Document Title	Explanation of Firmware Over-The-Air
Document Category	Explanation (EXP)
Document UID	945
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• No content changes
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• Rework FOTA internal buffer handling • Refined FOTA state machine • Refined Rollback procedure description • Editorial reworks
2019-11-28	R19-11	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction 5
1.1 Motivation . . . 5
1.2 Scope . . . 7
1.3 General Use Case . . . 7
1.4 Limitations of this Document . . . 7
2 Constraints and Assumptions 9
2.1 Prerequisites to the memory stack . . . 9
2.2 Realization of FOTA Master instance . . . 10
2.2.1 Further reading . . . 10
3 Terminology 12
3.1 Acronyms and Abbreviations . . . 12
3.1.1 Not listed acronyms in TR_Glossary . . . 12
3.2 System Component Terminology . . . 13
3.2.1 FOTA Target . . . 13
3.2.2 FOTA Master . . . 13
3.2.3 (FOTA-) Image . . . 13
3.2.4 Backend . . . 13
3.2.5 Data Chunk . . . 14
3.3 Process terminology . . . 14
3.3.1 Update . . . 14
3.3.2 Download . . . 14
3.3.3 Installation . . . 15
3.3.4 Verification . . . 15
3.3.5 Activation . . . 15
3.3.6 Rollback . . . 16
4 Requirements 17
5 Detailed Technical Solution 18
5.1 Functional and Architectural Elements . . . 19
5.1.1 FOTA Target ECU . . . 19
5.1.1.1 Functional Description of the FOTA Handler Module . . . 19
5.1.1.2 Diagnostics (Dcm) . . . 20
5.1.1.3 Nv Data . . . 21
5.1.1.4 FOTA Image Types . . . 21
5.1.2 FOTA Master (UCM-Master) . . . 22
5.1.3 Backend . . . 23
5.2 FOTA Procedure . . . 24
5.2.1 FOTA Handler Module . . . 24
5.2.1.1 Internal FOTA Status . . . 24
5.2.1.2 Installation . . . 26

5.2.1.3 Activation (Switching Partitions) . . . 26
5.2.1.4 Rollback . . . 27
5.2.1.5 Cancellation . . . 28
5.2.1.6 Invalidate “old” runtime partition . . . 28
5.2.2 FOTA Master . . . 28
5.2.3 Dcm Interaction . . . 29
5.2.3.1 Interface FOTA Target instance via diagnostic services (Dcm) . . . 29
5.2.3.2 Diagnostic services used by FOTA . . . 29
5.2.3.3 FOTA Diagnostic Session . . . 30
5.2.3.4 Buffer handling in context of FOTA . . . 30
5.2.3.5 Processing the FOTA buffer . . . 31
5.2.3.6 Preemption of the FOTA protocol . . . 31
5.2.3.7 Resume of a preempted FOTA procedure . . . 31
5.2.3.8 Exposed FOTA specific Diagnostic Services . . . 32
5.3 Migration Scenario . . . 34
5.4 Interfacing Additional Features . . . 35
5.4.1 Security Features . . . 35
5.4.2 Safety Features . . . 35
5.5 Error Handling . . . 36
5.6 Sequence Charts . . . 37
5.6.1 (Re-)Initialization of the FOTA procedure . . . 37
5.6.2 Processing of FOTA Data Chunks . . . 38
5.6.3 Resume interrupted/preempted FOTA procedure . . . 40
5.6.4 Activation of successfully flashed FOTA-Image . . . 40
5.6.5 Rollback of FOTA procedure . . . 40
6 Appendix 41

1 Introduction

The FOTA approach shall introduce a generic mechanism to update ECU software during runtime. While the current ECU software is executed and fully available from functional point of view (e.g. during driving) a new ECU software shall be programmed in the background (installation phase). At the time of the installation, which can be interrupted and continued over several driving cycles, the authenticity and integrity of the new SW shall be verified. In case of positive verification results, the ECU shall be able to activate the new SW. The activation of the SW shall always require a special ECU mode (e.g. boot), hence the activation of new SW must not be started or even executed while driving. The activation shall be done in a vehicle safe-state, e.g. stand-still, engine off and applied parking brake. In case of detected anomalies or errors after or during activation of the new SW, the ECU shall be able to realize an ECU internal rollback to previous SW. ECU internal rollback implies the approach, that the previous SW is still present on the ECU and can be re-activated.

Important: The whole realization approach of FOTA features strongly depends on concepts and change requests currently under development or discussion within AUTOSAR. These discussion points are expected to be solved in the next release phase:

- UCM Master (see specification documents [1] and [2], also see chapter 2.2 within this document)
- Concurrent access to flash memory (read-while-write feature, see chapter 2.1)

1.1 Motivation

Due to a growing SW complexity driven by evolving security requirements, distributed and connected functions the need to keep a system in a vehicle up-to-date is continuously increasing. In order to avoid time-consuming and recurring service garage visits because of an upcoming SW update, the SW deployment to fleets shall be orchestrated over-the-air. Different wireless techniques (UMTS, LTE, Bluetooth, WiFi, 5G) can be used to connect the vehicle to a backend/cloud system to provide a capability to download SW to the vehicle. The distribution of the new SW to target ECUs affected by the update is done via vehicle busses as CAN, CAN-FD, Flexray or Automotive Ethernet.

Most ECUs nowadays have an on-board reprogramming capability, which is used in the service garage infrastructure to deploy bugfixes or functional improvements in the field.

This interface, usually a flashbootloader, could also be addressed via an Over-The-Air (OTA) update master ECU in order to install a new SW, when the vehicle is outside of the service garage. In other words, you can realize an over-the-air SW update through the transfer of programming functionalities from an external diagnostic tester into a connected in-car-ECU (OTA-Master). From the perspective of target ECUs, it can be

even more transparent, whether the target ECU is conducted via diagnostic tester or OTA-Master.

This approach brings several aspects that are to be considered:

- During the entire SW update process the target ECU is not operational from functional point of view. This usually leads to a down-time of the entire vehicle during SW update.
- In case of distributed functions, where several ECUs must be updated in one campaign, the vehicle down-time is even longer compared to a single ECU update.
- In case of errors during or after the installation of the new SW a re-installation of the previous SW could be initiated and realized via OTA-Master. However, this additionally increases the vehicle downtime and could finally lead to a situation of functionally bricked ECUs.
- In case a rollback is necessary, all dependent ECUs related to the update have to be rolled back as well.
- The power supply concept and residual capacity of battery shall be also taken with care following this approach.

In order to achieve a higher degree of maturity, reliability and finally comfort other approaches of OTA update shall be analyzed and their impacts on the existing BSW architecture determined.

To reduce the vehicle downtime because of a running OTA update, parts of OTA update process shall be executed during normal operation of ECUs, e.g. while driving.

While the current SW is normally executed, the new SW shall be installed in the background. This shall not have any negative impacts on the performance and availability of the currently running SW. The installation of the new SW shall be interruptible, i.e. the installation process shall not be entirely dropped at the end of a driving cycle but continued in the next driving cycle.

At the end of background installation process, the new SW shall be ready for activation.

Due to different HW capabilities of μCs and the used flash memory devices the activation of the new SW could be realized (depending on HW capabilities) by either just booting from a previously inactive partition of the μC or could initiate the ECU internal activation processes, including ECU internal copying of the new SW from pure storage to the executing partitions. The pure storage partition can be allocated on both, μC internal or μC external flash, e.g. connected via SPI (refer to chapter 5.2.1.3 for details).

During the activation of the new SW according to the HW capabilities of the given product, the previous SW shall be kept within the ECU. In case of detected errors during or after the activation, an ECU-internal rollback to the previous SW shall be initiated. This shall be done either by defined internal criteria, e.g. detection of broken authenticity of the new SW, or external triggering by OTA-Master.

This approach would resolve or essentially improve the drawbacks of the OTA update approaches listed above, where installation and activation are realized in functionally non-operational mode. The vehicle downtime will be significantly reduced, and the new SW can be installed on several ECUs in parallel. As soon as the new SW is installed on all affected ECUs the activation can be centrally triggered. In case of unexpected anomalies while activation, a rollback shall be triggered and will be executed on all ECUs being affected by the last OTA update.

1.2 Scope

The objective of this document is to provide new SW update concepts, which support installing new SW on the ECUs, while current SW is normally executed. This includes the scenario, where a vehicle is driving and the new SW is installed on the affected ECUs without functional degradation of those.

Strategies to avoid interferences and blockings with the runtime system will be evaluated within this document. This means all running applications as well as system services (e.g. diagnostics, NvMemory) shall be fully available during the update process and may impact the delay of any request processing from functional point of view only in a minimal way.

Investigation of different approaches for ECU internal backup and rollback capabilities will also be detailed in this document.

1.3 General Use Case

The new ECU software shall be transmitted using diagnostic communication services (Dcm using UDS) and be transferred via the FOTA Target module to the low level memory driver. The memory stack shall program the new software into an inactive memory partition. A buffering mechanism is to be used to piecewise transfer the SW image to the memory stack. After the installation process has succeeded and the integrity of new SW was verified, an activation mechanism will be triggered to finally initiate the new SW. In case of failures during or after the activation of the new SW a rollback to previous SW, which shall be still available ECU-internally, shall be possible.

A complete list of specified requirements necessary to achieve the FOTA process can be found here: [3]

1.4 Limitations of this Document

Use cases and topics that are not covered within this document:

- Update of ECUs without UDS support
- Verification strategy of the newly flashed SW

- ECU SW version handling and checking (vendor specific)
- ECU SW compatibility/integrity check (vendor specific)
- Any details about the SW architecture of the memory stack

2 Constraints and Assumptions

2.1 Prerequisites to the memory stack

Within this document, the term *memory stack* shall substitute the memory driver (e.g. FlashDriver) and memory management modules (e.g. Fee), regardless if they are realized as internal or external driver, since no decisions on architectural solutions have been made yet within AUTOSAR. As long as these prerequisites are under discussion and not yet part of the specification documents, the following assumptions must be considered and implemented in the memory stack, in order to realize the FOTA approach as described in this document.

- Access to program flash memory

  Up to now AUTOSAR only foresees the access to the data flash sections during runtime. The access to program flash sections was not addressed within AUTOSAR yet and needed to be realized in a proprietary manner (e.g. in bootloader mode). To realize the FOTA approach, it is necessary to get read and write access to the program flash sections during runtime. Therefore an extension of the AUTOSAR memory stack in order to gain write access to the program memory is currently under development and planned to be released in R20-11.

- Read-while-write feature

  Assuming FOTA shall be realized on HW solutions with A/B swap capability (memory abstraction), there is a need to write the new software to an inactive memory section while the active section is accessed for reading during normal execution in parallel. This mainly affects the planned realization of the program flash access, where one partition is executed (active) and accessed for reading, while another non-runtime related partition (inactive) needs to have write access in order to apply new software in the background. In order to realize the read-while-write approach, the HW vendor needs to provide the needed functionality in their MCAL implementation.

- Use of different memory solutions

  The FOTA procedure is meant to work with all different kinds memory architectures and techniques. However, the hardware and driver vendors are responsible to provide the needed interfaces, defined by AUTOSAR MCAL layer, in order to instrument the FOTA features in the here described manner. Differences between existing and future hardware solutions and low-level drivers, are expected to be invisible within all provided interfaces to the memory stack APIs. In other words, all existing memory architectures shall be supported without affecting any upper layers in any way.

- Handling of Nv (user) data

  While user data (Nv-Data, see section 5.1.1.3) updates are handled by the NvM module, using its provided features and interfaces, an appropriate mechanism to

migrate e.g. data models while they are still accessed by the active partition must be defined.

From architectural point of view this will result in an overview as shown in figure 2.1:

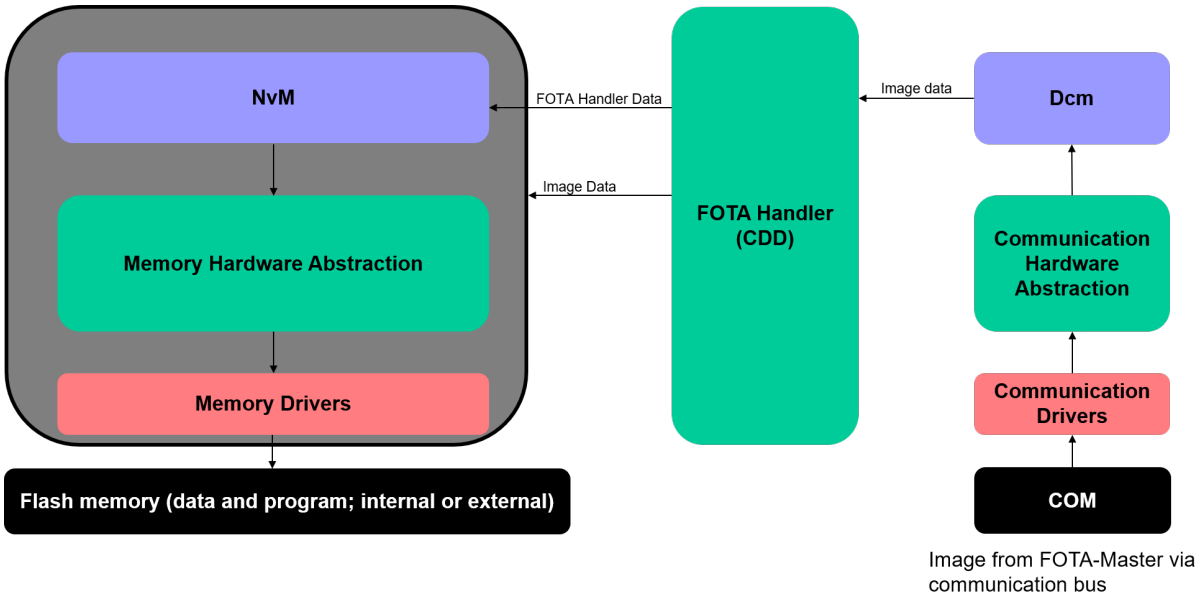


Figure 2.1: Assumed architectural layout for the memory stack containing data and program flash drivers

2.2 Realization of FOTA Master instance

Since the FOTA approach shall be applicable to both the classic and the adaptive platform of AUTOSAR, it was decided to place the FOTA-Master realization into the adaptive platform, since adaptive ECUs are expected to be way more powerful than classic ones. This FOTA-Master realization is done by the WG-UCM team in the adaptive platform context and will be called UCM-Master. The UCM-Master shall address, handle and steer updates also to the classic platform and is expected to have the first release in R19-11 together with this document.

However, since this document focuses on FOTA features, it will always refer to the master instance as FOTA-Master where the distinct implementation is realized in the adaptive platform as the UCM-Master.

2.2.1 Further reading

- For more information about the UCM-Master implementation see specification document [1].
- For more information about the role of the here referred FOTA-Master see chapters 3.2.2 and 5.1.2 for details about its expected behavior.

- For more information about potential non-volatile data handling and migration see specification document [4].

3 Terminology

In this chapter all prerequisites, assumptions, process and system terminologies used within the FOTA process are listed.

3.1 Acronyms and Abbreviations

Commonly known acronyms defined by AUTOSAR and listed in AUTOSAR Glossary [5].

Abbreviation/Acronym	Description
BSW	Basic Software (See LayeredSWArchitecture slide collection in AUTOSAR releases)
ECU	Electronic control unit

3.1.1 Not listed acronyms in TR_Glossary

Here is the copy paste ready list of acronyms which are not yet part of the TR_Glossary [5] document:

Abbreviation/Acronym	Description
DCM	Diagnostic Communication Manager (AUTOSAR BSW Module, Classic Platform) (Surprisingly not yet listed!)
FOTA	FirmwareOverTheAir
HSM	Hardware Security Module (cryptographic features realized in HW) (Should actually come from a different doc, e.g. safety or security)
NvM	Non-Volatile Memory, BSW Module, sometimes referred to as NV Memory (Surprisingly not yet listed!)
OTA	Over-The-Air technologies in general, regardless of AUTOSAR
UCM	Update And Configuration Management (Functional Cluster in the Adaptive Platform, AUTOSAR Workgroup)
UDS	Unified diagnostic services (see ISO-14229) (Surprisingly not yet listed!)

3.2 System Component Terminology

3.2.1 FOTA Target

This term applies to all ECUs which are capable to be updated via the FOTA procedure in general.

3.2.2 FOTA Master

The FOTA Master specifies the instance which handles all software updates with all their related information needed to apply the respective update to the FOTA Target ECU addressed by the FOTA update process.

Note: Keep in mind, that the FOTA Master is realized in context of the UCM Master concept, which is currently under development and might change in the future. A collaboration between UCM and the FOTA group is considered and planned for future releases.

See chapter 5.1.2 for more details.

3.2.3 (FOTA-) Image

The term Image describes a full downloadable ECU software collection. This shall be regardless if partial, incremental or full featured ECU software is downloaded and installed (see figure 3.1). Applying a new Image to an ECU will update at least one of the following ECU software parts:

- Application software
- Basic software
- Calibration data

After applying a new Image to an ECU, it must be in a fully functional state and be compatible to the rest of the system.

3.2.4 Backend

This identifies the remote instance which provides the FOTA Image to be downloaded into the FOTA Target ECU (e.g. via Wifi, 3G, 4G, 5G, etc.).

3.2.5 Data Chunk

Data Chunks are data packets, transferred from the FOTA Master ECU to the FOTA Target ECU, containing the raw image data (UDS TransferData service payload). This term is used since the length of each data chunk may vary.

3.3 Process terminology

The overall FOTA update process can be divided into several phases, which are shown in this chapter. A general overview is shown in figure 3.1:

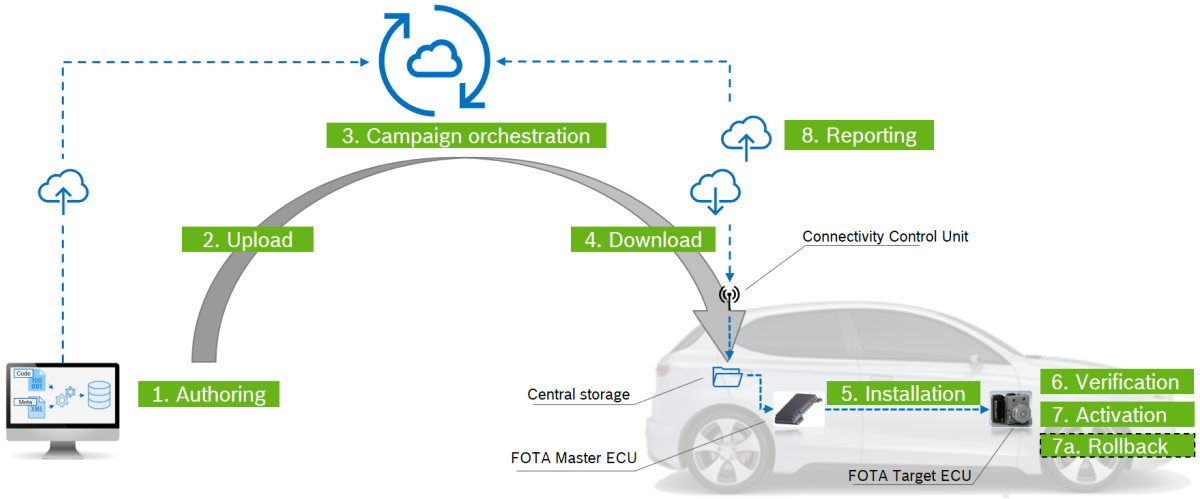


Figure 3.1: FOTA Phases

3.3.1 Update

The term “(FOTA) update” is used to indicate a whole update procedure of an ECU (or multiple ECUs in case of other dependent ECU updates). It contains all below described tasks such as download, installation, verification, activation and so on. The roll-back functionality can be considered as part of the update procedure in case it is necessary.

3.3.2 Download

Downloading the image refers to the transfer of all related and necessary ECU software, data and configuration needed for a complete update of a FOTA Target ECU from the backend server to the FOTA Master instance.

3.3.3 Installation

The term “installation” refers to the transfer of the new ECU software from the FOTA Master ECU to the FOTA Target ECU. Since it is not handy to completely transfer the whole ECU software at once to the FOTA Target ECU, this process is realized using data chunks (see chapter 3.2.5 for details). This means that the installation process can be active even over several driving cycles. The installation process is finished, when there is no more chunks left to transfer to the FOTA Target ECU and all of them have successfully been written to the memory stack.

In addition, the installation process also covers the actual writing of software into the inactive target partition by the program flash driver within the FOTA Target ECU.

3.3.4 Verification

The verification process is implementation specific and shall assure the correctness of the newly flashed ECU software. This only affects the plain ECU software (e.g. an image or even a differential update) which was flashed into the respective *FOTA Target* ECU.

Note: The initial verification, authentication and conditions checks related to integrity and authenticity of the FOTA Image as software update package is done by the FOTA Master. Verification checks in context of the FOTA procedure from FOTA Target point of view shall focus on the semantic and functional integrity and are therefore implementation specific. Integrity and authenticity verification can be done again on the FOTA Target if needed.

Note: In case of using external flash memory, which needs to be copied to the internal memory during activation, a second verification step after copying shall be considered. This shall ensure that no unintended modifications of the e.g. SPI transfer from external to internal memory has happened.

3.3.5 Activation

The activation process describes the actual switch of the ECU boot partition. In case of non-switchable memory architecture ECUs (e.g. external flash memory, fixed multi partition memory), this also covers the copy process from the temporary (e.g. external) flash memory to the internal one. The activation procedure is completed, when also the backup of the previous ECU software (n-1) is confirmed.

Note: The final activation of the newly installed software must be done in a vehicle-safe state. It is up to the integrator to ensure this vehicle-safe state.

3.3.6 Rollback

During the rollback process all ECU and user data from the previous running software must be restored. After the rollback has finished there must be no difference to the ECU software and user data compared to before the whole update procedure has been started.

4 Requirements

All requirements defined for FOTA so far can be found in the AUTOSAR RS document [3].

5 Detailed Technical Solution

This chapter provides all needed information in order to setup FOTA Target features for the Firmware-Over-The-Air procedure. Affected architectural components, outside of FOTA Target ECUs, are also listed here for system completeness.

Figure 5.1 shows an architectural overview of all involved components and modules within the overall FOTA process.

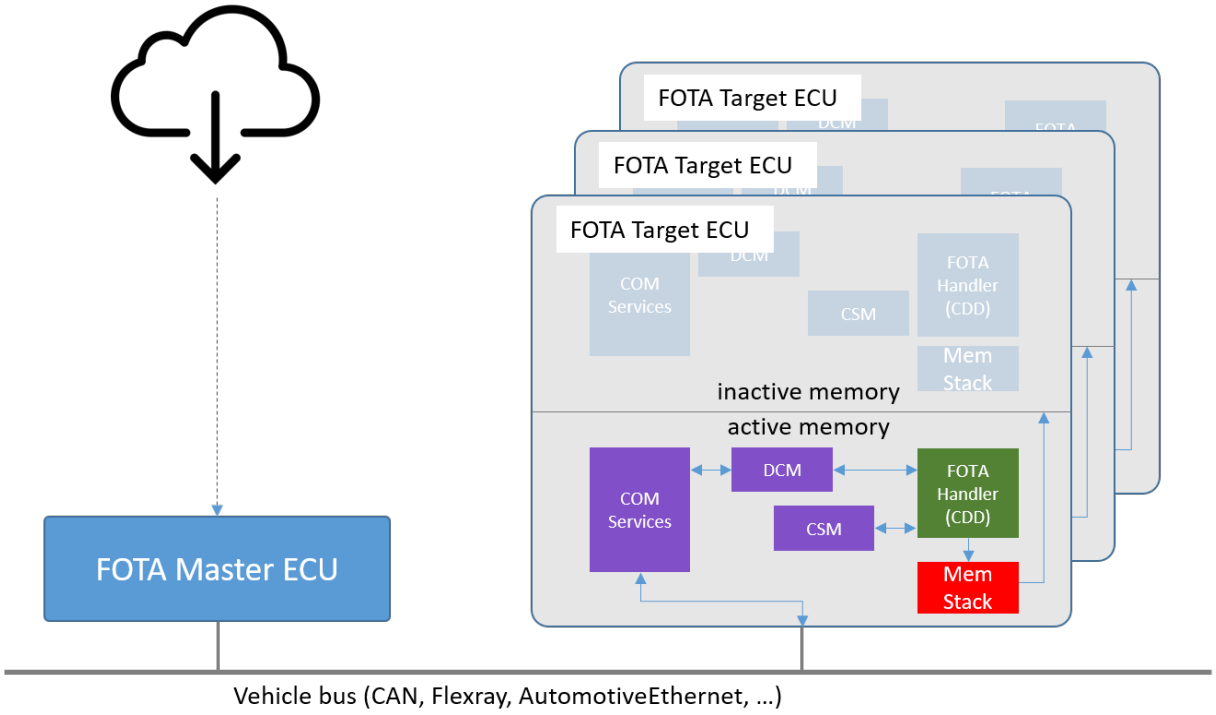


Figure 5.1: Overview of relevant FOTA components (program memory related)

5.1 Functional and Architectural Elements

The overall FOTA architectural design (as shown in figure 5.1) consists of:

- The FOTA Target ECU

  which receives the ECU software to be flashed from the FOTA Master and forwards the software to the low level memory stack instance (actual ECU software flashing process).

- The FOTA Master ECU

  which caches all new ECU software artifacts to be flashed to a FOTA Target ECU.

- A backend server

  which provides the FOTA master ECU with the image to be installed to the FOTA Target ECU.

5.1.1 FOTA Target ECU

Since the FOTA Handler specification is currently evolving and not all features are specified and defined yet, the FOTA Handler module is seen as a CDD for now. This decision was made due to the fact, that up to now, not enough features were defined to release a complete new AUTOSAR Basic Software Module. If, in future, the feature set grows to a BSW module worth complexity, a definition of a FOTA BSW module will be considered. However, since the architectural decision on how the “read-while-write” flash approaches and the program flash access, that will be realized by the related AUTOSAR MCAL group, has not been decided, the CDD approach leaves the option to connect through all BSW layers.

5.1.1.1 Functional Description of the FOTA Handler Module

The FOTA Handler module is meant to take care about the FOTA chunk buffer (located in the DCM module as part of the protocol row configuration) handling and to finally trigger the memory stack when new chunk data has arrived, and the memory driver is available for writing.

Within the FOTA Handler module, the respective callout function to serve the received requests by the Dcm module, which provides the FOTA data chunks, needs to be implemented. Inside this callout function the reception of the FOTA chunk needs to be handled.

The solution approach described in this document implements two functions within the FOTA Handler module:

- *FOTA_ProcessTransferDataWrite(...)* (Dcm callout)

This function is called by the Dcm module each time an image chunk is received by the Dcm module. The main task of this function is to inform about the reception of a new data chunk, to be processed by the *FOTAHandlerMain(...)* function. Additionally, the *OpStatus* call parameter shall be set to `DCM_PENDING` until the data chunk processing is finished. During the processing of the data chunk, the callout function shall always return with `Dcm_ReturnWriteMemoryType` = `DCM_WRITE_PENDING`. This informs the Dcm about the current processing of the service which in addition gets forwarded to the FOTA Master in order to prevent a timeout. Once the indication, that the chunk processing is finished by the *FOTAHandlerMain(...)* function is received, the *FOTA_ProcessTransferDataWrite(...)* shall return with `Dcm_ReturnWriteMemoryType` = `DCM_WRITE_OK` to inform the Dcm and furthermore the FOTA Master of the successful processing. New data chunks can be sent by the FOTA Master ECU afterwards.

- *FOTAHandlerMain(...)*

  This function does the actual processing of the new data chunk indicated by the *FOTA_ProcessTransferDataWrite(...)* function and forwards it to the memory stack for programming. The *FOTAHandlerMain(...)* function shall be scheduled cyclically by the AUTOSAR OS and is independent of the diagnostics and communication stack. Once the processing of the received data chunk has finished, the *FOTAHandlerMain(...)* function shall indicate this to the *FOTA_ProcessTransferDataWrite(...)* via e.g. using an *InterRunnableVariable*.

5.1.1.2 Diagnostics (Dcm)

As the AUTOSAR Dcm module implements the UDS (see [6]) diagnostics protocol, lots of useful features are delivered “free of charge” for realizing the FOTA functionalities. For the FOTA procedure, the following ones can be considered:

- Session Handling
- Security Access
- Authentication
- Service Handling (user jobs), e.g.:
  - 0x22/0x2E Read/WriteDataByIdentifier
  - 0x31 Routine Control
  - 0x34 RD/0x36 TD/0x37 TE (request download/transfer data/transfer exit)
- Error Handling
- Check Programming Conditions
- Reset/Restart ECU

Goal is to use as much features already provided by the DCM module as possible. Any additional extensions due to a lack of features required for the FOTA procedure shall be realized without extending any core features of the DCM module.

Additional needed extensions to achieve the FOTA requirements shall be implemented by either the FOTA Handler module instance or the FOTA Master module instance (UCM-Master, see chapter 5.1.2 for details).

5.1.1.3 Nv Data

The NvM module in AUTOSAR typically takes care about the data flash (user data, calibration data, errors and snapshots, additional runtime data for the ECU SW) and aligns them into so called NvDataBlocks. The NvM module provides features and interfaces to read, write or delete these data blocks. In order to migrate or modify NvData the features provided by the NvM module shall be used. Neither the FOTA Master nor a FOTA Target may interact with the data flash directly to avoid interferences, blockings or corruption of data. This means that e.g. data model changes that affect the NvData or user data migrations must be handled by the implementer. Refer to chapter 5.3 for details.

However, to safely store FOTA process related information (e.g. current FOTA process state, last successfully written memory address by the FOTA handler, etc.) in order to persist during interruption, the NvM module shall be used to handle (FOTA specific) user data. The mechanisms defined in the respective specification documents shall be used (see [7] for details).

5.1.1.4 FOTA Image Types

In general there are several ways to update an ECU, e.g. installing a full system image or just parts of the SW. Even partial and incremental updates can be possible. However, since this document does not specify in detail how to install new software onto a FOTA Target ECU, all types of updates are possible as long as the used installation mechanism allows them.

Based on the capability of the used HW several options to place a new image are possible:

- Runtime execution from fixed addresses

  In this configuration no swapping of partitions is possible due to HW limitations. In that case the image data needs to be placed in a temporary storage location (regardless if internal or external memory) and gets copied to the executable runtime partition during the activation phase. Backing up the previous SW version is required for rollback scenarios.

- Runtime execution from flexible addresses

Swapping partitions is supported by the HW, therefore the new SW can be placed in any available partition and will be executed after activation from that location. Adaptions or conversions of addresses and offsets within the image must be ensured by the implementer.

The following table lists several attributes for both partition types:

Task/Item	Fixed Runtime Address Mapping	Flexible Runtime Address Mapping
Installation	stored in temporary memory (external or internal)	directly into memory
Activation	backup old SW, copy new SW, reboot	switch (logical) partitions
Activation duration	Slow (copying of image to target partition needed)	Fast (immediate partition switch at boottime)
Image type	relocatable and nonrelocatable images	relocatable images only (non-relocatable only if supported by HW)
Addressing method	fixed	logical (abstracted memory layout)
Minimum number of partitions	3	2
Typical use cases	• Non-relocatable images • The microcontroller is extended with external memory because internal memory can only contain one single image	ECUs with physical flash memory size is larger than 2 times the image

5.1.2 FOTA Master (UCM-Master)

The FOTA Target ECUs need to communicate with a corresponding Master instance in order to receive FOTA Image data. This Master instance must store the image data for all relevant ECUs within the vehicle network, which can be quite a lot of data.

Usually embedded ECUs, which are targeted by the Classic Platform, do only have limited resources in means of memory, storage and computing power. Therefore it makes sense to move the responsibility of the Master instance to ECUs, which offer way more power, the adaptive platform.

The realization of the Master instance communicating with the FOTA Target ECU is currently done by the Work Group *Update And Configuration Management* (WG-UCM). This would result in following logical architecture:

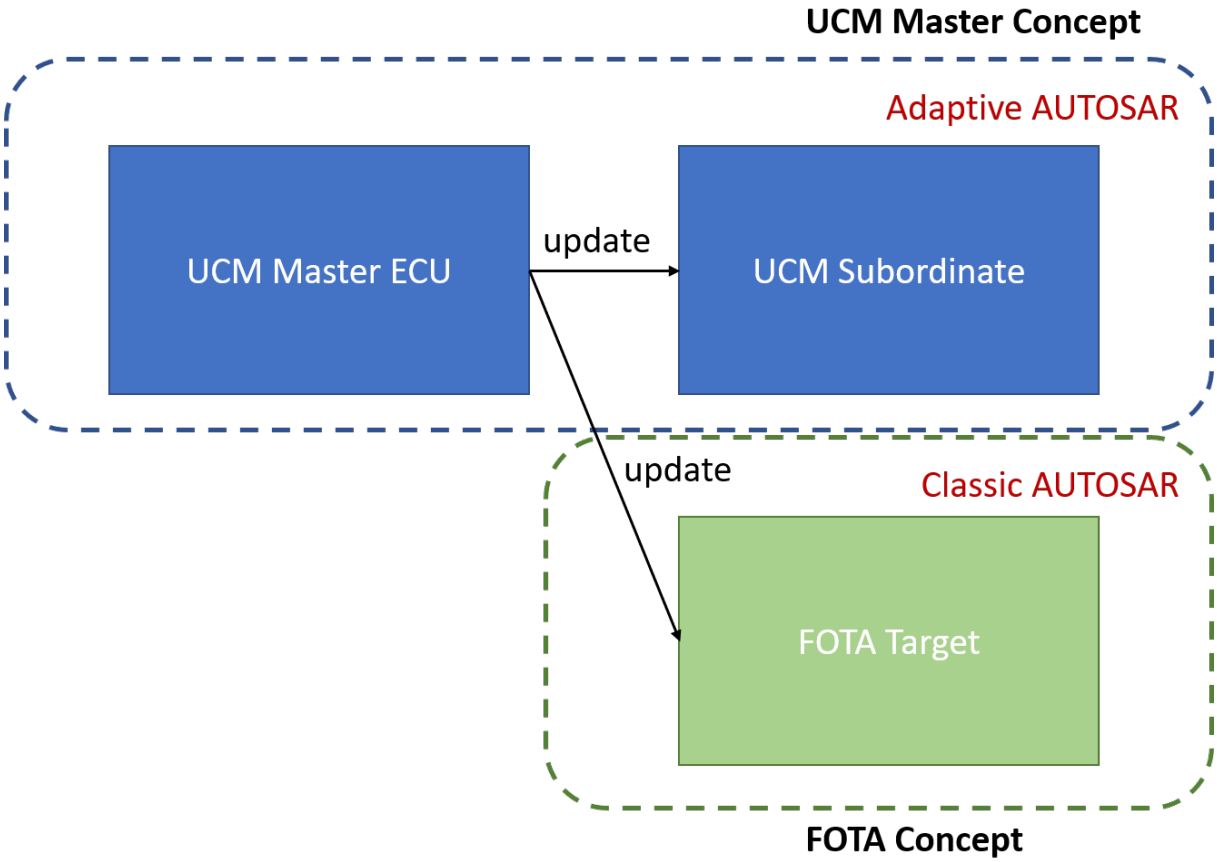


Figure 5.2: Correlation between UCM-Master and FOTA concepts (logical view)

Note: Keep in mind, that the UCM-Master concept is currently under development and might change. A potential collaboration between WG-UCM and the FOTA concept group is considered for future releases.

5.1.3 Backend

The Backend instance represents the initial image provider and notification interface to the vehicle to be updated. A notification about available new software could either be triggered from the backend instance to the car or the vehicle decides on its own when to ask if new software is available (polling approach). However, if a new software for one or more ECUs within the vehicle network are available, the fetching process is started and stores the images in an appropriate location within the vehicle (e.g. FOTA Master). Information about dependencies to other software packages must also be provided by the backend. Status and update progress information about the FOTA processes could be shared with the backend for analytic and diagnostic purposes.

5.2 FOTA Procedure

Note: Keep in mind, that all instructions in this chapter are mainly best practices and shall only act as guide line to realize the FOTA procedure.

5.2.1 FOTA Handler Module

During a FOTA update process the FOTA Handler module needs to take care about certain actions in order to create a solid update process:

- Receive data chunks from FOTA Master
- Handle data chunks
- Forward data chunks to the memory stack
- Exchange status information with the FOTA Master

This chapter explains how those task shall be realized and gives hints and best practices to do so.

5.2.1.1 Internal FOTA Status

The FOTA Handler module needs to handle and to indicate all different states during the processing of FOTA Images. This also includes the recovery of interrupted (due to e.g. driving cycle change) or suspended (by e.g. higher priority diagnostic requests). To provide this information to the FOTA Master, which initiates and triggers the FOTA procedure, a diagnostic service shall be realized to provide and update the current state of installation procedure from FOTA Target point of view. These different states shall be reflected by the following enumeration:

- IDLE

  Initial state of the FOTA Handler after the ECU startup procedure

- INIT

  The FOTA Handler is initialized and Dcm is set into the correct state(in Dcm FOTA session and security access has been granted).

- READY

  All FOTA data chunks have been installed, activation procedure can be triggered.

- PROCESSING

  The FOTA Handler is triggered by the Dcm callout since a new chunk has been received and is processed in the callout context.

- WAIT

The FOTA Handler has successfully processed the last received data chunk, returned the Dcm callout function and is waiting for the next data chunk.

- VERIFY

  Optional and implementer specific step, since the FOTA Target does not specify any details on the verification process.

- ACTIVATE

  FOTA installation has finished and received a respective service job from the FOTA Master that indicates the partition switch during the next boot process.

- ERROR

  Optional and implementer specific. Reserved state for e.g. implementer specific error handling, which is not (yet) covered by the FOTA Target.

All the above listed states shall help to keep the FOTA Target state and therefore the whole FOTA update procedure deterministic, solid and recoverable. This will result in a state machine for processing the FOTA update as shown in figure 5.3 below:

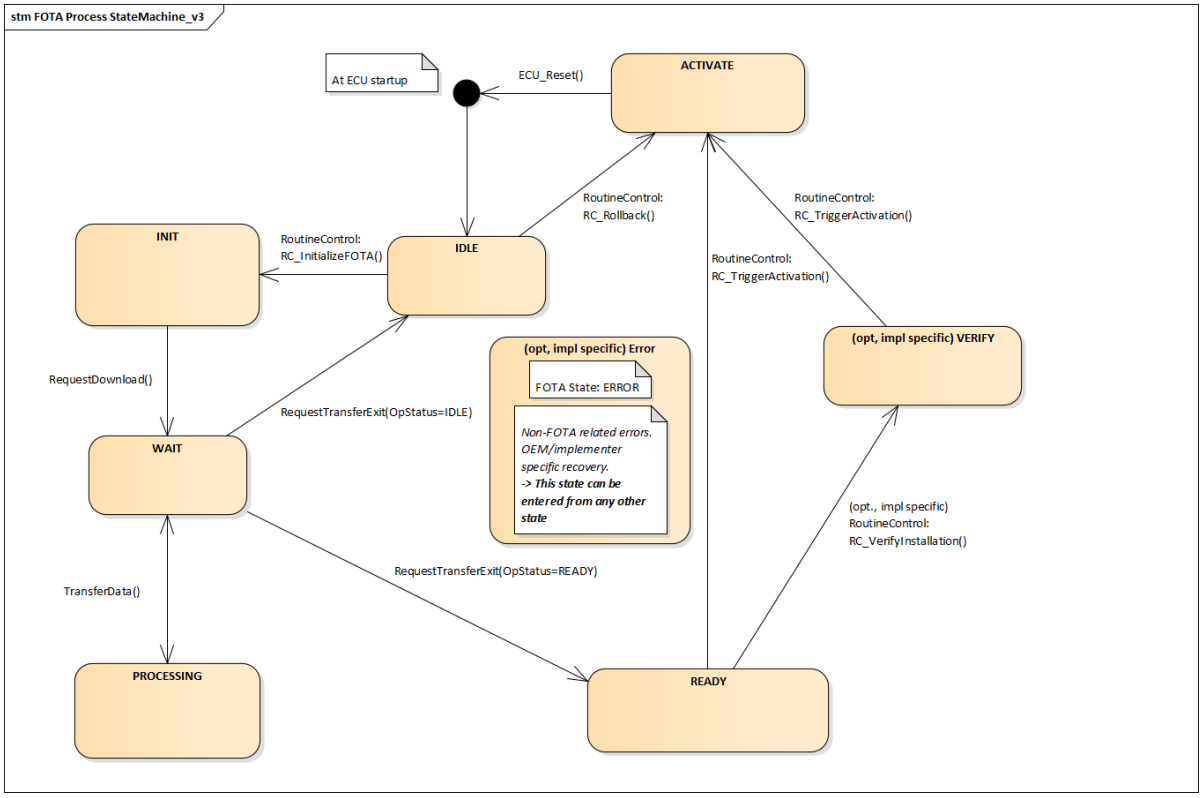


Figure 5.3: State machine of the FOTA Target

Note: The transition to IDLE or READY can be distigunished by the OpCode parameter in the RequestTransferExit() call, see SWS_DiagnosticCommunicationManager[8] for details.

Note: FOTA Master triggered rollback would be the transition from any state into ACTIVATE state.

The internal FOTA status enumeration shall be available to the FOTA Master ECU as either Read/WriteDataByIdentifier or RoutineControl UDS service provided by the FOTA Target ECUs Dcm module. The state transitions of the FOTA Handler module are controlled by the FOTA Master instance in normal (uninterrupted) processing. In case of interruption the internal FOTA state indicates the recovery strategy. All additional information in order to resume any interrupted FOTA procedure shall be provided as user jobs by the FOTA Target ECUs Dcm module (see section 5.2.3.8 for more information).

5.2.1.2 Installation

The installation process starts in the *WAIT* state and consists of waiting for the TransferData service request from the FOTA Master. Once this is received, the FOTA handler switches its state to *PROCESSING*. In this state, the FOTA chunk is received in the DCM RAM buffer (see chapter 5.2.3.4, also refer to specification document [8]). Once all the data of the chunk is received, the Dcm module will execute the FOTA callout function to process the chunk (see 5.2.3.5). The callout function takes the address of the chunk in Dcm RAM buffer as an input parameter and programs this data into the flash memory using the memory stack interface. The destination address (offset in partition) and size of the chunk are also input parameters to this function. Optionally (implementation specific), the callout function can verify the data after it is programmed. Once the programming is done, the callout function returns the final response (OK/NOK) to the DCM, so it can be transmitted to the FOTA Master. After this, the FOTA handlers switches back to *WAIT* state. The callout function must be implement asynchronously, i.e., within a FOTA main function, called cyclically by the OS.

5.2.1.3 Activation (Switching Partitions)

In order to enable a newly flashed ECU software, a switching mechanism is needed. This shall apply for all available flash memory solutions (e.g. A/B swappable or temporary storage). The switching mechanism needs to take care of all activities required to successfully switch the boot/runtime partition (e.g. moving new ECU software from external to the internal flash). Status information about the currently active partition and update status of the inactive partition must be available in all run modes (e.g. bootloader, application).

Note: This switching mechanism is also a prerequisite to the HW from system point of view.

Since it is not transparent to the FOTA Target ECU if the current update is dependent to other ECU updates and their status, the FOTA Master needs to take care of dependencies and the point in time when an activation of new SW is performed. A respective Dcm service job to trigger the FOTA state machine transition from *READY* to *ACTI-*

VATE e.g. using a RoutineControl (`RC_TriggerActivation`) service (refer to figure 5.3) needs to be configured. The implementation of this RoutineControl service is part of the FOTA Handler module and must take care of all required steps to prepare the actual activatioin step during the next ECU restart.

All needed steps to perform the activation (e.g. verification, informing other ECUs or FOTA-Master) are project specific and are therefore not closer specified here.

Note: Keep in mind, that the flag/data field to control the activation indication must be located in an memory area accessible by both the bootloader AND the application.

5.2.1.4 Rollback

The main use case of the rollback scenario is the master-triggered rollback. This means, the FOTA Master detects an error on one or more newly installed SW images and actively triggers a rollback, for example on request from backend or when version incompatibility is detected. This must be indicated by the FOTA Master in an appropriate way using again UDS services provided by the Dcm module (e.g. ReadDataByIdentifier/WriteDataByIdentifier, RoutineControl), see section 5.2.3.1 for more details.

Alternatively, the respective FOTA Target ECU, which is not capable to boot successfully, will initiate the rollback procedure (e.g. switch back partitions) autonomously in order to achieve a running and communicative state again. After the autonomous rollback and the FOTA Target ECU is responsive again, the respective SW version can be requested by the FOTA Master ECU.

To read the currently running SW version of a certain FOTA Target ECU back to the FOTA Master ECU, according to requirement [RS_FOTA_00035], an appropriate UDS service shall be in place (e.g. ReadDataByIdentifier/WriteDataByIdentifier, RoutineControl), see section 5.2.3.1 for more details.

Detailed realization aspects strongly depend on system design and architecture as well as specific OEM and supplier flavors. One possible realization approach is explained in section 5.2.1.4.1

5.2.1.4.1 Realization Approach as Example

In order to indicate a running FOTA Target ECU after boot to the FOTA Master ECU, the parameter *DcmResponseToEcuReset* shall be set to AFTER_RESET in the respective Dcm configuration element *DcmDspEcuReset* for every FOTA Target ECU. This will send a final positive response to the FOTA Master ECU to announce the availability of each FOTA Target ECU after reset individually. The FOTA Master ECU may then ask each FOTA Target ECU in the system about their currently running SW version. Dependencies between different FOTA Target ECUs and if a potential rollback shall be applied, must be evaluated and triggered by the FOTA Master ECU.

Note: Consider setting configuration parameter DcmSendRespPendOnRestart to TRUE in order to inform the FOTA Master ECU about the restart for activation.

5.2.1.5 Cancellation

Cancellation describes the abortion of an ongoing FOTA software update process. The cancellation process can be triggered by the FOTA Master ECU at any time between the initiation of the FOTA update and the activation trigger after finalizing the installation procedure. In detail cancellation simply stops the transmission of FOTA chunks from the FOTA Master ECU to the FOTA Target ECU by executing the *RequestTransferExit* service on FOTA Master ECU side. Additionally the target partition, to where the just canceled software should be applied to, shall be invalidated in order to indicate to the FOTA Target ECU that no interrupted update needs to be resumed after e.g. an ECU reset/restart due to a new driving cycle. Also refer to chapter 5.2.1.6 about the invalidation procedure.

5.2.1.6 Invalidate “old” runtime partition

After the switch to the new software was successful and the whole boot process and runtime availability was confirmed, the previous (now inactive) partition may be invalidated e.g. in order to avoid unintended or forced unauthorized rollbacks. This ensure that the previously active software and its potential vulnerabilities cannot be activated anymore. The decision when to invalidate the "‘old"’ SW or not is up to the implementer.

Note: The trigger to invalidate the previous partition is expected to be sent by the FOTA Master, since a rollback might still be necessary, if another ECU fails the update in a multi-ECU update campaign.

Note: How to achieve this feature is not yet defined within the FOTA context and is therefore implementer specific (e.g. via setting a flag, delete “old” partition, invalidate partition checksum, aso.).

5.2.2 FOTA Master

This FOTA Master instance connects to a specifically defined user job which was configured within the Dcm module (see chapter 5.2.3 for details). The transfer of the data between the Dcm module and the FOTA Handler module shall be done using FOTA data chunks in order to give the ability to have the FOTA download and flash procedure interruptible (preemptable). This is necessary as all functionality of the FOTA Target ECU shall be available during the flash process (e.g. Application-SWCs, BSW-Services).

In general, an Over The Air procedure might be interrupted at any time due to service issues during the reception of the FOTA image from a remote endpoint. Due to this

situation, it cannot be assured that all data packages are received in a consecutive order or even in one connection and/or driving cycle.

5.2.3 Dcm Interaction

The Dcm module features lots of functionality which can be used by the FOTA procedure e.g. identification, authentication, security mechanisms and even flow control and consistency functions. As these features are provided out of the box, there is no need to reimplement any of these.

5.2.3.1 Interface FOTA Target instance via diagnostic services (Dcm)

A list of required steps to be fulfilled by the Dcm module during the FOTA update procedure is listed below:

- Readback FOTA status information
- Preinstall SW revision check (by FOTA Master)
- Switch into FOTA session (user session)
- Verify access control for FOTA session
- Installation phase (data transfer from Master to Client and flashing)
- Verification phase (implementer specific, no details here)
- Pre-activation conditions (e.g. migration of user data, see chapter 5.3 for details)
- Activation phase (in vehicle safe state; ECU resets)
- Post-activation SW revision check (by FOTA Master)

5.2.3.2 Diagnostic services used by FOTA

In particular, the above mentioned steps needed to fulfill the FOTA procedure will be covered by the following Dcm services:

- Transfer FOTA-Image data

  To transport the data received from the FOTA Master to the *FOTA Target*. Proposed services:

  – Request Download/Transfer Data/Transfer Exit (0x34/0x36/0x37)

- Exchange Status Information

  To read out the current update status or FOTA Target ECU conditions and also set them if needed by the Master. Proposed services:

  - Routine Control (0x31)
  - Read/Write Data by Identifier (0x22/0x2E)

- Additional control functions (e.g. for activation, (in-)validation or rollback)

  In order to set specific status, flags or to execute additionally needed functions (e.g. set activation flag or invalidate the “old” partition):

  - Routine Control (0x31)
  - Read/Write Data by Identifier (0x22/0x2E)

5.2.3.3 FOTA Diagnostic Session

Since the FOTA procedure may not impact any runtime execution and availability of services on BSW and application level, it shall define its own diagnostic session in which context the FOTA processing shall take place. This allows the Dcm to use the *diagnostic protocol preemption* feature which is part of the Dcm specification (see specification document [8] Chapter 7.2.4.14.3 *“Preemption of protocol”* and also chapter 5.2.3.6 within this document).

The change into the FOTA diagnostic session should in addition be secured using the secure access mechanisms specified by the Dcm module (see [8]).

5.2.3.4 Buffer handling in context of FOTA

In general FOTA data chunks are transmitted by the FOTA-Master ECU using UDS (0x36 TransferData). From FOTA Target point of view, the DCM module will receive the data chunks from the communication stack. According to the Dcm module specification the internal buffer can be configured per diagnostic protocol (e.g UDS, OBD, ...).

Since the FOTA process shall run in its own low priority context within the Dcm, an own protocol entry for the use of FOTA shall be defined using the `DcmDslProtocolRow` configuration container. This means that a second UDS protocol only processing FOTA requests (with lowest priority defined with `DcmDslProtocolPriority`) needs to be introduced which shall reference its own data buffer within the Dcm module.

The maximum amount of data to be received from the FOTA-Master ECU depends on the DCM configuration parameter `DcmDslBufferSize` (see DCM module specification for details [8]) specific for the FOTA related diagnostic protocol handling. Since the DCM service *TransferData*(0x36) is realized as callout function, the FOTA Handler module must implement such a function, where the received pointer to the DCM data buffer, containing the FOTA data chunk, is processed. This function callout is triggered cyclically from the Dcm main function until a final response (OK/NOK) has been provided by FOTA Handler. During the processing of the data chunk the FOTA Handler shall return the state "‘pending"’.

Note: Determining the appropriate size of the FOTA data chunk buffer is highly hardware and implementation specific. However, a best practice is not to exceed the maximum size of other UDS protocol configuration in the Dcm module.

5.2.3.5 Processing the FOTA buffer

The TransferData callout only acts as indicator for an additionally provided FOTA main function which is called asynchronously by the OS. Within this FOTA main function, the actual processing of the FOTA chunk to the memory stack is done. The TransferData service callout shall also wait until the FOTA chunk processing of the FOTA main function has finished before returning and triggering the positive response to the FOTA Master by the Dcm module. This will be indicated by the service return value (Dcm_OpStatus) and controlled by the FOTA Handler module.

In case a protocol preemption arises, it is up to the implementer to decide if the already received FOTA chunk shall be processed by the FOTA main function in the background. However, in any case the last successfully written data to the memory stack shall be reflected in a specific Dcm service job (e.g. using a DID).

5.2.3.6 Preemption of the FOTA protocol

In case of a preemption due to a higher protocol request received by the Dcm module, the current FOTA processing will be stopped immediately and the new request is processed first. A callback function in case of preemption can be configured in the Dcm for each protocol using the configuration container `DcmDslCallbackDCMRequestService`. This will create a call to the here defined function once a protocol preemption has been received. Within the implementation of this callout additional adjustments shall be done, to ensure the consistent and recoverable state of the FOTA procedure.

Once a FOTA session is interrupted due to preemption, there is no information transferred to the FOTA Master ECU. User defined notification jobs within the FOTA Target ECUs Dcm module to inform the FOTA Master about any process interruption/preemption are not in scope of this document and are therefore fully implementer specific.

However, in this document it is expected to have no notification at all of the FOTA Master ECU in case of preemption. This means the only way to detect such a preemption by the FOTA Master ECU is a timeout. This timeout is in general dependent on the needs of the ECU. After the timeout has occurred, the FOTA Master ECU needs to reestablish and reinitiate the FOTA process.

5.2.3.7 Resume of a preempted FOTA procedure

Assuming the protocol preemption on the FOTA Target ECU side has finished and the FOTA Master ECU run into a timeout of processing the last provided FOTA chunk. From FOTA Master ECU point of view only a timeout occurs on UDS protocol level, but

no further information about the data chunk processing state is provided by the FOTA Target ECU due to protocol restrictions. Therefore it cannot be determined, if the last chunk was completely received by the FOTA Target ECU and could still be processed in the background or if the whole chunk needs to be resent after reinitialization.

The FOTA Master ECU may use FOTA specific Dcm service jobs to exchange needed information with the FOTA Target ECU in order to resume the preempted FOTA procedure e.g.:

- current processing state (see chapter 5.2.1.1 for details)
- programming state of the last data chunk
- last successfully programmed address
- last chunk id

The strategy of resume, either always resume after last successfully transferred data chunk (FOTA Master exclusive) or exchange last succeeded write position with the FOTA Target ECU, is up to the implementer. Once this and all other necessary information has been exchanged, the FOTA process shall proceed at the indicated position.

While the FOTA process initiation is triggered by the FOTA Master ECU, the general flow of the image data (installation step, see also chapter 3.3.3) is controlled by the FOTA Target ECU. Since the FOTA Master ECU may only provide the FOTA image data via UDS, the return code sent by the Dcm from the FOTA Target ECU indicates the availability to receive data chunks. If the communication terminates (e.g. due to protocol preemption) the timeout on FOTA Master side indicates an interruption. After FOTA Master and Target ECU got in sync again, based on the state information (e.g. DIDs or results of RIDs) provided by the FOTA Target ECU, the FOTA procedure can continue.

5.2.3.8 Exposed FOTA specific Diagnostic Services

Proposed approaches for FOTA related information exchange and feature triggering:

- Routines
  - Verification (implementation specific, no details in this document)
  - Activation
  - Cancel
  - Rollback (In case of dependent update and system wide rollback)
  - Installation precondition checks
  - Post installation checks
  - Post activation checks

- Data Identifier (DIDs)
  - Last successfully written memory address of the FOTA image
  - Get installed SW version
  - Additional implementation specific information exchange

5.3 Migration Scenario

This chapter covers aspects regarding the migration scenario, which deals with the porting of e.g. user or application specific data used and potentially modified during runtime. More details about the process and affected components is described in chapter 2.1 and 5.1.1.3. Within AUTOSAR the NvM BSW module takes care about all modifiable user and application data (see specification document [7]).

The general precondition is NOT to extend any features or functionalities already provided by the NvM module. All functionalities needed in addition to the NvM module (or any other BSW module) shall be realized by either the FOTA Handler module instance or the FOTA Master ECU (see chapter 5.1.2 for details).

The main task to be covered with the migration scenario is the handling of changed data models shipped with the FOTA-Image (see chapter 3.2.3). This means a mechanism to migrate the “old” data model into a “new” one, before switching the partitions to the updated one by the FOTA update process, must be defined.

Since the NvM module provides features for creating, deleting and moving of NV data blocks, these shall be used.

5.4 Interfacing Additional Features

Additional features provided by AUTOSAR shall be considered during the realization of FOTA. However, this document does not describe how to use, implement or realize those features. Refer to the related AUTOSAR specification document for more details.

5.4.1 Security Features

The security features provided and supported by AUTOSAR shall be considered during the realization of FOTA. Those could be e.g.:

- Encrypted transfer from FOTA Master to FOTA Target
- Decrypt encrypted FOTA-Image
- Encrypt new data in flash memory
- Encrypted transfer from Dcm to the memory stack
- Support of HSM (hardware security module) features

5.4.2 Safety Features

In order to avoid communication errors between the FOTA Master and the FOTA Target ECUs, it always should be considered to use end-to-end protection mechanisms provided by AUTOSAR.

5.5 Error Handling

In general, all FOTA related errors and recovering action are expected to be done, or at least controlled, by the FOTA Master ECU. All error handling that is not covered by the FOTA Master ECU, the communication stack or the diagnostics stack, is implementation specific. This document does not deal with those implementation specific errors.

5.6 Sequence Charts

In this chapter relevant sequence charts are illustrated. They show the interaction between all necessary modules and instances from FOTA Target point of view. These affected instances are (at least):

- FOTA Master (ECU)
- Diagnostics stack (Dcm)
- FOTA Handler module (CDD)
- Program memory stack (Low-level Memory Driver)

Note: Keep in mind that the sequence charts listed in this chapter are proposals and are not fixed yet. Changes or extensions of features are up to the implementer.

Details about status information and additional data exchanged between FOTA Master and FOTA Target can be found in chapter 5.2.3.8.

5.6.1 (Re-)Initialization of the FOTA procedure

The initialization (and reinitialization due to interrupted or preempted FOTA procedure) sequences of the FOTA Target involve the following components:

- FOTA Master (ECU)
- Dcm (FOTA Target ECU)

The initialization of the FOTA procedure always looks the same, regardless if the update is triggered for the first time or if its resumed due to interruption (e.g. driving cycle change) or preemption (higher priority diagnostic protocol request). Information exchange using FOTA specific diagnostic service jobs shall indicate, at which position the installation procedure shall be resumed.

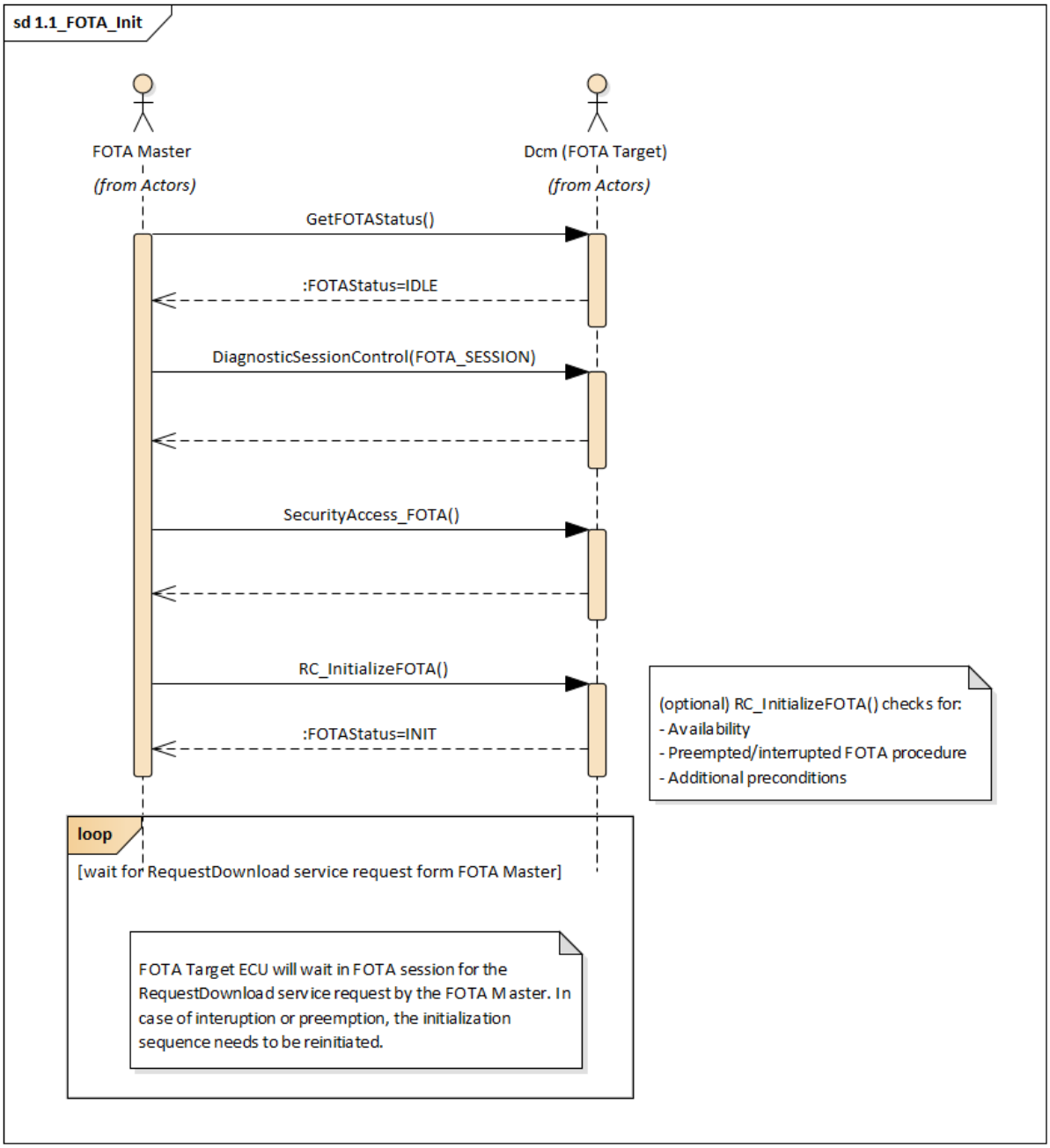


Figure 5.4: Initialization Sequence of the FOTA Target

5.6.2 Processing of FOTA Data Chunks

The installation sequence within a FOTA update procedure affects the following components:

- FOTA Master (ECU)
- Dcm (FOTA Target ECU)

- FOTA Handler module (CDD)

The data transfer of FOTA Data Chunks is triggered and controlled by the FOTA Master central instance. The Dcm module receives the data chunks in the Dcm RAM buffer. This RAM buffer is then forwarded to the FOTA Handler module implementation where it will be processed.

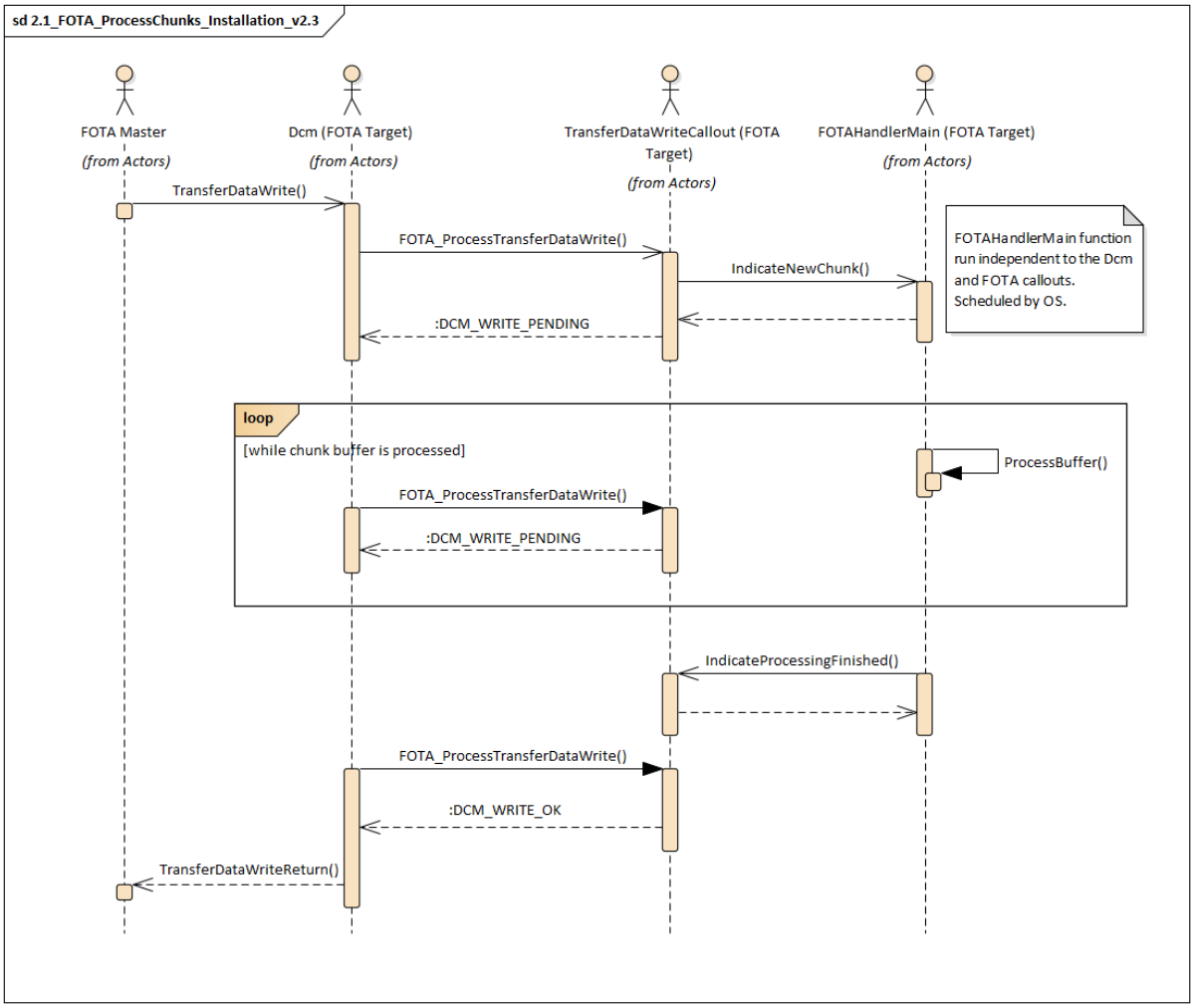


Figure 5.5: Installation sequence where a single FOTA chunk is transferred between UCM Master and FOTA Target ECU. This sequence is executed for each necessary FOTA chunk.

The solution described in this document shall work on the Dcm buffer as provided by the callout signature. Since the Dcm buffer, containing the FOTA data chunks, is exclusively accessed by the FOTA Handler module, the buffer payload can be expected as consistent even over several diagnostic protocol changes. Therefore the Dcm buffer for FOTA data chunks is blocked by the FOTA Handler until it has been processed completely. During the processing of the FOTA data chunk, the Dcm module returns a pending state which indicates the processing activity to the FOTA Master ECU. Once the FOTA data chunk has been processed completely, the TransferData service callout function tiggered by the Dcm module will return with a positive response.

5.6.3 Resume interrupted/preempted FOTA procedure

In order to resume an interrupted or preempted FOTA procedure, the FOTA diagnostic session needs to be reinitialized as well as the security access (as described in 5.6.1). The use of a FOTA specific diagnostic service jobs (refer to section 5.2.3.8 for details) shall return all required information needed to resume the FOTA procedure at a given position. After the reinitialization the FOTA procedure follows the same sequence as described in 5.6.2.

5.6.4 Activation of successfully flashed FOTA-Image

Since the sequences of activation of a newly installed SW image are mainly done in the bootloader context, which is out of scope of AUTOSAR, no sequence charts are listed.

For details about the activation process refer to section 5.2.1.3.

5.6.5 Rollback of FOTA procedure

Since the sequences of rollback to the previously active SW image (n-1) are mainly done in the bootloader context, which is out of scope of AUTOSAR, no sequence charts are listed.

For details about the rollback process refer to section 5.2.1.4.

6 Appendix

[1] Specification of Update and Configuration Management
AUTOSAR_AP_SWS_UpdateAndConfigurationManagement

[2] Requirements on Update and Configuration Management
AUTOSAR_AP_RS_UpdateAndConfigurationManagement

[3] Requirements on Firmware Over-The-Air
AUTOSAR_CP_RS_FirmwareOverTheAir

[4] Specification of Bulk NvData Manager
AUTOSAR_CP_SWS_BulkNvDataManager

[5] Glossary
AUTOSAR_FO_TR_Glossary

[6] ISO 14229-1(2013) – Unified diagnostic services (UDS) – Part 1: Application layer (Release 2013-03)
https://www.iso.org

[7] Specification of NVRAM Manager
AUTOSAR_CP_SWS_NVRAMManager

[8] Specification of Diagnostic Communication Manager
AUTOSAR_CP_SWS_DiagnosticCommunicationManager