BMW GROUP

Rolls-Royce Motor Cars Limited

StdDiag Classic Integration Manual

Project	BMW AUTOSAR 4 Core Rel. 3
Author	BMW AG
Release Date	2017-12-14
Version	5.4.0
Status	Release
Hotline	+49 89 382 - 32233
Contact	bac@bmw.de https://asc.bmw.com/jira/browse/BSUP (extern) https://asc.bmwgroup.net/jira/browse/BSUP (intern)

Revision History

Version	Date	Description
5.4.0	2017-12-14	BAC-6257: Add integration of functionality "Application Data Transfer" (ADT)
5.3.0	2017-11-09	BAC-6249: Move SgbdIndex to adapter part and add post build support
5.2.0	2017-10-12	Version Update
5.1.0	2017-08-10	Version Update
5.0.0	2017-06-08	Initial version for SP2021

Company
Bayerische Motoren Werke Aktiengesellschaft
Postal address
BMW AG
80788 München
Office address
Forschungs- und Innovationszentrum (FIZ)
Hufelandstr. 1
80937 München
Telephone
Switchboard
+49 89 382-0
Internet
www.bmwgroup.com

Table of Contents

1 Introduction

This Integration Manual describes the basic functionality of the BMW system function "Standard Diagnostics" (StdDiag), the configuration of the StdDiag module and of dependant modules, and the integration of the StdDiag module into BAC4 or aBAC.

Functional overview

The main functionality of the StdDiag module is to handling the active session states ("subsessions") of the default diagnostic session and the extended diagnostic session. It ensures that the programming preparation process (i.e. the transition from the diagnostic default session via the extended diagnostic session to the programming session) is only successful, if the diagnostic requests are received in the correct sequence and all necessary preconditions are fulfilled. It also checks whether diagnostic requests shall be rejected or allowed in the different session states.

The StdDiag module also provides diagnostic service handlers for

- clearing the secondary error memory
- reading the active session state
- reading the programming preconditions
- reading the SGBD-Index
- diagnostic communication loopback functionality
- handling Diagnostic Log and Trace settings
- handling IDRL basic functionality

2 Related documentation

References

3 Limitations

Unreleased Dlt specification in AUTOSAR

StdDiag optionally uses services of the AUTOSAR BSW module Dlt (Diagnostic Log and Trace). The specification of the ClientServer-Interface "DLTService", which provides the services used by StdDiag, is released with AUTOSAR 4.3.0. As required in the document "AUTOSAR features for SP2021", projects using Dlt shall support Dlt based on AUTOSAR 4.3.

4 Software Architecture

Dependencies on AUTOSAR modules

The current version of the Module StdDiag depends on the following BSW modules:

RTE

As a software component, the StdDiag module uses Rte client/server and sender/receiver communication to communicate with other SWCs and BSW modules.

Det

StdDiag optionally reports development errors to the Det.

Dcm

StdDiag is tightly coupled with the Dcm. The StdDiag implements several RDBI/WDBI and RC services, as well as services for upload download functionality. Dcm shall be configured in a way, that it dispatches theses jobs to the StdDiag SWC. In addition, the StdDiag requests information about the current diagnostic session.

Moreover the StdDiag receives information about incoming requests and the corresponding result of the request via a Manufacturer Notification. The StdDiag uses this feature to accept / deny a number of requests, and to handle the active session state.

Dem

StdDiag sets an EnableCondition to allow / deny writing of error entries and clears DTCs in the secondary error memory.

BswM

StdDiag receives and requests mode switches from the BswM to switch the current StdDiag operational mode. It further requests Communication Control mode switches, and receives a mode switch from the BswM when the active diagnostic session has changed.

Dlt

StdDiag optionally receives diagnostic requests for Diagnostic Log and Trace control and forwards them to the Dlt service interface.

EcuC

StdDiag provides post build configurable parameters. If post build support is used, an EcuC configuration is needed to evaluate available variants.

Dependencies to other modules

Darh

The StdDiag suspends / resumes sending Response on Events.

Omc

The StdDiag needs the current operating mode and extended operating mode from the Omc module. In addition, StdDiag provides a handler to allow / deny changing the operating mode.

Stm

The StdDiag reads the current PWF state from the Stm.

Other SWC

The StdDiag optionally calls another Application Software Component to establish intrinsic safety and to check the users programming preconditions.

The StdDiag optionally calls another Application Software Component to get the active session state in Sessions that have their own active session handling.

The StdDiag calls another Application Software Component (e.g. PiaClient) to read, write or reset the individual data, if the feature "Individual Data Recovery Light" (IDRL) is activated.

The StdDiag calls another Application Software Component (e.g. Bs) to upload or download application data, if the feature "Application Data Transfer" (ADT) is activated.

5 Integration

Configuration of other Modules

The following modules shall be configured, before this module can be generated, compiled and linked.

Dcm

Read Data By Identifier

The following RDBI-requests shall be configured in the Dcm module:

Read Active Session State (0x22 0xF1 0x00)

[IM_StdDiagClassic_0007] ⌈The following parameters have to be configured: ⌋(FL896, FL897, FL898, FL899)

- a container "DcmDspData" with
  - a "DcmDspDataSize" of 32 bit (4 bytes)
  - "DcmDspDataType" = UINT8_N
  - "DcmDspDataUsePort" = USE_DATA_SYNCH_CLIENT_SERVER
  - "DcmDspDataConditionCheckReadFncUsed" = TRUE
- a container "DcmDspDid" with
  - "DcmDspDidIdentifier" = 0xF100
  - only read-access, without session or security restrictions
  - one DID Signal "DcmDspDidSignal" with Data Position = 0 and a Data Reference to the "DcmDspData" container configured before.

Read SGBD Index (0x22 0xF1 0x50)

[IM_StdDiagClassic_0006] ⌈The following parameters have to be configured: ⌋(DK_T3_318)

- a container "DcmDspData" with
  - a "DcmDspDataSize" of 24 bit (3 bytes)
  - "DcmDspDataType" = UINT8_N
  - "DcmDspDataUsePort" = USE_DATA_SYNCH_CLIENT_SERVER
  - "DcmDspDataConditionCheckReadFncUsed" = FALSE
- a container "DcmDspDid" with
  - "DcmDspDidIdentifier" = 0xF150
  - only read-access, without session or security restrictions
  - one DID Signal "DcmDspDidSignal" with Data Position = 0 and a Data Reference to the "DcmDspData" container configured before.

Read Individual Data Identifier Table (0x22 0x17 0x34)

This request is needed, if the configuration container "StdDiagProvideIDRL" is available. The response to this request contains the table format identifier (2 byte) and the individual data IDs that are configured in the parameter "DIDValue" of the StdDiag configuration. The response has a fixed length that is calculated with the lengths of the table format and of the individual data IDs.

[IM_StdDiagClassic_0008] ⌈The following parameters have to be configured: ⌋(ADUE_3927, ADUE_3929)

- a container "DcmDspData" with
  - a "DcmDspDataSize": 16 bit for the format identifier plus 16 bit for each configured individual data ID.
  - "DcmDspDataType" = UINT8_N
  - "DcmDspDataUsePort" = USE_DATA_SYNCH_CLIENT_SERVER
  - "DcmDspDataConditionCheckReadFncUsed" = FALSE
- a container "DcmDspDid" with
  - "DcmDspDidIdentifier" = 0x1734
  - only read-access, only allowed in extended diagnostic session
  - one DID Signal "DcmDspDidSignal" with Data Position = 0 and a Data Reference to the "DcmDspData" container configured before.

Note: If IDRL clients with individual data IDs are configured (see parameter "DIDValue"), the user shall configure the following for each individual data ID:

- a container "DcmDspData" describing the individual data with
  - a "DcmDspDataSize": the maximum length of the individual data in bit (see parameter "MaxDataSize").
  - "DcmDspDataType" = UINT8_DYN
  - "DcmDspDataUsePort" = USE_DATA_ASYNCH_CLIENT_SERVER
  - "DcmDspDataConditionCheckReadFncUsed" = TRUE
- a container "DcmDspDid" with
  - "DcmDspDidIdentifier" is the individual data ID "DIDValue" configured in the DID table of the StdDiag.
  - read/write-access, only allowed in extended diagnostic session
  - one DID Signal "DcmDspDidSignal" with Data Position = 0 and a Data Reference to the "DcmDspData" container configured before.

Read log channel names for Dlt (0x22 0x18 0x28)

[IM_StdDiagClassic_0017] ⌈This request is only needed, if the configuration container "StdDiagProvideDLTSupport" is available. The response to this request contains the log channel names supported by Dlt module. The response has a fix length that is derived from the number of configured log channels in the Dlt.

The following parameters have to be configured: ⌋(DK_T3_1637, DK_T3_1639)

- a container "DcmDspData" with
  - a "DcmDspDataSize": 32 bit for each configured DLT Log Channel.
  - "DcmDspDataType" = UINT8_N
  - "DcmDspDataUsePort" = USE_DATA_SYNCH_CLIENT_SERVER

  - "DcmDspDataConditionCheckReadFncUsed" = FALSE
- a container "DcmDspDid" with
  - "DcmDspDidIdentifier" = 0x1828
  - only read-access, without session or security restrictions
  - one DID Signal "DcmDspDidSignal" with Data Position = 0 and a Data Reference to the "DcmDspData" container configured before.

RoutineControl

The following RoutineControl-requests shall be configured in the Dcm module:

Diagnostic Communication Loopback (0x31 0x01 0x03 0x03)

[IM_StdDiagClassic_0004] ⌈This request has a variable request length and a variable response length. The following parameters have to be configured: ⌋(DK_T3_792, DK_T3_799, DK_T3_800)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x0303
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" with
    - "DcmDspRoutineSignalLength" = max. diagnostic buffer size (e.g. 8192 bits)
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = VARIABLE_LENGTH
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineOut" with
  - a "StartRoutineOutSignal" with
    - "DcmDspRoutineSignalLength" = max. diagnostic buffer size (e.g. 8192 bits)
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = VARIABLE_LENGTH

[IM_StdDiagClassic_0003] ⌈Note: According to the "LH Diagnose Part 3" this request shall handle a variable data length in a range from 0 bytes up to the maximum length that the diagnostic buffer can handle. In this example configuration the value is set to 1024 bytes (8192 bits). ⌋(DK_T3_794)

Check Programming Preconditions (0x31 0x01 0x02 0x03)

[IM_StdDiagClassic_0019] ⌈This request has no input signals, and a variable response length. Therefore, it is sufficient to configure only an Out-Signal, but no In-Signal: ⌋(FL1082, FL1083)
[IM_StdDiagClassic_0001] ⌈This request shall be available in all sessions. ⌋(FL194)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x0203
  - "DcmDspRoutineUsePort" = TRUE
  - a reference to "DcmDspCommonAuthorization" which references all (or none) diagnostic sessions
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineOut" with
  - a "StartRoutineOutSignal" with

- "DcmDspRoutineSignalLength" = 8 bit for each programming precondition specified for the ECU (max. 255 preconditions, i.e. 2048 bits)
- "DcmDspRoutineSignalPos" = 0
- "DcmDspRoutineSignalType" = VARIABLE_LENGTH

Reset individual data (0x31 0x01 0x10 0x1A)

[IM_StdDiagClassic_0009] ⌈This request is supported, if the configuration container "StdDiagProvideIDRL" is available. The request has no input signals, and a fixed response length. The response contains the routine result (1 byte). If all resets of individual data are successful, the routine result is 0x00. Otherwise, the routine result is 0x01. Therefore, it is sufficient to configure only an Out-Signal, but no In-Signal: ⌋(ADUE_3981, ADUE_3982)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x101A
  - "DcmDspRoutineUsePort" = TRUE
  - a reference to "DcmDspCommonAuthorization" with a reference to the extended diagnostic session
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineOut" with
  - a "StartRoutineOutSignal" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT8

Clear DTCs Secondary Error Memory (0x31 0x01 0x0F 0x06)

[IM_StdDiagClassic_0005] ⌈This diagnostic request has to be configured only, if the ECU supports DTCs in the secondary error memory. The configuration container StdDiagClearSecondaryErrorMemory has to be available. This request has no input signals and no output signals.

The following parameters have to be configured: ⌋(DK_T3_862, DK_T3_863, DK_T3_864, DK_T3_865)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x0F06
  - "DcmDspRoutineUsePort" = TRUE
  - no further sub-containers configured

Upload/Download Pre/Post-Processing (0x31 0x01 0x70 0x00)

[IM_StdDiagClassic_0023] ⌈This request is supported, if the configuration container "StdDiagApplicationDataTransfer" is available. The request has a variable request length, and a fixed response length. The response contains the routine result (1 byte): ⌋(ADUE_2773, ADUE_2774, ADUE_3618)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x7000
  - "DcmDspRoutineUsePort" = TRUE

  - a reference to "DcmDspCommonAuthorization" with a reference to the extended diagnostic session, but no reference to default or programming session
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "Data" with
    - "DcmDspRoutineSignalLength" = 64 bit - 2184 bit (see note below)
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = VARIABLE_LENGTH
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineOut" with
  - a "StartRoutineOutSignal" named "Result" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT8

Note: Request data consists of 3 byte fix + size of memoryObjectIdentifier + size of applicationSpecificParameter. According to "LH Applikationsdatenuebertragung" the memoryObjectIdentifier size is in range 5 byte to 15 byte, applicationSpecificParameter size is in range 0 byte to 255 byte. So minimum value for "DcmDspRoutineSignalLength" is 64 bit (8 byte), maximum value is 2184 bit (273 byte).

Note: This RoutineControl (RID 0x7000) is a common job for all ADT clients. "LH Applikationsdatenuebertragung" requires further Routine Control jobs for each application specific routine control identifier. These jobs are not part of StdDiag.

Set log level for Dlt (0x31 0x01 0x10 0x90)

[IM_StdDiagClassic_0010] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has only input signals, and a fixed request length.

The following parameters have to be configured: ⌋(DK_T3_1583, DK_T3_1586)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1090
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "applicationId" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT32
  - a "StartRoutineInSignal" named "contextId" with
    - "DcmDspRoutineSignalPos" = 32
    - "DcmDspRoutineSignalType" = UINT32
  - a "StartRoutineInSignal" named "newLogLevel" with
    - "DcmDspRoutineSignalPos" = 64
    - "DcmDspRoutineSignalType" = UINT8

Reset Dlt to default configuration (0x31 0x01 0x10 0x91)

[IM_StdDiagClassic_0011] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has neither input nor output signals.

The following parameters have to be configured: ⌋(DK_T3_1590, DK_T3_1592)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1091
  - "DcmDspRoutineUsePort" = TRUE
  - no further sub-containers configured

Set message filtering state for Dlt (0x31 0x01 0x10 0x92)

[IM_StdDiagClassic_0012] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has only input signals, and a fixed request length.

The following parameters have to be configured: ⌋(DK_T3_1596, DK_T3_1599)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1092
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "newFilterStatus" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT8

Set log channel threshold for Dlt (0x31 0x01 0x10 0x93)

[IM_StdDiagClassic_0022] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has only input signals, and a fixed request length.

The following parameters have to be configured: ⌋(DK_T3_1603, DK_T3_1606)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1093
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "logChannelName" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT32
  - a "StartRoutineInSignal" named "newLogLevelThreshold" with
    - "DcmDspRoutineSignalPos" = 32
    - "DcmDspRoutineSignalType" = UINT8
  - a "StartRoutineInSignal" named "newTraceStatus" with
    - "DcmDspRoutineSignalPos" = 40
    - "DcmDspRoutineSignalType" = UINT8

Store Dlt configuration persistently (0x31 0x01 0x10 0x94)

[IM_StdDiagClassic_0013] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has neither input nor output signals.

The following parameters have to be configured: ⌋(DK_T3_1610, DK_T3_1612)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1094
  - "DcmDspRoutineUsePort" = TRUE
  - no further sub-containers configured

Set trace state for Dlt (0x31 0x01 0x10 0x95)

[IM_StdDiagClassic_0014] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has only input signals, and a fixed request length.

The following parameters have to be configured: ⌋(DK_T3_1616, DK_T3_1619)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1095
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "applicationId" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT32
  - a "StartRoutineInSignal" named "contextId" with
    - "DcmDspRoutineSignalPos" = 32
    - "DcmDspRoutineSignalType" = UINT32
  - a "StartRoutineInSignal" named "newTraceStatus" with
    - "DcmDspRoutineSignalPos" = 64
    - "DcmDspRoutineSignalType" = UINT8

Set default log level for Dlt (0x31 0x01 0x10 0x96)

[IM_StdDiagClassic_0015] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has only input signals, and a fixed request length.

The following parameters have to be configured: ⌋(DK_T3_1623, DK_T3_1626)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1096
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "newDefaultLogLevel" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT8

Set default trace state for Dlt (0x31 0x01 0x10 0x97)

[IM_StdDiagClassic_0016] ⌈This request is only supported, if the configuration container "StdDiagProvideDLTSupport" is available. The request has only input signals, and a fixed request length.

The following parameters have to be configured: ⌋(DK_T3_1630, DK_T3_1633)

- a container "DcmDspRoutine" with
  - "DcmDspRoutineIdentifier" = 0x1097
  - "DcmDspRoutineUsePort" = TRUE
- a subcontainer "DcmDspStartRoutine" and "DcmDspStartRoutineIn" with
  - a "StartRoutineInSignal" named "newDefaultTraceStatus" with
    - "DcmDspRoutineSignalPos" = 0
    - "DcmDspRoutineSignalType" = UINT8

Service Request Manufacturer Notification

A StdDiag Service Request Manufacturer Notification shall be configured in the container "DcmDsdServiceRequestManufacturerNotification". The Mapping to the StdDiag module is done by connecting the corresponding ports.

Service Handler for Upload Download Services

To implement the functionality "Application Data Transfer" (ADT, german "Applikationsdatenuebertragung, ADUE) (see LH "Applikationsdatenuebertragung"), the UDS Services 0x34 (RequestDownload), 0x35 (RequestUpload), 0x36 (TransferData) and 0x37 (RequestTransferExit) are necessary. For these services AUTOSAR Dcm basically provides callouts like "Dcm_ProcessRequestDownload" and "Dcm_WriteMemory", which has two constraints: The parameter MemoryAddress within these APIs is of type uint32, i.e. values are restricted to 4 bytes. ADT needs at least 5 byte for the memory object identifier, which is ISO-14229 compliant. Furthermore Dcm currently only provides C-API callouts, but no Service Interface. For these reasons StdDiag requires the following configuration:

- configure container "DcmDspService" for service "RequestDownload" with
  - "DcmDsdSidTabServiceId" = 0x34
  - "DcmDsdSidTabSubfuncAvail" = 0
  - "DcmDsdSidTabFnc" = StdDiag_RequestDownload
- configure container "DcmDspService" for service "RequestUpload" with
  - "DcmDsdSidTabServiceId" = 0x35
  - "DcmDsdSidTabSubfuncAvail" = 0
  - "DcmDsdSidTabFnc" = StdDiag_RequestUpload
- configure container "DcmDspService" for service "TransferData" with
  - "DcmDsdSidTabServiceId" = 0x36
  - "DcmDsdSidTabSubfuncAvail" = 0
  - "DcmDsdSidTabFnc" = StdDiag_TransferData
- configure container "DcmDspService" for service "RequestTransferExit" with
  - "DcmDsdSidTabServiceId" = 0x37

- "DcmDsdSidTabSubfuncAvail" = 0
- "DcmDsdSidTabFnc" = StdDiag_RequestTransferExit

Det

A StdDiag entry shall be added to the Software Component List from Det. This is only necessary, if the parameter "StdDiagDevErrorDetect" is set to "true".

Dem

[IM_StdDiagClassic_0002] ⌈In the container "DemEnableCondition" a new Enable Condition has to be configured (StdDiag Enable Condition). This Enable condition has to be referenced by each Enable Condition Group configured in the container "DemEnableConditionGroup". In addition, each configured Event ("DemEventParameter") has to refer to an Enable Condition Group ("DemEnableConditionGroupRef") in its Event Class ("DemEventClass"). ⌋(FL257)

Exceptions: The Events related to the DTCs "CODING_EVENT_NOT_CODED" (see IntegrationManual_Coding.pdf) and "Energy saving mode active" (see IntegrationManual_Omc.pdf) shall not be locked by the StdDiag Enable Condition. These Events shall refer to either no Enable Condition Group, or to an Enable Condition Group that does not include the StdDiag Enable Condition.

BswM

The following **BswMModeRequestPorts** have to be configured in the BswM:

Port_StdDiag_LifeCycle
This port shall have a BswMModeRequestSource of type "BswMSwcModeNotification", and receives mode switch notifications of the ModeDeclarationGroup "StdDiag_LifeCycle" from the StdDiag module.

Port_StdDiag_ComControlModeRequest
This port shall have a BswMModeRequestSource of type "BswMSwcModeRequest", and receives mode requests of the ModeDeclarationGroup "StdDiag_NormalCommunicationModeGroup" from the StdDiag module.

Port_Dcm_CommunicationControl
This port shall have a BswMModeRequestSource of type "BswMDcmComModeRequest", and shall receive the current communication mode of the ModeDeclarationGroup "Dcm_CommunicationControl_<Channel>" defined by the Dcm.

Port_Dcm_SessionControl
This port shall have a BswMModeRequestSource of type "BswMSwcModeNotification", and shall receive the current diagnostic session of the ModeDeclarationGroup "Dcm_DiagnosticSessionControl" defined by the Dcm.

The following **BswMRteModeRequestPort** have to be configured in the BswM:

StdDiagLifeCycleRequestPort

To request modes of the ModeDeclarationGroup "StdDiag_LifeCycle", this port shall have a BswMRteModeRequestPortInterfaceRef to the variable data prototype "requestMode" related to the Port "LifeCycle" of the StdDiag module.

The following **BswMSwitchPorts** have to be configured in the BswM:

StdDiagSessionChangeIndicationPort
To switch modes of the ModeDeclarationGroup "StdDiag_SessionModeGroup", this port shall have a BswMModeSwitchInterfaceRef to the ModeSwitchInterface "SessionChangeIndicationInterface" of the StdDiag module.

StdDiagComControlNormalNotificationPort
To switch modes of the ModeDeclarationGroup "StdDiag_NormalCommunicationModeGroup", this port shall have a BswMModeSwitchInterfaceRef to the ModeSwitchInterface "ComControlNormalNotificationInterface" of the StdDiag module.

The following **Rules** have to be configured in the BswM:

LifeCycle handling:
To initialize the StdDiag module the BswM has to provide a rule, that results in an action that requests the mode "STDDIAG_INITIALIZED" of the mode declaration group "StdDiag_LifeCycle".

To set the StdDiag module to normal operation mode the BswM has to provide a rule, that results in an action that requests the mode "STDDIAG_RUNNING" of the mode declaration group "StdDiag_LifeCycle". This rule is triggered by a mode switch to the mode "STDDIAG_INITIALIZED" by the StdDiag module itself. When the StdDiag module is in normal operation mode, the StdDiag module switches the mode to "STDDIAG_RUNNING".

To deactivate the StdDiag module the BswM has to provide a rule, that results in an action that requests the mode "STDDIAG_STOPPED" of the mode declaration group "StdDiag_LifeCycle".

For details on how to initialize / deactivate the StdDiag module, please refer to chapter 5.5.

Diagnostic Session handling:
To notify the StdDiag module when the DefaultSession is entered, the BswM has to provide a rule, that results in an action that switches the mode of the mode declaration group "StdDiag_SessionModeGroup" to "STDDIAG_DEFAULT_SESSION" via BswMSwitchPort "StdDiagSessionChangeIndicationPort" (see above), when the Dcm module indicates a mode switch of the mode declaration group "DcmDiagnosticSessionControl" to "DefaultSession" via BswMModeRequestPort "Port_Dcm_SessionControl".

To notify the StdDiag module when the DefaultSession is left, the BswM has to provide a rule, that results in an action that switches the mode of the mode declaration group "StdDiag_SessionModeGroup" to "STDDIAG_OTHER_SESSION" via BswMSwitchPort "StdDiagSessionChangeIndicationPort" (see above), when the Dcm module indicates a mode switch of the mode declaration group "DcmDiagnosticSessionControl" to any other than the "DefaultSession" via BswMModeRequestPort "Port_Dcm_SessionControl".

Communication Control handling:
To notify the StdDiag module about a change of the normal communication mode, the BswM has to provide rules that result in actions that switch the corresponding mode of the mode declaration group

"StdDiag_NormalCommunicationModeGroup" via BswMSwitchPort "StdDiagComControlNormalNotificationPort" (see above).

These rules have to be triggered either when the Dcm switches the mode of the mode declaration group "DcmCommunicationControl_<Channel>", or when the StdDiag module itself requests a mode of the mode declaration group "StdDiag_NormalCommunicationModeGroup".

(**Note:** These rules also have to result in actions that enable / disable the corresponding Pdu-Groups)

To achieve a correct combination of the two request sources StdDiag and Dcm, the following four rules should be configured:

BswMRule_ComControl_Enable_Rx_Enable_Tx
if ((LogEx_Enable_Rx == TRUE) && (LogEx_Enable_Tx == TRUE))
{
Switch StdDiagComControlNormalNotificationPort to ENABLE_RX_AND_TX_NORMAL
}

BswMRule_ComControl_Enable_Rx_Disable_Tx
if ((LogEx_Enable_Rx == TRUE) && (LogEx_Disable_Tx == TRUE))
{
Switch StdDiagComControlNormalNotificationPort to ENABLE_RX_DISABLE_TX_NORMAL
}

BswMRule_ComControl_Disable_Rx_Enable_Tx
if ((LogEx_Disable_Rx == TRUE) && (LogEx_Enable_Tx == TRUE))
{
Switch StdDiagComControlNormalNotificationPort to DISABLE_RX_ENABLE_TX_NORMAL
}

BswMRule_ComControl_Disable_Rx_Disable_Tx
if ((LogEx_Disable_Rx == TRUE) && (LogEx_Disable_Tx == TRUE))
{
Switch StdDiagComControlNormalNotificationPort to DISABLE_RX_AND_TX_NORMAL
}

The corresponding **locigal expressions** are defined as follows:

Both StdDiag and Dcm allow to enable Rx:
LogEx_Enable_Rx = (LogEx_Enable_Rx_StdDiag) && (LogEx_Enable_Rx_Dcm)

Both StdDiag and Dcm allow to enable Tx:
LogEx_Enable_Tx = (LogEx_Enable_Tx_StdDiag) && (LogEx_Enable_Tx_Dcm)

Either StdDiag or Dcm want to disable Rx:
LogEx_Disable_Rx = !((LogEx_Enable_Rx_StdDiag) && (LogEx_Enable_Rx_Dcm))

Either StdDiag or Dcm want to disable Tx:
LogEx_Disable_Tx = !((LogEx_Enable_Tx_StdDiag) && (LogEx_Enable_Tx_Dcm))

LogEx_Enable_Rx_StdDiag =
Port_StdDiag_ComControlModeRequest == ENABLE_RX_AND_TX_NORMAL ||

Port_StdDiag_ComControlModeRequest == ENABLE_RX_DISABLE_TX_NORMAL

LogEx_Enable_Rx_Dcm =
Port_Dcm_CommunicationControl == DCM_ENABLE_RX_TX_NORM ||
Port_Dcm_CommunicationControl == DCM_ENABLE_RX_DISABLE_TX_NORM ||
Port_Dcm_CommunicationControl == DCM_ENABLE_RX_TX_NORM_NM ||
Port_Dcm_CommunicationControl == DCM_ENABLE_RX_DISABLE_TX_NORM_NM

LogEx_Enable_Tx_StdDiag =
Port_StdDiag_ComControlModeRequest == ENABLE_RX_AND_TX_NORMAL ||
Port_StdDiag_ComControlModeRequest == DISABLE_RX_ENABLE_TX_NORMAL

LogEx_Enable_Tx_Dcm =
Port_Dcm_CommunicationControl == DCM_ENABLE_RX_TX_NORM ||
Port_Dcm_CommunicationControl == DCM_DISABLE_RX_ENABLE_TX_NORM ||
Port_Dcm_CommunicationControl == DCM_ENABLE_RX_TX_NORM_NM ||
Port_Dcm_CommunicationControl == DCM_DISABLE_RX_ENABLE_TX_NORM_NM

EcuC

If post build support is needed, the EcuC configuration shall provide a collection of toplevel PostBuildSelectable variants in containers "EcucPostBuildVariants". These containers shall refer to predefined variants, which in turn refer to post build variant criterion value sets. During the generation process, PAGe evaluates all available variants whether the criterion value matches the value given in the variation point of the post build selectable configuration parameter.

Configuration of generic part

StdDiagGeneral

This container contains the configuration parameters of the generic part of the StdDiag module

StdDiagDevErrorDetect

This parameter activates/deactivates the Development Error Detection and Notification.

If set to true: Development Error Detection and Notification is activated.
If set to false: Development Error Detection and Notification is deactivated.

StdDiagUserEstablishIntrinsicSafety

[IM_StdDiag_0001] ⌈During UDS Routine Service 0x31 01 0F 0C 03 (set EnergyMode = Flash) an ECU shall establish intrinsic safety. StdDiag checks whether the PWF status is "PruefenAnalyseDiagnose", which is mandatory for all ECUs. During UDS Routine Service 0x31 01 0F 0C

00 (set EnergyMode = Normal), if flash mode was active before, an ECU shall revert all actions taken to establish intrinsic safety. If there are more actions to be taken by the ECU to establish or revert intrinsic safety, this parameter has to be set to "true". ⌋(FL215, FL216, FL217, FL283, FL284, FL285)

If set to true: StdDiag calls a user specific function to establish or revert intrinsic safety.
If set to false: StdDiag only checks for correct PWF state.

StdDiagUserActiveSessionState

The positive response on UDS service 0x22 F1 00 provides the active session state in its second byte. StdDiag provides the active session state within the application default session and the application extended diagnostic session. For all other sessions StdDiag sets the active session state to "0", or optionally calls a user specific application that provides this information.

If set to true: StdDiag call a user specific function that provides the active session state.
If set to false: StdDiag uses the default value "0" for the active session state.

StdDiagUserProgrammingPreconditionsCheck

[IM_StdDiag_0002] ⌈To evaluate EUC specific programming preconditions, StdDiag optionally provides a callout to a user specific application. If an ECU has to evaluate programming preconditions, the integrator shall provide this configuration container and configure its parameters. This container defines whether a user specific function for checking additional programming preconditions is used. ⌋(FL191, FL193)

MaxNumberUserProgrammingPrecondition

This parameter defines the maximum number of failed user programming preconditions.

StdDiagProvideIDRL

This configuration container defines if the StdDiag module shall provide the IDRL feature.

Container is available: StdDiag provides the IDRL feature Container is not available: StdDiag does not provide the IDRL feature

If this container is available, the following parameters and subcontainers, which are part of this container, shall be configured.

The user has at least to configure the format identifier of the DID table.

DIDTableFormatIdentifier

This parameter defines the format identifier of the individual DID table. According to LH ADUE this parameter has to be set to 0x0001.

IDRLClient

This container defines the name of an application software component that implements an IDRL client, i.e. that provides individual data. This container has a subcontainer IndivData. This container shall be configured once for each application component implementing an IDRL client.

IndivData

The container IndivData defines the individual data of an IDRL client. The configured name of this container represents a symbolic name for the individual data. This container has a configuration parameter DIDValue.

DIDValue

Each individual data has a 2 bytes data identifier. This identifier shall be configured in parameter DIDValue.

MaxDataSize

This parameter defines the maximum size of individual data for the corresponding DIDValue.

Configuration of adapter part

StdDiagGeneral

This container contains the configuration parameters of the classic adapter of the StdDiag module.

StdDiagClearSecondaryErrorMemory

This parameter defines if the StdDiag module provides a service handler to clear the secondary error memory via the UDS service 0x31 01 0F 06. true: StdDiag provides a service handler false: StdDiag does not provide a service handler

If the ECU supports DTCs in the secondary error memory, this parameter shall be set to "true", and the following configuration container "StdDiagUserDefinedMemory" as well as the two parameters, which are part of this container, shall be configured.

StdDiagUserDefinedMemory

This configuration container and the parameters within this container have to be configured, if the parameter "StdDiagClearSecondaryErrorMemory" is set to 'true'

StdDiagUserDefinedMemoryName

This string parameter shall be set to the short-name of the Dem configuration container DemUserDefinedMemory, which represents the secondary error memory. To clear the secondary error memory, StdDiag calls the operation "ClearDTC" of the Interface "CddIf" with Parameter "DTCOrigin" set to "DEM_DTC_ORIGIN_USERDEFINED_MEMORY_XX", where "XX" is the short-name of DemUserDefinedMemory. StdDiag uses the parameter "StdDiagUserDefinedMemoryName" to replace "XX" here.

StdDiagUserDefinedMemoryId

This parameter shall be set to the value of "Dem_DTCOriginType" (i.e. the value of the compu-scale "DEM_DTC_ORIGIN_USERDEFINED_MEMORY_XX"), which represents the secondary error memory.

Basically this is the value of the Dem parameter "DemUserDefinedMemoryIdentifier". According to AUTOSAR 4.2.2 the value of the Dem parameter is restricted to a range 16..255. As BMW requires a value of 0x01 for the parameter "MemorySelection" of UDS-Service 0x19 with subfunctions 0x17, 0x18 or 0x19, some Stack vendors might provide different workarounds to realize this. StdDiag provides the configuration parameter "StdDiagUserDefinedMemoryId" to configure the value of the compu-scale "DEM_DTC_ORIGIN_USERDEFINED_MEMORY_XX" according to the stack specific implementation.

With RfC70370 the range of "DemUserDefinedMemoryIdentifier" will be changed to 0..255.

StdDiagSgbdIndex

[IM_StdDiagClassic_0020] ⌈This container defines the SGBD-Index which is read out by UDS service 0x22 F1 50. ⌋(DK_T3_312, DK_T3_316)

SgbdIndex

[IM_StdDiagClassic_0021] ⌈This parameter shall be set to the value of the SGBD-Index. This value shall be in range 0x000000 to 0xFFFFFF. ⌋(DK_T3_312, DK_T3_316)

Note: The parameter "SgbdIndex" is post build configurable. If a post build selectable SgbdIndex is needed, the parameter "SgbdIndex" shall be configured once for each variant. Each instance of "SgbdIndex" shall have a variation point with a post build variant condition, that specifies the necessary value of the corresponding post build variant criterion.

StdDiagApplicationDataTransfer

This configuration container and its parameters and subcontainers has to be configured to enable functionality application data transfer (ADT). ADT is necessary to support e.g. certificate and key managemet of system function "Basic Security" (BS).

ApplicationRoutineControlIdentifier

This configuration container shall exist for each supported applicationRoutineControlIdentifier.

RoutineIdentifierValue

This parameter shall be set to the value of the applicationRoutineControlIdentifier (e.g. 0x10AA for BS).

ApplicationSubRoutineControlIdentifier

This configuration container shall exist for each supported applicationSubRoutineControlIdentifier. If no applicationSubRoutineControlIdentifier is needed, this container shall be configured anyway, and parameter SubRoutineIdentifierValue shall be set to 0x0000.

SubRoutineIdentifierValue

This parameter shall be set to the value of the applicationSubRoutineControlIdentifier (e.g. 0x0000 for BS).

controlIDs

This parameter has a multiplicity of 1..n. For each cntrlID of the corresponding combination applicationRoutineControlIdentifier/applicationSubRoutineControlIdentifier one instance of this parameter shall exist, and shall be set to the value of the cntrlID (e.g. one instance with value 0x00 for BS).

StdDiagProvideDLTSupport

This configuration container defines if the StdDiag module shall provide the implementation for BMW specific support of Diagnostic Log and Trace (DLT).

container is available: StdDiag provides the DLT feature container is not available: StdDiag does not provide the DLT feature

If this container is available, the following parameter, which is part of this container, shall be configured.

StdDiagNumberSupportedDLTLogChannels

This parameter defines the number of supported DLT log channels. This has to be equal to the number of DltLogChannel containers in Dlt module configuration.

Configuration of the RTE

After performing the steps indicated in chapters 5.1, 5.2 and 5.3, the RTE configuration can be started. In other way, the RTE will report an interface incompatibility error.

Assembly Software Connectors

The ports of the StdDiag module have to be connected with ports of other modules as follows:

Dcm

ActiveSessionState <-> DataServices_<Data>
where <Data> is the name of the corresponding container "DcmDspData" describing the active Session State configured in the Dcm module according to chapter 5.1.1.

SgbdIndex <-> DataServices_<Data>
where <Data> is the name of the corresponding container "DcmDspData" describing the SGBD Index configured in the Dcm module according to chapter 5.1.1.

DiagCommLoopback <-> RoutineServices_<Routine>
where <Routine> is the name of the corresponding container "DcmDspRoutine" configured in the Dcm module according to chapter 5.1.1.

CheckProgrammingPreconditions <-> RoutineServices_<Routine>
where <Routine> is the name of the corresponding container "DcmDspRoutine" configured in the Dcm module according to chapter 5.1.1.

ClearSecondaryErrorMemory <-> RoutineServices_<Routine>
where <Routine> is the name of the corresponding container "DcmDspRoutine" configured in the Dcm module according to chapter 5.1.1. (Only necessary when configuration container "StdDiagClearSecondaryErrorMemory" is available, see chapter 5.3.)

ServiceRequestManufacturerNotificationPort <-> ServiceRequestNotification_<Name>
where <Name> is the name of the corresponding container "DcmDslServiceRequestManufacturerNotification" configured in the Dcm module according to chapter 5.1.1.

DCMServicesPort <-> DCMServices

If the configuration container "StdDiagProvideIDRL" is available (see chapter 5.2), the following ports of the StdDiag module have to be connected with the ports of DCM as follows:
IdrIDidTable <-> DataServices_<Data>

where <Data> is the name of the corresponding container "DcmDspData" configured in the Dcm module according to chapter 5.1.1.

ResetIdrlData <-> RoutineServices_<Routine>
where <Routine> is the name of the corresponding container "DcmDspRoutine" configured in the Dcm module according to chapter 5.1.1.

If the configuration container "StdDiagProvideIDRL" is available, and the IDRL clients and the individual data IDs are also configured (see chapter 5.2), the following ports of the StdDiag module have to be connected with the ports of DCM as follows:
IdrlData<IDRLClient><DID> <-> DataServices_<Data>
where <IDRLClient> is the name of an IDRL client configured in container IDRLClient, <DID> is the symbolic name of the individual data configured in container IndivData, and <Data> is the name of the corresponding container "DcmDspData" configured in the Dcm module according to Note of chapter 5.1.1.

If the configuration container "StdDiagApplicationDataTransfer" is available (see chapter 5.3), the following port of the StdDiag module has to be connected with the port of DCM as follows:
UpDownloadPrePostProcessingPort <-> RoutineServices_<Routine>
where <Routine> is the name of the corresponding container "DcmDspRoutine" configured in the Dcm module according to chapter 5.1.1.

If the configuration container "StdDiagProvideDLTSupport" is available (see chapter 5.3), the following ports of the StdDiag module have to be connected with the ports of DCM as follows:

DltReadLogChannelNames <-> DataServices_<Data>
where <Data> is the name of the corresponding container "DcmDspData" configured in the Dcm module according to chapter 5.1.1.
DltSetLoglevel <-> RoutineServices_<Routine>
DltResetToDefault <-> RoutineServices_<Routine>
DltSetMessagefilteringstate <-> RoutineServices_<Routine>
DltSetLogchannelThreshold <-> RoutineServices_<Routine>
DltStoreConfiguration <-> RoutineServices_<Routine>
DltSetTracestate <-> RoutineServices_<Routine>
DltSetDefaultLoglevel <-> RoutineServices_<Routine>
DltSetDefaultTracestate <-> RoutineServices_<Routine>
where <Routine> is the name of the corresponding container "DcmDspRoutine" configured in the Dcm module according to chapter 5.1.1.

Dem

EnableConditionPort <-> EnableCond_<Name>
where <Name> is the name of the corresponding container "DemEnableCondition" configured in the Dem module according to chapter 5.1.3.

ClearDTCPort<-> Dem_Cdd
(Only necessary when configuration container "StdDiagClearSecondaryErrorMemory" is available, see chapter 5.3)

Det

ReportErrorPort <-> DS<xxx>
where <xxx> is an identifier of the StdDiag configured in the Det module (see chapter 5.1.2).

Darh

RoEStatePort <-> RoeStatePort

Omc

OperatingModeControlPort <-> operatingModeSwitchPort
ExtendedOperatingModeControlPort <-> extendedOperatingModeSwitchPort
AllowOpModeChangePort <-> StdDiag AllowOpModeChangeCbkPort <-> StdDiagCbk

Stm

VehicleStatePort <-> VehicleStateSP2015ModeSwitchPort

BswM

LifeCycle <-> BswMModeRequestPort_xxx
where BswMModeRequestPort_xxx means the R-Port of the BswM that receives a mode switch of the ModeDeclarationGroup "StdDiag_LifeCycle" from the StdDiag module. (see "Port_StdDiag_LifeCycle" in chapter 5.1.4)

LifeCycleRequest <-> RteModeRequestPort_xxx
where RteModeRequestPort_xxx means the P-Port of the BswM that provides a mode requret of the ModeDeclarationGroup "StdDiag_LifeCycle" to the StdDiag module. (see "StdDiagLifeCycleRequestPort" in chapter 5.1.4)

SessionChangeIndicationPort <-> ModeSwitchPort_xxx
where ModeSwitchPort_xxx means the P-Port of the BswM that provides a mode switch of the ModeDeclarationGroup "StdDiag_SessionModeGroup" to the StdDiag module. (see "StdDiagSessionChangeIndicationPort" in chapter 5.1.4)

ComControlNormalModeAccessPort <-> ModeSwitchPort_xxx
where ModeSwitchPort_xxx means the P-Port of the BswM that provides a mode switch of the ModeDeclarationGroup "StdDiag_NormalCommunicationModeGroup" to the StdDiag module. (see "StdDiagComControlNormalNotificationPort" in chapter 5.1.4)

ComControlModeRequestPort <-> ModeRequestPort_xxx
where ModeRequestPort_xxx means the R-Port of the BswM that receives a mode request of the ModeDeclarationGroup "StdDiag_NormalCommunicationModeGroup" from the StdDiag module. (see "Port_StdDiag_ComControlModeRequest" in chapter 5.1.4)

Other Application SWC

UserEstablishIntrinsicSafetyPort <-> ApplicationPort
where ApplicationPort means the P-Port of an Application SWC that establishes intrinsic safety. (Only necessary when parameter "StdDiagUserEstablishIntrinsicSafety" is set to 'true', see chapter 5.2.)

UserEstablishIntrinsicSafetyCbkPort <-> ApplicationPort
where ApplicationPort means the R-Port of an Application SWC that notifies StdDiag when intrinsic safety is established. (Only necessary when parameter "StdDiagUserEstablishIntrinsicSafety" is set to 'true', see chapter 5.2.)

UserProgrammingPreconditionsCheckPort <-> ApplicationPort
where ApplicationPort means the P-Port of an Application SWC that provides a user programming precondition check. (Only necessary when parameter "UserProgrammingPreconditionsCheck" is set to 'true', see chapter 5.2.)

UserActiveSessionStatePort <-> ApplicationPort
where ApplicationPort means the P-Port of an Application SWC that provides the active session state for HDD Update Session. (Only necessary when parameter "StdDiagUserActiveSessionState" is set to 'true', see chapter 5.2.)

If the configuration container "StdDiagProvideIDRL" is available, and the IDRL clients and the individual data IDs are also configured (see chapter 5.2), the following ports of the StdDiag module have to be connected with the ports of the IDRL clients as follows:

IdrlDataSwc<IDRLClient><DID> <-> ApplicationPort
where <IDRLClient> is the name of an IDRL client configured in container IDRLClient, and <DID> is the symbolic name of the individual data configured in container IndivData (see chapter 5.2).

ResetIdrlData<IDRLClient> <-> ApplicationPort
where <IDRLClient> is the name of an IDRL client configured in container IDRLClient (see chapter 5.2).

If the configuration container "StdDiagApplicationDataTransfer" is available, and the applicationRoutineControlIdentifer, applicationSubRoutineControlIdentifer and cntrlIDs are also configured (see chapter 5.3), the following ports of the StdDiag module have to be connected with the ports of the ADT clients as follows:

Adt_<applRID>_<applSubRID>_CtrlID_<cntrlID> <-> ApplicationPort
where ApplicationPort means the P-Port of an Application SWC that implements an ADT client (e.g. port "StdDiag_AdueCertificates" of module Bs), and where <applRID> is the name of the container "ApplicationRoutineControlIdentifier", <applSubRID> is the name of the container "ApplicationSubRoutineControlIdentifier", and <cntrlID> is the hex representation of parameter "controlIDs" (see chapter 5.3).

Note: <applSubRID> is an empty string if corresponding parameter "SubRoutineIdentifierValue" is 0.

Adt_<applRID>_<applSubRID>_CtrlID_<cntrlID>_Cbk <-> ApplicationPort
This is the corresponding optional callback port connection, which is only necessary if an ADT client handles requests asynchronously.

Event Mapping

The mode switch events "Event_SessionChange_DefaultSession" and "Event_SessionChange_OtherSession" have to be mapped to an os-task.

Data Mapping

No Data Mapping necessary for the StdDiag module.

Exclusive Areas

No Exclusive Area available in the StdDiag module.

Software Integration

Startup/Initialization

Before initialization of the StdDiag module, the modules Det, Dcm, Dem, Omc, Darh and the RTE have to be initialized. To initialize the StdDiag module, the BswM shall request the mode "STDDIAG_INITIALIZED" of the mode declaration group "StdDiag_LifeCycle".

When initialization of the StdDiag has been finished successfully, the StdDiag module switches the mode to "STDDIAG_INITIALIZED".

Normal Operation

When the StdDiag module switches the mode to "STDDIAG_INITIALIZED", the BswM shall request the mode "STDDIAG_RUNNING". There are no further conditions that have to be considered. When the StdDiag module switches the mode to "STDDIAG_RUNNING", the module is in normal operation mode.

Shutdown/Deactivation

To deactivate the StdDiag module, the BswM has to request to the mode "STDDIAG_STOPPED". There are no further conditions that have to be considered. When the StdDiag module switches the mode to "STDDIAG_STOPPED", the module is deactivated.

Select Post Build Configuration

If the configuration parameter "SgbdIndex" is configured for two or more variants, the integrator shall call the API `StdDiag_SetConfiguration(const StdDiag_PBConfigType * selectedConfig)`, with parameter `selectedConfig` set to the desired post build variant. Possible variants and necessary

type definitions are available in the generated header file `StdDiagClassic_PBCfg.h`. If only one SgbdIndex is configured, this configuration is used by default, i.e. it is not necessary to call this API.

SWCD

After generating the Dcm SWC description files the integrator shall perform changes within StdDiag_ext_interface.arxml.pgen to make the Rte interfaces between Dcm and StdDiag compatible from Rte point of view. Within the StdDiag_ext_interface.arxml.pgen the <ARRAY-SIZE> of the parameters Indication_ArrayType and DiagCommLoopback_ArrayType shall be set to the values given by Dcm. If the RTE importer still complains about incompatible interfaces, please check the compatibility of the StdDiag configuration and the Dcm configuration.

Prevent sleep mode

[IM_StdDiagClassic_0018] ⌈LH Fahrzeugprogrammierung (10505691-000-02) requires in FL247 that an ECU shall not go into sleep mode as long as EnergyMode is set to "Flash". This has to be ensured by the integrator, and can be acheived in different ways. One possibility is the following:

- Evaluate the current operating mode (=EnergyMode) provided by module "Omc" via Port "operatingModeSwitchPort"
- Create BswM-Rules that realize the following items
- If the operating mode equals "Flash" (0x03), trigger communication request for Partial Network "Fahrzeug Infrastruktur"
- If the operating mode does not equal "Flash" (0x03), release communication request for Partial Network "Fahrzeug Infrastruktur"

For details on the operating mode refer to OmcClassic_IntegrationManual.pdf ⌋(FL247)