Document Title	Remote Event Communication Protocol Specification
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	812

Document Status	Final
Part of AUTOSAR Standard	Foundation
Part of Standard Release	1.3.0

Document Change History

Date	Release	Changed by	Description
2017-12-08	1.3.0	AUTOSAR Release Management	• No content changes
2017-10-27	1.2.0	AUTOSAR Release Management	• Enhanced formal quality of requirements and requirements tracing
2017-03-31	1.1.0	AUTOSAR Release Management	• Editorial changes
2016-11-30	1.0.0	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 and overview 6
1.1 Protocol purpose and objectives . . . 10
1.2 Applicability of the Protocol . . . 10
1.2.1 Constraints and Assumptions . . . 10
1.2.2 Limitations . . . 10
1.2.2.1 Applicability to car domains . . . 11
1.2.2.2 Applicability to emission–related environments (OBD) 11
1.3 Dependencies . . . 11
1.3.1 Dependencies to other protocol layers . . . 11
1.3.2 Dependencies to other standards and norms . . . 11
1.3.3 Dependencies to the Application Layer . . . 11
2 Use Cases 13
2.1 Status Messages . . . 14
2.1.1 UC_000: Processing Heartbeat (Keep–Alive) Status . . . 14
2.1.2 UC_001: Processing DTC Status . . . 15
2.1.3 UC_002: Processing Monitor Disabled Status . . . 16
2.1.4 UC_003: Processing IUMPR Fault Detect Status . . . 17
2.1.5 UC_004: Processing DTR Data Status . . . 18
2.1.6 UC_005: Processing Vehicle Information Status . . . 19
2.1.7 UC_006: Processing Mode $4 Return Value Status . . . 20
2.1.8 UC_007: Processing Version Numbers Status . . . 21
2.2 Management Messages . . . 22
2.2.1 UC_008: Processing Mode $4 Notification Management Command . . . 22
2.2.2 UC_009: Processing Synchronization–All Management Command . . . 23
3 Requirements traceability 24
4 Acronyms, abbreviations and definitions 29
4.1 Acronyms and abbreviations . . . 29
4.2 Acronyms used in Protocol fields . . . 29
4.3 Terms and Definitions . . . 30
5 Protocol Specification 33
5.1 Message formats . . . 33
5.1.1 Common Message fields . . . 33
5.1.1.1 Message Type (MSG_TYPE) field format . . . 33
5.1.1.2 Client Identifier (CLIENT_ID) field format . . . 33
5.1.1.3 Reserved field format . . . 34
5.1.1.4 Message counter field format . . . 34
5.1.2 Heartbeat Status Message format . . . 34
5.1.3 DTC Status Message format . . . 36

5.1.3.1 Diagnostic Status Information (STATUS_1 – STATUS_X) field format . . . 36
5.1.4 Monitor Disabled Status Message format . . . 37
5.1.4.1 Monitor Disabled Status Information (STATUS_1 – STATUS_X) field format . . . 38
5.1.5 IUMPR Fault Detected Status Message format . . . 39
5.1.5.1 Fault Detected Status Information (STATUS_1 – STATUS_X) field format . . . 40
5.1.6 DTR Data Status Message format . . . 41
5.1.6.1 DTR Data Status Information (STATUS_1 – STATUS_X) field format . . . 42
5.1.7 Vehicle Information Values Status Message format . . . 43
5.1.7.1 Calibration identifier (CAL_ID) field format . . . 44
5.1.7.2 Calibration verification number (CVN) field format . . . 44
5.1.8 Mode $4 Return Value Status Message . . . 45
5.1.8.1 Mode $4 Return Value (MODE4_RET) field format . . . 45
5.1.9 Version Numbers Status Message format . . . 46
5.1.9.1 Version Numbers (VERSION) field format . . . 47
5.1.10 Mode $04 Notification Management Message format . . . 48
5.1.11 Diagnostic Synchronize–All Management Message format . . . 50
5.2 Communication matrix . . . 52
5.3 Communication Specification . . . 53
5.3.1 Fire & Forget Communication . . . 53
5.4 Endian Specification . . . 54
5.5 Message Fields Handling . . . 54
5.5.1 Message counter field . . . 54
5.6 Error Handling . . . 54
5.7 Data Types . . . 54
5.7.1 Type Client Identifier (TYP_CLID) . . . 54
5.7.2 Type Reserved structure (STRCT_RESVD) . . . 54
5.7.3 Type Message Counter (TYP_MSGCTR) . . . 55
5.7.4 Type DTC Index Identifier (TYP_DTC_INDEX_ID) . . . 55
5.7.5 Type Mode $4 Return Value (TYP_MODE4_RET) . . . 55
5.7.6 Diagnostic_Status_Information (DSI) structures . . . 55
5.7.6.1 Type DTC Status structure (STRCT_DTC_STAT) . . . 56
5.7.6.2 Type Monitor Disabled Status structure (STRCT_MON_DIS_STAT) . . . 57
5.7.6.3 Type IUMPR Fault Detected Status structure (STRCT_FLT_DET_STAT) . . . 57
5.7.6.4 Type DTR Monitoring Data Status structure (STRCT_DTR_DATA_STAT) . . . 59
5.7.6.5 Vehicle Information (CAL–ID/CVN) Types . . . 60
5.7.6.6 Type Version Numbers Status structure (STRCT_VERS_STAT) . . . 61
5.8 Type Definitions . . . 62
5.8.1 Type Message Type (TYP_MSTP) . . . 62

5.9 Sequence diagrams . . . 63
5.9.1 Process Synchronization Commands after Initialization . . . 63
5.9.1.1 D–OBD Scenario . . . 63
5.9.1.2 Thin–Client Scenario . . . 64
5.9.2 Process Mode $04 Notification Commands . . . 65
6 Configuration specification 66
6.1 Status Messages . . . 66
6.1.1 Heartbeat Status Message . . . 66
6.1.2 Diagnostic Status Messages . . . 66
6.1.3 Vehicle Information Values Status Message . . . 67
6.1.4 Mode $4 Return Status Message . . . 67
6.1.5 Version Numbers Status Messages . . . 67
6.2 Management Messages . . . 68
6.2.1 Mode $4 Notification Management Message . . . 68
6.2.2 Synchronize–All Management Message . . . 68
6.3 Configuration Parameters . . . 69
6.3.1 Heartbeat Interval Time . . . 69
6.3.2 Synchronize–All Message Return Time . . . 69
6.4 Published Information . . . 70
7 Protocol usage and guidelines 71
7.1 How–to and guidance for implementors . . . 71
7.2 Implementation examples for use cases . . . 71
8 Related documentation 72
8.1 Input documents . . . 72
8.2 Related standards and norms . . . 73
8.3 Related specification . . . 73

1 Introduction and overview

This RecM Bus Protocol specification specifies the format, message sequences and semantics of the AUTOSAR Bus Protocol "Remote Event Communication Manager" (RecM).

The protocol is employed for the sending and reception of OBD–relevant Diagnostic status information and RecM Bus Protocol management commands over a communication network by a RecM Bus Protocol compatible module. The specified messages are those exchanged between a RecM Bus Protocol compatible module and the underlying communication structure.

The RecM Bus Protocol targets distributed architectures where a standardized method to transmit the status of OBD diagnostics from one ECU to another ECU over a bus system is required because of limitations in the external tool (Scantool) used to extract the diagnostic information from the system (see Figure 1.1 for a distributed OBD system overview).

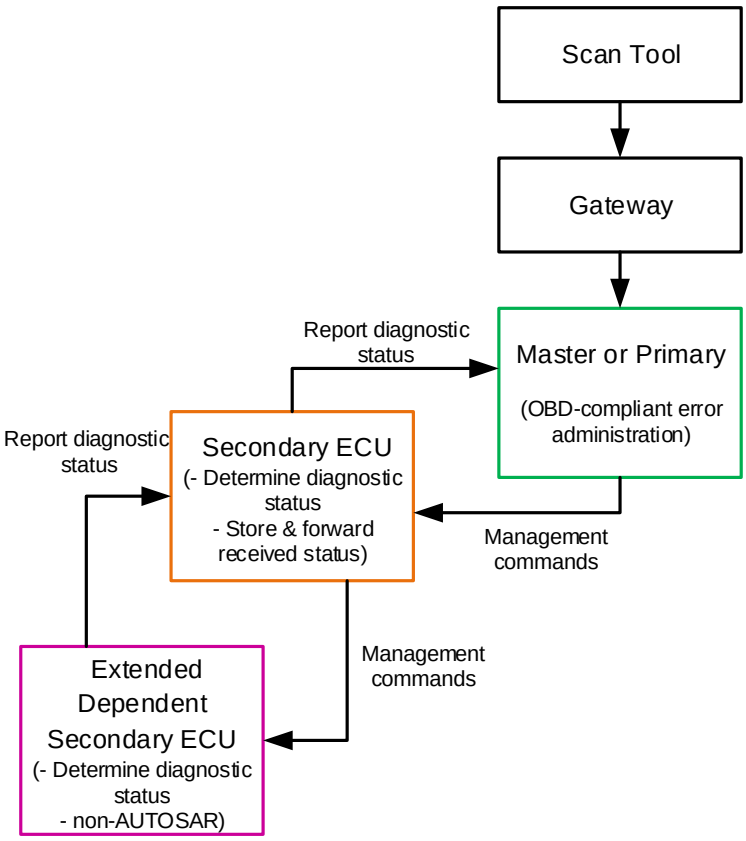


Figure 1.1: Distributed OBD Overview

The RECM environment, composed of RecM Bus Protocol compatible modules, supports the implementation of a "remote Diagnostic Event Status storage capability" (see

Figure 1.1). This functionality will allow the transfer of OBD diagnostic information from a local ECU to a remote ECU over a bus and storage of that data into NvRam by the Diagnostic Management module of the remote ECU. This will thus enable the extending of the Diagnostic Monitor interface across ECU boundaries (Inter–ECU communication). The RecM Bus Protocol compatible modules handle the communication part of a "remote fault–error manager".

There are two main functions implemented by the RecM Bus Protocol compatible modules. The first is the handling of the diagnostic events; i.e. the forwarding of diagnostic event status information and vehicle information generated in a source ECU (normally a dependent secondary) to a destination ECU (normally a primary) which is accessible by an OBD Scantool. This defined as the upstream direction since the direction of data transfer is from event source to destination.

The second functionality is the remote diagnostic management of the dependent secondary from the destination ECU, i.e. the forwarding of erase and synchronization commands from the primary to the secondaries. This is a separate channel and is defined as the downstream direction since the communication is from the destination to the event source.

Notes:

1. The term "RecM Bus Protocol compatible module" refers to a module that is capable of implementing the inter–ECU RecM Bus Protocol compatible communication specified by this document.
2. The terms "Source" or "Src" refer to the ECU that is the source of the diagnostic status information. The diagnostic status is generated in this ECU.
3. The terms "Destination" or "Des" refer to the ECU that is the destination storage location for the diagnostic status information. The diagnostic status is administrated in an OBD–compliant manner in this ECU.

RecM Bus Protocol compatible module sections:

The RecM Bus Protocol compatible modules serve either one or two functional roles. Each role provides distinct functionality:

- Source Role - This section contains the sub–modules necessary to transfer local DTC status data to the RecM Destination role section in a remote ECU to which it is attached. This section typically receives the status data from a Diagnostic Management module which, in turn, has received the status from a diagnostic status source. The Source role section of the RecM Bus Protocol compatible module is found in a Secondary ECU which is attached to a Primary ECU (as seen from an OBD architecture point of view). The Source role section of the RecM Bus Protocol compatible module is also found in an Extended Dependent Secondary which is attached to a Secondary ECU (as seen from an OBD architecture point of view).
- Destination Role - This section contains the sub–modules necessary to receive DTC status data from a RecM Source role section in an attached ECU (Sec-

ondary or Extended Dependent Secondary). This section typically forwards the status data to a Diagnostic Management module for management and memory storage. The Destination section of the RecM Bus Protocol compatible module is normally found in a Primary ECU which has one or more attached Secondary ECUs. It is also employed, in addition to a Source section, in a Secondary ECU which has one or more attached Extended Dependent Secondary ECUs.

A RecM Bus Protocol compatible module can have one or both of the above functional sections implemented, depending upon the ECU OBD type: Primary, Secondary or Extended Dependent Secondary. See RecM Bus Protocol Communication Overview, Figure 1.2.

Notes:

1. Master ECU: On the level of the RecM Bus Protocol, the Master ECU is considered to be equivalent to the Primary ECUs.
2. In the OBD architecture, the various OBD ECU types are defined as having the OBD–related connection attributes shown in Table 1.1 below. For the purposes of the RecM Bus Protocol, the type of OBD ECU defines the possible roles that an ECU can have, as follows in Table 1.1 below:

ECU Type:	Connection Attributes:	Roles:
Primary (incl. Master)		Destination Role only
Secondary	Attached to Primary	- Source role - Additional Destination role (if an Ext. Dep. Sec. is attached)
Extended Dependent Secondary	Attached to Secondary	Source role only

Table 1.1: ECU Attributes

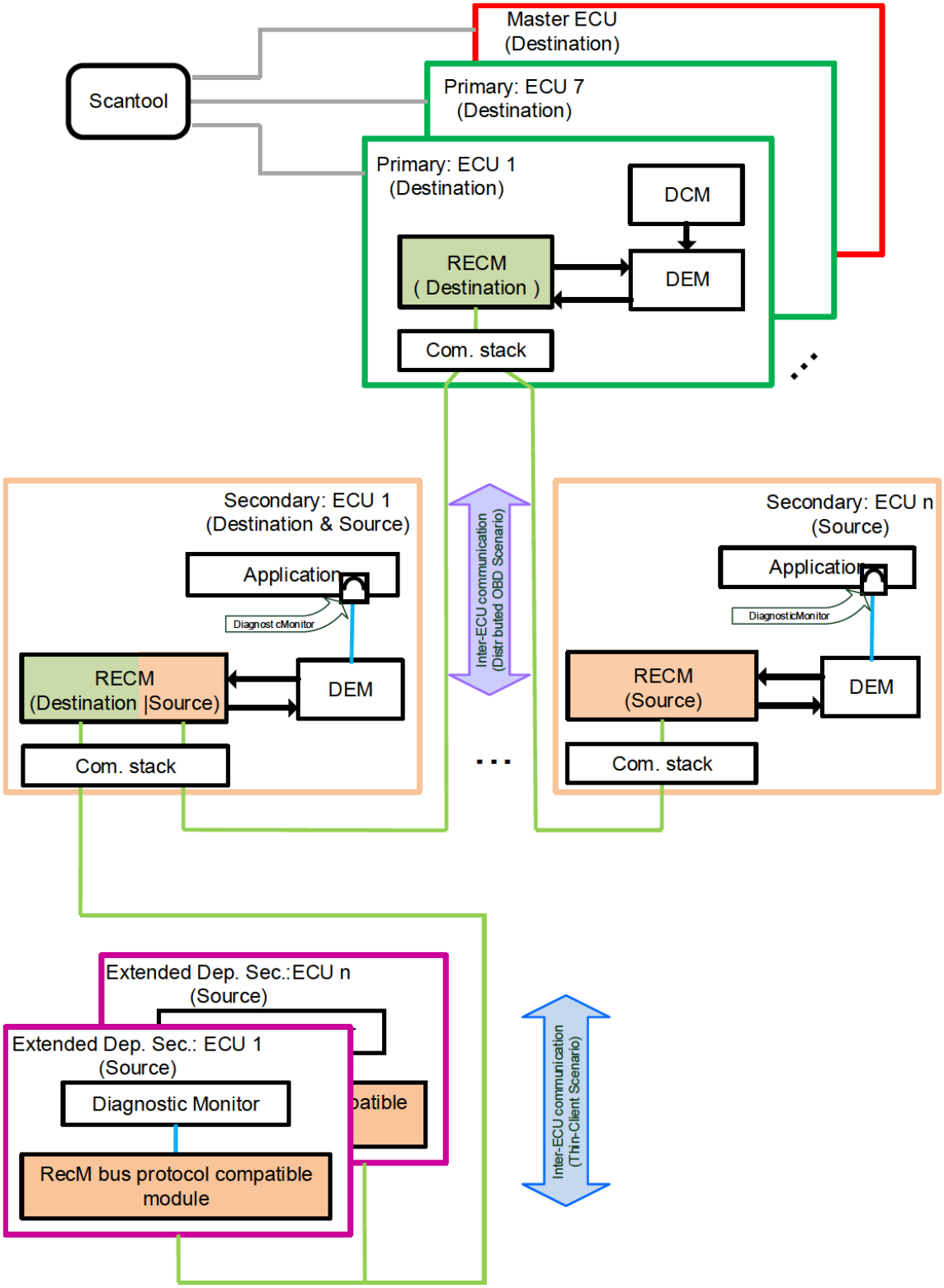


Figure 1.2: RecM Bus Protocol Communication Overview

1.1 Protocol purpose and objectives

The RecM Bus Protocol is used to communicate diagnostic status and vehicle information from a diagnostic state determination module in a (local source) ECU to a diagnostic administration module in another (remote destination) ECU.

The RecM bus protocol is also used to communicate RecM bus protocol management commands from a diagnostic administration module in a (remote destination) ECU to a diagnostic state determination module in another (local source) ECU.

The Recm bus protocol covers both of the two defined RecM environment scenarios: Distributed OBD and Thin–Client. The Distributed OBD scenario covers the interaction between a Primary ECU and its attached Secondary ECUs. The Thin–Client scenario covers the interaction between a Secondary ECU and its attached Dependant Secondary ECUs. The RecM Bus Protocol Communication Overview, Figure 1.2, shows a general overview of the communication elements and paths in a system with both scenarios implemented.

The RecM Bus Protocol specifies the message interface between a RecM Bus Protocol compatible module and the underlying communication structure.

1.2 Applicability of the Protocol

It is intended to use the RecM Bus Protocol for the communication between the RecM Bus Protocol compatible Source role section in one "source" ECU and the RecM Bus Protocol compatible Destination role section in another "destination" ECU.

1.2.1 Constraints and Assumptions

The RECM Bus Protocol is network–independent.

The RECM Bus Protocol does not specify the underlying communication stack with which a RecM Bus Protocol compatible module communicates.

1.2.2 Limitations

The corresponding requirements specification to this document is missing in this release even though the specification contains traceability information.

The introduction of a corresponding SRS Remote Event Communication is planned for the next Foundation release R1.1.0.

1.2.2.1 Applicability to car domains

No restrictions. The RecM Bus Protocol can be used for all car domains.

1.2.2.2 Applicability to emission–related environments (OBD)

This RecM Bus Protocol is intended to facilitate/support the fulfillment of the the emission–related requirements on distributed OBD systems given by legislator.

1.3 Dependencies

1.3.1 Dependencies to other protocol layers

None

1.3.2 Dependencies to other standards and norms

N/A

1.3.3 Dependencies to the Application Layer

The Source role of the RecM Bus Protocol compatible module in a Secondary ECU provides the Vehicle Information (CAL–ID & CVN) interface to a CAL–ID & CVN "generator" SWC to use for reporting CAL–ID & CVN values to the RecM Bus Protocol compatible module.

The diagram, Figure 1.3: RecM and Generator SW–C, illustrates this interaction in a "classic" AUTOSAR environment.

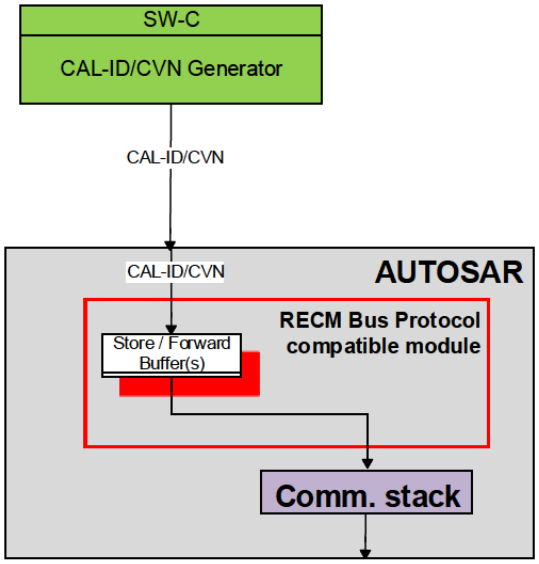


Figure 1.3: Cal–ID/CVN Generator SW–C and RecM Bus Protocol compatible module interaction

The Destination role of the RecM Bus Protocol compatible module in a Primary ECU requires a Vehicle Information (CAL–ID & CVN) interface to a CAL–ID & CVN "handler" SWC to use for reporting CAL–ID & CVN values to this "handler" SWC.

The diagram, Figure 1.4: RecM and Handler SW–C, illustrates this interaction in a "classic" AUTOSAR environment..

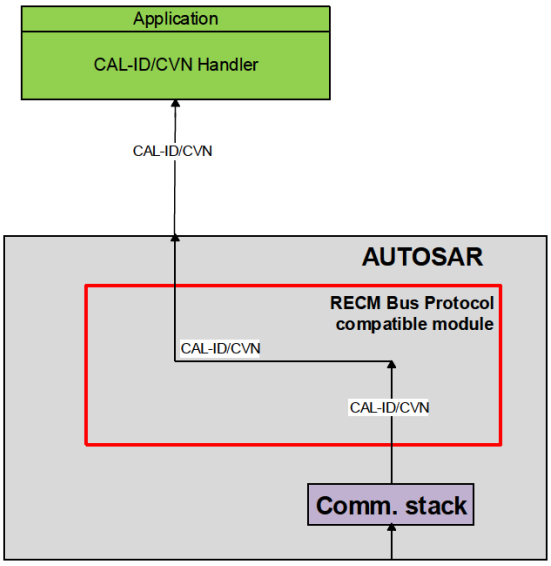


Figure 1.4: RecM Bus Protocol compatible module and Cal–ID/CVN Handler SW–C interaction

2 Use Cases

This chapter describes the use cases which can be realized in an environment whereby an ECU implements the RecM Bus Protocol. The Use Cases are described in Table 2.1.

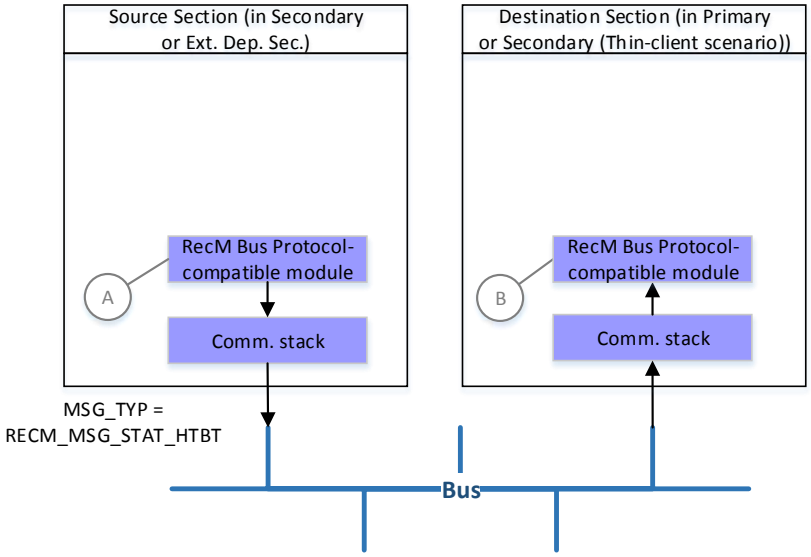
ID:	Name:	Description:
UC_000	Processing Heartbeat (Keep–Alive)	Src: Heartbeat timer timeout RecM Protocol Compatible Module (RPCM) sends Heartbeat msg to Destination ECU Des: RPCM receives Heartbeat msg from Source ECU RPCM initiates synchronization
UC_001	Processing DTC Status	Src: DTC status forwarded to RecM–BPCM RPCM sends DTC msg Des: RPCM receives DTC msg DTC status saved to memory
UC_002	Processing Monitor Disabled Status	Src: Monitor Disabled report forwarded to RPCM RPCM sends Mon. Dis. msg Des: RPCM receives Mon. Dis. msg Mon. Dis. status saved to memory
UC_003	Processing IUMPR Fault Detected Status	Src: Fault Detection report forwarded to RPCM RPCM sends Flt. Det. msg Des: RPCM receives Flt. Det. msg Flt. Det. status saved to memory
UC_004	Processing DTR Data Status	Src: DTR Data status forwarded to RecM–BPCM RPCM sends DTR Data msg Des: RPCM receives DTR Data msg DTR Data status saved to memory
UC_005	Processing Vehicle Information Status	Src: SWC forwards Veh. Info to RPCM RPCM sends Veh. Info msg Des: RPCM receives Veh. Info msg RPCM forwards Veh. Info to SWC
UC_006	Processing Mode $4 Return Value Status	Des: Mode $4 ret forwarded to RPCM RPCM sends Ret msg Src: RPCM receives Ret msg RPCM forwards Clr. DTC ret for processing
UC_007	Processing Versions Status	Des: RPCM sends version after initialization RPCM sends Version msg Src: RPCM receives Version msg RPCM verifies version
UC_008	Processing Mode $4 Notification Management Command	Des: Mode $4 command forwarded to RPCM RPCM sends notification msg Src: RPCM receives notification msg RPCM forwards notification for processing
UC_009	Processing Synchronization–All Management Command	Des: RPCM requires for (re)synchronization RPCM sends Sync–All msg Src: RPCM receives Sync–All msg RPCM processes Sync command

Table 2.1: RecM Protocol Use Cases

2.1 Status Messages

Status messages are all messages that originate from a RecM Bus Protocol compatible module's Source Section in one ECU and which are sent to the RecM Bus Protocol compatible module's Destination Section in another ECU.

2.1.1 UC_000: Processing Heartbeat (Keep–Alive) Status

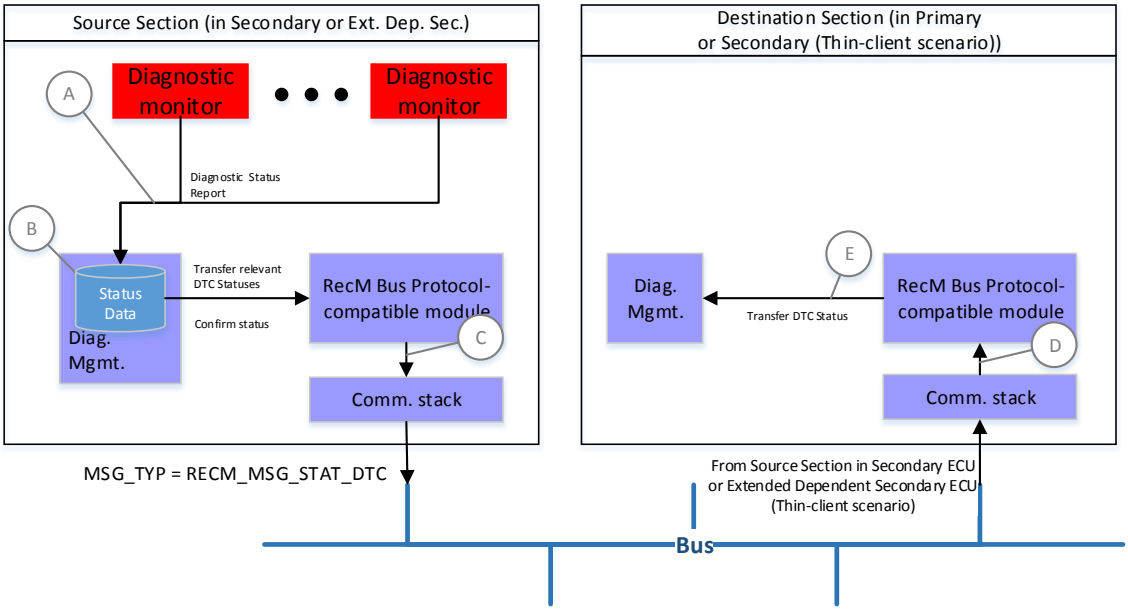


A The „RecM Bus Protocol"-compatible module in the Source Section ECU sends a Heartbeat message over the bus via the communication stack interface when the maximum allowable time interval between sent messages has been exceeded.

B The „RecM Bus Protocol"-compatible module on the Destination Section receives the Heartbeat Message from the communication stack and processes it. This includes resetting the message timeout timer.

Figure 2.1: UseCase 000: Processing Heartbeat (Keep–Alive)

2.1.2 UC_001: Processing DTC Status



A A Diagnostic Monitor sends diagnostis status information to the Diagnostic Management module.

B The Diagnostic Management module processes the diagnostic status information and stores the resulting status data. For OBD-relevant DTCs, the Diagnostic Management module informs the „RecM Bus Protocol“-compatible module whenever the status data has changed.

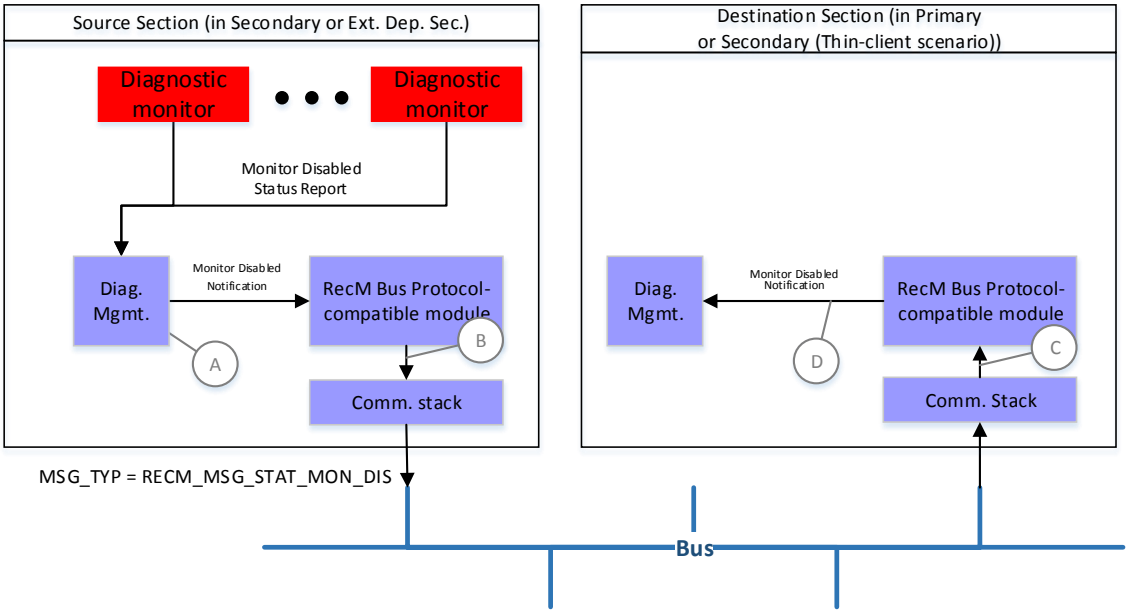
C The „RecM Bus Protocol“-compatible module buffers the changed diagnostic information and sends it via the communication stack interface over the bus when a trigger condition is met.

D Diagnostic Status messages are received over the bus from the communication stack interface and decoded by the „RecM Bus Protocol“-compatible module. The source ECU is either a Secondary (D-OBD scenario) or Extended Dependent Secondary (Thin-client scenario)

E The „RecM Bus Protocol“-compatible module sends the received DTC status to the Diagnostic Management module which then processes the received status data.

Figure 2.2: UseCase 001: Processing DTC Status

2.1.3 UC_002: Processing Monitor Disabled Status



A The Diagnostic Management module notifies the „RecM Bus Protocol“-compatible module whenever it has received a Monitor Disabled Status report (SetEventDisabled) from a Diagnostic Monitor interface.

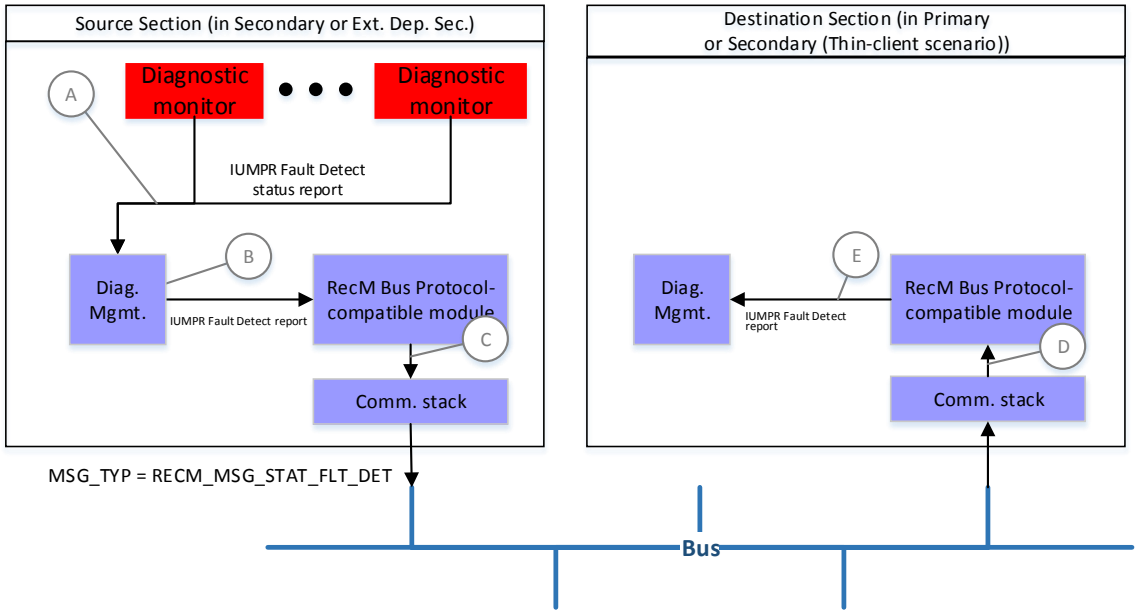
B „RecM Bus Protocol“-compatible module buffers the Monitor Disabled Status and sends it via the communication stack interface over the bus when a trigger condition is met.

C Monitor Disabled Status messages are received over the bus from the communication stack interface and decoded by the „RecM Bus Protocol“-compatible module.

D The „RecM Bus Protocol“-compatible module sends the received Monitor Disabled indication to the Diagnostic Management module which then processes the received Monitor status data.

Figure 2.3: UseCase 002: Processing Monitor Disabled Status

2.1.4 UC_003: Processing IUMPR Fault Detect Status



A Diagnostic Monitor reports Fault Detection Possible, individual denominator or locked Status to the Diagnostic Management module.

B The Diagnostic Management module notifies the „RecM Bus Protocol“-compatible module whenever it has received a Fault Detection report (numerator, denominator or locked) from a Diagnostic Monitor.

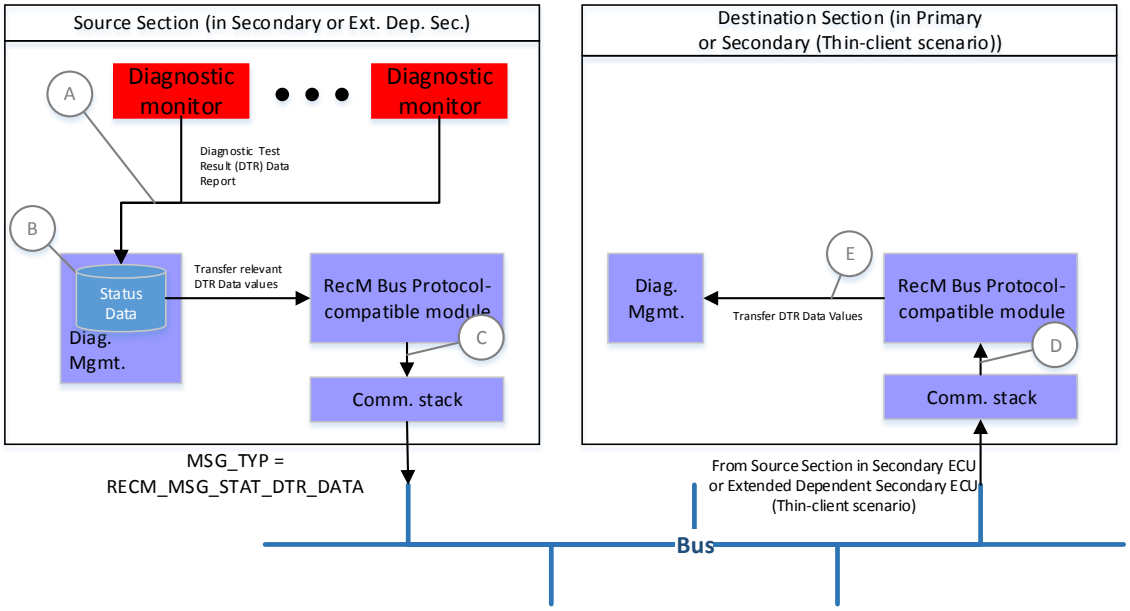
C The „RecM Bus Protocol“-compatible buffers the changed status information and sends it via the communication stack interface over the bus when a trigger condition is met.

D Fault Detection Possible Status messages are received over the bus from the communication stack interface and decoded by the „RecM Bus Protocol“-compatible module.

E The „RecM Bus Protocol“-compatible module informs the Diagnostic Management module that a monitor reported that a Fault Detected indication (numerator, denominator or locked) has been received. The Diagnostic Management module subsequent increments the numerator or denominator of the respective In–Use–Monitor Performance Ratio.

Figure 2.4: UseCase 003: Processing IUMPR Fault Detect Status

2.1.5 UC_004: Processing DTR Data Status



A A Diagnostic Monitor sends „diagnostic test result“ data status information to the Diagnostic Management module.

B The Diagnostic Management module processes the DTR Data status information and stores the resulting status data. For OBD-relevant DTCs, the Diagnostic Management module informs the „RecM Bus Protocol“-compatible module whenever the status data has changed.

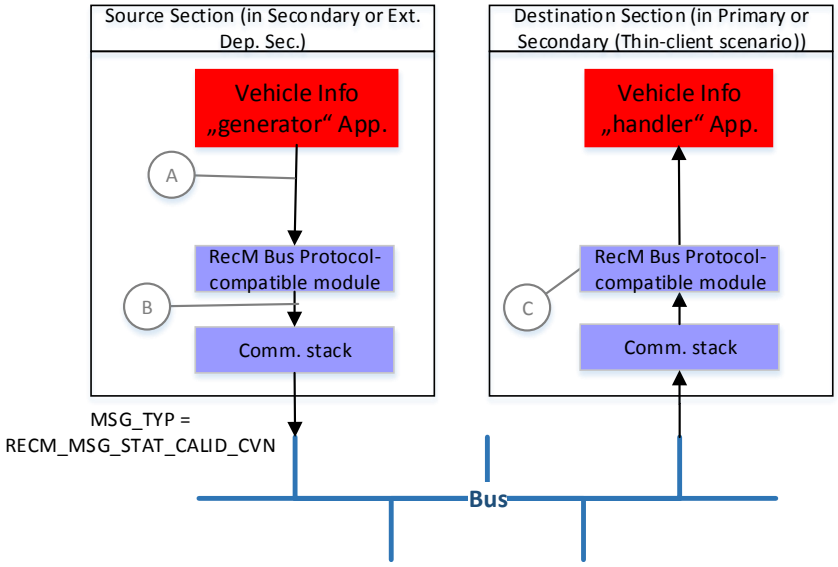
C The „RecM Bus Protocol“-compatible module buffers the changed DTR Data information and sends it via the communication stack interface over the bus when a trigger condition is met.

D Diagnostic Status messages are received over the bus from the communication stack interface and decoded by the „RecM Bus Protocol“-compatible module. The source ECU is either a Secondary (D-OBD scenario) or Extended Dependent Secondary (Thin-client scenario)

E The „RecM Bus Protocol“-compatible module sends the received DTR Data status to the Diagnostic Management module which then processes the received status data.

Figure 2.5: UseCase 004: Processing DTR Data Status

2.1.6 UC_005: Processing Vehicle Information Status



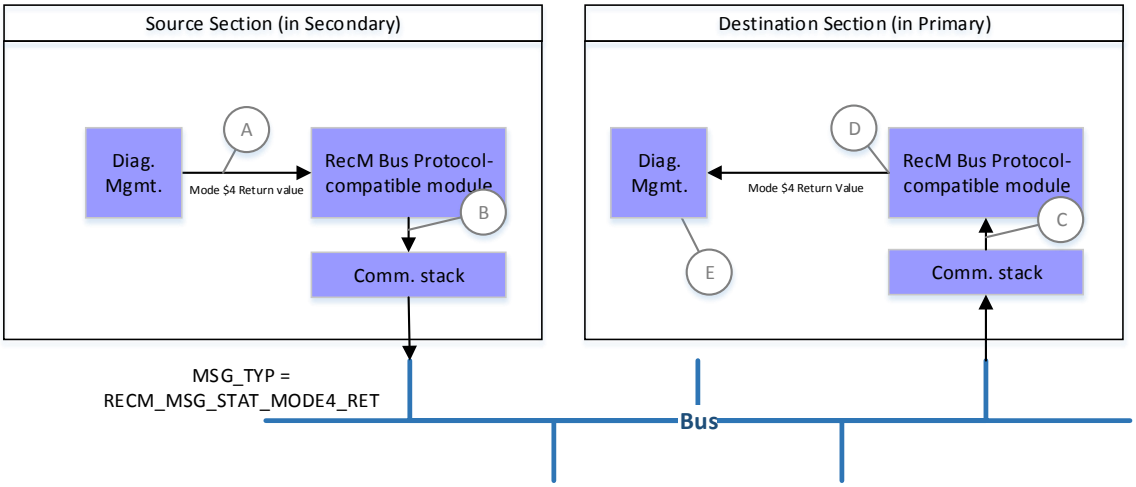
A A CAL–ID/CVN „generator“ application in the Source Section ECU sends the Vehicle Information to the „RecM Bus Protocol“-compatible module.

B The „RecM Bus Protocol“-compatible module stores the CALID/CVN Values in a buffer and sends it via the communicaion stack interface over the bus.

C The „RecM Bus Protocol“-compatible module on the Destination Section receives the CALID/CVN Values via the communication stack interface and forwards it to the application that handles the vehicle information.

Figure 2.6: UseCase 005: Processing Vehicle Information Status

2.1.7 UC_006: Processing Mode $4 Return Value Status



A — After the Diagnostic Management module has processed the Mode $4 indication, it informs the „RecM Bus Protocol“-compatible module of the result via a return value.

B — The „RecM Bus Protocol“-compatible module sends the Diagnostic Management module return code via the communication stack interface over the bus.

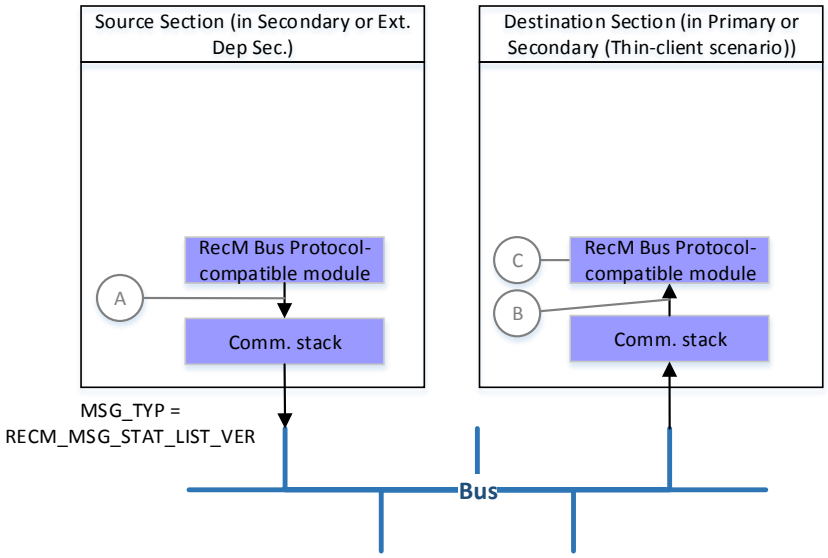
C — The „RecM Bus Protocol“-compatible module on the Destination side receives one return code from each attached Secondary ECU.

D — The „RecM Bus Protocol“-compatible module combines all received return codes and forwards the combined result to the Diagnostic Management module.

E — The Diagnostic Management module combines the return value received from the „RecM Bus Protocol“-compatible module with its own internal return value to form a combined return value.

Figure 2.7: UseCase 006: Processing Mode $4 Return Value

2.1.8 UC_007: Processing Version Numbers Status



A The „RecM Bus Protocol“-compatible module in the Source Section ECU sends the Master List Version via the communication stack interface over the bus after initialization is completed.

B The „RecM Bus Protocol“-compatible module on the Destination Section receives the Master Lists Version via the communication stack interface and compares this version number with its internal list.

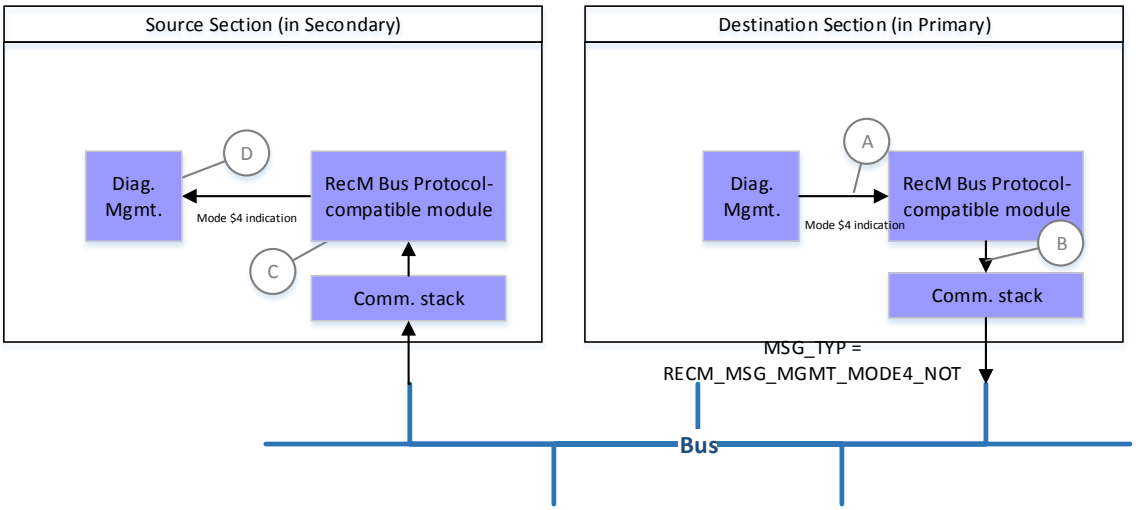
C If the version is incompatible, an error is generated. All communication from the sender ECU will then be ignored.

Figure 2.8: UseCase 007: Processing Version Numbers

2.2 Management Messages

Management messages are all messages that originate in the RecM Bus Protocol compatible module's Destination Section in one ECU and which are sent to the RecM Bus Protocol compatible module's Source Section in another ECU.

2.2.1 UC_008: Processing Mode $4 Notification Management Command



A The Diagnostic Management module forwards the OBD Mode $4 command received from the Scantool to the „RecM Bus Protocol"-compatible module.

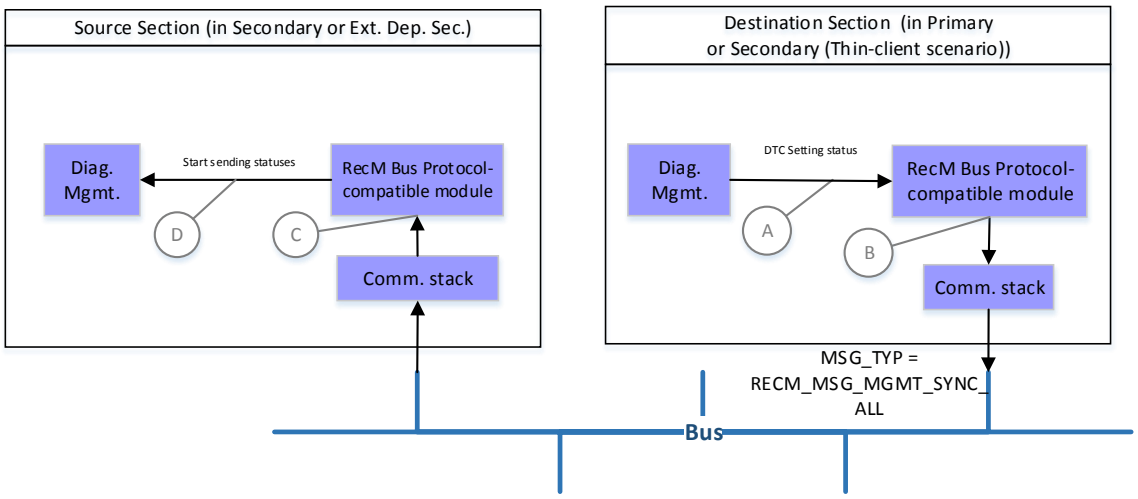
B The „RecM Bus Protocol"-compatible moduel sends the Mode $4 Received Notification to all attached Secondary ECUs via the communication stack iterface over the bus.

C The „RecM Bus Protocol"-compatible module in the Secondary receives the Mode $4 Received Notification via the communication stack interface and informs the Diagnostic Management module.

D The Diagnostic Management module clears all OBD-relevant DTCs.

Figure 2.9: UseCase 008: Processing Mode $4 Notification Management Command

2.2.2 UC_009: Processing Synchronization–All Management Command



A Whenever the Destination Section Diagnostic Management module informs the „RecM Bus Protocol“-compatible module that the current „DTC Setting“ is set to START, a Synchronization ‚all‘ message is sent to all attached Secondary ECUs. In addition, when the „RecM Bus Protocol“-compatible module detects a communication error (such as a message not received timeout) associated with a particular Secondary, a Synch. ‚all‘ message is sent to that Secondary ECU.

B The Destination Section „RecM Bus Protocol“-compatible module sends a Synch-All synchronization message via the communication stack interface over the bus to the Source Section „RecM Bus Protocol“-compatible module.

C The „RecM Bus Protocol“-compatible Source Section module receives the Sync message from the bus via the communicatio stack interface. The „RecM Bus Protocol“-compatible module sends all stored current diagnostic statuses (DTC, Monitor Disabled & IUMPR Fault Detected statuses) to the Destination Section and/or notifies the Dem to send all current statuses.

D The „RecM Bus Protocol“-compatible Source Section module notifies the Diagnostic Management module to start transferring the latest status information.

E The „RecM Bus Protocol“-compatible Source Section module starts sending the required status messages after receiving the updated status data.

Figure 2.10: UseCase 009: Processing Synchronization–All Management Command

3 Requirements traceability

The following table references the SRS requirements which are fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_RECM_00001]	No description	[PRS_RECM_00001] [PRS_RECM_00002] [PRS_RECM_00003] [PRS_RECM_00004] [PRS_RECM_00005] [PRS_RECM_00006] [PRS_RECM_00015] [PRS_RECM_00016] [PRS_RECM_00018] [PRS_RECM_00019] [PRS_RECM_00020] [PRS_RECM_00022] [PRS_RECM_00023] [PRS_RECM_00024] [PRS_RECM_00026] [PRS_RECM_00027] [PRS_RECM_00028] [PRS_RECM_00030] [PRS_RECM_00031] [PRS_RECM_00035] [PRS_RECM_00036] [PRS_RECM_00039] [PRS_RECM_00040] [PRS_RECM_00043] [PRS_RECM_00044] [PRS_RECM_00045] [PRS_RECM_00047] [PRS_RECM_00048] [PRS_RECM_00049] [PRS_RECM_00051] [PRS_RECM_00052] [PRS_RECM_00053] [PRS_RECM_00058] [PRS_RECM_00059] [PRS_RECM_00060] [PRS_RECM_00063] [PRS_RECM_00065] [PRS_RECM_00067] [PRS_RECM_00068] [PRS_RECM_00069] [PRS_RECM_00070] [PRS_RECM_00071] [PRS_RECM_00072] [PRS_RECM_00073] [PRS_RECM_00075] [PRS_RECM_00076] [PRS_RECM_00078] [PRS_RECM_00080]

		[PRS_RECM_00084] [PRS_RECM_00085] [PRS_RECM_00086] [PRS_RECM_00087] [PRS_RECM_00088] [PRS_RECM_00089] [PRS_RECM_00090] [PRS_RECM_00091] [PRS_RECM_00092] [PRS_RECM_00094] [PRS_RECM_00095] [PRS_RECM_00096] [PRS_RECM_00098] [PRS_RECM_00099] [PRS_RECM_00100] [PRS_RECM_00101] [PRS_RECM_00102] [PRS_RECM_00103] [PRS_RECM_00104] [PRS_RECM_00105]
[SRS_RECM_00004]	No description	[PRS_RECM_00050] [PRS_RECM_00079]
[SRS_RECM_00007]	No description	[PRS_RECM_00060]
[SRS_RECM_00008]	No description	[PRS_RECM_00001] [PRS_RECM_00002] [PRS_RECM_00003] [PRS_RECM_00004] [PRS_RECM_00005] [PRS_RECM_00006] [PRS_RECM_00009] [PRS_RECM_00010] [PRS_RECM_00011] [PRS_RECM_00012] [PRS_RECM_00013] [PRS_RECM_00014] [PRS_RECM_00017] [PRS_RECM_00021] [PRS_RECM_00025] [PRS_RECM_00029] [PRS_RECM_00032] [PRS_RECM_00034] [PRS_RECM_00037] [PRS_RECM_00038] [PRS_RECM_00041] [PRS_RECM_00042] [PRS_RECM_00046] [PRS_RECM_00051]

		[PRS_RECM_00056] [PRS_RECM_00057] [PRS_RECM_00061] [PRS_RECM_00062] [PRS_RECM_00066] [PRS_RECM_00074] [PRS_RECM_00077] [PRS_RECM_00081] [PRS_RECM_00082] [PRS_RECM_00083] [PRS_RECM_00093]
[SRS_RECM_00012]	No description	[PRS_RECM_00017] [PRS_RECM_00018] [PRS_RECM_00019] [PRS_RECM_00020]
[SRS_RECM_00013]	No description	[PRS_RECM_00050] [PRS_RECM_00079]
[SRS_RECM_00019]	No description	[PRS_RECM_00046] [PRS_RECM_00047] [PRS_RECM_00048] [PRS_RECM_00049]
[SRS_RECM_00023]	No description	[PRS_RECM_00017] [PRS_RECM_00018] [PRS_RECM_00019] [PRS_RECM_00020]
[SRS_RECM_00026]	No description	[PRS_RECM_00011] [PRS_RECM_00012] [PRS_RECM_00013] [PRS_RECM_00020] [PRS_RECM_00042] [PRS_RECM_00046] [PRS_RECM_00047] [PRS_RECM_00048] [PRS_RECM_00056] [PRS_RECM_00057] [PRS_RECM_00061] [PRS_RECM_00062] [PRS_RECM_00066] [PRS_RECM_00074] [PRS_RECM_00077] [PRS_RECM_00081] [PRS_RECM_00082] [PRS_RECM_00083]
[SRS_RECM_00027]	No description	[PRS_RECM_00081] [PRS_RECM_00083]
[SRS_RECM_00031]	No description	[PRS_RECM_00014] [PRS_RECM_00015] [PRS_RECM_00016] [PRS_RECM_00050]

[SRS_RECM_00036]	No description	[PRS_RECM_00018] [PRS_RECM_00022] [PRS_RECM_00026] [PRS_RECM_00030] [PRS_RECM_00035] [PRS_RECM_00039] [PRS_RECM_00044] [PRS_RECM_00048] [PRS_RECM_00053] [PRS_RECM_00094]
[SRS_RECM_00037]	No description	[PRS_RECM_00018] [PRS_RECM_00022] [PRS_RECM_00026] [PRS_RECM_00030] [PRS_RECM_00035] [PRS_RECM_00039] [PRS_RECM_00044] [PRS_RECM_00048] [PRS_RECM_00053] [PRS_RECM_00094]
[SRS_RECM_00038]	No description	[PRS_RECM_00014] [PRS_RECM_00015] [PRS_RECM_00016] [PRS_RECM_00050] [PRS_RECM_00082]
[SRS_RECM_00039]	No description	[PRS_RECM_00020] [PRS_RECM_00024] [PRS_RECM_00028] [PRS_RECM_00041] [PRS_RECM_00053] [PRS_RECM_00096]
[SRS_RECM_00044]	No description	[PRS_RECM_00042] [PRS_RECM_00043] [PRS_RECM_00044] [PRS_RECM_00045]
[SRS_RECM_00045]	No description	[PRS_RECM_00009] [PRS_RECM_00012] [PRS_RECM_00014]
[SRS_RECM_00046]	No description	[PRS_RECM_00010]
[SRS_RECM_00047]	No description	[PRS_RECM_00011] [PRS_RECM_00013]
[SRS_RECM_00049]	No description	[PRS_RECM_00046]
[SRS_RECM_00050]	No description	[PRS_RECM_00020] [PRS_RECM_00024] [PRS_RECM_00028] [PRS_RECM_00032] [PRS_RECM_00037] [PRS_RECM_00041] [PRS_RECM_00096]
[SRS_RECM_00051]	No description	[PRS_RECM_00048] [PRS_RECM_00049]
[SRS_RECM_00052]	No description	[PRS_RECM_00043] [PRS_RECM_00044] [PRS_RECM_00047]

[SRS_RECM_00067]	No description	[PRS_RECM_00050]
[SRS_RECM_00068]	No description	[PRS_RECM_00011]
[SRS_RECM_00074]	No description	[PRS_RECM_00021] [PRS_RECM_00022] [PRS_RECM_00023] [PRS_RECM_00024]
[SRS_RECM_00075]	No description	[PRS_RECM_00025] [PRS_RECM_00026] [PRS_RECM_00027] [PRS_RECM_00028]
[SRS_RECM_00076]	No description	[PRS_RECM_00029] [PRS_RECM_00030] [PRS_RECM_00031] [PRS_RECM_00032] [PRS_RECM_00038] [PRS_RECM_00039] [PRS_RECM_00040] [PRS_RECM_00041] [PRS_RECM_00052] [PRS_RECM_00053]

4 Acronyms, abbreviations and definitions

There are acronyms and abbreviations relevant to this document that are included in the [1, AUTOSAR glossary].

4.1 Acronyms and abbreviations

Abbreviation	Description
BSW	Basic Software
CAL-ID	Calibration ID
CAN	Controller Area Network
CVN	Calibration Verification Number
Dem	Diagnostic Event Manager
Sec	Secondary ECU
DTC	Diagnostic Trouble Code
DTC ID	DTC identifier
DTR	Diagnostic Test Result
ECU	Electronic Control Unit
HW	Hardware
I–PDU	Interaction Layer Protocol Data Unit
I–PDU–ID	PDU Identifier
ID	Identification/Identifier
ISO	International Standardization Organization
IUMPR	In Use Monitoring Performance Ratio (OBD Term)
LIN	Local Interconnect Network
MIL	Malfunction Indicator Light (SAE J1979) or Lamp (SAE J1939)
msg	message
NRC	Negative Response Code
NVRAM	Non volatile RAM
OBD	On–Board–Diagnostics
OEM	Original Equipment Manufacturer (Automotive Manufacturer)
OS	Operating System
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PID	Parameter Identification (SAE J1587 or SAE J1979)
RAM	Random Access Memory
RecM	Remote Event Communication Manager
RecM	Remote Event Communication Manager – Protocol Compatible Module
RTE	Runtime Environment
SW	Software
SW-C	Software Component
TP	Transport Protocol

4.2 Acronyms used in Protocol fields

Protocol Field Acronyms	Description
CALID_CVN_STAT	Calid/CVN Status structure type
CAL-ID	Calibration Identifier field
CLIENT_ID	Client ECU identifier field
COMMAND	Synchronization command field
CVN	Calibration Verification Number field
DTC_CLR_RET	Clear–DTC return value field
DTC_STAT	DTC Status structure type
FLT_DET_STAT	IUMPR Fault Detect Status structure type
DTR_DATA_STAT	DTR Data Values structure type
FORMAT	Status Field format field
INDEX_	Index identifier field
MON_DIS_STAT	Monitor Disabled Status structure type
MSG_CNT	Message counter field
MSG_TYPE	Message type field
RESERVED	Reserved field
STATUS_	Status field

4.3 Terms and Definitions

Term	Description
Calibration ID	(Cal–Id) Vehicle–specific Calibration Identifier: returned during a Service $09 – Request Vehicle Information request.
Calibration Verification Number	(CVN) Vehicle–specific Calibration Verification Number: returned during a Service $09 – Request Vehicle Information request.
Diagnostic Monitor	A diagnostic monitor is a routine entity determining the proper functionality of a component. Alternatively the term "diagnostic function" can be used.
Diagnostic Monitor Interface	A diagnostic monitor interface is the interface between a diagnostic monitor and a Diagnostic Management module.
Diagnostic Index Identifier	An identifier value used to identify a particular OBD DTC in a status or management message. The Diagnostic Index Identifier value used for a DTC is obtained from the respective Master List.
Diagnostic Management module	A module capable of managing diagnostic information received from a Diagnostic Monitor and interacting with a "RecM Bus Protocol"–compatible module.
Diagnostic trouble code	A 'Diagnostic trouble code' defines a unique identifier (shown to the diagnostic tester) mapped to a 'Diagnostic event' of the Dem module. The Dem provides the status of 'Diagnostic trouble codes' to the RecM module.
Denominator	The denominator of a specific monitor m (Denominator) is a counter indicating the number of vehicle driving events, taking into account conditions specific to that specific monitor.
Dependent Secondary ECU	Dependent / Secondary ECUs and Dependent / Secondary (or dep. / sec. or Secondary) ECUs are always related to a Master or a Primary ECU.
Destination	The Destination refers to the ECU that is the destination storage location for the diagnostic status information. The diagnostic status is administrated in an OBD–compliant manner in this ECU.
Destination Section	The Destination section of a RecM module is the remote section where the statuses are stored. It is located in a Primary ECU (RecMRole parameter of RecmRoleConfig container = Primary) or a Secondary ECU (RecMRole = SecondaryDualRole).

Dynamic Length Signal	A dynamic length signal is a signal whose length can vary at run–time.
Dynamic Length I–PDU	A dynamic length I–PDU is an I–PDU containing a dynamic length signal. It length varies depending on the length of the included dynamic length signal. Dynamic length I–PDUs will be transmitted via TP.
Event memory	An event memory (e.g. Primary memory) consists of several event memory entries.
Extended Dependent Secondary ECU	Extended Dependent Secondary ECUs (or Ext. Dep. Sec.) ECUs are always related to a Dependent Secondary ECU.
Fire & forget	Messages whereby there is no response message after a request is sent.
General Denominator	The general denominator is a counter indicating the number of times a vehicle has been operated, taking into account general conditions.
In-Use performance ratio	The in-use performance ratio (IUPR) of a specific monitor m of the OBD system is: IUPRm = Numeratorm / Denominatorm
Init Value	I–PDUs and signals are set to the Initial Value by the RECM module after start–up. This value is used until it is overwritten.
I–PDU	Interaction Layer Protocol Data Unit. An I–PDU carries signals. An I–PDU consists of data (buffer), length and I–PDU ID. The PDU router will mainly route I–PDUs (exception is routing–on–the–fly) It is defined in [2, OSEK COM].
Inter–ECU–communication	Communication between two or more ECU; for example via a CAN or FlexRay network
Lower Layer Modules (Lo)	Modules below the PDU Router. This layer may include CAN, LIN, FlexRay, Ethernet communication interface modules and the respective TP modules. Modules used are configured in the configuration.
Master ECU	As a primary ECU a Master ECU stores “it’s own” and “reported errors” of related dep. / sec ECUs in it’s `event memory`. Beside this a Master has to fulfill special Master tasks as MIL Master or provision of “general nominator” information.
Master Index List	A double–entry list with each entry containing an UDS DTC value and its corresponding universal Diagnostic Index Identifier value. Universal in this case meaning that the DTC value represented by the Diagnostic Index Identifier is known system–wide.
Master IUMPR List	A double–entry list with each entry containing an UDS DTC value for an IUMPR–related diagnostic and its corresponding universal Diagnostic Index Identifier value. Universal in this case meaning that the DTC value represented by the Diagnostic Index Identifier is known system–wide.
Master Monitor Disabled List	A double–entry list with each entry containing an UDS DTC value for a Monitor Disabled related diagnostic and its corresponding universal Diagnostic Index Identifier value. Universal in this case meaning that the DTC value represented by the Diagnostic Index Identifier is known system–wide.
Message	OSEK–COM always uses the synonym *message*. In AUTOSAR, *message* is replaced by *signal* but with the same meaning.
Message counter	Counter attached to each status message (from Source to Destination) which is incremented after each message is sent.
Mode $4	OBD Mode $4 notification from Scantool indicates that every OBD–relevant DTC should be cleared.

Numerator	The numerator of a specific monitor m (Numeratorm) is a counter indicating the number of times a vehicle has been operated such that all monitoring conditions necessary for that specific monitor to detect a malfunction have been encountered.
Operating cycle	An 'Operating cycle' is the base of the event qualifying and also Dem scheduling (e.g. ignition key off-on cycles, driving cycles, etc.)
OBD	On-Board Diagnostics, or OBD is a generic term referring to a vehicle's self-diagnostic and reporting capability. OBD systems give the vehicle owner or a repair technician access to state of health information for various vehicle sub-systems.
OBD ECUs	"In a vehicle there can be 4 different types of OBD ECUs: • Master ECU (one per vehicle) • Primary ECU (several per vehicle) • Secondary ECU (several per vehicle) • Extended Dependent Secondary ECU (several per vehicle)
PDU Router	The PDU Router is a module transferring I–PDUs from one module to another module. The PDU Router can be utilized for gateway operations and for internal routing purposes.
Primary ECU	A primary ECU stores "it's own" and "reported errors" of related dep. / sec ECUs in it's event memory
Readiness	The readiness refers to the tested bits TestNotCompletedSinceLastClear (bit 4) and TestNotCompleteThisOperationCycle (bit 6) of the UDS DTC Status Byte.
"RecM Bus Protocol"–compatible module	A module capable of encoding and sending RecM Bus Protocol messages to a communication stack and/or capable of receiving and decoding RecM Bus Protocol messages from a communication stack.
Scantool	An external tool used to obtain the various OBD diagnostic status data available in a master or primary ECU
Secondary ECU(s)	see Dependent Secondary ECU
Signal	A signal in the RECM module's context is equal to a message in OSEK COM, see [2, OSEK COM]
Source	Source refers to the ECU that is the source of the diagnostic status information. The diagnostic status is generated in this ECU.
Source Section	The Source section of a RecM module is the local section in which the statuses originate. It is located in a Secondary ECU.
Upper Layer Modules (Up)	Modules above the PDU Router. This layer includes COM, Diagnostic Communication Manager (DCM) and RECM.

5 Protocol Specification

5.1 Message formats

This chapter specifies all of the message formats of the RecM Bus Protocol.

Unless otherwise specified, all messages are of the "fire and forget" type; meaning that, after a request is sent, there is no response message.

For both the Status and Management messages, the same basic RecM Bus Protocol message format is used. It consists of the Header segment, the Payload segment and the Tail segment.

Table 5.1: RecM Messsage Structure

The Header of all Status messages contains three fields: a Message Type, a Client Identifier and a Reserved field.
The Header of all Management messages contains two fields: a Message Type and a Reserved field.
The Tail of both the Status and Management messages contains a common Message Counter field.

Note: The structures used in all of the messages are defined in the Data Types section (See Section 5.7).

5.1.1 Common Message fields

5.1.1.1 Message Type (MSG_TYPE) field format

[PRS_RECM_00001] ⌈ The Message Type (MSG_TYPE) shall be of type TYP_MSTP. ⌋*(SRS_RECM_00001, SRS_RECM_00008)*

See Section 5.8.1 for the definition of the type TYP_MSTP.

[PRS_RECM_00002] ⌈ The Message Type (MSG_TYPE) shall be one of the listed messages defined of type in Table Table 5.23. ⌋*(SRS_RECM_00001, SRS_RECM_00008)*

See Section 5.8.1 for the definitions of the message types.

5.1.1.2 Client Identifier (CLIENT_ID) field format

The Client Identifier (CLIENT_ID) is a Secondary ECU identifier which is obtained from the Diagnostic Management module.

[PRS_RECM_00003] ⌈ The Client Identifier (CLIENT_ID) field shall be of type TYP_CLID. ⌋*(SRS_RECM_00001, SRS_RECM_00008)*

See Section 5.7.1 for the definition of the type TYP_CLID.

[PRS_RECM_00004] ⌈ The Client Identifier (CLIENT_ID) field shall contain the local ECU identifier obtained from the Diagnostic Management module. ⌋ *(SRS_RECM_00001, SRS_RECM_00008)*

[PRS_RECM_00005] ⌈ The Client Identifier (CLIENT_ID) field shall only be employed in messages originating in a Source role section. ⌋*(SRS_RECM_00001, SRS_RECM_00008)*

5.1.1.3 Reserved field format

[PRS_RECM_00006] ⌈ The Reserved (RESERVED) field shall consist of the STRCT_RESVD structure. ⌋*(SRS_RECM_00001, SRS_RECM_00008)*

See Section 5.7.2 for the definition of the STRCT_RESVD structure.

5.1.1.4 Message counter field format

[PRS_RECM_00009] ⌈ The Message Counter (MSG_CTR) field shall contain the message counter number. ⌋*(SRS_RECM_00008, SRS_RECM_00045)*

[PRS_RECM_00010] ⌈ The Message Counter (MSG_CTR) field shall be of the TYP_MSGCTR type. ⌋*(SRS_RECM_00008, SRS_RECM_00046)*

See Section 5.7.3 for the definition of the TYP_MSGCTR type.

5.1.2 Heartbeat Status Message format

(See Use Case UC_000: Processing Heartbeat (Keep–Alive) Status)

The Heartbeat Status Message is sent when the Heartbeat Interval Time, defined by the RecMTriggerTimeout parameter, has been exceeded; i.e. no message has been sent over the bus within this configurable time interval. See Heartbeat Interval Time parameter definition: Configuration Section 6.3.1

[PRS_RECM_00014] ⌈ The Heartbeat Status Message format (Heartbeat_Status_Array) shall be used for the transmission of the heartbeat message over the bus from a source ECU to the destination ECU. ⌋*(SRS_RECM_00008, SRS_RECM_00031, SRS_RECM_00038, SRS_RECM_00045)*

[PRS_RECM_00015] ⌈ The Heartbeat Status Message format shall apply the Heartbeat_Status_Array (HSA) I–PDU structure. The Heartbeat_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Byte 5: MSG_CTR: Message counter

⌋(*SRS_RECM_00001*, *SRS_RECM_00031*, *SRS_RECM_00038*)

[PRS_RECM_00016] ⌈ The Message Type (MSG_TYPE) for the Heartbeat Status Message shall be RECM_MSG_STAT_HTBT. ⌋(*SRS_RECM_00001*, *SRS_RECM_00031*, *SRS_RECM_00038*)

The Heartbeat_Status_Array I–PDU structure is shown in Figure 5.1 and in Table 5.2 below:

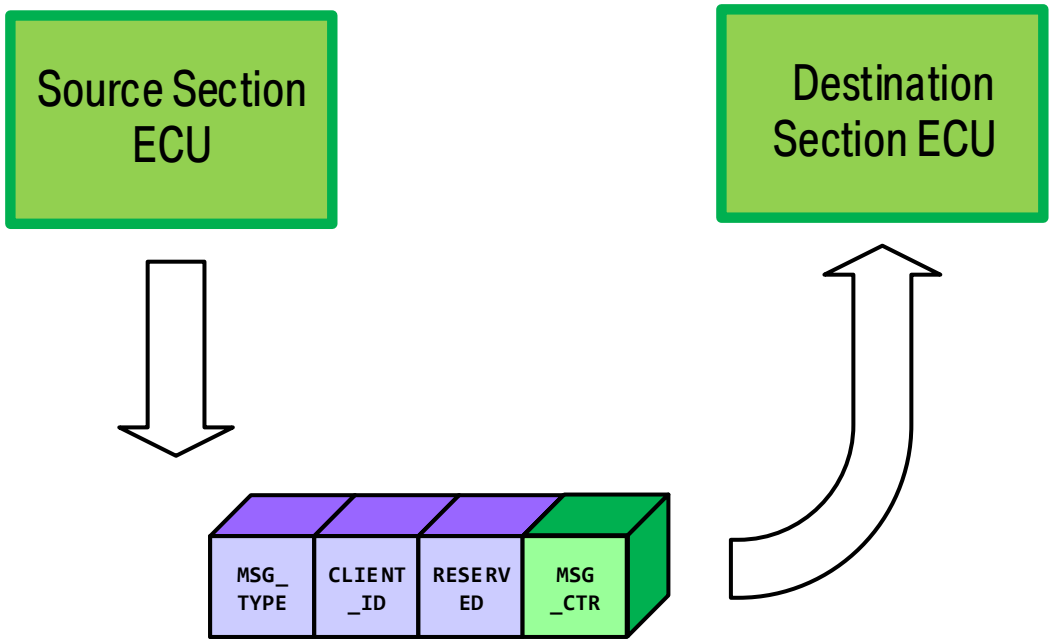


Figure 5.1: Heartbeat Status Message Structure

	Field:	Contents:	Type:	Byte #:
Header	MSG_TYPE	Message type = RECM_MSG_STAT_HTBT	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	5

Table 5.2: Heartbeat_Status_Array Structure

5.1.3 DTC Status Message format

(See Use Case UC_001: Processing DTC Status)

[PRS_RECM_00017] ⌈ The DTC Status Message format (DTC_Status_Array) shall be used for the transmission of the diagnostic DTC status data over the bus from a source ECU to the destination ECU. ⌋*(SRS_RECM_00008, SRS_RECM_00012, SRS_RECM_00023)*

[PRS_RECM_00018] ⌈ The DTC Status Message format shall apply the DTC_Status_Array (DSA) I–PDU structure. The Diagnostic_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Bytes 5–(4+(NUM_ELEM *sizeof(STATUS element))): STATUS_1 – STATUS_X
  (where X = NUM_ELEM value)
  (If NUM_ELEM = 0, then there is no STATUS field.)
  : Status Fields
- Byte 5+(NUM_ELEM *sizeof(STATUS element)): MSG_CTR: Message counter

  (where NUM_ELEM equals the number of status structures in the STATUS payload field)

⌋*(SRS_RECM_00001, SRS_RECM_00012, SRS_RECM_00023, SRS_RECM_00036, SRS_RECM_00037)*

[PRS_RECM_00019] ⌈ The Message Type (MSG_TYPE) for the DTC Status Message shall be RECM_MSG_STAT_DTC. ⌋*(SRS_RECM_00001, SRS_RECM_00012, SRS_RECM_00023)*

5.1.3.1 Diagnostic Status Information (STATUS_1 – STATUS_X) field format

This variable–length field contains a one–dimensional array of Diagnostic Status Information elements with a size equal to the NUM_ELEM value.

[PRS_RECM_00020] ⌈ Each element of the status element array shall apply structure STRCT_DTC_STAT type and contains an identifier and the associated status field. The identifier and status field data are in the format defined by the structure STRCT_DTC_STAT type. ⌋*(SRS_RECM_00001, SRS_RECM_00012, SRS_RECM_00023, SRS_RECM_00026, SRS_RECM_00039, SRS_RECM_00050)*

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the STRCT_DTC_STAT data type structure.

The DTC_Status_Array Structure (DSA) consists of the fields shown in Figure 5.2 and in Table 5.3 below:

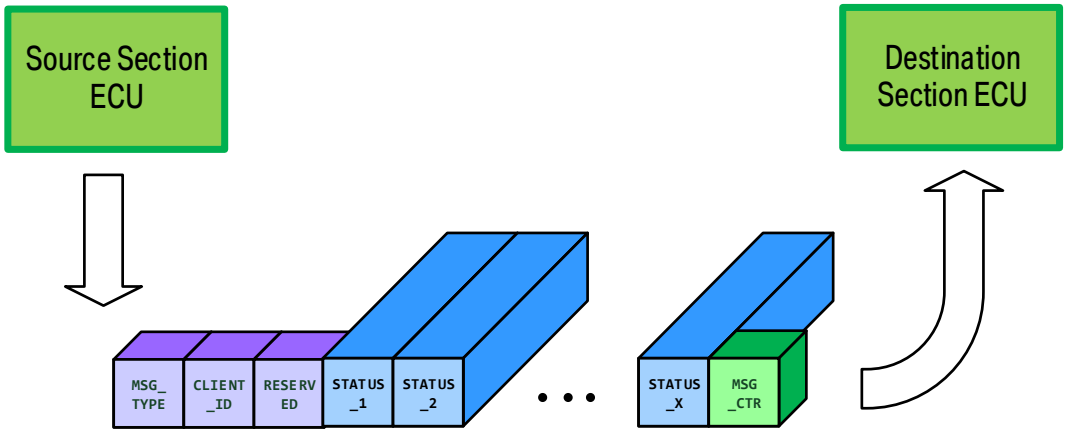


Figure 5.2: DTC Status Message Structure

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload	STATUS_1	Status Element #1	STRCT_DTC_STAT	5–(5+ (sizeof (STATUS struct))
	.		"	
	.		"	
	.		"	
	STATUS_X	Status Element #X	"	5+((NUM_ELEM-1) *sizeof(STATUS))–4+((NUM_ELEM-1) *sizeof(STATUS))+(sizeof(STATUS))
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	5+(NUM_ELEM *sizeof(STATUS))

Table 5.3: DTC_Status_Array Structure

5.1.4 Monitor Disabled Status Message format

(See Use Case UC_002: Processing Monitor Disabled Status)

[PRS_RECM_00021] ⌈ The Monitor Disabled Status Message format (Monitor_Disabled_Status_Array) shall be used for the transmission of the Monitor Disabled status data over the bus from a source ECU to the destination ECU. ⌋ *(SRS_RECM_00008, SRS_RECM_00074)*

[PRS_RECM_00022] ⌈ The Monitor Disabled Status Message format shall apply the Monitor_Disabled_Status_Array (MDSA) I–PDU structure. The Monitor_Disabled_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Bytes 5–(4+(NUM_ELEM *sizeof(STATUS element))): STATUS_1 – STATUS_X (where X = NUM_ELEM value)
  (If NUM_ELEM = 0, then there is no STATUS field.)
  : Status Fields
- Byte 5+(NUM_ELEM *sizeof(STATUS element)): MSG_CTR: Message counter

  (where NUM_ELEM equals the number of status structures in the STATUS payload field)

⌋*(SRS_RECM_00001, SRS_RECM_00036, SRS_RECM_00037, SRS_RECM_00074)*

[PRS_RECM_00023] ⌈ The Message Type (MSG_TYPE) for the DTC Status Message shall be RECM_MSG_STAT_MON_DIS. ⌋*(SRS_RECM_00001, SRS_RECM_00074)*

5.1.4.1 Monitor Disabled Status Information (STATUS_1 – STATUS_X) field format

This variable–length field contains a one–dimensional array of Diagnostic Status Information elements with a size equal to the NUM_ELEM value.

[PRS_RECM_00024] ⌈ Each element of the status element array shall apply structure STRCT_MON_DIS_STAT type and contains an identifier and the associated status field. The identifier and status field data are in the format defined by the structure STRCT_MON_DIS_STAT type. ⌋*(SRS_RECM_00001, SRS_RECM_00039, SRS_RECM_00050, SRS_RECM_00074)*

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the STRCT_MON_DIS_STAT data type structure.

The Monitor_Disabled_Status_Array Structure (MDSA) has the same field structure as in Figure 5.2 with the only difference being that the STATUS elements are of a different type.

The Monitor_Disabled_Status_Array Structure (DSA) shall consist of the following fields shown in Table 5.4 below:

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload	STATUS_1	Status Element #1	STRCT_MON_DIS _STAT	7–(6+ (sizeof (STATUS struct))
	.		"	
	.		"	
	.		"	
	STATUS_X	Status Element #X	"	7+((NUM_ELEM-1) *sizeof(STATUS))–6+((NUM_ELEM-1) *sizeof(STATUS)) +(sizeof(STATUS))
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	7+(NUM_ELEM *sizeof(STATUS))

Table 5.4: Monitor_Disabled_Status_Array Structure

5.1.5 IUMPR Fault Detected Status Message format

(See Use Case UC_003: Processing IUMPR Fault Detected Status)

[PRS_RECM_00025] ⌈ The IUMPR Fault Detected Status Message format (DTC_Status_Array) shall be used for the transmission of the IUMPR Fault Detected status data over the bus from a source ECU to the destination ECU. ⌋ *(SRS_RECM_00008, SRS_RECM_00075)*

[PRS_RECM_00026] ⌈ The IUMPR Fault Detected Status Message format shall apply the Fault_Detect_Status_Array (FDSA) I–PDU structure. The Fault_Detect_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Bytes 5–(4+(NUM_ELEM *sizeof(STATUS element))): STATUS_1 – STATUS_X (where X = NUM_ELEM value)

(If NUM_ELEM = 0, then there is no STATUS field.)
: Status Fields

- Byte 5+(NUM_ELEM *sizeof(STATUS element)): MSG_CTR: Message counter

  (where NUM_ELEM equals the number of status structures in the STATUS payload field)

⌋(*SRS_RECM_00001*, *SRS_RECM_00036*, *SRS_RECM_00037*, *SRS_RECM_00075*)

[PRS_RECM_00027] ⌈ The Message Type (MSG_TYPE) for the DTC Status Message shall be RECM_MSG_STAT_FLT_DET. ⌋(*SRS_RECM_00001*, *SRS_RECM_00075*)

5.1.5.1 Fault Detected Status Information (STATUS_1 – STATUS_X) field format

This variable–length field contains a one–dimensional array of Fault Detected Status Information elements with a size equal to the NUM_ELEM value.

[PRS_RECM_00028] ⌈ Each element of the status element array shall apply the structure STRCT_FLT_DET_STAT type and contains an identifier and the associated status field. The identifier and status field data are in the format defined by the structure STRCT_FLT_DET_STAT type. ⌋(*SRS_RECM_00001*, *SRS_RECM_00039*, *SRS_RECM_00050*, *SRS_RECM_00075*)

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the STRCT_FLT_DET_STAT data type structure.

The Fault_Detected_Status_Array Structure (FDSA) has the same field structure as in Figure 5.2 with the only difference being that the STATUS elements are of a different type.

The Fault_Detected_Status_Array Structure (FDSA) shall consist of the following fields shown in Table 5.5 below:

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload	STATUS_1	Status Element #1	STRCT_FLT_DET _STAT	7–(6+ (sizeof (STATUS struct))
	.		"	
	.		"	
	.		"	
	STATUS_X	Status Element #X	"	7+((NUM_ELEM-1) *sizeof(STATUS))–6+((NUM_ELEM-1) *sizeof(STATUS)) +(sizeof(STATUS))
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	7+(NUM_ELEM *sizeof(STATUS))

Table 5.5: IUMPR Fault_Detected_Status_Array Structure

5.1.6 DTR Data Status Message format

(See Use Case UC_004: Processing DTR Data Status)

[PRS_RECM_00093] ⌈ The DTR Data Status Message format (DTR_Data_Status_Array) shall be used for the transmission of the Diagnostic Test Report (DTR) status data over the bus from a source ECU to the destination ECU. ⌋ *(SRS_RECM_00008)*

[PRS_RECM_00094] ⌈ The DTR Data Status Message format shall apply the DTR_Data_Status_Array (DDSA) I–PDU structure. The DTR_Data_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Bytes 5–(4+(NUM_ELEM *sizeof(STATUS element))): STATUS_1 – STATUS_X (where X = NUM_ELEM value)
  (If NUM_ELEM = 0, then there is no STATUS field.)
  : Status Fields
- Byte 5+(NUM_ELEM *sizeof(STATUS element)): MSG_CTR: Message counter

(where NUM_ELEM equals the number of status structures in the STATUS payload field)

⌋(*SRS_RECM_00001*, *SRS_RECM_00036*, *SRS_RECM_00037*)

[PRS_RECM_00095] ⌈ The Message Type (MSG_TYPE) for the DTR Data Status Message shall be RECM_MSG_STAT_DTR_DATA. ⌋(*SRS_RECM_00001*)

5.1.6.1 DTR Data Status Information (STATUS_1 – STATUS_X) field format

This variable–length field contains a one–dimensional array of DTR Data Status Information elements with a size equal to the NUM_ELEM value.

[PRS_RECM_00096] ⌈ Each element of the status element array shall apply structure STRCT_DTR_DATA_STAT type and contains an identifier and the associated status field. The identifier and status field data are in the format defined by the structure STRCT_DTR_DATA_STAT type. ⌋(*SRS_RECM_00001*, *SRS_RECM_00039*, *SRS_RECM_00050*)

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the STRCT_DTR_DATA_STAT data type structure.

The DTR_Data_Status_Array Structure (DDSA) consists of the fields shown in Figure 5.3 and in Table 5.6 below:

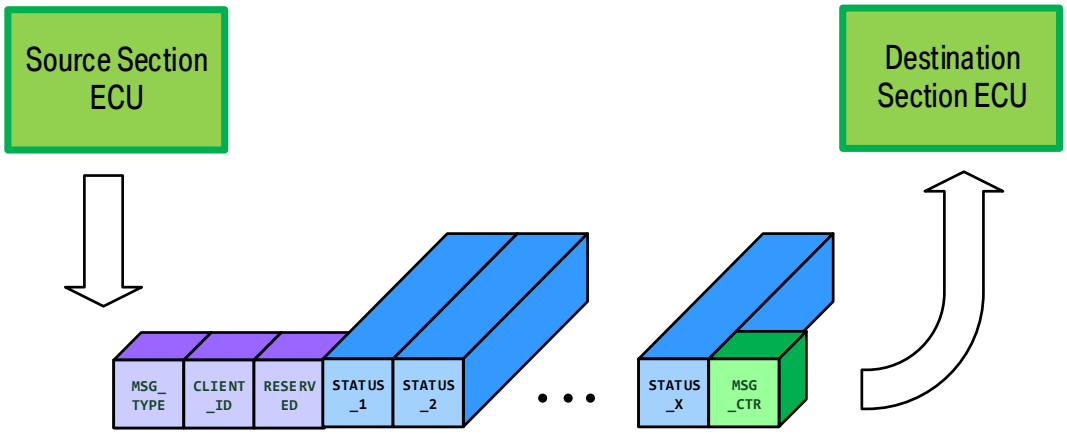


Figure 5.3: DTR Data Status Message Structure

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload	STATUS_1	Status Element #1	STRCT_DTC_STAT	5–(5+ (sizeof (STATUS struct))
	.		"	
	.		"	
	.		"	
	STATUS_X	Status Element #X	"	5+((NUM_ELEM-1) *sizeof(STATUS))–4+((NUM_ELEM-1) *sizeof(STATUS)) +(sizeof(STATUS))
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	5+(NUM_ELEM *sizeof(STATUS))

Table 5.6: DTR_Data_Status_Array Structure

5.1.7 Vehicle Information Values Status Message format

(See Use Case UC_005: Processing Vehicle Information Status)

[PRS_RECM_00029] ⌈ The Vehicle Information Values Status Message format (Vehicle_Information_Status_Array) shall be used for the transmission of the Vehicle Information Values status data over the bus from a source ECU to the destination ECU. ⌋ *(SRS_RECM_00008, SRS_RECM_00076)*

[PRS_RECM_00030] ⌈ The Vehicle Information Values Status Message format shall apply the Vehicle_Information_Status_Array (VISA) I–PDU structure. The Vehicle_Information_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Bytes 5–20: CAL_ID: Calibration Identifier
- Bytes 21–24: CVN: Calibration Verification Number
- Byte 25: MSG_CTR: Message counter

⌋(*SRS_RECM_00001*, *SRS_RECM_00036*, *SRS_RECM_00037*, *SRS_RECM_00076*)

[PRS_RECM_00031] ⌈ The Message Type (MSG_TYPE) for the DTC Status Message shall be RECM_MSG_STAT_CALID_CVN. ⌋(*SRS_RECM_00001*, *SRS_RECM_00076*)

5.1.7.1 Calibration identifier (CAL_ID) field format

[PRS_RECM_00032] ⌈ The Calibration identifier (CAL_ID) field shall apply structure STRCT_CALID type. ⌋(*SRS_RECM_00008*, *SRS_RECM_00050*, *SRS_RECM_00076*)

[PRS_RECM_00052] ⌈ Each byte of the structure STRCT_CALID shall be set to the default value of 0x00 if the Calibration Identifier is not yet known by the Source role RecM Bus Protocol-compatible module. ⌋(*SRS_RECM_00001*, *SRS_RECM_00076*)

5.1.7.2 Calibration verification number (CVN) field format

[PRS_RECM_00053] ⌈ Each byte of the structure STRCT_CVN shall be set to the default value of 0x00 if the Calibration Verification Number is not yet known by the Source role RecM Bus Protocol-compatible module. ⌋(*SRS_RECM_00001*, *SRS_RECM_00036*, *SRS_RECM_00037*, *SRS_RECM_00039*, *SRS_RECM_00076*)

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the STRCT_CALID and STRCT_CVN data type structures.

The Vehicle_Information_Status_Array Structure (DSA) consists of the fields shown in Figure 5.4 and in Table 5.7 below:

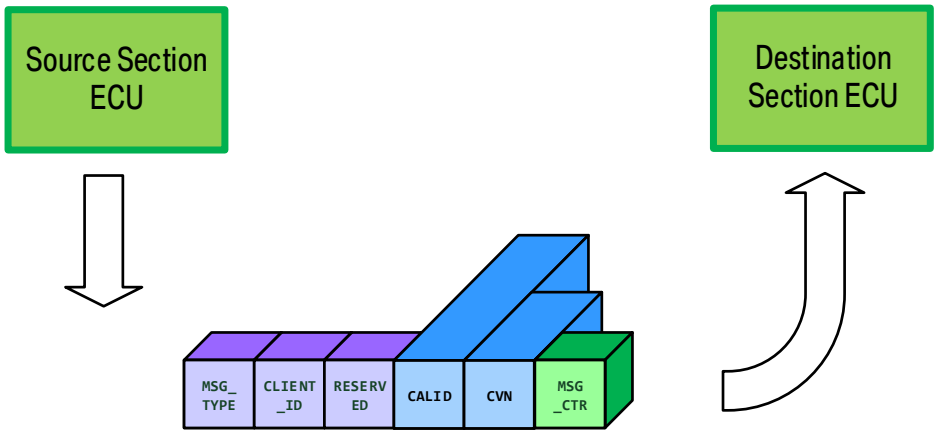


Figure 5.4: Vehicle Information Values Status Message Structure

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload	CAL_ID	Calibration Identifier	STRCT_CALID	5–20
	CVN	Calibration Verification Number	STRCT_CVN	21–24
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	25

Table 5.7: Vehicle_Information_Status_Array Structure

5.1.8 Mode $4 Return Value Status Message

(See Use Case UC_006: Processing Mode $4 Return Value Status)

[PRS_RECM_00034] ⌈ The Mode $4 Return Value Status Message format (Mode4_Return_Status_Array) shall be used for the transmission of the Mode $4 Return Value over the bus from a source ECU to the destination ECU. ⌋ *(SRS_RECM_00008)*

[PRS_RECM_00035] ⌈ The Mode $4 Return Value Status Message format shall apply the Mode4_Return_Status_Array (MRSA) I–PDU structure. The Mode4_Return_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Byte 5: MODE4_RET: Mode $4 Return Value
- Byte 6: MSG_CTR: Message counter

⌋*(SRS_RECM_00001, SRS_RECM_00036, SRS_RECM_00037)*

[PRS_RECM_00036] ⌈ The Message Type (MSG_TYPE) for the DTC Status Message shall be RECM_MSG_STAT_MODE4_RET. ⌋*(SRS_RECM_00001)*

5.1.8.1 Mode $4 Return Value (MODE4_RET) field format

[PRS_RECM_00037] ⌈ The Mode $4 Return Value (MODE4_RET) field format shall apply the structure TYP_MODE4_RET type. ⌋*(SRS_RECM_00008, SRS_RECM_00050)*

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the TYP_MODE4_RET data type.

The Mode4_Return_Status_Array (MRSA) consists of the fields shown in Figure 5.5 and in Table 5.8 below:

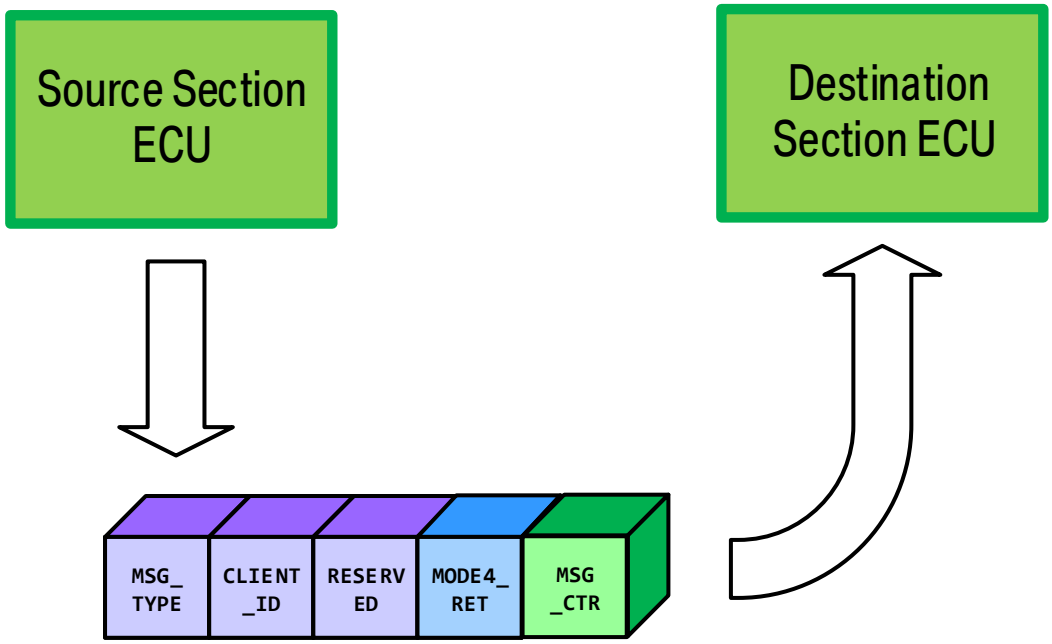


Figure 5.5: Mode $4 Return Value Status Message Structure

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload	MODE4_RET	Mode $4 Return value	TYP_MODE4_RET	5
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	6

Table 5.8: Mode $4 Return Value Status Array Structure

5.1.9 Version Numbers Status Message format

(See Use Case UC_007: Processing Versions Status)

The Master List and RecM Bus Protocol-compatible module Version numbers can be transmitted using the Version Numbers Status Message format.

[PRS_RECM_00038] ⌈ The Version Numbers Status Message format (Version_Numbers_Status_Array) shall be used for the transmission of various version numbers over the bus from a source ECU to the destination ECU. ⌋ *(SRS_RECM_00008, SRS_RECM_00076)*

[PRS_RECM_00039] ⌈ The Version Numbers Status Message format shall apply the Version_Numbers_Status_Array (VNSA) I–PDU structure. The Version_Numbers_Status_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Byte 1: CLIENT_ID: Client identifier
- Bytes 2–4: RESERVED: Reserved for future use
- Bytes 5–6: NUM_ELEM: Number of elements in Versions field
- Bytes 7–(6+(NUM_ELEM *sizeof(VERSION element))): VERSION_1 – VERSION_X (where X = NUM_ELEM value)
  (If NUM_ELEM = 0, then there is no VERSION field.)
  : Version Fields
- Byte 7+(NUM_ELEM *sizeof(VERSION element)): MSG_CTR: Message counter

⌋ *(SRS_RECM_00001, SRS_RECM_00036, SRS_RECM_00037, SRS_RECM_00076)*

[PRS_RECM_00040] ⌈ The Message Type (MSG_TYPE) for the DTC Status Message shall be RECM_MSG_STAT_VERS. ⌋ *(SRS_RECM_00001, SRS_RECM_00076)*

5.1.9.1 Version Numbers (VERSION) field format

This variable–length field contains a one–dimensional array of Version Numbers Status Information elements with a size equal to the NUM_ELEM value.

[PRS_RECM_00041] ⌈ Each element of the status element array shall apply structure STRCT_VERS_STAT type and contains an version identifier and the associated version number field. The identifier and version number field data are in the format defined by the structure STRCT_VERS_STAT type. ⌋ *(SRS_RECM_00008, SRS_RECM_00039, SRS_RECM_00050, SRS_RECM_00076)*

See Section 5.7.6 Diagnostic_Status_Information (DSI) structures for the definition of the STRCT_VERS_STAT data type structure.

The Version_Numbers_Status_Array Structure (VNSA) consists of the fields shown in Figure 5.6 and in Table 5.9 below:

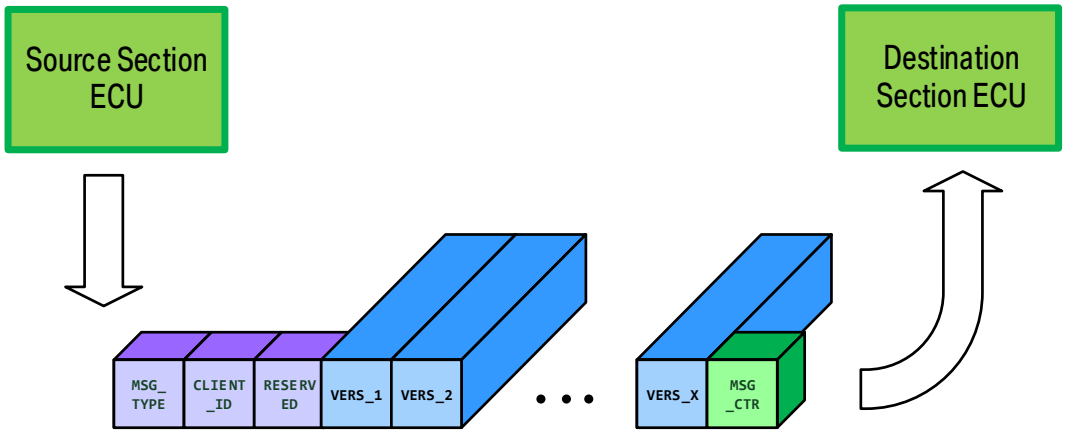


Figure 5.6: Version Numbers Status Message Structure

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	CLIENT_ID	Client ECU identifier	TYP_CLID	1
	RESERVED	Reserved	STRCT_RESVD	2-4
Payload				(6+ (sizeof (VERSIONS struct))
	.		"	
	.		"	
	.		"	
	VERSION_X	Versions Element #X	"	7+((NUM_ELEM-1) *sizeof(VERS.))– 6+((NUM_ELEM-1) *sizeof(VERS.)) +(sizeof(VERS.))
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	7+(NUM_ELEM *sizeof(VERS.))

Table 5.9: Version_Numbers_Status_Array Structure

5.1.10 Mode $04 Notification Management Message format

(See Use Case UC_008: Processing Mode $4 Notification Management Command)

[PRS_RECM_00042] ⌈ The Mode $04 Notification Management Message format (Mode4_Not_Management_Array) shall be used for the transmission of manage-

ment command messages over the bus from a destination ECU to a source ECU. ⌋ *(SRS_RECM_00008, SRS_RECM_00026, SRS_RECM_00044)*

[PRS_RECM_00043] ⌈ The Mode $04 Notification Management format shall apply the Mode4_Not_Management_Array (MNMA) structure. ⌋*(SRS_RECM_00001, SRS_RECM_00044, SRS_RECM_00052)*

[PRS_RECM_00044] ⌈ The Mode4_Not_Management_Array structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Bytes 1-3: RESERVED: Reserved for future use
- Byte 4: MSG_CNT: Message counter

⌋*(SRS_RECM_00001, SRS_RECM_00036, SRS_RECM_00037, SRS_RECM_00044, SRS_RECM_00052)*

[PRS_RECM_00045] ⌈ The Message Type (MSG_TYPE) for the Mode $04 Notification Management Message shall be RECM_MSG_MGMT_MODE4_NOT. ⌋ *(SRS_RECM_00001, SRS_RECM_00044)*

The structure of the Mode4__Not_Management_Array I–PDU is shown in the Table 5.10 below:

The Mode4_Not_Management_Command_Array I–PDU structure is shown in Figure 5.7 and Table 5.10 below:

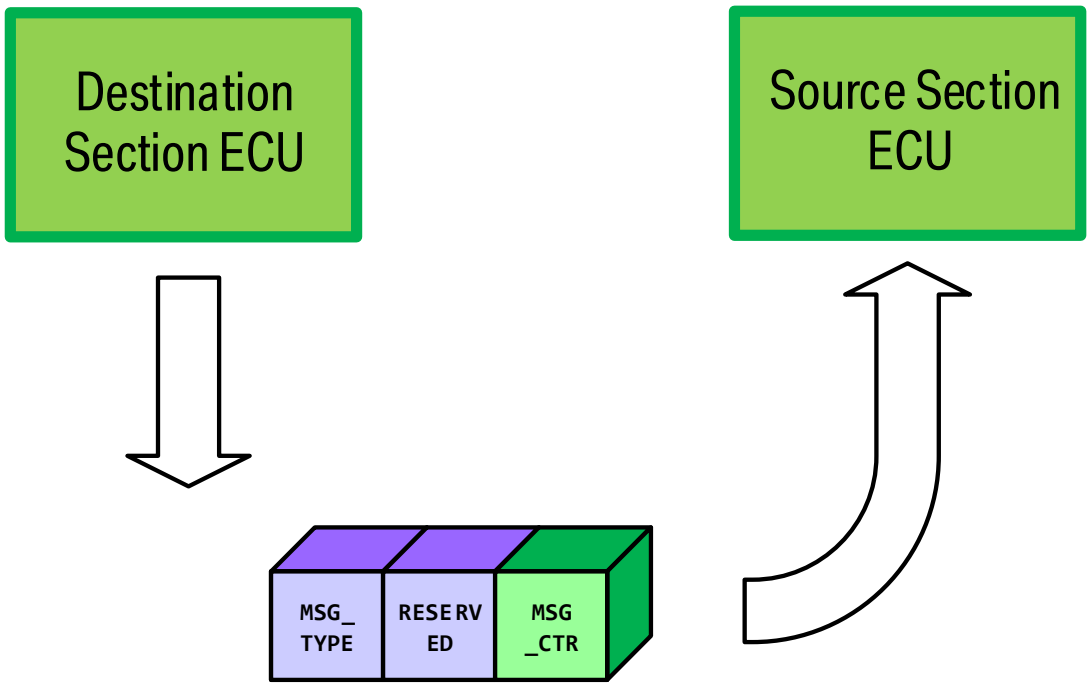


Figure 5.7: Mode $4 Notification Management Message Structure

	Field:	Contents:	Type:	Byte(s) #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	RESERVED	Reserved	STRCT_RESVD	1–3
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	4

Table 5.10: Mode4_Management_Command_Array Structure

5.1.11 Diagnostic Synchronize–All Management Message format

(See Use Case UC_009: Processing Diagnostic Synchronize–All Management Command)

[PRS_RECM_00046] ⌈ The Diagnostic Synchronize–All Management Message format (Diagnostic_Synchronize_All_Management_Array) shall be used for the transmission of the diagnostic synchronize 'all' diagnostics command over the bus from the destination ECU to a source ECU. ⌋*(SRS_RECM_00008, SRS_RECM_00019, SRS_RECM_00026, SRS_RECM_00049)*

[PRS_RECM_00047] ⌈ The Diagnostic Synchronize–All Management Message format shall apply the Diagnostic_Synchronize_All_Management_Array (DSAMA) structure. ⌋ *(SRS_RECM_00001, SRS_RECM_00019, SRS_RECM_00026, SRS_RECM_00052)*

[PRS_RECM_00048] ⌈ The Diagnostic_Synchronize_All_Management_Array Structure shall consist of the following fields in the following order:

- Byte 0: MSG_TYPE: Message type
- Bytes 1-3: RESERVED: Reserved for future use
- Byte 4: MSG_CNT: Message counter

⌋*(SRS_RECM_00001, SRS_RECM_00019, SRS_RECM_00026, SRS_RECM_00036, SRS_RECM_00037, SRS_RECM_00051)*

[PRS_RECM_00049] ⌈ The Message Type (MSG_TYPE) for the Diagnostic Synchronize–All Management Message shall be RECM_MSG_MGMT_SYNC_ALL. ⌋ *(SRS_RECM_00001, SRS_RECM_00019, SRS_RECM_00051)*

The Diagnostic_Synchronize_All_Management_Array I–PDU structure is shown in Figure 5.8 and Table 5.11.

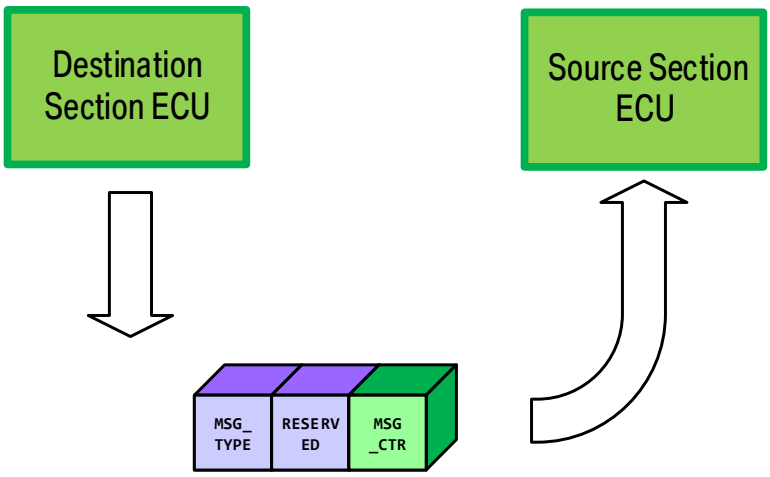


Figure 5.8: Diagnostic Synchronize–All Management Message Structure

	Field:	Contents:	Type:	Byte #:
Header	MSG_TYPE	Message type	TYP_MSTP	0
	RESERVED	Reserved	STRCT_RESVD	1–3
Tail	MSG_CTR	Message Counter	TYP_MSGCTR	4

Table 5.11: Diagnostic_Synchronize–All_Management_Array I–PDU Structure

[PRS_RECM_00081] ⌈ Upon reception of a Diagnostic Synchronize–All Management Message, the Source Role ECU shall send the following list of messages to the Destination Role ECU from which the message was received within the time interval specified by the Sync–All Message Return Time parameter (SyncAllReturnTime).

1. Version numbers status message (RECM_MSG_STAT_VERS)
2. Cal–ID & CVN Values status message (RECM_MSG_STAT_CALID_CVN)
3. DTC Status message (RECM_MSG_STAT_DTC)
4. Monitor Disabled Status message (RECM_MSG_STAT_MON_DIS)
5. IUMPR Fault Detected Status message (RECM_MSG_STAT_FLT_DET)

⌋*(SRS_RECM_00008, SRS_RECM_00026, SRS_RECM_00027)*

[PRS_RECM_00082] ⌈ If the Destination Role ECU has not received every one of the following messages from the Source Role ECU within the time interval specified by the Sync–All Message Return Time parameter (SyncAllReturnTime), the Destination Role ECU shall send the Diagnostic Synchronize–All Management Message again.

1. Version numbers status message (RECM_MSG_STAT_VERS)
2. Cal–ID & CVN Values status message (RECM_MSG_STAT_CALID_CVN)

3. DTC Status message (RECM_MSG_STAT_DTC)
4. Monitor Disabled Status message (RECM_MSG_STAT_MON_DIS)
5. IUMPR Fault Detected Status message (RECM_MSG_STAT_FLT_DET)

⌋(*SRS_RECM_00008*, *SRS_RECM_00026*, *SRS_RECM_00038*)

[PRS_RECM_00083] ⌈ If there are OBD–relevant Monitor Disabled diagnostics present in a Source Role ECU, then no Monitor Disabled Status message shall be sent by the Source Role ECU after the reception of a Diagnostic Synchronize–All Management Message and no Monitor Disabled Status message shall be expected by the Destination Role ECU.

If there are OBD–relevant IUMPR Fault Detected diagnostics present in a Source Role ECU, then no IUMPR Fault Detected Status message shall be sent by the Source Role ECU after the reception of a Diagnostic Synchronize–All Management Message and no IUMPR Fault Detected Status message shall be expected by the Destination Role ECU. ⌋(*SRS_RECM_00008*, *SRS_RECM_00026*, *SRS_RECM_00027*)

5.2 Communication matrix

The communication matrix between the OBD–relevant ECUs is shown in Table 5.12 below:

Scenario:	Type:	Sender:	Receiver:	Message Type:
D–OBD	Status	Secondary	Primary	RECM_MSG_STAT_HTBT
				RECM_MSG_STAT_DTC
				RECM_MSG_STAT_MON_DIS
				RECM_MSG_STAT_FLT_DET
				RECM_MSG_STAT_DTR_DATA
				RECM_MSG_STAT_CALID_CVN
				RECM_MSG_STAT_MODE4_RET
				RECM_MSG_STAT_VERS
	Mgmt.	Primary	Secondary	RECM_MSG_MGMT_MODE4_NOT
				RECM_MSG_MGMT_SYNC_ALL
Thin-Client	Status	Ext. Dep. Sec.	Secondary	RECM_MSG_STAT_HTBT
				RECM_MSG_STAT_DTC
				RECM_MSG_STAT_MON_DIS
				RECM_MSG_STAT_FLT_DET
				RECM_MSG_STAT_CALID_CVN
				RECM_MSG_STAT_VERS
	Mgmt.	Secondary	Ext. Dep. Sec.	RECM_MSG_MGMT_SYNC_ALL

Table 5.12: Communication Matrix

5.3 Communication Specification

5.3.1 Fire & Forget Communication

Messages that have requests without a corresponding response message are called "fire & forget".

[PRS_RECM_00051] ⌈ Unless otherwise specifically noted, all RecM Bus Protocol messages shall be of the "fire & forget" type. The implementation is basically the same as for Request/Response communication with the following differences:

- There is no response message (i.e., no handshaking).

⌋(*SRS_RECM_00001*, *SRS_RECM_00008*)

"Fire & forget" messages do not return an error.

Error handling and return codes are implemented by an application and/or module in an upper layer above the RecM Bus Protocol compatible module when needed.

5.4 Endian Specification

[PRS_RECM_00080] ⌈ All message fields shall be encoded in big endian format. ⌋*(SRS_RECM_00001)*

5.5 Message Fields Handling

5.5.1 Message counter field

[PRS_RECM_00011] ⌈ The Message Counter shall be incremented after every Diagnostic Status message sent by the RecM Bus Protocol-compatible module. ⌋*(SRS_RECM_00008, SRS_RECM_00026, SRS_RECM_00047, SRS_RECM_00068)*

[PRS_RECM_00012] ⌈ After initialization of the RecM Bus Protocol-compatible module, the Message Counter (MSG_CNT) shall be set to '0'. ⌋*(SRS_RECM_00008, SRS_RECM_00026, SRS_RECM_00045)*

[PRS_RECM_00013] ⌈ If the Message Counter reaches 255, the counter shall wrap around and start with the value '0' at the next Diagnostic Status message to be transmitted. ⌋*(SRS_RECM_00008, SRS_RECM_00026, SRS_RECM_00047)*

5.6 Error Handling

The error handling is done in the communication layers below the RecM Bus Protocol compatible module.

5.7 Data Types

5.7.1 Type Client Identifier (TYP_CLID)

The Client Identifier type is used in the Client identifier (CLIENT_ID) field which indicates the number of the sending Source ECU.

[PRS_RECM_00056] ⌈ The size of the Client Identifier (TYP_CLID) field shall be 1 byte unsigned integer (uint8). ⌋*(SRS_RECM_00008, SRS_RECM_00026)*

5.7.2 Type Reserved structure (STRCT_RESVD)

[PRS_RECM_00058] ⌈ Type STRCT_RESVD shall consist of a one–dimensional array of data type unsigned integer (UINT). ⌋*(SRS_RECM_00001)*

[PRS_RECM_00059] ⌈ The size of the STRCT_RESVD array shall be equal to 3 bytes. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00060] ⌈ Each element of the Type STRCT_RESERVED array shall be set to the default value of 0xAA. ⌋*(SRS_RECM_00001, SRS_RECM_00007)*

5.7.3 Type Message Counter (TYP_MSGCTR)

The Message Counter is incremented with every Diagnostic Status message sent by the RecM Bus Protocol-compatible module. With the Message Counter, lost messages can be recognized.

[PRS_RECM_00061] ⌈ The size of the Message counter (MSG_CNT) field shall be 8-bit (an unsigned 0–255 integer). ⌋*(SRS_RECM_00008, SRS_RECM_00026)*

5.7.4 Type DTC Index Identifier (TYP_DTC_INDEX_ID)

The DTC Index Identifier type is used in many of the Diagnostic_Status_Information (DSI) structures as an identifier for the respective status.

[PRS_RECM_00057] ⌈ The size of the DTC Index Identifier (TYP_DTC_INDEX_ID) field shall be a 16–bit unsigned integer (uint16). ⌋*(SRS_RECM_00008, SRS_RECM_00026)*

5.7.5 Type Mode $4 Return Value (TYP_MODE4_RET)

The Mode $4 Return Value type is used in the Mode $4 Return Value (MODE4_RET) field which is used to store the return value resulting from a Mode $4 Notification management message.

[PRS_RECM_00062] ⌈ The size of the Mode $4 Return Value (TYP_MODE4_RET) field shall be 1 byte unsigned integer (uint8). ⌋*(SRS_RECM_00008, SRS_RECM_00026)*

5.7.6 Diagnostic_Status_Information (DSI) structures

The various Diagnostic_Status_Information (DSI) structures are shown in the following table:

DSI Structure Type:	Status Identifier:	Status Field:
DTC Status (STRCT_DTC_STAT)	TYP_DTC_INDEX_ID DTC Index Identifier (Index from Master Index List)	Dem_UdsStatusByte (8 bits) See Note 2 below
Monitor Disabled Status (STRCT_MON_DIS_STAT) IUMPR Fault Detected Status (STRCT_FLT_DET_STAT)	TYP_DTC_INDEX_ID DTC Index Identifier	uint8 (1 bit used)
	TYP_DTC_INDEX_ID DTC Index Identifier	uint8 (3 bits used)
DTR Data Status (STRCT_DTR_DATA_STAT)	TYP_DTC_INDEX_ID DTC Index Identifier	–TestResult (4 x uint8) –LowerLimit (4 x uint8) –UpperLimit (4 x uint8) –DtrCtrlType (1 x uint8)
Vehicle Info Values (STRCT_CALID_CVN_STAT)	none	–CAL–ID (16 x uint8) –CVN (4 x uint8)
Versions (STRCT_LST_VER_STAT)	Version Type Identifier (uint16)	Version number uint16

Notes:

1. Each DSI structure consists of a status identifier and a status field.
2. See the Types Definition section of the [3, Specification of Diagnostic Event Manager] (AUTOSAR_SWS_DiagnosticEventManager) for the definition of the Dem_UdsStatusByte Type.

5.7.6.1 Type DTC Status structure (STRCT_DTC_STAT)

[PRS_RECM_00063] ⌈ The DTC Status type (STRCT_DTC_STAT) shall apply Table 5.13 below: ⌋*(SRS_RECM_00001)*

Byte:	Type:	Field:
0 – 1	Identifier	DTC Index Identifier (TYP_DTC_INDEX_ID) (Index from Master Index List)
2	Status	Dem_UdsStatusByte (defined in SWS DEM spec., [3, Specification of Diagnostic Event Manager]) (See Note 2 above)

Table 5.13: DTC Status type (STRCT_DTC_STAT)

5.7.6.2 Type Monitor Disabled Status structure (STRCT_MON_DIS_STAT)

[PRS_RECM_00105] ⌈ The Monitor Disabled Status type (STRCT_MON_DIS_STAT) shall apply Table 5.14 below: ⌋*(SRS_RECM_00001)*

Byte:	Type:	Field:
0 – 1	Identifier	DTC Index Identifier (TYP_DTC_INDEX_ID) (Index from Master List)
2	Status	TYP_MON_DIS_STAT

Table 5.14: DTC Status type (STRCT_DTC_STAT)

5.7.6.2.1 Type Monitor Disabled Status (TYP_MON_DIS_STAT)

The Monitor Disabled Status type is used in the Monitor Disabled Status structure (STRCT_MON_DIS_STAT) and represents the current Monitor Disabled status.

[PRS_RECM_00077] ⌈ The size of the Monitor Disabled Status (TYP_MON_DIS_STAT) type shall be a 1 byte unsigned integer (uint8). ⌋ *(SRS_RECM_00008, SRS_RECM_00026)*

[PRS_RECM_00078] ⌈ The bit encoding for the various status bits in the Monitor Disabled Status (TYP_MON_DIS_STAT) type shall be defined as below:

- Bit 0 DIS: Monitor disabled status (0=not disabled,1=disabled)
- Bits 1–7 RESERVED: set to 0x00

⌋*(SRS_RECM_00001)*

TableTable 5.15 illustrates the bit placement in the Monitor Disabled Status byte.

Monitor Disabled Status (TYP_MON_DIS_STAT)								
Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Short Name:	RESERVED							DIS

Table 5.15: Monitor Disabled Status Encoding

[PRS_RECM_00101] ⌈ If the DIS bit (bit '0') is not set, it shall be interpreted as meaning "not disabled". If the DIS bit is set, it shall be interpreted as meaning "disabled". ⌋ *(SRS_RECM_00001)*

5.7.6.3 Type IUMPR Fault Detected Status structure (STRCT_FLT_DET_STAT)

[PRS_RECM_00065] ⌈ The IUMPR Fault Detect Status type (STRCT_FLT_DET_STAT) shall apply Table 5.16 below: ⌋*(SRS_RECM_00001)*

Byte:	Type:	Field:

0 – 1	Identifier	DTC Index Identifier (TYP_DTC_INDEX_ID) (Index from Master Index List)
2	Status	TYP_FLT_DET_STAT

Table 5.16: IUMPR Fault Detect Status type (STRCT_FLT_DET_STAT)

5.7.6.3.1 Type Fault Detected Status (TYP_FLT_DET_STAT)

The Fault Detected Status type is used in the IUMPR Fault Detected Status structure (STRCT_FLT_DET_STAT) and represents the current Fault Detected status.

[PRS_RECM_00066] ⌈ The size of the Fault Detected Status (TYP_FLT_DET_STAT) type shall be a 1 byte unsigned integer (uint8). ⌋*(SRS_RECM_00008, SRS_RECM_00026)*

[PRS_RECM_00067] ⌈ The bit encoding for the various status bits in the Fault Detected Status (TYP_FLT_DET_STAT) type shall be defined as below:

- Bit 0 NUM: Numerator reported status (0=not reported,1=reported)
- Bit 1 DEN: Denominator reported status (0=not reported,1=reported)
- Bit 2 LCKD: Locked status (0=not locked,1=locked)
- Bits 3–7 RESERVED: set to 0x00

⌋*(SRS_RECM_00001)*

Table Table 5.17 illustrates the bit placement in the Fault Detected Status byte.

Fault Detected Status (TYP_FLT_DET_STAT)								
Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Short Name:	RESERVED					LCKD	DEN	NUM

Table 5.17: Fault Detected Status Encoding

[PRS_RECM_00102] ⌈ If the NUM bit (bit '0') is not set, it shall be interpreted as meaning "not reported". If the NUM bit is set, it shall be interpreted as meaning "reported". ⌋*(SRS_RECM_00001)*

[PRS_RECM_00103] ⌈ If the DIS bit (bit '0') is not set, it shall be interpreted as meaning "not reported". If the DIS bit is set, it shall be interpreted as meaning "reported". ⌋ *(SRS_RECM_00001)*

[PRS_RECM_00104] ⌈ If the LCKD bit (bit '2') is not set, it shall be interpreted as meaning "not locked". If the LCKD bit is set, it shall be interpreted as meaning "locked". ⌋*(SRS_RECM_00001)*

5.7.6.4 Type DTR Monitoring Data Status structure (STRCT_DTR_DATA_STAT)

[PRS_RECM_00084] ⌈ The DTR Data Status type (STRCT_DTR_DATA_STAT) shall apply Table 5.18 below: ⌋*(SRS_RECM_00001)*

Byte:	Type:	Field:
0 – 1	Identifier	DTC Index Identifier (TYP_DTC_INDEX_ID) (Index from Master Index List)
2 – 5	Status	STRCT_TEST_RESULT
6 – 9	Status	STRCT_LOWER_LIMIT
10 – 13	Status	STRCT_UPPER_LIMIT
14	Status	TYP_DTR_CTRL_TYPE

Table 5.18: DTR Data Status type (STRCT_DTR_DATA_STAT)

5.7.6.4.1 DTR Data Status Data Types

5.7.6.4.2 Type Test Result Value structure (STRCT_TEST_RESULT)

[PRS_RECM_00085] ⌈ Type STRCT_TEST_RESULT shall consist of a one–dimensional array of data type unsigned integer (UINT) containing a Test Result value. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00086] ⌈ The size of the Type STRCT_TEST_RESULT array shall be equal to 4 bytes. ⌋*(SRS_RECM_00001)*

5.7.6.4.3 Type Lower Limit Value structure (STRCT_LOWER_LIMIT)

[PRS_RECM_00087] ⌈ Type STRCT_LOWER_LIMIT shall consist of a one–dimensional array of data type unsigned integer (UINT) containing a Lower Limit value. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00088] ⌈ The size of the Type STRCT_LOWER_LIMIT array shall be equal to 4 bytes. ⌋*(SRS_RECM_00001)*

5.7.6.4.3.1 Type Upper Limit Value structure (STRCT_UPPER_LIMIT)

[PRS_RECM_00089] ⌈ Type STRCT_UPPER_LIMIT shall consist of a one–dimensional array of data type unsigned integer (UINT) containing a Upper Limit value. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00090] ⌈ The size of the Type STRCT_UPPER_LIMIT array shall be equal to 4 bytes. ⌋*(SRS_RECM_00001)*

5.7.6.4.3.2 Type DTR Control Type Value (TYP_DTR_CTRL_TYPE)

[PRS_RECM_00091] ⌈ The size of the TYP_DTR_CTRL_TYPE type shall be a 1 byte unsigned integer (uint8). ⌋*(SRS_RECM_00001)*

[PRS_RECM_00092] ⌈ The Type TYP_DTR_CTRL_TYPE shall contain one of the range of values in the following table:

DTR Control Type:	Value:	Description:
DEM_DTR_CTL_NORMAL	0x00	Values are reported and regarded as valid test result.
DEM_DTR_CTL_NO_MAX	0x01	Values are reported, but maximum limit is not available (not valid); upper limit value is ignored.
DEM_DTR_CTL_NO_MIN	0x02	Values are reported, but minimum limit is not available (not valid); lower limit value is ignored.
DEM_DTR_CTL_RESET	0x03	Values are all ignored. External representation will be all zeros as initialized (e.g. after fault clear)
DEM_DTR_CTL_INVISIBLE	0x04	Values are all ignored. This DTR is treated for the external view (tester) as if not integrated.

Table 5.19: DTR Control Type

⌋*(SRS_RECM_00001)*

5.7.6.5 Vehicle Information (CAL–ID/CVN) Types

5.7.6.5.1 Type CAL–ID/CVN Values structure (STRCT_CALID_CVN_STAT)

[PRS_RECM_00068] ⌈ The CAL–ID/CVN Values Status type (STRCT_CALID_CVN_STAT) shall apply Table 5.20 below: ⌋*(SRS_RECM_00001)*

Byte:	Type:	Field:
0 – 15	Status	STRCT_CALID
16 – 19	Status	STRCT_CVN

Table 5.20: CAL–ID/CVN Values Status type (STRCT_CALID_CVN_STAT)

5.7.6.5.2 CAL–ID/CVN Values Data Types

5.7.6.5.2.1 Type Calibration–Identifier (STRCT_CALID)

[PRS_RECM_00069] ⌈ Type STRCT_CALID shall consist of a one–dimensional array of data type unsigned integer (UINT) containing a Calibration Identification number. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00070] ⌈ The size of the Type STRCT_CALID array shall be equal to 16 bytes. ⌋*(SRS_RECM_00001)*

5.7.6.5.2.2 Type Calibration Verification Number (STRCT_CVN)

[PRS_RECM_00071] ⌈ Type STRCT_CVN shall consist of a one–dimensional array of data type unsigned integer (UINT) containing a Calibration Verification Number. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00072] ⌈ The size of the Type STRCT_CVN array shall be equal to 4 bytes. ⌋*(SRS_RECM_00001)*

5.7.6.6 Type Version Numbers Status structure (STRCT_VERS_STAT)

[PRS_RECM_00073] ⌈ The Version Numbers Status type (STRCT_VERS_STAT) shall apply Table 5.21 below: ⌋*(SRS_RECM_00001)*

Byte:	Type:	Field:
0 – 1	Identifier	Version Number type (TYP_VERSION_TYPE)
2 – 3	Status	Version Number (STRCT_VERS_NUM)

Table 5.21: Version Numbers Status type (STRCT_VERS_STAT)

5.7.6.6.1 Type Version Type (TYP_VERSION_TYPE)

The Version Number Types (TYP_VERSION_TYPE) used in the Type Version Numbers Status structure (STRCT_VERS_STAT) indicates the type of Version Number being sent.

[PRS_RECM_00098] ⌈ The size of the Version Number Types (TYP_VERSION_TYPE) field shall be 1 byte (uint8). ⌋*(SRS_RECM_00001)*

[PRS_RECM_00076] ⌈ The Version Number Types (TYP_VERSION_TYPE) used in the Type Version Numbers Status structure (STRCT_VERS_STAT) shall be defined as in TableTable 5.22 below: ⌋*(SRS_RECM_00001)*

Version Type (TYP_VERSION_TYPE)		
Short name:	**Version Type:**	**Value:**
TYP_VERS_MASTER_LIST	Master List Version	0x0
TYP_VERS_RECM_MODULE	RecM Bus Protocol-compatible module version	0x1

Table 5.22: Version Number Types definition

5.7.6.6.2 Type Version Number (STRCT_VERS_NUM)

[PRS_RECM_00099] ⌈ Type STRCT_VERS_NUM shall consist of a one–dimensional array of data type unsigned integer (UINT) containing a Version Number. ⌋*(SRS_RECM_00001)*

[PRS_RECM_00100] ⌈ The size of the Type STRCT_VERS_NUM array shall be equal to 2 bytes. ⌋*(SRS_RECM_00001)*

5.8 Type Definitions

5.8.1 Type Message Type (TYP_MSTP)

The Message Type type is used in the Message Type field which indicates the type of message being sent.

[PRS_RECM_00074] ⌈ The size of the Message Type (TYP_MSTP) field is 1 byte (uint8). ⌋*(SRS_RECM_00008, SRS_RECM_00026)*

[PRS_RECM_00075] ⌈ The Message Types (MSG_TYPE) used for the various messages shall be defined as in Table Table 5.23 below: ⌋*(SRS_RECM_00001)*

Message Type (TYP_MSTP)		
Short name:	**Message Type:**	**Value:**
RECM_MSG_STAT_HTBT	Heartbeat Status message	0x0
RECM_MSG_STAT_DTC	DTC Status message	0x1
RECM_MSG_STAT_MON_DIS	Monitor Disabled Status message	0x2
RECM_MSG_STAT_FLT_DET	IUMPR Fault Detect Status message	0x3
RECM_MSG_STAT_DTR_DATA	DTR Data status message	0x4
RECM_MSG_STAT_CALID_CVN	Cal–ID & CVN Values status message	0x5
RECM_MSG_STAT_MODE4_RET	Mode $4 return value status message	0x6
RECM_MSG_STAT_VERS	Version numbers status message	0x7
RECM_MSG_MGMT_MODE4_NOT	Mode $4 command management message	0x8
RECM_MSG_MGMT_SYNC_ALL	Synchronize All Diagnostics mgmt. message	0x9

Table 5.23: Message Type Parameters definition

5.9 Sequence diagrams

5.9.1 Process Synchronization Commands after Initialization

5.9.1.1 D–OBD Scenario

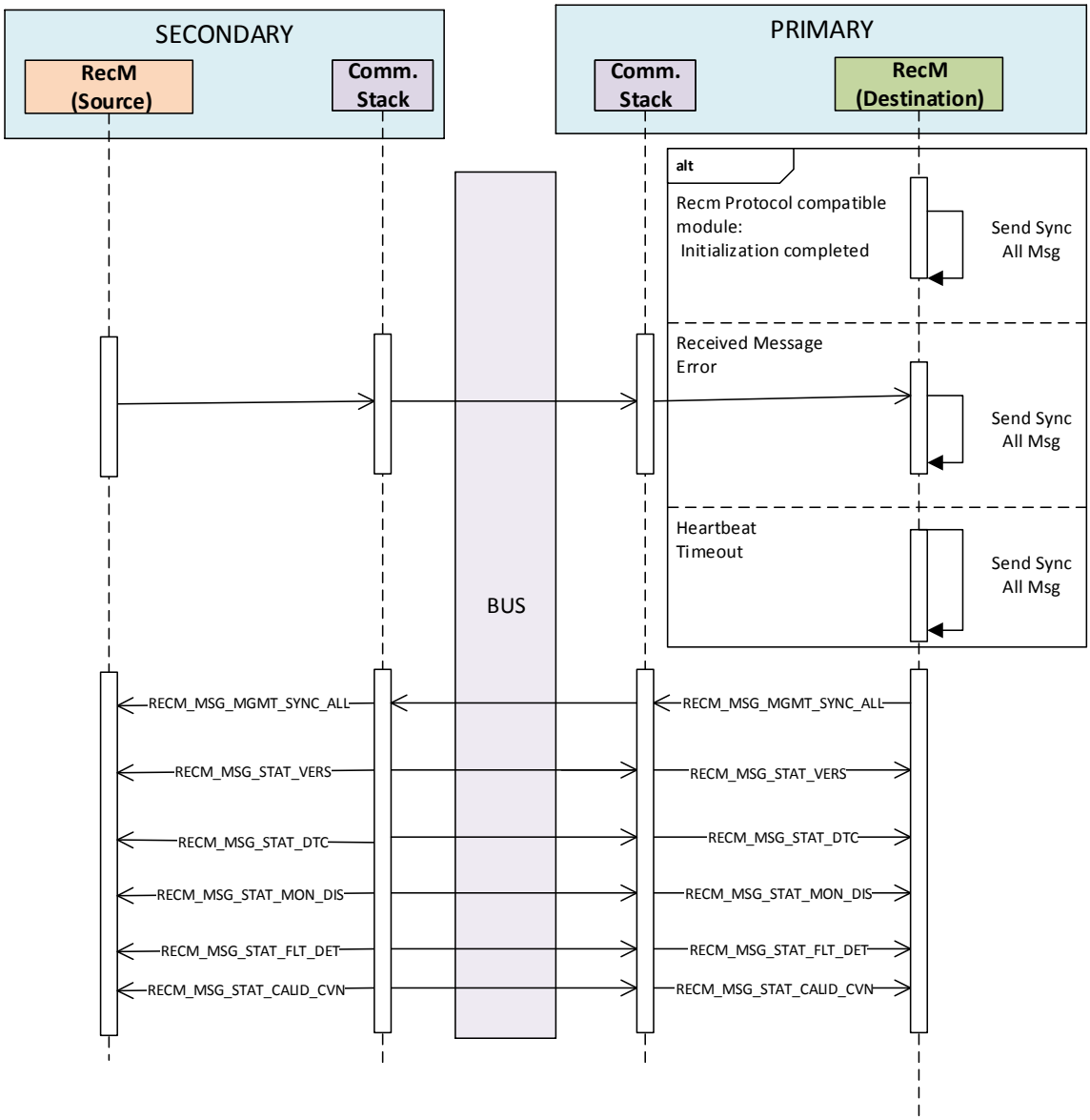


Figure 5.9: Sequence diagram of Process Synchronize All Commands (D–OBD Scenario)

5.9.1.2 Thin–Client Scenario

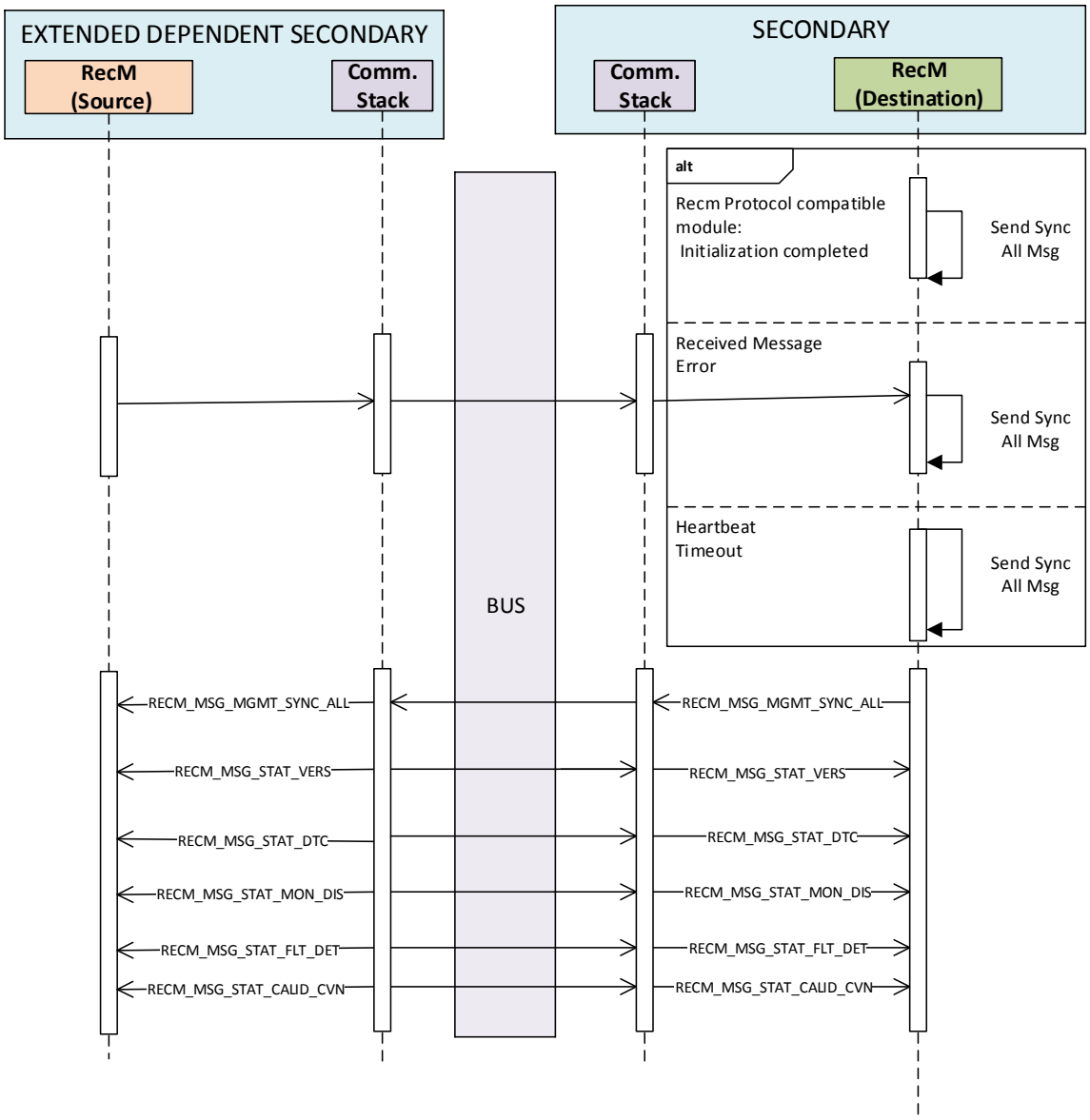


Figure 5.10: Sequence diagram of Process Synchronize All Commands (Thin–Client Scenario)

5.9.2 Process Mode $04 Notification Commands

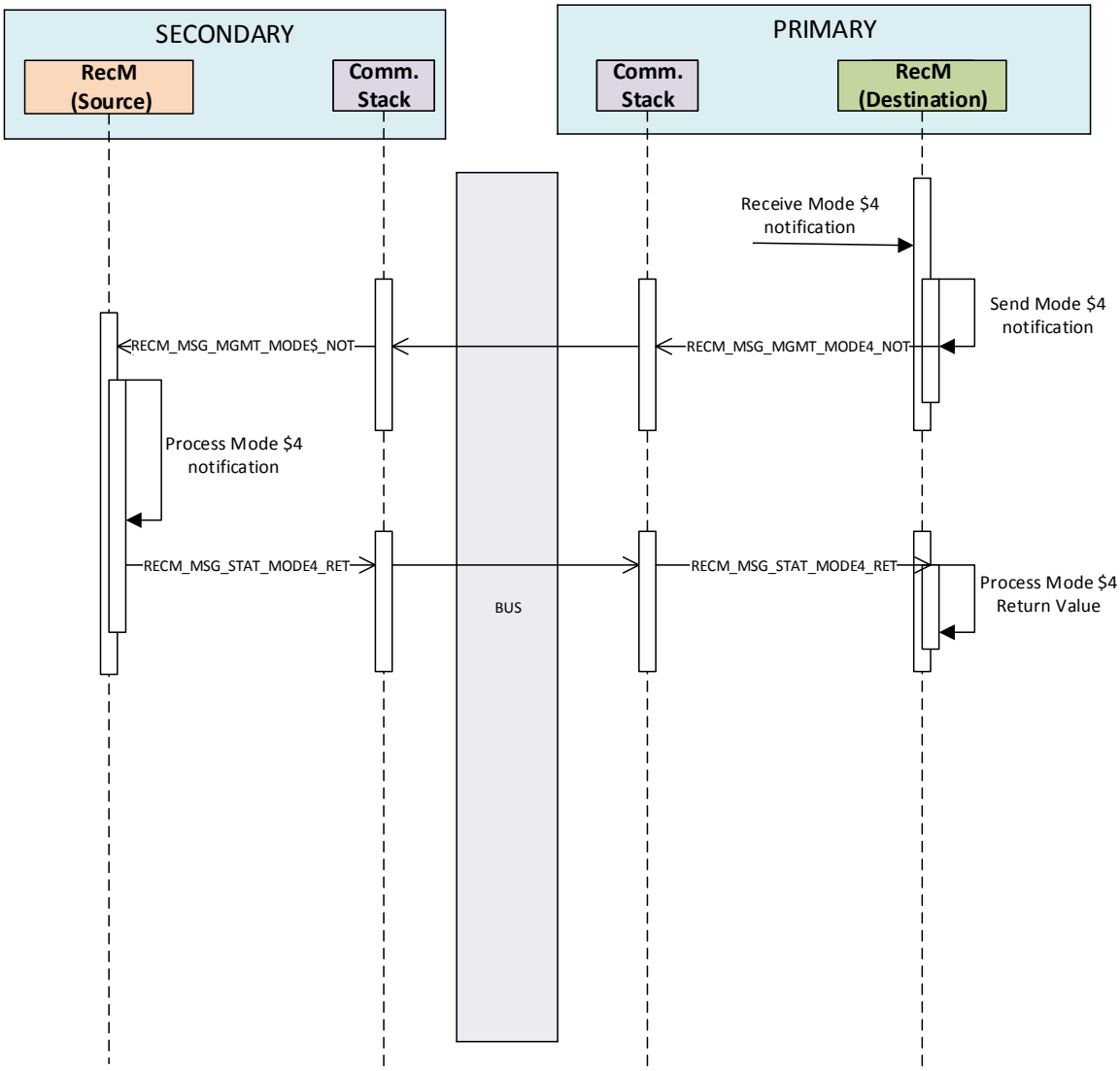


Figure 5.11: Sequence diagram of Process Mode $04 Notification

6 Configuration specification

This chapter summarizes all of parameters that the RecM bus protocol uses.

6.1 Status Messages

6.1.1 Heartbeat Status Message

The Heartbeat Status Message format (Heartbeat_Status_Array) message format is shown in Table 6.1.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_STAT_HTBT
	Client Identifier	CLIENT_ID	Client ECU identifier
	RESERVED	RESERVED	Reserved for future use
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.1: Heartbeat_Status_Array structure parameters

6.1.2 Diagnostic Status Messages

The DTC Status (DTC_Status_Array), Monitor Disabled Status (Monitor_Disabled_Status_Array), IUMPR Fault Detected Status (DTC_Status_Array) and DTR Data Status (DTR_Data_Status_Array) messages all share the same message format shown in Table 6.2.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_STAT_DTC or RECM_MSG_STAT_MON_DIS or RECM_MSG_STAT_FLT_DET
	Client Identifier	CLIENT_ID	Client ECU identifier
	RESERVED	RESERVED	Reserved for future use
Payload	Status Field #1 – Status Field X	STATUS_1– STATUS_X	Status Element Field (status structures are dependent upon message type)
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.2: Diagnostic Status Array structure parameters

6.1.3 Vehicle Information Values Status Message

The Vehicle Information Values Status (Vehicle_Information_Status_Array) message format is shown in Table 6.3.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_STAT_CALID_CVN
	Client Identifier	CLIENT_ID	Client ECU identifier
	RESERVED	RESERVED	Reserved for future use
Payload	CAL–ID Value	STRCT_CALID	Calibration Identifier value
	CVN Value	STRCT_CVN	Calibration Verification Number value
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.3: Vehicle_Information_Status_Array structure parameters

6.1.4 Mode $4 Return Status Message

The Mode $4 Return Status Message format (Mode4_Return_Status_Array) message format is shown in Table 6.4.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_STAT_MODE4_RET
	Client Identifier	CLIENT_ID	Client ECU identifier
	RESERVED	RESERVED	Reserved for future use
Payload	Mode $4 return Value	TYP_MODE4_RET	Return code resulting from Mode $4 notification
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.4: Mode4_Return_Status_Array structure parameters

6.1.5 Version Numbers Status Messages

The Version Numbers Status (Version_Numbers_Status_Array) message format is shown in Table 6.5.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_STAT_VERS
	Client Identifier	CLIENT_ID	Client ECU identifier
	RESERVED	RESERVED	Reserved for future use
Payload	Versions Field #1 – Versions Field X	VERSIONS_1– VERSIONS_X	Versions Element Field (STRCT_VERS_STAT)
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.5: Version_Numbers_Status_Array structure parameters

6.2 Management Messages

6.2.1 Mode $4 Notification Management Message

The Mode $4 Notification Management (Mode4_Not_Management_Array) message format is shown in Table 6.6.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_MGMT_MODE4_NOT
	RESERVED	RESERVED	Reserved for future use
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.6: Mode4_Not_Management_Array structure parameters

6.2.2 Synchronize–All Management Message

The Diagnostic Synchronize–All Management Message format (Diagnostic_Synchronize_All_Management_Array) is shown in Table 6.7.

	Long name:	Short name:	Description:
Header	Message Type	MSG_TYPE	Message type = RECM_MSG_MGMT_SYNC_ALL
	RESERVED	RESERVED	Reserved for future use
Tail	Message Counter	MSG_CTR	Message counter field

Table 6.7: Diagnostic_Synchronize_All_Management_Array structure parameters

6.3 Configuration Parameters

6.3.1 Heartbeat Interval Time

[PRS_RECM_00050] ⌈ The Heartbeat Interval Time parameter, RecMTriggerTimeout, shall define the maximum allowable time interval, in milliseconds, between two messages sent on the network by a Source Role ECU. This parameter shall be configurable. ⌋*(SRS_RECM_00004, SRS_RECM_00013, SRS_RECM_00031, SRS_RECM_00038, SRS_RECM_00067)*

6.3.2 Synchronize–All Message Return Time

[PRS_RECM_00079] ⌈ The Sync–All Message Return Time parameter, SyncAllReturnTime, shall define the maximum time interval, in milliseconds, within which a Source Role ECU shall send all required messages in answer to a Synchronize–All management message. This parameter shall be configurable. ⌋*(SRS_RECM_00004, SRS_RECM_00013)*

6.4 Published Information

None

7 Protocol usage and guidelines

7.1 How–to and guidance for implementors

In an AUTOSAR implementation, all network-specific functionality (the specifics of networks like CAN, LIN, FlexRay or Ethernet) is handled outside of the RecM Bus Protocol. In a "classic" AUTOSAR ECU, the RecM Bus Protocol communication would be between a RecM Bus Protocol compatible module and the TP Interface of the PduR module. The PDU Router (PduR) module provides a network–independent upper–level TP interface to the RECM Bus Protocol compatible module.

The RecM Bus Protocol compatible module in the local Diagnostic Source ECU sends RecM Bus Protocol diagnostic status messages to the PduR module to be forwarded to the remote Destination ECU. The RecM Bus Protocol compatible module in the remote Destination ECU sends RecM Bus Protocol diagnostic management messages to the PduR module to be forwarded to the local Diagnostic Source ECU.

Although, this specification is intended to specify the RecM Bus Protocol communication between two AUTOSAR ECUs, the actual implementation in an ECU is, of course, not required to be realized with AUTOSAR modules. The Extended Dependent Secondary implementations will, in all probability, be mostly non-AUTOSAR modules. The only constraint is that ECUs that employ this bus protocol specification comply completely with this specification.

Although some of the modules shown in the figures, diagrams and sequence diagrams are AUTOSAR classic platform modules, they can be interpreted to represent generic diagnostic management functionality (i.e.: SWC & Application –> generation; DEM –> processing; RecM & DCM –> communication, etc. of diagnostic status information).

7.2 Implementation examples for use cases

In the Thin–client scenario, the Destination role section of the RecM Bus Protocol compatible module in a Secondary ECU receives RecM Bus Protocol diagnostic status messages from the PduR module which were received from a remote Extended Dependent Secondary ECU.

8 Related documentation

8.1 Input documents

Bibliography

[1] Glossary
AUTOSAR_TR_Glossary

[2] Communication
http://portal.osek-vdx.org/files/pdf/specs/
osekcom303.pdf

[3] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager

8.2 Related standards and norms

Not applicable.

8.3 Related specification

Not applicable.