SV62 MCU Platform Software Architecture Report

Table of Content

Revision History

*to be updated in reverse chronological order

Version	Date	Comment

Version	Date	Comment
0.2	21.03.2025	1. Major update to L1 Building Block View and Runtime View for all domains. 2. Introduction of L2 Building Block View.
0.1.1	23.09.2024	'Information' work items containing images are replaced by 'EA Diagram' work items that reference exported EA diagrams.
0.1	17.09.2024	First draft export based on the architecture parts that were demonstrated in the common architecture meeting between TTTech, Mobileye, and PAG

1 Introduction and Goals

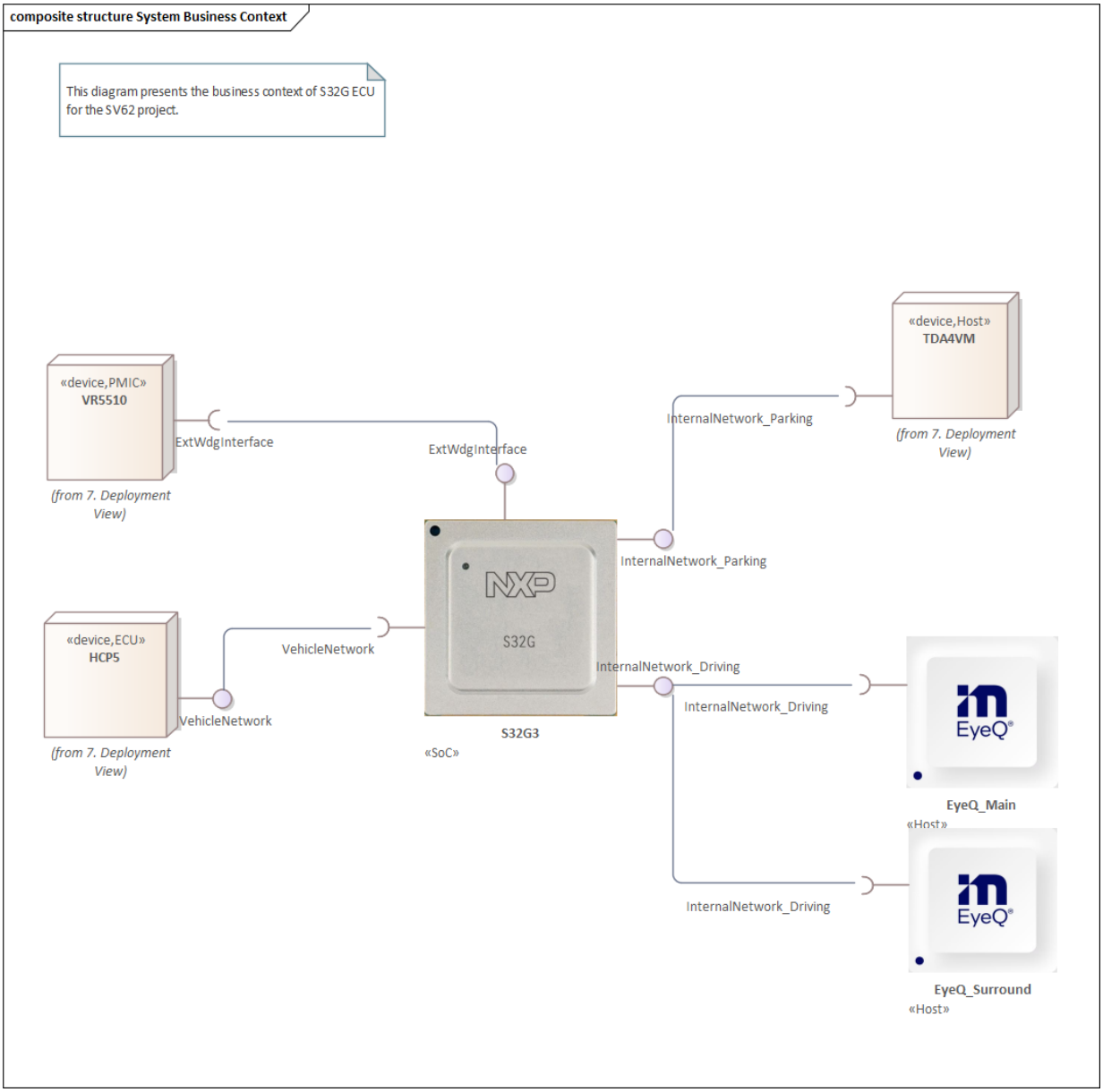
SV62 is a continuation of the HCP2.low/tv project, targeting Level 2+ ADAS functionalities for series vehicle projects. TTTech Auto AG is the Software Tier-2 to Mobileye and is responsible for the Platform Software on NXP S32G, which is the gateway for vehicle communication on the SV62 ECU.
This document is an extract of the architecture of the SV62 platform documented at ASPICE level SWE.2 supported by a model in Enterprise Architect. The aim of the document is to show the deployment of the platform software and its configuration in SV62, and to list new concepts that extend the existing platform software. **[**HCP 2MEP-226351 **]**

2 Architecture Constraints

The platform SW is expected to be a reuse of HCP2.low SW to the extent possible. The platform SW is built upon TTTech's MotionWise platform; constraints for the Application SWC architecture are captured in the Safety Manual for Application Developrs (SMAD) document. **[**HCP2MEP-226352 **]**

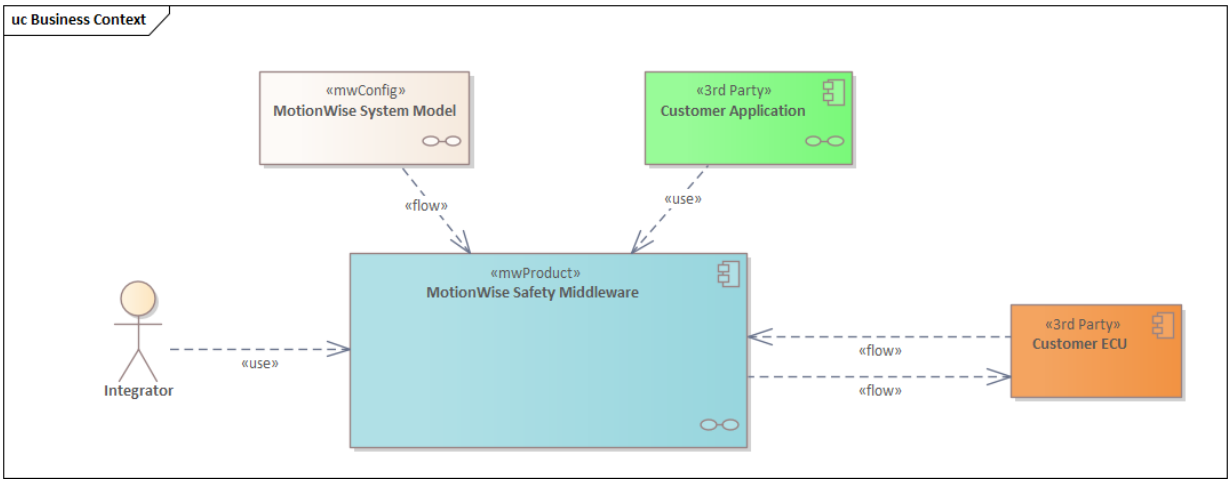
3 System Scope and Context

3.1 Business Context



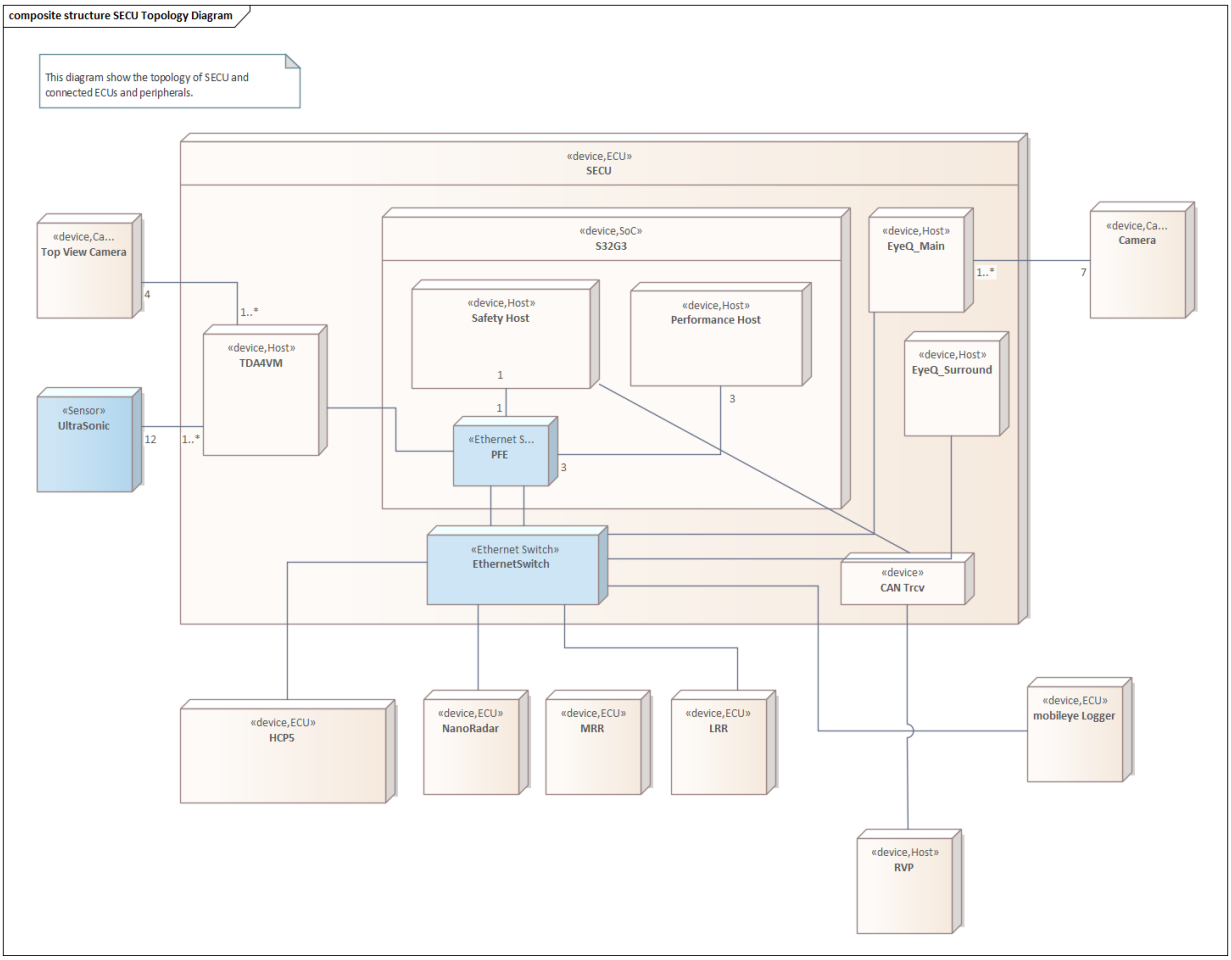
3.1.1 Platform Software View

MotionWise Middleware Platform Software is deployed on the S32G SoC to integrate necessary software components. 'Customer applications' denotes any software components that are not MotionWise Middleware Platform components (example: EyeQ Handlers, TDA Handler). The MotionWise Middleware enables communication use-cases between integrated software components on the S32G and remote ECUs ('Customer ECU'). **[**HCP2MEP-226353 **]**

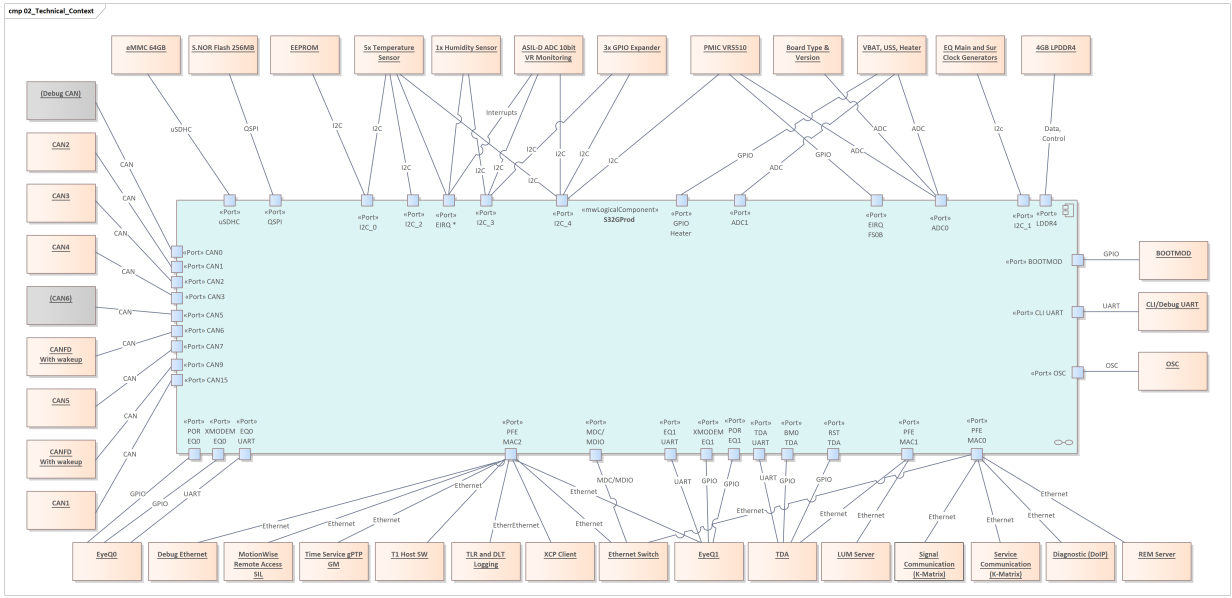


3.2 Technical Context

TTTech platform Software is deployed on S32G on Safety Host and Performance Host and it is marked with Blue color on the diagram below: **[**HCP2MEP-226372 **]**



3.2.1 Technical Interfaces

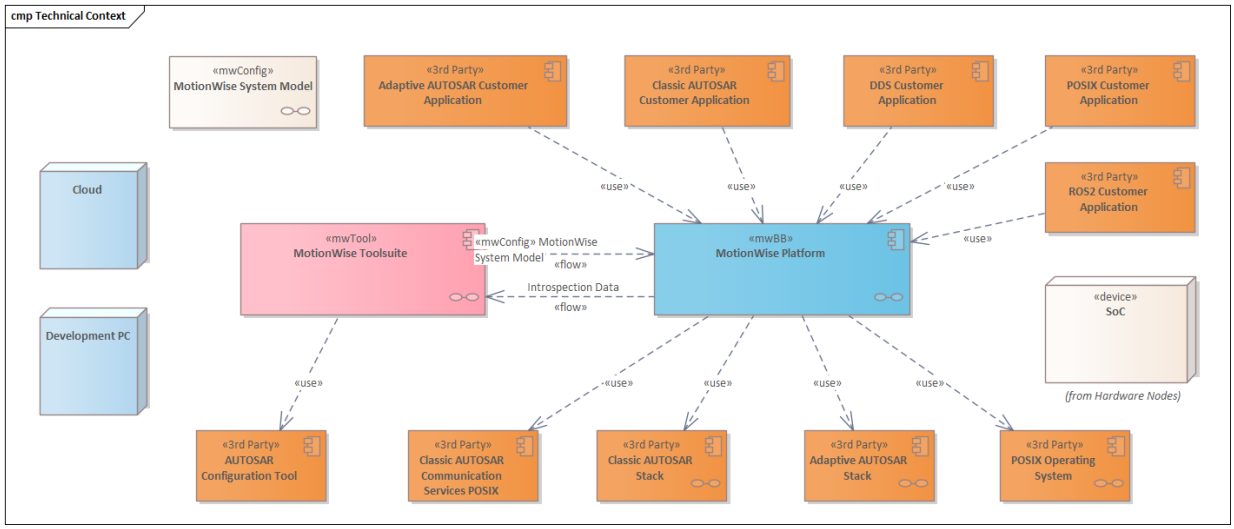


3.2.2 Platform Software View

Classic AUTOSAR Applications (Software Components) are integrated and deployed in the platform. In SV62 context the platform relies on:

- a Classic AUTOSAR Stack (SMH_SAFE) and
- a POSIX Operating System (QNX RTOS on SMH_PERF), which provides Classic AUTOSAR Communication Services (Classic AUTOSAR Communication Services POSIX).

The MotionWise Toolsuite is used to perform the integration based on the System Model. 3rd Party AUTOSAR Configuration tools are used to configure the AUTOSAR stack. **[**HCP2MEP-226354 **]**

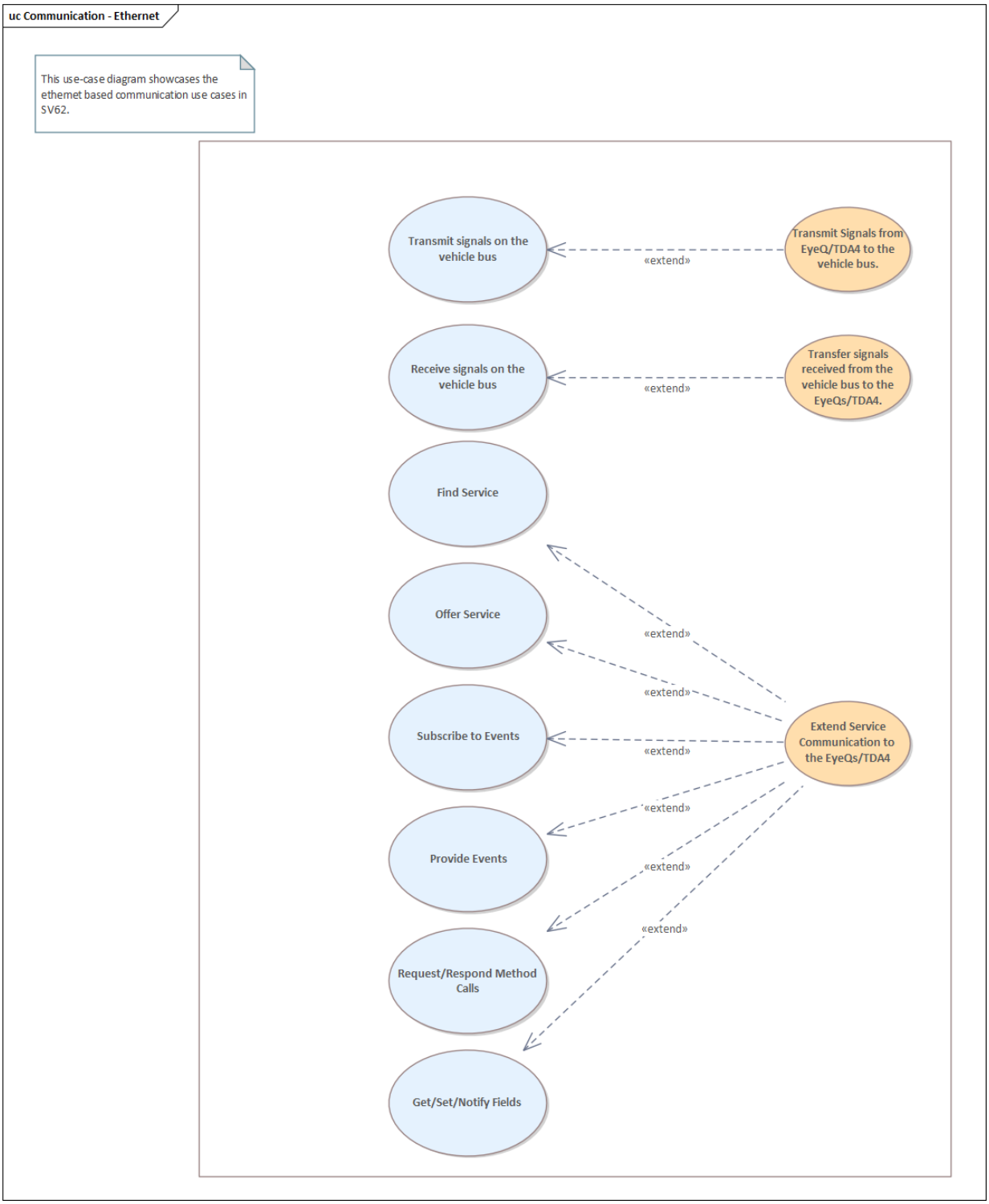


3.3 Use Cases

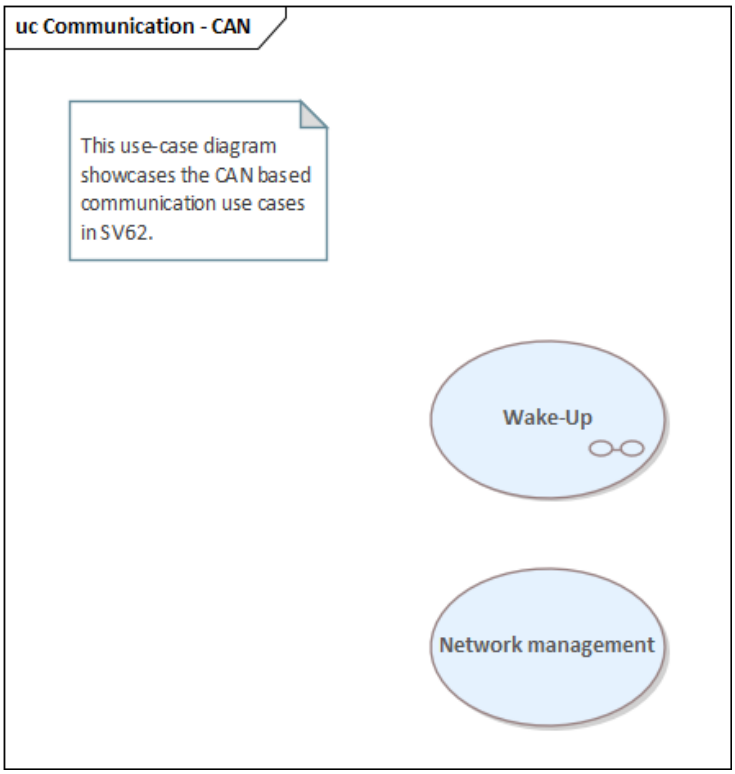
The following view presents the domains supported by SV62 platform SW: [HCP2MEP-148045]

3.3.1 Communication

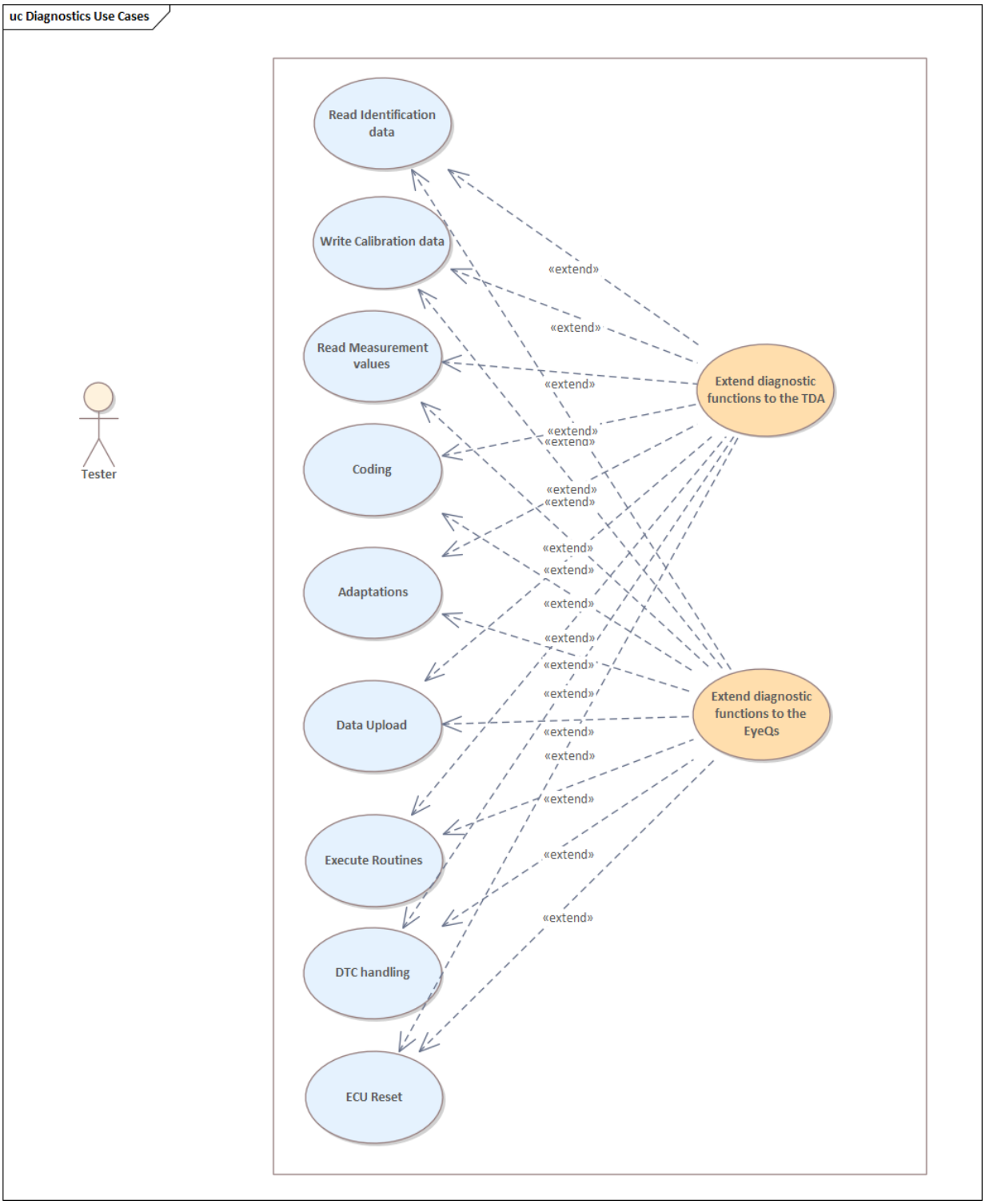
3.3.1.1 Communication - Ethernet



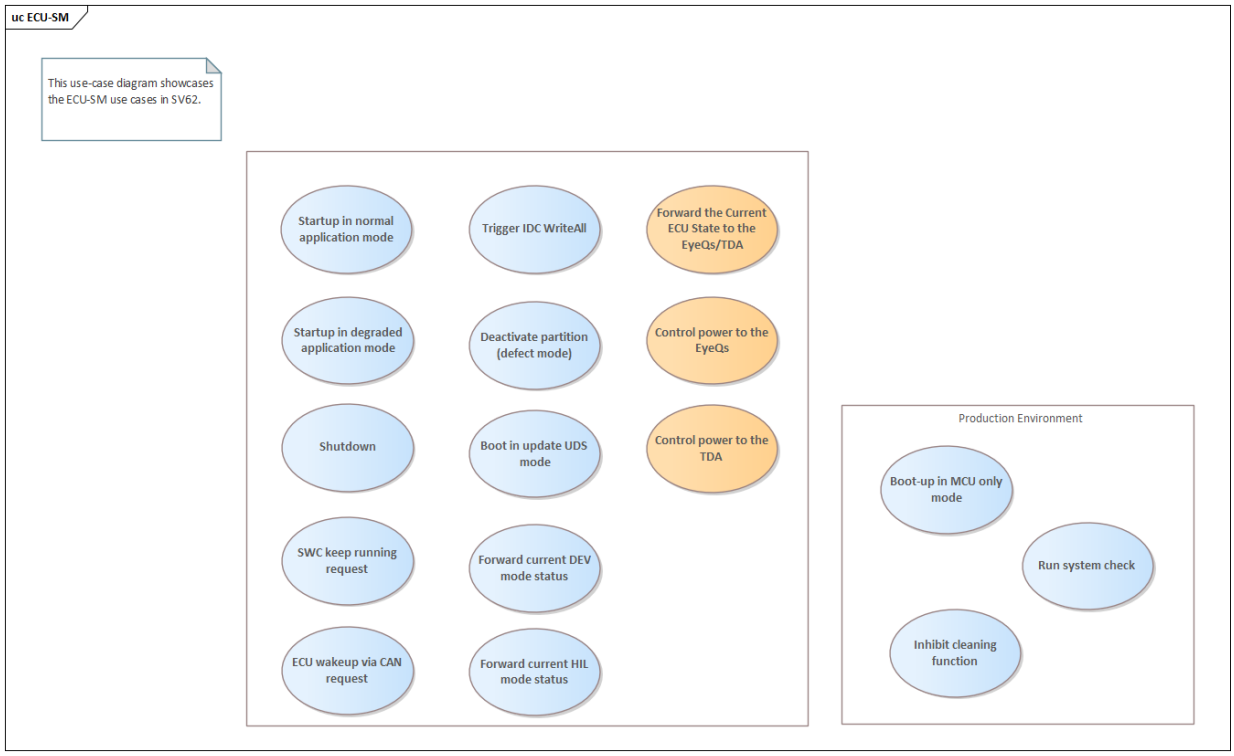
3.3.1.2 Communication - CAN



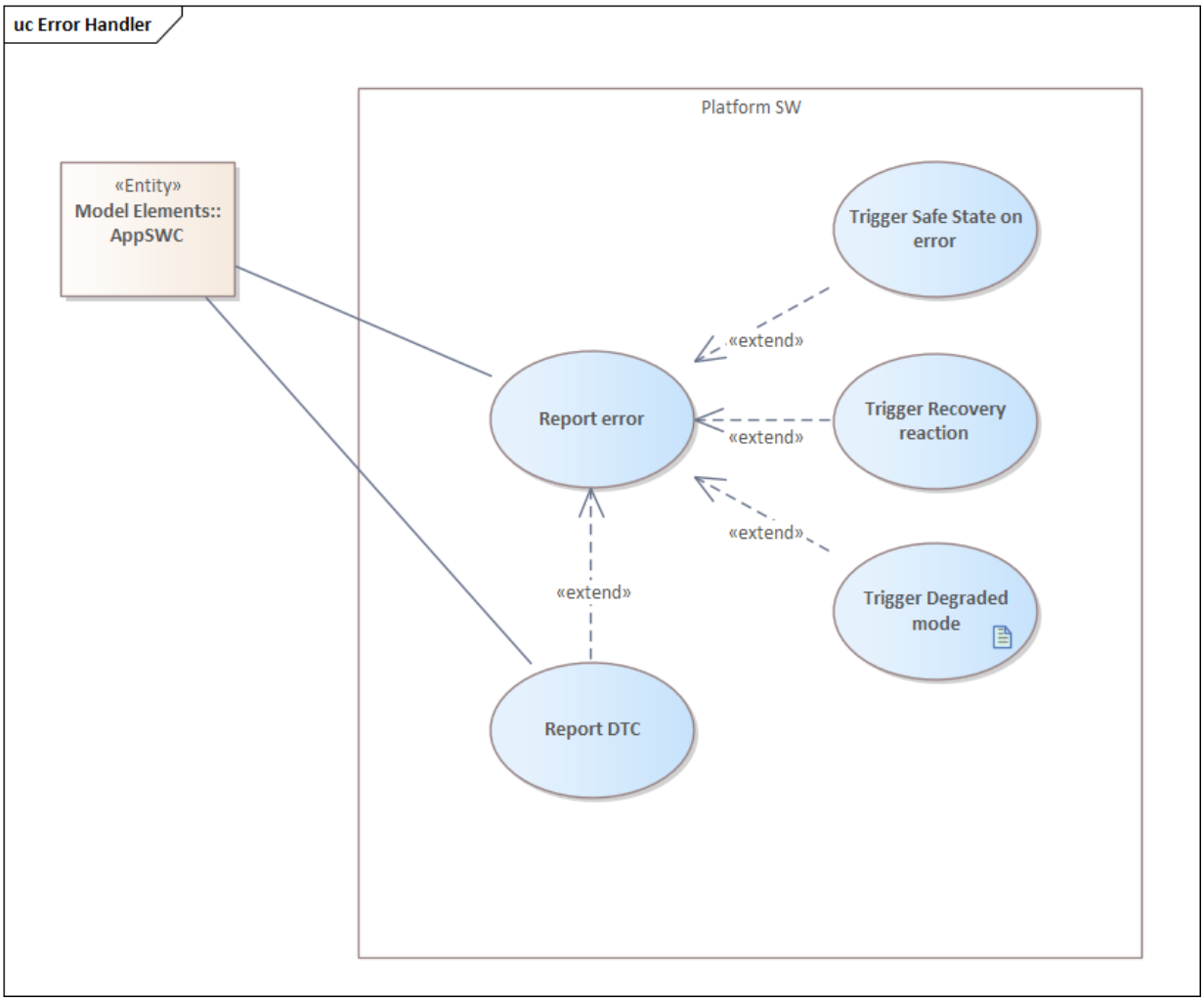
3.3.2 Diagnostic



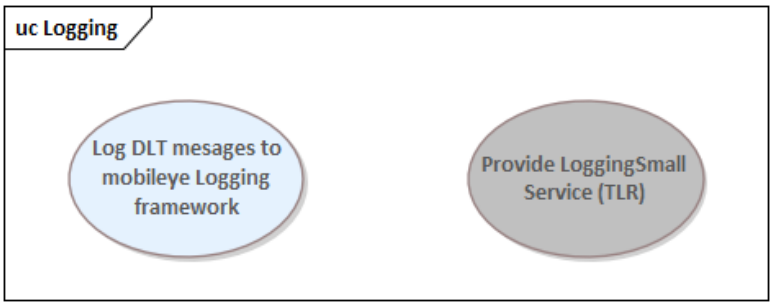
3.3.3 State management



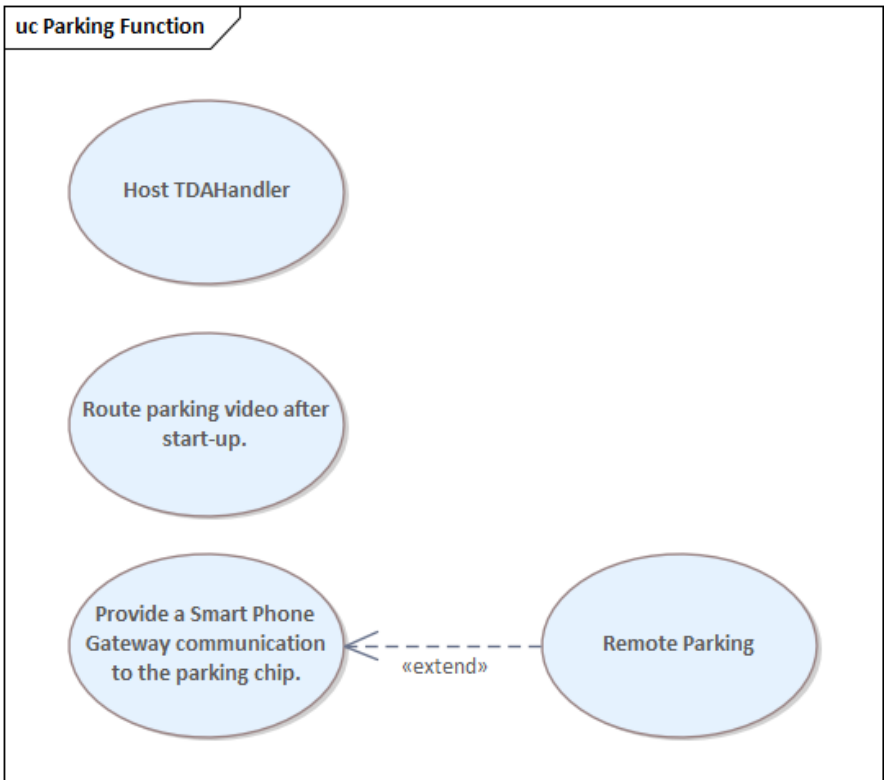
3.3.4 Error Handling



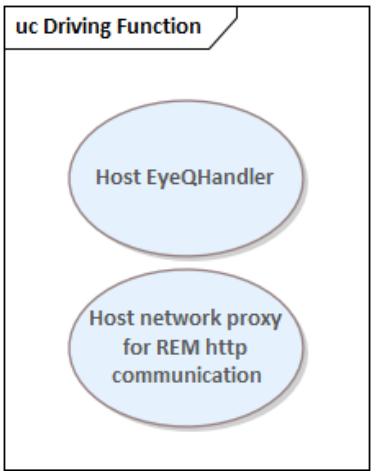
3.3.5 Logging



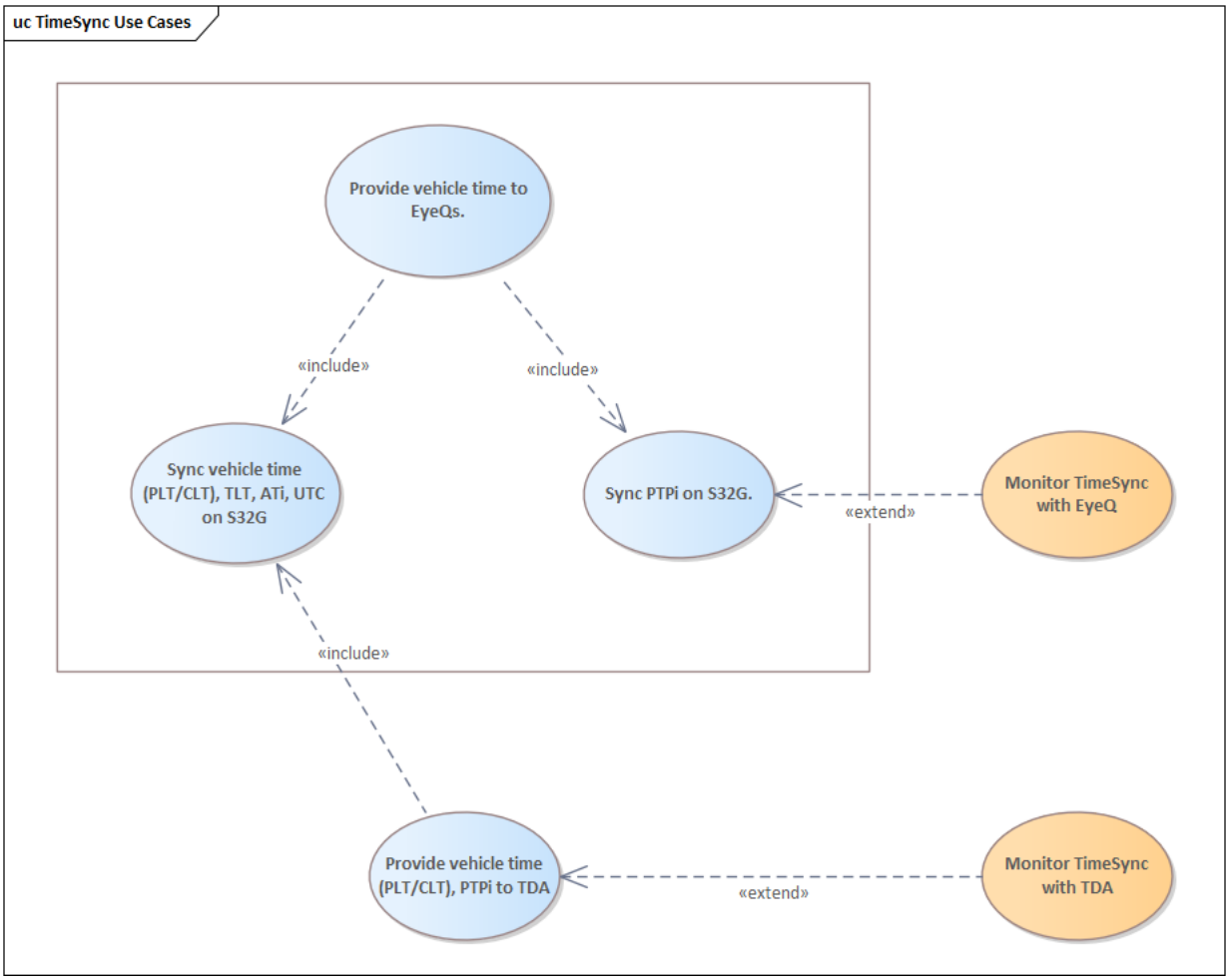
3.3.6 Parking



3.3.7 Driving

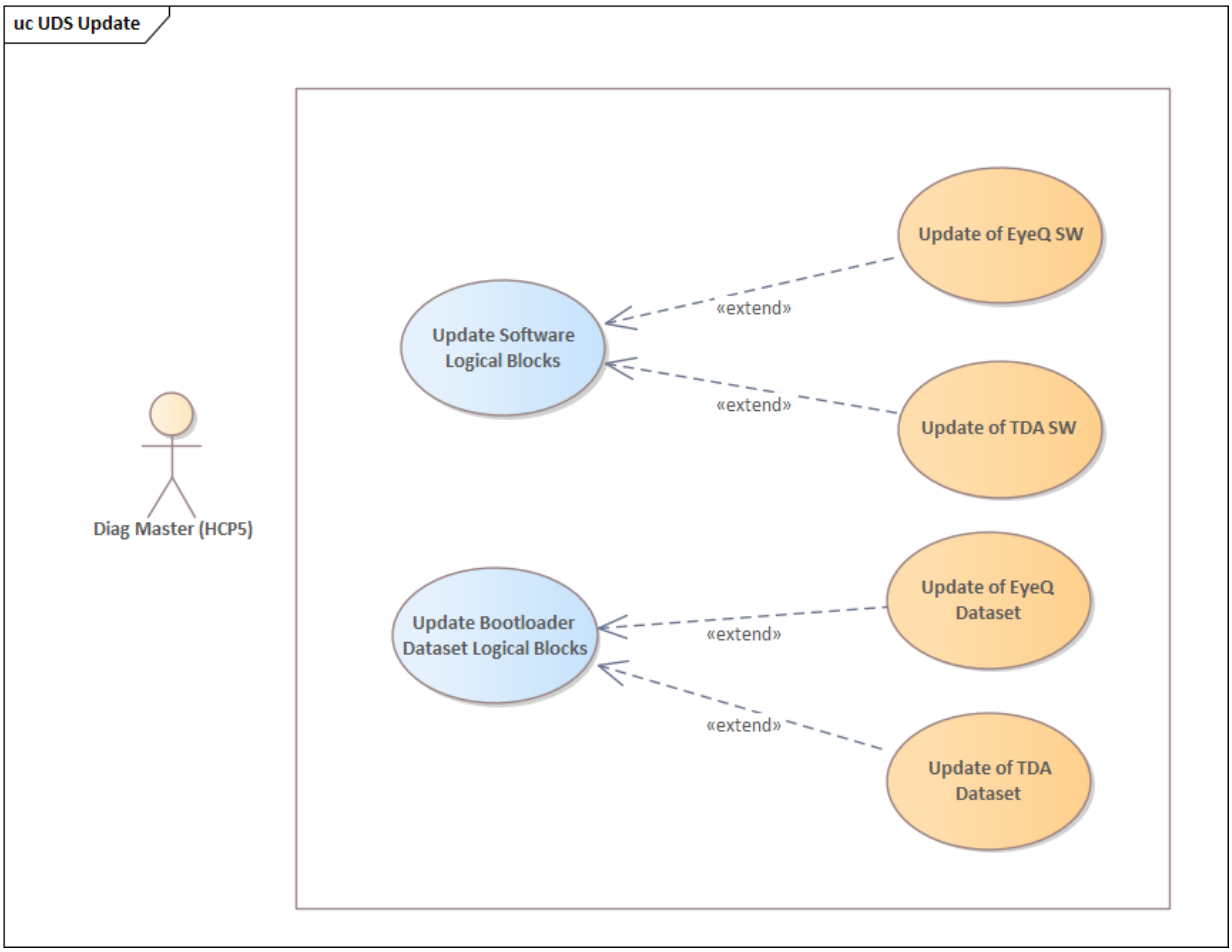


3.3.8 Time sync

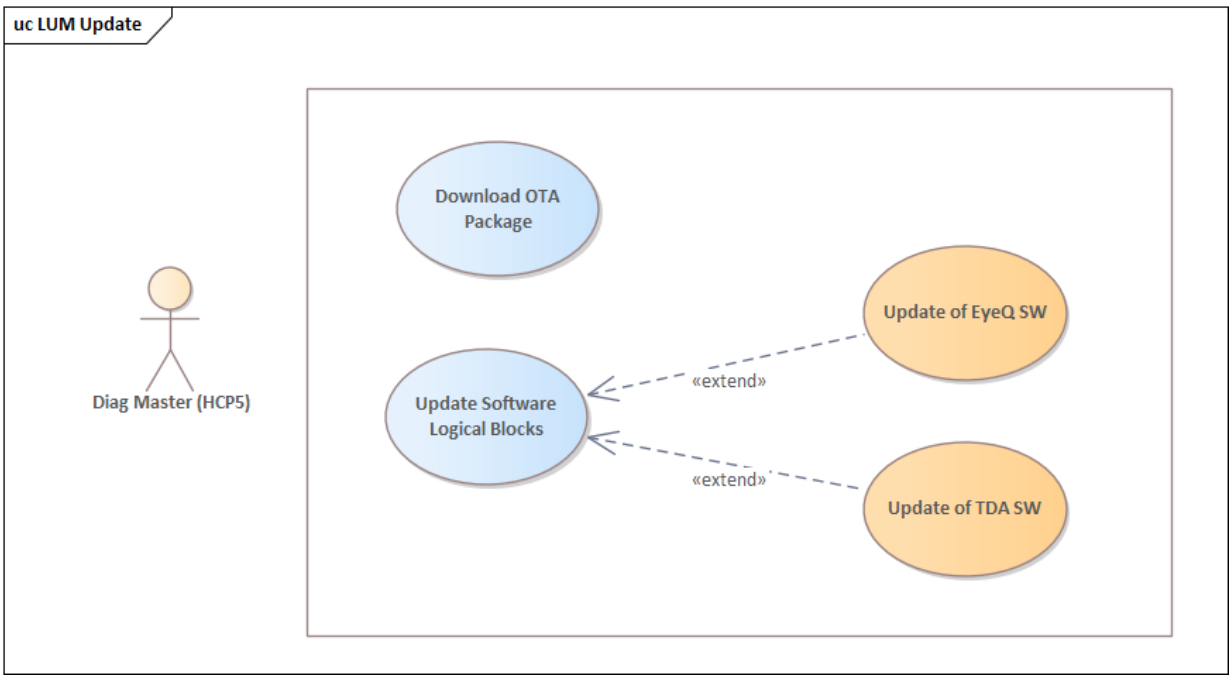


3.3.9 SW Update

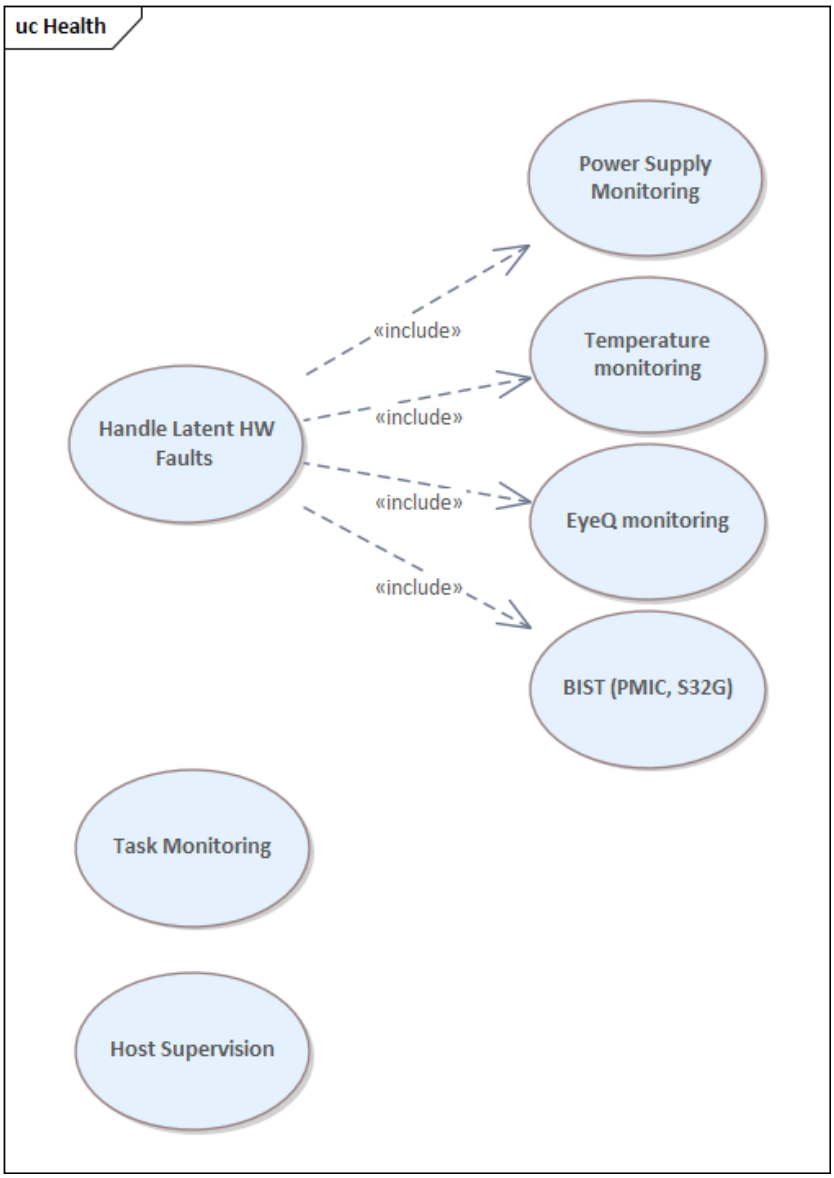
3.3.9.1 UDS Update



3.3.9.2 LUM Update

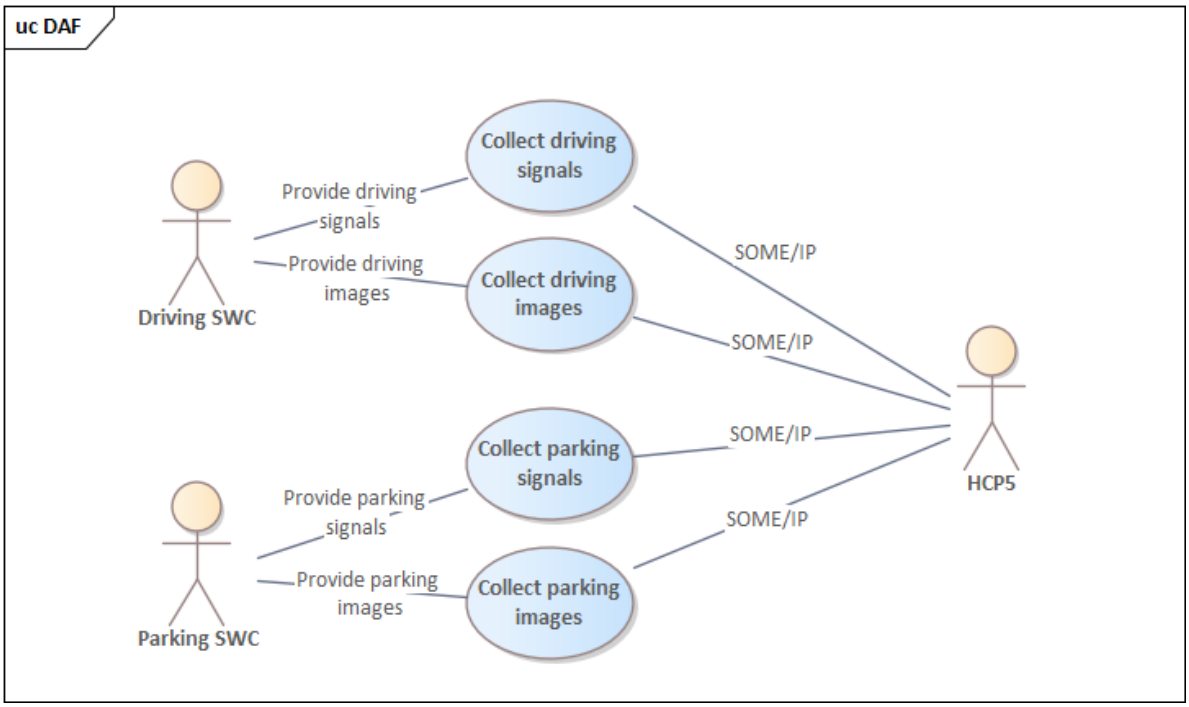


3.3.10 Health

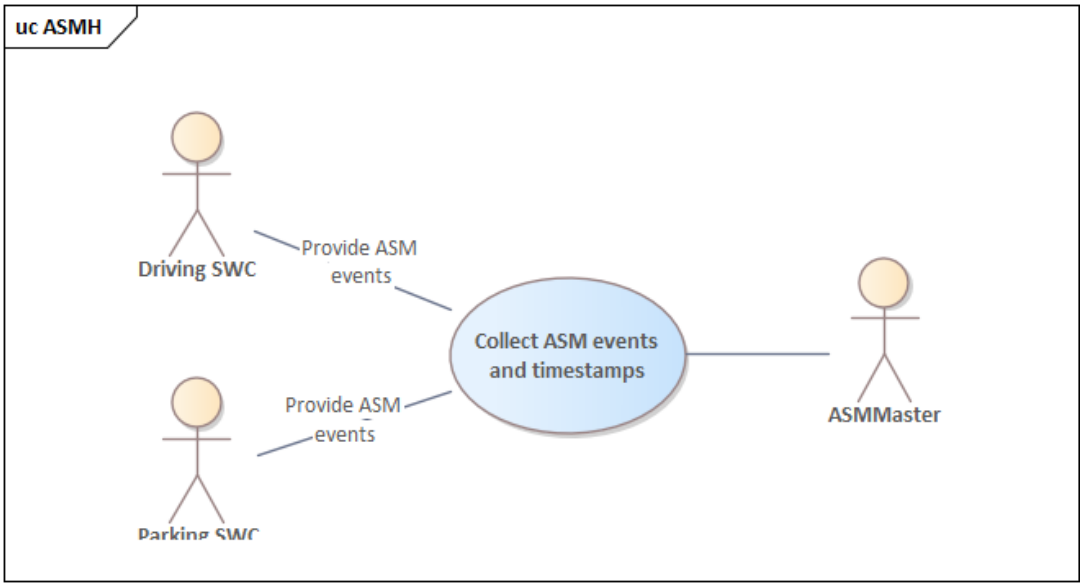


3.3.11 IDC

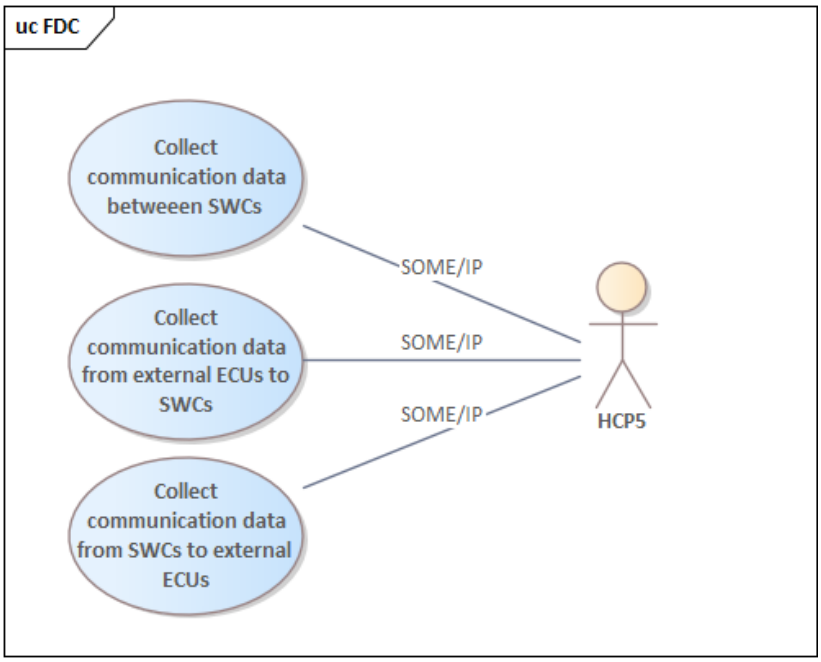
3.3.11.1 DAF



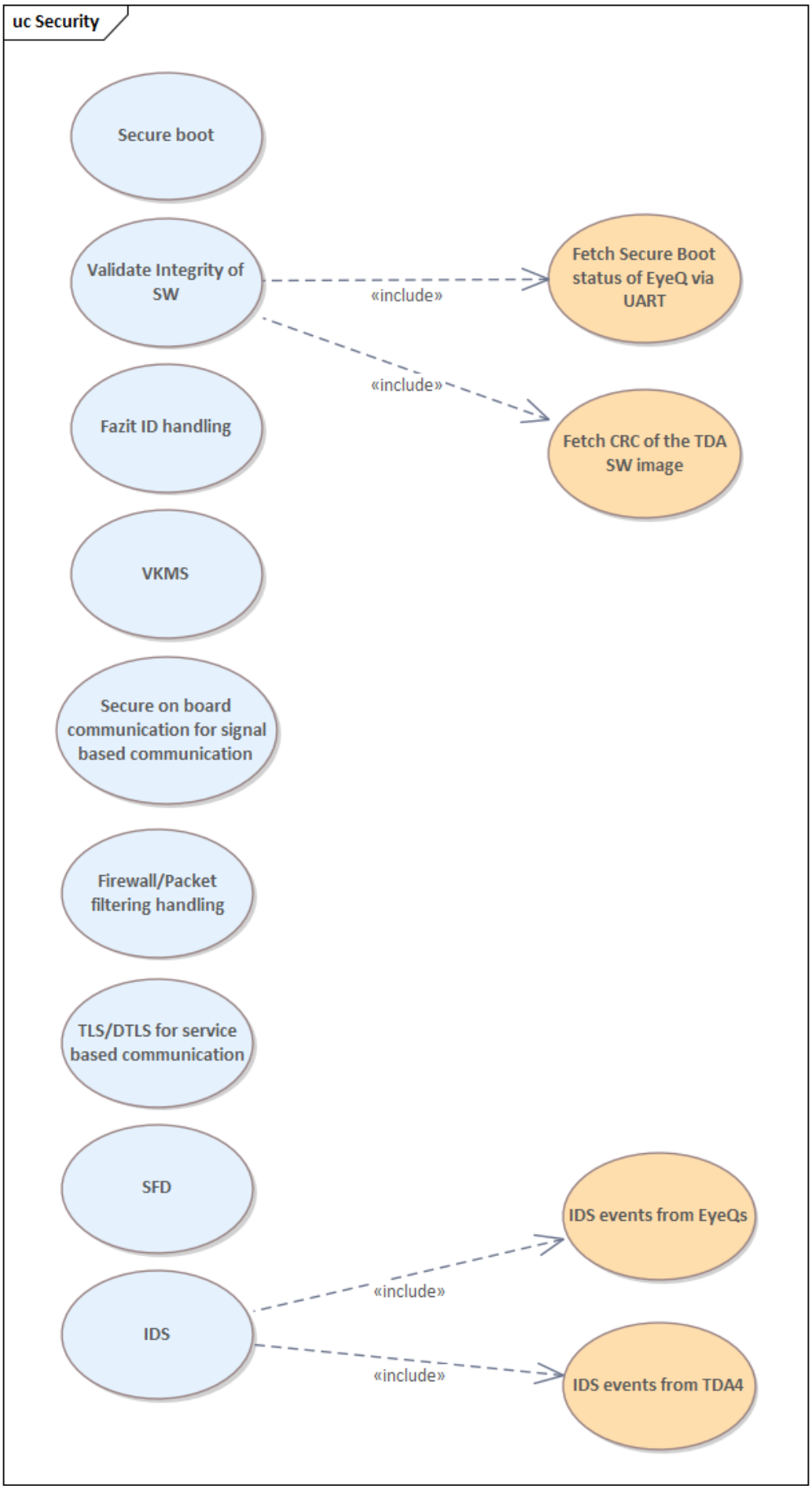
3.3.11.2 ASMH



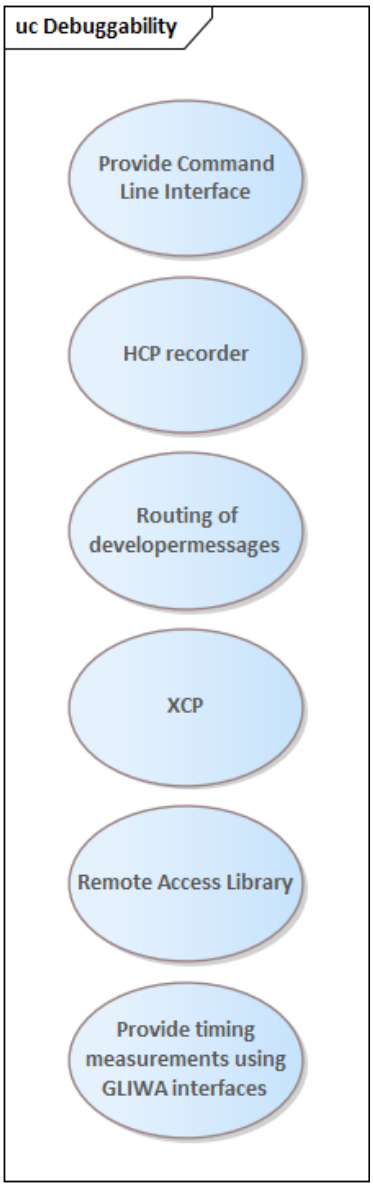
3.3.11.3 FDC



3.3.12 Security

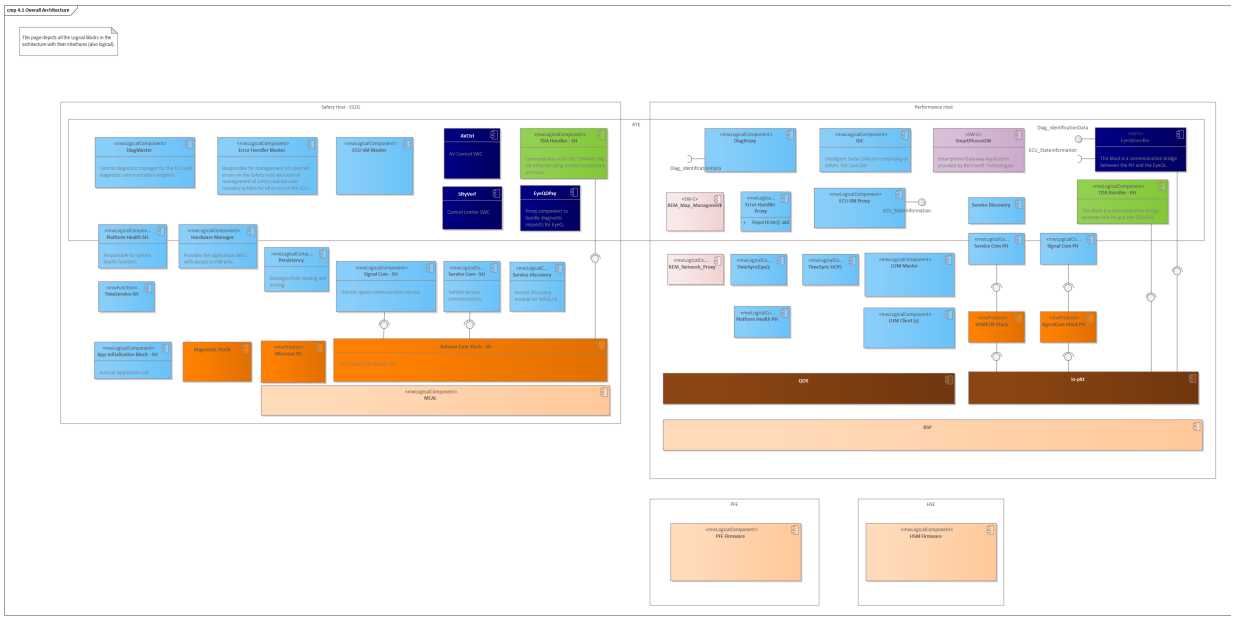

uc Security
Secure boot
Validate Integrity of SW
«include»
Fetch Secure Boot status of EyeQ via UART
«include»
Fazit ID handling
Fetch CRC of the TDA SW image
VKMS
Secure on board communication for signal based communication
Firewall/Packet filtering handling
TLS/DTLS for service based communication
SFD
IDS events from EyeQs
«include»
IDS
«include»
IDS events from TDA4

3.3.13 Debuggability



4 Solution Strategy

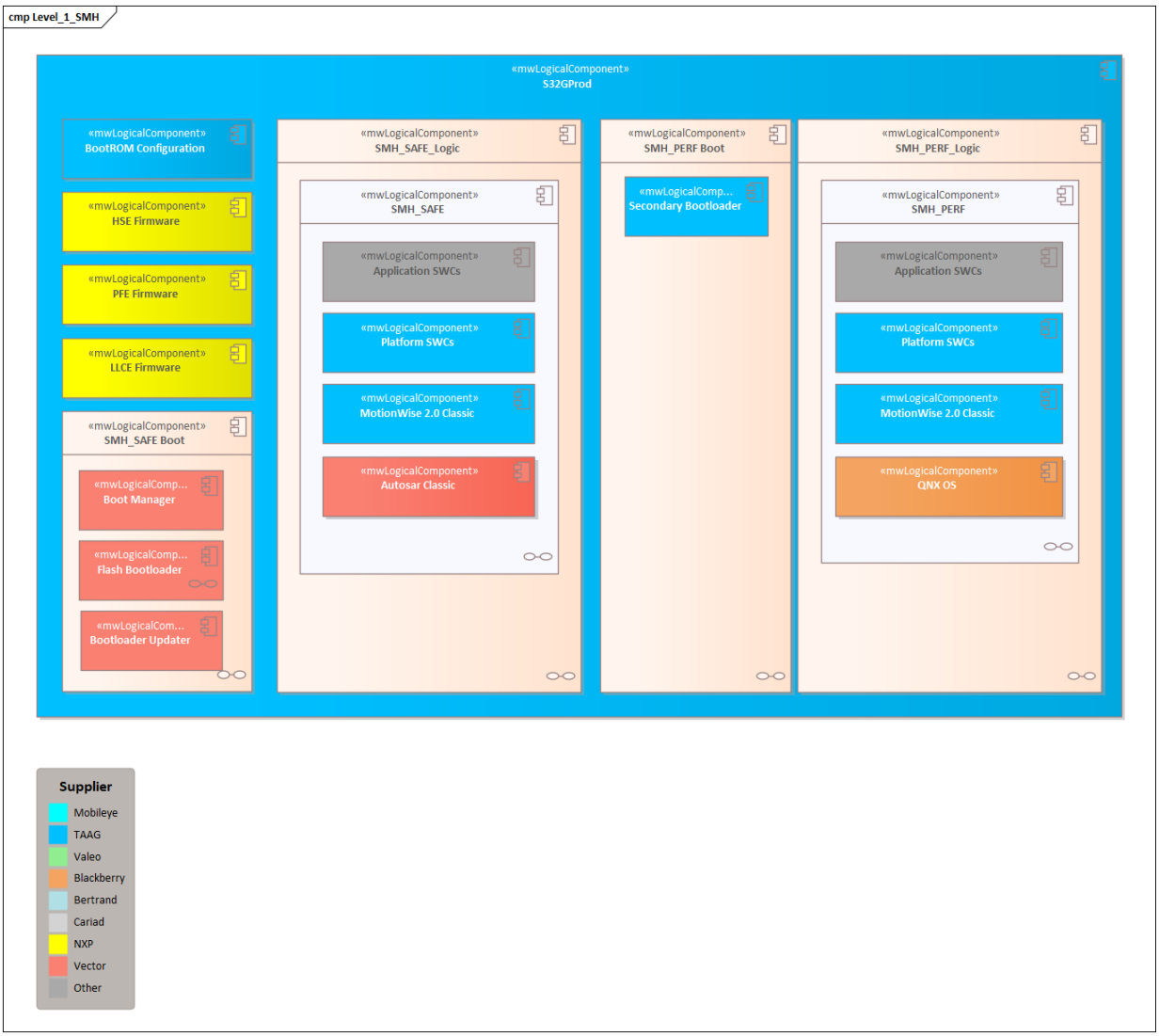
Logical Architecture View: [HCP2MEP-148292]



5 Building Block View

5.1 Level 1: High-level building blocks

The following diagram shows the high-level building blocks of the S32GProd platform software. [S32GPRODP-256479]

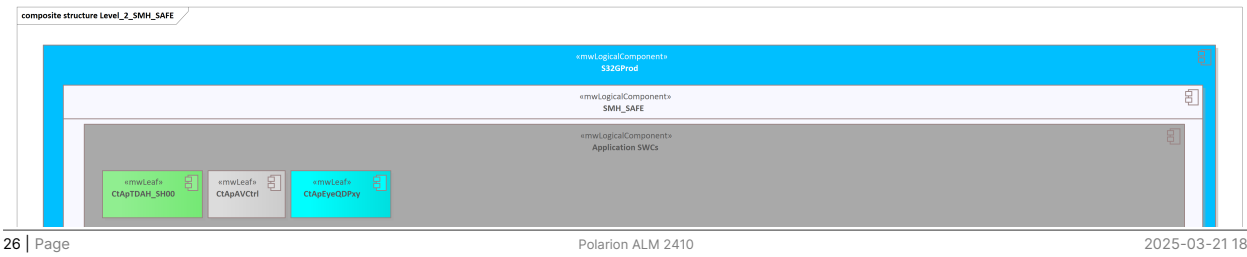


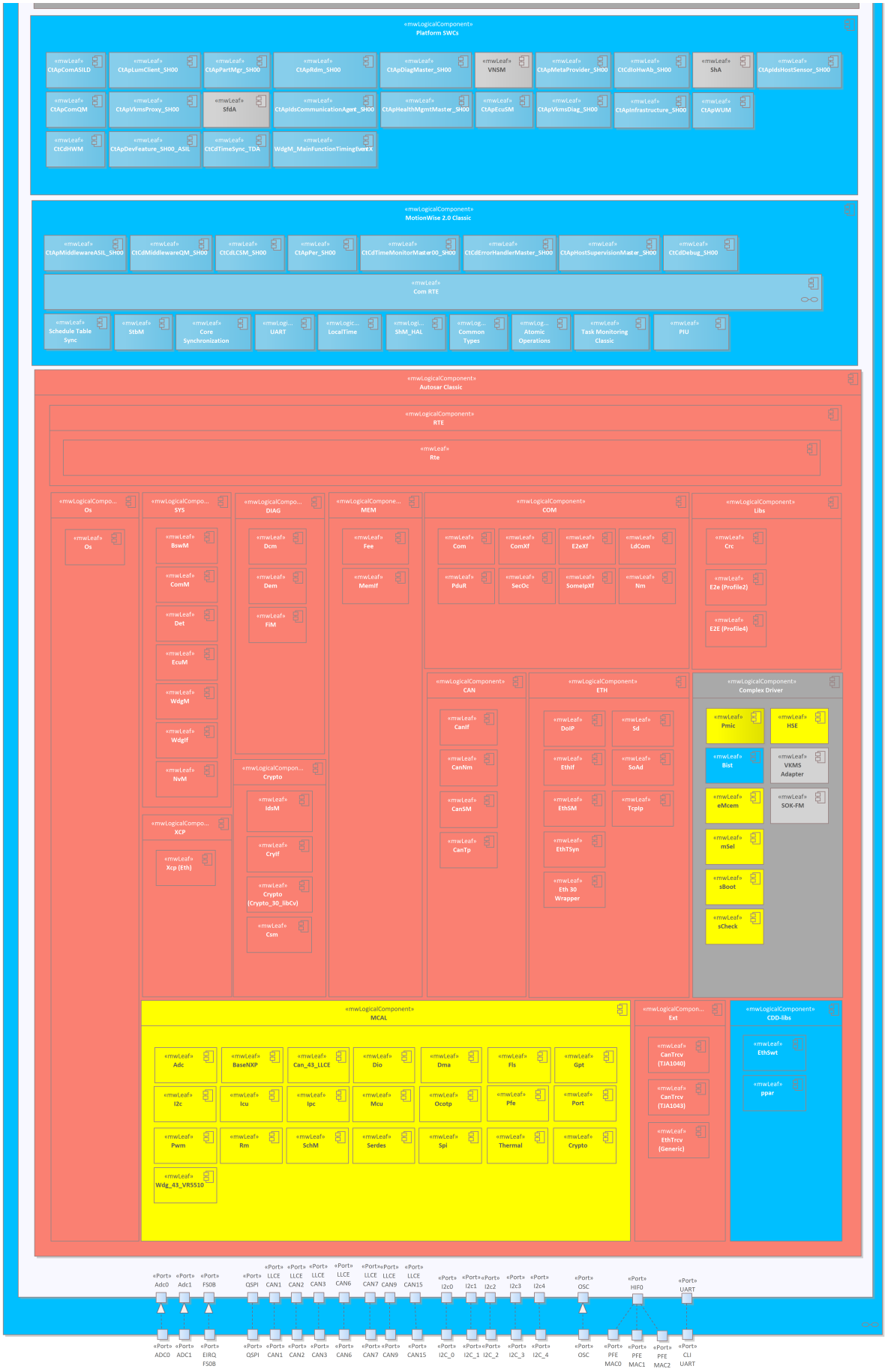
SMH_SAFE (safety host) is based on Classic Autosar stack and SMH_PERF (performance host) is based on QNX OS for Safety.

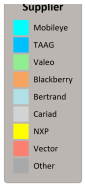
On top of Autosar/QNX MotionWise platform is TTTech middleware used for hosting of SWCs which includes communication middleware which provides RTE interface to the SWCs. [HCP2MEP-220075]

5.1.1 Safety Partition - SMH_SAFE

The following diagram shows the building block view of the SMH_SAFE. [S32GPRODP-332726]

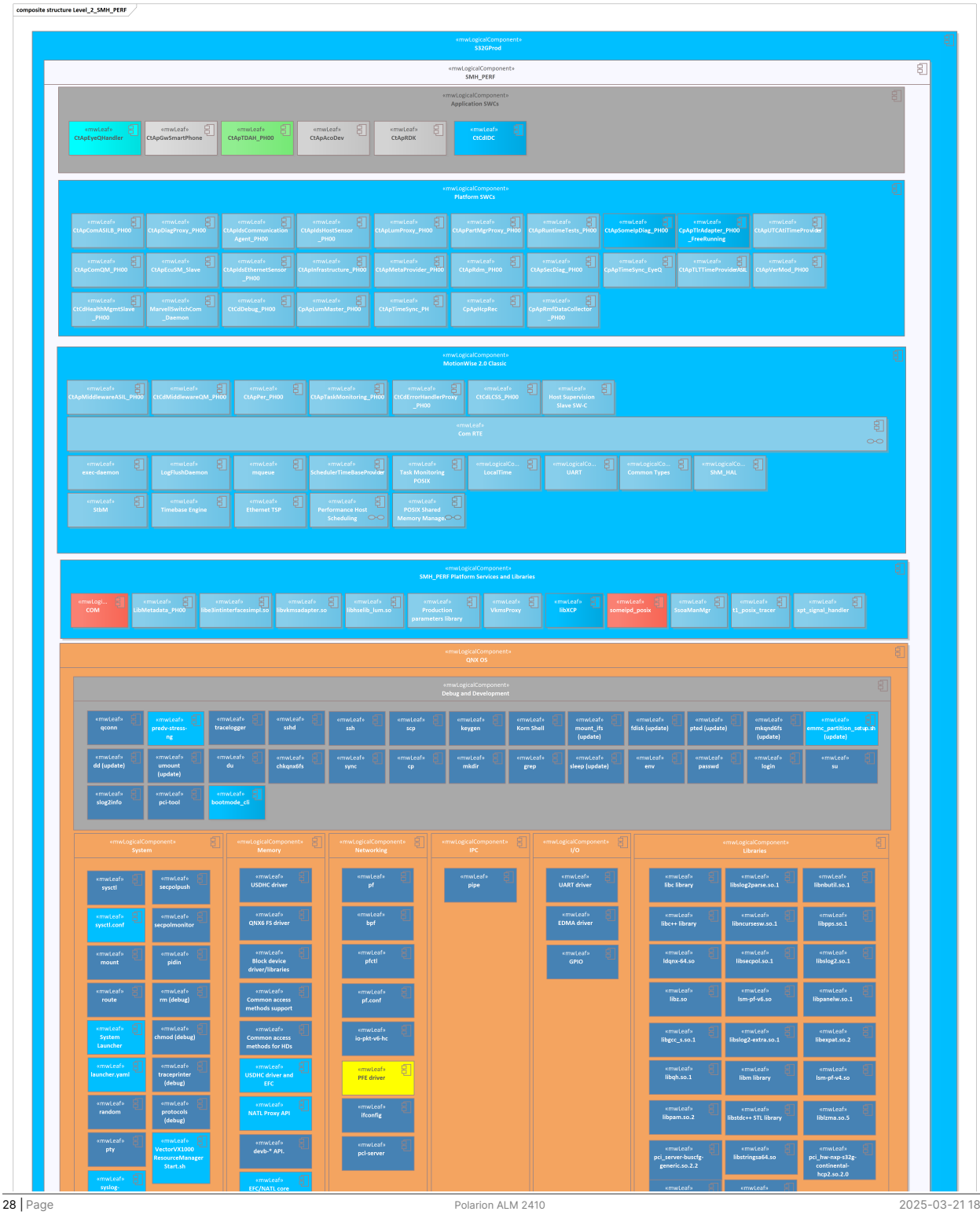



«mwLogicalComponent»
Platform SWCs
«mwLeaf» CtApComASILD
«mwLeaf» CtApLumClient_SH00
«mwLeaf» CtApPartMgr_SH00
«mwLeaf» CtApRdm_SH00
«mwLeaf» CtApDiagMaster_SH00
«mwLeaf» VNSM
«mwLeaf» CtApMetaProvider_SH00
«mwLeaf» CtCdIoHwAb_SH00
«mwLeaf» ShA
«mwLeaf» CtApIdsHostSensor_SH00
«mwLeaf» CtApComQM
«mwLeaf» CtApVkmsProxy_SH00
«mwLeaf» SfdA
«mwLeaf» CtApIdsCommunicationAgent_SH00
«mwLeaf» CtApHealthMgmtMaster_SH00
«mwLeaf» CtApEcuSM
«mwLeaf» CtApVkmsDiag_SH00
«mwLeaf» CtApInfrastructure_SH00
«mwLeaf» CtApWUM
«mwLeaf» CtCdHWM
«mwLeaf» CtApDevFeature_SH00_ASIL
«mwLeaf» CtCdTimeSync_TDA
«mwLeaf» WdgM_MainFunctionTimingEvent
«mwLogicalComponent»
MotionWise 2.0 Classic
«mwLeaf» CtApMiddlewareASIL_SH00
«mwLeaf» CtCdMiddlewareQM_SH00
«mwLeaf» CtCdLCSM_SH00
«mwLeaf» CtApPer_SH00
«mwLeaf» CtCdTimeMonitorMaster00_SH00
«mwLeaf» CtCdErrorHandlerMaster_SH00
«mwLeaf» CtApHostSupervisionMaster_SH00
«mwLeaf» CtCdDebug_SH00
«mwLeaf» Com RTE
«mwLeaf» Schedule Table Sync
«mwLeaf» StbM
«mwLeaf» Core Synchronization
«mwLeaf» UART
«mwLeaf» LocalTime
«mwLeaf» ShM_HAL
«mwLeaf» Common Types
«mwLeaf» Atomic Operations
«mwLeaf» Task Monitoring Classic
«mwLeaf» PIU
«mwLogicalComponent»
Autosar Classic
«mwLogicalComponent» RTE
«mwLeaf» Rte
«mwLogicalCompo... Os
«mwLeaf» Os
«mwLogicalCompo... SYS
«mwLeaf» BswM
«mwLeaf» ComM
«mwLeaf» Det
«mwLeaf» EcuM
«mwLeaf» WdgM
«mwLeaf» Wdgif
«mwLeaf» NvM
«mwLogicalCompo... XCP
«mwLeaf» Xcp (Eth)
«mwLogicalCompo... DIAG
«mwLeaf» Dcm
«mwLeaf» Dem
«mwLeaf» FiM
«mwLogicalCompon... Crypto
«mwLeaf» IdsM
«mwLeaf» Cryif
«mwLeaf» Crypto (Crypto_30_libCv)
«mwLeaf» Csm
«mwLogicalCompone... MEM
«mwLeaf» Fee
«mwLeaf» MemIf
«mwLogicalComponent» COM
«mwLeaf» Com
«mwLeaf» ComXf
«mwLeaf» E2eXf
«mwLeaf» LdCom
«mwLeaf» PduR
«mwLeaf» SecOc
«mwLeaf» SomeIpXf
«mwLeaf» Nm
«mwLogicalComponent» Libs
«mwLeaf» Crc
«mwLeaf» E2e (Profile2)
«mwLeaf» E2E (Profile4)
«mwLogicalComponent» CAN
«mwLeaf» CanIf
«mwLeaf» CanNm
«mwLeaf» CanSM
«mwLeaf» CanTp
«mwLogicalComponent» ETH
«mwLeaf» DoIP
«mwLeaf» Sd
«mwLeaf» EthIf
«mwLeaf» SoAd
«mwLeaf» EthSM
«mwLeaf» TcpIp
«mwLeaf» EthTSyn
«mwLeaf» Eth 30 Wrapper
«mwLogicalComponent» Complex Driver
«mwLeaf» Pmic
«mwLeaf» HSE
«mwLeaf» Bist
«mwLeaf» VKMS Adapter
«mwLeaf» eMcem
«mwLeaf» SOK-FM
«mwLeaf» mSel
«mwLeaf» sBoot
«mwLeaf» sCheck
«mwLogicalComponent» MCAL
«mwLeaf» Adc
«mwLeaf» BaseNXP
«mwLeaf» Can_43_LLCE
«mwLeaf» Dio
«mwLeaf» Dma
«mwLeaf» Fls
«mwLeaf» Gpt
«mwLeaf» I2c
«mwLeaf» Icu
«mwLeaf» Ipc
«mwLeaf» Mcu
«mwLeaf» Ocotp
«mwLeaf» Pfe
«mwLeaf» Port
«mwLeaf» Pwm
«mwLeaf» Rm
«mwLeaf» SchM
«mwLeaf» Serdes
«mwLeaf» Spi
«mwLeaf» Thermal
«mwLeaf» Crypto
«mwLeaf» Wdg_43_VR5510
«mwLogicalCompon... Ext
«mwLeaf» CanTrcv (TJA1040)
«mwLeaf» CanTrcv (TJA1043)
«mwLeaf» EthTrcv (Generic)
«mwLogicalComponent» CDD-libs
«mwLeaf» EthSwt
«mwLeaf» ppar
«Port» Adc0
«Port» Adc1
«Port» FS0B
«Port» QSPI
«Port» LCCE CAN1
«Port» LCCE CAN2
«Port» LCCE CAN3
«Port» LCCE CAN6
«Port» LCCE CAN7
«Port» LCCE CAN9
«Port» LCCE CAN15
«Port» I2c0
«Port» I2c1
«Port» I2c2
«Port» I2c3
«Port» I2c4
«Port» OSC
«Port» HIF0
«Port» UART
«Port» ADC0
«Port» ADC1
«Port» EIRQ FS0B
«Port» QSPI
«Port» CAN1
«Port» CAN2
«Port» CAN3
«Port» CAN6
«Port» CAN7
«Port» CAN9
«Port» CAN15
«Port» I2C_0
«Port» I2C_1
«Port» I2C_2
«Port» I2C_3
«Port» I2C_4
«Port» OSC
«Port» PFE MAC0
«Port» PFE MAC1
«Port» PFE MAC2
«Port» CLI UART



5.1.2 Performance Partition - SMH_PERF

The following diagram shows the building block view of the SMH_Perf. **[**S32GPRODP-332725 **]**




«mwLeaf» monitor
«mwLeaf» emmc fstab
«mwLeaf» emmc_partitions _postinit.sh
«mwLeaf» CRC32 procedure
«mwLogicalComponent» Diagnostics
«mwLeaf» System logger Service
«mwLeaf» dumper
«mwLogicalComponent» OS
«mwLeaf» procNTO for Safety (intrumented)
«mwLogicalComponent» Startup Function
«mwLeaf» Startup script
«mwLeaf» Startup code
«mwLeaf» Startup Verifier
«mwLeaf» pci_slog2.so.2.1
«mwLeaf» libcatalog.so.1
«mwLeaf» pci_strings.so.2.0
«mwLeaf» libregex.so.1
«mwLeaf» pci_cap-0x11.so.2.2
«mwLeaf» libsocket.so.3
«mwLeaf» pci_cap-0x01.so.2.0
«mwLeaf» libpci.so.2.3
«mwLeaf» pcie-xcap-0x0001.so.2.3
«mwLeaf» pci_bkwd_compat.so.2.1
«mwLeaf» libssl.so.2.1
«mwLeaf» pci_debug2.so.2.1
«mwLeaf» libqcrypto.so.1.0
«mwLeaf» pci_cap-0x10.so.2.3
«mwLeaf» libqcrypto-openssl.so
«mwLeaf» pcie_xcap-0x000b-ffffffff.so.2.0
«mwLeaf» librpc.so.2
«mwLeaf» libcrypto.so.2.1
«mwLeaf» libtracelog.so.1 (debug)
«mwLeaf» libjson.so.1
«Port» DDR4
«Port» uSDHC
«Port» UART
«Port» HIF1
«Port» HIF3
«Port» HIF2
«Port» RST EQ0
«Port» RST EQ1
«Port» RST TDA
«Port» XMODEM EQ0
«Port» XMODEM EQ1
«Port» BM TDA
«Port» I2C_3
«Port» I2C_4
«Port» LDDR4
«Port» uSDHC
«Port» CLI UART
«Port» PFE MAC0
«Port» PFE MAC1
«Port» PFE MAC2
«Port» POR EQ0
«Port» POR EQ1
«Port» RST TDA
«Port» XMODEM EQ0
«Port» XMODEM EQ1
«Port» BM0 TDA
«Port» I2C_3
«Port» I2C_4
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

5.2 Level 1: Logical Services

The following two diagrams show the high-level logical service decomposition of the S32GProd platform software. The logical services are assigned to the two partitions (SMH_SAFE and SMH_PERF) provided by the S32G. **[**S32GPRODP-296177 **]**

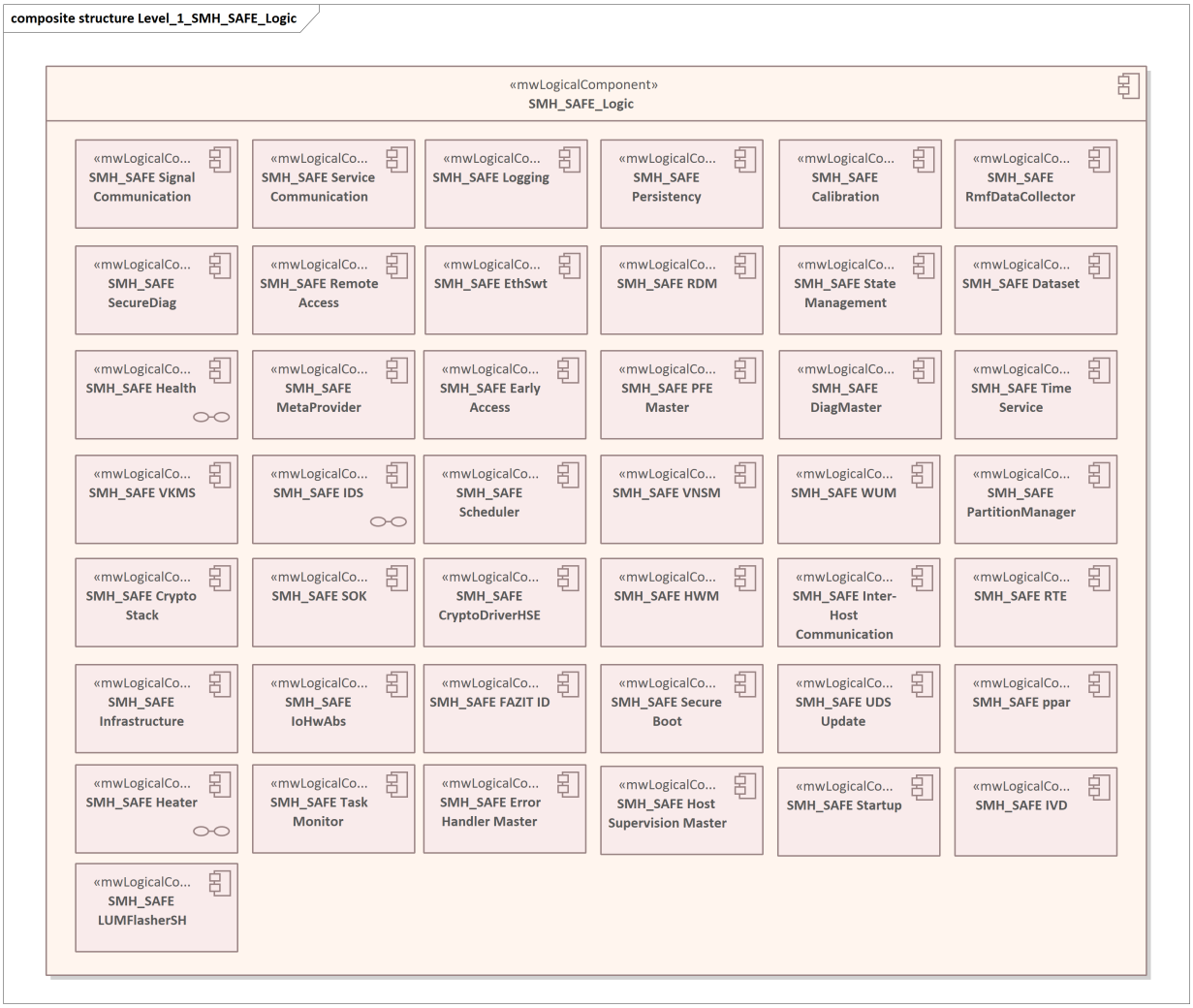


Table below describes core functionality of logical components deployed on Safety Host. For more details about components, check the usage of concrete component in the building block and runtime diagrams in this document.

Logical component	Functionality

Logical component	Functionality
SMH_SAFE Signal Communication	Signal Communication block is responsible for Signal Oriented Communication (PDU based) and includes: • Abstraction of AutoSAR BSW and COM stack for Application SWCs through RTE Data Elements • Serialization and de-serialization of data • End-to-end communication handling • PDU state machines • Message time stamping • Raw-to-physical conversion and physical-to-raw conversion • PDU groups management
SMH_SAFE Service Communication	Service Communication block is responsible for Service Oriented Communication (SOME/IP based) and includes: • Abstraction of AutoSAR BSW and COM stack for Application SWCs through RTE Data Elements • Serialization and de-serialization of data • End-to-end communication handling
SMH_SAFE Logging	Logging provides logging capabilities complaint with AUTOSAR standard over UART and Ethernet. Component itself is responsible: • Collecting data from SWCs • Internally store data • Flush logs to configured sink channel • Store configuration persistently • Restore persistent configuration
SMH_SAFE Persistency	Persistency is responsible for the reading/writing data from/to non-volatile memory, and provides data with its status to applications via the MotionWise RTE. Before distribution of the persistent data, it checks integrity.
SMH_SAFE Calibration	Calibration component provided access to the internal variables of SH SWCs and non queued RTE ports of the SWCs (via XCP).
SMH_SAFE RmfDataCollector	Resource Measurement Framework is a component which integrates GLIWA T1 within and utilizes tracing for each SWC integrated into MotionWise Time-Triggered scheduler. RMF provides APIs towards to SWCs for custom instrumentation for profiling with finer granularity.
SMH_SAFE SecureDiag	SecureDiag (SFD) is responsible to secure specific diagnostic objects within an ECU to prevent unauthorized access. Each diagnostic object that is protected by SFD is either assigned a SFD Role (Role protected) or is end-to-end (E2E) protected. If the ECU does not have authorization to access a diagnostic object, it will receive a Security Access Denied.
SMH_SAFE Remote Access	Remote Access is a component to handle communication with Debug Host (Debug PC) and providing tooling for debug PC to monitor and re-configure network (RTE) traffic on hosts that execute the MotionWise Classic Platform.
SMH_SAFE EthSwt	Component representing handling related to Marvell Ethernet Switch.

Logical component	Functionality
SMH_SAFE RDM	RDM (Routing of developer messages) provides logging capabilities on production/series SW over vehicle ethernet. Component itself is responsible: • Collecting data from SWCs via RTE ports • Send logs to Signal Communication Component • Store configuration persistently • Allow runtime configuration of which PDUs to be used for which SWC over diagnostics • Restore persistent configuration
SMH_SAFE State Management	Ecu State Management (EcuSM) specifies the handling of HCP2 system modes like application, degraded and UDS update, and power cycle states like startup, running and shutdown. The platform function covers the following architectural tasks: • Wakeup handling of the ECU • Startup of the ECU, including all processors, memory units and external interfaces • Startup/Shutdown of software components • Recovery from platform or software component faults • Start of communication interfaces • Coordinated shutdown of communication interfaces (network management) • Shutdown of the ECU, including all processors, memory units and external interfaces • Entering degradation or defect mode in case of errors • Dynamometer Operation Mode control • HIL-Mode activation and deactivation • Development mode activation and deactivation
SMH_SAFE Dataset	Dataset is responsible for the reading datasets from non-volatile memory, and provides data with its status to applications via the MotionWise RTE. Before distribution of the datasets, it checks validity and plausibility of the datasets.
SMH_SAFE Health	SMH_SAFE Health embodies following logical components deployed on the SMH_SAFE: • HWM (Hardware Monitoring) • Error Handler • Task Monitoring • Host Supervision
SMH_SAFE MetaProvider	Metaprovider is responsible for the provision of information (Name, Version, CRC etc.) for the logical blocks and hardware/sofware to the application SWC's (like Diagnostic). This information, called metadata, is stored in the non-volatile memory.
SMH_SAFE PFE Master	This block represents the Master driver for the PFE. It's responsibilities include initialisation and configuration of the PFE for enabling ethernet communication from the Safety host.
SMH_SAFE DiagMaster	The DiagMaster component handles the diagnostic jobs from the BSW component DCM and distribute them to the platform SWCs, optional application SWCs, EQ handler and TDA handler instances. The communication channels (using RTE) depend on the type of diagnostic information. The DiagMaster component also forwards information to the DiagProxy if it needs to be processed on the performance partition. Conversely, the DiagMaster component receives the Diagnostic Events from platform and optional application SWCs and EQ handler instance on the safety partition and forwards them to the DCM or DEM. DiagMaster handles persistent storage of adaption DIDs and Coding distribution.

Logical component	Functionality
SMH_SAFE Time Service	Time Service is responsible for synchronization of time domains like EGT, PLT, TLT, PTPi, Precise UTC and Basic UTC. It also provides interfaces to all SWCs for accessing the time values of the above domains.
SMH_SAFE VKMS	The main goal of the VKMS is to provide possibility of centralized key provisioning and management on vehicle level. VKMS consists of a single VKMS core, VKMS adapter on SMH-S and multiple VKMS proxies on Sh and PH. VKMS core is running inside HSM of ECU, VKMS proxies are running on each partition. Pre-shared keys have to be provisioned at production time using DLC (Download Containers) through a diagnostic routine.
SMH_SAFE IDS	IDS provides an interface to the services and components that will report qualified security events to the IDS system with abstraction from the low level dependencies on the IDS components and modules. This shall be possible for qualified security events delivered by the TDA4 or for qualified events reported by the EyeQs though the S32G. For such external events, it shall be possible to utilize a bypass service that would forwarding the security events coming from the TDA4. Similarly, a suitable EyeQ Bypass service shall be available for the qualified events collected at the EyeQ handler. Those shall be then forwarded via the before mentioned service to the PH-Handler at the Safety host and sent then to the IDS core without modification.
SMH_SAFE Scheduler	Time triggered scheduler based on scheduling tables. It controls how schedulable entities (RTE runnables; schedulable threads) have access to CPU cores. It uses Vector OS scheduler.
SMH_SAFE VNSM	Vehicle Network State Manager (VNSM) is a SWC responsible for forwarding the NM messages to all bus channels of a given PNC independent of the bus protocol. VNSM gets activation / deactivation requests whether the given PNC shall be activated or not, VNSM is not responsible to request / release PNCs on its own.
SMH_SAFE WUM	The WakeUpMonitor (WUM) is a software component for monitoring the activity of communication buses and ECUs when the vehicle is parked.
SMH_SAFE PartitionManager	PartitionManager on SAFE partition receives the partition management commands from PartitionManager on PERF via RTE, and performs the partition switch between active and inactive partitions.
SMH_SAFE Crypto Stack	Crypto Stack consists of following modules: • Crypto Service Manager (Csm) • Crypto Interface (CryIf) • Crypto Driver (Crypto) The Crypto Stack offers a standardized access to cryptographic services for applications and system functions. The cryptographic services are the computation of hashes, the verification of asymmetrical signatures, or the symmetrical encryption of data.

Logical component	Functionality
SMH_SAFE SOK	SOK enables the communication of data within vehicle in a way that protects the integrity of the data against tampering. SOK ensures manipulated data is identified, logged and not processed by the data sink. The entire communication of a function may not necessarily be protected by SOK. Rather, only the part of the function's communication for which it is deemed necessary to guarantee integrity must be protected by SOK. Signal based communication can be secured in Autosar with the Secure On Board Communication (SecOC) standard. SecOC is implemented as a service next to the PDU Router (PUDR). SOK protects the integrity of a communication by attaching signatures to the data to be protected. The data itself remains unchanged.
SMH_SAFE CryptoDriverHSE	The Crypto Driver module is located in the micro controller abstraction layer and is below the Crypto Interface module and Crypto Service Manager module. It implements a generic interface for synchronous and asynchronous cryptographic primitives. It also supports key storage, key configuration, and key management for cryptographic services.

SMH_SAFE HWM	SMH_SAFE HWM is responsible for monitoring of the HW signals specified to be monitored by the SMH, like:
	• voltages, temperatures, humidity, EyeQ (CREER/NCRERR)
	• executing and monitoring of BIST tests, like
	  ◦ BIST tests for SMH_SAFE (LBIST, MBIST)),
	  ◦ PMIC,
	  ◦ EyeQ (BGA tests)

SMH_SAFE Inter-Host Communication	Inter Host communication provides communication abstraction to Application SWCs deployed on different Hosts. The host-to-host communication can be based on shared memory and Ethernet. End-to-end protection (E2EP) is implemented for safety-related communication. E2E state for shared-memory based communication is ensured via memory FFI, and for Ethernet communication, E2E protection is implemented using standard profiles. Note: Interhost communication in S32G product is implemented over Ethernet (PFE based communication) for the first releases (v1, v2). Future releases will switch to shared memory communication.
SMH_SAFE RTE	This block represents TTTech's MotionWise RTE layer.
SMH_SAFE Infrastructure	Component which implements some/ip testability services based on K-matrix
SMH_SAFE IoHwAbs	Responsibility of IoHwAbs is to obtain and collect data and signals from the peripherals attached to the S32G3 SoC. Before the data and signals are provided to the upper layers, they are converted to represent physical information (eq. conversion to physical unit).
SMH_SAFE FAZIT ID	The FAZIT Identification String is used for assignment and tracking of every individual control unit. The FAZIT ID string is stored within the MFG Data section (that contains also the HW Variant information, ECU Serial Number, APTIV Part Number, MAC Address...) and will be flashed at the EOL and protected by an OTP mechanism.

Logical component	Functionality
SMH_SAFE Secure Boot	This logical entity representing the secure boot, where the Bootrom initiates the secure boot process, to enable the HSE and HSE FW based on the provided configurations. This process allows the proper establishment of the root of trust, starting from the device key and the HSE FW decryption, until triggering the process for the secure boot manager. To start the secure boot processes the different components and processes on both SH and PH. This includes the enablement of the FBL for secure software update and secure version control.
SMH_SAFE UDS Update	UDS Software Update is performed in Flash Bootloader context when the programming session is triggered by the Diagnostics Client tools (ODIS, PASDT etc...). Flash Bootloader receives the UDS Software Update diagnostics requests from the diagnostics client and orchestrates the Software Update for memory partitions (Software and Dataset logical blocks) of S32G external memory devices (NOR and eMMC) and external SoCs (TDA, EyeQ).
SMH_SAFE ppar	Component for handling MCU Manufacturing Information. It is responsible for reading of MCU MI information from the flash and providing to the applications.
SMH_SAFE HEATER	The Heater function is used to control the High Side and Low Side switches connected to the S32G through the IoHwAbs component. It is used to switch on or off the camera heater, based on user request (e.g. by a diagnostic routine).
SMH_SAFE Task Monitor	Task Monitoring is responsible to provide program execution monitor functionality: • aliveness monitor, • deadline monitor, • program flow monitor. On the SMH_SAFE, the external Watchdog of the PMIC contributes to Task Monitor concept.
SMH_SAFE Error Handler Master	Error Handler Master is responsible for collecting of the errors reported from Platform SW and customer SW-Cs. Configuration of Error Handler Master provides a means that each error can be configured with: • qualification and disqualification times • safety and recovery reactions • optionally, triggering the associated DTC
SMH_SAFE Host Supervision Master	Host Supervision is based on a master-slave concept. Host Supervision Master uses question/answer protocol to check the status of SMH_PERF hosts. In case of incorrect answer, appropriate recovery and safety reactions are performed.
SMH_SAFE Startup	This logical component covers the "normal" startup of the MCU. More details are covered by the runtime diagrams.
SMH_SAFE IVD	IVD (Integrity validation data) is a module that guarantees the integrity and authenticity of software. IVD uses two hash-values per ECU to determine the integrity of the software (by using a programming hash) and configuration (by using the configuration hash).
SMH_SAFE LUMFlasherSH	LUMFlasherSH receives the flash data from LUMClientSH via RTE and performs the flashing of NOR partitions of S32G.

[HCP2MEP-226377]

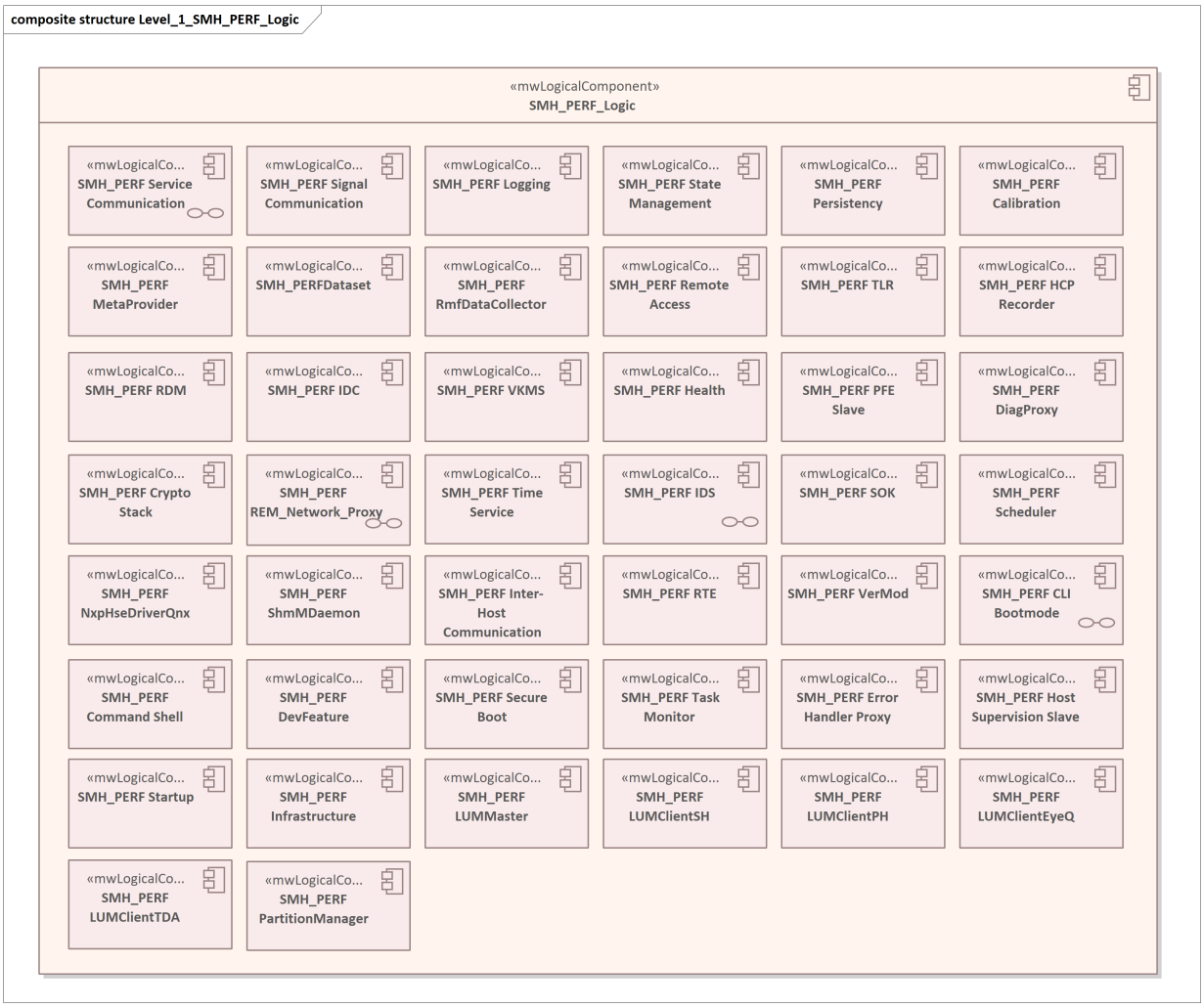


Table below describes core functionality of logical components deployed on Performance Host. For more details about components, check the usage of concrete component in the building block and runtime diagrams in this document.

Logical component	Functionality
SMH_PERF Service Communication	SMH_PERF Service Communication is logical block responsible for Service Oriented Communication (SOME/IP based). This block provides following major functionality: • Provides C APIs around C++ Class methods to upper layers, and performs conversion between C++/C data Classes/Structures. • Provides RTE interfaces to Application SWCs • It is adaptation layer between Classic and Adaptive AUTOSAR world. All upper layers (SWCs) are written according to Classic AUTOSAR documentation, lower layer of Communication stack is in line with Adaptive AUTOSAR. • Initialization of Vector SOME/IP Stack for all SWC which use service-based communication.

Logical component	Functionality
SMH_PERF Signal Communication	Signal Communication block is responsible for Signal Oriented Communication (PDU based) and includes: • Abstraction of AutoSAR Posix ETH and COM stack for Application SWCs through RTE Data Elements • Serialization and de-serialization of data • End-to-end communication handling • PDU state machines • Message time stamping • Raw-to-physical conversion and physical-to-raw conversion • PDU groups management
SMH_PERF Logging	Logging provides logging capabilities complaint with AUTOSAR standard over UART and Ethernet. Component itself is responsible: • Collecting data from SWCs • Internally store data • Flush logs to configured sink channel • Store configuration persistently • Restore persistent configuration
SMH_PERF State Management	EcuSM-Slave (EcuSM-S) is responsible to coordinate the host state and host boot process until partitions can be handled independently (ECUSM Salve is deployed on performance host). Each EcuSM-S handles partition states of its partition. EcuSM Master is deployed on Safety Host.
SMH_PERF Persistency	Persistency is responsible for the reading/writing data from/to non-volatile memory, and provides data with its status to applications via the MotionWise RTE. Before distribution of the persistent data, it checks integrity.
SMH_PERF Calibration	Calibration component provides the access to internal variables via libXCP.
SMH_PERF MetaProvider	Metaprovider is responsible for the provision of information (Name, Version, CRC etc.) for the logical blocks and hardware/sofware to the application SWC's (like Diagnostic). This information, called metadata, is stored in the non-volatile memory.
SMH_PERF Dataset	Dataset is responsible for the reading datasets from non-volatile memory, and provides data with its status to applications via the MotionWise RTE. Before distribution of the datasets, it checks validity and plausibility of the datasets.
SMH_PERF RmfDataCollector	Resource Measurement Framework is a component, integrates GLIWA T1 within and utilizes tracing for each SWC integrated into MotionWise Time-Triggered scheduler. RMF provides APIs towards to SWCs for custom instrumentation for profiling with finer granularity.
SMH_PERF Remote Access	Remote Access is a component to handle communication with Debug Host (Debug PC) and providing tooling for debug PC to monitor and re-configure network (RTE) traffic on hosts that execute the MotionWise Classic Platform.
SMH_PERF TLR	TLR component collects the logging data from SWCs on Performance host, packs them into frames according to TLR protocol specified by OEM and sends this via SOME/IP to TLR server on HCP5.

Logical component	Functionality
SMH_PERF HCP Recorder	HCP Recorder is a component which is targeted to be used for supporting the analysis of problems and defects for the production/series SW. It is responsible for: • gather events, data, operating conditions, etc. from other platform software functions • during the usage of the vehicle and store persistently • Send the collected data requested over diagnostics when Service is triggered.
SMH_PERF RDM	RDM (Routing of developer messages) provides logging capabilities on production/series SW over vehicle ethernet. Component itself is responsible:) provides logging capabilities on production/series SW over vehicle ethernet. Component itself is responsible: • Collecting data from SWCs via RTE ports • Send logs to Signal Communication Component • Store configuration persistently • Allow runtime configuration of which PDUs to be used for which SWC over diagnostics • Restore persistent configuration
SMH_PERF IDC	The IDC (Inteligent data recorder) consists of the following functions: - ASM Handler - Global Data Collector - DAF
SMH_PERF VKMS	The main goal of the VKMS is to provide possibility of centralized key provisioning and management on vehicle level. VKMS consists of a single VKMS core, VKMS adapter on SMH-S and multiple VKMS proxies on Sh and PH. VKMS core is running inside HSM of ECU, VKMS proxies are running on each partition. Pre-shared keys have to be provisioned at production time using DLC (Download Containers) through a diagnostic routine.
SMH_PERF Health	SMH_PERF Health embodies following logical components on deployed on the SMH_PERF: • HWM (Hardware Monitoring) • Error Handler • Task Monitoring • Host Supervision
SMH_PERF PFE Slave	This block represents the Slave driver for the PFE. It's responsibilities include initialisation and configuration of the PFE for enabling ethernet communication from the Performance host.
SMH_PERF DiagProxy	The DiagProxy component is integrated on the the performance partition to handle requests locally on this partition. DiagMaster on the safety partition connects to the BSW and forwards requests to the DiagProxy, which distributes the information to software parts on this partition. Conversely, the DiagProxy component receives the Diagnostic Events from platform and optional application SWCs, e.g. EQ handler and TDA handler on the performance partition and forwards them to the DiagMaster component. The DiagProxy handles persistency for adaption DIDs and Coding for its clients.

Logical component	Functionality
SMH_PERF Crypto Stack	Crypto Stack consists of following modules: 1. Crypto Service Manager (Csm) 2. Crypto Interface (CryIf) 3. Crypto Driver (Crypto) The Crypto Stack offers a standardized access to cryptographic services for applications and system functions. The cryptographic services are the computation of hashes, the verification of asymmetrical signatures, or the symmetrical encryption of data.
SMH_PERF REM_Network_proxy	Gateway between the REM client on EyeQ and the Cloud server which converts IPv4 to IPv6 http traffic.
SMH_PERF Time Service	Time Service is responsible for synchronization of time domains like EGT, PLT, TLT, PTPi, Precise UTC and Basic UTC. It also provides interfaces to all SWCs for accessing the time values of the above domains.
SMH_PERF IDS	IDS on PH is responsible for handling the IDS events generated on the performance host. IDS events collected from the IDS host and ethernet sensors, in addition to potential events from the external handlers, are collected and directed to the IdsM on the safety host in order to send the qualified events to IDS core on HCP5
SMH_PERF SOK	SOK enables the communication of data within vehicle in a way that protects the integrity of the data against tampering. SOK ensures manipulated data is identified, logged and not processed by the data sink. The entire communication of a function may not necessarily be protected by SOK. Rather, only the part of the function's communication for which it is deemed necessary to guarantee integrity must be protected by SOK. Signal based communication can be secured in Autosar with the Secure On Board Communication (SecOC) standard. SecOC is implemented as a service next to the PDU Router (PUDR).SOK protects the integrity of a communication by attaching signatures to the data to be protected. The data itself remains unchanged.
SMH_PERF Scheduler	Time triggered scheduler based on scheduling tables. It controls how schedulable entities (RTE runnables; schedulable threads) have access to CPU cores.
SMH_PERF NxpHseDriverQnx	This is a 3rd party driver, developed by NXP to allow the access to the HSE via the VKMS core interface. This is required to allow the security dependent components VKMS, SOK, TLS, etc, to access their relevant keys for requests coming though PH and the QNX
SMH_PERF ShmMDaemon	ShmMDaemon represents background process responsible for handling of shared memory (creation, permissions handling)
SMH_PERF Inter-Host Communication	Inter Host communication provides communication abstraction to Application SWCs deployed on different Hosts. The host-to-host communication can be based on shared memory and Ethernet. End-to-end protection (E2EP) is implemented for safety-related communication. E2E state for shared-memory based communication is ensured via memory FFI, and for Ethernet communication, E2E protection is implemented using standard profiles. Note: Interhost communication in S32G product is implemented over Ethernet (PFE based communication) for the first releases (v1, v2). Future releases will switch to shared memory communication.
SMH_PERF RTE	This block represents TTTech's MotionWise RTE layer.

Logical component	Functionality
SMH_PERF VerMod	Version Module is a component targeted for version verification. Version Module checks each component for its version number and build date and compares them to a list, which is created by the official build chain for a release. If deviations are found, the version module logs and records the components and their version number. Component responsibilities: • notify software version deviations • logging of any software version deviations • logging of the version numbers of each software component
SMH_PERFCLI Bootmode	Command line tool accessible through the command shell on PH. It is used to control the boot mode pins of the connected EQ and TDA SoCs and to trigger a reset of the connected SoCs.
SMH_PERF Command Shell	Component which represents command shell on PH.
SMH_PERF DevFeature	Enable and disable command shell upon diagnostic request via adaption channel in runtime
SMH_PERF Secure Boot	This logical entity representing the secure boot, where as soon as the root of trust is propagated to the PH, the relevant PH SWCs, QNX IFS are then loaded thought the secondary boot loader. As soon as the secure boot has verified and authenticate the proper root of trust, the A53 are released to allow the PH SWCs and QNX IFS are then loaded from the DDR. This entity is also tightly related to other security functionalities, namely enablement of secure software update and secure version control.
SMH_PERF Task Monitor	SMH_PERF Task Monitoring is responsible to provide program execution monitor functionality: • aliveness monitor, • deadline monitor, • program flow monitor. Violations of monitored entities detected by the SMH_PERF Task monitor are forwarded to SMH_PERF Error Handler Proxy and SMH_PERF Host Supervision Slave.
SMH_PERF Error Handler Proxy	SMH_PERF Error Handler Proxy is responsible for collecting and proxying the errors, reported from platform software and customer SW-Cs on the SMH_PERF, to the Error Handler Master.
SMH_PERF Host Supervision Slave	Host Supervision is based on a master-slave concept. Host Supervision Slave collects the status of Task Monitor as SMH_PERF and provides it to the Host Supervision Master.
SMH_PERF Startup	This logical component covers the "normal" startup of the MCU. More details are covered by the runtime diagrams.
SMH_PERF Infrastructure	Component which implements some/ip testability services based on K-matrix
SMH_PERF LUMMaster	LUM Master SSW (Standard Software) receives the LUM update commands (via SOMEIP) and download packages (via HTTPs) and orchestrates the LUM update sequence for the ECU by sending LUM commands to the LUM Clients via ara::com.

Logical component	Functionality
SMH_PERF LUMClientSH	LUMClientSH offers LUM Client ara::com service for LUM Master and handles update commands from LUM Master. Forwards the flash data to LUMFlasherSH via RTE.
SMH_PERF LUMClientPH	LUMClientPH offers LUM Client ara::com service for LUM Master and handles update commands from LUM Master and performs the flashing of eMMC partitions of S32G.
SMH_PERF LUMClientEyeQ	LUMClientEyeQ offers LUM Client ara::com service for LUM Master and handles update commands from LUM Master for EyeQ SW.
SMH_PERF LUMClientTDA	LUMClientTDA offers LUM Client ara::com service for LUM Master and handles update commands from LUM Master for TDA SW.
SMH_PERF PartitionManager	PartitionManager on PERF partition receives the partition management commands from LUM Master and forwards the commands to PartitionManager on SAFE partition. It also performs the transfer of the EyeQ and TDA SW.

[HCP2MEP-226376]

5.2.1 Communication

5.2.1.1 Inter-ECU Communication (Vehicle communication)

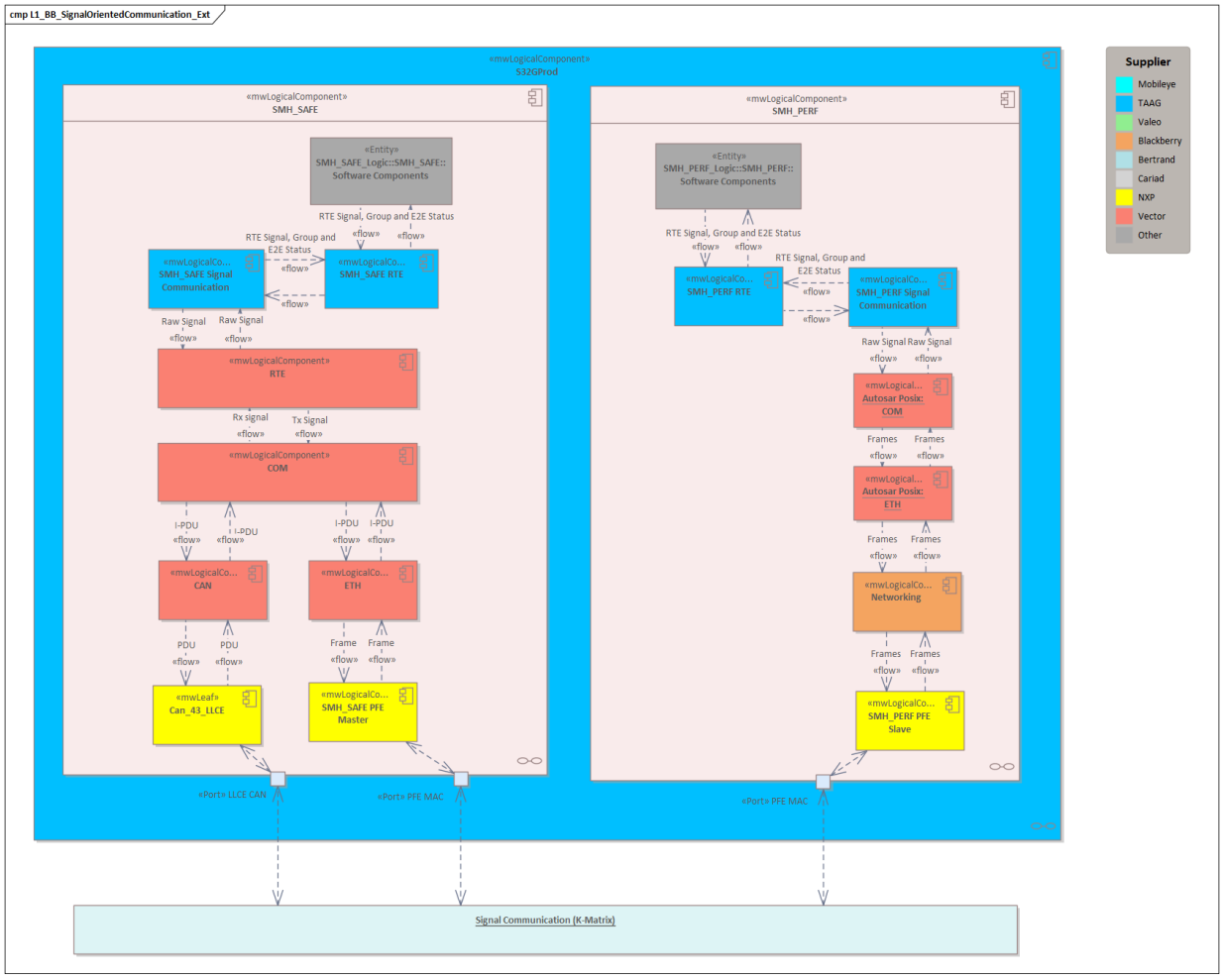
Vehicle Communication enables communication of MCU with other ECUs and covers follow use cases:

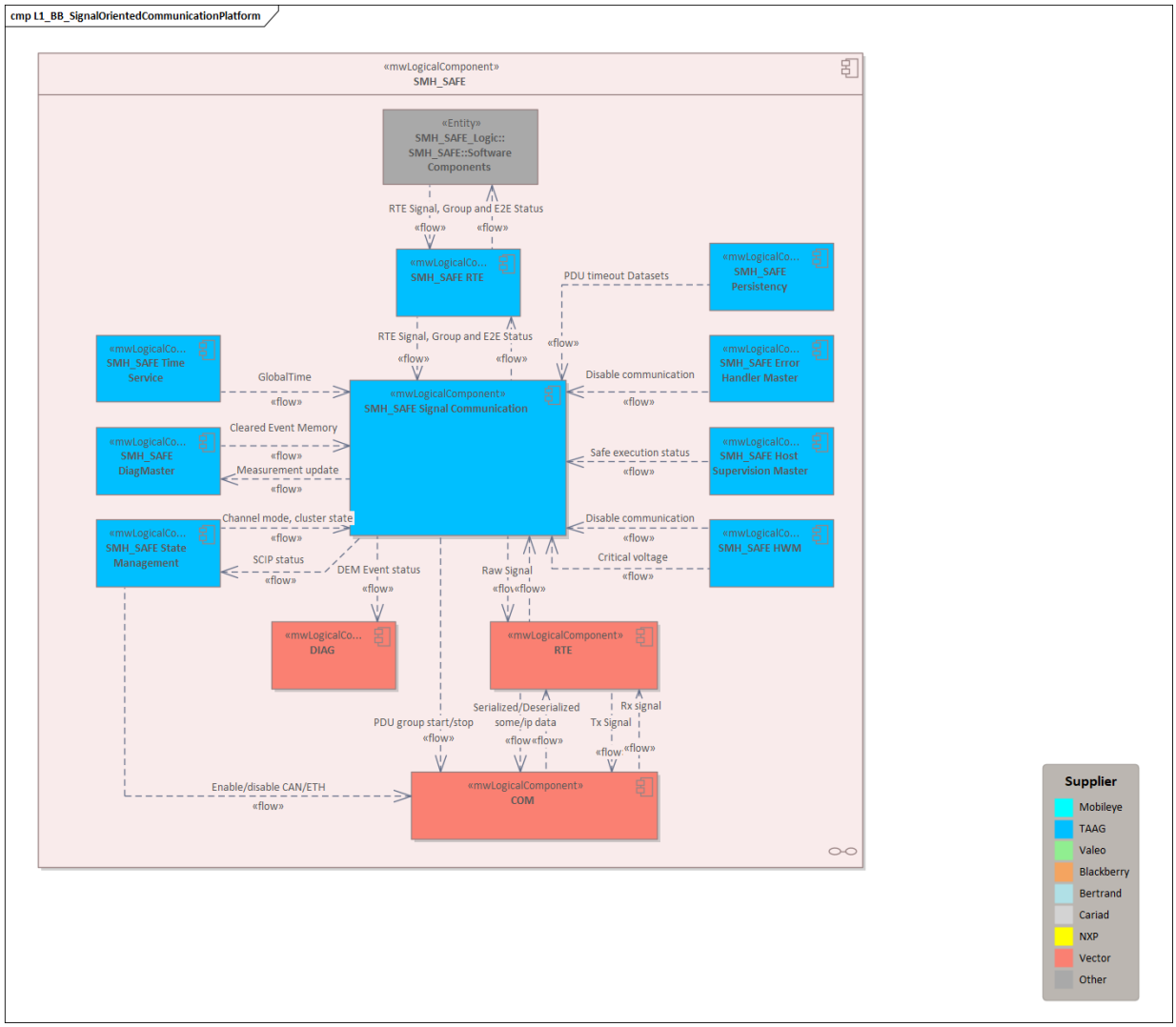
- Signal-based communication on SH (via CAN and Ethernet)
- Signal-based communication on PH (via Ethernet)
- Service-oriented communication on SH (via Ethernet)
- Service-oriented communication on PH (via Ethernet)
- REM use case

[HCP2MEP-220076]

5.2.1.1.1 Signal oriented communication

Signal Communication includes signal oriented communication based on K-matrix via CAN (on SH) and Ethernet (both SH and PH).


cmp L1_BB_SignalOrientedCommunication_Ext
«mwLogicalComponent»
S32GProd
«mwLogicalComponent»
SMH_SAFE
«Entity»
SMH_SAFE_Logic::SMH_SAFE::
Software Components
RTE Signal, Group and E2E Status
«mwLogicalCo...
SMH_SAFE Signal
Communication
«mwLogicalCo...
SMH_SAFE RTE
Raw Signal
«mwLogicalComponent»
RTE
Rx signal
Tx Signal
«mwLogicalComponent»
COM
I-PDU
«mwLogicalCo...
CAN
«mwLogicalCo...
ETH
PDU
Frame
«mwLeaf»
Can_43_LLCE
«mwLogicalCo...
SMH_SAFE PFE
Master
«Port» LLCE CAN
«Port» PFE MAC
«mwLogicalComponent»
SMH_PERF
«Entity»
SMH_PERF_Logic::SMH_PERF::
Software Components
«mwLogicalCo...
SMH_PERF RTE
«mwLogicalCo...
SMH_PERF Signal
Communication
«mwLogical...
Autosar Posix:
COM
Frames
«mwLogical...
Autosar Posix:
ETH
«mwLogicalCo...
Networking
«mwLogicalCo...
SMH_PERF PFE
Slave
«Port» PFE MAC
Signal Communication (K-Matrix)
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other



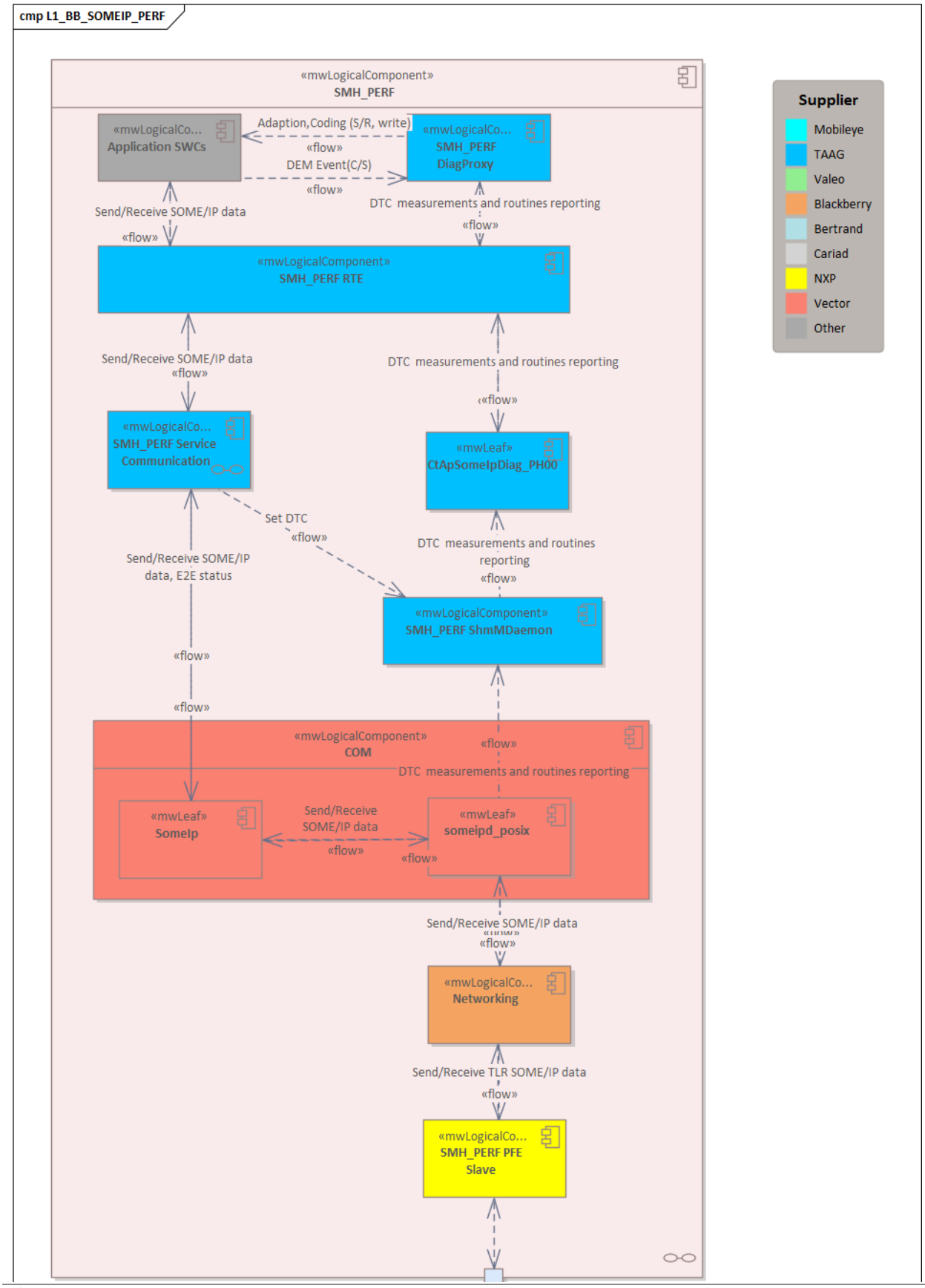
Signal Oriented Communication provides:

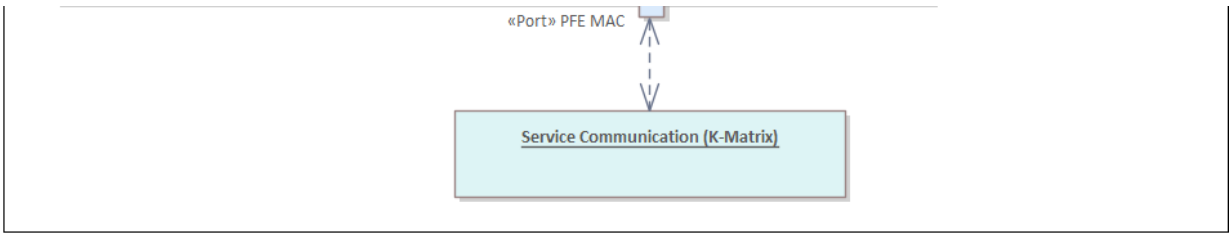
- Abstraction of AutoSAR BSW and COM stack for Application SWCs through RTE Data Elements
- Serialization and de-serialization of data
- End-to-end communication handling with AUTOSAR end-to-end transformers
- PDU state machines
- Message time stamping
- Raw-to-physical conversion and physical-to-raw conversion
- Network management
- PDU groups management

[S32GPRODP-296165]

5.2.1.1.2 Service oriented communication

5.2.1.1.2.1 Performance host



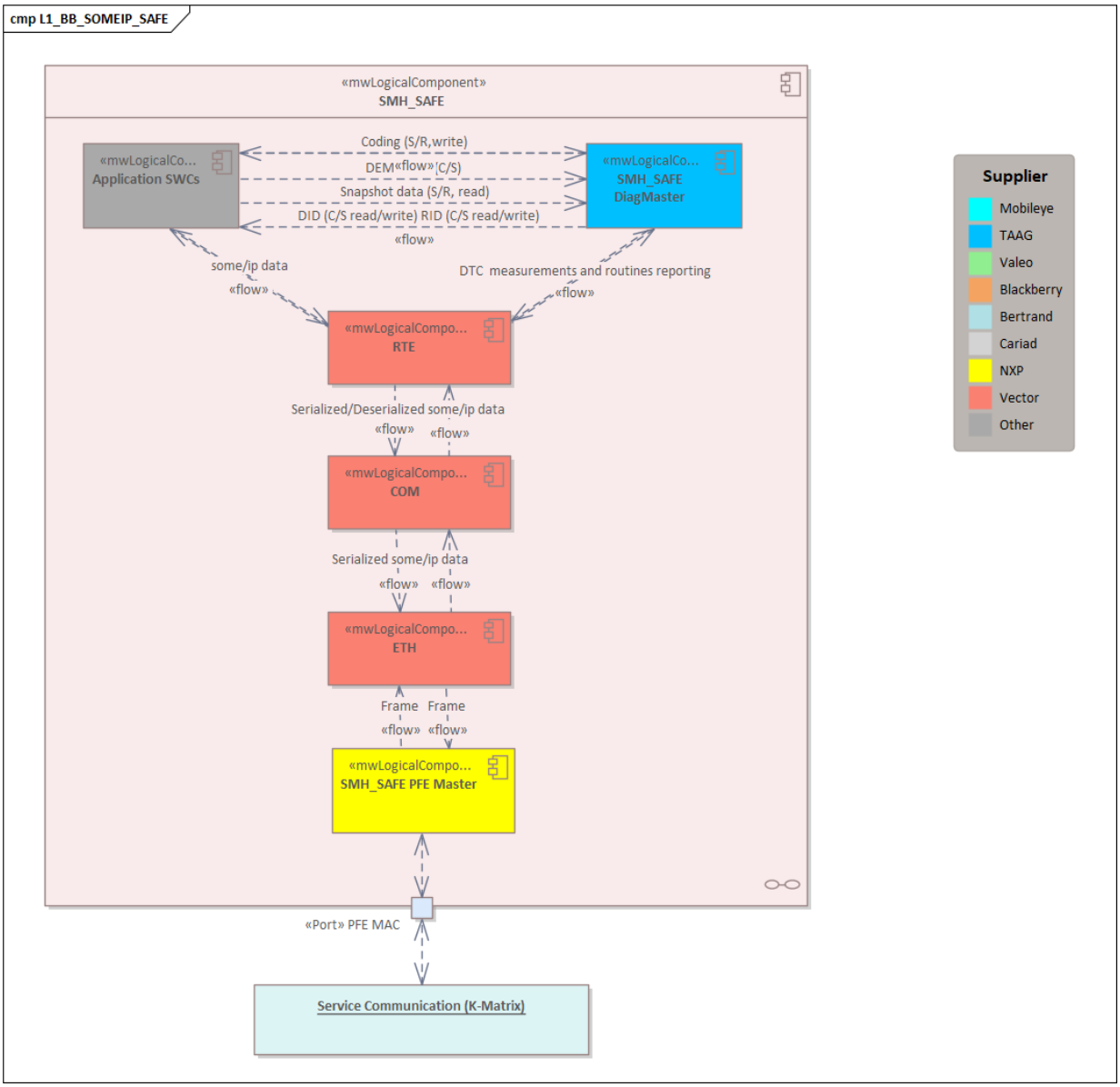


All SWC can be divided into following logical layers (units):

- Aplication layer - Classic AUTOSAR model
- RTE - TTTech Classic AUTOSAR
- Service Communication - Glue Layer
- Communication Management API (ara:com) - Vector Adaptive SOME/IP Stack

[S32GPRODP-296873]

5.2.1.1.2.2 Safety Host

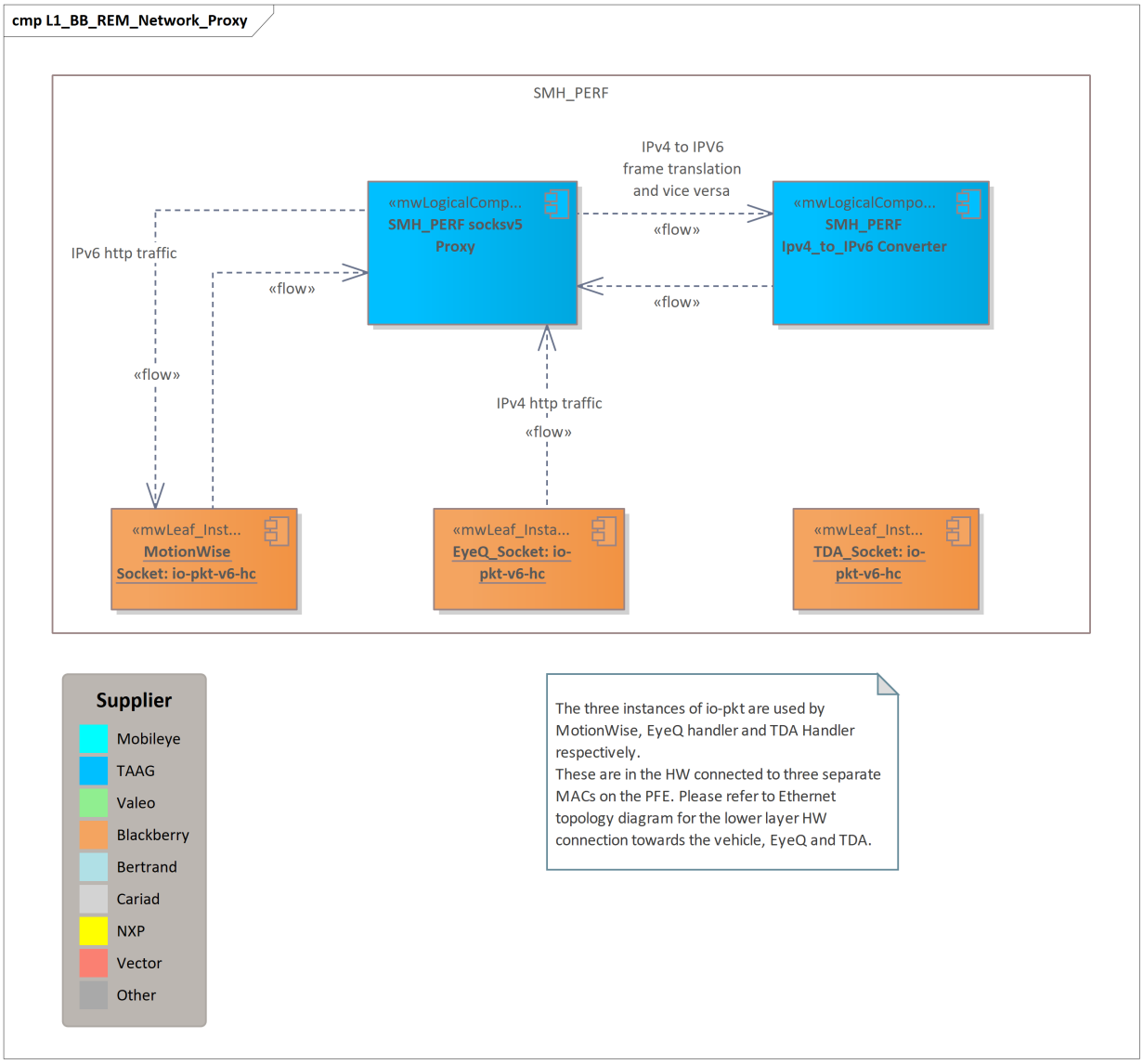


All SWC on SH use the following blocks (units) for SOME/IP communication:

- Aplication layer - Classic AUTOSAR model
- RTE - provided by the Vector
- COM - provided by the Vector
- ETH- provided by the Vector

[S32GPRODP-296880]

5.2.1.1.3 REM Network Proxy



The platform SW hosts an RFC 1928 compliant proxy REM_Netwrok_Proxy on the PH, which is the gateway between the REM client on EyeQ and the Cloud server. [S32GPRODP-296402]

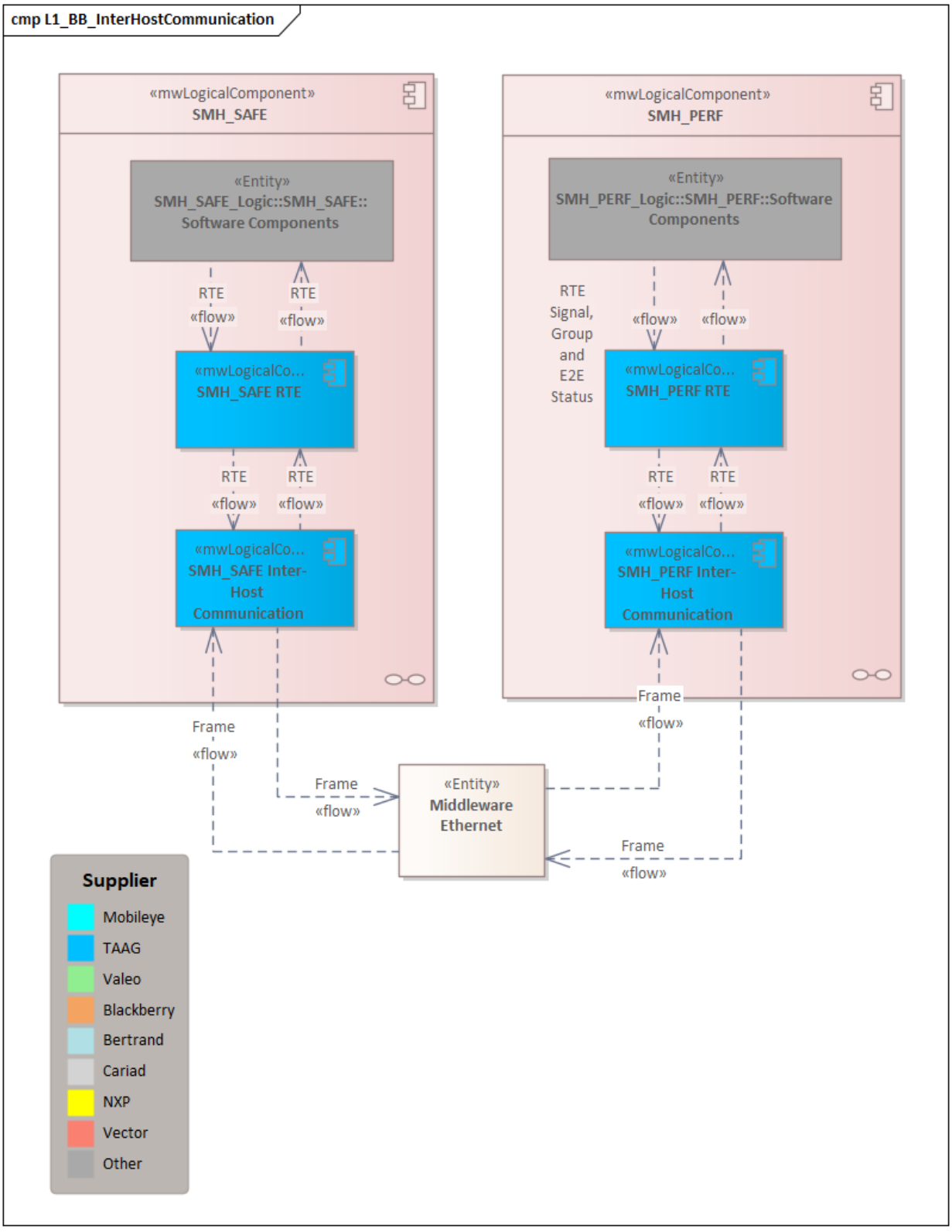
5.2.1.2 Intra-ECU communication

Intra-Ecu communication enables communication within MCU partitions and covers follow use cases:

- Communication inside partitions (Intra-Host)
- Communication between SH and PH partitions (Inter-Host)
- Proprietary communication between SH/PH and EyeQs/TDA4 (not in scope of TTTech platform)

[HCP2MEP-220147]

5.2.1.2.1 Inter-Host communication



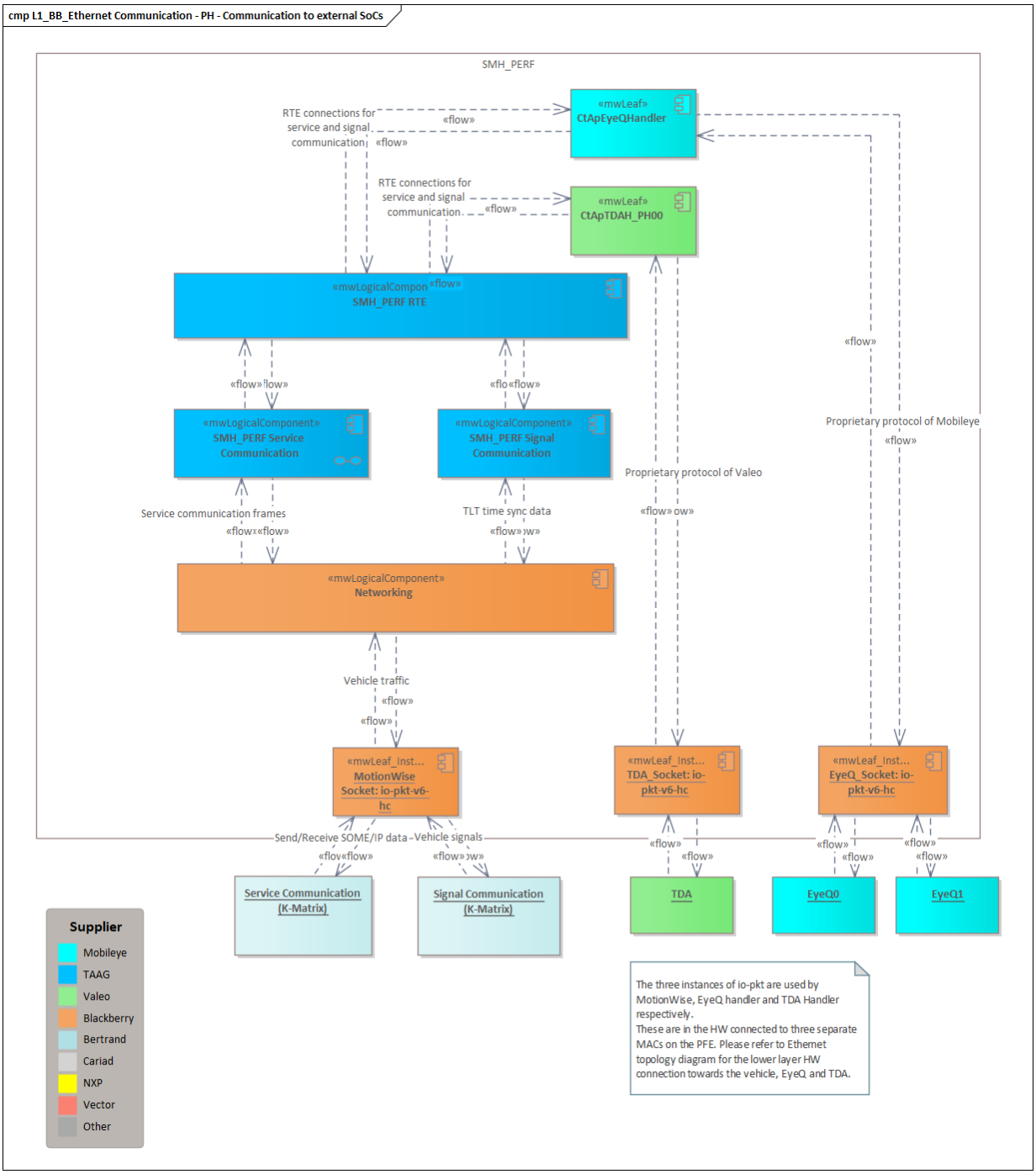
Inter Host communication provides communication abstraction to Application SWCs deployed on different Hosts, SoCs or MCUs over AutoSAR RTE that execute the MotionWise platform software.
End-to-end protection (E2EP) is implemented for safety-related communication. E2E state for shared-memory based communication is ensured via memory FFI, and for Ethernet communication, E2E protection is implemented using standard profiles.

The host-to-host communication can be based on shared memory and Ethernet (TSN):

- Inter-host communication via Ethernet is designed specifically for Time-Sensitive Networking, but it can also provide best-effort communication. MotionWise uses Ethernet frame optimization mechanisms such as frame grouping and segmentation.
- Shared memory communication is available to MotionWise hosts that are deployed on the same SoC in a common address space.

Note: Interhost communication is implemented over Ethernet (PFE based communication) for the first releases (v1, v2). Future releases will switch to shared memory communication.
[S32GPRODP-296183]

5.2.1.2.2 PH - Communication to driving and parking SoCs

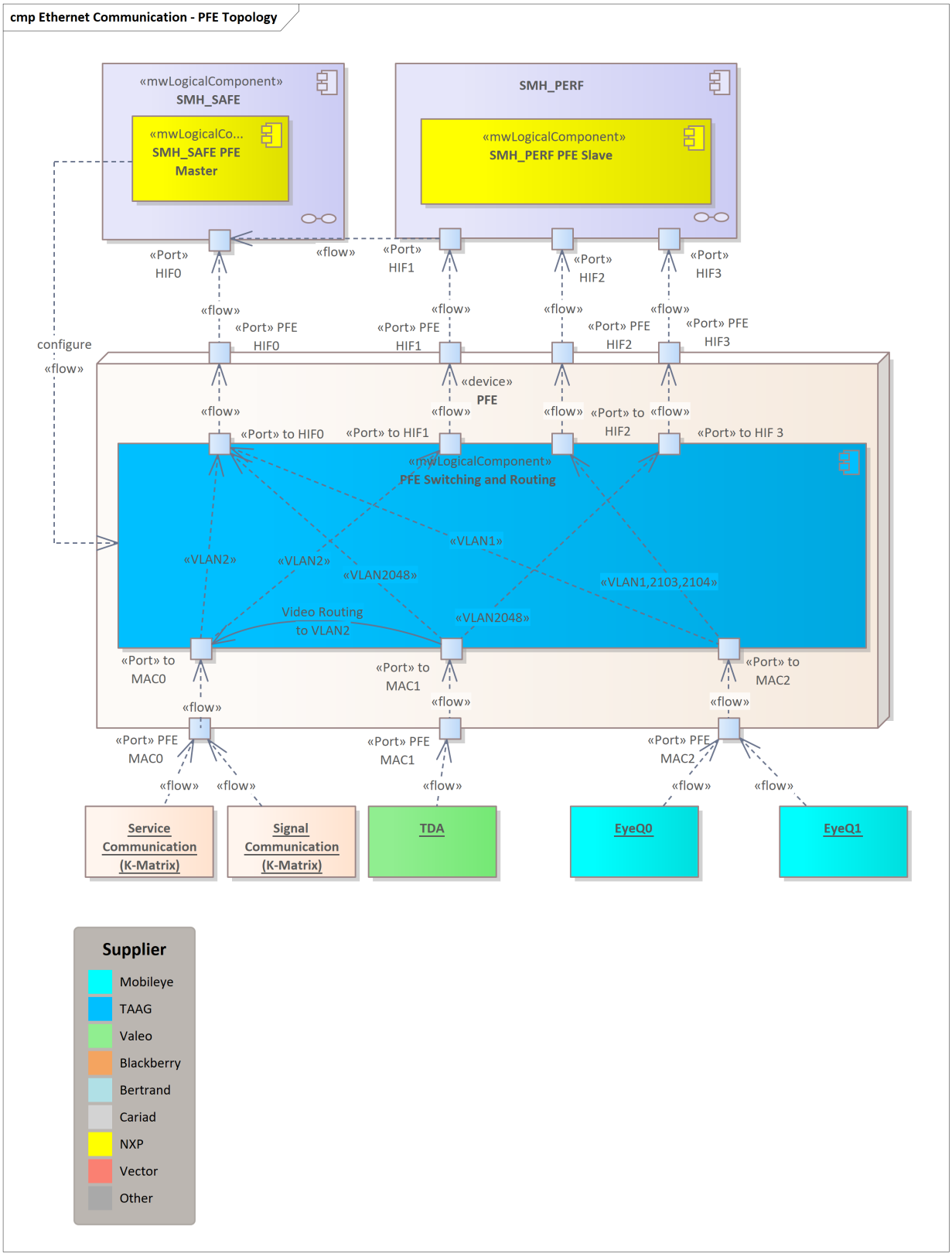


MotionWise platform, the EyeQHandler and the TDA Handler have separate instances of io-pkt to communicate over Ethernet. [S32GPRODP-296410]

For Vehicle communication, the Handlers use the platform SW RTE and the communication stack like any other WSWC on the platform. [S32GPRODP-296412]

The communication between the Handlers and the external SoCs EyeQs and TDA is not via the platform communication stack, and the protocol is proprietary to mobileye and Valeo respectively. [S32GPRODP-296411]

5.2.1.3 Ethernet BUS (PFE Topology)



S32G PFE block on S32G SoC is configured to operate as an Ethernet switch. In order to isolate traffic logical

interfaces using VLANs, so called VLAN bridges are defined with VLAN ID and member MAC and HIF ports. This enables only Ethernet frames with specific VLAN tags to be exchanged among Ethernet MAC ports and HIF (Host Interfaces).

As an example: PFE MAC0 is set to carry only vehicle traffic with VLAN=2. Ethernet frames tagged with VLAN=2 (included in the Ethernet header), will be forwarded only among PFE MAC0 and HIF 0 and HIF1. No other traffic will pass this port. In addition to VLAN=2 , the MAC0 port can also include filtering functions which is not shown here, and can forward traffic relevant only for HIF0 or HIF1 or both. This helps to reduce overhead on network stack servicing HIF0 and HIF1.

Other VLANs (2104, 2103, 1, 2048) communicate via MAC1 and MAC2 ports, and are used for intra-ECU interfacing. VLAN ID=2048 is used for TDA4 function handling while VLAN ID=2104 is used for EQ function handling. VLAN ID=1 is used to handle for any untagged (no-VLAN) traffic.
The selection of interfaces is architected to enable separation and isolation of resources dedicated to vehicle, EQ and TDA functions.

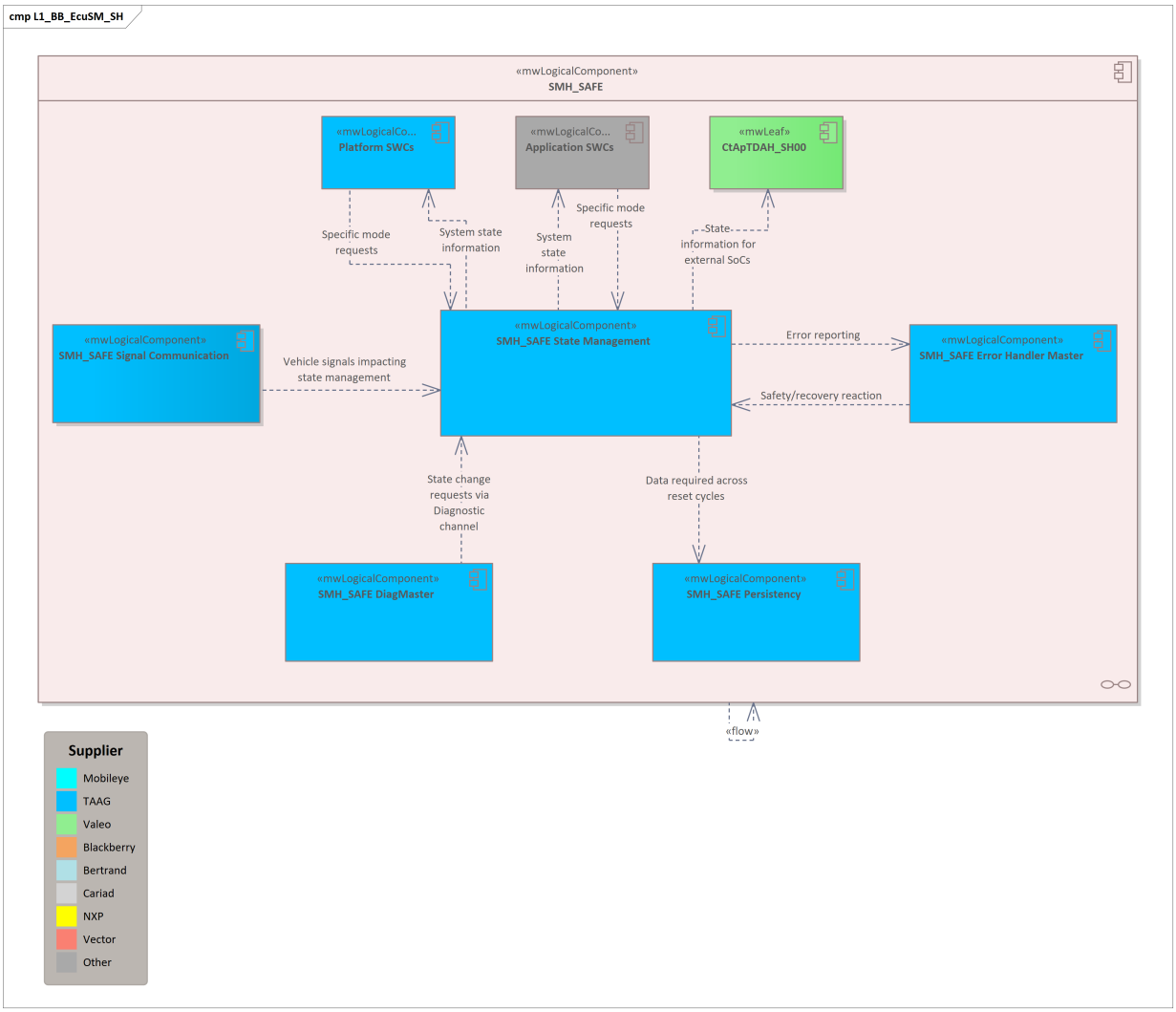
PFE MAC1 also includes a routing function to PFE MAC0 which enables unidirectional routing of IPv6 video traffic sent over VLAN=2048 and translated into PFE MAC2 VLAN=2. There is no other interaction of vehicle functions with TDA4 SoC functions. **[**HCP2MEP-220175 **]**

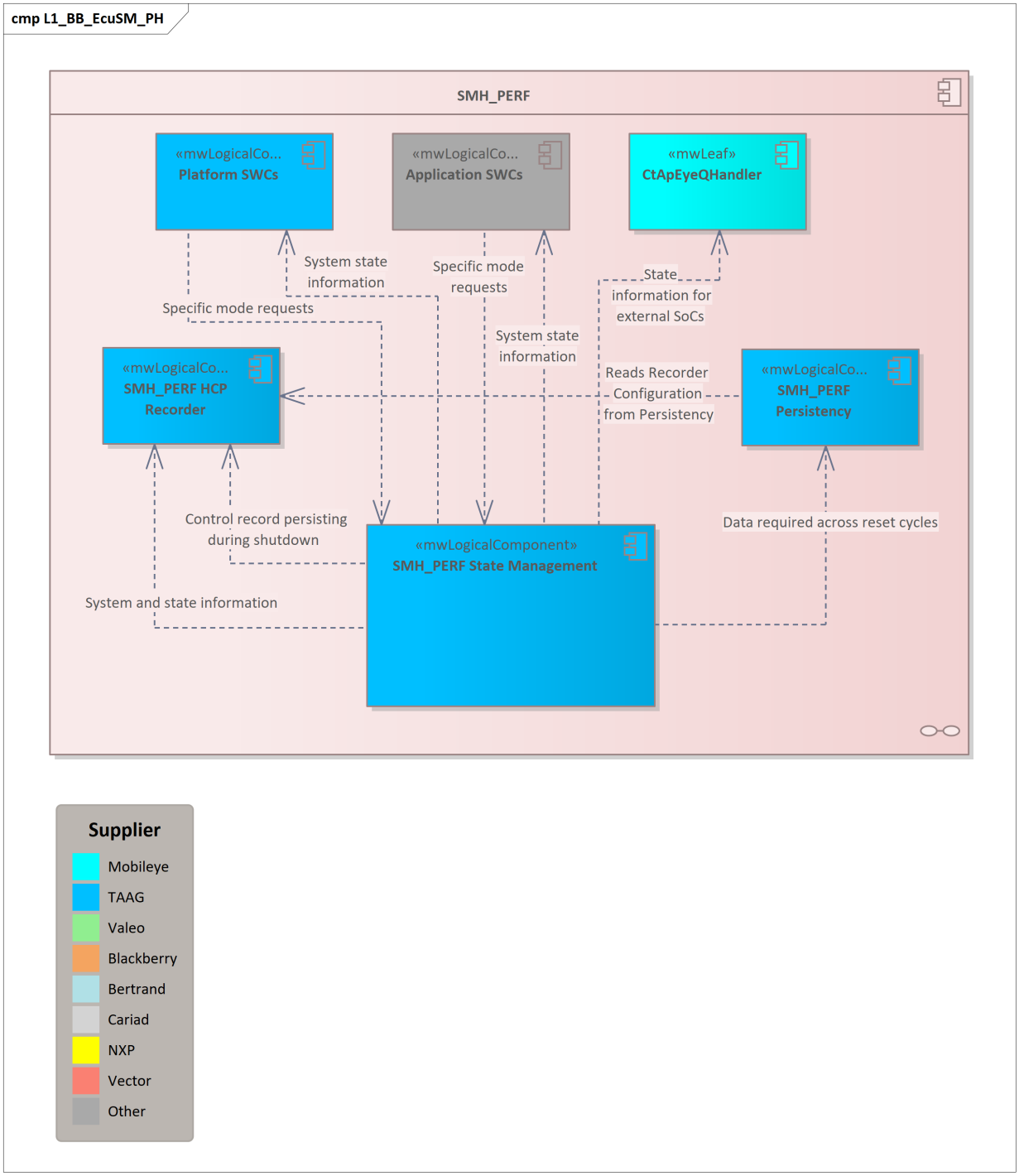
5.2.2 ECU state management

EcuStateManagement (EcuSM) specifies the handling of HCP2 system modes like application, degraded and UDS update, and power cycle states like startup, running and shutdown. The platform function covers the following architectural tasks:

- Wakeup handling of the ECU
- Startup of the ECU, including all processors, memory units and external interfaces
- Startup of software components
- Recovery from platform or software component faults by (partial) reset.
- Start of communication interfaces
- Coordinated shutdown of communication interfaces (network management)
- Shutdown of software components
- Shutdown of the ECU, including all processors, memory units and external interfaces
- Shutdown of the ECU, based on Online Remote Update (ORU) status
- Entering degradation or defect mode in case of errors
- Dynamometer Operation Mode control
- HIL-Mode activation and deactivation
- Development mode activation and deactivation
- Improper Module shutdown monitoring

[S32GPRODP-296248 **]**


cmp L1_BB_EcuSM_SH
«mwLogicalComponent»
SMH_SAFE
«mwLogicalCo...
Platform SWCs
«mwLogicalCo...
Application SWCs
«mwLeaf»
CtApTDAH_SH00
Specific mode requests
System state information
System state information
Specific mode requests
State information for external SoCs
«mwLogicalComponent»
SMH_SAFE Signal Communication
Vehicle signals impacting state management
«mwLogicalComponent»
SMH_SAFE State Management
Error reporting
Safety/recovery reaction
«mwLogicalComponent»
SMH_SAFE Error Handler Master
State change requests via Diagnostic channel
Data required across reset cycles
«mwLogicalComponent»
SMH_SAFE DiagMaster
«mwLogicalComponent»
SMH_SAFE Persistency
«flow»
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other



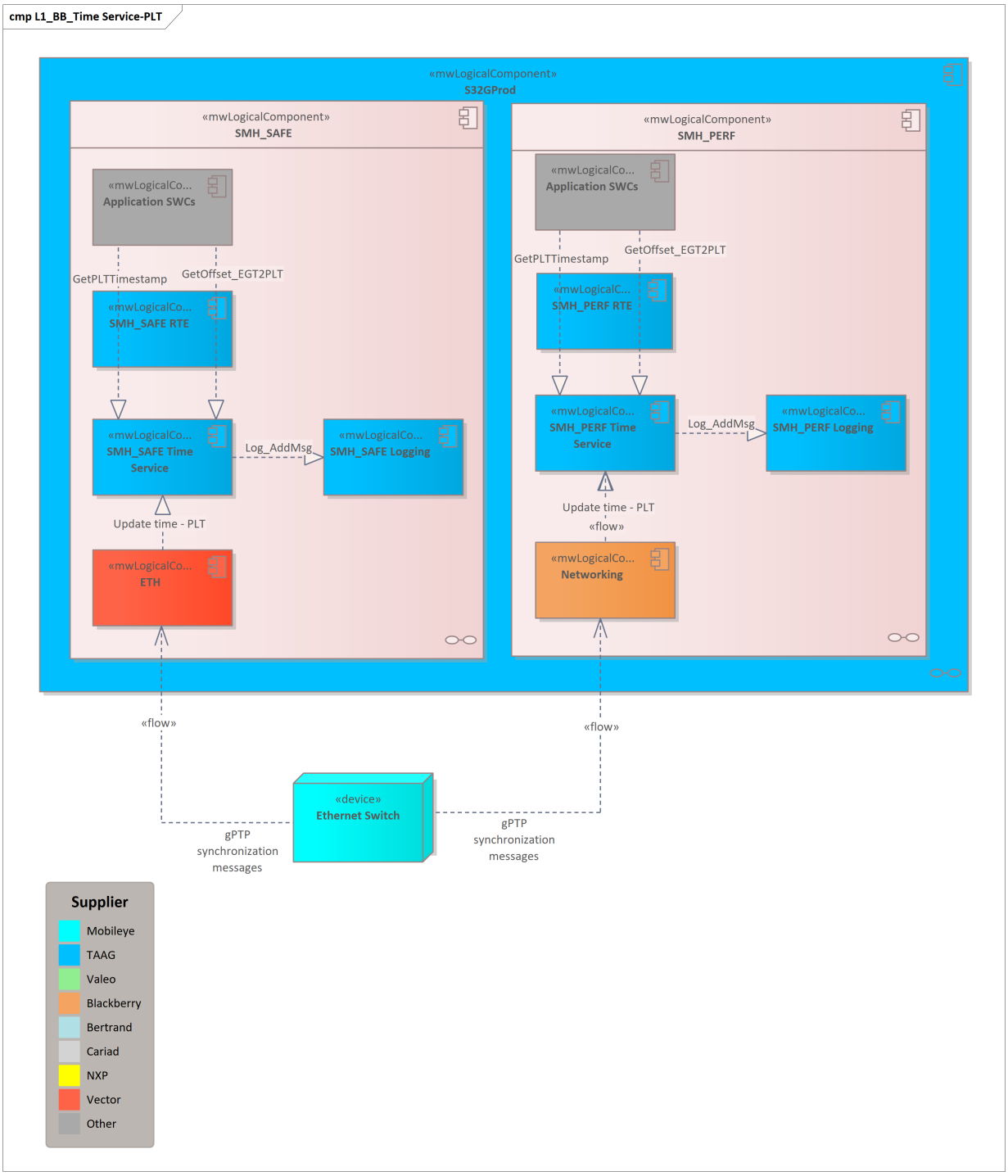
5.2.3 Time service

Time services provide synchronization with time domains like Precise Local Time (Vehicle time/PLT), Trustworthy Local Time (TLT), Authentic Time, Precise UTC. It makes available these time domains via platform interface to all platform and application SWCs.

Also time service implements mechanism for synchronization with EyeQs(Driving Soc) and TDA (Parking Soc). [S32GPRODP-296825]

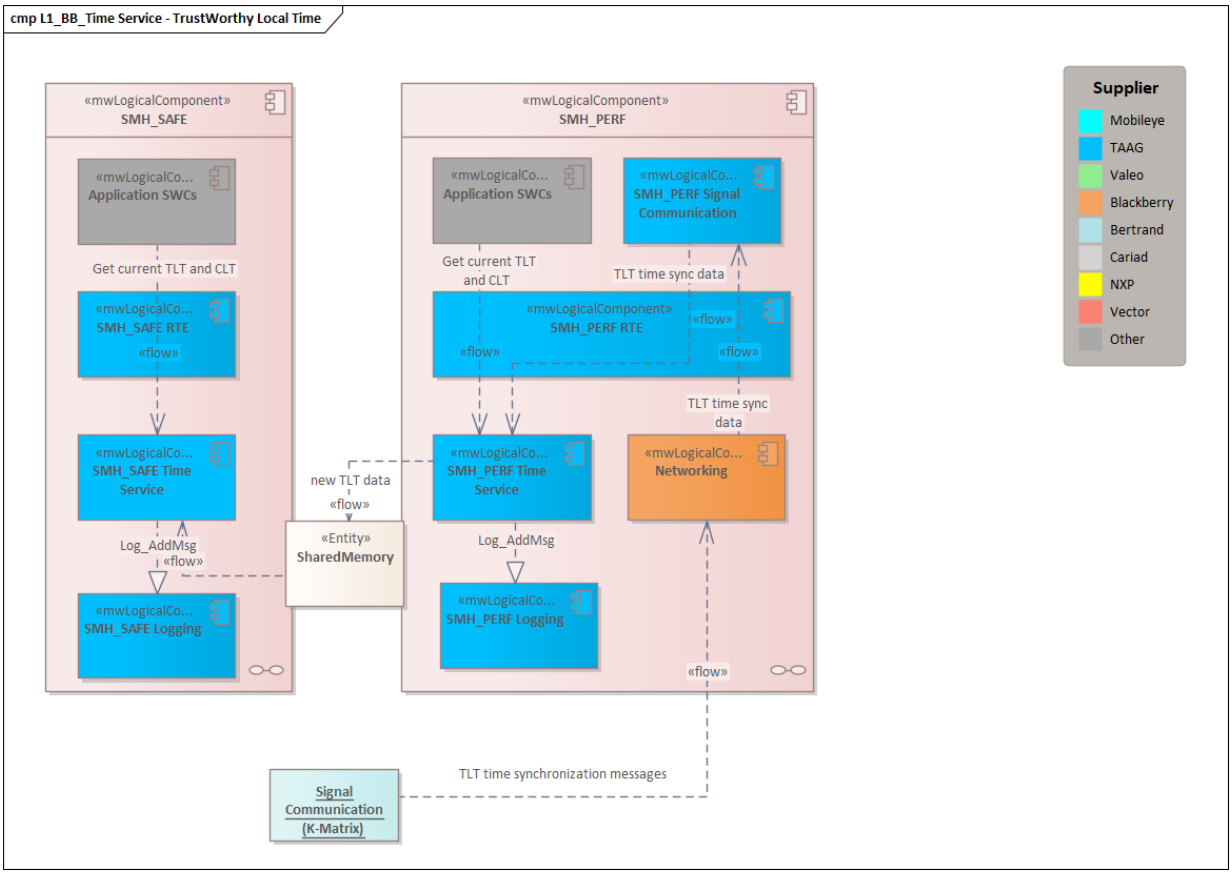
5.2.3.1 Precise Local Time (PLT)

Precise Local Time (PLT) is continuous, monotonic representation of vehicle time, which is used to have accurate and precise sensor data fusion. It has minimum resolution of at least 1 nanosecond and is provided in QM quality. High-level architecture of PLT feature of Time Service is depicted below. **[**S32GPRODP-296826 **]**



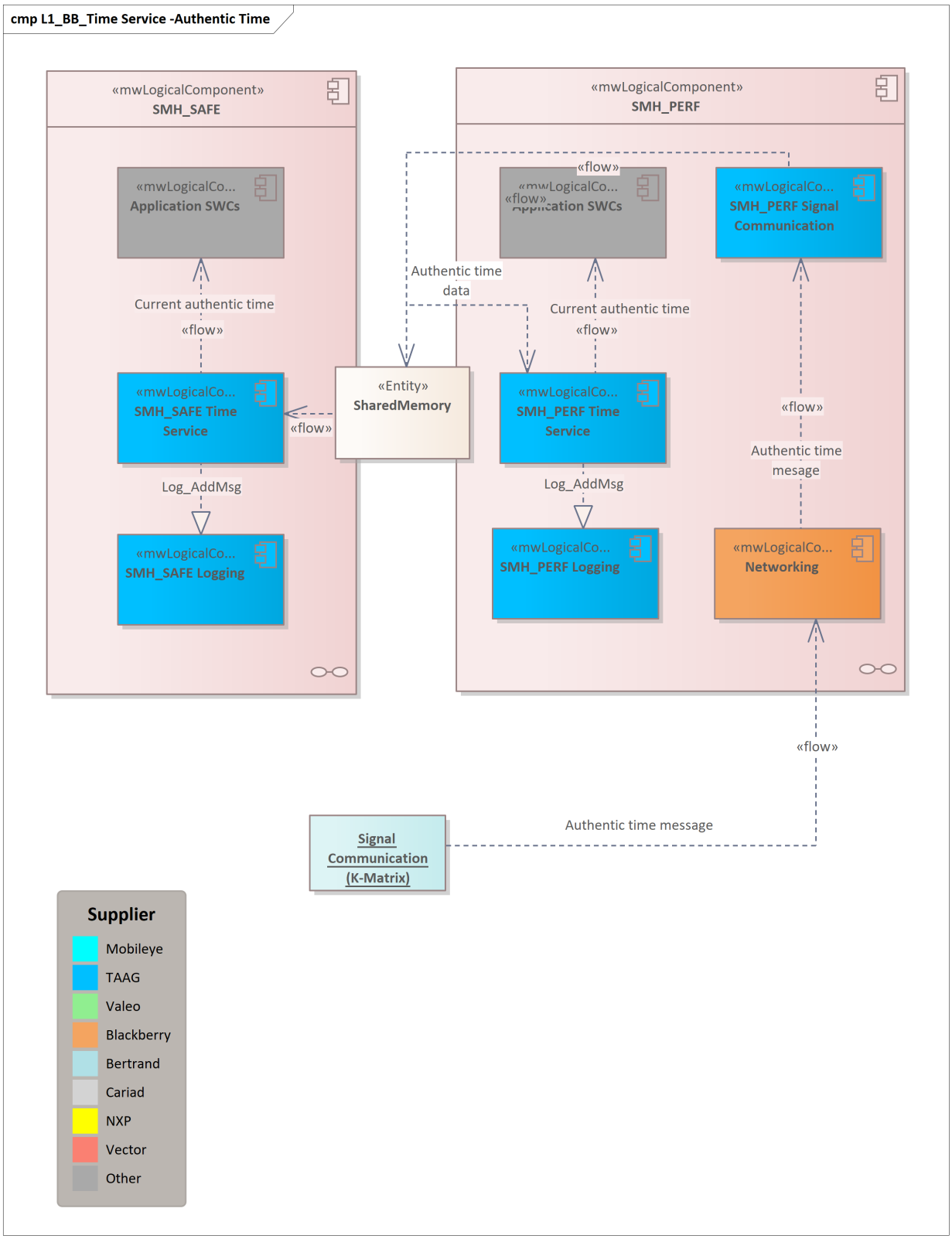
5.2.3.2 Trustworthy Local Time (TLT)

Trustworthy Local Time (TLT) is continuous, monotonic representation of vehicle time, which is used in sensor data fusion purposes. It has minimum resolution of at least 1 nanosecond and is provided in ASIL B quality.
High-level architecture of TLT feature of Time Service is depicted below. **[**S32GPRODP-296831 **]**



5.2.3.3 Authentic Time

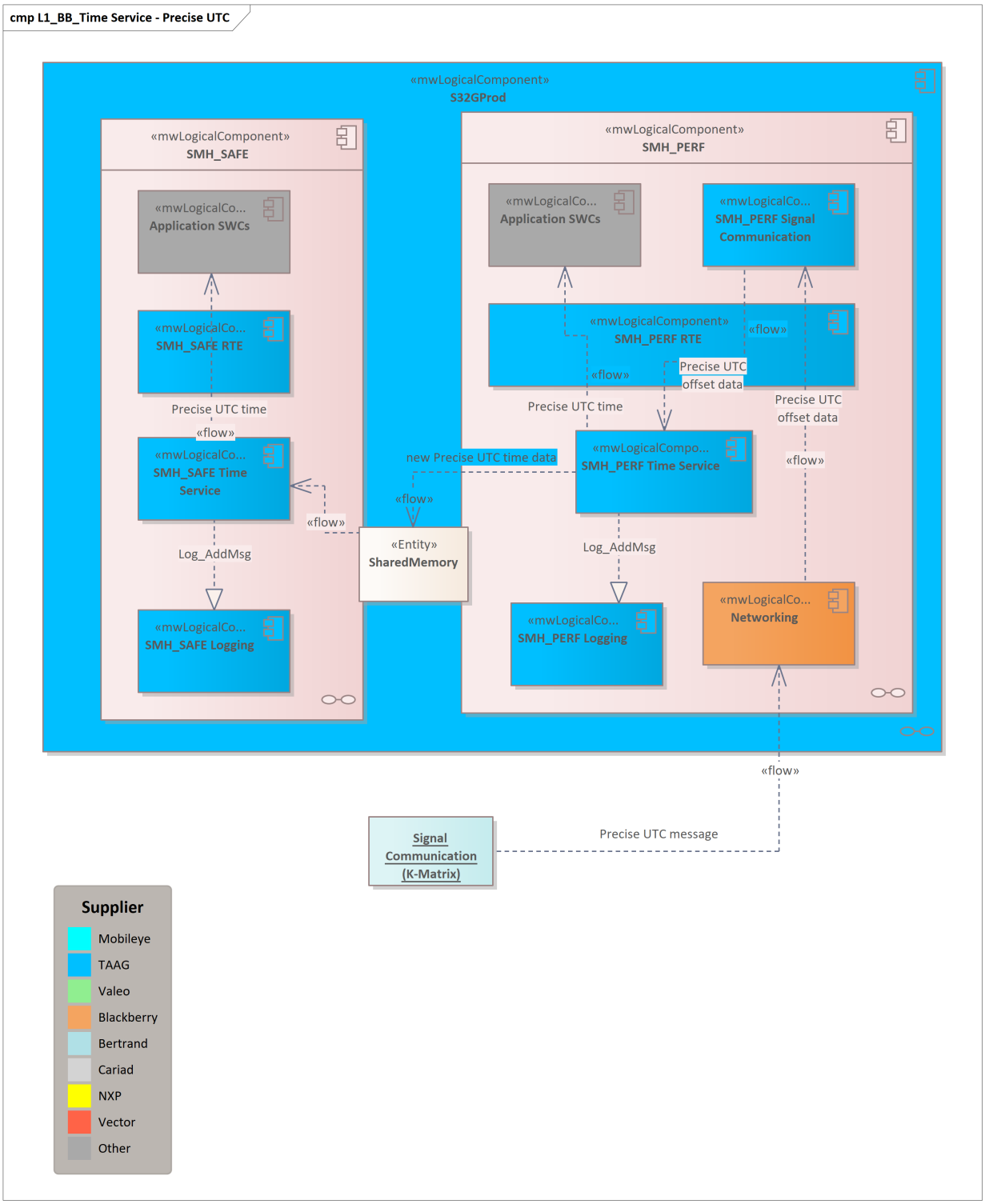
Authentic Time (ATi) is continuous, monotonic representation of time, which is used to put time limit on usage of certain vehicle functions. It has minimum resolution of 1 second and is provided in QM quality.
High-level architecture of ATi feature of Time Service is depicted below: **[**S32GPRODP-296827 **]**


cmp L1_BB_Time Service -Authentic Time
«mwLogicalComponent»
SMH_SAFE
«mwLogicalComponent»
SMH_PERF
«mwLogicalCo...
Application SWCs
«mwLogicalCo...
Application SWCs
«mwLogicalCo...
SMH_PERF Signal Communication
«flow»
«flow»
Authentic time data
Current authentic time
«flow»
Current authentic time
«flow»
«mwLogicalCo...
SMH_SAFE Time Service
«flow»
«Entity»
SharedMemory
«mwLogicalCo...
SMH_PERF Time Service
«flow»
Authentic time mesage
Log_AddMsg
Log_AddMsg
«mwLogicalCo...
SMH_SAFE Logging
«mwLogicalCo...
SMH_PERF Logging
«mwLogicalCo...
Networking
«flow»
Signal Communication (K-Matrix)
Authentic time message
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

5.2.3.4 Precise UTC

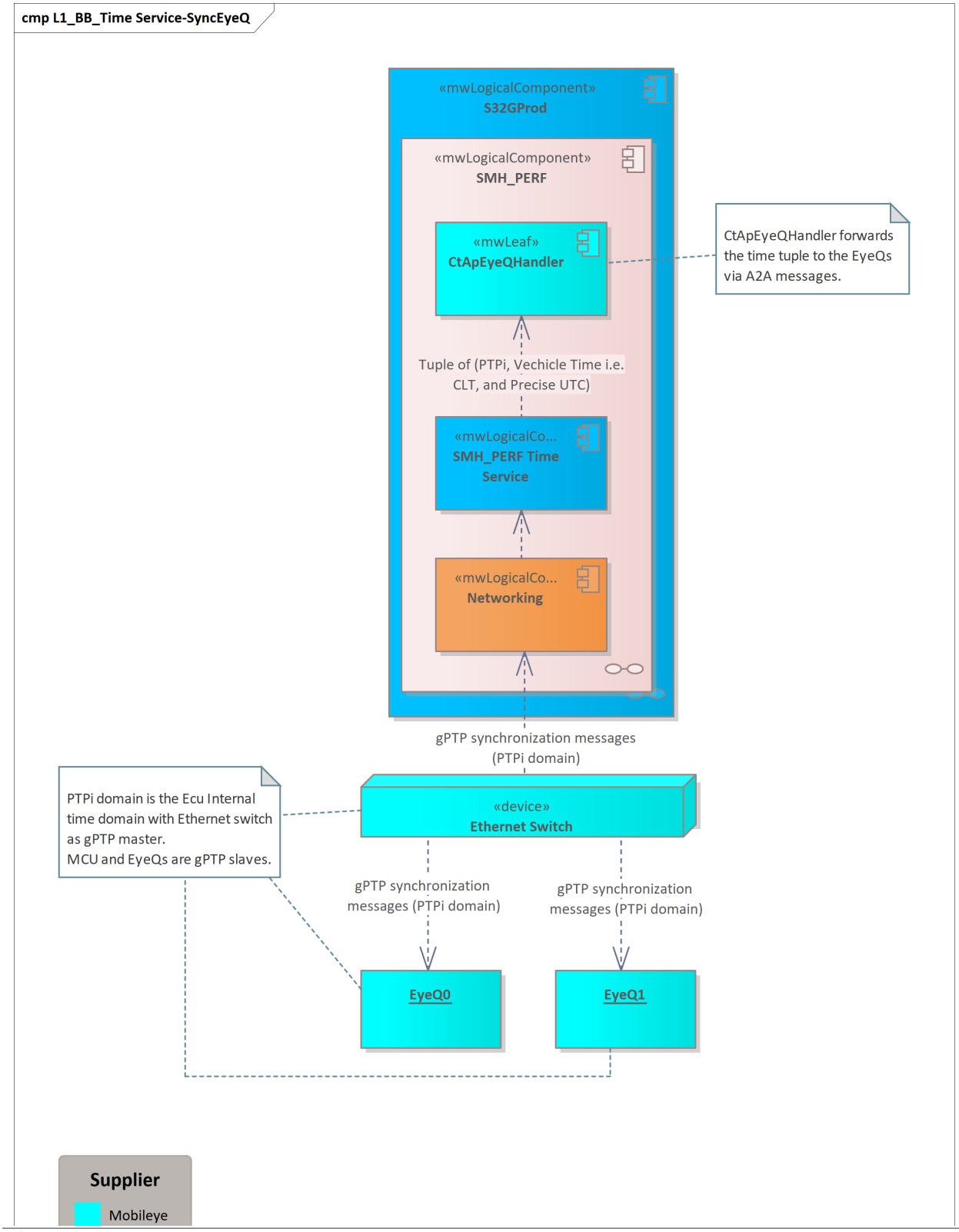
PreciseUTC is continuous, monotonic representation of Coordinated Universal Time. It has minimum resolution of 1 nanosecond and is provided in QM quality.

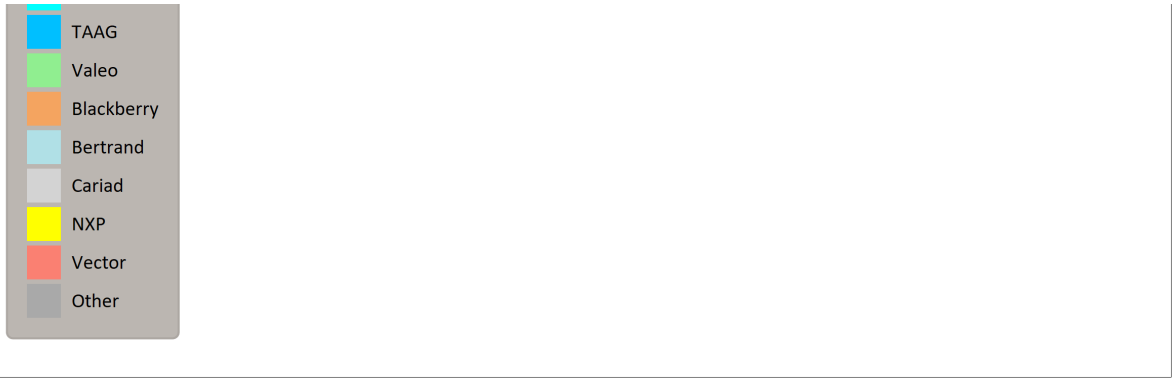
High-level architecture of PreciseUTC feature of Time Service is depicted below: [S32GPRODP-296828]



5.2.3.5 TimeSync with EyeQs

Time synchronization with EyeQs is established using gPTP synchronization with the time domain PTPi (ECU internal time) with Ethernet switch as master. S32G and the EyeQs are slaves of the PTPi domain.
The TimeService creates a time tuple of the PTPi and Vehicle time domains and forwards it the EyeQ handler SWC. The EyeQ handler SWC then forwards the time tuple to the EyeQs. **[**S32GPRODP-296829 **]**

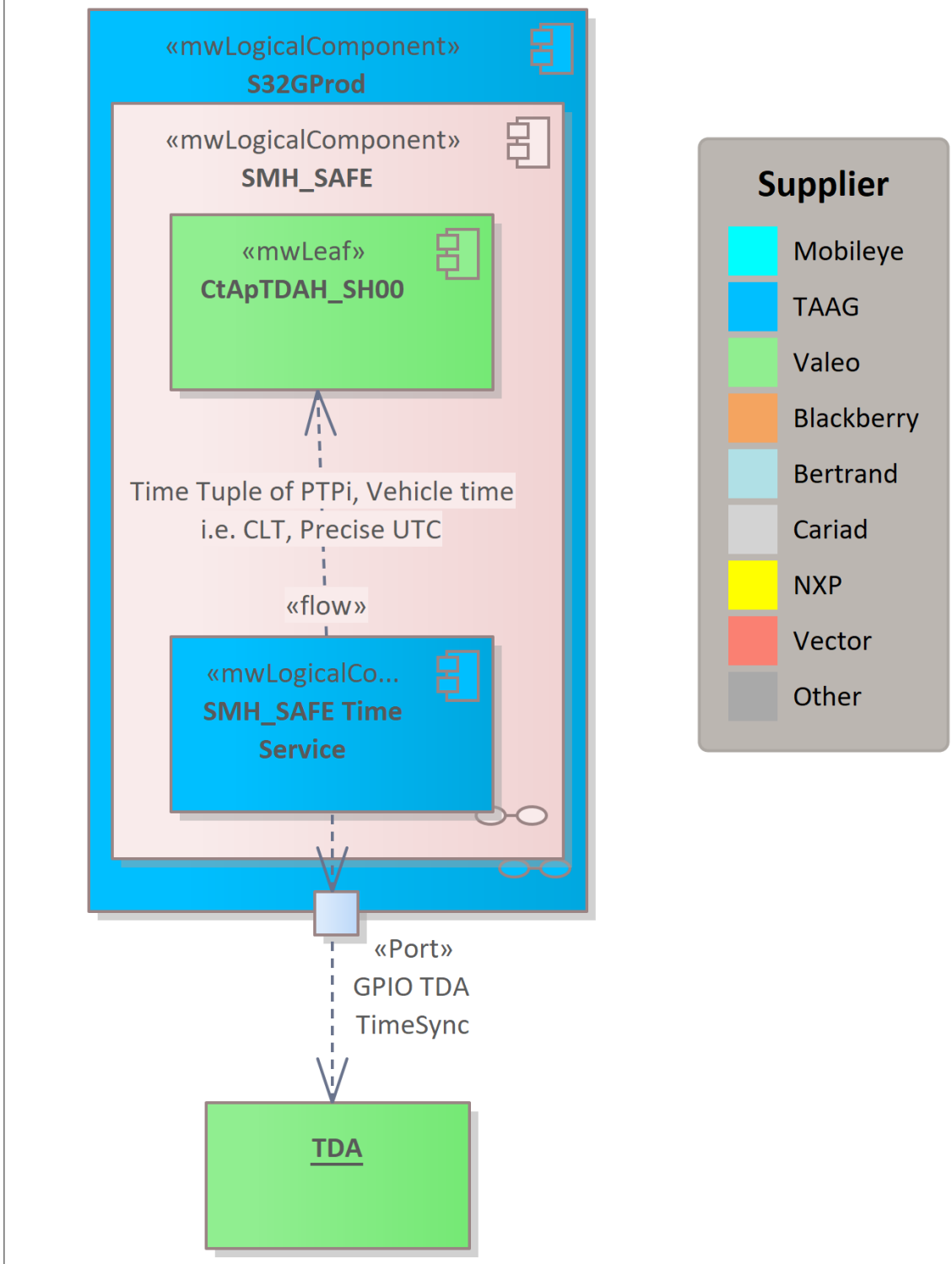




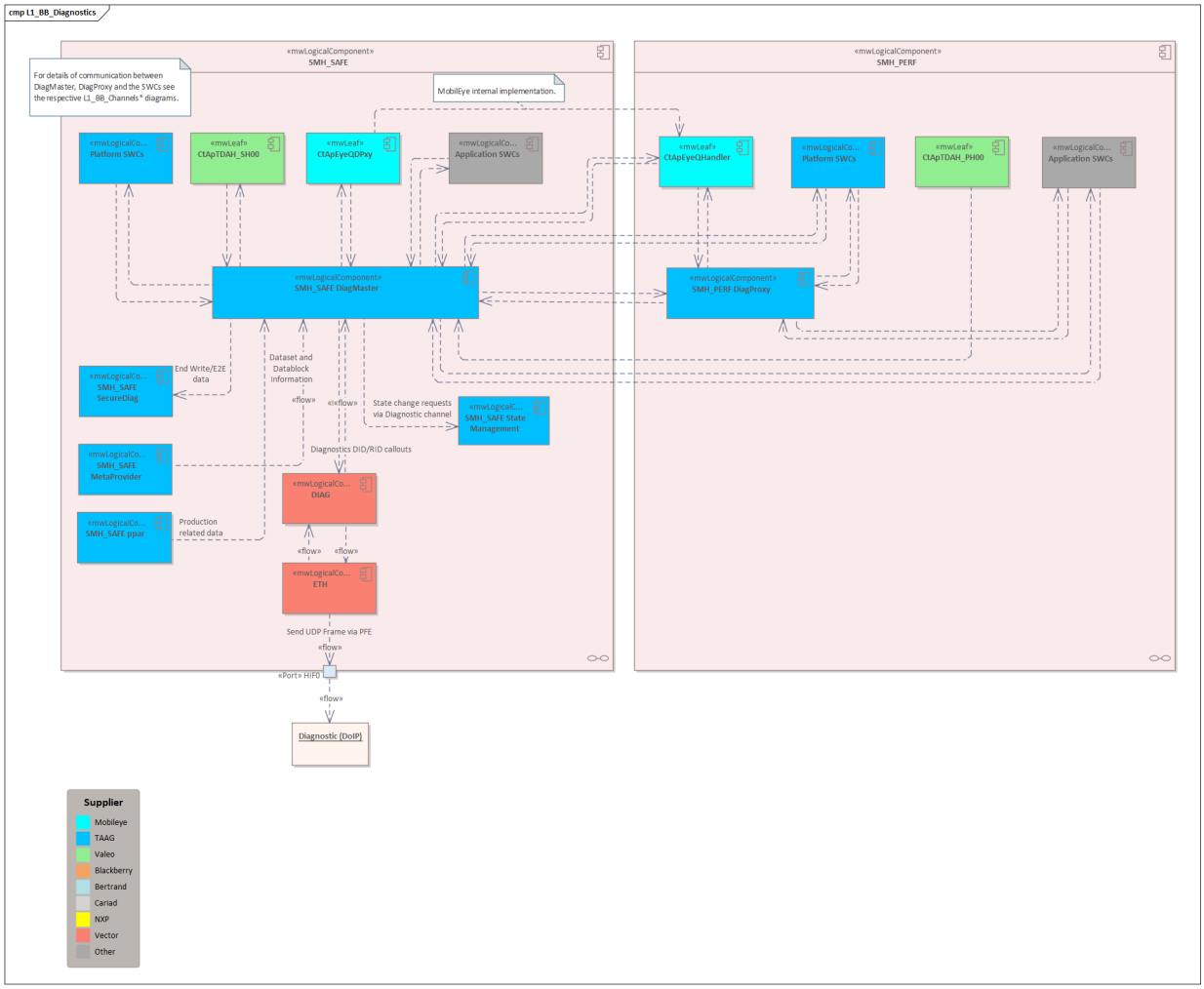
5.2.3.6 TimeSync with TDA

Time synchronization with TDA is done using GPIO.
Time Service component toggles the pin and captures the tuple of PTPi and Vehicle time at the instance of toggle and provides it to the TDA handler via port interface. [S32GPRODP-296830]

cmp L1_BB_Time Service-SyncTDA



5.2.4 Diagnostics



The communication with the diagnostic client and error memory management is done by the Autosar BSW (DoIP, DCM, DEM).

The DiagMaster component is integrated on the safety partition to handle the diagnostic jobs from the DCM and distribute them to the platform SWCs, optional application SWCs, EQ handler and TDA handler instances. The communication channels (using RTE) depend on the type of diagnostic information.

The DiagMaster component also forwards information to the DiagProxy if it needs to be processed remotely (for details see the specific diagrams below).

The DiagMaster component receives the Diagnostic Events from platform and optional application SWCs and EQ handler instance on the safety partition and forwards them to the DEM.

The DiagProxy component is integrated on the the performance partition to handle requests locally on this partition (for details see the specific diagrams below).

The DiagProxy component receives the Diagnostic Events from platform and optional application SWCs, EQ handler and TDA handler on the performance partition and forwards them to the DiagMaster component.

The SecureDiag component provides the security protection functionality to DCM and processes security related DIDs.

The Metaprovider and ppar component provide information items used in DIDs

DiagMaster communicates with the StateManagement component to trigger state changes and read status information **[**S32GPRODP-296750 **]**

The diagnostic subsystem supports the following services:

Diagnostic Services	Description
ECU Reset(0x11)	The ECUReset service is used by the client to request a server reset.
DiagnosticSessionControl(0x10)	UDS session functionality
ReadDTCInformation(0x19)	Read the content of the error memory.
ReadDataByIdentifier(0x22)	The ReadDataByIdentifier service allows the client to request data record values from the server identified by one or more dataIdentifiers
CommunicationControl(0x28)	The purpose of this service is to switch on/off the transmission and/or the reception of certain messages of (a) server(s) (e.g. application communication messages).
WriteDataByIdentifier(0x2E)	The WriteDataByIdentifier service allows the client to write information into the server at an internal location specified by the provided data identifier.
RoutineControl(0x31)	The RoutineControl service is used by the client to execute a defined sequence of steps and obtain any relevant results.
TesterPresent(0x3E)	This service is used to indicate to a server (or servers) that a client is still connected to the vehicle.
ControlDTCSetting(0x85)	The ControlDTCSetting service shall be used by a client to stop or resume the updating of DTC status bits in the server(s).

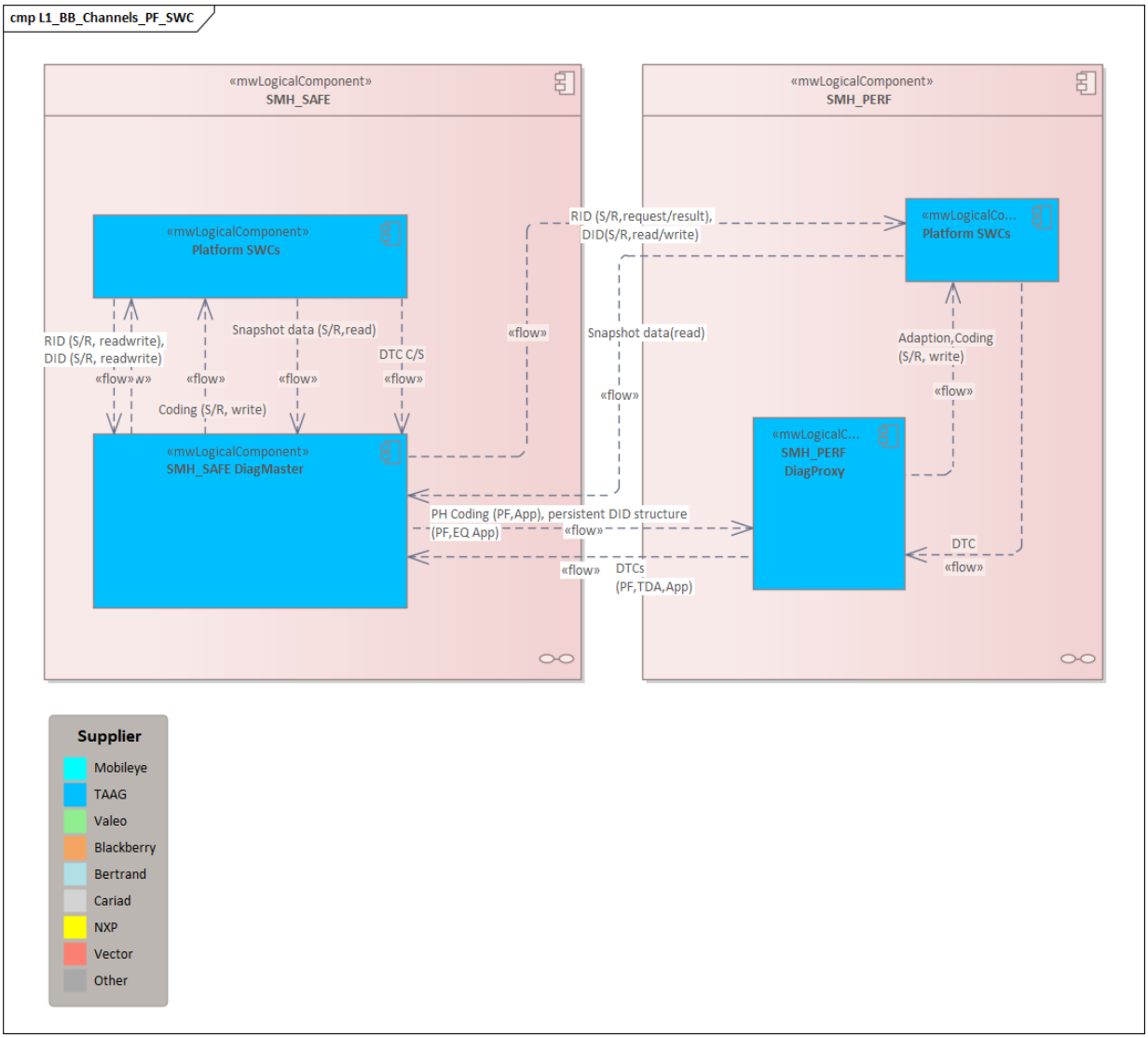
[S32GPRODP-296260 **]**

The diagnostic service handles the requests from an external diagnostic client (diagnostic tester) and orchestrates the respective actions within the ECU.
The diagnostic platform service handles the following classes of diagnostic objects:

Diagnostic Class	Description
Identification	Retrieve identification data from platform SWCs, EQ and TDA component.
Messwerte (Measurement Values)	Retrieve measurement values from platform SWCs, EQ and TDA component.
Routines	Start/Stop/RequestStatus of routines for platform SWCs, EQ and TDA component.
Coding	Set the coding configuration for platform SWCs, EQ and TDA component.
adaption	Set adaption configuration for platform SWCs, EQ and TDA component.

The details concerning the diagnostic objects are provided by the customer in diagnostic tables (CDD or DEXT format). **[**S32GPRODP-296256 **]**

5.2.4.1 Platform SWCs related diagnostic

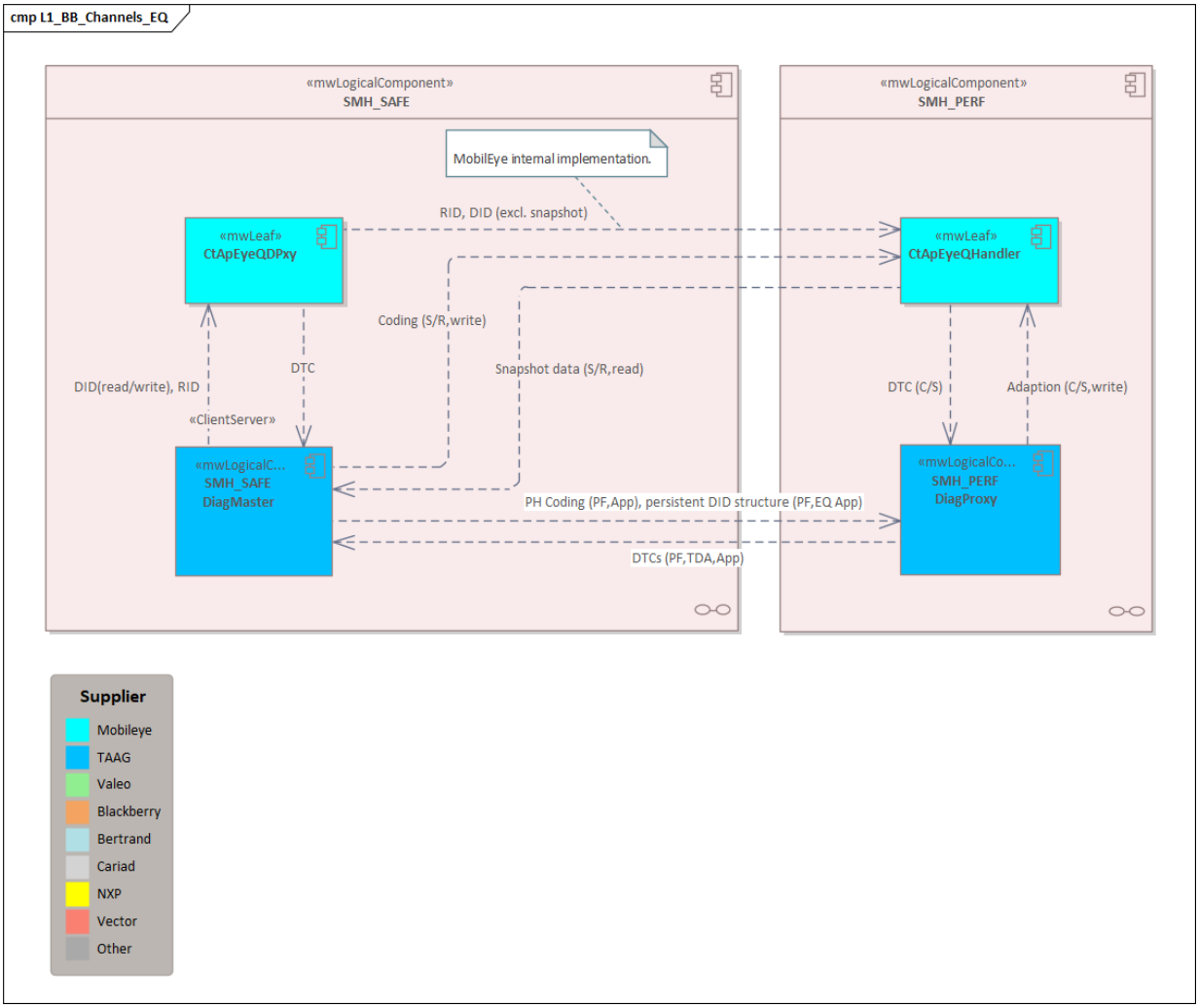


The communication channels used depend on the type of diagnostic information processed and the location of the involved component:

- *Non-persisted adaption* values are read and written via S/R ports directly from the respective component by DiagMaster component, Routines are triggered in the same way.
- The *coding word* is persistently stored by the DiagMaster and distributed as whole via S/R interface to the PF components on safety partition, and sent to DiagProxy who in turn persists the data locally and distributes it to the PF components on the performance partition.
  Persisted adaption values are stored in NVM together in a single structure and distributed identically to the coding value.
- DIDs that serve as snapshot data will be provided by the SWC on both partitions using a dedicated S/R port which is connected to DiagMaster component. DiagMaster will use this information in case a DTC is raised and the DIDs are requested by DEM.
- PF SWCs reporting a DTC event use a C/S port to report to the respective local diagnostic component. The DiagProxy component will forward the report to the DiagMaster which in turn will call the respective DEM port.

[S32GPRODP-296751]

5.2.4.2 EyeQ related diagnostic

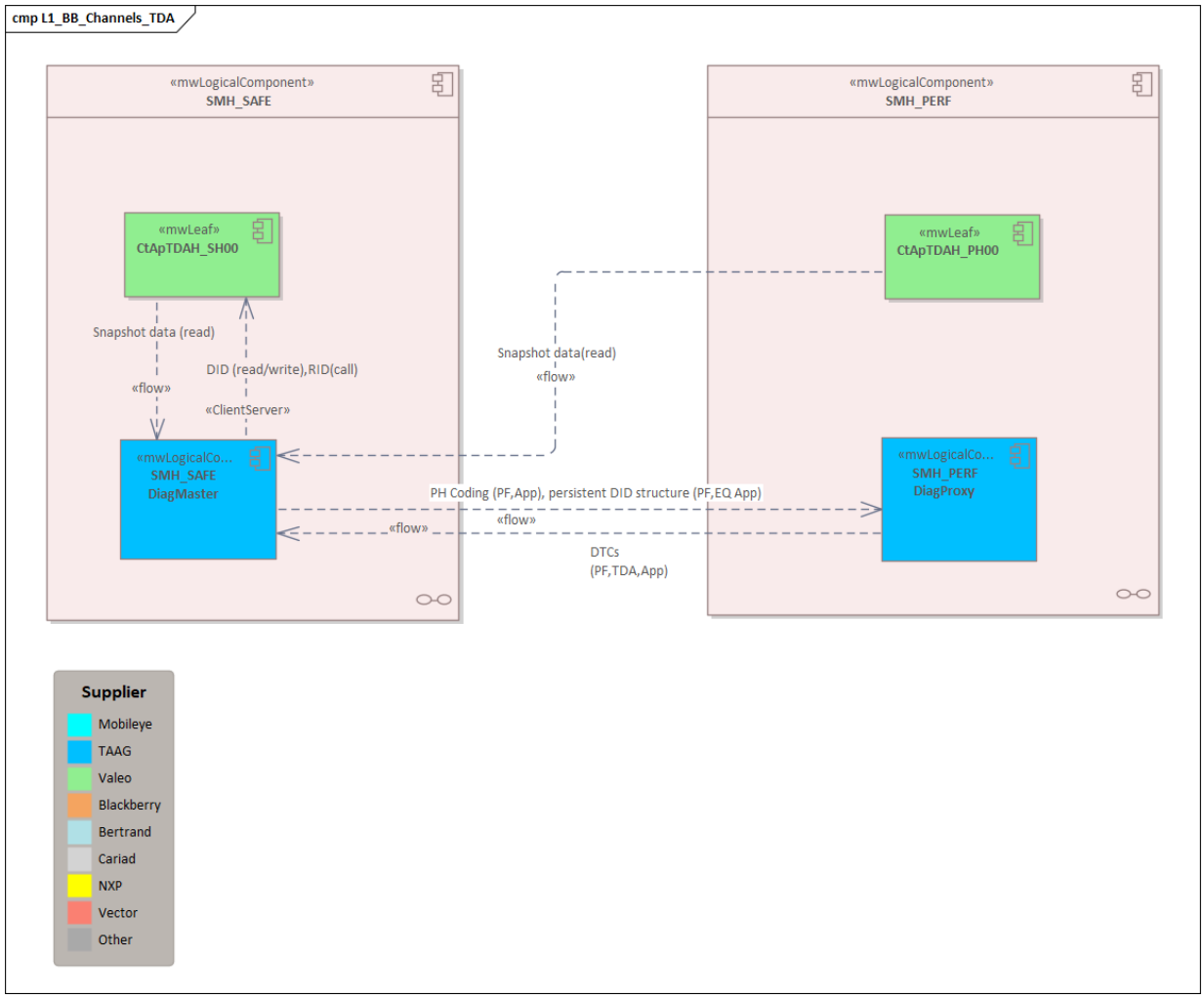


The communication channels used depend on the type of diagnostic information processed and the location of the involved component:

- *Non-persisted adaption* values are read and written via a C/S port by DiagMaster component to EyeQDProxy component (which internally forwards it to the EyeQHandler if appropriate, yet these mechanisms are outside of the scope of this architecture). Routines are triggered in the same way.
- The *coding word* is persistently stored by the DiagMaster and distributed as whole via S/R interface to the EyeQHandler on the performance partition.
- *Persisted adaption* values are stored in NVM together in a single structure and distributed by sending to the DiagProxy component which in turn stores the data locally to NVM and forwards to the EyeQHandler component.
- DIDs that serve as snapshot data will be provided by the EyeQHandler on the performance partition using a dedicated S/R port which is connected to DiagMaster component. DiagMaster will use this information in case a DTC is raised and the DIDs are requested by DEM.
- Both EyeQ Handler components reporting a DTC event use a C/S port to the respective local diagnostic component. The DiagProxy component will forward the report to the DiagMaster which in turn will call the respective DEM port.

[S32GPRODP-296754]

5.2.4.3 TDA related diagnostic

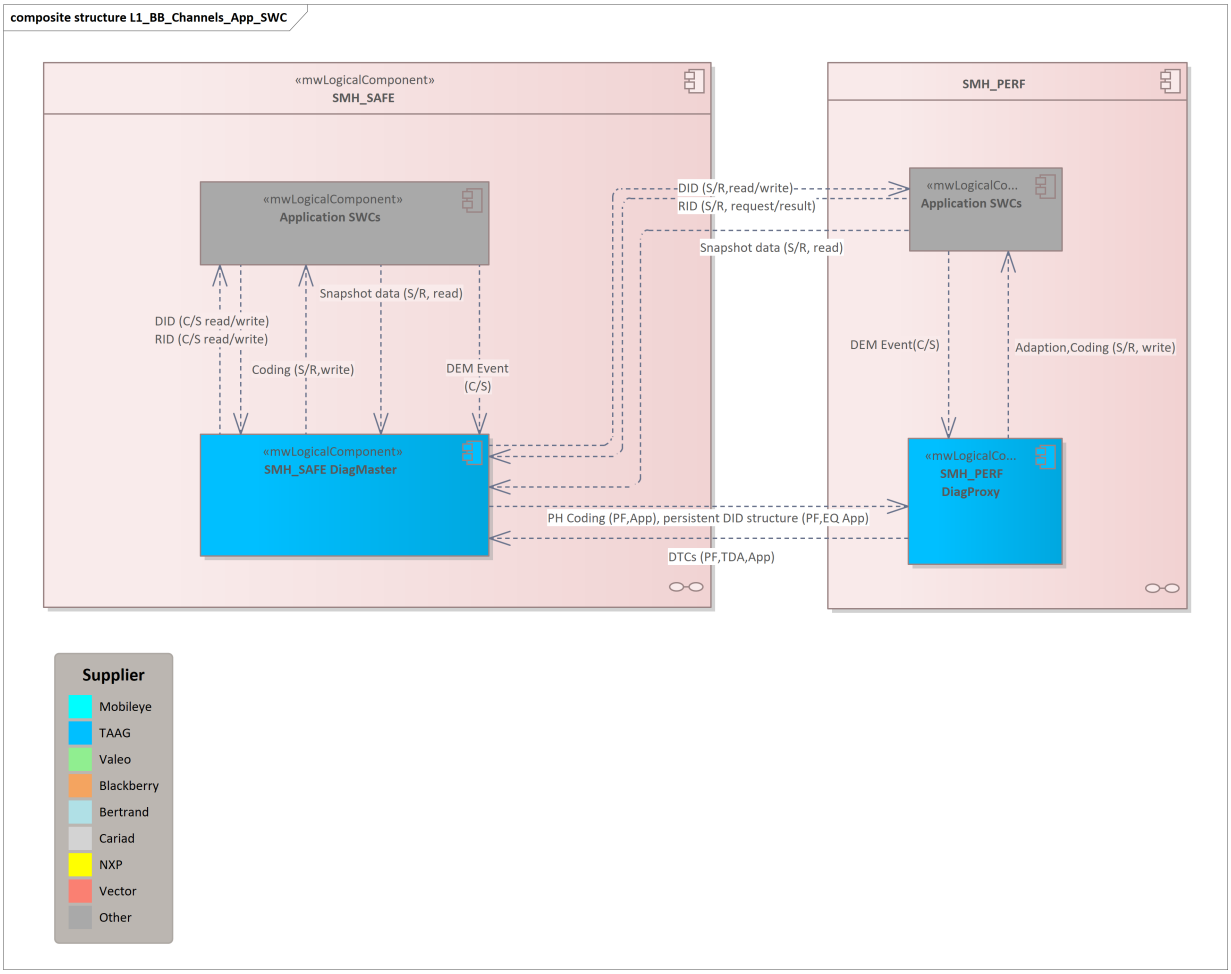


The communication channels used depend on the type of diagnostic information processed and the location of the involved component:

- *Non-persisted adaption* values are read and written via a C/S port by DiagMaster component to TDA Handler component. Routines are triggered in the same way.
- The *coding word* is persistently stored by the DiagMaster and distributed as whole via S/R interface to the TDA Handler on the safety partition. *Persisted adaption* values are stored in NVM together in a single structure and handled identically to coding procedure.
- DIDs that serve as snapshot data will be provided by theTDA components on both partitions using a dedicated S/R port which is connected to DiagMaster component. DiagMaster will use this information in case a DTC is raised and the DIDs are requested by DEM.
- The TDA Handler components reporting a DTC event use a C/S port to the respective local diagnostic component. The DiagProxy component will forward the report to the DiagMaster which in turn will call the respective DEM port

[S32GPRODP-296755]

5.2.4.4 App SWCs related diagnostic



This diagram depicts the general architecture pattern for optional application SWCs, that are added to the diagnostic infrastructure as project specific configuration. This pattern is based on the assumption that there is no exchange of diagnostic information done between SH00 and PH00 components (ie., no master proxy architecture of these SWCs).
The communication channels used depend on the type of diagnostic information processed and the location of the involved component:

- *Non-persisted adaption and measurement* values are read and written
  - via C/S ports on the safety partition
  - via S/R ports on the performance partition
  directly from the respective component by DiagMaster component,
  Routines are triggered in the same way and the results delivered as return value (S/R interfaces need to support bidirectional data transfer).
- The *coding word* is persistently stored by the DiagMaster and distributed as whole via S/R interface to the application components on safety partition, and sent to DiagProxy which in turn persists the data locally and distributes it to the application components on the performance partition.
  Persisted adaption values are stored in NVM together in a single structure and distributed identically to the coding value.
- DIDs that serve as snapshot data will be provided by the SWC on both partitions using a dedicated S/R port

which is connected to DiagMaster component. DiagMaster will use this information in case a DTC is raised and the DIDs are requested by DEM.

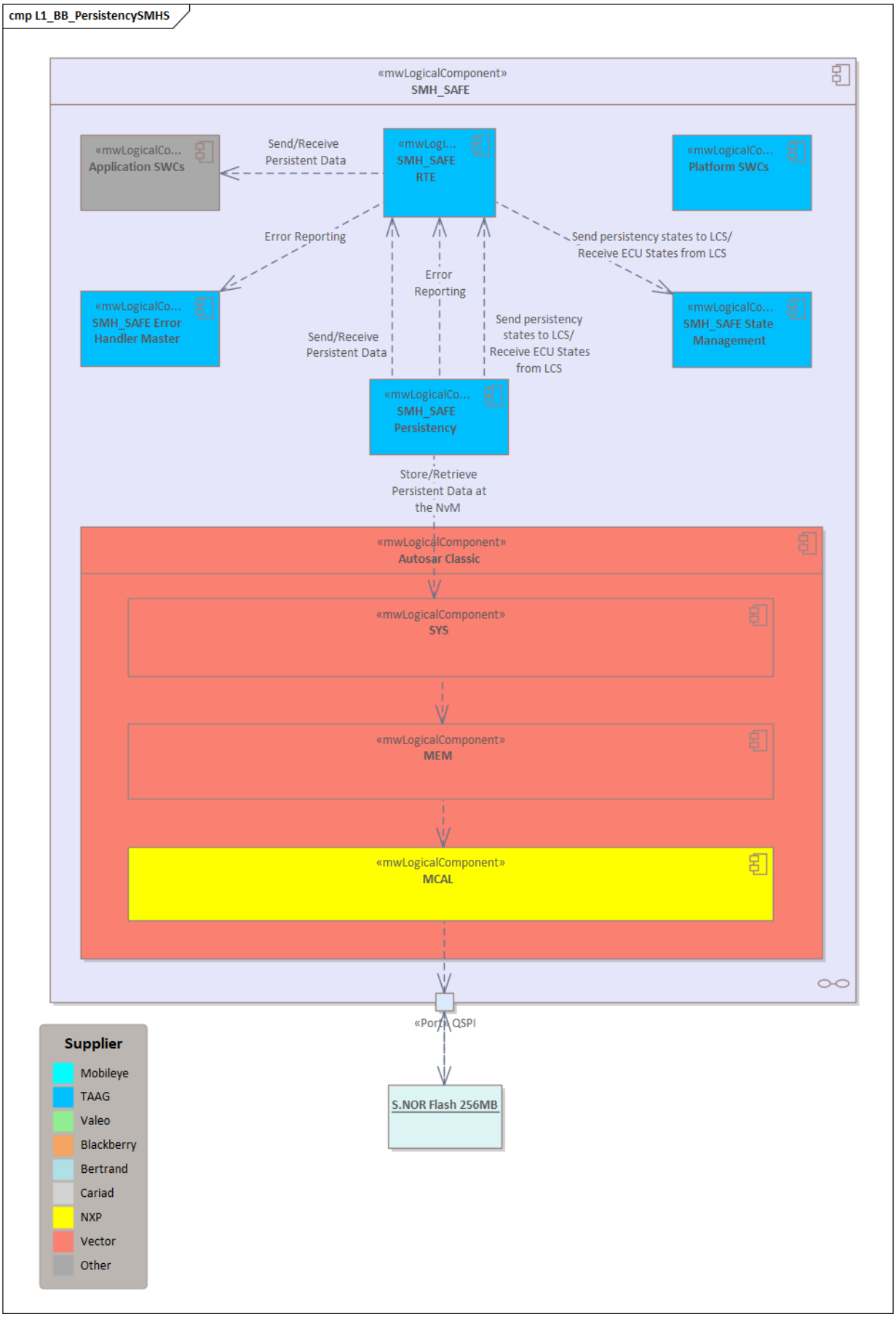
- Application SWCs reporting a DTC event use a C/S port to report to the respective local diagnostic component. The DiagProxy component will forward the report to the DiagMaster which in turn will call the respective DEM (service) port.

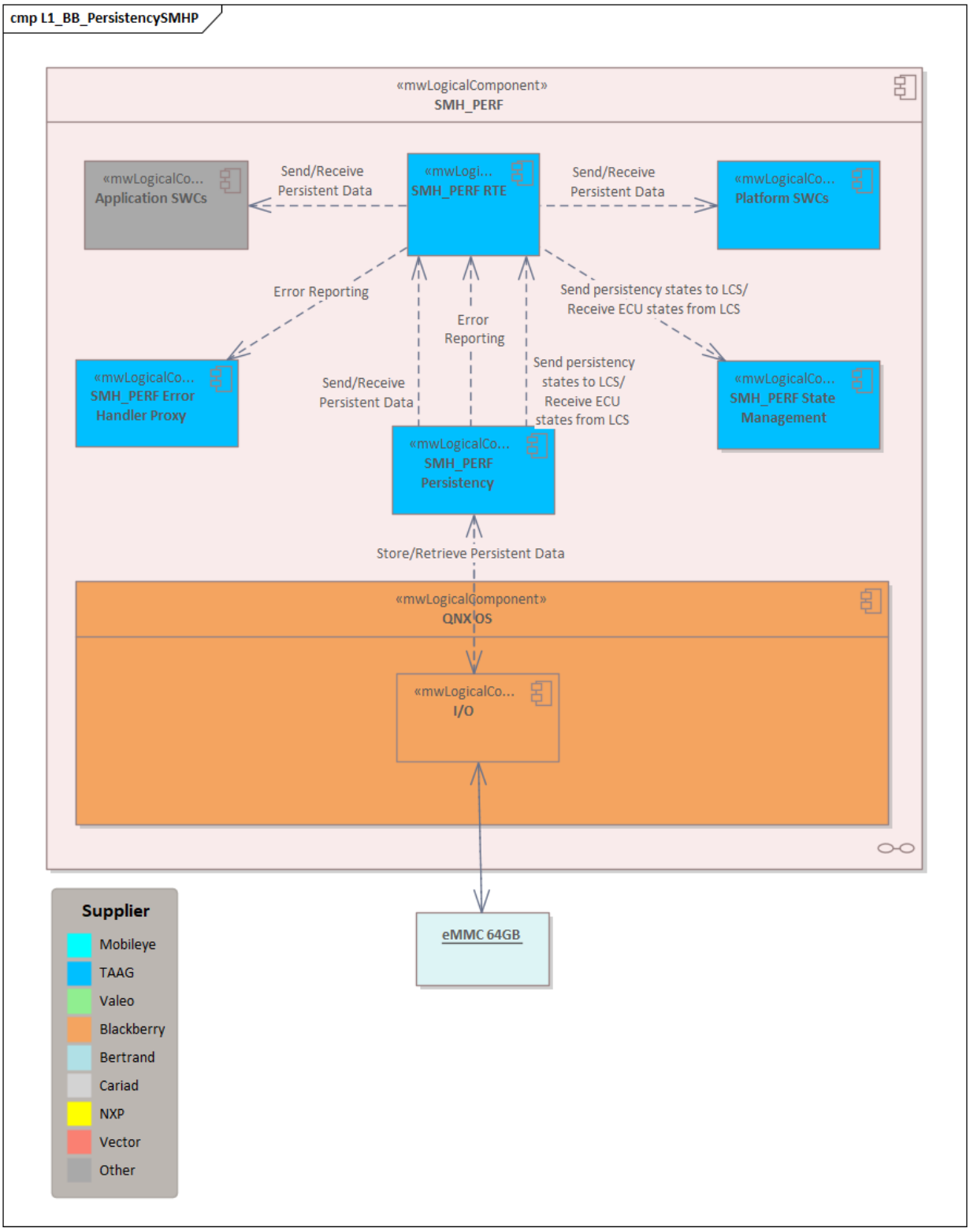
[S32GPRODP-310428]

5.2.5 Persistent storage

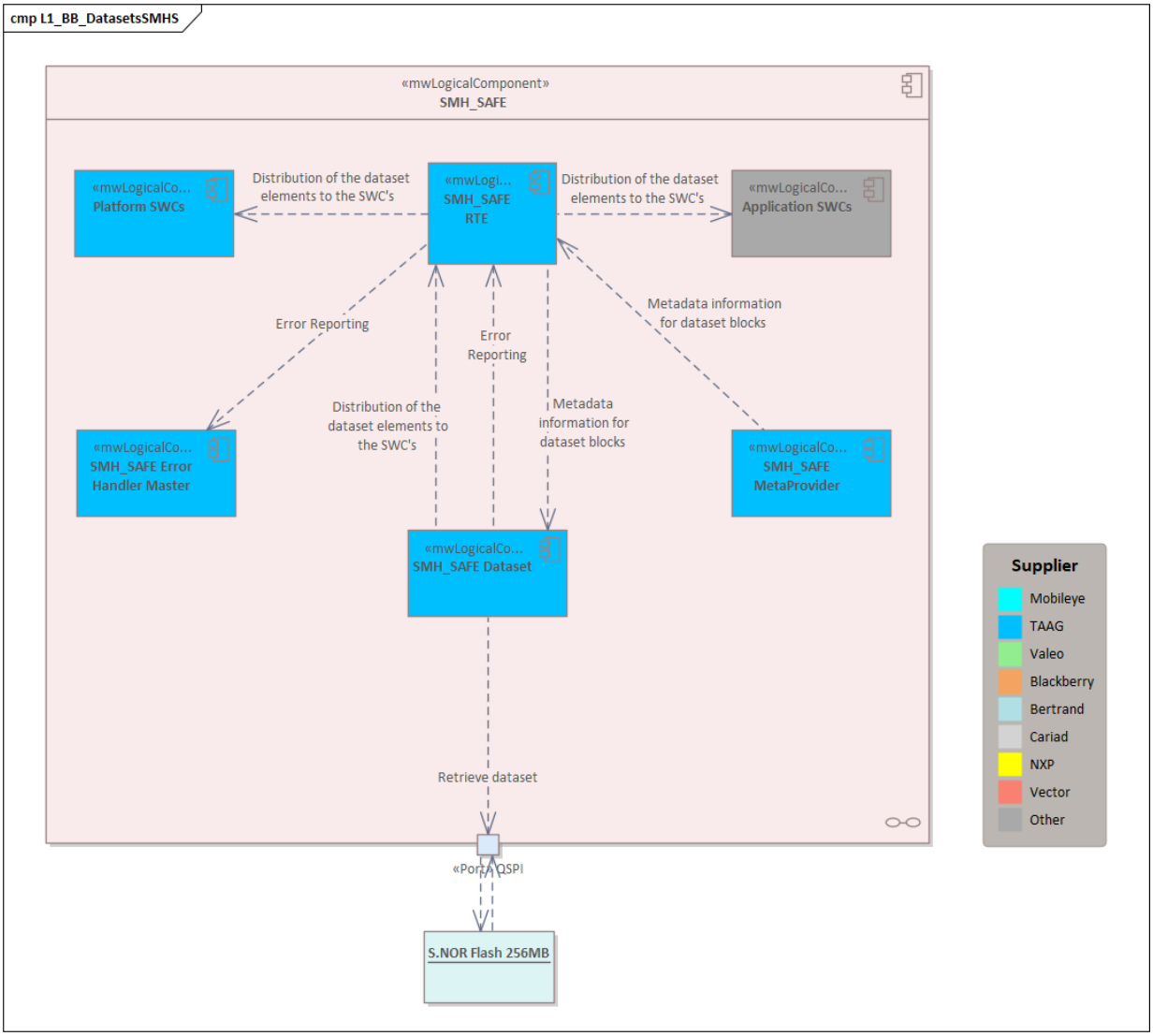
Persistency Service manages the persistent safe, non-volatile preservation of data across the ECU life cycle, including shutdown and restart. It reads/writes data from/to NVM, and provides it to applications via the MotionWise RTE. Persistency Service handles both persistent data and dataset. [S32GPRODP-296322]

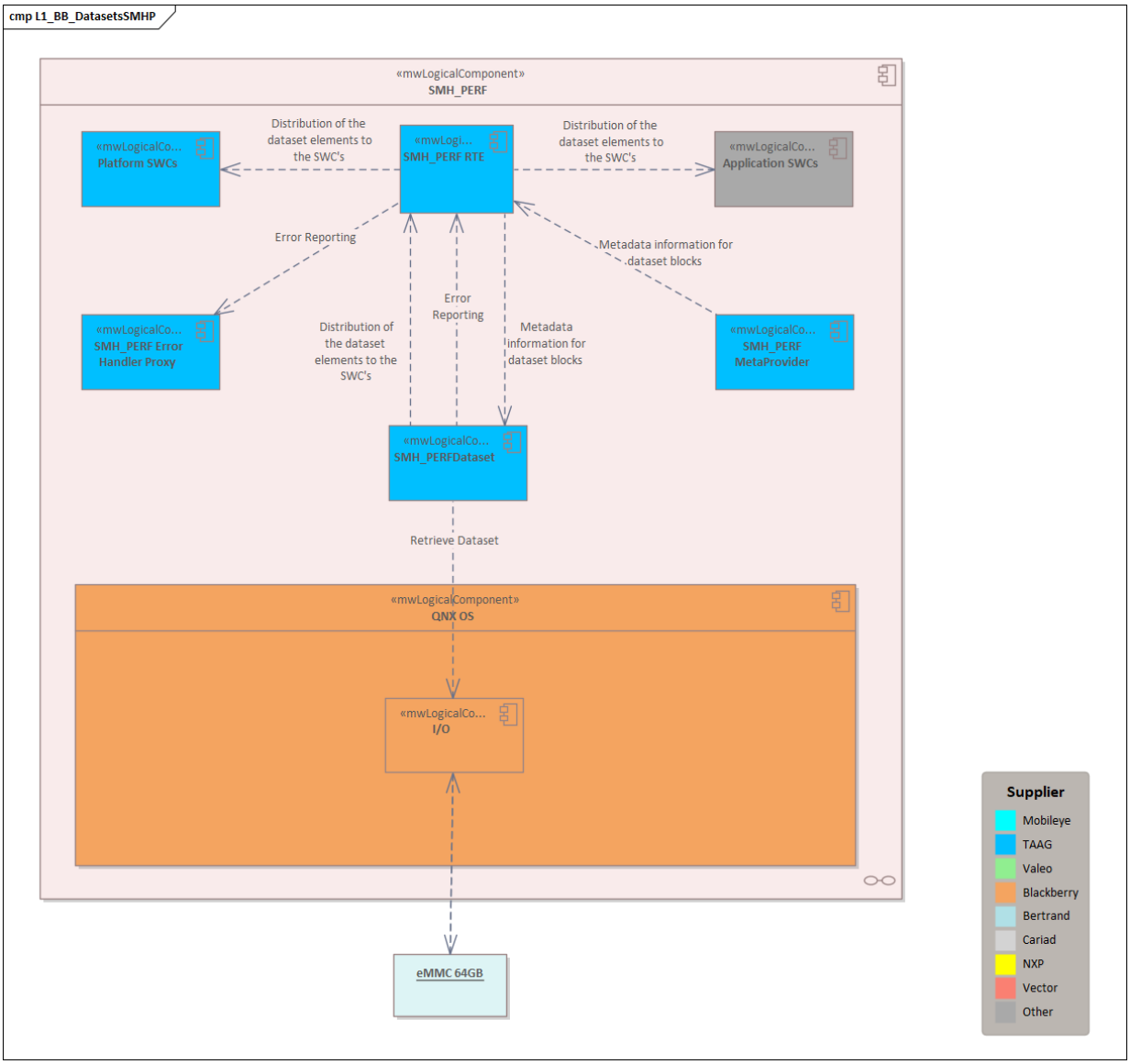
5.2.5.1 Persistency


cmp L1_BB_PersistencySMHS
«mwLogicalComponent»
SMH_SAFE
«mwLogicalCo...
Application SWCs
Send/Receive
Persistent Data
«mwLogi...
SMH_SAFE
RTE
«mwLogicalCo...
Platform SWCs
Error Reporting
Send persistency states to LCS/
Receive ECU States from LCS
Error
Reporting
«mwLogicalCo...
SMH_SAFE Error
Handler Master
Send/Receive
Persistent Data
Send persistency
states to LCS/
Receive ECU States
from LCS
«mwLogicalCo...
SMH_SAFE State
Management
«mwLogicalCo...
SMH_SAFE
Persistency
Store/Retrieve
Persistent Data at
the NvM
«mwLogicalComponent»
Autosar Classic
«mwLogicalComponent»
SYS
«mwLogicalComponent»
MEM
«mwLogicalComponent»
MCAL
«Port» QSPI
S.NOR Flash 256MB
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other


cmp L1_BB_PersistencySMHP
«mwLogicalComponent»
SMH_PERF
«mwLogicalCo...
Application SWCs
Send/Receive
Persistent Data
«mwLogi...
SMH_PERF RTE
Send/Receive
Persistent Data
«mwLogicalCo...
Platform SWCs
Error Reporting
Send persistency states to LCS/
Receive ECU states from LCS
Error
Reporting
Send persistency
states to LCS/
Receive ECU
states from LCS
Send/Receive
Persistent Data
«mwLogicalCo...
SMH_PERF Error
Handler Proxy
«mwLogicalCo...
SMH_PERF State
Management
«mwLogicalCo...
SMH_PERF
Persistency
Store/Retrieve Persistent Data
«mwLogicalComponent»
QNX OS
«mwLogicalCo...
I/O
eMMC 64GB
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

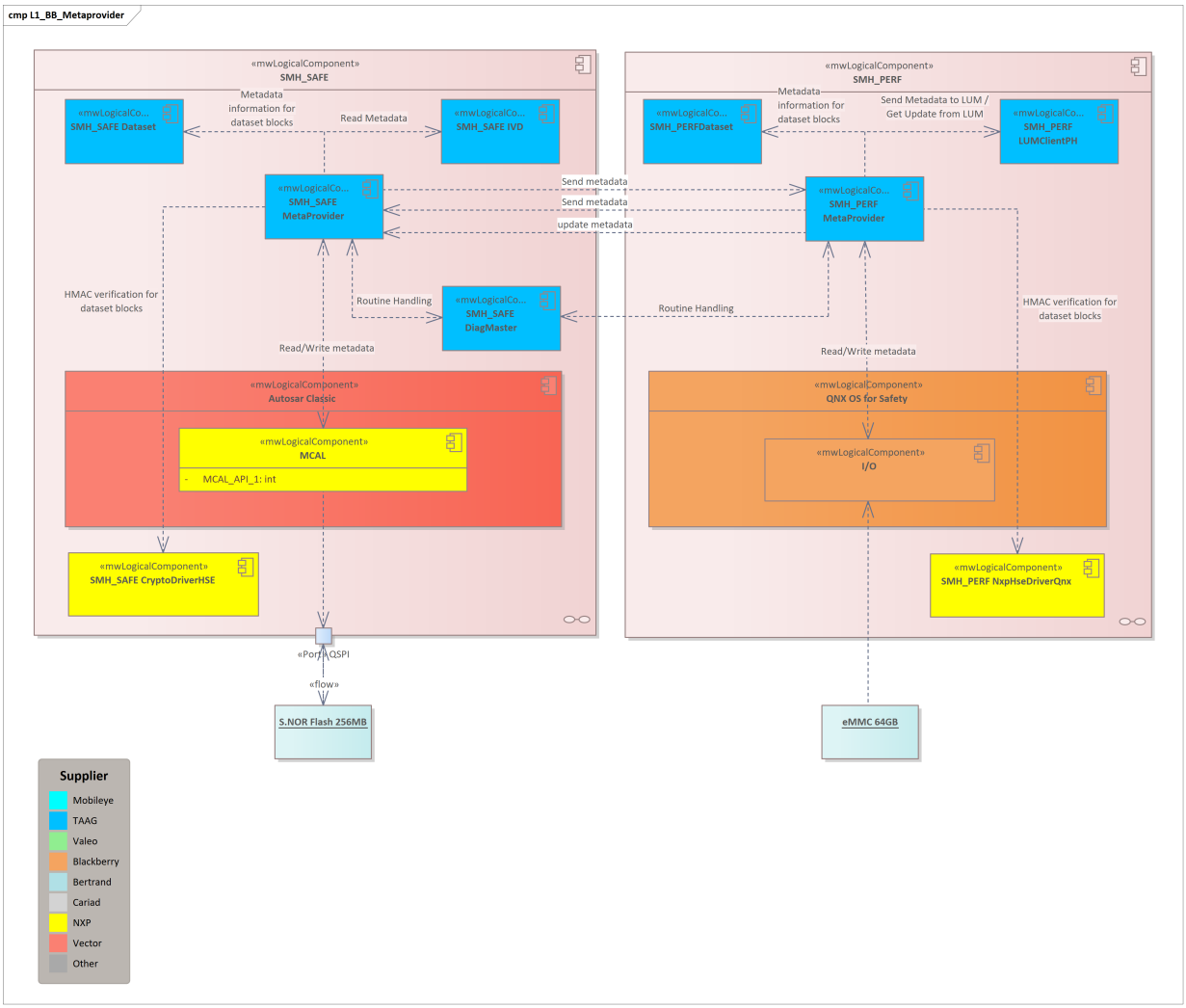
5.2.5.2 Datasets





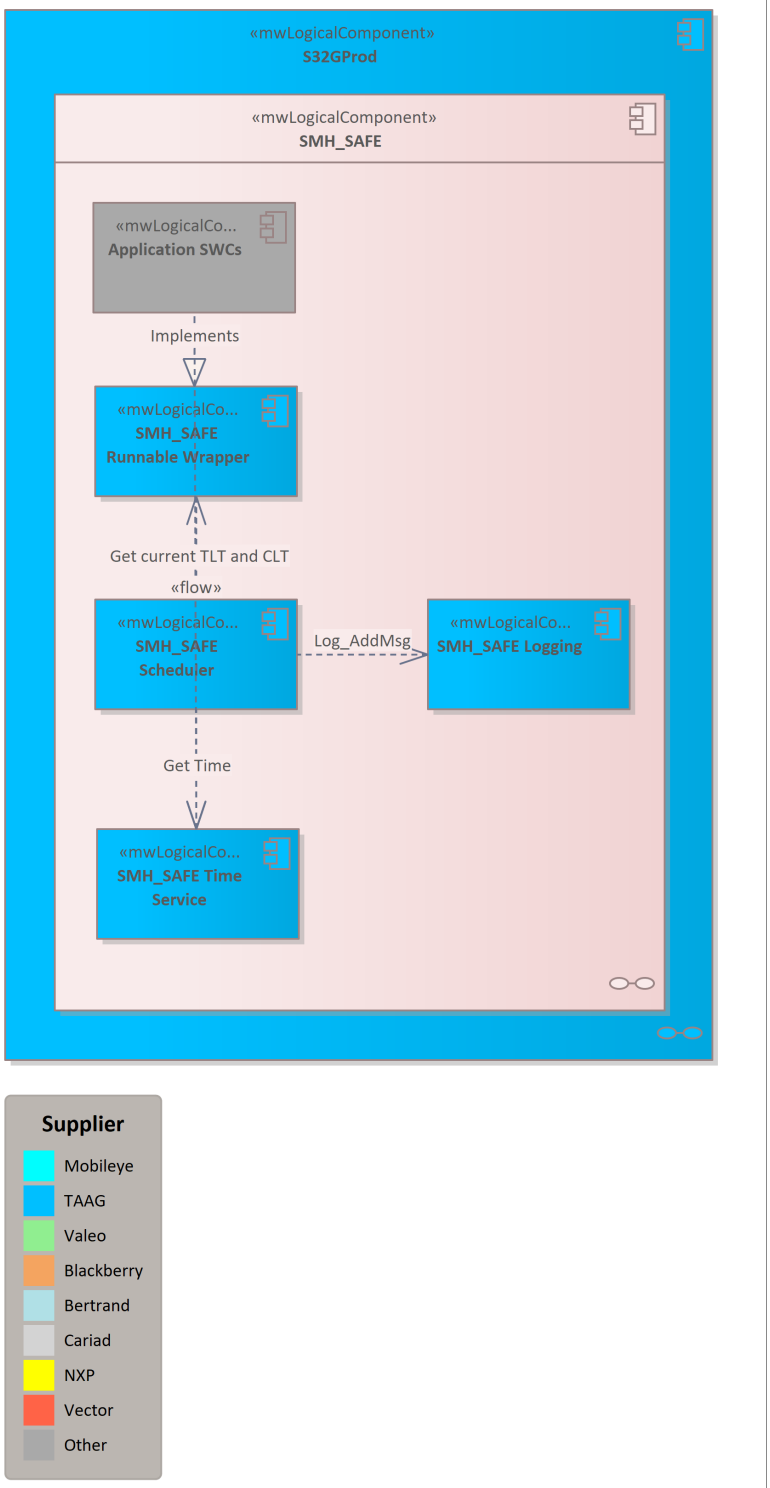
5.2.5.3 Metaprovider

The Metaprovider SWC is responsible for the provision of information (Name, Version, CRC etc.) for the logical blocks and hardware/sofware to the application SWC's. This information, called metadata, is stored in the non-volatile memory. **[**S32GPRODP-296323 **]**



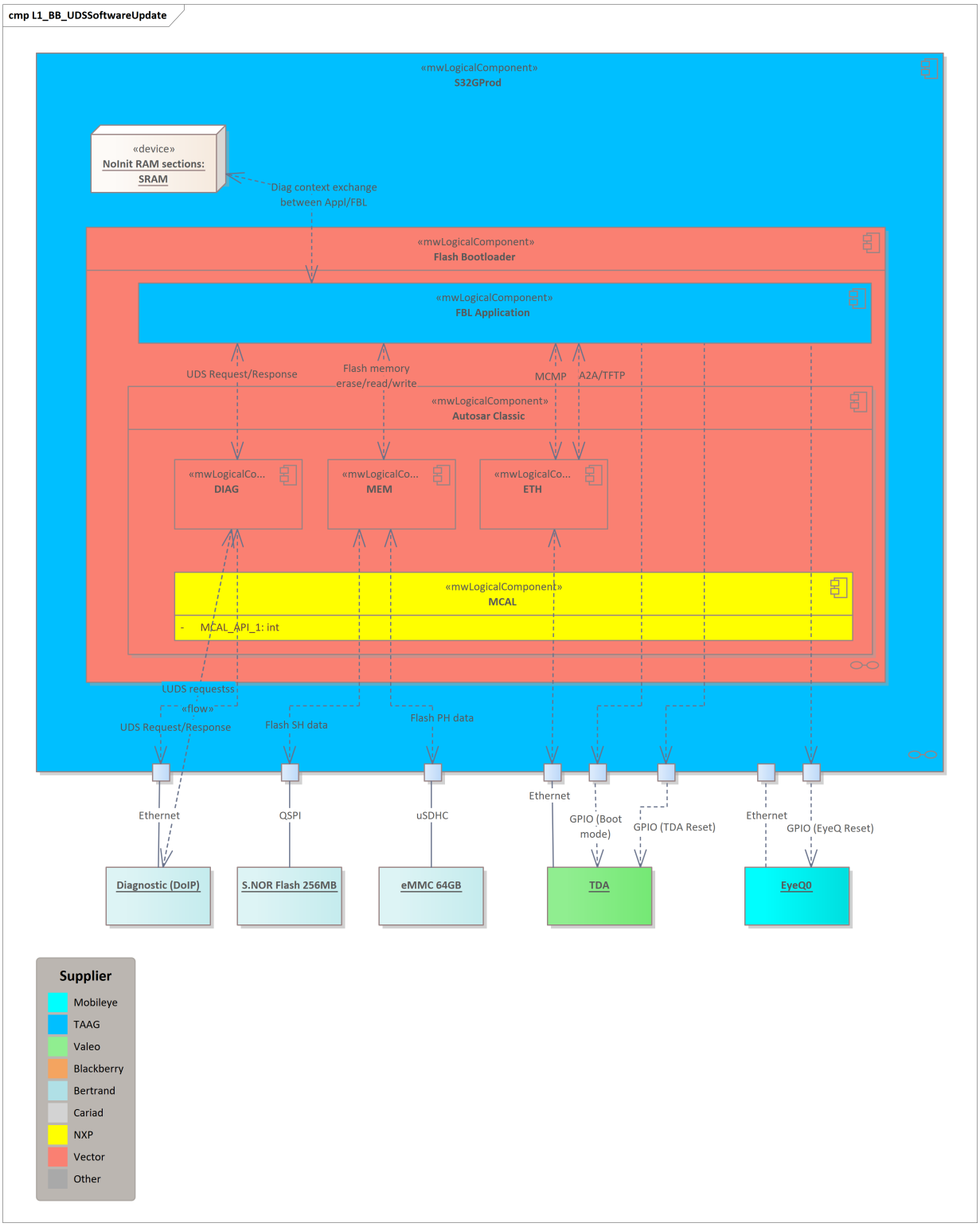
5.2.6 Task scheduling

cmp L1_BB_Task Scheduler SH



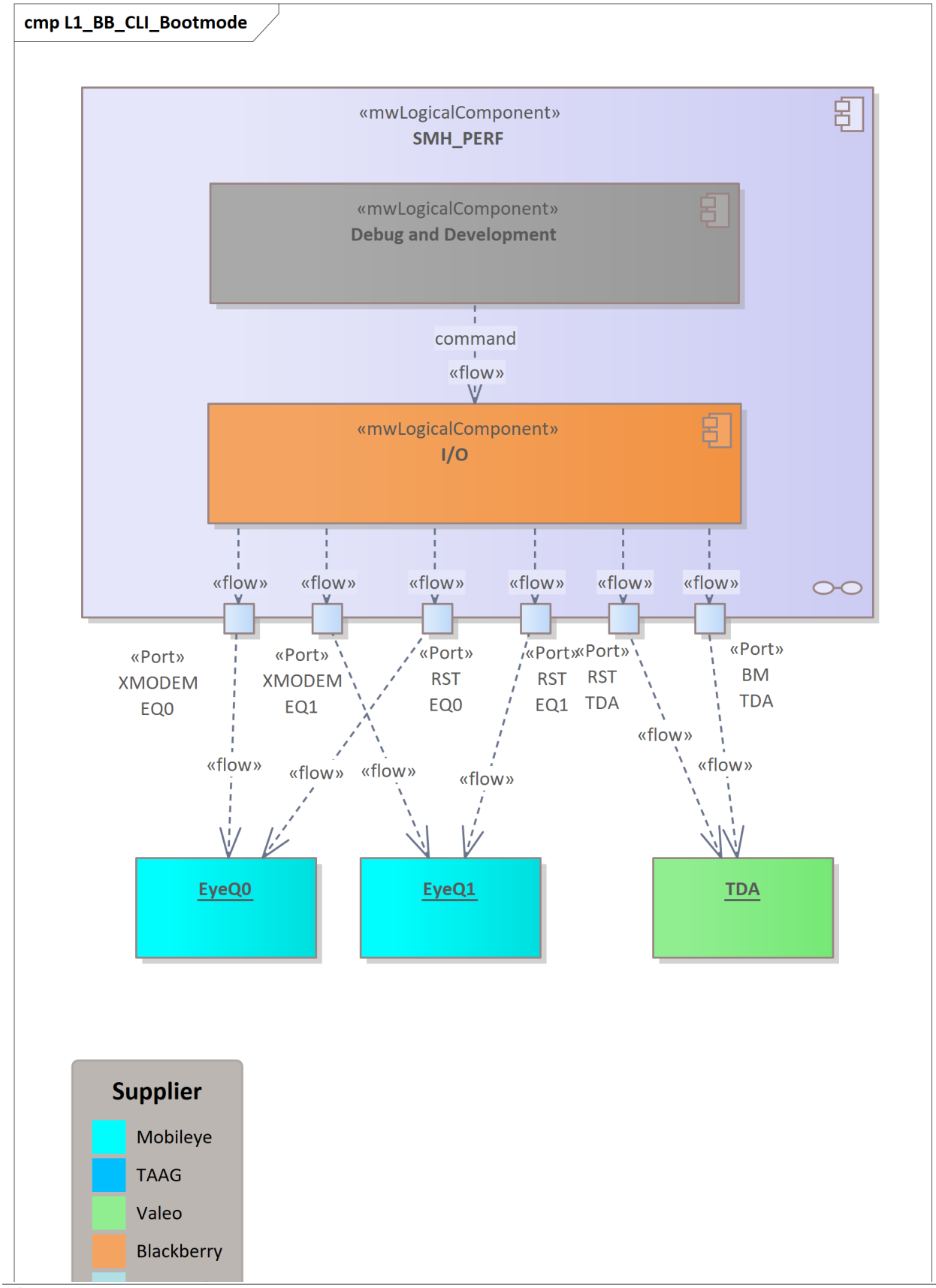
5.2.7 Software update

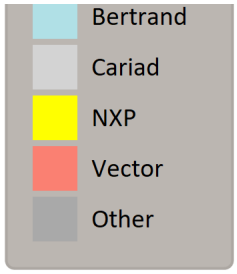
5.2.7.1 UDS Software Update



5.2.8 Analysis framework

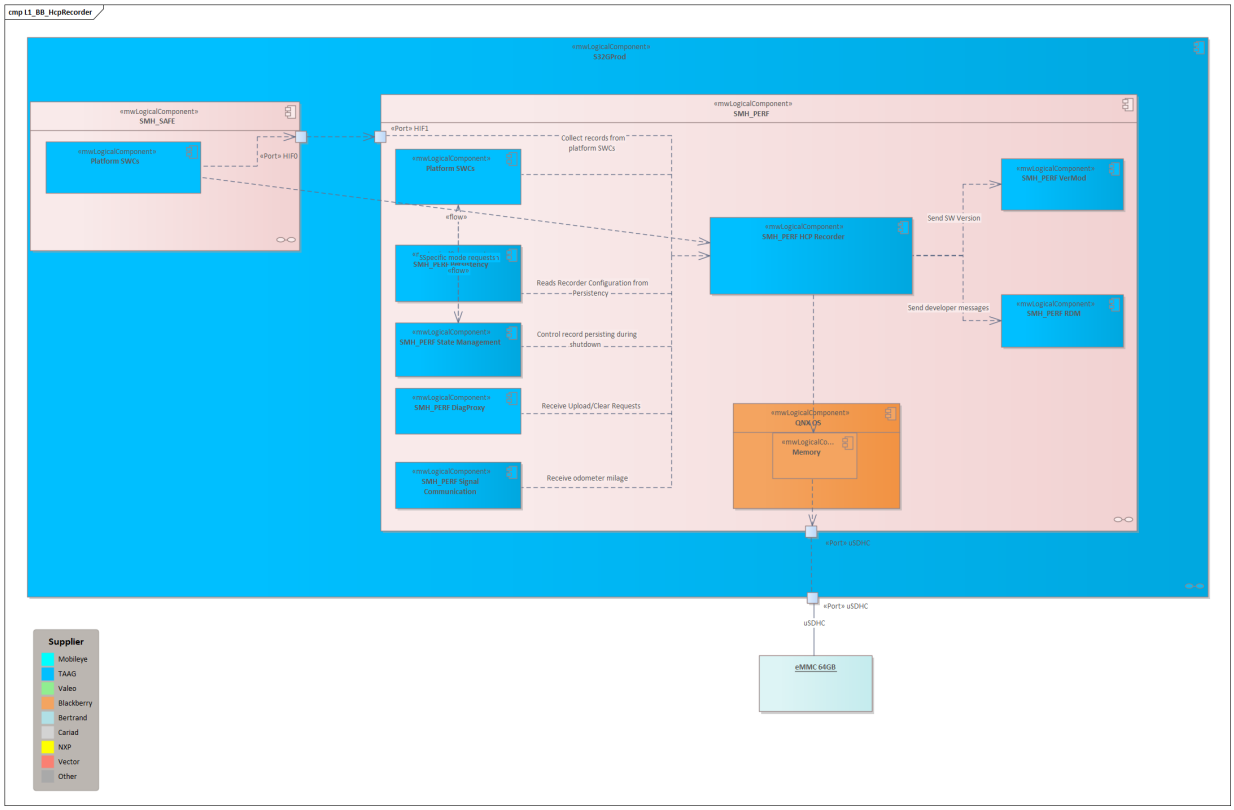
5.2.8.1 Bootmode CLI





The Bootmode CLI tool is a debug and development tool, which is implemented on the SMH_PERF partition and accessible through the QNX Korn Shell. The tool uses I/O drivers of the QNX OS to control reset and bootmode hardware pins of the connected EyeQ and TDA SoC's. **[**S32GPRODP-296163 **]**

5.2.8.2 HCP recorder



HCP Recorder is a series production software feature used for supporting the analysis of problems and defects that may occur when a vehicle is in the field. Its sole purpose is to gather events, data, operating conditions, etc. from other platform software functions during the usage of the vehicle and store them persistently in the ECU. Gathering of information does not involve intrusive tooling and stored information are to be downloaded, at service station, via diagnostic services for further analysis.
The following goals shall be achieved:

- collecting records from other platform software components and storing them in an internal buffer during the current power cycle
- persist the content of the buffer during shutdown/reset of the ECU
- provide diagnostic interface to access the persisted data
- prevent certain records, which are marked as irreplaceable, from being overwritten

- logging of the current mileage from odometry every 30 seconds

[S32GPRODP-296747]

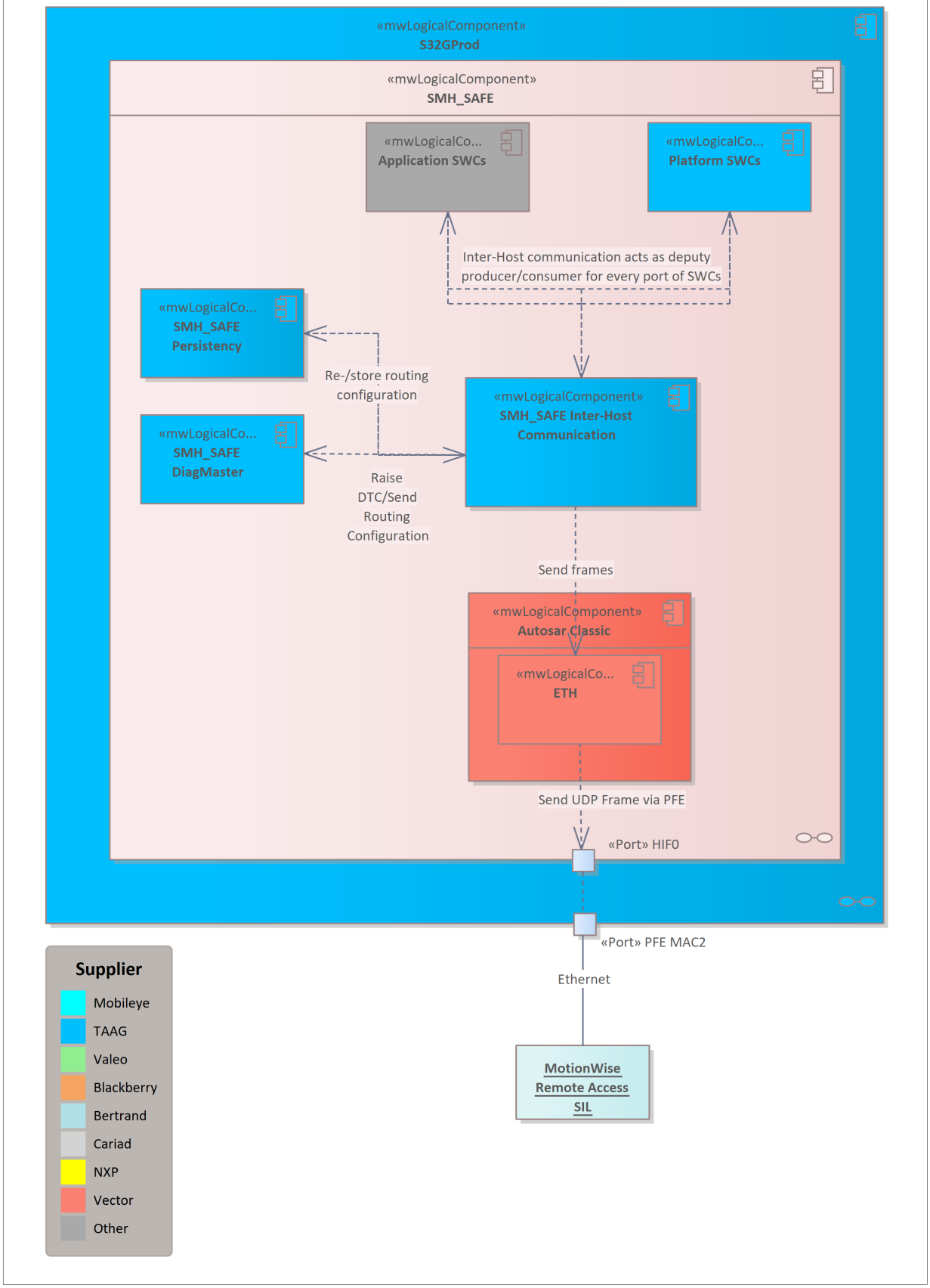
5.2.8.3 Remote access

Remote Access is a feature that enables a debug PC to monitor and re-configure network (RTE) traffic on hosts that execute the MotionWise Classic Platform. Remote Access consists of three parts:

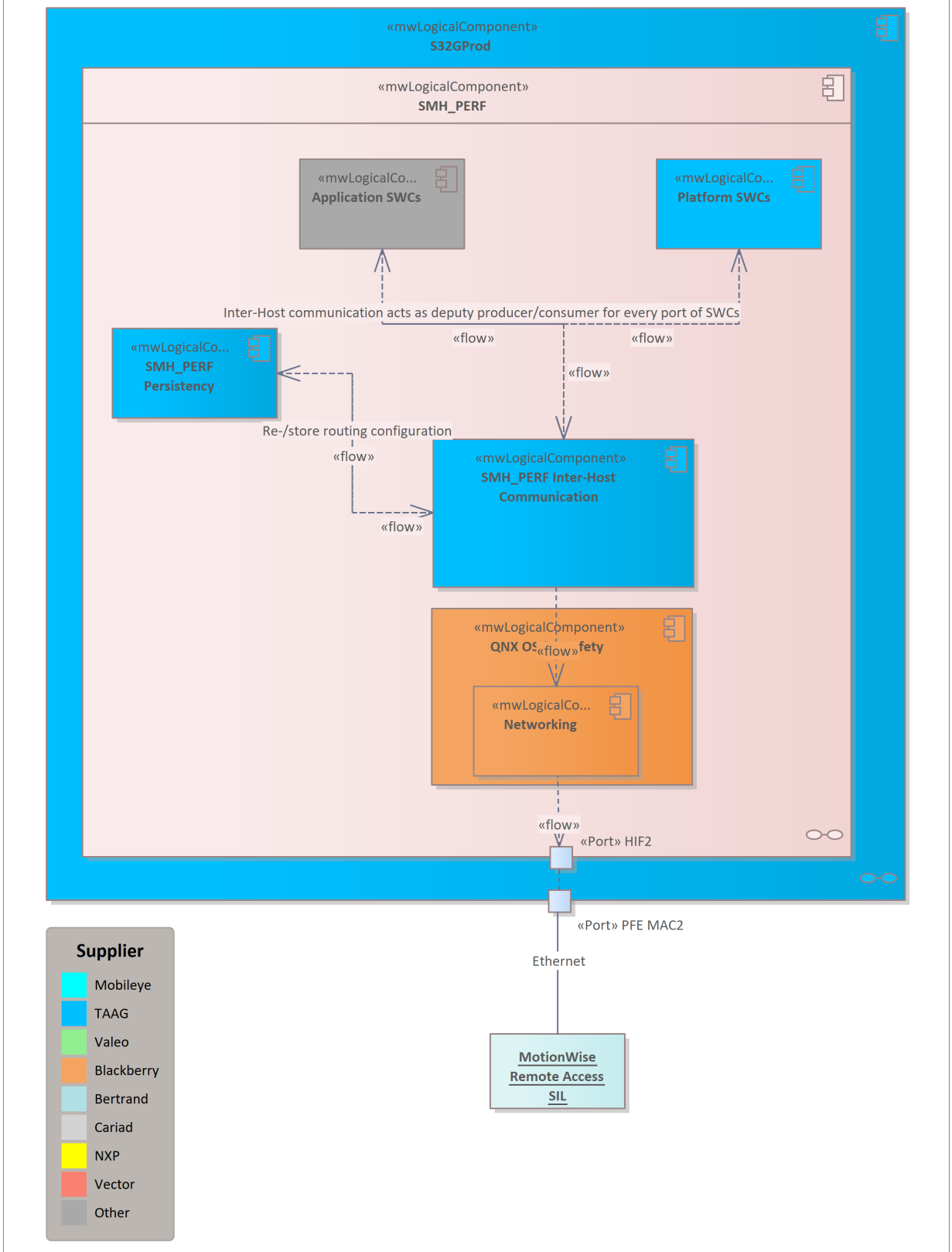
- Inter-Host Communication: A communication middleware where all the traffic between SWCs are going through and redirects traffic for inter-host (among SMH_S, SMH_P and Debug Host).
- The Remote Access Library (RA-Lib): A collection of C functions which can monitor RTE communication and reconfigure some platform services. The RA-Lib is a C library and contains no dependencies to a specific test environment. The library can be embedded in a variety of simulation environments.
- The Remote Access Tools (RA Tools): A collection of Python scripts which use the Remote Access Library. These scripts can be used to perform standard tasks like logging, configuring the routing table, etc.

Inter-Host Communication mechanism implemented on different partition does not have any differences apart from the underlying network stack. [S32GPRODP-296836]

cmp L1_BB_RemoteAccessSMHS



cmp L1_BB_RemoteAccessSMHP

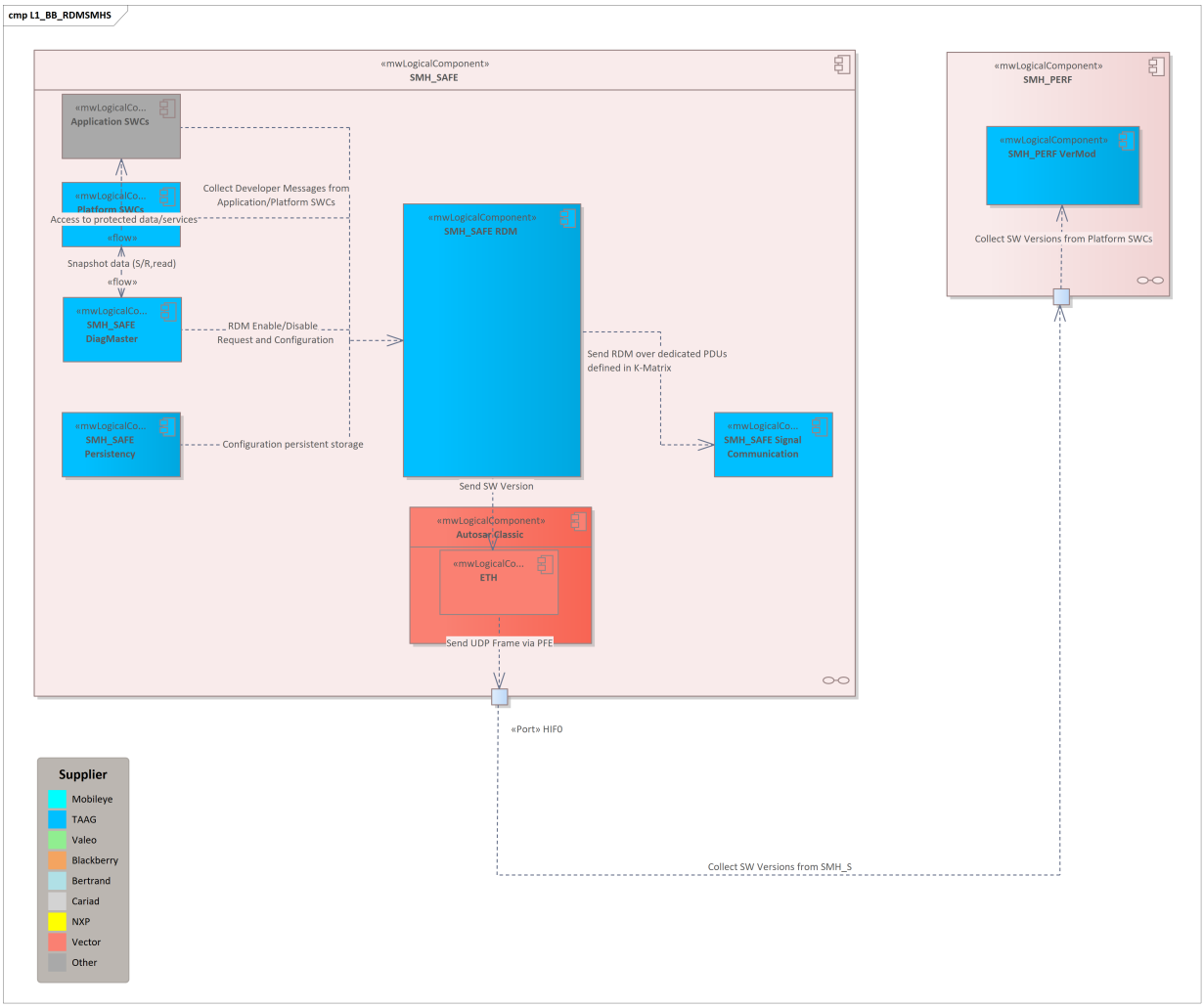


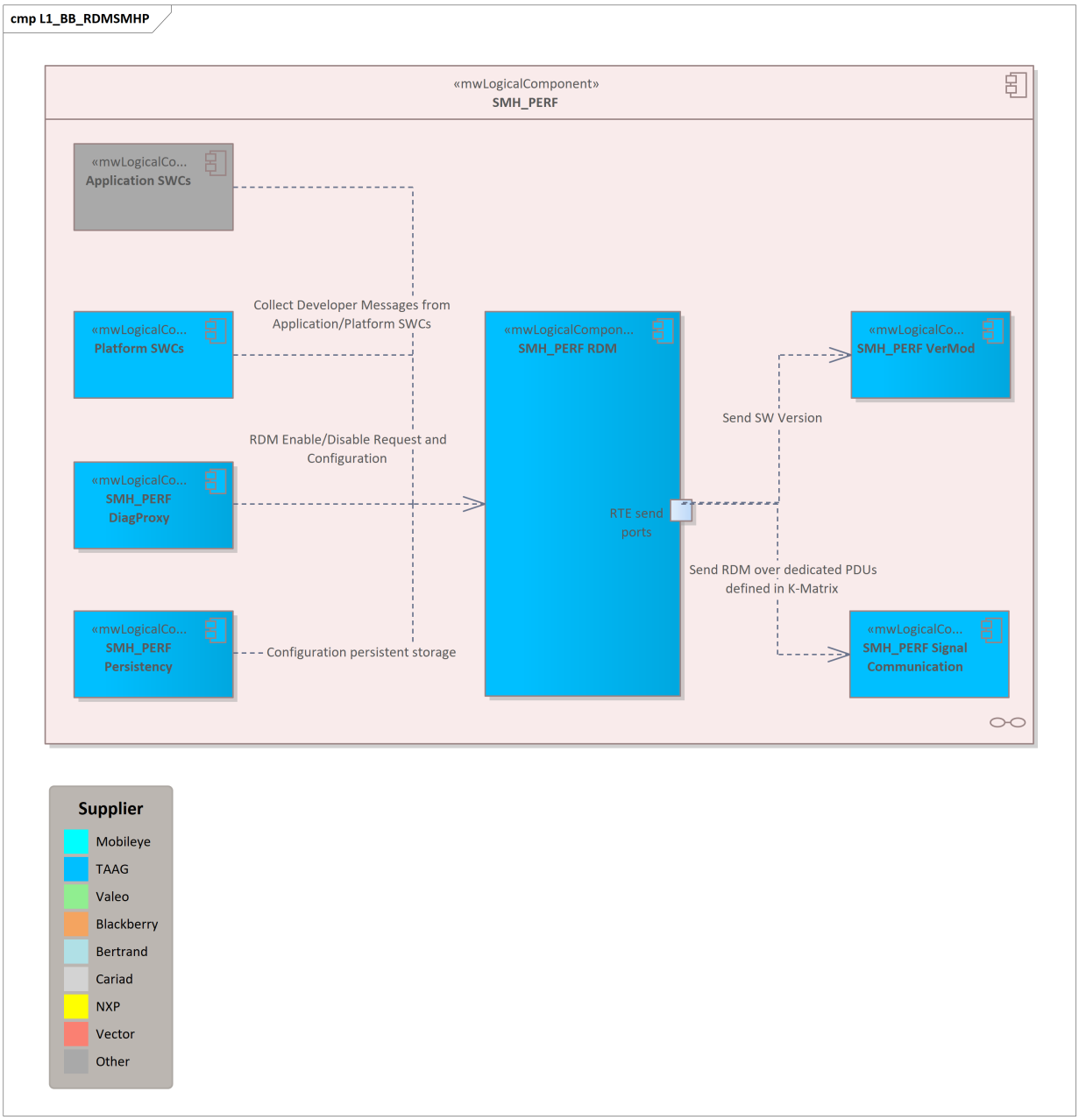
5.2.8.4 Routing of Developer Messages (RDM)

For development purposes and analysis each application software component and platform software component shall have the possibility to send developer messages via Ethernet. The messages are defined with a fixed size, but the content and layout can be individually chosen by each component. Only one application component's messages are send at a time. The platform components shall deliver their messages to RDM and be aggregated in one message send on Ethernet. Configuration is done via diagnostics. Messages will be send periodically every 40ms.
The following goals shall be achieved:

- route and filter the developer messages from application and platform components on each partition
- enable configuration and enable/disable via diagnosis
- provide API on the RTE to be used by SWCs to send developer messages
- document message layout for platform component message and all SWC messag

[S32GPRODP-296728]



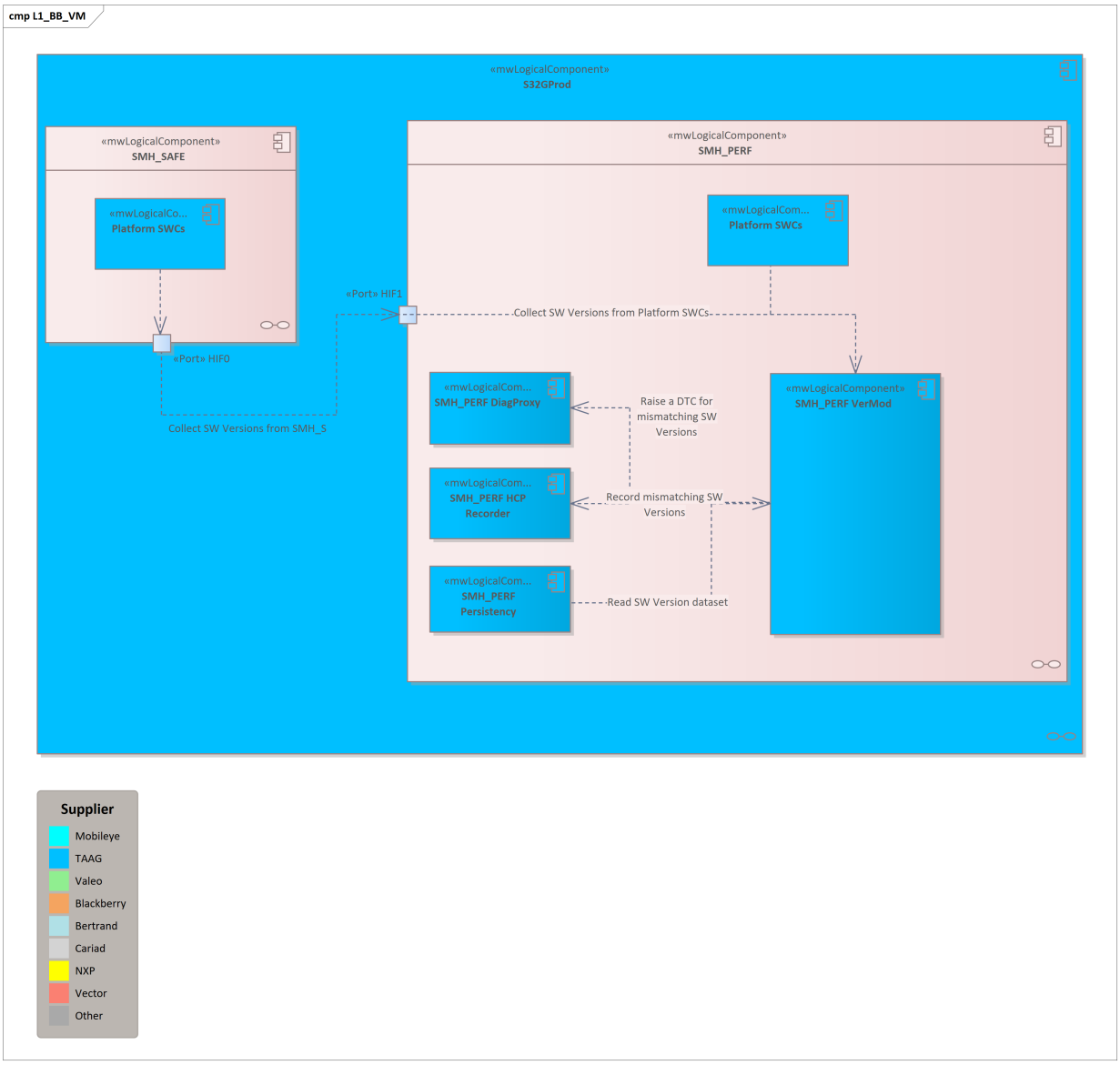


5.2.8.5 Version Module (VM)

In series and development software it is important that a device under test can be unambiguously identified. All variable SW modules/components must therefore be identifiable. The Version Module checks each component for its version number and build date and compares them to a list, which is created by the official build chain for a release. If deviations are found, the version module logs and records the components and their version number. The following goals shall be achieved:

- notify software version deviations
- logging of any software version deviations
- logging of the version numbers of each software component

[S32GPRODP-296734]



5.2.8.6 Logging framework

Any customer SW-C runnable can log messages during its execution. Messages are logged within the execution runtime of the relevant SW-C runnable, at up to ASIL-D quality according to the SW-C's safety level.
Message content includes

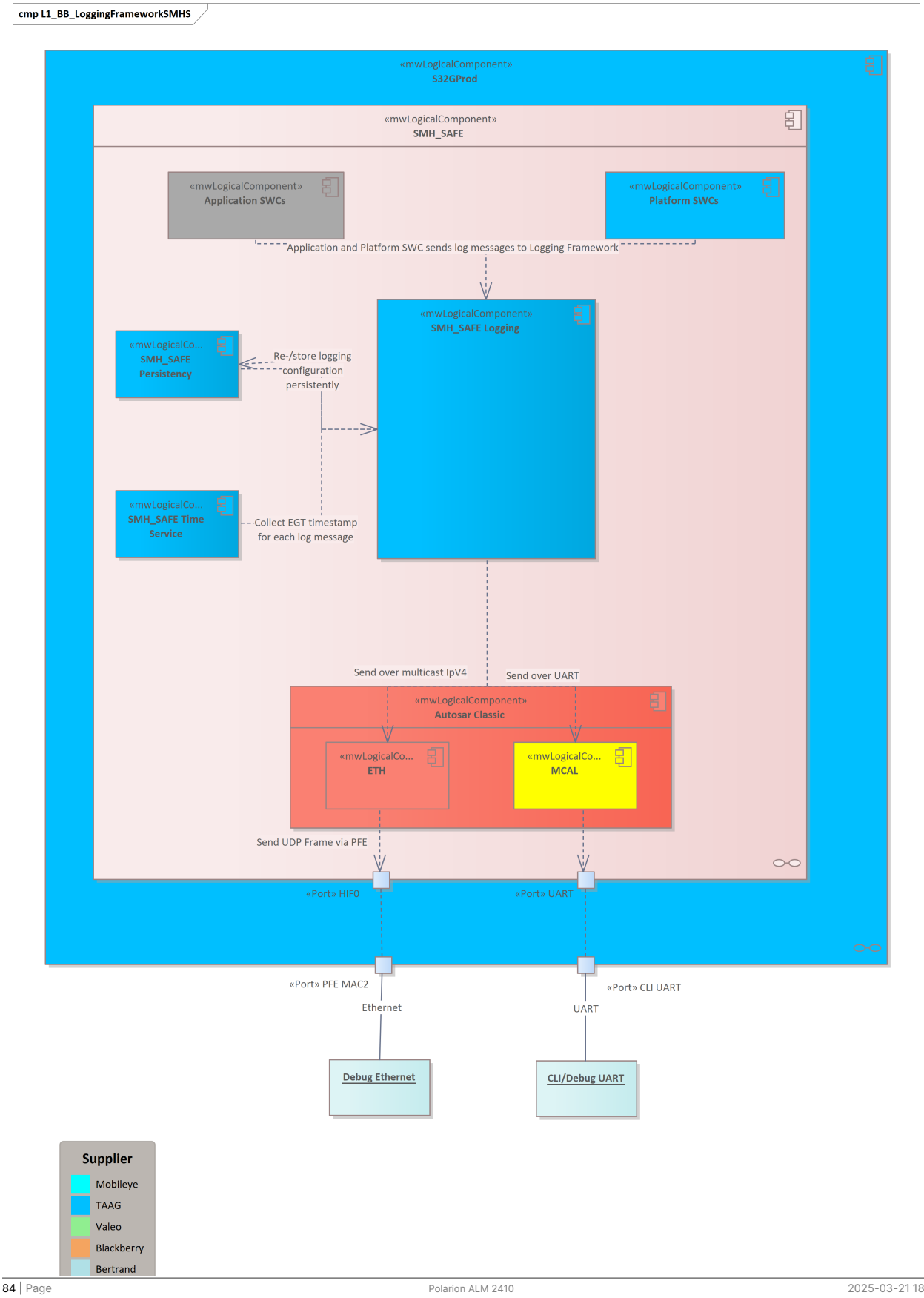
- integer values,
- strings,
- pointers to shared memory, or
- a binary stream

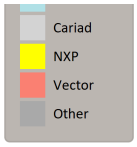
Saved messages are written to a host-internal memory buffer according to the desired output sink(s). The buffers are ring buffers: when a buffer is full, the oldest entry will be overwritten first. The size of each output sink's buffer is static, and changing its size requires recompiling the platform software. Each SW-C must allocate a subset of the available memory to which it logs messages.
Messages are dumped to a debug PC via one of three output sinks: UART, or Ethernet and addition to SMH_S partition Logging Framework on SMH_P also supports TLR via TLR Adapter SWC. (Note that UART is not available

for binary streams.)

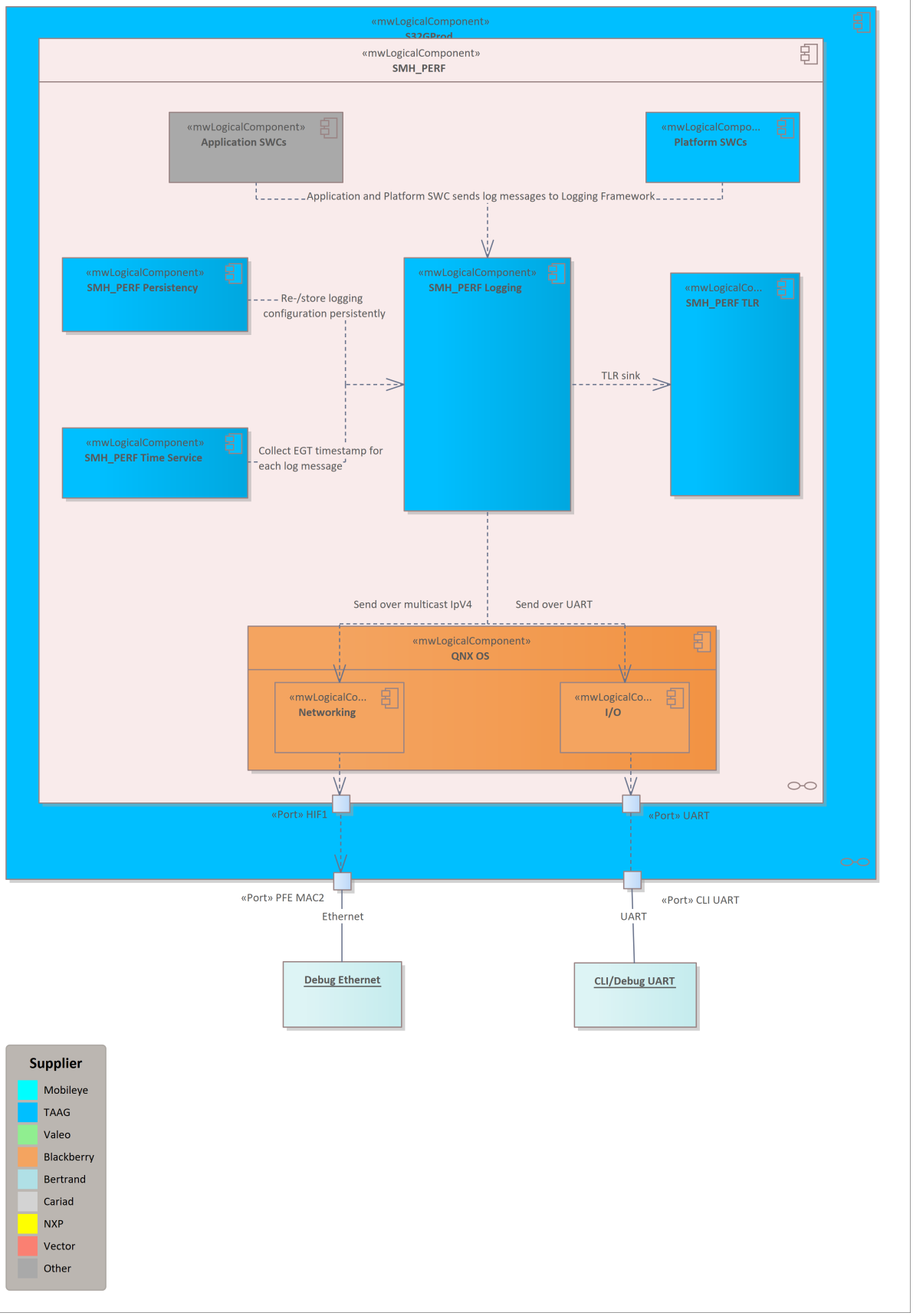
The output is provided in either decimal, hexadecimal, or binary format according to the message's configuration.

[S32GPRODP-296737 **]**



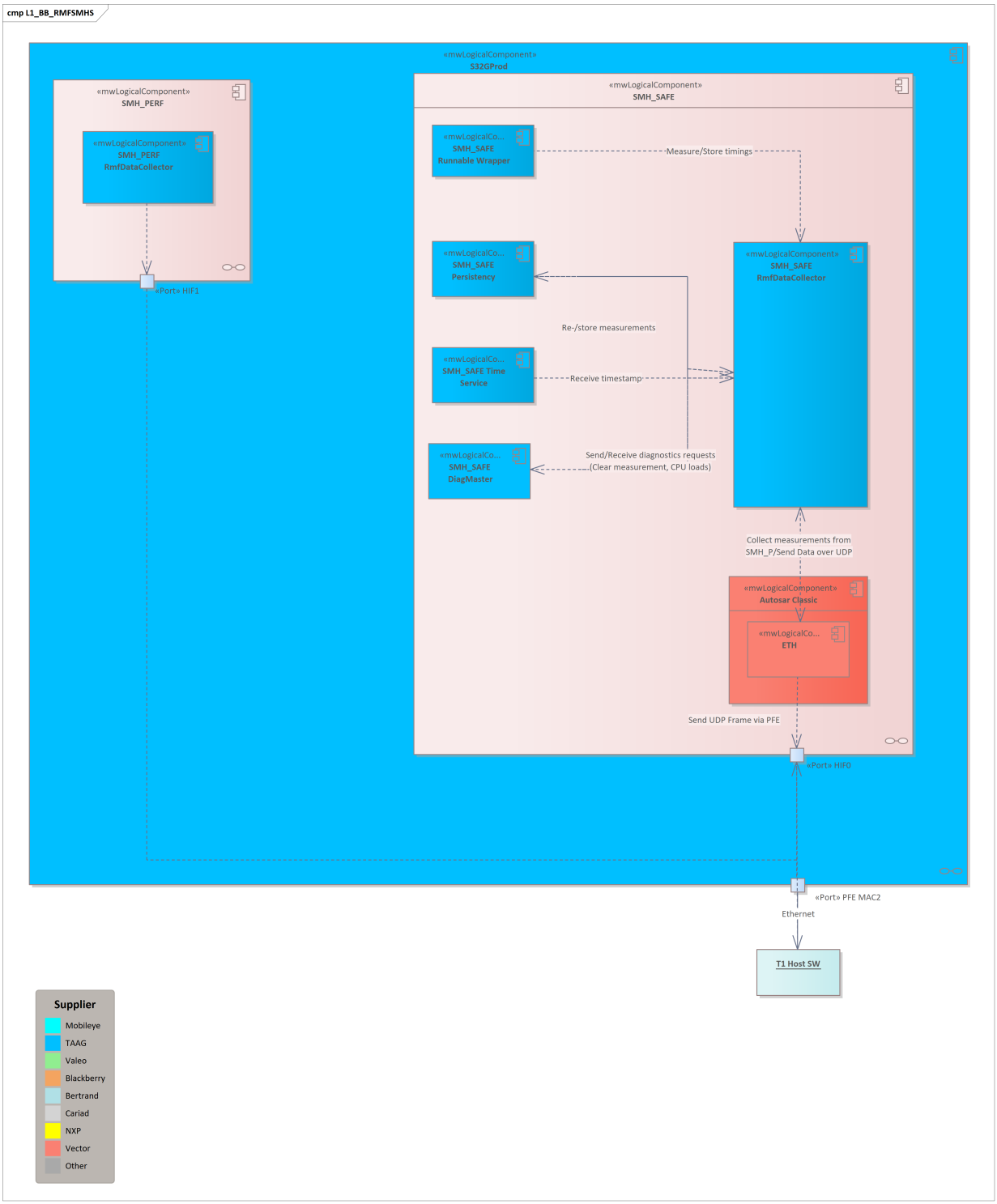

Cariad
NXP
Vector
Other

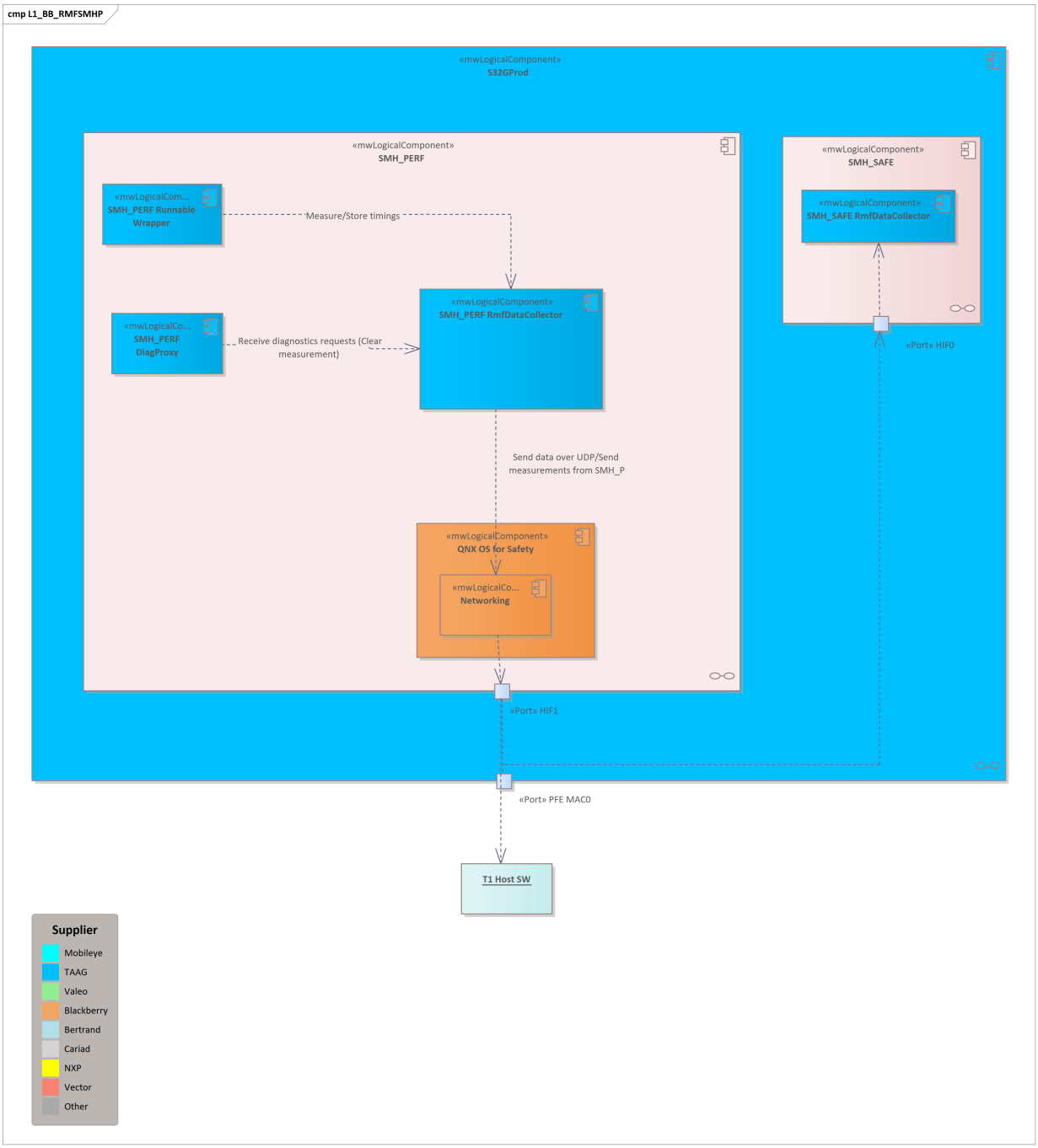
cmp L1_BB_LoggingFrameworkSMHP



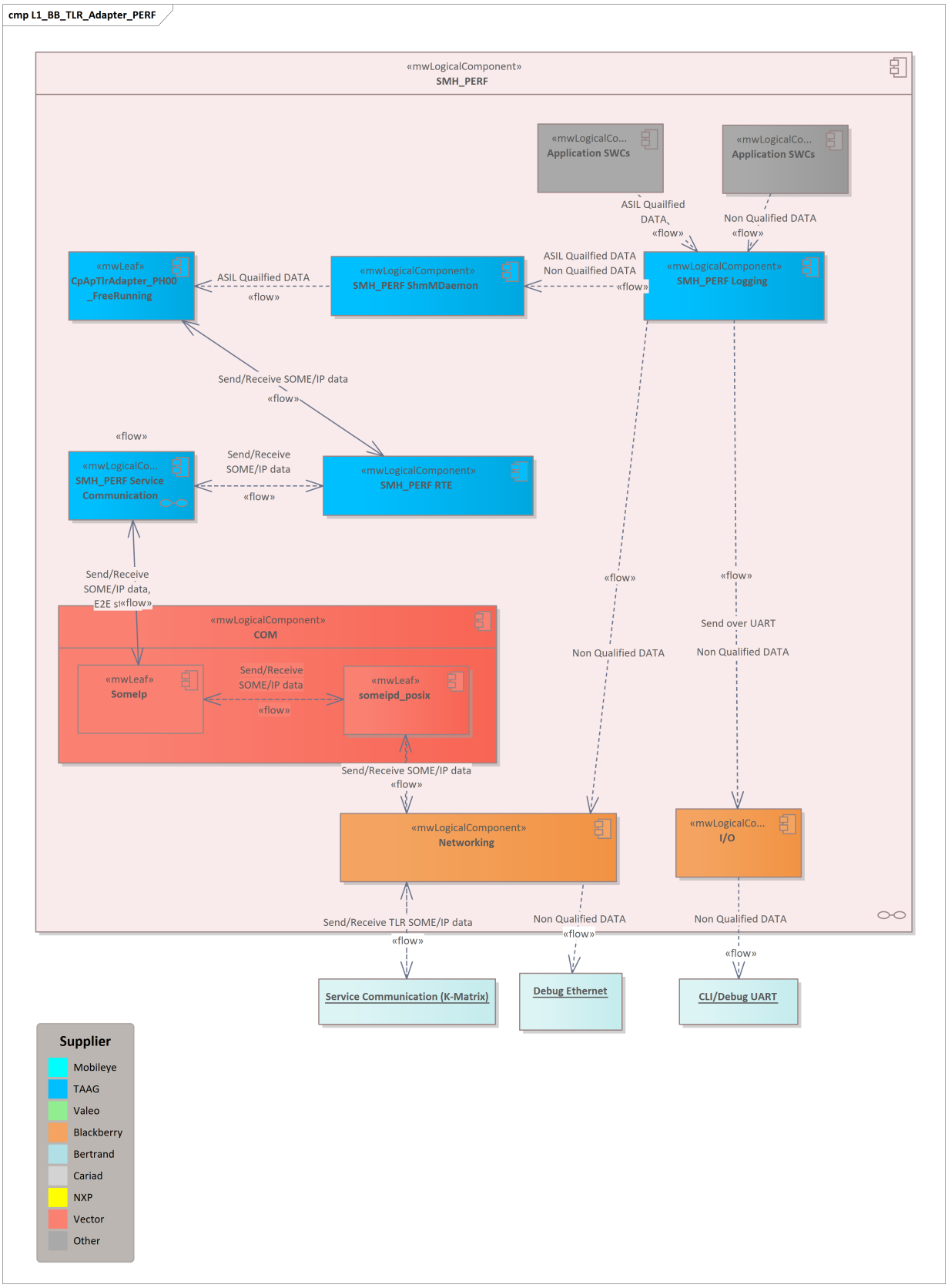
5.2.8.7 Resource Measurement Framework (RMF)

Resource Measurement Framework (RMF) is a platform service, deployed on both partition, which integrates GLIWA T1 within and utilizes tracing for each SWC integrated into MotionWise Time-Triggered scheduler. These tracing information is later sent over UDP to be decoded on T1-HOST-SW (GUI provided by GLIWA) which resolves and vusializes tracing information. RMF also provides APIs towards to SWCs for custom instrumentation for profiling with finer granularity.
Each RMF service works independently apart from the persistent storage and connection to diagnostics. RMF on SMH_S collects data (min/max CPU utilization for each core) from RMF on SMH_P to be persistently stored or send it over diagnostics. **[**S32GPRODP-296738 **]**

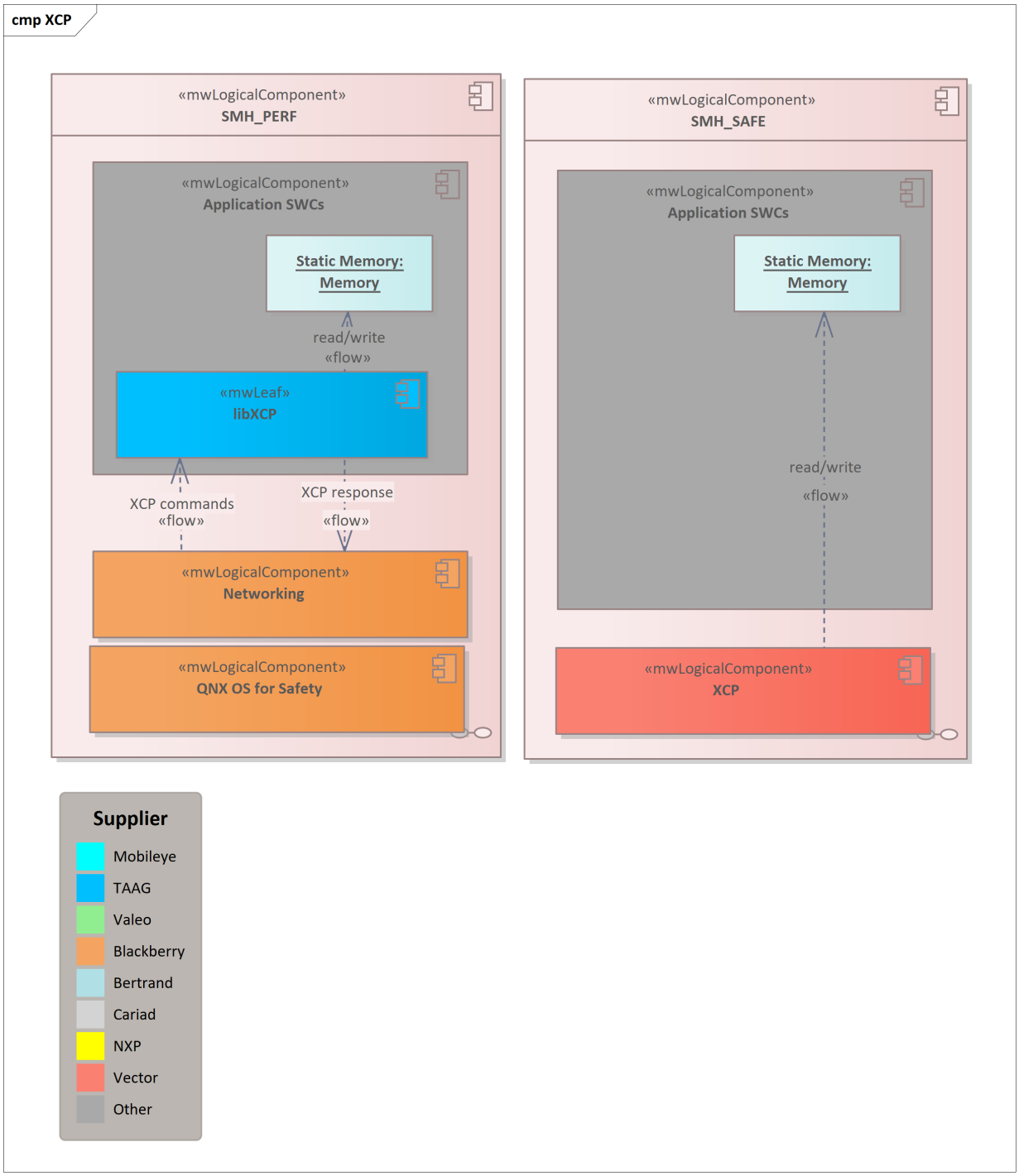

cmp L1_BB_RMFSMHS
«mwLogicalComponent»
S32GProd
«mwLogicalComponent»
SMH_PERF
«mwLogicalComponent»
SMH_PERF
RmfDataCollector
«Port» HIF1
«mwLogicalComponent»
SMH_SAFE
«mwLogicalCo...
SMH_SAFE
Runnable Wrapper
Measure/Store timings
«mwLogicalCo...
SMH_SAFE
Persistency
«mwLogicalComponent»
SMH_SAFE
RmfDataCollector
Re-/store measurements
«mwLogicalCo...
SMH_SAFE Time
Service
Receive timestamp
«mwLogicalCo...
SMH_SAFE
DiagMaster
Send/Receive diagnostics requests
(Clear measurement, CPU loads)
Collect measurements from
SMH_P/Send Data over UDP
«mwLogicalComponent»
Autosar Classic
«mwLogicalCo...
ETH
Send UDP Frame via PFE
«Port» HIF0
«Port» PFE MAC2
Ethernet
T1 Host SW
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other


cmp L1_BB_RMFSMHP
«mwLogicalComponent»
S32GProd
«mwLogicalComponent»
SMH_PERF
«mwLogicalCom...
SMH_PERF Runnable Wrapper
Measure/Store timings
«mwLogicalCo...
SMH_PERF DiagProxy
Receive diagnostics requests (Clear measurement)
«mwLogicalComponent»
SMH_PERF RmfDataCollector
Send data over UDP/Send measurements from SMH_P
«mwLogicalComponent»
QNX OS for Safety
«mwLogicalCo...
Networking
«Port» HIF1
«mwLogicalComponent»
SMH_SAFE
«mwLogicalComponent»
SMH_SAFE RmfDataCollector
«Port» HIF0
«Port» PFE MAC0
T1 Host SW
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

5.2.8.8 TLR


cmp L1_BB_TLR_Adapter_PERF
«mwLogicalComponent»
SMH_PERF
«mwLogicalCo...
Application SWCs
«mwLogicalCo...
Application SWCs
ASIL Quailfied DATA
«flow»
Non Qualified DATA
«flow»
«mwLeaf»
CpApTlrAdapter_PH00
_FreeRunning
ASIL Quailfied DATA
«flow»
«mwLogicalComponent»
SMH_PERF ShmMDaemon
ASIL Quailfied DATA
Non Qualified DATA
«flow»
«mwLogicalComponent»
SMH_PERF Logging
Send/Receive SOME/IP data
«flow»
«flow»
«mwLogicalCo...
SMH_PERF Service
Communication
Send/Receive
SOME/IP data
«flow»
«mwLogicalComponent»
SMH_PERF RTE
Send/Receive
SOME/IP data,
E2E s«flow»
«flow»
«flow»
«mwLogicalComponent»
COM
Send over UART
Non Qualified DATA
Non Qualified DATA
«mwLeaf»
SomeIp
Send/Receive
SOME/IP data
«flow»
«mwLeaf»
someipd_posix
Send/Receive SOME/IP data
«flow»
«mwLogicalComponent»
Networking
«mwLogicalCo...
I/O
Send/Receive TLR SOME/IP data
«flow»
Non Qualified DATA
«flow»
Non Qualified DATA
«flow»
Service Communication (K-Matrix)
Debug Ethernet
CLI/Debug UART
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

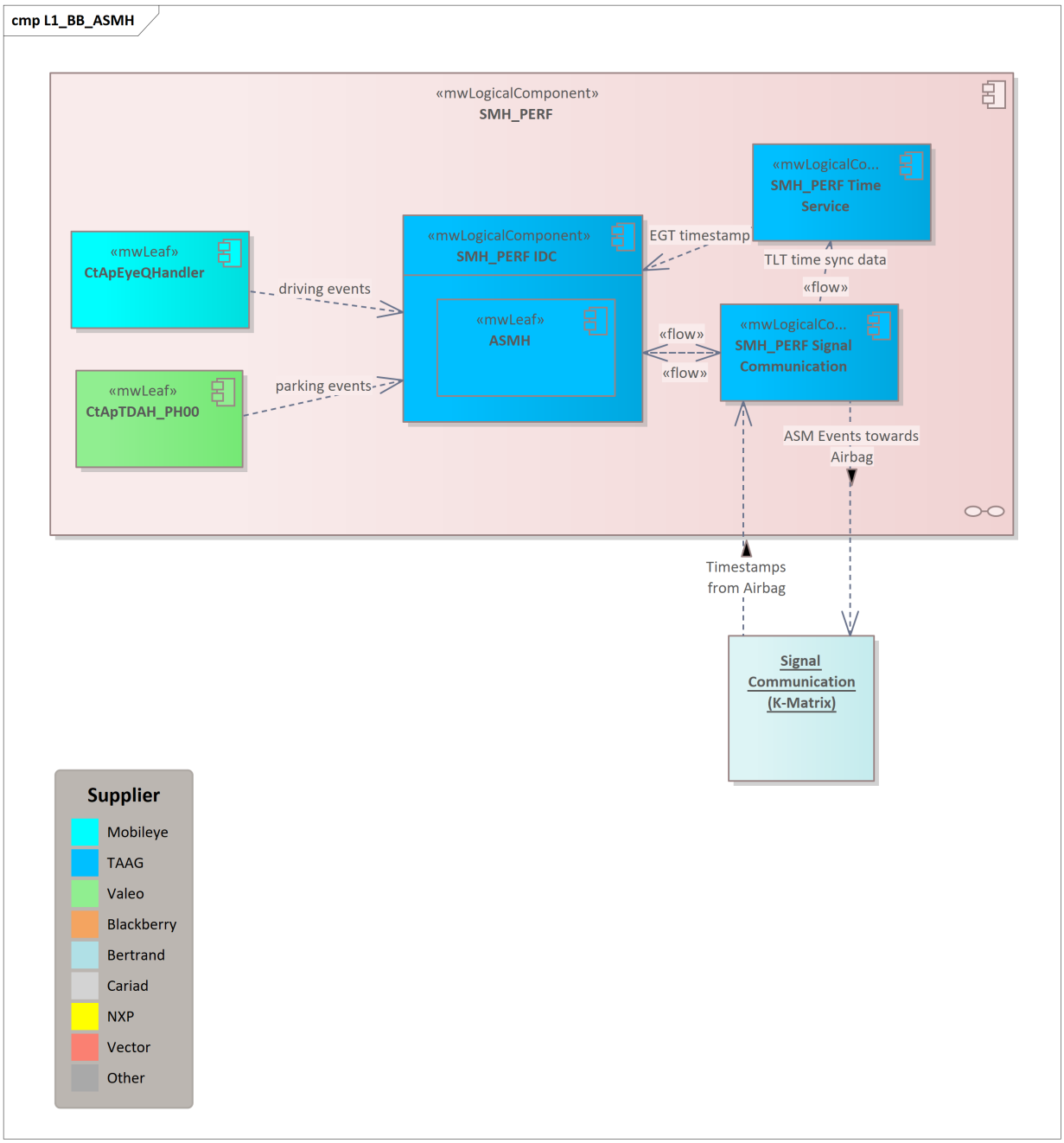
5.2.8.9 Calibration (XCP)



Calibration Service uses XCP protocol to access the memory of the host for calibration and measurement. XCP is based on the master/slave principle. Safety Host and Performance Host are slaves and the measurement and calibration tool (typically a PC equipped with necessary XCP hardware and software) act as the master.
Vector XCP module is used on the Safety host. While on the performance host libXCP is used which is integrated with every App SWC to provide the XCP functionality. **[**S32GPRODP-296835 **]**

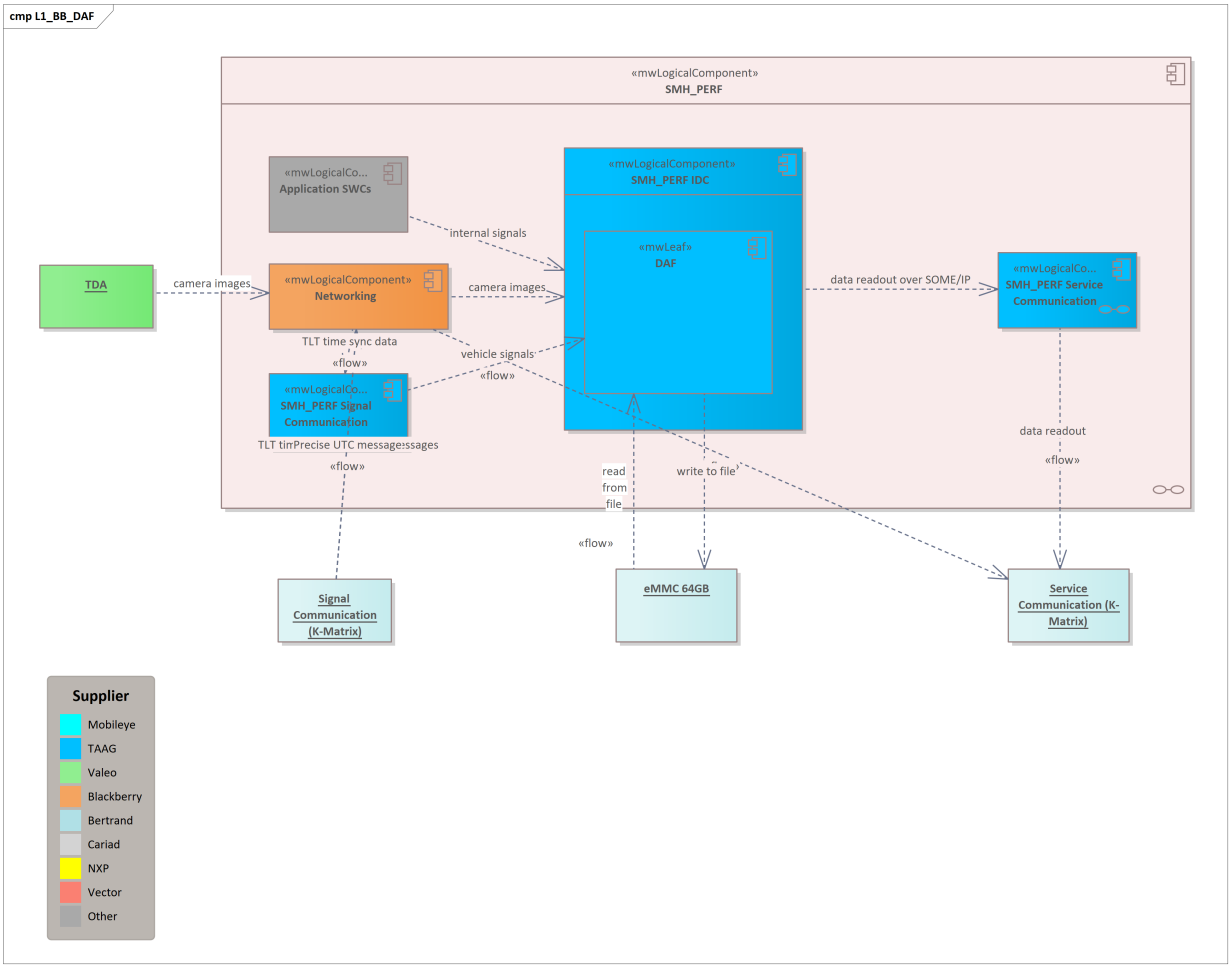
5.2.9 Data recording

5.2.9.1 ASMH



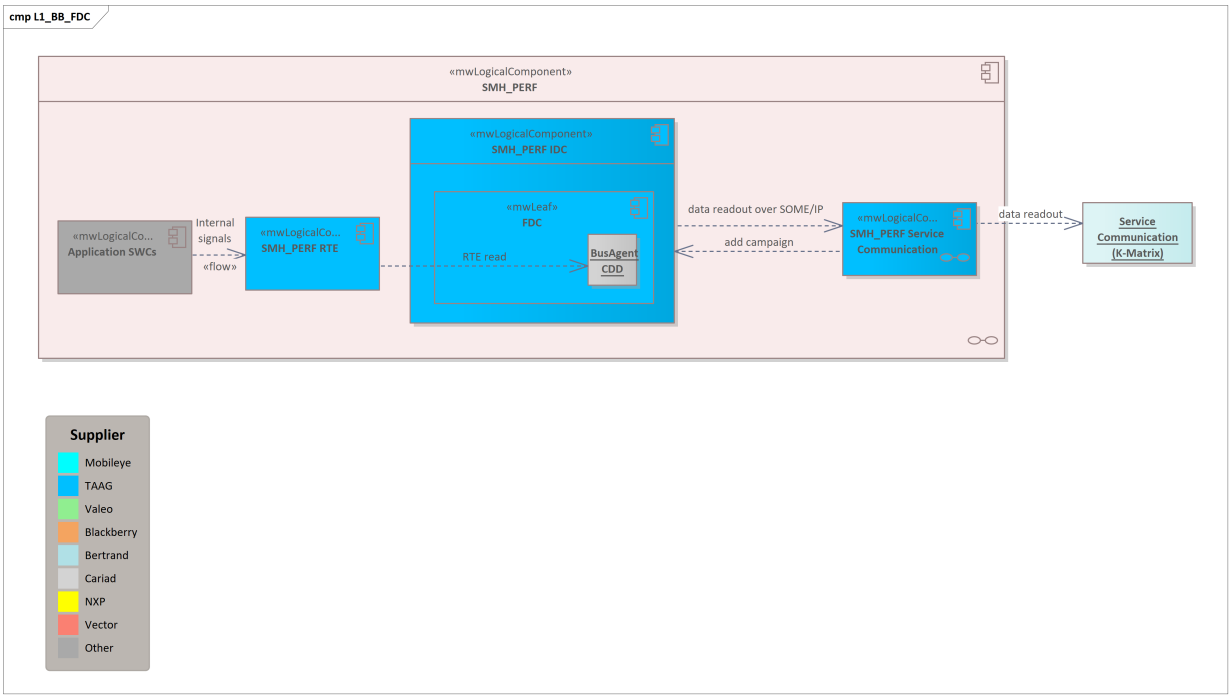
The Assistance System Monitor (ASM) stores ADAS data and integral safety (IS) data in the vehicle. This distributed function prepares status data of ADAS system, which are transmitted event-based via vehicle networking to central data storage. If an event that is worth recording occurs (e.g., an accident), the recorded data is stored in non-volatile memory in the airbag controller for subsequent evaluation. The OEM can use diagnostic tools to read out the data.
[S32GPRODP-296313]

5.2.9.2 DAF



The DAF records data from participating customer functions for product liability and accident reconstruction or for development purpose or for customer purpose. In addition, DAF also saves images from parking and driving cameras. [S32GPRODP-296317]

5.2.9.3 FDC



The FDC is a feature that is used to collect several types of data:

- communication data between SWCs
- communication data from external ECUs to SWCs
- communication data from SWCs to external ECUs

[S32GPRODP-296314]

From the structural point of view, FDC comprises of:

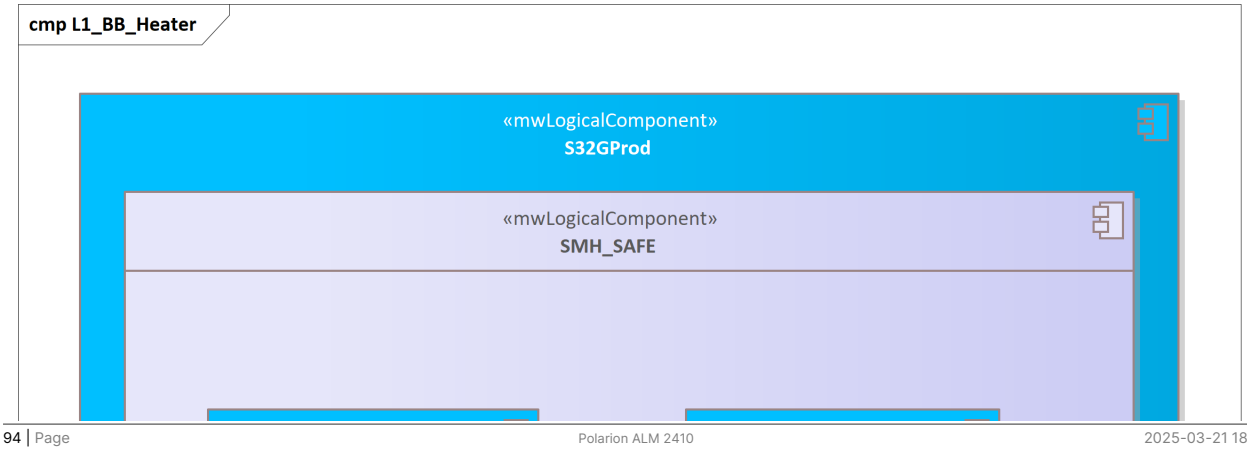
- **BusAgentCDD** updates the BusAgentSWC shared queue with the values collected from the RTE;
- **BusAgentSWC** (a.k.a. GDC) in charge of external communication and configuration management;

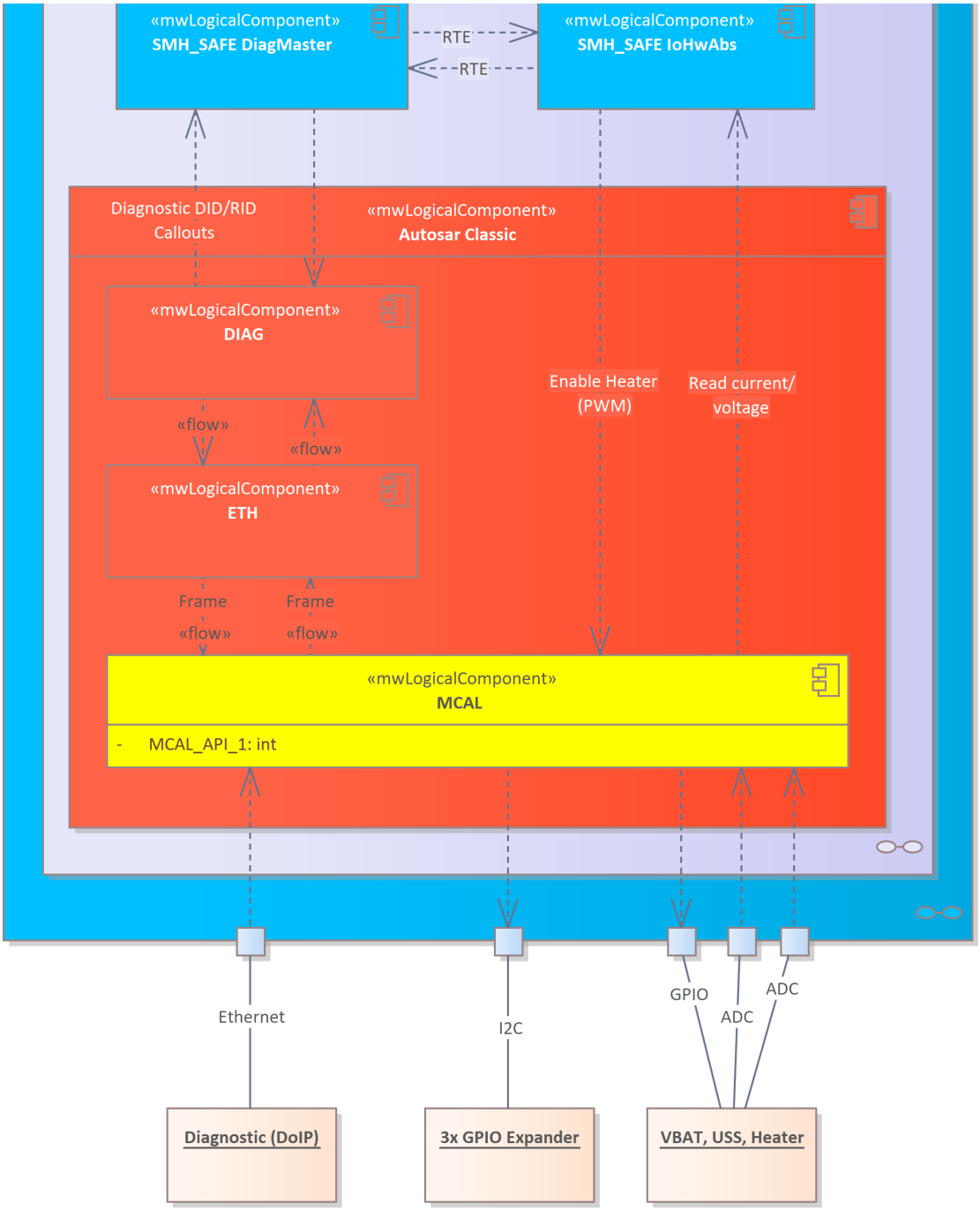
[S32GPRODP-296315]

The request for what data to capture is received from outside of the ECU over Protobuf via SOME\IP. [S32GPRODP-296316]

5.2.10 I/O Hardware Abstraction

5.2.10.1 Heater




«mwLogicalComponent»
SMH_SAFE DiagMaster
RTE
RTE
«mwLogicalComponent»
SMH_SAFE IoHwAbs
Diagnostic DID/RID
Callouts
«mwLogicalComponent»
Autosar Classic
«mwLogicalComponent»
DIAG
«flow»
«flow»
Enable Heater
(PWM)
Read current/
voltage
«mwLogicalComponent»
ETH
Frame
«flow»
Frame
«flow»
«mwLogicalComponent»
MCAL
- MCAL_API_1: int
Ethernet
I2C
GPIO
ADC
ADC
Diagnostic (DoIP)
3x GPIO Expander
VBAT, USS, Heater

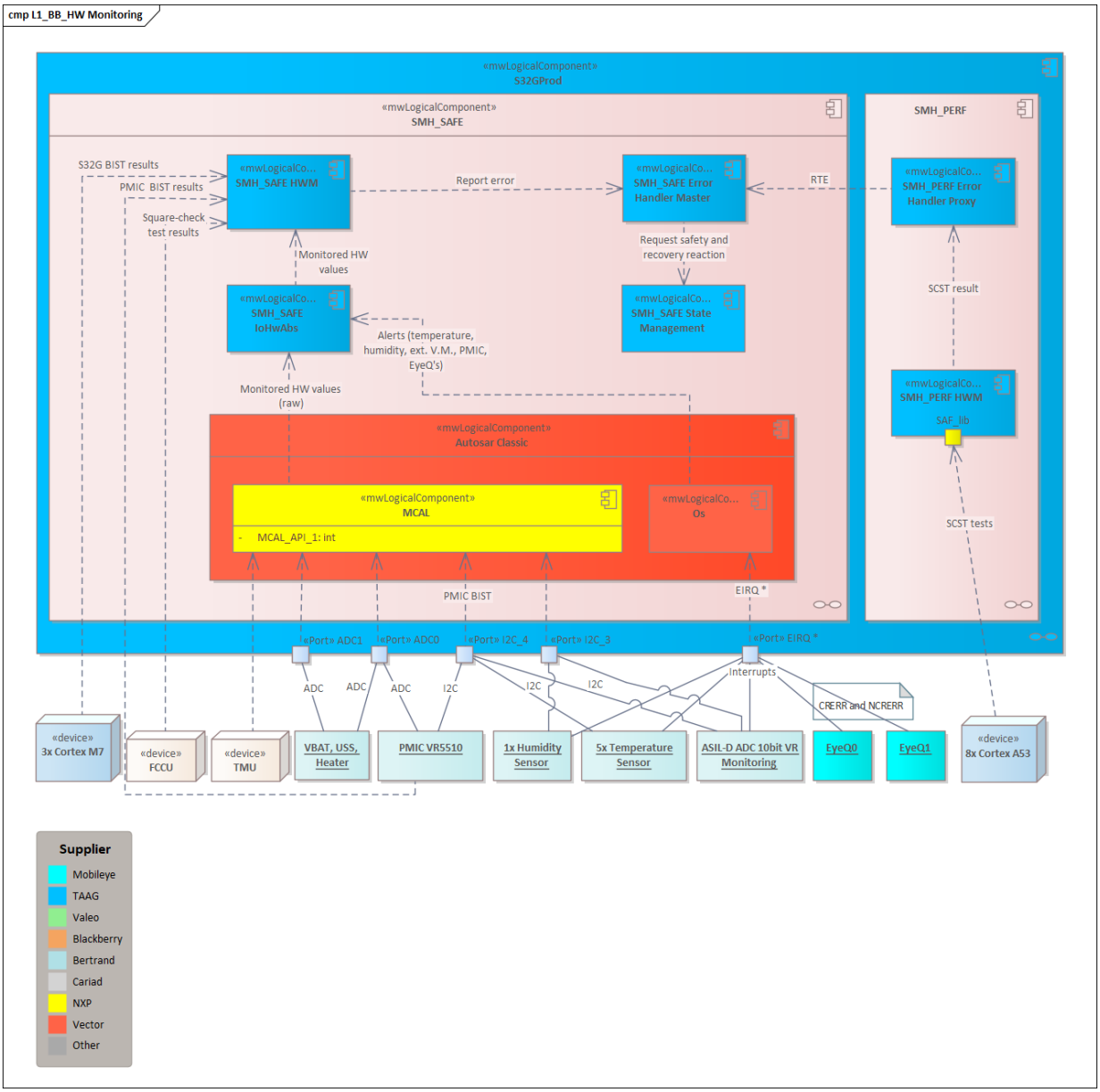

Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector

NOTE: This is a temporary implementation, the camera heater function logic will be moved to a customer software component. [S32GPRODP-296168]

The camera heater is a functionality, which is implemented inside the I/O Hardware Abstraction component on the SMH_SAFE partition. It's purpose is to control and monitor the high-side (HSS) and low-side (LSS) switches, that are powering the camera heater.
Heater requests are triggered by a diagnostic routine (0x318) and are forwarded by the diagnostic stack to the I/O Hardware Abstraction component. Based on diagnostic routine parameters, I/O Hardware Abstraction generates a PWM pulse, which enables the camera heater. During activation, HSS and LSS error statuses are monitored, [S32GPRODP-296167]

5.2.11 Health

5.2.11.1 HW monitoring

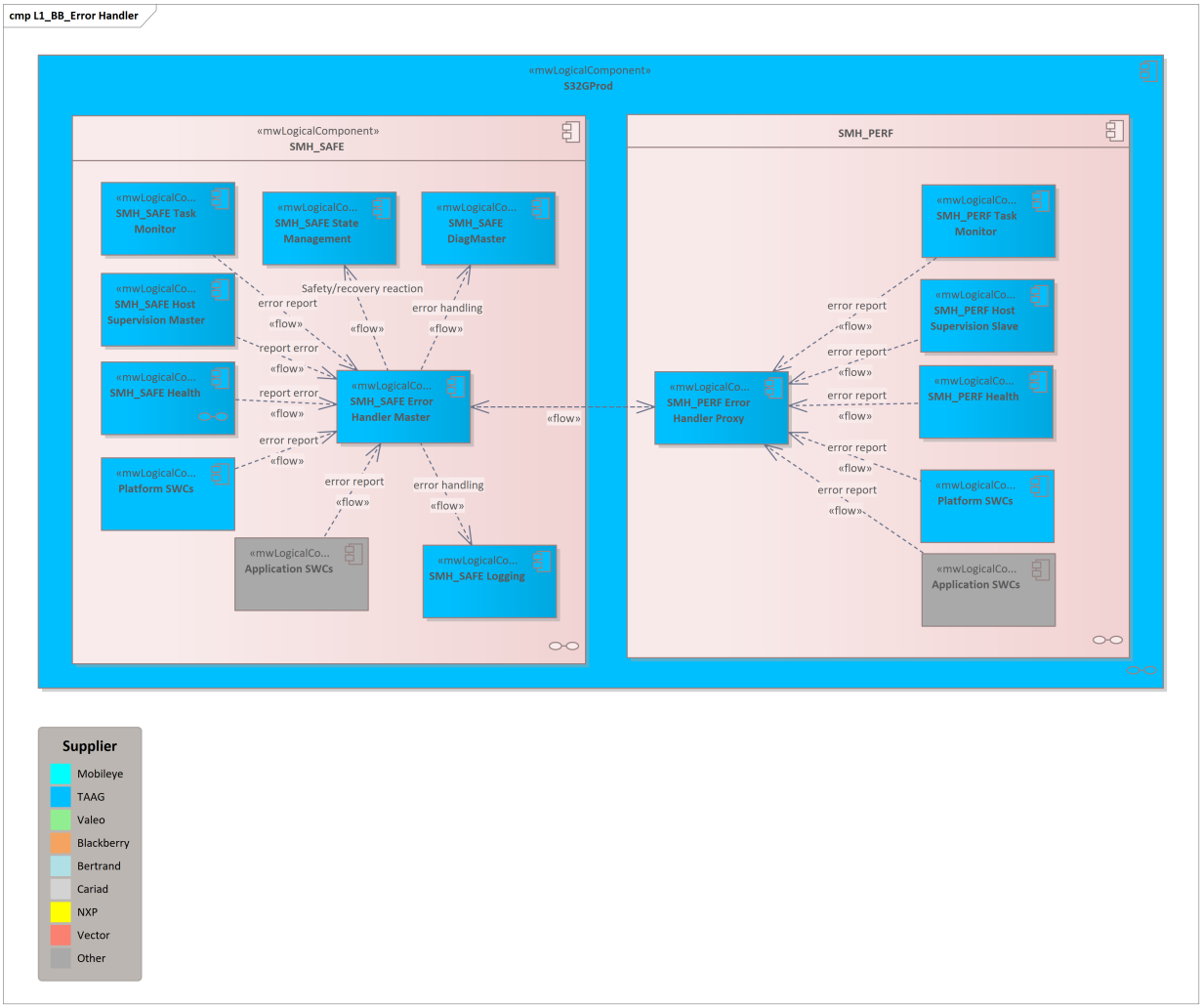


Health HW Monitoring is responsible for:

- monitoring of the HW signals specified to be monitored by the SMH, like: voltages, temperatures, humidity, EyeQ (CREER/NCRERR)
- executing and monitoring of BIST tests, like BIST tests for SMH_SAFE (LBIST, MBIST)), SMH_PERF (SCST), PMIC, EyeQ (BGA tests)

[S32GPRODP-296800]

5.2.11.2 Error handler

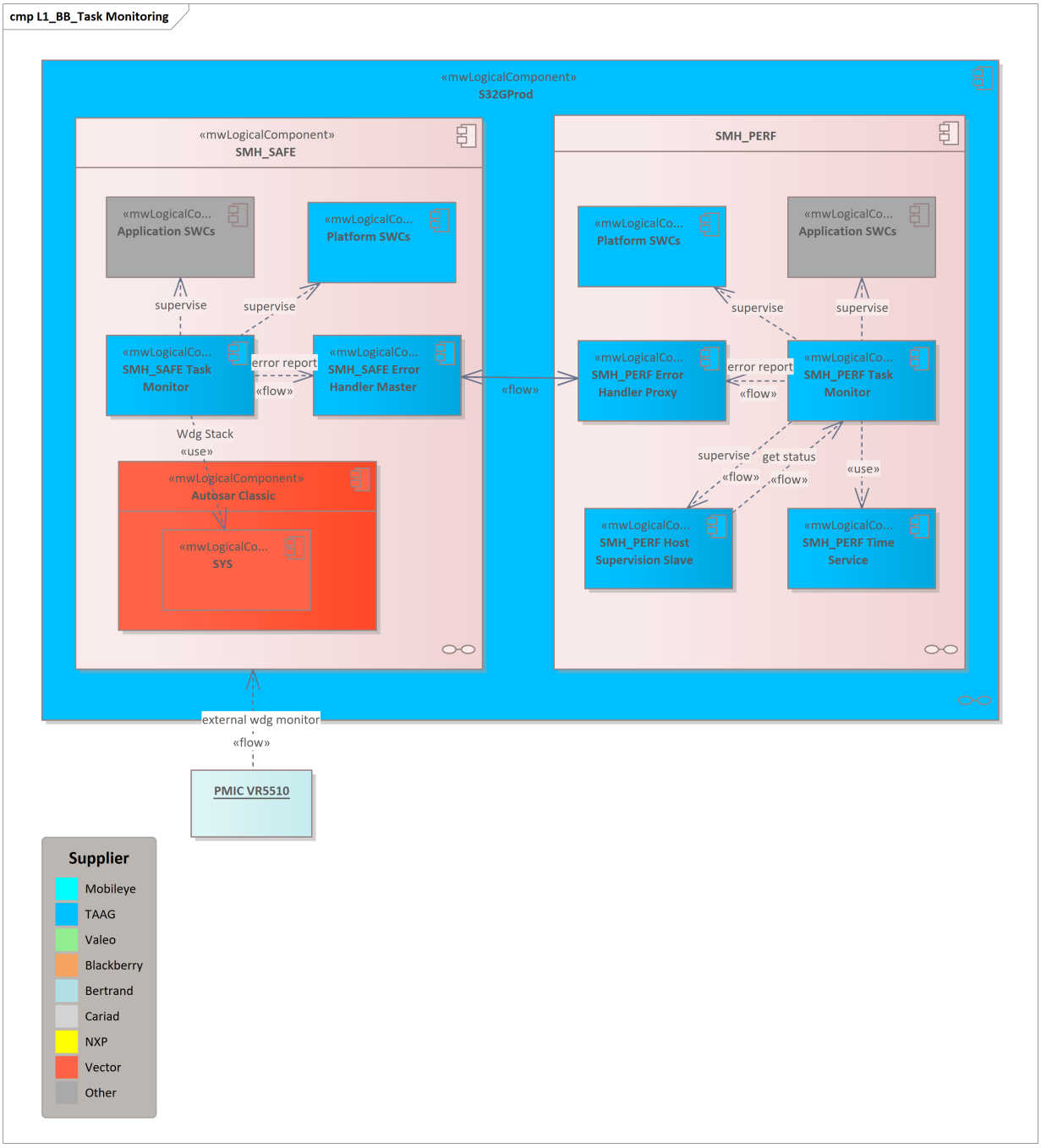


Error Handler is taking care of collecting errors from platform applications and customer applications (also platform middleware elements as well); based on different error type/severity, triggering error handling activities which may include but not limited to error logging, diagnostic recording, taking safety reaction and in case of necessary taking recovery action.

Error Handler has a master-proxy structure across safety host and performance host. Error Handler Proxy is deployed on performance host and collect errors from different modules from this host; it would report collected errors to the Error Handler Master which is deployed on safety host. Error Handler Master is the central module of the concept: errors received from Error Handler Proxy and local errors reported from safety host modules would be proceeded and mitigated by this module; the error reaction is overall decided by Error Handler Master.

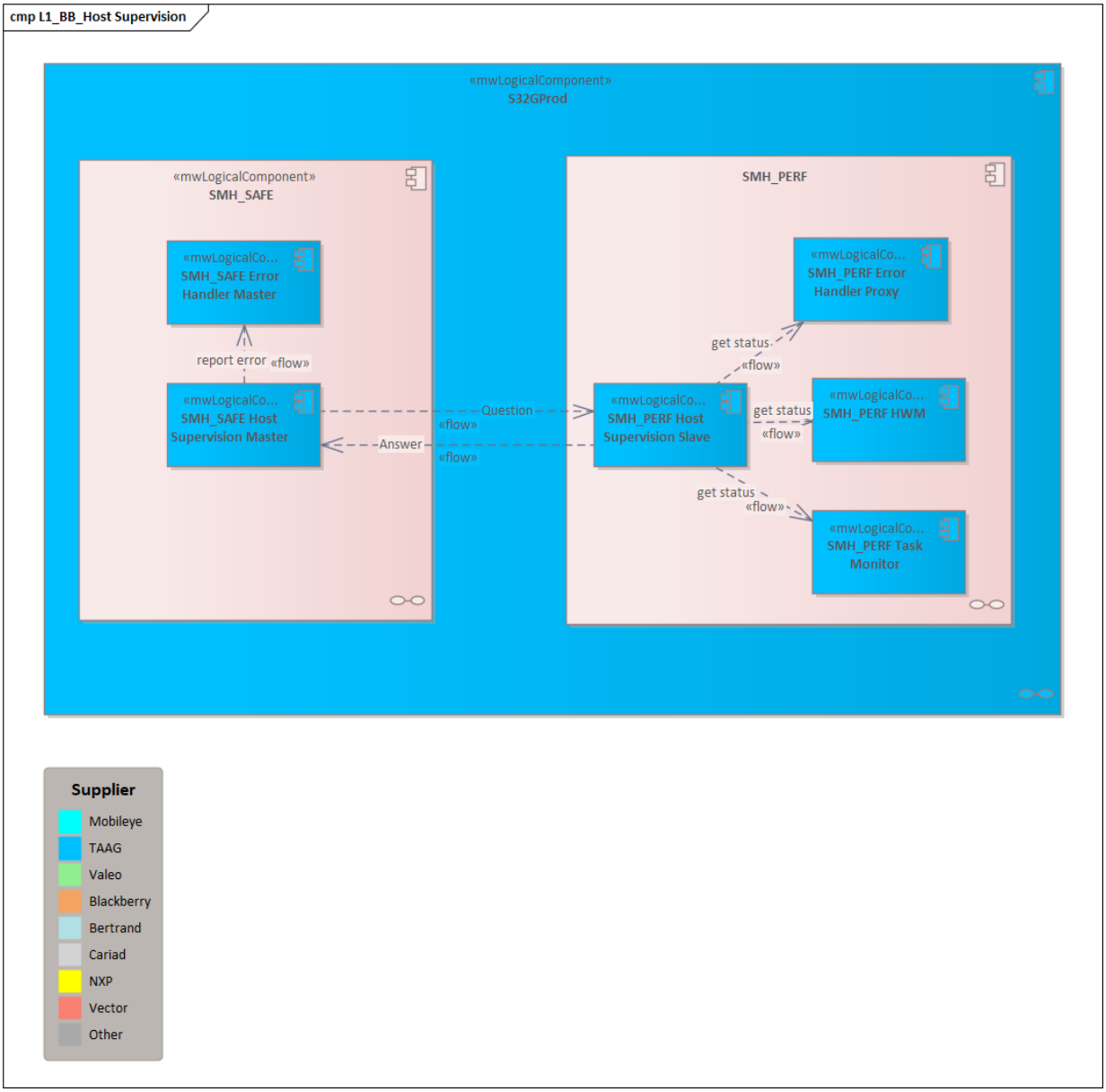
Error handler is also the central part of system safety concept as well. **[**S32GPRODP-296834 **]**

5.2.11.3 Task monitoring



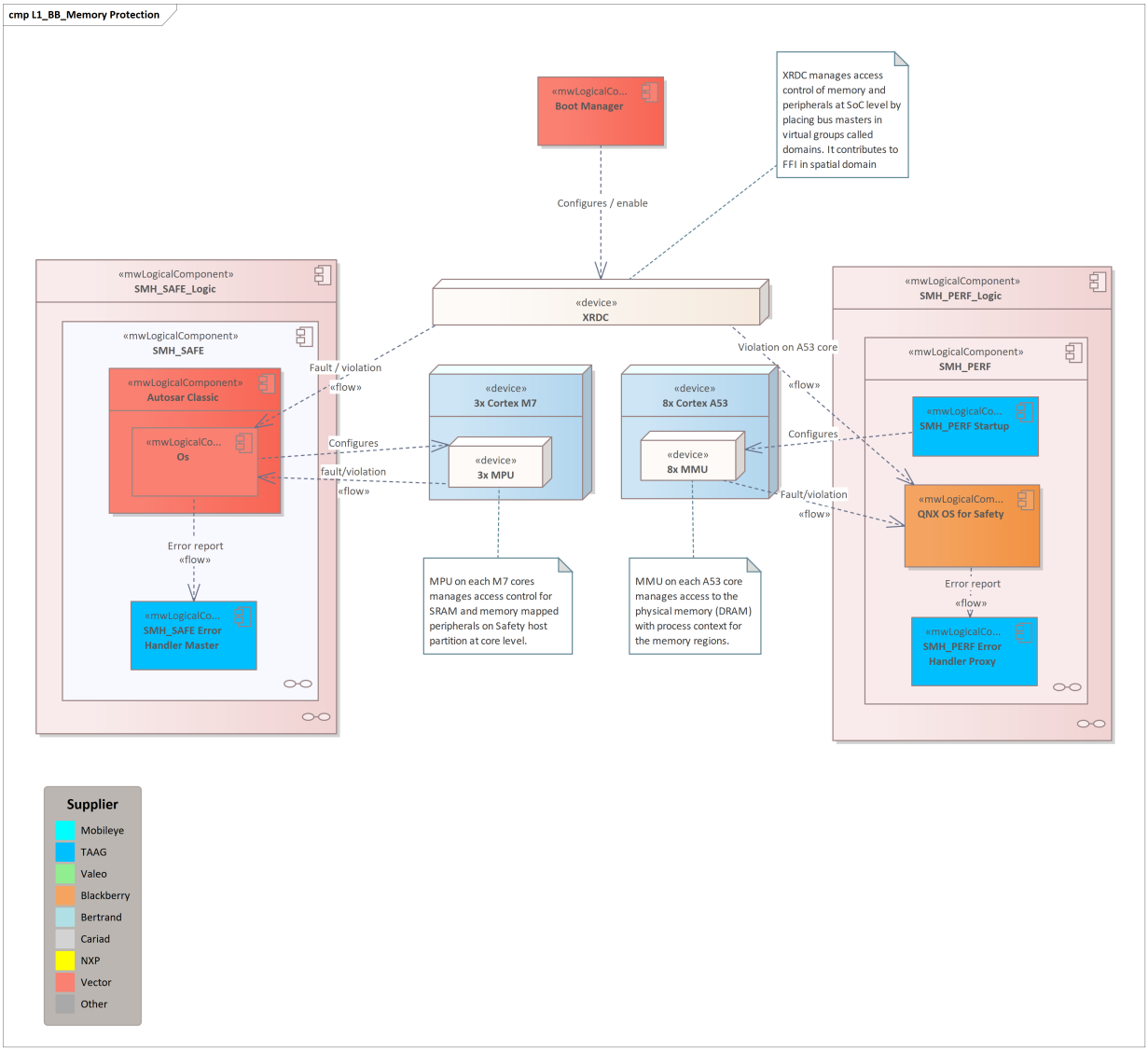
Task Monitoring is targeted to provide program execution monitor functionalities: aliveness monitor, deadline monitor, program flow monitor. It has 2 independent task monitor modules on safety host and performance host; each of them relies on watchdog stack from AutoSAR and platform implementation. Additionally external watchdog of PMIC contributes to this task monitor concept as well. [S32GPRODP-296832]

5.2.11.4 Host supervision



Host Supervision is based on a master-slave concept to allow safety host to check the status of performance host. Host Supervision Master (HSVM) on safety host would cyclically send predefined questions to Host Supervision Slave (HSVS) on performance host; on receiving the questions from HSVM, HSVS would collect status from dependent modules on performance host and then formulate answer back to HSVM. If the answer is incorrectly provided, HSVM would regard as error and report to error handler master.
Host Supervision works closely with task monitoring and Error Handler; it works as a redundant path for Error Handler. From safety point of view, Host Supervision, Task Monitoring and Error Handling compose a sound loop of safe supervision. **[**S32GPRODP-296833 **]**

5.2.12 Memory protection

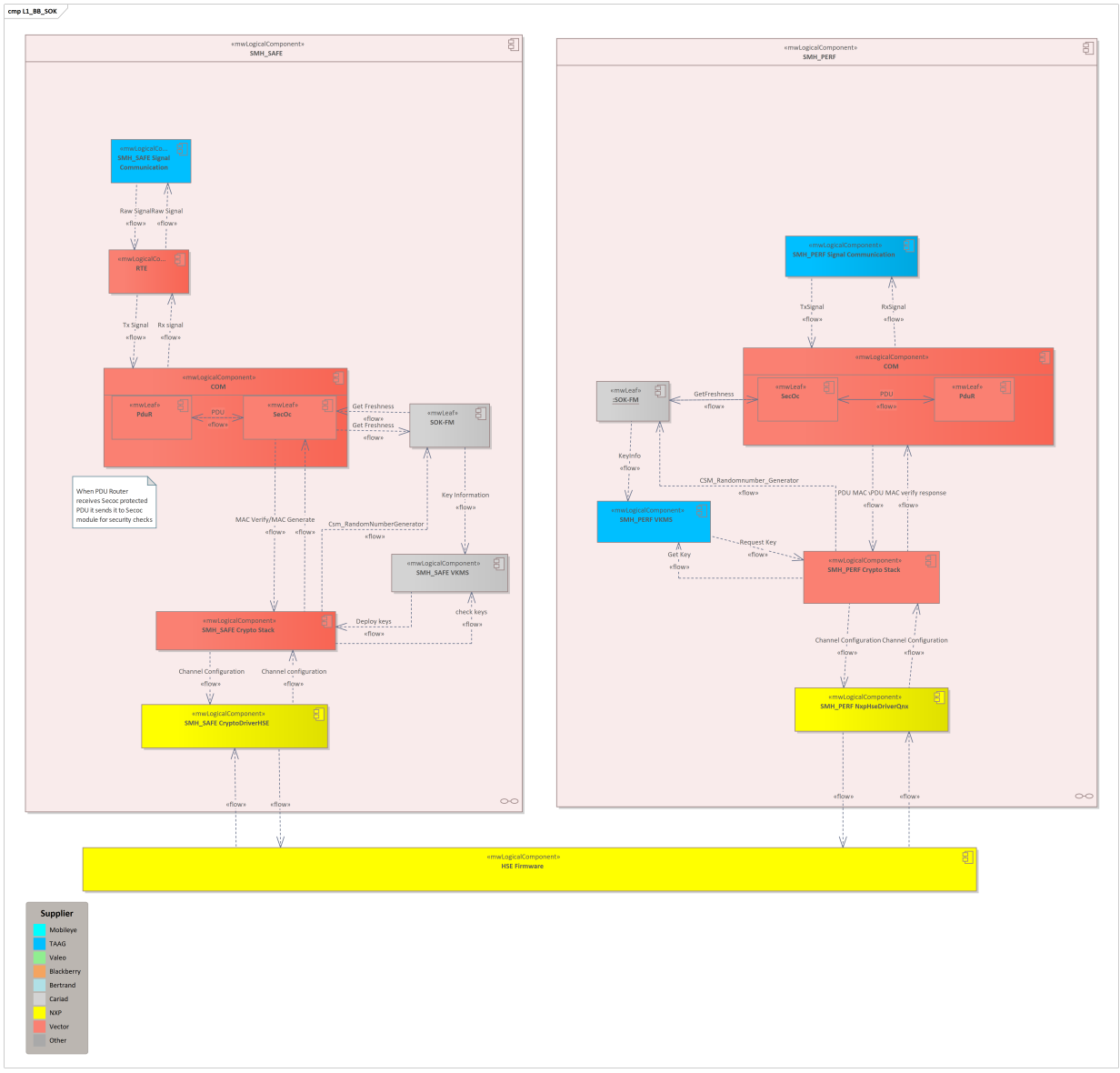


Memory protection feature ensures FFI (Freedom from Interference) in spatial domain by controlling access to memory and peripherals on Safety host and performance host partition by using MPU, MMU and XRDC. Each M7 core has its own MPU which is configured by Microsar OS. MPU configuration ensures lower ASIL components can't access the higher ASIL assigned memory regions. On PH cores, memory access is controlled by MMU with virtual address space creation for each process context. XRDC provides SOC system level memory access control and peripheral isolation by assigning chip resources (memory, peripherals) to processing domains. XRDC is configured by Boot manager component during startup phase of SH with fault handlers on the respective host cores where fault is reported.

[S32GPRODP-296033]

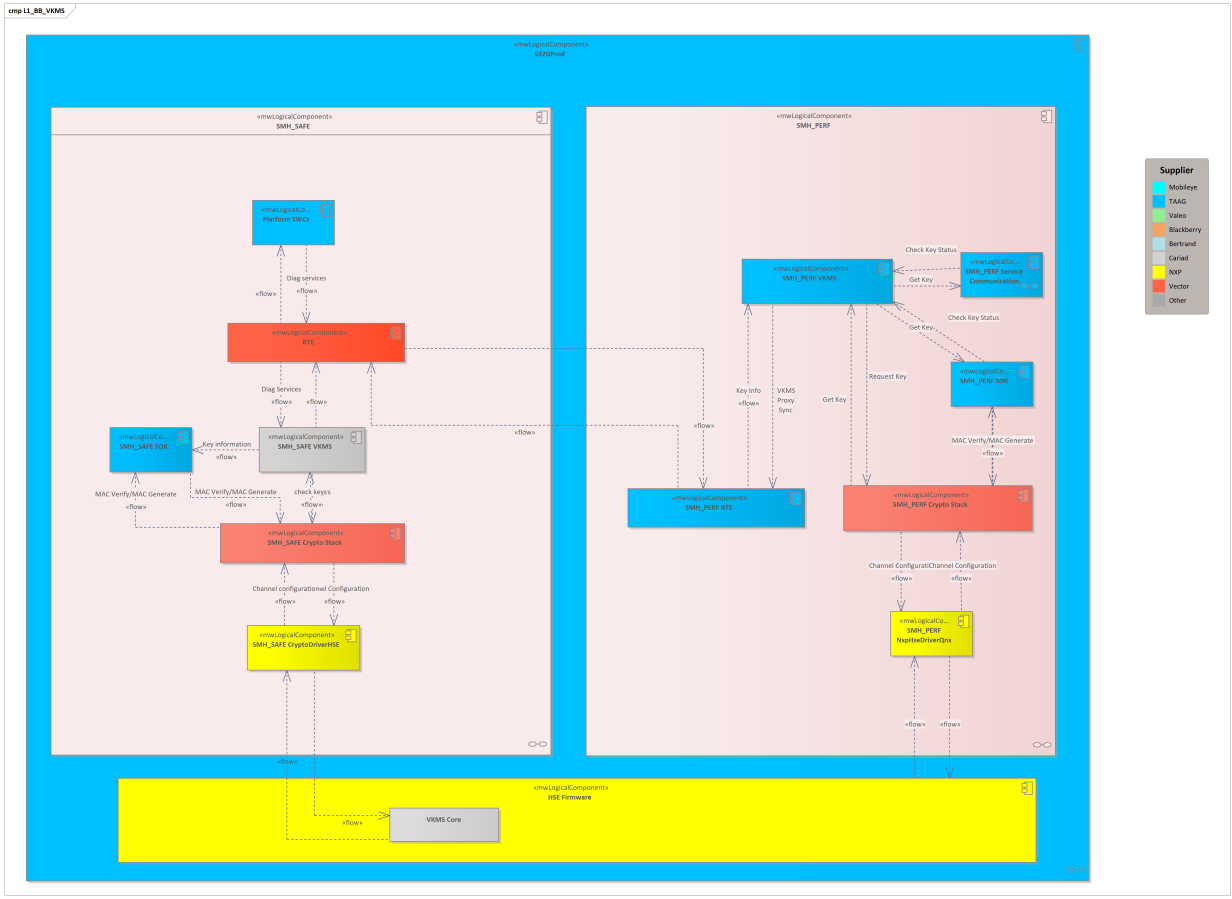
5.2.13 Security

5.2.13.1 SOK



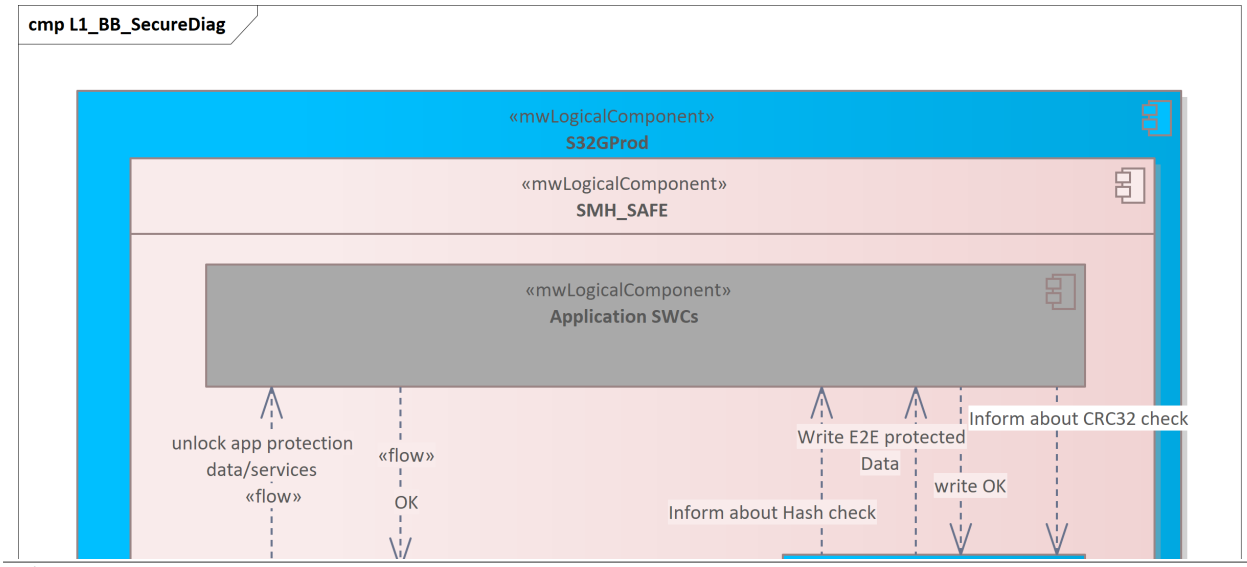
SOK (Sichere on-board Kommunikation) is based on the AUTOSAR SecOC concept.
The purpose of the secure on-board Communication (SecOC) module is to provide an AUTOSAR BSW Module to transmit secured data between two or more ECUs exchanging information. SOK enables the communication of data within an ECU in a way that protects the integrity of the data against tampering. SOK protects the integrity of a communication by attaching signatures to the data to be protected. The data itself remains unchanged. This can be achieved by various modules like SOK-FM, CSM, VKMS. The SOK- FM serves the freshness value to prevent replay attacks. The Truncated authenticator is the result of the calculation of the cryptographic signature.
[S32GPRODP-296114]

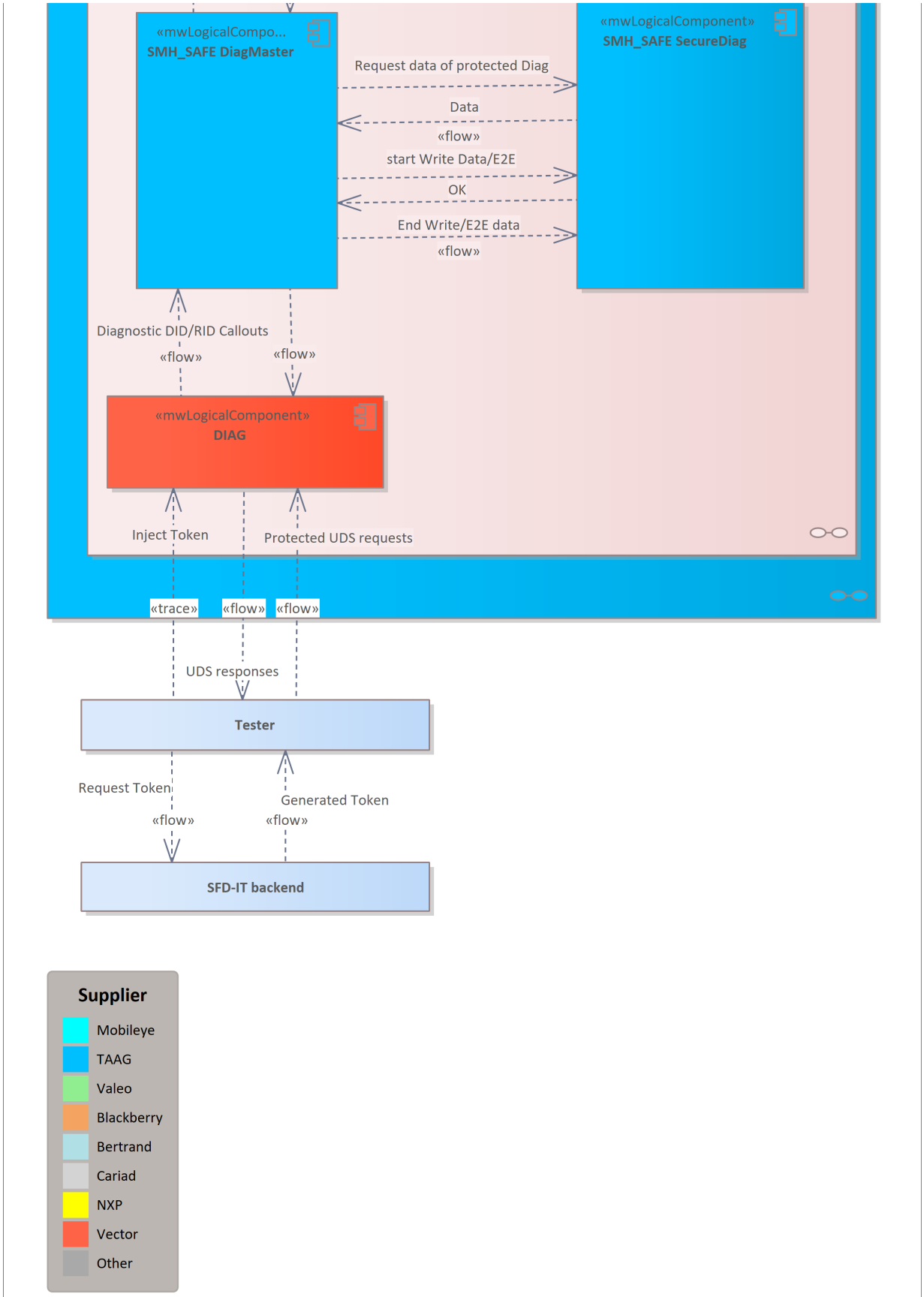
5.2.13.2 VKMS



VKMS is to provide possibility of centralized key provisioning and management on vehicle level. VKMS of ECU consists of a single VKMS core, VKMS adapter on SMH-SH and multiple VKMS proxies. VKMS core is running inside HSM of SMH, VKMS proxies are running on each partition including the ones outside of SMH that require centralized key management services. A mechanism of keys synchronization must be in place to ensure freshness and availability of the keys to VKMS clients. **[**S32GPRODP-296200 **]**

5.2.13.3 SFD (Secure Diagnostic)

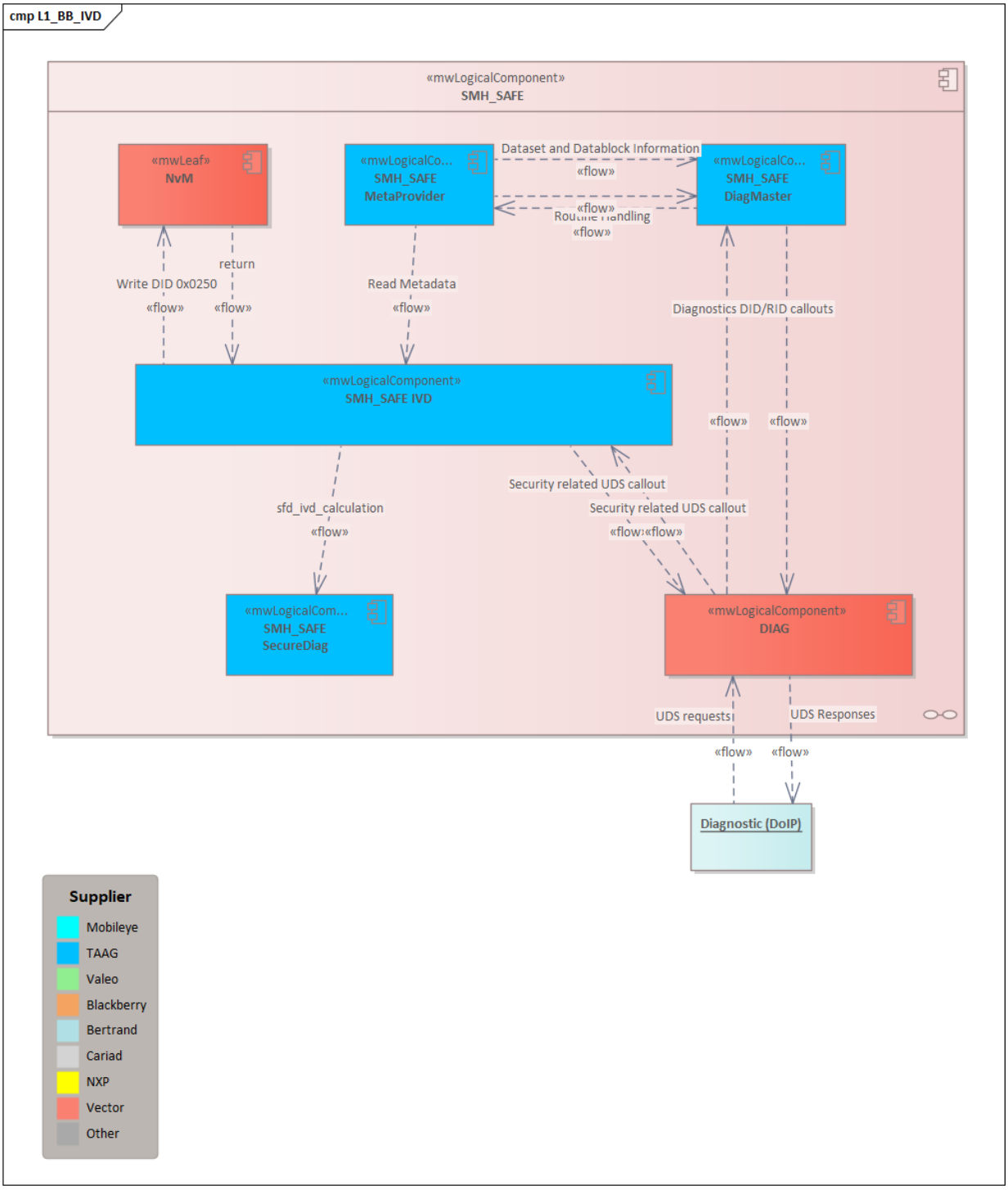




SFD is a module designed to secure specific diagnostic objects within an ECU to prevent unauthorized access.

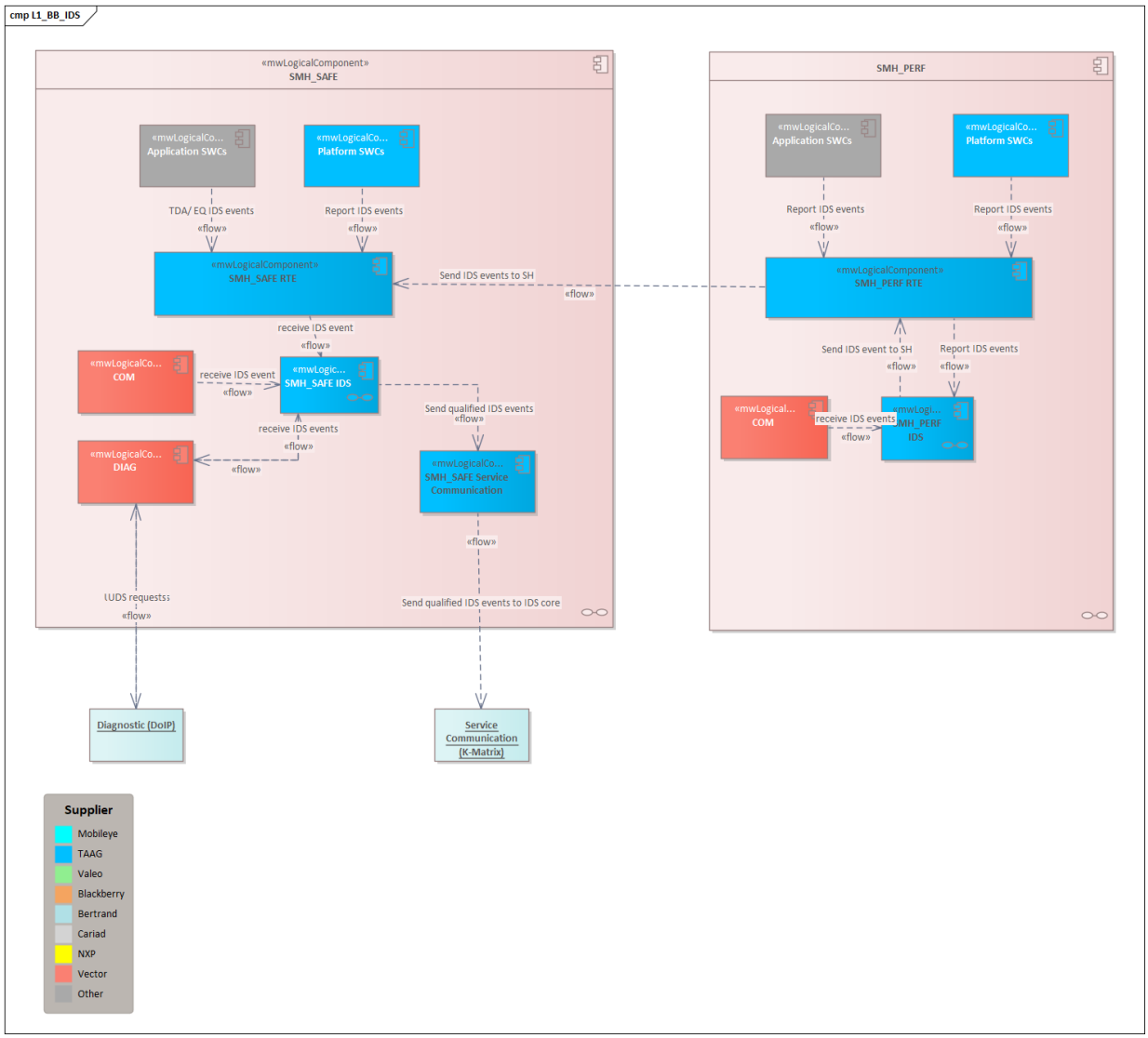
Each diagnostic object that is protected by SFD is either assigned a SFD Role (Role protected) or is end-to-end (E2E) protected. If the ECU does not have authorization to access a diagnostic object, it will receive a Security Access Denied. [S32GPRODP-296371]

5.2.13.4 IVD



IVD (Integrity validation data) is a module that guarantees the integrity and authenticity of software. IVD uses two hash-values per ECU to determine the integrity of the software (by using a programming hash) and configuration (by using the configuration hash). [S32GPRODP-296124]

5.2.13.5 IDS



IDS is to provide an interface to the services and components that will report qualified security events to the IDS system with abstraction from the low level dependencies on the IDS components and modules. This shall be possible for qualified security events delivered by the TDA4 or for qualified events reported by the EyeQs though the S32G. For such external events, it shall be possible to utilize a bypass service that would forwarding the security events coming from the TDA4. Similarly, a suitable EyeQ Bypass service shall be available for the qualified events collected at the EyeQ handler. Those shall be then forwarded via the before mentioned service to the PH-Handler at the Safety host and sent then to the IDS core without modification. [S32GPRODP-296373]

5.2.14 Design Validation (DV)

Design validation software implements, integrate, or configure checks and statistics for following:

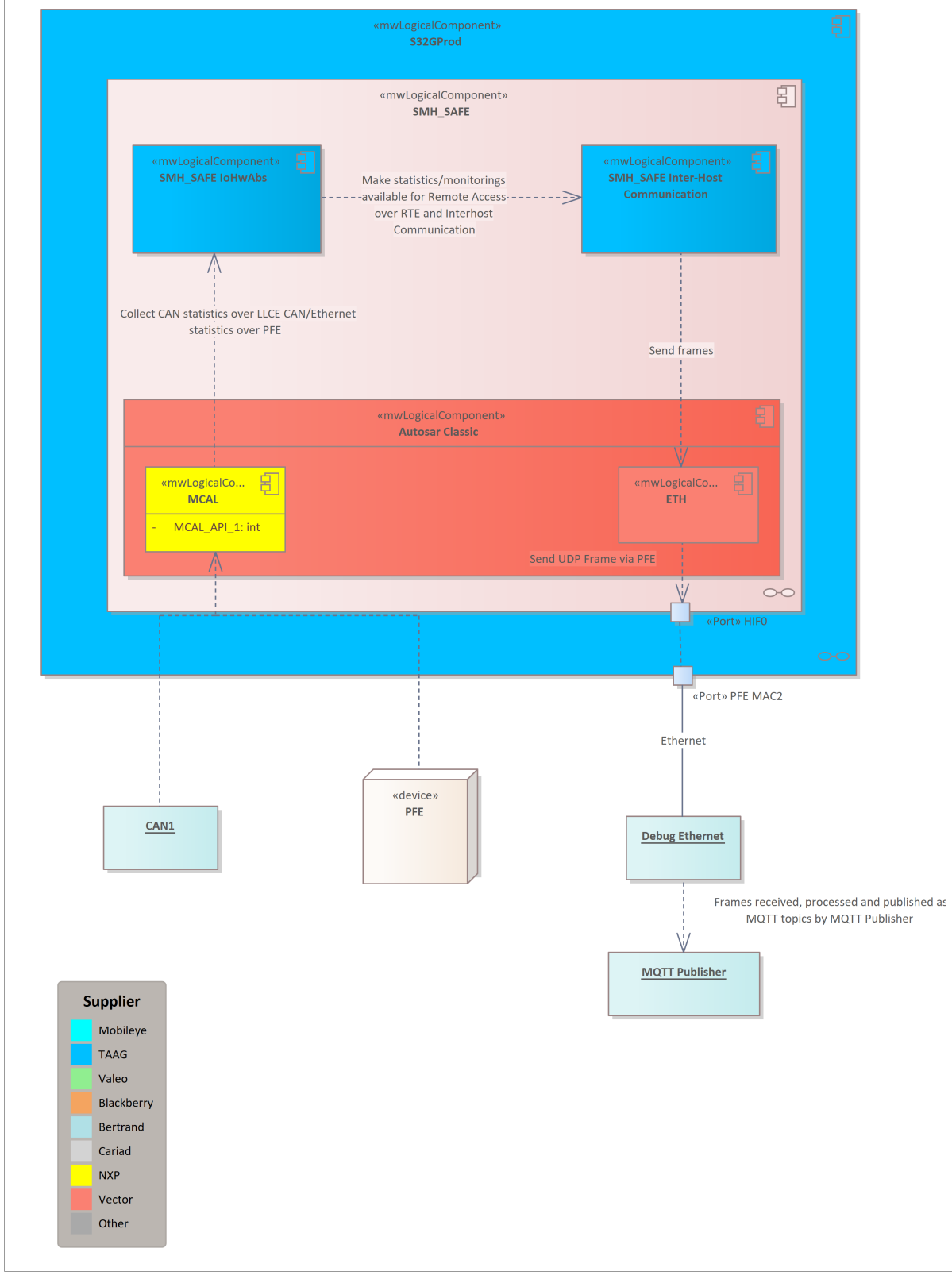
- Internal/External Voltage measurements
- Internal/External Current measurements
- Internal/External Temperature measurements

- Internal/External Humidity measurements
- Runtime Tests on Memory devices:
  - DRAM
  - EEPROM
  - eMMC
  - Flash
- External Communication Statistics:
  - Ethernet statistics for all traffic going through MCU(S32G)
    - Statistics to be collected from PFE level
  - CAN communication statistics for all active CAN transceivers

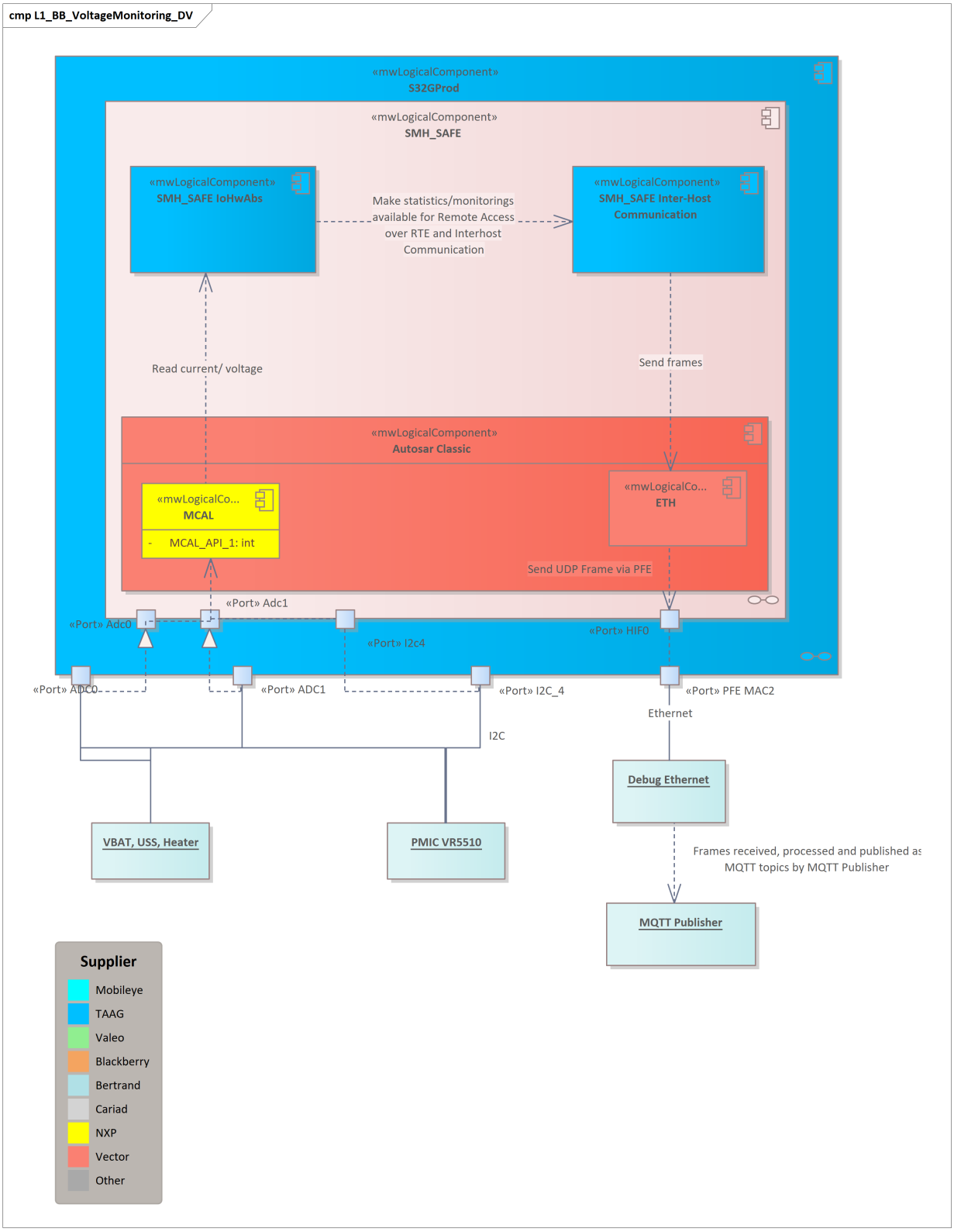
which are available to MCU(S32G).
Ddesign validation software implements, integrates, or configure tooling for publishing relevant test information over MQTT protocol **[**S32GPRODP-296744 **]**

5.2.14.1 Ethernet/CAN Statistics

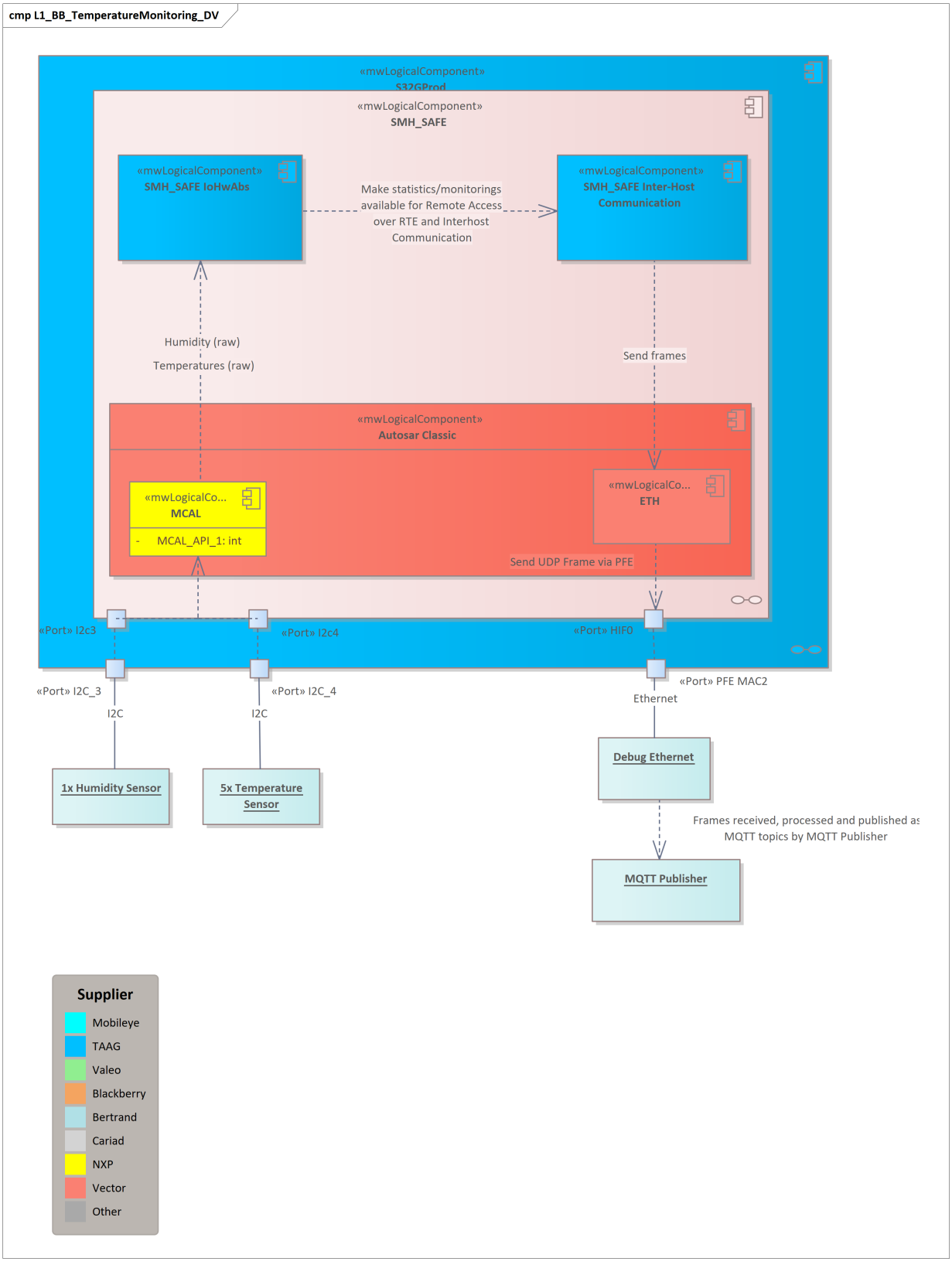
cmp L1_BB_CommunicationTrafficMonitoring_DV



5.2.14.2 Voltage/Current Monitoring

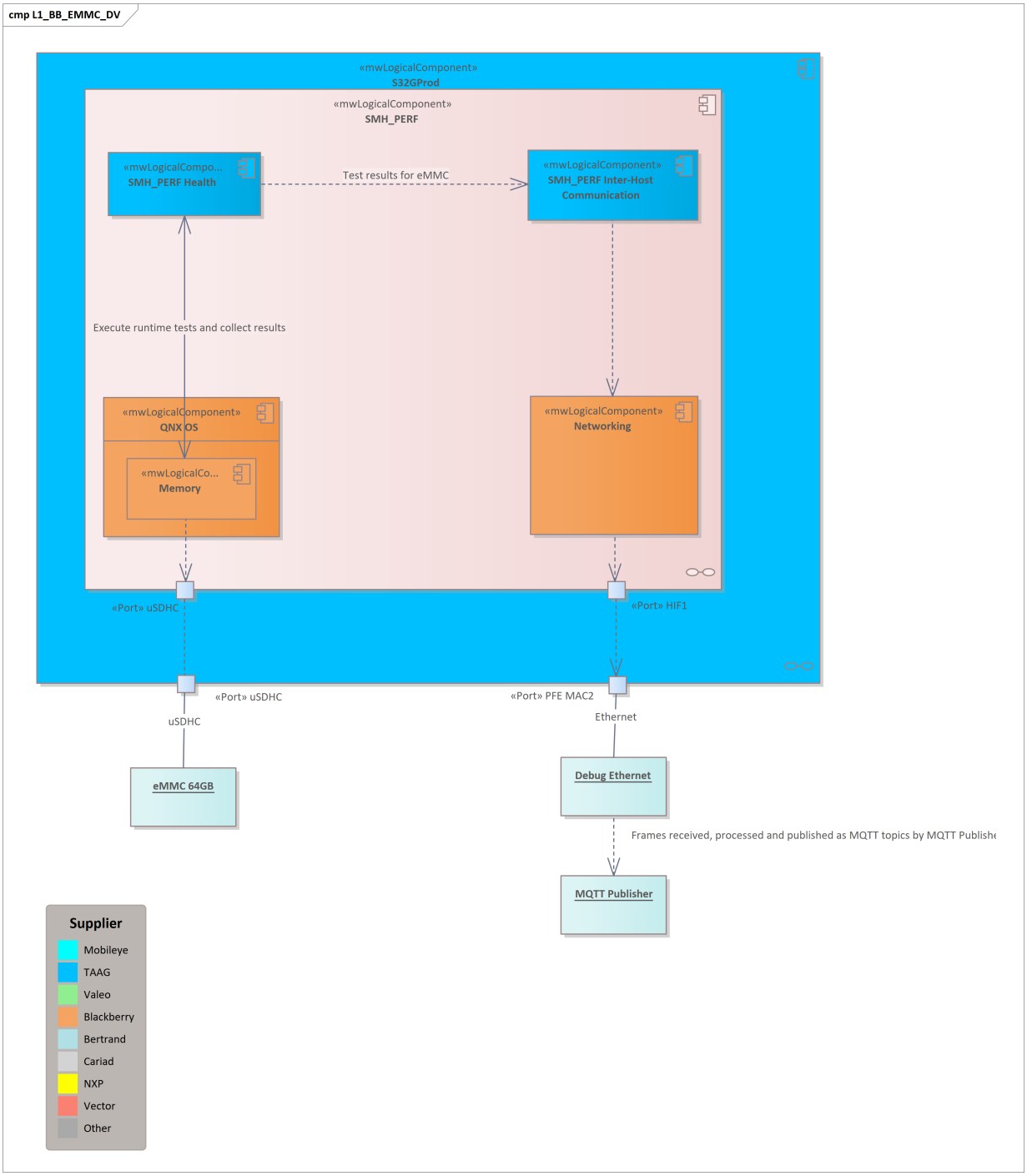


5.2.14.3 Temperature/Humidity Monitoring

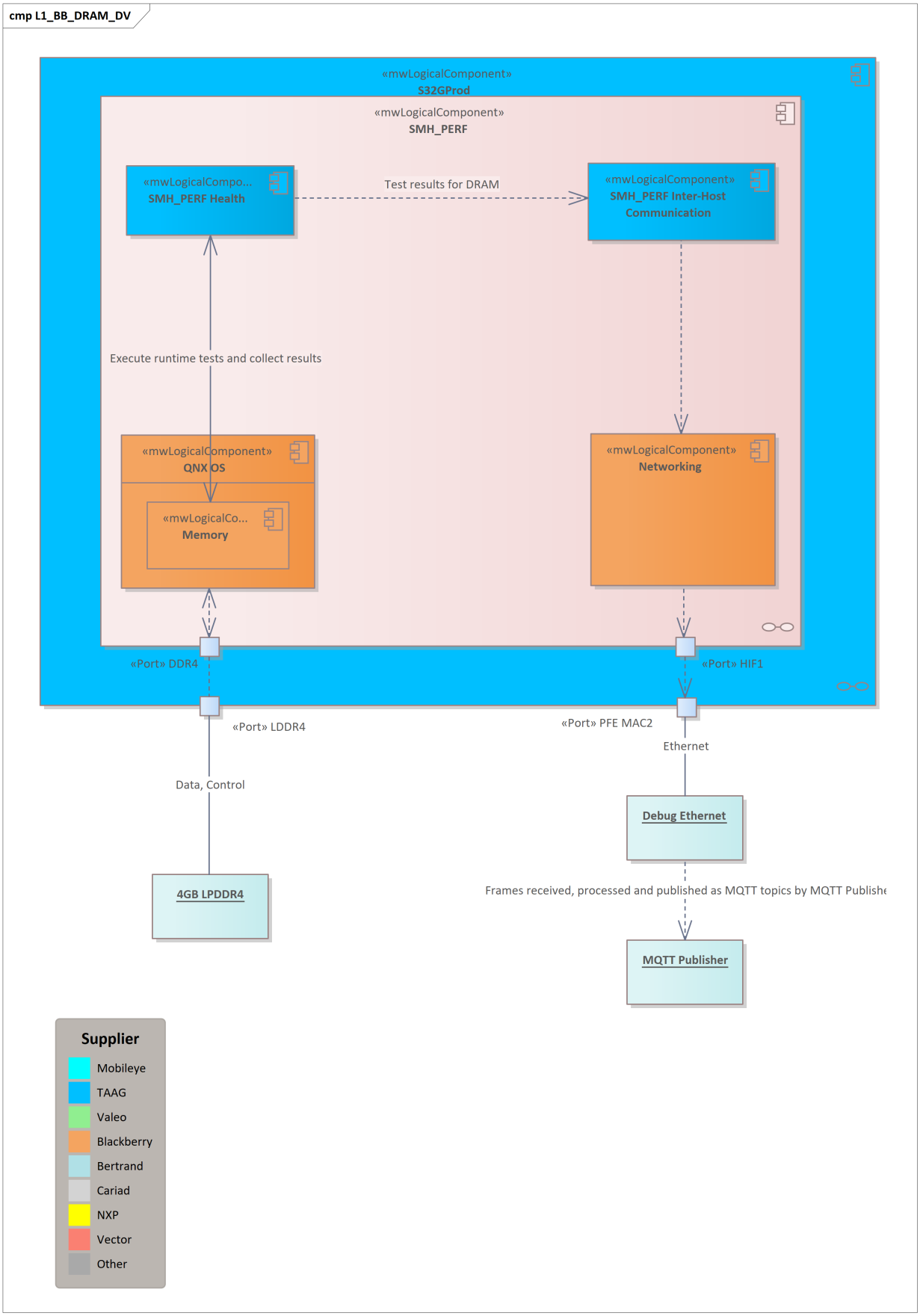


5.2.14.4 Memory Peripheral Tests

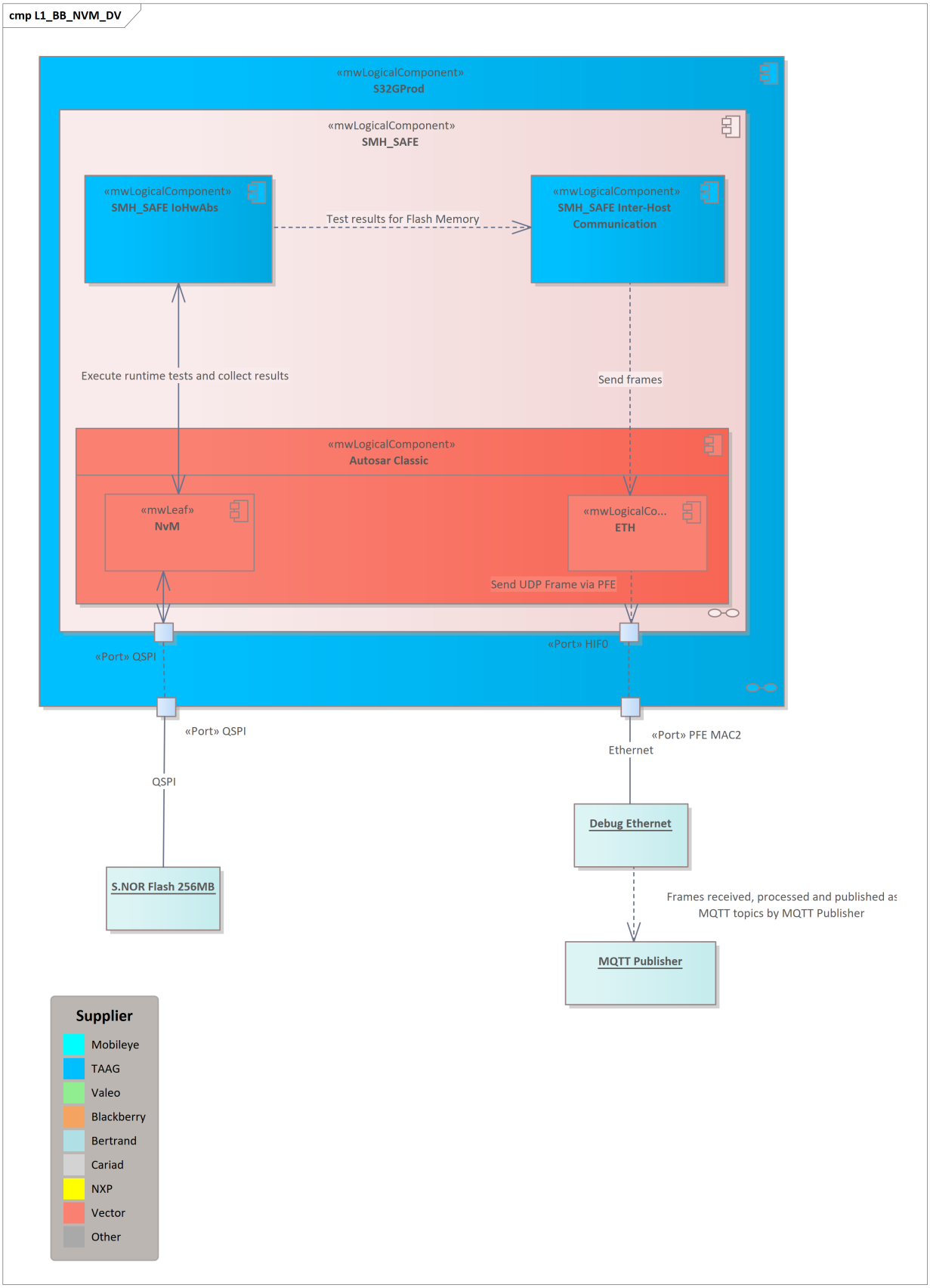
5.2.14.4.1 eMMC



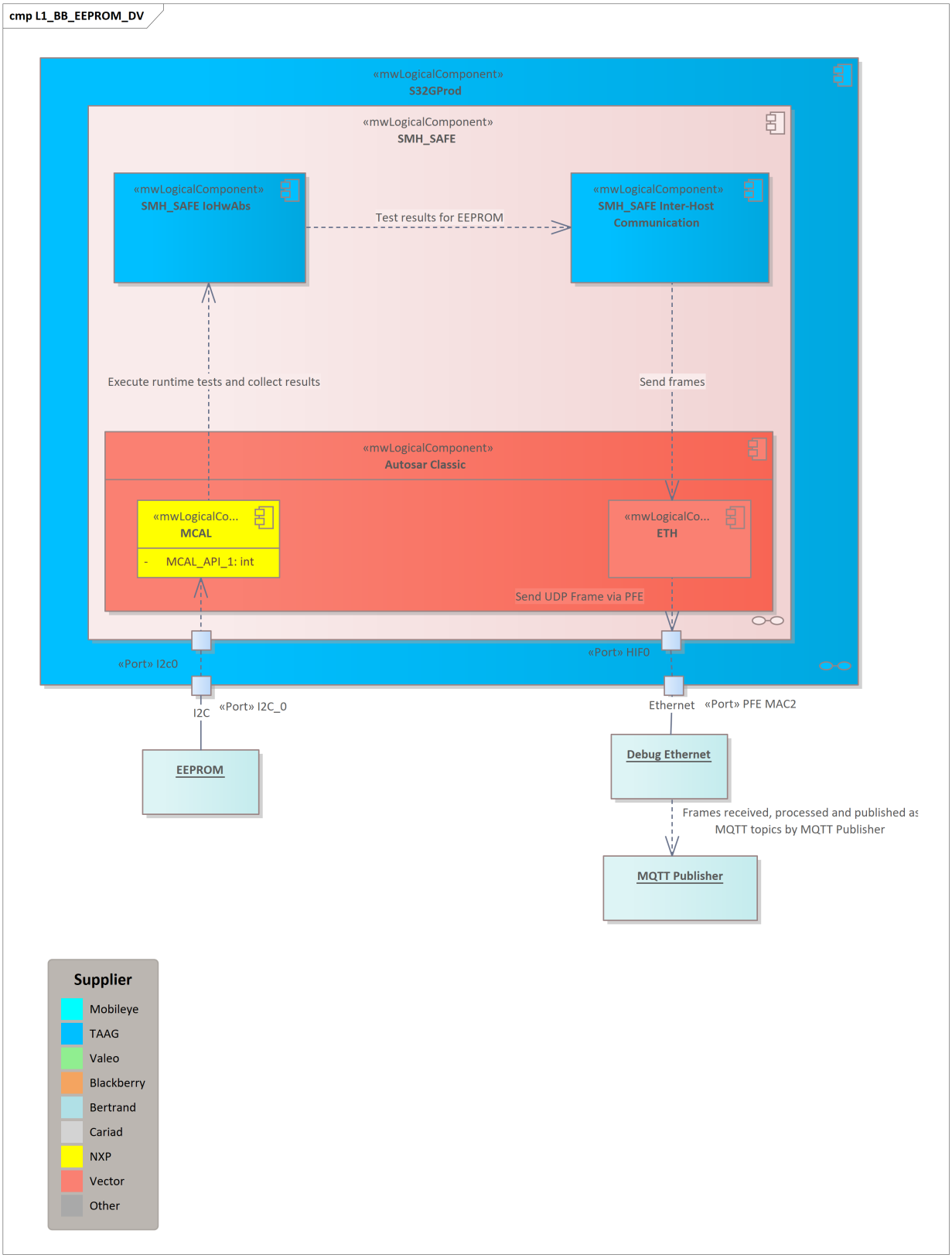
5.2.14.4.2 LPDDR4



5.2.14.4.3 Flash



5.2.14.4.4 EEPROM



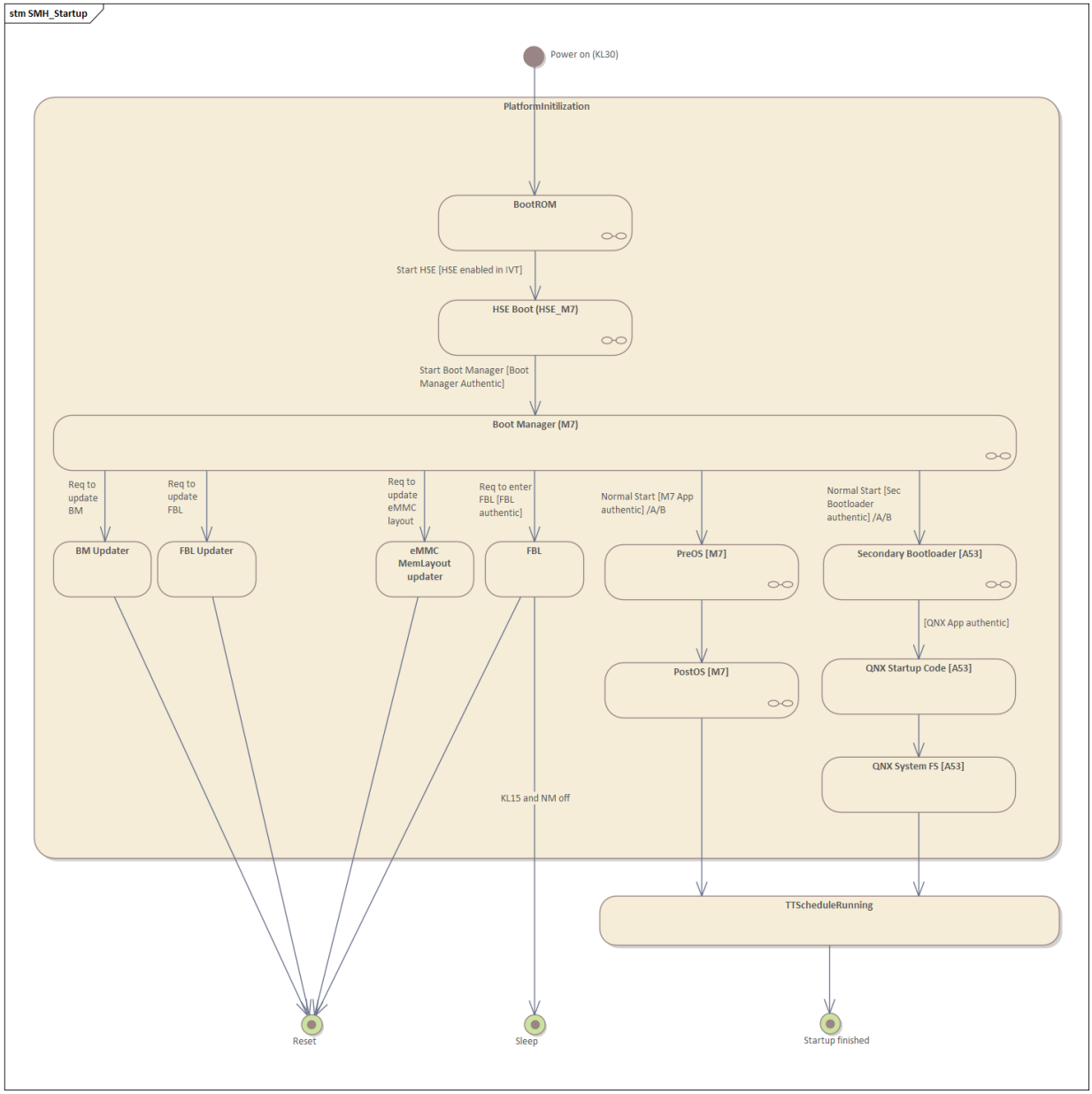
5.2.15 Application SWCs

Diagram below shows 3rd party Application SWCs which are deployed on S32G.

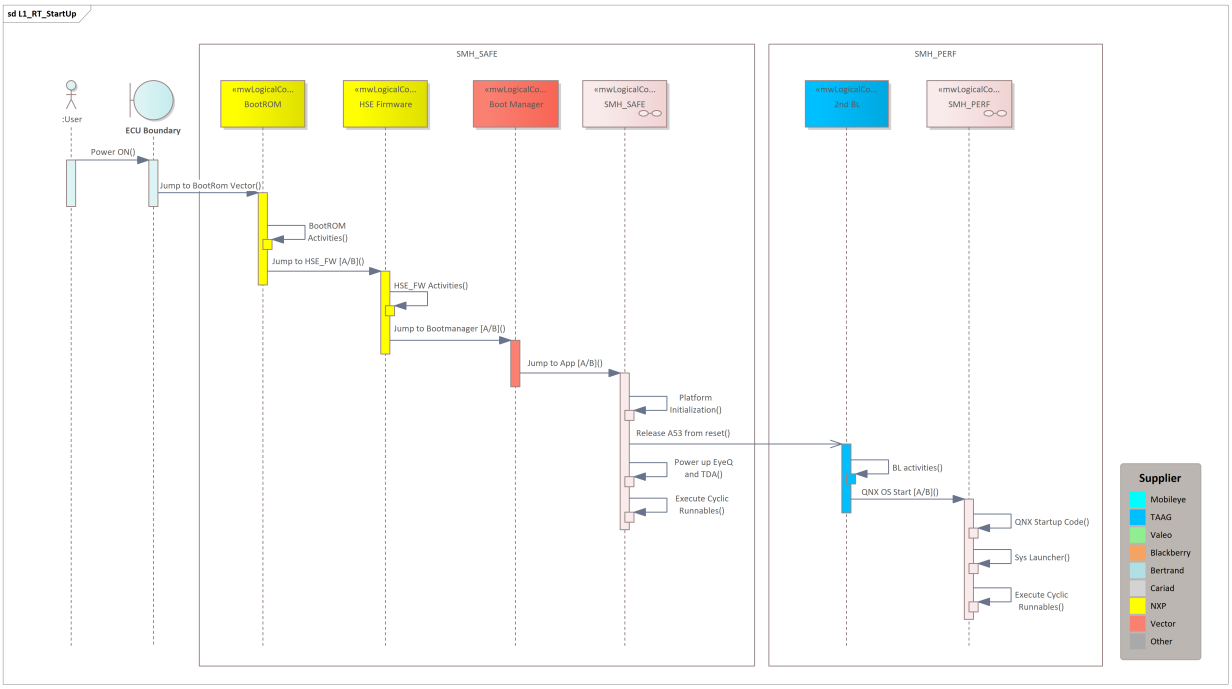
6 Runtime View

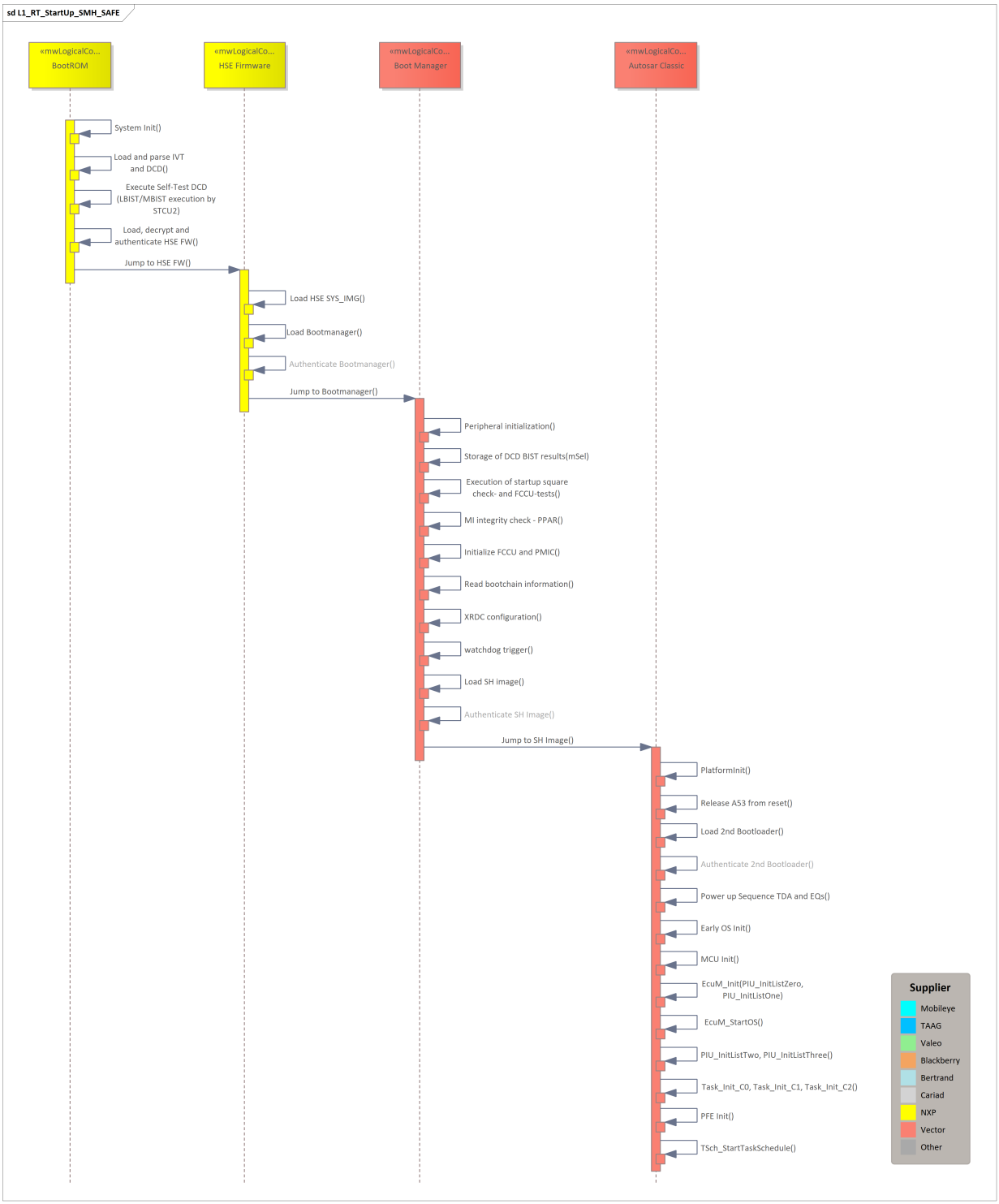
6.1 Overall ECU start-up

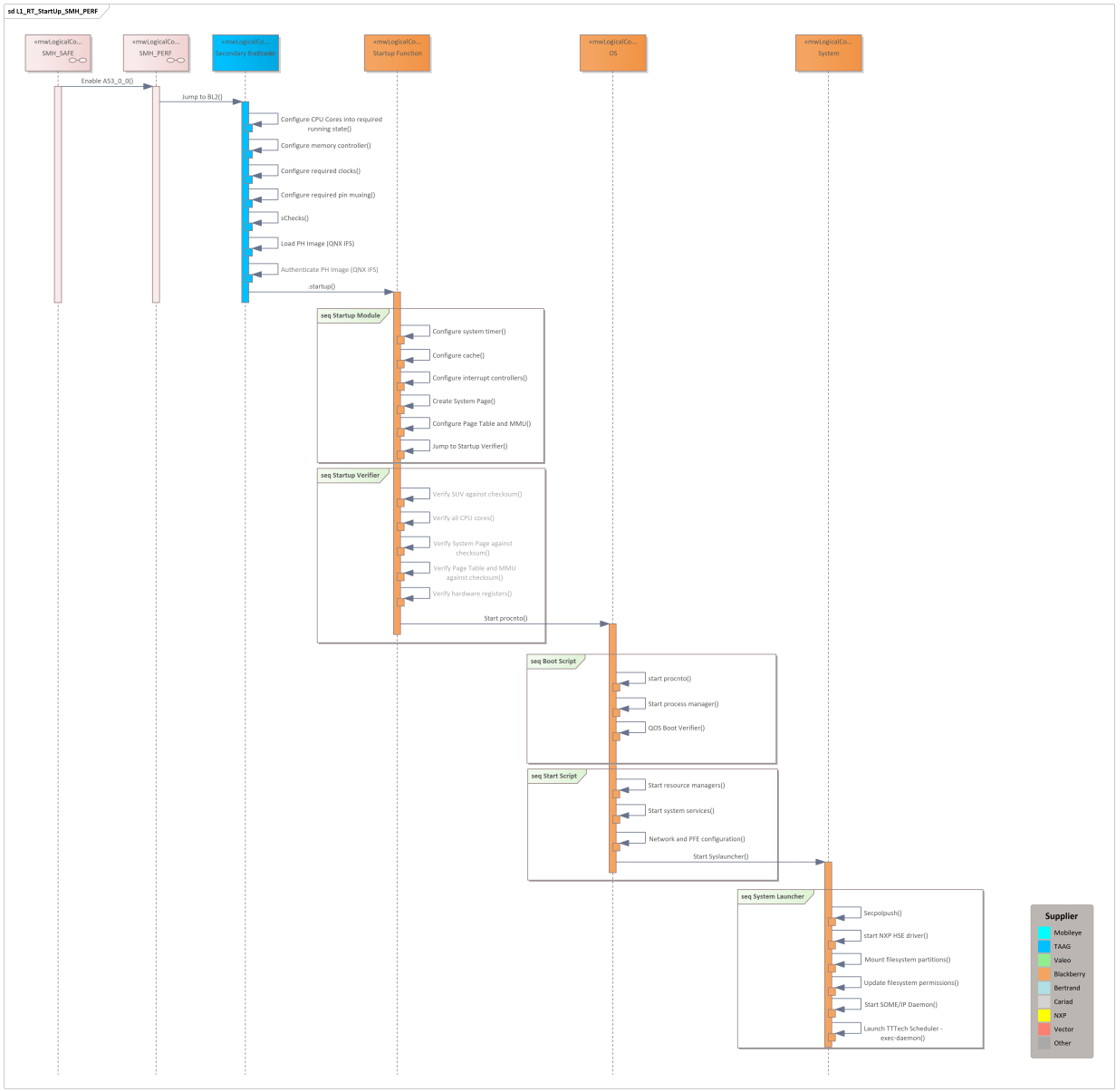
This view shows the default startup of the SMH_SAFE. The Boot Manager selects the boot path, either to the application on M7 and A53 cores, or it can boot into the flash bootloader (FBL) to update the ECU. [S3 2GPRODP-296370]



The start-up sequence diagram shows the sequence of initialization of deployable entities at the highest level, until the first runnable execution on Safety Host and Performance Host. [HCP2MEP-226356]


sd L1_RT_StartUp
SMH_SAFE
SMH_PERF
:User
ECU Boundary
«mwLogicalCo...
BootROM
«mwLogicalCo...
HSE Firmware
«mwLogicalCo...
Boot Manager
«mwLogicalCo...
SMH_SAFE
«mwLogicalCo...
2nd BL
«mwLogicalCo...
SMH_PERF
Power ON()
Jump to BootRom Vector()
BootROM Activities()
Jump to HSE_FW [A/B]()
HSE_FW Activities()
Jump to Bootmanager [A/B]()
Jump to App [A/B]()
Platform Initialization()
Release A53 from reset()
Power up EyeQ and TDA()
Execute Cyclic Runnables()
BL activities()
QNX OS Start [A/B]()
QNX Startup Code()
Sys Launcher()
Execute Cyclic Runnables()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other


sd L1_RT_StartUp_SMH_SAFE
«mwLogicalCo... BootROM
«mwLogicalCo... HSE Firmware
«mwLogicalCo... Boot Manager
«mwLogicalCo... Autosar Classic
System Init()
Load and parse IVT and DCD()
Execute Self-Test DCD (LBIST/MBIST execution by STCU2)
Load, decrypt and authenticate HSE FW()
Jump to HSE FW()
Load HSE SYS_IMG()
Load Bootmanager()
Authenticate Bootmanager()
Jump to Bootmanager()
Peripheral initialization()
Storage of DCD BIST results(mSel)
Execution of startup square check- and FCCU-tests()
MI integrity check - PPAR()
Initialize FCCU and PMIC()
Read bootchain information()
XRDC configuration()
watchdog trigger()
Load SH image()
Authenticate SH Image()
Jump to SH Image()
PlatformInit()
Release A53 from reset()
Load 2nd Bootloader()
Authenticate 2nd Bootloader()
Power up Sequence TDA and EQs()
Early OS Init()
MCU Init()
EcuM_Init(PIU_InitListZero, PIU_InitListOne)
EcuM_StartOS()
PIU_InitListTwo, PIU_InitListThree()
Task_Init_C0, Task_Init_C1, Task_Init_C2()
PFE Init()
TSch_StartTaskSchedule()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other



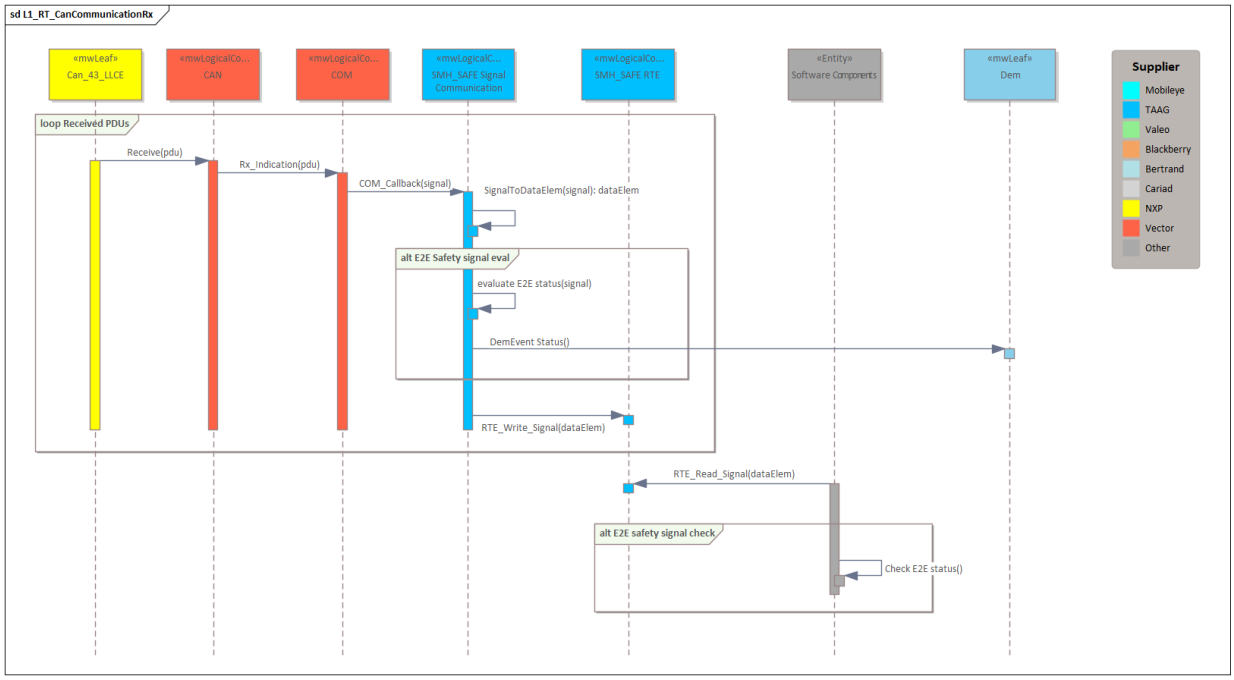
6.2 Runtime

6.2.1 Communication

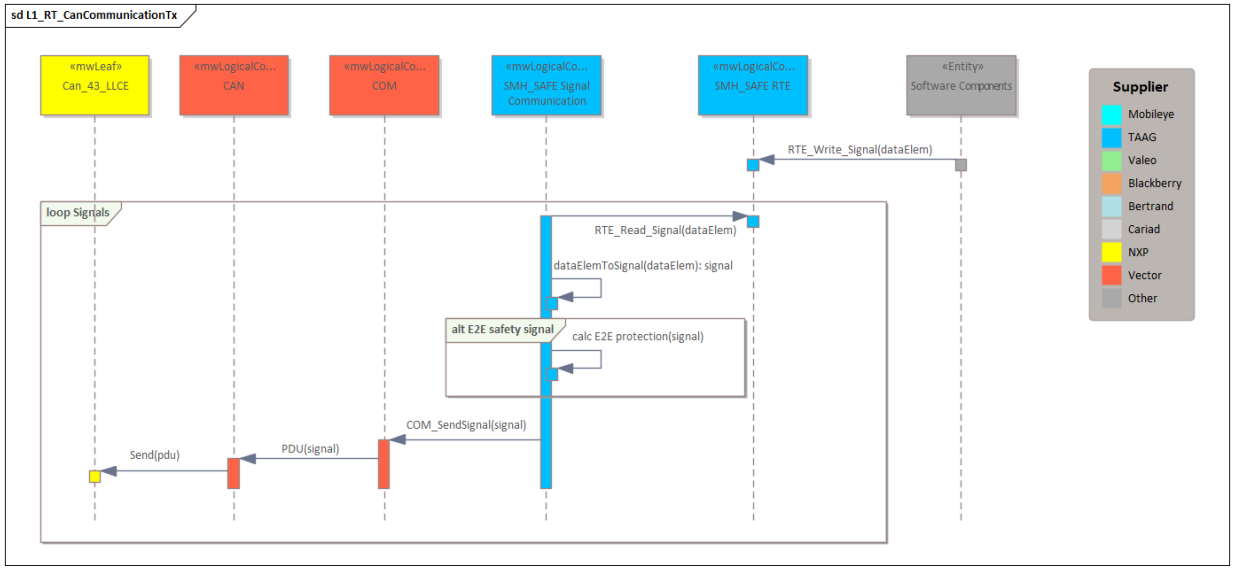
6.2.1.1 Inter-ECU Communication (Vehicle communication)

6.2.1.1.1 Signal oriented communication

6.2.1.1.1.1 CAN signal communication

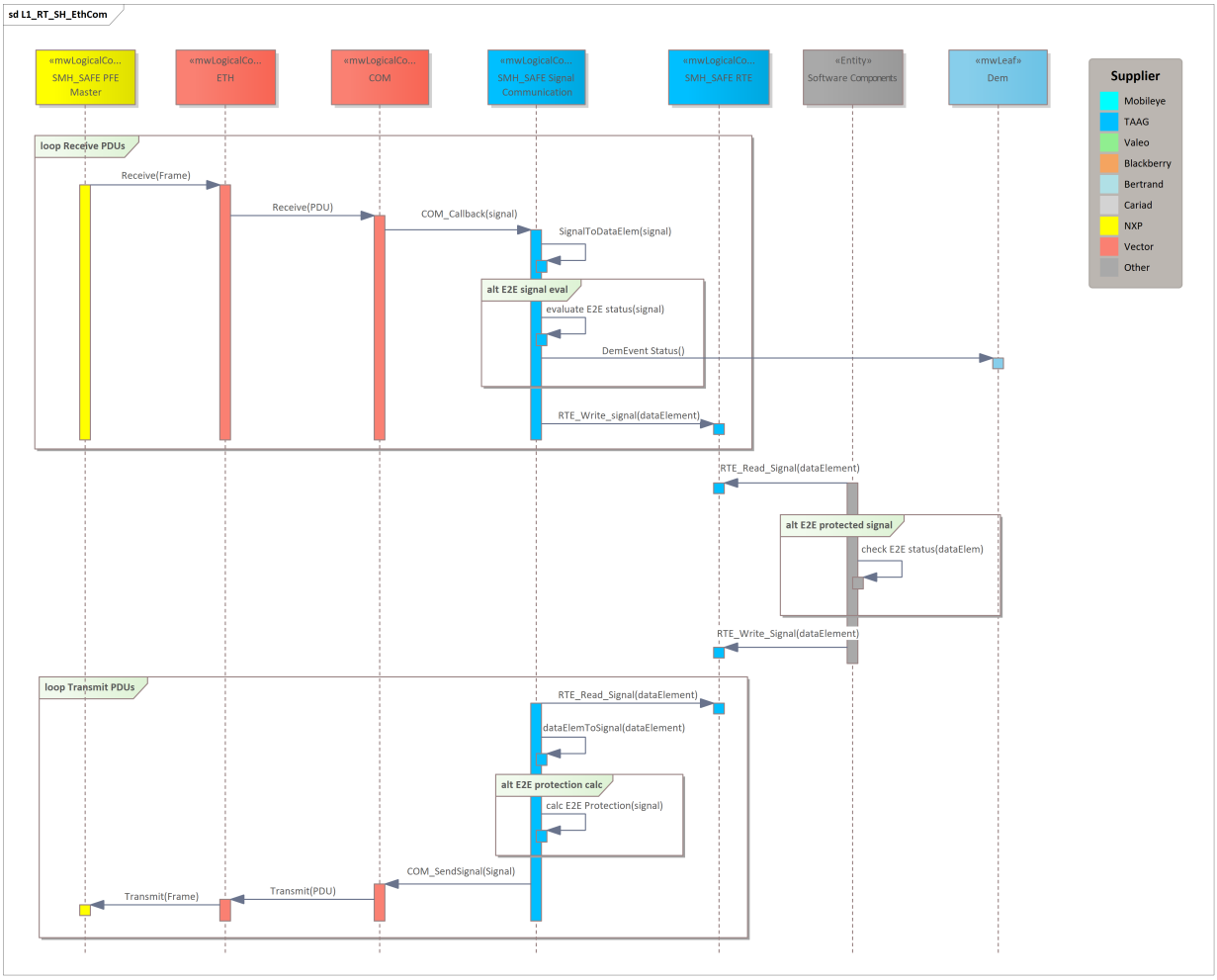


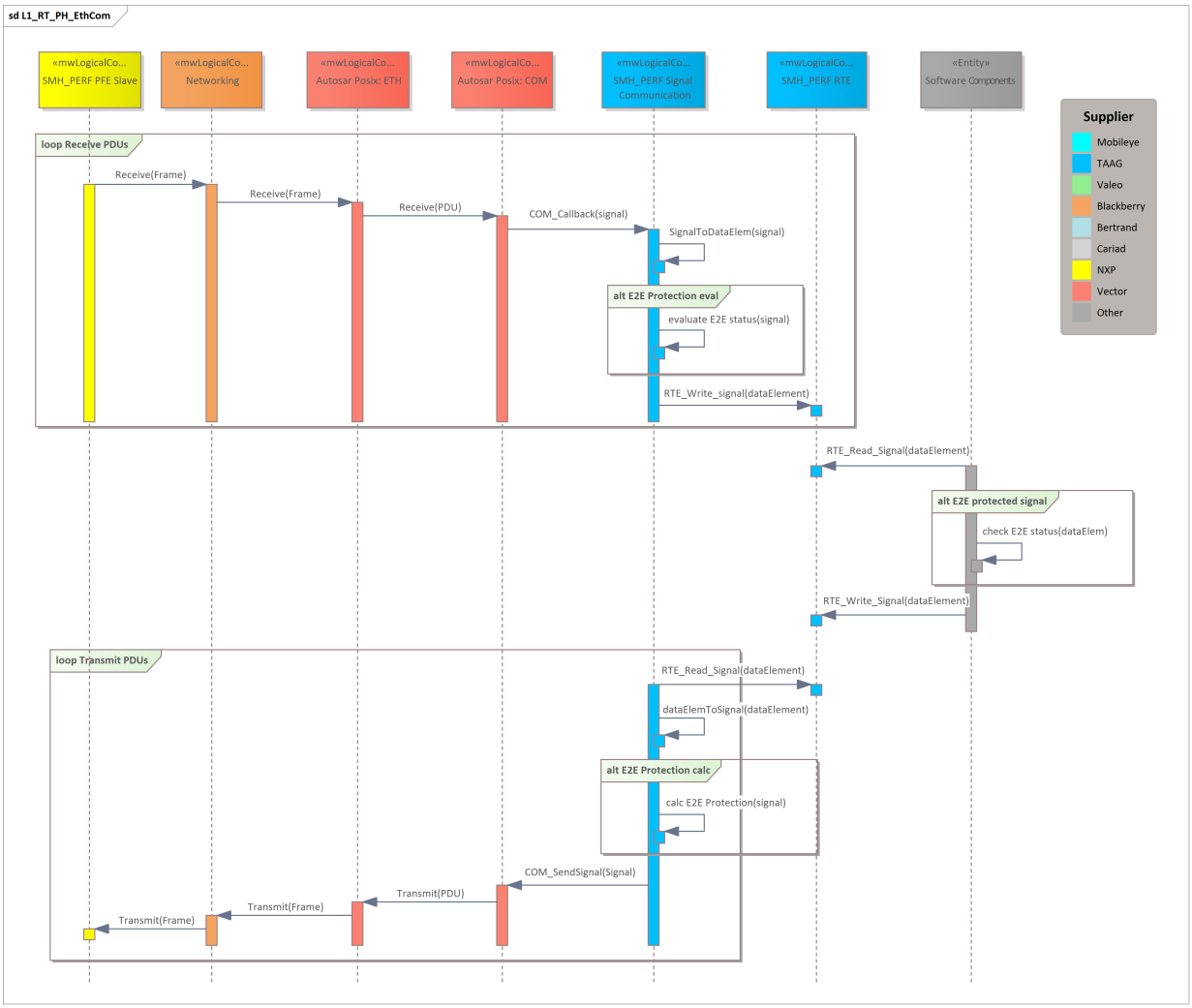
Signals are periodically read from the 3rd party COM BSW stack, transformed and written to RTE Data Elements structures in preallocated buffer pages. Application SWCs will get latest values through RTE API explicit access to same buffer page with zero copy operation. **[**S32GPRODP-296250 **]**



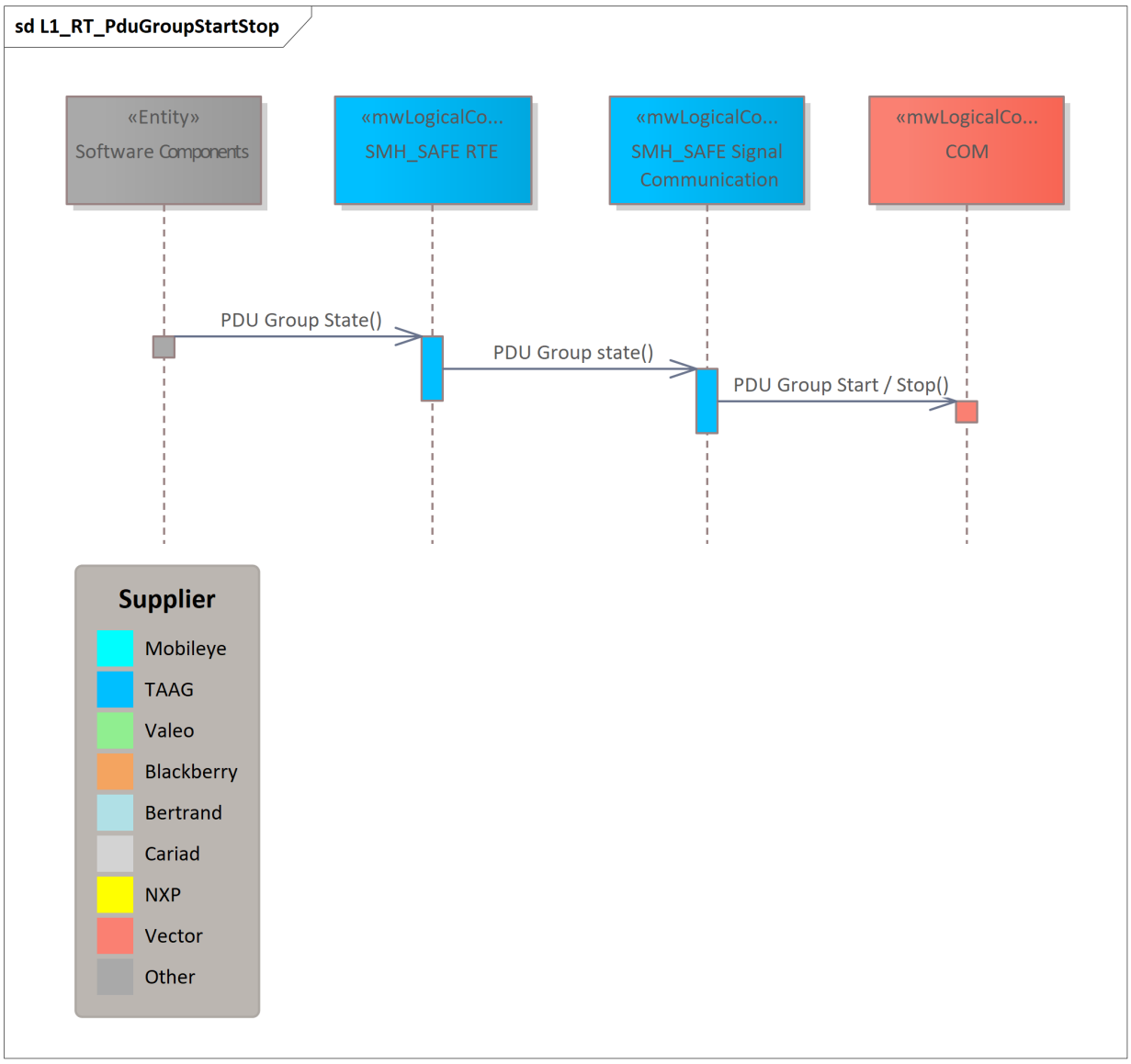
Application SWC writes Data Element structures bind to a bus Signal, using RTE API. Signal communication performs transformations to bus Signal and provides values to 3rd Party COM stack. **[**S32GPRODP-296252 **]**

6.2.1.1.1.2 Ethernet signal communication





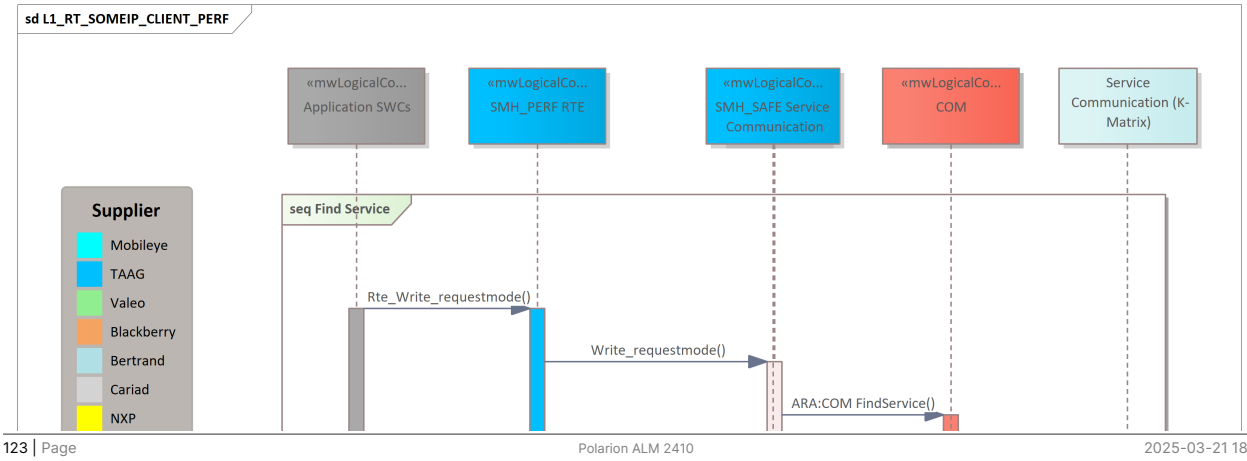
6.2.1.1.1.3 PDU groups enabling/disabling

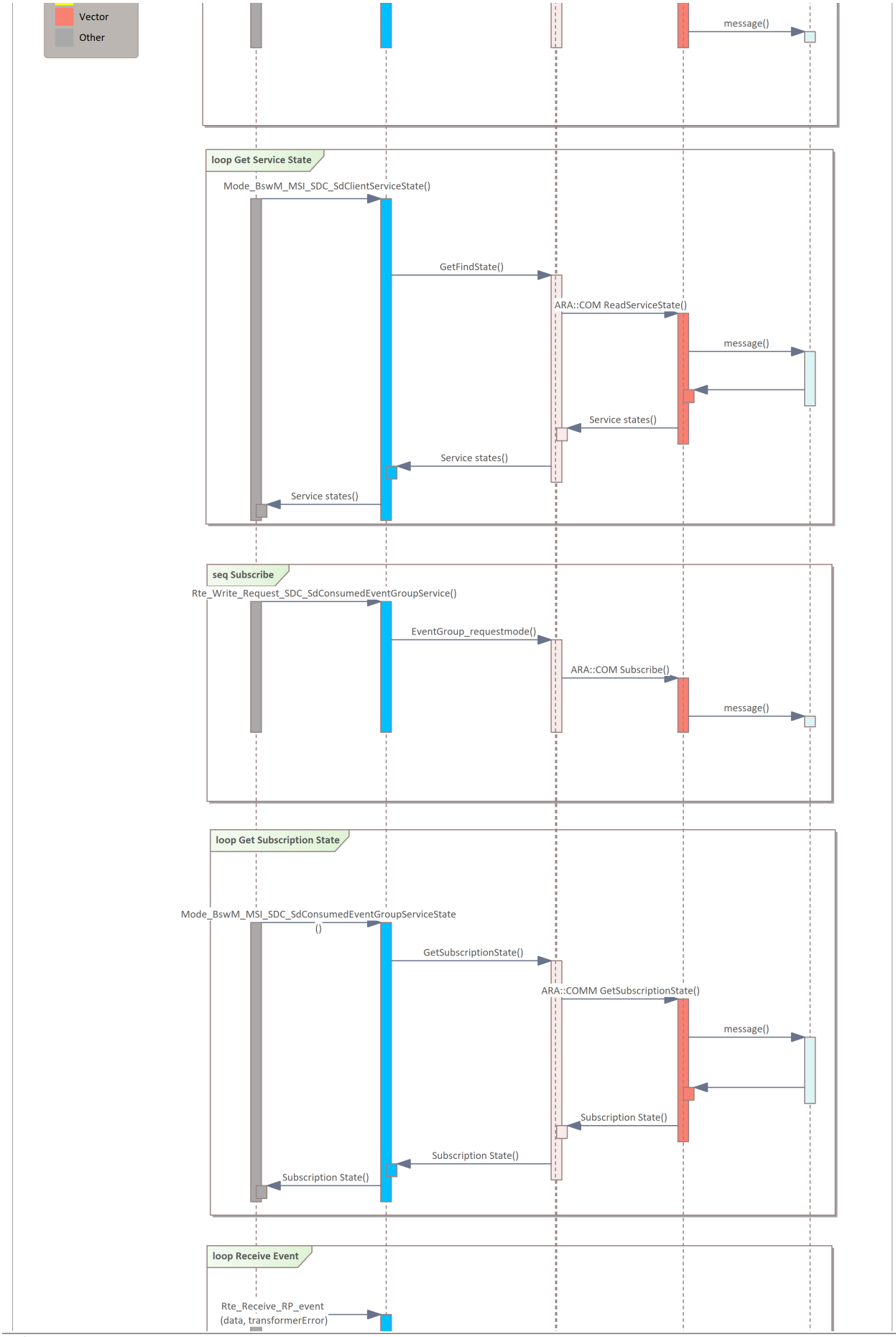


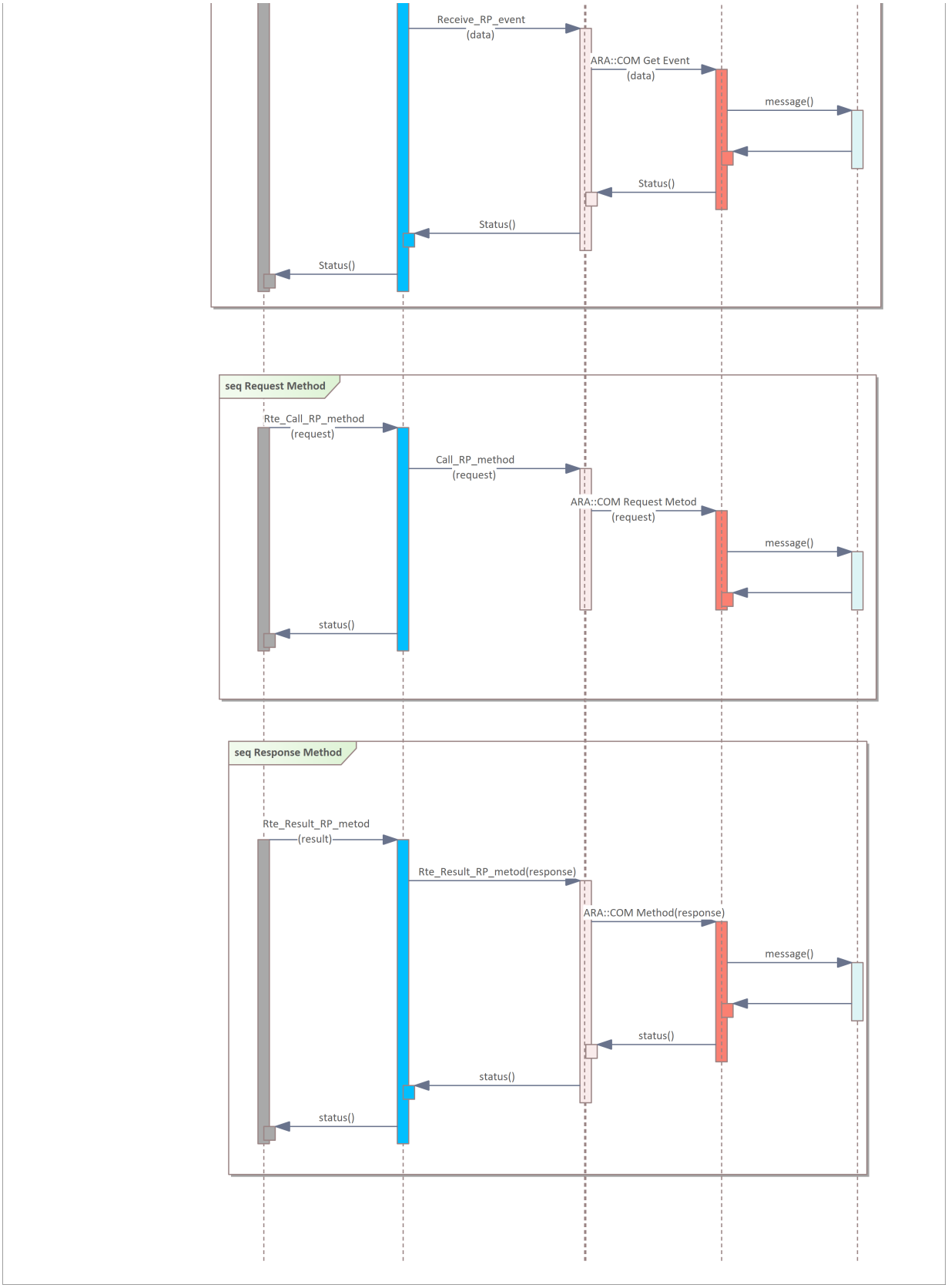
Application SWC can control transmission of defined PDU groups through particular PDU Group RTE API interface. [S32GPRODP-296310]

6.2.1.1.2 Service oriented communication

6.2.1.1.2.1 Client services - PH




Vector
Other
message()
loop Get Service State
Mode_BswM_MSI_SDC_SdClientServiceState()
GetFindState()
ARA::COM ReadServiceState()
message()
Service states()
Service states()
Service states()
seq Subscribe
Rte_Write_Request_SDC_SdConsumedEventGroupService()
EventGroup_requestmode()
ARA::COM Subscribe()
message()
loop Get Subscription State
Mode_BswM_MSI_SDC_SdConsumedEventGroupServiceState
()
GetSubscriptionState()
ARA::COMM GetSubscriptionState()
message()
Subscription State()
Subscription State()
Subscription State()
loop Receive Event
Rte_Receive_RP_event
(data, transformerError)



The Adaptive AUTOSAR provides method FindService to find service instances. FindService function is mapped to Rte requestMode function which can be used from Application Layer. **[**S32GPRODP-296867 **]**

ReadServiceState function is mapped to RTE Mode_BswM_MSI_SDC_SdClientServiceState. This method provide

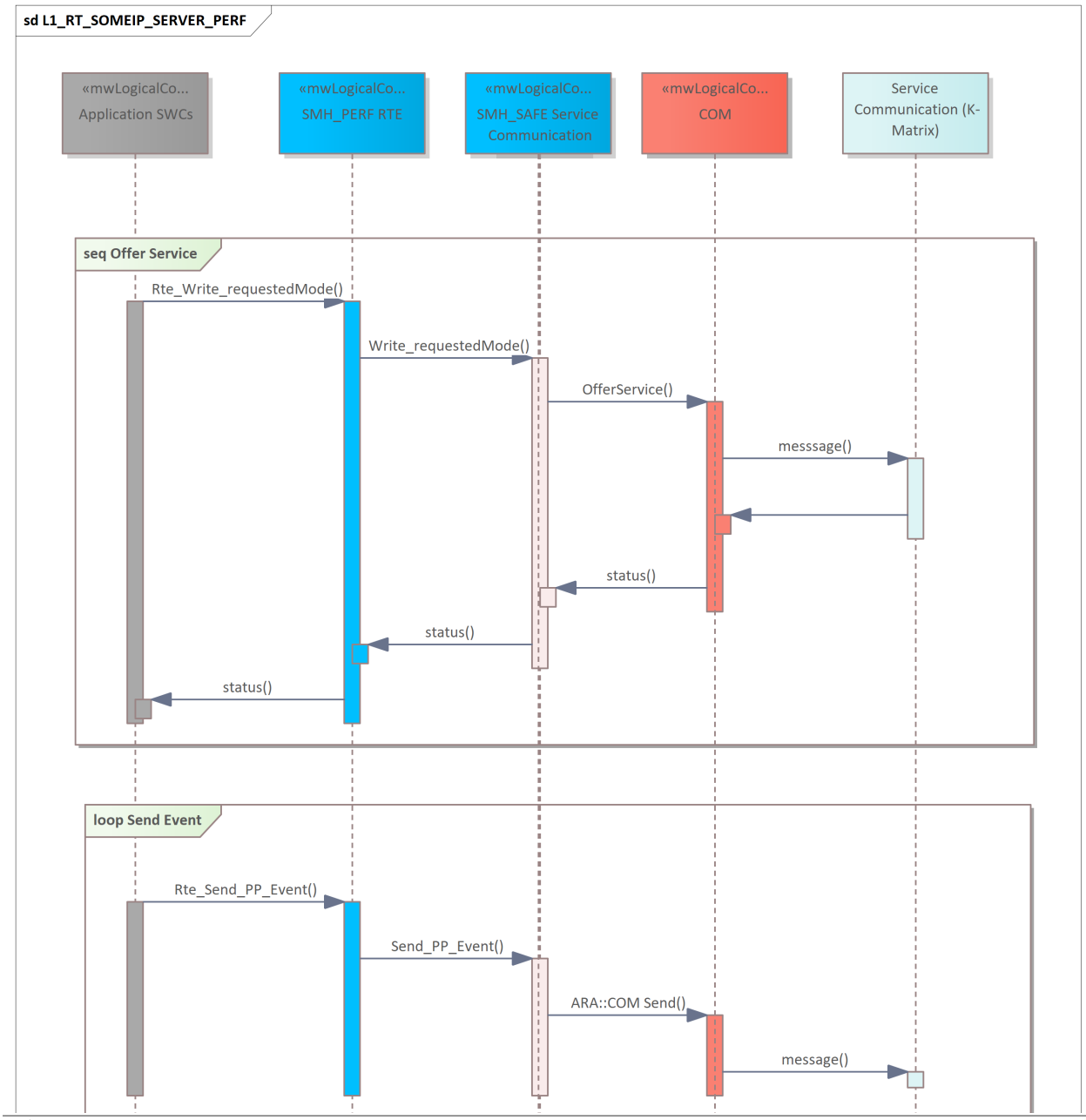
support for polling the service state of an offered service on client side. [S32GPRODP-296863]

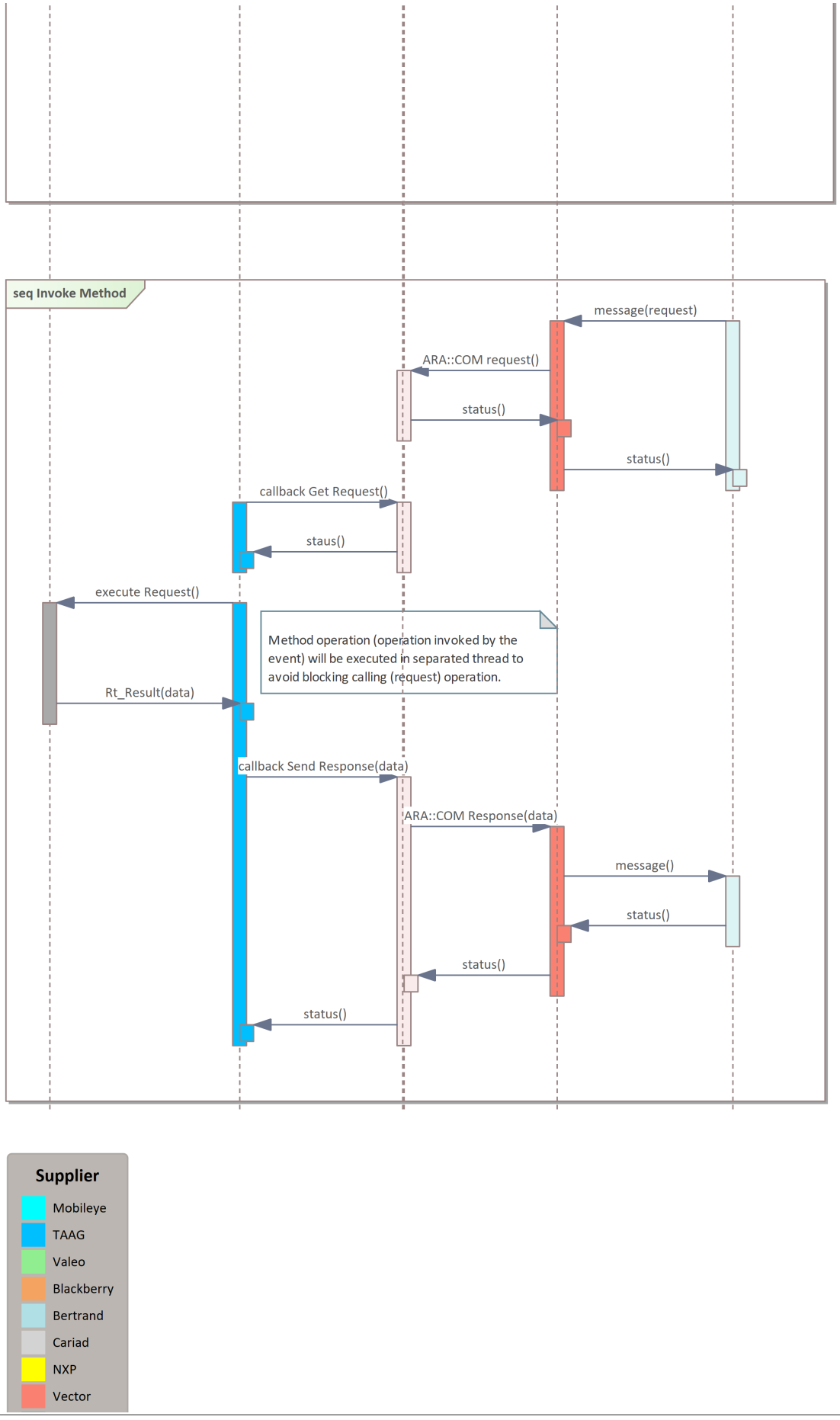
When a client application has connected to a server, it can subscribe to events that is offered by the server. RTE SdConsumedEventGroupService call is used for subscription to the events from the Application level. This RTE call is mapped to Subscribe() Adaptive AUTOSAR. [S32GPRODP-296864]

The call to the Subscribe() method is asynchronous call. For the monitoring of subscription state Adaptive AUTOSAR provide GetSubscriptionState metod. This method is mapped to the RTE Mode_BswM_MSI_SDC_SdConsumedEventGroupServiceState which is used from Application layer. [S32GPRODP-296865]

Corresponding RTE calls are used for server event receiving and server-side method invocations. Polling mechanism from the Application Layer is used for events. For the methods request-response mechanism is used. [S32GPRODP-296866]

6.2.1.1.2.2 Server Services - PH



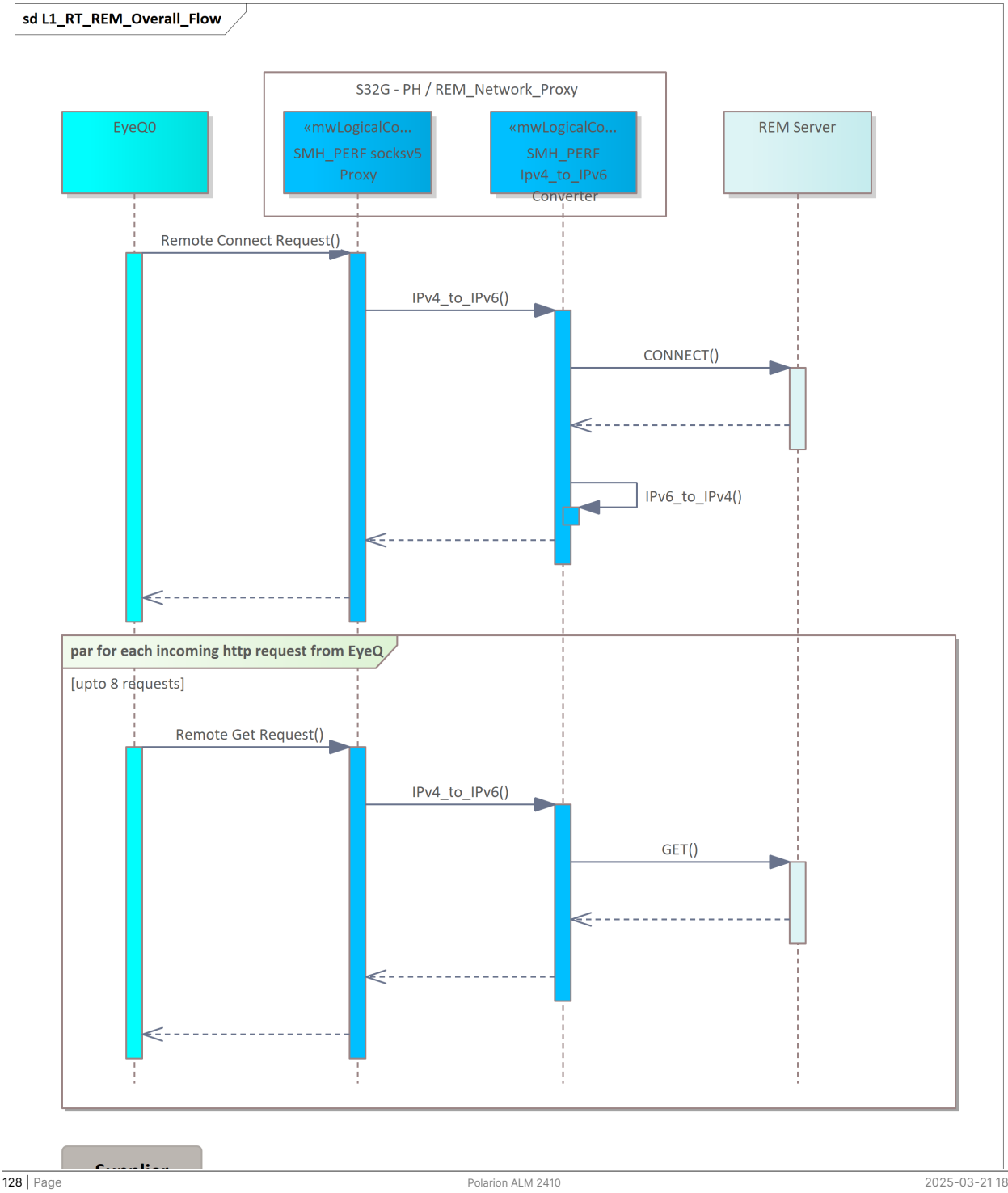

seq Invoke Method
message(request)
ARA::COM request()
status()
status()
callback Get Request()
staus()
execute Request()
Method operation (operation invoked by the event) will be executed in separated thread to avoid blocking calling (request) operation.
Rt_Result(data)
callback Send Response(data)
ARA::COM Response(data)
message()
status()
status()
status()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector

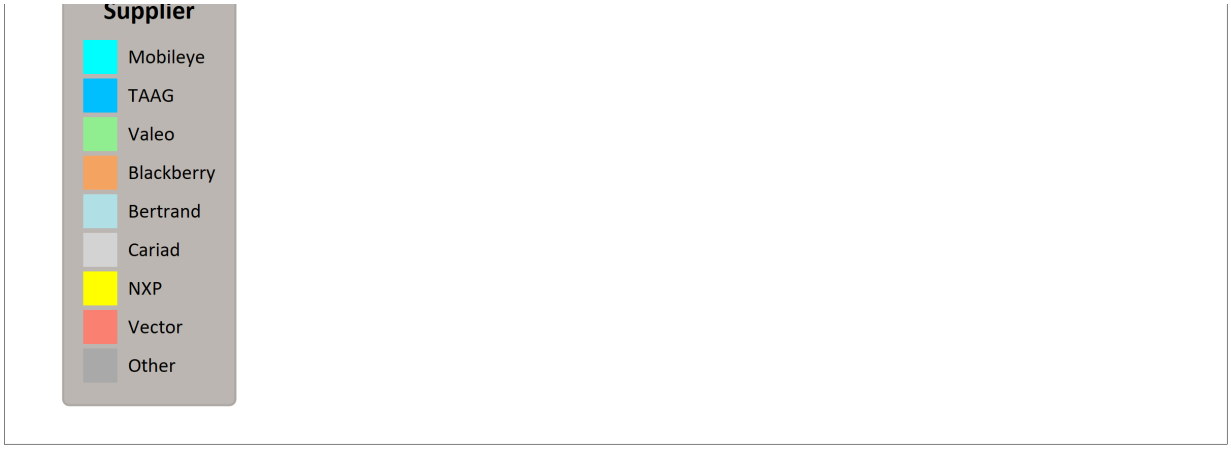
Other

The Adaptive Autosar provides the method OfferService() for Offering Service instance. Application layer of service provider side (server) need to instantiate RTE requestMode() method call which is mapped to OfferService() for Offering Service instance. [S32GPRODP-296856]

After RTE requestMode() method call server can serve requests (method calls) and provide events to subscribing consumers. [S32GPRODP-296855]

6.2.1.1.3 REM Network Proxy

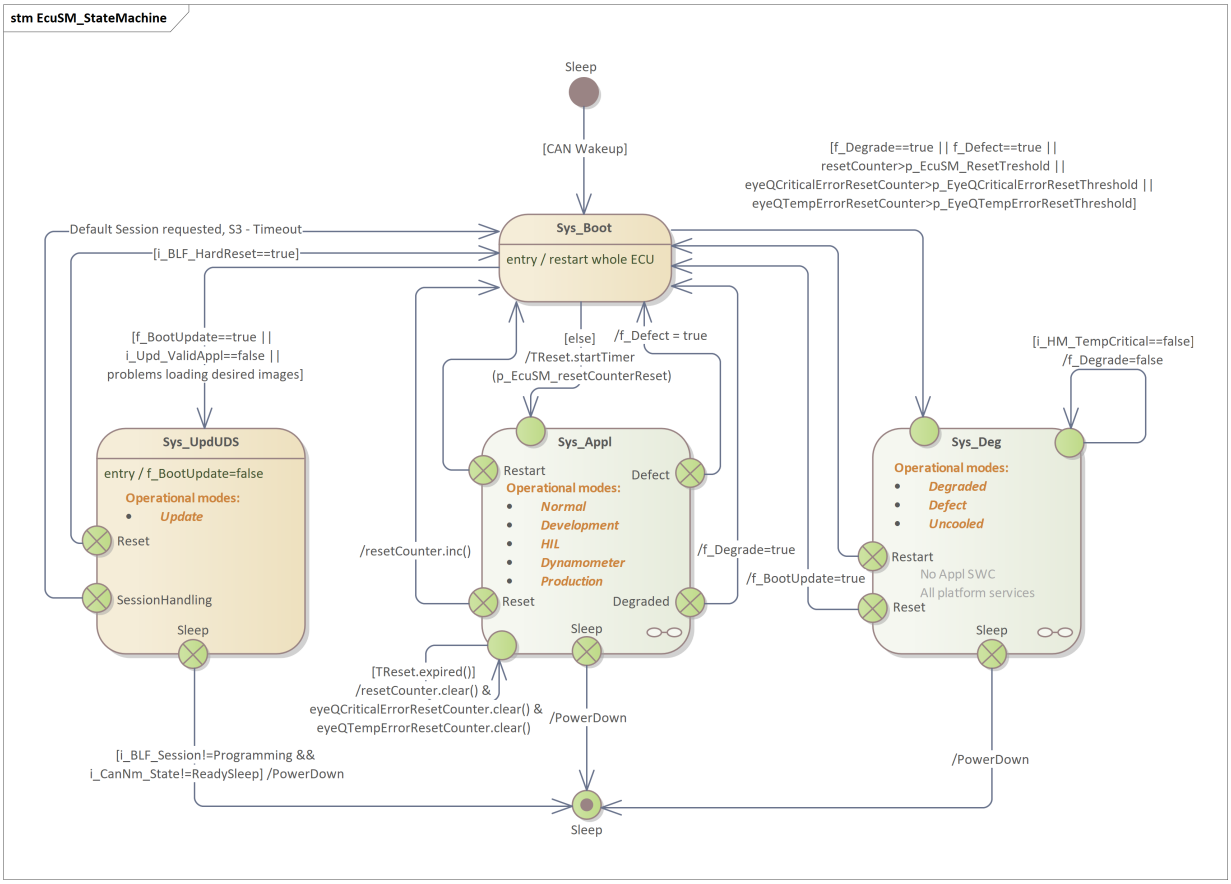




REM_Network_proxy has the above runtime behaviour. The proxy translates IPv4 frames from EyeQ to IPv6 frames towards the REM cloud server and vice-versa. [S32GPRODP-287703]

6.2.2 ECU state management

6.2.2.1 System state machine



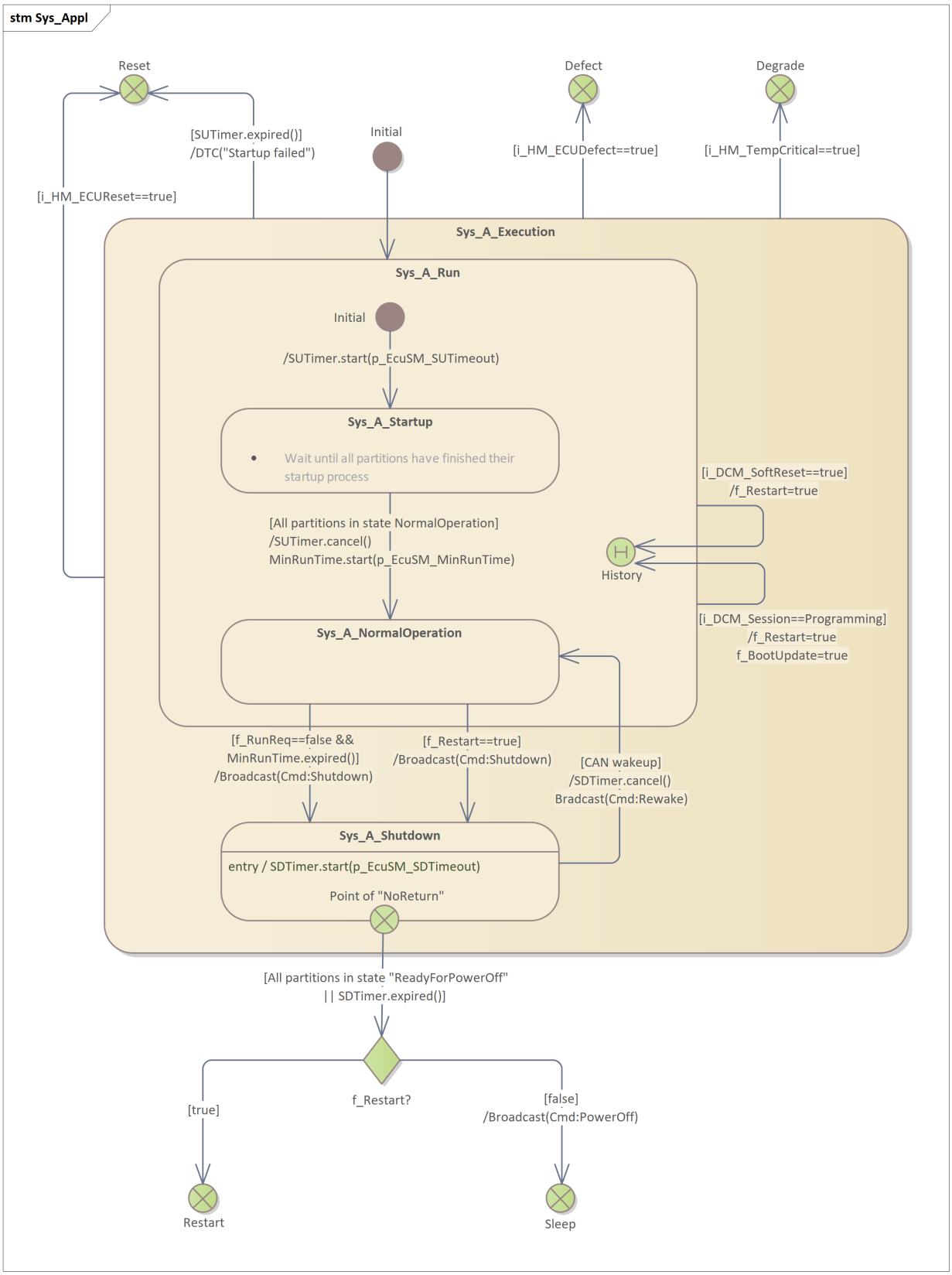
This view shows the System State Machine with its basic system modes:

- UDS Update (Sys_UpdUDS) - system shall start, run and shutdown only software that is required to support flashing over UDS protocol
- Application (Sys_Appl) - system shall start, run and shutdown all Application SWCs

- Degraded (Sys_Deg) - system shall start, run and shutdown only Platform SWCs and not run any Application SWCs

In Sys_Boot state shall be decided which system state the ECU shall boot to. It is an intermediate state, not externally visible. [S32GPRODP-296245]

6.2.2.1.1 Application mode state machine

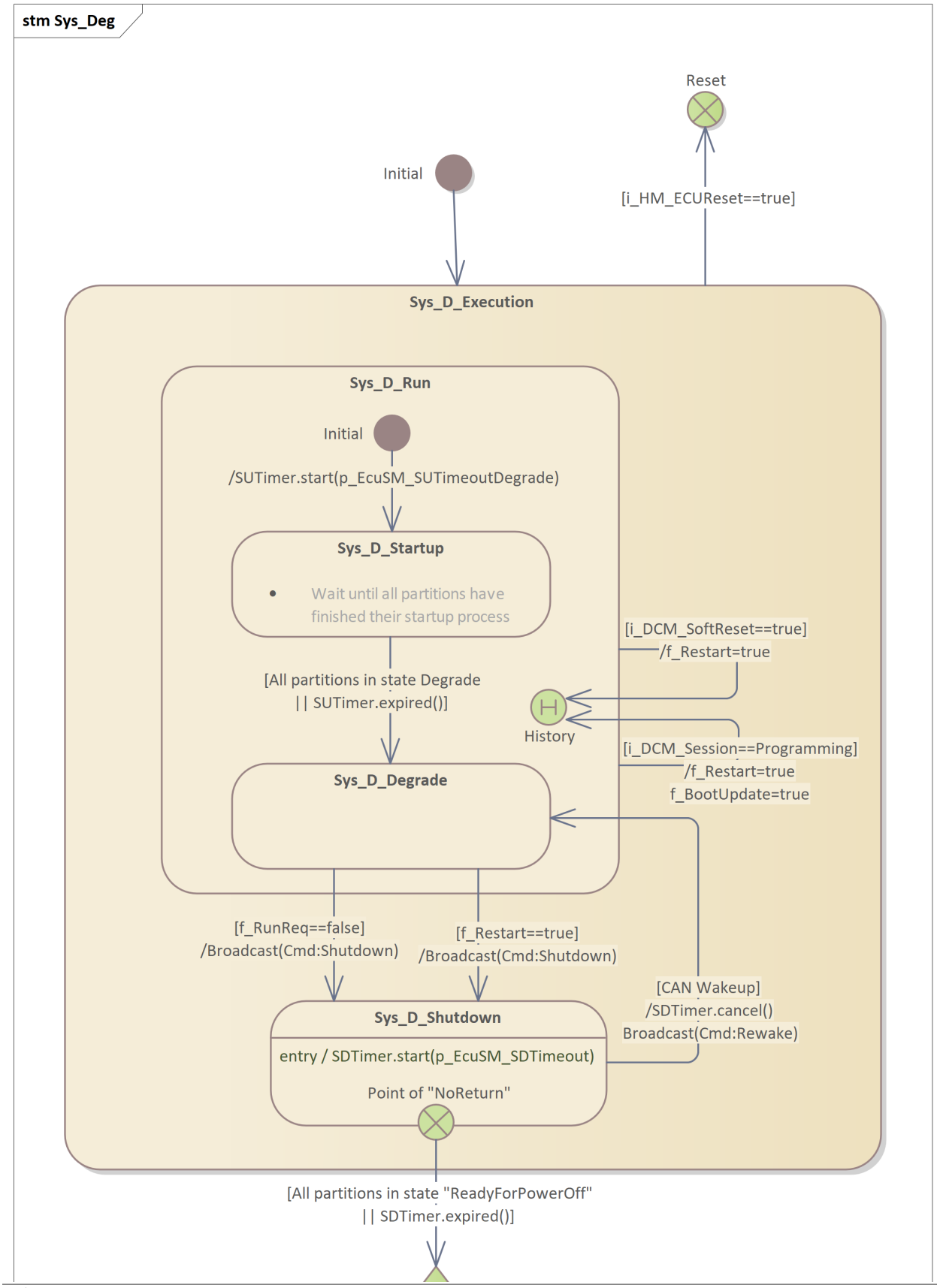


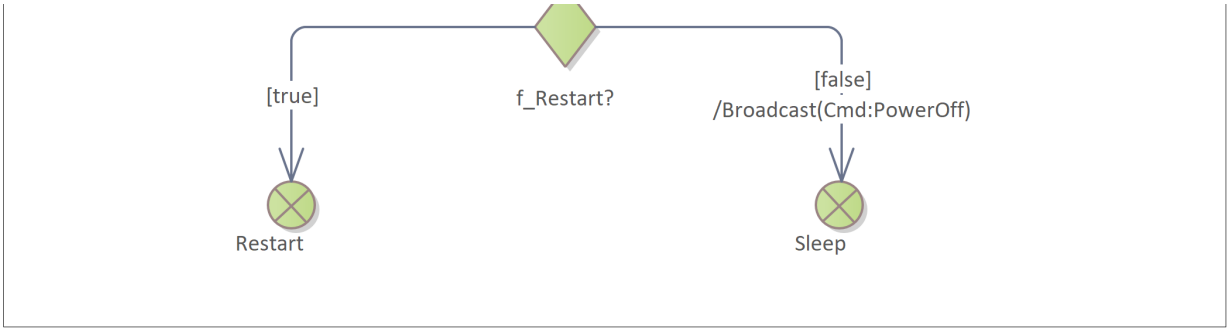
System Application Mode has the following states:

- Sys_A_Startup
- Sys_A_NormalOperation
- Sys_A_Shutdown

[S32GPRODP-296895]

6.2.2.1.2 Degraded mode state machine



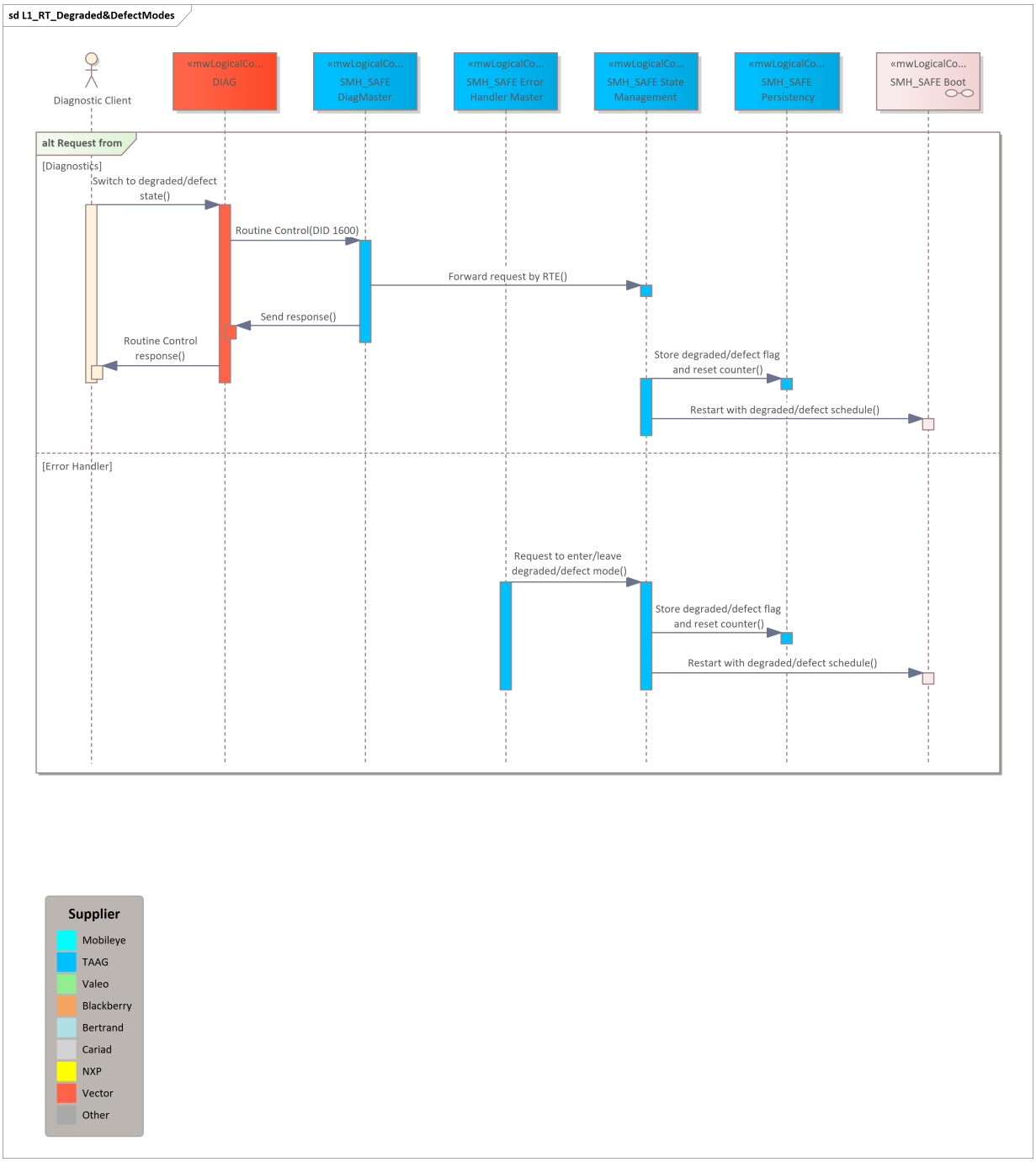


System Degraded Mode has the following states:

- Sys_D_Startup
- Sys_D_Degrade
- Sys_D_Shutdown

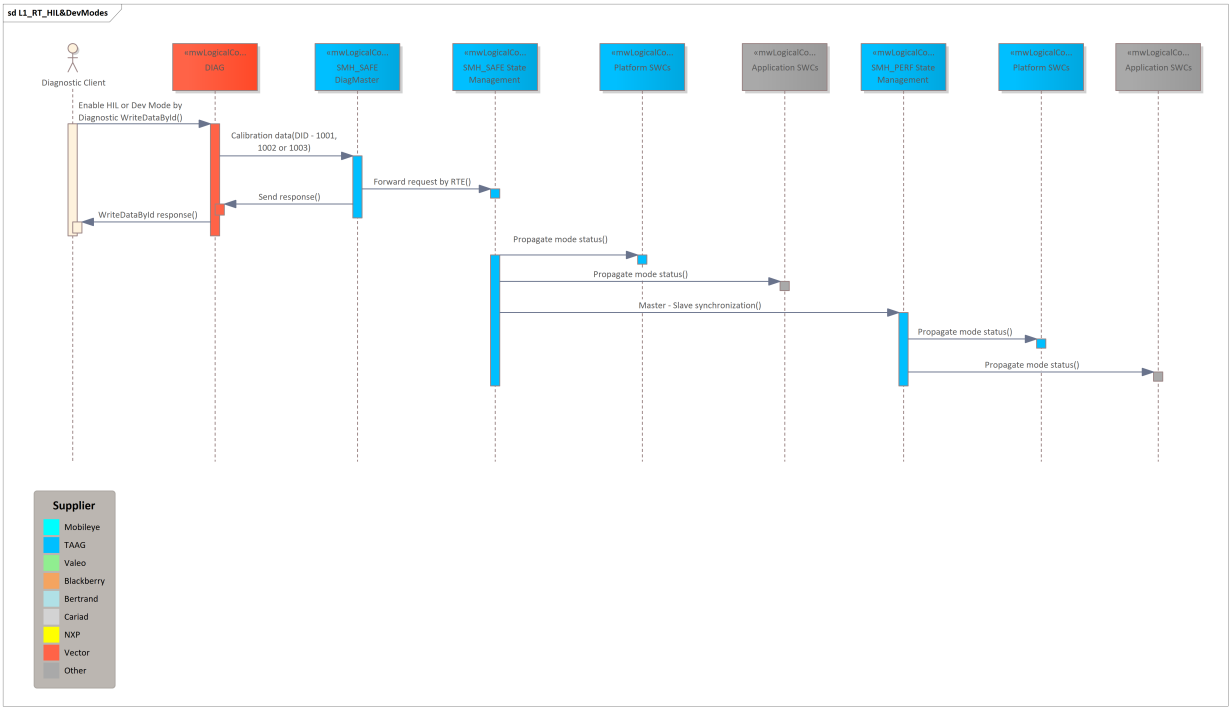
[S32GPRODP-296896]

6.2.2.2 Degraded mode and defect mode



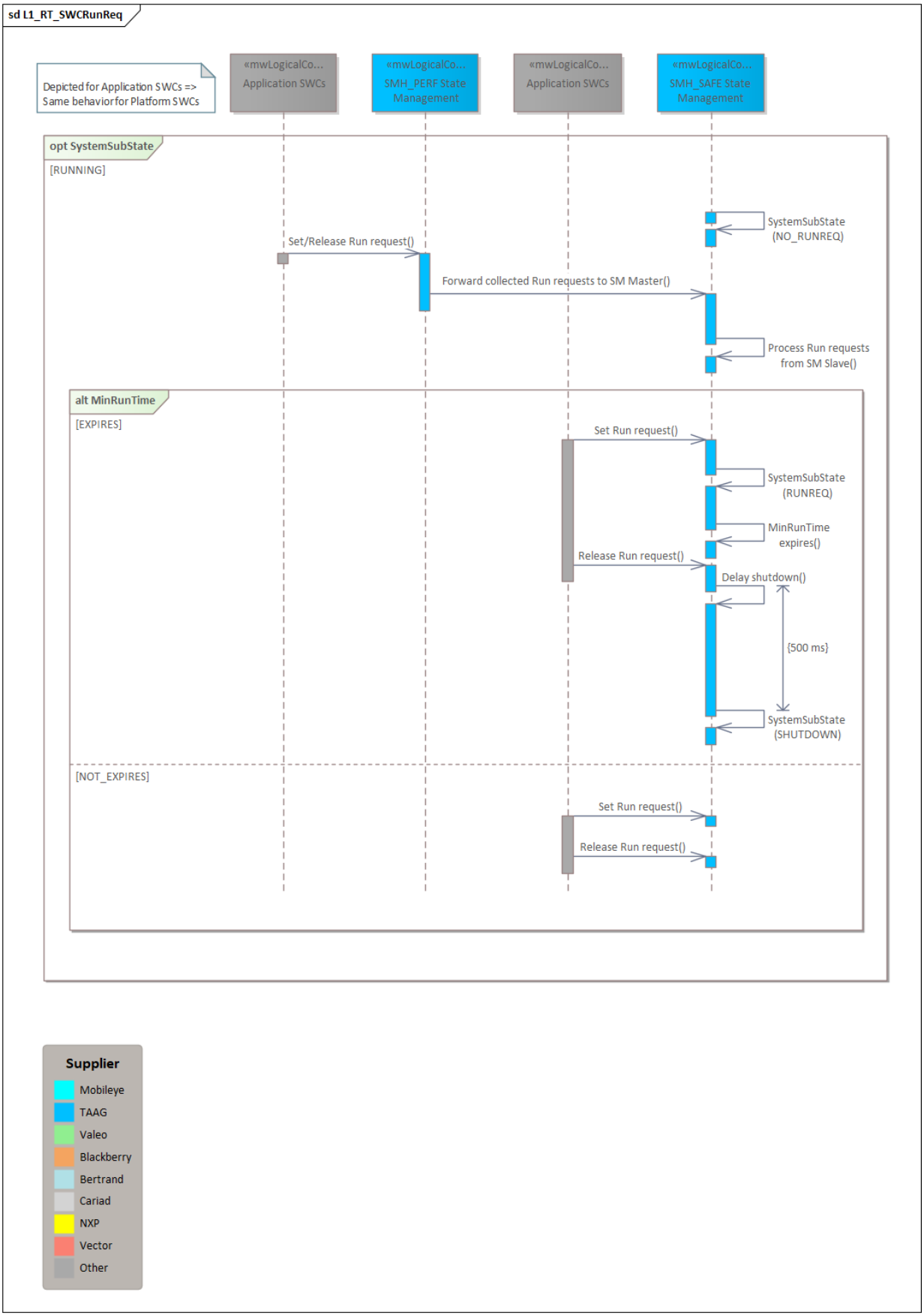
Degraded mode is used to improve availability of the overall system by enabling it to safely stop/restart SWCs instead of whole ECU resets while maintaining safe execution of non-failed components.
Defect mode is similar to Degraded mode except that it is based on having the SMH SH host running only, while other hosts do not boot.
Degraded and Defect modes can be started only during the boot process and can be triggered as a diagnostic request or by Error Handler as a consequence of permanent defects. [S32GPRODP-296210]

6.2.2.3 HIL mode and development mode



Hardware-in-the-loop (HIL) simulation is a method for testing the functions, system integration, and communication of ECUs thoroughly.
Development mode allows the ECU to enable/disable debugging functionalities and secure remote access for development purposes.
These modes are triggered via DIDs from diagnostics and the information on whether they are active or not is provided to all Platform and Application SWCs. [S32GPRODP-296209]

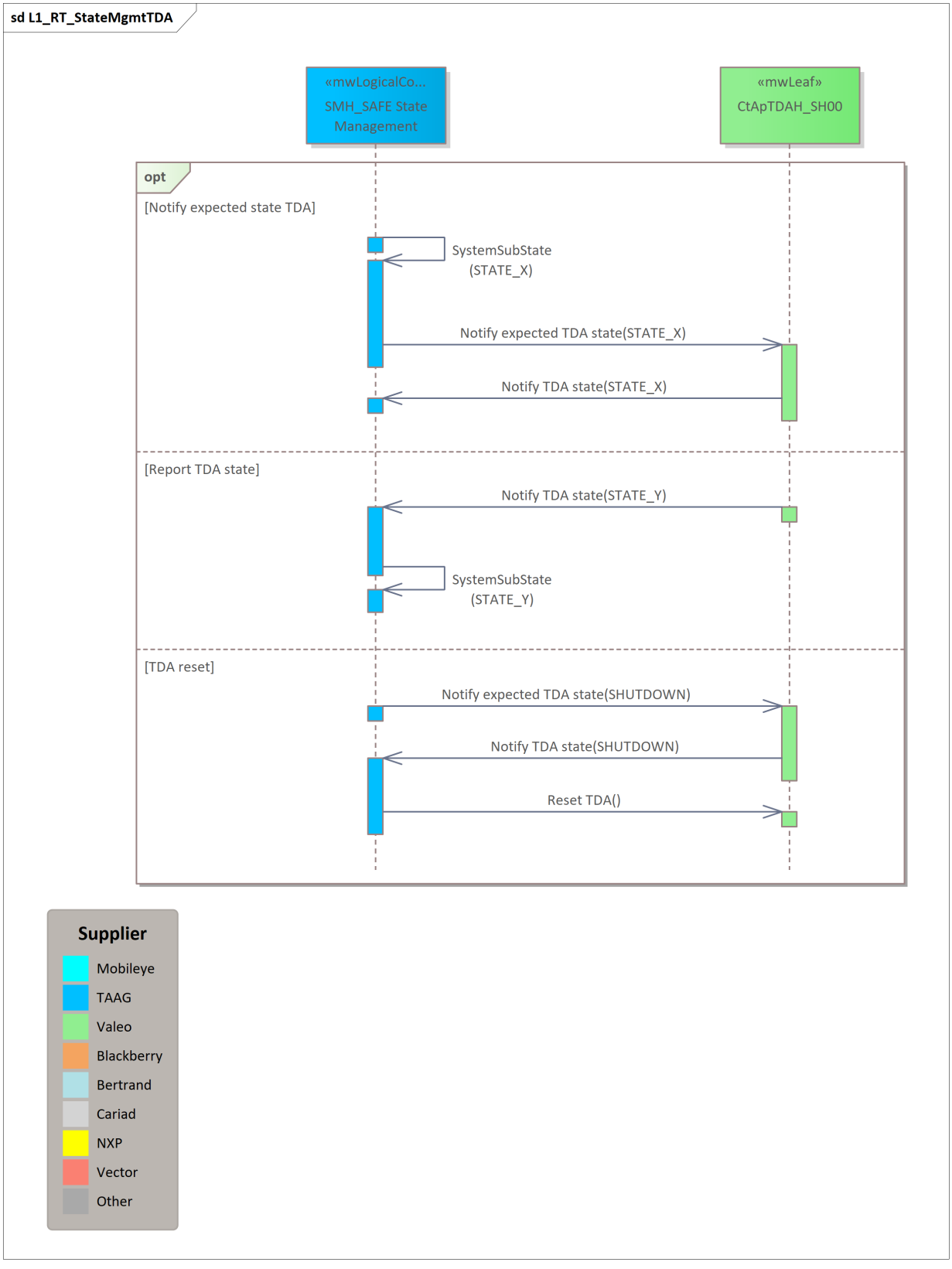
6.2.2.4 SWC Run Request



Application SWCs can request EcuSM to keep the subsystem running by sending a run request. Otherwise, the subsystem starts to shutdown after MinRunTime expires. If the Application SWCs release the run request, the

shutdown is delayed for 500 ms. [S32GPRODP-296180]

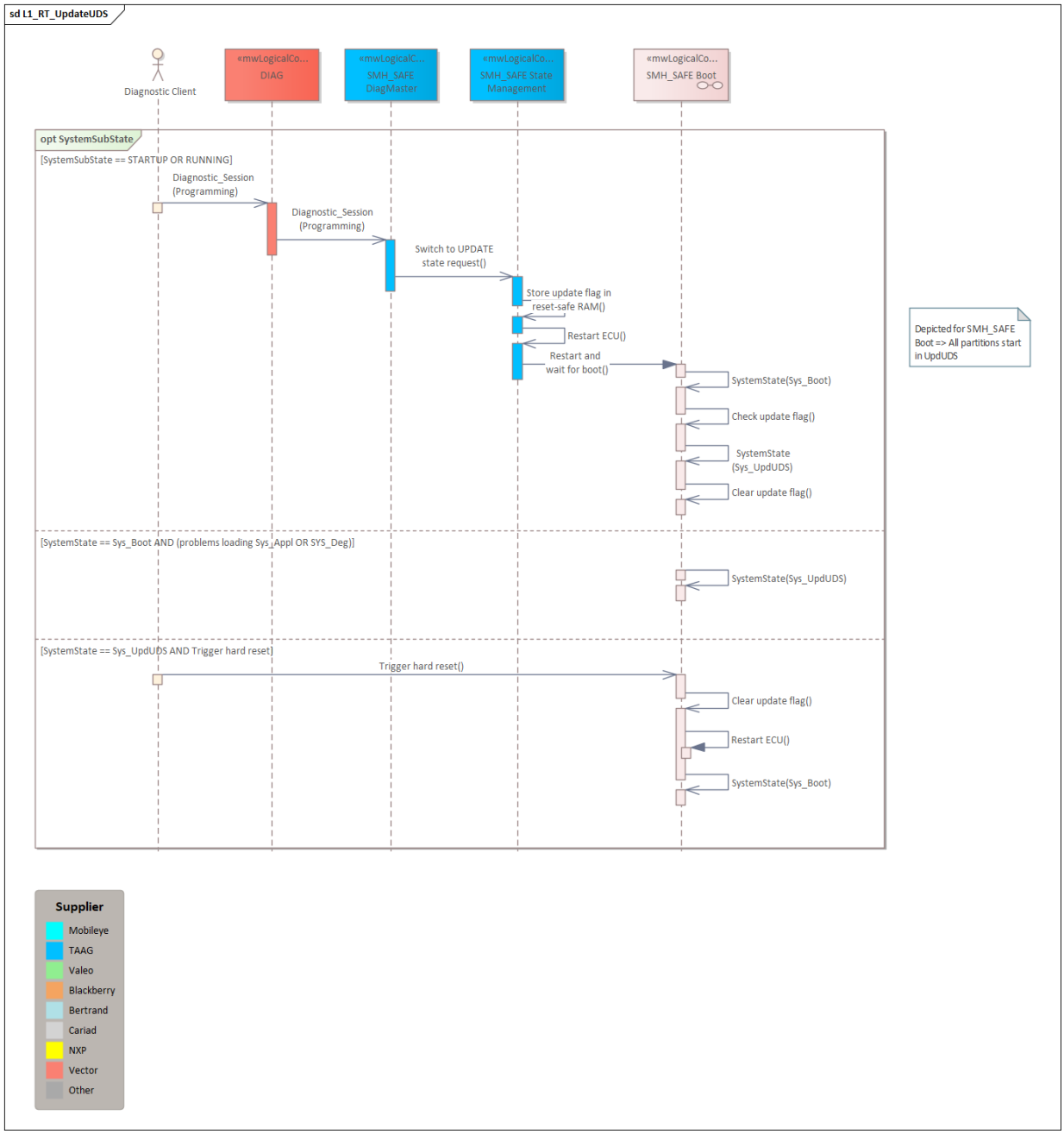
6.2.2.5 State management TDAs and EyeQs



EcuSM shares the SystemSubState with the TDAs and EyeQs. The TDAs and EyeQs also share their status with

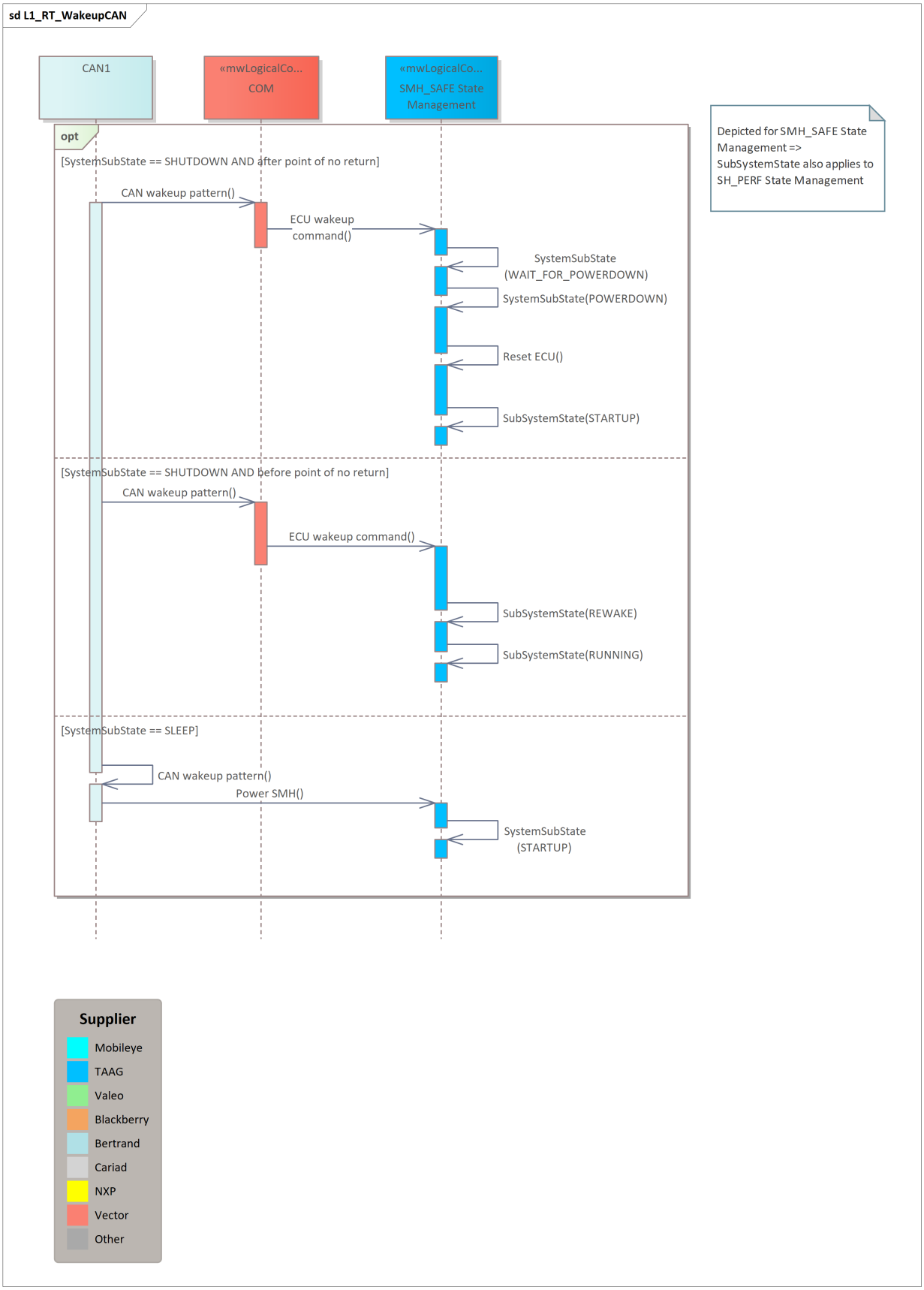
EcuSM, which could influence the SystemSubState where EcuSM is hosted. [S32GPRODP-296181]

6.2.2.6 Update UDS



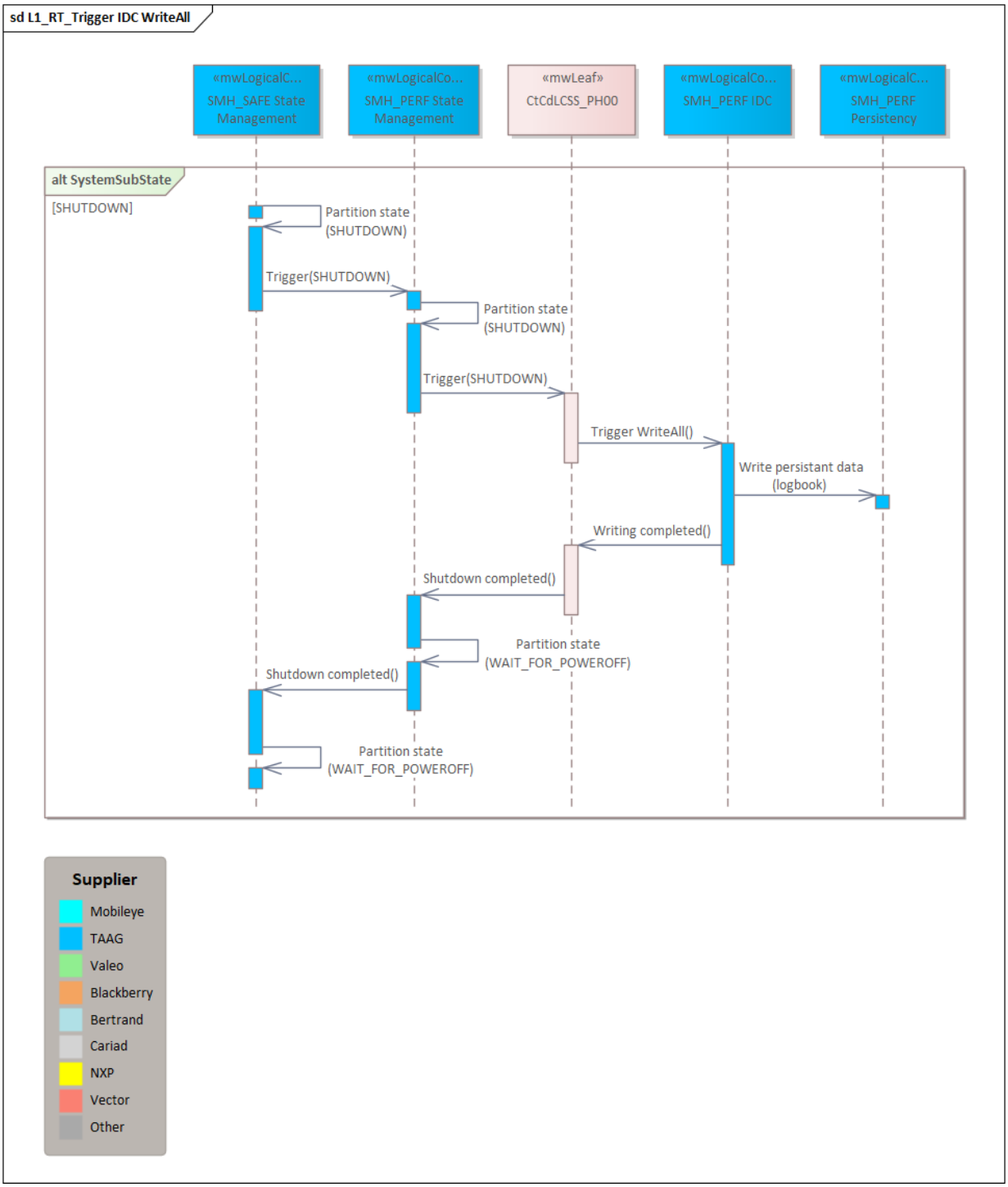
A diagnostic programming session request causes the system to reset and start in Sys_UpdUDS. The diagram also shows the cases when the system is booting (Sys_Boot) and there is a problem loading the normal application (Sys_Appl) or degraded mode images (Sys_Deg). [S32GPRODP-296184]

6.2.2.7 Wakeup via CAN



The ECU can be put in a sleep state. The sleep state can be left if receiving a wakeup pattern command via CAN bus. The final goal is to have the system up and running. The diagram shows different scenarios when the wakeup pattern is received. In the first of them, the command is received when the SystemSubState is SHUTDOWN and after the point in which the system cannot be put back into RUNNING, so the system resets. In the second scenario, the wakeup command is received when the system is still able to be put back at RUNNING, which is exactly what happens. In the third scenario, the system is sleeping, and the CAN module powers the ECU, which starts up. [S32GPRODP-296192]

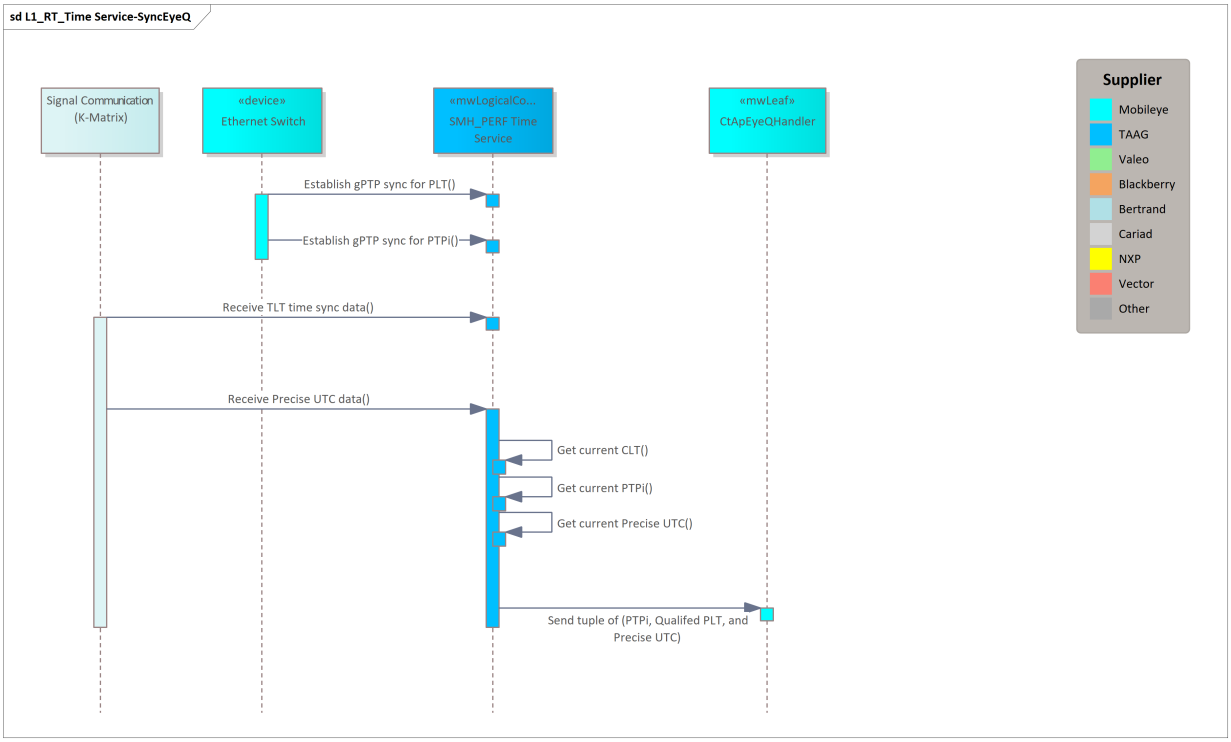
6.2.2.8 Trigger IDC WriteAll



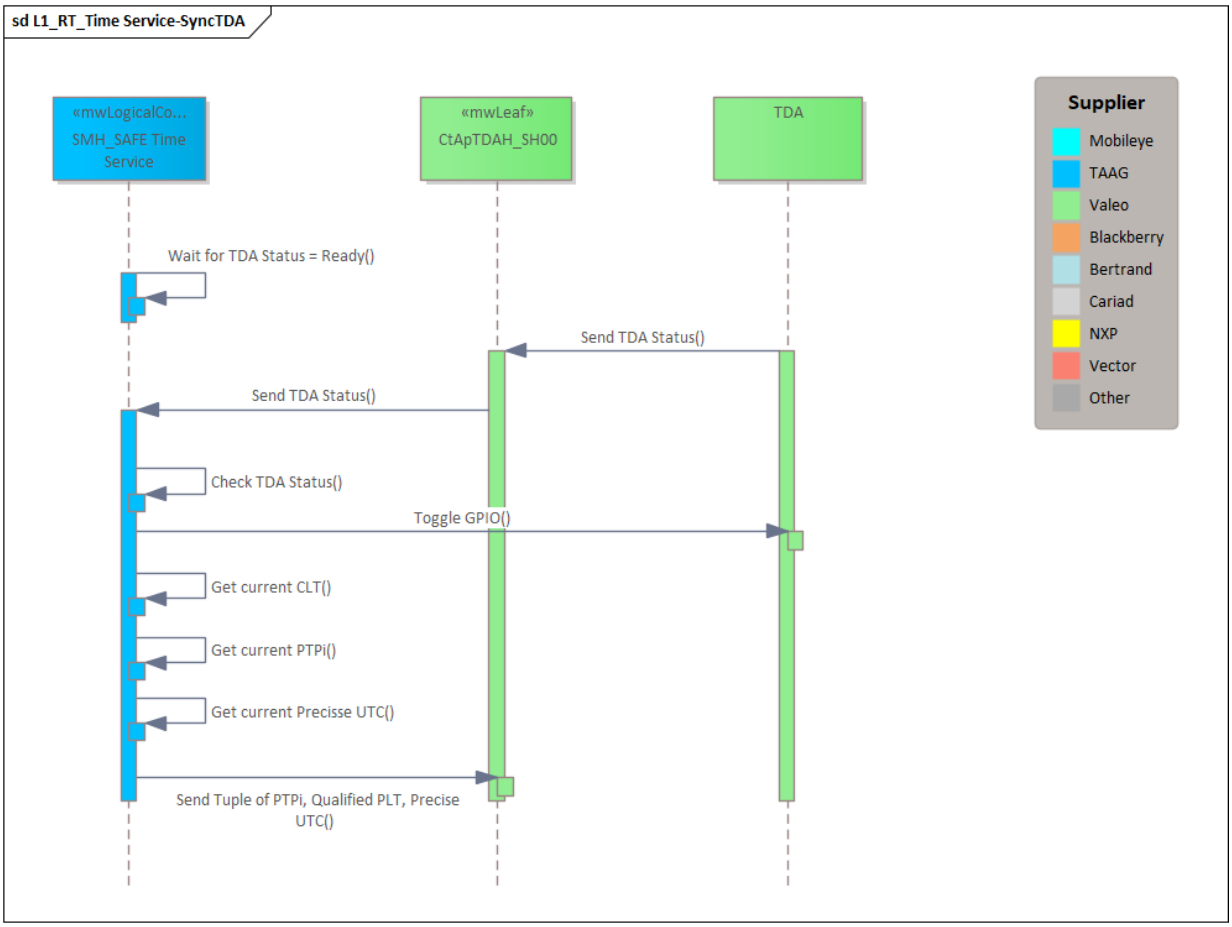
When a shutdown is triggered the IDC has to write some data persistently. [S32GPRODP-296193]

6.2.3 Time service

6.2.3.1 TimeSync with EyeQs



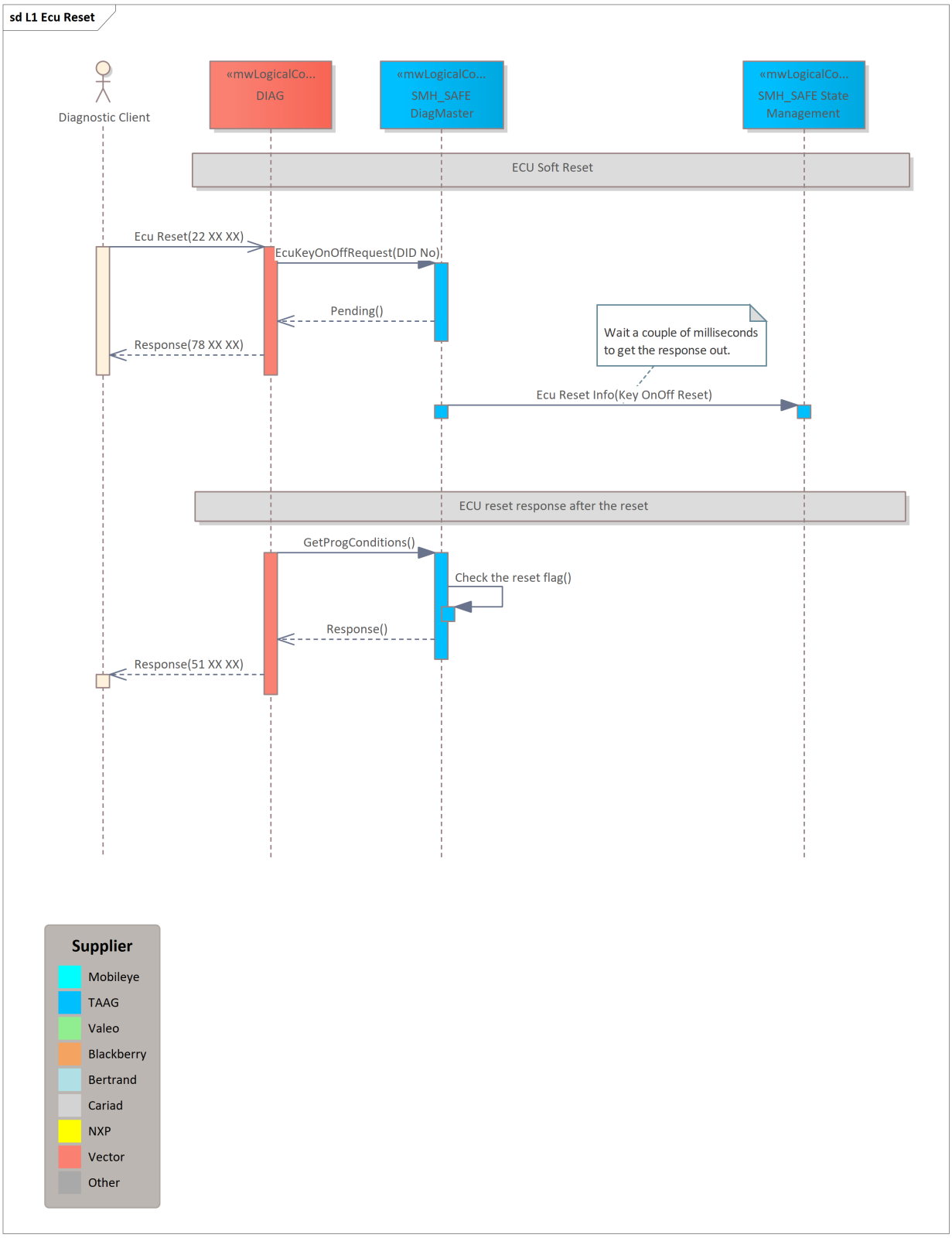
6.2.3.2 TimeSync with TDA



6.2.4 Diagnostic

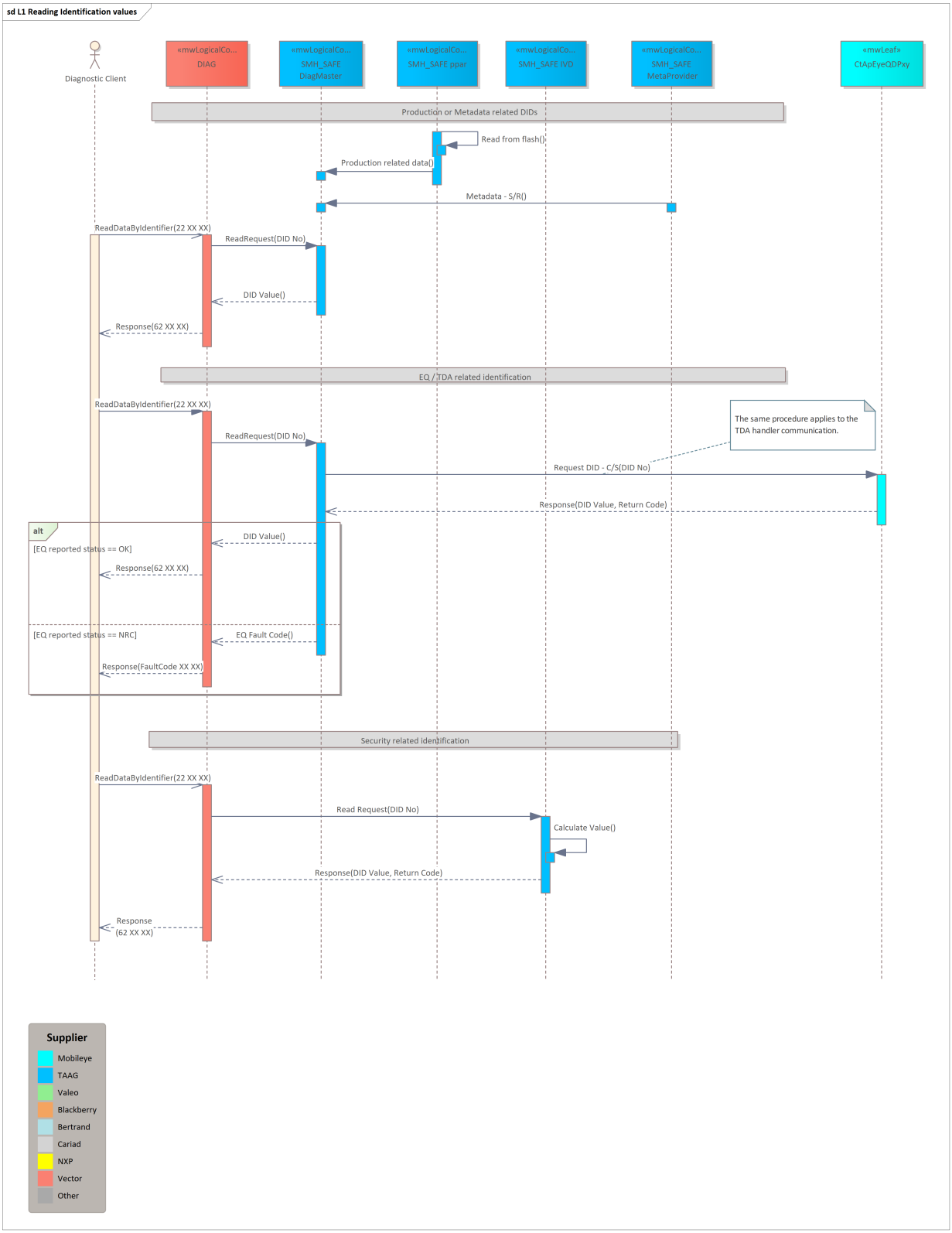
6.2.4.1 Diagnostic Services

Diagnostic Session Control is implemented by the BSW DIAG component. Positive and Negative Responses to the tester are handled by this component as well. **[**S32GPRODP-296308 **]**



The EcuReset service connects to the EcuSM component to perform the shutdown/reset. The diagnostic master sends a pending response to the tester before the reset and finishes the service after the reset has been performed. **[**S32GPRODP-296309 **]**

6.2.4.2 Diagnostic Objects



Identification

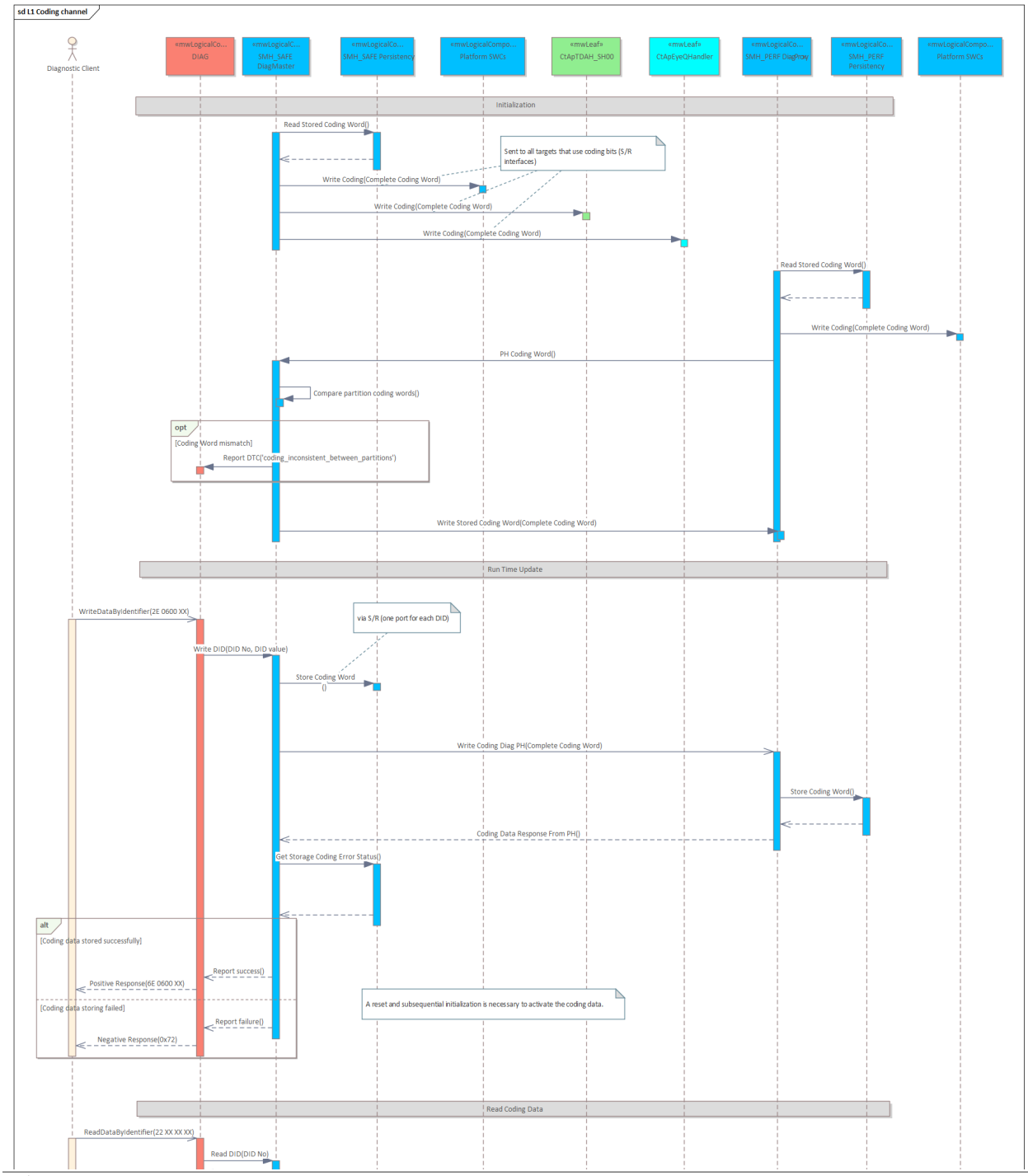
Identification is the Data that is used to identify different components (SW and HW) of the system.

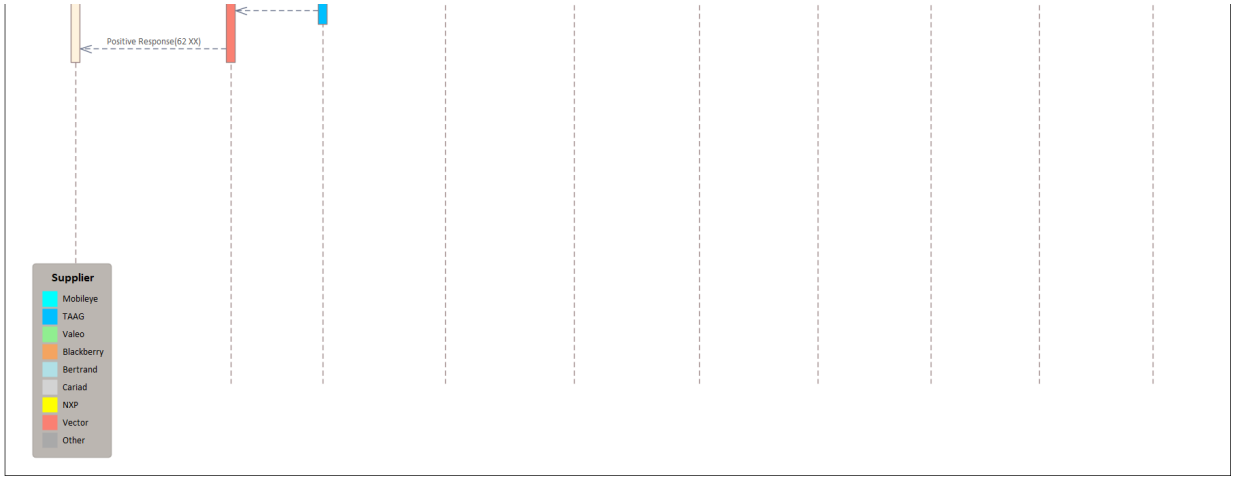
Software components responsible for giving the Identification data can run on safety partition or performance partition.

Production related parameters (PPAR) are stored in the flash at fixed location at production plant. Upon every start-up that PPAR structure would be loaded from Flash to RAM such that this RAM copy of PPAR structure is available to DiagMaster to read out the parameters from the PPAR structure and to give response to corresponding Identification DID.

The Metadata Provider serves the identification of 'Logical Blocklist' and 'Dataset Logical Blocklist'.

The components EyeQHandler and TDA Handler also provide Identification DIDs which are just read and forwarded by the DiagMaster.
[S32GPRODP-296762 **]**

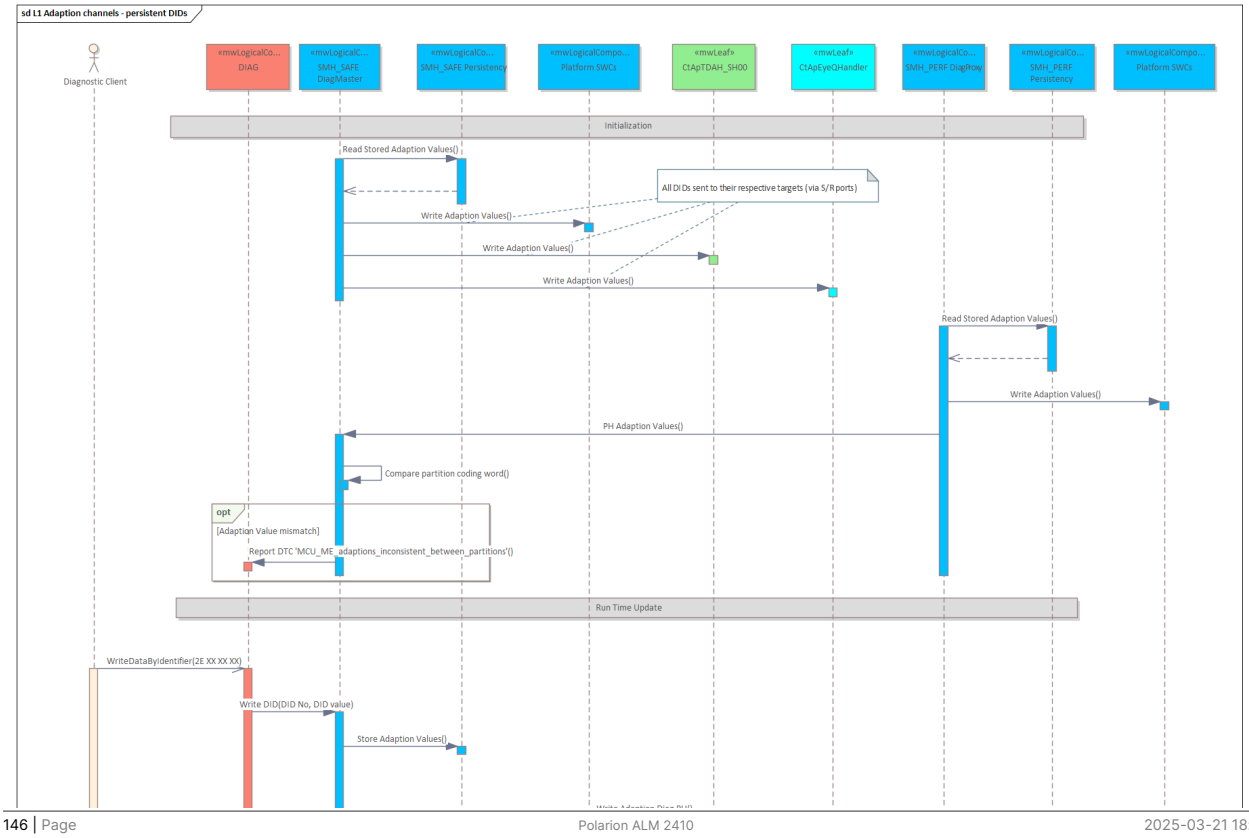


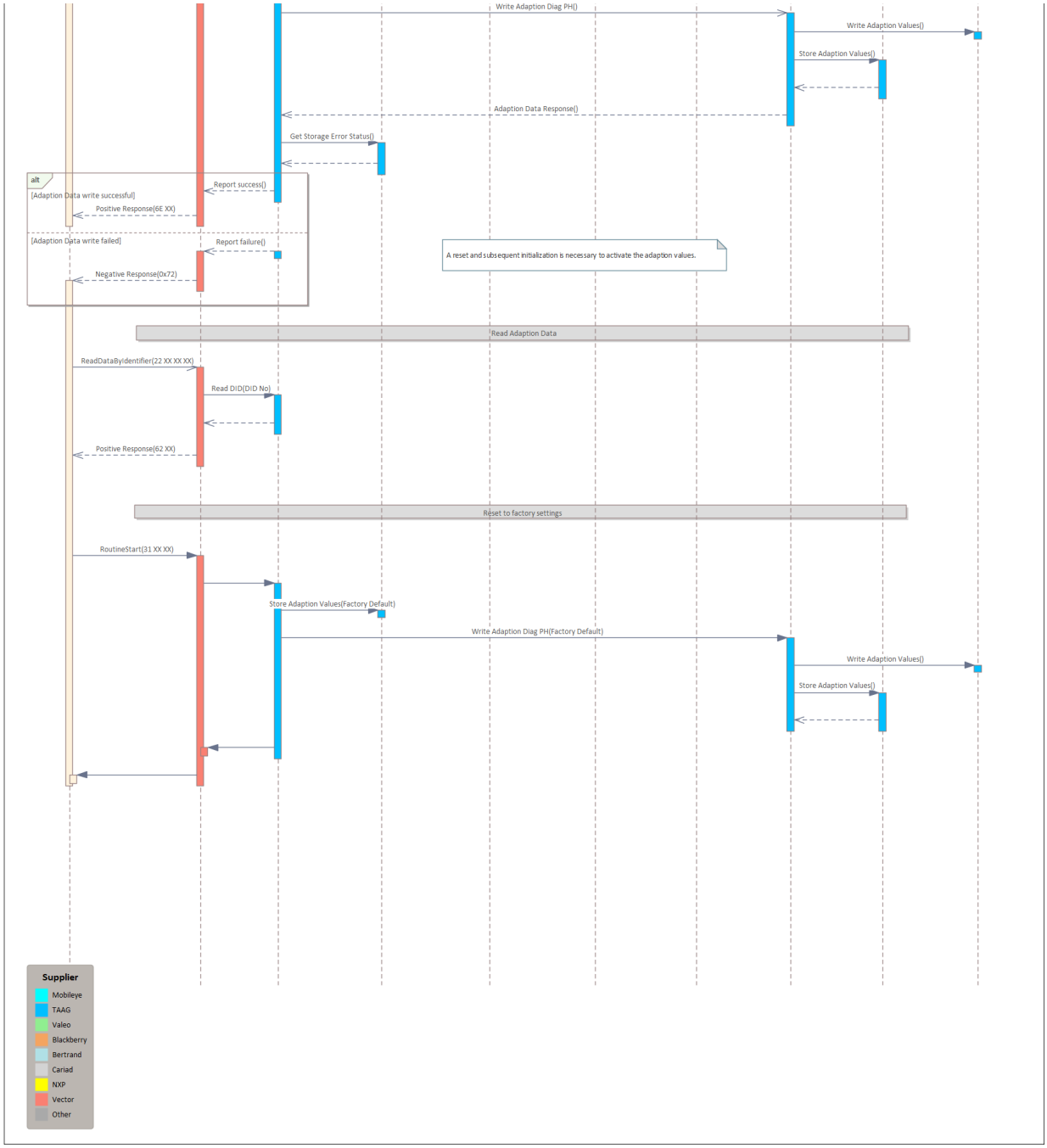


Coding channel

1. DiagMaster receives the coding request from DCM.
2. Contacts the persistence component on the Safety partition to persist the coding data.
3. transfers the coding data on to the DiagProxy on the Performance partition via MiddleWare such that DiagProxy on Performance partition contacts persistence component locally to persist the coding data on the Performance partition.
4. Sends back the positive response upon successful coding.
5. DiagMaster also ensures the consistency of the coding across the the cores and raises a DTC in case of inconsistency.
6. DiagMaster and DiagProxy are responsible for providing the Coding data to all the SW-Cs (which is realized through the modeling between DiagMaster, DiagProxy and other SW-Cs).

[S32GPRODP-296758]



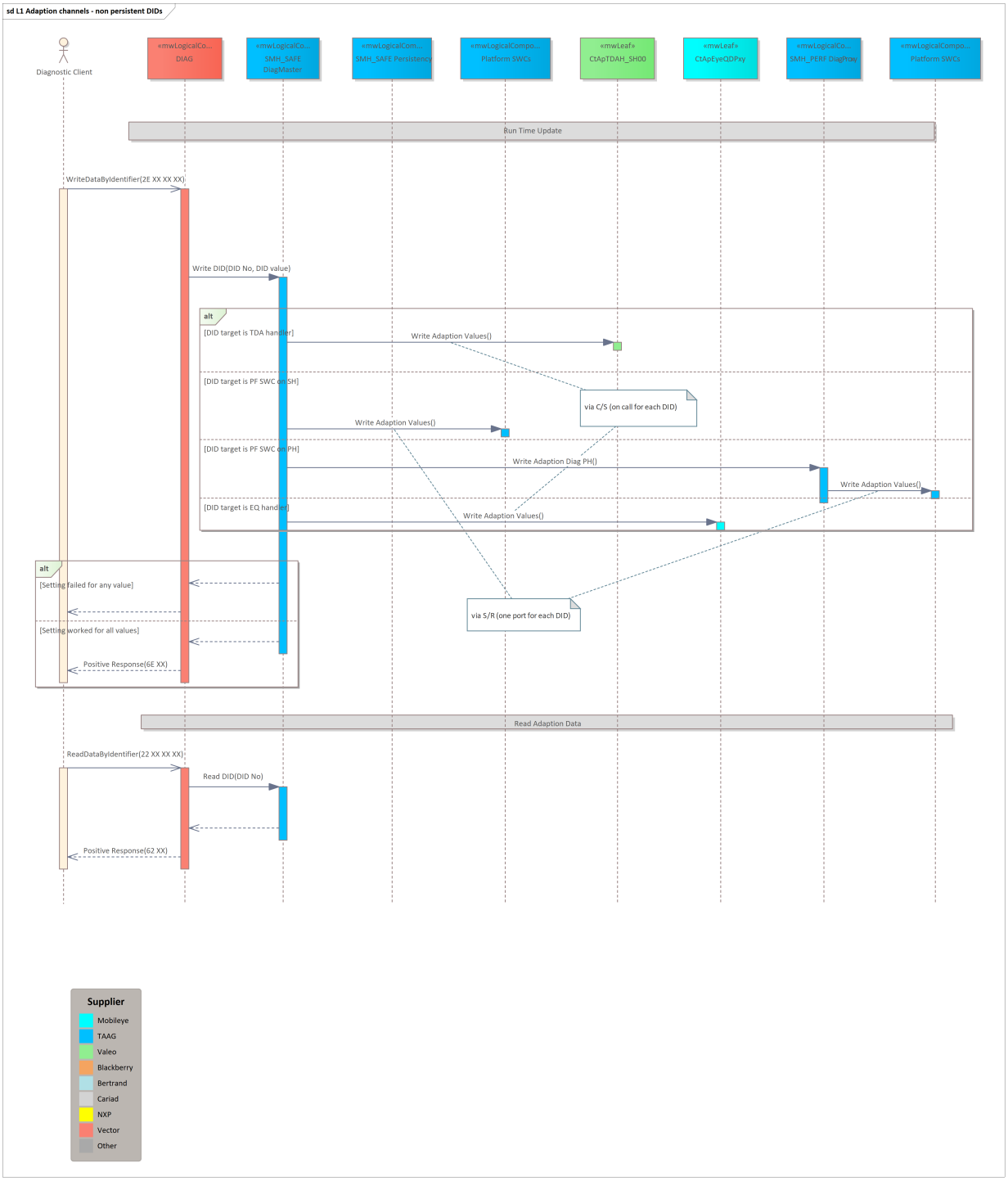


Adaption channels - persistent DIDs

1. DiagMaster and DiagProxy are responsible to provide the adaption data to all the necessary SW-Cs (done through the modeling).
2. DiagMaster & DiagProxy provides the default adaption data when there is no request from outside to change the adaption data.
3. DiagMaster receives the adaption request from the DCM and provides the data to Persistency component to persist the same. DiagMaster also sends the necessary adaption data to DiagProxy component on the Performance partition to be persisted locally on Performance partition with the help of Persistency SW-C. Sends back the positive response upon successful adaption data saving.
4. Upon KeyOffOnReset DiagMaster & DiagProxy components hold the latest adaption data available to all SW-Cs.

5. DiagMaster also handles the routine related to adaption ResetOfAllAdaptions-0x0366 to reset all the adaptation data back to factory resettings/default data.

[S32GPRODP-296759]

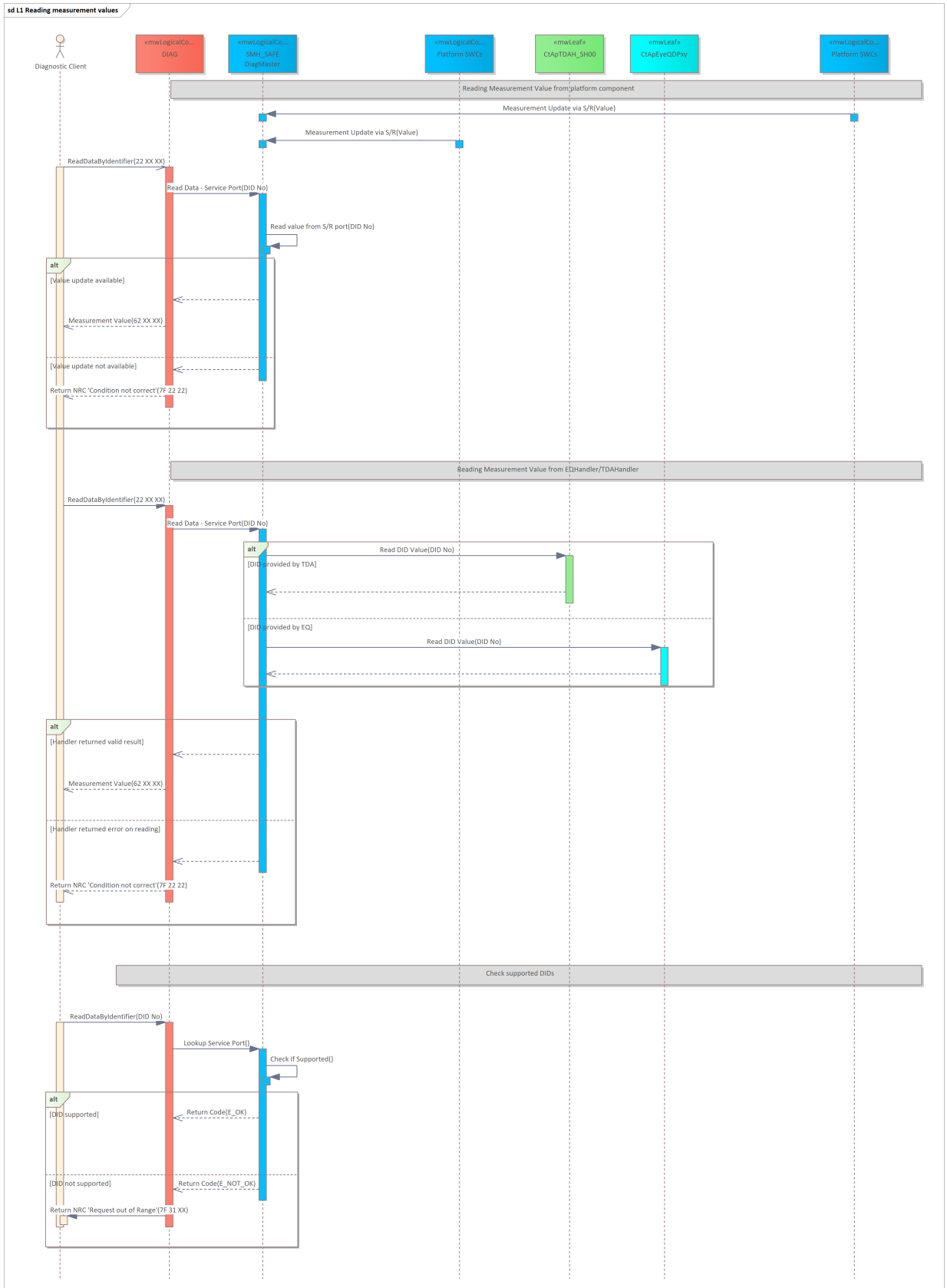


Reading Adaption channels - non-persistent DIDs

1. DiagMaster and DiagProxy are responsible to provide the adaption data to all the necessary SW-Cs (done through the modeling).
2. The default Adaption data has to be provided by the component.
3. DiagMaster receives the Adaption request from the DCM, sends the necessary Adaption data to DiagProxy

component on the Performance partition, which distributes it to the components on the performance partition

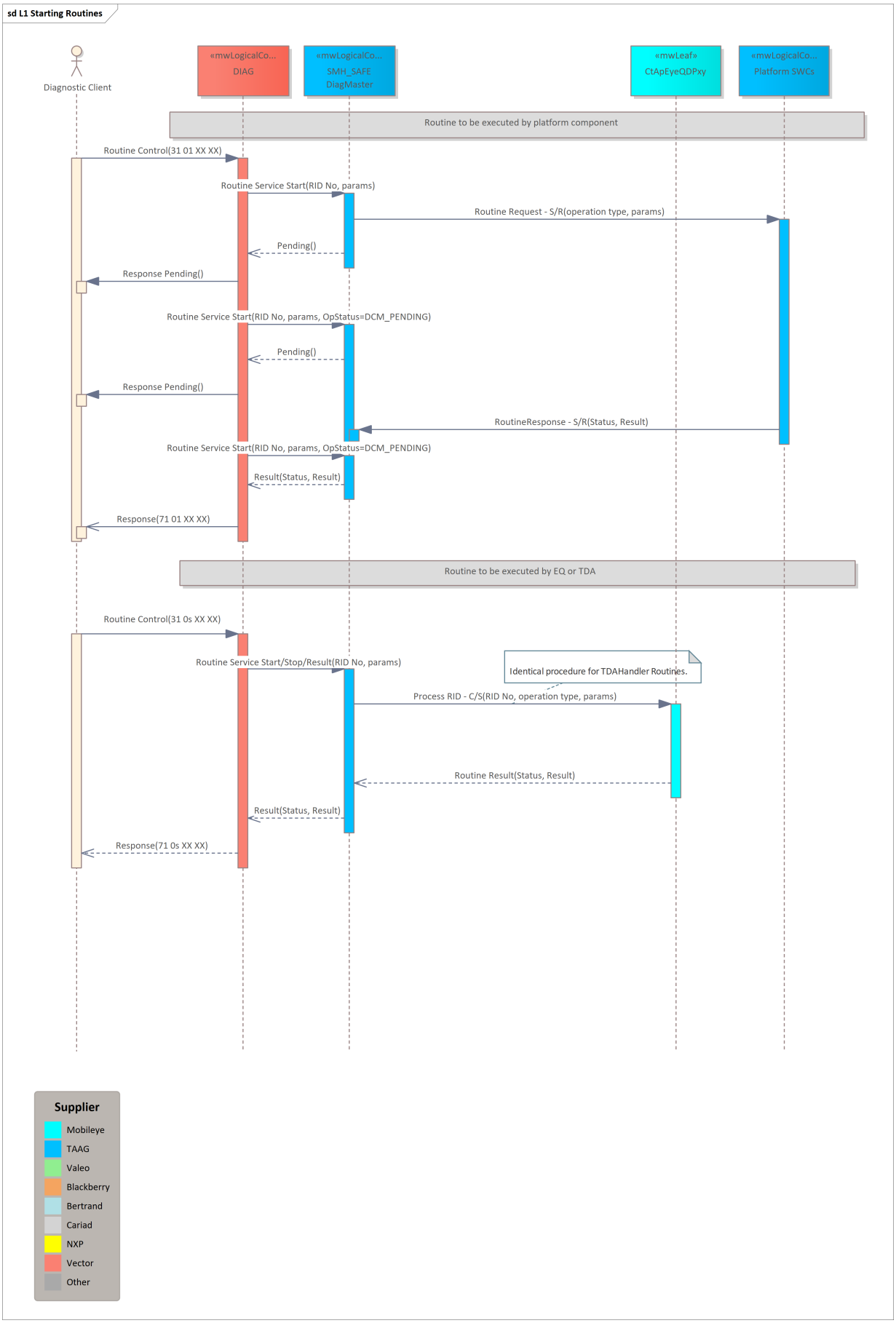
[S32GPRODP-296760]



Reading Measurement values

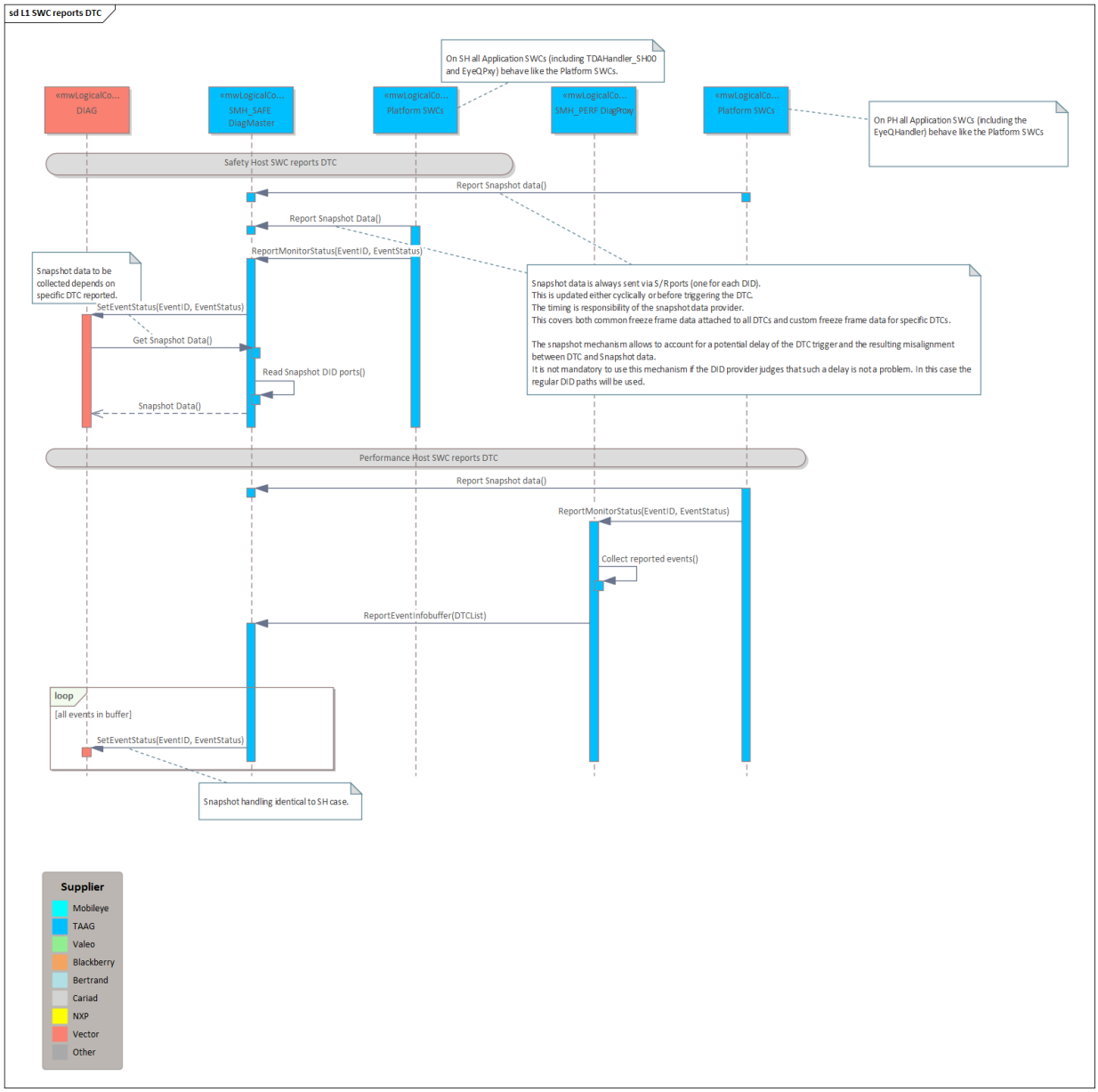
1. Receives the Diagnostic request (Read Data By Identifier) from DCM
2. Gets the updated measurement value from other SW-C (this update could be either periodically or event based update from other SW-C)
3. Sends back the Measurement value as response to DCM.

[S32GPRODP-296756 **]**


sd L1 Starting Routines
Diagnostic Client
«mwLogicalCo...
DIAG
«mwLogicalCo...
SMH_SAFE
DiagMaster
«mwLeaf»
CtApEyeQDPxy
«mwLogicalCo...
Platform SWCs
Routine to be executed by platform component
Routine Control(31 01 XX XX)
Routine Service Start(RID No, params)
Routine Request - S/R(operation type, params)
Pending()
Response Pending()
Routine Service Start(RID No, params, OpStatus=DCM_PENDING)
Pending()
Response Pending()
RoutineResponse - S/R(Status, Result)
Routine Service Start(RID No, params, OpStatus=DCM_PENDING)
Result(Status, Result)
Response(71 01 XX XX)
Routine to be executed by EQ or TDA
Routine Control(31 0s XX XX)
Routine Service Start/Stop/Result(RID No, params)
Identical procedure for TDAHandler Routines.
Process RID - C/S(RID No, operation type, params)
Routine Result(Status, Result)
Result(Status, Result)
Response(71 0s XX XX)
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

1. Receives the diagnostics request related to routines (Start, Stop and RequestResults) from DCM.
2. Both routines to be handled in the Safety partition as well as routines handled in the Performance partition will be managed completely within DiagMaster and forwarded to the respective software component (the type of port used however depends on the software component and the partition). The DiagMaster also collects the final result/response to be sent out.
3. The response would be sent back to DCM.

[S32GPRODP-296757]



1. Snapshot data has to be provided by the respective component at either

- regular intervals so it is up-to-date when the DTC becomes qualified.
- in a timely manner before calling the DTC, so it will be available at the DiagMaster when the DEM requests the data

2. Component reports the monitor status, DiagMaster forwards the report to DEM.

3. If the DTC becomes qualified, DEM will ask for the respective snapshot data,
4. DiagMaster will read the data from the respective S/R ports and return to DEM.
5. If the reporting component is located on the safety partition it will directly trigger the C/S port of the DiagMaster, if the reporting component is located on the performance partition it will trigger the C/S port of the DiagProxy, which will collect the the reports and send it to the DiagMaster, which in turn will process the requests like in 2.)

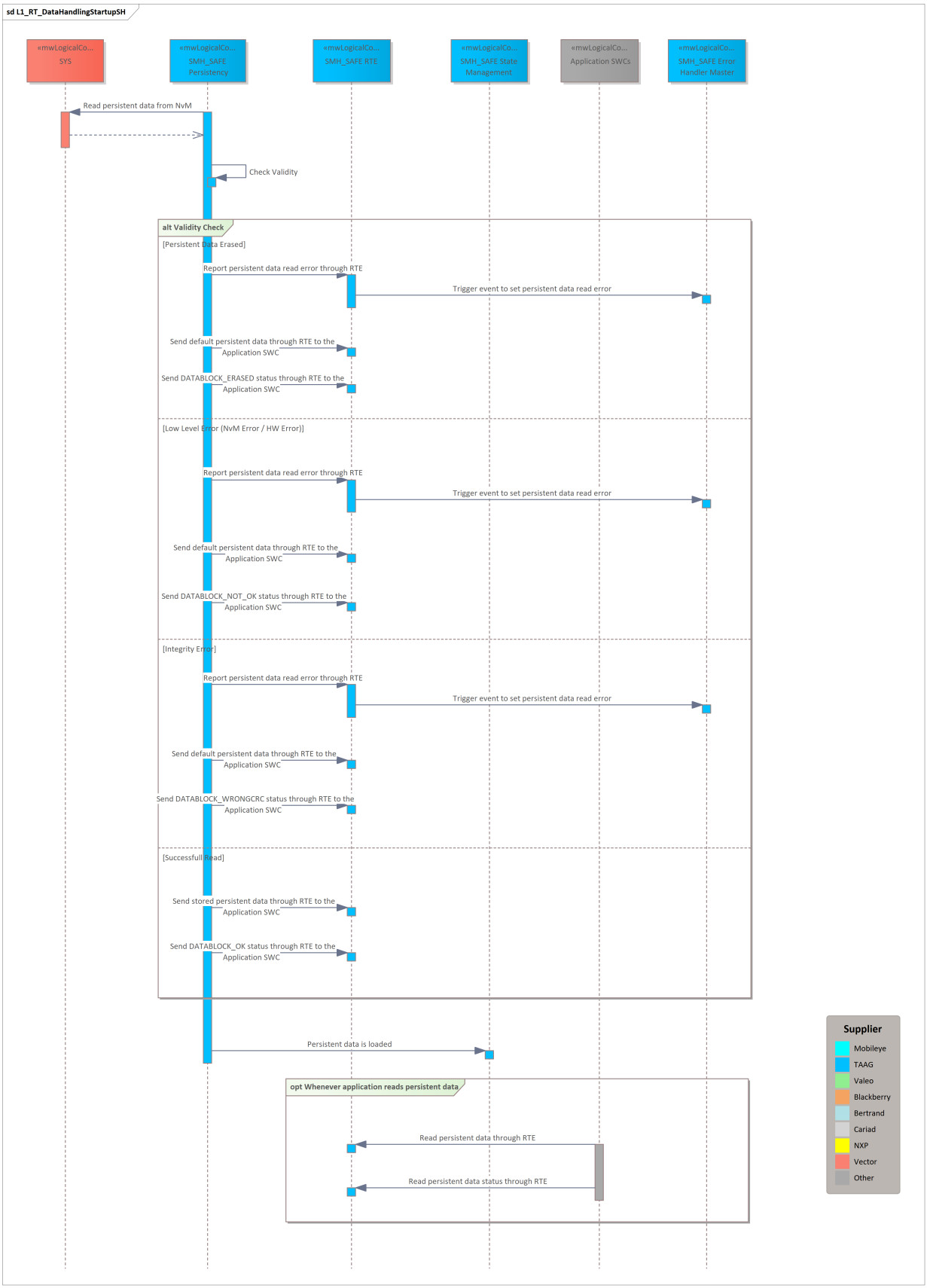
[S32GPRODP-296761 **]**

6.2.5 Persistent storage

6.2.5.1 Persistency

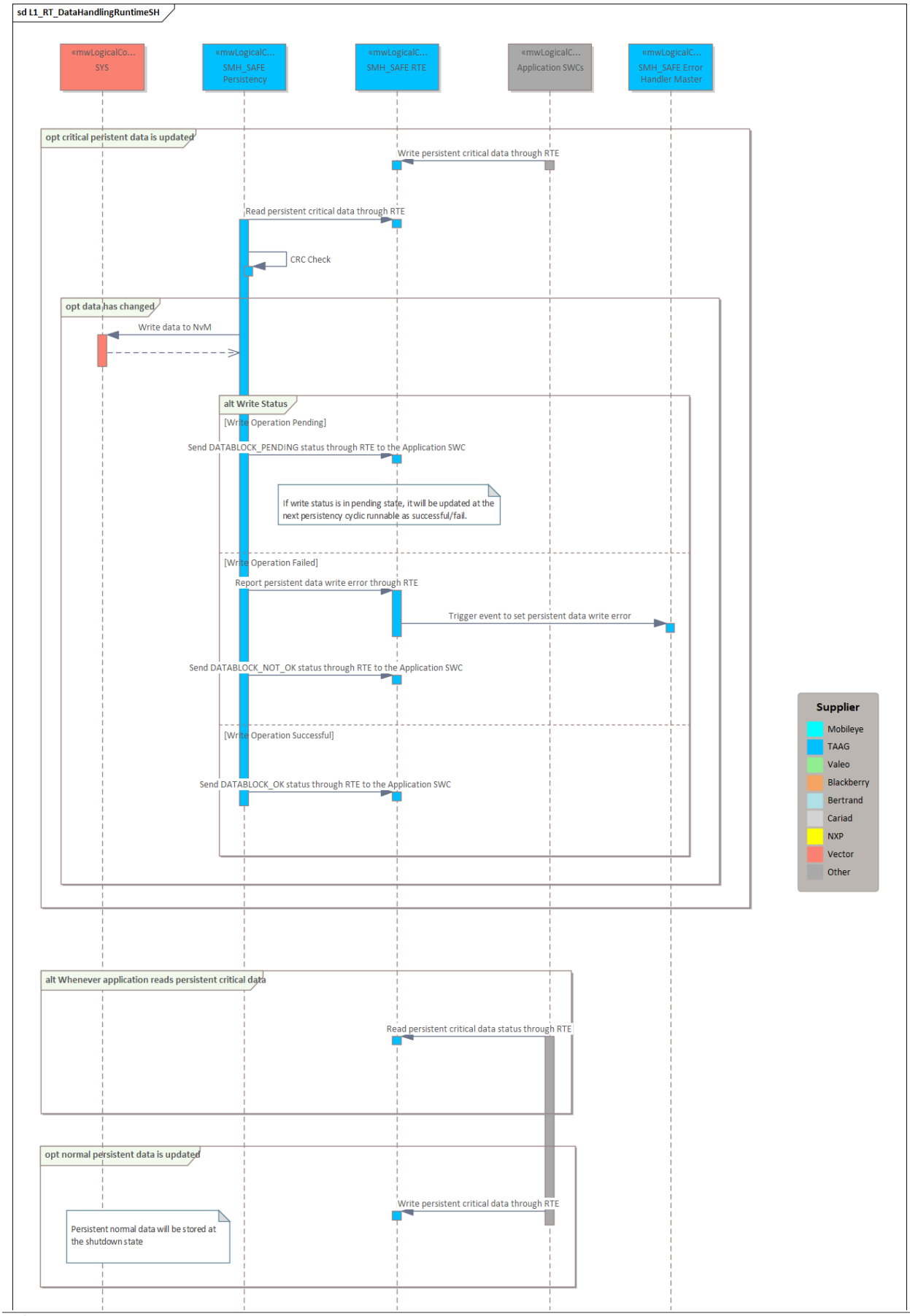
6.2.5.1.1 Safety host Persistency

At the init phase, Persistency Service reads persistent data from NvM and distributes them to the App SWC's. **[**S32GPRODP-296339 **]**

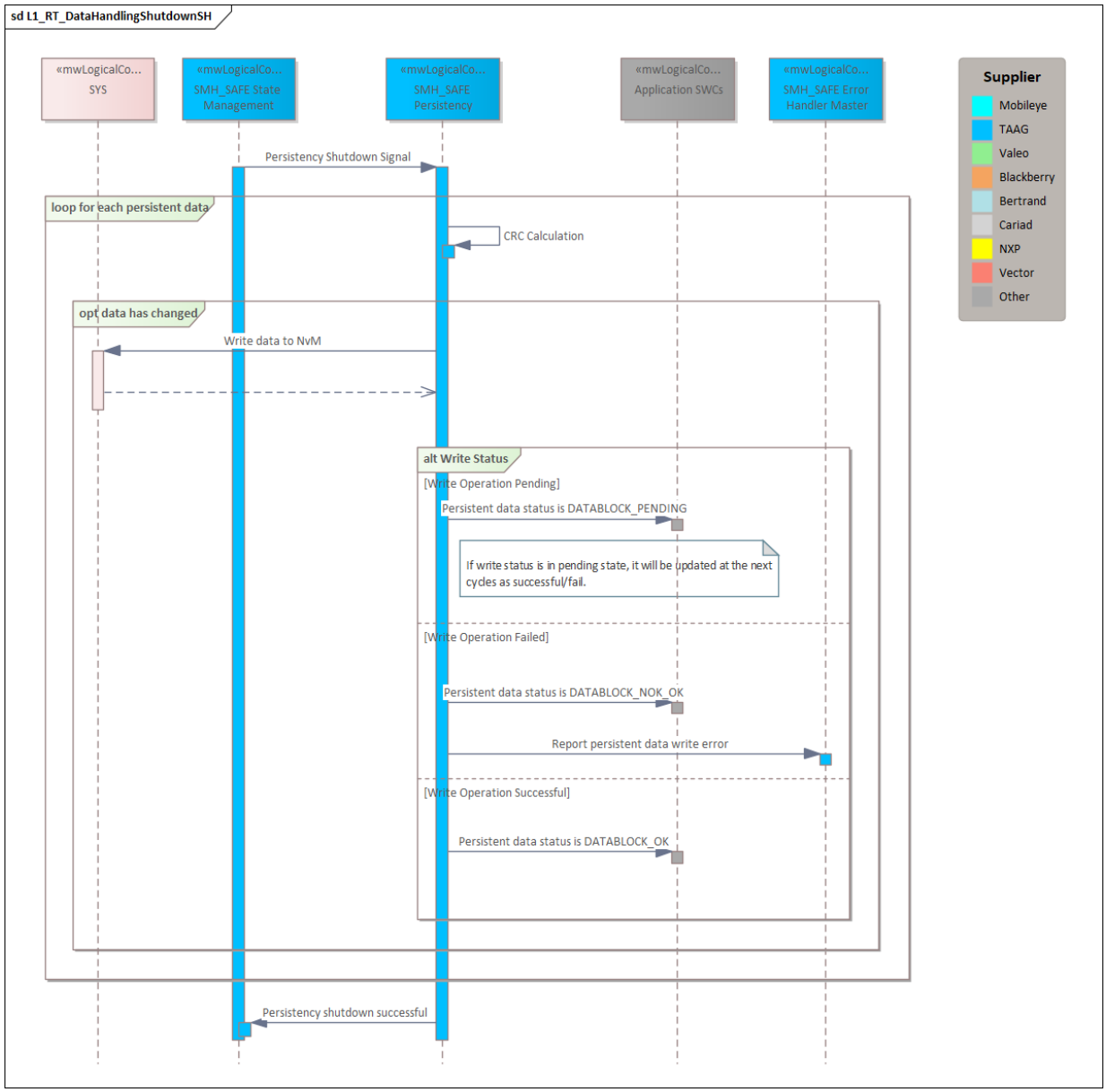


At the runtime, Persistency Service checks whether persistent critical data is updated by App SWC or not. If persistent critical data has been updated, Persistency Service writes this data to NvM (only persistent data, modelled as critical, is written at the runtime. Persistent data, modelled as normal data, will be written at the

shutdown state). **[**S32GPRODP-296340 **]**

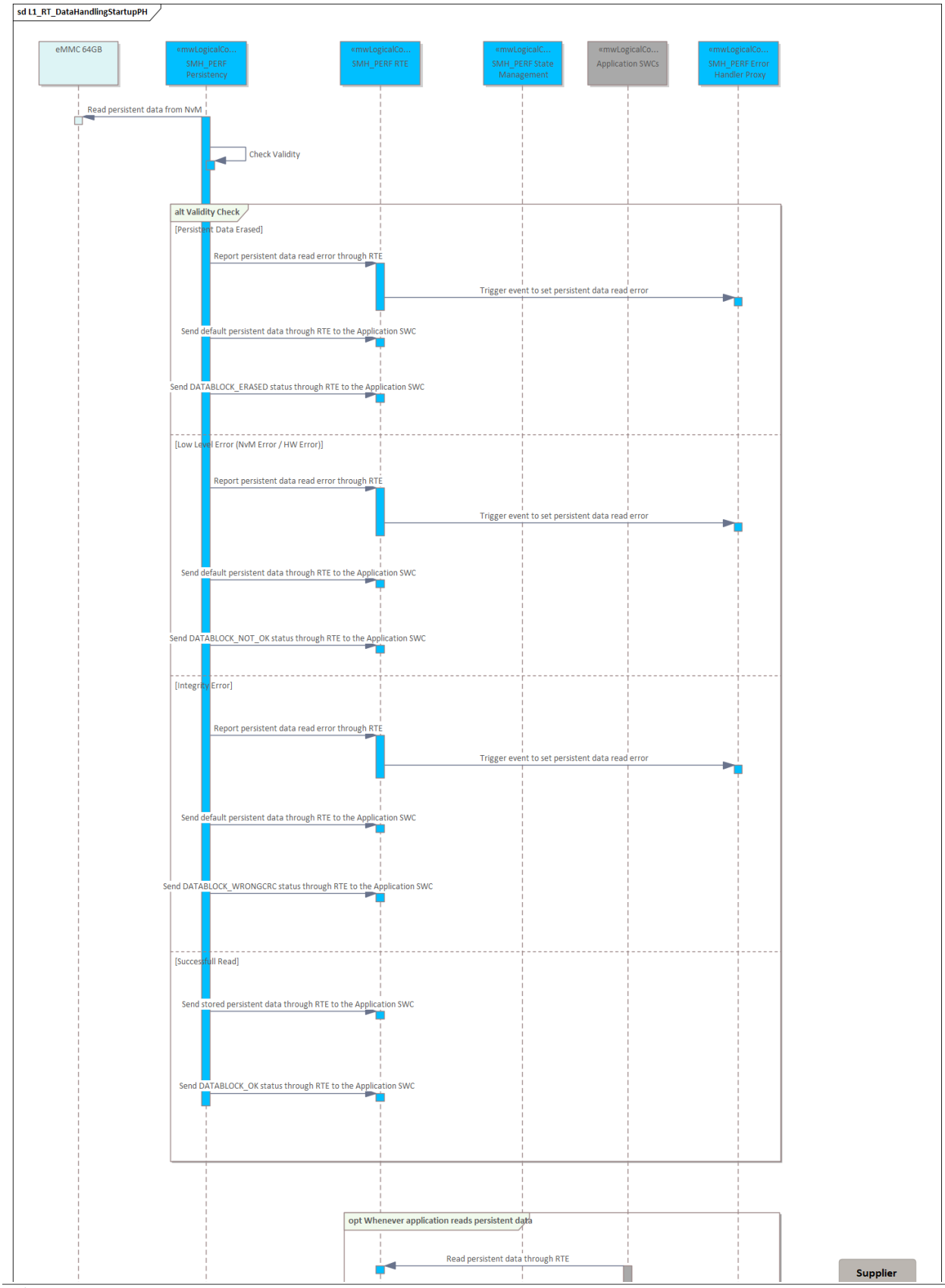


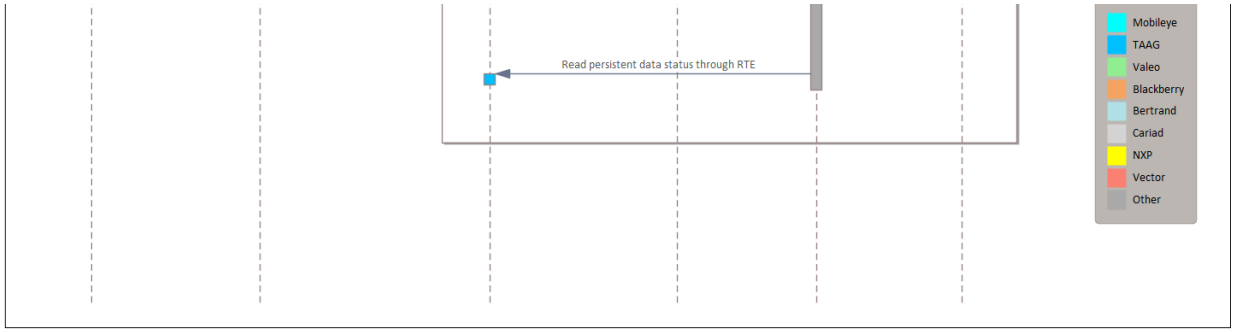
At the shutdown, Persistency Service checks all persistent data whether it is updated by App SWC or not. If persistent data has been updated, Persistency Service writes this data to NvM (both persistent critical data and persistent normal data will be written to NvM if it is updated by App SWC's). **[**S32GPRODP-296342 **]**



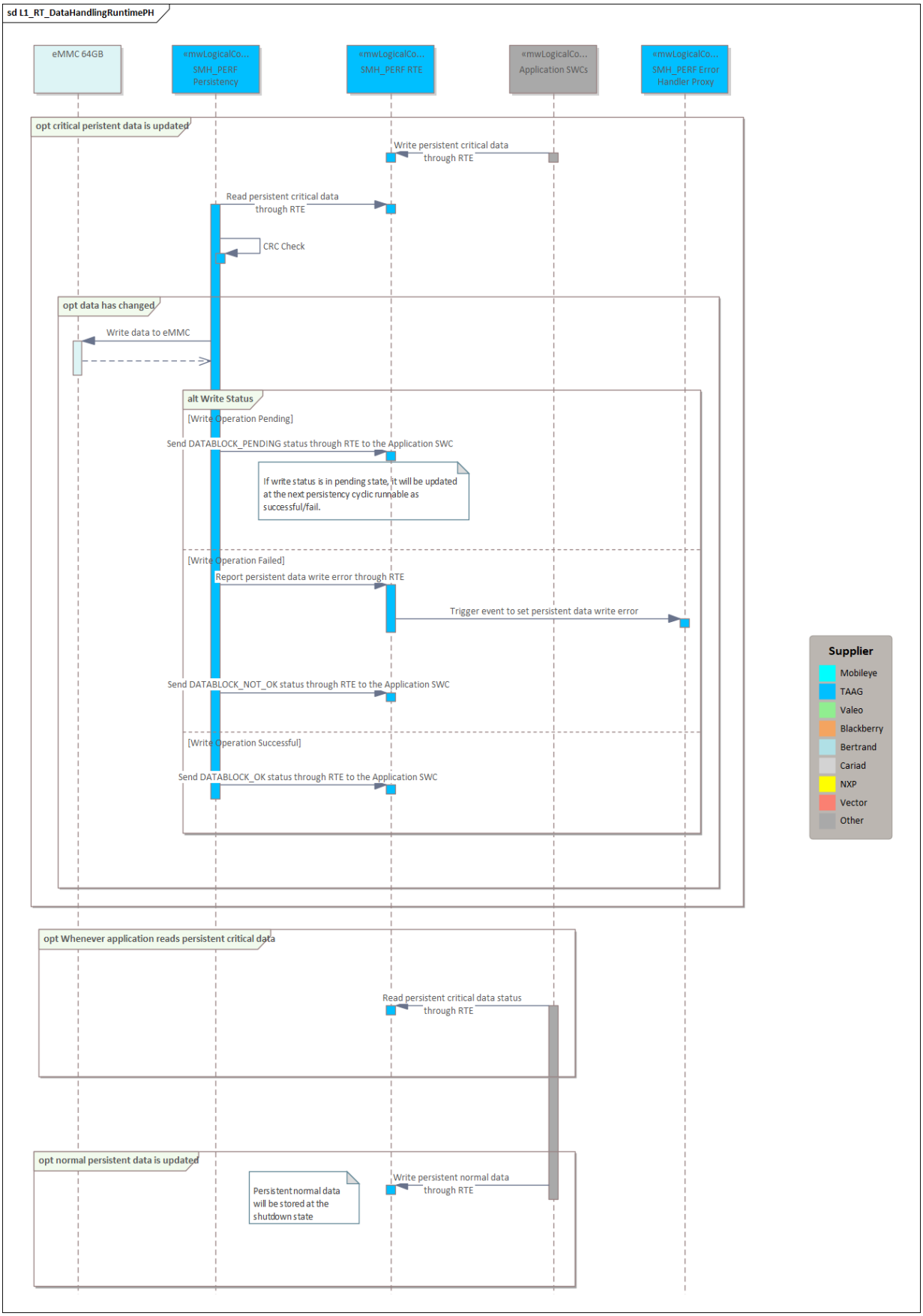
6.2.5.1.2 Performance host Persistency

At the init phase, Persistency Service reads persistent data from NvM and distributes them to the App SWC's. [S32GPRODP-296344]



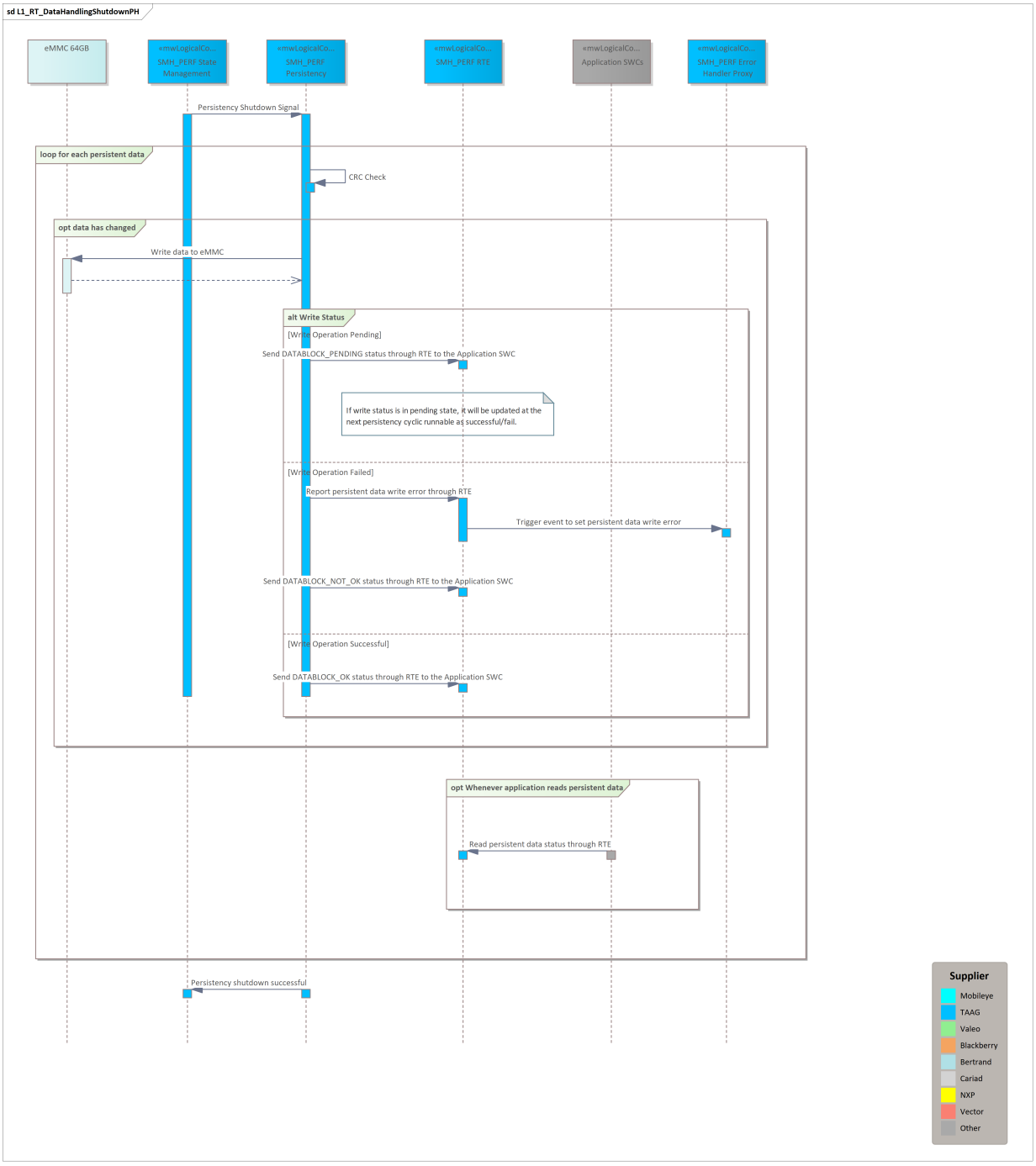


At the runtime, Persistency Service checks whether persistent critical data is updated by App SWC or not. If persistent critical data has been updated, Persistency Service writes this data to NvM (only persistent data, modelled as critical, is written at the runtime. Persistent data, modelled as normal data, will be written at the only shutdown state). **[**S32GPRODP-296345 **]**



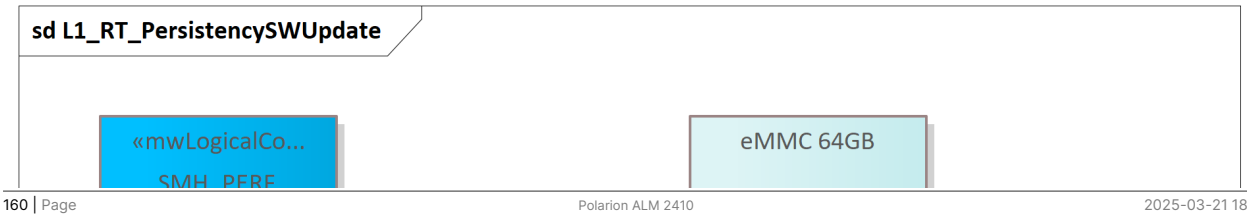
At the shutdown, Persistency Service checks all persistent data whether it is updated by App SWC or not. If persistent data has been updated, Persistency Service writes this data to NvM (both persistent critical data and

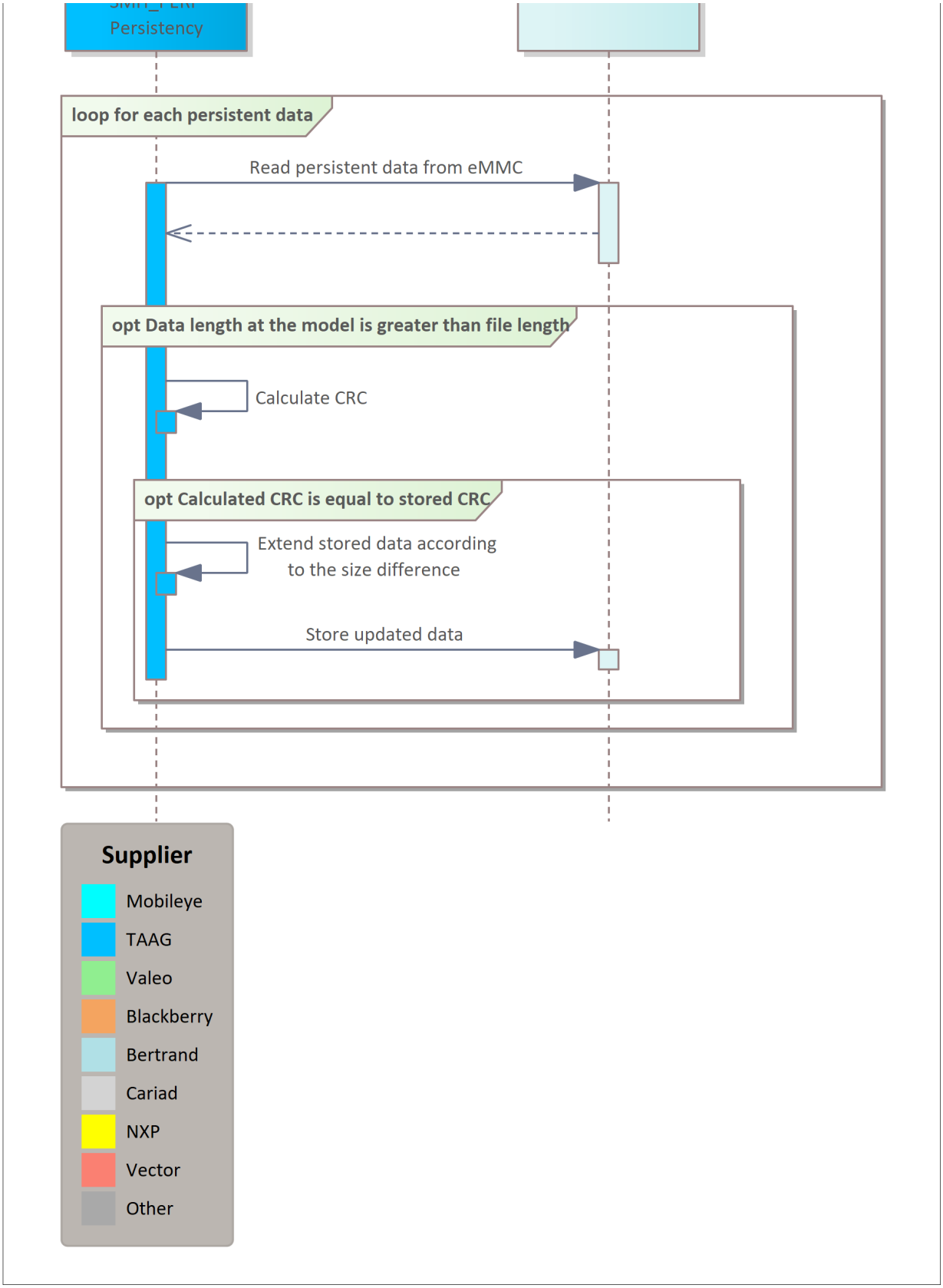
persistent normal data will be written to NvM if it is updated by App SWC's). [S32GPRODP-296346]



6.2.5.1.3 Persistency Software Update and Migration

At the SMH-Safe, Persistency Service will have 15% reserved area for the future extension of the data.
At the SMH-Perf, Persistency Service will extend the file size according to the model size change. [S32GPRODP-296347]

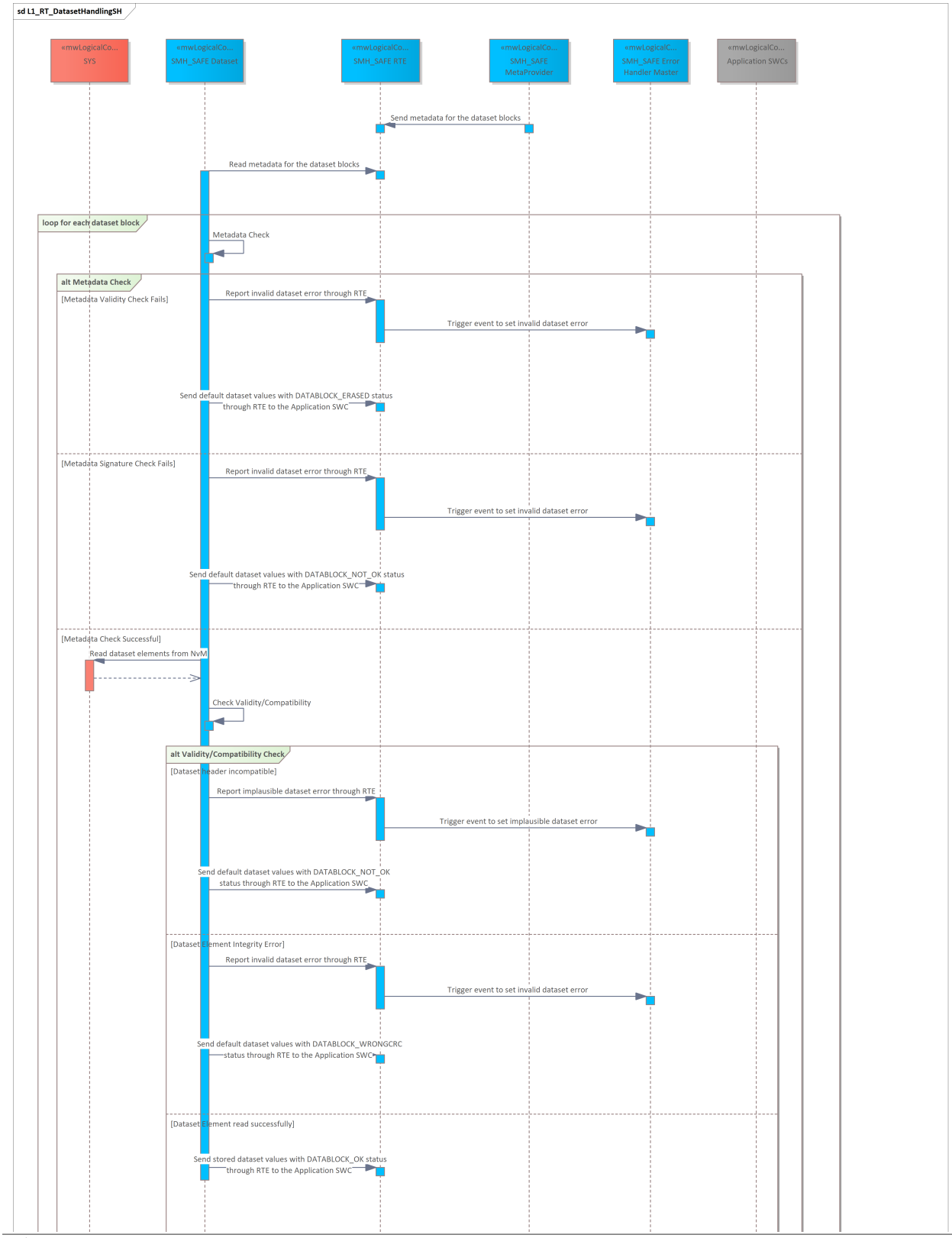


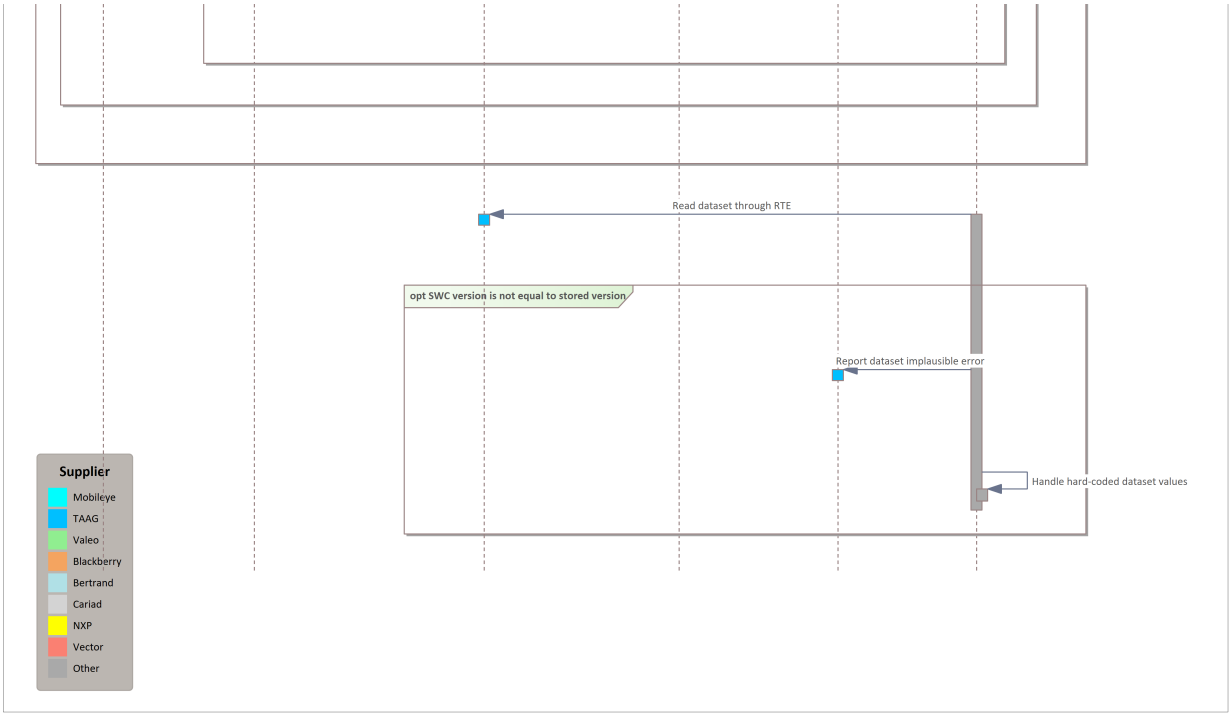

Persistency
loop for each persistent data
Read persistent data from eMMC
opt Data length at the model is greater than file length
Calculate CRC
opt Calculated CRC is equal to stored CRC
Extend stored data according to the size difference
Store updated data
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

6.2.5.2 Datasets

6.2.5.2.1 Safety host Datasets

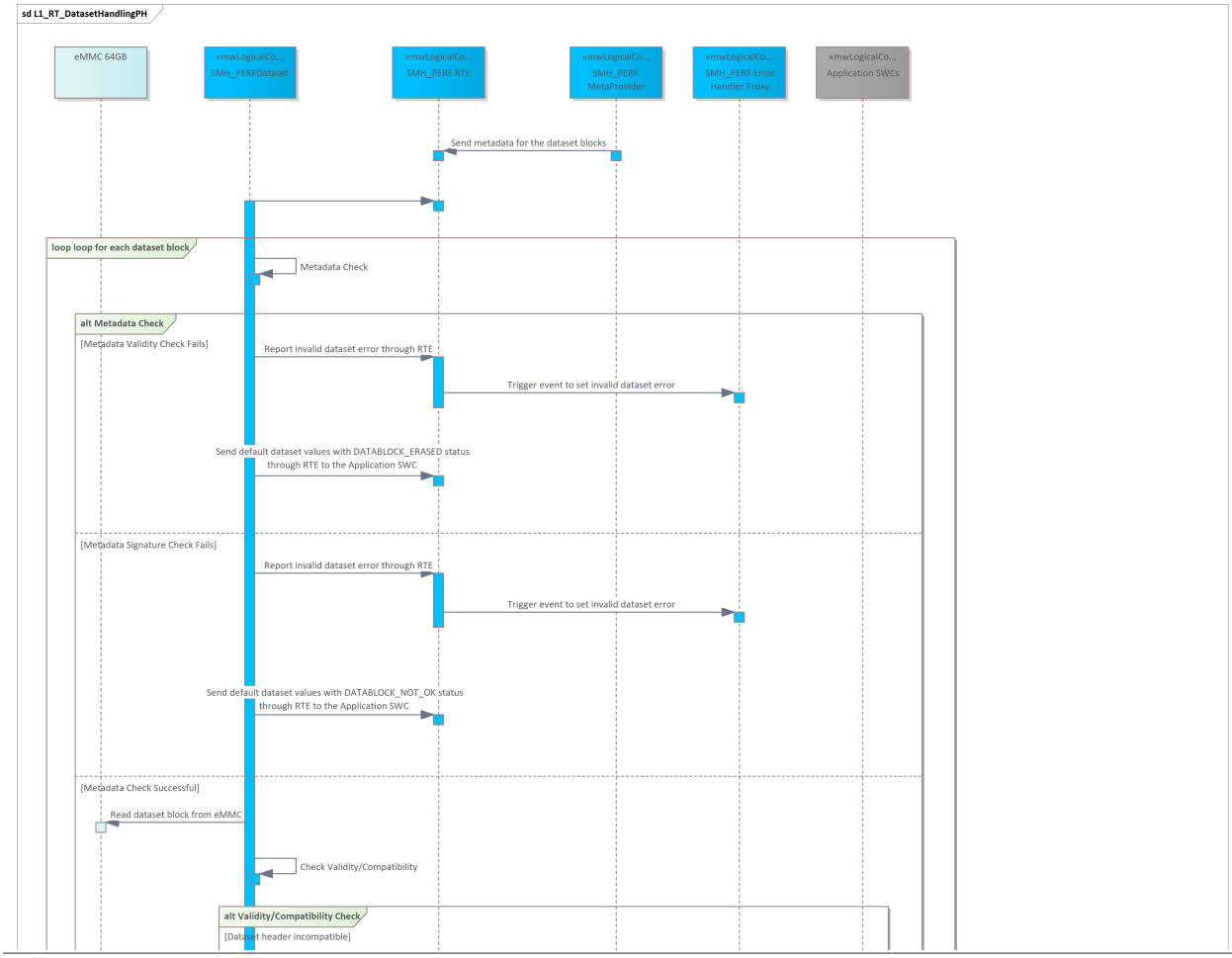
At the init phase, Persistency Service reads datasets from NvM and distributes them to the App SWC's. [S32GPR ODP-296343]

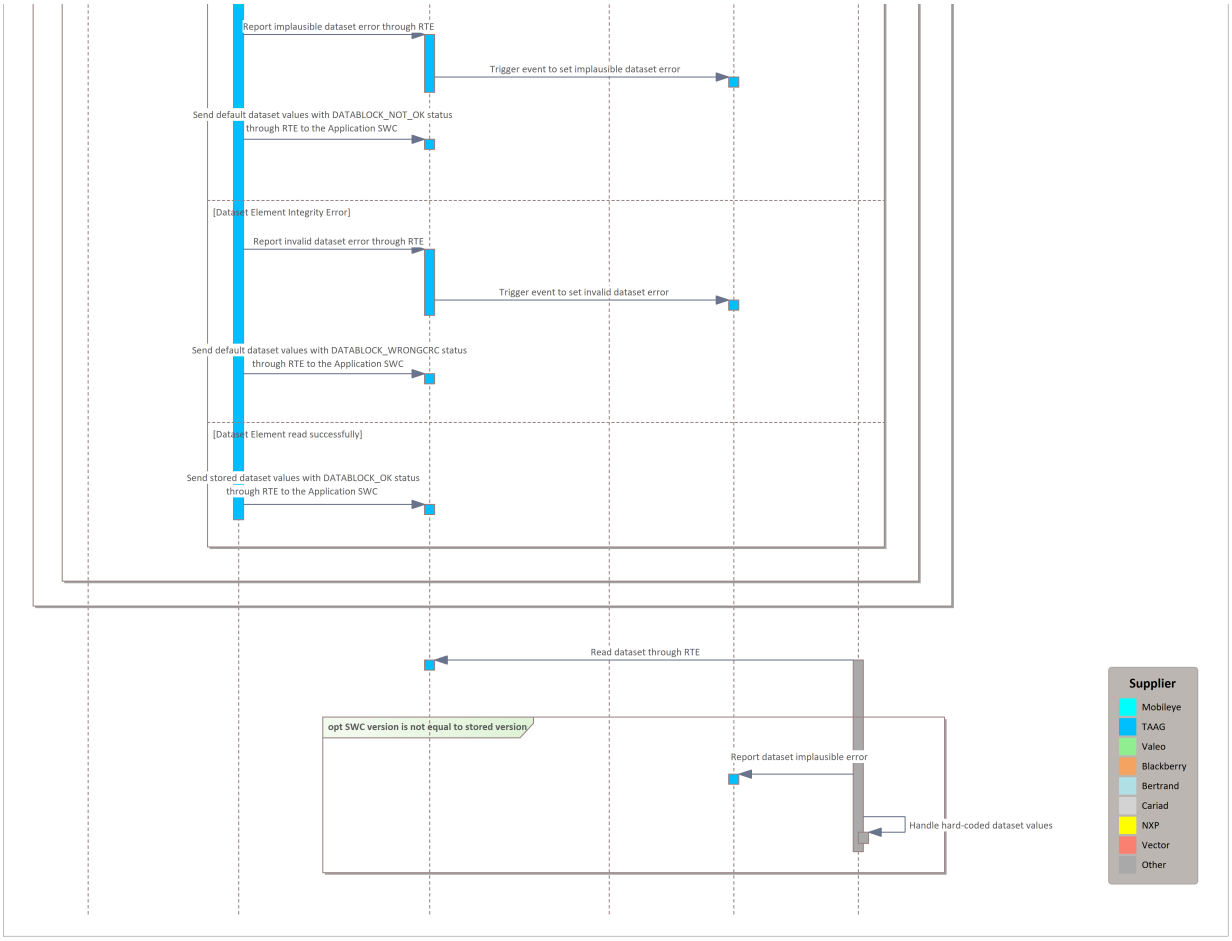




6.2.5.2.2 Performance host Datasets

At the init phase, Persistency Service reads datasets from NvM and distributes them to the App SWC's. **[**S32GPR ODP-296338 **]**

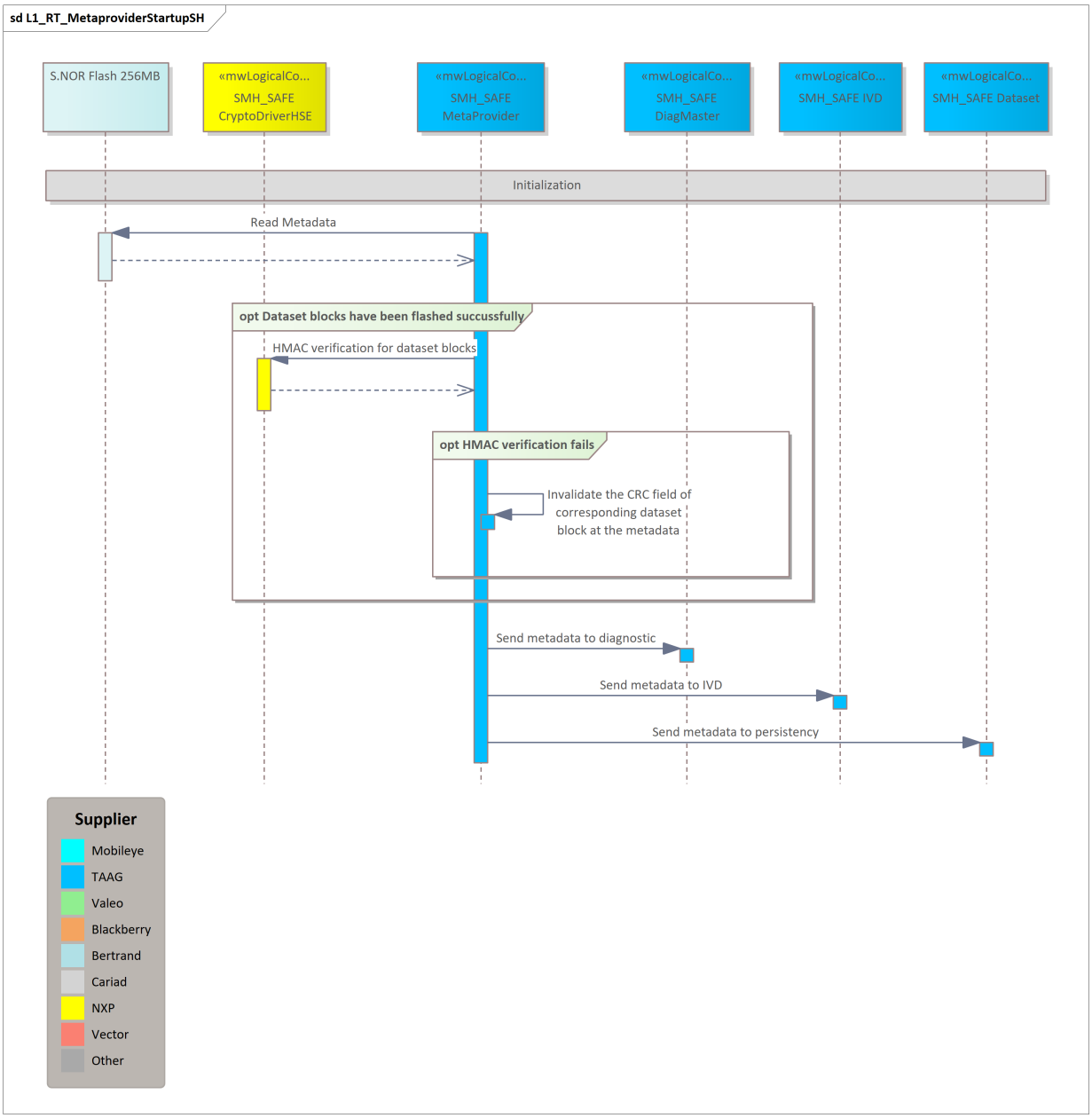




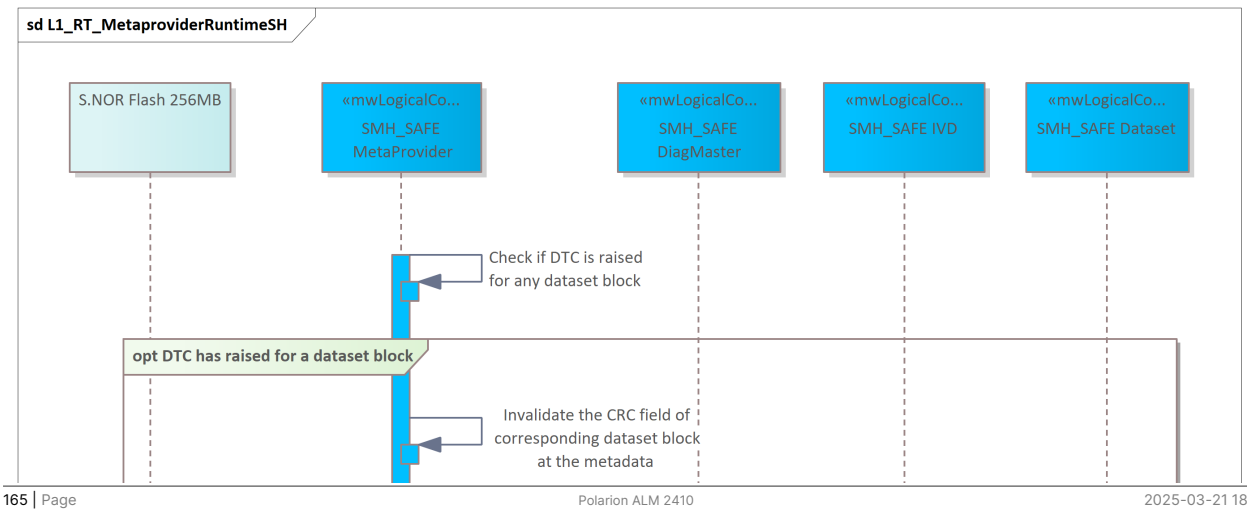
6.2.5.3 Metaprovider

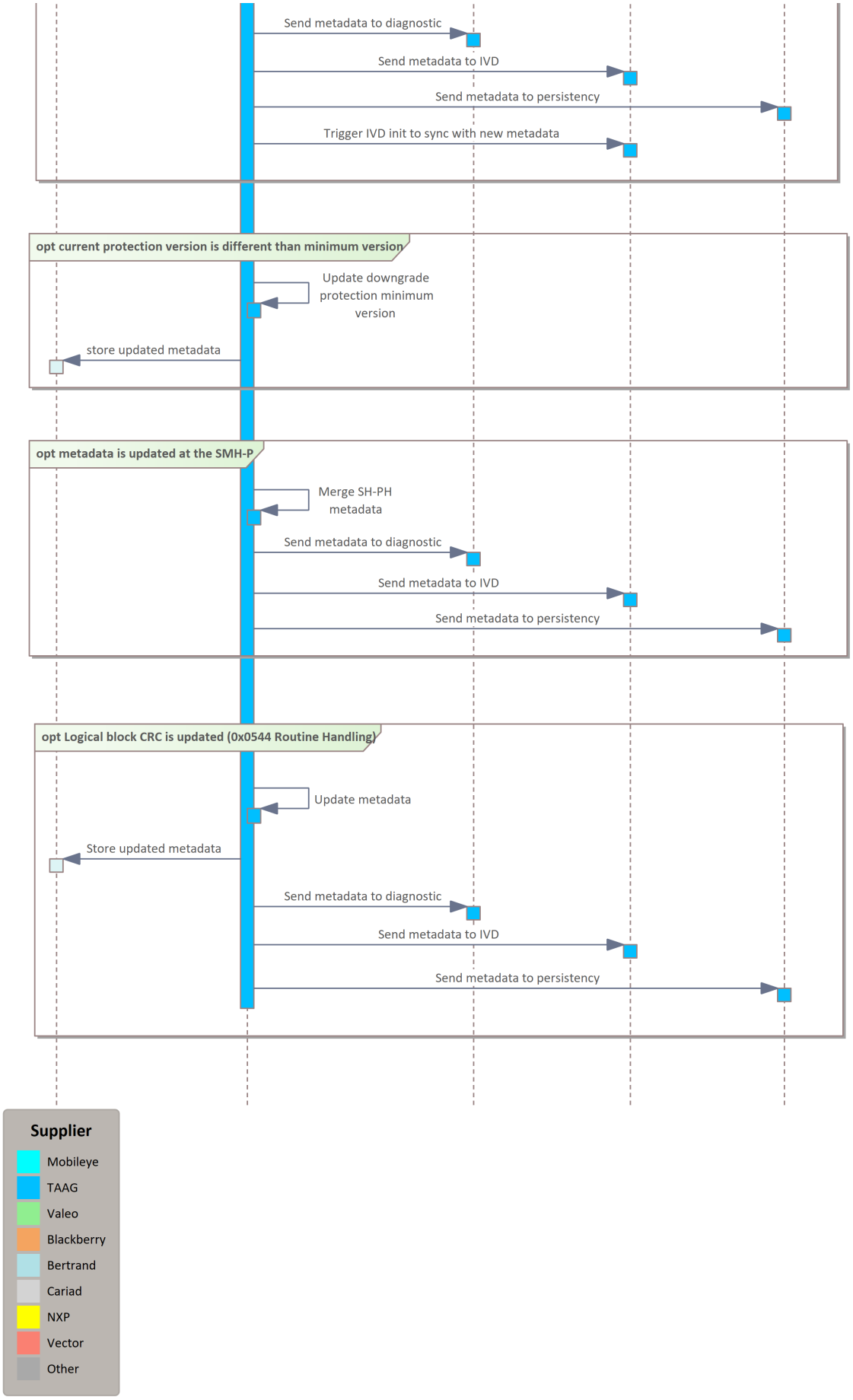
6.2.5.3.1 Safety Host Metaprovider

At the init phase, Metaprovider SWC reads metadata from NvM and distributes information to the SWC's. **[**S32GP RODP-296350 **]**



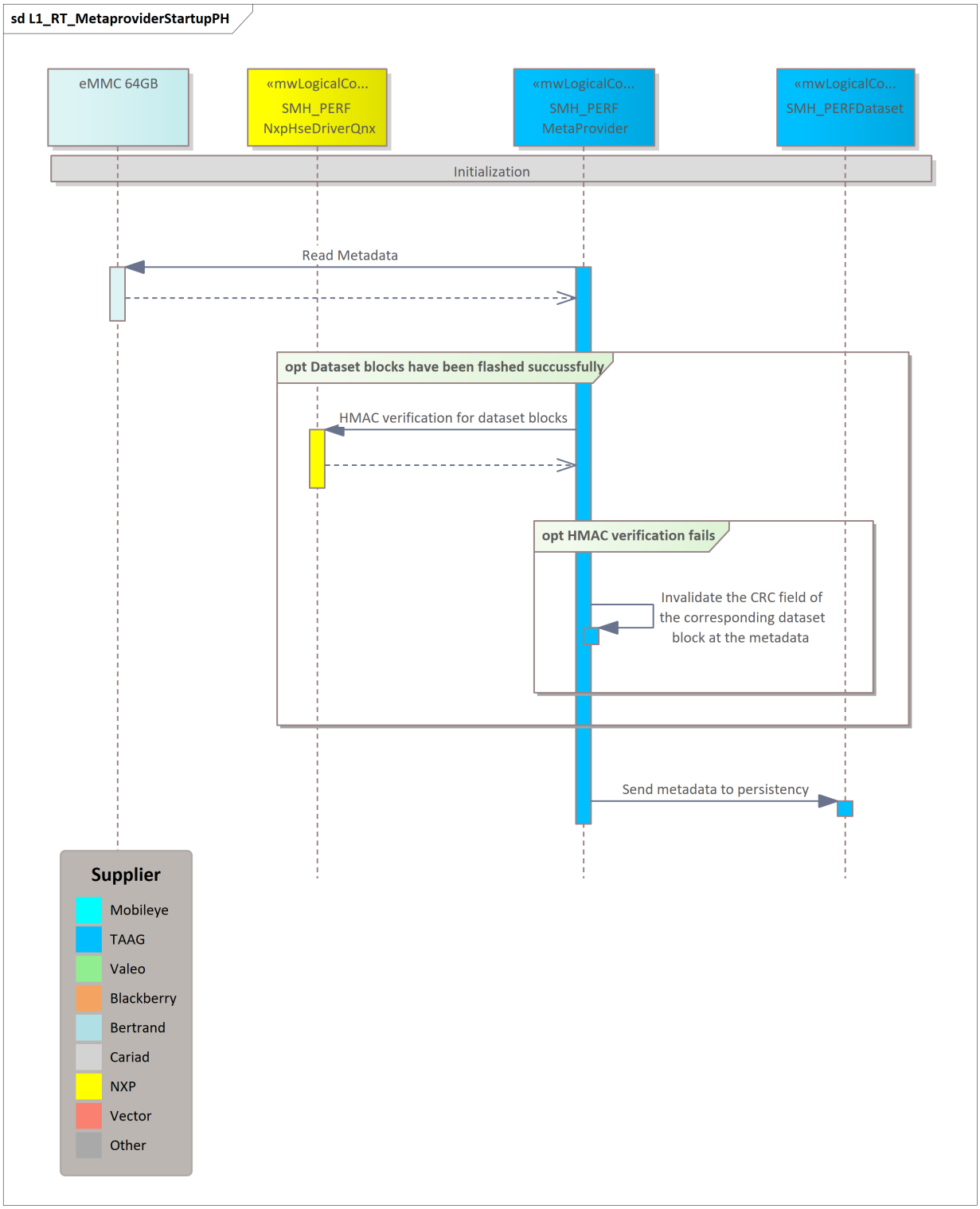
At the runtime, Metaprovider SMH-Safe checks Metaprovider SMH-Perf update status and handles 0x0544 routine requests. **[**S32GPRODP-296368 **]**



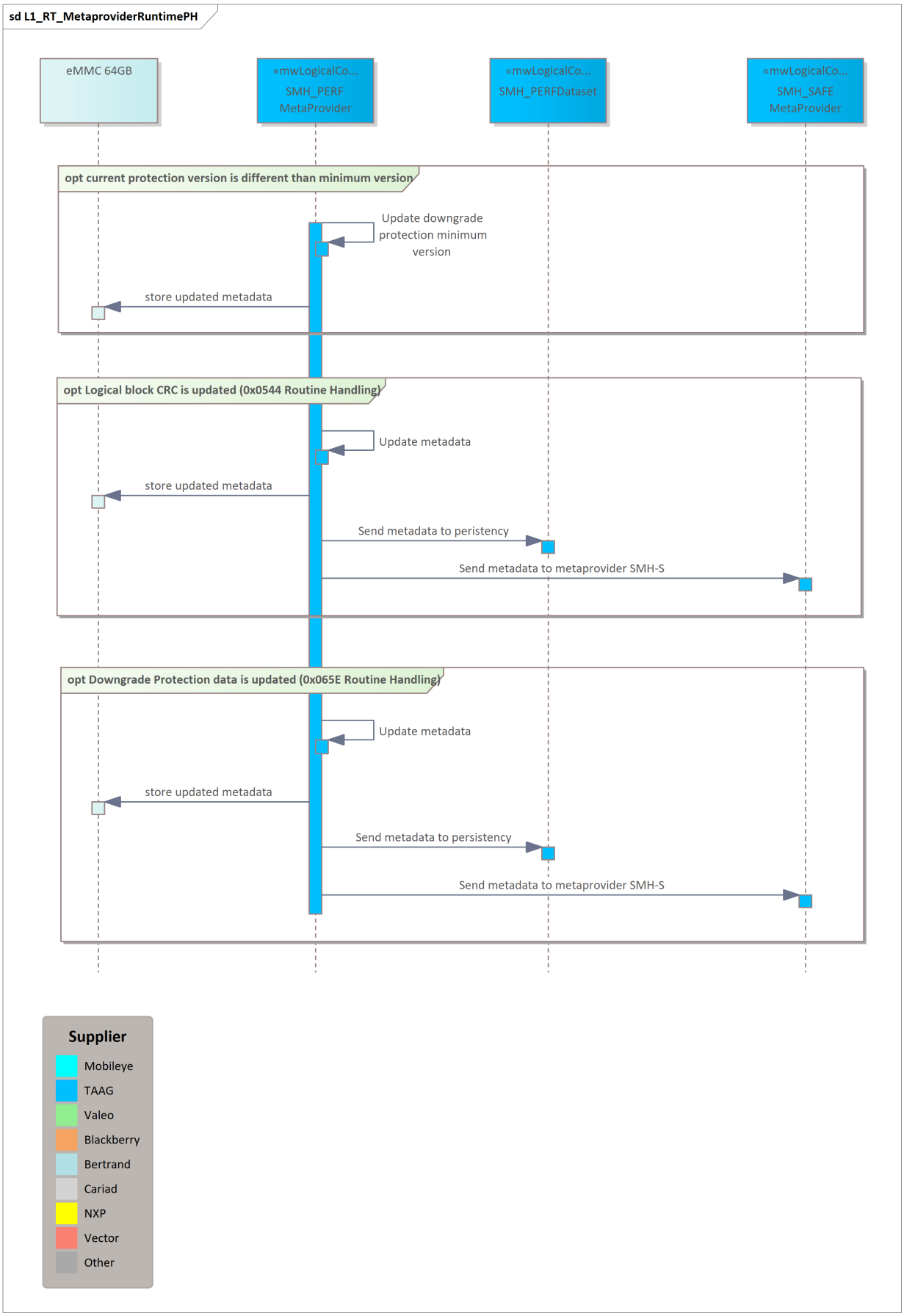

Send metadata to diagnostic
Send metadata to IVD
Send metadata to persistency
Trigger IVD init to sync with new metadata
opt current protection version is different than minimum version
Update downgrade protection minimum version
store updated metadata
opt metadata is updated at the SMH-P
Merge SH-PH metadata
Send metadata to diagnostic
Send metadata to IVD
Send metadata to persistency
opt Logical block CRC is updated (0x0544 Routine Handling)
Update metadata
Store updated metadata
Send metadata to diagnostic
Send metadata to IVD
Send metadata to persistency
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

6.2.5.3.2 Performance Host Metaprovider

At the init phase, Metaprovider SWC reads metadata from NvM and distributes information to the SWC's. [S32GPRODP-296351]



At the runtime, Metaprovider SWC handles 0x0544 and 0x065E routine requests. [S32GPRODP-296369]

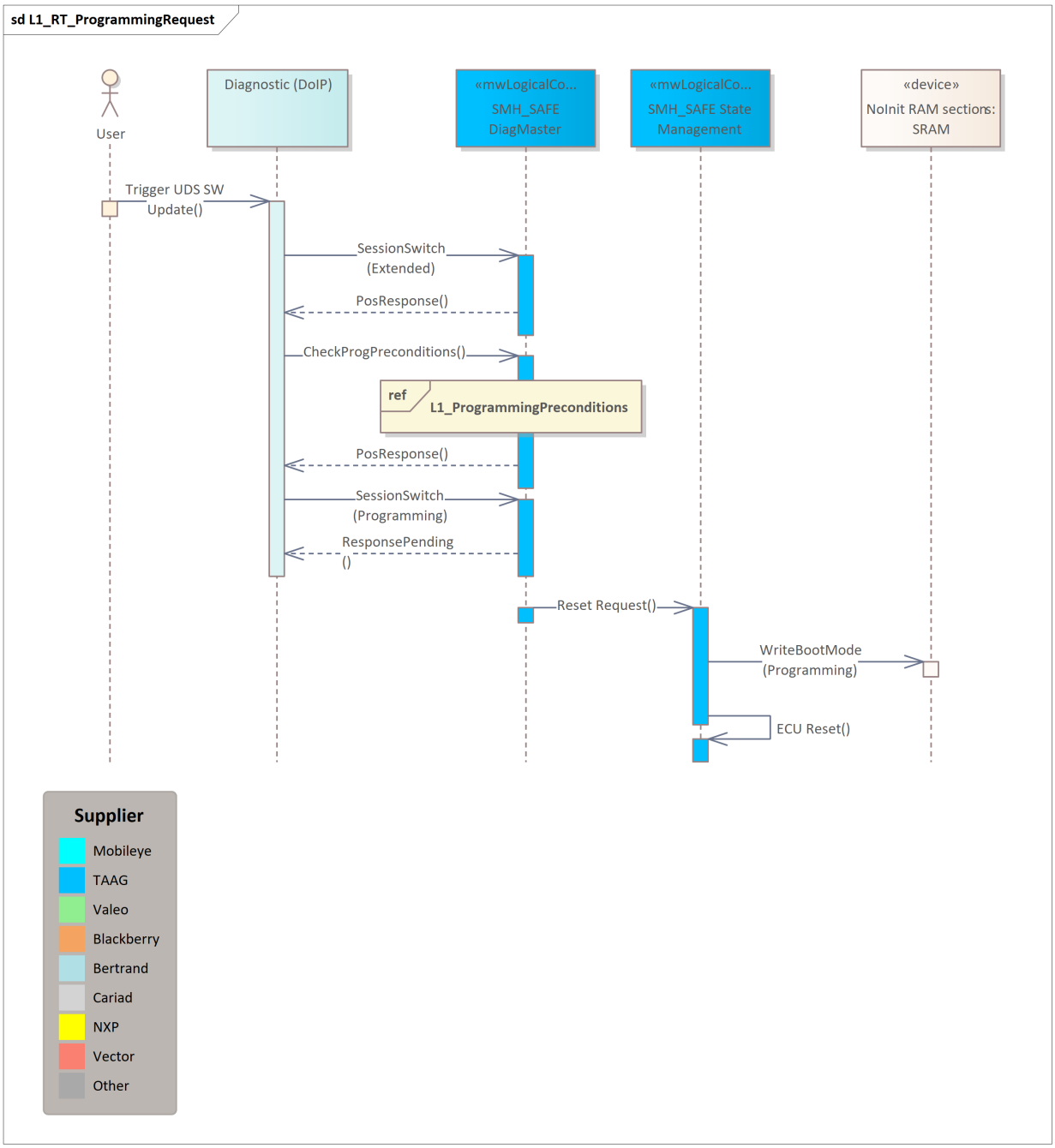

sd L1_RT_MetaproviderRuntimePH
eMMC 64GB
«mwLogicalCo...
SMH_PERF
MetaProvider
«mwLogicalCo...
SMH_PERFDataset
«mwLogicalCo...
SMH_SAFE
MetaProvider
opt current protection version is different than minimum version
Update downgrade protection minimum version
store updated metadata
opt Logical block CRC is updated (0x0544 Routine Handling)
Update metadata
store updated metadata
Send metadata to peristency
Send metadata to metaprovider SMH-S
opt Downgrade Protection data is updated (0x065E Routine Handling)
Update metadata
store updated metadata
Send metadata to persistency
Send metadata to metaprovider SMH-S
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

6.2.6 Software update

6.2.6.1 UDS Software Update

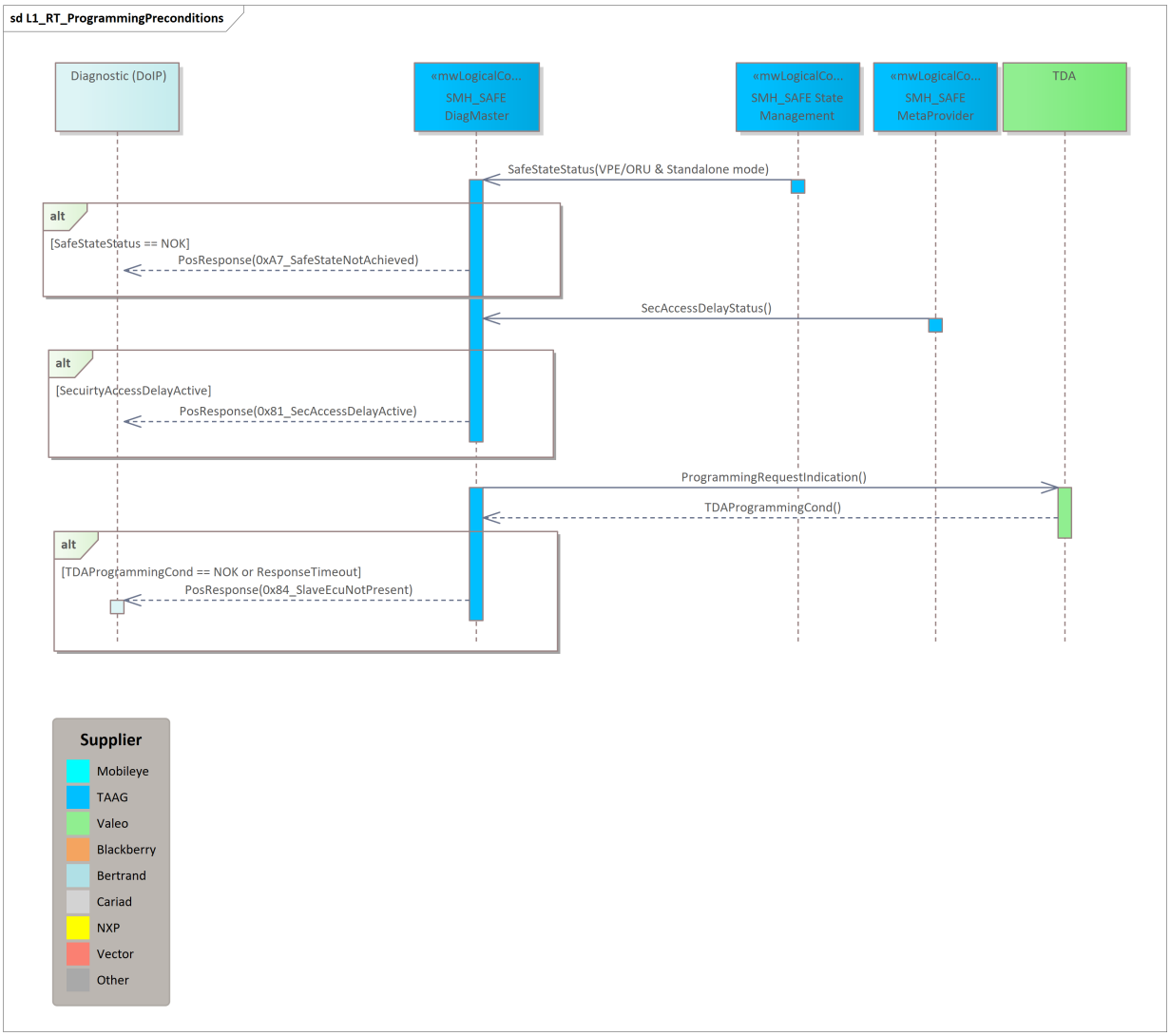
6.2.6.1.1 Programming Request

When the system is in application mode, if the user triggers a UDS Software Update sequence using a PDX container, DiagMaster handles the extended diagnostic session control request, 0x0203 Check Programming Preconditions and programming session request. A response pending is send to the client and ECU reset is performed in order to boot to FlashBootloader. **[**S32GPRODP-296403 **]**



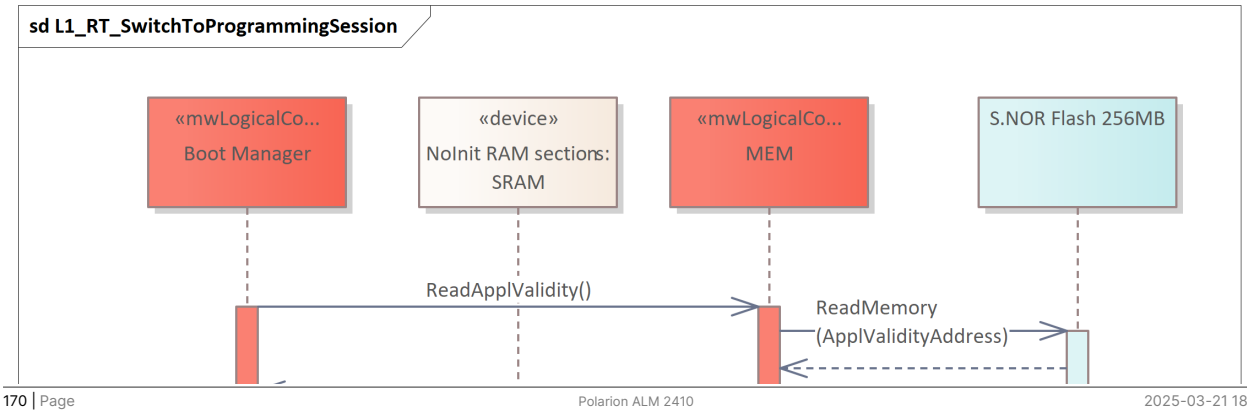
6.2.6.1.2 Programming Preconditions

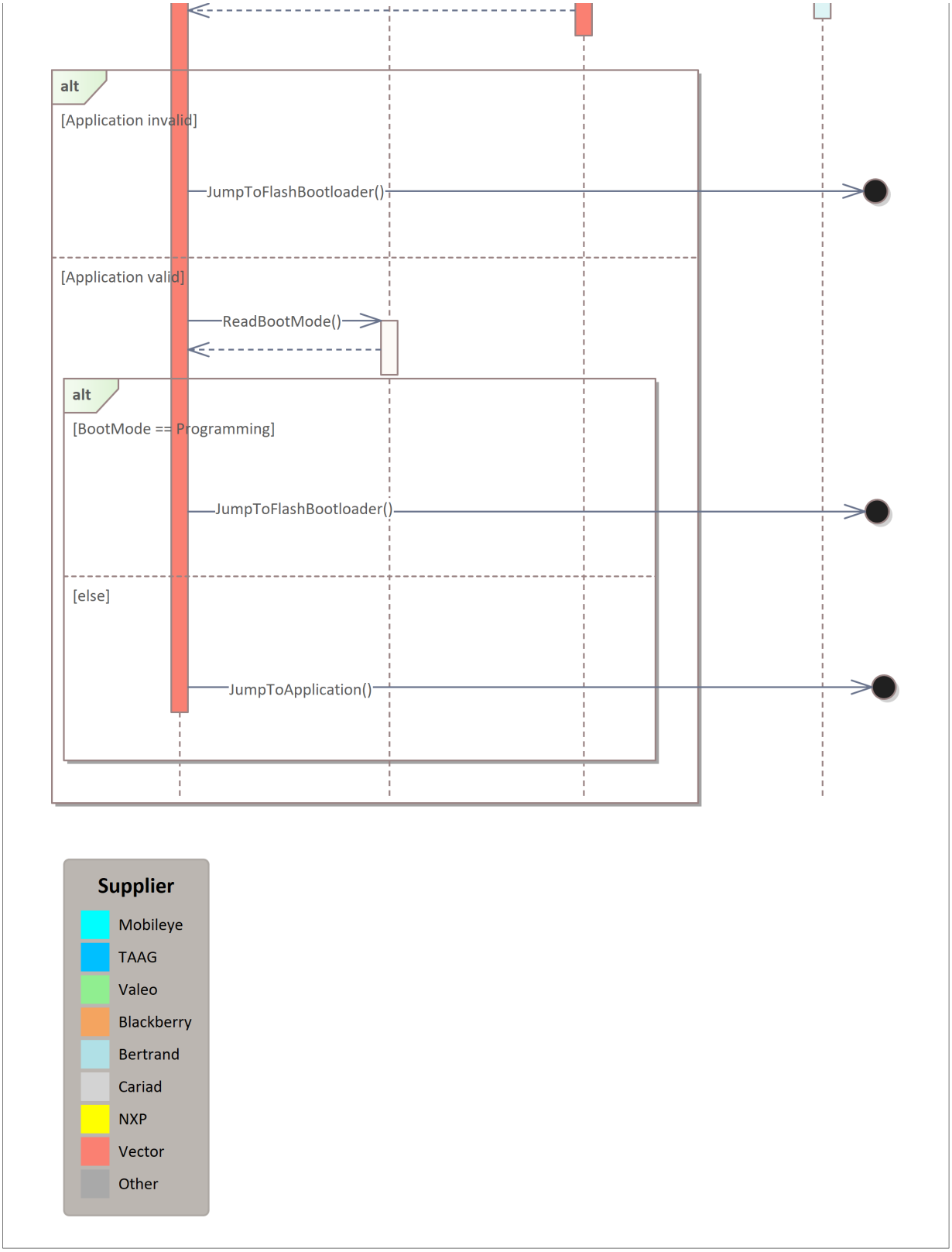
Programming preconditions are checked in DiagMaster when the 0x0203 Check Programming Preconditions routine is executed. **[**S32GPRODP-296405 **]**



6.2.6.1.3 Booting to Flash Bootloader by Boot Manager

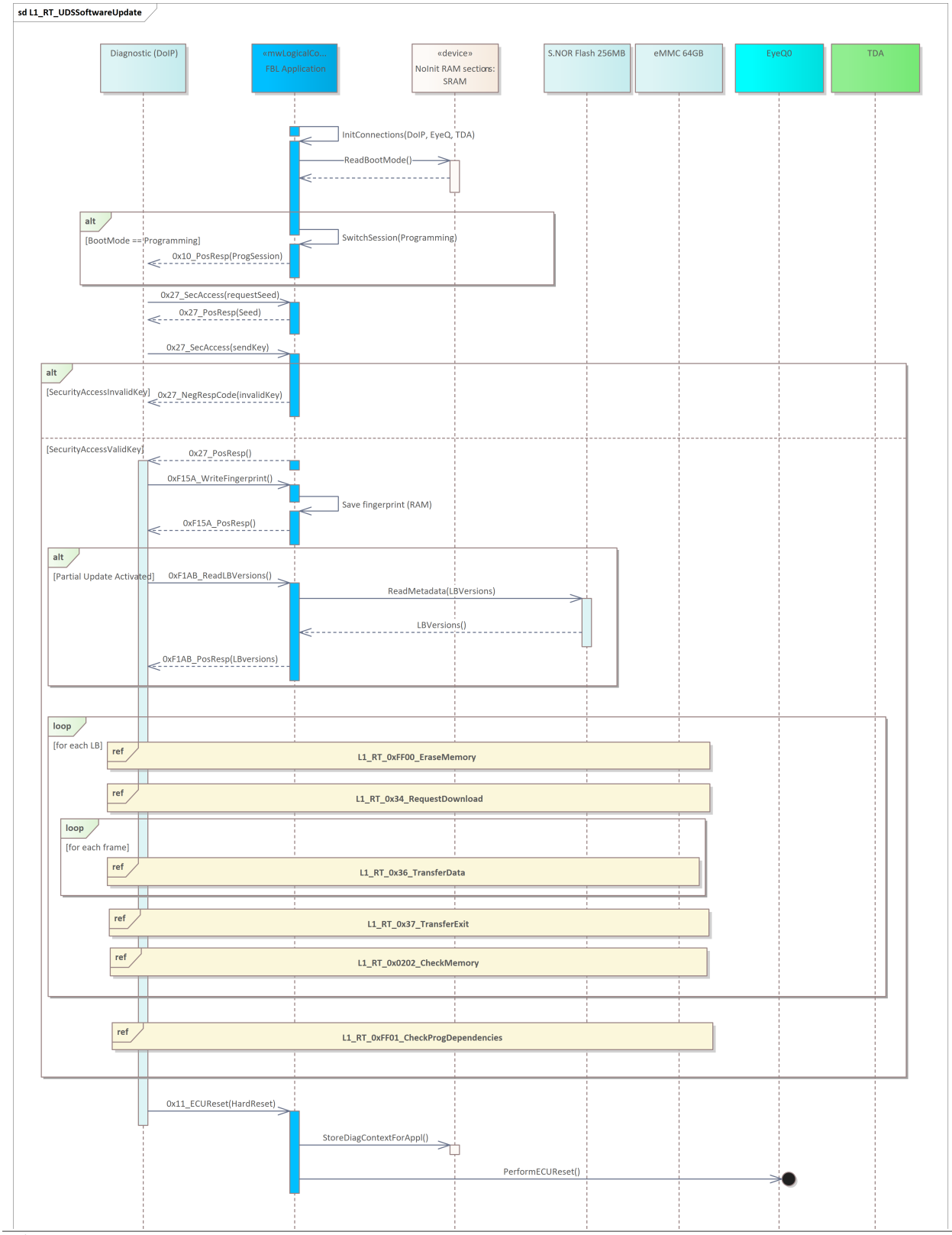
Boot Manager reads the application validity flag from NOR flash and programming request flag from NoInit SRAM section. In case of invalid application or programming request from application, Boot Manager jumps to Flash Bootloader context. **[**S32GPRODP-296407 **]**

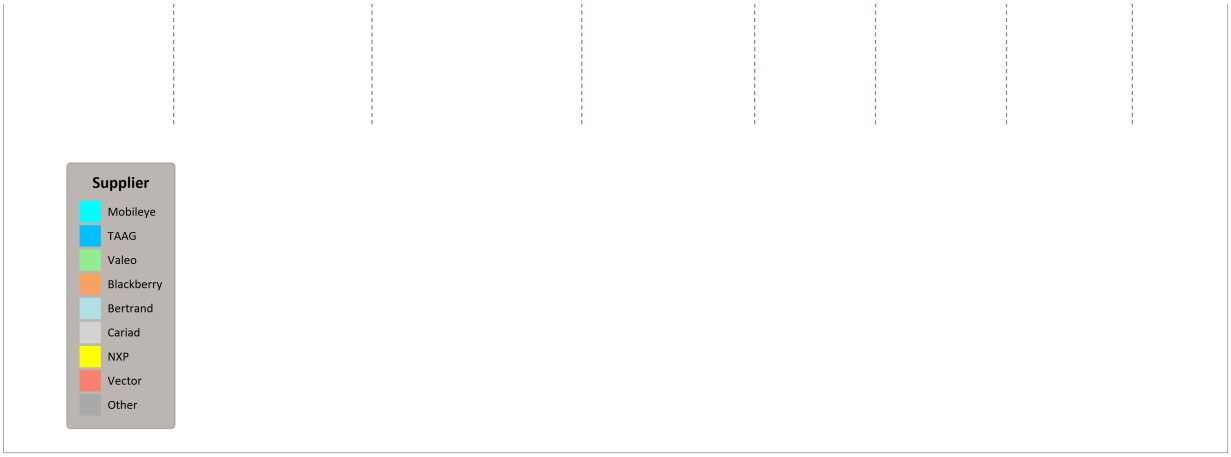



alt
[Application invalid]
JumpToFlashBootloader()
[Application valid]
ReadBootMode()
alt
[BootMode == Programming]
JumpToFlashBootloader()
[else]
JumpToApplication()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

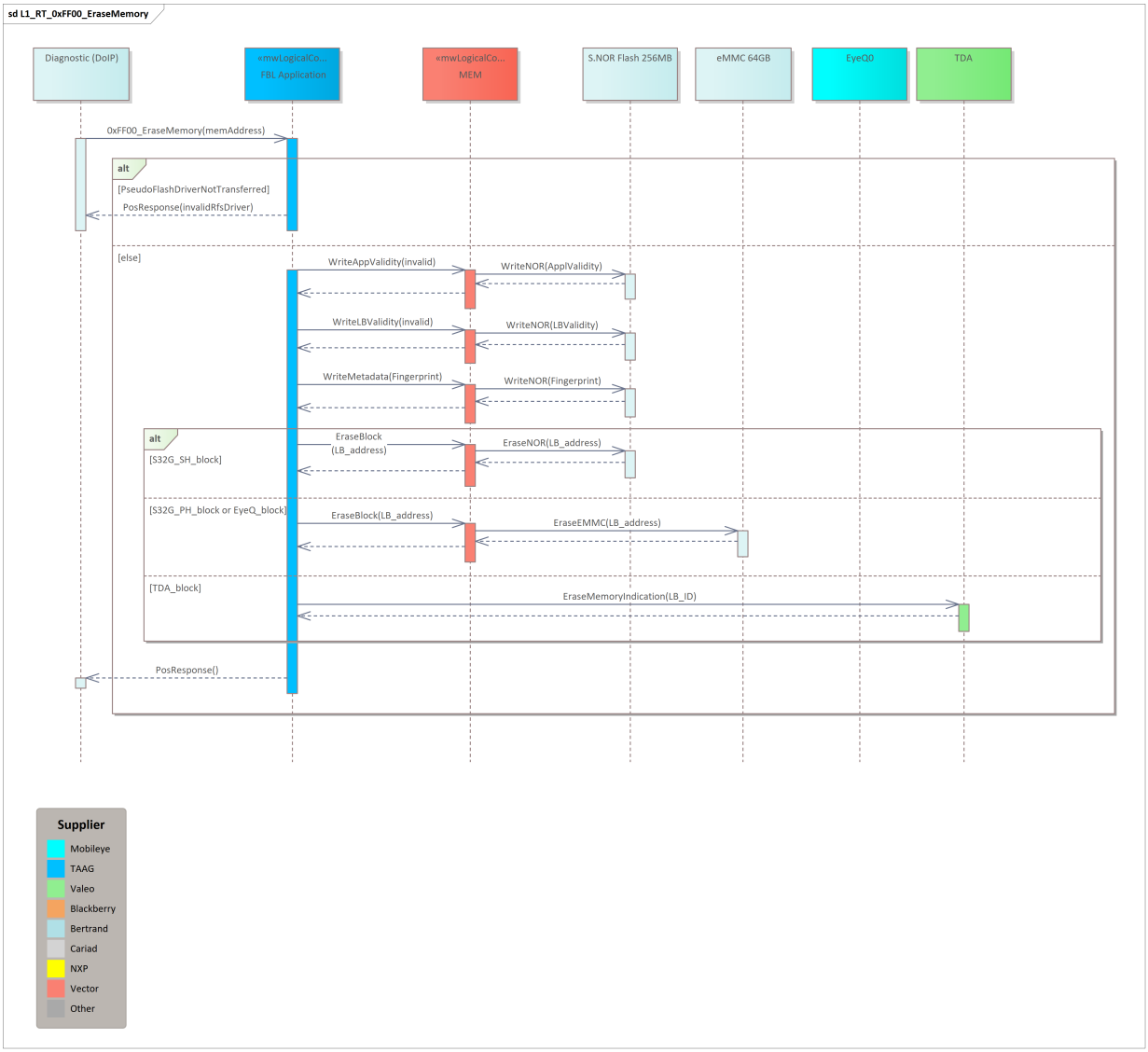
6.2.6.1.4 Overall UDS Software Update Sequence

When Flash Bootloader is started by Boot Manager, it initializes all the ethernet connections for DoIP, EyeQ and TDA communication and then sends the positive response to the diagnostics client in case Flash Bootloader is started due to a programming request. Then Flash Bootloader handles all UDS Software Update sequence requests from the client. **[**S32GPRODP-296417 **]**



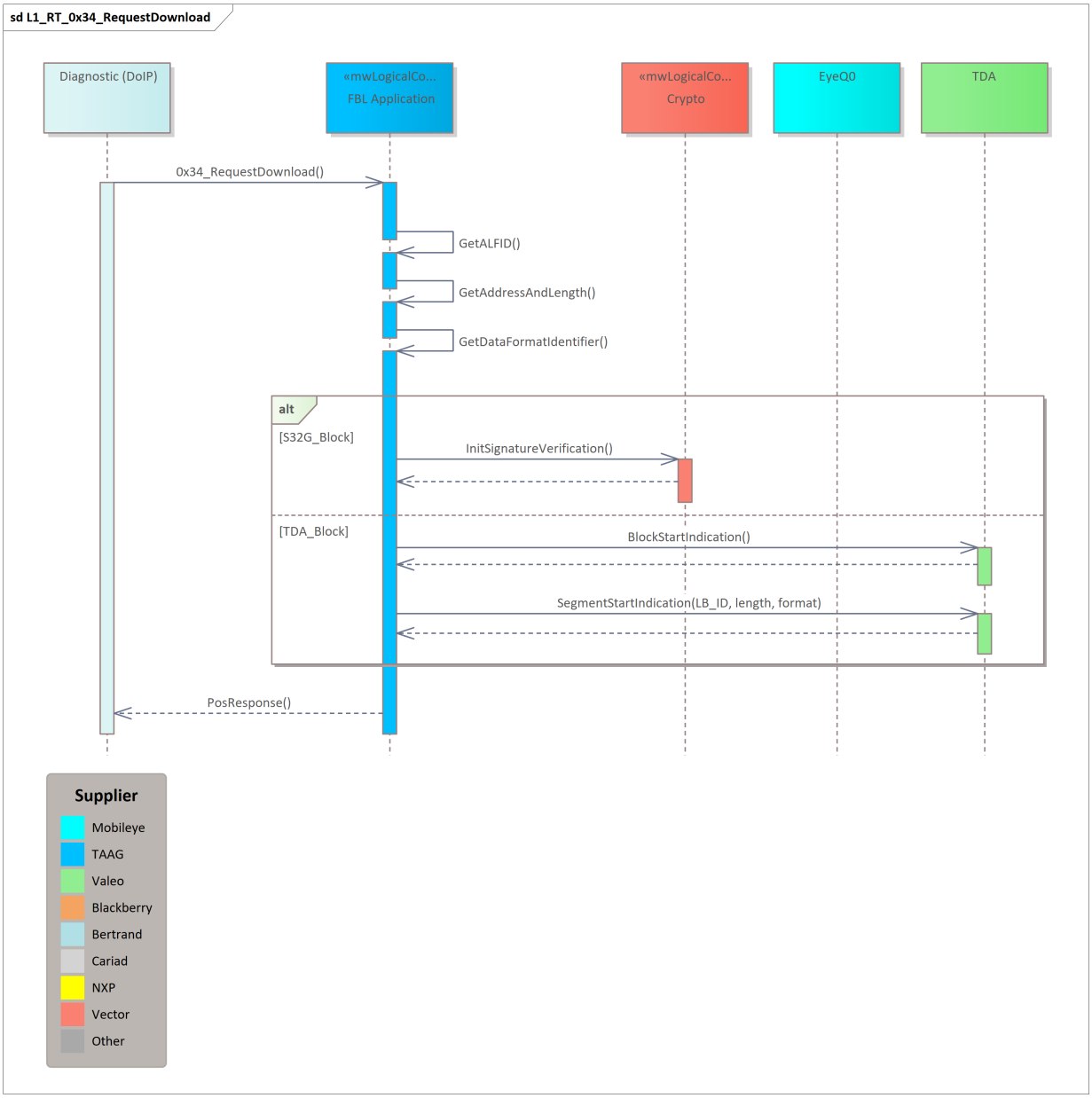


When 0xFF00 EraseMemory routine is executed for a logical block, Flash Bootloader sets application validity and logical block validity to "invalid" and stores the fingerprint in NvM metadata container, and then performs erasing of S32G or EyeQ block. In case of TDA block, Erase Memory request is forwarded to TDA. **[**S32GPRODP-296418 **]**

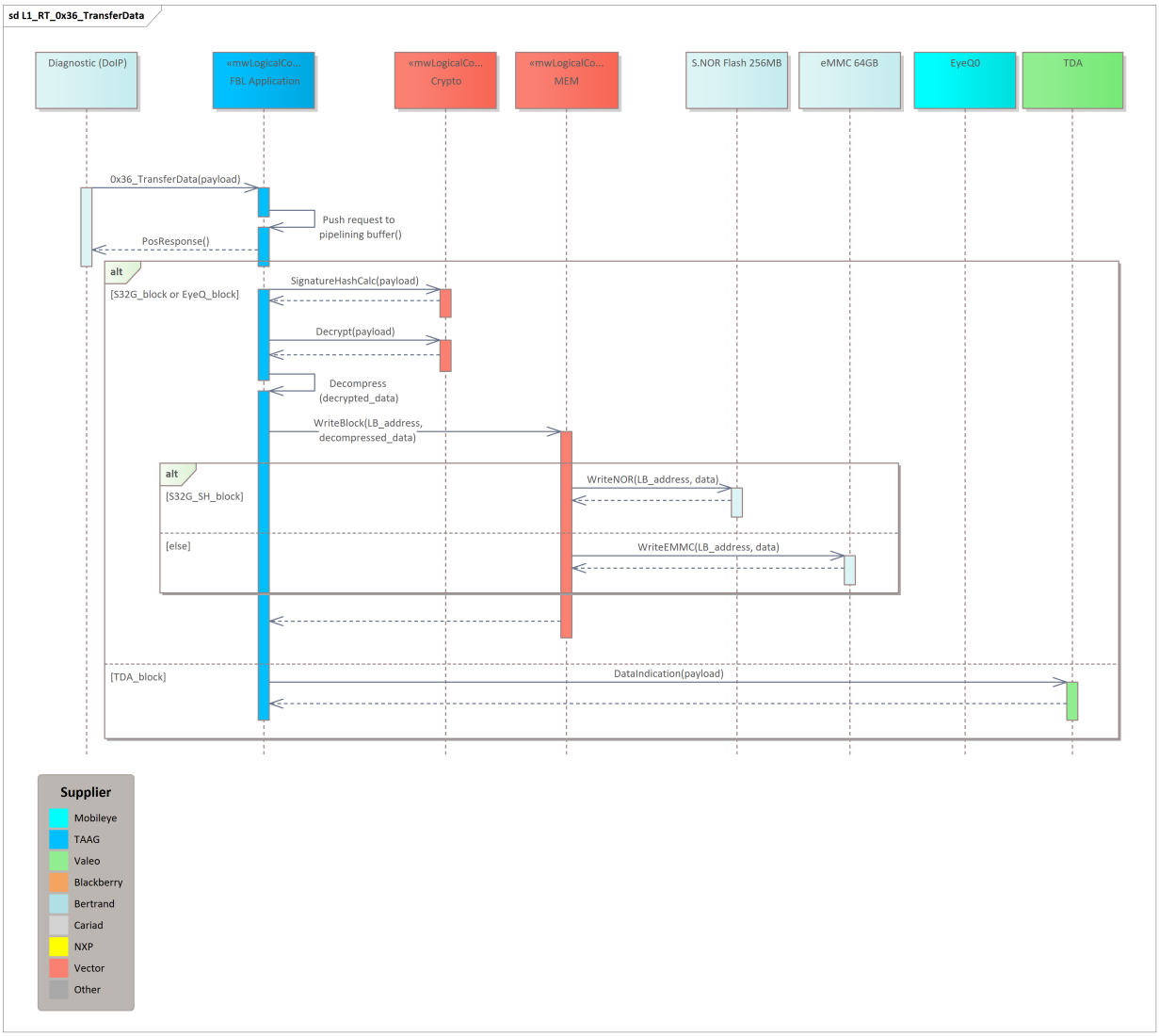


When 0x34 RequestDownload request is received, Flash Bootloader gets the necessary logical block information from diagnostics buffer and initializes the signature hash calculation for S32G or EyeQ block. In case of TDA

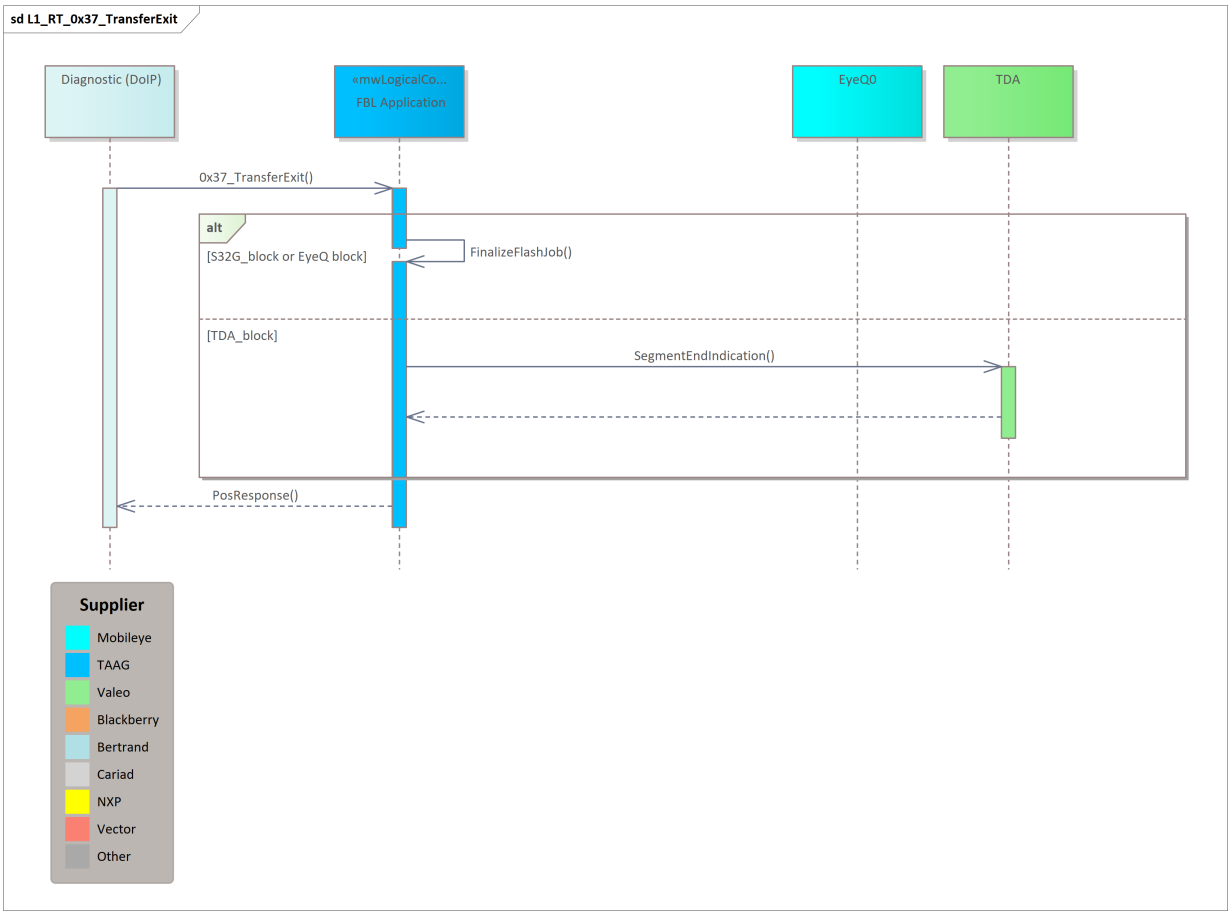
block, request is forwarded to TDA. [S32GPRODP-296420]



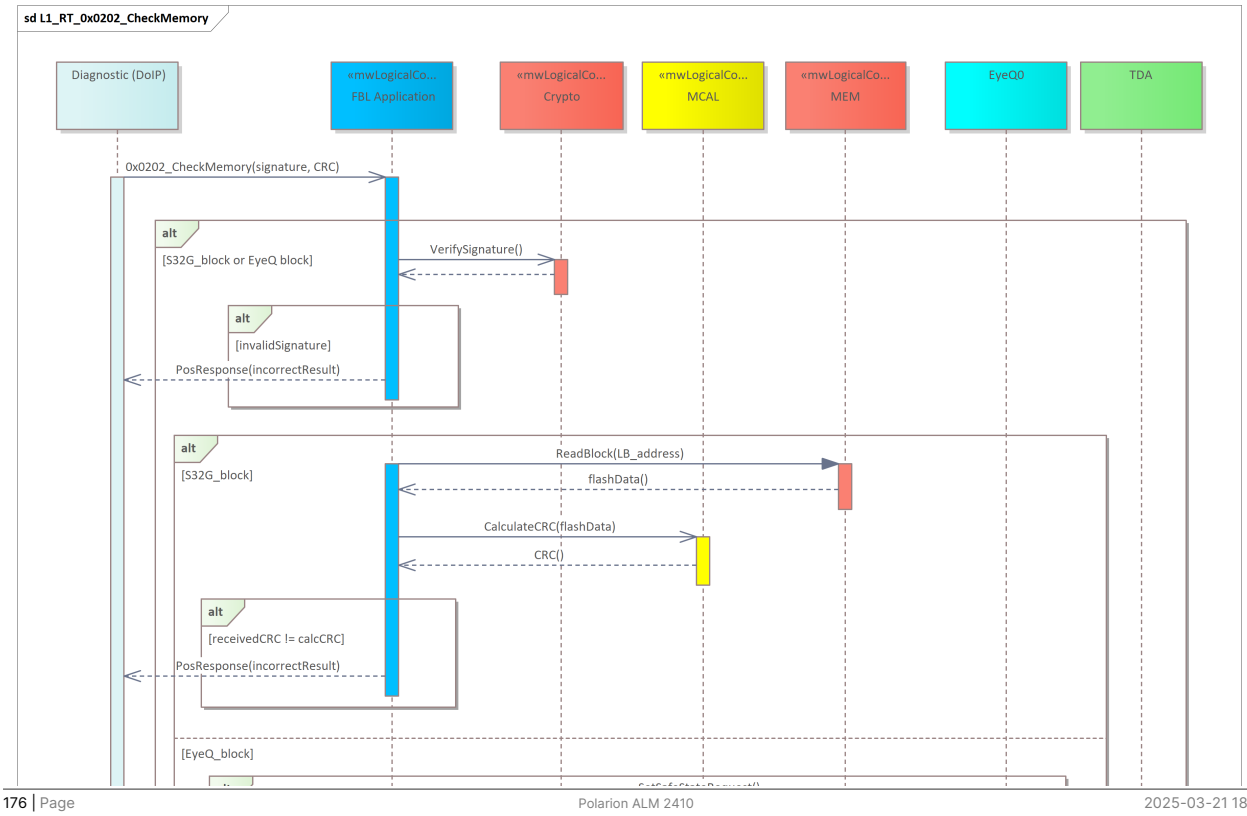
When 0x36 TransferData request is received, Flash Bootloader sends the positive response as soon as possible and then calculates signature hash, process the data (decrypt/decompress) and flashes to memory for S32G or EyeQ blocks. In case of TDA blocks, request is forwarded to TDA. [S32GPRODP-296421]

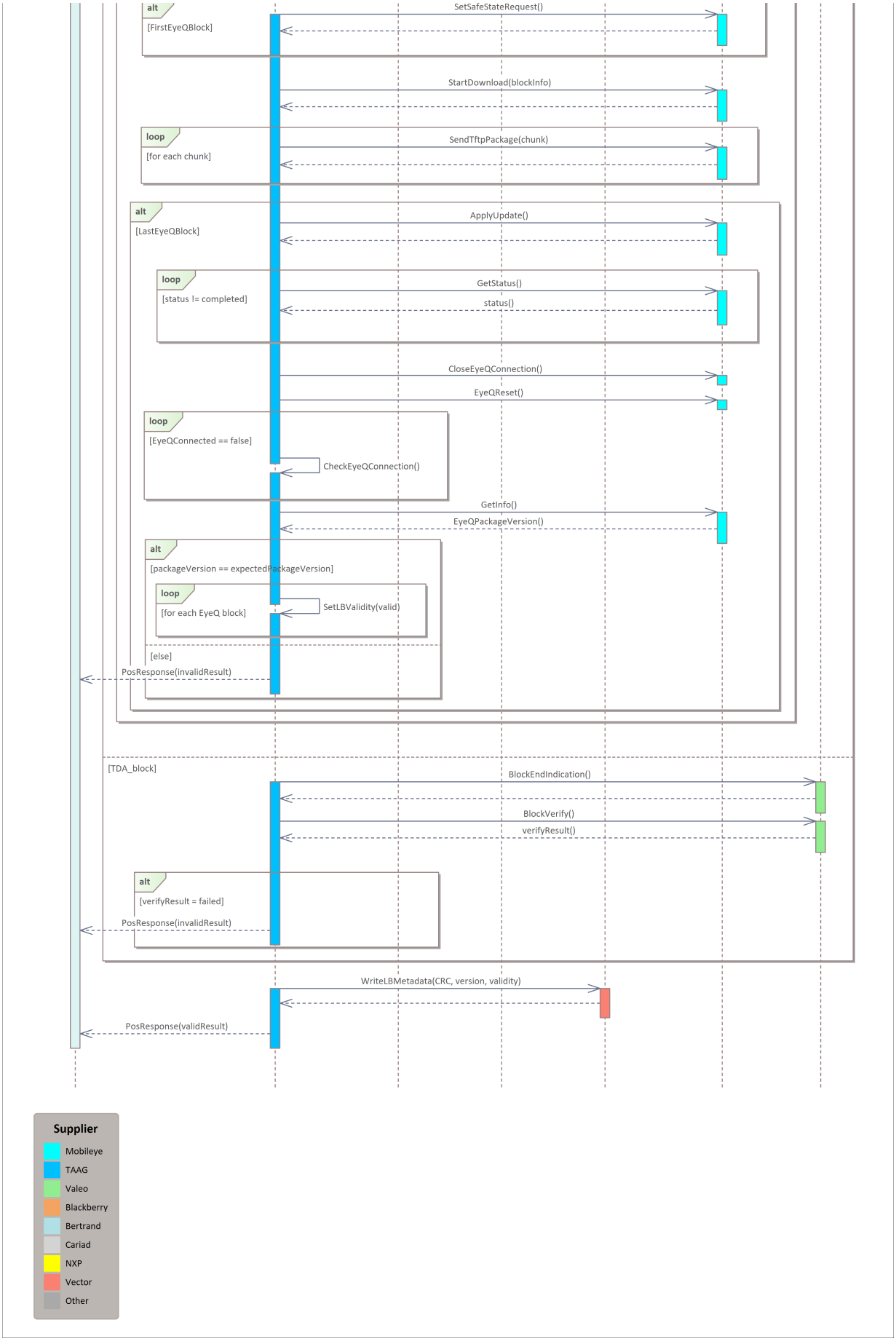


When 0x37 TransferExit request is received, Flash Bootloader finalize the last flash job for S32G or EyeQ blocks. In case of TDA block, request is forwarded to TDA. [S32GPRODP-296416]

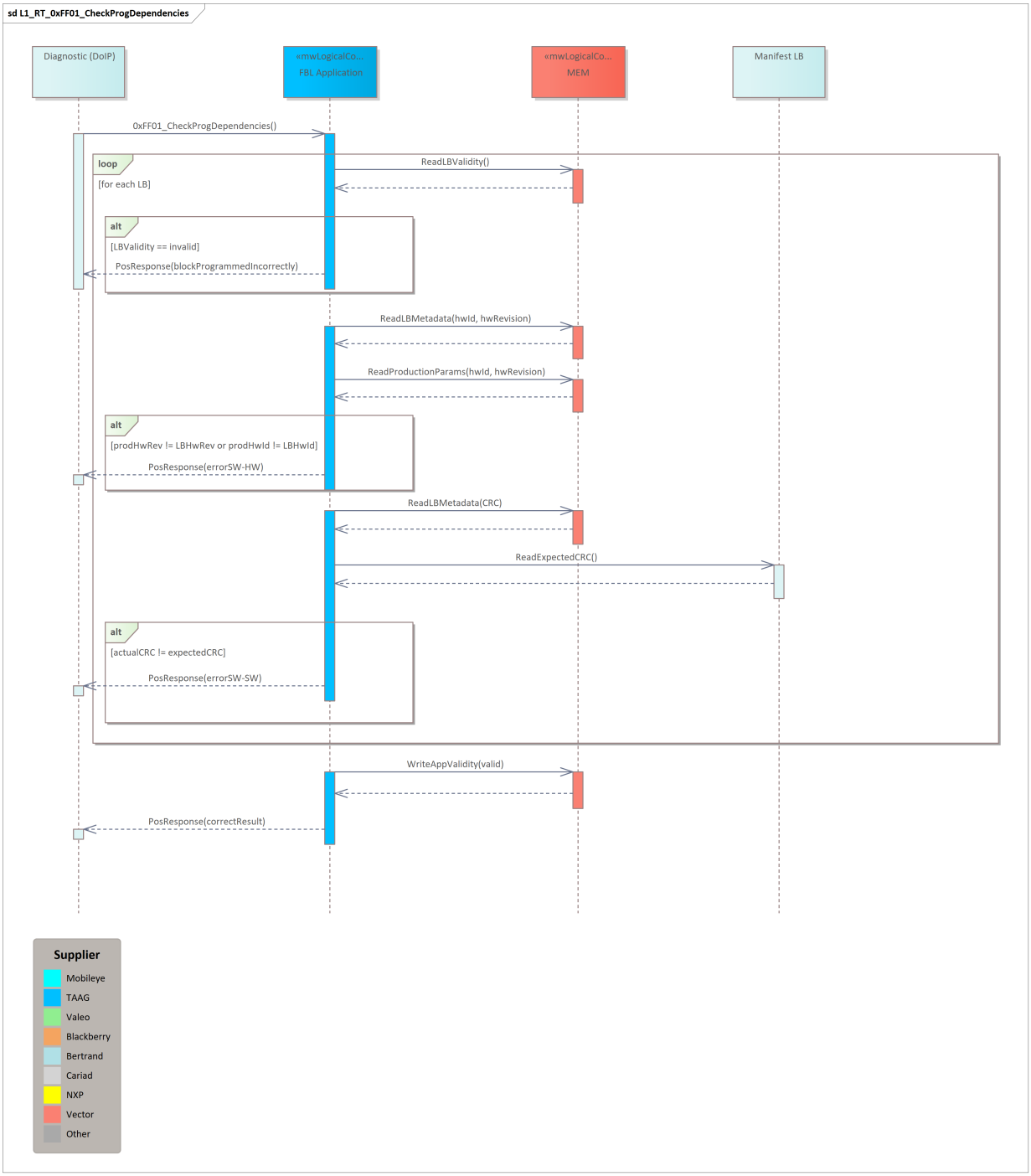


When 0x0202 CheckMemory request is received, Flash Bootloader performs the verification of the signature, calculates and compares the CRC for S32G blocks. For EyeQ blocks, transferring of the images and update of EyeQ are performed at this stage. In case of TDA blocks, request is forwarded to TDA. **[**S32GPRODP-296419 **]**



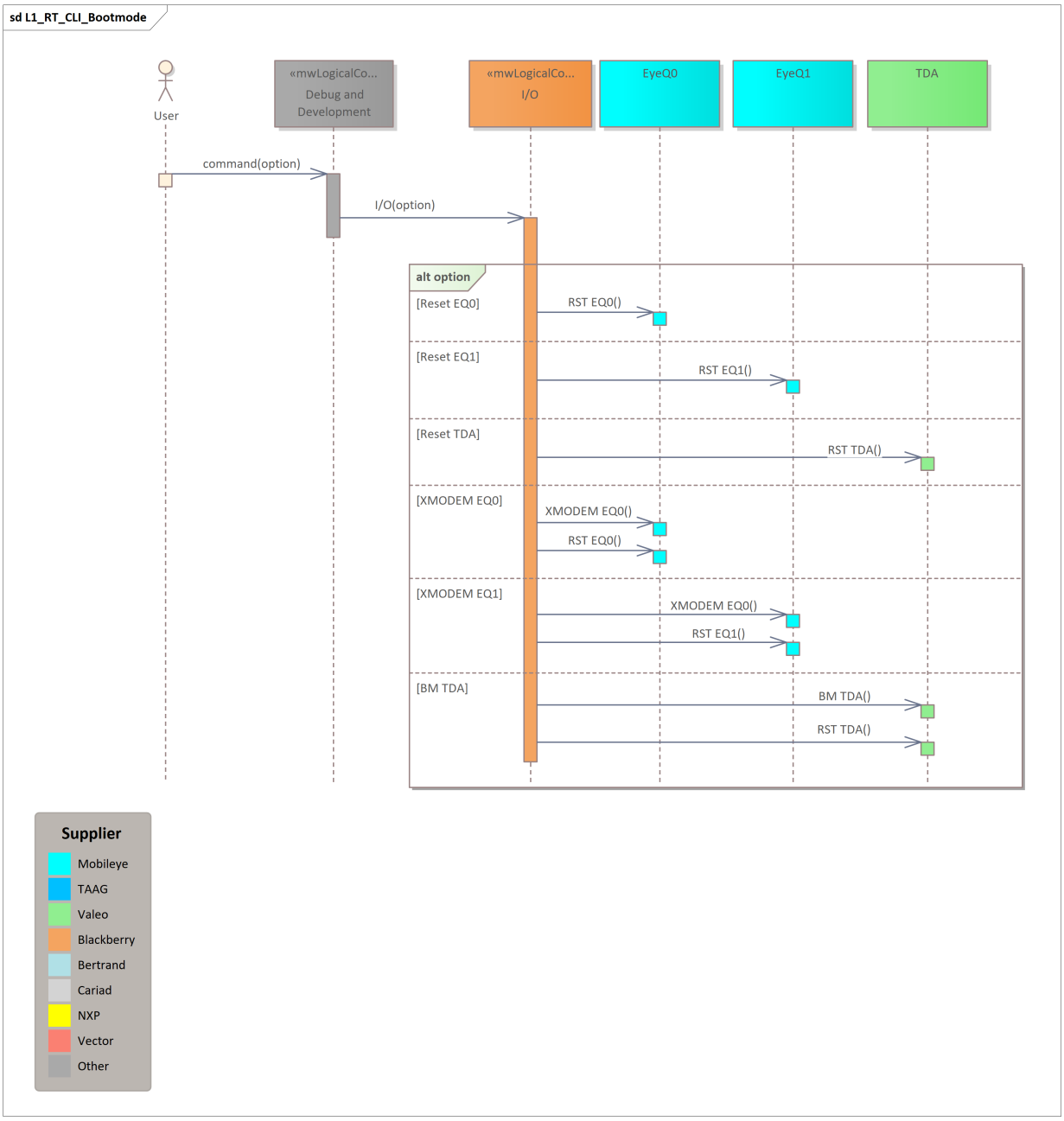

alt
[FirstEyeQBlock]
SetSafeStateRequest()
StartDownload(blockInfo)
loop
[for each chunk]
SendTftpPackage(chunk)
alt
[LastEyeQBlock]
ApplyUpdate()
loop
[status != completed]
GetStatus()
status()
CloseEyeQConnection()
EyeQReset()
loop
[EyeQConnected == false]
CheckEyeQConnection()
GetInfo()
EyeQPackageVersion()
alt
[packageVersion == expectedPackageVersion]
loop
[for each EyeQ block]
SetLBValidity(valid)
[else]
PosResponse(invalidResult)
[TDA_block]
BlockEndIndication()
BlockVerify()
verifyResult()
alt
[verifyResult = failed]
PosResponse(invalidResult)
WriteLBMetadata(CRC, version, validity)
PosResponse(validResult)
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

When 0xFF01 CheckProgrammingDependencies request is received, for each logical block, Flash Bootloader checks if validity flag is "valid", checks if HW revision and ID from metadata and production data are equal and checks if current CRC values (from metadata) and expected CRC values (from manifest logical block) are equal. **[** S32GPRODP-296422 **]**



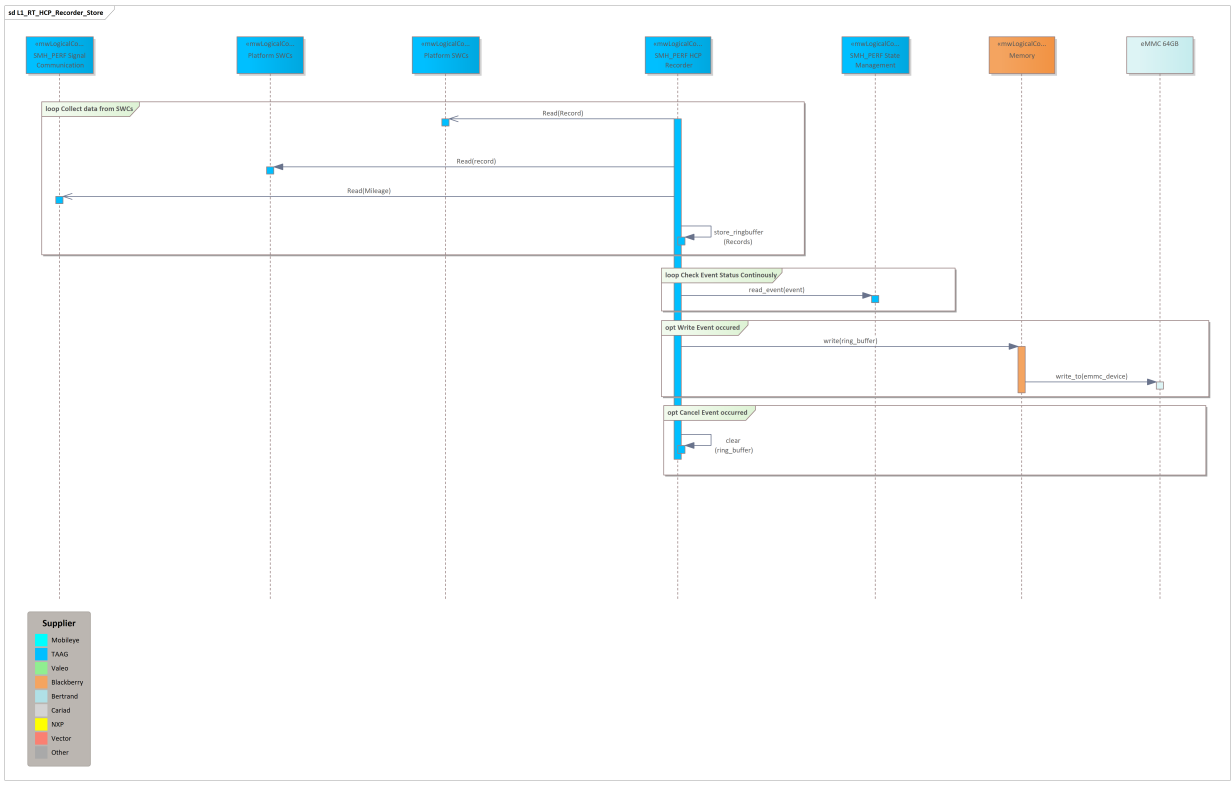
6.2.7 Analysis framework

6.2.7.1 Bootmode CLI

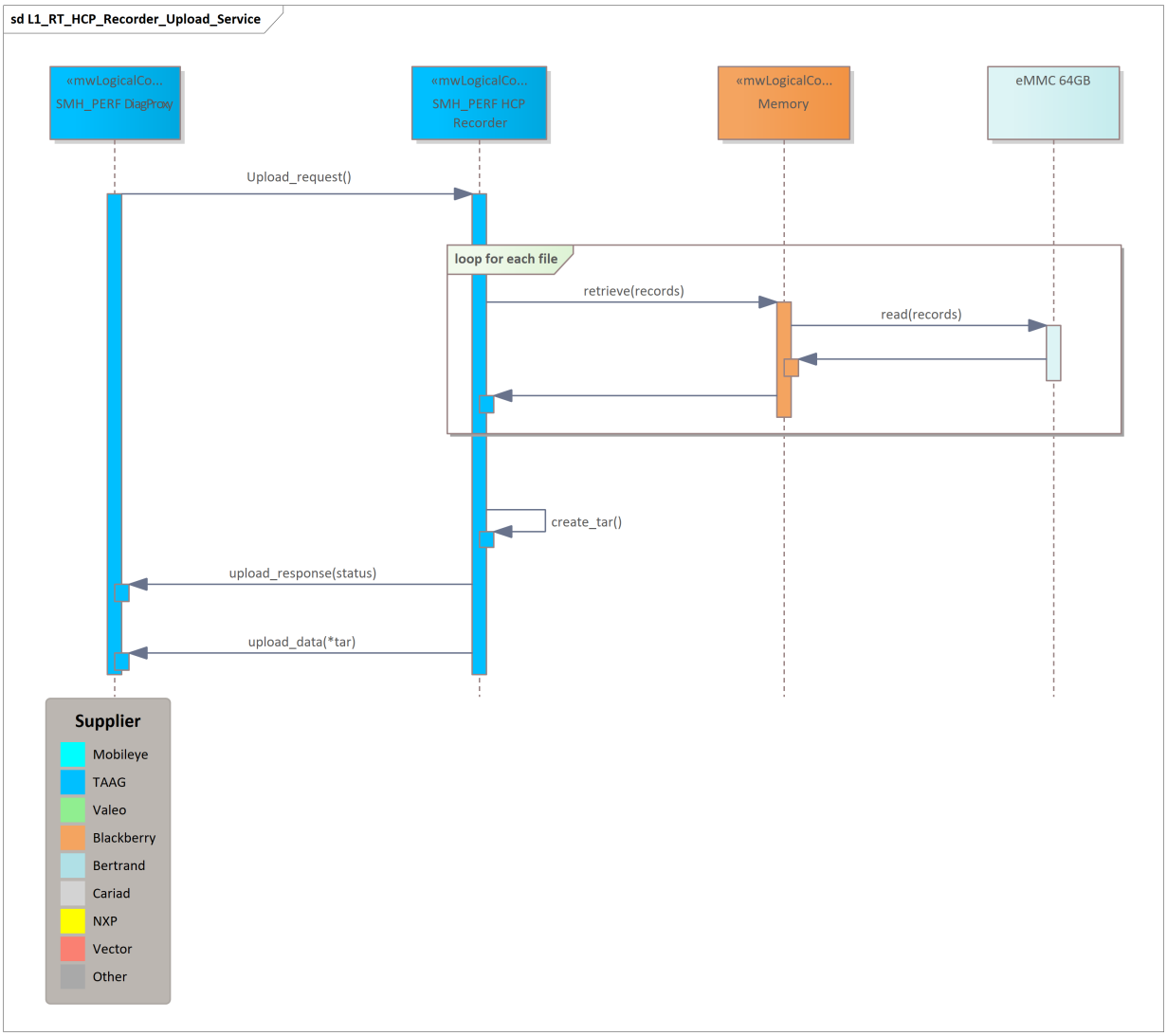


The Bootmode CLI tool is accessible on SMH_PERF through the QNX Korn Shell and offers different command options to the user. By selecting the command option, the user can control the hardware boot mode and reset pins of the connected SoC's. [S32GPRODP-296179]

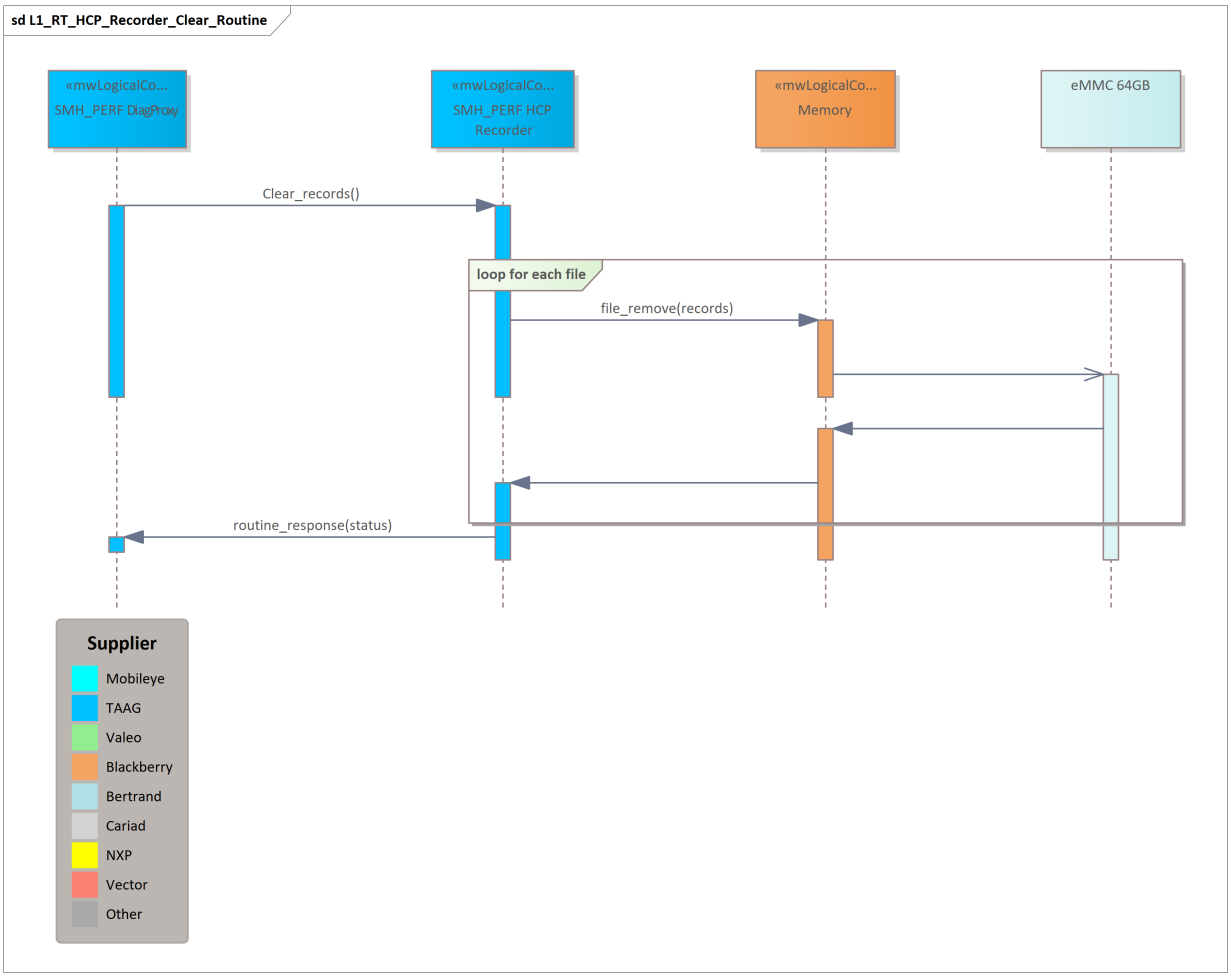
6.2.7.2 HCP recorder



Above diagram explains how does HCP Recorder works during runtime and shutdow. During normal operation, HCP Recorder SWC collects required data from SWCs and store it in a internal ring buffer without storing persistently. It continously checks ECUSM if any shutdown or reset event triggered. If so, it starts to write data into eMMC. **[**S32GPRODP-296774 **]**



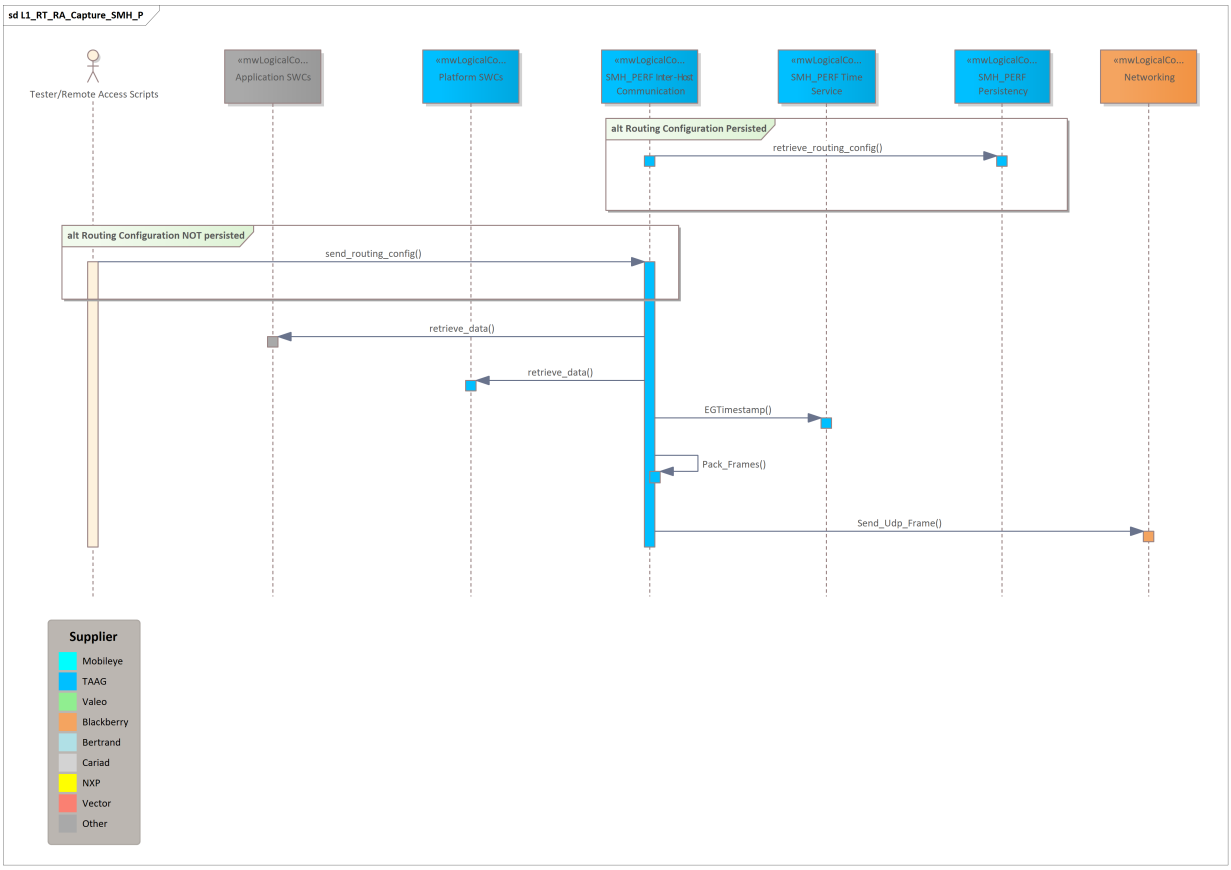
Above diagram explains how does HCP recorder uploads records via diagnostics. A upload diagnostic request is propogated to HCP Recorder SWC by Diagnostic Proxy with a certain sequence, checking preconditions, preparing upload and sending the data. **[**S32GPRODP-296776 **]**



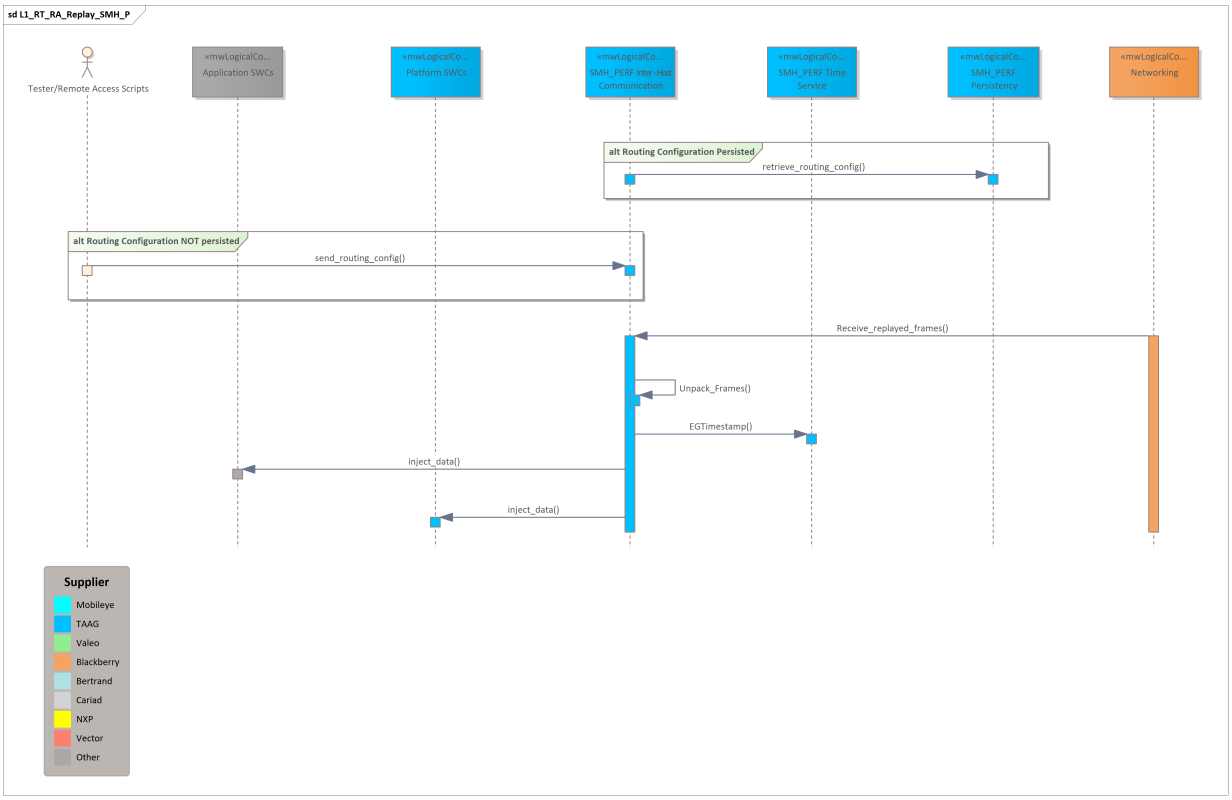
Above diagram explains how does HCP recorder clears it records with a trigger from diagnostics. A clear diagnostic request is propogated to HCP Recorder SWC by Diagnostic Proxy with a certain RID, afterwards HCP Recorder clears all persistent data it stores.

6.2.7.3 Remote access

6.2.7.3.1 Remote access on Performance host

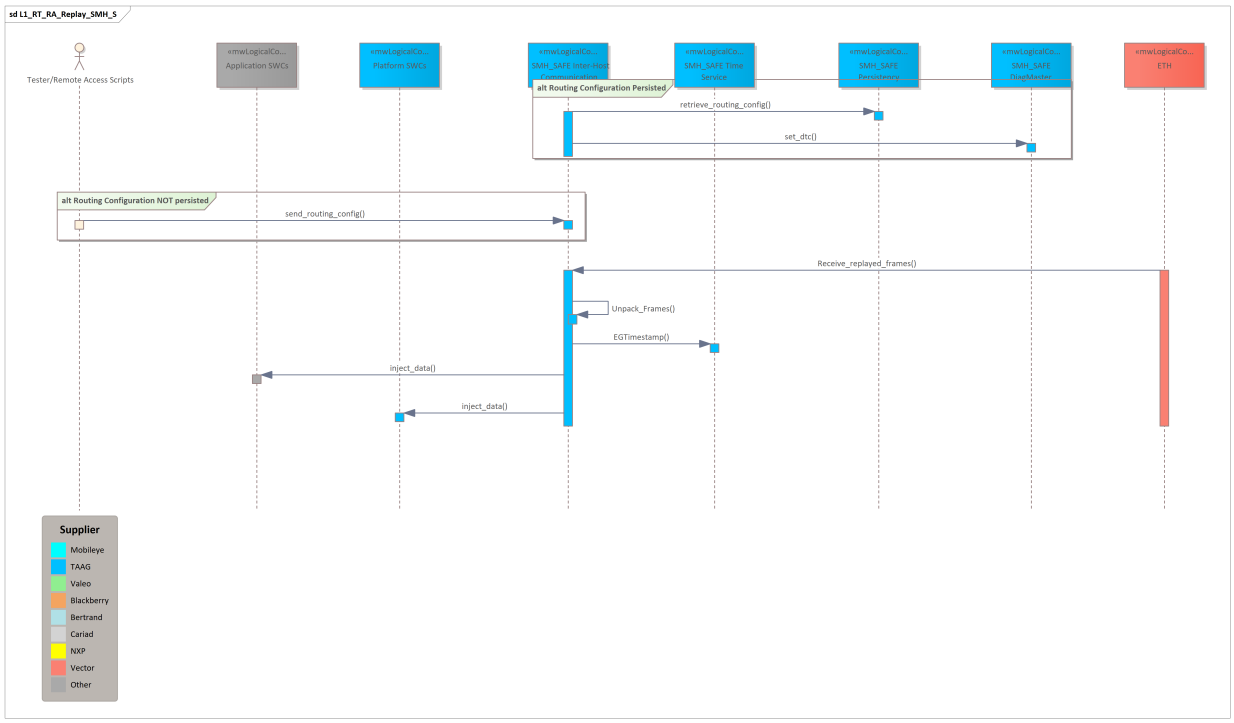


Above diagram explains how capturing data from SMH_P SWCs RTE works. Inter-Host Communication component receives routing configuration from Debug PC and rearrange routing tables to send requested data to Debug PC. [S32GPRODP-296780]

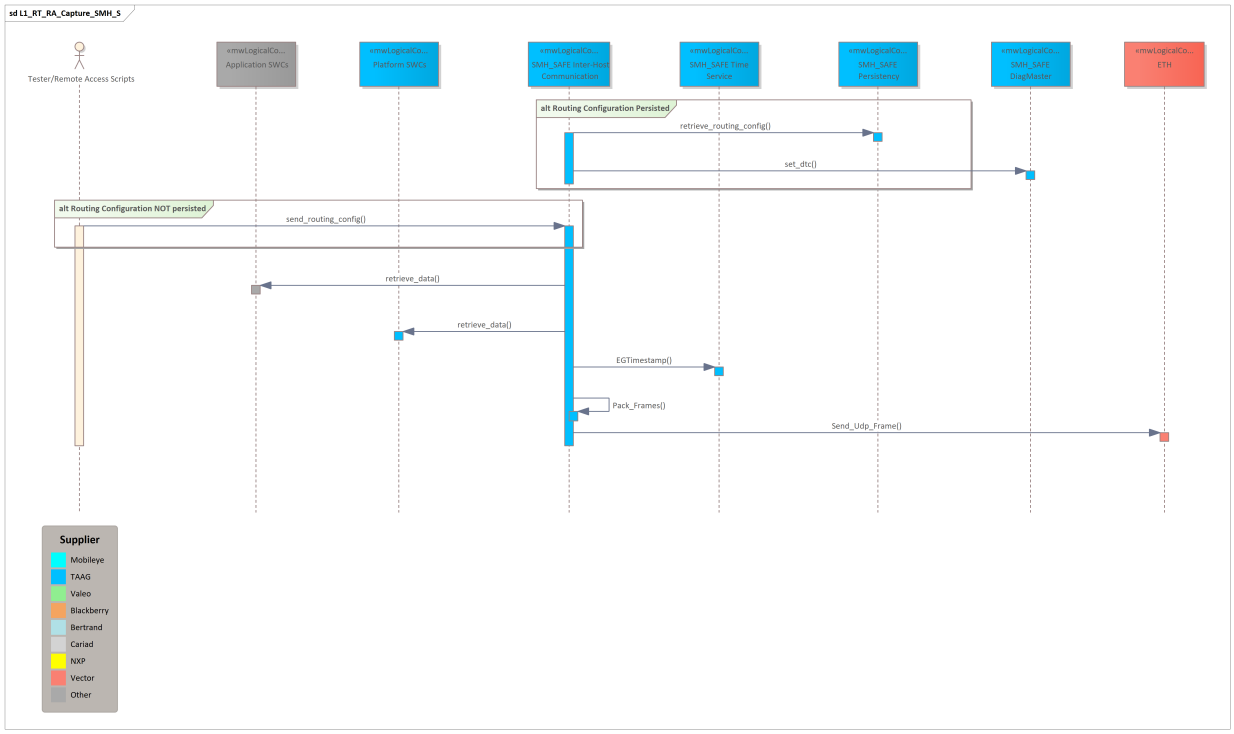


Above diagram explains how replaying data from Debug PC to SMH_P SWCs works. Inter-Host Communication component receives routing configuration from Debug PC indicating which data is going to be received and mutes the original ECU on ECU. After receiving the data it injects to RTE as it is produced by the original producer SWC. [S32GPRODP-296782]

6.2.7.3.2 Remote access on Safety host



Above diagram explains how capturing data from SMH_S SWCs RTE works. Inter-Host Communication component receives routing configuration from Debug PC and rearrange routing tables to send requested data to Debug PC. [S32GPRODP-296781]

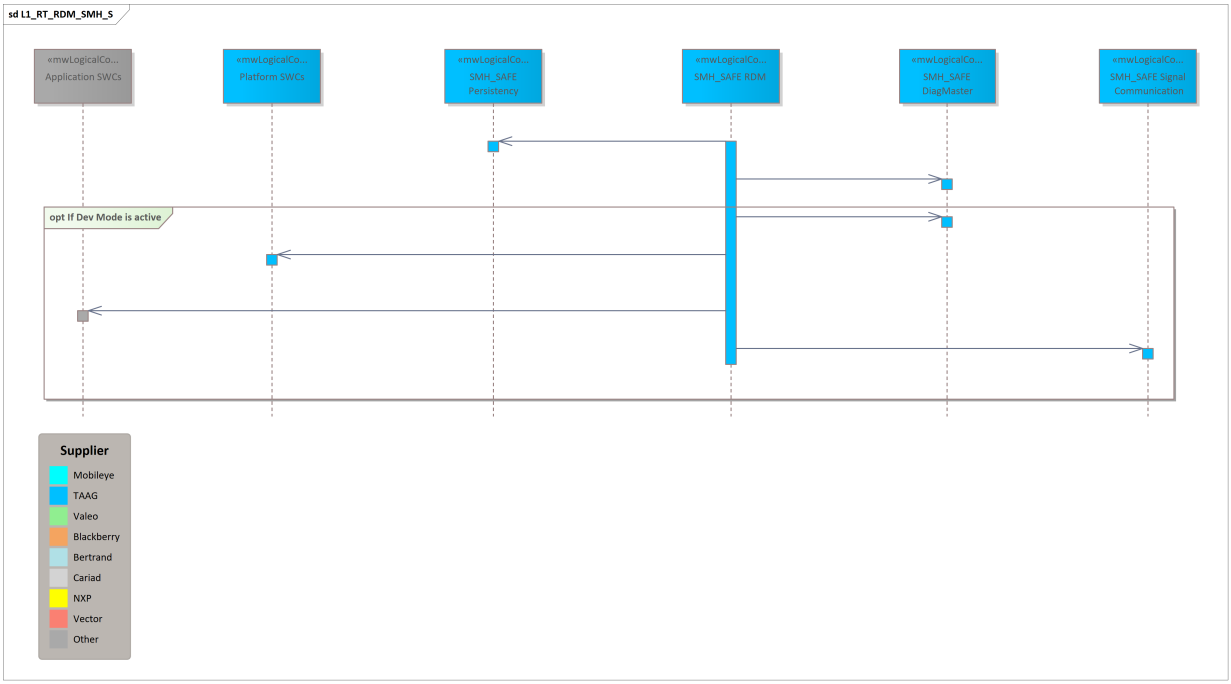


Above diagram explains how replaying data from Debug PC to SMH_S SWCs works. Inter-Host Communication component receives routing configuration from Debug PC indicating which data is going to be received and mutes the original ECU on ECU. After receiving the data it injects to RTE as it is produced by the original producer SWC. [S32GPRODP-296775]

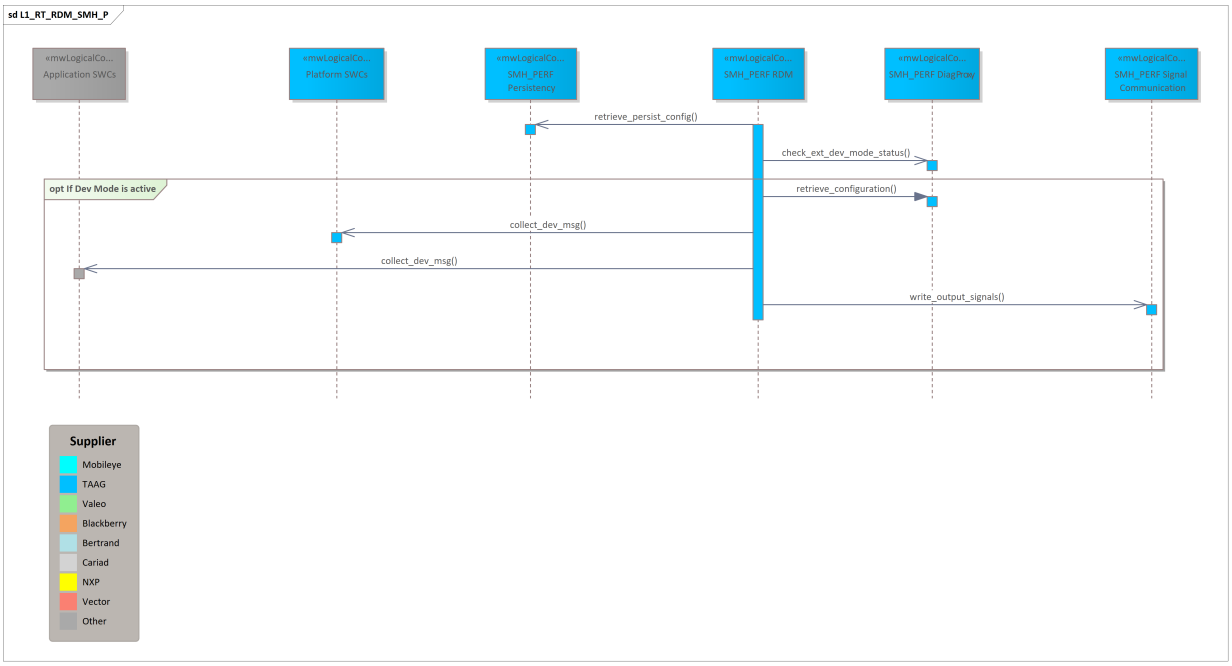
6.2.7.4 Routing of Developer Messages (RDM)

Routing of Developer Messages functionality is a platform service targeted for series SW to allow users to receive developer messages without requiring any additional debug interface. RDM, collects all the data from SWCs and aggregate the data with necessary information (e.g. SWCID, timestamp etc.) and propagates it to signal communication SWC which later sends the data over dedicated PDUs. [S32GPRODP-296797]

6.2.7.4.1 Routing of Developer Messages on Safety Host

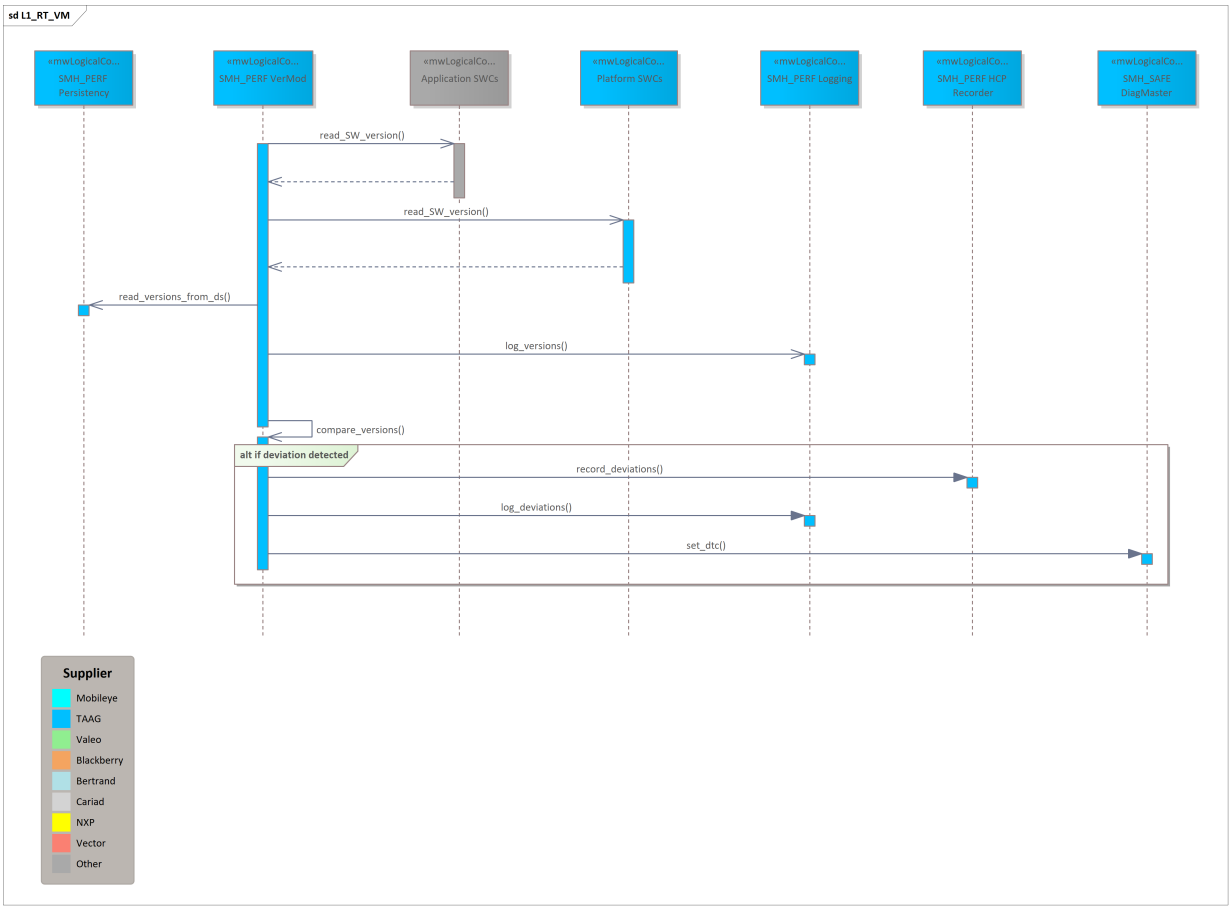


6.2.7.4.2 Routing of Developer Messages on Performance Host



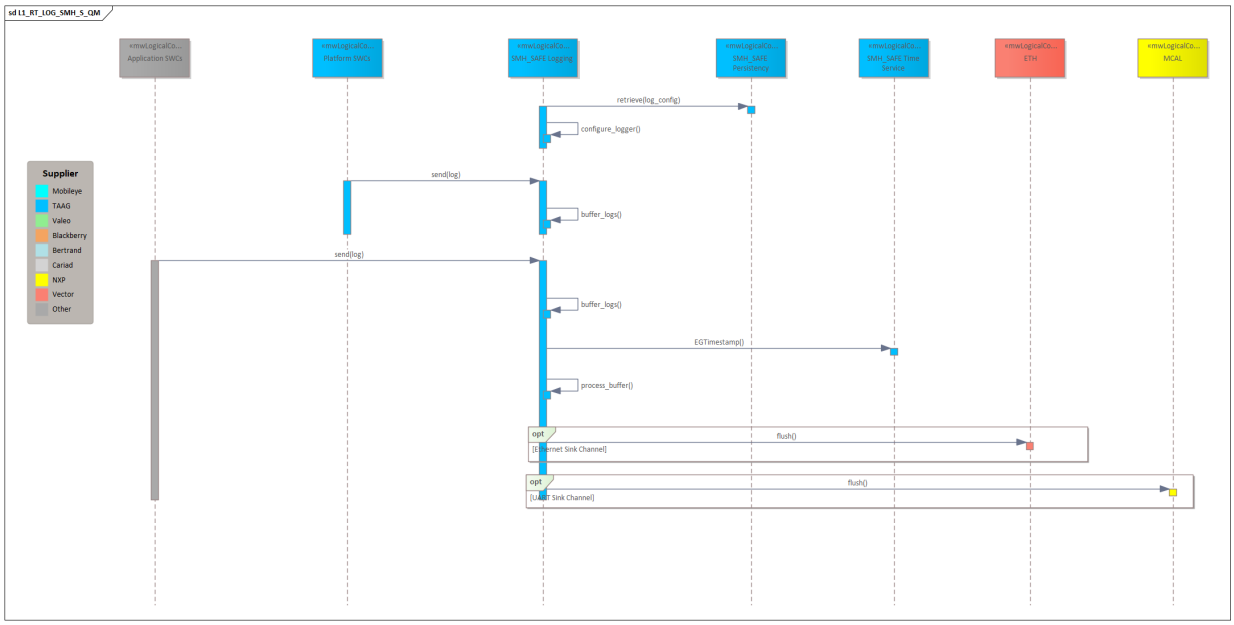
6.2.7.5 Version Module (VM)

As described in illustration below, VM collects SW versions from each SWC and compare it to versions retrieved from dedicated datasets, if there is any deviation it will be reported over HCP Recorder, Logging and a DTC will be raised. [S32GPRODP-296798]

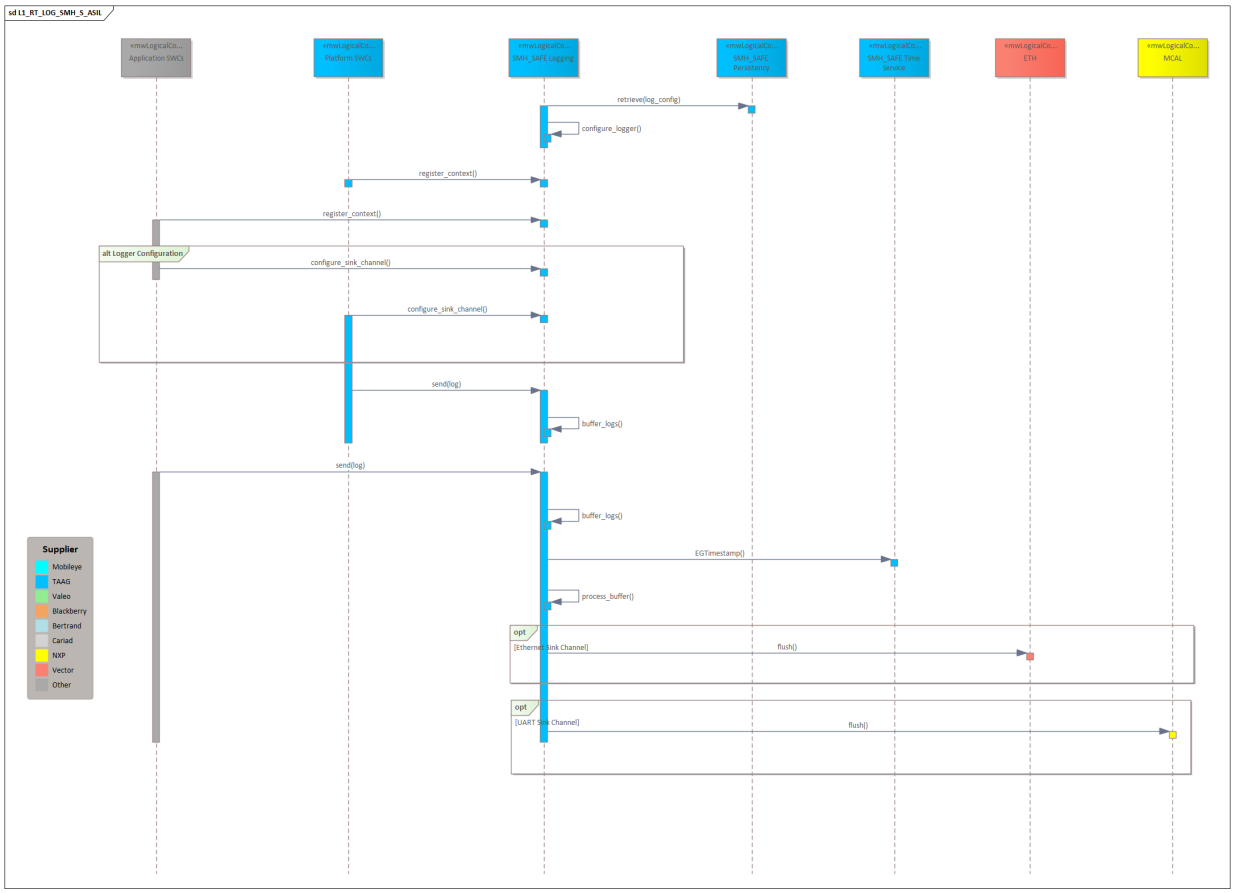


6.2.7.6 Logging framework

6.2.7.6.1 Logging Framework on Safety host

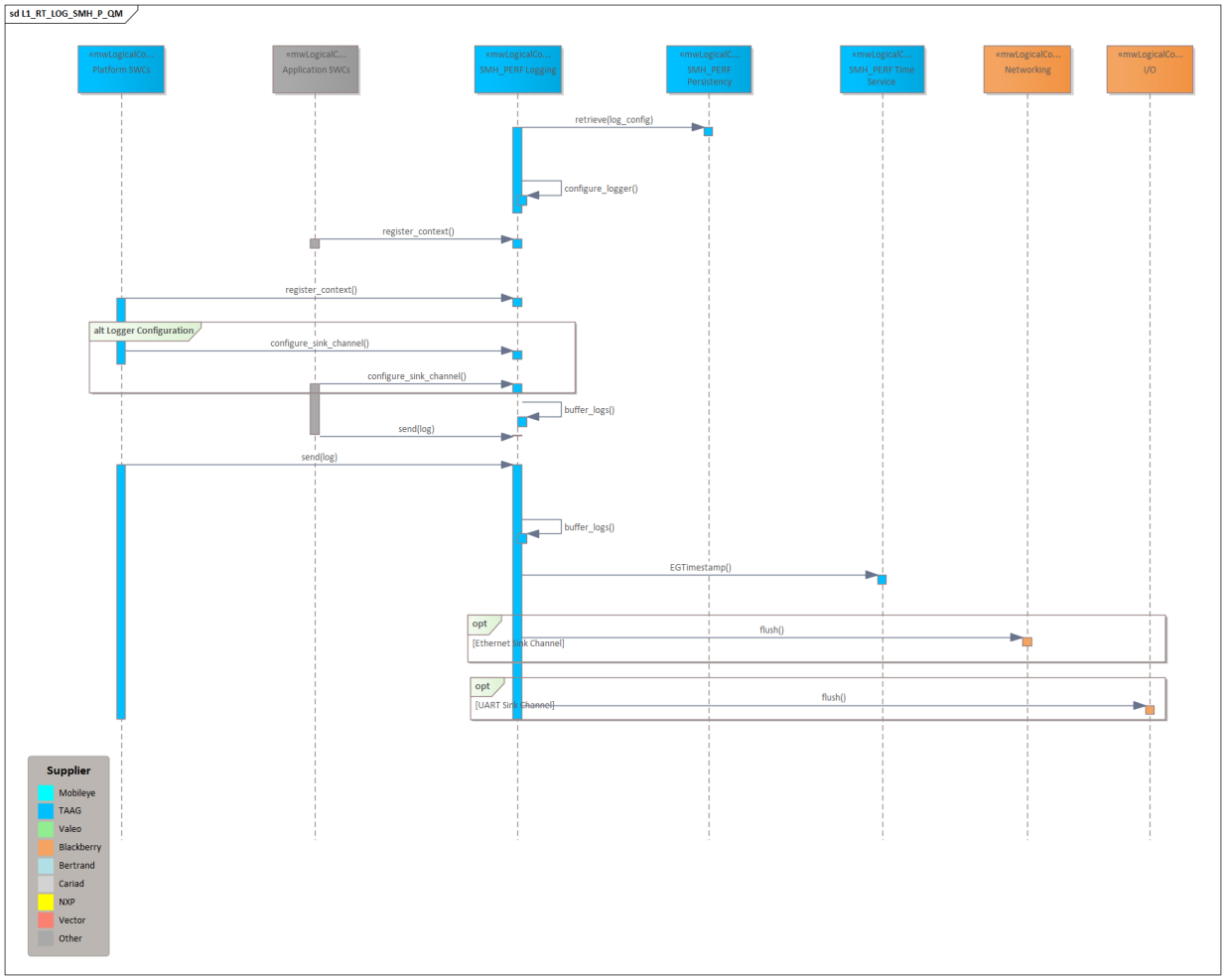


Above diagram shows how QM Logger works. On initialization it retrieves its configuration from persistent storage, if available. It collects logs from SWCs and stores internally until it is scheduled to flush its buffers to configured sink channel. **[**S32GPRODP-296777 **]**

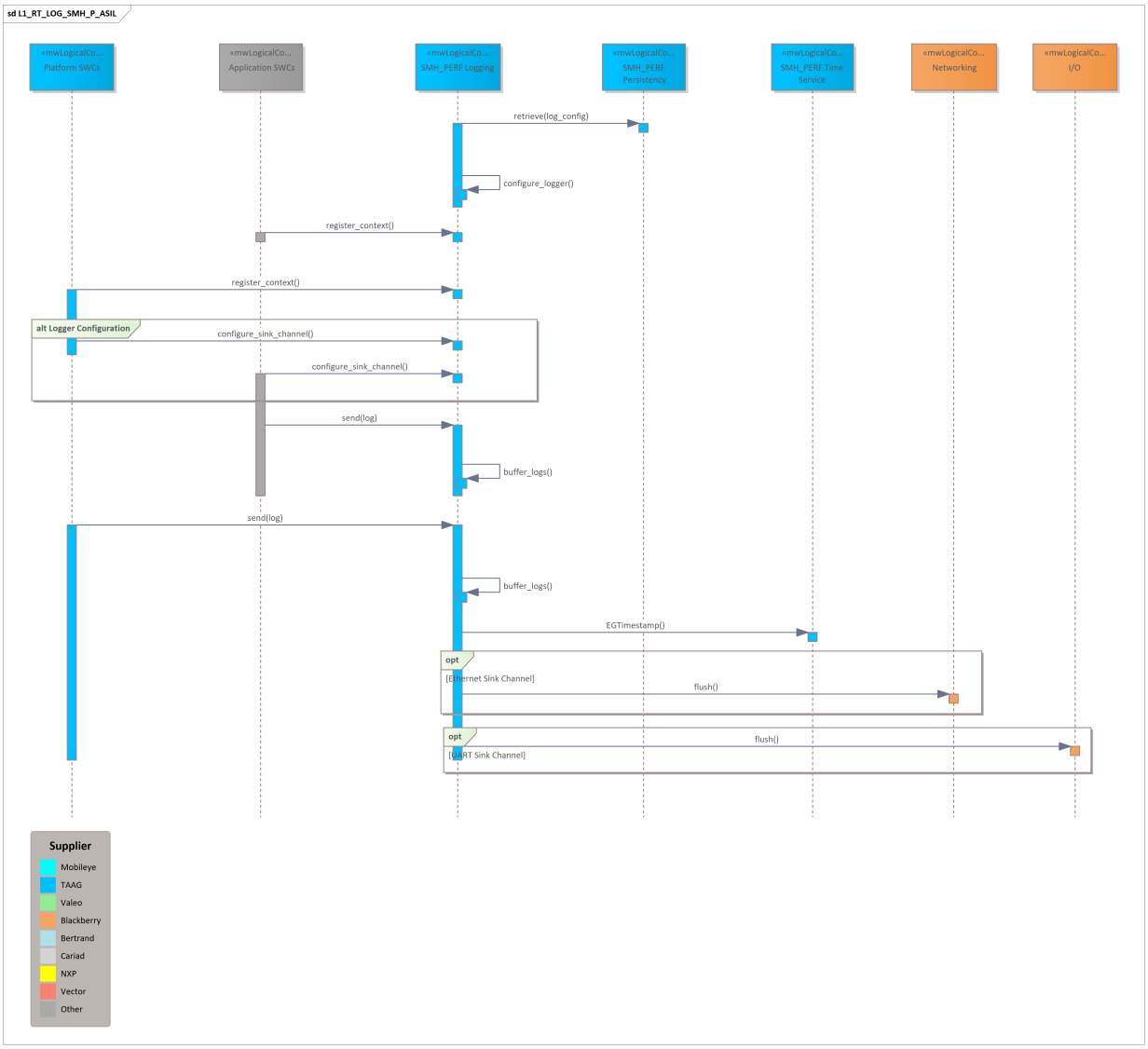


Above diagram shows how ASIL Logger works. On initialization it retrieves its configuration from persistent storage, if available. Each SWC should at least register one context in order to log anything. It collects logs from SWCs which registered contexts and stores internally until it is scheduled to flush its buffers to configured sink channel. **[**S32GPRODP-296779 **]**

6.2.7.6.2 Logging Framework on Performance host

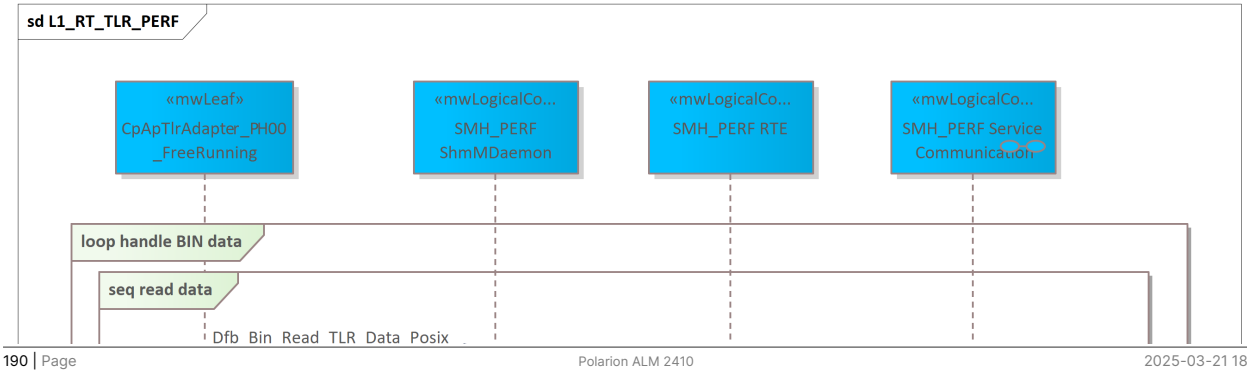


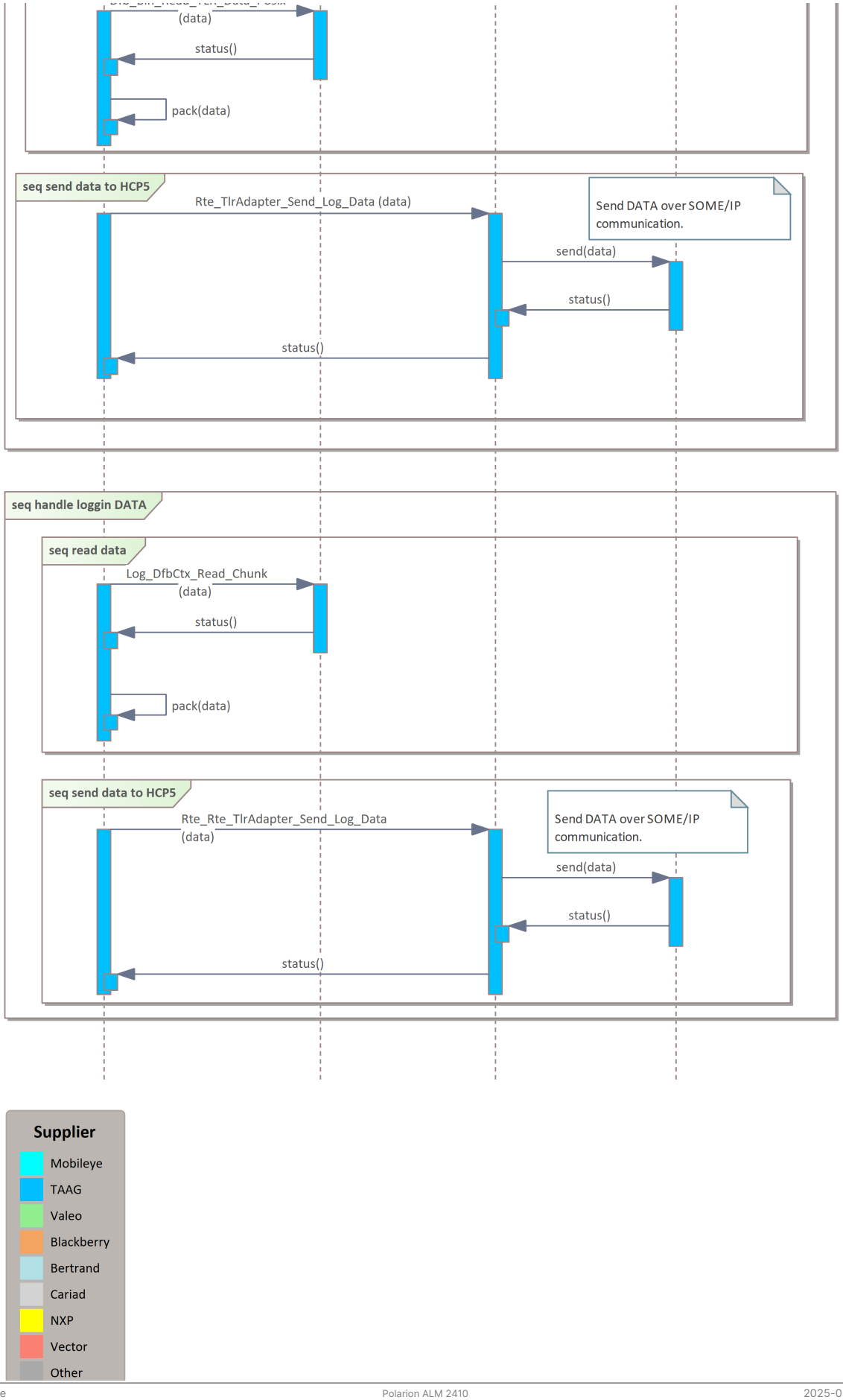
Above diagram shows how QM Logger works. On initialization it retrieves its configuration from persistent storage, if available. It collects logs from SWCs and stores internally until it is scheduled to flush its buffers to configured sink channel. [S32GPRODP-296783]



Above diagram shows how ASIL Logger works. On initialization it retrieves its configuration from persistent storage, if available. Each SWC should at least register one context in order to log anything. It collects logs from SWCs which registered contexts and stores internally until it is scheduled to flush its buffers to configured sink channel. Via shared memory available to TLR Adapter SWC, TLR is an sink option as well for SMH_P. **[**S32GPRODP-296784 **]**

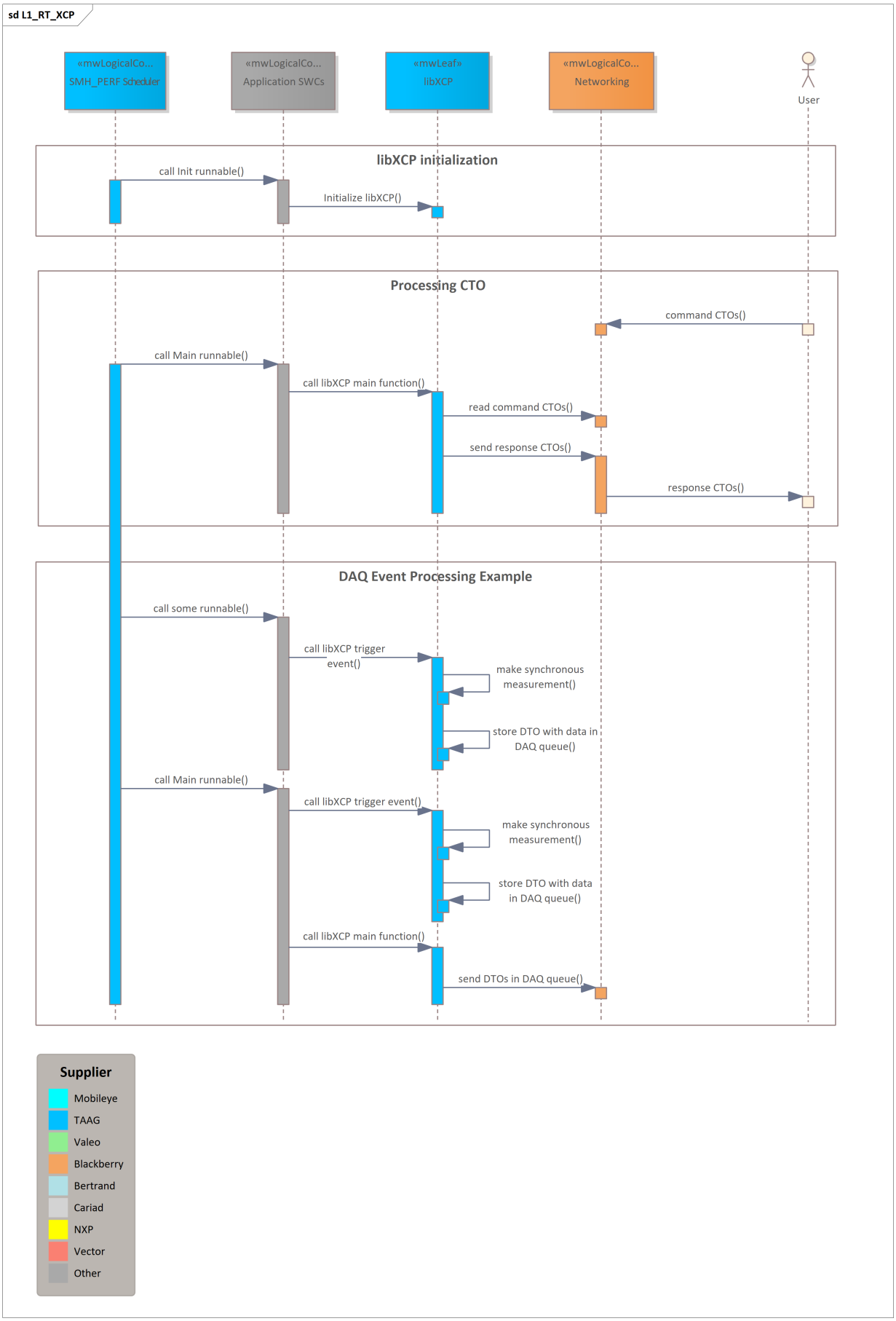
6.2.7.7 TLR




(data)
status()
pack(data)
seq send data to HCP5
Rte_TlrAdapter_Send_Log_Data (data)
Send DATA over SOME/IP communication.
send(data)
status()
status()
seq handle loggin DATA
seq read data
Log_DfbCtx_Read_Chunk
(data)
status()
pack(data)
seq send data to HCP5
Rte_Rte_TlrAdapter_Send_Log_Data
(data)
Send DATA over SOME/IP communication.
send(data)
status()
status()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

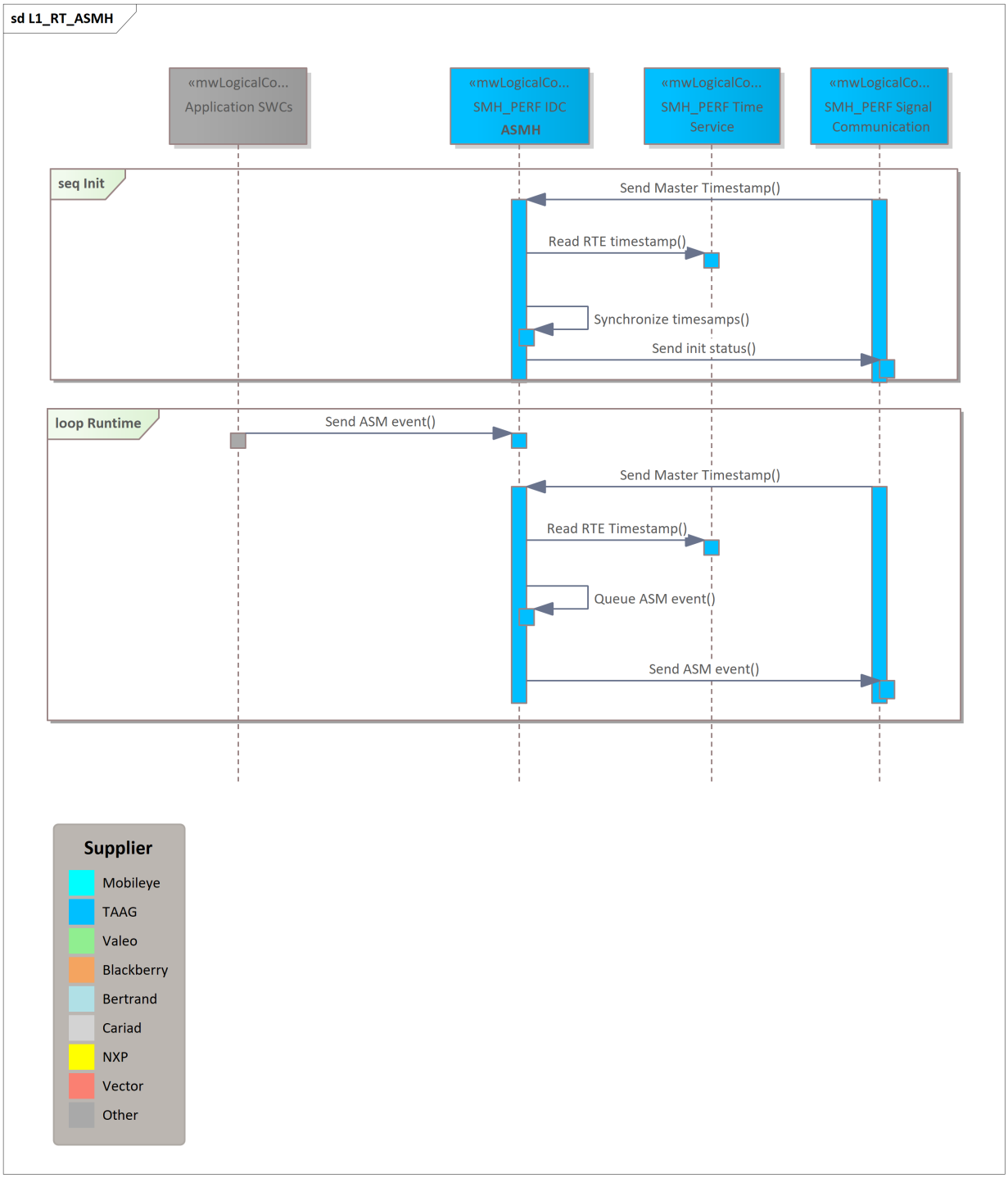
6.2.7.8 Calibration (XCP)

The following diagram shows the main interactions of libXCP, namely the processing of CTOs and DAQ events for synchronous measurements. [S32GPRODP-296869]


sd L1_RT_XCP
«mwLogicalCo...
SMH_PERF Scheduler
«mwLogicalCo...
Application SWCs
«mwLeaf»
libXCP
«mwLogicalCo...
Networking
User
libXCP initialization
call Init runnable()
Initialize libXCP()
Processing CTO
command CTOs()
call Main runnable()
call libXCP main function()
read command CTOs()
send response CTOs()
response CTOs()
DAQ Event Processing Example
call some runnable()
call libXCP trigger event()
make synchronous measurement()
store DTO with data in DAQ queue()
call Main runnable()
call libXCP trigger event()
make synchronous measurement()
store DTO with data in DAQ queue()
call libXCP main function()
send DTOs in DAQ queue()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

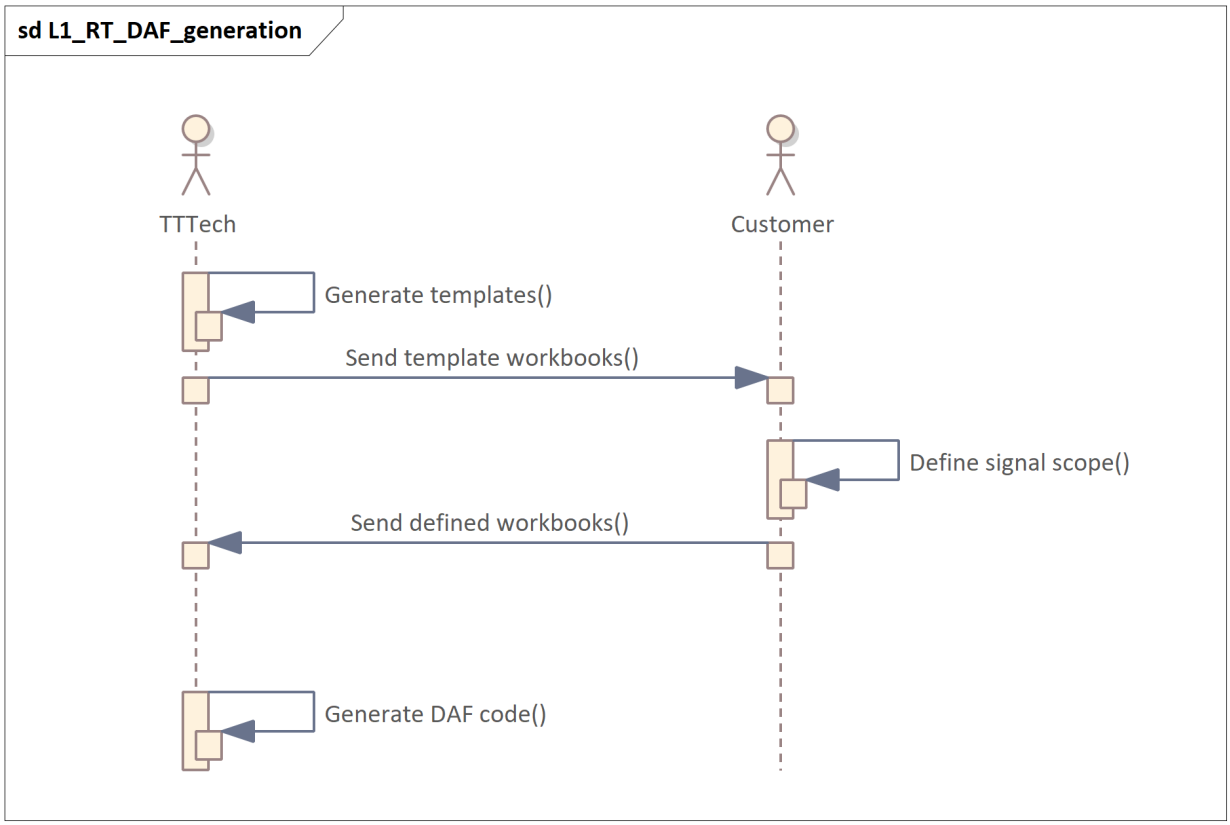
6.2.8 Data recording

6.2.8.1 ASMH

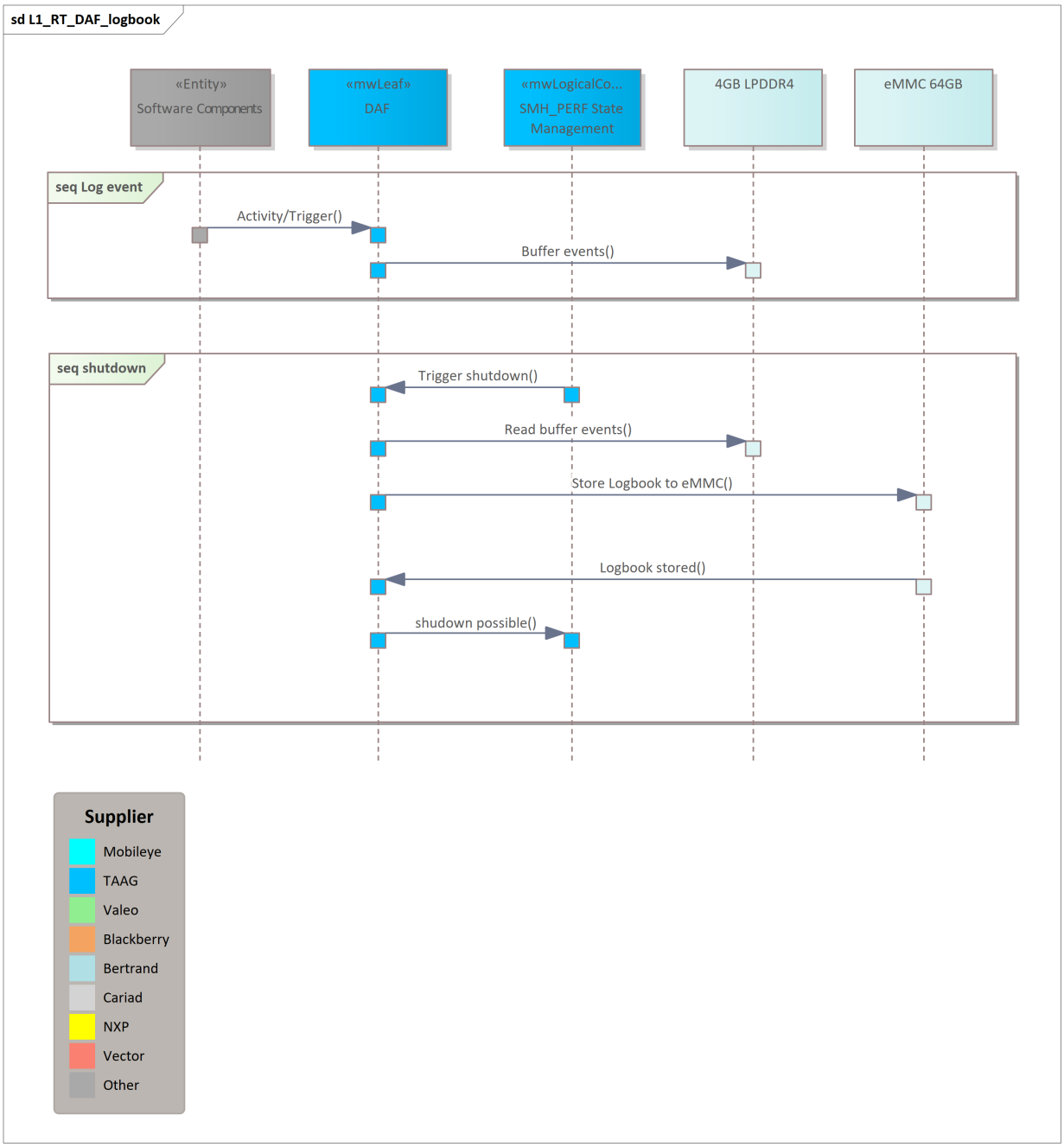


After initialization ASM Handler periodically sends asynchronous events to the ASM master over the RTE. In case there is no event from Applications SWCs (a.k.a. participating functions) empty event is being sent. [S32GPRODP-296894]

6.2.8.2 DAF



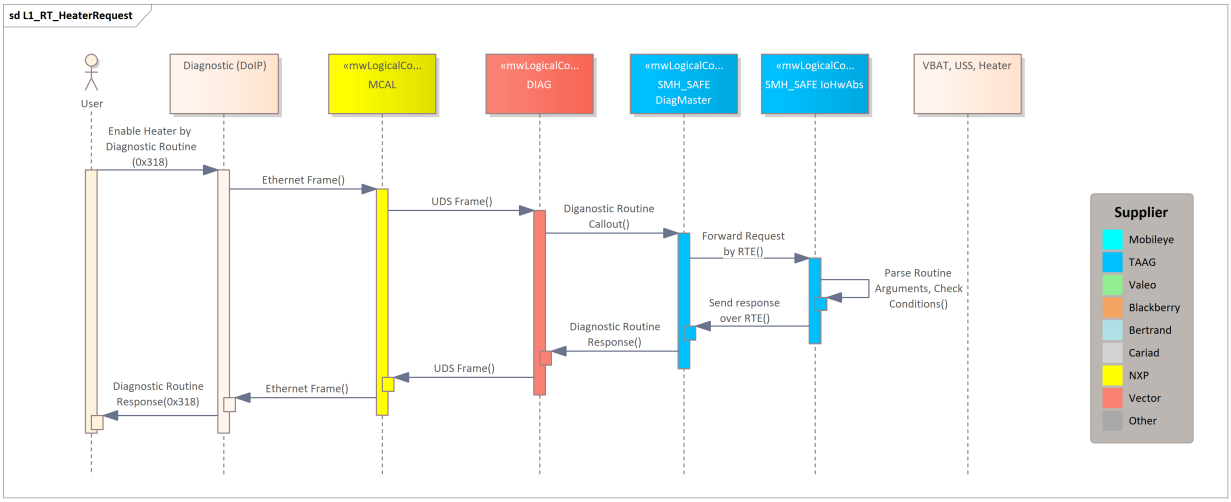
The definition of the DAF signal scope is the responsibility of the customer. Customer has to fallout the workbook template, based on which DAF part which reads the data from RTE is being generated. **[**S32GPRODP-296892 **]**



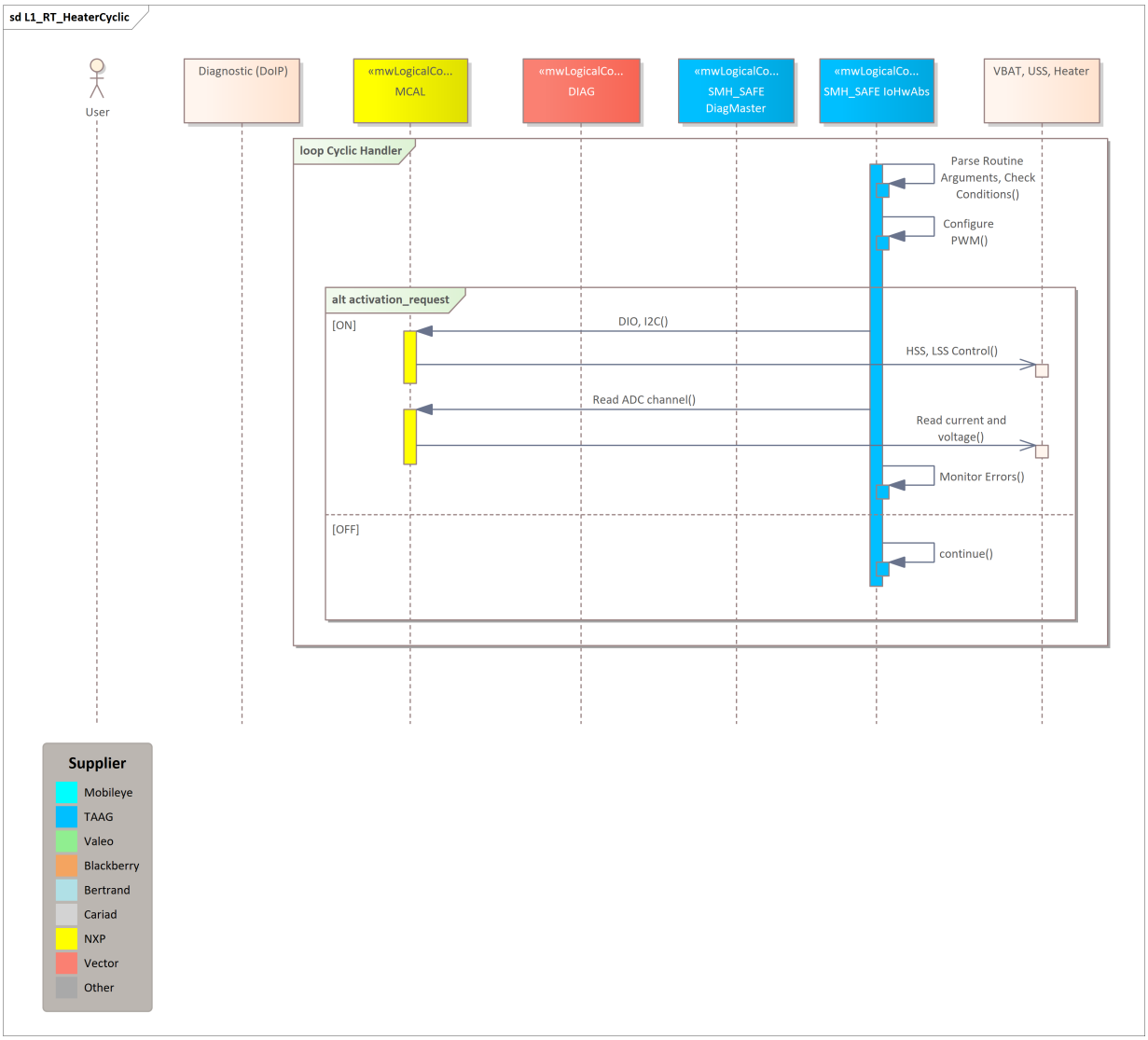
The logbook events in runtime are only saved in the RAM buffer. Only on shutdown request they are being saved to the persistent memory. [S32GPRODP-296893]

6.2.9 I/O Hardware Abstraction

6.2.9.1 Heater



Camera Heater requests can be triggered by the user through a diagnostic routine (0x318). The UDS request is processed by the diagnostic stack and forwarded to the I/O Hardware Abstraction component. There it is parsed and the parameters are stored to configure the heater control. A positive response is returned through the diagnostic stack to the user. **[**S32GPRODP-296182 **]**

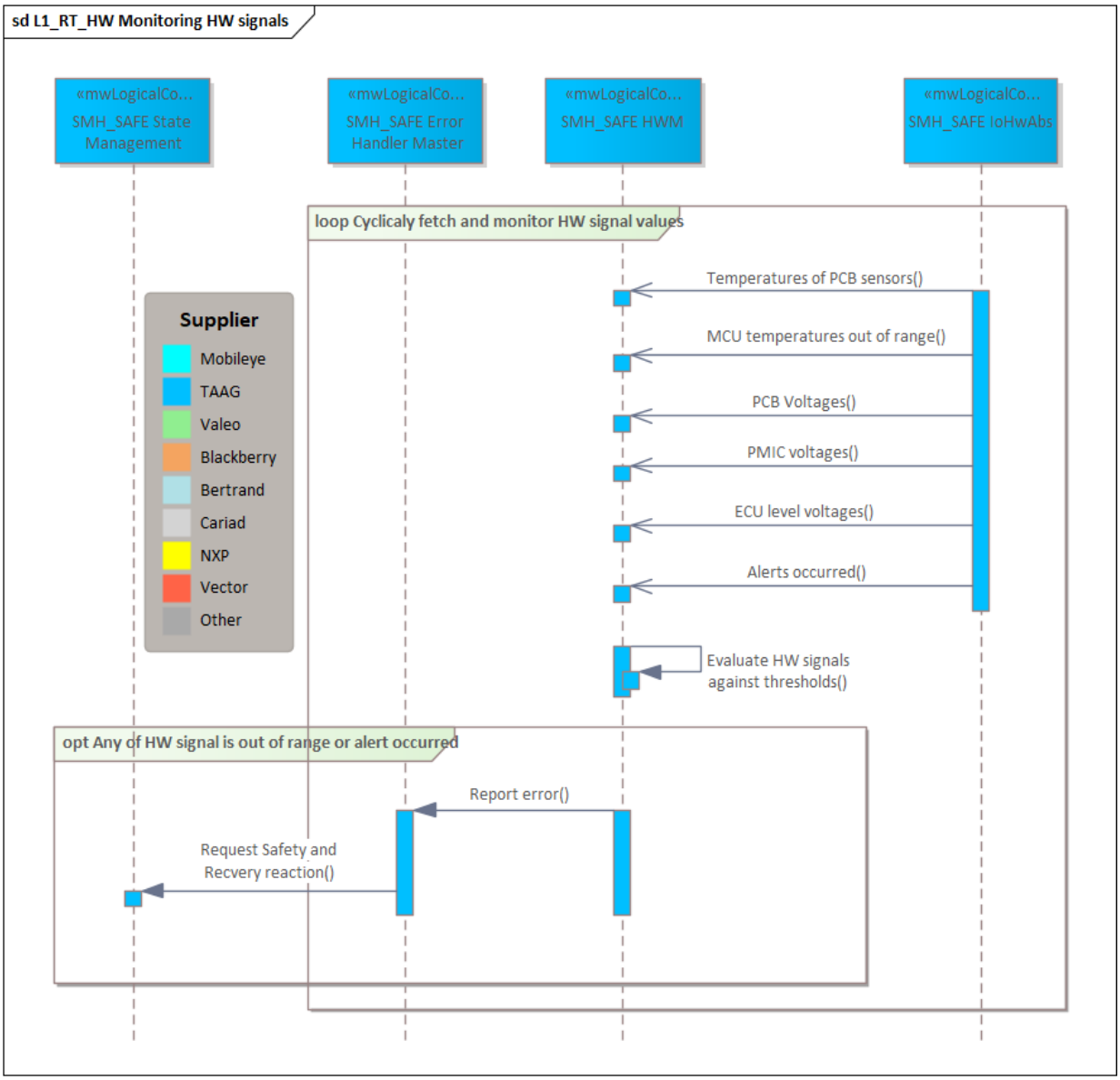


Stored parameters received be the diagnostic routine are parsed cyclically. In case of an available activation request, the PWM is configured according to the received parameters. In case of no activation request, the PWM is not configured and the heater deactivated. **[**S32GPRODP-296197 **]**

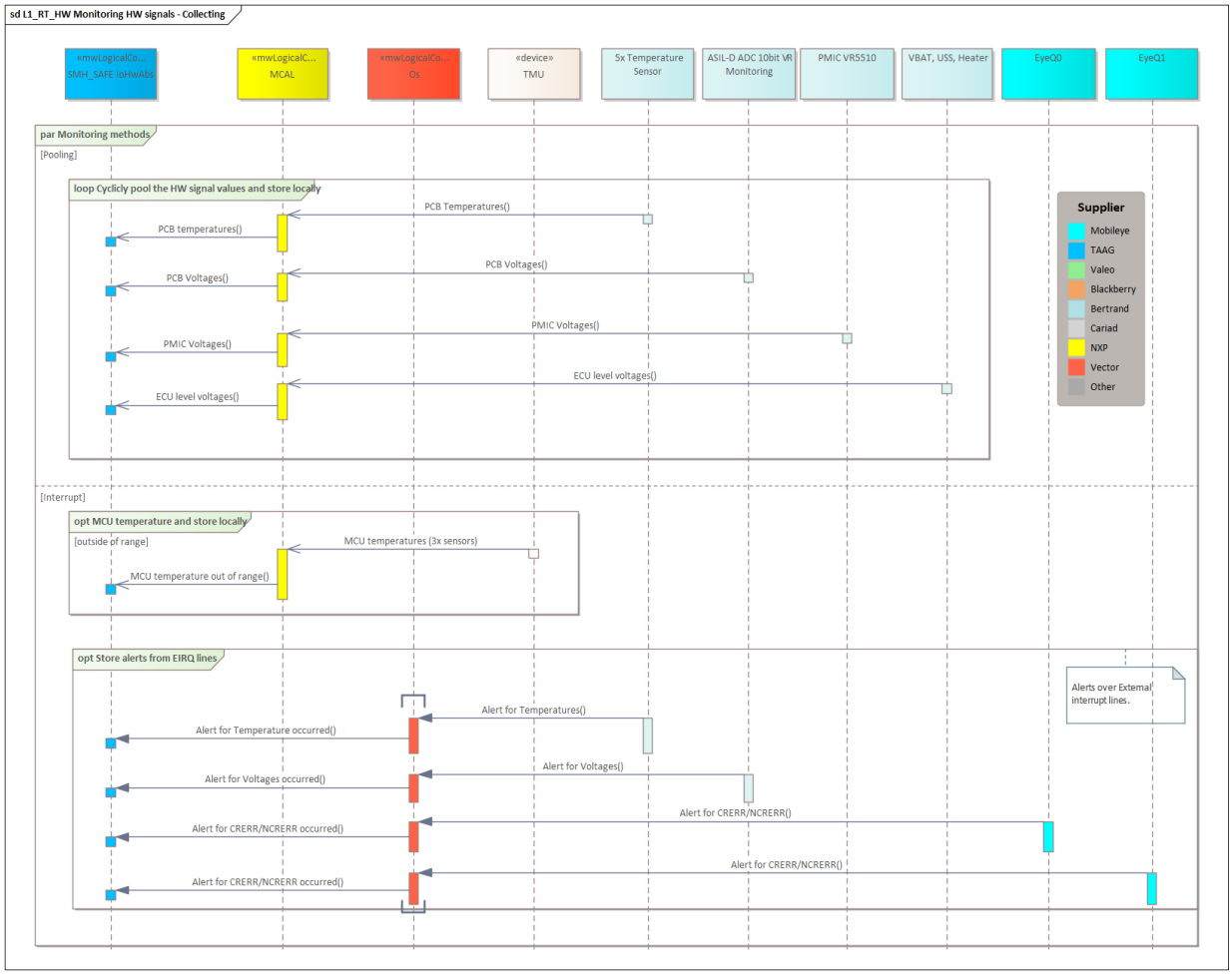
6.2.10 Health

6.2.10.1 HW monitoring

6.2.10.1.1 Physical values



HW Monitoring (physical values) evaluates the physical values of the collected HW signals against the configured thresholds.

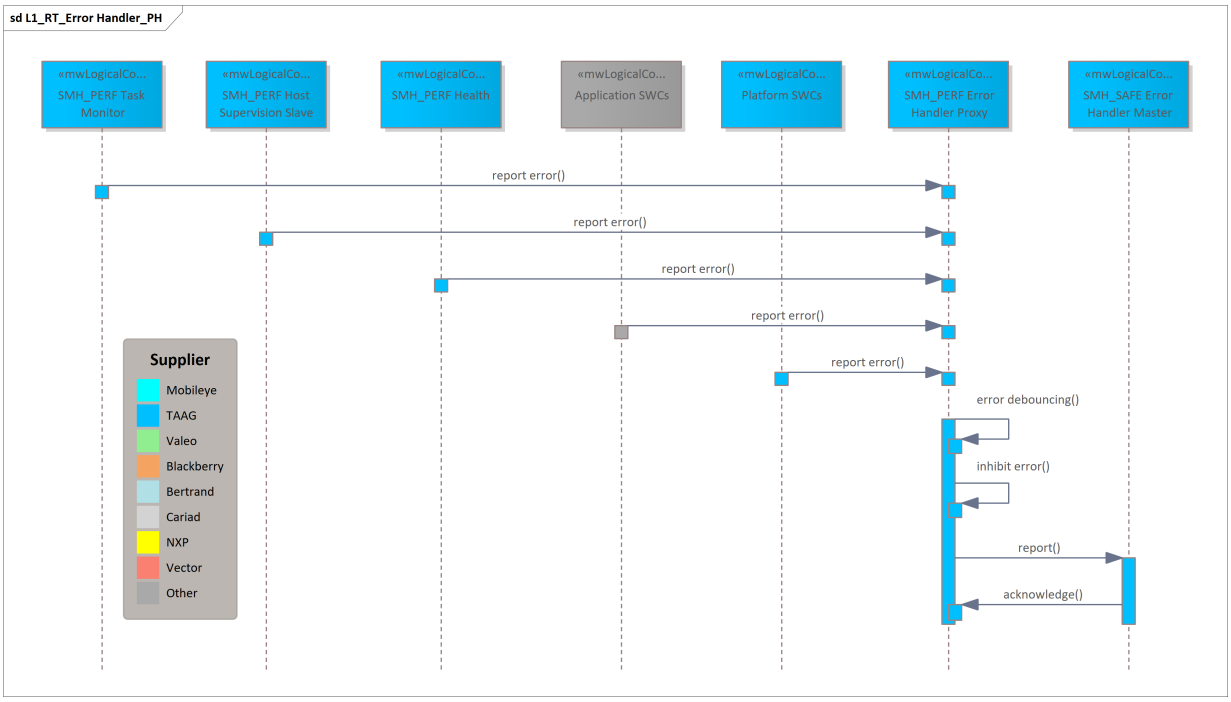


Monitoring of physical values is based on two diverse methods:

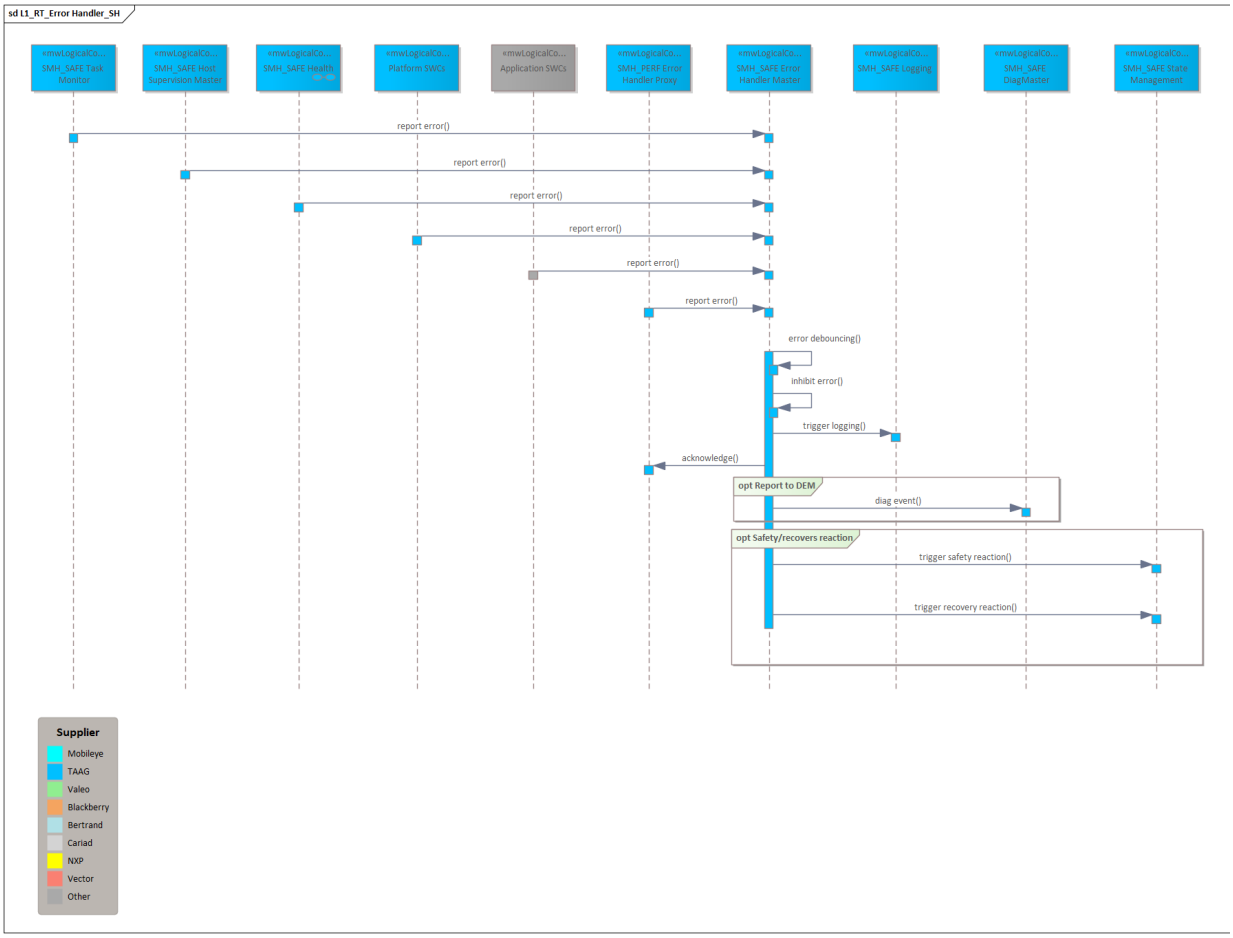
- periodically pooling the physical values
- servicing alerts generated by the sensors, configured to generate the same in case a values are out of allowed ranges.

[S32GPRODP-296806]

6.2.10.2 Error handler

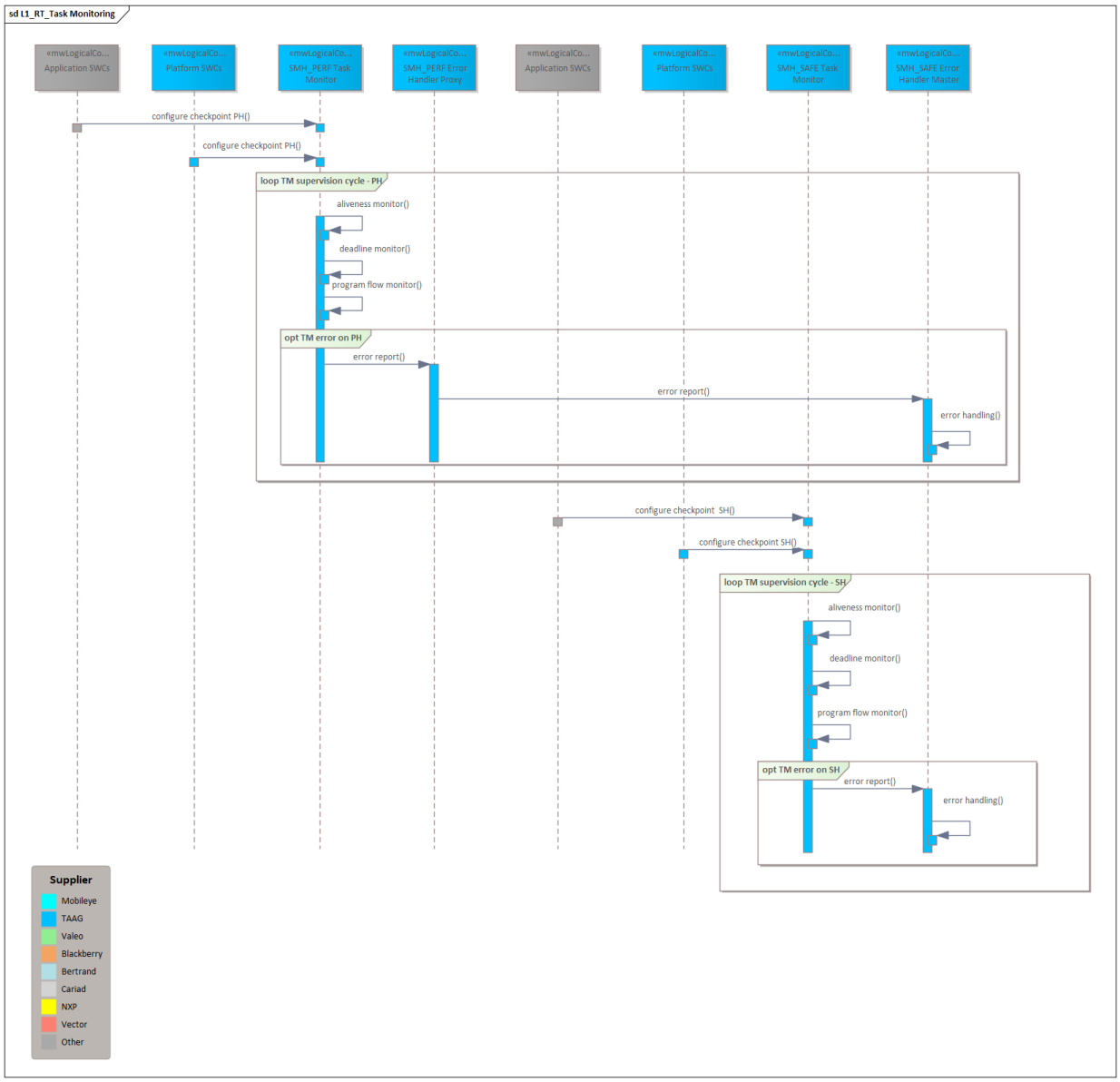


Error Handler Proxy gets error reports from different modules from PH, debounce them, makes error inhibition, possibly take some local handling (e.g. logging), then forward confirmed error to Error Handler Master. Error Handler Master would further send commands back to Error Handler Proxy when it decides to perform necessary safety or recovery reaction on PH. [S32GPRODP-296891]



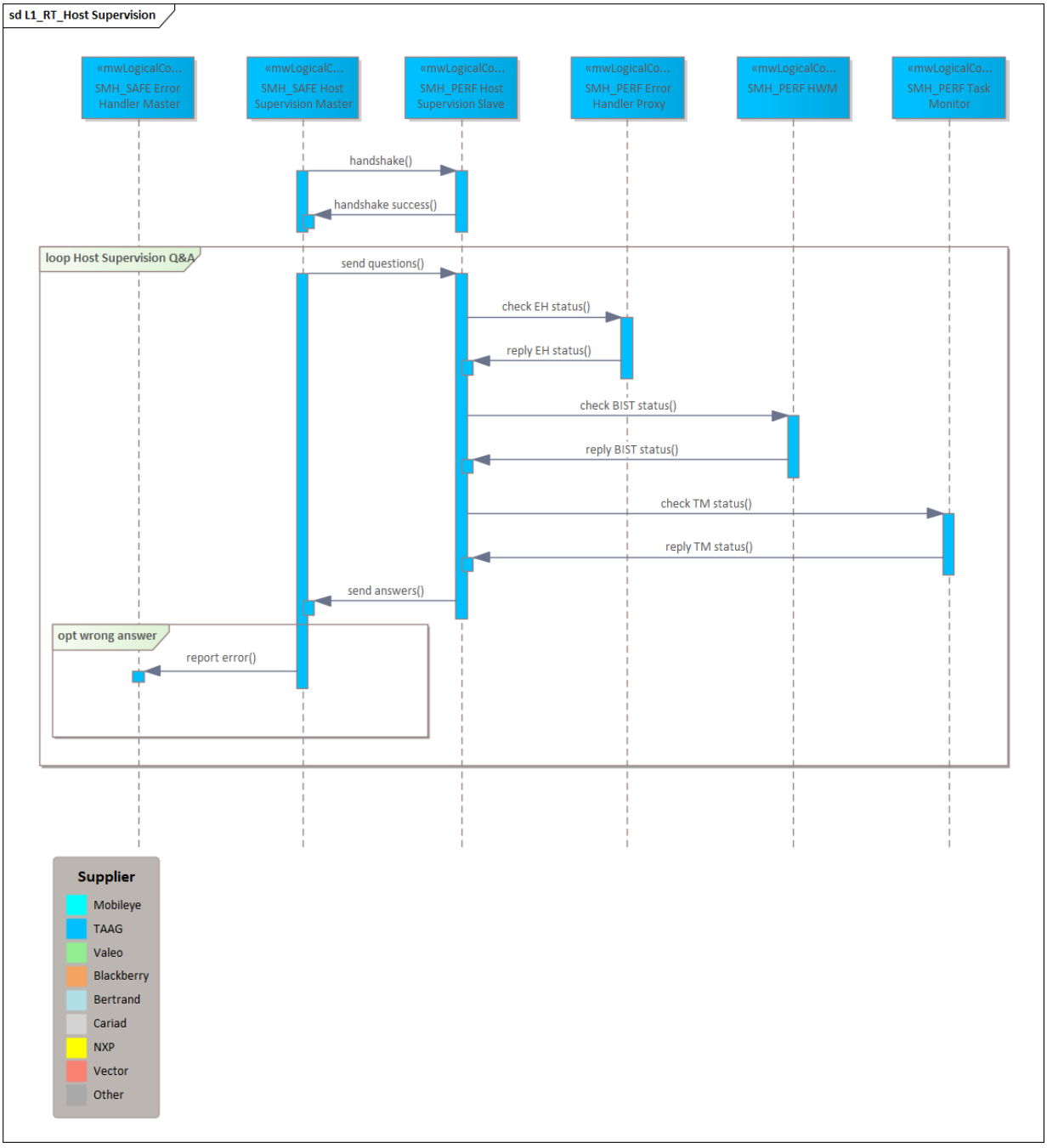
Error Handler Master receives error from different SH modules and also from Error Handler Proxy. After errors are received, Error Handler Master would debounce the errors, make error inhibition, log necessary ones, command to Error Handler Proxy if in need, then take diagnostic recording, finally make safety reaction or possibly recovery reaction. [S32GPRODP-296888]

6.2.10.3 Task monitoring



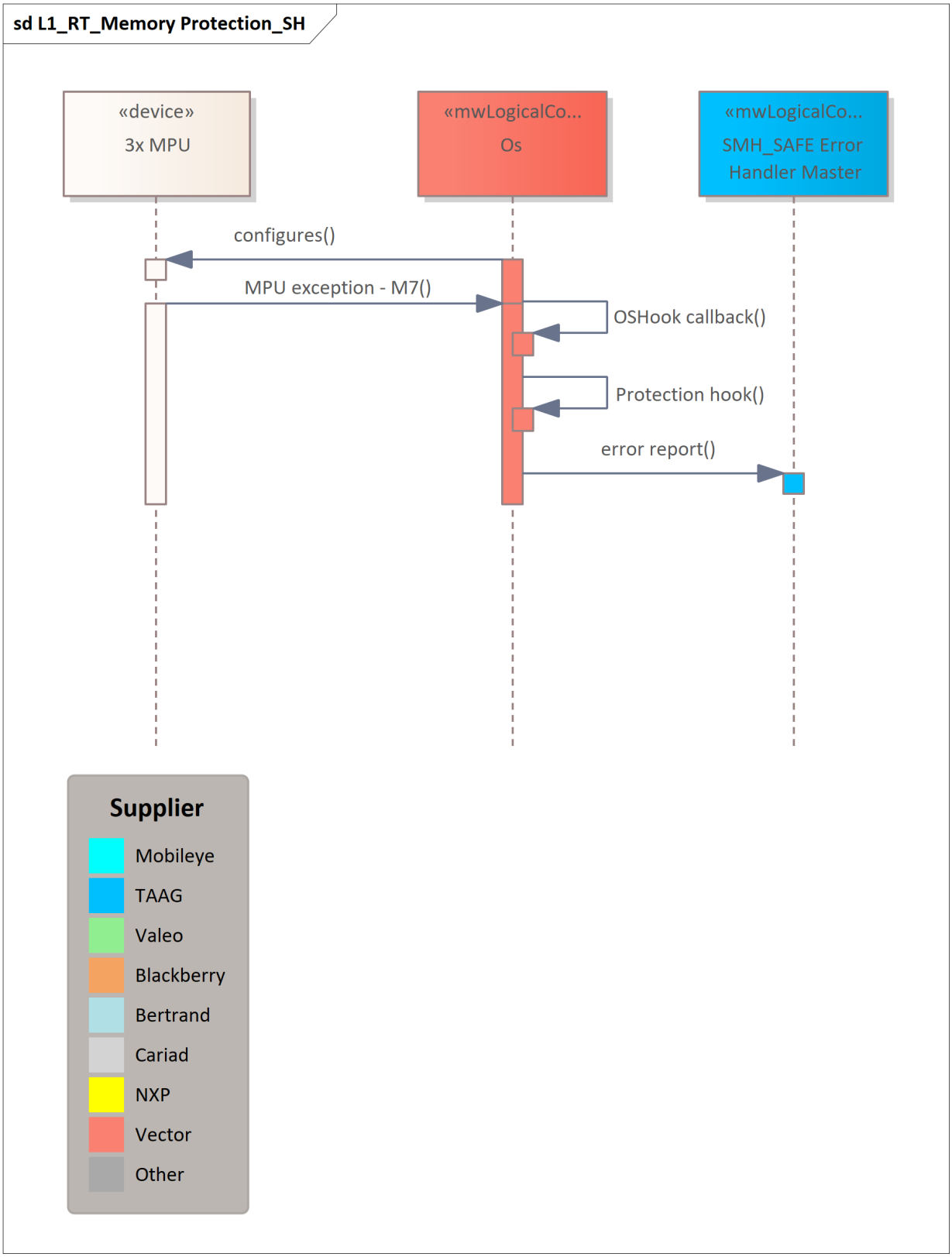
Checkpoints would be separately configured for different customer applications and platform applications on performance host and safety host. Watchdogs on PH (TTTech stack) and SH (AutoSAR stack) would cyclically perform aliveness check, deadline check and program flow check on top of that. If any deviation occurs, error would be reported to error handler. [S32GPRODP-296889]

6.2.10.4 Host supervision

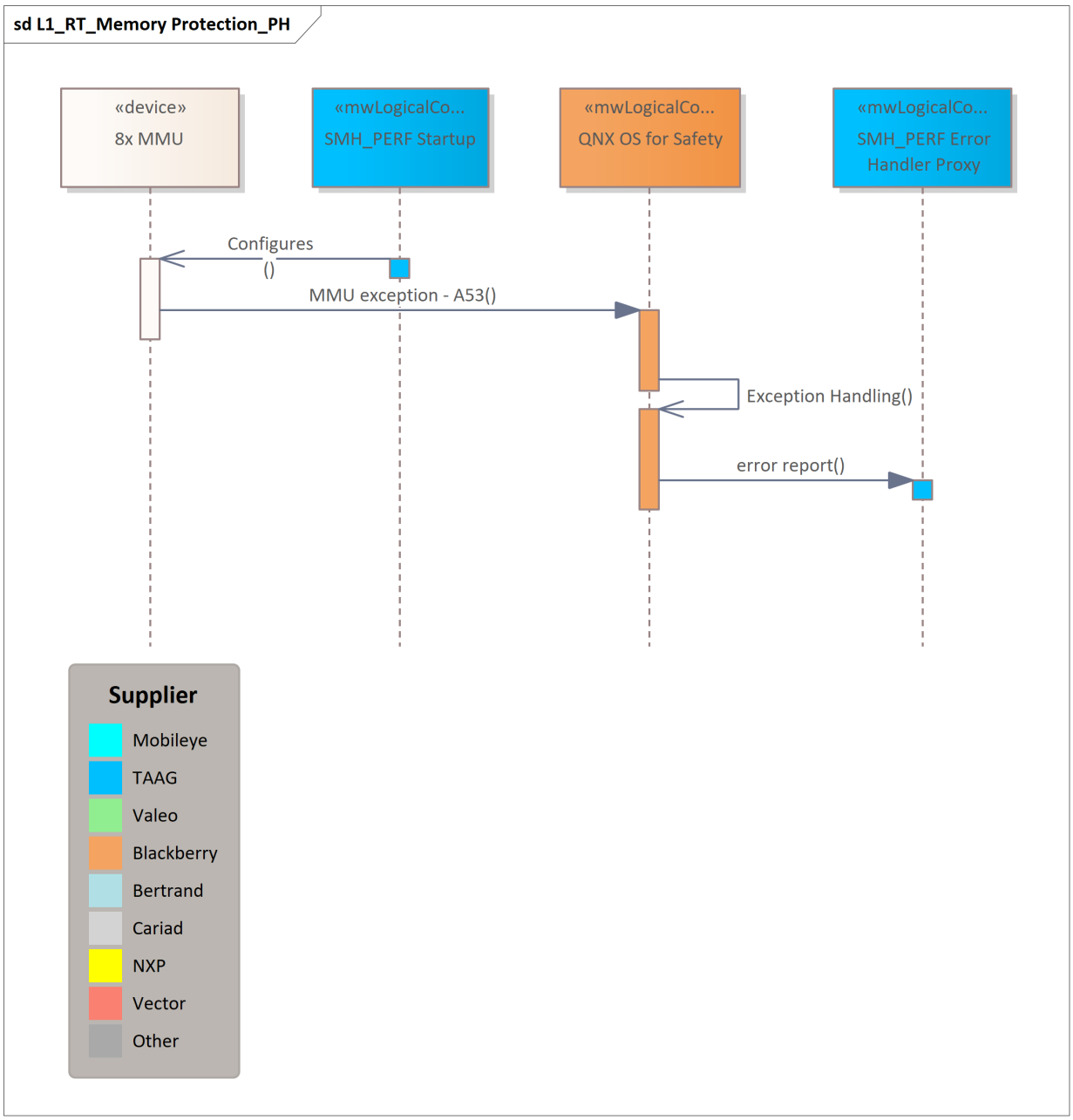


Host Supervision Master would have handshake with Host Supervision Slave in the beginning. Once handshake established, HSVM would cyclically send questions to HSVS, then HSVS would sequentially check status from Error Handler Proxy, BIST, Task Monitoring status and formulate answer back to HSVM. If HSVM detects answer deviation, it would report error to error handler. **[**S32GPRODP-296890 **]**

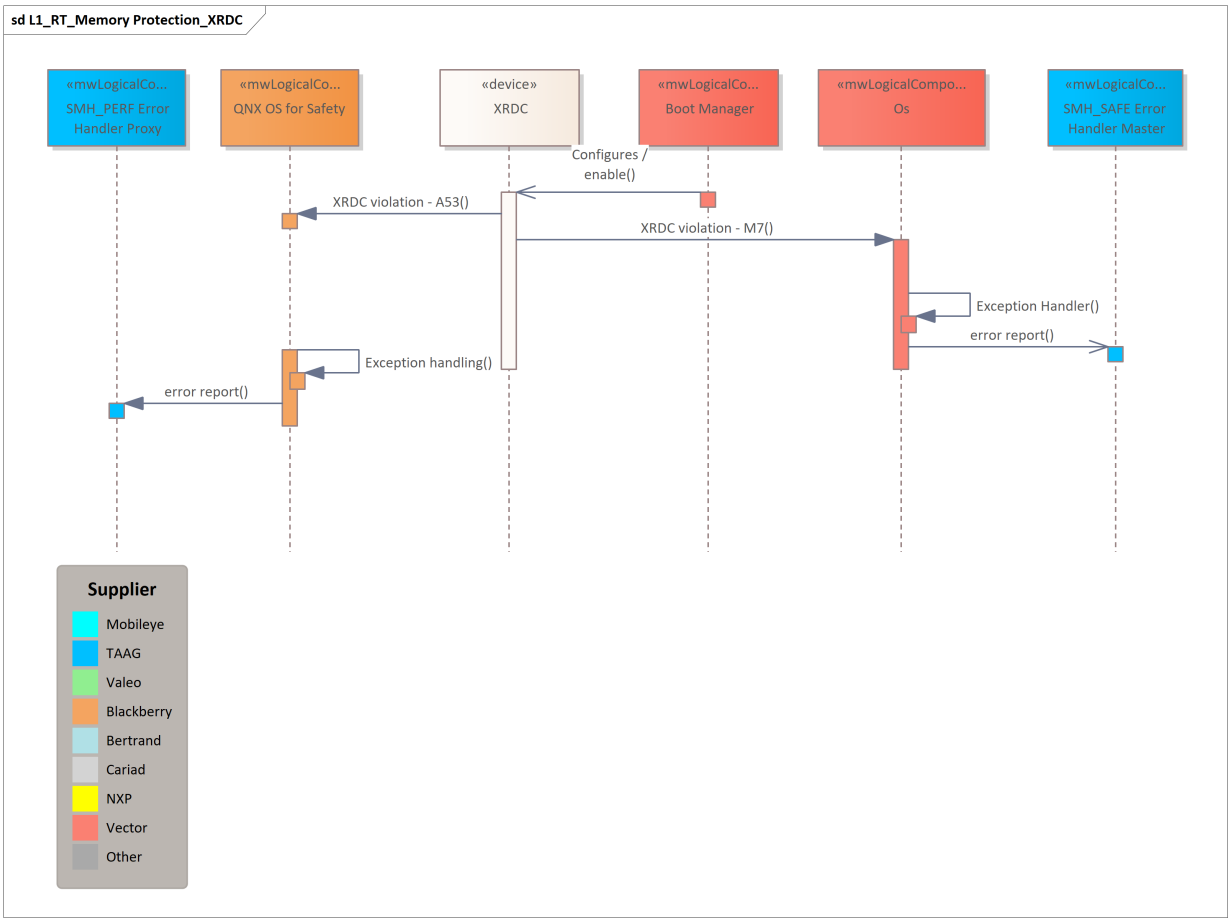
6.2.11 Memory protection



Microsar OS configures and enables MPU after OS is started and it also handles any exceptions with OS Hook errors mapped to memory faults. Error Handler Master component handles failure reaction and recovery strategy. [S32GPRODP-296199]



Each MMU on a core is configured and enabled by Performance partition startup and violation of MMU is handled by QNX OS. Error Handler proxy handles the error reporting to SH. [S32GPRODP-296337]



XRDC is configured and enabled at Boot manager phase and XRDC violations detected on M7 and A53 cores are handled on respective cores with exception handlers from the OS. Errors are further reported to Error Handler Master component, which takes care of failure reaction and recovery action. **[**S32GPRODP-296201 **]**

6.2.12 Security

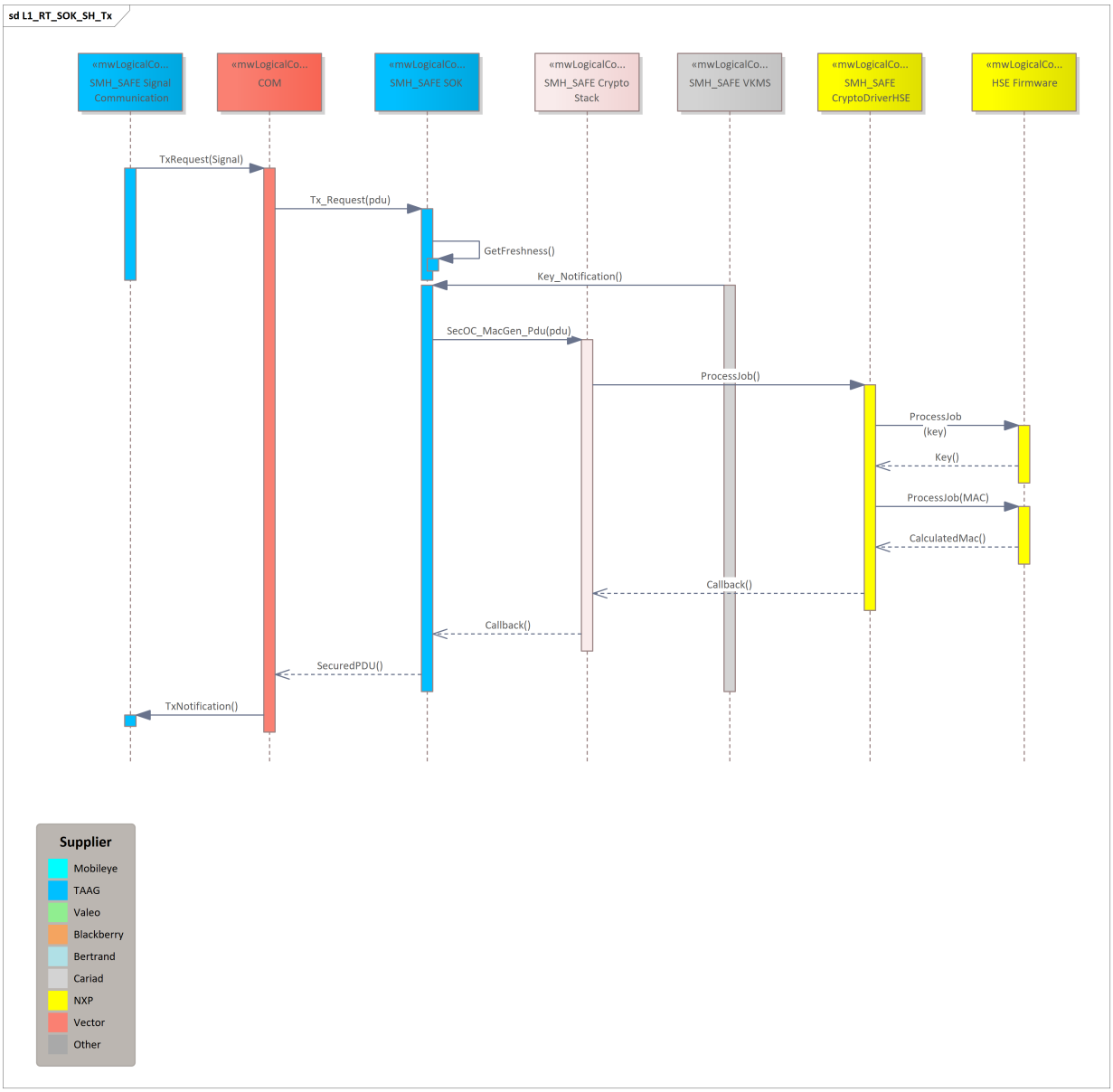
6.2.12.1 SOK

SOK on SH for transmit PDUs, It creates secure PDU by adding freshness value, truncated MAC to the authentic PDU.

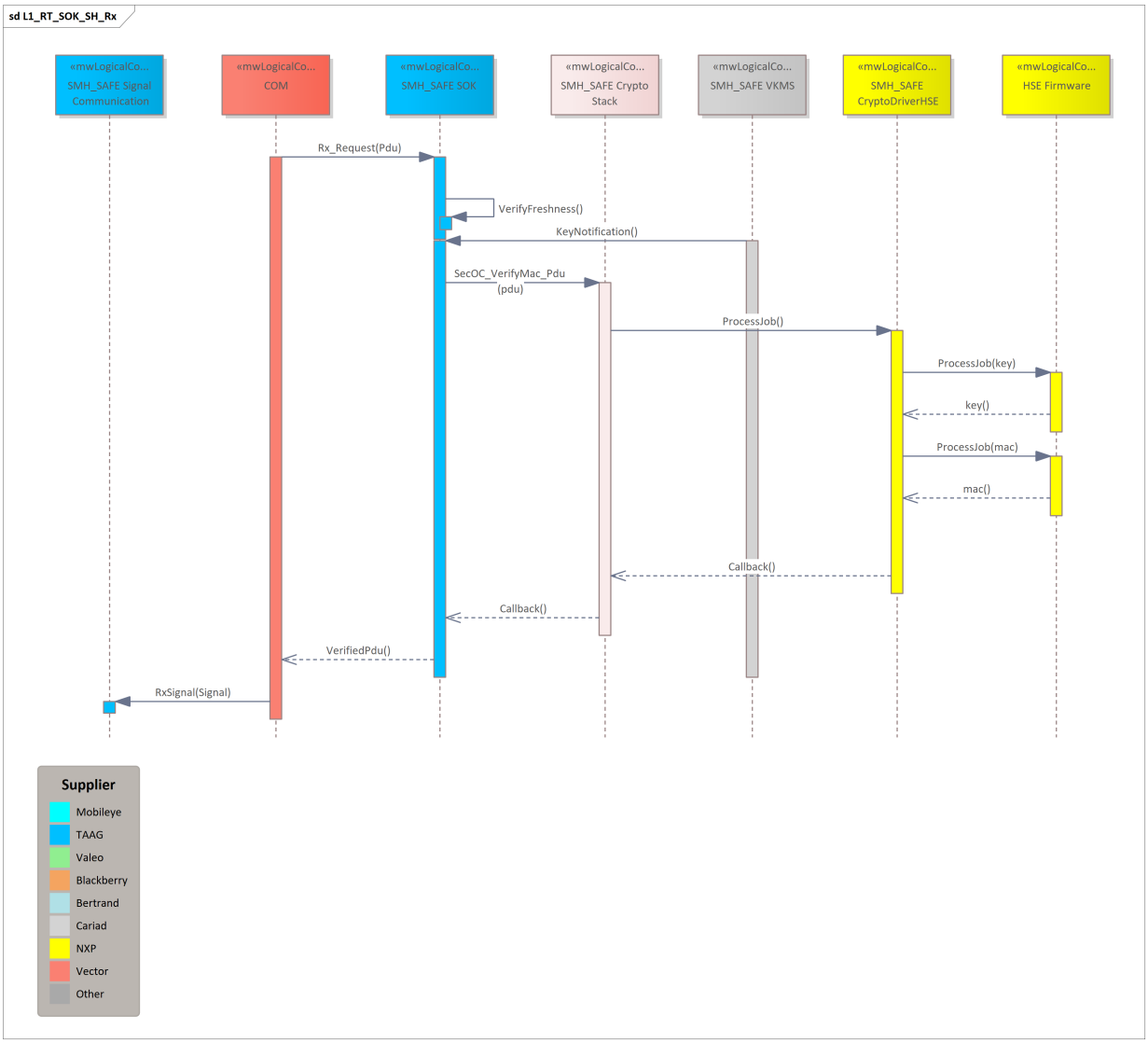
Secured PDU = Authentic PDU + Cryptographic PDU.

Cryptographic PDU = Truncated freshness value + Truncated MAC

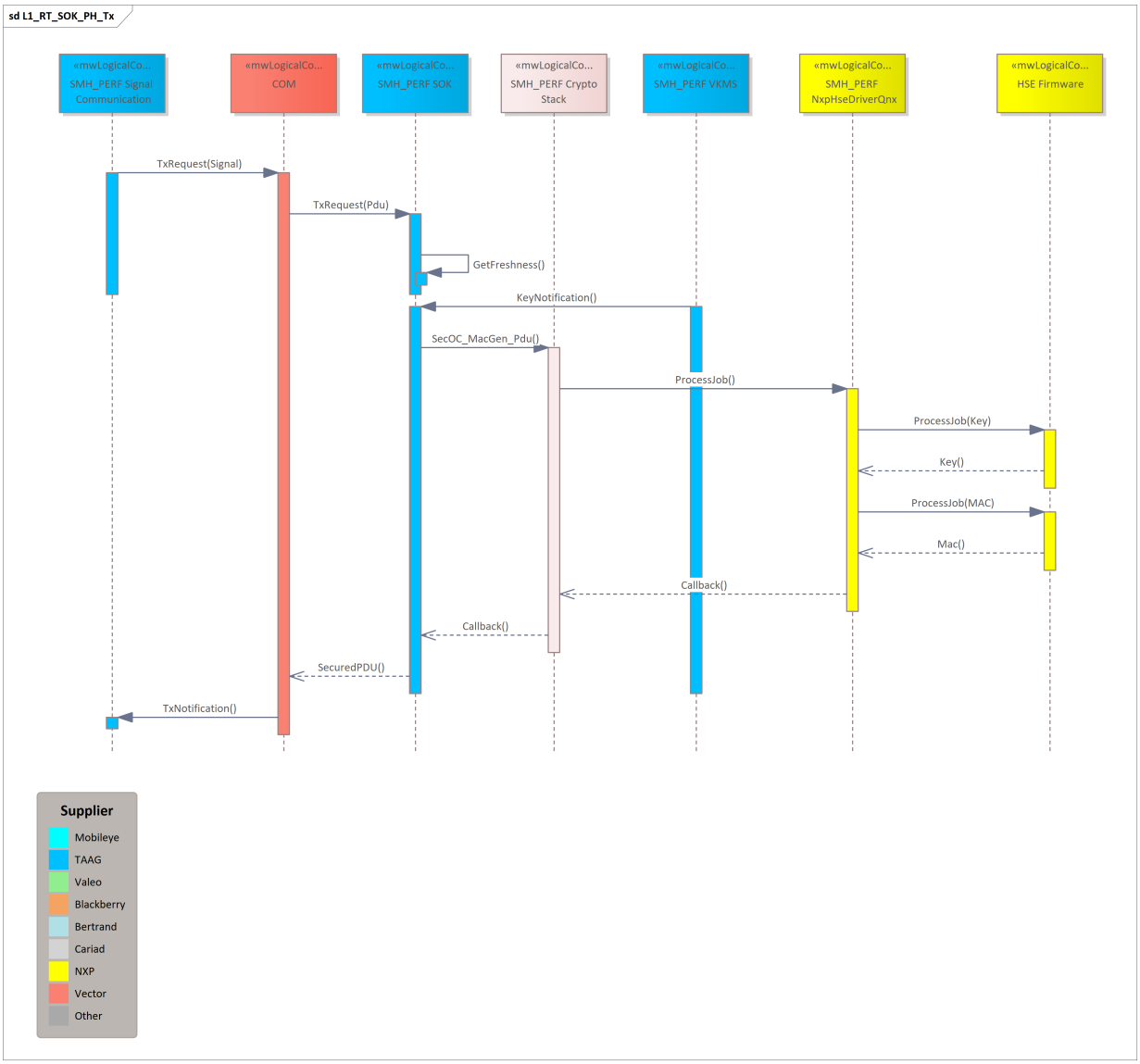
If PDUR receives authentic PDU, it forwards the authentic PDU to SecOC. Then SecOC will be responsible to generate secure PDU for Tx PDU and verification for Rx PDU. **[**S32GPRODP-296390 **]**



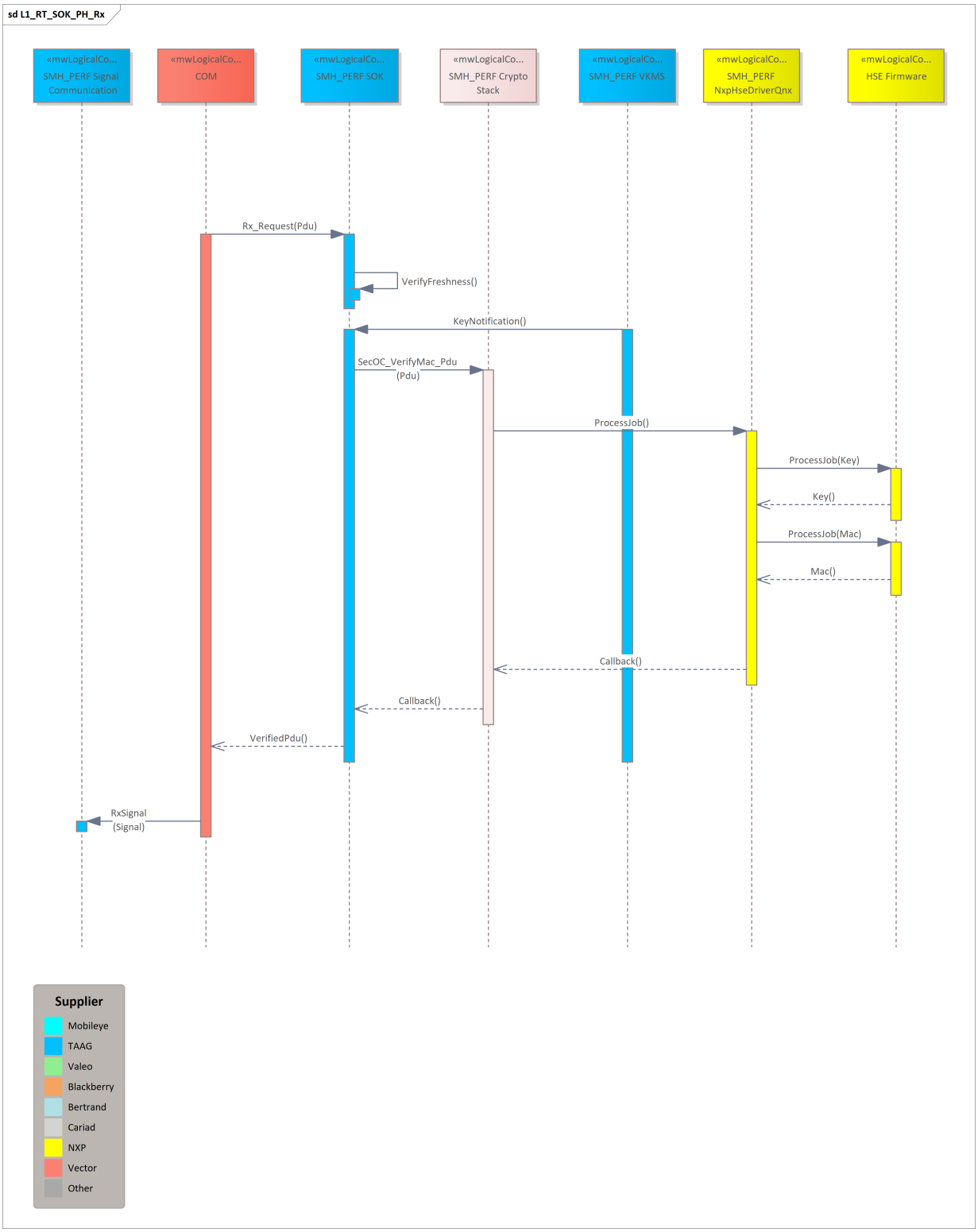
SOK on SH for receive PDUs, It verifies the freshness value and MAC for secured PDU. **[**S32GPRODP-296389 **]**



SOK on PH for transmit PDUs, It creates secure PDU by adding freshness value, truncated MAC to the authentic PDU. **[**S32GPRODP-296398 **]**



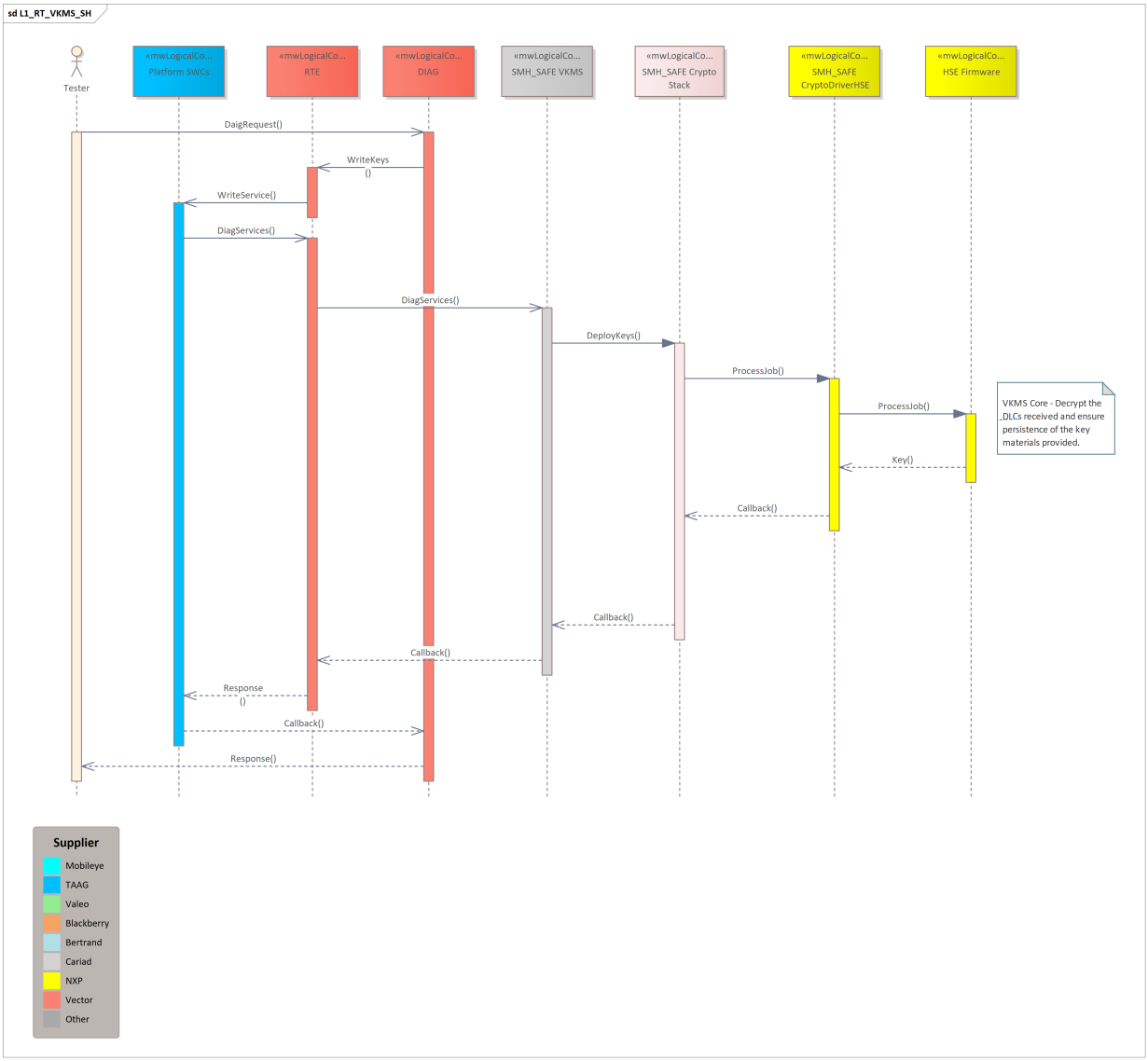
SOK on PH for receive PDUs, It verifies the freshness value and MAC for secured PDU. **[**S32GPRODP-296399 **]**


sd L1_RT_SOK_PH_Rx
«mwLogicalCo... SMH_PERF Signal Communication
«mwLogicalCo... COM
«mwLogicalCo... SMH_PERF SOK
«mwLogicalCo... SMH_PERF Crypto Stack
«mwLogicalCo... SMH_PERF VKMS
«mwLogicalCo... SMH_PERF NxpHseDriverQnx
«mwLogicalCo... HSE Firmware
Rx_Request(Pdu)
VerifyFreshness()
KeyNotification()
SecOC_VerifyMac_Pdu (Pdu)
ProcessJob()
ProcessJob(Key)
Key()
ProcessJob(Mac)
Mac()
Callback()
Callback()
VerifiedPdu()
RxSignal (Signal)
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

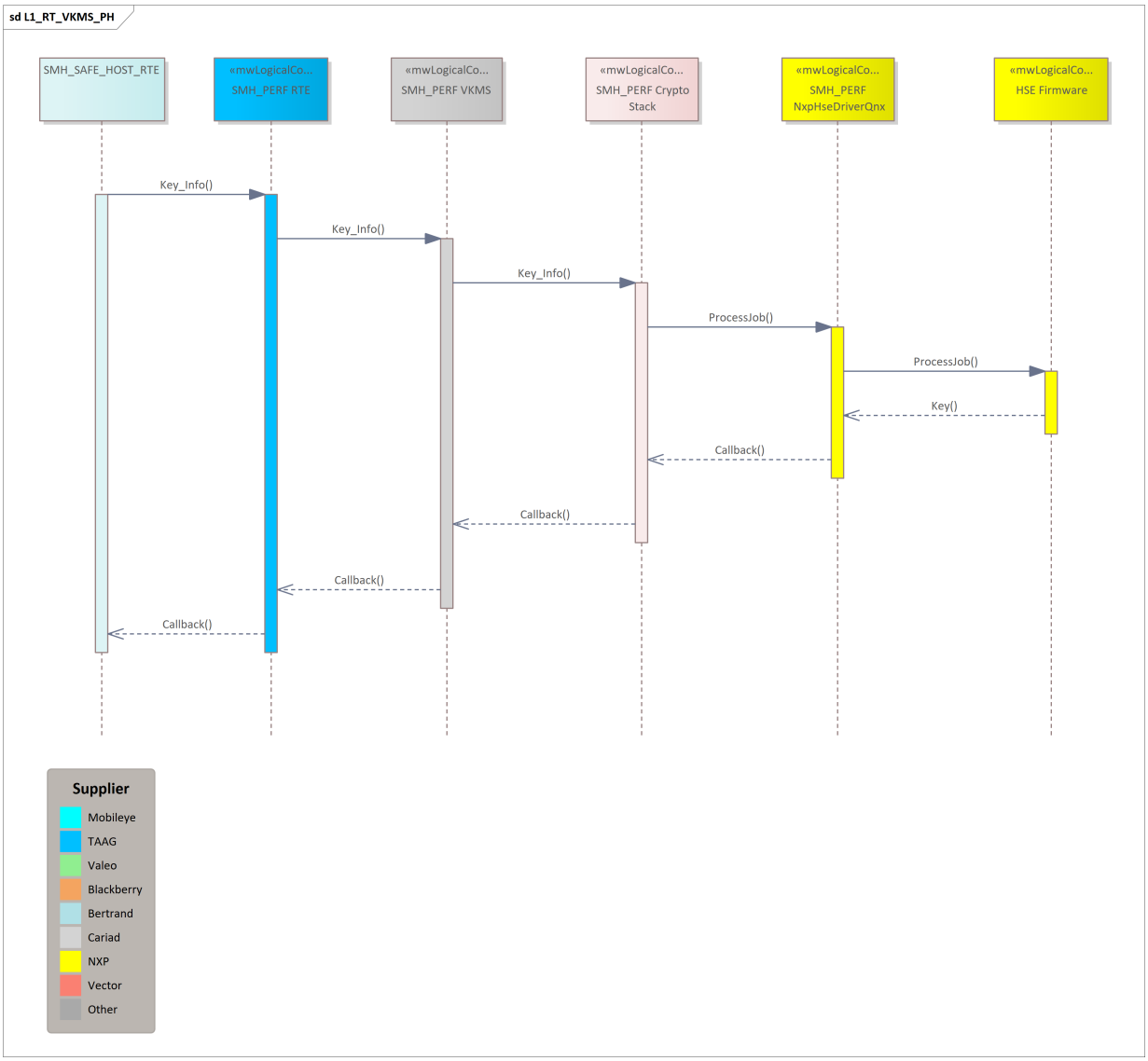
6.2.12.2 VKMS

VKMS of ECU consists of a single VKMS core, VKMS adapter on SMH-SH and multiple VKMS proxies. VKMS core is running inside HSM of SMH, VKMS proxies are running on each partition including the ones outside of SMH that require centralized key management services. **[**S32GPRODP-296392 **]**

The below sequence shows the deploying keys through VKMS Adapter on SH. **[**S32GPRODP-296394 **]**

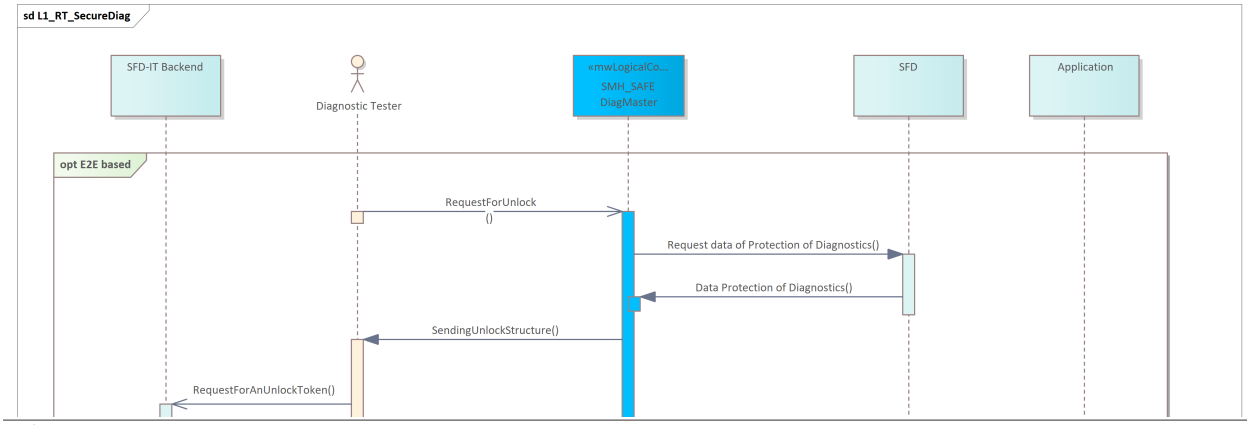


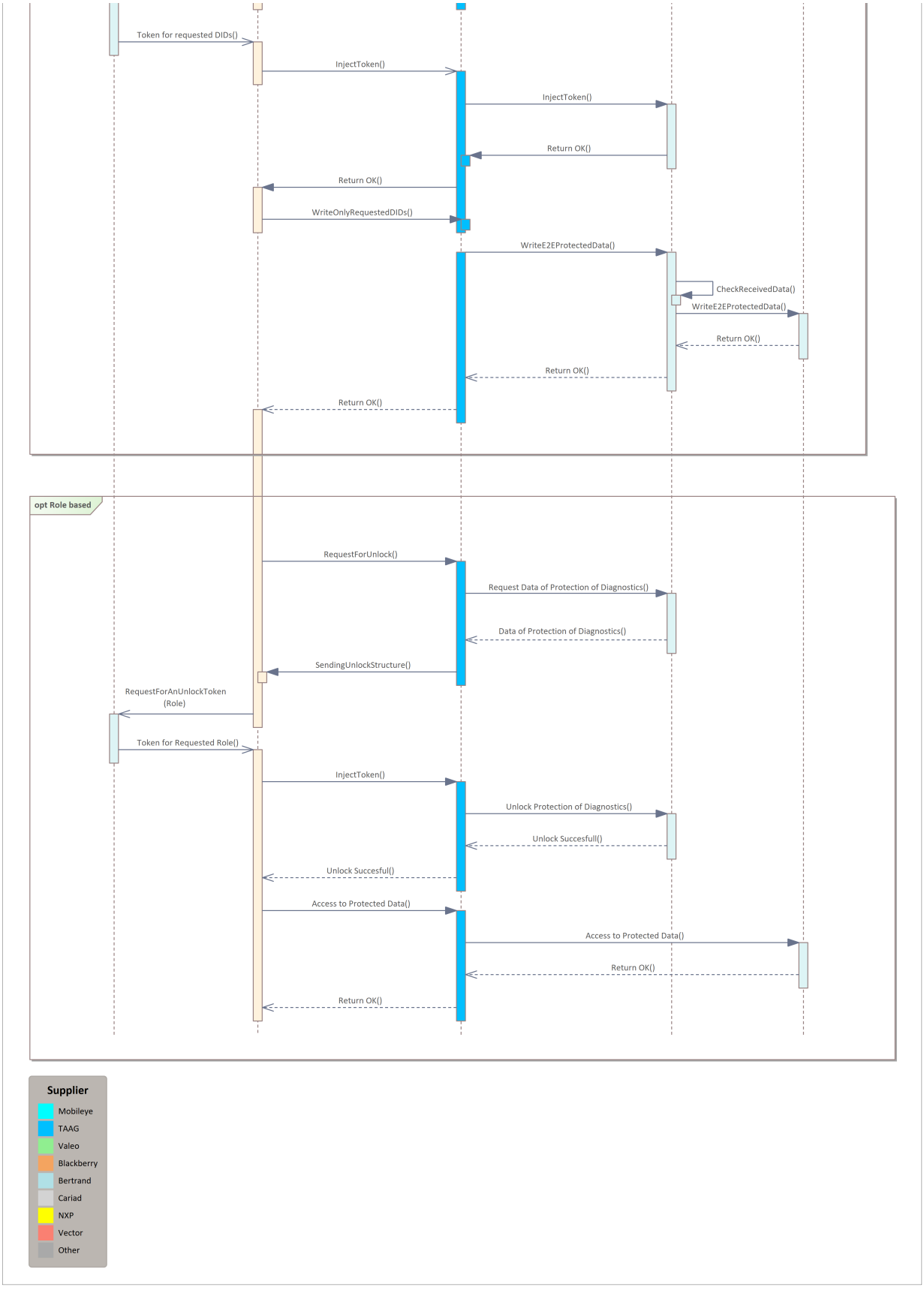
The below sequence show notifying the key information through VKMS proxy on PH. **[**S32GPRODP-296393 **]**



6.2.12.3 SFD (Secure Diagnostic)

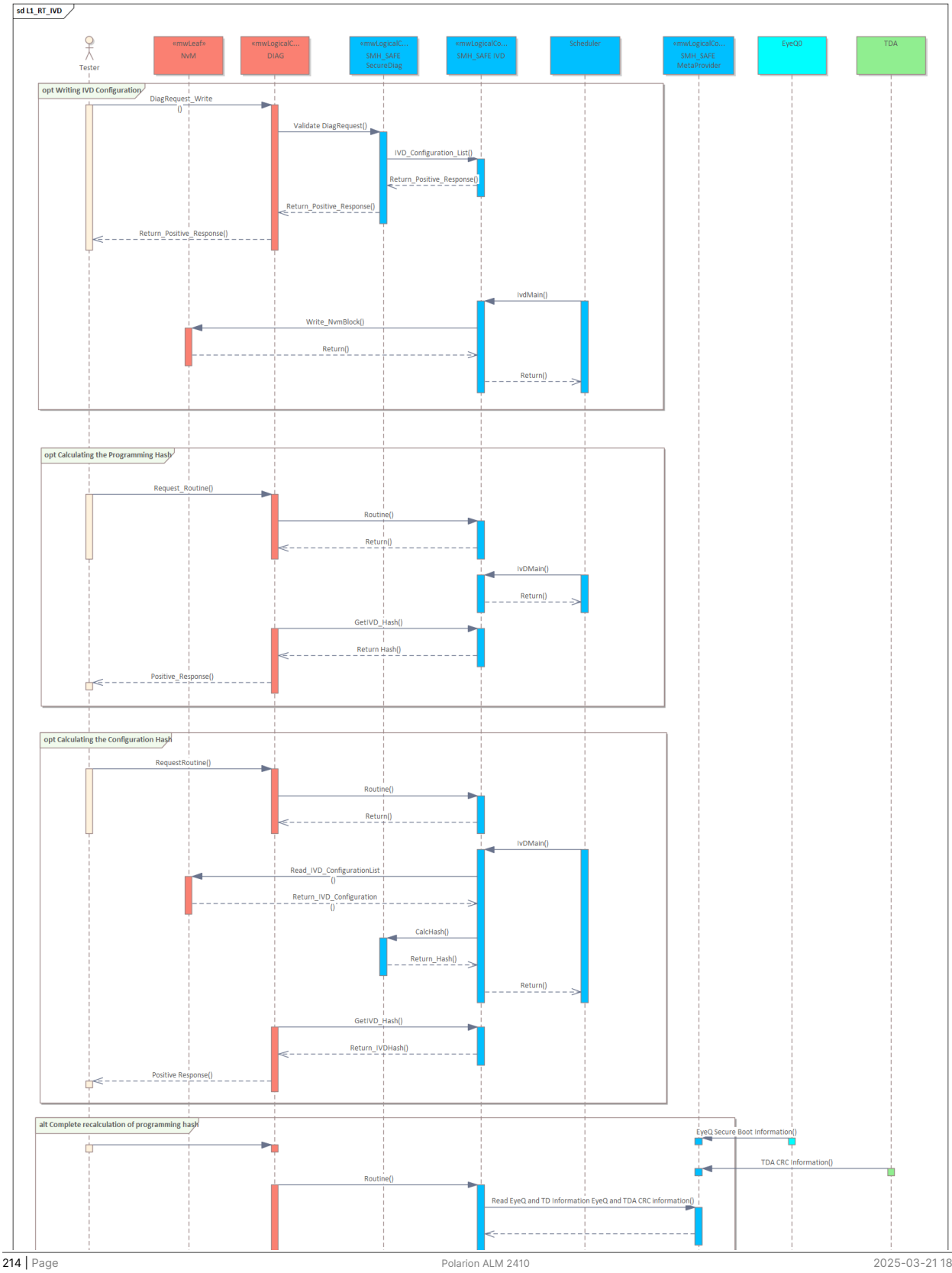
SFD is a module designed to secure specific diagnostic objects within an ECU to prevent unauthorized access. Each diagnostic object that is protected by SFD is either assigned a SFD Role (Role protected) or is end-to-end (E2E) protected. If the ECU does not have authorization to access a diagnostic object, it will receive a Security Access Denied. **[**S32GPRODP-296396 **]**

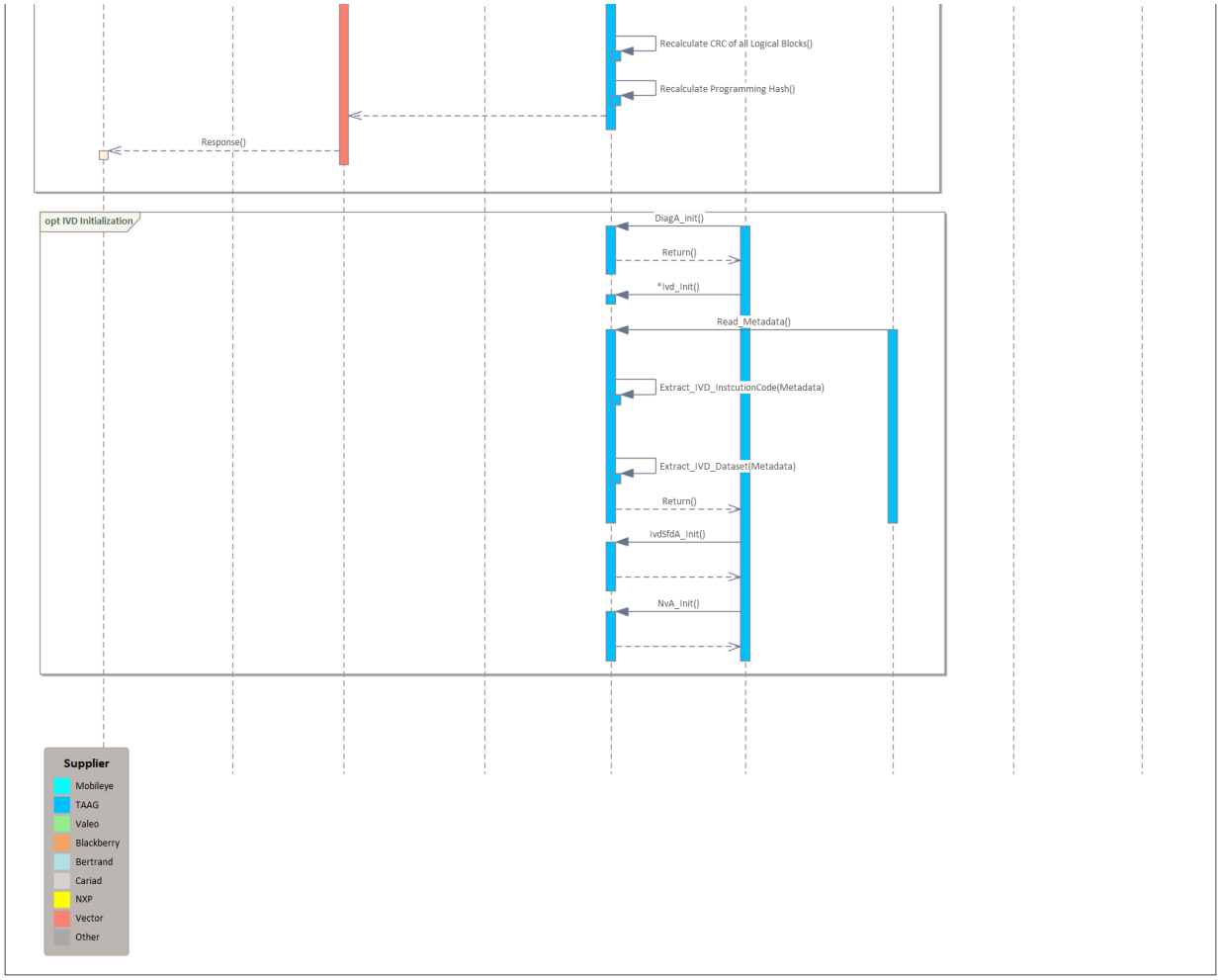



Token for requested DIDs()
InjectToken()
InjectToken()
Return OK()
Return OK()
WriteOnlyRequestedDIDs()
WriteE2EProtectedData()
CheckReceivedData()
WriteE2EProtectedData()
Return OK()
Return OK()
Return OK()
opt Role based
RequestForUnlock()
Request Data of Protection of Diagnostics()
Data of Protection of Diagnostics()
SendingUnlockStructure()
RequestForAnUnlockToken (Role)
Token for Requested Role()
InjectToken()
Unlock Protection of Diagnostics()
Unlock Succesfull()
Unlock Succesful()
Access to Protected Data()
Access to Protected Data()
Return OK()
Return OK()
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

6.2.12.4 IVD

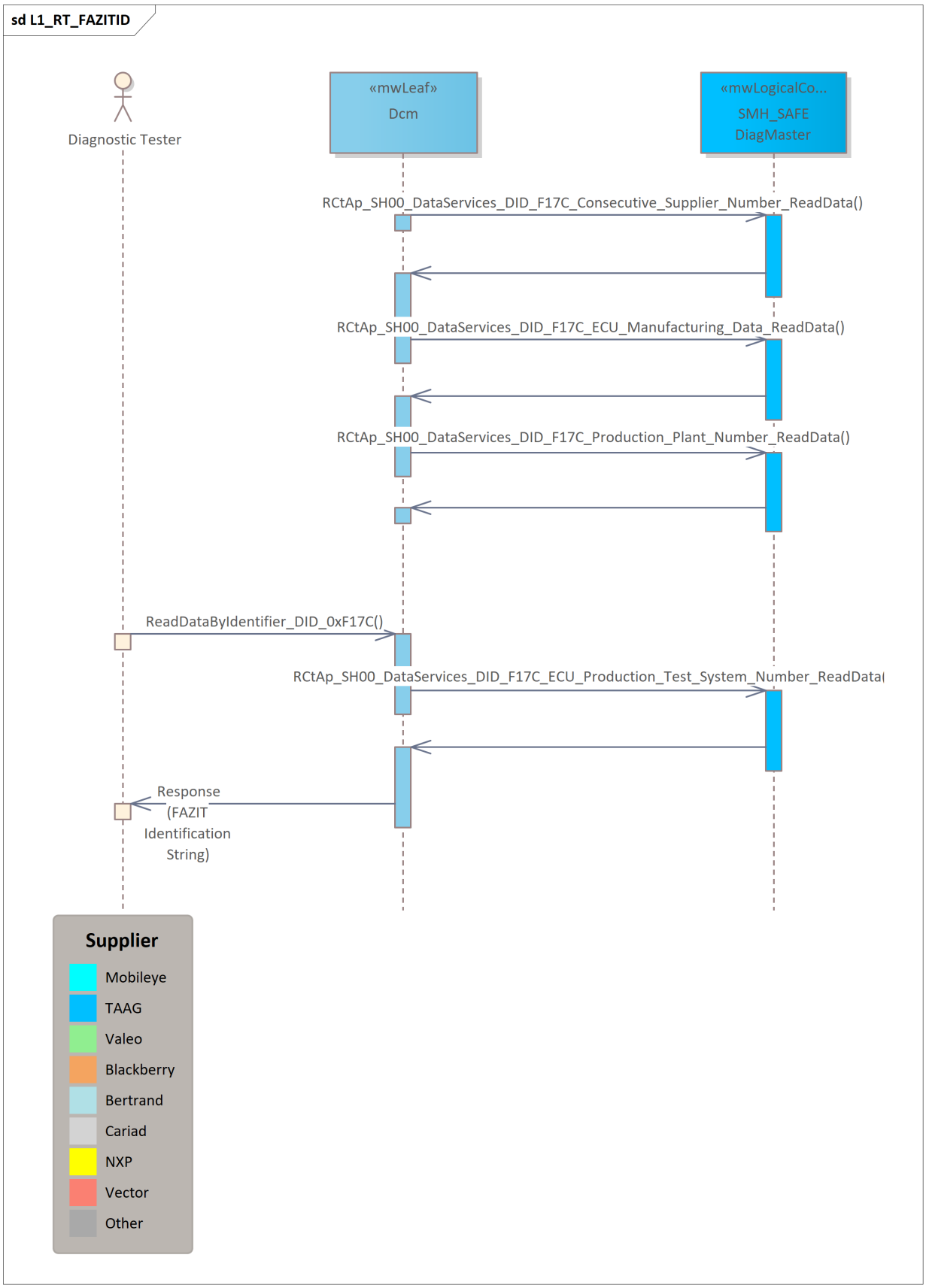
IVD (Integrity validation data) is a module that guarantees the integrity and authenticity of software. IVD uses two hash-values per ECU to determine the integrity of the software (by using a programming hash) and configuration (by using the configuration hash). **[**S32GPRODP-296395 **]**





6.2.12.5 Fazit id

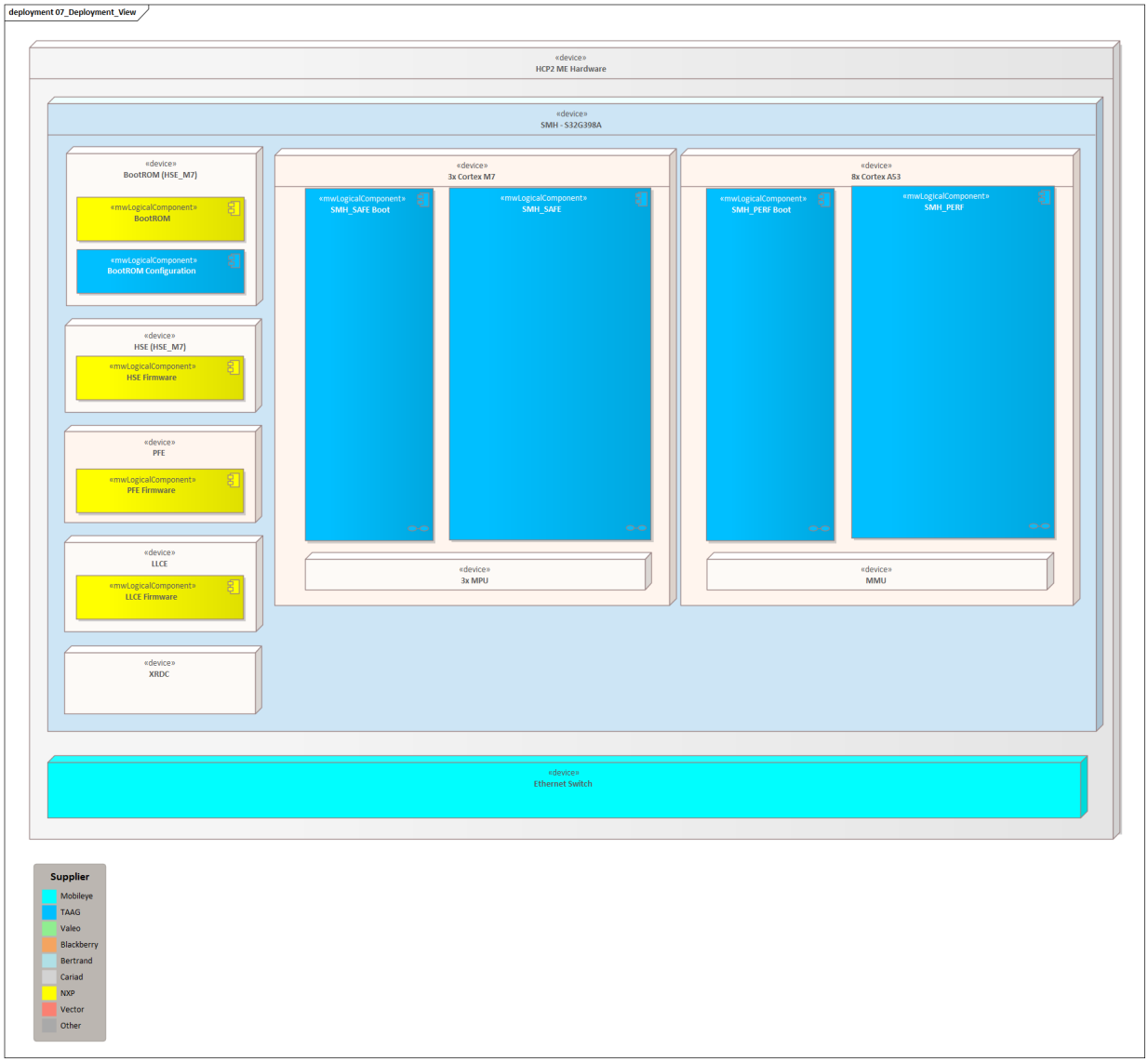
The FAZIT Identification String is used for assignment and tracking of every individual control unit. The FAZIT ID string will be stored within the MFG Data section (that contains also the HW Variant information, ECU Serial Number, APTIV Part Number, MAC Address...) and will be flashed at the EOL and protected by an OTP mechanism. **[**S32GPRODP-296400 **]**


sd L1_RT_FAZITID
Diagnostic Tester
«mwLeaf»
Dcm
«mwLogicalCo...
SMH_SAFE
DiagMaster
RCtAp_SH00_DataServices_DID_F17C_Consecutive_Supplier_Number_ReadData()
RCtAp_SH00_DataServices_DID_F17C_ECU_Manufacturing_Data_ReadData()
RCtAp_SH00_DataServices_DID_F17C_Production_Plant_Number_ReadData()
ReadDataByIdentifier_DID_0xF17C()
RCtAp_SH00_DataServices_DID_F17C_ECU_Production_Test_System_Number_ReadData
Response
(FAZIT
Identification
String)
Supplier
Mobileye
TAAG
Valeo
Blackberry
Bertrand
Cariad
NXP
Vector
Other

7 Deployment View

7.1 System level deployment

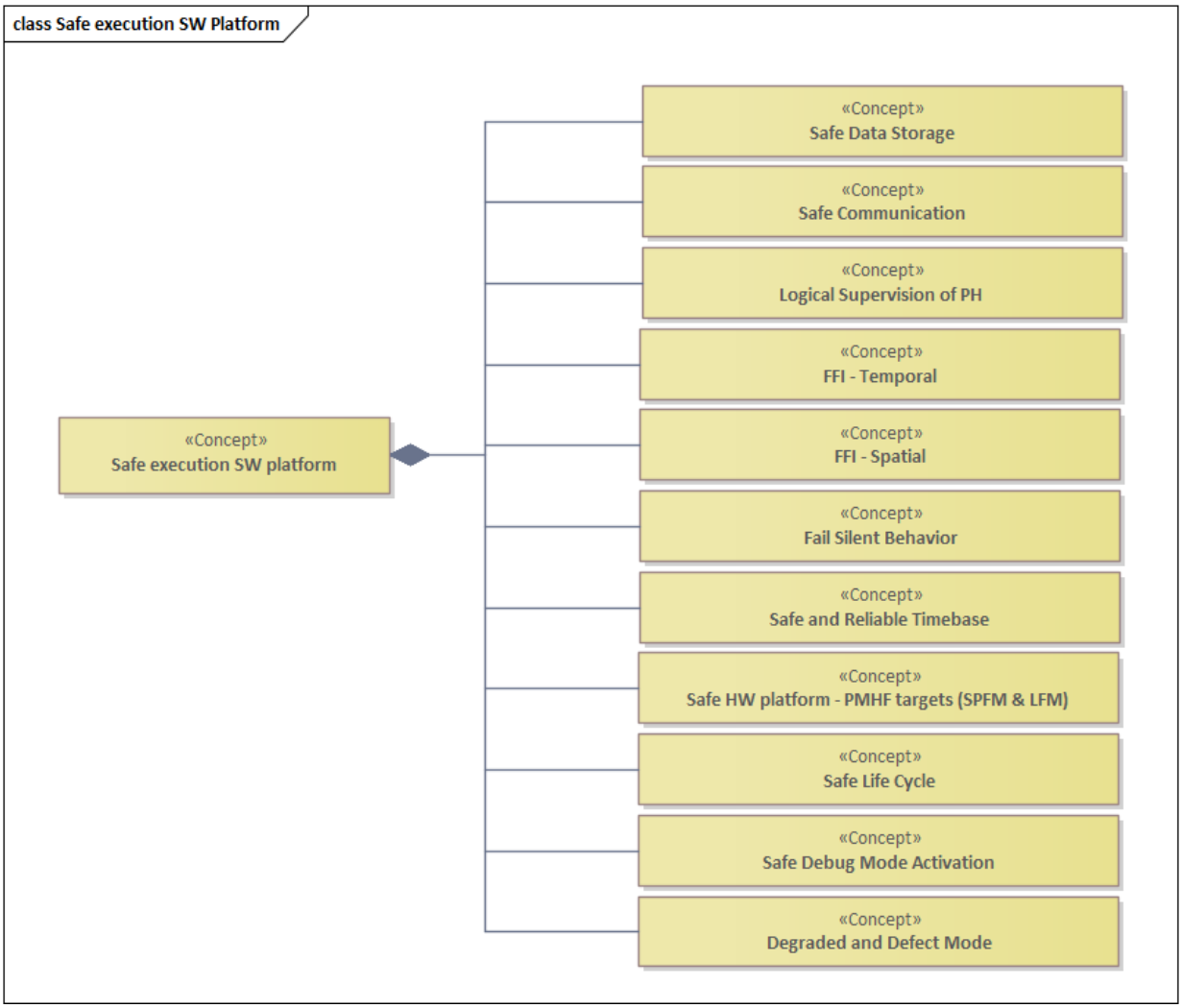
7.2 S32G3 device deployment



8 Crosscutting Concepts

8.1 High level Safety Concept (TSC)

Safe Execution SW platform (overall concept), is composed of following concepts: [HCP2MEP-148298]



These concepts above are decomposed further to the actual architecture items which are described in chapters 5 - Building Block View and 6 - Runtime View of this document. [HCP2MEP-226373]