

AUTOSAR SOW Appendix A



AUTOSAR Statement of Work Appendix A: AUTOSAR Integration Guide

Version: 2019.0
Date Issued 2019-08-27

Electrical/Electronic Systems Engineering
Architecture and Software Platform

FORD MOTOR COMPANY CONFIDENTIAL AND PROPRIETARY

This document contains Ford Motor Company confidential and proprietary information. Disclosure of the information contained in any portion of this document is not permitted without the expressed, written consent of a duly authorized representative of Ford Motor Company, Dearborn, Michigan, USA.

© Copyright 1998 – 2019, Ford Motor Company - All Rights Reserved

AUTOSAR SOW Appendix A

CHANGE LOG

Version	Revision Description	Author	Date
2019.0	Added Ford AUTOSAR workflow and comparison to ideal AUTOSAR workflow. Revised integration techniques section. Removed module descriptions and other information duplicated in AUTOSAR SWS documents.	STRAICOF	2019-08-27
2018.2	Removed table 2.	JMILOSER	2018-09-12
2017.1	Capture integration techniques for initial product release.	PRAJKU14	2017-08-31

AUTOSAR SOW Appendix A

TABLE OF CONTENTS

1	INTRODUCTION	4
1.1	Purpose.....	4
1.2	Scope.....	4
1.3	References	4
1.4	Abbreviations/Acronyms.....	Error! Bookmark not defined.
1.5	About This Specification	5
2	IMPLEMENTATION GUIDELINES.....	Error! Bookmark not defined.
2.1	Introduction to AUTOSAR	7
2.2	AUTOSAR Methodology.....	9
2.2.1	Ideal AUTOSAR Workflow	Error! Bookmark not defined.
2.2.2	Ford AUTOSAR Workflow.....	9
2.2.3	Work Products and Tools	9
3	AUTOSAR INTEGRATION RELATED TOPICS.....	Error! Bookmark not defined.
3.1	Mode Management.....	10
3.1.1	Power Up and Initialization.....	10
3.1.2	Run, Sleep, Wakeup, shutdown	10
3.2	Communication	10
3.2.1	Communication Management	10
3.2.2	CAN stack configuration	11
3.3	Watchdog Handling.....	12
3.4	Operating System	12
3.4.1	SCALABILITY Class.....	12
3.4.2	Scheduling Strategy.....	12
3.4.3	Stack MONITORING	12
3.4.4	Multicore Handling	12
3.5	Interrupt Handling.....	13
3.5.1	ISR Execution Times	13
3.5.2	Interrupt latency and CAN Rx Overwrites	13
3.5.3	Interrupt latency and Background Task	13
3.5.4	ISR Nesting	13
3.5.5	ISR Category.....	13
3.6	Application Design	14
3.6.1	Application SWCs	14
3.6.2	Complex Device Drivers (CDDs)	14
3.7	Diagnostics Handling	14
3.8	Memory Management.....	15
3.9	Security Handling.....	16
3.9.1	Crypto Services.....	16
3.10	E2E Protection	16
3.11	I/O Handling	16
3.12	Flash BootLoader	16
3.13	Ecu Configuration Tool Usage.....	16

AUTOSAR SOW Appendix A

1 INTRODUCTION

1.1 PURPOSE

This document is intended to provide guidance to software engineers of Ford suppliers who are required to deliver AUTOSAR compliant ECUs. These ECUs are mandated to use AUTOSAR BSW, RTE, OS, and Configuration Tools from the BSW Vendors approved by Ford. It addresses topics that may require advanced knowledge of AUTOSAR and its design. By using this document, supplier personnel can integrate the AUTOSAR product more swiftly, with fewer errors, and in the most preferred way desired by Ford Motor Company.

ATTENTION

THIS IS NOT A FORMAL PRODUCT REQUIREMENTS DOCUMENT.

1.2 SCOPE

This guideline covers concepts regarding integration of AUTOSAR BSW, RTE, Application SWCs, CDDs and MCALs. It covers aspects of integration outside of the feature domain.

1.3 REFERENCES

The following documents are either referenced by this specification, or contain information that is relevant to this specification.

#	Source	Title	Document Control No.
1	FMC	Generic Global Diagnostics Specification, Issue 005	00.06.15.001
2	AUTOSAR	Specification of ECU State Manager, release 4.3.1	078
3	AUTOSAR	Guide to Mode Management, release 4.3.1	440
4	AUTOSAR	Specification of RTE Software, release 4.3.1	084

Table 1: Applicable Documents Table

1.4 ACRONYMS AND ABBREVIATIONS

The following abbreviations and acronyms are used throughout this document:

Acronym	Term/Definition
AUTOSAR	Automotive Open System Architecture
ARXML	AUTOSAR XML (Extensible Markup Language)
ASIL	Automotive Software Integrity Level
BSW	AUTOSAR Basic Software
BswM	AUTOSAR Basic Software Mode Manager
CAN	Controller Area Network
CAN-FD	CAN with Flexible Data-Rate
CDD	AUTOSAR Complex Device Driver or Vector CANdela Diagnostic Data file
ComM	AUTOSAR Communication Manager
CPU	Central Processing Unit

AUTOSAR SOW Appendix A

Acronym	Term/Definition
Cry	AUTOSAR Crypto Library/Driver
CryIf	AUTOSAR Crypto Interface
Csm	AUTOSAR Crypto Service Manager
DBC	CAN Database file
Dcm	AUTOSAR Diagnostic Communication Manager
Dem	AUTOSAR Diagnostic Event Manager
DID	Data Identifier
DTC	Diagnostic Trouble Code
ECU	Electrical Control Unit
ECUC	AUTOSAR ECU Configuration
EcuM	AUTOSAR ECU State Manager
EESE	Ford Electrical/Electronic Systems Engineering
FMC	Ford Motor Company
GGDS	Generic Global Diagnostic Specification
HW	Hardware
ID	Identifier
ISR	Interrupt Service Routine
LDF	LIN Description File
LIN	Local Interconnect Network
MCAL	AUTOSAR Microcontroller Abstraction Layer
MPU	Memory Protection Unit
NM	Network Management
NV	Non-Volatile
NvM	AUTOSAR Non-Volatile RAM Manager
OS	Operating System
RAM	Random Access Memory
RMS	Rate Monotonic Scheduling
RTE	AUTOSAR Runtime Environment
Rx	Receive/Reception
SecOC	AUTOSAR Secure On-board Communication
SME	Subject Matter Expert
SW	Software
SWC	AUTOSAR Software Component
Tx	Transmit/Transmission

Table 2: Acronyms and Abbreviations

1.5 ABOUT THIS SPECIFICATION

This document is split into two main sections followed by an Appendix for showing examples.

AUTOSAR SOW Appendix A

Section 2.0 briefly describes the implementation aspects of AUTOSAR, the business model used to support it, and the formal mechanisms for handling exceptions.

Section 3.0 Integration guidelines and hints for using AUTOSAR efficiently and effectively to meet all Ford Motor Company (FMC), subsidiary, and affiliate production intent implementations of the ECU.

Disclaimer: *Though every attempt was made to cover all aspects of implementing AUTOSAR compliant ECUs for FMC there could still be design attributes that may lack detail or are overlooked. As these and other attributes are further studied requirements will be added or modified within this document to reflect the latest industry and corporate knowledge, directions, and trends. Future versions will be made available on a periodic and as needed basis only.*

AUTOSAR SOW Appendix A

2 PLATFORM AND METHODOLOGY

2.1 INTRODUCTION TO AUTOSAR

With ever increasing on-board vehicular networking and diagnostic demands coupled with an expanding supply base, the desire to commonize software components across automotive systems is necessary. Hence there is a need of standardization of software development in automotive domain. AUTOSAR fits in perfectly to realize this need.

AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of vehicle manufacturers, suppliers and other companies from the electronics, semiconductor and software industry.

AUTOSAR:

- Paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness
- Is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation"
- Is a key enabling technology to manage growing electrical/electronic complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality
- Facilitates the exchange and update of software and hardware over the service life of the vehicle

AUTOSAR provides a set of specifications that describes basic software modules, defines application interfaces and builds a common development methodology based on standardized exchange format. Basic software modules made available by the AUTOSAR layered software architecture can be used in vehicles of different manufacturers and electronic components of different suppliers, thereby reducing expenditures for research and development and mastering the growing complexity of automotive electronic and software architectures.

AUTOSAR uses a three-layered architecture:

- **Basic Software (BSW):** Standardized software modules without any functional job themselves that offer services necessary to run the functional part of the application.
- **Runtime Environment (RTE):** Middleware which abstracts from the network topology for the inter- and intra-ECU information exchange between the Application Software Components (SWCs) and between the BSW and Application SWCs.
- **Application Layer:** Application Software Components (SWCs) that interact with each other and the BSW through the RTE.

Note: Refer to Figure 1 for detailed AUTOSAR Layered Architecture.

The AUTOSAR development methodology defines a set of tools and activities used by various development roles to generate work products. Ford does not currently use the entire defined AUTOSAR methodology, but instead parts of it. Ford's current adoption of the methodology is further described in section 2.2.1.

AUTOSAR SOW Appendix A

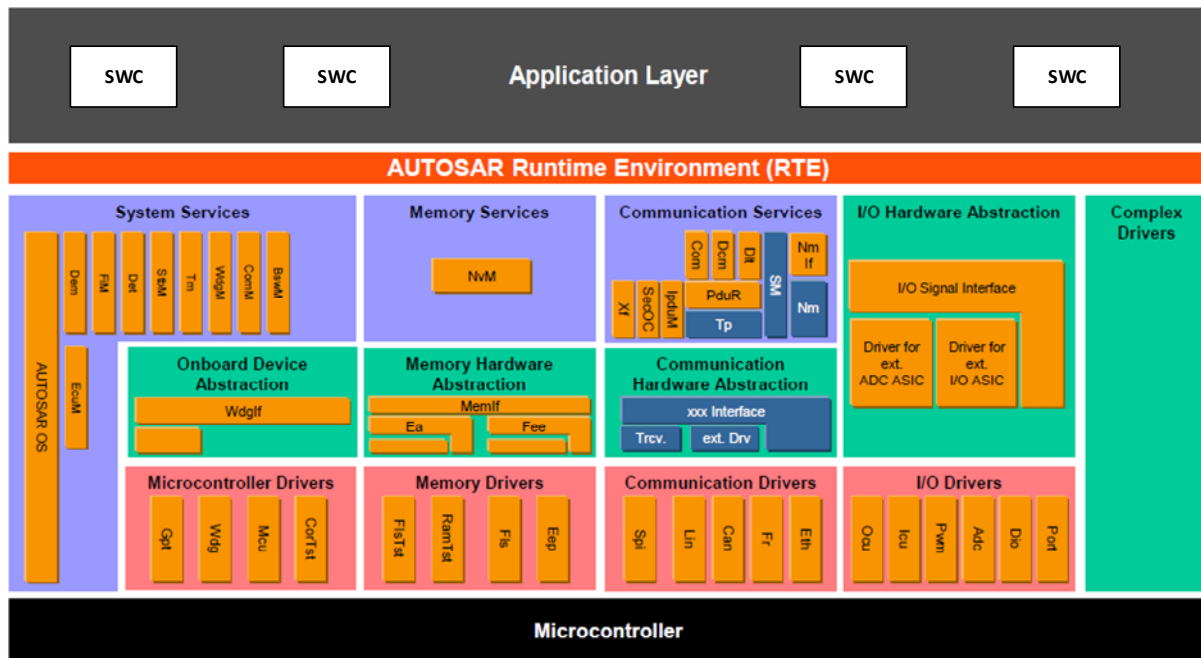


Figure 1: AUTOSAR Architecture

AUTOSAR SOW Appendix A

2.2 AUTOSAR METHODOLOGY

AUTOSAR Methodology involves exchange of information in the standardized ARXML (AUTOSAR Extensible Markup Language) format. Ford does not currently follow the complete AUTOSAR Methodology with regards to the items used or provided to the Tier 1 Supplier.

2.2.1 FORD AUTOSAR WORKFLOW

Ford does not currently use all steps of the ideal AUTOSAR workflow. The current workflow is described below.

- The ECU Supplier will receive communication artifacts from Ford.
 - If the ECU supports CAN/CAN-FD, this will be one or more DBC files.
 - If the ECU supports LIN, this will be one or more LDF files.
 - If the ECU supports AUTOSAR-based Ethernet, this will be in an AUTOSAR ECU Extract.
- For DBC or LDF files, the ECU Supplier shall use their BSW vendor's AUTOSAR tooling to convert these files into an ECU Extract.
 - If multiple communication buses are used in the ECU, multiple input files need to be converted to a single ECU Extract.
- The ECU Supplier shall create appropriate Application Software components and complete the contents of the ECU Extract using an AUTOSAR Authoring Tool.
 - Depending on the ECU use case and functionality, Ford may also provide SWCs to the ECU Supplier which need to be integrated into the ECU Extract.
- The ECU Supplier shall configure the AUTOSAR BSW, OS, and RTE in the AUTOSAR ECU Configuration tool.
 - This will include configuring GGDS and possibly OBD-II diagnostics. A Vector CANdela Diagnostic Data document (CDD) can be imported to partially configure Diagnostic modules (DCM and DEM).
- The ECU Supplier shall generate appropriate code for SWCs, BSW, OS, and RTE. This shall be built together with static code to generate an executable.

2.2.2 WORK PRODUCTS AND TOOLS

The below table depicts the different work products, tools required and responsibilities:

#	Work Product	Tool	Responsibility
1	CAN Database (DBC)	DBC Editor	Ford
2	LIN Database (LDF)	LDF Editor	Ford
3	Partial ECU Extract as input for Ethernet	AUTOSAR Authoring Tool	Ford
4	Software Component Description	AUTOSAR Authoring Tool	Ford and/or ECU Supplier
5	Completed ECU Extract	AUTOSAR Authoring Tool	ECU Supplier
6	Diagnostic database (CDD, ODX)	Vector CANdela Studio	ECU Supplier
7	ECU Configuration Description(s)	AUTOSAR ECU Configuration Tool	ECU Supplier

Table 3: Work Products and Tools

AUTOSAR SOW Appendix A

3 AUTOSAR INTEGRATION TOPICS

This section outlines some guidelines and recommendations for AUTOSAR software integration. Descriptions of core software functionality shall be found in the related public AUTOSAR documents.

3.1 MODE MANAGEMENT

The Mode Management and ECU State Management shall be handled by the AUTOSAR BSW modules Basic Software Mode Manager (BswM) and ECU State Manager (EcuM).

The BSW shall be configured to fulfill the requirements for the appropriate ECU power type:

- Switched: Power is switched on/off due to power input (e.g., ignition switch).
- Latched: Power is switched on, latched, and removed at discretion of ECU application logic.
- Sleep: Power is applied at all times while battery is connected. Sleep state minimizes current draw by using low power mode of microprocessor. Cyclic micro wakeups from low power mode are used to evaluate wakeup input circuits.
- Unpowered Sleep: Power is applied at all times while battery is connected. Sleep state minimizes current draw by removing power from microcontroller. Transceiver is used to evaluate wakeup input circuits.

3.1.1 POWER UP AND INITIALIZATION

In AUTOSAR, EcuM is the first component that needs to be initialized. It is recommended that the integrator take the following steps at initialization time:

1. Execute the microcontroller start-up code, which in turn shall call the main() function
2. Within the main() function, the code shall perform any necessary initialization (per the BSW or OS vendor) then call EcuM_Init() as the last step. EcuM_Init() does not return to the caller.
3. It is recommended that BSW module initialization done by the EcuM follow either the order given in [2] or [3]. This assumes use of EcuM Flexible.
4. An OS task shall be coded to invoke EcuM_StartupTwo(). This task shall be configured in the following way.
 - a. Basic Task
 - b. Highest priority in system
 - c. Auto-started

3.1.2 RUN, SLEEP, WAKEUP, SHUTDOWN

These states shall be handled with EcuM and BswM depending on ECU requirements.

If using Vector BSW, it is allowed to use DaVinci Configurator's BswM Auto-Configuration for configuring this.

Note: Ford EESE Architecture and Software Platform is working on creating a Mode Management user guide.

3.2 COMMUNICATION

3.2.1 COMMUNICATION MANAGEMENT

The Communication Manager (ComM) module handles resource management by controlling of more than one communication bus channel of an ECU by implementing a channel state machine for every channel.

AUTOSAR SOW Appendix A

ComM abstracts the type of communication channel from the user (SWC) by providing mapping of Users to Channels (ComMUser, ComMChannel, ComMUserPerChannel). It also coordinates the availability of the bus communication stack (allow sending and receiving of signals) of multiple independent software components on one ECU.

Different Communication modes supported by AUTOSAR communication stack are:

- Full Communication – Both Tx and Rx is allowed
- No Communication – No Tx and Rx
- Silent Communication – Only Rx allowed (Internal state/mode, necessary for synchronization)

Configuration of Communication Management:

1. If the ECU is an Active ECU, Application SWC can request Full Communication or No Communication directly to ComM through Client-Server Port interfaces or through BswM.
2. In case of Passive ECU, the ECU shall wait for wakeup message to keep the application active.
3. ComM_CommunicationAllowed API needs be invoked through BswM configuration to Enable/Disable the communication on a given channel.
4. Starting and stopping IPDU group (IPDU Group switching) needs to be handled in BswM based on the notification received from ComM.
5. ComM provides configuration parameters & APIs to set ECU Group Classification for Mode Inhibition (Limitation) or Bus Wakeup Inhibition. This feature needs to be used appropriately.

3.2.2 CAN STACK CONFIGURATION

The CAN Communication Services are a group of modules for vehicle network communication with the communication system CAN. It provides a uniform interface to the CAN network.

The CAN Communication Stack supports:

- Classic CAN communication (CAN 2.0)
- CAN FD communication, if supported by hardware

3.2.2.1 HARDWARE ACCEPTANCE FILTERS

The CAN hardware acceptance filters should be optimized to filter out as many of the irrelevant passing messages as possible. This helps reduce unnecessary interrupts. This can be achieved through the configuration parameters 'CanHwFilterCode' and 'CanHwFilterMask' of CAN Driver.

3.2.2.2 SEARCH ALGORITHM & RX INTERRUPT RUNTIME

If flash memory is not an issue, the Rx interrupt service routine run time can be greatly reduced by configuring CanIfPrivateSoftwareFilterType based on the performance requirements of the controller. This is only relevant to BasicCAN and when more than 10 messages are received.

Note: 'CanIfPrivateSoftwareFilterType' needs to be configured based on the Performance requirements.

3.2.2.3 TX/RX FULLCAN LAYOUT

There are several factors which influence the balancing the FullCAN layout:

1. If there are ample FullCAN objects, assign all application Tx & Rx IDs to FullCAN.
2. If Priority Inversion Avoidance (PIA) is a required, Tx IDs should be assigned to FullCAN. In cases where this is difficult to achieve, approval is needed from the responsible Ford SME.
3. Rx frames with signals states which absolutely must be received should be assigned to FullCAN objects.

AUTOSAR SOW Appendix A

4. The IDs received should be balanced among the filters used. Having some filters wide open increases the chance of data overwrites.

3.2.2.4 CAN HARDWARE FAILURES

CAN HW failures can occur at CAN cell initialization, bus off recovery, power up, reset, sleep, and wakeup. The CAN driver loops waiting for hardware acknowledgement. The integrator needs to refer the controller reference manual and CAN Driver implementation to determine how to handle these.

3.3 WATCHDOG HANDLING

Triggering of the hardware Watchdog needs to be handled only through Watchdog Manager (WdgM) module. There shall not be any direct interaction to HW/Driver.

3.4 OPERATING SYSTEM

3.4.1 SCALABILITY CLASS

The supplier should ensure that the OS implements the correct required Scalability Class:

- Scalability Class 1 doesn't support Timing Protection or Memory Protection
- Scalability Class 2 supports Timing Protection but not Memory Protection
- Scalability Class 3 supports Memory Protection but not Timing Protection
- Scalability Class 4 supports both Timing Protection and Memory Protection

For an ECU which must implement software to support a higher ASIL rating, the OS can provide Timing and/or Memory Protection; a hardware MPU is required for Memory Protection. When an MPU is used, Stack Memory will be designated as non-executable.

This is driven by the Software Functional Safety requirements and guidelines provided by Ford.

3.4.2 SCHEDULING STRATEGY

AUTOSAR OS supports fixed priority-based FIFO scheduling. The priority of the OS Tasks drives the scheduling. The System Integrator needs to design the required Tasks and assign proper priority to each of these Tasks .

1. Scheduling is handled by the AUTOSAR OS and RTE for all Runnable Entities. There shall not be any scheduling in Application.
2. The Application shall be decomposed into SWCs containing one or more Runnable Entities. It remains the System Integrator's responsibility to perform 'Runnable-to-Task Mapping' correctly while configuring the AUTOSAR RTE.

Any exception to this need shall be approved by Ford.

3.4.3 STACK MONITORING

1. Stack monitoring of Tasks/Category 2 ISRs 'OsStackMonitoring' needs to be configured as 'true' if available.

3.4.4 MULTICORE HANDLING

In case of multicore ECU, the following configurations need to be handled:

AUTOSAR SOW Appendix A

1. In AUTOSAR Operating System module, the parameter 'OsNumberOfCores' needs to be configured with Maximum number of cores that are controlled by the OS.
2. Each OsApplication needs to have reference to 'EcucCoreDefinition' through the configuration parameter 'OsApplicationCoreRef'.
3. In AUTOSAR ECU State Manager module, the parameter 'EcuMPartitionRef' needs to refer to the 'EcucPartition'. Each partition need to have one 'EcuMPartitionRef'. Also, for each core there needs to be 'EcuMOSResource' configured which refers to 'OsResource'.

3.5 INTERRUPT HANDLING

3.5.1 ISR EXECUTION TIMES

Measurements of the runtimes of all interrupt service routines should be made to evaluate the robustness of the software architecture. Also necessary is an understanding of the frequency of each interrupt.

3.5.2 INTERRUPT LATENCY AND CAN RX OVERWRITES

Interrupt latency (including higher priority interrupts if nesting is used) should be examined to ensure CAN data overwrites are unlikely. If the latency is excessive, a number of ways can be employed to remedy the situation:

1. Assign IDs vulnerable to overwrites to FullCAN objects.
2. Optimize the CAN Rx interrupt run time (see search algorithm below).
3. Keep ISR run times to a minimum by only including code statements which are absolutely necessary.
4. If possible, allocate two hardware buffers per acceptance filter to double the time the CPU has to process the Rx message before it gets overwritten.
5. If possible, increase the number of available acceptance filters to allow for better filter optimization and blocking of unwanted messages.
6. Distribute the Rx IDs between the filters such that none of the filters are wide open.

3.5.3 INTERRUPT LATENCY AND BACKGROUND TASK

The interrupt latency should not be so excessive as to starve the application software of running in the background during high bus load conditions.

3.5.4 ISR NESTING

If interrupt nesting is employed, it is paramount that the CAN interrupts have the same interrupt priority. This would preclude one CAN interrupt service routine from preempting another and corrupting shared resources.

3.5.5 ISR CATEGORY

AUTOSAR OS supports Interrupts of types Category 1 and Category 2. Category 1 ISRs are not handled by operating system and the operating system is not aware of any Category 1 ISRs being invoked. Category 2 ISRs are completely handled by operating system.

1. If Category 1 interrupts AND OS-Applications are used together, then all Category 1 ISRs must belong to a trusted OS-Application.
2. With exception of highly time critical interrupt handling, all ISRs are recommended to be configured as Category 2.

AUTOSAR SOW Appendix A

3.6 APPLICATION DESIGN

3.6.1 APPLICATION SWCS

The Application needs to be designed using an AUTOSAR Authoring Tool. The Application should constitute multiple independent Atomic Software Components (SWCs). An Atomic Software Component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.

An Atomic Software Component can be one of the following different types:

- ApplicationSwComponentType
 - EcuAbstractionSwComponentType
 - NvBlockSwComponentType
 - SensorActuatorSwComponentType
 - ComplexDeviceDriverSwComponentType
 - ServiceSwComponentType
 - ServiceProxySwComponentType
 - ParameterSwComponentType
1. In case of Application development by Ford, the teams involved share decide on the exchange formats (e.g., source code, object code, or models).
 2. In case of Application collaboration between multiple Tier 1 Suppliers, exchange formats must be agreed upon at time of sourcing.
 3. In case of multiple RunnableEntities exchanging information between them within the same SWC, Inter Runnable Variables or Exclusive Areas need to be used. RunnableEntity can have data protection through configuration of Exclusive Areas.

3.6.2 COMPLEX DEVICE DRIVERS (CDDS)

In case of the functionalities which are not supported by the AUTOSAR BSW & in case of time-critical/complex sensor or actuator implementation, the AUTOSAR concept of Complex Device Drivers (CDD) can be used.

All CDDs need to be developed with coding standards defined by Ford and need to be reviewed with Ford team.

3.7 DIAGNOSTICS HANDLING


Diagnostics in AUTOSAR is mainly handled by Diagnostic Communication Manager (Dcm) and Diagnostic Event Manager (Dem) modules. Diagnostics need to be implemented as per Ford GGDS specification [1].

1. The Diagnostics configuration is stored in the format of Vector CANdela Diagnostic Data file (.cdd). The integrator needs to import this file into the BSW vendor's ECU Configuration Tool and ensure Dcm and Dem modules are configured correctly based on the content in the input files.
2. Configuration parameter 'DcmDslBufferSize' represents the size of the diagnostic buffer in bytes. The integrator needs to ensure this configured to sufficiently handle all the Diagnostics services.

The paged buffer can be used (parameter 'DcmPagedBufferEnabled') to reduce RAM usage used for diagnostic responses. The Diagnostic buffer is the RAM buffer used to store diagnostic requests and responses. Analysis is needed prior to selecting the size in the ECU Configuration Tool.

AUTOSAR SOW Appendix A

- a. Paged buffer enabled: the diagnostic buffer must be as long as the longest diagnostic request from the tester.
- b. Paged buffer disabled: the diagnostic buffer must be as long as the longest request from the tester, or as long as the longest response whichever is longer.
- c. Paged buffer disabled: the diagnostic buffer must be long enough to report all configured DTCs – This takes 4 bytes per DTC plus three bytes of overhead:
(4 bytes * number of DTCs + 3)
- d. Paged buffer disabled: the diagnostic buffer must be long enough to report the maximum number of identifiers per request times the longest request in bytes as follows:
(maximum # of identifiers per request x (largest DID + 2)) + 1

 CAUTION	Diagnostic buffer size analysis is imperative since under sizing it can result in writing past the end of the buffer and corrupting contiguous memory! A memory corruption issue can be next to impossible to troubleshoot depending on the memory corrupted. Oversizing the buffer is not an issue.
---	--

Note: The maximum number of identifiers per request can be configured in the CANdela database. The minimum numbered allowed is two. Please see the figure below for how to configure.

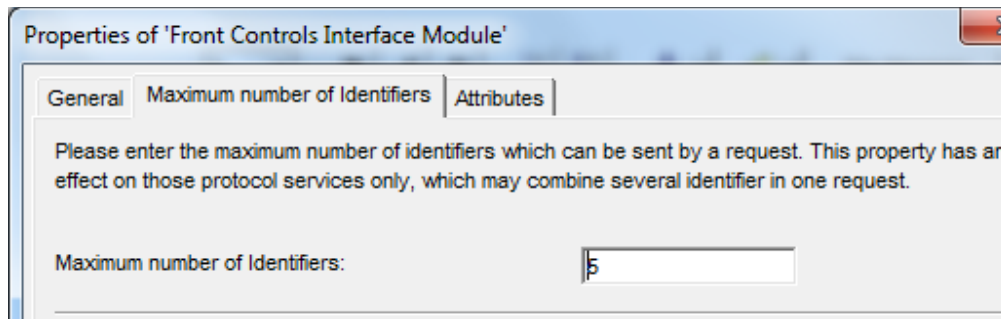


Figure 2: Maximum Number of DIDs per Request

3.8 MEMORY MANAGEMENT

Data storage and maintenance of NV (non-volatile) data needs to handle through AUTOSAR NVRAM Manager Module (NvM). AUTOSAR NvM provides the required synchronous/asynchronous services for the management and the maintenance of NV data (init/read/write/control).

1. During Power up, the NV data from EEPROM/Flash needs to be read into RAM by configuring a call to the API NvM_ReadAll in BswM.
2. During Power down or transition into sleep, the NV data from RAM block needs to be written to EEPROM/Flash by configuring a call to the API NvM_WriteAll in BswM before entering No Power or Low Power Mode.

AUTOSAR SOW Appendix A

3. On demand data read/write/control can be handled through the Client Server Interfaces provided by NVM module.
4. Use of the `NvBlockSwComponentType` is recommended when NV data is shared between Application SWCs. See the RTE Software Specification [4] for further details.

3.9 SECURITY HANDLING

The ECUs which requires Cyber Security features for CAN Message Authentication need to have AUTOSAR SecOC and the Crypto stack (Csm, Crylf, and Crypto driver modules).

3.9.1 CRYPTO SERVICES

The mechanism for Software Security to be handled as per Requirements & Guidelines provided by Ford.

Note: Ford EESE Architecture and Software Platform will provide a separate guide for integrating the CAN Message Authentication modules.

3.10 E2E PROTECTION

End-to-End (E2E) Protection can protect Functional Safety-related data exchange against runtime faults. The mechanism and profile needs to be chosen based on the Software Functional Safety Requirements & Guidelines for the particular ECU.

3.11 I/O HANDLING

1. The I/O Hardware Abstraction (SWC) module shall be modeled as an SWC of `EcuAbstractionSwComponentType`.
2. Sensors and Actuators within the Application shall be modeled as Sensor and Actuator SWCs, respectively, with interfaces to the `IoHwAb` SWC.
3. Where necessary or appropriate, Complex Device Drivers may be used for I/O.
 - a. In case a required hardware driver is not defined by AUTOSAR, it needs to be implemented as Complex Device Driver.
 - b. All CDDs need to be developed with coding standards defined by Ford and need to be reviewed with Ford team.

3.12 FLASH BOOTLOADER

The Flash Bootloader is out of scope for AUTOSAR Integration. However, care needs to be taken to handle the diagnostics services in Bootloader in accordance with the Application.

3.13 ECU CONFIGURATION TOOL USAGE

For the details of features and usage of ECU Configuration Tool, the integration needs to refer to the BSW vendor ECU Configuration Tool user guide.

In case Vector AUTOSAR BSW is used, please consult the following documents delivered with the software:

- `Startup_Ford_SLP1.pdf`