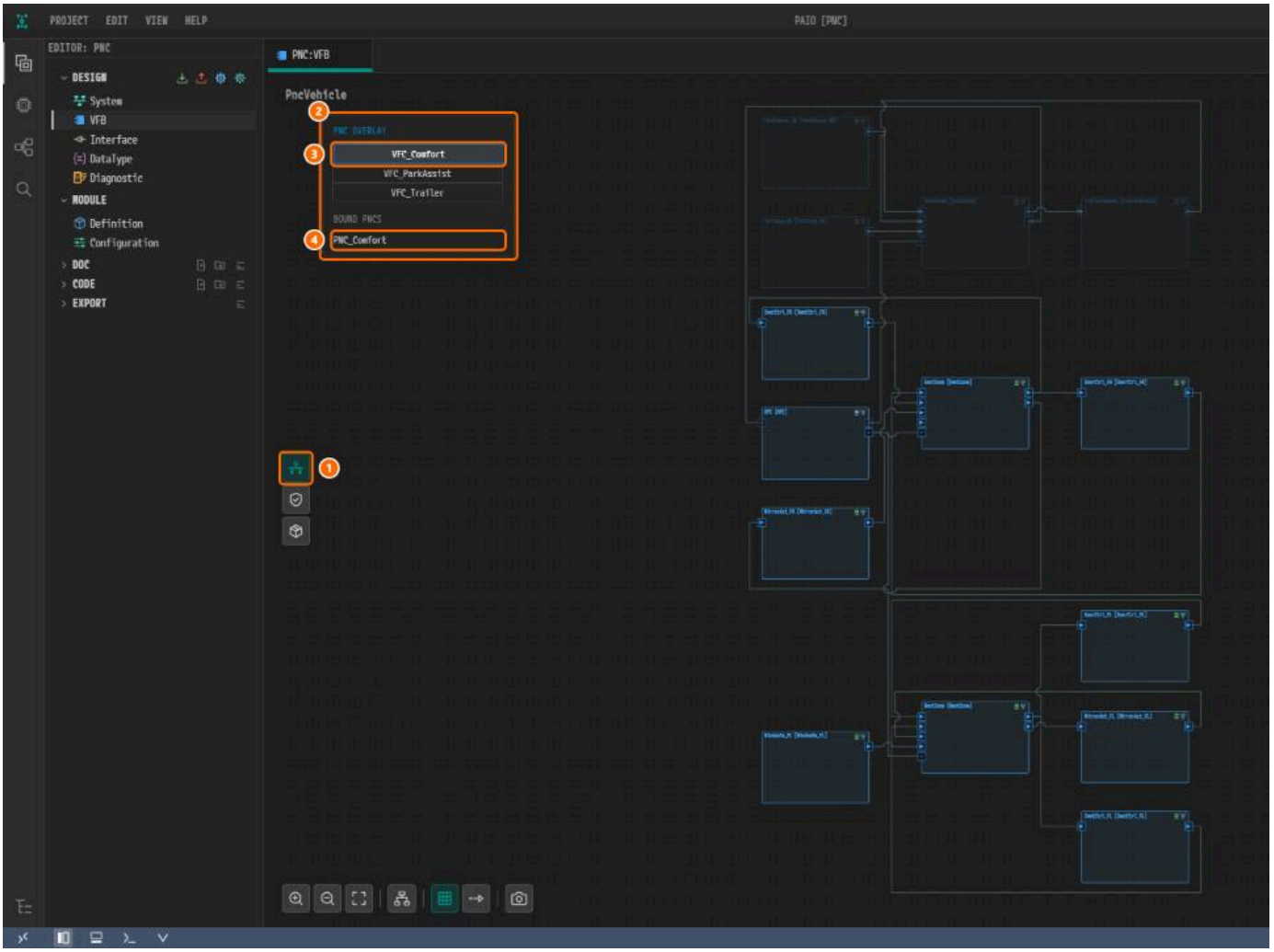


*그림 6-27. ① **VFC (PNC Overlay)**, ② **PNC OVERLAY**, ③ `VFC_Comfort`, ④ **BOUND PNCS** = `PNC_Comfort`.*



*그림 6-28. **PNC OVERLAY** 패널. **BOUND PNCS**는 그 VFC를 참조하는 PNC이다(읽기 전용).*

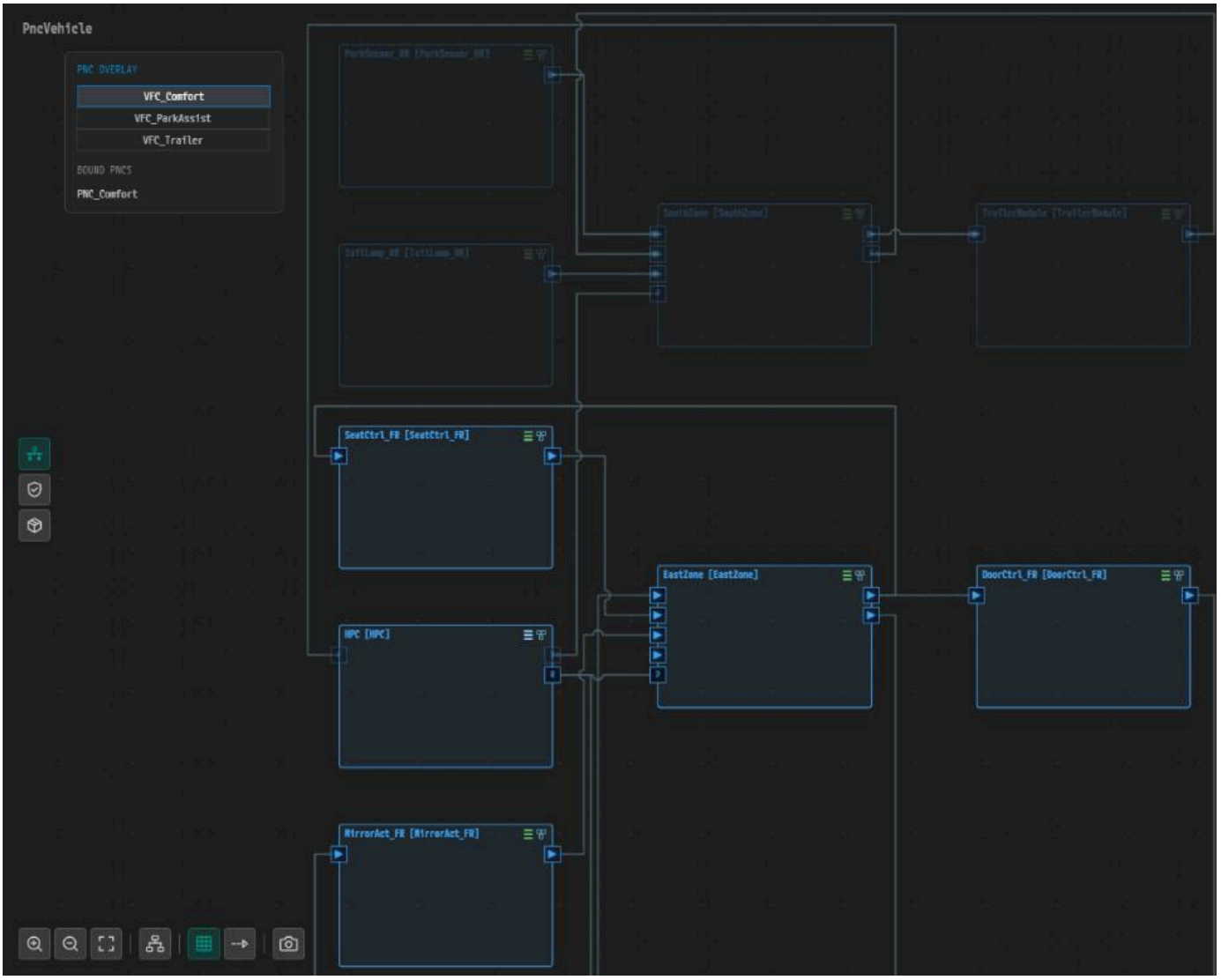


그림 6-29. `ParkSensor_RR`, `TailLamp_RR`, `SouthZone`, `TrailerModule`이 흐리게 표시된다.

왜 이렇게 설계하는가

- **PNC와 VFC.** PNC는 VFC를 참조해 그 PNC가 깨어 있어야 동작하는 기능을 기록한다.
- **모든 계층에서 같은 이름.** VFC는 여러 ECU와 HPC에 걸친다. 같은 이름의 PortGroup이어야 루트의 VFC 하나에서 모든 포트를 따라간다.
- **컴포지션 이름 = 노드 이름.** 컴포지션에서 노드를 바로 찾으므로 VFC가 걸치는 ECU와 채널을 안다.

확인하기

- [ ] 루트 `PncVehicle`에 System 편집기 노드 이름과 같은 컴포지션 14개가 있다.
- [ ] `ParkSensor_RR` 안의 `ParkDistanceSensing`이 경계 포트 `ParkDistance`로 위임되고, 그 포트가 `SouthZone`의 `ParkDistance`(R)와 이어져 있다.
- [ ] 편집기를 다시 열면 머리글이 ECU 아이콘과 함께 `ParkSensor_RR [ParkSensor_RR]`로 보인다.
- [ ] `ParkDistanceSensing`에 `VFC_ParkAssist`(**PARTIAL_NETWORKING**, OUTER PORTS `ParkDistance`)가, `ParkSensor_RR`에 INNER GROUPS `ParkDistanceSe`
다.
- [ ] 나머지 PortGroup의 원자 SWC PortGroup을 모두 만들었다.
- [ ] 오버레이 목록에 `VFC_Comfort`, `VFC_ParkAssist`, `VFC_Trailer`가 있고, `VFC_ParkAssist`에서는 `HPC`, `SouthZone`, `ParkSensor_RR`만 흐리지 않다. **BOUND PN**
- [ ] **PNC:VFB** 탭에 저장하지 않은 변경 표시가 없다.

다음 단계

PNC 다이얼로그 — PNC별 바인딩에서 PNC마다 VFC, I-PDU Group, 물리 채널, 서비스 인스턴스, 웨이크업 프레임을 바인딩한다.

PNC 다이얼로그 — PNC별 바인딩

이 장에서 할 일

이 장에서는 PNC 다이얼로그에서 PNC마다 VFC, I-PDU Group, 물리 채널, 서비스 인스턴스, 웨이크업 프레임을 바인딩한다. `PNC_ParkAssist`를 단계별로 따라 하고, 웨이크업 프
다.

시작하기 전에

- VFB와 VFC(PortGroup)를 마친 뒤 시작한다.
- PNC `PNC_Comfort` 17, `PNC_ParkAssist` 18, `PNC_Trailer` 19가 있고, I-PDU Group은 4장에서 각 PNC에 바인딩되었다. PortGroup `VFC_Comfort`, `VFC_ParkAss`
  않았다.
- 그림 7-2부터 7-9까지와 그림 7-14, 7-15는 아무것도 바인딩되지 않은 상태에서 찍었다. 독자의 화면에는 **PNC GROUPS**에 이미 두 행이 보인다.

섹션	`PNC_ParkAssist`의 값
VIRTUAL FUNCTION CLUSTERS (VFC)	`VFC_ParkAssist`
PNC GROUPS	`RearCAN_PNC_ParkAssist_Rx`(`ParkDistance_Pdu`), `RearCAN_PNC_ParkAssist_Tx`(PDU 없음)
PHYSICAL CHANNELS	CAN `RearCAN` / `RearCAN_Channel`, ETHERNET `Backbone` / `VLAN_Rear (VLAN 200)`
SERVICE INSTANCES	`ParkDistance_Provider`, `ParkDistance_Consumer`
WAKEUP FRAMES	없음

PNC 다이얼로그 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System**을 눌러 탭 `PNC:System`을 연다.

2. 캔버스 왼쪽 가장자리의 두 번째(아래쪽) 아이콘(툴팁 **Partial Network (PNC)**)을 누른다. **Partial Network (PNC) Configuration** 다이얼로그가 열린다.

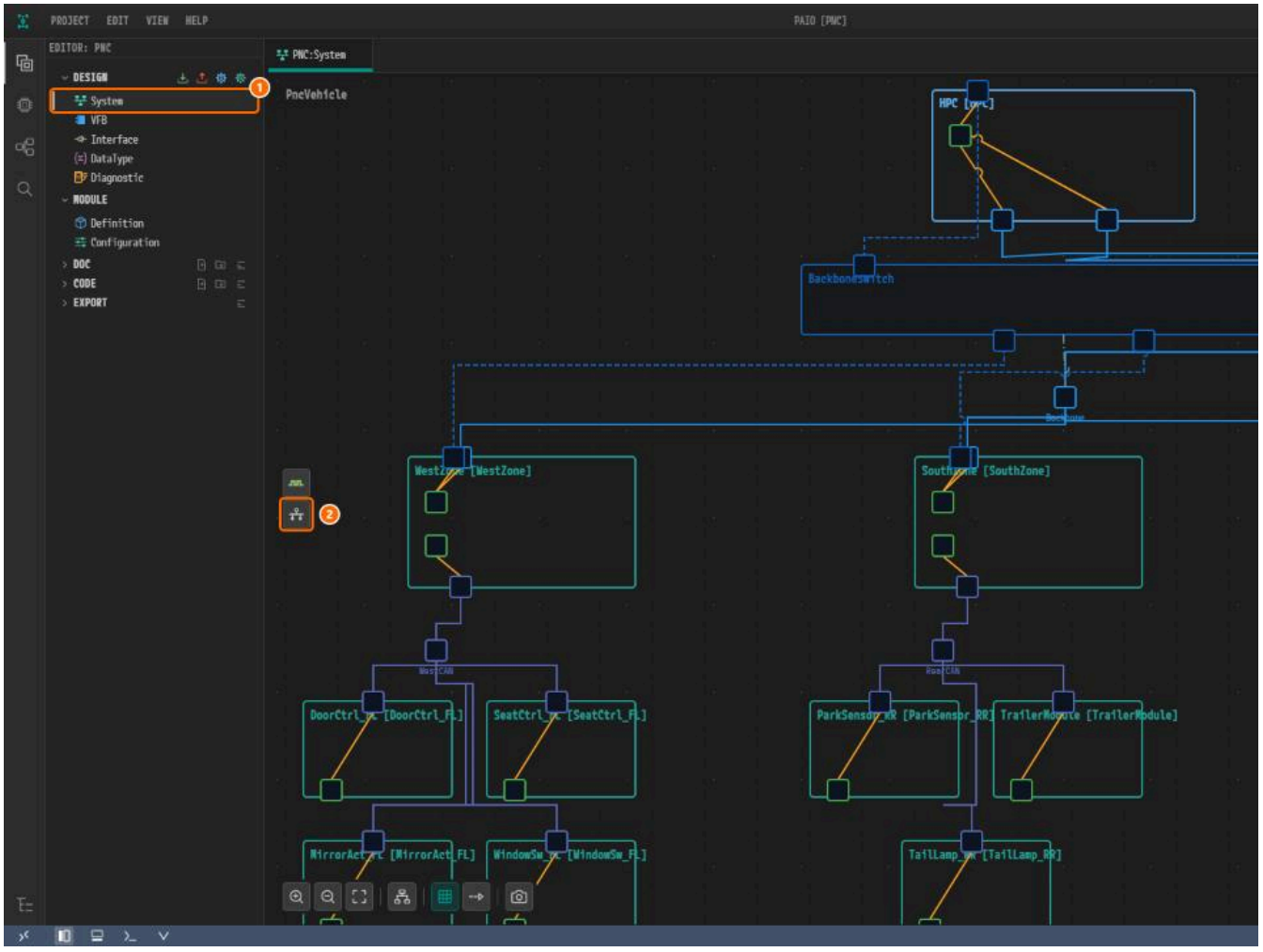


*그림 7-1. ① **DESIGN** 아래의 **System**, ② PNC 버튼.*

3. **PARTIAL NETWORKS** 목록에서 **PNC 18 PNC_ParkAssist**를 누른다.

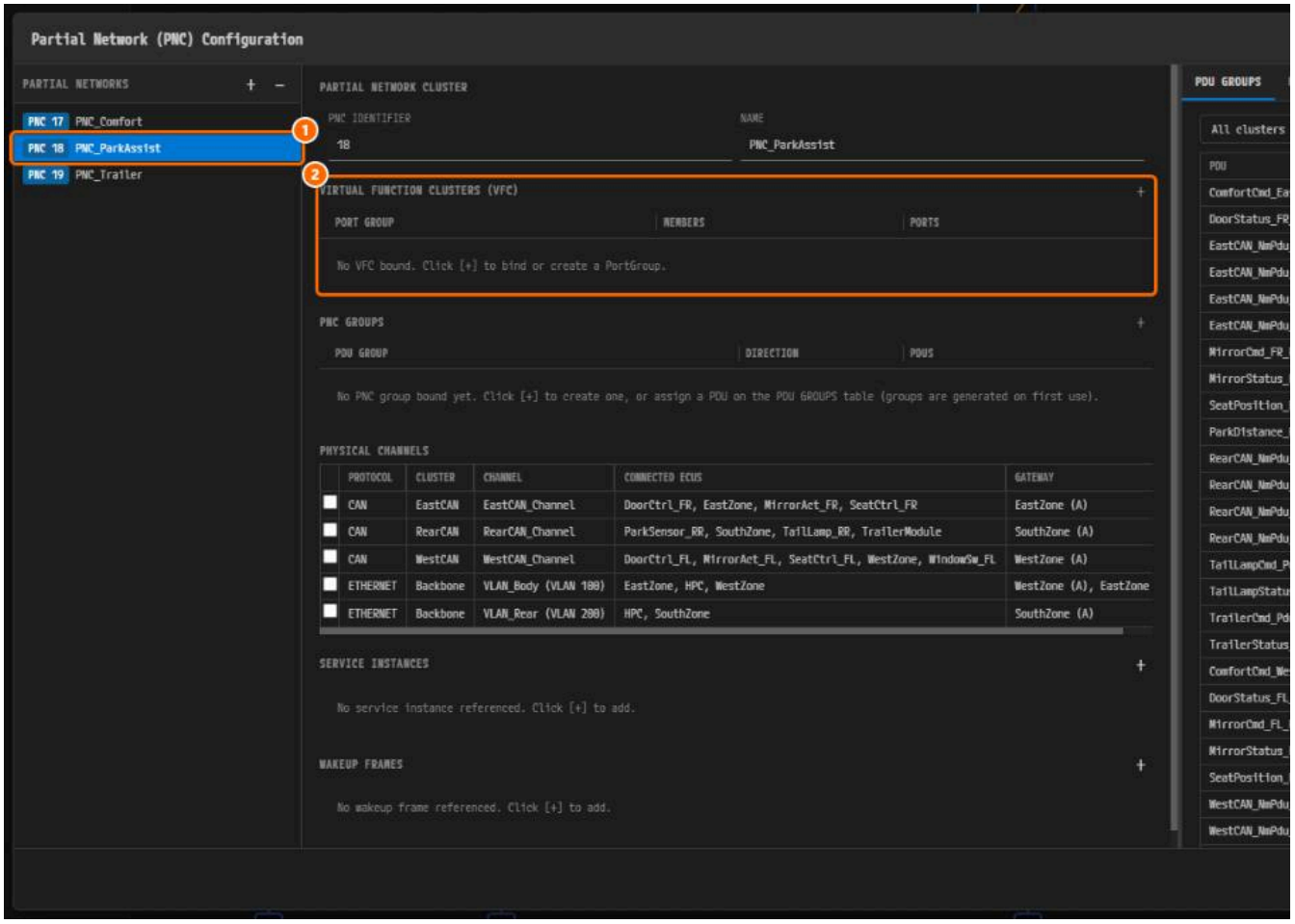


*그림 7-2. ① **PNC 18 PNC_ParkAssist**, ② 아직 비어 있는 **VIRTUAL FUNCTION CLUSTERS (VFC)**.*

가운데에는 선택한 PNC의 섹션이, 오른쪽에는 뷰어 탭 **PDU GROUPS**, **MATRIX**, **NM VECTOR**가 있다. 다이얼로그 제목 앞의 **●**는 저장하지 않은 변경을 뜻한다. 저장은 **Ctrl+S**

VFC 바인딩하기

1. **VIRTUAL FUNCTION CLUSTERS (VFC)** 머리글 오른쪽의 **+**를 누른다. 메뉴가 열린다.
2. **VFC_ParkAssist**를 누른다.

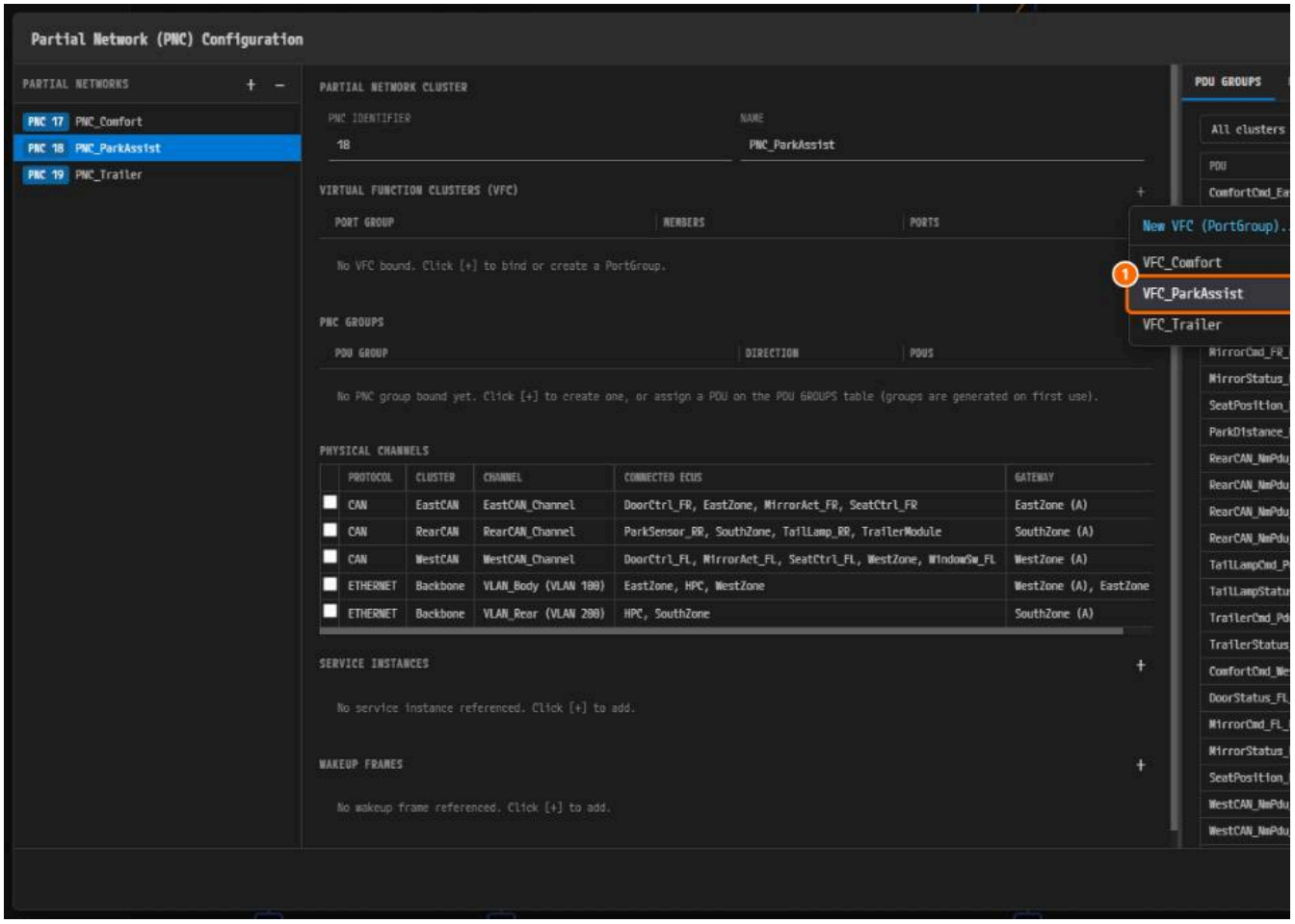


그림 7-3. ① `VFC_ParkAssist`.

VFC 표에 행이 생긴다.

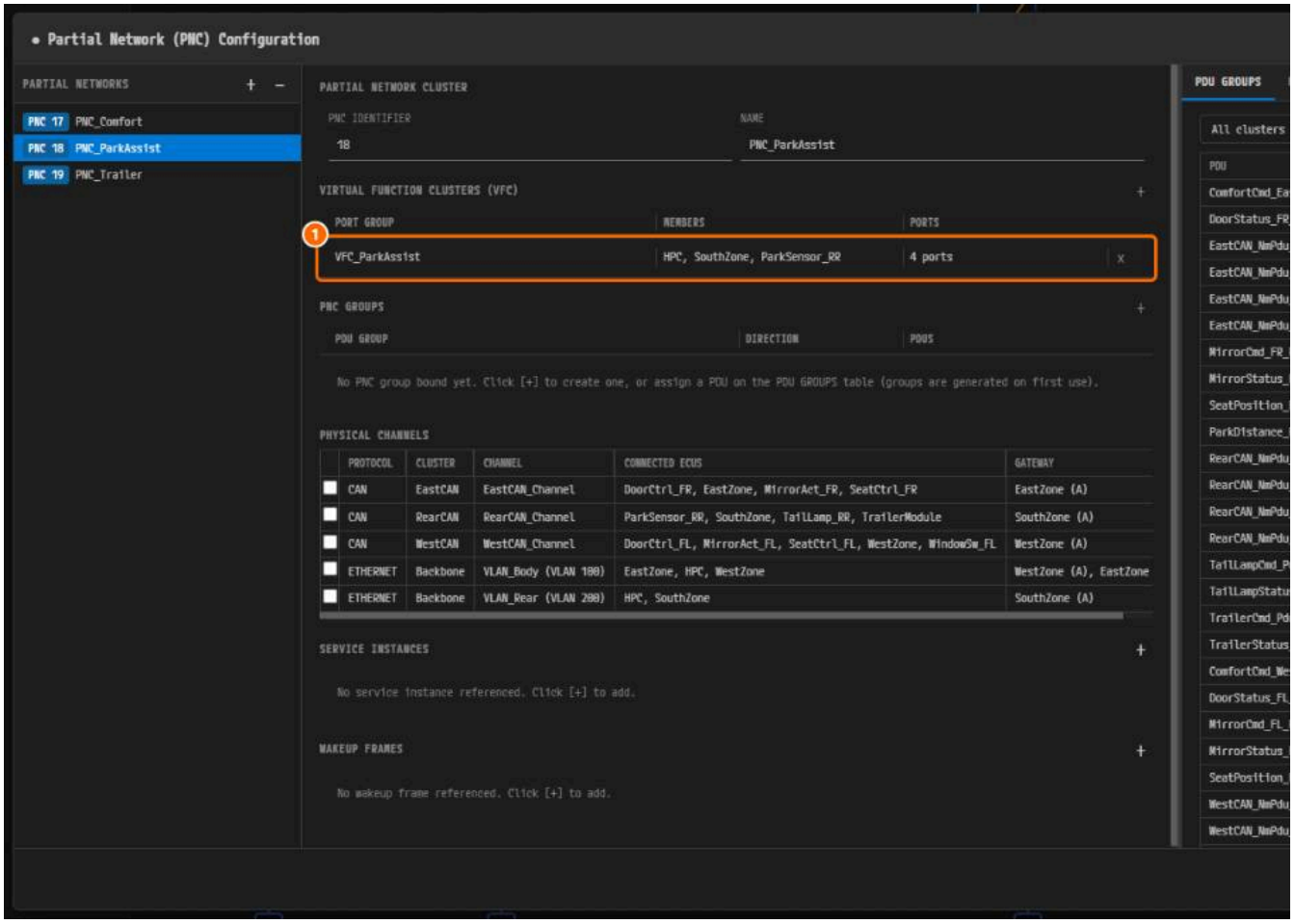


*그림 7-4. ① **MEMBERS** `HPC, SouthZone, ParkSensor_RR`, **PORTS** `4 ports`.*

3. `VFC_ParkAssist` 행을 눌러 펼친다.

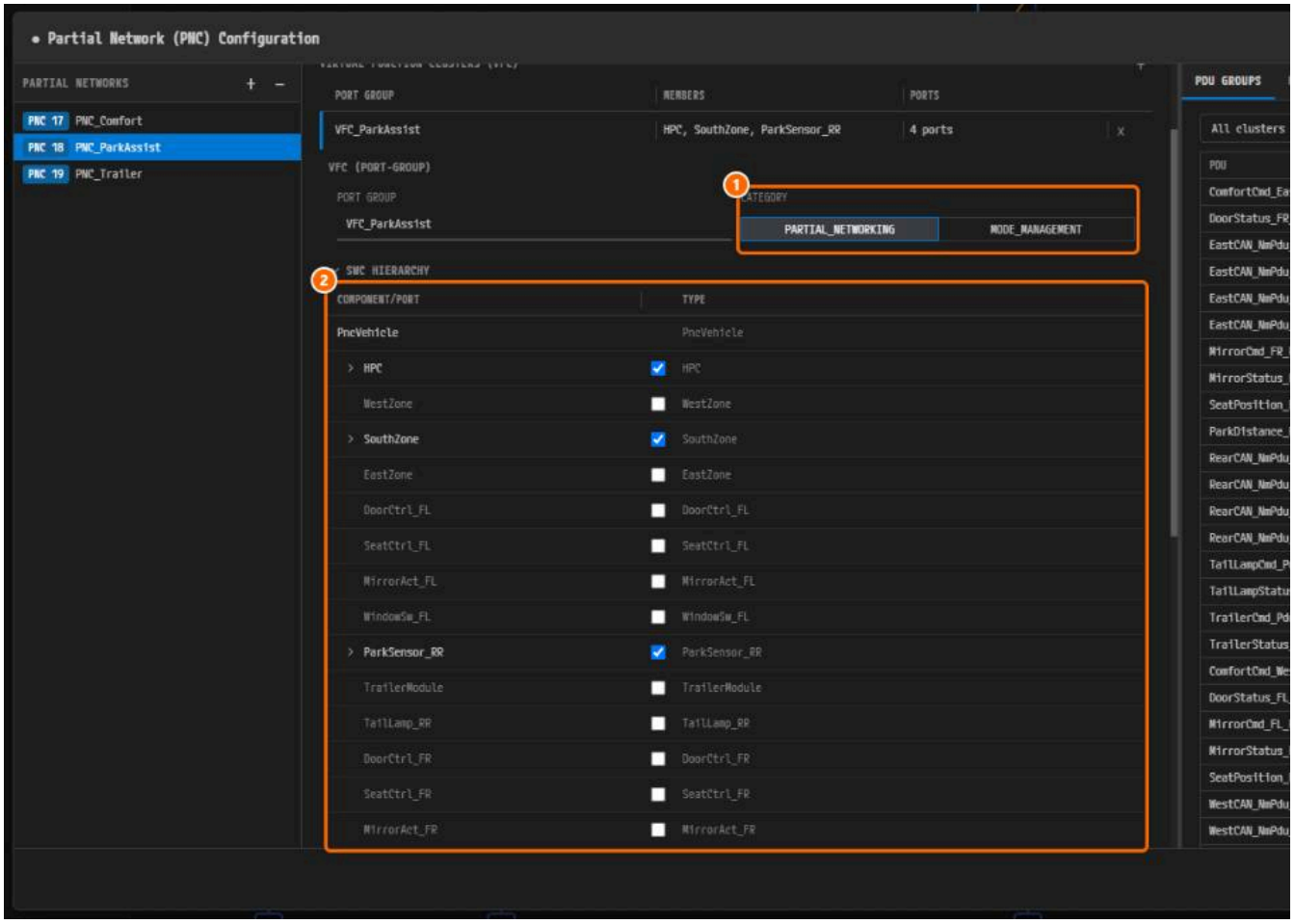


*그림 7-5. ① **CATEGORY PARTIAL_NETWORKING**, ② **SWC HIERARCHY**.*

4. **SWC HIERARCHY**에서 컴포지션 `HPC`, `SouthZone`, `ParkSensor_RR`을 체크한다. 각 행을 펼쳐 포트를 체크한다.

컴포지션	체크할 포트
`HPC`	`ParkDistanceSvc`
`SouthZone`	`ParkDistance`, `ParkDistanceSvc`
`ParkSensor_RR`	`ParkDistance`

   `HPC` 안의 AA `ParkAssistApp`의 포트 `ParkDistanceSvc`, `VehicleStateSvc`도 체크한다.

5. 행을 다시 눌러 접는다.

> **주의.** **SWC HIERARCHY**의 체크는 **Ctrl+S** 없이 바로 모델에 기록된다. 컴포지션 체크를 풀면 그 PortGroup이 지워지므로 표에 있는 행만 건드린다.

PNC GROUPS 확인하기

PNC GROUPS에는 4장에서 만든 그룹이 이미 있다. `RearCAN_PNC_ParkAssist_Rx`(**RX**, `1 PDU`)와 `RearCAN_PNC_ParkAssist_Tx`(**TX**, `0 PDUs`)가 보이는지 확인한다.

아래 단계는 그룹을 바인딩하는 다른 방법, 곧 **PDU GROUPS** 표에서 PDU에 PNC를 지정하는 방법이다. 아직 PNC가 없는 PDU에만 쓴다. 예제에서는 이미 끝났으므로 읽기만 한다.

1. **PDU GROUPS**의 검색칸에 `ParkDistance`를 입력한다. 뷰어 왼쪽 경계선을 왼쪽으로 끌어 **PNC MEMBERSHIP**이 보이게 한다.

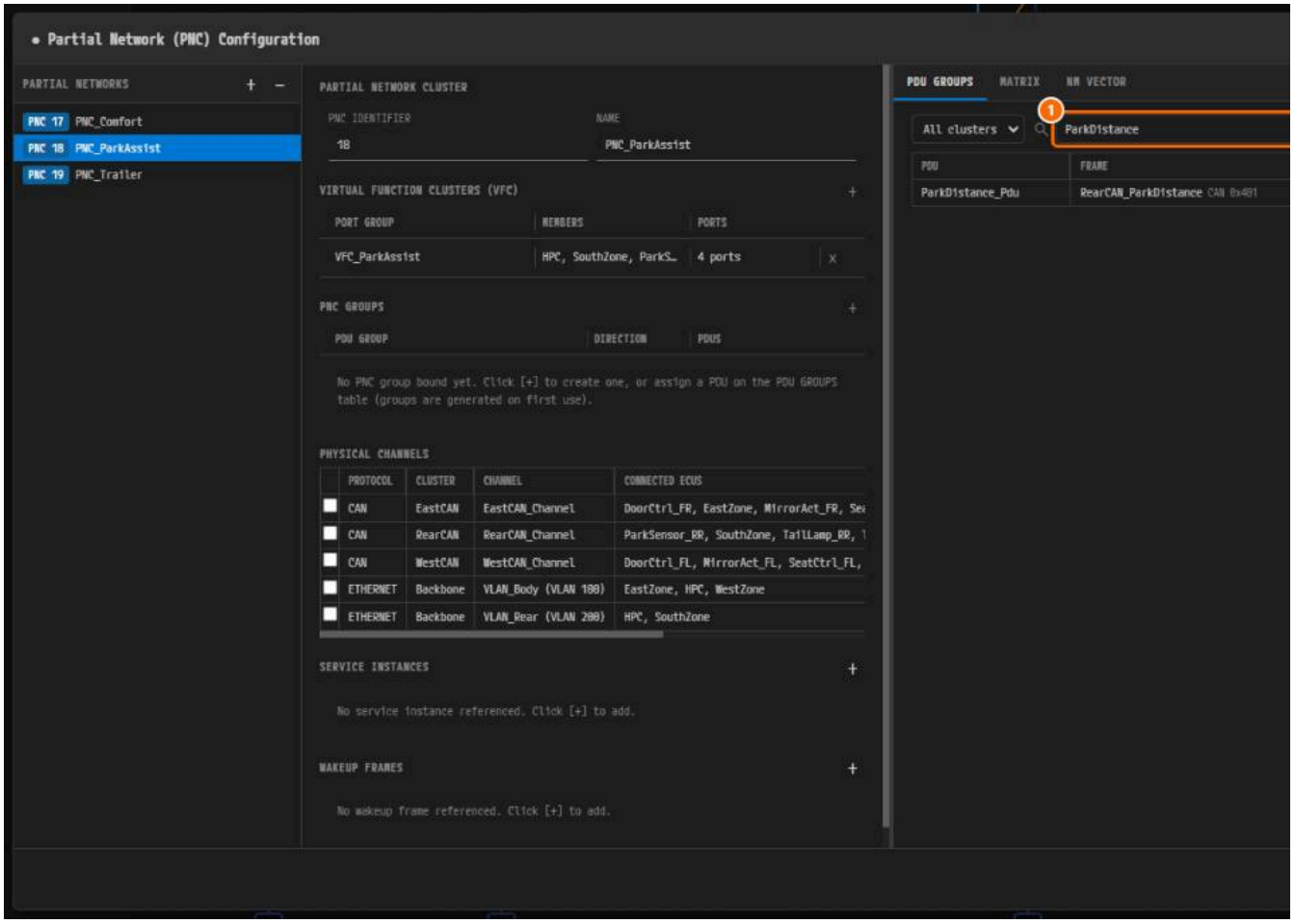


*그림 7-6. ① 검색칸, ② **PNC MEMBERSHIP.***

2. **PNC MEMBERSHIP** 칸을 누르고 **PNC_ParkAssist**를 체크한다.

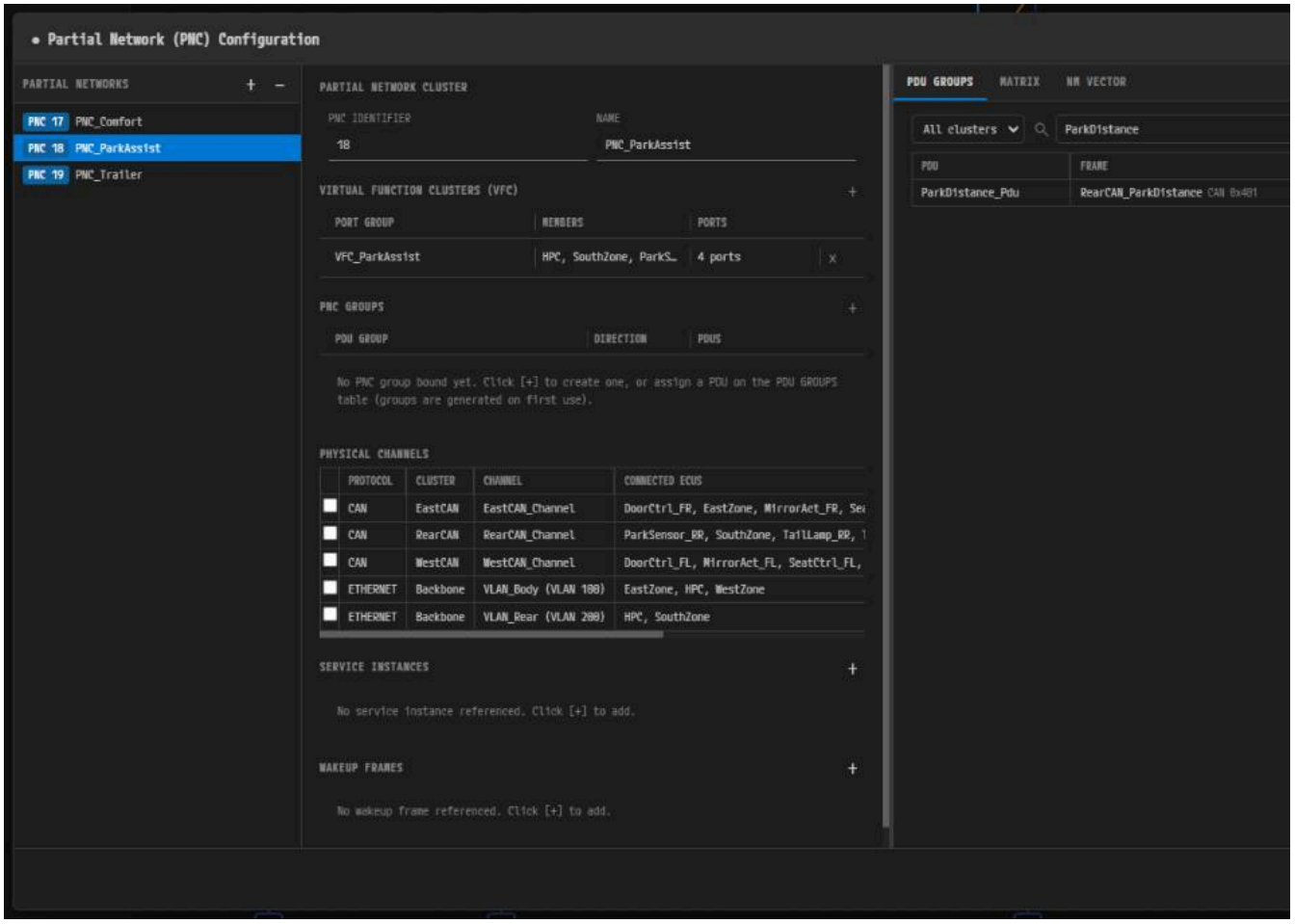


그림 7-7. ① `PNC_ParkAssist`.

3. **Esc**를 누른다. 그룹 `RearCAN_PNC_ParkAssist_Rx`와 `RearCAN_PNC_ParkAssist_Tx`가 바인딩된다.

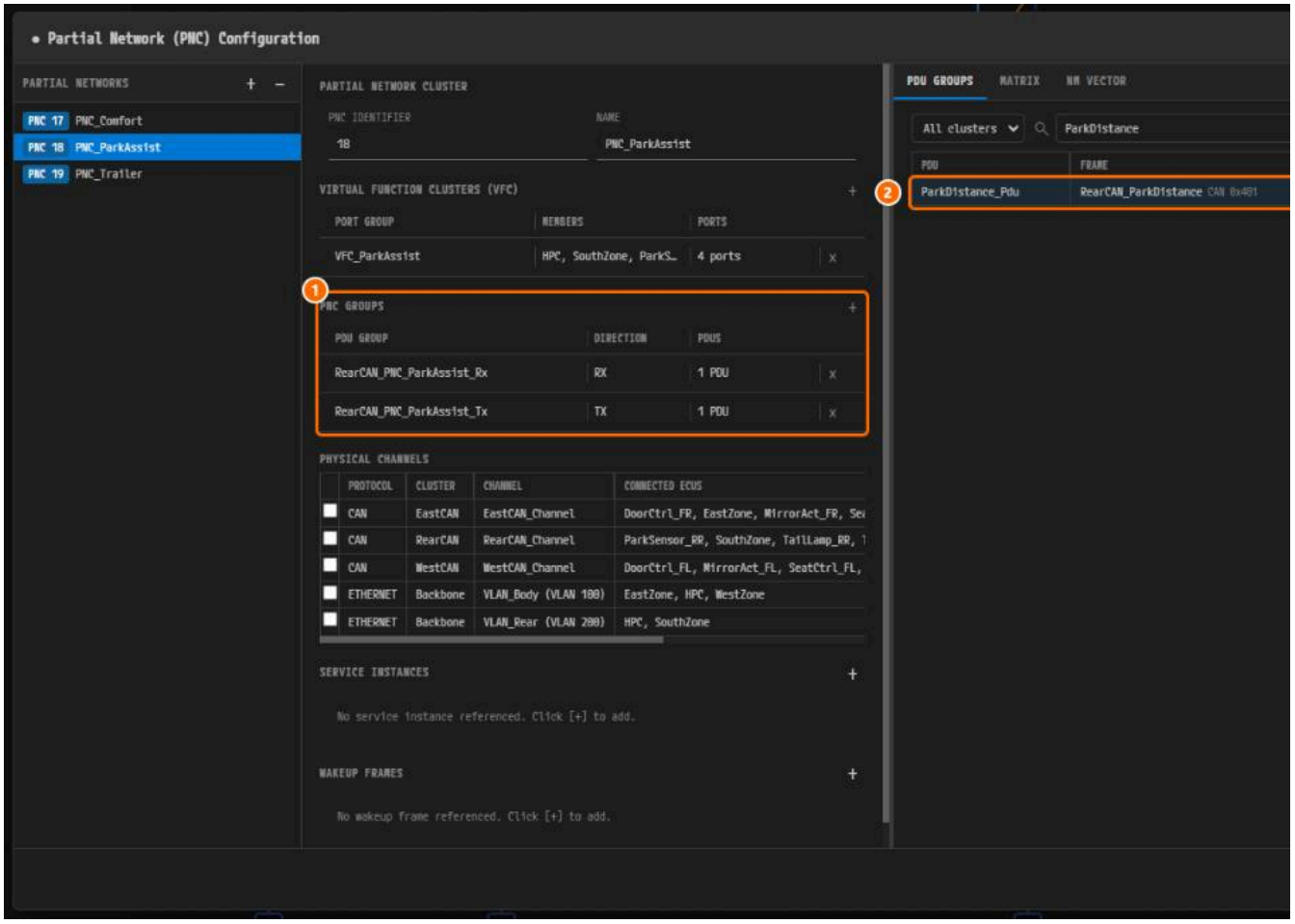


그림 7-8. ① 바인딩된 그룹 쌍, ② `PNC_ParkAssist`가 표시된 `ParkDistance_Pdu`.

PDU는 두 그룹 모두에 들어간다. `SouthZone`이 받는 PDU이므로 Rx 그룹에만 둔다.

4. 경계선을 되돌린다. `RearCAN_PNC_ParkAssist_Tx`를 펼치고 **CONTAINED PDUS**의 `ParkDistance_Pdu` 행에서 **×**를 누른다.

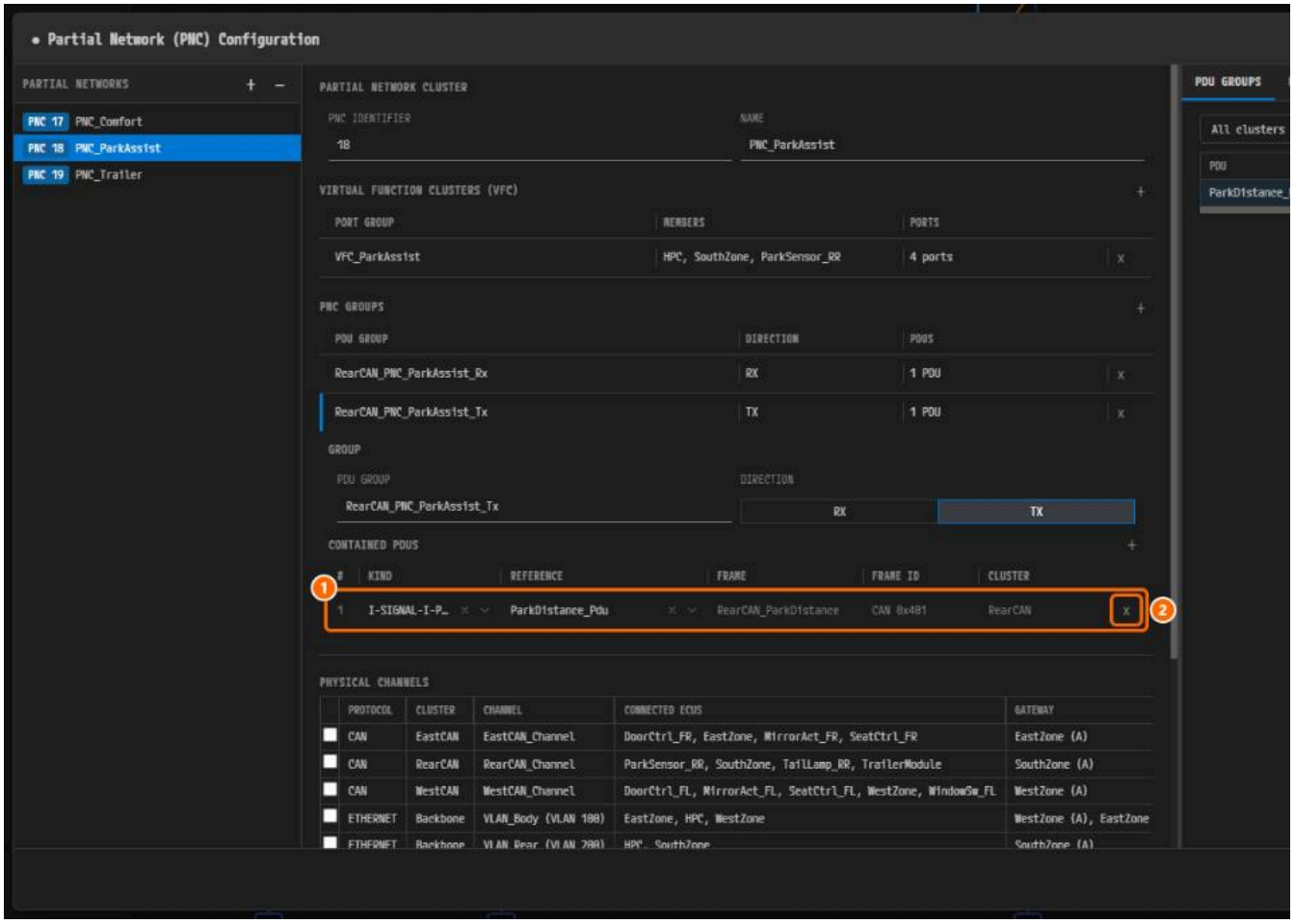


그림 7-9. ① `ParkDistance_Pdu`, ② **×**.

Tx 그룹이 `0 PDUs`가 된다.

5. 행을 접고 검색칸을 비운다.

> **주의.** 그룹 행 오른쪽 끝의 **×**는 그룹 자체를 삭제한다.

물리 채널 선택하기

1. **PHYSICAL CHANNELS**에서 **CAN** / `RearCAN_Channel`과 **ETHERNET** / `VLAN_Rear (VLAN 200)`을 체크한다.

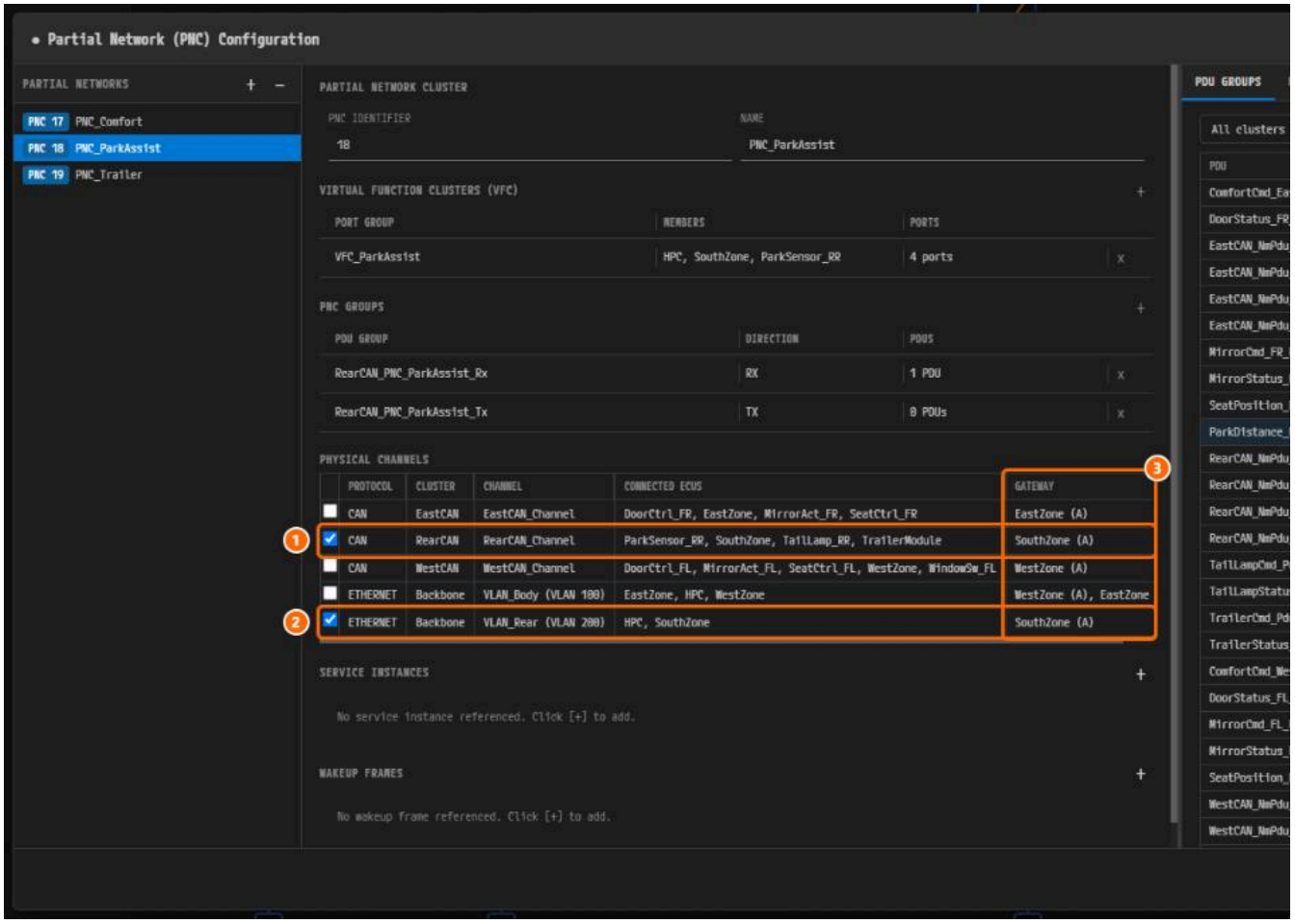


*그림 7-10. ① `RearCAN_Channel`, ② `VLAN_Rear (VLAN 200)`, ③ 표시 전용인 **GATEWAY**.*

서비스 인스턴스 추가하기

1. **SERVICE INSTANCES** 오른쪽의 **+**를 누르고 `ParkDistance`를 입력한 뒤 **ParkDistance_Provider**를 누른다.

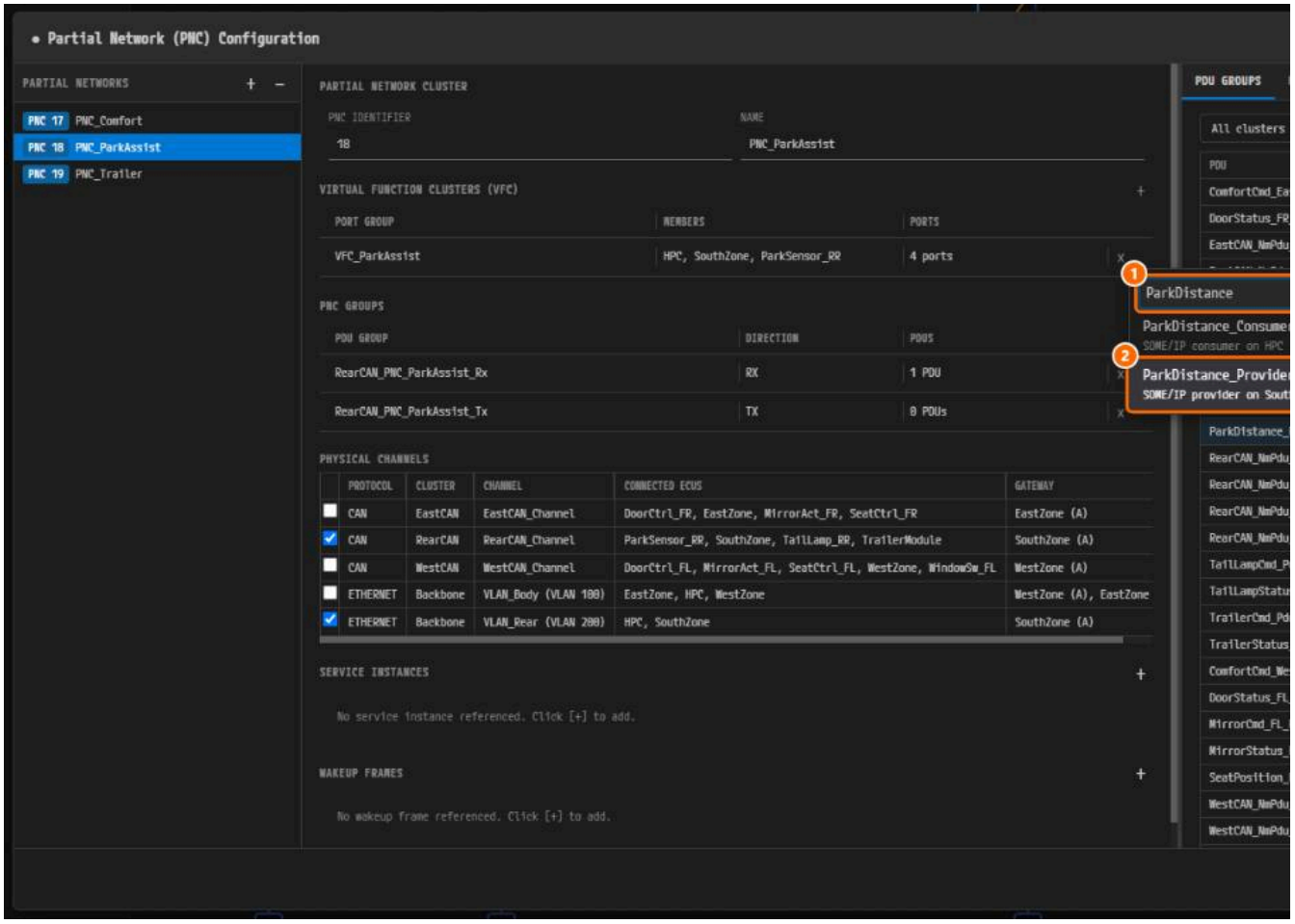


그림 7-11. ① 검색칸, ② `ParkDistance_Provider`.

2. 같은 방법으로 **ParkDistance_Consumer**를 추가한다. 카드 2개가 생긴다. 카드를 누르면 상세가 보인다.

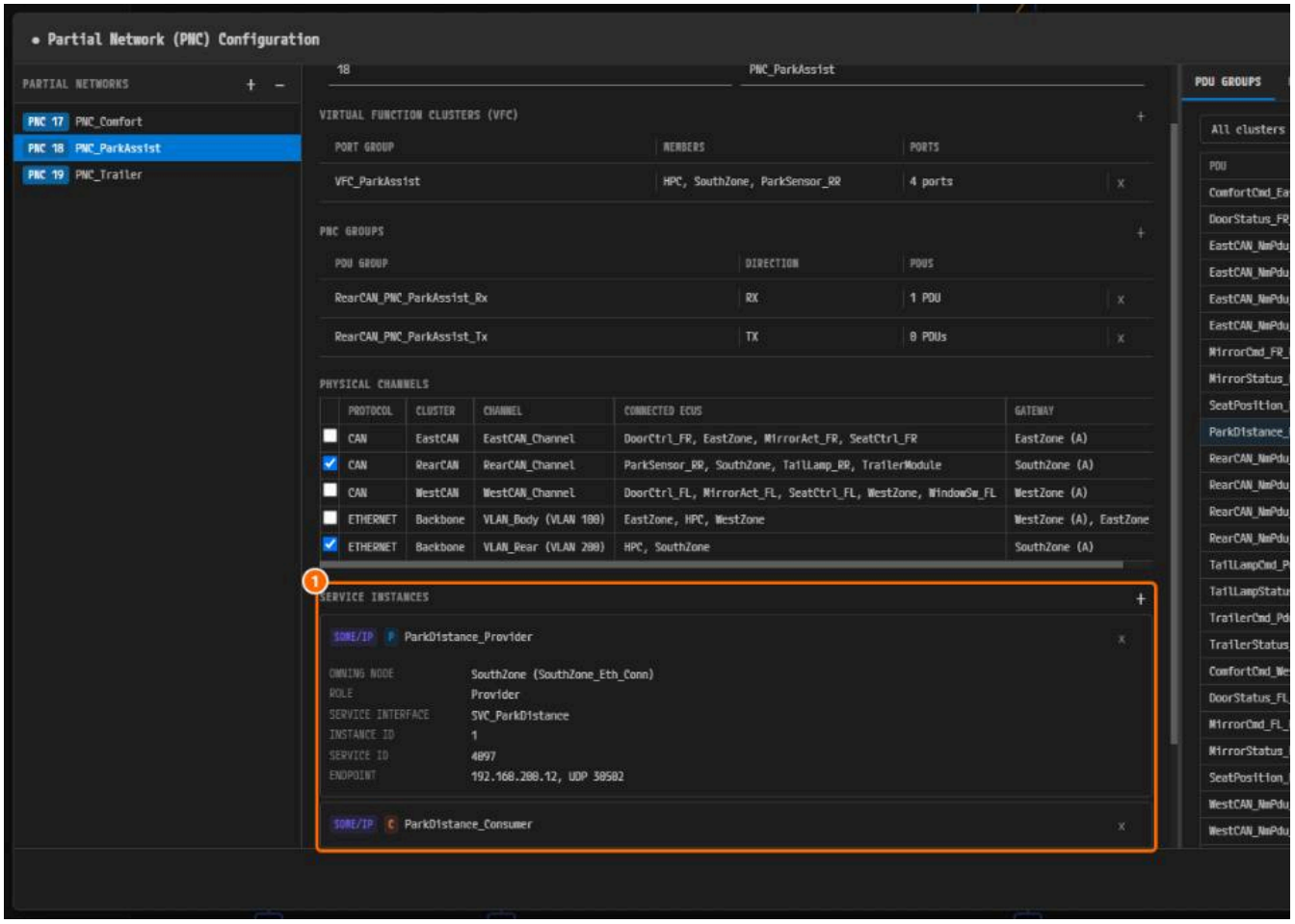


*그림 7-12. ① `ParkDistance_Provider` 를 펼친 **SERVICE INSTANCES**.*

저장하기

1. **Ctrl+S**를 누른다. 제목 앞의 **●**가 사라진다.

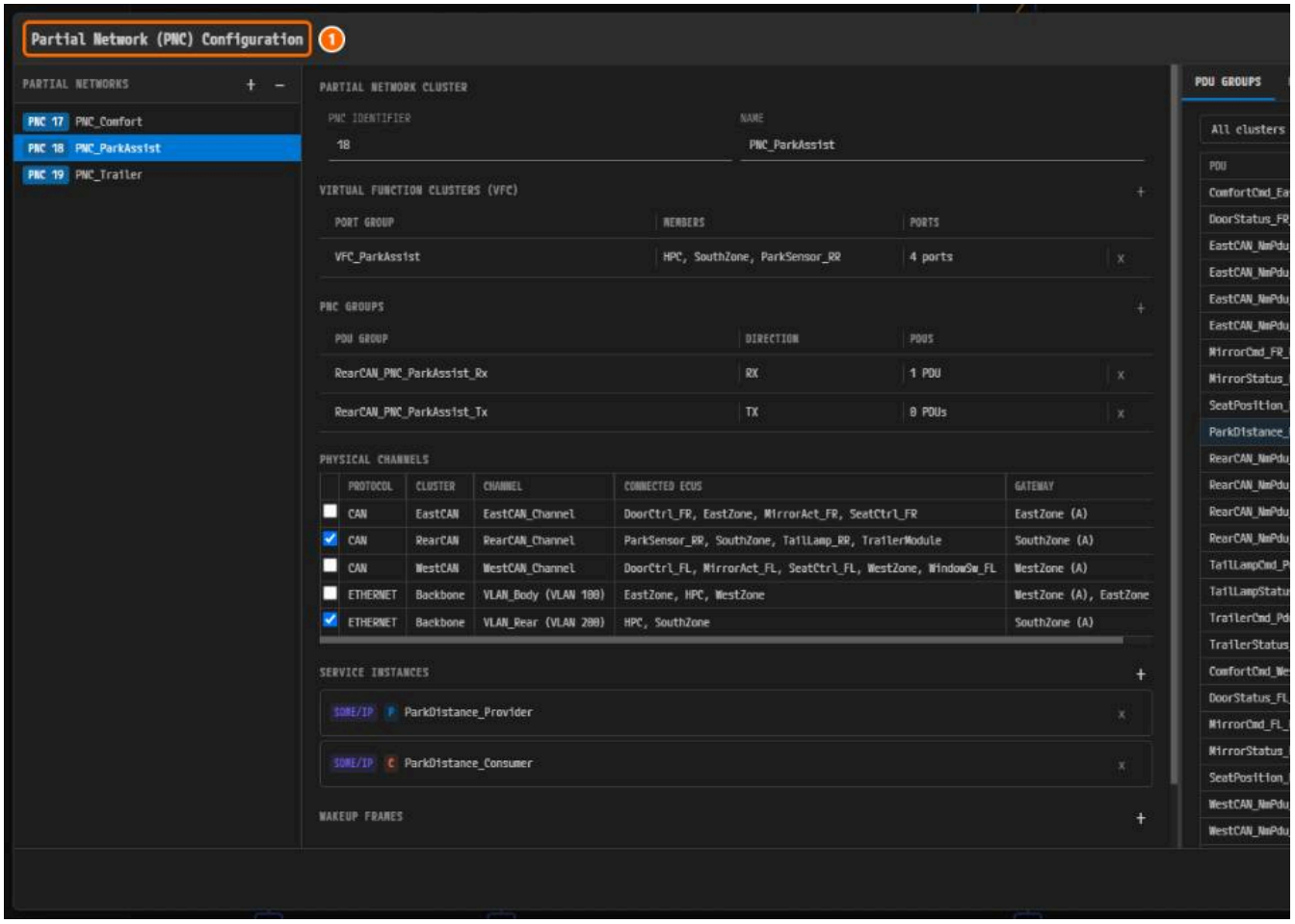


*그림 7-13. ① **●*가 없는 제목.*

웨이크업 프레임 추가하기

`PNC_ParkAssist`에는 웨이크업 프레임이 없다. `PNC_Trailer`에 입력한다.

1. 목록에서 **PNC 19 PNC_Trailer**를 누른다.
2. **WAKEUP FRAMES** 오른쪽의 **+**를 누르고 `TrailerModule`을 입력한 뒤 **RearCAN_NmFrame_TrailerModule**을 누른다.

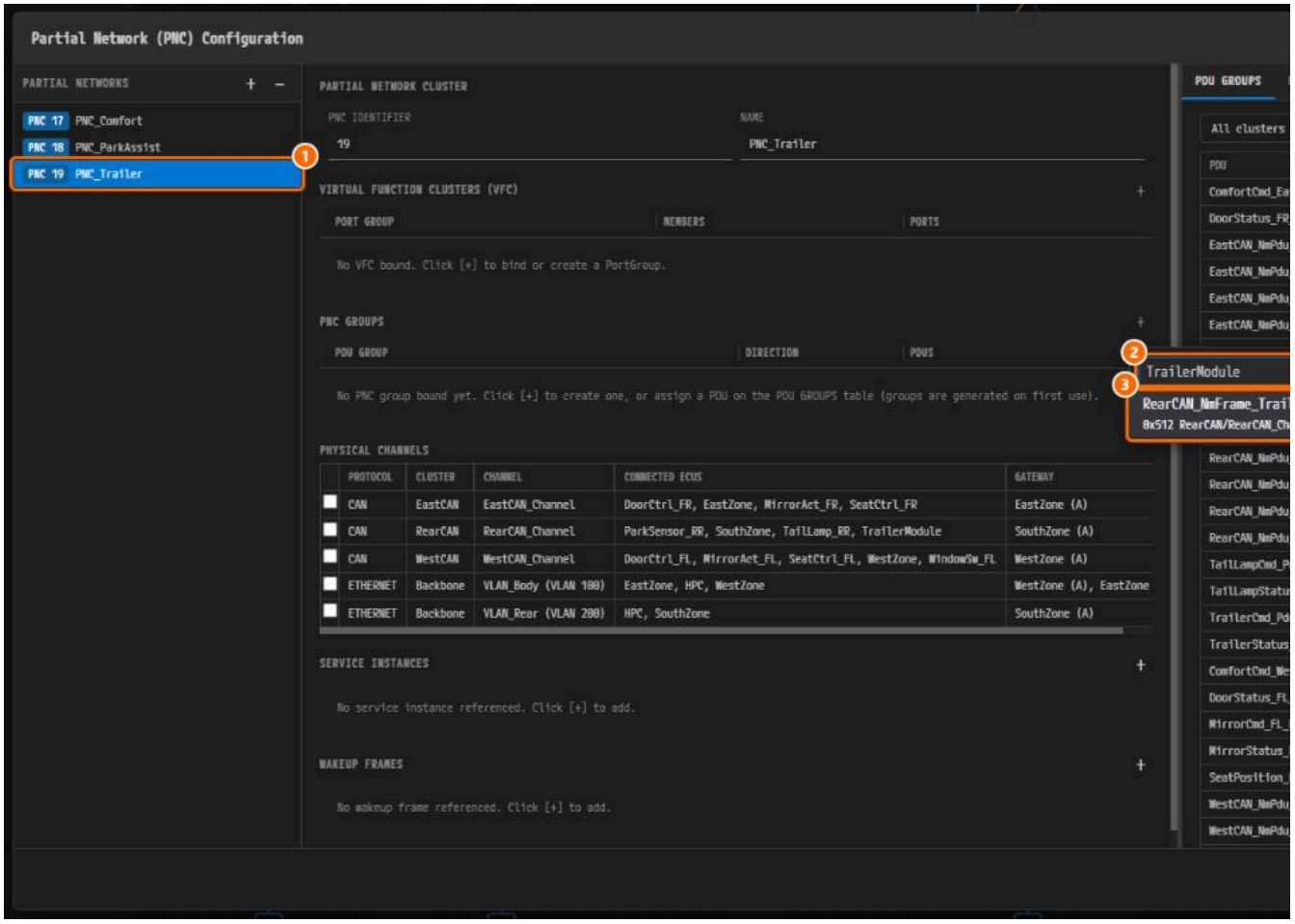


*그림 7-14. ① **PNC 19 PNC_Trailer**, ② 검색칸, ③ `RearCAN_NmFrame_TrailerModule(0x512)`. 독자의 화면에서는 목록이 **PNC RX FRAMES** 머리글로 시작한다.*

3. **CAN** 카드가 생긴다. 카드를 누르면 상세가 보인다.

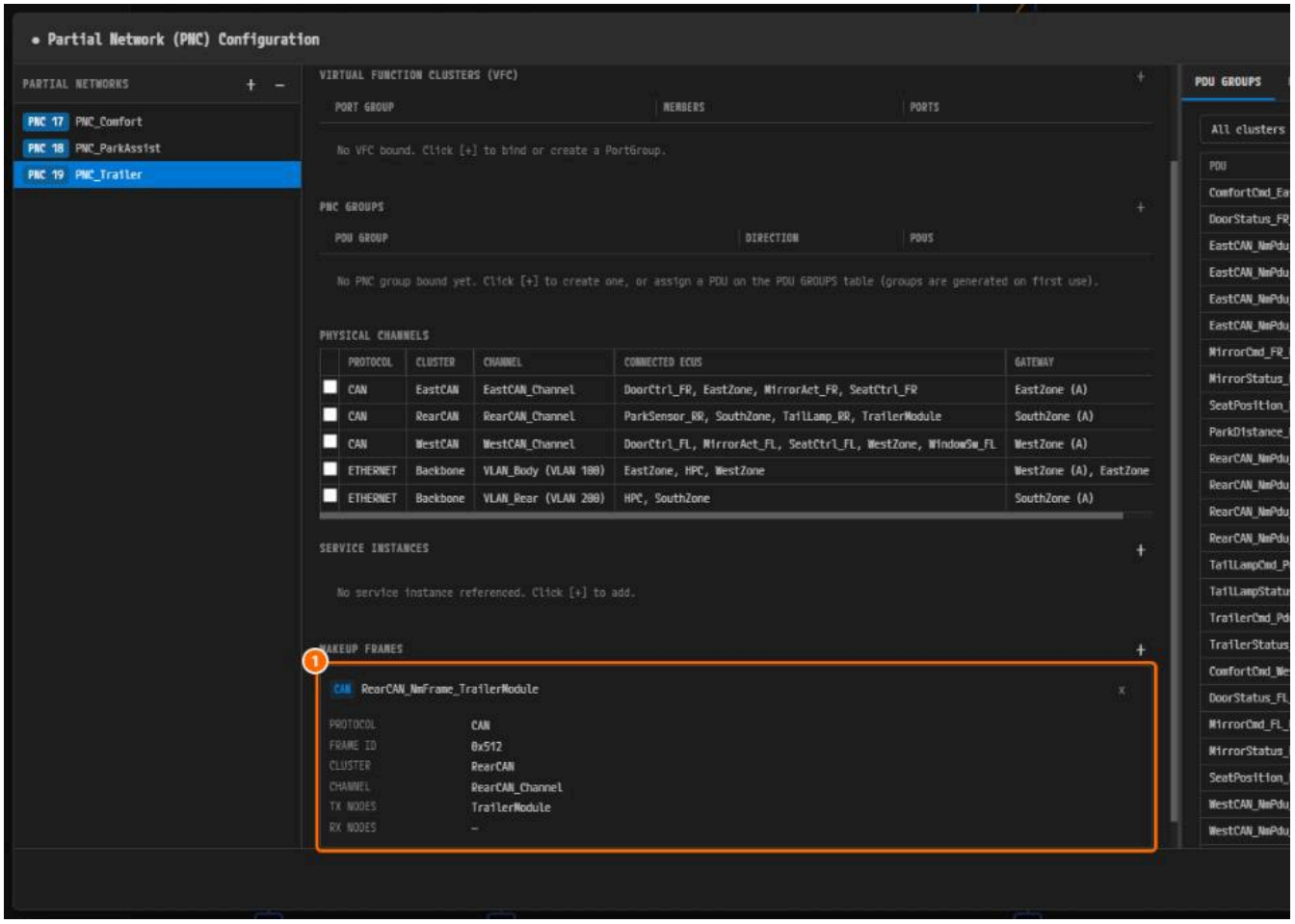


*그림 7-15. ① **FRAME ID** `0x512`, **TX NODES** `TrailerModule`.*

4. **Ctrl+S**를 누르고 **Close**를 누른다.

나머지 PNC의 값

`PNC_Comfort`와 `PNC_Trailer`도 표대로 같은 방법으로 바인딩한다. **PNC GROUPS**는 이미 채워져 있으므로 확인만 한다.

PNC	VFC	PNC GROUPS	PHYSICAL CHANNELS	SERVICE INSTANCES
PNC 17 `PNC_Comfort`	`VFC_Comfort`	`WestCAN_PNC_Comfort_Rx`, `WestCAN_PNC_Comfort_Tx`, `EastCAN_PNC_Comfort_Rx`, `EastCAN_PNC_Comfort_Tx`	`EastCAN_Channel`, `WestCAN_Channel`, `VLAN_Body (VLAN 100)`	`ComfortControl_Provider_West`(P), `ComfortControl_Provider_East`(P), `ComfortControl_Consumer`(C)
PNC 18 `PNC_ParkAssist`	`VFC_ParkAssist`	`RearCAN_PNC_ParkAssist_Rx`, `RearCAN_PNC_ParkAssist_Tx`	`RearCAN_Channel`, `VLAN_Rear (VLAN 200)`	`ParkDistance_Provider`(P), `ParkDistance_Consumer`(C)
PNC 19 `PNC_Trailer`	`VFC_Trailer`	`RearCAN_PNC_Trailer_Rx`, `RearCAN_PNC_Trailer_Tx`	`RearCAN_Channel`, `VLAN_Rear (VLAN 200)`	`VehicleState_Provider`(P), `VehicleState_Consumer`(C)

VFC를 바인딩한 뒤 **SWC HIERARCHY**에서 멤버를, 그리고 VFB와 VFC(PortGroup)의 **OUTER PORTS**대로 그 포트를 체크한다.

VFC	MEMBERS	PORTS
`VFC_Comfort`	`HPC`, `WestZone`, `EastZone`, `DoorCtrl_FL`, `SeatCtrl_FL`, `MirrorAct_FL`, `WindowSw_FL`, `DoorCtrl_FR`, `SeatCtrl_FR`, `MirrorAct_FR` (10개)	`28 ports`
`VFC_ParkAssist`	`HPC`, `SouthZone`, `ParkSensor_RR` (3개)	`4 ports`
`VFC_Trailer`	`HPC`, `SouthZone`, `TrailerModule`, `TailLamp_RR` (4개)	`8 ports`

> **참고.** 4장을 건너뛰었다면 **PDU GROUPS**에서 `TrailerStatus_Pdu`, `TailLampStatus_Pdu`, `TrailerCmd_Pdu`, `TailLampCmd_Pdu`에 **PNC_Trailer**를 지정한다. 이어 `TailLampStatus_Pdu`를, Rx 그룹에서 `TrailerCmd_Pdu`, `TailLampCmd_Pdu`를 뺀다. 버스 두 개에 걸치는 `PNC_Comfort`에는 이 방법을 쓰지 않는다.

완성 화면

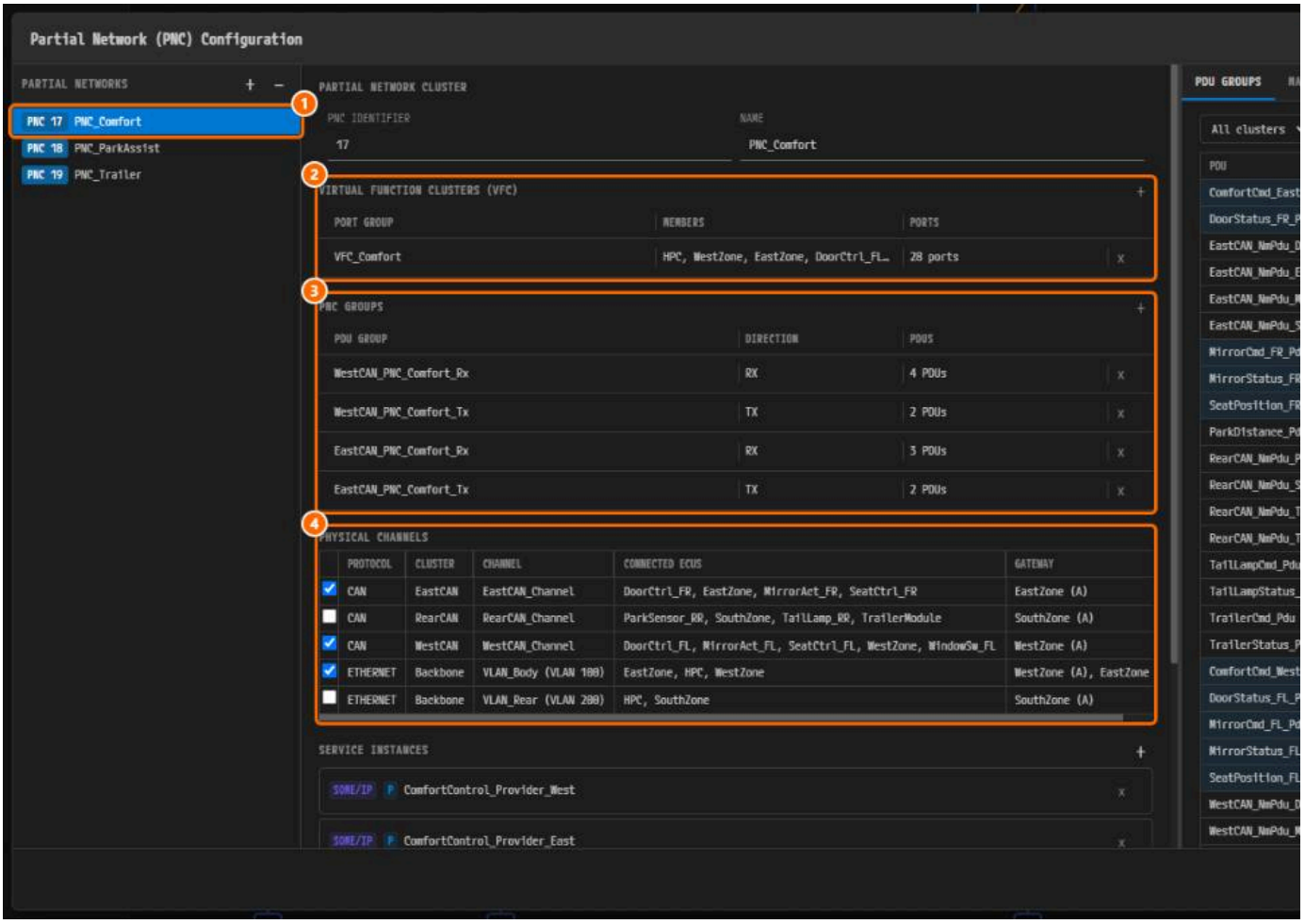


*그림 7-16. `PNC_Comfort` 위쪽. ① **PNC 17 PNC_Comfort**, ② `VFC_Comfort`, ③ **PNC GROUPS** 4개, ④ **PHYSICAL CHANNELS** 3개.*

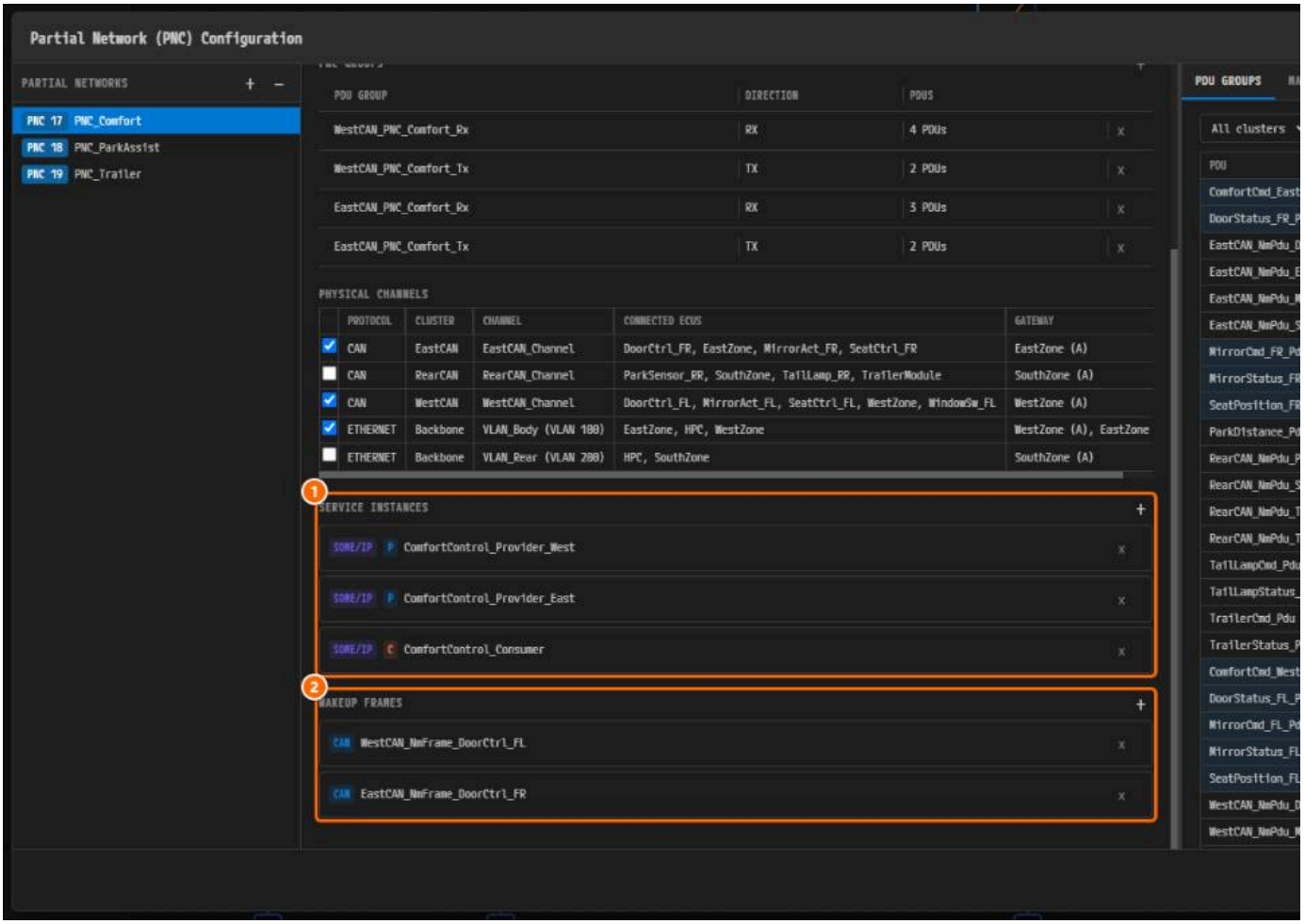


*그림 7-17. `PNC_Comfort` 아래쪽. ① **SERVICE INSTANCES**, ② **WAKEUP FRAMES**.*

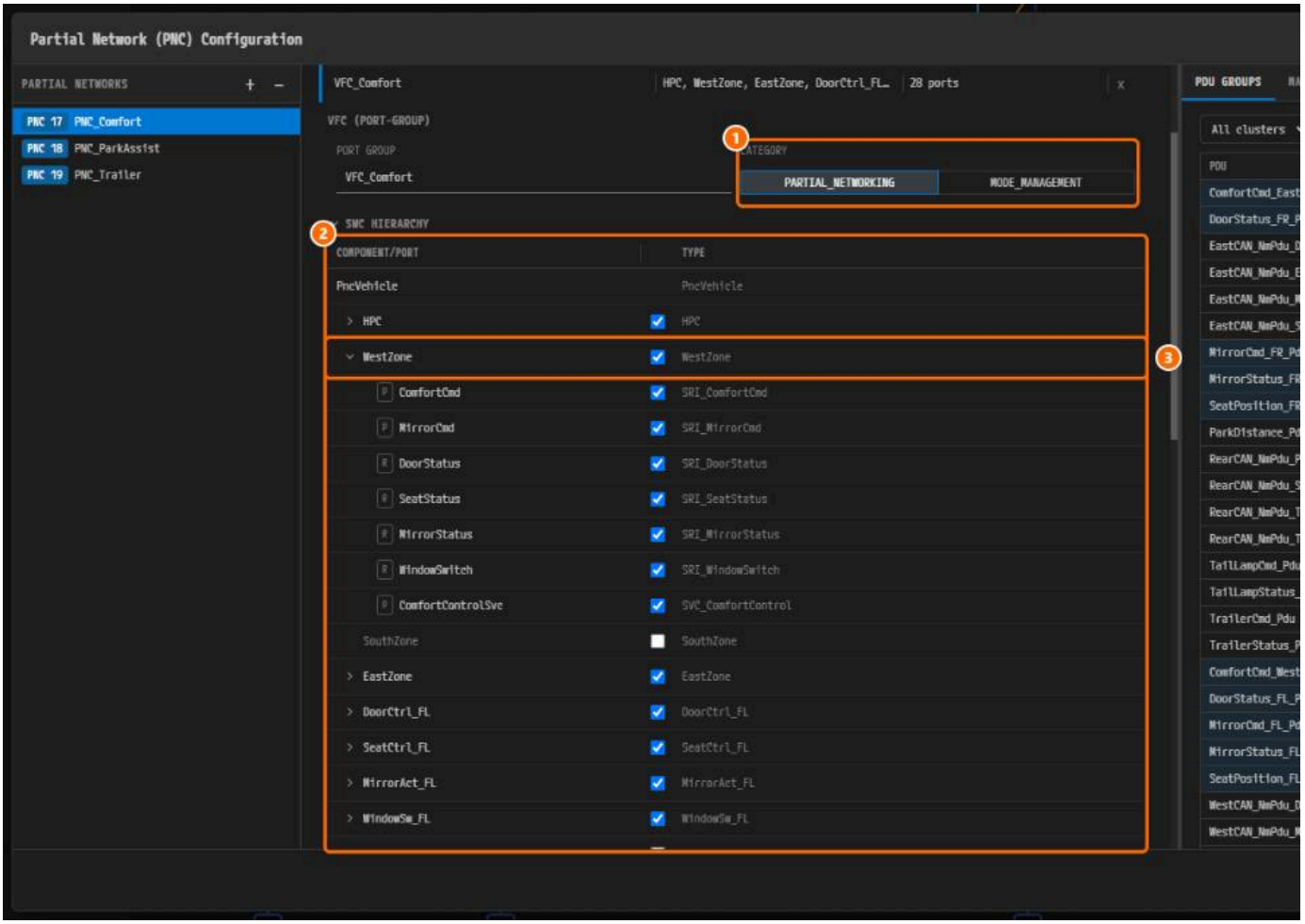


*그림 7-18. 펼친 `VFC_Comfort`. ① **CATEGORY**, ② **SWC HIERARCHY**, ③ 포트 7개가 모두 체크된 `WestZone`.*

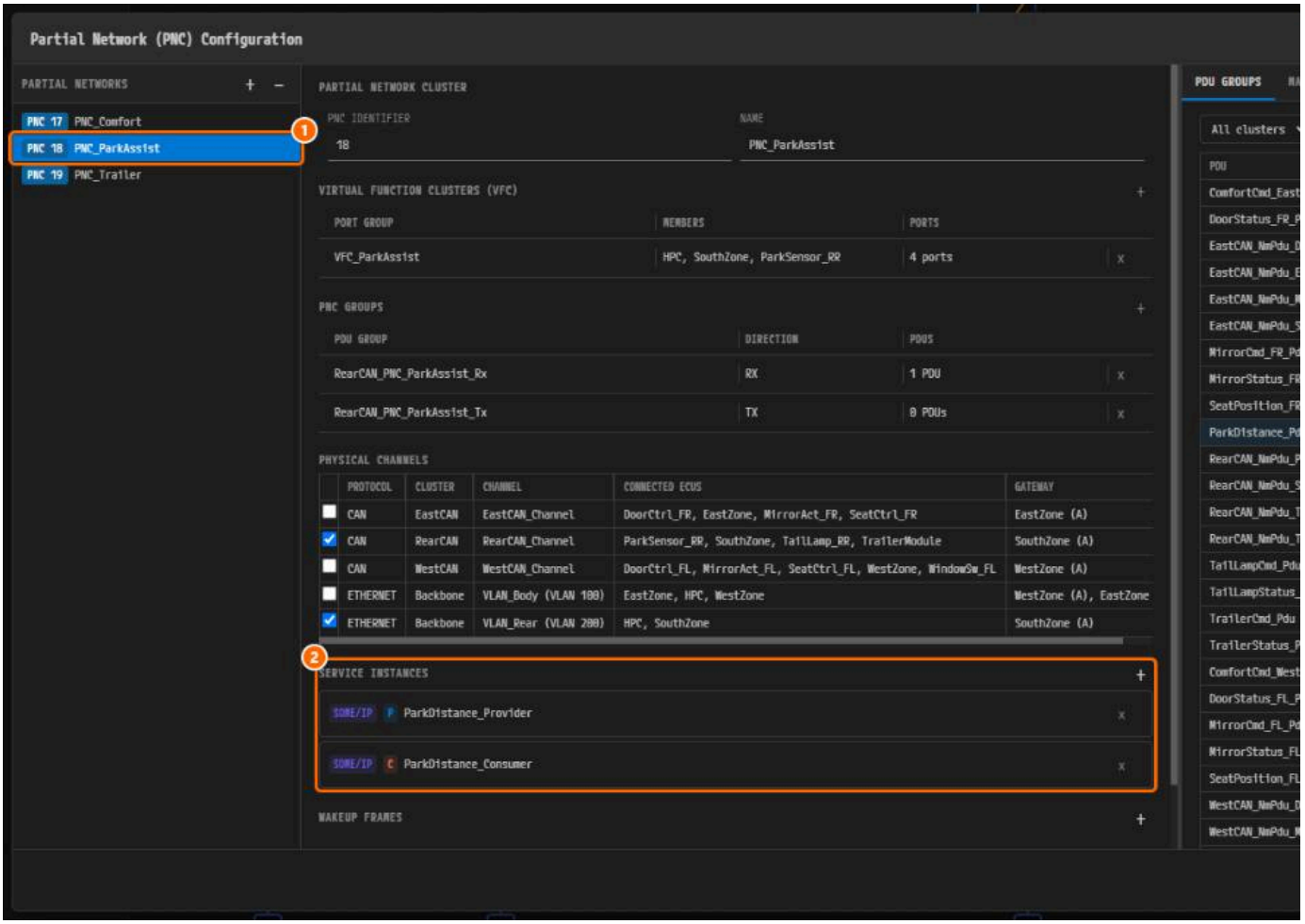


*그림 7-19. ① **PNC 18 PNC_ParkAssist**, ② **SERVICE INSTANCES**.*

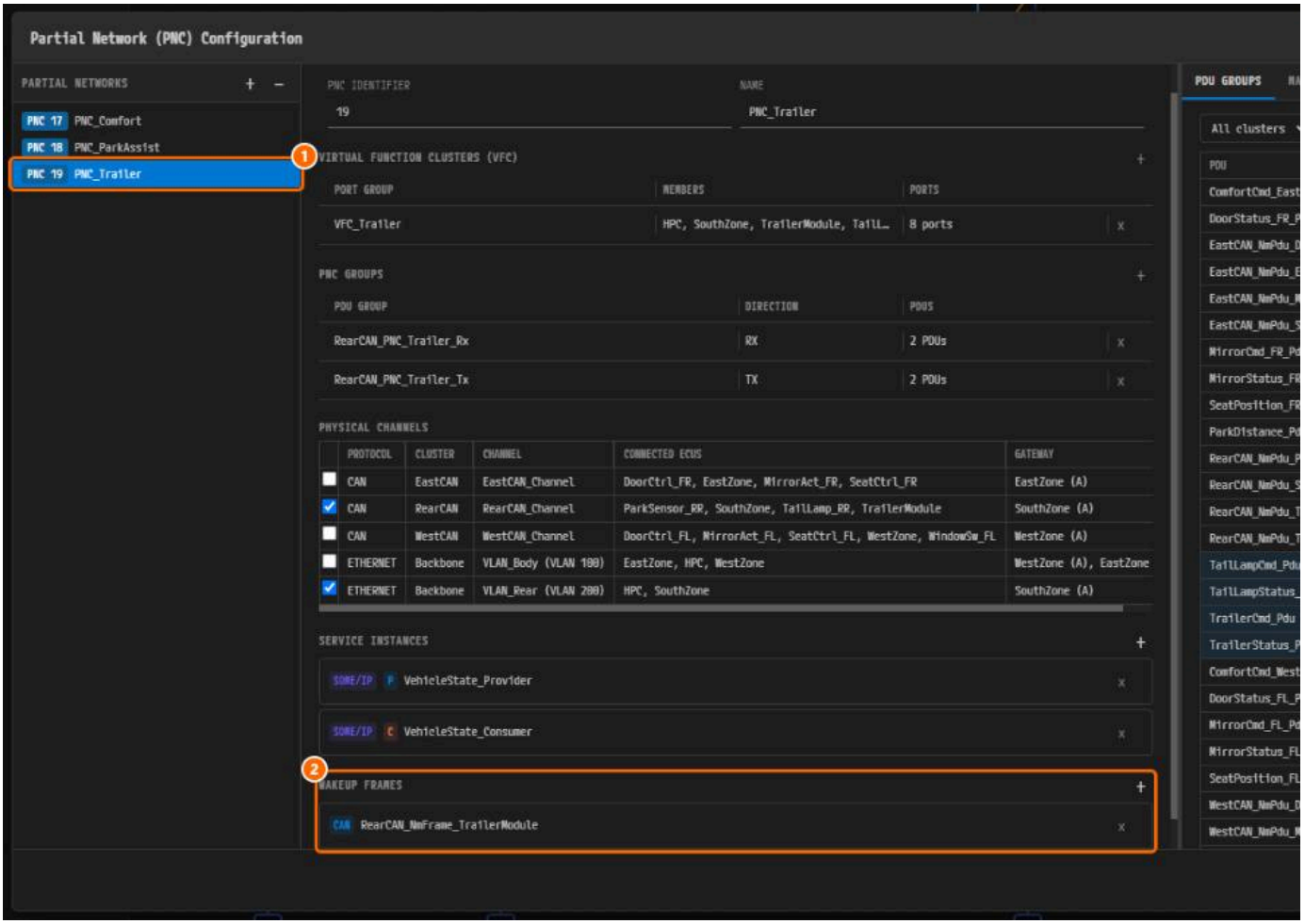


*그림 7-20. 아래로 스크롤한 `PNC_Trailer`. ① **PNC 19 PNC_Trailer**, ② **WAKEUP FRAMES**.*

왜 이렇게 설계하는가

- **PNC 하나에 화면 하나.** PNC의 참조가 한곳에 모여 있어 빠진 것이 바로 보인다.
- **물리 채널은 직접 체크한다.** 그룹이나 PDU에서 자동으로 유도되지 않는다.
- **`PNC_ParkAssist`에는 웨이크업 프레임이 없다.** HPC의 FunctionGroup 상태가 이 PNC를 요청한다(AP 머신 — NM 핸들과 FunctionGroup).
- **`PNC_Trailer`는 `RearCAN_NmFrame_TrailerModule`로 깨어난다.** `0x512`는 웨이크업 필터 블록 `0x500`..`0x5FF` 안에 있다.

확인하기

- [ ] `PNC_ParkAssist`에 `VFC_ParkAssist`(`4 ports`), 그룹 2개(**RX** `1 PDU`, **TX** `0 PDUs`), 서비스 인스턴스 2개가 있다.
- [ ] **PHYSICAL CHANNELS**에서 `RearCAN_Channel`과 `VLAN_Rear (VLAN 200)`만 체크되어 있다.
- [ ] `PNC_Trailer`의 **WAKEUP FRAMES**에 `RearCAN_NmFrame_TrailerModule`이 있다.
- [ ] `PNC_Comfort`와 `PNC_Trailer`가 나머지 PNC의 값의 표와 같다.
- [ ] 다이얼로그 제목 앞에 **●**가 없다.

다음 단계

PDU GROUPS·MATRIX·NM VECTOR 검토에서 뷰어의 세 탭으로 설계를 교차 검토한다.

PDU GROUPS·MATRIX·NM VECTOR 검토

이 장에서 할 일

이 장에서는 PNC 다이얼로그의 뷰어 탭 **PDU GROUPS**, **MATRIX**, **NM VECTOR** 세 개를 열어 PNC 설계를 교차 검토한다. 보기만 하며, 값을 바꾸거나 저장하지 않는다.

시작하기 전에

- PNC 다이얼로그 — PNC별 바인딩을 마치고, 모든 PNC가 나머지 PNC의 값대로 바인딩된 상태에서 시작한다.
- NM 값은 NM 설정에서 왔다. 값이 다르면 전체 NM 값과 맞춰 본다.

항목	값
PNC	`PNC_Comfort` 17, `PNC_ParkAssist` 18, `PNC_Trailer` 19
PNC VECTOR OFFSET / PNC VECTOR LENGTH	`2` / `6` (NM PDU의 Byte ~~27, 비트 4663~~)
NM 클러스터	`NmCluster_EastCAN`, `NmCluster_RearCAN`, `NmCluster_WestCAN`(CanNm), `NmCluster_VLAN_Body`, `NmCluster_VLAN_Rear`(UdpNm

뷰어 탭 열기

1. 앞 장처럼 System 편집기(탭 `PNC:System`)에서 PNC 다이얼로그를 연다.
2. **PNC 19 PNC_Trailer**를 누른다.

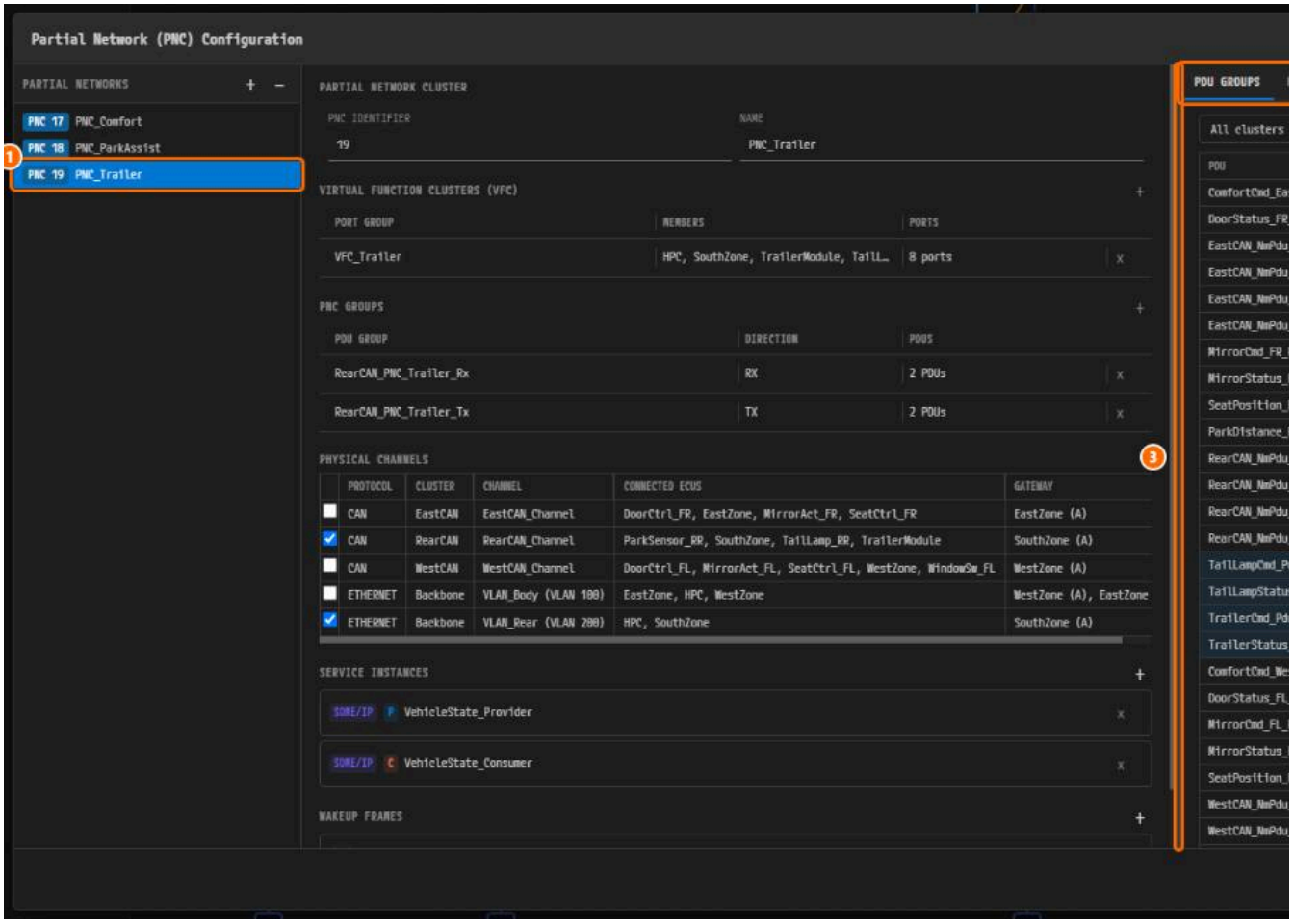


*그림 8-1. ① **PNC 19 PNC_Trailer**, ② 뷰어 탭, ③ 경계선.*

3. 경계선을 왼쪽으로 약 340px 끌어 **PDU**, **FRAME**, **CLUSTER**, **PNC MEMBERSHIP**이 모두 보이게 한다.

PDU GROUPS 읽기

한 행이 PDU 하나이다. **PNC MEMBERSHIP**은 그 PDU를 담은 I-PDU Group이 바인딩된 PNC를 보여 준다.

1. `PNC_Trailer`의 4행이 강조되고 **PNC MEMBERSHIP**이 `PNC_Trailer`인지 확인한다.

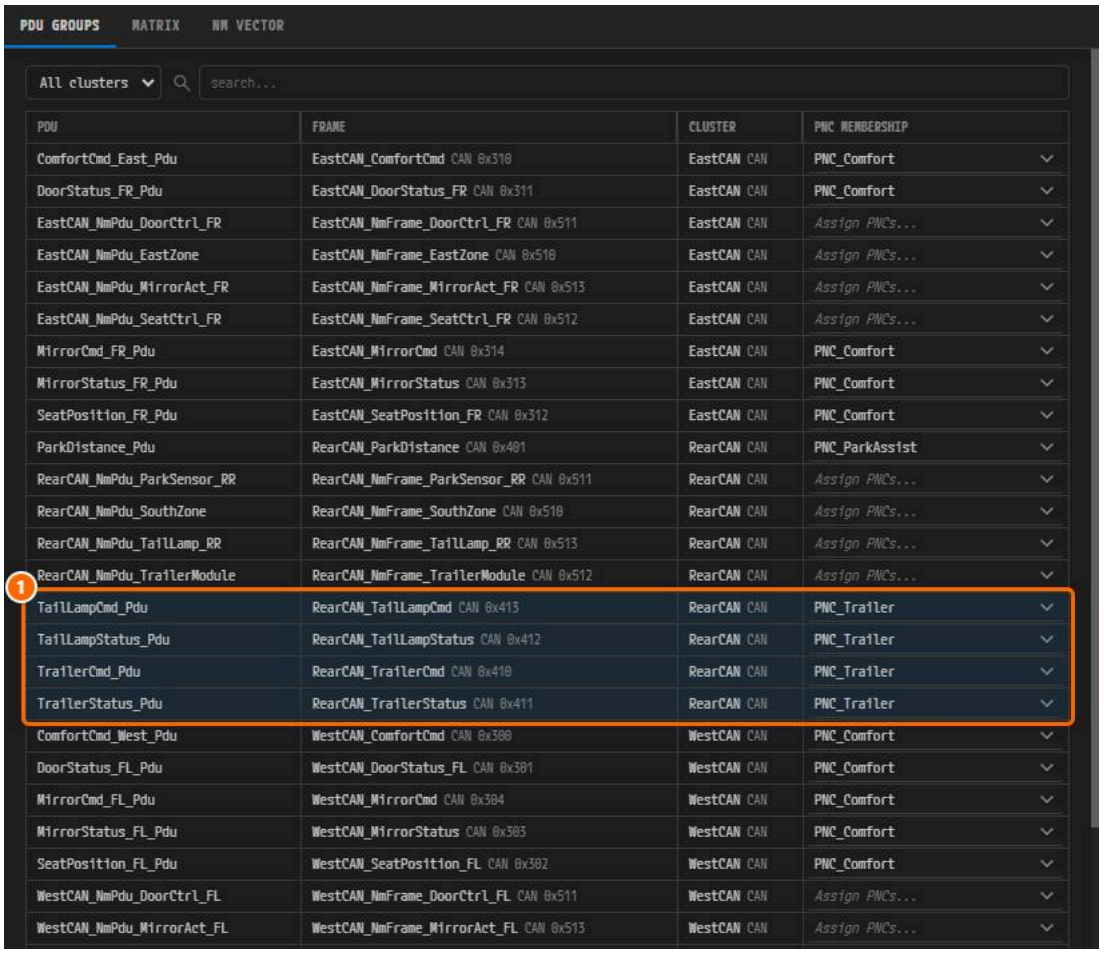


그림 8-2. ① `PNC_Trailer`의 4행.

NM PDU(`<클러스터>_NmPdu_<ECU>`)는 "Assign PNCs..."로 보인다. NM PDU는 어느 PNC의 그룹에도 없으므로 정상이다.

> **참고.** 직접 만든 프로젝트에서는 행이 더 적을 수 있다. 부록 — 직접 만든 프로젝트와 완성 예제의 차이를 참고한다.

클러스터로 좁히기

1. 클러스터 필터(**All clusters**)를 **RearCAN**으로 고른다. 9행이 남는다.

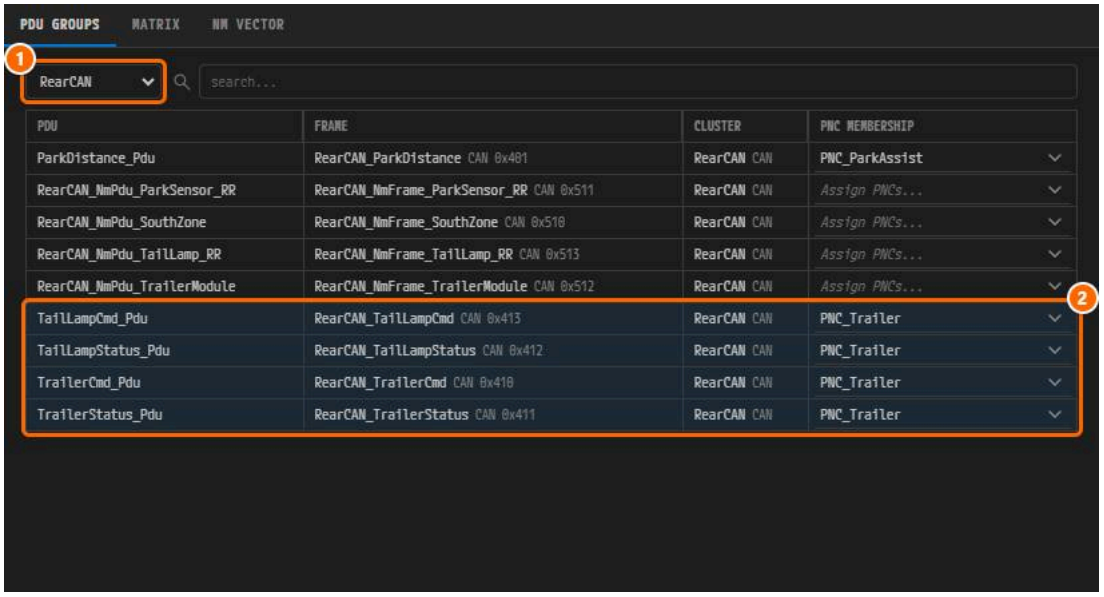


*그림 8-3. ① **RearCAN**, ② `PNC_Trailer`의 4행.*

2. 필터를 **All clusters**로 되돌린다.

이름으로 찾기

1. 검색칸에 `Trailer`를 입력한다. 3행이 남는다.

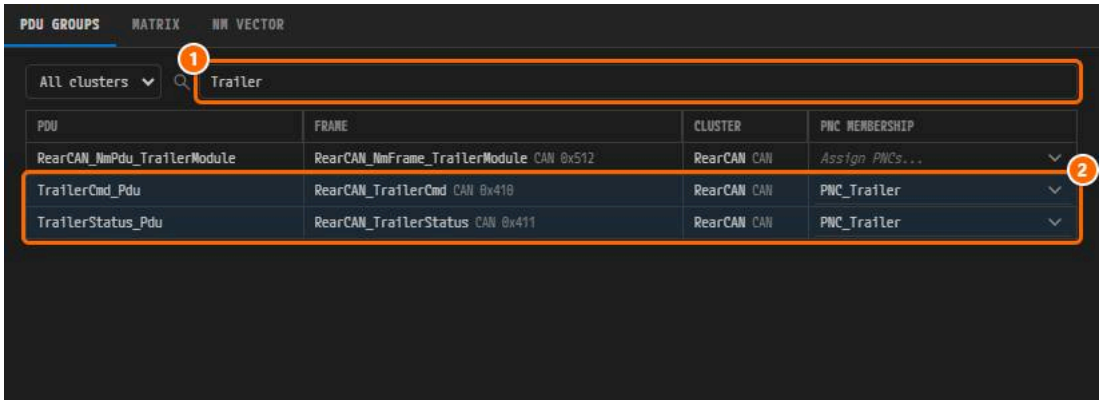


그림 8-4. ① `Trailer`, ② `TrailerCmd_Pdu`와 `TrailerStatus_Pdu`.

`PNC_Trailer`의 `TailLamp*` PDU는 이름에 `Trailer`가 없어서 나오지 않는다.

2. 검색칸을 비운다.

MATRIX 읽기

행은 PNC, 열은 물리 채널이다.

1. 경계선을 오른쪽으로 되돌리고 **MATRIX** 탭을 누른다.

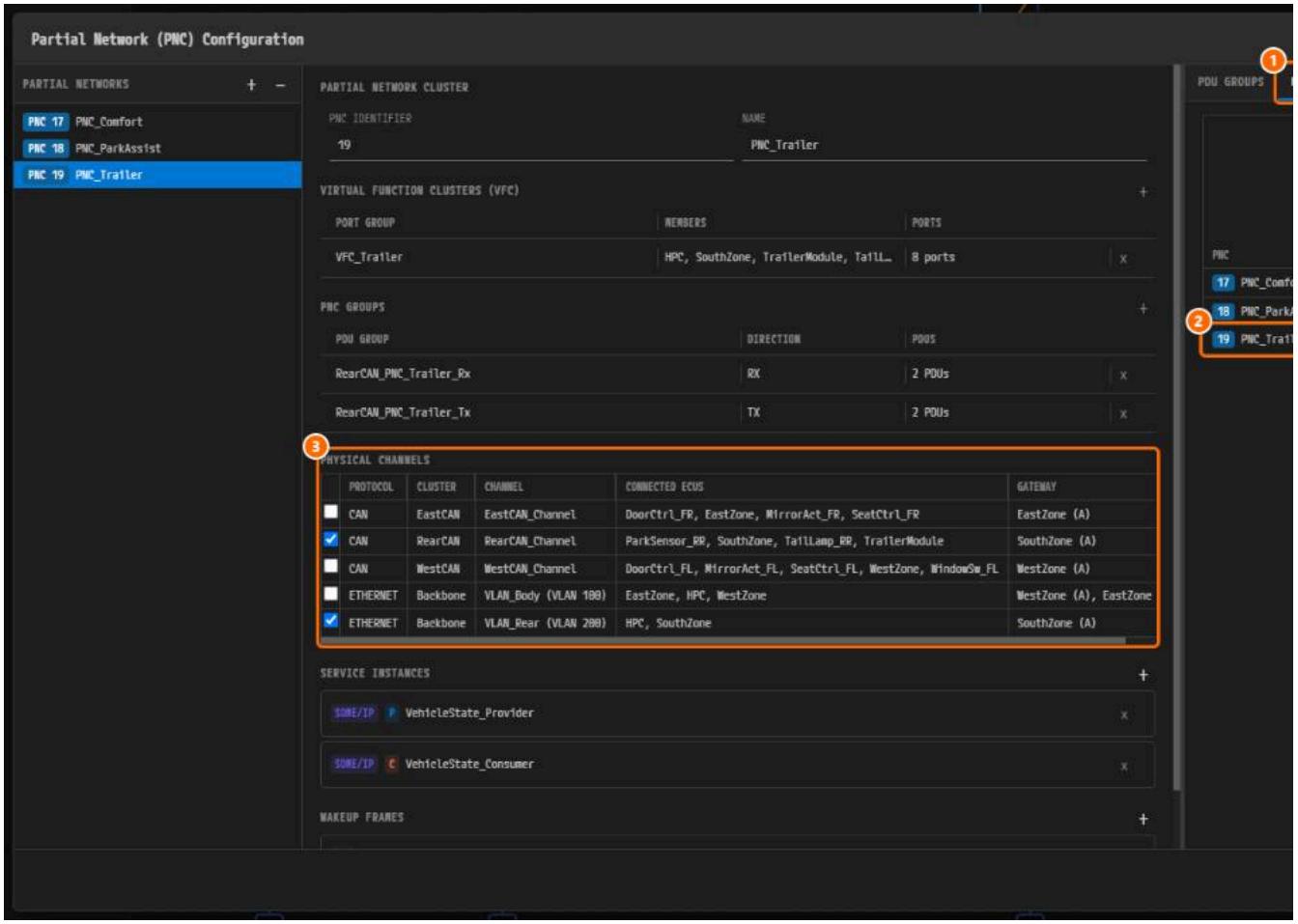


*그림 8-5. ① **MATRIX**, ② `PNC_Trailer` 행, ③ ②와 같은 값을 보여 주는 **PHYSICAL CHANNELS**.*

2. 매트릭스를 표와 맞춰 본다.



그림 8-6. ① `PNC_Comfort`, ② `PNC_ParkAssist`와 `PNC_Trailer`.

PNC	EastCAN / EastCAN_Channel	RearCAN / RearCAN_Channel	WestCAN / WestCAN_Channel	Backbone / VLAN_Body	Backbone / VLAN_R
17 `PNC_Comfort`	✓		✓	✓	
18 `PNC_ParkAssist`		✓			✓
19 `PNC_Trailer`		✓			✓

> **주의.** 매트릭스 칸을 클릭하면 물리 채널 참조가 바로 바뀐다. 클릭하지 않는다. 클릭했다면 같은 칸을 다시 클릭하거나 **Ctrl+Z**를 누른다.

NM VECTOR 읽기

PNC 식별자 N은 NM PDU의 Byte `N ÷ 8의 몫`, bit `N ÷ 8의 나머지`에 놓인다.

1. **PNC 17 PNC_Comfort**를 누르고 뷰어를 다시 넓힌 뒤 **NM VECTOR** 탭을 누른다.

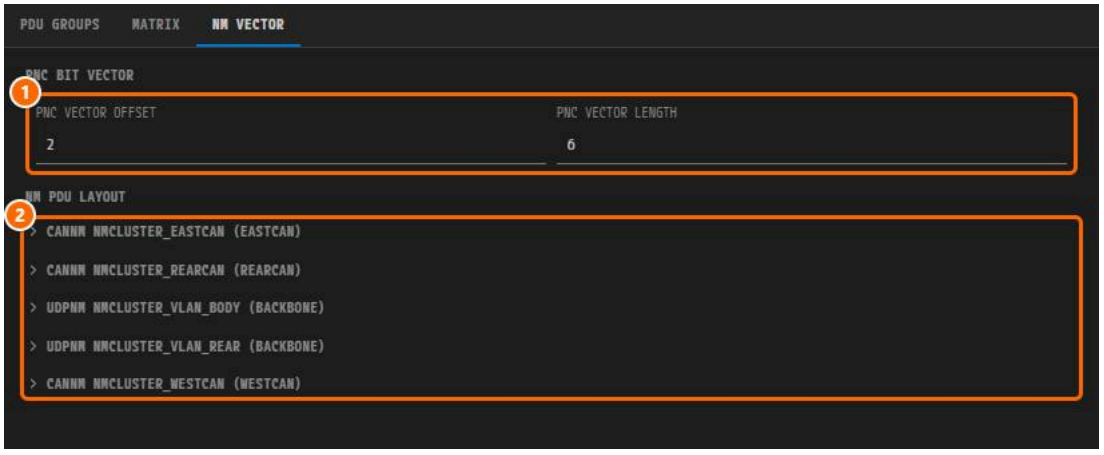


*그림 8-7. ① **PNC VECTOR OFFSET** `2`, **PNC VECTOR LENGTH** `6`, ② NM 클러스터 머리글 5개.*

두 값은 모든 NM 클러스터에 적용된다. PNC 비트 벡터 설정하기에서 입력했다.

2. 맨 아래 머리글 **CANNM NMCLUSTER_WESTCAN (WESTCAN)**을 눌러 펼친다.

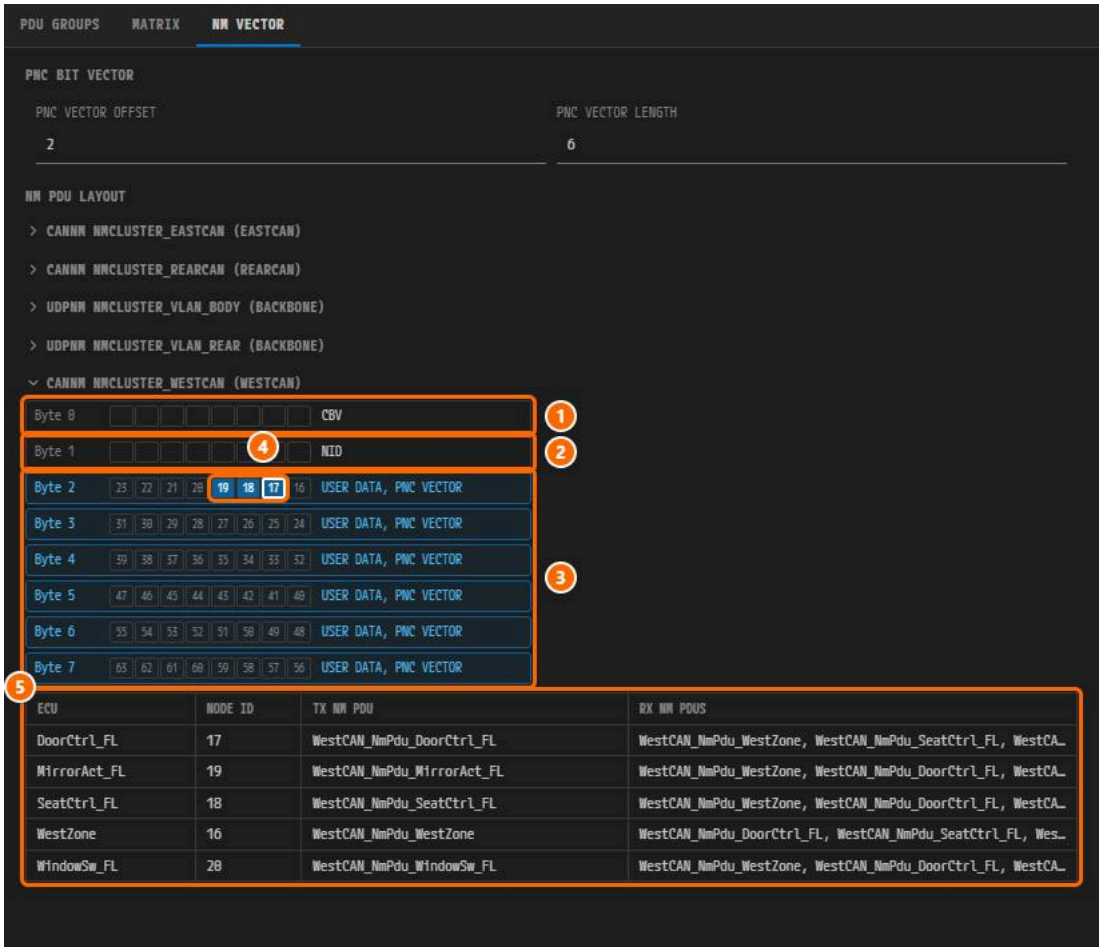


*그림 8-8. ① Byte 0 **CBV**, ② Byte 1 **NID**, ③ Byte 2~7 **USER DATA, PNC VECTOR**, ④ 채워진 칸 `19`, `18`, `17`. 선택한 PNC 17에는 흰 테두리가 있다, ⑤ NM 노드 표.*

3. `NmCluster_WestCAN`의 노드 표를 확인한다.

ECU	NODE ID	TX NM PDU
`DoorCtrl_FL`	17	`WestCAN_NmPdu_DoorCtrl_FL`
`MirrorAct_FL`	19	`WestCAN_NmPdu_MirrorAct_FL`
`SeatCtrl_FL`	18	`WestCAN_NmPdu_SeatCtrl_FL`
`WestZone`	16	`WestCAN_NmPdu_WestZone`
`WindowSw_FL`	20	`WestCAN_NmPdu_WindowSw_FL`

4. 다른 NM 클러스터도 같은 방법으로 펼친다. 바이트 배치는 모두 같다. 표의 ECU는 NODE ID 순이다.

NM 클러스터	ECU (NODE ID)
`NmCluster_EastCAN` (EastCAN)	`EastZone`(16), `DoorCtrl_FR`(17), `SeatCtrl_FR`(18), `MirrorAct_FR`(19)
`NmCluster_RearCAN` (RearCAN)	`SouthZone`(16), `ParkSensor_RR`(17), `TrailerModule`(18), `TailLamp_RR`(19)
`NmCluster_VLAN_Body` (Backbone)	`HPC`(0), `WestZone`(1), `EastZone`(2), `SouthZone`(3), `HPC`(4)
`NmCluster_VLAN_Rear` (Backbone)	`HPC`(0), `SouthZone`(1), `WestZone`(2), `EastZone`(3), `HPC`(4)

> **참고.** 두 UdpNm 클러스터는 `Backbone`의 모든 커넥터마다 노드를 하나씩 보여 준다. 다른 VLAN의 노드는 NM PDU 칸이 비어 있고, `HPC`는 두 번 나온다.

5. **Close**를 누른다. 저장 여부를 묻지 않고 닫힌다.

필터 마스크와 NM VECTOR의 관계

필터 마스크 계산에서 입력한 **PNC FILTER ARRAY MASKS**는 NM PDU의 Byte 2~7만 담는다. 마스크의 Byte 0이 NM PDU의 Byte 2이다.

```
byteIndex = pncId ÷ 8 의 몫 - pncVectorOffset    # 마스크 배열에서의 바이트
bit       = pncId ÷ 8 의 나머지                  # 그 바이트 안의 비트
value     = 2 ^ bit
```

PNC	NM PDU 위치(툴팁)	마스크 배열 위치	값
17 `PNC_Comfort`	byte 2, bit 1	Byte 0, bit 1 (17 ÷ 8 = 2, 2 − 2 = 0)	`0x02`
18 `PNC_ParkAssist`	byte 2, bit 2	Byte 0, bit 2	`0x04`
19 `PNC_Trailer`	byte 2, bit 3	Byte 0, bit 3	`0x08`

- PNC 17만 통과시키는 커넥터(`WestZone_CanFd_Conn`)의 마스크는 `02 00 00 00 00 00`이다.
- PNC 18과 19를 통과시키는 커넥터(`SouthZone_CanFd_Conn`)는 OR로 합친다: `0x04` | `0x08` = `0x0C`, 즉 `0C 00 00 00 00 00`.
- 마스크 길이는 **PNC VECTOR LENGTH** 6과 같아야 한다(`constr_3686`).

왜 이렇게 설계하는가

- **한 설계를 세 방향에서 본다.** PDU→PNC, PNC→채널, PNC→NM PDU 비트로 보므로 한 탭에서 빠진 것이 다른 탭에서 드러난다.
- **PNC 식별자는 비트 번호이다.** 식별자가 NM PDU 안의 위치와 필터 마스크 값을 정한다.

확인하기

- [ ] **PDU GROUPS**에서 `PNC_Trailer`를 고르면 `TailLampCmd_Pdu`, `TailLampStatus_Pdu`, `TrailerCmd_Pdu`, `TrailerStatus_Pdu`가 강조된다.
- [ ] **MATRIX**가 MATRIX 읽기의 표와 같다.
- [ ] **NM VECTOR**에서 offset이 `2`, length가 `6`이고 Byte 2의 17, 18, 19 칸이 채워져 있다.
- [ ] **Close**를 눌러도 저장 여부를 묻지 않는다.

다음 단계

AP 머신 — NM 핸들과 FunctionGroup에서 HPC 머신에 NM 핸들과 FunctionGroup을 설정한다.

AP 머신 — NM 핸들과 FunctionGroup

이 장에서 할 일

HPC는 PNC 게이트웨이가 아니라 AP 머신이다. HPC는 NM 네트워크 핸들로 PNC를 요청하고 해제한다. 이 장에서는 HPC의 **Machine Configuration** 다이얼로그에서 PNC마다 Fu
들어 FunctionGroup 상태에 매핑한다.

시작하기 전에

- PDU GROUPS·MATRIX·NM VECTOR 검토를 마친 뒤 시작한다.
- `ParkAssistFG`와 `NH_ParkAssist`를 단계별로 만들고, 나머지는 표를 보고 만든다.
- `MachineFG`는 PAIO가 직접 추가한다. 만들거나 지우지 않는다.

항목	값
FunctionGroup **NAME**	`ParkAssistFG`
FunctionGroup **DESCRIPTION**	`Park-assist function group driven by PNC_ParkAssist`
FunctionGroup **GROUP NAME** / **INITIAL MODE**	`ParkAssistState` / `Off`
FunctionGroup **MODE DECLARATION**	`Off`, `Active`
MAX ACTION LIST DURATION	`1` s
핸들 **NAME**	`NH_ParkAssist`
핸들 **VLANS (ETHERNET CONNECTORS)**	`HPC_Eth_Rear_Conn`
핸들 **PARTIAL NETWORKS**	`PNC_ParkAssist (PNC 18)`
매핑 1	`ParkAssistFG:Active` → `FUNCTION-GROUP-STATE-TO-NM-HANDLE`
매핑 2	`ParkAssistFG:Off` → `NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE`

Machine Configuration 다이얼로그 열기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **VFB**를 누른다. 탭 **PNC:VFB**에 루트 컴포지션 `PncVehicle`이 보인다.

2. **HPC [HPC]** 헤더 오른쪽의 하늘색 머신 아이콘을 누른다. 맨 오른쪽의 네모 아이콘은 누르지 않는다.

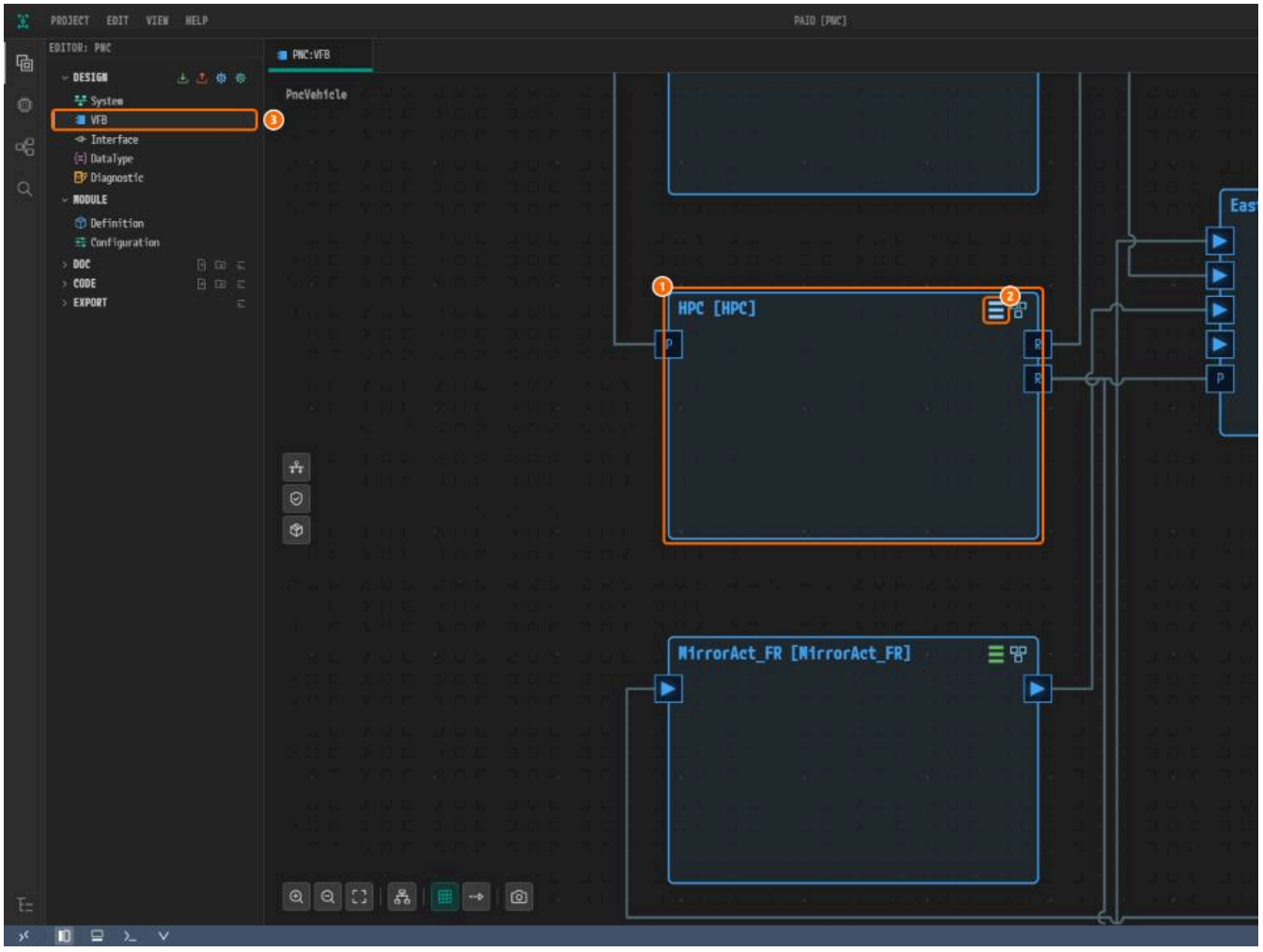


*그림 9-1. ① **HPC [HPC]**, ② 머신 아이콘, ③ **VFB**.*



그림 9-2. ① 머신 아이콘.

3. **Machine Configuration [HPC]** 다이얼로그가 **MACHINE** 탭으로 열린다.

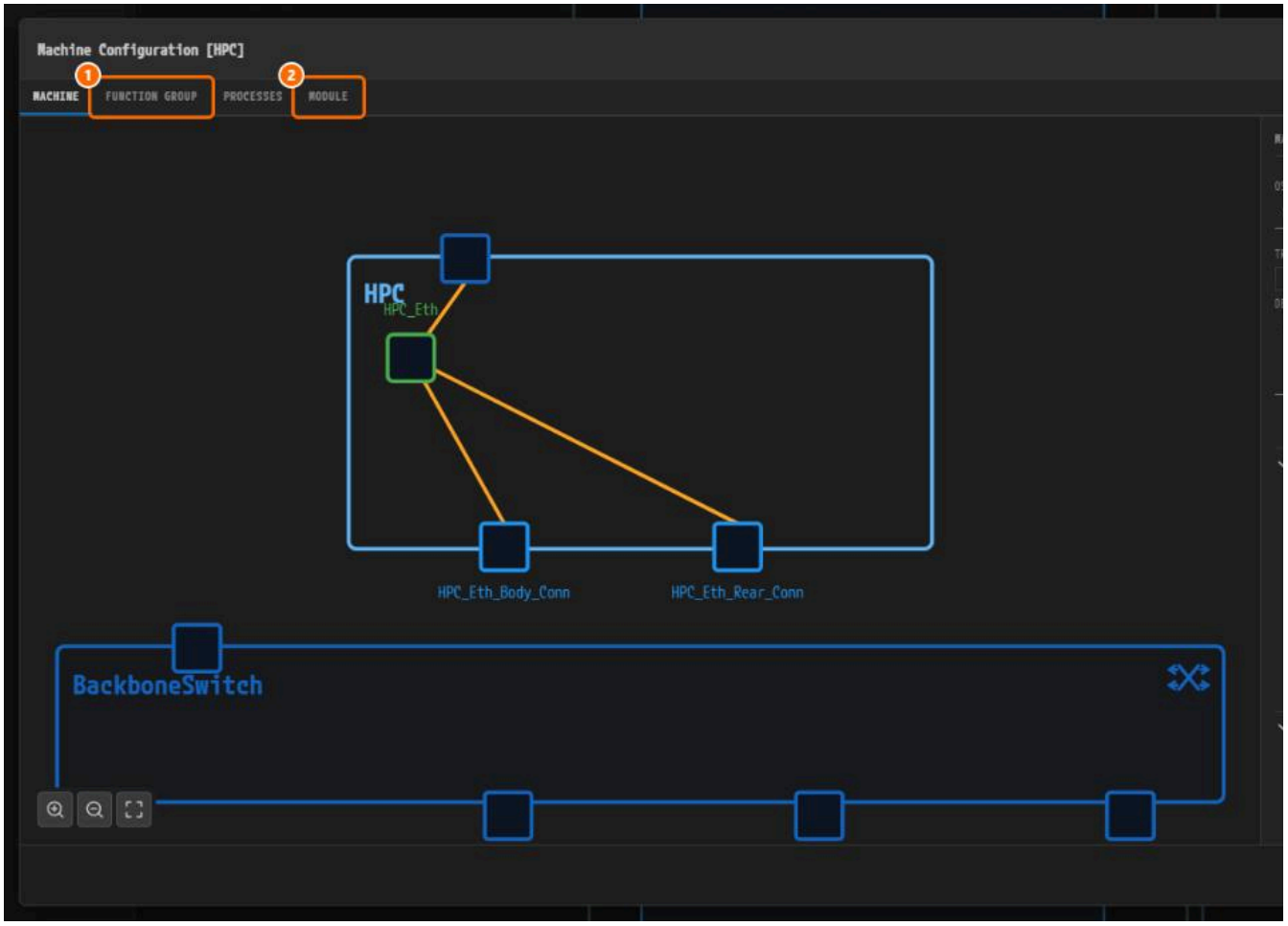


*그림 9-3. ① **FUNCTION GROUP**, ② **MODULE**. 이 장에서는 **PARA Platform Import**를 누르지 않는다.*

FunctionGroup 살펴보기

NM 핸들은 FunctionGroup 상태를 참조하므로 FunctionGroup을 먼저 만든다.

1. **FUNCTION GROUP** 탭을 누르고 `MachineFG`를 누른다. 값은 그대로 둔다.

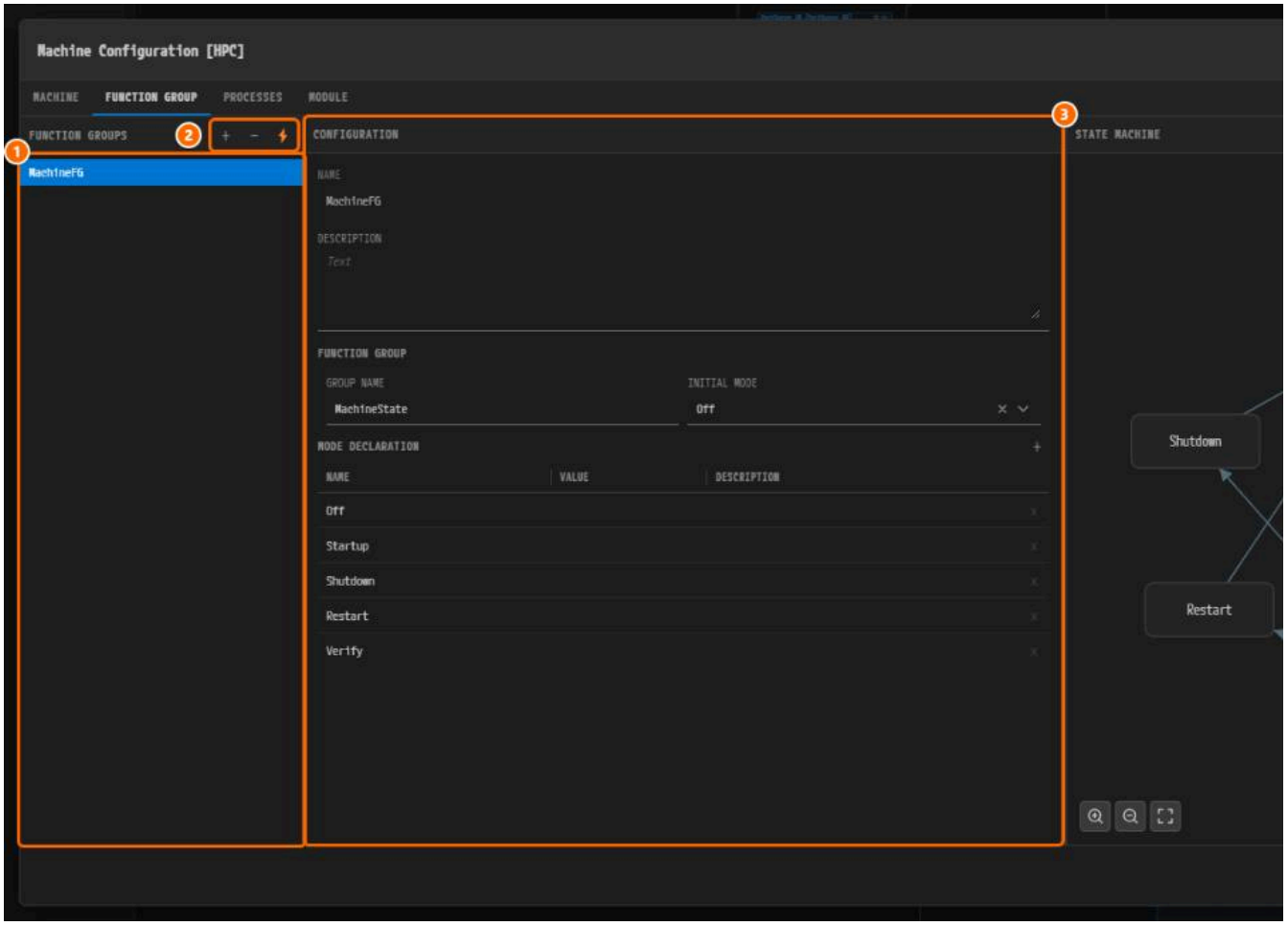


*그림 9-4. ① **FUNCTION GROUPS**, ② **+**, **−**, ⚡, ③ `MachineFG`의 **CONFIGURATION**. 이 장에서는 ** ⚡ **를 누르지 않는다.*

FunctionGroup 만들기

1. **+(Add Function Group)**를 누른다. `FunctionGroup_2`가 추가되고 선택된다.

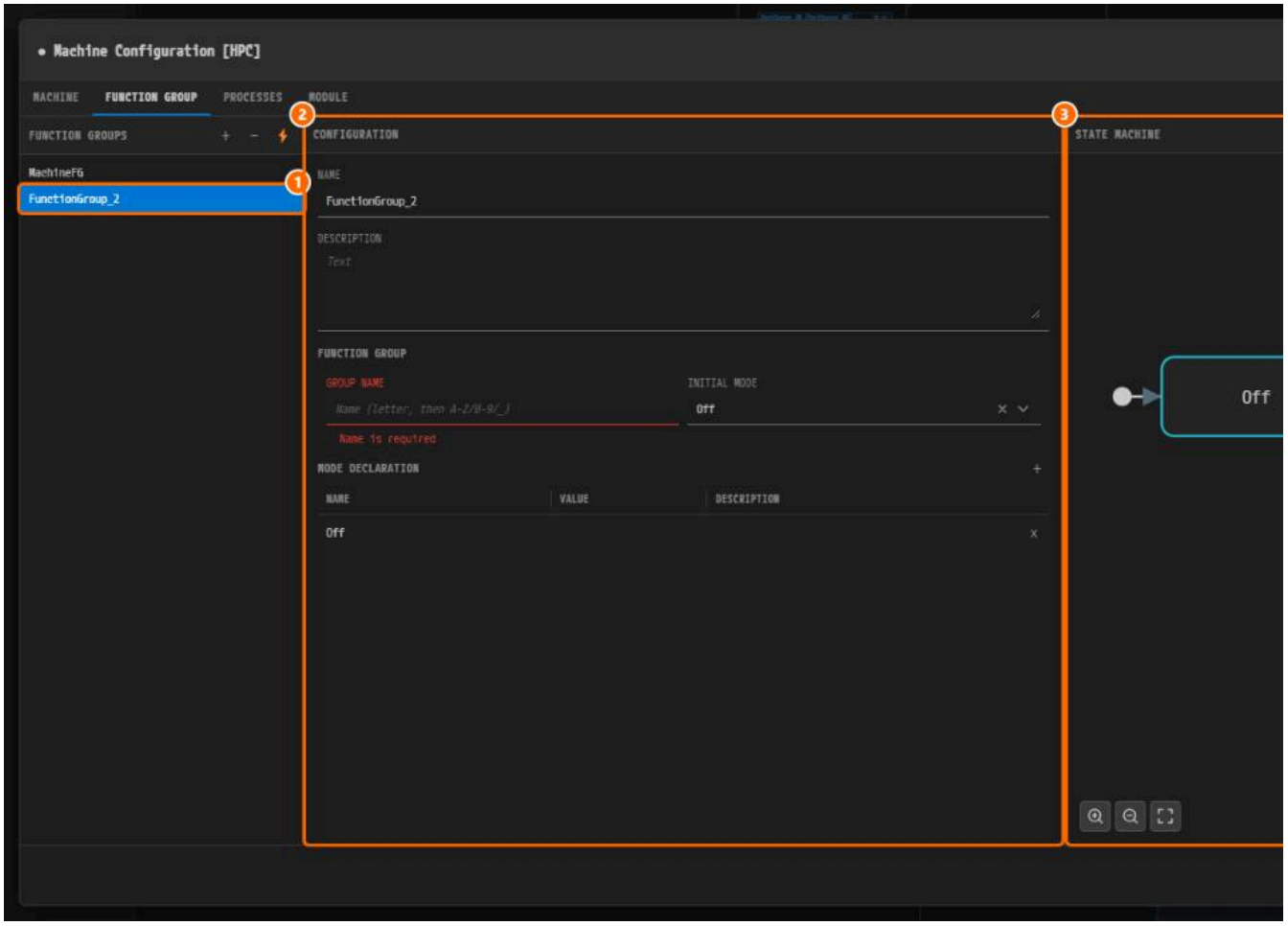


*그림 9-5. ① `FunctionGroup_2`, ② **CONFIGURATION**, ③ **STATE MACHINE**.*

2. **NAME**에 `ParkAssistFG`를 입력하고 `Tab`을 누른다. 목록 라벨은 나중에 바뀐다.

3. **DESCRIPTION**에 `Park-assist function group driven by PNC_ParkAssist`를 입력한다.

4. **GROUP NAME**에 `ParkAssistState`를 입력하고 `Tab`을 누른다.

5. **MODE DECLARATION**에는 `Off`가 이미 있다. **+(Add Mode Declaration)**를 누르고 새 행의 **NAME**을 `Active`로 바꾼 뒤 표 밖을 누른다.

6. **INITIAL MODE**가 `Off`인지 확인한다.

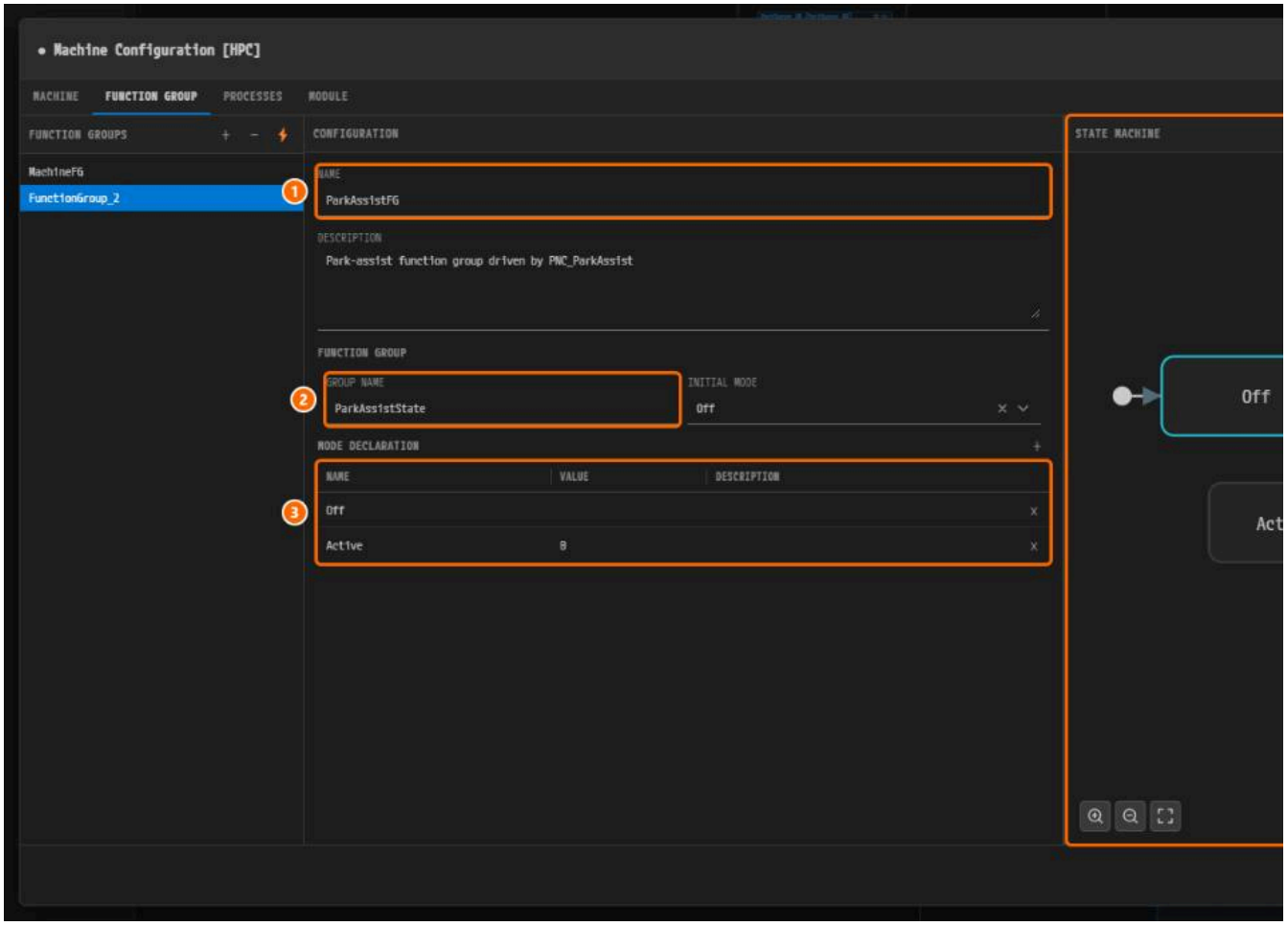


*그림 9-6. ① **NAME**, ② **GROUP NAME**, ③ `Off`와 `Active`, ④ 전이가 없는 **STATE MACHINE**.*

7. **STATE MACHINE**에서 `Off` 상자를 오른쪽 클릭하고 **Transition to** > **Active**를 누른다. `Active` 상자에서는 **Transition to** > **Off**를 누른다.



*그림 9-7. ① **Transition to**, ② **Active**.*

8. 목록에서 `FunctionGroup_2`를 누른다. 라벨이 `ParkAssistFG`로 바뀐다.

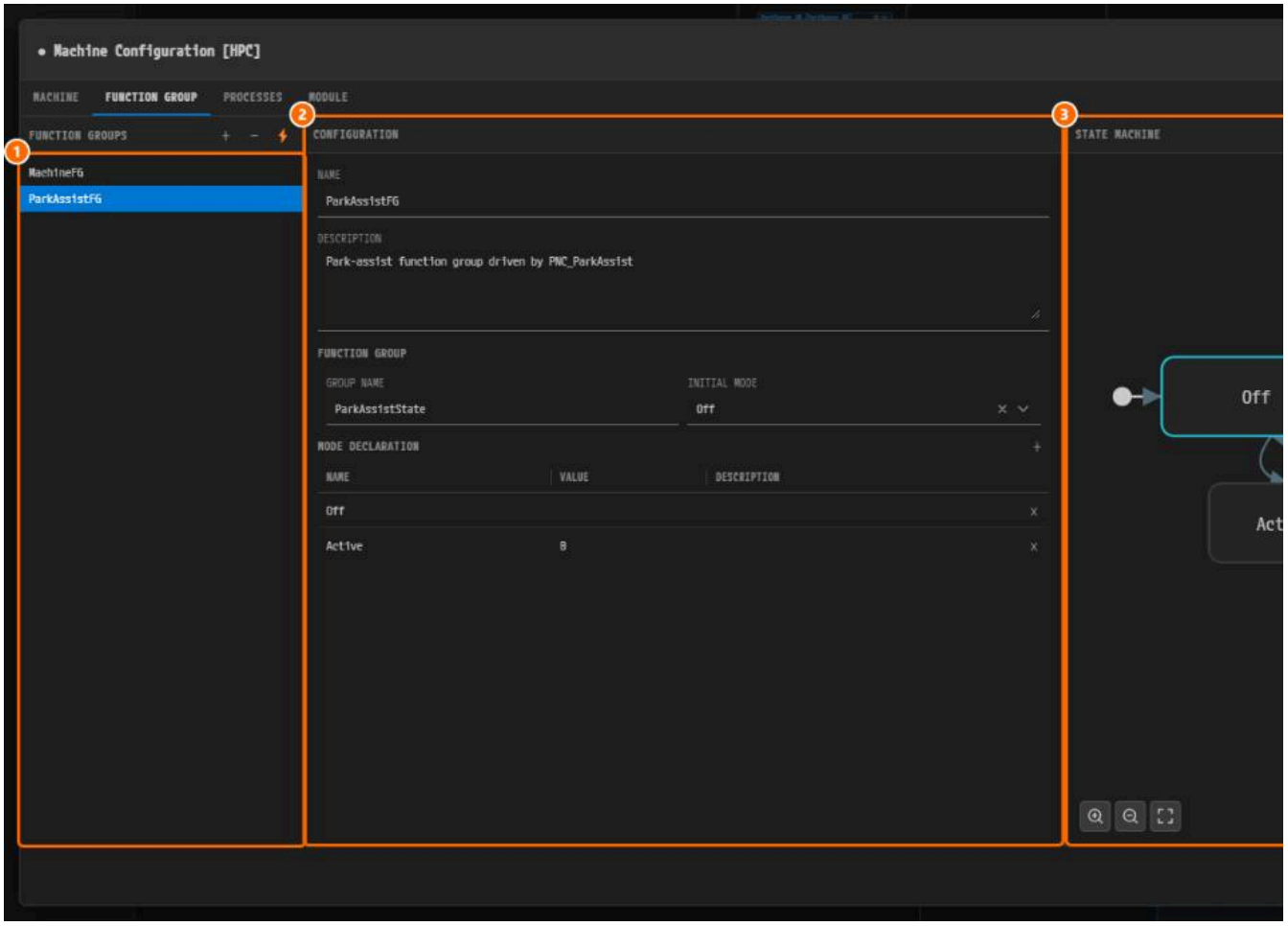


*그림 9-8. ① **FUNCTION GROUPS**, ② **CONFIGURATION**, ③ 전이 2개.*

9. `Ctrl+S`로 저장한다.

> **참고.** 새 모드 행에는 **VALUE** `0`이 들어가 `Active (0)`으로 보인다. 예제 모델에는 이 값이 없다. 결과에 영향이 없으므로 그대로 둔다.

나머지 FunctionGroup 만들기

`TrailerFG`와 `ComfortFG`를 같은 방법으로 만들고, 전이 `Off` → `Active`, `Active` → `Off`를 넣는다.

NAME	DESCRIPTION	GROUP NAME	INITIAL MODE	MODE DECLARATION	
`MachineFG`	(비어 있음)	`MachineState`	`Off`	`Off`, `Startup`, `Shutdown`, `Restart`, `Verify`	자
`ParkAssistFG`	`Park-assist function group driven by PNC_ParkAssist`	`ParkAssistState`	`Off`	`Off`, `Active`	0
`TrailerFG`	`Trailer function group driven by PNC_Trailer`	`TrailerState`	`Off`	`Off`, `Active`	0
`ComfortFG`	`Comfort function group driven by PNC_Comfort`	`ComfortState`	`Off`	`Off`, `Active`	0

1. FunctionGroup 만들기의 1~9단계로 `TrailerFG`를 만든다.
2. 같은 방법으로 `ComfortFG`를 만든다.
3. `ParkAssistFG`를 누르고 그림 9-9와 비교한다.

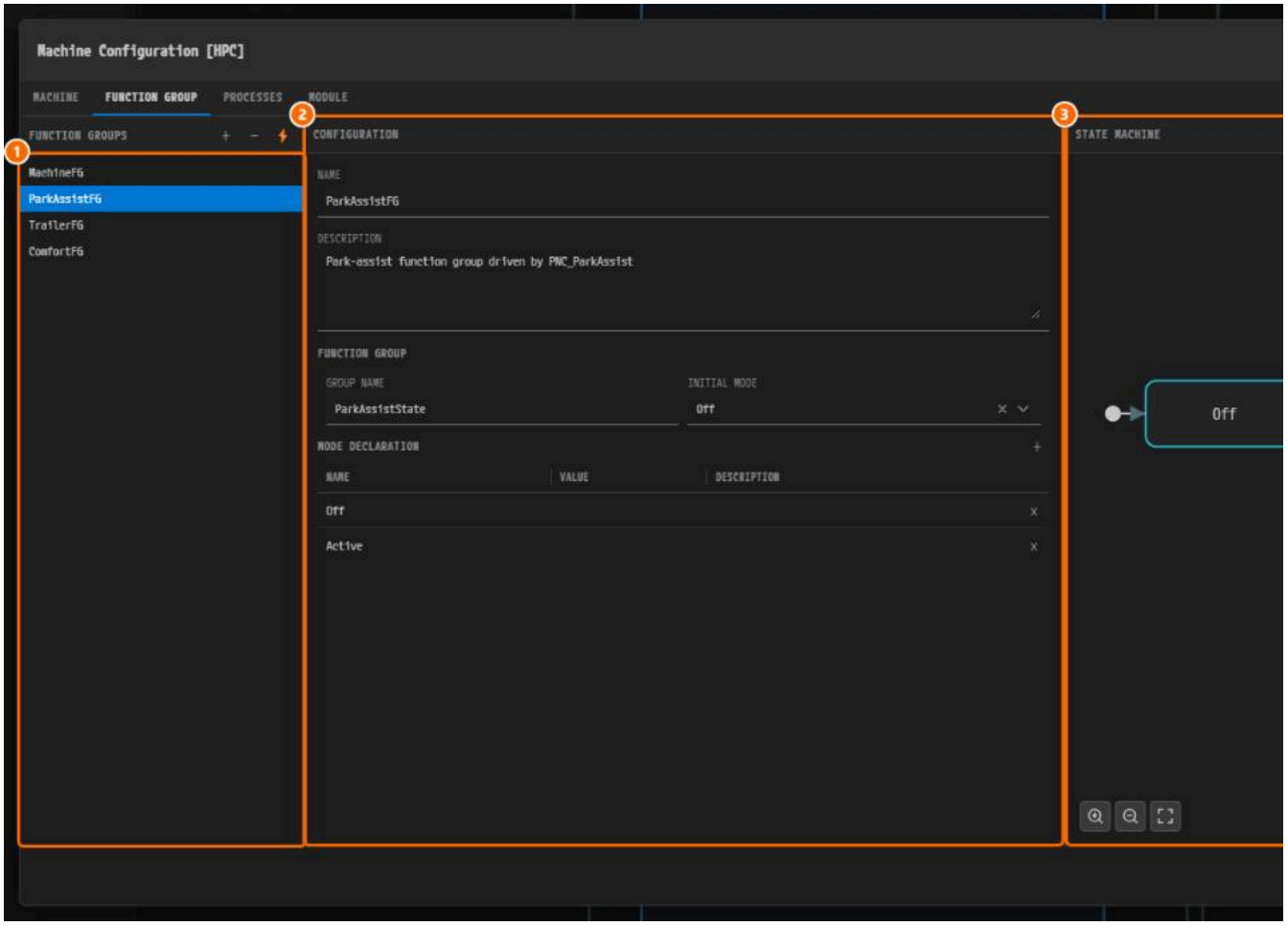


*그림 9-9. ① FunctionGroup 4개, ② `ParkAssistFG`의 **CONFIGURATION**, ③ **STATE MACHINE**.*

이 FunctionGroup들은 뒤의 FunctionGroup 상태 매핑 추가하기에서 `<FunctionGroup>:<상태>` 후보 11개로 나온다. 이름이 표와 다르면 후보에 없다.

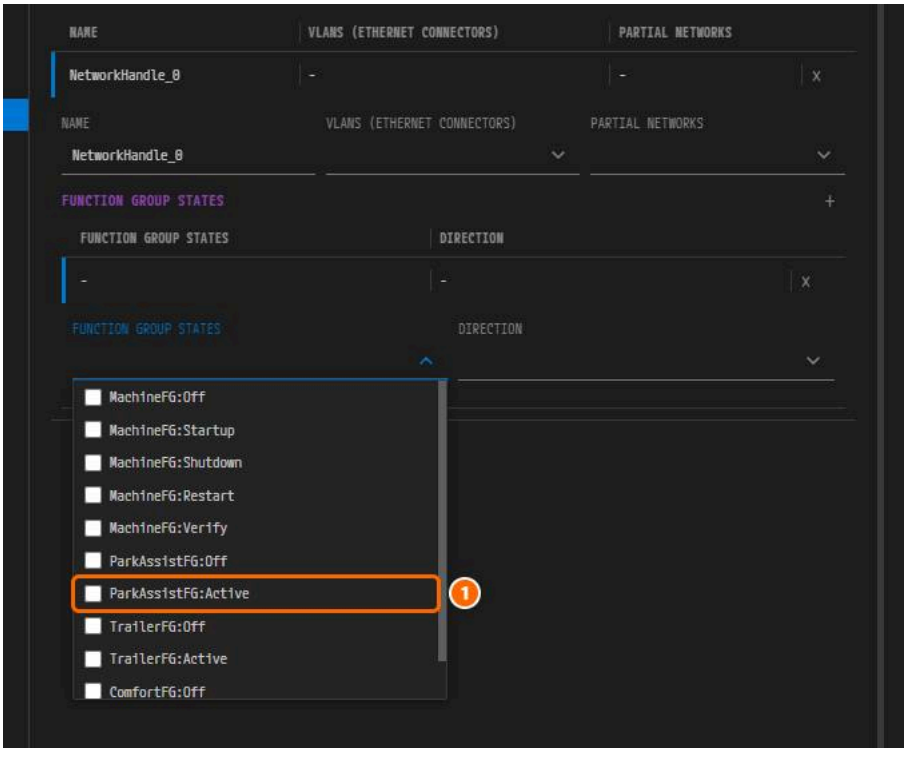


그림 9-10. 뒤 절에서 보는 목록. ① `ParkAssistFG:Active`.

State Management의 MAX ACTION LIST DURATION 입력하기

1. **MODULE** 탭을 누르고 **MODULES** 목록에서 **State Management**를 누른다. 모듈은 모두 활성 상태로 둔다.
2. **ACTION ITEMS** 하위 탭을 누른다.
3. **MAX ACTION LIST DURATION**에 `1`을 입력하고 다른 곳을 누른다. 단위는 `s` 그대로 둔다.

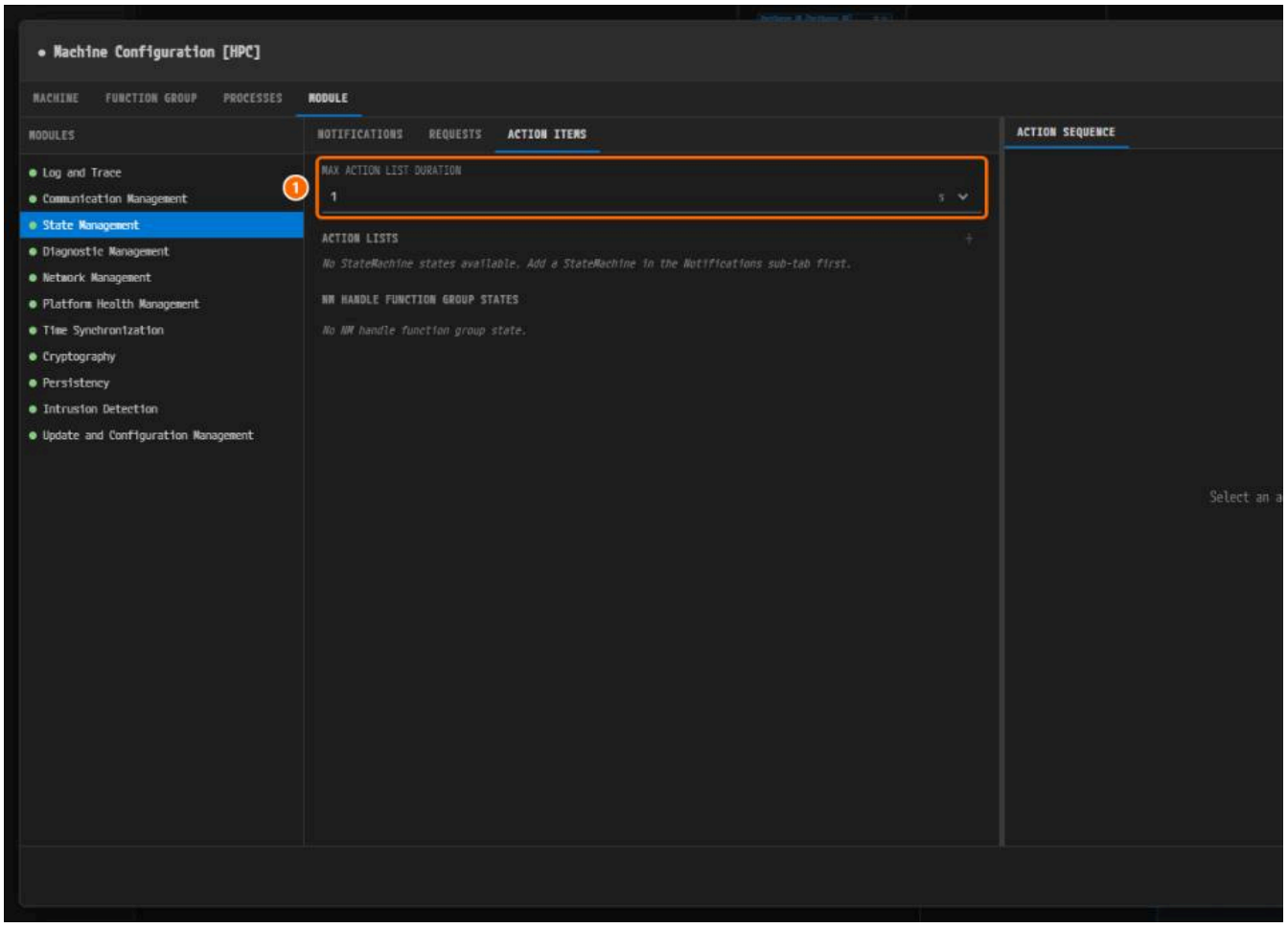


*그림 9-11. ① **MAX ACTION LIST DURATION** `1` `s`.*

4. `Ctrl+S`로 저장한다.

> **주의.** 이 값이 없으면 검증에서 `constr_10641`이 보고된다.

NM 네트워크 핸들 추가하기

1. **MODULES** 목록에서 **Network Management**를 누른다.

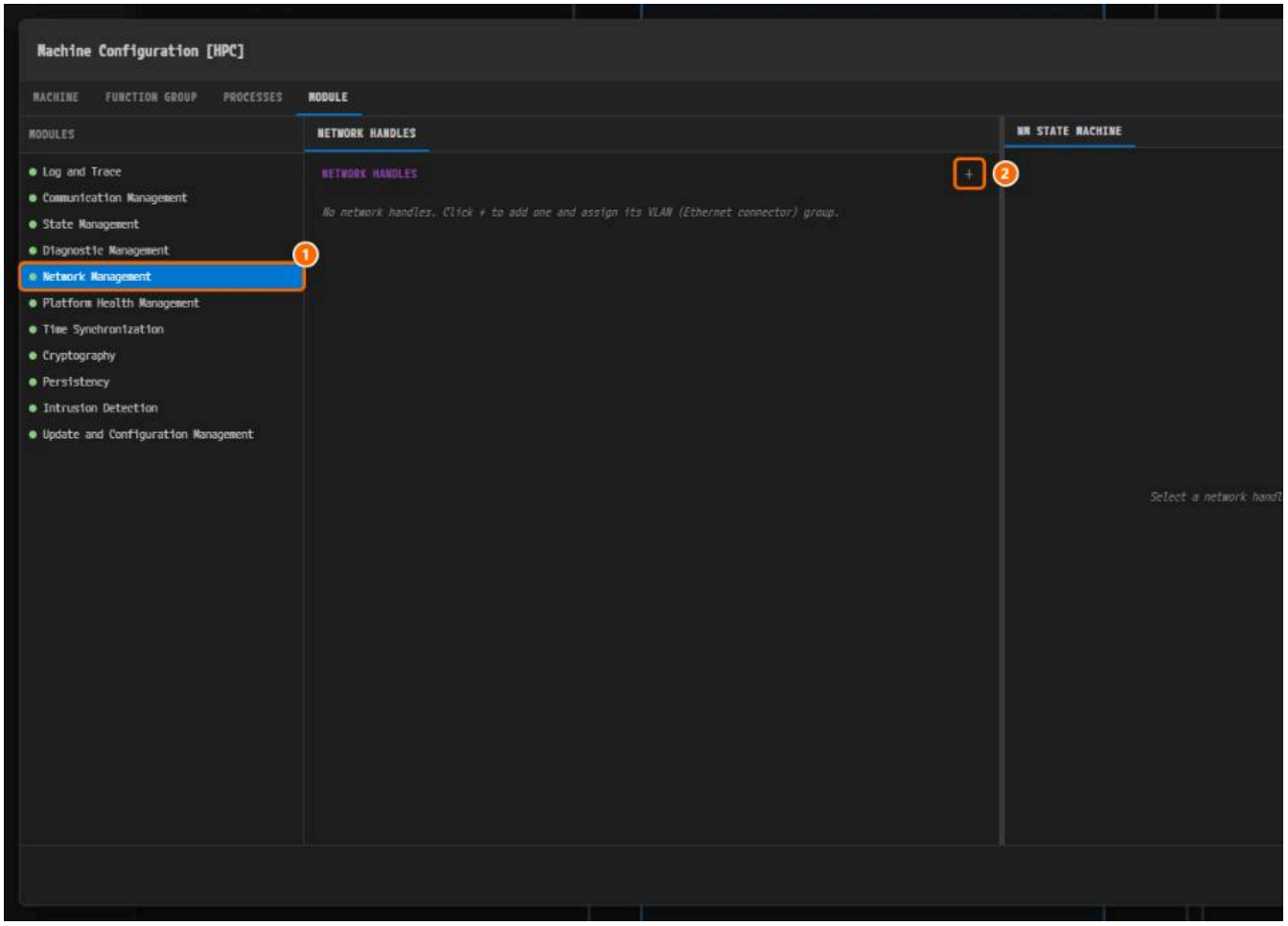


*그림 9-12. ① **Network Management**, ② **+(Add Network Handle)**.*

2. **NETWORK HANDLES** 옆의 **+**를 누른다. `NetworkHandle_0`이 추가되고 펼쳐진다.

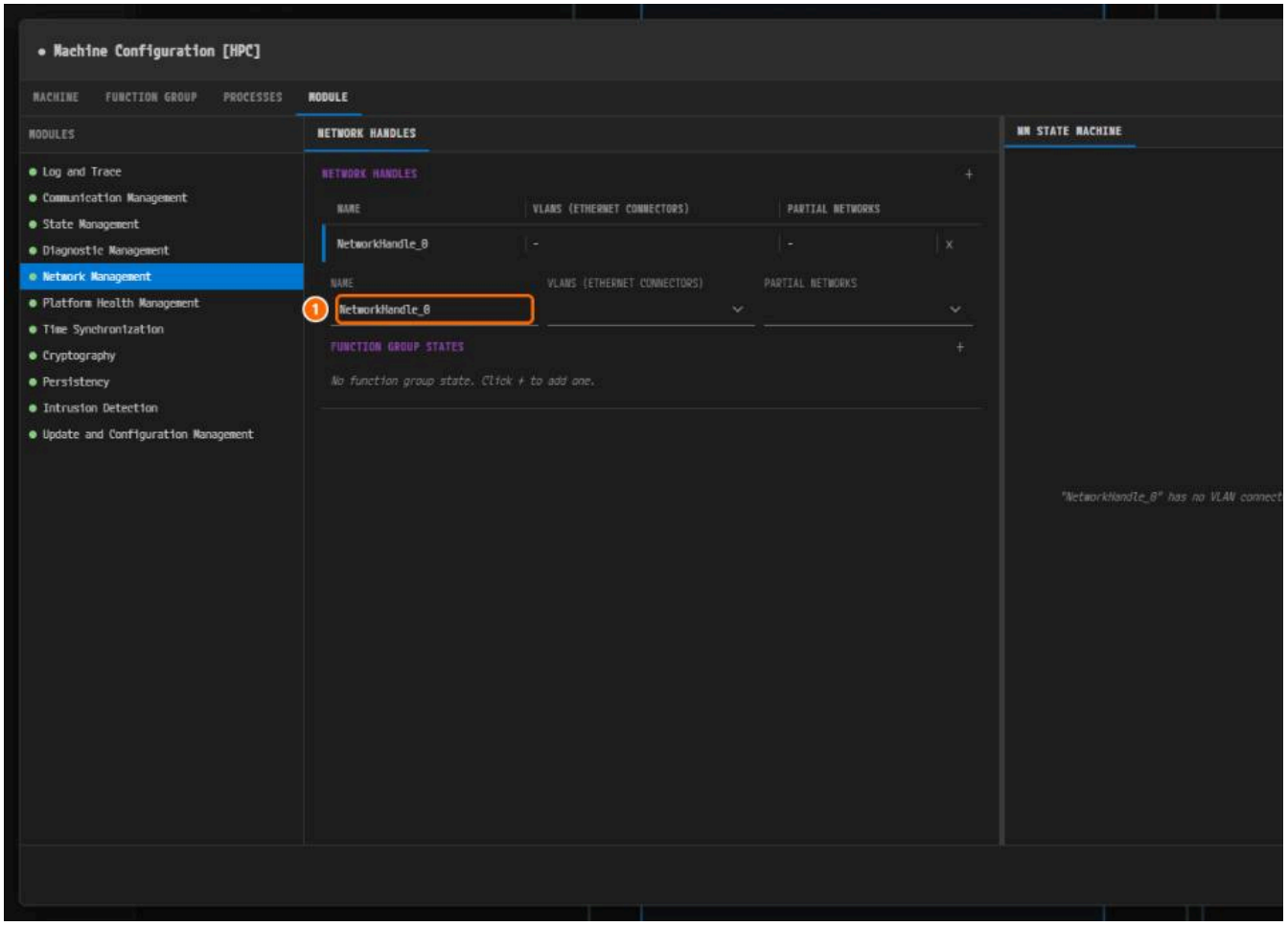


*그림 9-13. ① **NAME.***

3. **NAME**에 `NH_ParkAssist`를 입력하고 `Tab`을 누른다.

4. 요약 행에 `NH_ParkAssist`가 보인다.

5. **VLANS (ETHERNET CONNECTORS)**를 누르고 **HPC_Eth_Rear_Conn**을 체크한 뒤 `Esc`를 누른다.

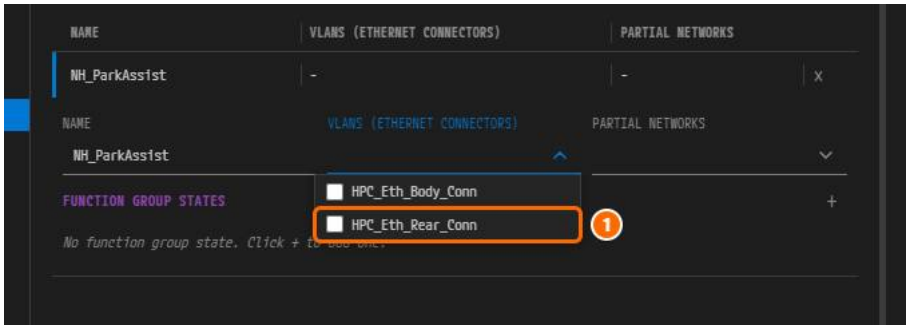


*그림 9-14. ① **HPC_Eth_Rear_Conn.***

6. **PARTIAL NETWORKS**를 누르고 **PNC_ParkAssist (PNC 18)**을 체크한 뒤 `Esc`를 누른다.

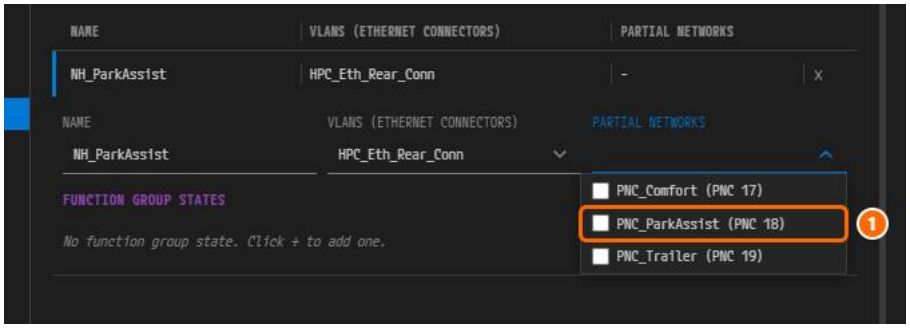


*그림 9-15. ① **PNC_ParkAssist (PNC 18)**.*

FunctionGroup 상태 매핑 추가하기

첫 매핑은 `ParkAssistFG`가 `Active`가 되면 `PNC_ParkAssist`를 요청한다. 두 번째 매핑은 PNC가 비활성이 되면 `ParkAssistFG`를 `Off`로 바꾼다.

1. 핸들 안의 **FUNCTION GROUP STATES** 옆 **+(Add Function Group State)**를 누른다. 매핑 행이 열린다.
2. **FUNCTION GROUP STATES**를 누르고 **ParkAssistFG:Active**를 체크한 뒤 `Esc`를 누른다.

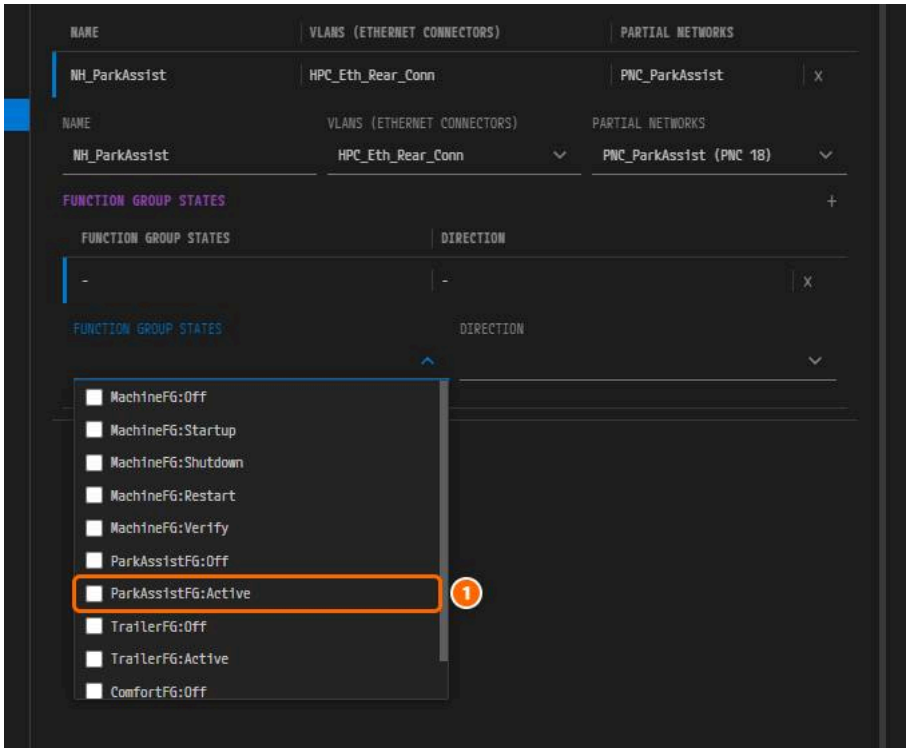


*그림 9-16. ① **ParkAssistFG:Active**.*

3. **DIRECTION**에서 **FUNCTION-GROUP-STATE-TO-NM-HANDLE**을 고른다.

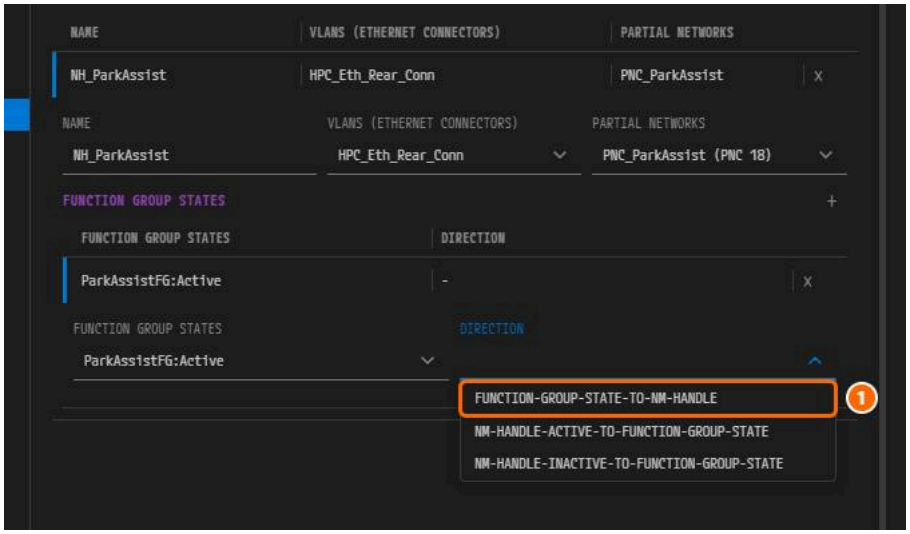


*그림 9-17. ① **FUNCTION-GROUP-STATE-TO-NM-HANDLE.***

4. **FUNCTION GROUP STATES** 옆의 **+**를 다시 누른다.

5. 새 행에서 **ParkAssistFG:Off**와 **NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE**를 고른다.

6. `Ctrl+S`로 저장한다.

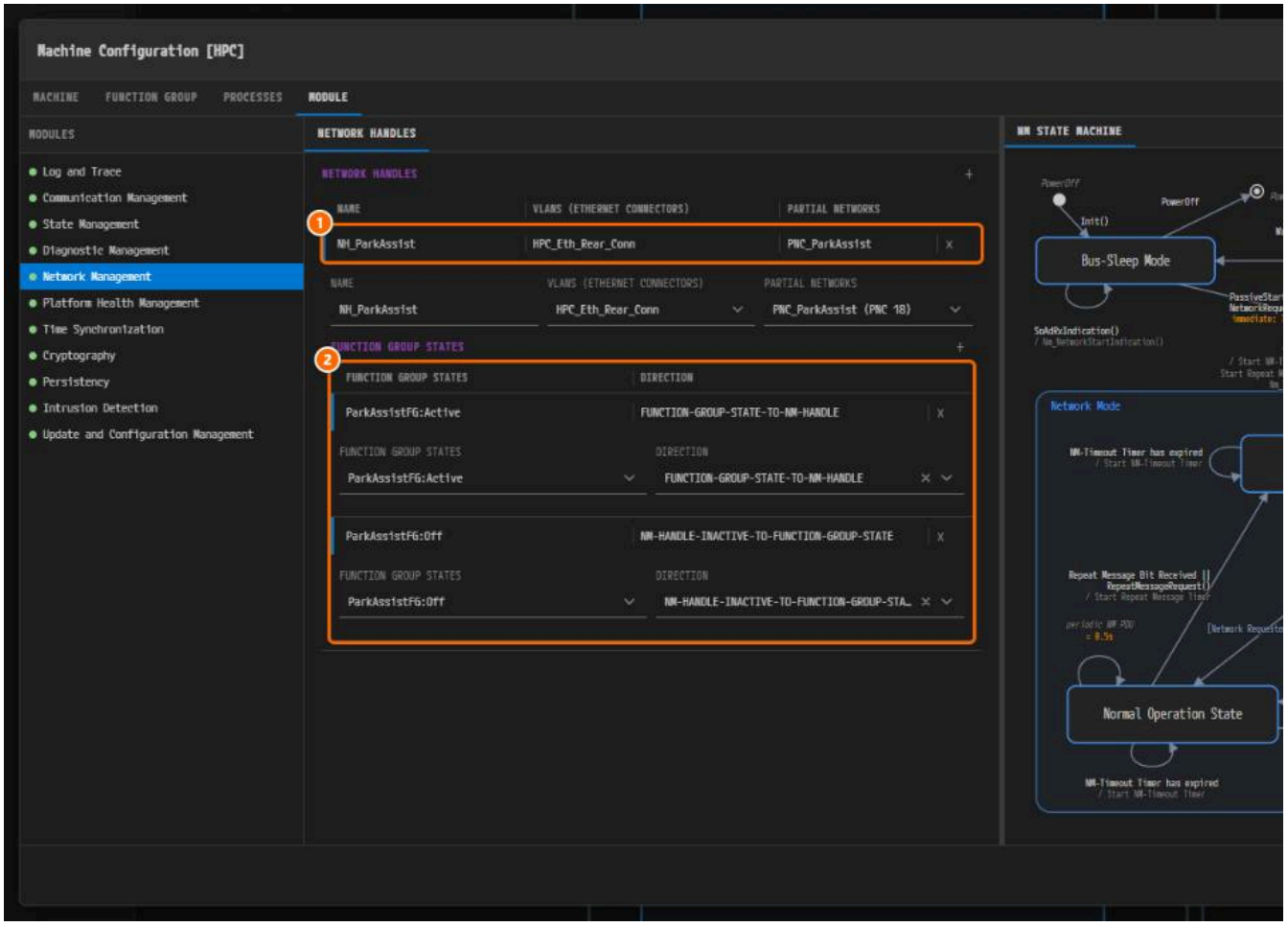


그림 9-18. ① 핸들 행, ② 매핑 2개.

7. **State Management** > **ACTION ITEMS**를 누른다. **NM HANDLE FUNCTION GROUP STATES**에 매핑 2개가 보인다.

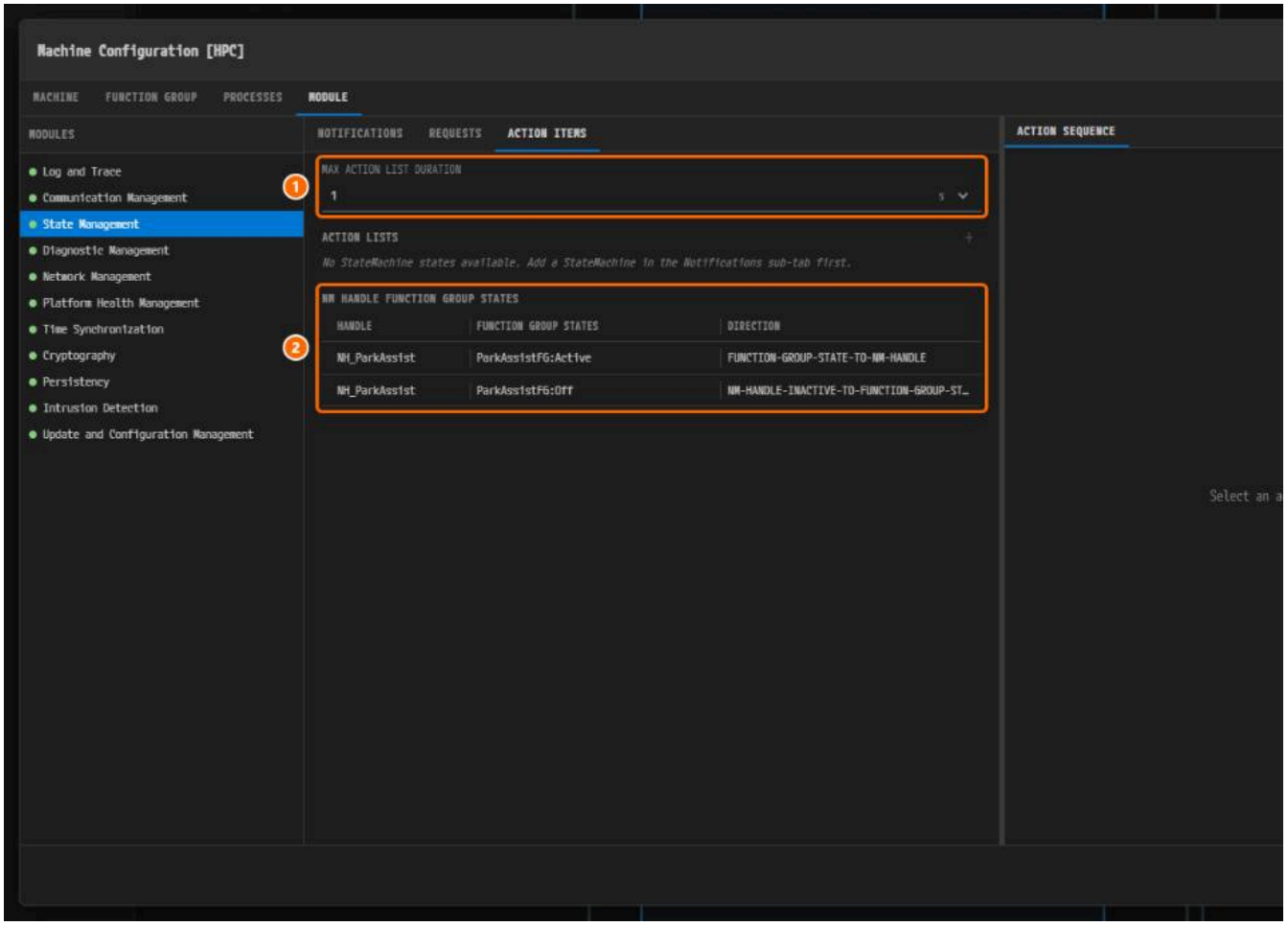


*그림 9-19. ① **MAX ACTION LIST DURATION**, ② **NM HANDLE FUNCTION GROUP STATES**(읽기 전용).*

나머지 핸들 추가하기

`NH_Comfort`와 `NH_Trailer`를 같은 방법으로 만든다.

NAME	VLANS (ETHERNET CONNECTORS)	PARTIAL NETWORKS	FUNCTION GROUP STATES	DIRECTION
`NH_Comfort`	`HPC_Eth_Body_Conn`	`PNC_Comfort (PNC 17)`	`ComfortFG:Active`	`NM-HANDLE-ACTIVE-TO-FUNCTION-GROUP-STATE`
			`ComfortFG:Off`	`NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE`
`NH_ParkAssist`	`HPC_Eth_Rear_Conn`	`PNC_ParkAssist (PNC 18)`	`ParkAssistFG:Active`	`FUNCTION-GROUP-STATE-TO-NM-HANDLE`
			`ParkAssistFG:Off`	`NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE`
`NH_Trailer`	`HPC_Eth_Rear_Conn`	`PNC_Trailer (PNC 19)`	`TrailerFG:Active`	`NM-HANDLE-ACTIVE-TO-FUNCTION-GROUP-STATE`
			`TrailerFG:Off`	`NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE`

DIRECTION	의미
`FUNCTION-GROUP-STATE-TO-NM-HANDLE`	FunctionGroup이 이 상태가 되면 핸들이 PNC를 요청한다.
`NM-HANDLE-ACTIVE-TO-FUNCTION-GROUP-STATE`	PNC가 활성이 되면 FunctionGroup이 이 상태로 바뀐다.
`NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE`	PNC가 비활성이 되면 FunctionGroup이 이 상태로 바뀐다.

1. **Network Management**를 다시 누른다.
2. NM 네트워크 핸들 추가하기의 2~~6단계와 FunctionGroup 상태 매핑 추가하기의~~ 4~~5~~단계로 `NH_Comfort`를 만든다.

3. 같은 방법으로 `NH_Trailer`를 만든다.

4. `Ctrl+S`로 저장한다.

5. `NH_Comfort`를 펼쳐 그림 9-20과 비교한다. 행 순서와 펼친 행은 다를 수 있다.

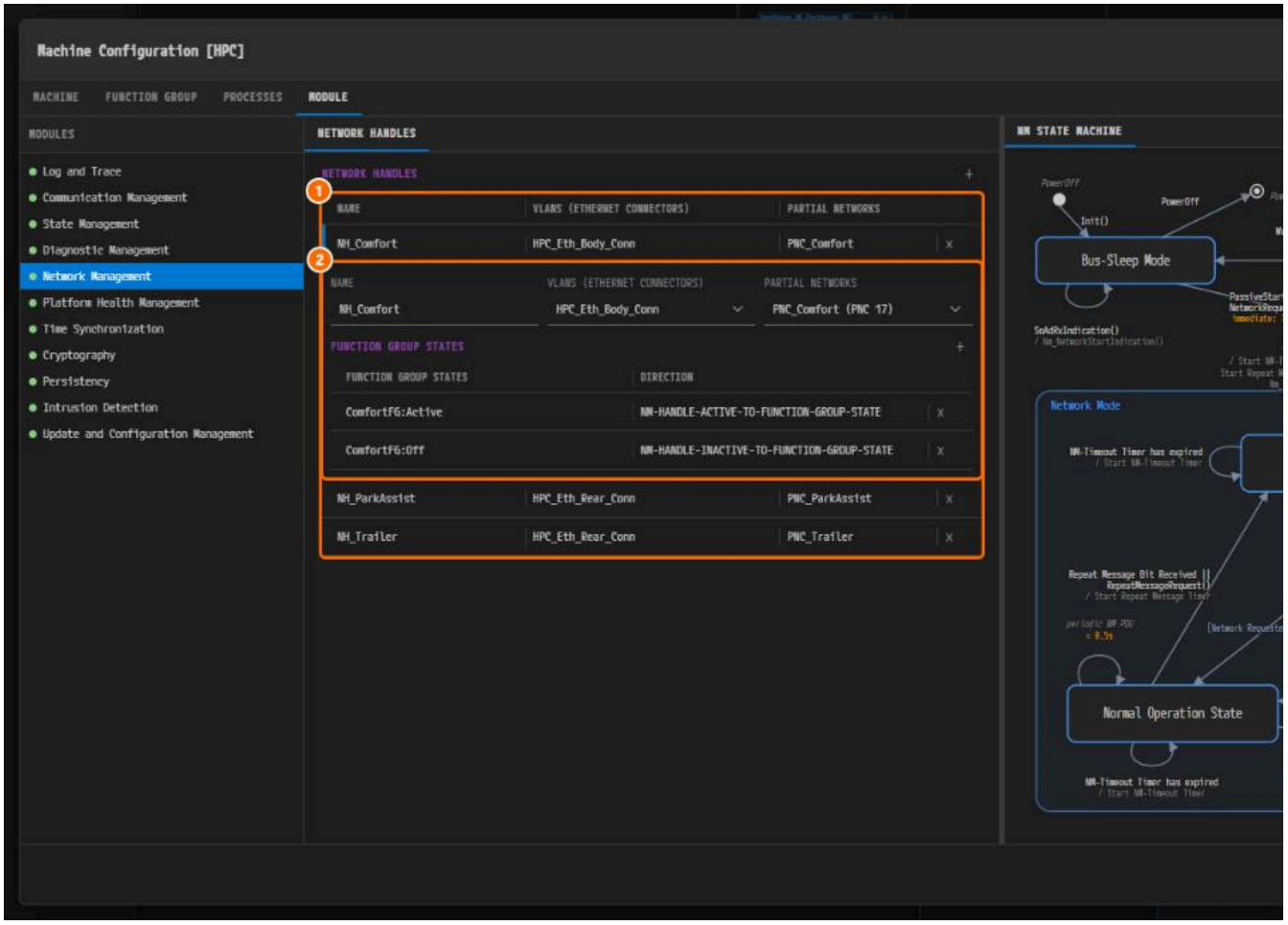


그림 9-20. ① 핸들 3개, ② `NH_Comfort`의 상세.

6. **State Management** > **ACTION ITEMS**에 매핑 6개가 보인다.

7. **Close**를 누른다.

> **주의.** PNC가 있는 핸들에 `NM-HANDLE-INACTIVE-TO-FUNCTION-GROUP-STATE` 매핑이 없으면 검증에서 "PNC release stops nothing" 오류가 난다.

왜 이렇게 설계하는가

- **HPC는 게이트웨이가 아니다. PNC GATEWAY TYPE**은 CP ECU 커넥터에만 있으므로, AP 머신은 ara::nm 네트워크 핸들로 PNC에 참여한다.
- **PNC마다 FunctionGroup 하나.** PNC가 살아 있는 동안 동작하는 애플리케이션이 한 상태를 공유한다.
- **양방향 매핑.** 후진 기어가 `ParkAssistFG`를 `Active`로 만들어 `PNC_ParkAssist`를 요청하고, `PNC_Comfort`가 활성이 되면 `ComfortFG`가 `Active`가 된다.
- **`PNC_ParkAssist`에는 웨이크업 프레임이 없다.** 버스 프레임이 아니라 HPC의 FunctionGroup 상태가 요청한다.

확인하기

- [ ] **FUNCTION GROUPS**에 `MachineFG`, `ParkAssistFG`, `TrailerFG`, `ComfortFG`가 있고 값과 전이가 표와 같다.
- [ ] **MAX ACTION LIST DURATION**이 `1` s이다.
- [ ] **NETWORK HANDLES**에 `NH_Comfort`, `NH_ParkAssist`, `NH_Trailer`가 있고 값이 나머지 핸들 추가하기와 같다.
- [ ] **NM HANDLE FUNCTION GROUP STATES**에 6행이 있다.
- [ ] 다이얼로그가 저장 확인 없이 닫혔다.

다음 단계

PARA 플랫폼 임포트와 PNC 요청 포트에서 PARA 플랫폼을 임포트하고, 이 핸들로 PNC를 요청하는 AA를 연결한다.

PARA 플랫폼 임포트와 PNC 요청 포트

이 장에서 할 일

이 장에서는 HPC 머신에 PARA 기본 플랫폼을 임포트한다. 이어서 각 응용 AA에 ara::nm P-Port를 달고 NM 네트워크 핸들과 매핑해, AA가 PNC를 요청하고 해제할 수 있게 한다. P 표를 따른다.

시작하기 전에

- AP 머신 — NM 핸들과 FunctionGroup을 마친 뒤 시작한다. HPC 머신에 핸들 `NH_Comfort`, `NH_ParkAssist`, `NH_Trailer`가 있다.
- 참고: VFB 편집기의 8.9~8.11절, Interface 편집기의 9.3.5절.

항목	값
NM 포트 인터페이스 **NAME** / **C++ NAMESPACE**	`NMI_PncRequest` / `pnc::nm`
`ParkAssistApp`의 P-Port **NAME**	`PncParkAssist`
P-Port의 **PORT INTERFACE**	`NM.NetworkHandle: NMI_PncRequest`
NetworkHandle 매핑 **PORT** / **NETWORK HANDLE**	`PncParkAssist` / `NH_ParkAssist`

PARA 플랫폼이란

PARA는 PopcornSAR AUTOSAR Adaptive Platform의 기본 구성 세트이다. 한 번의 임포트로 AP 머신이 동작하는 데 필요한 것을 넣는다.

항목	내용
플랫폼 AA 11개	`EM`, `CM`, `PHM`, `SM`, `TBM`, `DM`, `IDSM`, `NM`, `UCM`, `VUCM`, `CSM`. 컴포지션 `PARA_HPC`에 묶인다(HPC 안의 프로토타입 `PARA`)
인터페이스	서비스 인터페이스 8개와 상태 관리 인터페이스 `StateClient_HPC_MachineFG`
머신	FunctionGroup `MachineFG`와 모듈 7개(`LOG`, `CM`, `SM`, `NM`, `PHM`, `TIME_SYNC`, `CRYPTO`)
기타	오류 도메인 4개, 데이터 타입 62개, `PARA_HPC` 내부 커넥터 5개

PARA에는 응용 AA용 NM 포트 인터페이스가 없다. `NMI_PncRequest`는 NM 포트 인터페이스 만들기에서 만든다.

PARA 플랫폼 임포트하기

1. **DESIGN** 아래의 **VFB**를 누른다. 루트 화면에서 9장처럼 **HPC [HPC]** 머리글 오른쪽의 머신 아이콘을 누른다.

2. **Machine Configuration [HPC]**의 오른쪽 위에서 **PARA Platform Import**를 누른다.

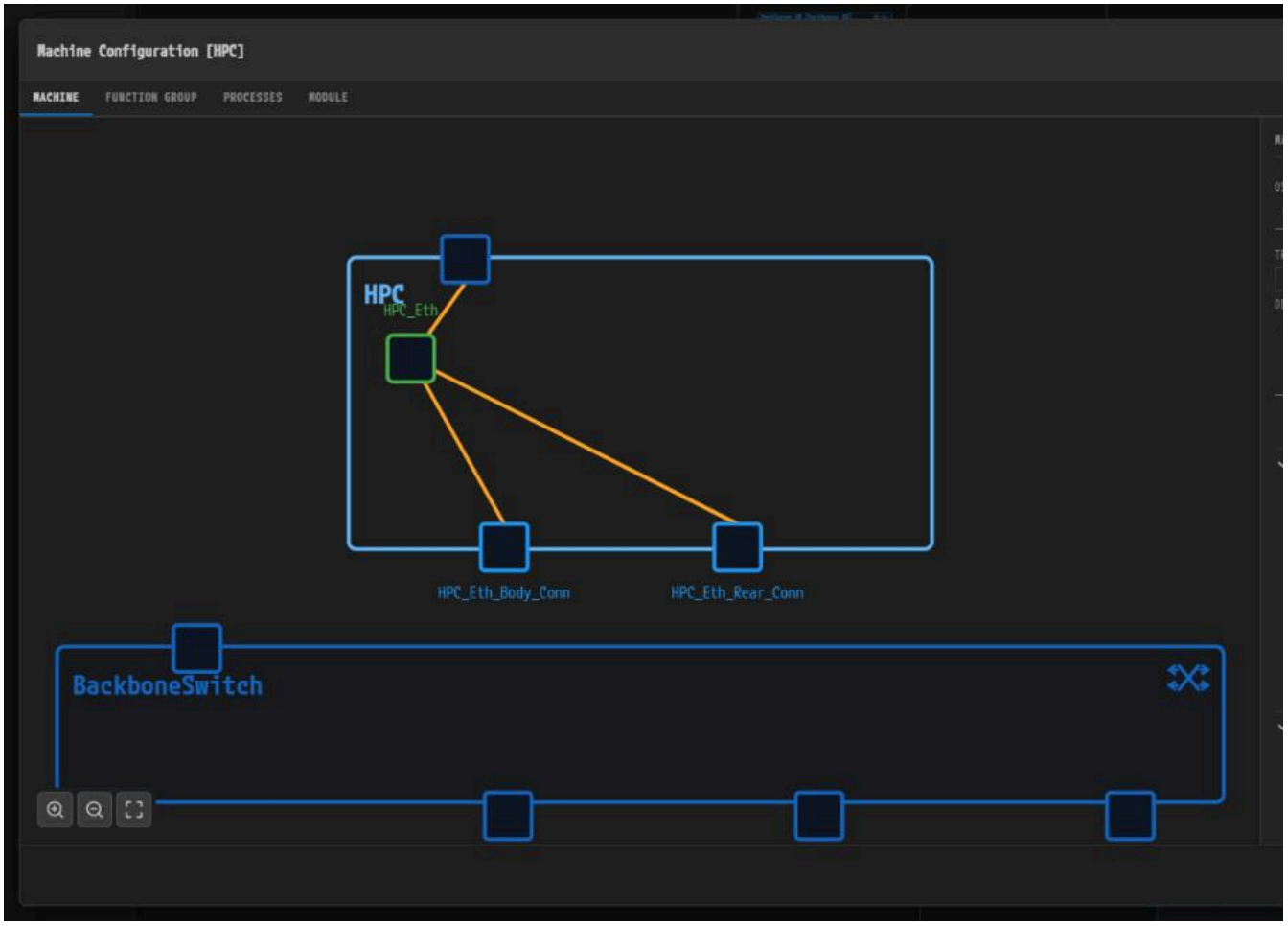


*그림 10-1. ① **PARA Platform Import.***

3. **Import**를 누른다.

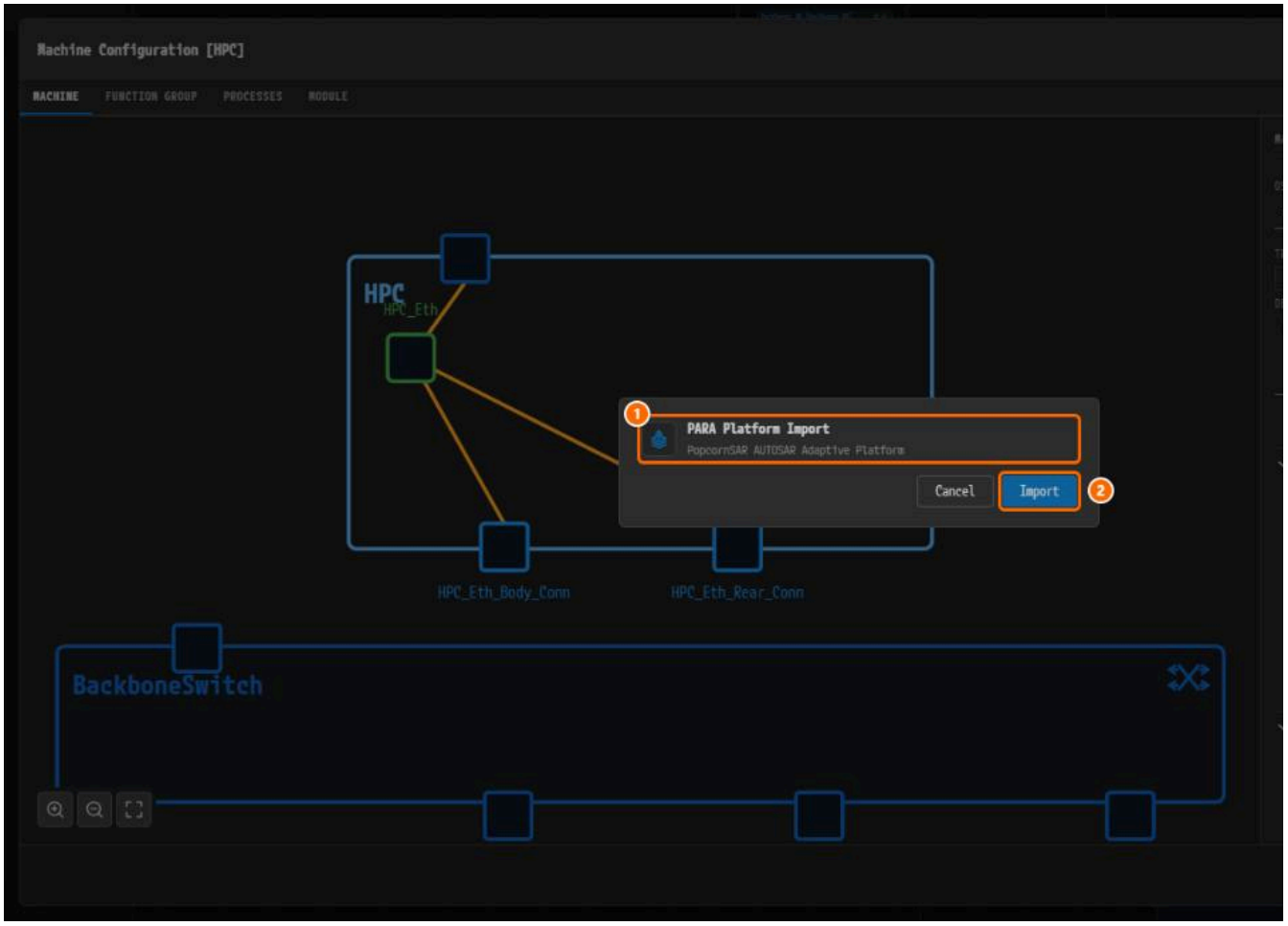


*그림 10-2. ① **PARA Platform Import** 창, ② **Import**.*

4. 충돌 하나(**Machine (1)** > **HPC**)를 담은 **Conflict Resolution** 다이얼로그가 열린다. PARA는 **Max Action List Duration**을 `5`로 두고, 9장에서는 `1`을 입력했다. **Max Actic**
Resolve 1/1을 누른다.

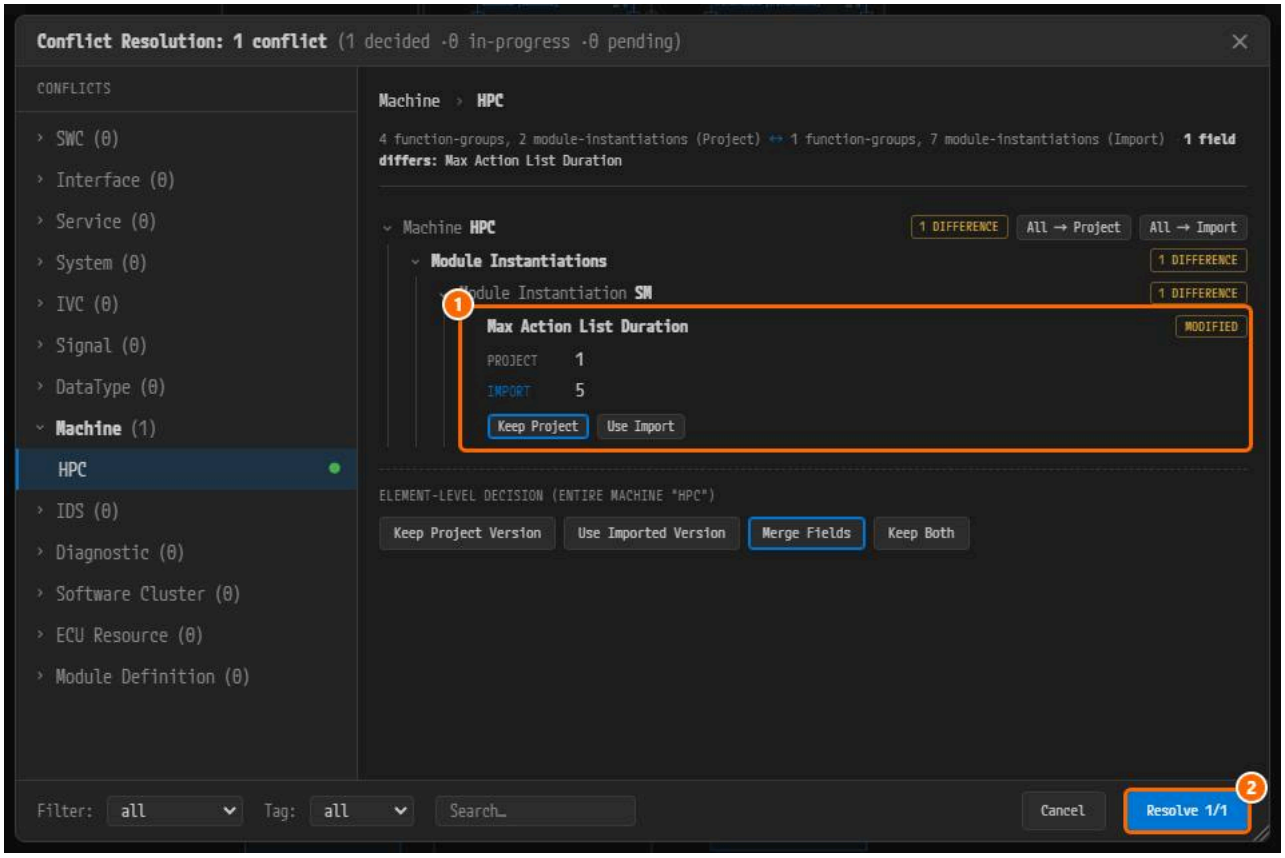


*그림 10-19. ① **Max Action List Duration** 아래의 **Keep Project**, ② **Resolve 1/1**.*

> **주의.** **Resolve**를 누르기 전에 결정한다. 결정하지 않은 충돌은 임포트 값으로 덮인다. 머신 단위 버튼은 쓰지 않는다. **Use Imported Version**은 NM 네트워크 핸들을 없애고 하지 않는다.

5. 임포트가 끝나면 **Machine Configuration [HPC]** 다이얼로그가 다시 열린다.

임포트 결과 확인하기

1. **MODULE** 탭에서 **Network Management**를 누른다. **NETWORK HANDLES** 표에 핸들 3개가 그대로 있다.

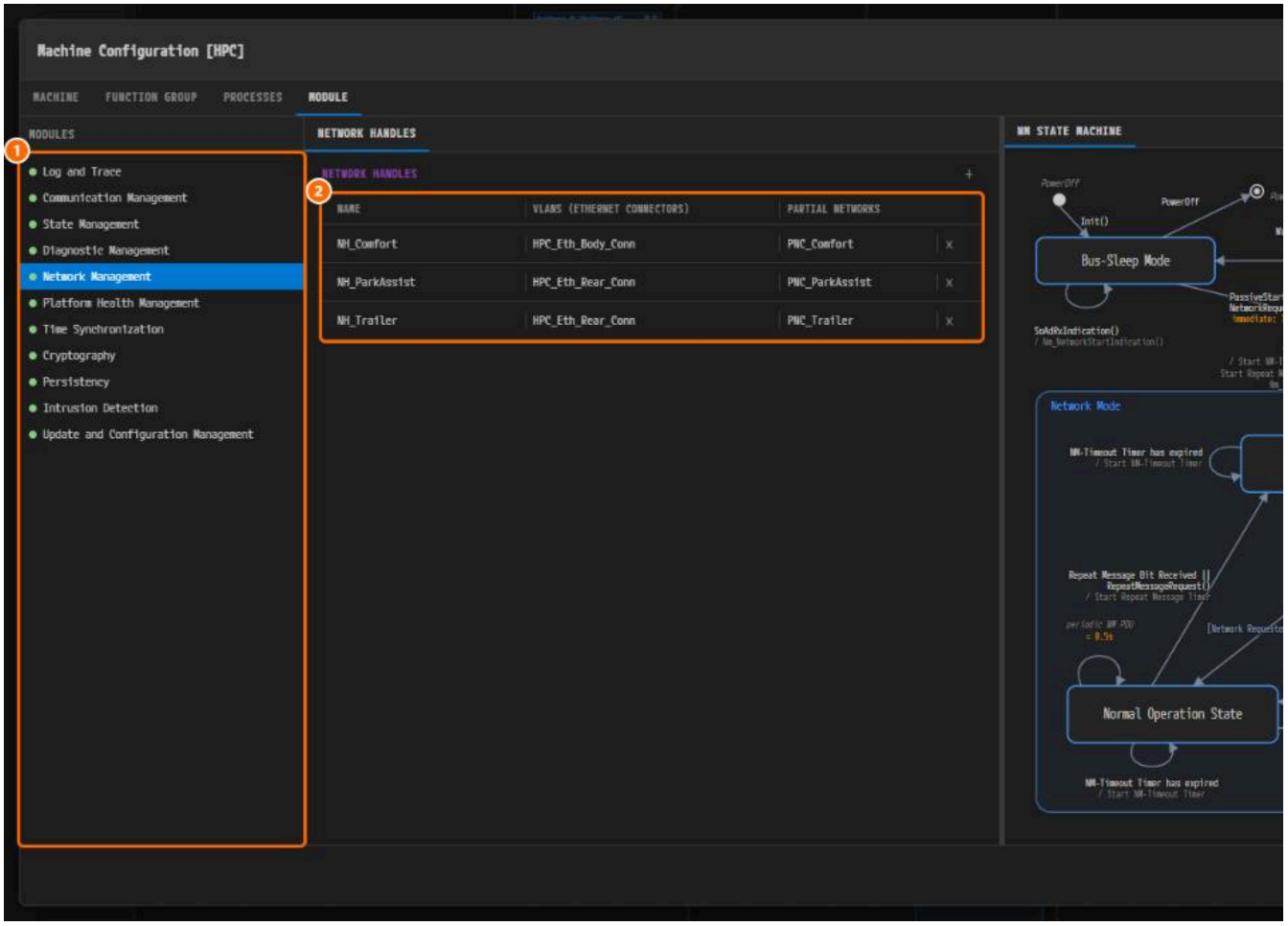


*그림 10-3. ① **MODULES**(모듈 종류 11개가 늘 표시된다), ② **NETWORK HANDLES**.*

2. **PROCESSES** 탭을 누른다. 응용 프로세스 3개와 PARA 프로세스 11개가 보인다.

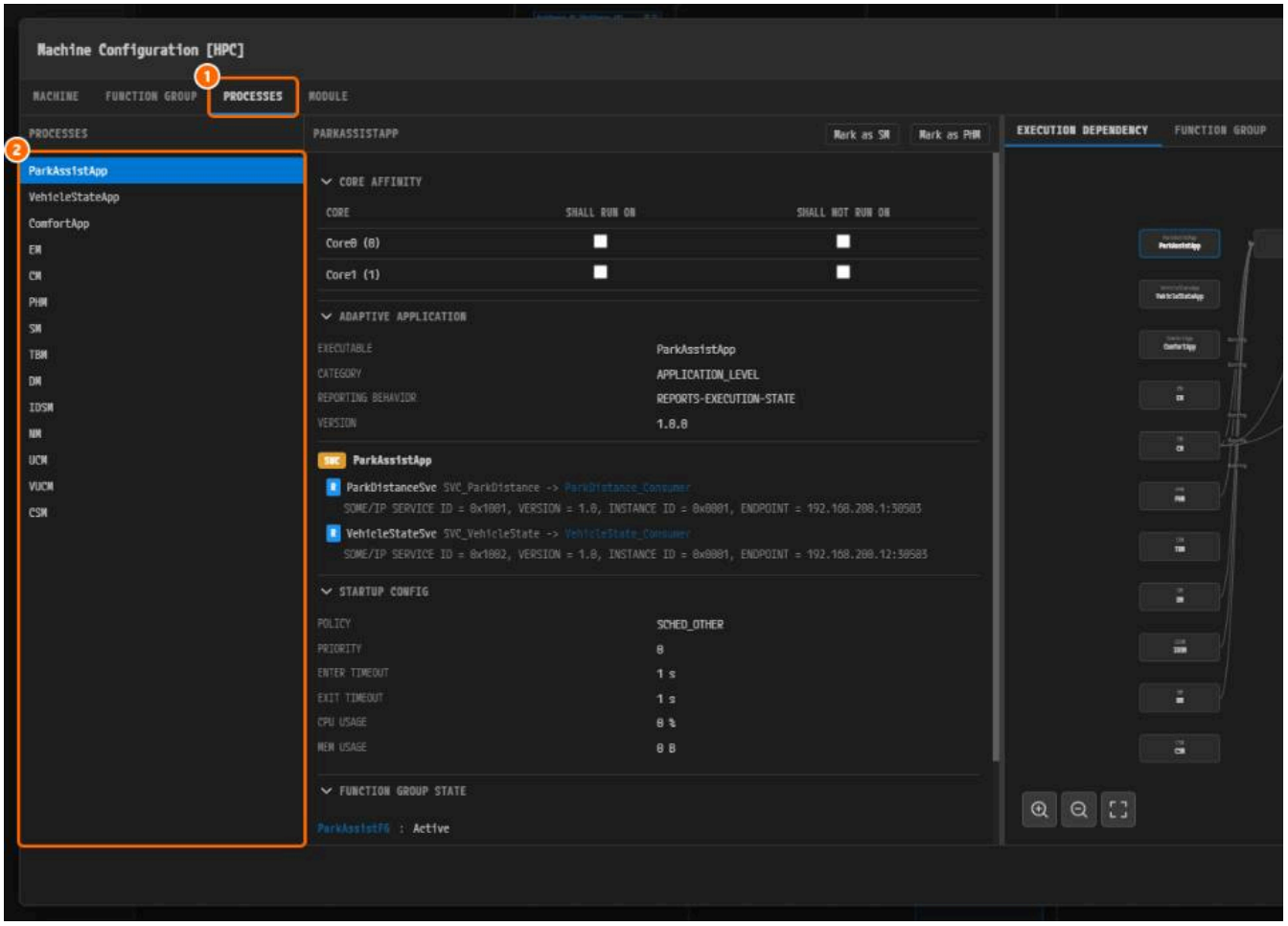


*그림 10-4. ① **PROCESSES**, ② 프로세스 14개.*

3. **Close**를 누른다.

HPC 컴포지션 정리하기

1. VFB 캔버스에서 **HPC [HPC]**를 더블클릭한다. 새 노드 **PARA_HPC**가 **ComfortApp**과 겹치면 **Auto Layout**을 누른다.

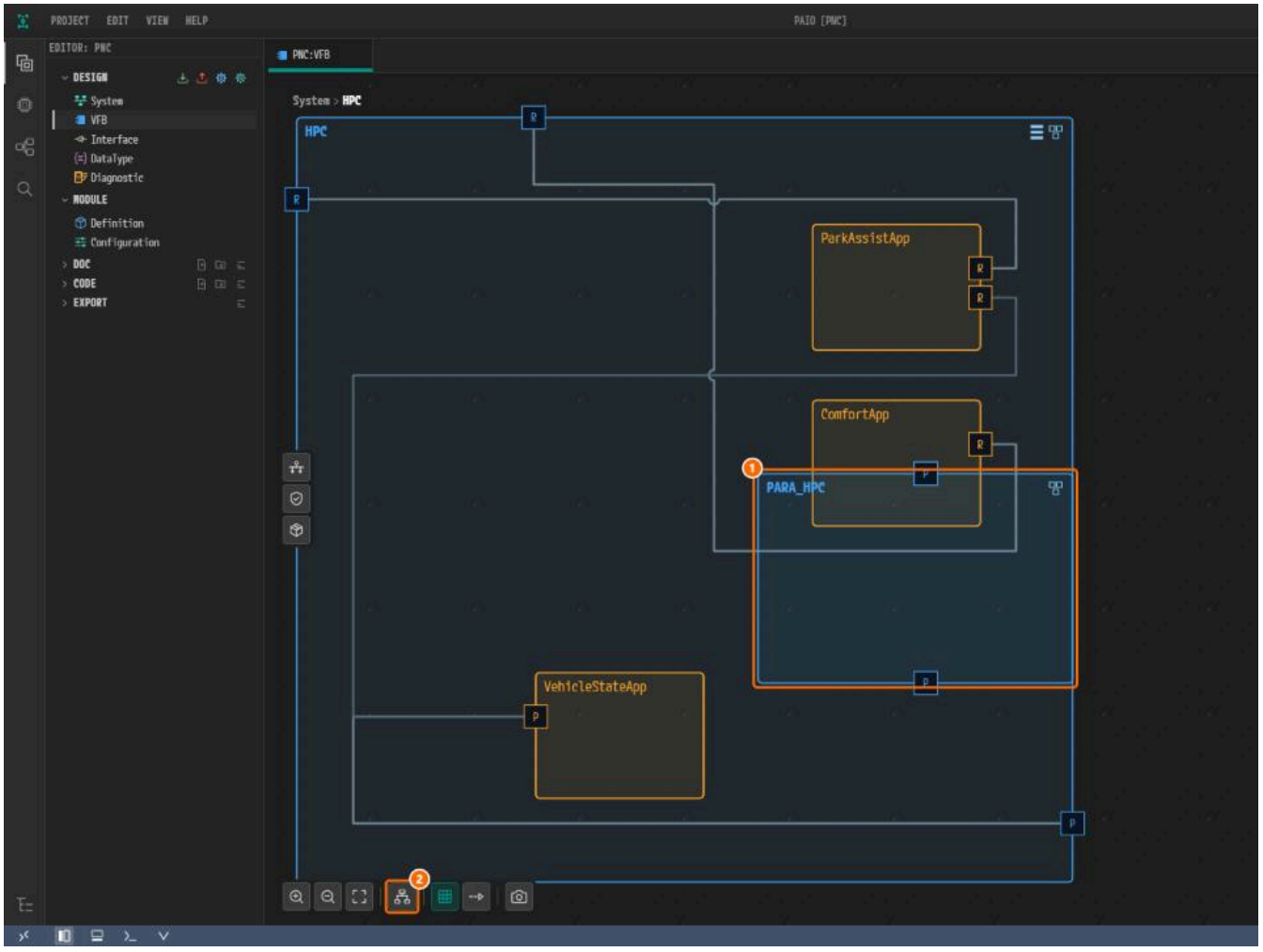


*그림 10-5. ① **PARA_HPC**, ② **Auto Layout**.*

2. `Ctrl+S`로 저장한다.

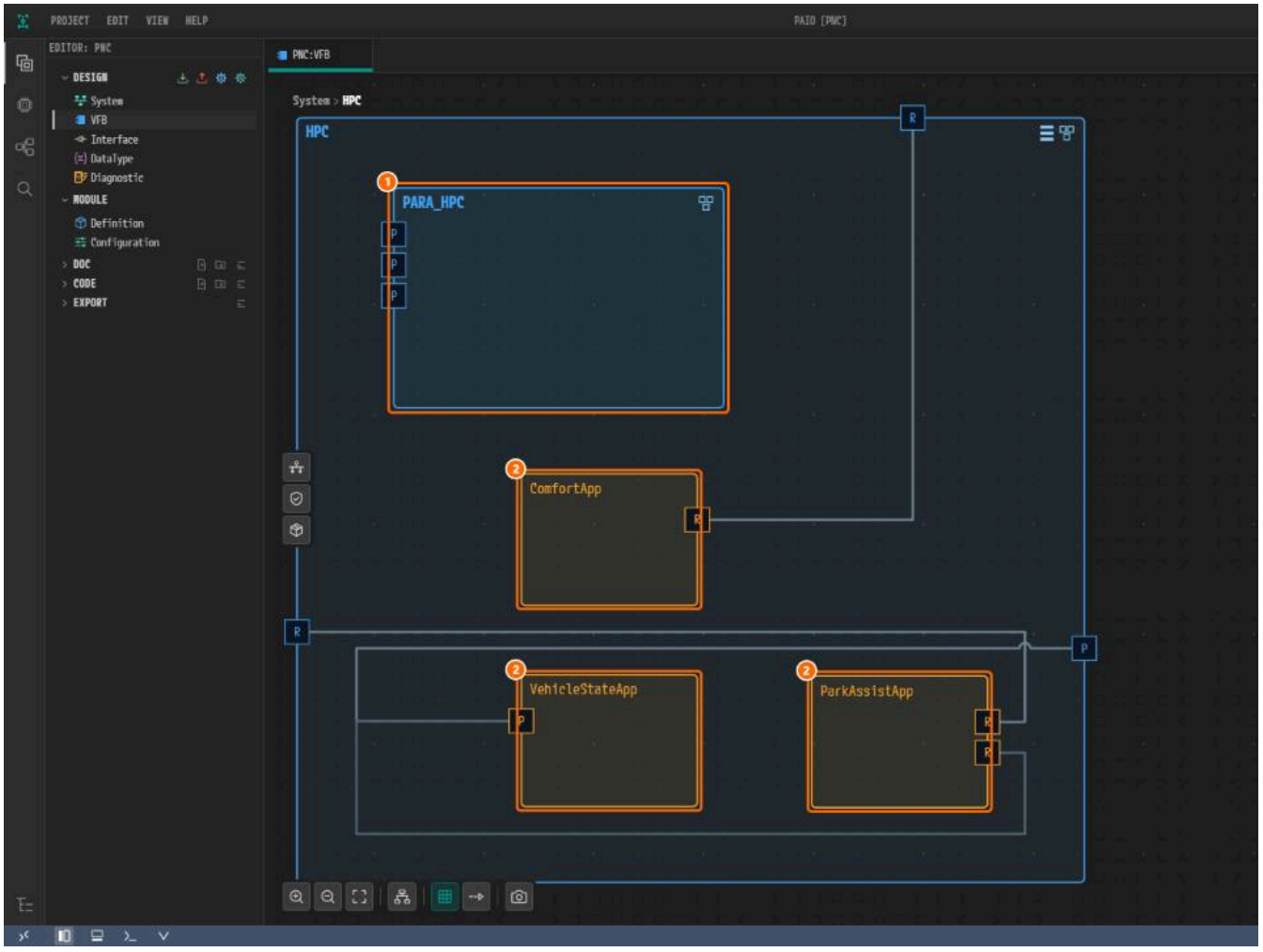


*그림 10-6. ① **PARA_HPC**로 표시되는 프로토타입 `PARA`, ② 응용 AA 3개.*

NM 포트 인터페이스 만들기

1. **DESIGN** 아래의 **Interface**를 누르고 **ADAPTIVE** 탭을 연다.

2. **INTERFACES** 옆의 **+**를 누르고 **ara::nm** > **NetworkManagement**를 고른다.

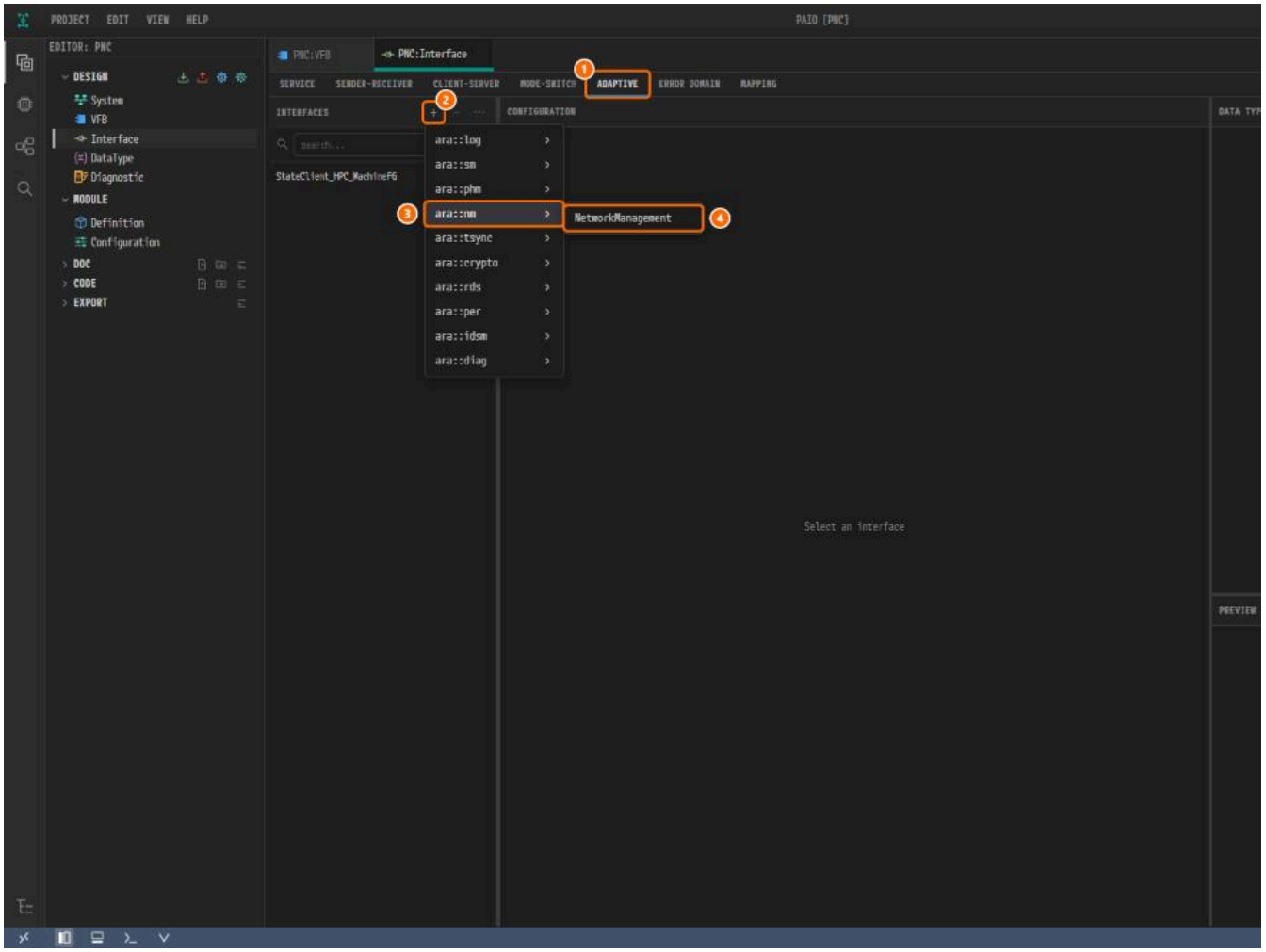


*그림 10-7. ① **ADAPTIVE**, ② **+**, ③ **ara::nm**, ④ **NetworkManagement**.*

3. **CONFIGURATION**의 **NAME**에 `NMI_PncRequest`, **C++ NAMESPACE**에 `pnc::nm`을 입력한다. `Ctrl+S`로 저장한다.

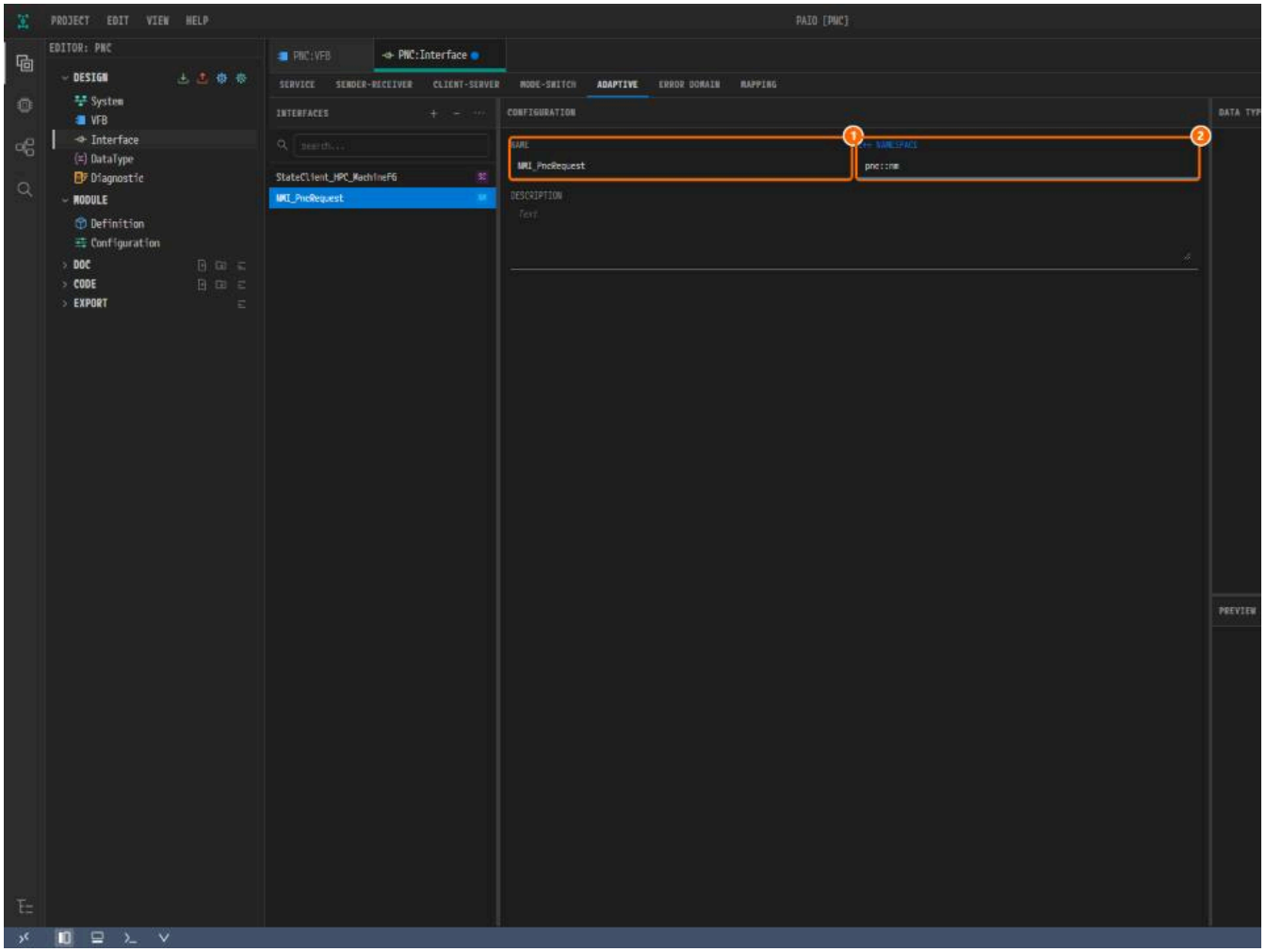


*그림 10-8. ① **NAME**, ② **C++ NAMESPACE**.*

ParkAssistApp에 PNC 요청 포트 추가하기

1. HPC 내부가 열린 채인 탭 **PNC:VFB**로 돌아간다. **ParkAssistApp**을 우클릭하고 **Add Port** > **PPort**를 고른다.

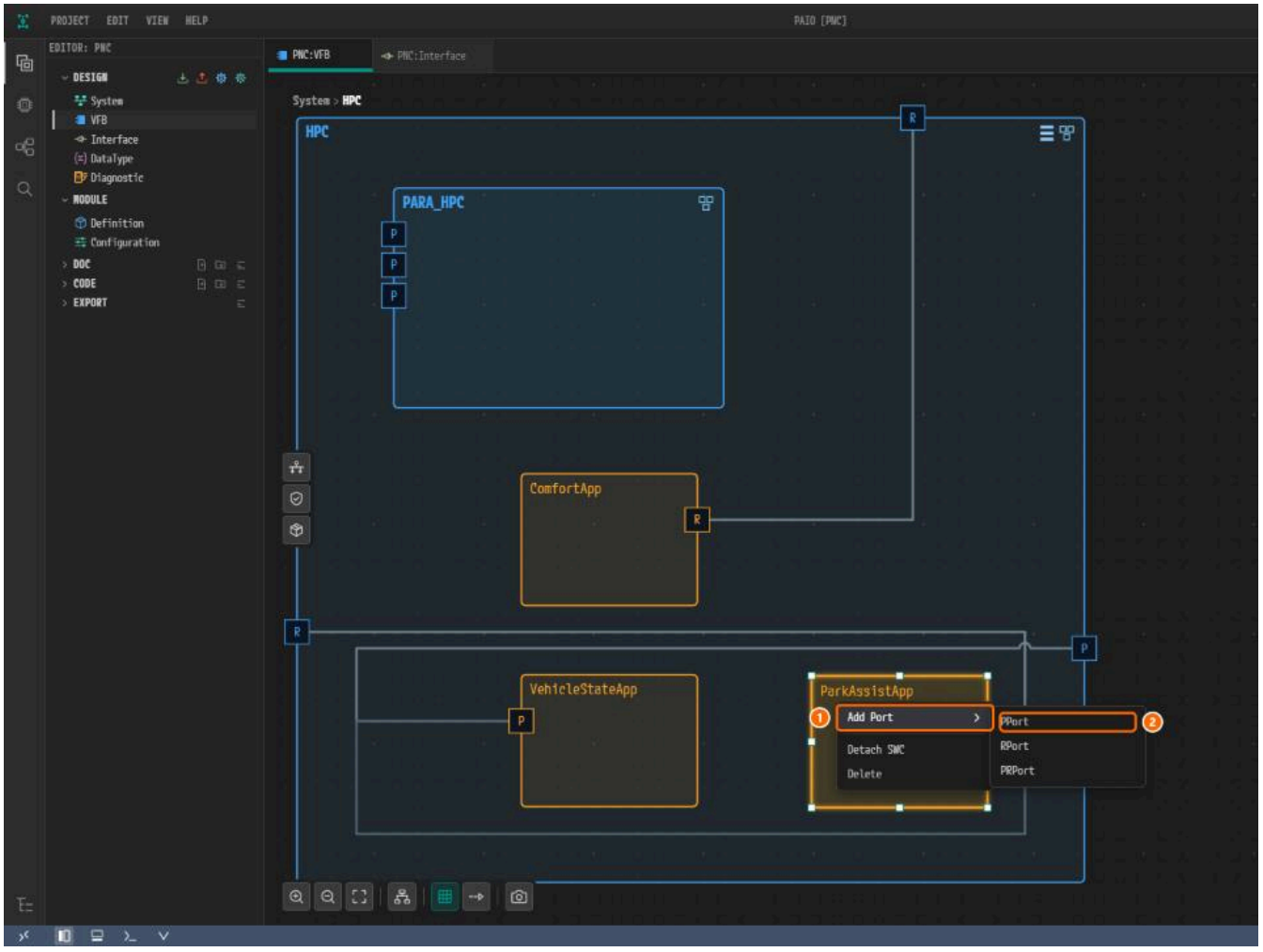


*그림 10-9. ① **Add Port**, ② **PPort**.*

2. 새 포트를 누른다. **NAME**을 `PncParkAssist`로 바꾸고 `Tab`을 누른다.

3. **PORT INTERFACE**에 `NMI`를 입력하고 **NM.NetworkHandle: NMI_PncRequest**를 고른다.



*그림 10-10. ① **NAME**, ② **PORT INTERFACE**, ③ **NM.NetworkHandle: NMI_PncRequest**.*

4. `Ctrl+S`로 저장한다. 포트가 NM 포트를 뜻하는 원형으로 바뀐다.

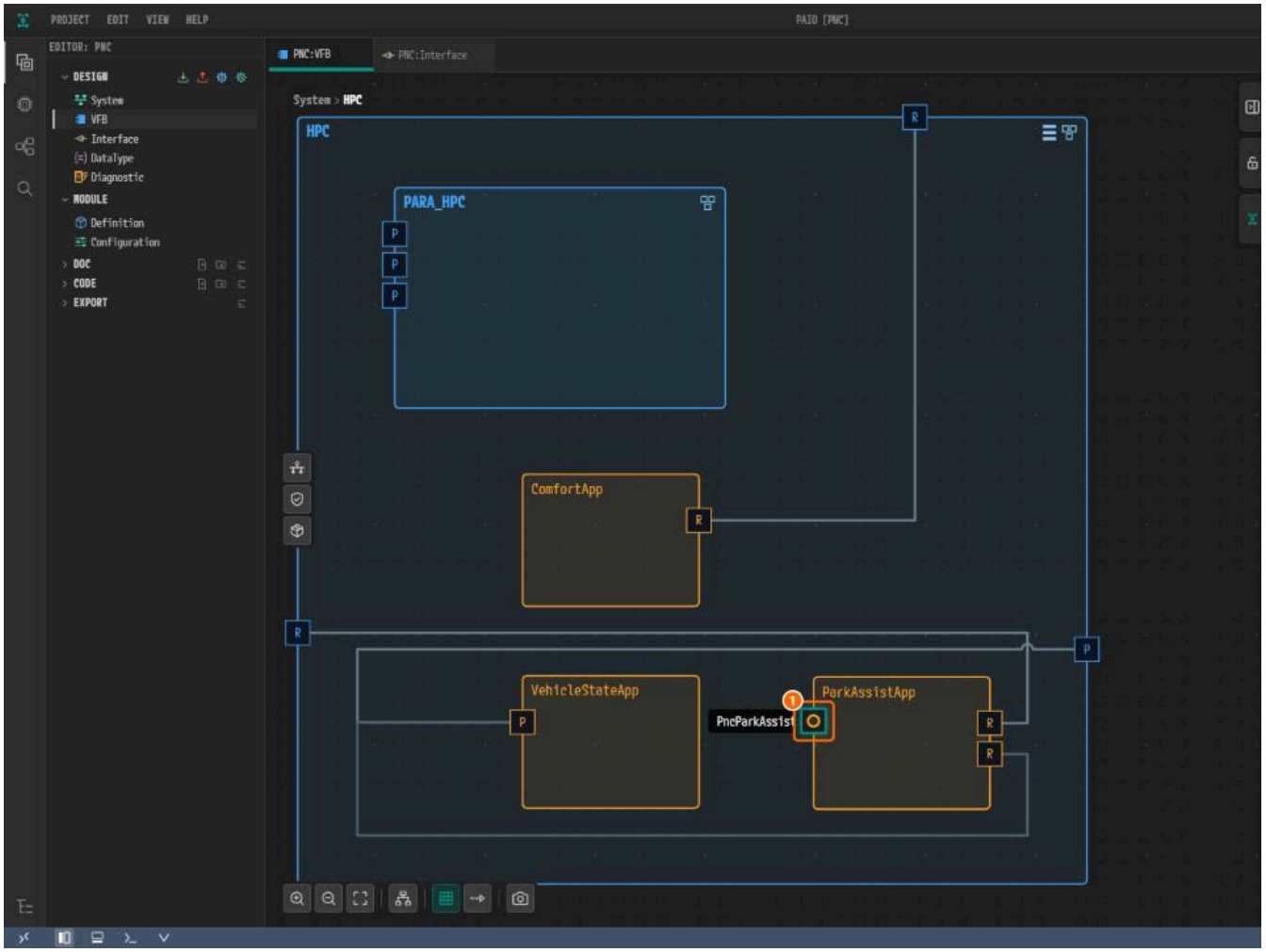


*그림 10-11. ① `PncParkAssist`, ② 그 **PORT INTERFACE**.*

NetworkHandle 매핑하기

1. **ParkAssistApp**을 더블클릭한다. **AA Internal Behavior [ParkAssistApp]** 다이얼로그가 열린다.

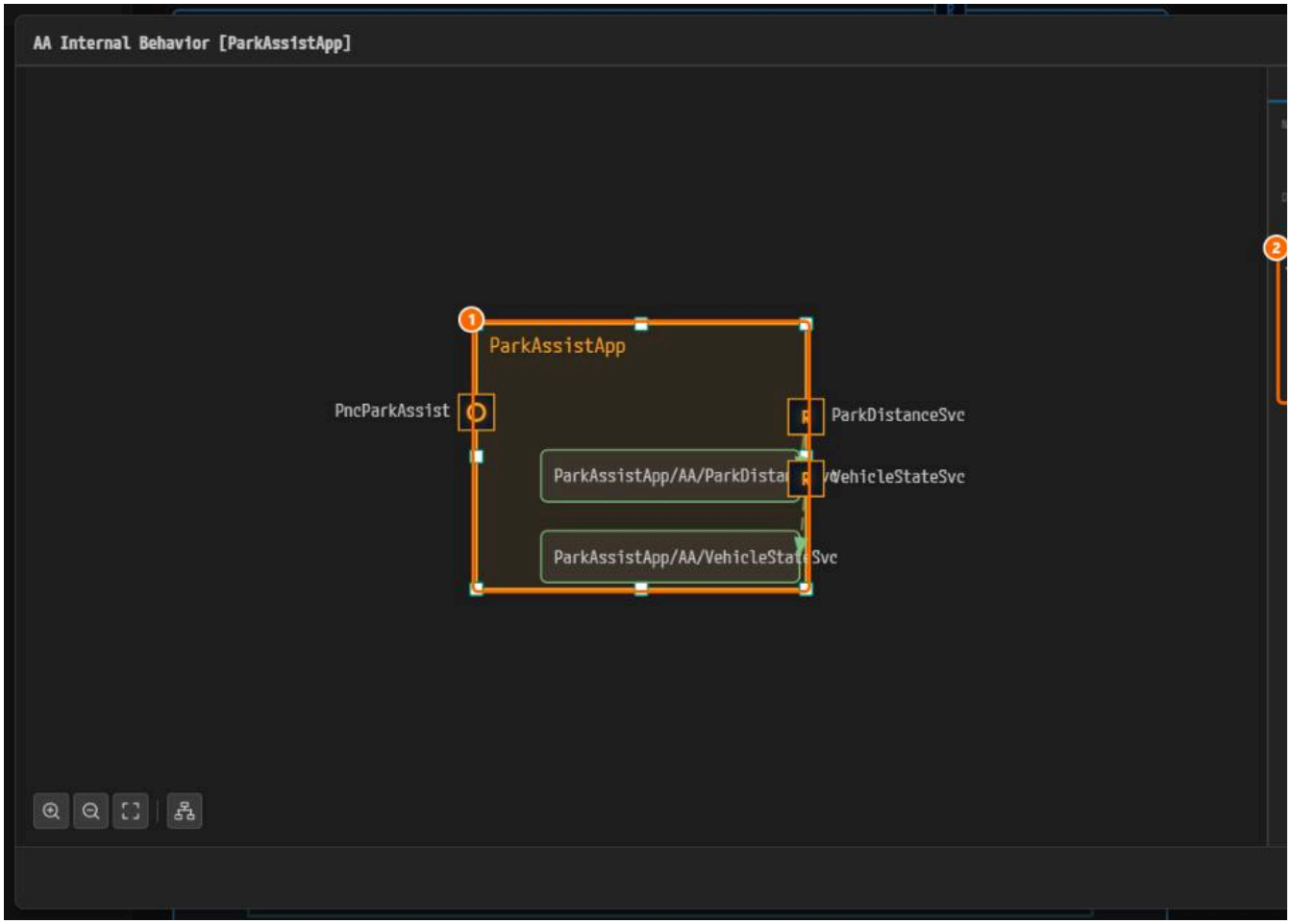


*그림 10-12. ① 포트 `PncParkAssist`가 있는 AA 박스, ② **PORTS**.*

2. AA 박스 안의 빈 곳을 우클릭하고 **Add Instance** > **ara::nm** > **Network Handle**을 고른다.

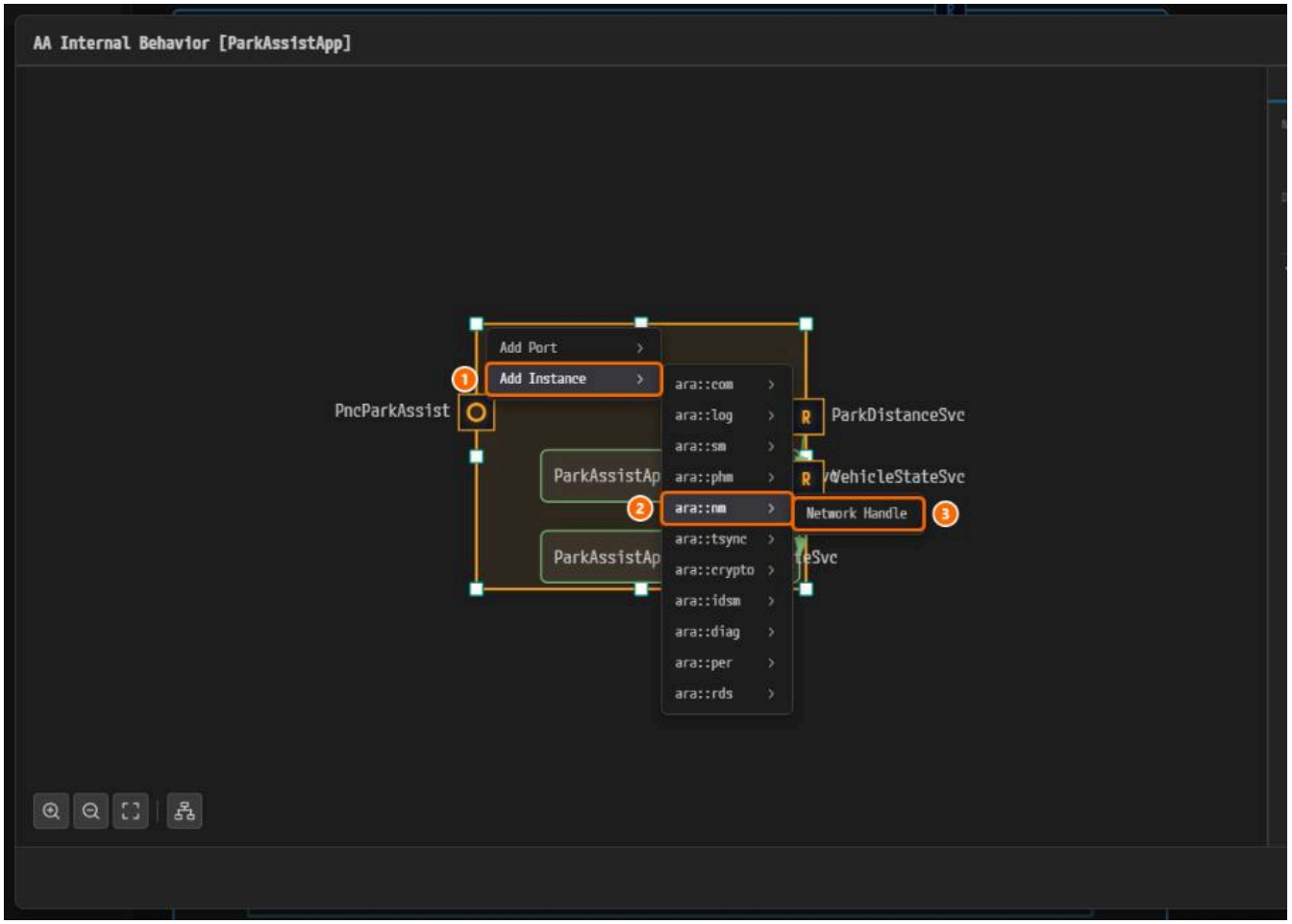


*그림 10-13. ① **Add Instance**, ② **ara::nm**, ③ **Network Handle**.*

3. **NETWORK HANDLE MAPPING** 패널의 **PORT**에서 `PncParkAssist`를 고른다.

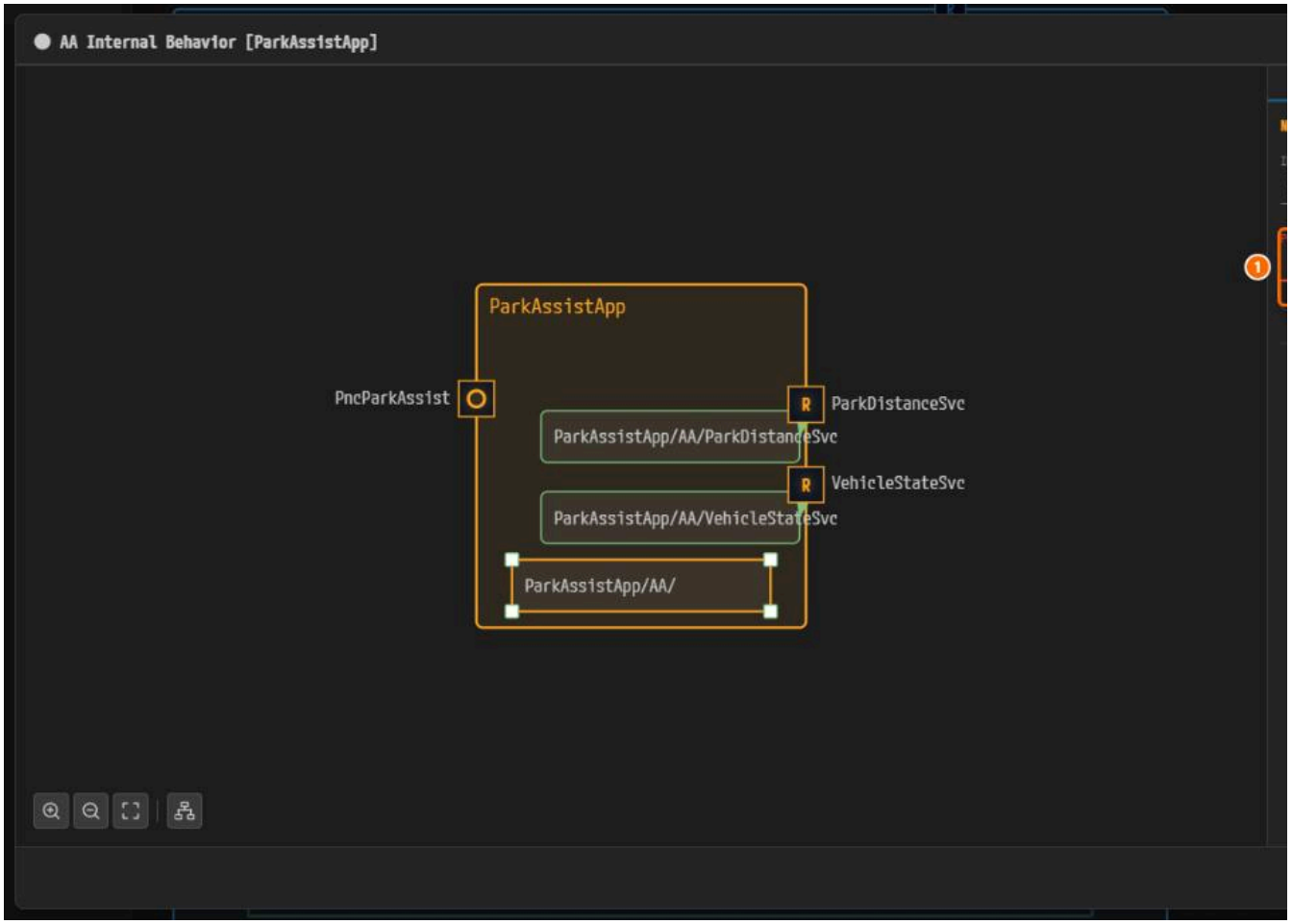


*그림 10-14. ① **PORT**, ② `PncParkAssist`.*

4. **NETWORK HANDLE**에서 `NH_ParkAssist`를 고른다.

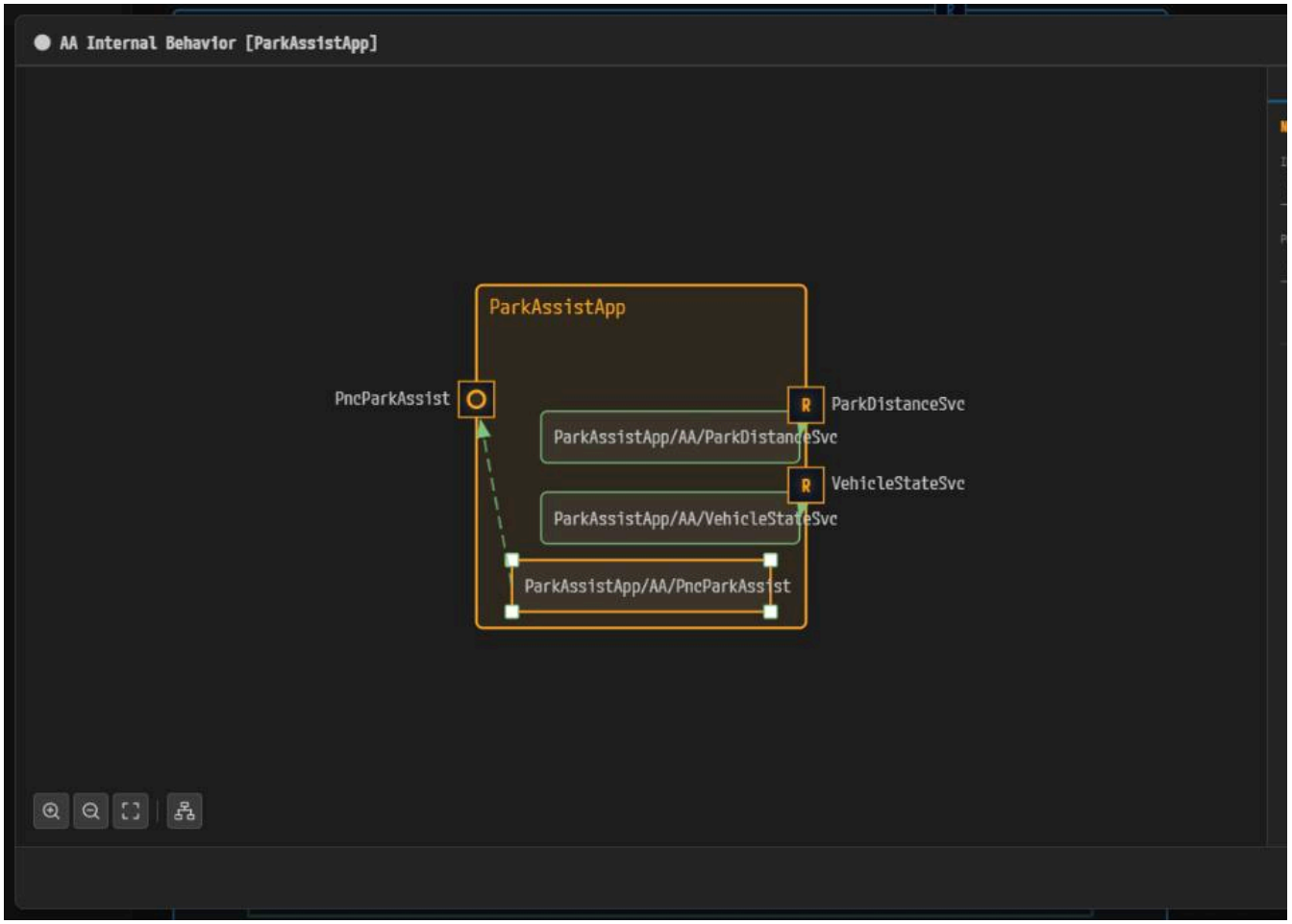


*그림 10-15. ① **NETWORK HANDLE**, ② `NH_ParkAssist`.*

5. `Ctrl+S`로 저장한다.

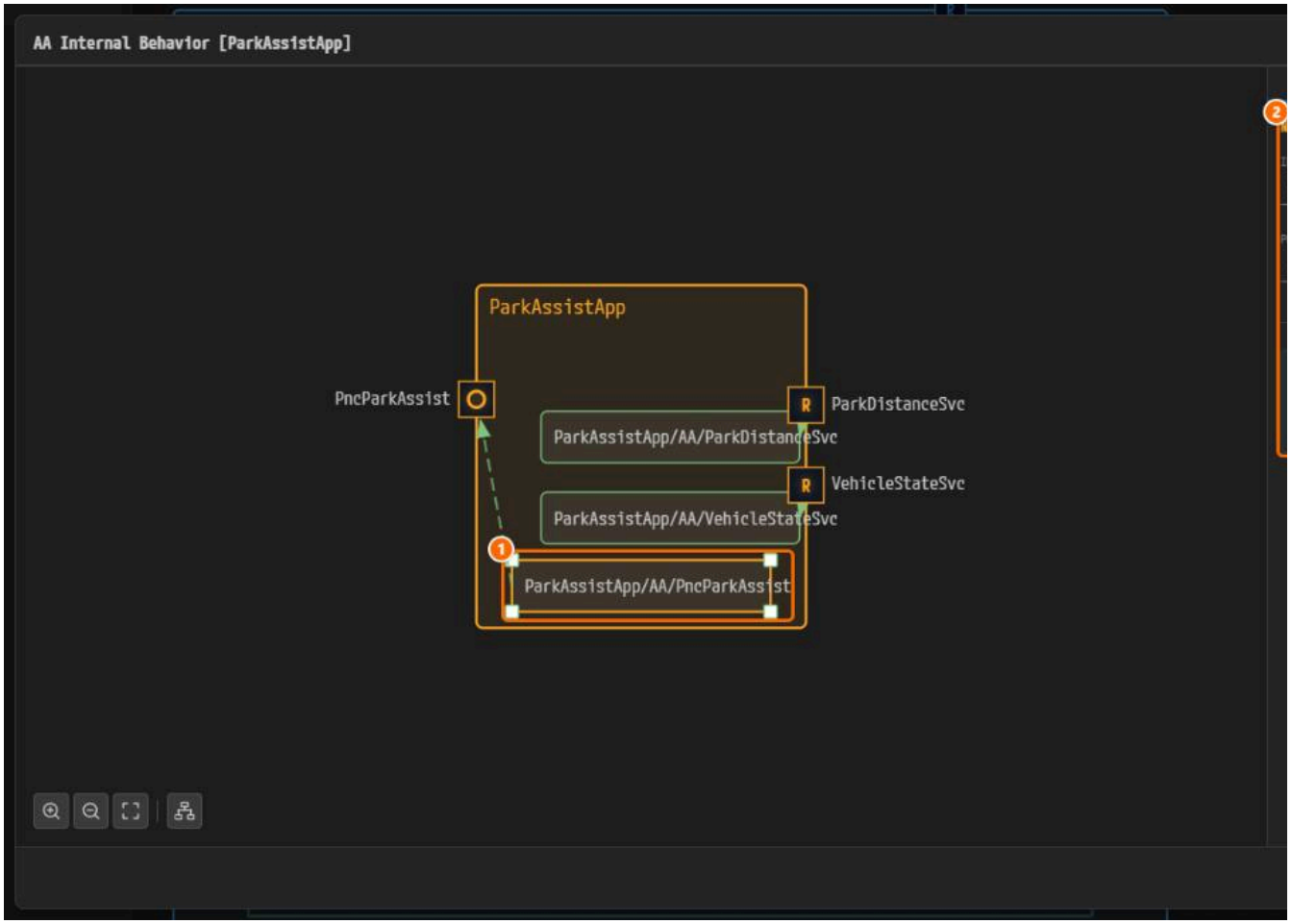


그림 10-16. ① 매핑 `ParkAssistApp/AA/PncParkAssist`, ② 그 값과 핸들 요약.

> **주의.** 요약의 **NM CLUSTER** 줄은 `NmCluster_VLAN_Body (Backbone)`로 보이지만 `NH_ParkAssist`가 속한 것은 `NmCluster_VLAN_Rear`이다. 이 줄은 표시용일

6. **Close**를 누른다.

나머지 AA 연결하기

`VehicleStateApp`과 `ComfortApp`에도 같은 방법으로 P-Port를 달고 NetworkHandle 매핑을 만든다. 모든 포트의 **PORT INTERFACE**는 `NM.NetworkHandle: NMI_PncR`

AA(노드 라벨 / 프로토타입)	P-Port NAME	NETWORK HANDLE	저장되는 매핑 이름
ParkAssistApp / `ParkAssist`	`PncParkAssist`	`NH_ParkAssist`	`ParkAssistApp_AA_PncParkAssist`
VehicleStateApp / `VehicleState`	`PncTrailer`	`NH_Trailer`	`VehicleStateApp_AA_PncTrailer`
ComfortApp / `Comfort`	`PncComfort`	`NH_Comfort`	`ComfortApp_AA_PncComfort`

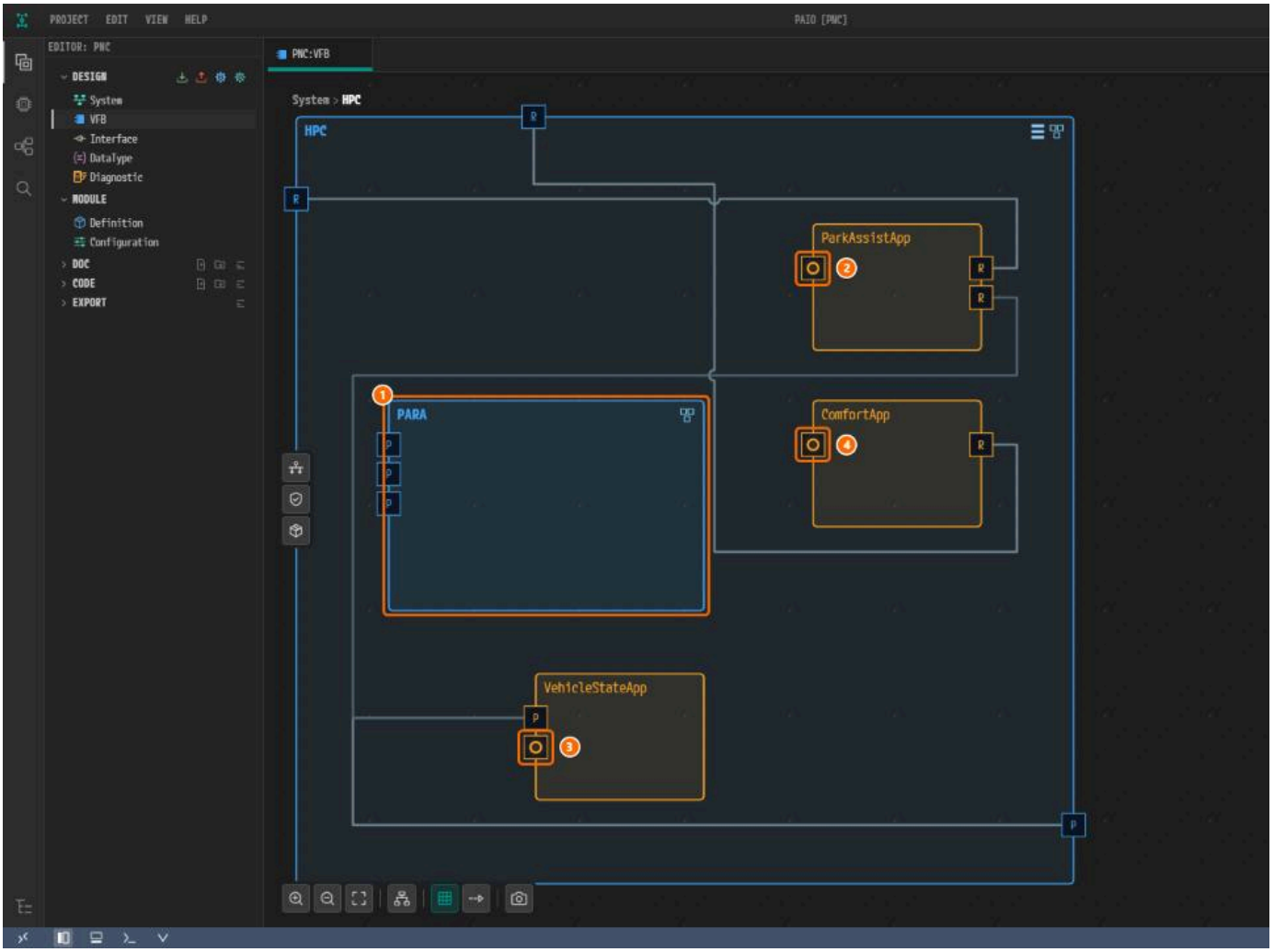


*그림 10-17. 완성 예제. ① `PARA`, ② `PncParkAssist`, ③ `PncTrailer`, ④ `PncComfort`. 예제는 이전 형식의 PARA를 쓰므로 노드가 **PARA_HPC** 대신 **PARA**로 보이고, 노드 위*

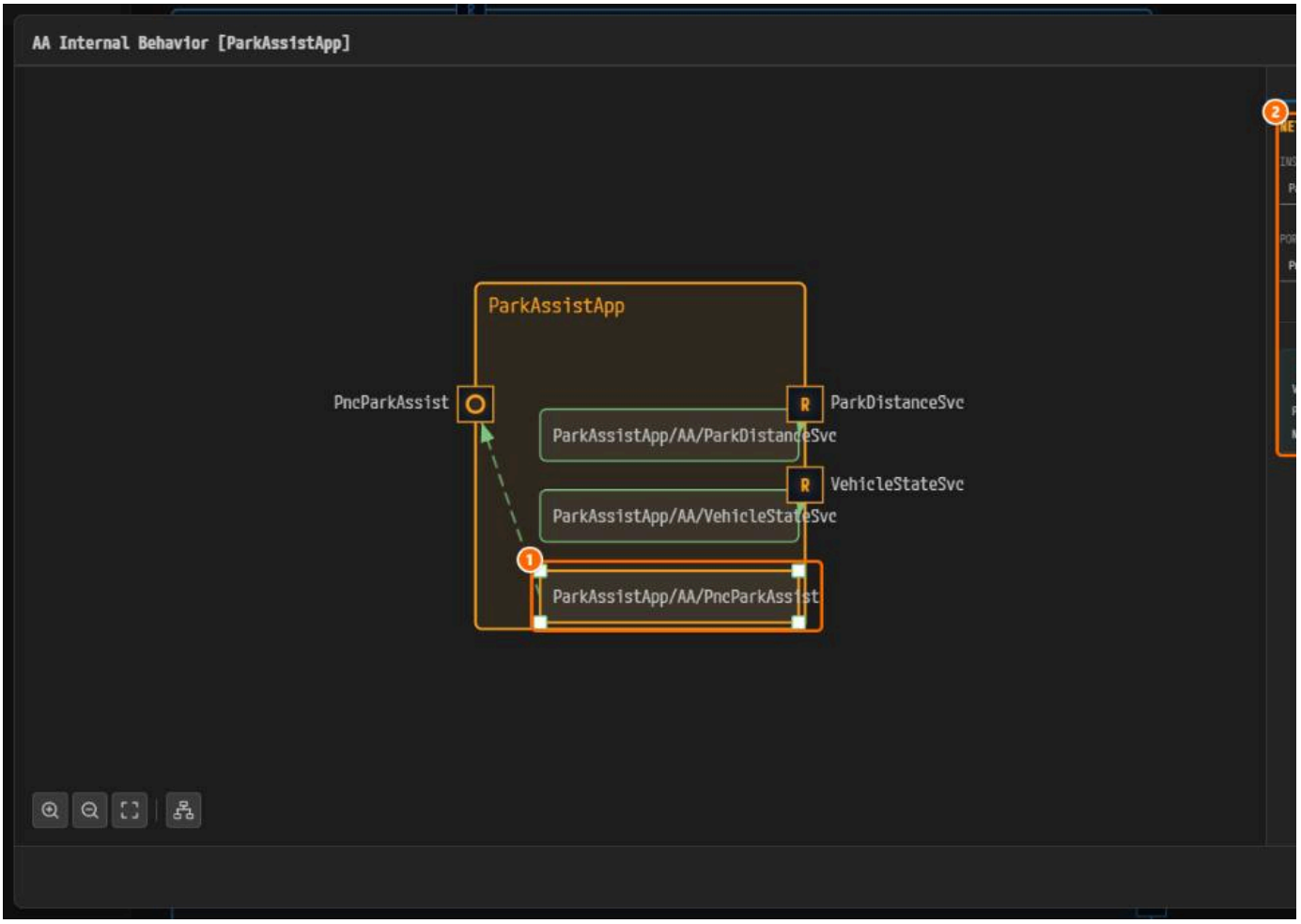


*그림 10-18. 완성 예제의 **AA Internal Behavior [ParkAssistApp]**. ① NetworkHandle 매핑, ② 그 값. 그림 10-16과 같아야 한다.*

익스포트되는 내용

이 장에서 만든 것	익스포트되는 요소
`NMI_PncRequest`	인터페이스 문서의 `NETWORK-MANAGEMENT-PORT-INTERFACE`
응용 AA의 NetworkHandle 매핑 3개	AA Executable Manifest의 `NETWORK-HANDLE-PORT-MAPPING`

왜 이렇게 설계하는가

- **두 가지 결정.** 9장의 NM 네트워크 핸들은 어느 VLAN의 어느 PNC인지를, 이 장의 매핑은 어느 응용이 그 핸들을 요청하는지를 정한다.
- **응용은 핸들만 본다.** ara::nm P-Port로 핸들을 요청하고 해제하며, VLAN이나 PNC 번호는 다루지 않는다.
- **임포트는 병합이다.** 머신 모듈을 플랫폼 AA(`SM`, `NM`)와 잇고, 앞 장의 머신 설정은 그대로 둔다.

확인하기

- [ ] HPC 머신의 **Network Management**에 `NH_Comfort`, `NH_ParkAssist`, `NH_Trailer`가 그대로 있다.
- [ ] HPC 안에 **PARA_HPC**가 있고 다른 노드와 겹치지 않는다.
- [ ] **ADAPTIVE** 탭에 **C++ NAMESPACE**가 `pnc::nm`인 `NMI_PncRequest`가 있다.
- [ ] 응용 AA마다 나머지 AA 연결하기의 표대로 원형 P-Port와 NetworkHandle 매핑이 있다.
- [ ] 탭과 다이얼로그 제목에 수정 표시가 없다.

다음 단계

System 에디터와 검증에서 설계를 검증하고 보고된 항목을 고친다.

System 에디터와 검증

이 장에서 할 일

이 장에서는 System 편집기와 VFB 편집기의 SYSTEM 패널에서 설계를 확인한다. 이어서 **VALIDATION** 패널을 열어 오류를 고치고 "No validation issues." 상태로 만든다.

시작하기 전에

- PARA 플랫폼 임포트와 PNC 요청 포트를 마친 뒤 시작한다.
- 그림 11-4부터 11-12까지는 SOME/IP 소비자 포트가 제공자와 겹친 초안 상태이다. 앞 장을 따라 했다면 이 오류는 없으므로 절차만 읽어 익힌다.
- 직접 만든 프로젝트와의 차이는 부록 — 직접 만든 프로젝트와 완성 예제의 차이를 참고한다.
- 전체 설명은 PAIO 매뉴얼의 Validation과 System Editor에 있다.

SYSTEM 패널 보기

1. **EDITOR** 뷰의 **DESIGN** 아래에서 **System**을 클릭한다(탭 `PNC:System`). 캔버스 오른쪽 가장자리의 세 번째 버튼인 **System** 버튼(PAIO 로고)을 클릭한다.

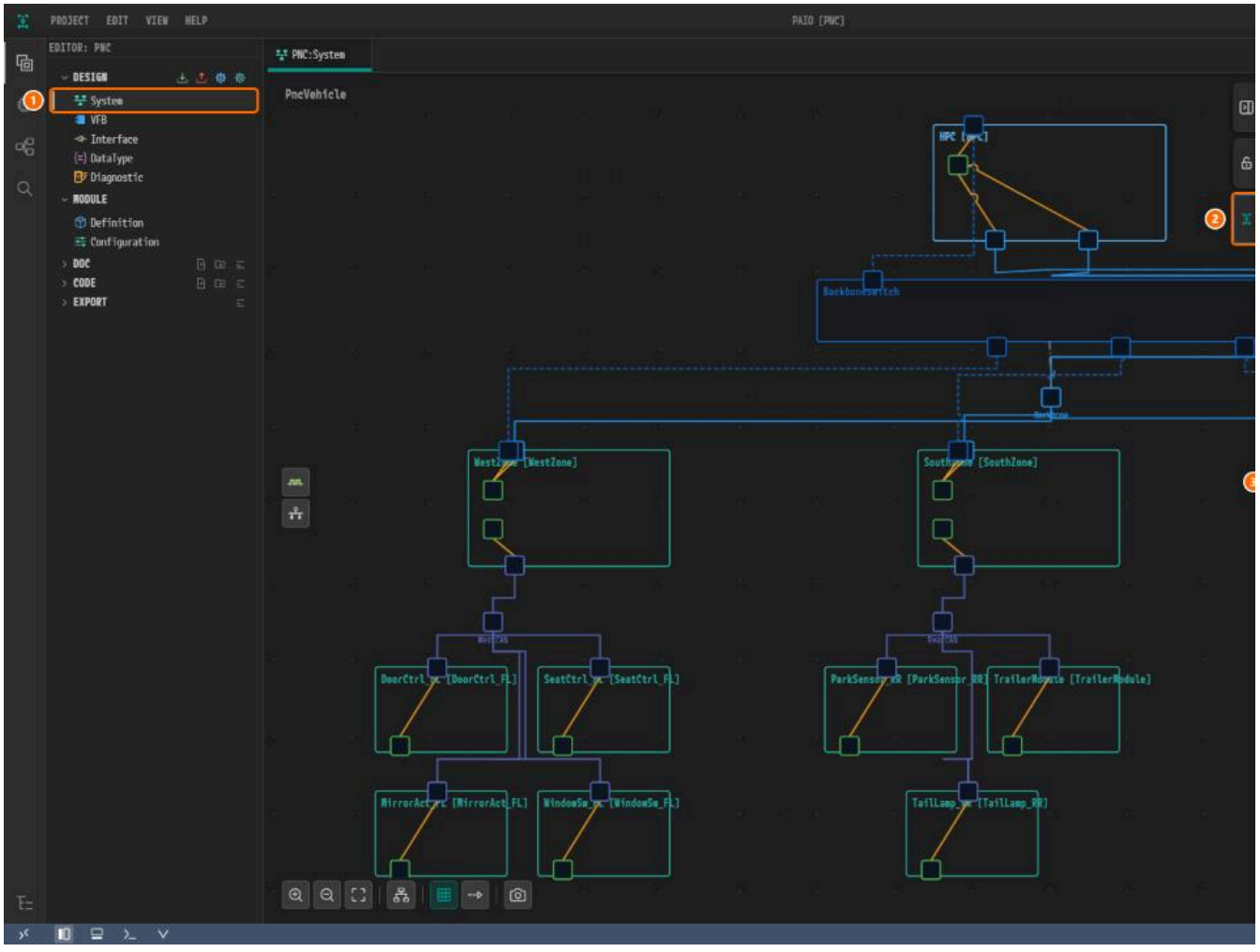


*그림 11-1. ① **DESIGN** 아래의 **System**, ② **System** 버튼, ③ **ECU MAPPING**.*

2. **ECU MAPPING**에 14행이 있고, 노드마다 같은 이름의 컴포지션이 매핑되어 있는지 확인한다.

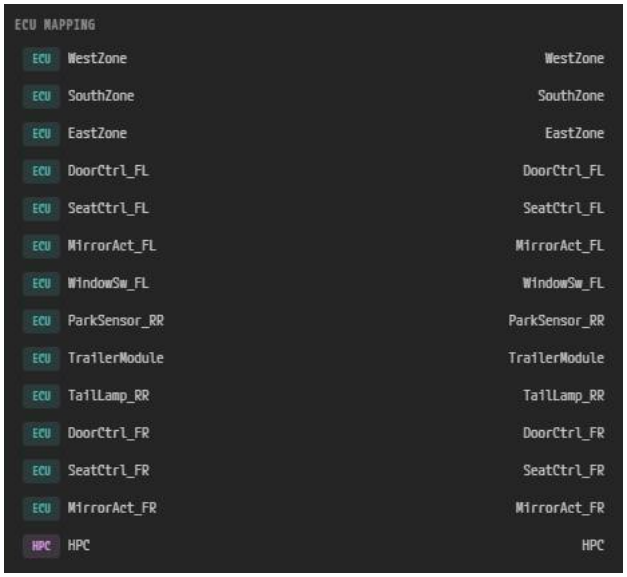


*그림 11-3. **ECU MAPPING: ECU** 13행과 **HPC** 1행.*

3. **DESIGN** 아래에서 **VFB**를 클릭하고(탭 `PNC:VFB`) 이 캔버스의 **System** 버튼을 클릭한다.

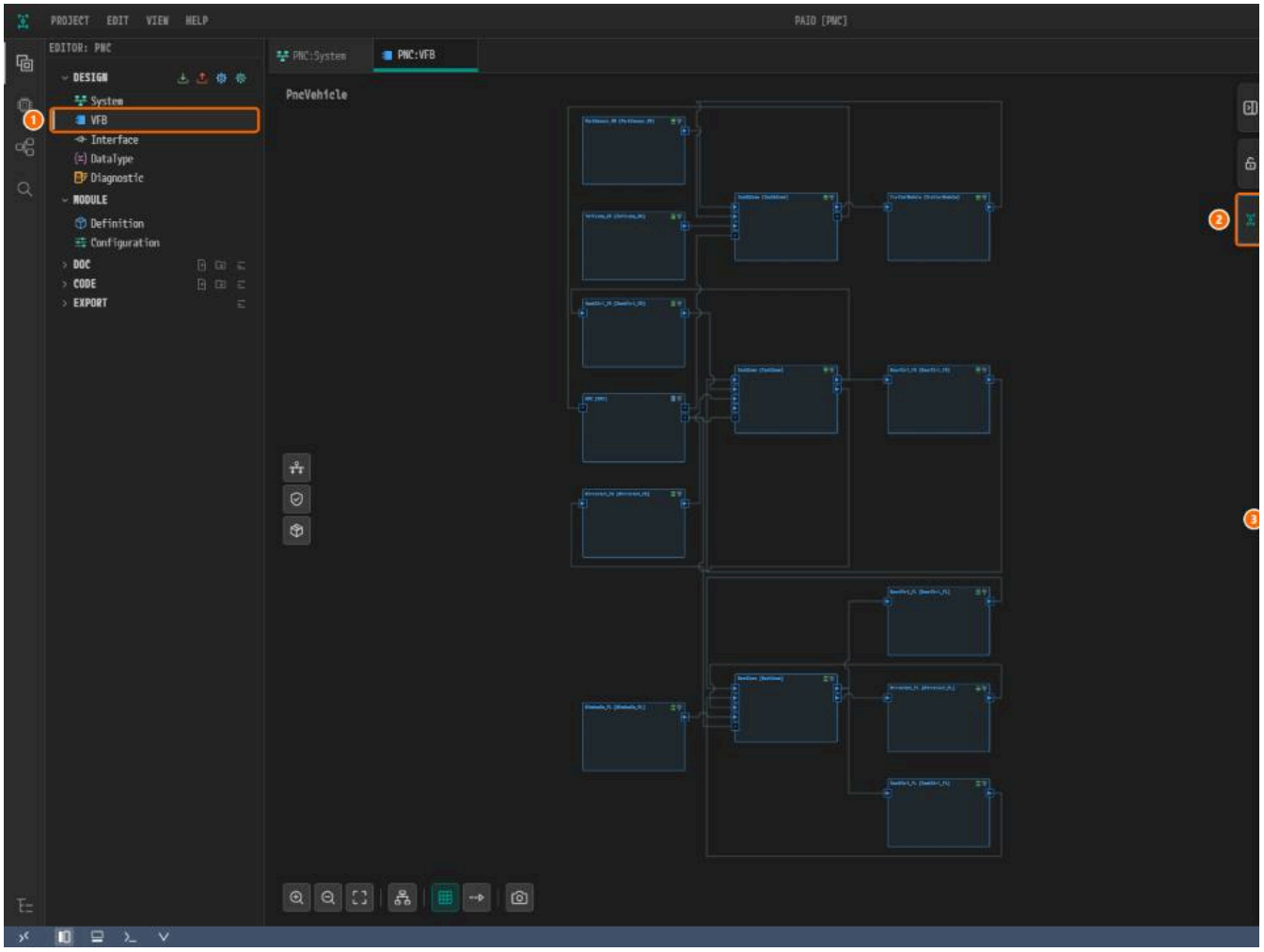


*그림 11-17. ① **VFB**, ② **System** 버튼, ③ **COMPOSITIONS**. 4단계를 마친 뒤의 모습이다.*

4. **COMPOSITIONS**에서 루트 `PncVehicle` 아래 컴포지션 14개를 접는다.

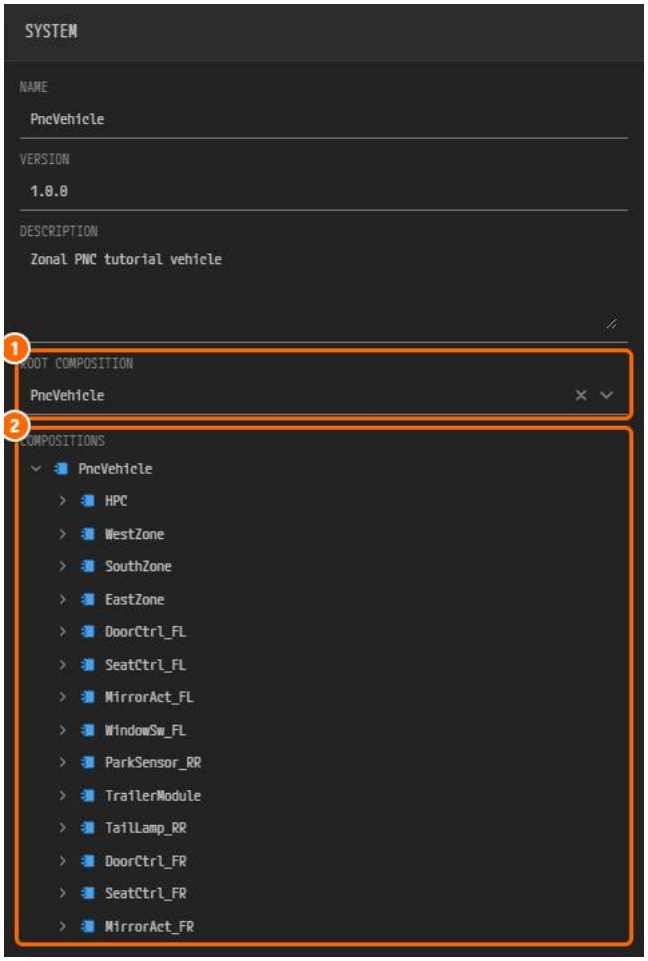


*그림 11-2. ① **ROOT COMPOSITION** `PncVehicle`, ② **COMPOSITIONS**.*

> **참고.** `PARA`는 `HPC` 안에 있으므로 **ECU MAPPING**에 따로 행이 없다.

검증 패널 열기

검증에는 실행 버튼이 없다. PAIO는 프로젝트를 열 때와 편집할 때마다 다시 검사한다.

1. 상태 표시줄 왼쪽 끝의 검증 아이콘(`V`, 툴팁 **Toggle Validation**)을 찾는다. 빨간 배지가 오류 건수이다.

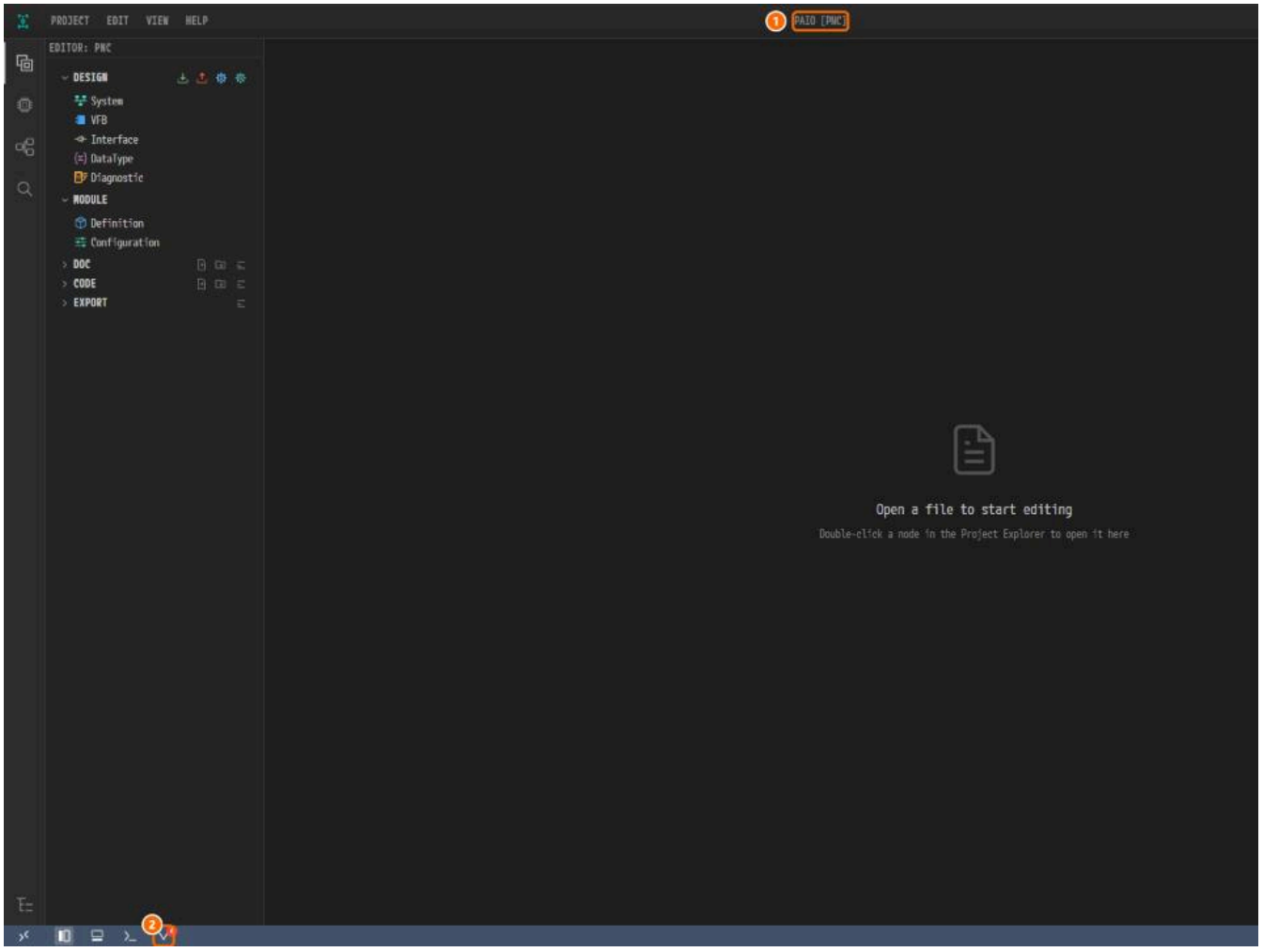


그림 11-4. 초안 상태. ① 제목 표시줄 `PAIO [PNC]`, ② 배지 `4`가 붙은 검증 아이콘.

2. 아이콘을 클릭한다. 편집기 아래에 **VALIDATION** 패널이 열린다.

검증 패널 읽기

한 행이 위반 하나이다. 행을 클릭하면 그 요소로 이동한다.

열	내용
ELEMENT TYPE	요소의 종류. 예) `SomeipServiceInstanceToMachineMapping`, `PhysicalChannel`, `PncMapping`
NAME	요소 이름
DESCRIPTION	무엇이 잘못되었는지
CONSTR	AUTOSAR 제약 번호. 예) `constr_5093`. PAIO 자체 규칙이면 비어 있다



*그림 11-5. PAIO 자체 규칙(**CONSTR** 비어 있음)의 오류 4건. ① `VehicleState_Consumer` 행.*

오류를 찾아 고치기: SOME/IP 포트 중복

초안에서는 소비자 2개가 커넥터의 제공자와 같은 UDP 포트를 썼다. 소비자를 `30503`으로 옮겨 고친다.

커넥터 (IP 주소)	인스턴스 (역할)	초안 UDP 포트	수정 후
`HPC_Eth_Rear_Conn` (`192.168.200.1`)	`VehicleState_Provider` (P)	`30501`	`30501` (그대로)
`HPC_Eth_Rear_Conn` (`192.168.200.1`)	`ParkDistance_Consumer` (C)	`30501`	**`30503`**
`SouthZone_Eth_Conn` (`192.168.200.12`)	`ParkDistance_Provider` (P)	`30502`	`30502` (그대로)
`SouthZone_Eth_Conn` (`192.168.200.12`)	`VehicleState_Consumer` (C)	`30502`	**`30503`**

오류 행에서 요소 찾기

1. `VehicleState_Consumer` 행을 클릭한다. `PNC:Interface` 탭이 열리고 카드에 빨간 표시가 붙는다.

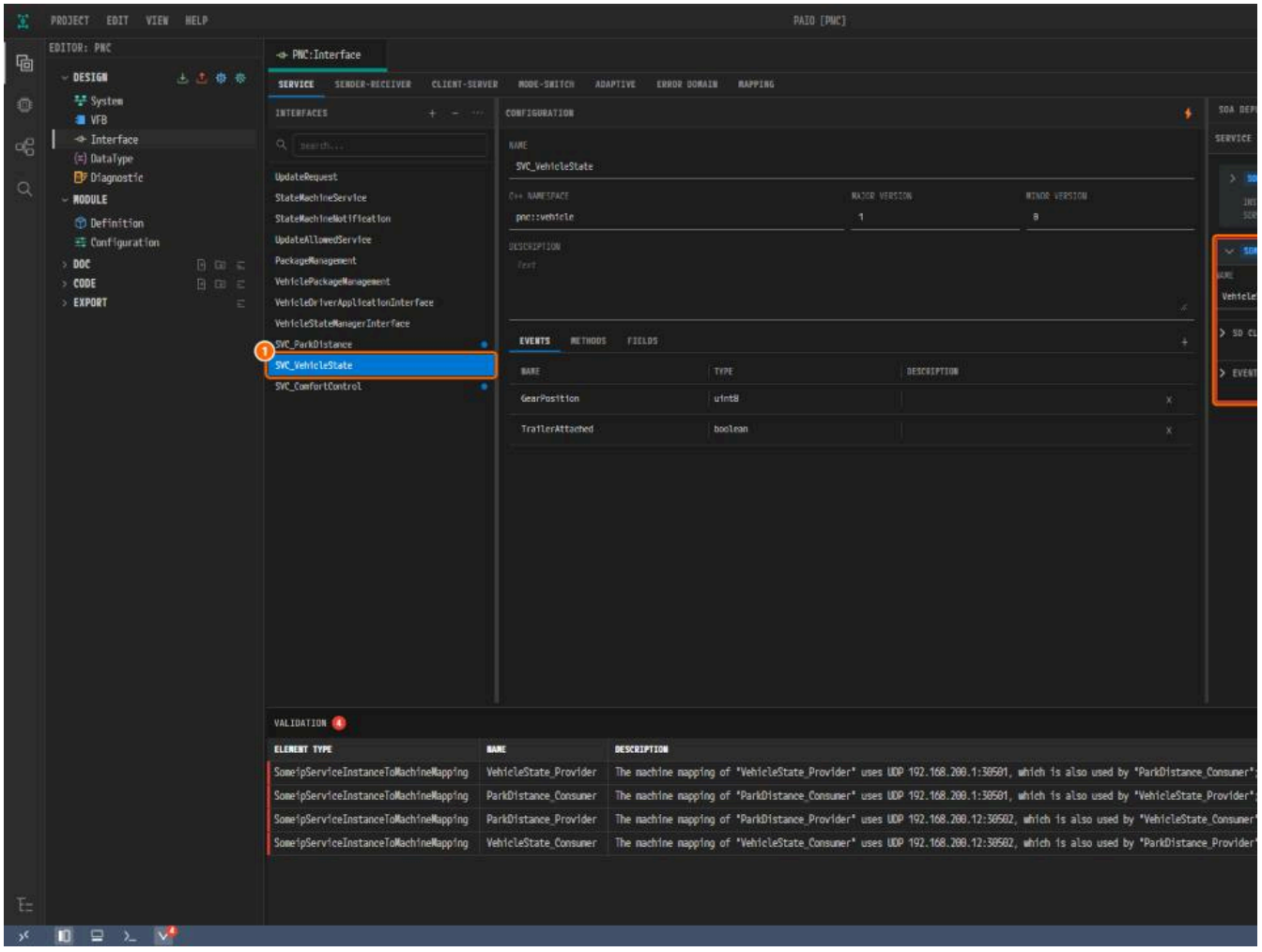


그림 11-6. ① `SVC_VehicleState`, ② `VehicleState_Consumer` 카드.

카드에는 포트 칸이 없다. 포트는 System 편집기의 커넥터에서 정한다.

SouthZone_Eth_Conn 포트 고치기

1. `PNC:System` 탭을 열고 `SouthZone` 위쪽의 Ethernet 커넥터(**NAME** `SouthZone_Eth_Conn`)를 클릭한다.
2. **SERVICE INSTANCES**에서 `VehicleState_Consumer`를 클릭해 펼친다.

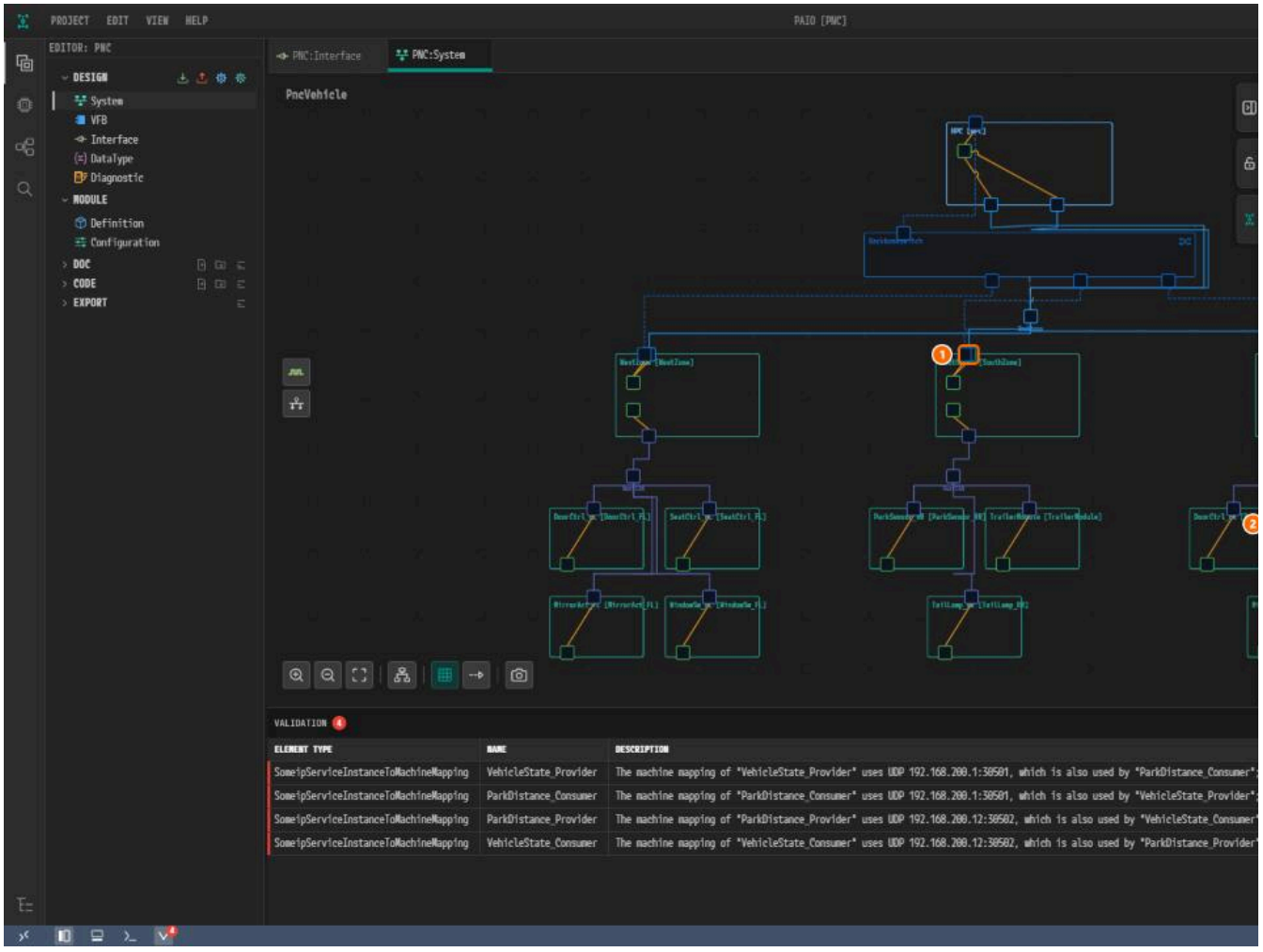


*그림 11-7. ① `SouthZone_Eth_Conn`, ② 둘 다 `UDP:30502`인 **SERVICE INSTANCES**, ③ **UDP PORT** `30502`.*

3. **UDP PORT**에 `30503`을 입력하고 **Tab**을 누른다. 배지가 `2`로 줄어든다.

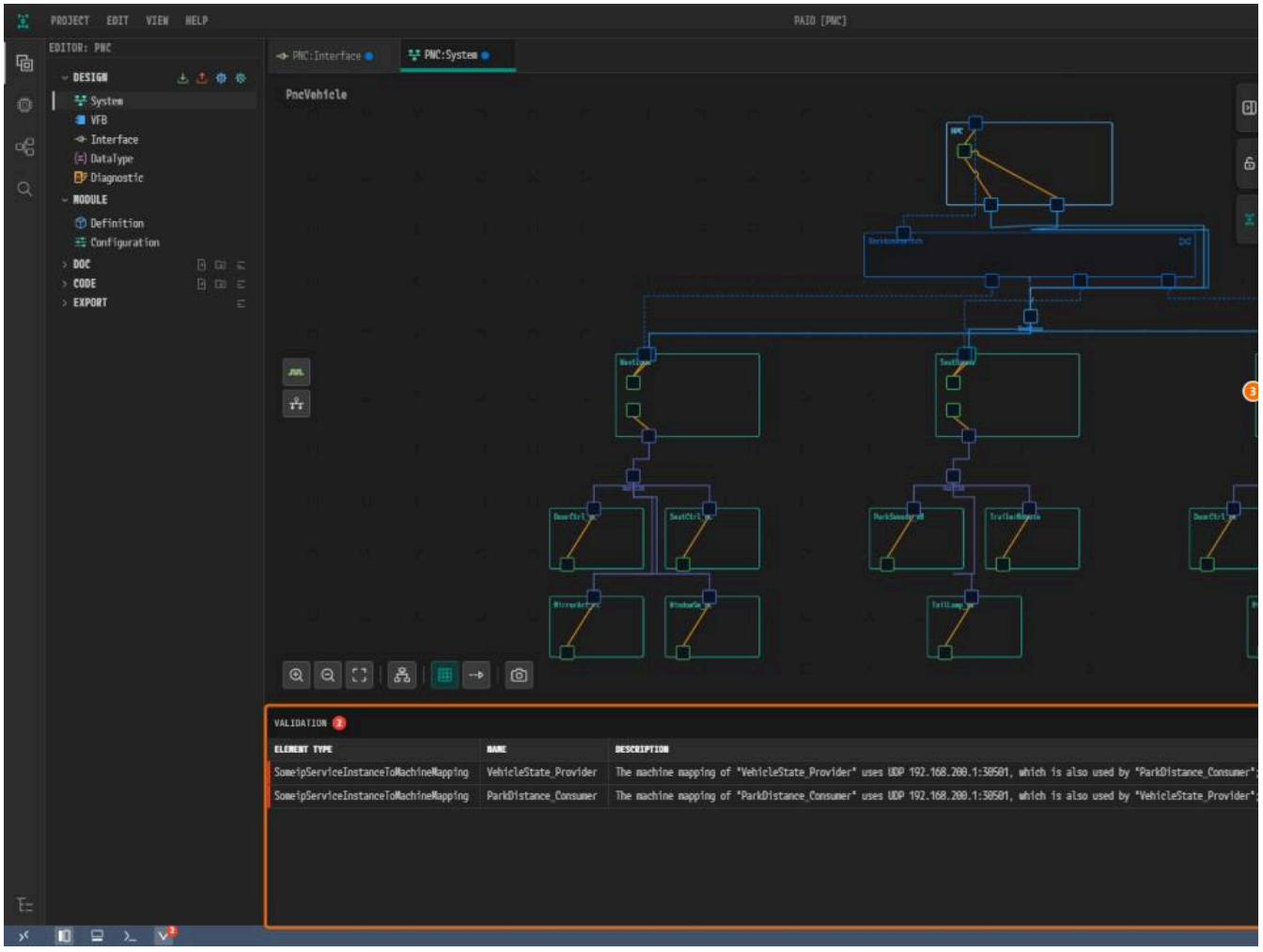


*그림 11-10. ① **UDP PORT** `30503`, ② 배지 `2`인 **VALIDATION** 패널, ③ 새 엔드포인트 `...:30503`.*

포트를 바꾸면 `PNC:Interface`에 파생 인터페이스가 추가되어 그 탭에도 **●**가 붙는다.

4. `Ctrl+S`를 누른 뒤 **PROJECT** > **Save All**을 선택해 `PNC:Interface`도 저장한다.

HPC_Eth_Rear_Conn 포트 고치기

1. `ParkDistance_Consumer` 행을 클릭한다.



그림 11-11. ① `ParkDistance_Consumer`.

2. `PNC:System`에서 `HPC` 아래쪽 가장자리의 오른쪽 커넥터(**NAME** `HPC_Eth_Rear_Conn`, **CONNECTED CHANNEL** `Backbone / VLAN_Rear (VLAN: 200)`)를 클릭한
3. `ParkDistance_Consumer`를 펼치고 **UDP PORT**에 `30503`을 입력한 뒤 **Tab**을 누른다. 패널에 "No validation issues."가 나온다.

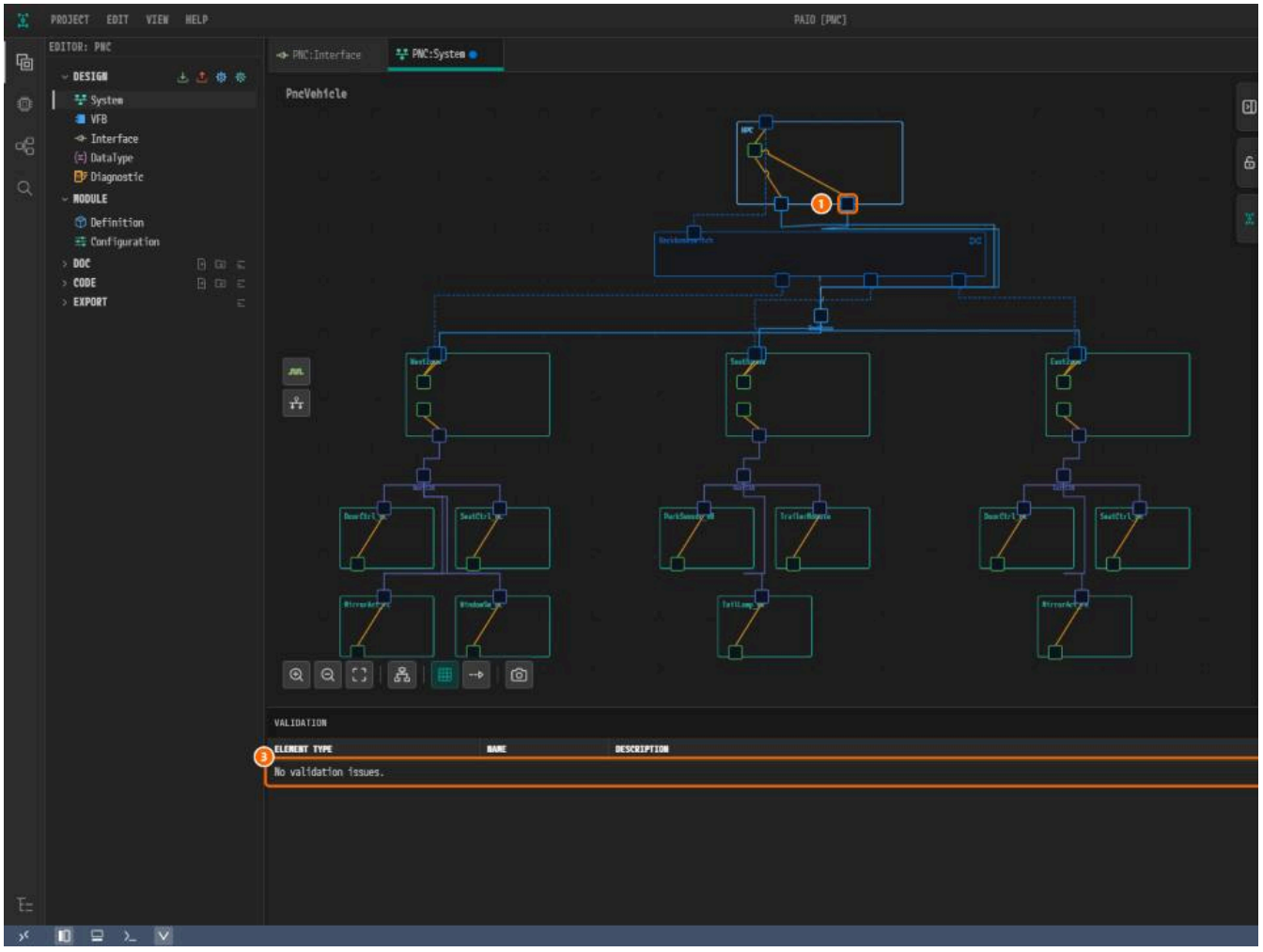


*그림 11-12. ① `HPC_Eth_Rear_Conn`, ② **UDP PORT** `30503`, ③ "No validation issues.". 요약 줄은 커넥터를 다시 선택할 때까지 `UDP:30501`이다.*

4. `Ctrl+S`를 누른다.

저장한 결과 다시 검증하기

1. 저장하면 다시 검사된다. "No validation issues."가 보이고 상태 표시줄 아이콘에 배지가 없는지 확인한다.

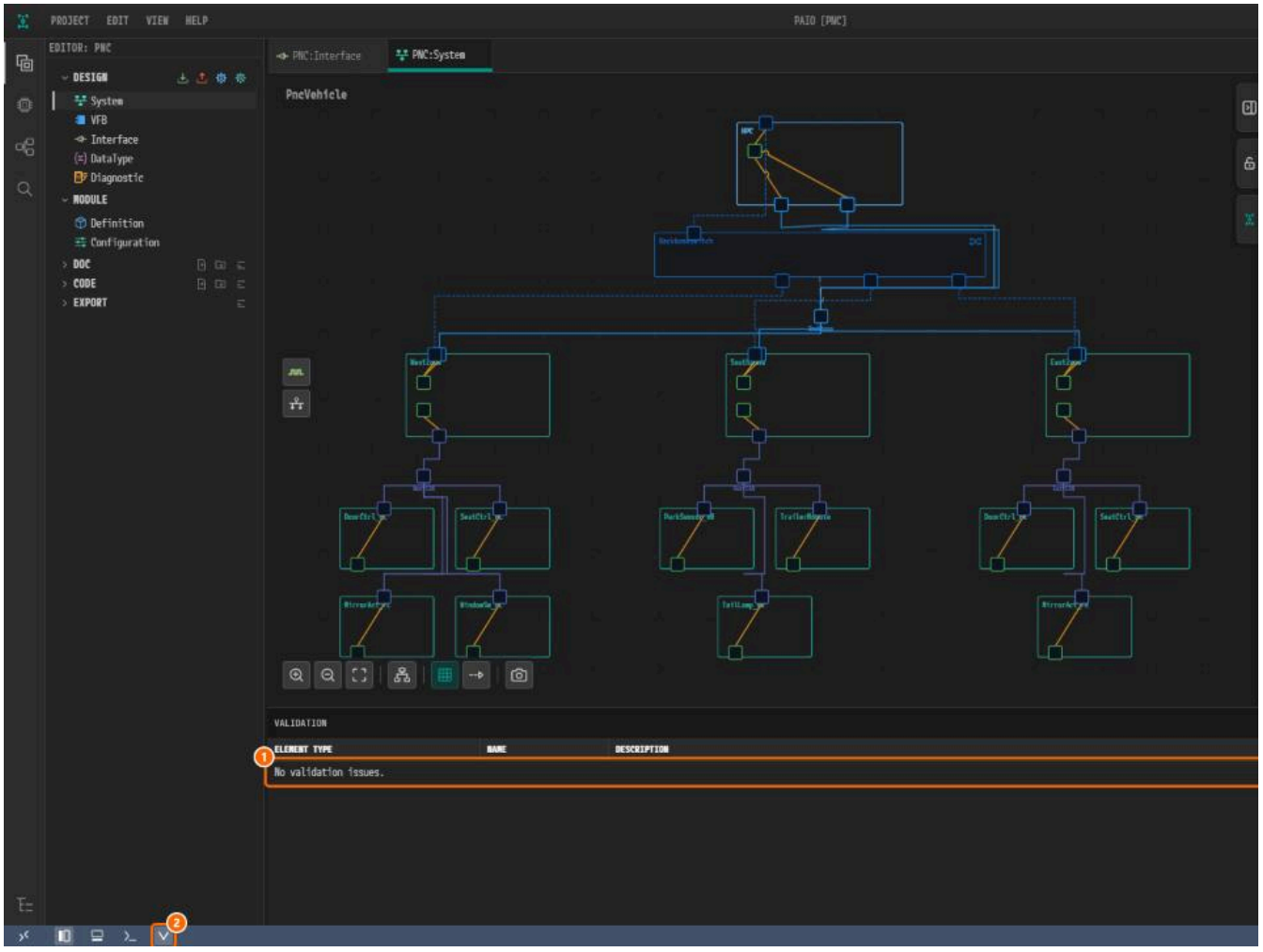


그림 11-13. ① "No validation issues.", ② 배지가 없는 아이콘.

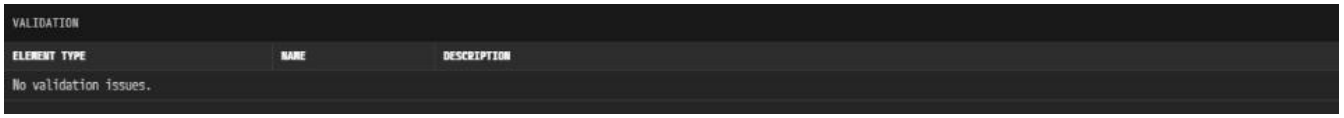


*그림 11-14. 오류 0건인 **VALIDATION** 패널.*

실시간 검증 확인하기: PNC GATEWAY TYPE

검증은 저장하지 않은 편집에도 반응한다. 설정 하나를 바꿔 오류를 보고 되돌린다. 저장하지 않는다.

1. `PNC:System`에서 `EastZone` 위쪽의 커넥터(**NAME** `EastZone_Eth_Conn`)를 클릭한다. **PARTIAL NETWORK**의 **PNC GATEWAY TYPE**은 **PASSIVE**이다.
2. **ACTIVE**를 클릭한다. 오류 2건이 나타난다.

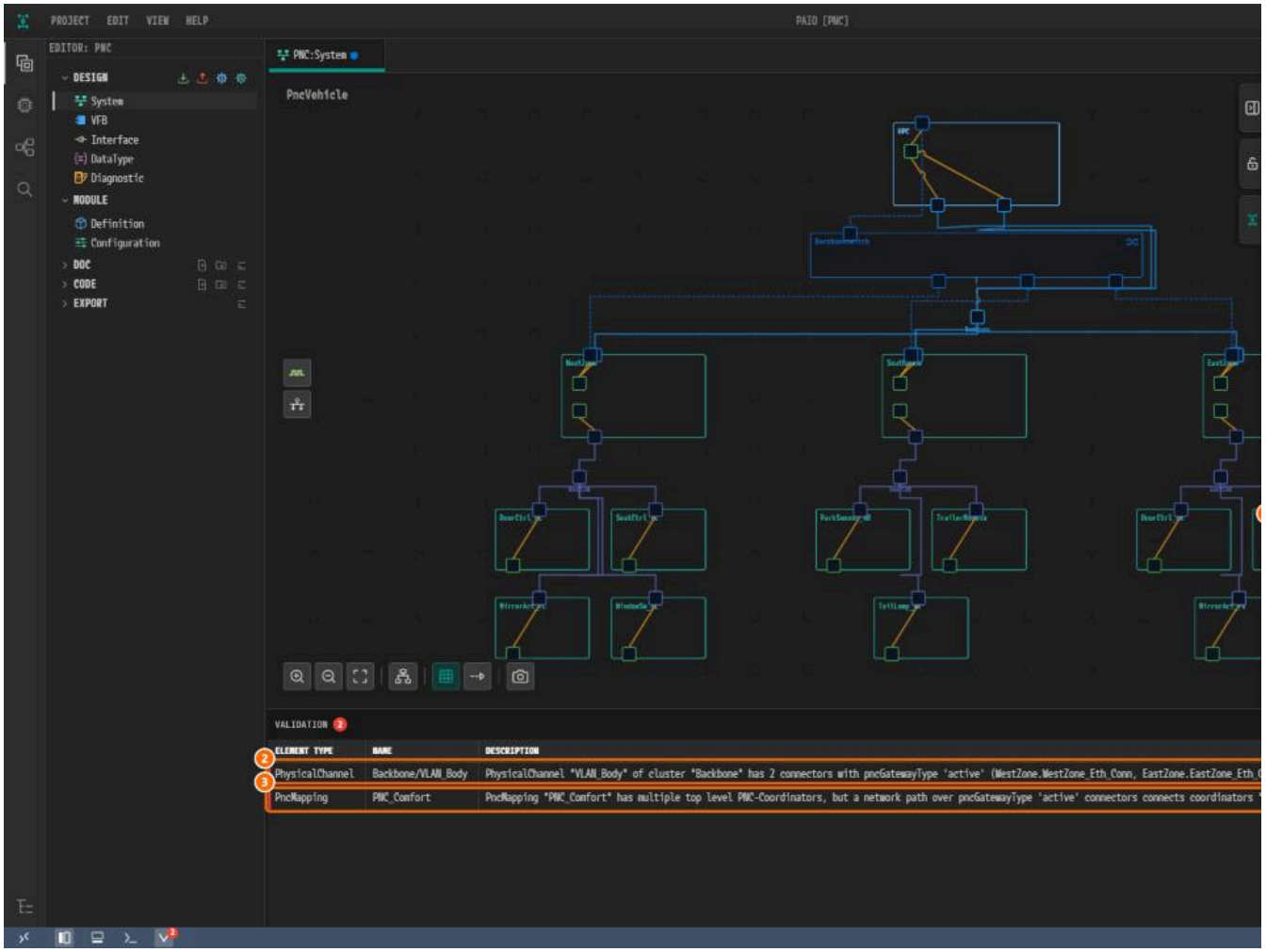


*그림 11-15. ① **ACTIVE**, ② `constr_5093: VLAN_Body`에 active 커넥터가 2개, ③ `constr_3714: PNC_Comfort`의 코디네이터 `WestZone`과 `EastZone`이 active 경*

3. **PASSIVE**를 클릭한다. 두 오류가 사라지지만 탭의 **●**는 남는다.

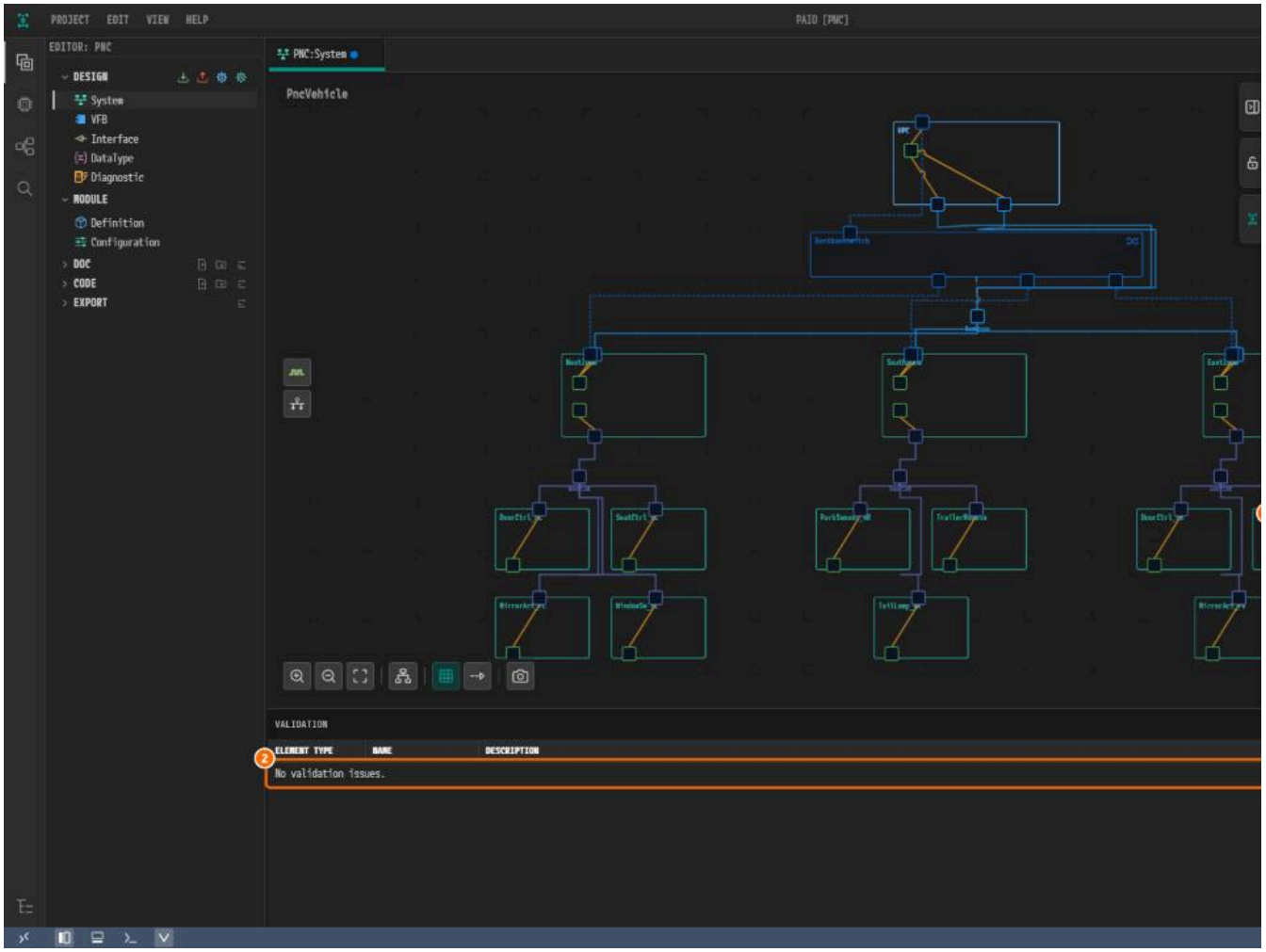


*그림 11-16. ① **PASSIVE**, ② "No validation issues.".*

4. `PNC:System` 탭의 ***로 탭을 닫고 **Don't Save**를 클릭한다.

PNC 설계와 관련된 주요 제약

완성 예제는 아래 제약을 모두 지킨다. 예제를 처음 만들 때 보고되었던 항목이다. 같은 오류가 보이면 관련 장을 다시 확인한다.

제약	검증 패널이 보고한 내용	수정
`constr_3040`	PNC 식별자 8/9/10이 PNC 벡터 비트 범위(16..63) 밖	식별자를 17/18/19로 변경
`constr_3686`	필터 마스크가 3바이트로, PNC VECTOR LENGTH 6과 다름	6바이트로 다시 계산
`constr_5093`	백본 채널에 active 게이트웨이 커넥터가 3개	백본을 `VLAN_Body` / `VLAN_Rear`로 나누고, VLAN마다 active 1개
`constr_3714` / `constr_3781`	top-level 코디네이터끼리 active 경로로 이어짐 / 코디네이터가 없음	`WestZone`, `SouthZone`은 양쪽 active(top-level), `EastZone`은 백본 쪽 passive
`constr_3514`	I-Signal(`LockCmd` 등) 하나가 PDU 두 개에 매핑됨	좌/우 신호를 `_FL` / `_FR`로 분리
`constr_9157` – `9159`, `constr_9162`	CanNm 클러스터의 필수 속성 누락	**CAN NM CLUSTER** 탭에서 `nmBusloadReductionActive`, `nmImmediateNmTransmis` `nmMessageTimeoutTime`, `nmRemoteSleepIndicationTime`을 채움
`constr_5222`	UdpNm 클러스터의 필수 속성 누락	**UDP NM CLUSTER** 탭에서 `nmMessageTimeoutTime`, `nmRemoteSleepIndicationT` `NmCluster_VLAN_Rear`는 화면에서 열 수 없다
`constr_9165` / `9166`	NM 클러스터 커플링의 필수 플래그 누락	커플링은 화면에서 고칠 수 없으며, 값은 ARXML 임포트로 들어온다
(PAIO NM 규칙)	NetworkHandle에 INACTIVE→FunctionGroup 상태 매핑이 없음 ("PNC release stops nothing")	`ParkAssistFG` / `TrailerFG`의 Off 매핑 추가
`constr_10641`	SM 모듈의 max-action-list-duration 누락	1.0 s 설정

왜 이렇게 설계하는가

- **검증은 막지 않는다.** 오류가 있어도 저장과 익스포트는 된다.
- **그래도 지적마다 실제 결함이다.** 익스포트한 ARXML을 ComM/Nm 설정 생성기 같은 다른 도구가 읽으면 잘못된 동작으로 이어지므로 "No validation issues."를 목표로 한다
- **스키마 검사는 따로 한다.** ARXML 스키마(XSD)는 익스포트 단계에서 검사한다.

확인하기

- [ ] **ECU MAPPING**에 노드 14개, **COMPOSITIONS**에 `PncVehicle` 아래 컴포지션 14개가 있다.
- [ ] **VALIDATION** 패널에 "No validation issues."가 보이고 상태 표시줄 아이콘에 배지가 없다.
- [ ] `HPC_Eth_Rear_Conn`의 `ParkDistance_Consumer`와 `SouthZone_Eth_Conn`의 `VehicleState_Consumer`가 `UDP:30503`이다.
- [ ] `EastZone_Eth_Conn`의 **PNC GATEWAY TYPE**이 **PASSIVE**이고, 어느 탭에도 **●**가 없다.

다음 단계

ARXML 익스포트에서 검증을 마친 설계를 ARXML로 내보내고 파일을 확인한다.

ARXML 익스포트

이 장에서 할 일

검증을 마친 설계를 AUTOSAR ARXML 로 내보내고, PNC 설계가 어디에 기록되는지 찾고, 스키마 검사 결과를 확인한다. 파일 5개 가운데 4개는 오류 0건이고, `PNC_COM.arxml` 에에서 12건).

시작하기 전에

- System 에디터와 검증을 마쳐 **VALIDATION** 패널이 "No validation issues." 상태여야 한다.
- 모든 탭을 `Ctrl+S` 로 저장해 ● 표시가 붙은 탭이 없게 한다.
- 익스포트에는 활성 라이선스와 서버 연결이 필요하다. 자세한 것은 Import and Export를 참고한다.

파일	PNC 관련 내용
`PNC_System.arxml`	`<PNC-MAPPING>` 3개, `<PNC-VECTOR-LENGTH>` `6` 과 `<PNC-VECTOR-OFFSET>` `2` (SYSTEM 에 한 번), SWC-ECU 매핑
`PNC_COM.arxml`	ECU 인스턴스와 머신 디자인: 커넥터 PNC 속성, 클러스터와 VLAN, `<I-SIGNAL-I-PDU-GROUP>` 20개, NM 설정, SOME/IP 서비스 인스턴스
`PNC_SWC.arxml`	SWC 타입과 VFC 의 `<PORT-GROUP>`, `<NM-NETWORK-HANDLE>` 3개, `<NETWORK-HANDLE-PORT-MAPPING>` 3개, PARA 플랫폼 AA
`PNC_Interface.arxml`	포트 인터페이스와 서비스 인터페이스. `<NETWORK-MANAGEMENT-PORT-INTERFACE>` 1개 포함
`PNC_DataType.arxml`	데이터 타입. PNC 전용 요소는 없다

개수는 완성 예제의 값이다. 직접 만든 프로젝트에서는 `I-SIGNAL-I-PDU-GROUP` 같은 일부 개수가 다르다. 부록 — 직접 만든 프로젝트와 완성 예제의 차이를 참고한다.

프로젝트를 ARXML로 익스포트하기

1. **EDITOR** 뷰에서 **DESIGN** 머리글 오른쪽의 두 번째 아이콘에 마우스를 올린다. 툴팁은 **Export** 이다.

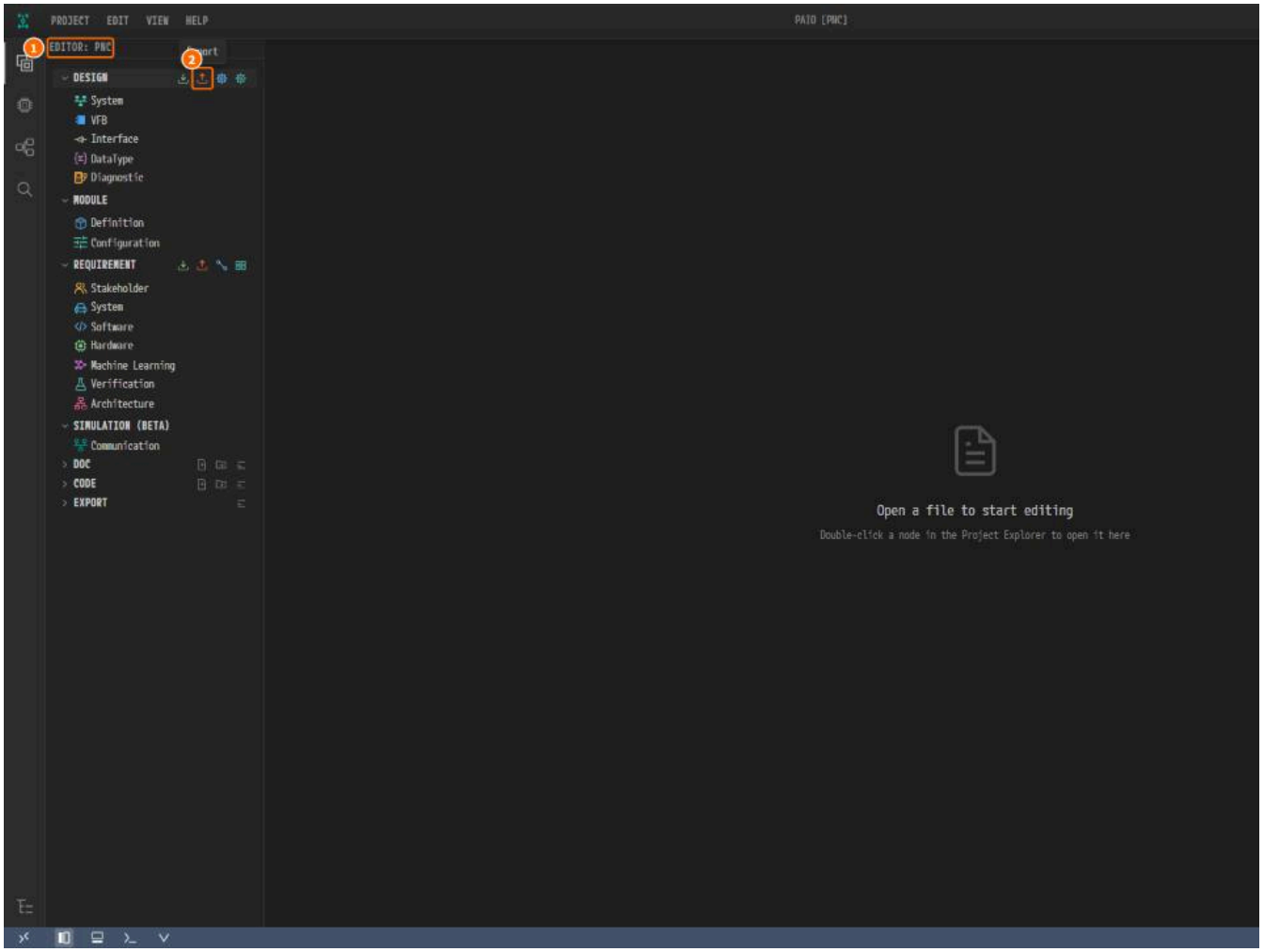


*그림 12-1. ① `EDITOR: PNC`, ② **Export**.*

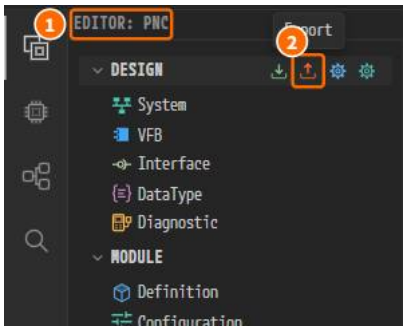


*그림 12-2. ① `EDITOR: PNC`, ② **Export**. 나머지 아이콘은 **Import**, **PRTE Code Generation**, **PARA Code Generation** 이다.*

2. **Export** 를 누른다. **Export** 창이 열린다. **FOLDER** 는 `export`, **FORMAT** 은 **ARXML (R25-11)** 그대로 두고 **Export** 를 누른다.



*그림 12-3. ① **FOLDER**, ② **FORMAT**, ③ **Export**.*

3. 창에 `✓ Complete!` 와 **ARXML** `R25-11, 298 elements` 줄이 표시된다. 요소 수는 모델에 따라 다르다. **Close** 를 누른다.

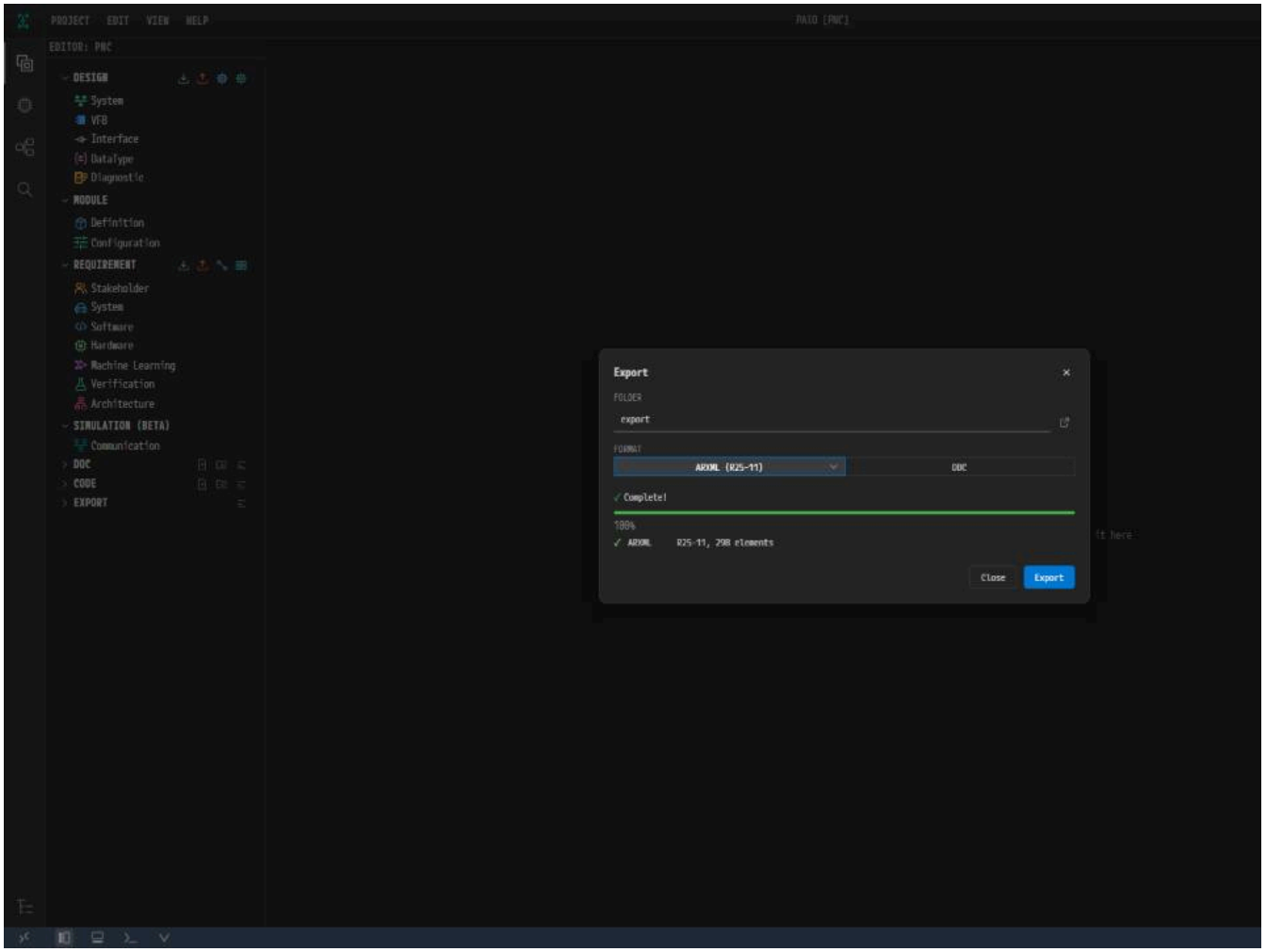


그림 12-4. 익스포트가 끝난 순간의 화면 전체.

> **주의.** 익스포트가 실패하면 `X Finished with errors` 가 표시되고 **ARXML** 줄에 사유가 보인다. `OFFLINE_EXPORT_DENIED` 는 서버 연결이 필요하다는 뜻이고, `PLAN_EX`

만들어진 파일 확인하기

1. **EDITOR** 뷰의 **EXPORT** 머리글을 누른다. `PNC_COM.arxml`, `PNC_DataType.arxml`, `PNC_Interface.arxml`, `PNC_SWC.arxml`, `PNC_System.arxml` 5개가 보인다.

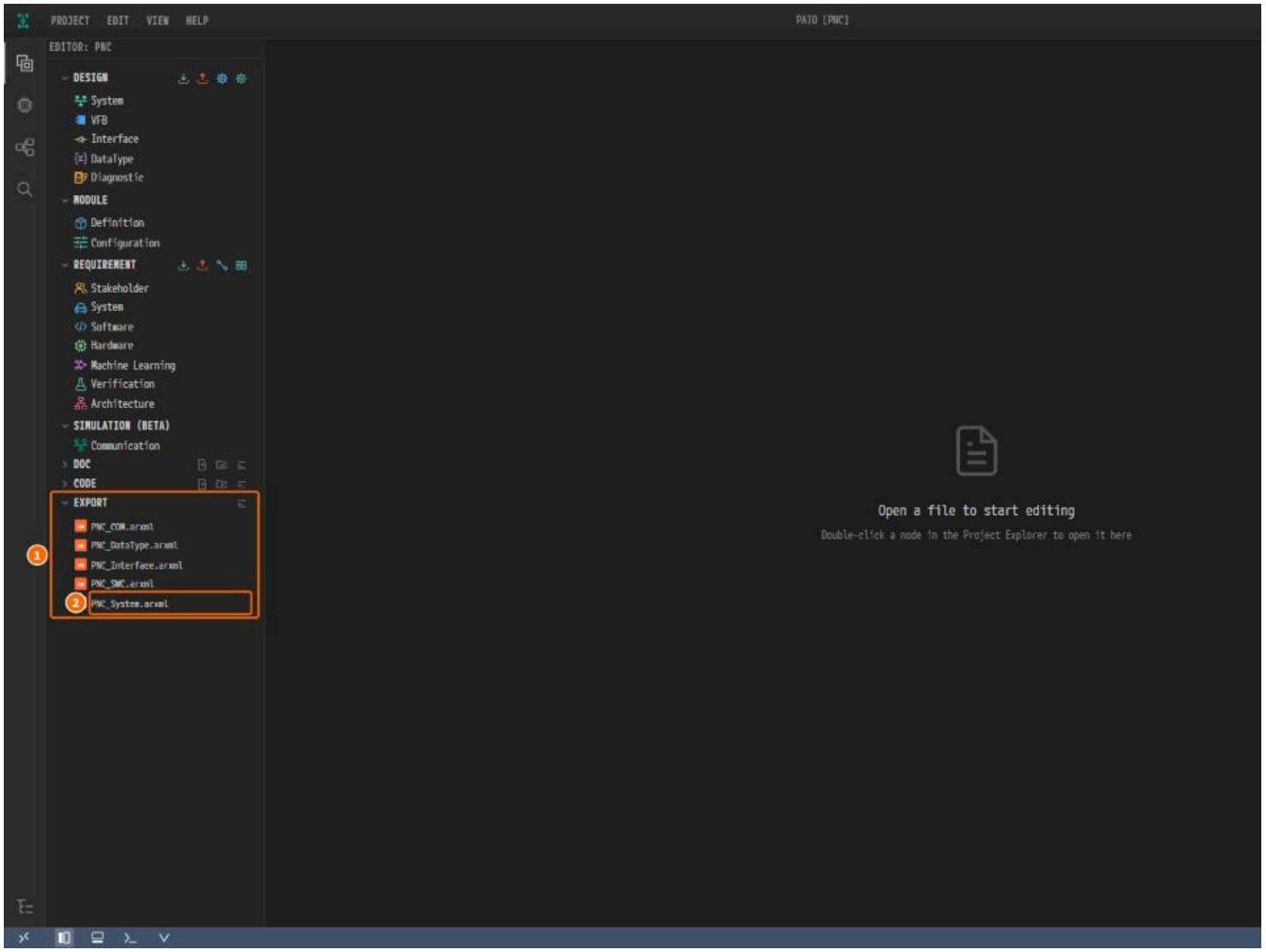


*그림 12-5. ① **EXPORT** 와 파일 5개, ② `PNC_System.arxml`.*

PNC_System.arxml에서 PNC-MAPPING 읽기

1. **EXPORT** 에서 **PNC_System.arxml** 을 더블클릭한다. `PNC:PNC_System.arxml` 탭이 열린다.
2. `Ctrl+F` 를 누르고 `<PNC-MAPPING>` 을 입력한다. 찾기 창에 `1 of 3` 이 표시되고 매핑 `PNC_Comfort` 로 이동한다.

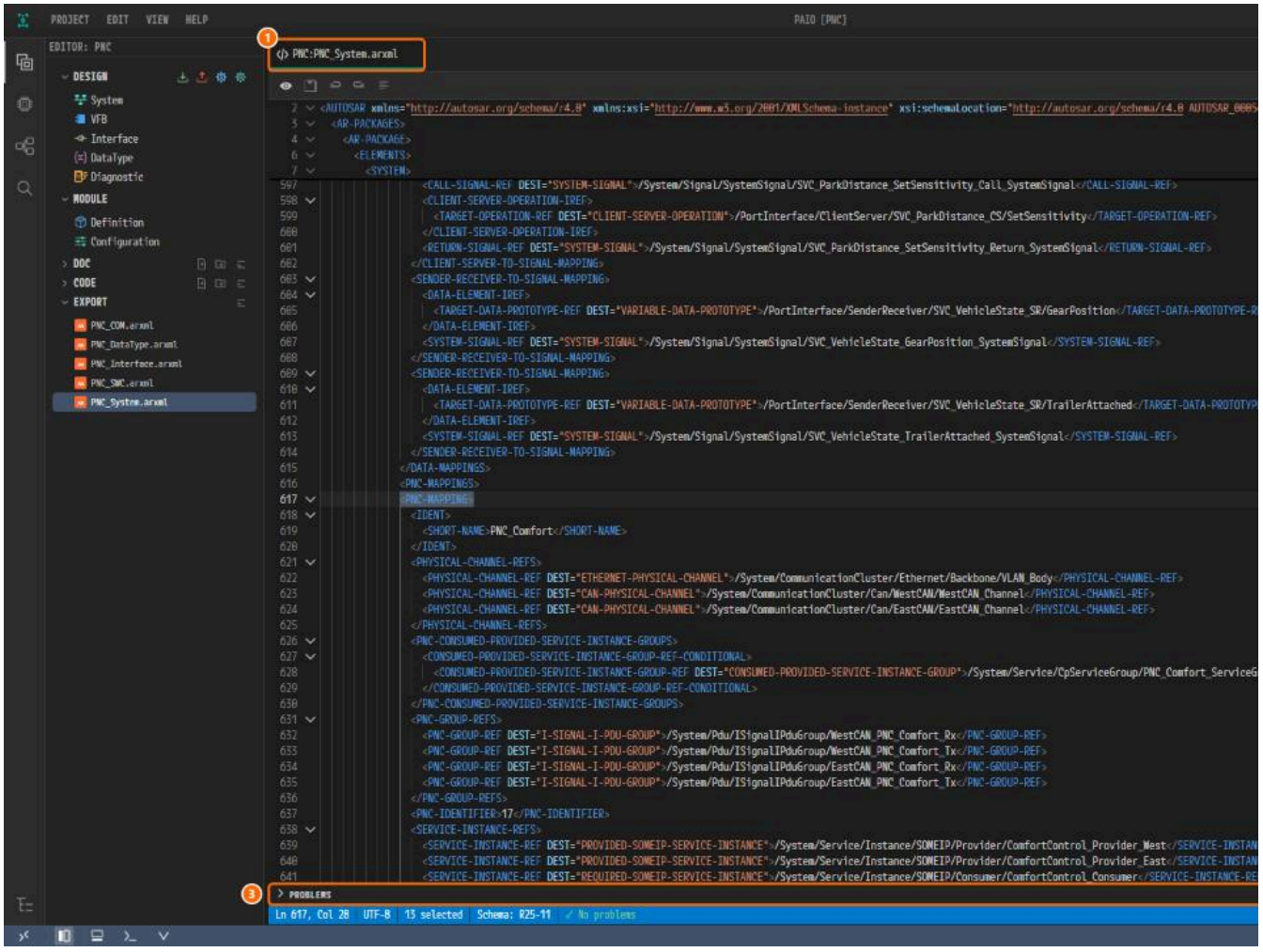


그림 12-6. ① `PNC:PNC_System.arxml`, ② `1 of 3` 이 보이는 찾기 창, ③ **PROBLEMS** 패널. 상태 표시줄에 `Schema: R25-11` 과 `✓ No problems` 가 보인다.

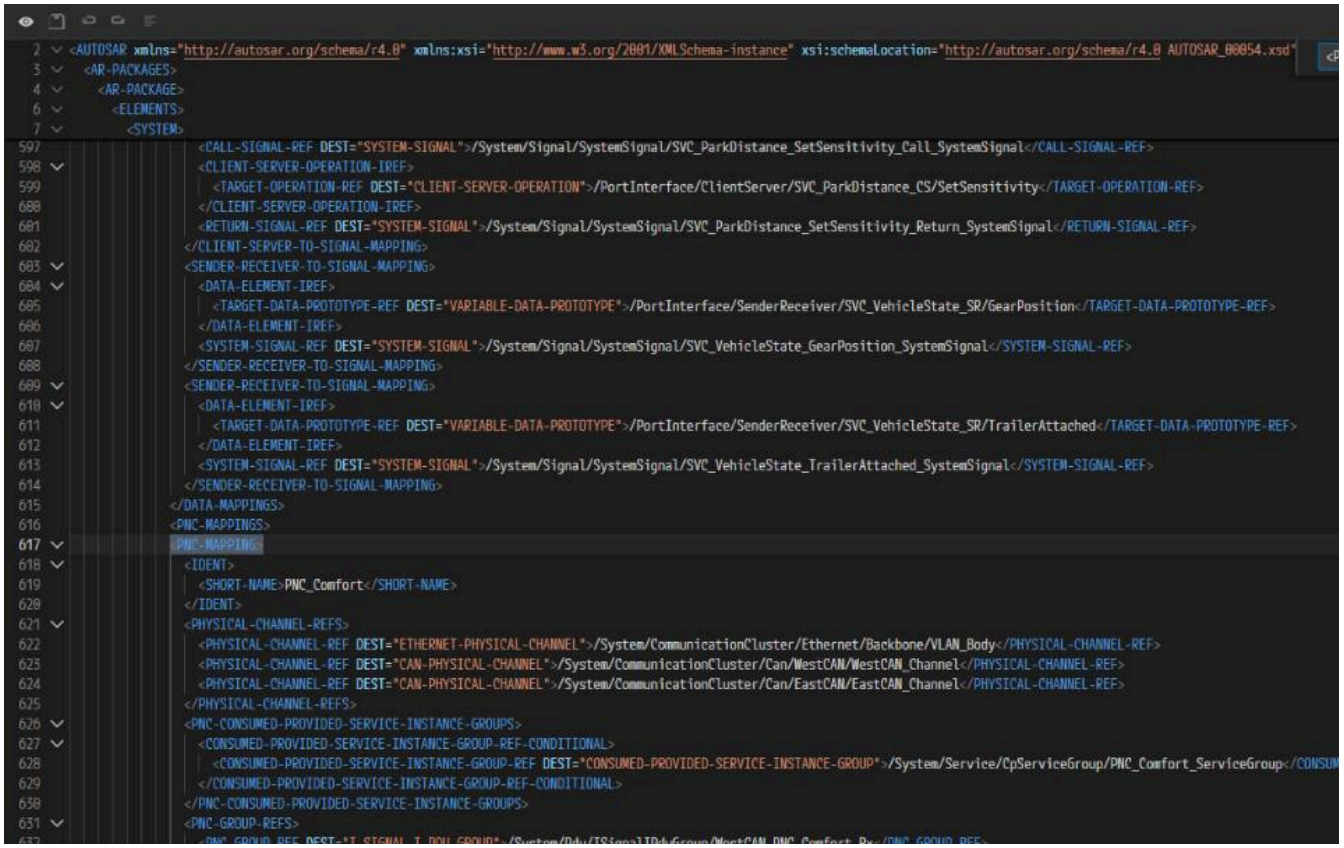


그림 12-7. 첫 `<PNC-MAPPING>` 블록 `PNC_Comfort`.

3. 블록을 그것을 만든 장과 대조한다.

요소	값	장
`PHYSICAL-CHANNEL-REFS`	`VLAN_Body`, `WestCAN_Channel`, `EastCAN_Channel`	토폴로지 설계
`PNC-IDENTIFIER`	`17`	PNC 정의와 커넥터 PNC 속성
`PNC-GROUP-REFS`	I-PDU Group 4개	신호·PDU·프레임과 I-PDU Group
`SERVICE-INSTANCE-REFS`	SOME/IP 인스턴스 3개	인터페이스와 SOME/IP 서비스
`VFC-IREFS`	`VFC_Comfort` (`PORT-GROUP`)	VFB와 VFC
`WAKEUP-FRAME-REFS`	`FT_WestCAN_NmFrame_DoorCtrl_FL`, `FT_EastCAN_NmFrame_DoorCtrl_FR`	신호·PDU·프레임과 I-PDU Group

4. 찾기 창의 ↓ 로 `PNC_ParkAssist` 와 `PNC_Trailer` 를 이어서 본다.

스키마 검사 결과 확인하기

XML 편집기는 파일을 열 때 AUTOSAR XSD 로 검사한다.

1. 편집기 아래의 **PROBLEMS** 머리글을 누른다. `PNC_System.arxml` 의 표에는 `No problems found` 만 있다.

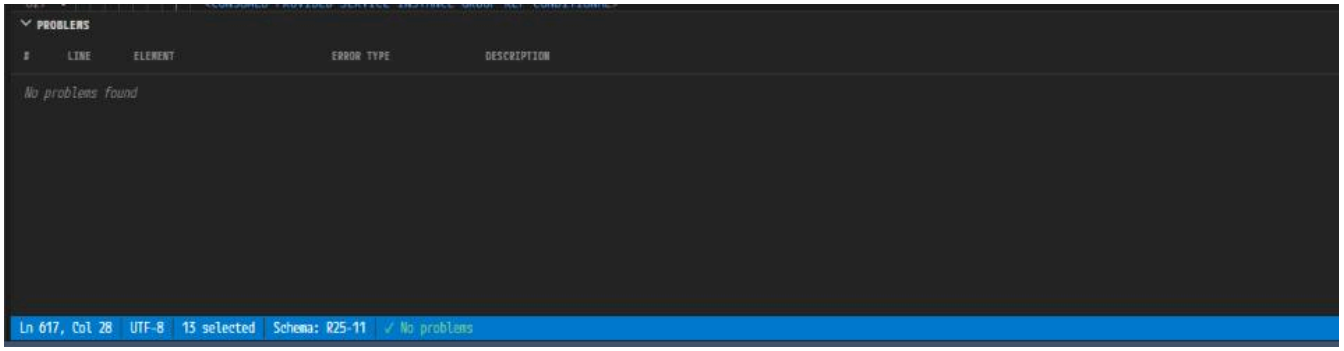


*그림 12-8. `No problems found` 가 있는 **PROBLEMS** 와 상태 표시줄의 `Schema: R25-11` · `✓ No problems`.*

2. 나머지 4개 파일도 같은 방법으로 연다. `PNC_DataType.arxml`, `PNC_Interface.arxml`, `PNC_SWC.arxml` 은 오류 0건이다. `PNC_COM.arxml` 은 **PROBLEMS** 패널이

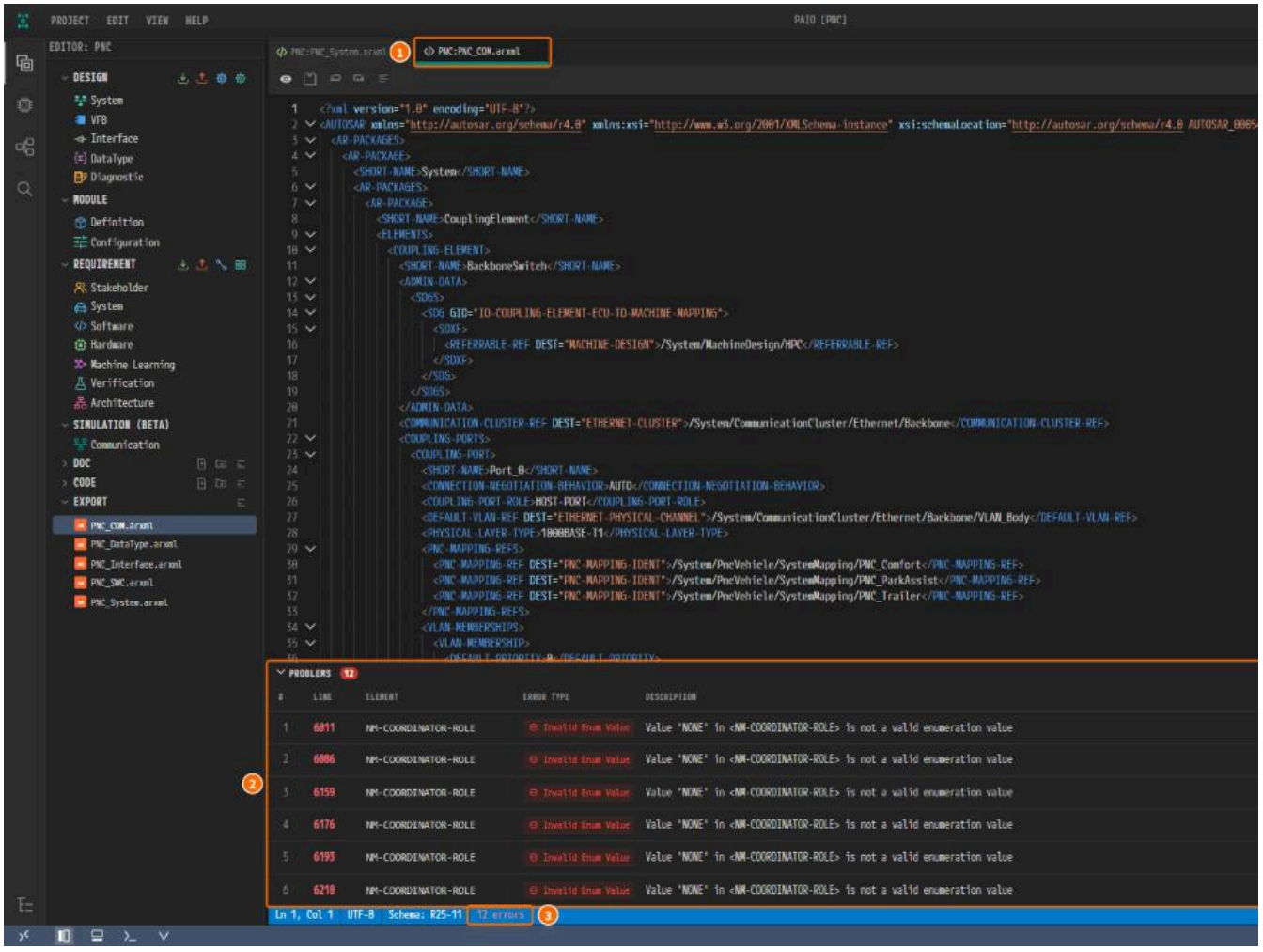


*그림 12-9. ① `PNC:PNC_COM.arxml`, ② 배지 `12` 가 붙은 **PROBLEMS**, ③ `12 errors`.*

> **참고.** 12건은 입력 실수가 아니라 익스포트에서 생긴다. **NM COORDINATOR ROLE** 이 **NONE** 인 NM 노드(NM 설정)의 `NM-COORDINATOR-ROLE` 이 `NONE` 으로 기록된 것이다.

다시 임포트해 확인하기

이 라운드트립 확인은 선택 사항이다.

1. **PROJECT** > **New Project...** 로 빈 프로젝트를 만든다. 새 창에서 열린다.
2. **DESIGN** 머리글 오른쪽의 첫 번째 아이콘 **Import** 를 누른다.
3. `PNC/export` 의 ARXML 5개를 모두 고른다.
4. 임포트가 끝나면 **DESIGN** 아래의 **System** 을 열고 PNC 3개, 식별자 `17` / `18` / `19`, 커넥터의 **PNC GATEWAY TYPE** 이 그대로인지 확인한다.

> **주의.** 같은 프로젝트에 임포트하면 덮어쓰지 않고 병합하며, 값이 다르면 **Conflict Resolution** 다이얼로그가 뜬다.

왜 이렇게 설계하는가

- **역할별 파일 5개.** 데이터 타입과 인터페이스는 공유하는 정의, SWC 는 소프트웨어 구조, COM 은 ECU 와 네트워크 배치, System 은 둘을 잇는 매핑이다.
- **PNC 설계가 나뉜다.** PNC 가 어느 채널·PDU 그룹·서비스 인스턴스·VFC 를 묶는지는 매핑이므로 System 에, 게이트웨이 종류와 필터 마스크는 ECU 배치이므로 COM 에 들어간
- **스키마 검사와 검증은 다르다.** 스키마 검사는 XSD 문법을 보고, **VALIDATION** 패널은 AUTOSAR 제약(`constr_*`)을 본다.

확인하기

- [ ] **EXPORT** 에 `PNC_*.arxml` 5개가 있다.
- [ ] `PNC_System.arxml` 에서 `<PNC-MAPPING>` 을 찾으면 `1 of 3` 이 보이고, 첫 매핑은 식별자 `17` 의 `PNC_Comfort` 다.
- [ ] `PNC_System.arxml`, `PNC_DataType.arxml`, `PNC_Interface.arxml`, `PNC_SWC.arxml` 은 오류 0건이다.
- [ ] `PNC_COM.arxml` 에는 익스포트에서 생기는 오류만 있다(완성 예제에서 12건).

다음 단계

부록에 설계 체크리스트, 프로젝트 파일 구조, 완성 예제와의 차이가 있다.

부록

이 장에서 할 일

이 부록은 따라 하는 장이 아니라 참고 자료이다. PNC 설계를 점검할 때, 값이 어디에 저장되는지 볼 때, 화면이 기대와 다르게 보일 때 쓴다.

설계 체크리스트

각 항목 뒤의 링크는 그 항목을 다룬 장이다.

- [ ] PNC 식별자가 `pncVectorOffset × 8` 이상, `(pncVectorOffset + pncVectorLength) × 8 - 1` 이하인가(`constr_3040`). 1장, 3장
- [ ] 모든 커넥터의 PNC 필터 배열 마스크 길이가 `pncVectorLength` 와 같은가. 3장
- [ ] 물리 채널마다 `pncGatewayType` 이 active 인 커넥터가 최대 1개인가(`constr_5093`). 3장, 11장
- [ ] PNC 마다 최상위 PNC 코디네이터, 곧 관련 채널 2개 이상에 연결되고 그 연결이 모두 active 인 ECU 가 있는가(`constr_3781`). 2장
- [ ] PNC 마다 Rx/Tx 쌍의 I-PDU Group 이 **PNC GROUPS** 에 있는가. 4장, 7장
- [ ] PNC 가 지나는 버스마다 **NM PNC PARTICIPATION** 이 **YES** 이고 NM 타이밍 필수 속성(`constr_9157`, `constr_9158`, `constr_9159`, `constr_9162`, UdpNm 은 `const`
- [ ] **NM NODE ID** 가 클러스터 안에서 겹치지 않고, **TX-NM-PDU-REFS** 가 자기 NM-PDU 1개(`constr_9388`), **RX-NM-PDU-REFS** 가 클러스터의 나머지인가. 3장, 4장
- [ ] VFC 이름이 원자 SWC, ECU 컴포지션, 루트 컴포지션에서 모두 같은가. 6장
- [ ] NM 네트워크 핸들마다 자기 PNC 를 참조하고 ACTIVE 와 INACTIVE 양쪽 FunctionGroup 상태 매핑을 갖는가. 9장
- [ ] 요청하는 응용 AA 가 `NetworkManagementPortInterface` P-Port 와 NetworkHandle 매핑을 갖고, PARA 플랫폼이 임포트되어 있는가. 10장
- [ ] 웨이크업 프레임이 버스에 실제로 있는 프레임을 가리키는가. 7장
- [ ] **VALIDATION** 이 0건이고, `PNC_COM.arxml` 의 익스포트 오류를 빼면 스키마 검사를 통과하는가. 11장, 12장

프로젝트 파일 구조

설계 파일은 모두 `design/` 아래에 있다.

```
PNC/
├ PNC.io.prj
└ design/
   ├ IVC.io.model            토폴로지, 커넥터 PNC 속성, 스위치 포트 PNC 참조, NM 설정
   ├ SIGNAL.io.model         신호, PDU, 프레임, NM-PDU, I-PDU Group
   ├ SYSTEM.io.model         PNC 매핑, PNC 벡터 오프셋과 길이, SWC-ECU 매핑
   ├ INTERFACE.io.model      SR 인터페이스, 서비스 인터페이스, NM 인터페이스
   ├ SERVICE.io.model        SOME/IP 배포, 제공자와 소비자 인스턴스
   ├ VFB.io.model            컴포지션 목록
   ├ DATATYPE.io.model       데이터 타입
   ├ SWC/
   │  ├ Application/                원자 SWC 9개. PortGroup(= VFC)이 여기 들어간다.
   │  ├ AdaptiveApplication/        AP 앱 3개. PARA 플랫폼 AA는 하위 폴더 HPC/에 들어간다.
   │  ├ Composition/                노드 컴포지션 14개, 루트 PncVehicle, PARA_HPC
   │  └ Layout/                     VFB 캔버스 배치(뷰 상태)
   └ Module/
      └ Machine/
         └ HPC.machine.io.model     FunctionGroup, NM 네트워크 핸들, AP 모듈 설정
```

프로젝트 폴더에는 이 밖에 `code`, `doc`, `export`, `import`, `workflow` 가 있다. ARXML 파일은 `export` 아래에 생긴다.

모델 조각

화면에 바로 보이지 않는 완성 예제의 값이다. 이 파일을 손으로 고칠 일은 없다.

파일	필드	값	
`SYSTEM.io.model`	`pnc-vector-offset`, `pnc-vector-length`	`2`, `6`	예제와 설계 목표의 식별자 범위
`IVC.io.model` (`SouthZone_CanFd_Conn`)	`pnc-filter-array-masks`	`[12, 0, 0, 0, 0, 0]`	길이 6 = `pnc-vector-length`. `12` 는 `000011(` `19`
`IVC.io.model` (`SouthZone_CanFd_Conn`)	`pnc-wakeup-can-id`, `pnc-wakeup-can-id-mask`	`1280`, `1792`	`0x500` 과 `0x700`: `0x500` 부터 `0x5FF` 까지가 웨
`IVC.io.model` (`SouthZone_CanFd_Conn`)	`pnc-mappings`	PNC 18, 19	이 커넥터가 참여하는 PNC. 화면에 편집란이 없다
`HPC.machine.io.model` (`NH_ParkAssist`)	`function-group-state-mappings`	`ParkAssistFG` `Active`, `Off`	`Active` 가 되면 PNC 를 요청하고, PNC 가 내려가면

직접 만든 프로젝트와 완성 예제의 차이

현재 버전에는 몇몇 값을 만드는 화면이 없어서, 토폴로지 설계부터 따라 만든 프로젝트는 완성 예제와 다르다. 어느 행도 검증 항목을 늘리지 않는다.

항목	완성 예제	따라 만든 프로젝트	
CAN NM 프레임	NM 프레임 13개, CAN 프레임 29개	NM 프레임 없음, 응용 프레임 16개. CAN 프레임에 NM-PDU 를 실을 수 없다	NM 통신 를 고를
NM 클러스터 커플링	UdpNm 커플링 1개, CanNm 커플링 1개	없음: 화면이 없다	`PNC_CO` `CLUSTE`
컨테이너와 NoPnc I-PDU Group	20개: PNC 리프 8개, 컨테이너 6개, NoPnc 6개	PNC 리프 8개만: 화면이 없다	`PNC_CO` 8개다. A
`tp-config` 와 VLAN 설정	`tp-config` 없음, `Backbone` 에 `vlan-configs` 있음	빈 `tp-config` (**Communication Cluster Configuration** 대화상자에서 `Ctrl+S` 로 추가됨), `vlan-configs` 없음	설계와 F
`NmCluster_VLAN_Rear` 의 값	NM 타이밍, **USER DATA**, IPv4 `239.200.0.1`	기본값: `Backbone` 을 더블클릭하면 언제나 `NmCluster_VLAN_Body` 가 열린다	NM 설정 NM 이 ㄷ
커넥터의 `pnc-mappings`	CAN 커넥터 13개에 있음	없음: 편집란이 없고 ARXML 에도 대응 요소가 없다	VFC 참조
루트 컴포지션과 시스템 정보	`PncVehicle`, **VERSION** `1.0.0`, **DESCRIPTION** `Zonal PNC tutorial vehicle`	`System` 과 새 프로젝트 기본값	`PNC_Sy`
AP 머신 모듈	7개: `LOG`, `CM`, `SM`, `NM`, `PHM`, `TIME_SYNC`, `CRYPTO`	11개: `PERSISTENCY`, `IDSM`, `UCM`, `DM` 이 더 있다	머신 파일
PARA 플랫폼	컴포지션 `PARA`, 인터페이스 `StateClient_MachineFG` 1개	컴포지션 `PARA_HPC`, AA 는 `HPC/` 아래, `StateClient_HPC_MachineFG` 와 FunctionGroup 마다 `StateClient_` 1개	노드가 F
SOME/IP 에서 파생된 신호와 PDU	없음	`SVC_ParkDistance_SR` 과 `SVC_…_PDU`	검증 오류
SOME/IP 세부 값	저장된 SD 값, `HPC_Rear_UDP_30501` 형식의 엔드포인트	SD 기본값, `<노드>_<커넥터>_UDP_<포트>` 형식의 엔드포인트	영향이 없 한다
CAN FD 세부 값	데이터 구간 2 Mbps, CAN FD 프레임 16개	컨트롤러의 **CAN FD** 토글만	영향이 없 **KBPS)** (
배치와 기본값	예제 배치, FunctionGroup 상태 **VALUE** 없음, Zone ECU 에 `pnc-synchronous-wakeup: false`	**Auto Layout** 배치, **VALUE** `0`	파일 값(

그래서 어떻게 하면 되는가

예제와 같은 모델이 필요하면 예제 프로젝트를 열거나 예제 ARXML 을 임포트한다. 아니면 차이를 그대로 두고 System 에디터와 검증의 "No validation issues." 를 목표로 한다.

자주 겪는 문제

대부분 잘못 입력한 것이 아니라 현재 버전의 표시 방식 때문이다.

캔버스와 선택

증상	할 일
클러스터의 종류와 전송 속도가 안 보인다	클러스터 사각형에 마우스를 올린다
노드를 우클릭하면 다른 메뉴가 열린다	커넥터나 컨트롤러가 아니라 노드 안의 빈 곳을 우클릭한다
커넥터나 컨트롤러가 선택되지 않는다	작은 아이콘을 정확히 클릭한다. 클러스터는 사각형을 클릭한다
속성 패널 뒤의 노드를 클릭할 수 없다	캔버스를 드래그해 옮기거나 패널을 접는다
Auto Layout 이 위에서 아래로 정렬하지 않는다	무시한다. 배치는 모델에 영향이 없다
VFB 노드 이름이 너무 작다	`Ctrl` 과 휠로 확대한다
VFB 드릴다운 경로가 `System` 으로 시작한다	표시만의 문제다. 루트는 `PncVehicle` 이다

값이 화면에 안 보이거나 편집란이 없을 때

증상	할 일
커넥터의 `pnc-mappings` 에 편집란이 없다	파일에 보존된다. 모델 조각을 본다
HPC 커넥터에 **PNC GATEWAY TYPE** 이 없다	정상이다. Classic Platform ECU 의 커넥터에만 있다
VLAN 이름의 편집란이 없다	채널에는 VLAN ID 만 입력한다
프레임 수준 CAN FD 가 안 보인다	화면에서 확인할 수 없는 값이다
CAN 클러스터의 **TRANSPORT PROTOCOL** 이 꺼져 있다	정상이다. 이 튜토리얼은 켜지 않는다
`Backbone` 이 언제나 `NmCluster_VLAN_Body` 를 연다	`NmCluster_VLAN_Rear` 를 여는 화면이 없다. 위 표를 본다
NM 노드 이름이 둘씩 보인다	**NAME** 을 `WestZone_CanNm`, `WestZone_UdpNm` 처럼 고친다
TX-NM-PDU-REFS 후보가 없다	신호·PDU·프레임과 I-PDU Group의 NM-PDU 만들기를 마치고 NM 설정으로 돌아온다
컨테이너와 NoPnc 그룹을 만드는 화면이 없다	**PNC GROUPS** 의 **+** 로 만들지 않는다. 선택한 PNC 에 묶인다

목록과 표

증상	할 일
I-PDU Group 후보에 NM-PDU 가 섞여 나온다	이름이 `_Pdu` 로 끝나는 응용 PDU 만 고른다
그룹 행 끝의 ×	연결 해제가 아니라 그룹 삭제다
PNC MEMBERSHIP 이 PDU 를 Rx 와 Tx 양쪽에 넣는다	필요하면 반대쪽 그룹에서 × 로 뺀다
뷰어의 열이 잘린다	뷰어를 넓히거나 셀에 마우스를 올린다
NM 클러스터 머리글이 대문자로 보인다	머리글 스타일일 뿐이다. 이름은 `NmCluster_WestCAN` 이다
검색 상자로 클러스터를 찾을 수 없다	옆의 필터를 쓴다

편집 중 동작

증상	할 일
NM 네트워크 핸들 행이 펼쳐진 채로 남는다	요약 행을 눌러 접는다
두 상태 버튼을 눌러도 아무 일이 없다	아직 값이 없다. 반대쪽을 눌렀다가 되돌린다
LENGTH 를 줄여도 **MAX** 가 따라 바뀌지 않는다	**MAX** 를 직접 고친다
새 신호가 마지막 신호 다음에 놓인다	**START** 를 직접 입력한다
포트를 바꾼 뒤 `PNC:Interface` 가 저장되지 않은 채 남는다	**PROJECT** > **Save All** 을 고른다(System 에디터와 검증)
SOME/IP 포트 오류가 Interface 편집기를 연다	포트는 System 편집기의 커넥터에서 고친다
Machine Configuration 이 항상 모듈 11개를 보여 준다	저장하면 11개가 모두 기록된다
핸들 요약의 NM CLUSTER 줄이 실제와 다르다	**VLANS** 와 **PARTIAL NETWORKS** 로 확인한다
하나를 고쳤는데 오류가 둘 사라진다	원인 하나가 `constr_5093` 과 `constr_3714` 처럼 여러 제약을 어길 수 있다

마치며

이제 PNC 3개가 토폴로지부터 ARXML 까지 기록된 모델 하나를 갖게 되었다. 더 자세한 것은 에디터와 작업 흐름에 있다.