

Diagnostic Functionality for Scan Tool and Service



Diesel Gasoline Systems

Imprint

- Publisher Robert Bosch GmbH
DGS-EC Training Centre, Fe052
Wernerstraße 1, 70469 Stuttgart
- Author DGS-EC/ESC DiagInf Team
- Person responsible for this topic Frank, Rainer (DGS-EC/ESC)
- Version 090924
- Year of publication 2009
- Place of printing DGS-EC Training Centre
- Note The printed material forms a working copy which is not subject to document control. You will find the current material on the Bosch intranet at:
http://www.intranet.bosch.com/ds/esq/100_topics/500_fit/059_files/020_Seminarunterlagen/index.html



Information on Presentation Updates

Title: DiagInf training for function developer and calibration engineer

Released on: 2009-09-24

Released by: Rainer Frank (DGS-EC/ESC)

Author : DiagInf Team (DGS-EC/ESC)

Initial version: 1.0.0 (this version)

Scope: 199

Format: PowerPoint

Next check: none planned

Slide No.	Title	Brief description of changes	Changes authorized by	Changes carried out by	Date



Lecturer

- Robert Bosch GmbH
DGS-EC/ESC
- Anne Reinhart
- Telephone: 0711 / 811 42272
- E-Mail: Anne.Reinhart@de.bosch.com







Introduction and Objectives

This training course should provide developers and calibration engineers with an overview of the diagnostic infrastructure

- Aim: Participants are able to use and calibrate the components of the diagnostic infrastructure.



Diagnostic Infrastructure Training Course

	09:00:00	Introduction	30min
	09:30:00	ATS	45min
	10:15:00	AVS	45min
	11:00:00	Break	15min
	11:15:00	ETC	45min
	12:00:00	Exercises part I	45min
	12:45:00	Lunchbreak	60min
	13:45:00	Signals	45min
	14:30:00	Signals/OBD	15min
	14:45:00	Break	15min
	15:00:00	I15031	45min
	15:45:00	Break	15min
	16:00:00	Exercises part II	60min
	17:00:00	End	



Agenda

→ Chapter 1 General: DiagInf

- **Function**
- Configuration (BCT)

→ Chapter 2 Service diagnostics

- ATS (Advanced Test Service) – actuators
- AVS (Adjustment Value Service) – adjustments
- ETC (Engine Test Coordinator) – engine tests
- Signals

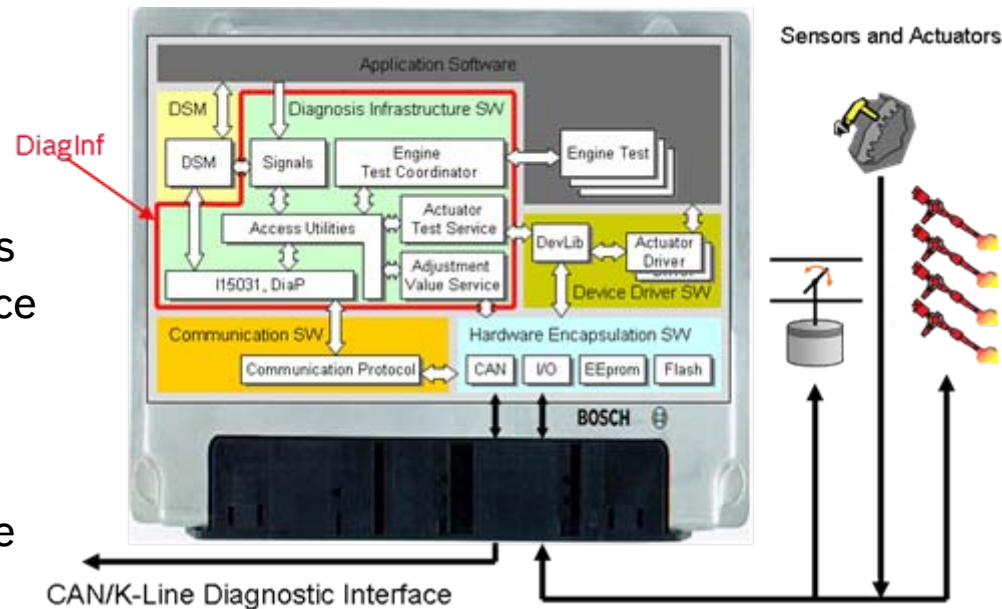
→ Chapter 3 OBD – Diagnostics

- Signals/OBD
- I15031

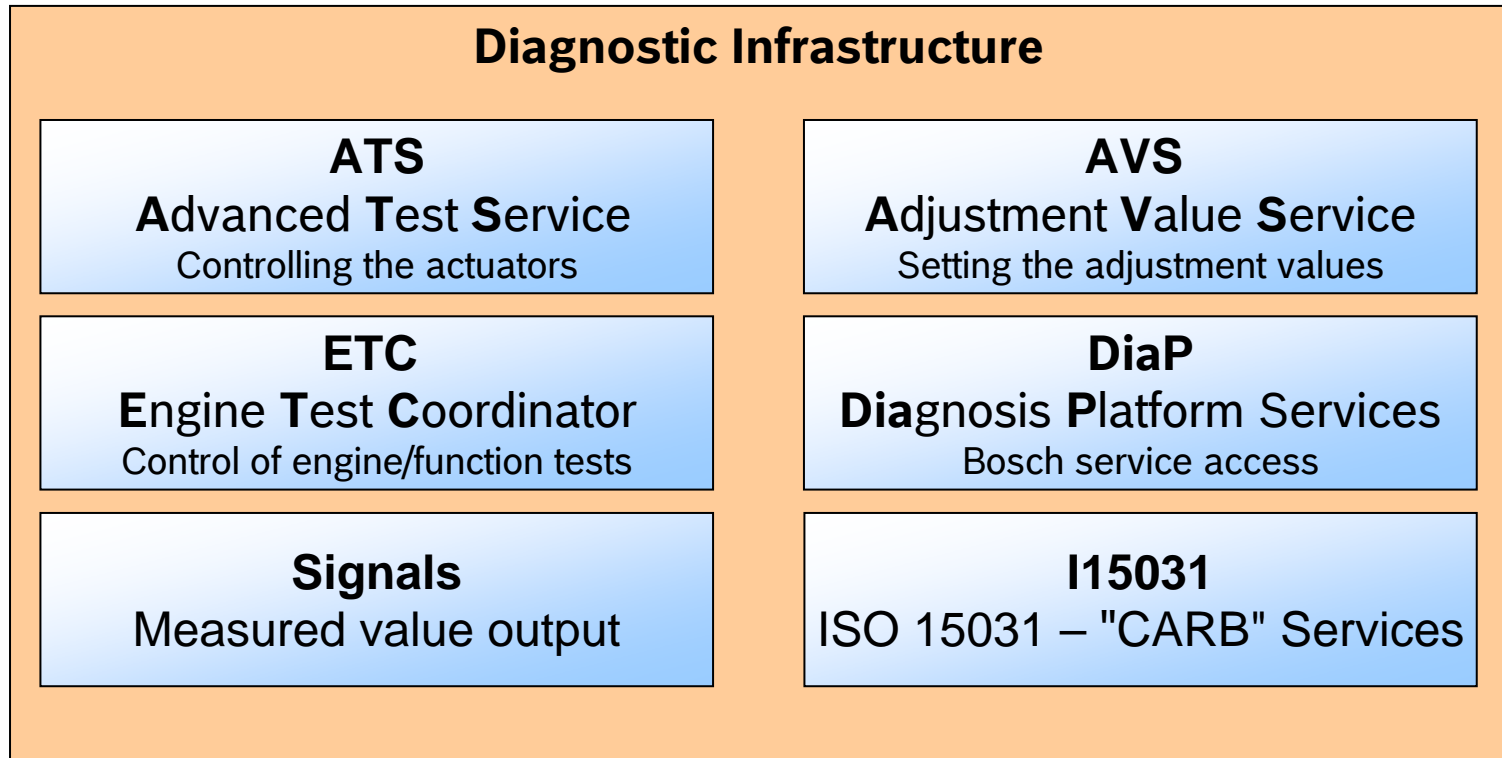


What does the Diagnostic Infrastructure offer?

- ➔ ... summarizes functional parts of diagnosis and provides a common interface for those
- ➔ ... supports the following services/diagnostic functions
 - functions for the customer-specific service diagnostics
 - services for scan tool access
 - services for the Bosch service diagnostics access
- ➔ ... provides
 - diagnostic data for exchange with OEM or AA ("ODX")
 - testing concept based on ODX
 - automatic generation of documentation

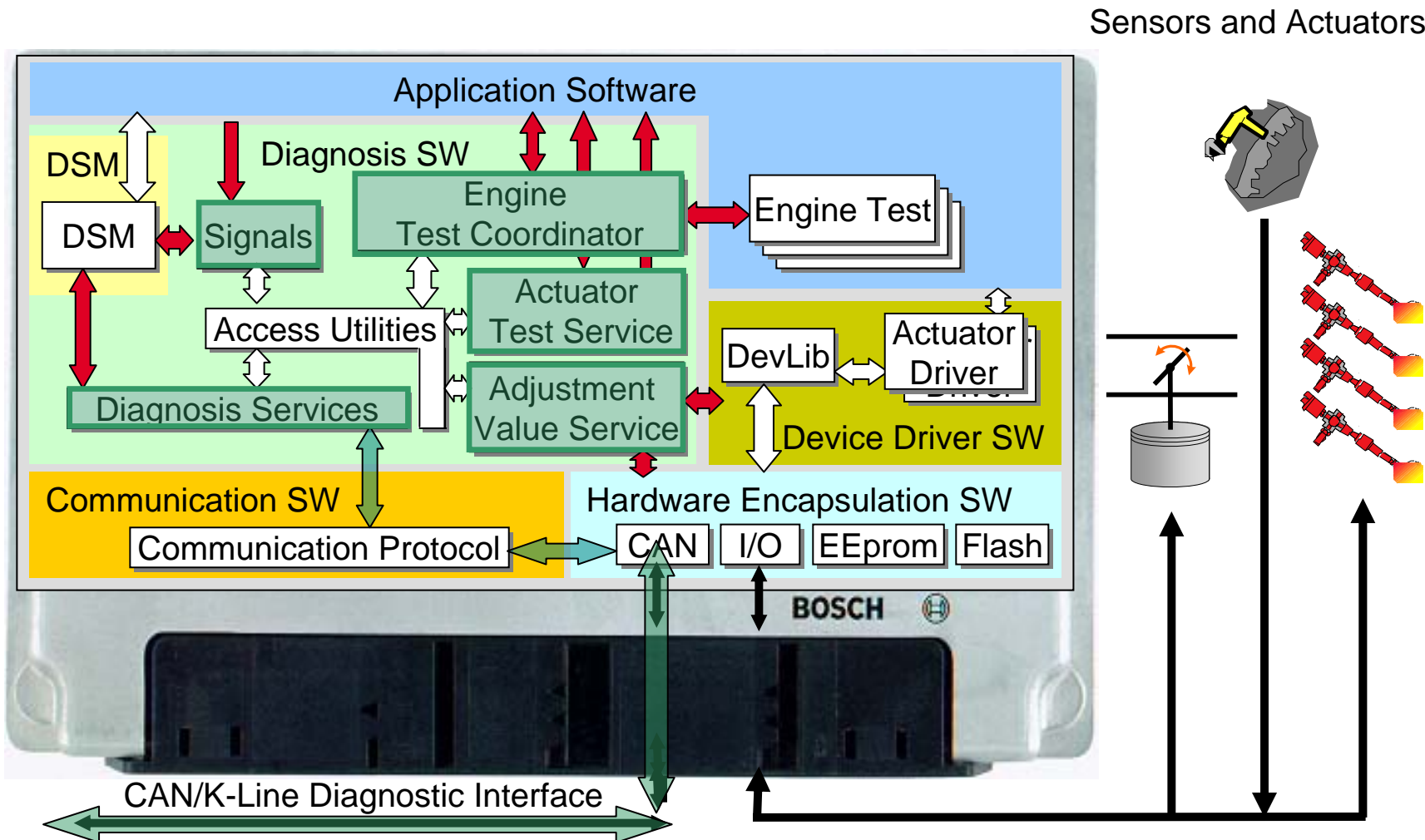


General: DiagInf - Components





Diagnostic Infrastructure Training Course

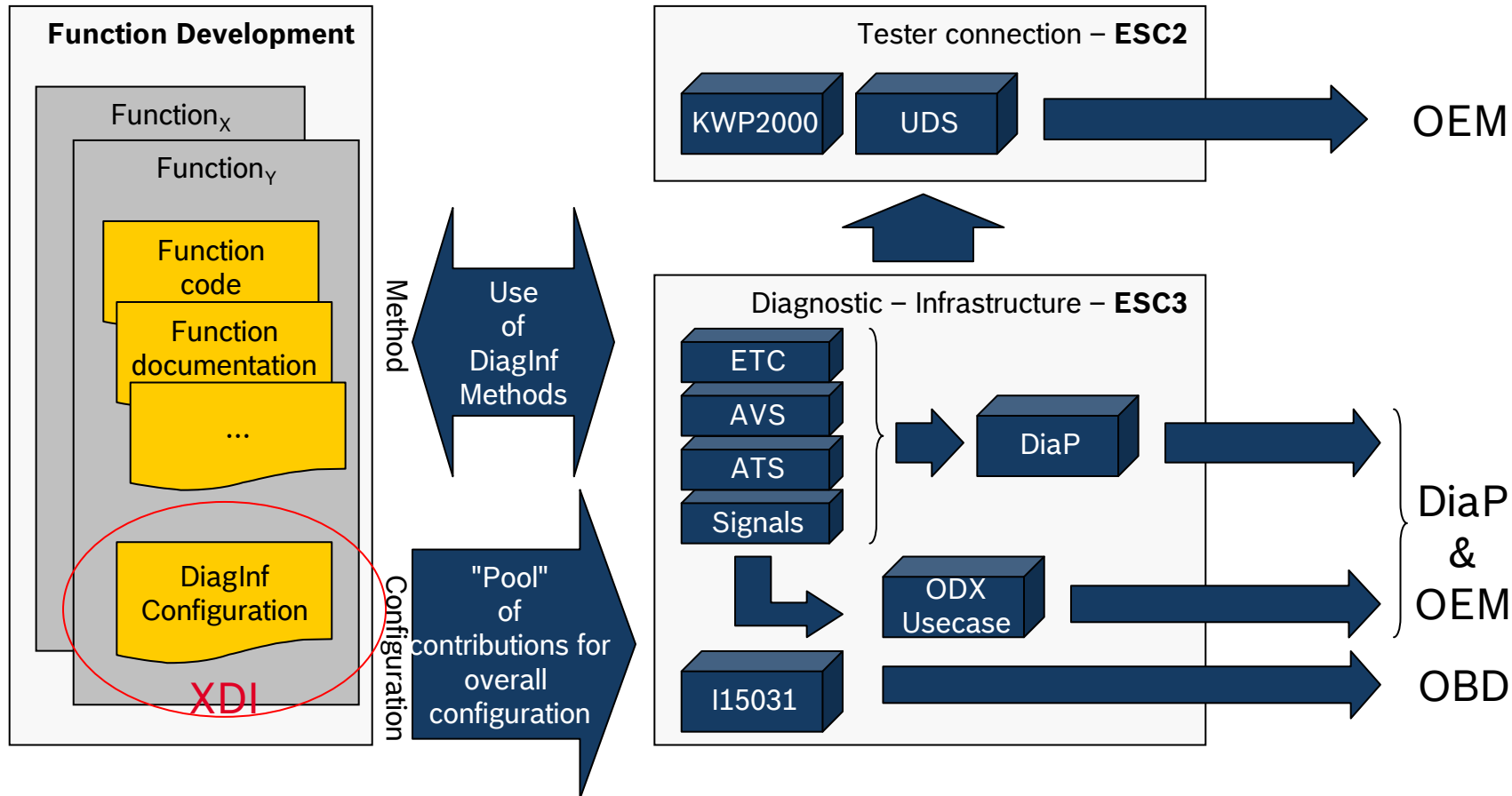


Diesel Gasoline Systems



BOSCH

Partitioning within the Overall System





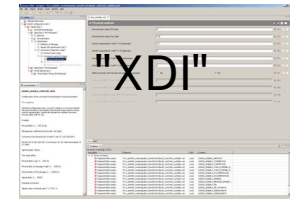
Process Steps for Diagnostics

Work steps/products from the user's view

1. Configuration [FD]

based on "XDI" with the " BCT" tool

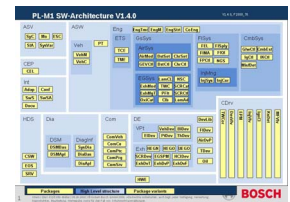
Aim: Specification



+

2. Integration of the software interfaces of DiagInf into the ECU software [FD]

Aim: Functional use of the services



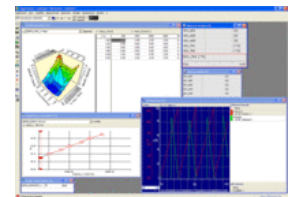
+

3. Connection to the diagnostic services [ESC2]

Aim: Connection of the diagnostic function to the tester

4. Possible calibration [App/]

Aim: Setting the parameters, variant handling





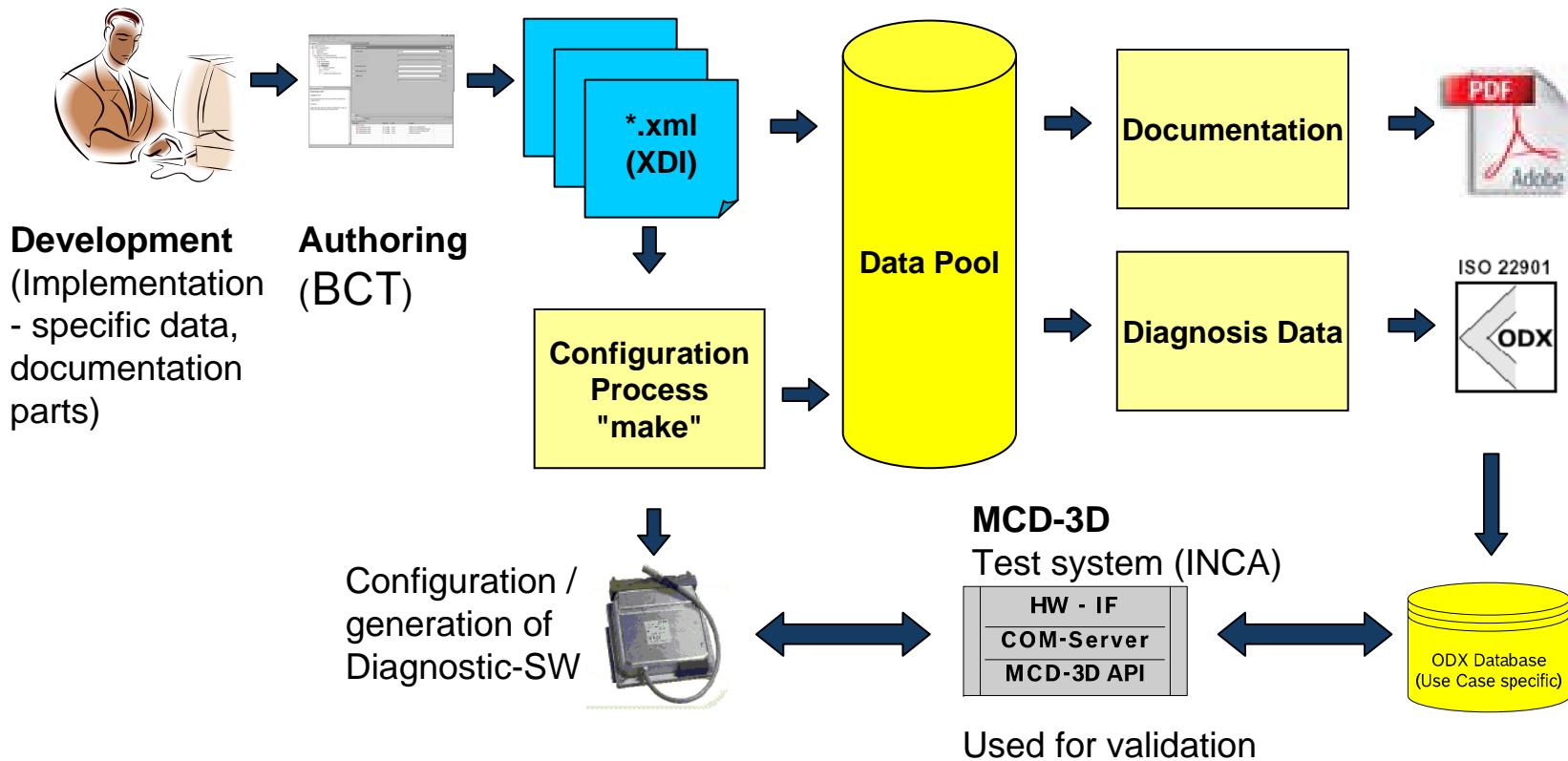
Configuration - General

- General notes
 - "Configuration": parameterization of components in SW
 - "Calibration": parameterization of components outside SW (⇒ Dataset)
- Method for DGS
 - XDI based on XML
- Tooling
 - BCT for editing these XML files
- Purpose
 - Abstraction of data
 - Automatic processing (e.g. documentation)
 - Automatic code generation
 - Automatic plausibility check



Configuration - General

→ Constant data flow





Configuration – Results & Validation

→ **Report file(s) & output of DGS SW-Analyser must be checked**

→ **Content:**

- correctly configured items
- errors
- warnings

→ **Location of report files:**

- *SWB for Diesel projects:* /log/coreproc/XXX_report.txt
- *Make for Gasoline projects:* /makeout/core/reports/XXX_report.txt
- **DGS Build:** /log/coreproc/XXX_report.txt

→ **XXX->** Name of the component (ATS, AVS, ETC, Signals)



Agenda

- Chapter 1 General: DiagInf
 - Function
 - Configuration (BCT)

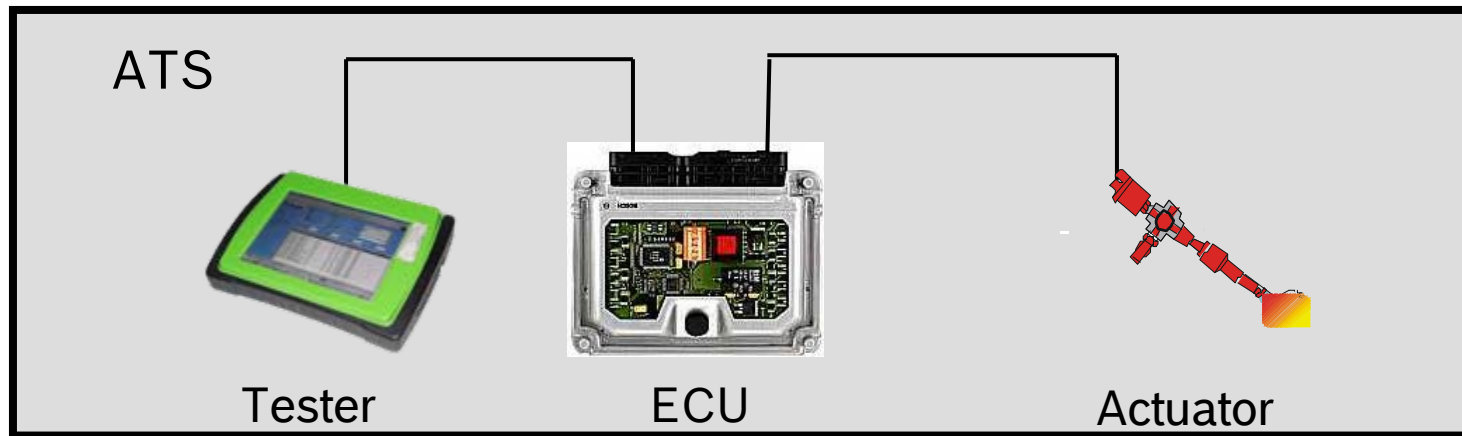
- **Chapter 2 Service diagnostics**
 - **ATS (Advanced Test Service) – actuators**
 - AVS (Adjustment Value Service) – adjustments
 - ETC (Engine Test Coordinator) – engine tests
 - Signals

- Chapter 3 OBD – Diagnostics
 - Signals/OBD
 - I15031



Overview

- ATS is used for tester access to actuators and sensors.
Via ATS, the tester influences those by setting new values.
- ATS = **A**dvanced **T**est **S**ervice





Functionality

- The ATS handles tester requests for actuators or sensors.
- Limitation of those requests can be achieved. For this,

- the intervention time
- engine speed
- vehicle speed
- zero engine speed limitation
- brake pedal shut off
- accelerator pedal shut off

will be monitored by ATS.

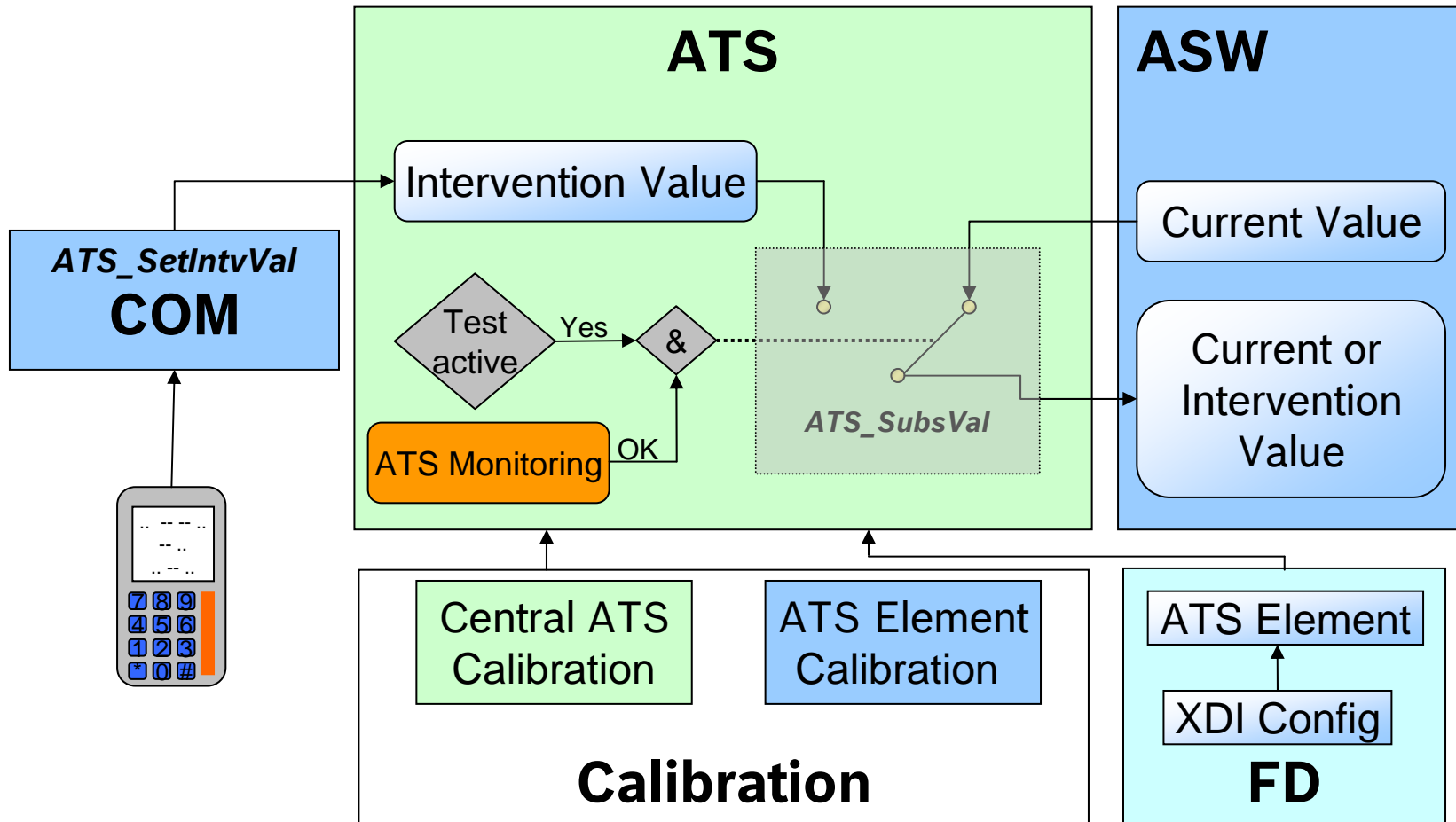
- CM (Configuration Management) storage:

- ClearCase: medc17/core/inf/ats
- Nestor: COCOMP : ATS / CONFIG: ATS
- SDOM: BC :DiaBas / MC : ATS



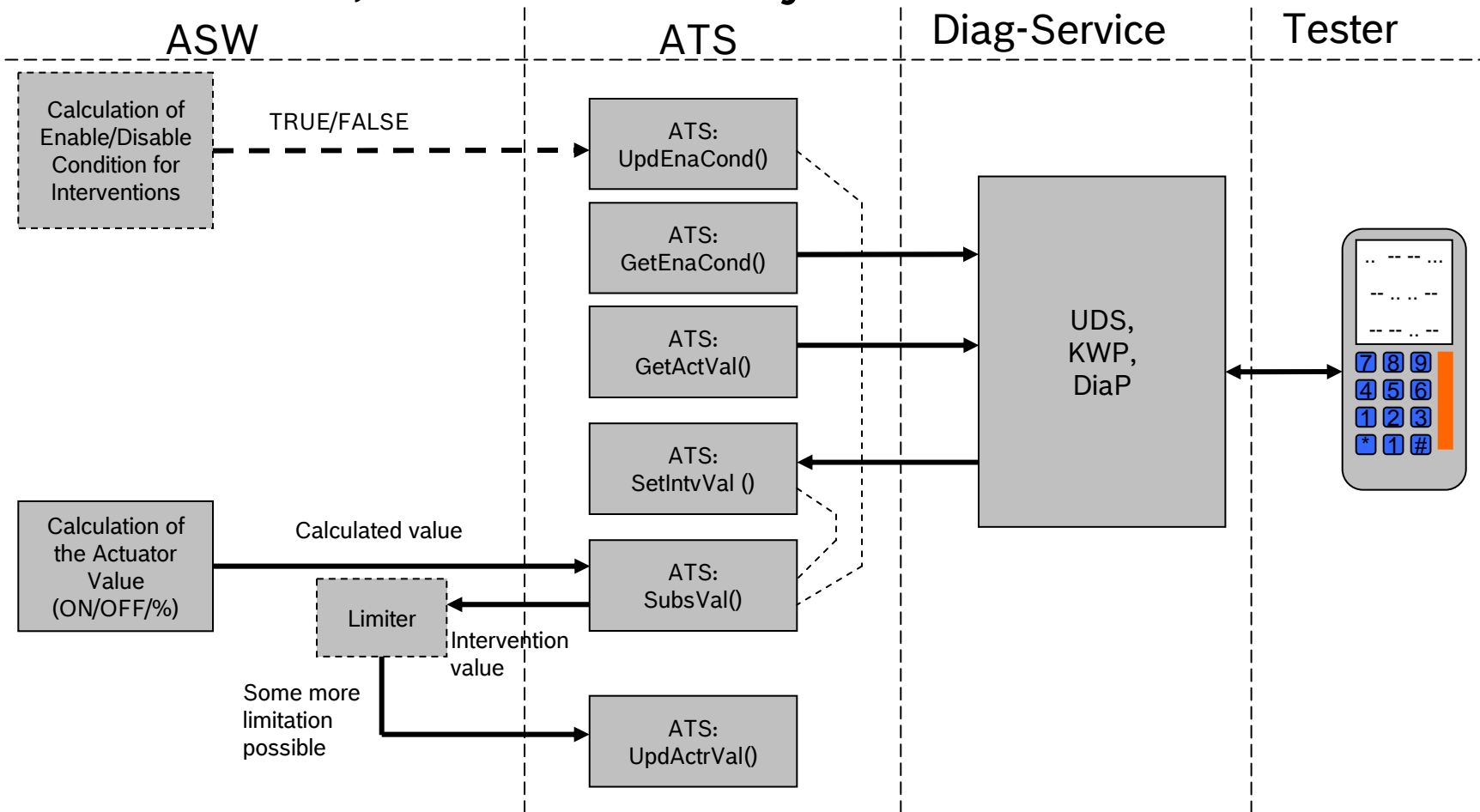


Data Flow, Interfaces in System



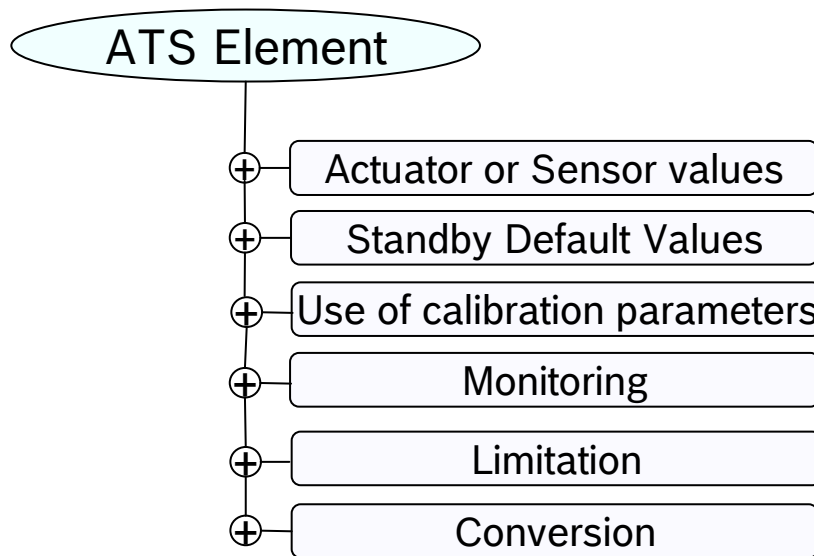


Data Flow, Interfaces in System





Features



Features:

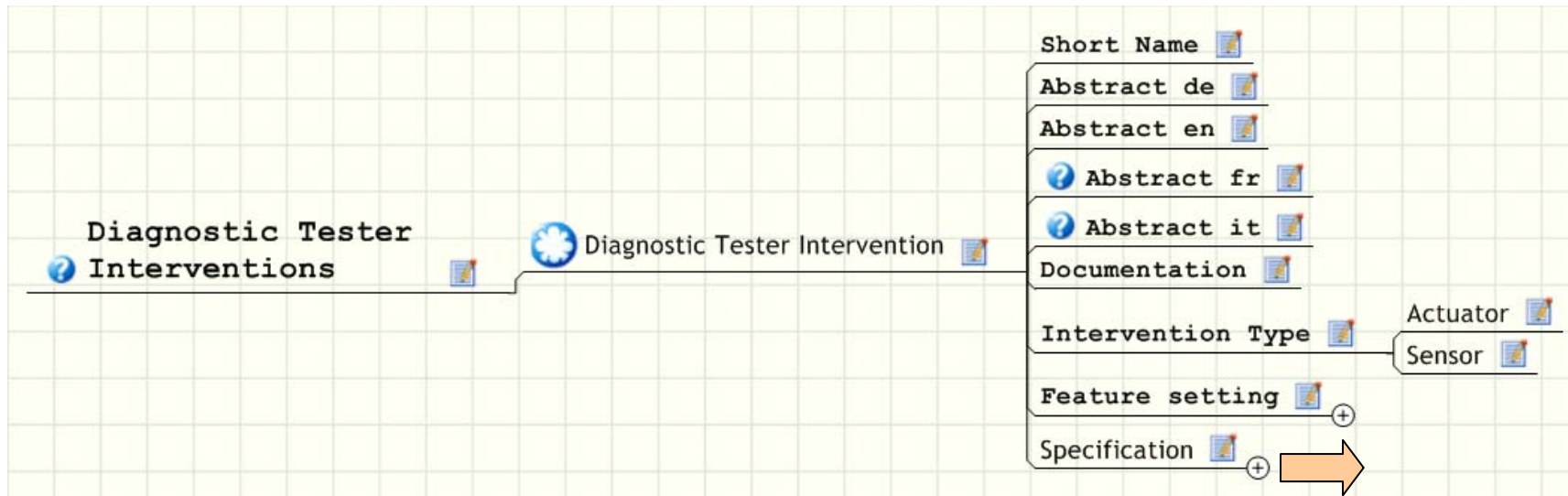
- Actuator (*access to actuator*)
- Sensor (*access to sensor*)
- Calibration of certain attributes
- Standby Default Values
(values for end of line test)
- Monitoring of conditions
- Conversion and limitation of
actuator values





Configuration MEDC17, XDI

→ Modul feature tree and options

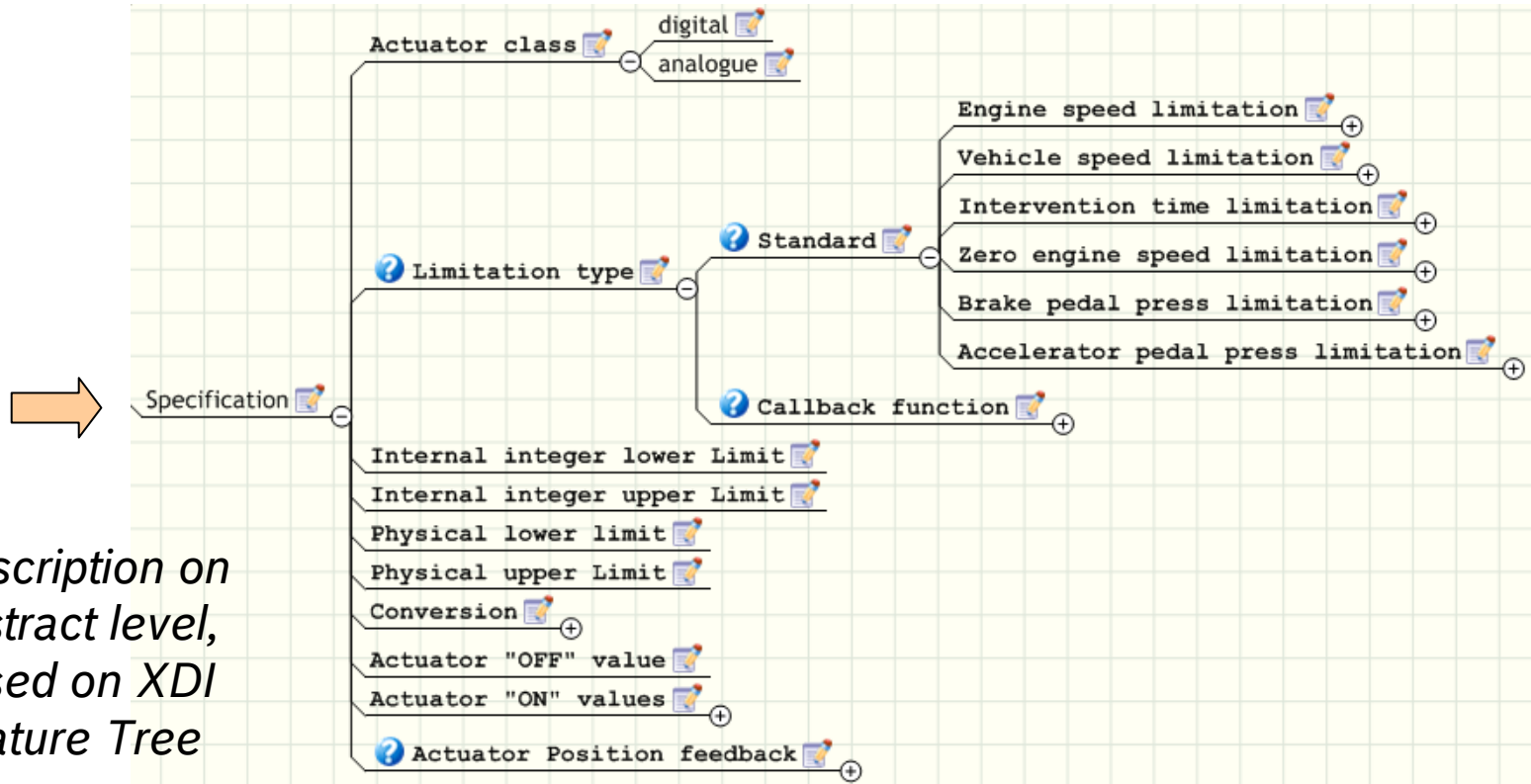


*Description on abstract level, based on
XDI Feature Tree*



Configuration MEDC17, XDI

→ Modul feature tree and options



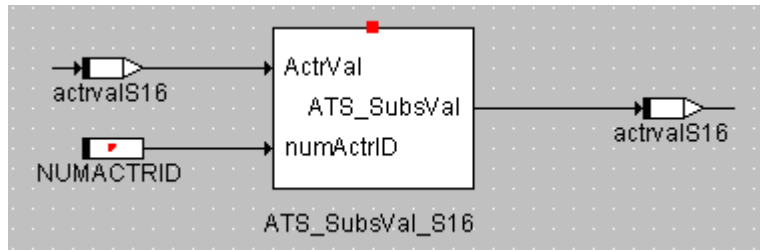
*Description on
abstract level,
based on XDI
Feature Tree*





Interface Description

void **ATS_SubVal** (sint16 *ActrVal, uint8 numActrID)



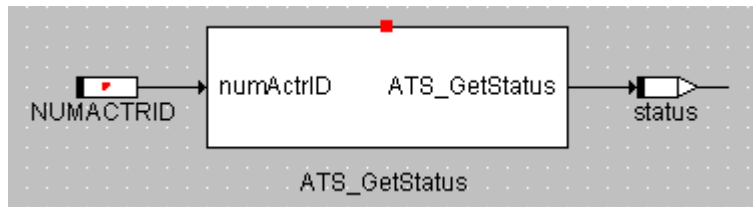
Substitutes the actuator value with the intervention value if the test is active

- Input:
ATS element ID
- Return value:
actual value of ATS element



Interface Description

uint8 **ATS_GetStatus** (uint8 numActrID)



Returns the current status of an actuator

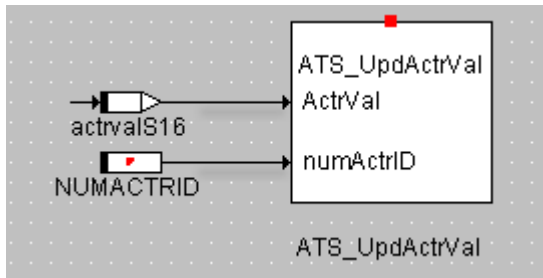
- Input:
ATS element ID
- Return value:
state of test





Interface Description

void **ATS_UpdActrVal** (sint16 ActrVal, uint8 numActrID)



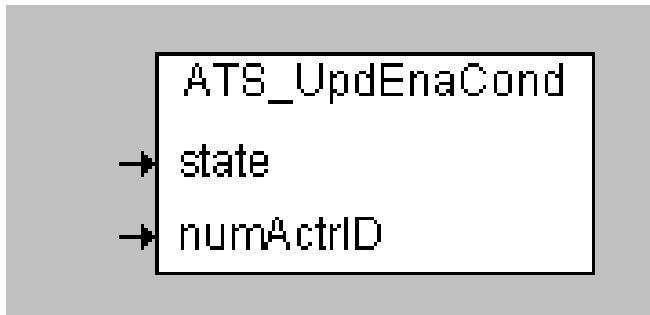
Stores the actuator value in the ATS if changed by the ASW function after intervention.

→ Input:
ATS element ID
Actuator value



Interface Description - C-Code

void **ATS_UpdEnaCond** (bool stEnable, uint8 numActrID)



Tells the ATS the enable status of the actuator function.

- Input:
ATS element ID
- State:
intervention is possible or not



Test Procedure

→ **Provided:**

- Actuator test XYZ

→ **Sought:**

- How can XYZ be tested?
- Test driver is available in SDOM. It can be used to test the various ATS interfaces without a tester.
- This is available under **COMP-A: DevInt \ AC-A: TESTCD**
BC: TESTCD_DiaBas_ASW
|_ FC: TESTCD_ATS_ASW ----->for testing actuators
|_ FC: TESTCD_ETC_ASW
|_ FC: TESTCD_AVS_ASW
- Integrate BC: TESTCD_DiaBas_ASW into PVER that needs to be tested for ATS functionality, and use INCA for testing.



Test Procedure

- Use the appropriate calibration parameters to select the interface that needs to be tested and also the parameters for a particular interface.
- TstDrv_Atts_Trigger_C is used to trigger the ATS interface.
- TstDrv_Atts_ActrVal16_C is used to specify the intervention value for an actuator
- TstDrv_Atts_ActrID_C is used to select the actuator
- TstDrv_Atts_Interface_C is used to select the interface that needs to be tested.



Test Procedure

- List of the interfaces that can be tested and the corresponding measurement points where the result will be updated is as shown below:

ATS_SetIntvVal
ATS_SetDfltVal
ATS_GetActrVal
ATS_GetActrVal8
ATS_GetEnaCond
ATS_GetActrValInt
ATS_GetActrValInt8
ATS_GetStatus

TstDrv_Ats_Status_mp
TstDrv_Ats_RetVal16_mp
TstDrv_Ats_RetVal_mp
TstDrv_AtsGetEna_Status_mp

Further information about usage of test driver is available in spec folder under FC: TESTCD_ATS_ASW

Test Procedure

→ Configuration check

- The test of the actual actuator test configuration is carried out at the time of FCI (**F**unctional **C**omponent **I**mplementation test).
- Here, attention must be paid to the correct configuration of:
 - Limits
 - Limit Type Mask for ATS monitoring





Interface Description for Calibration

→ Description of global calibration parameters

- There are six global calibration parameters for ATS
- These settings are valid for all actuators
 - ATS_TstDemMaxEngN_C (engine speed)
 - ATS_TstDemMaxVSSCDV_C (vehicle speed)
 - ATS_tiTstDemMax_C (intervention time)
 - ATS_TmrBehaviour_C (timer behaviour)
 - ATS_tiTstDemHeal_C (healing time)





Interface Description for Calibration

→ Description of actuator-specific calibration parameters

- There are some actuator-specific calibration parameters
 - These parameters will be generated optionally with the feature setting "Generate calibration parameters"
 - <ATS element name> indicates the actuator name
 - <ATS element name>.LimitTypeMsk_C (monitoring switches)
 - <ATS element name>.LowLim_C
 - <ATS element name>.UpLim_C
 - <ATS element name>.CnvOfs_C
 - <ATS element name>.CnvFac_C
 - <ATS element name>.CnvNorm_C
 - <ATS element name>.DfltVal_C
- } actuator limits
- } conversion internal-tester
- (standby test value)



Hints for Integration

→ No special hints for integration necessary.





Process

- **CE departments**
 - Ordering of ATS elements [FD]
 - Ordering of necessary changes in communication layer [ESC2]
- **Function development**
 - Realisation of functions
 - Configuration of ATS elements
 - Testing
- **Calibration**
 - If an ATS element is configured to be calibrateable, the calibration engineer can change limits, default values, parameters of conversion from internal value to tester value and activation of conditions for start permission
 - Testing



Process

- **Responsibility**

- Function development is responsible for the correctness of the ATS element configuration and implementation.
- Calibration department is responsible for correct calibration of all parameters that are made calibrateable by FD.
- CE department is responsible for completeness

- **Test concept (MCD-3D)**

Use of MCD-3D to test single ATS elements

- Garage: Testing with customer-specific services and DiaP

Agenda

- Chapter 1 General: DiagInf
 - Function
 - Configuration (BCT)

- Chapter 2 Service diagnostics
 - ATS (Advanced Test Service) – actuators
 - **AVS (Adjustment Value Service) – adjustments**
 - ETC (Engine Test Coordinator) – engine tests
 - Signals

- Chapter 3 OBD – Diagnostics
 - Signals/OBD
 - I15031





Overview

What are adjustment values and what purpose does the AVS serve?

- Adjustment values are parameters which can be used to permanently influence the behaviour of software functions.
- Adjustment values can be changed at any time via external access (service diagnostics tester).
- Example:
 - *Maximum speed limitation for reverse gear (vehicle-specific adjustment)*
 - *Injector quantity adjustment (component-specific adjustment of the manufacturing tolerance).*
- The AVS (**A**djustment **V**alue **S**ervice) serves as an interface between the adjustment values and service tester regarding read and write access.





Functional Description

- AVS provides universal access via specified interfaces and generated IDs for each adjustment value.
- Advantages:
 - Standardized yet flexible due to partially optional configuration elements.
 - Documentation and ODX data which can be generated automatically. Thus simple integration into the test process.
- Encapsulates access to EEPROM
- CM (Configuration Management) storage:
 - ClearCase: medc17/core/inf/avs
 - Nestor: COCOMP : AVS / CONFIG: AVS
 - SDOM: BC :DiaBas / MC : AVS





Functional Description, Storage Features

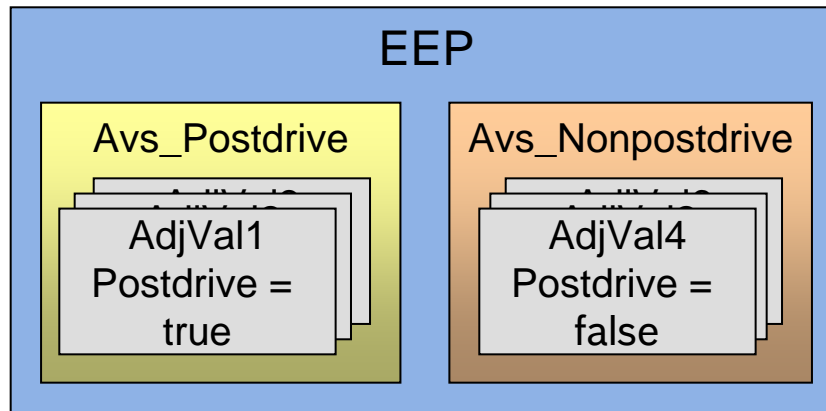
→ Non-volatile adjustment values

- Automatically written back in postdrive
- Not automatically written back in postdrive (manual store , tester triggered)

→ Postdrive behaviour will be represented by location in different EEP blocks

→ Volatile adjustment values

- Are only kept in RAM variables
- With reset of the ECU, the value is also reset to "0"
- Can be used for temporary calculations

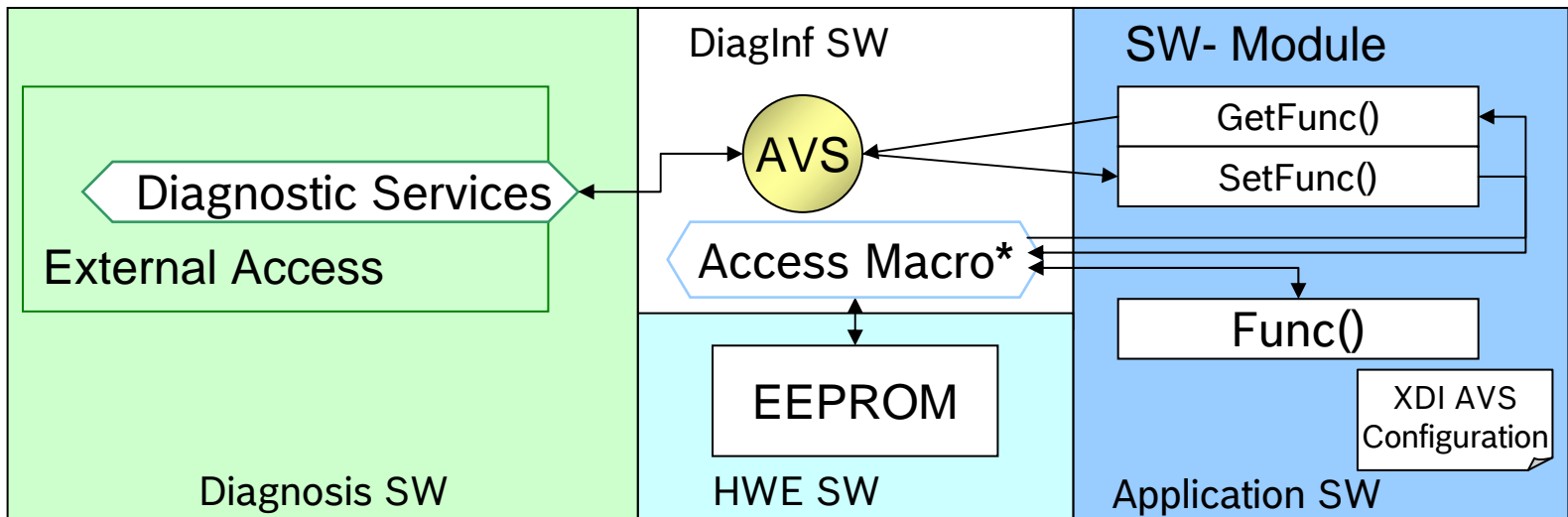




Functional Description, Access Variants and Data Flow

→ „Normal Access“ (High end)

- Values are set to or get from RAM-mirror through a module set- or get-function that allows to convert or plausibilize the input value.



* Called in ASW-process or -function.

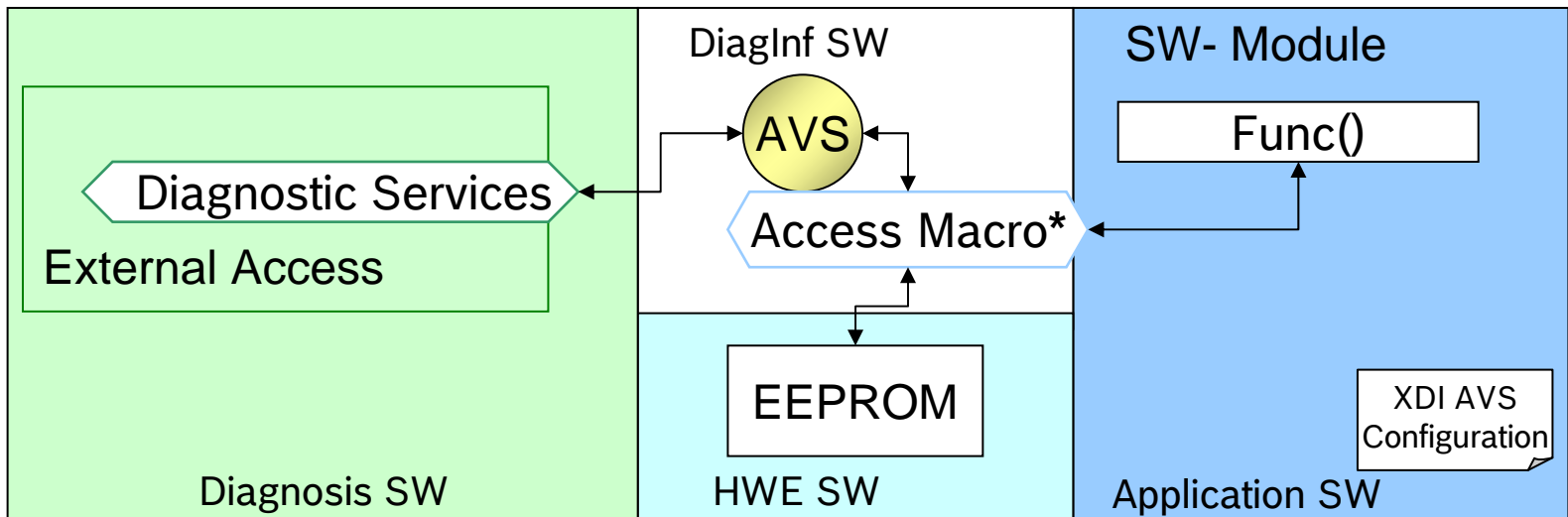




Functional Description, Access Variants and Data Flow

→ „Direct Access“ (Low end)

- Values can be set in or retrieved from the EEPROM mirror directly by AVS without having to use a module set- or get-function.
- „Flag“ type is equal to: Direct access, volatile, length 1 byte.



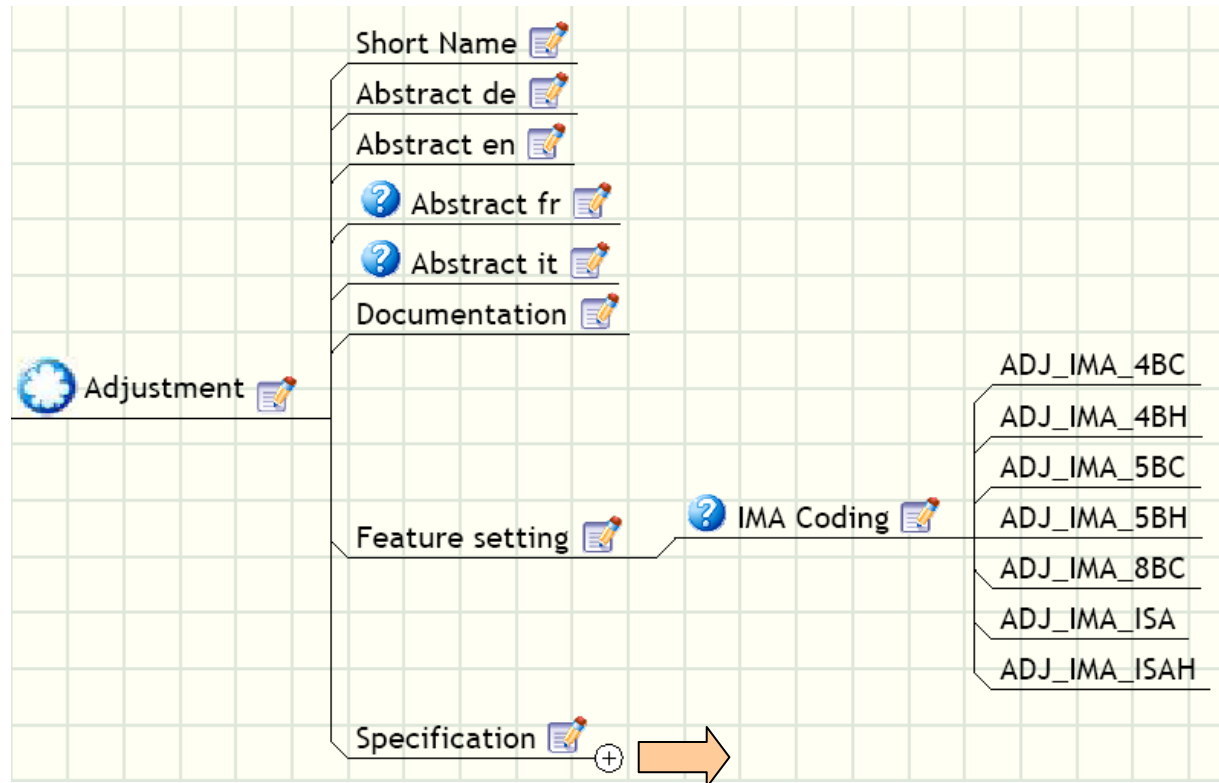
* Called in ASW-process or -function.





Configuration MEDC17, XDI

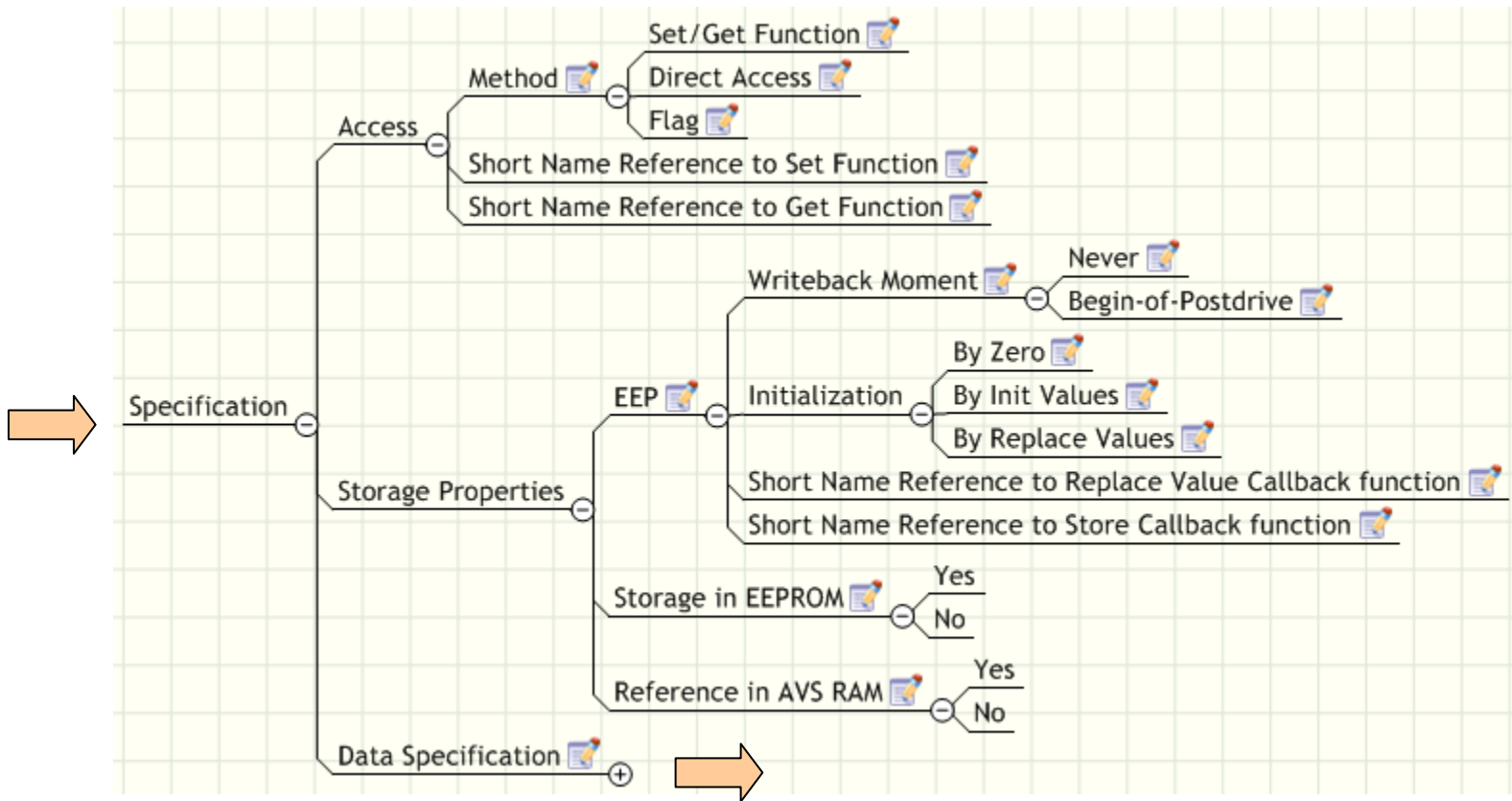
→ Modul feature tree and options





Configuration MEDC17, XDI

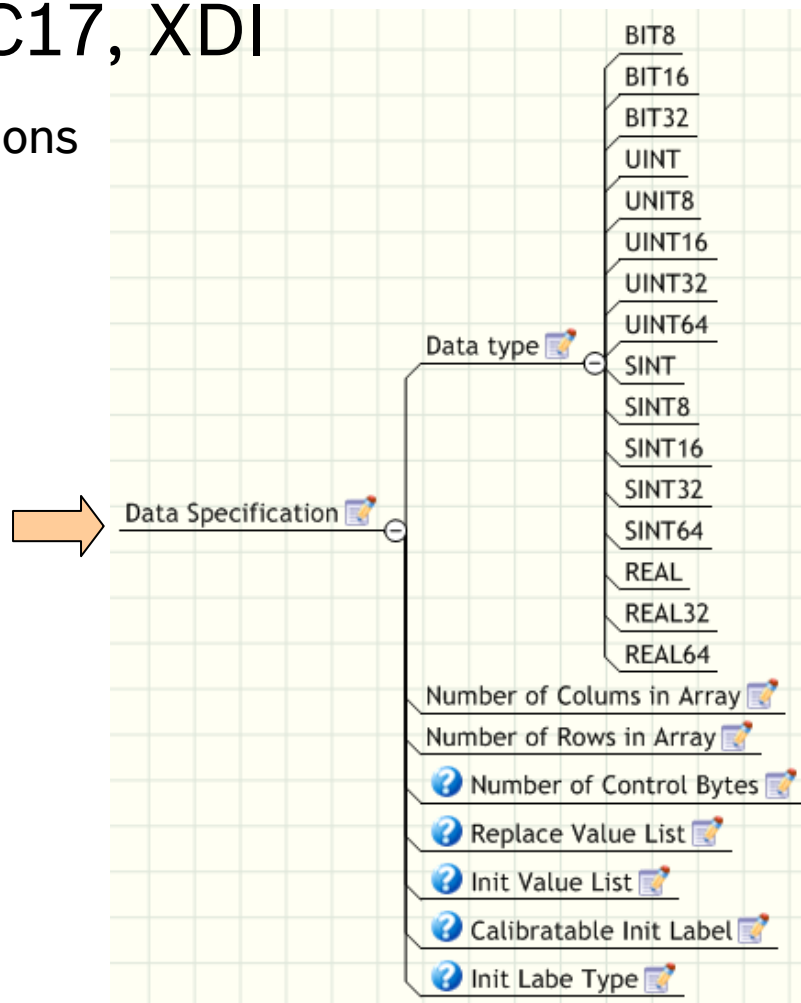
→ Modul feature tree and options





Configuration MEDC17, XDI

→ Modul feature tree and options





Interface Description – C-Code

- **Application SW access to adjustment value through Access Macro**
 - ASW accesses the AVS RAM through an auto generated Access Macro e.g. AVS_STSYS_TRQVAL_S16.

- **Diagnosis SW access to adjustment value through AVS “Normal Access” (High End)**
 - Diagnosis SW accesses the AVS RAM through Set/Get callback functions that are to provide by the ASW module.
 - Within these callback functions the „real“ access to AVS RAM is to implement again through the respective AVS Access Macro.





Interface Description – ASCET

→ Definition of AVS Access Macro

- The AVS Element Access Macro as the basic element of AVS access is to define in ASCET as System Constant with the following attributes:
 - Name: Name of auto generated AVS Access Macro
 - Rule: AVS_<Name from config>_<datatype>
 - Model Type: According to configuration, uint or sint
 - Scope: Imported (because auto generated by AVS)
 - Calibration: No

Element Editor for AVS_TESTAVSVAL_U16:audisc

Name: AVS_TESTAVSVAL_U16

Unit:

Comment:

Dimension: Scalar

Model Type: ☐ Logic ☐ Signed Discrete ☒ Unsigned Discrete ☐ Continuous ☐ Enumeration

Kind: ☐ Constant ☒ System Constant ☐ Parameter ☐ Variable ☐ Input ☐ Output

Scope: ☐ Local ☒ Imported ☐ Exported

Existence: ☐ Virtual ☒ Non-Virtual

Dependency: ☐ Dependent ☒ Independent

Memory: ☐ Volatile ☒ Non-Volatile

Calibration: ☐ Yes ☒ No

☒ Always show editor for new elements

OK Cancel

Formula

☐ Variants ☐ Set() ☐ Get()

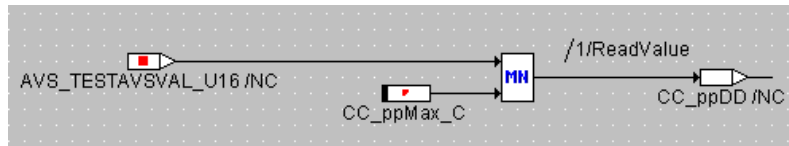




Interface Description – ASCET

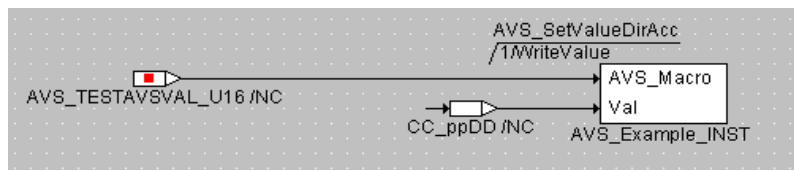
→ Application SW access to adjustment value through Access Macro

- Read Access:



- This example demonstrates how to read minimum of adjustment value (Access Macro AVS_TESTAVSVAL) and CC_ppMax_C to variable CC_pp_DD.

- Write Access:



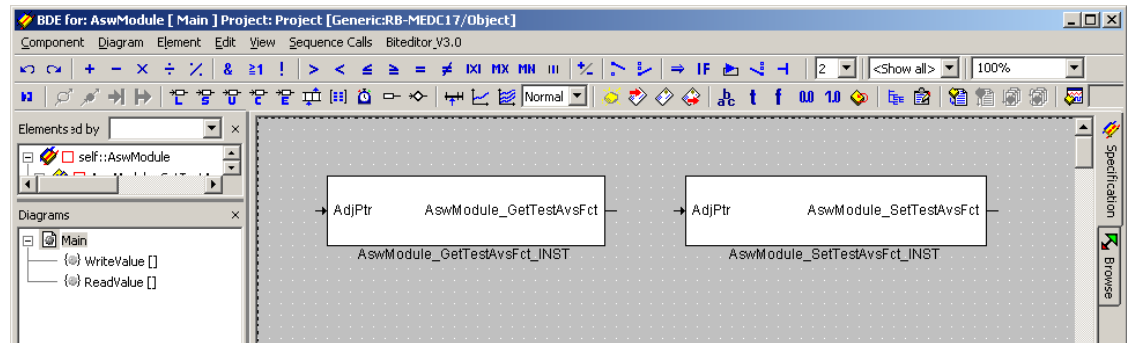
- This example demonstrates how to write variable CC_ppDD to adjustment value (Access Macro AVS_TESTAVSVAL).
- For access of bigger adjustment values (arrays) the methods are provided accordingly.





Interface Description - ASCET

- ➔ **Diagnostic SW access to adjustment value through AVS “Normal Access” (High End)**
 - For each Set/Get callback function an own ASCET class has to be defined.
 - Within these callback functions the „real“ access to AVS RAM is to implement again through the respective AVS access methods as described for the AVS RAM access in ASW.
 - Remark: To avoid the unwanted clean up of the callback class instances they must be placed in the model:





Test Procedure

→ **Provided:**

- AVS element XYZ

→ **Sought:**

- How can XYZ be tested?
- Test driver is available in SDOM. It can be used to test the various AVS interfaces without a tester.
- This is available under **COMP-A: DevInt \ AC-A: TESTCD**
BC: TESTCD_DiaBas_ASW
|_ FC: TESTCD_ATS_ASW
|_ FC: TESTCD_ETC_ASW
|_ FC: TESTCD_AVS_ASW ----- > for AVS test
- Integrate BC: TESTCD_DiaBas_ASW into PVER that needs to be tested for AVS functionality, and use INCA for testing.



Test Procedure

- Use the appropriate calibration parameters to select the interface that needs to be tested and also the parameters for a particular interface.
- TstDrv_Avs_Trigger_C is used to trigger the AVS interface.
- TstDrv_Avs_numBytes_C is used to specify the number of adjustment value in bytes.
- TstDrv_Avs_idxAdjVal_C is used to select the adjustment value.
- TstDrv_Avs_Interface_C is used to select the interface that needs to be tested.



Test Procedure

- List of the interfaces that can be tested and the corresponding measurement points where the result will be updated.

Avs_GetCallInfo
Avs_GetEepInfo
Avs_GetInfo
Avs_GetStatus
Avs_GetValue
Avs_GetCtlSize
Avs_GetValueSize
Avs_SetValue
Avs_StoreValue
Avs_GetValueType

TstDrv_Avs_Status_mp
TstDrv_Avs_GetInfoStatus_mp
TstDrv_Avs_GetStatus_mp
TstDrv_Avs_RetVal_mp
TstDrv_Avs_SetStatus_mp
TstDrv_Avs_StoreStatus_mp
TstDrv_Avs_RsltBuf_mp
TstDrv_Avs_RetVal_mp

Further information about usage of test driver is available in spec folder under
FC: TESTCD_AVS_ASW



Test Procedure

→ Configuration check

- The test of the AVS element configuration is carried out at the time of FCI (**F**unctional **C**omponent **I**mplementation test).
- Attention must be paid particularly to the correct configuration of:
 - Volatile/Non-Volatile
 - Block size and Data type
- Moreover all Warnings/Remarks are to consider.



Hints for Integration

→ System constants:

- Normally no system constants must be set.
- However a system constant DIABAS_AVS_SY is provided to disable AVS functionality completely, including DiaP service \$7E (AVS disabled: DIABAS _AVS_SY = 0).





Process

- **CE departments**
 - Ordering of AVS values [FD]
 - Ordering of necessary changes in Communication layer [ESC2]
 - Creation of the master configuration
- **Function development**
 - Configuration of adjustment values
 - Testing
- **Calibration**
 - Where applicable, calibration of first init value
 - Testing



Process

- **Responsibility**

- The function developer (FD) is responsible for the correctness of his/her adjustment value (Length, Data type, Access type etc.).
- The CE department is responsible for completeness

- **Test concept (MCD-3D)**

- Using MCD-3D to test individual adjustment values (DiaP, ODX-Link)

- **Alternative test options:**

- Testing with a customer-specific service tester
- Testing with SamDia/CANalyser





Diesel Gasoline Systems

Confidential | DGS-EC/ESC3 | 02.12.2008 | © Robert Bosch GmbH reserves all rights even in the event of industrial property rights. We reserve all rights of disposal such as copying and passing on to third parties.



BOSCH

Agenda

- Chapter 1 General: DiagInf
 - Function
 - Configuration (BCT)

- Chapter 2 Service diagnostics
 - ATS (Advanced Test Service) – actuators
 - AVS (Adjustment Value Service) – adjustments
 - **ETC (Engine Test Coordinator) – engine tests**
 - Signals

- Chapter 3 OBD – Diagnostics
 - Signals/OBD
 - I15031



Overview

- ETC is part of the functionality of the diagnosis infrastructure, which is used to coordinate engine tests.
- An engine test is a function that executing diagnostic tests on the engine.
- Abbreviations:
 - ET = **E**ngine **T**est
 - ETC = **E**ngine **T**est **C**oordinator
- CM (Configuration Management) storage:
 - ClearCase: medc17/core/inf/etc
 - Nestor: COCOMP : ETC / CONFIG: ETC
 - SDOM: BC :DiaBas / MC : ETC



Functionality

- ECU will have engine tests, which are used to diagnose engine failures.
- These tests aren't running in ECU drive mode and must be activated using a tester.
- ETC is responsible for coordination these engine tests.
- Engine tests need certain environmental conditions for being able to run
- When coordinating engine tests, the ETC also considers the mutual exclusion of some engine test.



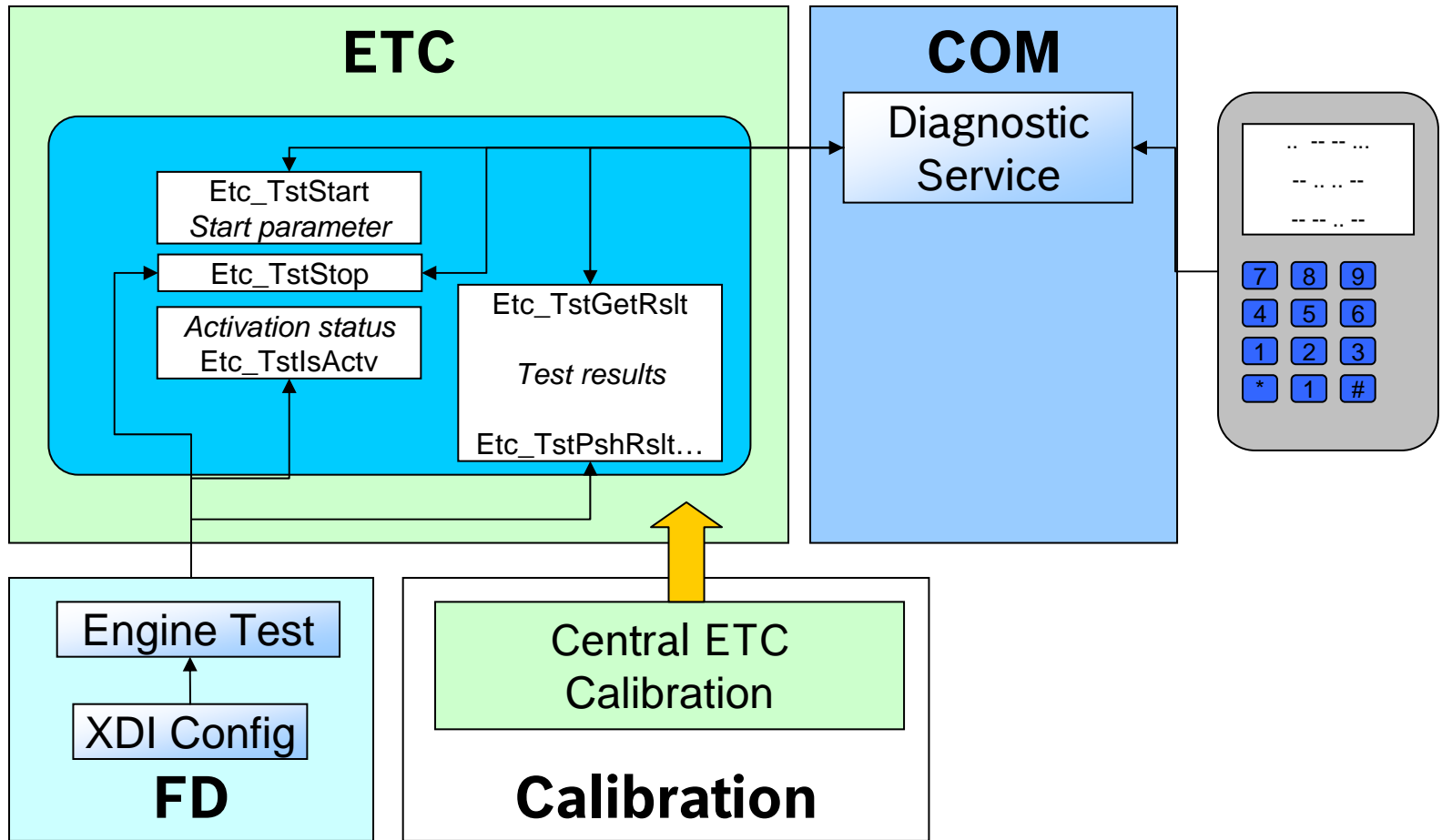
Function description, Modul Features

- Every engine test that is available in the ECU is registered using the configuration of the ETC and receives an ID
- Using this ID and the appropriate interfaces of the ETC, a start- or stop request can be sent
- The ETC checks beforehand mentioned conditions and gives the start permission to a test - if possible
- Each engine test is allowed to deliver test results to the ETC.
The ETC will store those and the tester may fetch them from the ETC

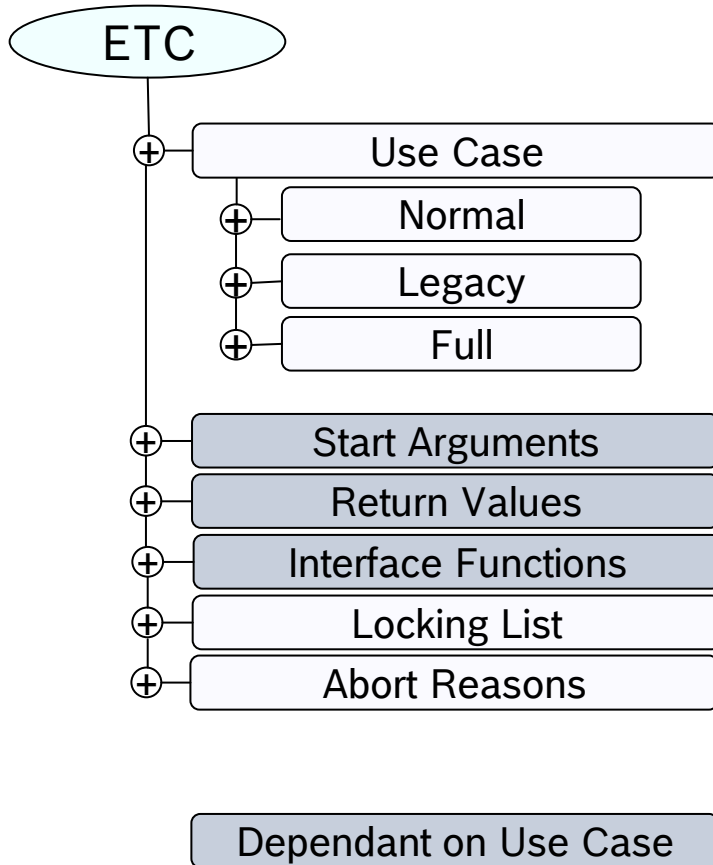




Data Flow, Interfaces in System



Features

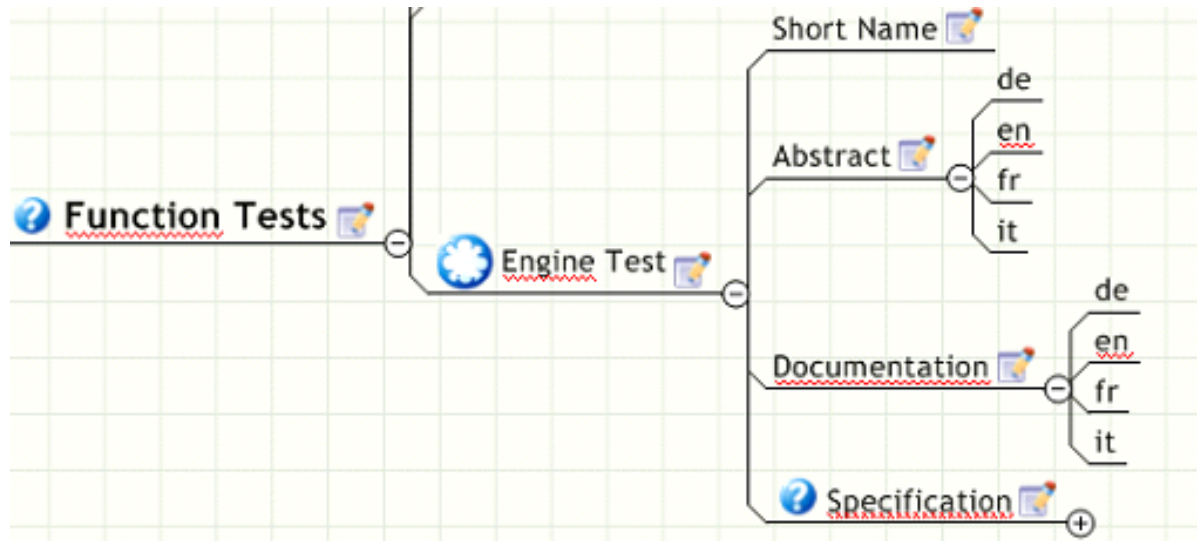


Features of each class:

- **Normal**
Interface for start permission, status and abort info is done using messages.
No start arguments or return values.
- **Legacy**
Interface (as above) is done using call back functions.
Start arguments and return values can be used optionally.
- **Full**
Interface (as above) is done using messages.
Start arguments and return values can be used optionally.

Configuration MEDC17, XDI

→ Modul feature tree and options



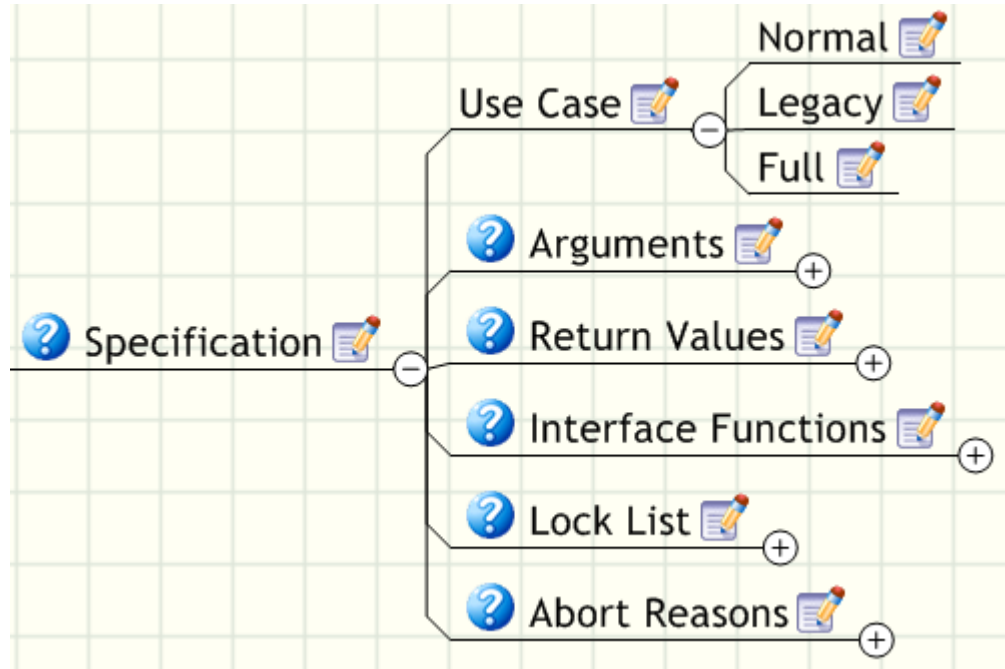
*Description on abstract level, based on
XDI Feature Tree*

Configuration MEDC17, XDI

→ Modul feature tree and options

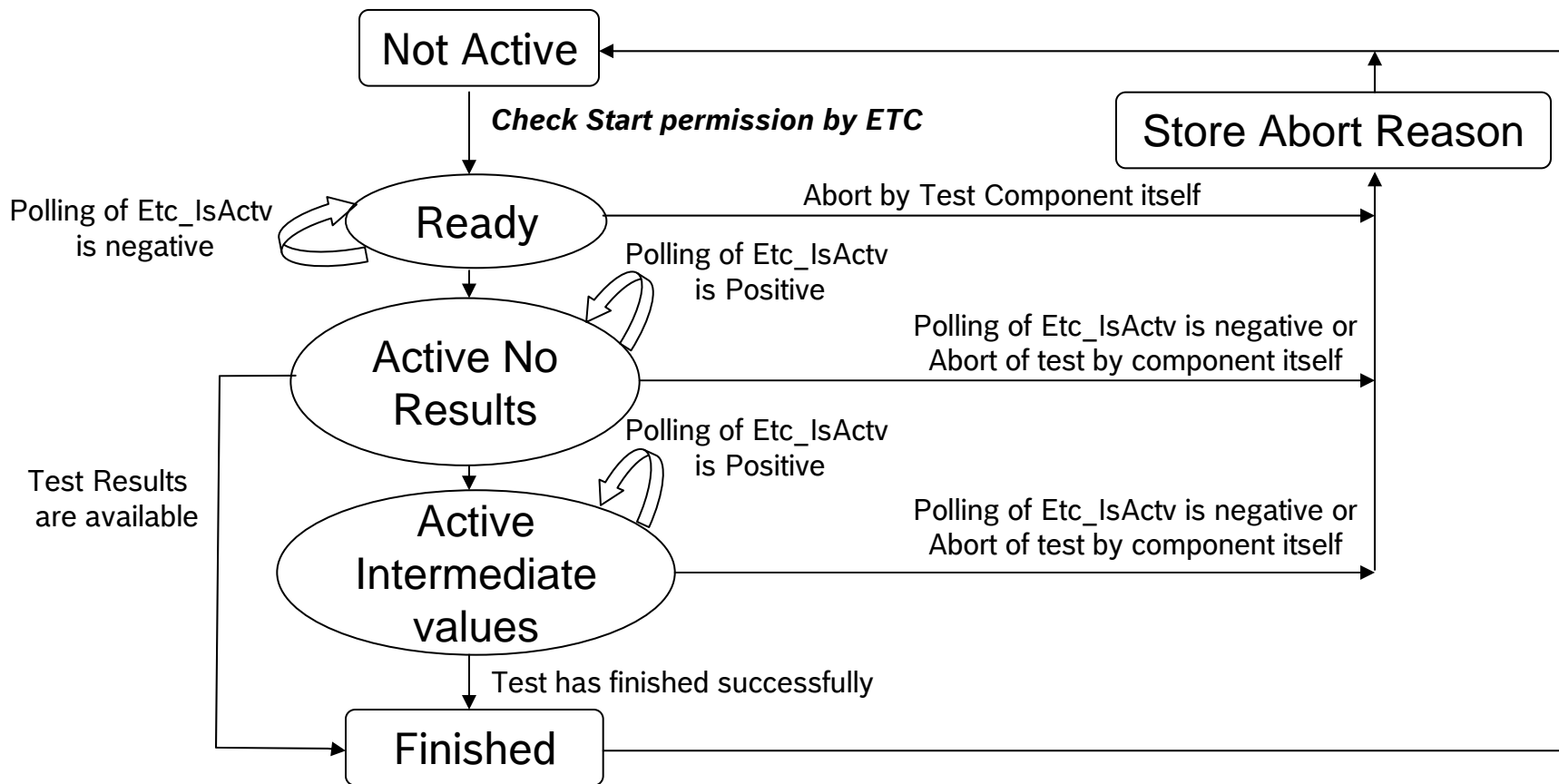


*Description on
abstract level,
based on XDI
Feature Tree*





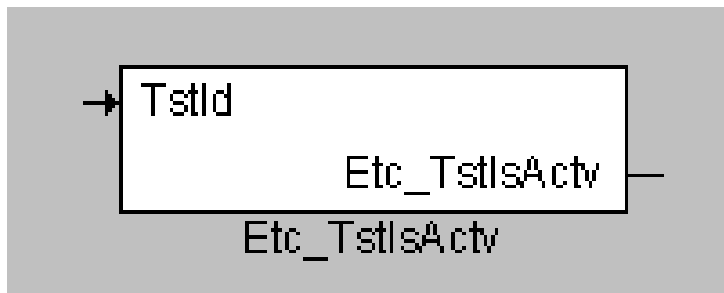
ETC for Function/Software Developers





Interface Description C-Code

```
uint8 Etc_TstIsActv(const Etc_TstId_t *TstId);
```



Get activation permission

The engine test function must permanently fetch its permission to be activated by using this interface.

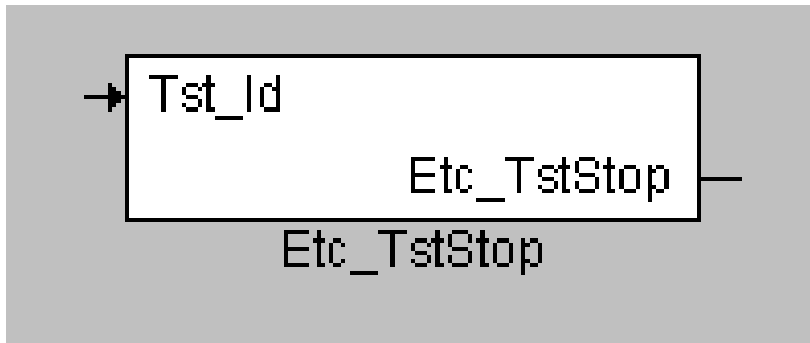
- Input:
ET ID
- Return value:
Activation state of ET





Interface Description C-Code

```
uint16 Etc_TstStop(const Etc_TstId_t *TstId);
```



Stopping the engine test

The engine test needs to stop itself after successful execution or in case of an abortion.

- Input:
ET ID
- Return value:
state of stopping the ET





Interface Description C-Code

```
bool Etc_TstPshRsltSet(const Etc_TstId_t *TstId, uint8 *RsltVal_pu8);
```



Push one result set

This function is for the engine tests only and is used to push one complete result set into the result value buffer.

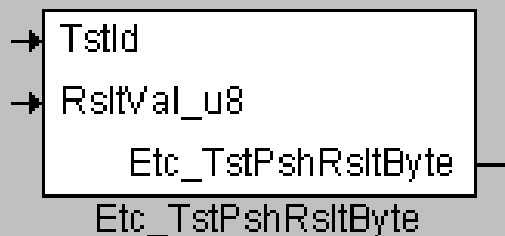
- Input:
 - ET ID
 - Pointer to result set (manual implementation)
- Return value:
 - success of the push action





Interface Description C-Code

bool Etc_TstPshRsltByte(const Etc_TstId_t *TstId, uint8 RsltVal_u8)



Push one result byte

This function is for the engine tests only and is used to push one byte into the result value buffer.

- Input:
 - ET ID
 - Byte that will be pushed
- Return value:
 - success of the push action



Test Procedure

→ Provided:

- Engine test XYZ

→ Sought:

- How can XYZ be tested?

Test driver is available in SDOM which replaces service tester and it should be used only to check engine test, by using required ETC interfaces

This is available under **COMP-A: DevInt \ AC-A: TESTCD**

BC: TESTCD_DiaBas_ASW

|_ FC: TESTCD_ATS_ASW

|_ FC: TESTCD_ETC_ASW -----> for testing engine test

|_ FC: TESTCD_AVS_ASW

- Integrate BC: TESTCD_DiaBas_ASW into PVER in which engine test needs to be tested, and use INCA for testing



Test Procedure

- Use of calibration parameters to select an engine test and to perform an action on engine test
- *TstDrv_Etc_TstID_C* is used to select an engine test
- *TstDrv_Etc_Order_C* to perform an action on engine test



Test Procedure

- For testing the flow of an engine test, select the option “*ETC_RUN*” in *TstDrv_Etc_Order_C* which will perform following actions:
 - Starts an engine test
 - Updates status of engine test continuously
 - Updates abort reason when engine test is aborted
 - Updates test Results to tester, once test is finished or intermediate results are available



Test Procedure

- Each action can be performed individually using TstDrv_Etc_Order_C, and these options will be useful for debugging
- TstDrv_Etc_Order_C

ETC_START
ETC_STOP
ETC_GETSTATUS
ETC_GETABORT
ETC_GETRSLT

TstDrv_Etc_RetVal_TstStrt_mp
TstDrv_Etc_RetVal_TstStop_mp
TstDrv_Etc_RetVal_TstGetSt_mp
TstDrv_Etc_RetVal_AbrtReason_mp
TstDrv_Etc_RetVal_TstGetRslt_mp

- Status of all the above actions will be updated in verbal form in corresponding measurement points as shown in the table above
- Further information about usage of test driver is available in spec folder under FC: TESTCD_ETC_ASW

Test Procedure

→ Configuration check

- The test of the actual engine test configuration is carried out at the time of FCI (**F**unctional **C**omponent **I**mplementation test).
- Here, attention must be paid to the correct configuration of:
 - Are the environmental conditions whilst start request checked properly
 - Is the engine test activated correctly and running fine?
 - Will be engine test stop itself after completion?
 - Is the tester able to fetch the results?



Interface Description for Calibration

→ Description of global calibration parameters

- Etc_MaxReqTO_C
 - Maximum time between two service tester request (at ETC) before currently running engine tests will be aborted
 - The resolution of Etc_MaxReqTO_C is 10ms/Bit
 - The init value of Etc_MaxReqTO_C is set to 500, which means 5s



Hints for integration

→ System constants:

- Normally no system constants must be set
- However a system constant ETC_Act_SY is provided to disable ETC functionality completely, including DiaP service \$7D (ETC disabled: $\text{ETC_Act_SY} = 0$)



Process

- **CE departments**
 - Ordering of customer-specific engine tests [FD]
 - Ordering of necessary changes in Communication layer [ESC2]
- **Function development**
 - Realization of an engine test
 - Configuration of an engine test
 - Testing
- **Calibration**
 - Where applicable calibration of start parameter
 - Calibration of tester communication time out
 - Testing



Process

- **Responsibility**
 - Function development is responsible for the correctness of the ETC configuration
 - CE department is responsible for completeness
- **Test concept (MCD-3D)**
 - Use of MCD-3D to test single ETC elements
 - garage: Testing with customer-specific services and DiaP



Exercises part I



Notes:

- Location of exercises: Desktop->“FIT-Share“->DiagInf\Exercises
- Tools:
 - BCT(Part of ECUWorx)





Short intro on BCT (1/2)

Start of ECUWorx:

Systray->Toolbase icon->Launchbox->ecu.worx (x.y.z)

Open project:

Open DiagInfTraining project provided in exercises

Create new XDI file

Open XDI file and solve exercises





Short intro on BTC (2/2)

Outline →

← Specification

Helptext →

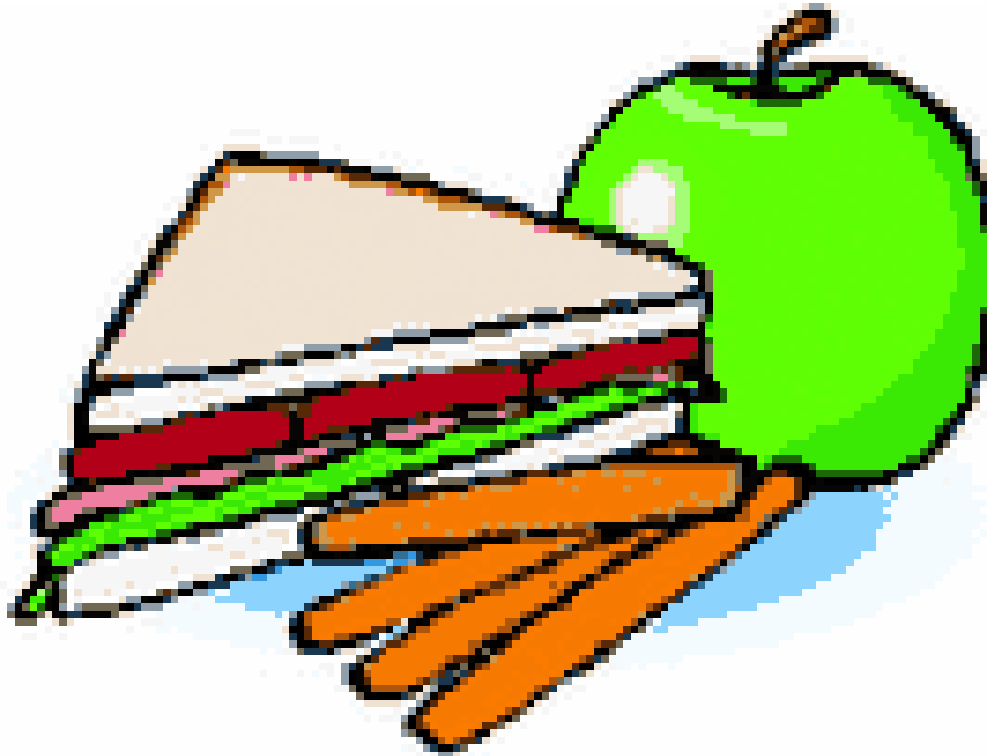
← Problems

The screenshot shows the Diagnostic Infrastructure Training Course software interface. The main window is titled "Venus/MSR_CONF_MX17 - [scripts] - M:\u_DiagInf_Test_mng2s\inf\diashd2_conf\Untitled_confdata.xml". The interface is divided into several panels:

- Outline:** A tree view on the left showing the project structure. The "Specification" folder is selected.
- Specification:** A central panel showing the configuration for the "Actuator class". It includes fields for "Internal Integer lower Limit", "Internal Integer upper Limit", "Physical lower Limit", "Physical upper Limit", "Visibility [opt.]", and "Actuator 'OFF' value".
- Helptext:** A panel on the bottom left showing the "Physical upper Limit" help text. It includes the category "VALUE", a description, and a detailed explanation of the physical upper limit.
- Problems:** A table at the bottom right showing the status of the configuration. It has columns for "Description", "Resource", "Path", and "Location".

Description	Resource	Path	Location
Errors (4 items)			
Required field is empty	M:\u_Diag...	Local	DIAXDLATS_CONVERSION
Required field is empty	M:\u_Diag...	Local	DIAXDLATS_DOCUMENTATION_EN
Required field is empty	M:\u_Diag...	Local	DIAXDLATS_PHYS_UPPER_LIMIT
Required field is empty	M:\u_Diag...	Local	Software Component





Agenda

- Chapter 1 General: DiagInf
 - Function
 - Configuration (BCT)

- Chapter 2 Service diagnostics
 - ATS (Advanced Test Service) – actuators
 - AVS (Adjustment Value Service) – adjustments
 - ETC (Engine Test Coordinator) – engine tests
 - **Signals**

- Chapter 3 OBD – Diagnostics
 - Signals/OBD
 - I15031



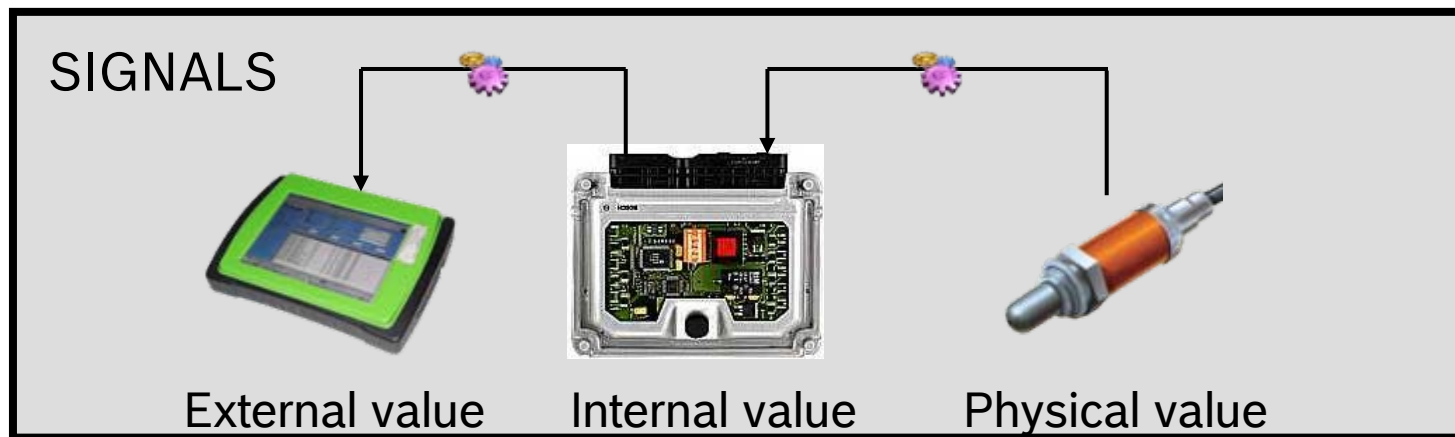
Overview

Signals...

- ... is used to read out internal ECU variables using a service tester
- ... provides universal access via defined interfaces
- ... provides generic connection to tester service (e.g. UDS, I15031)



Data flow Description





Function Description

- Signals is used to read out internal ECU variables using a service/scan tool.
- Signals provides universal access via defined interfaces and generated IDs for each ECU variable.
- Advantages:
 - Automatic generation of documentation.
 - Automatic integration into the testing process.
 - Signals carries out the conversion in the tester representation
- CM (Configuration Management) storage:
 - ClearCase: medc17/core/inf/signals
 - Nestor: COCOMP : Signals / CONFIG: Signals
 - SDOM: BC :DiaBas / MC : Signals





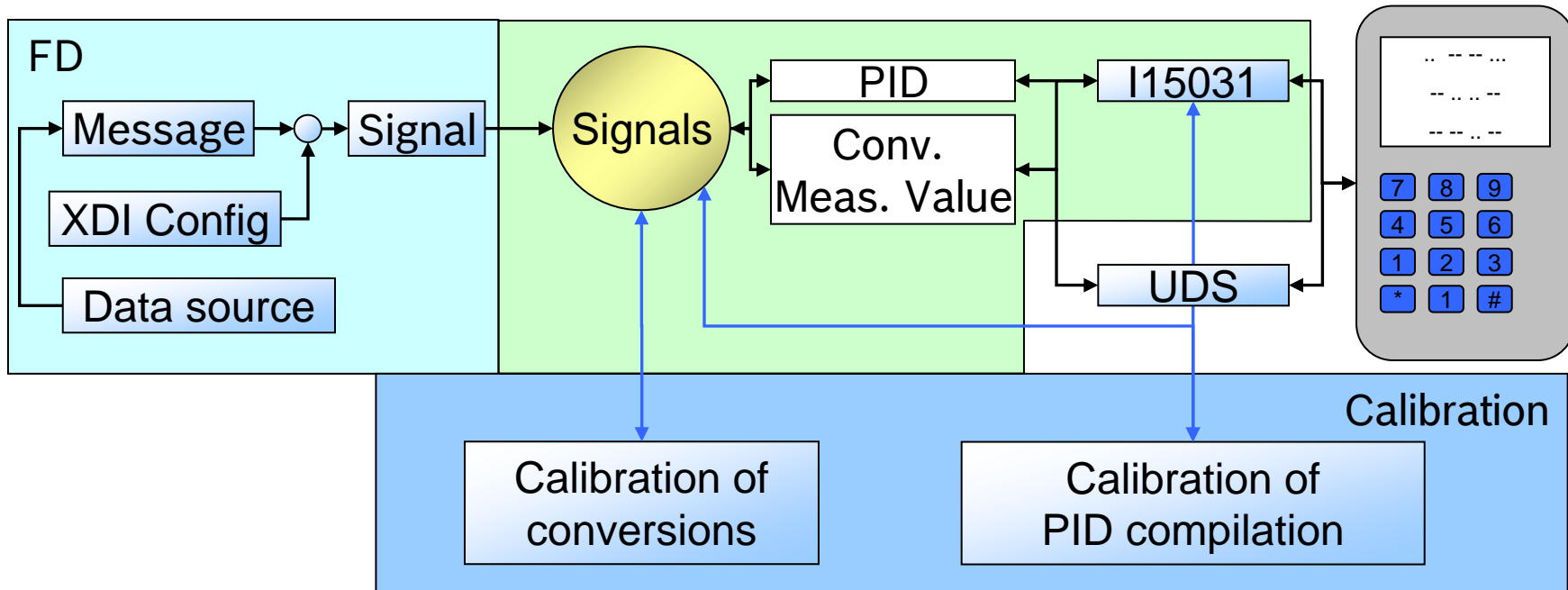
Data Flow, Interfaces in the System

- Signals functions as an access layer between the diagnostic services, which establish the connection to the diagnostic tester and internal ECU messages.
- The value of a message is
 - read out by Signals
 - converted according to the configuration
 - forwarded to the tester via the diagnostic services

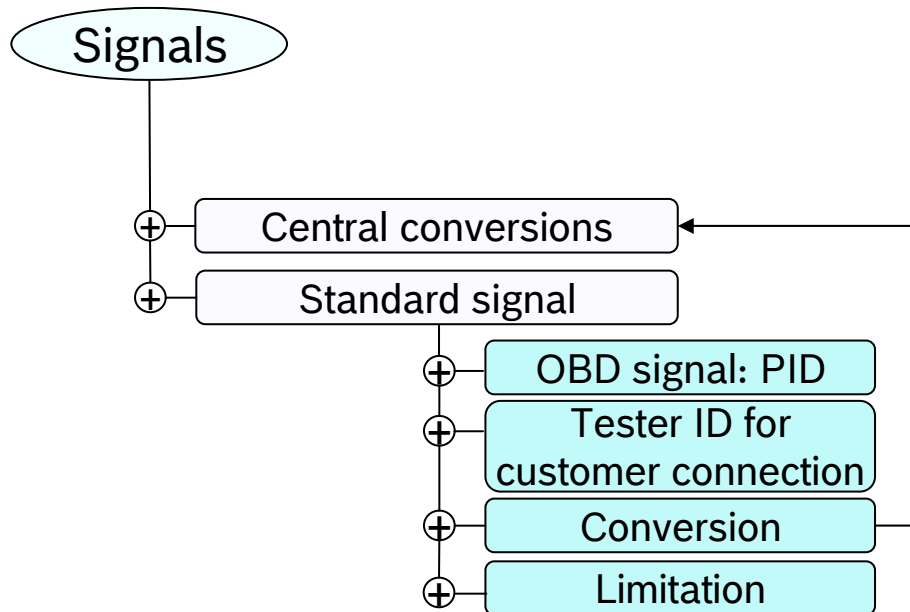




Data Flow, Interfaces in the System



Features



Features

- Standard signals
(*access to message*)
(OBD signals and customer signals)
- Central configuration of conversions
- PID assignments can be configured
- Calibration:
 - PID assignment
 - Conversion factors



Functional Description, Module Features

→ Standard signals

- When a signal value is called, the current value of the message assigned to the signal is read out. Depending on the configured conversion, this value is converted to the new representation and forwarded to the diagnostic services.
- The signal is generally called via the generic (internal) signal number (a signal's ID). It is also possible – depending on the support of the diagnostic services – to call a signal via its PID (OBD) or the customer tester ID.





Interface Description - C-Code

Note:

The interfaces are not relevant to the function development and are therefore not listed here.

(As a general rule, they are only used by the diagnostic services)



Interface Description - ASCET

- For Signals, it is sufficient to create a message + configuration, no further efforts in ASCET are necessary.



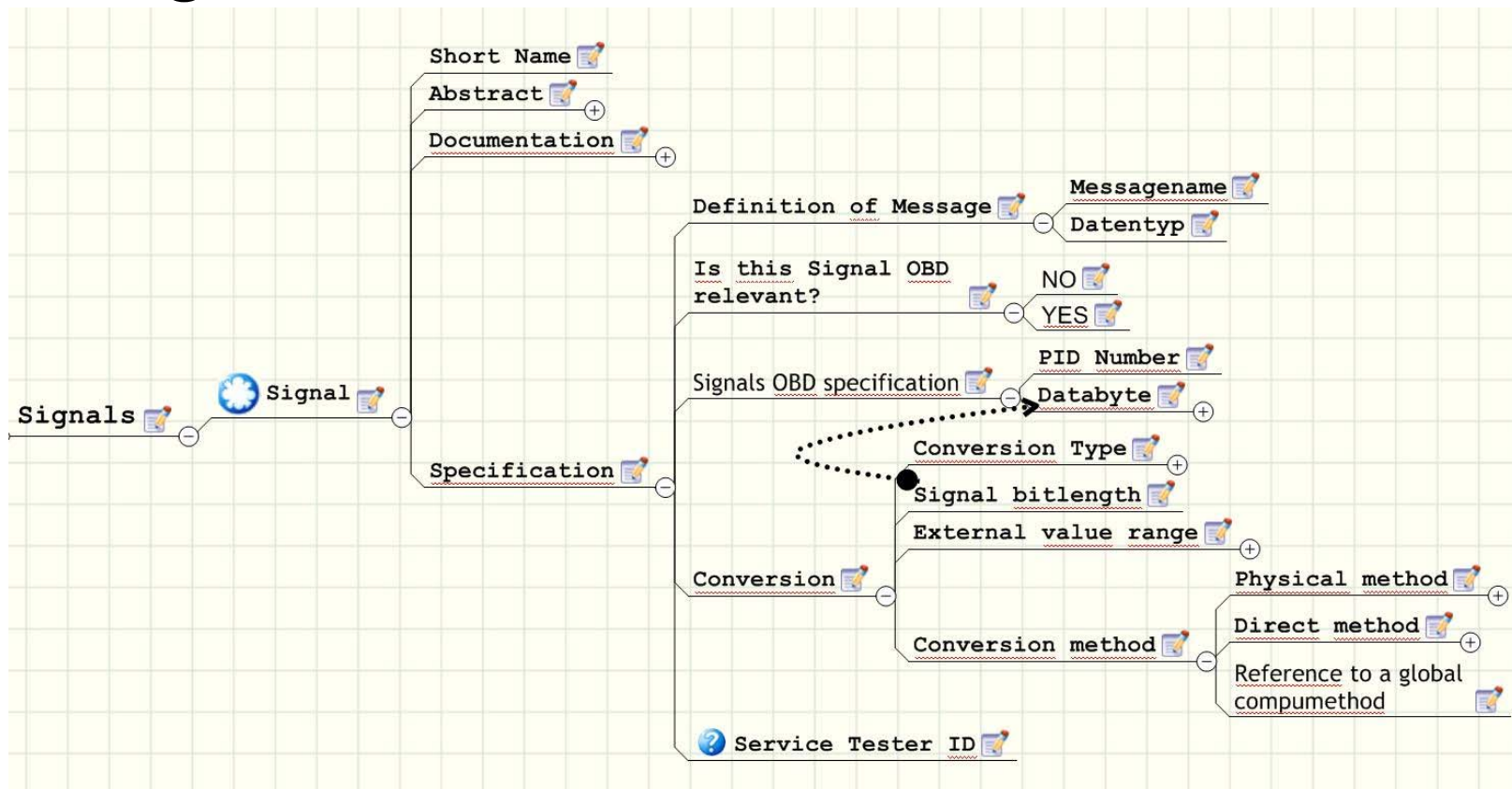
Configuration MEDC17, XDI - General

What must be considered in the configuration?

- Naming of the signal (OBD, not OBD)
- Not global variable, but message
NO MEASUREMENT-POINT!
- Conversion from PaVaSt
- Physical limits
- Desired length of the signal



Configuration MEDC17, XDI



Description on an abstract level based on the XDI Feature Tree

Test Procedure

→ Provided:

- Signal XYZ, which references message xyz

→ Sought:

- How can XYZ be tested?
 - Calling of the signal via the respective diagnostic service (*I15031, DiaP,...*)
 - Calling the signal via e.g. ODX-Link, SamDia, ...
 - Comparison of the current message value (e.g. *in INCA*) and the received signal value, and check whether the expected value was adhered to.
(*Observe conversion if necessary*)

Test Procedure

→ Setup of the testing environment

- A complete signal configuration is required; in case of a call via OBD, the calibration of the PID assignment must also be completed.
- INCA may be required to display the message value
- Connection to diagnostic interface
(ODX-Link, KTS tester, or SamDia/CANalyser for simulation).



Test Procedure

→ Configuration check

- The test of the actual Signals configuration is carried out at the time of FCI (**F**unctional **C**omponent **I**mplementation test).
(For calibratable conversions, this can only be completely carried out at this time)
- Testing of the calibration (conversion, PID assignment) for FCF (**F**unctional **C**omponent **F**unction test).
If only one calibration of the PID assignment is carried out, it is sufficient to test the mapping between the PID and signal.
When calibrating conversions, this part of the FCF must be repeated for the affected signals.



Test Procedure

→ Testing the configuration

- Here, attention must be paid to the correct configuration of:
 - Message assignment (*Is the correct message accessed?*)
 - Signal length (*Is the limitation correct?*)
 - Conversion (*Does the message value correspond to the desired converted signal value?*)

→ Testing the calibration

- Was the assignment to a PID carried out correctly?
Query of the PIDs via the I15031 service; the signal value must have the correct value (incl. conversion).
- Has the conversion for a signal been calibrated correctly?
This can only be done via a manual check.



Interface Description for Calibration

Depending on the configuration, Signals creates conversion formulas for signals which can be calibrated both binary and linearly.

Binary conversions:

- "SIGNALS_COMPU_" + < Name from the configuration >.StrtPos_C
Start position of the bit field.
- "SIGNALS_COMPU_" + < Name from the configuration >.Lngh_C
Length of the bit field



Interface Description for Calibration

Linear conversions:

- "SIGNALS_COMPU_" + < *Name from the configuration* >.Norm_C
This is used to set the desired resolution of the slope factor if this is not to be an integer. "Norm_C" thus represents a bit value by which "Fac_C" is shifted in the calculation (instead of a division -> run time)
- "SIGNALS_COMPU_" + < *Name from the configuration* >.Fac_C
slope factor. The actually desired slope factor (usually floating point value) must be multiplied by $2^{\text{Norm_C}}$ to obtain the actual value as an integer.
- "SIGNALS_COMPU_" + < *Name from the configuration* >.Ofs_C
Offset value, must be an integer.

Notes Regarding Integration

- No special hints for integration necessary.



Process

- **Tasks in function development**

Configuration of all OBD signals, as well as all signals required for the service diagnostics

- **Calibration**

Disabling of all OBD signals which are not supported, and, if necessary, temporary correction of the PID assignment for OBD (permanent change due to function development (FD))
Calibration of conversions for correspondingly demanded service signals



Process

- **CE department**
 - Ordering of OBD [FD]/ customer-specific [ESC2] signals
 - Ordering of necessary changes in Communication layer [ESC2]
- **Function development**
 - Configuration of Signals
 - Assignment of OBD relevant Signals to the according PID
 - Testing
- **Calibration**
 - Calibration of OBD (PID) assignment
 - Where applicable calibration of conversions



Process

- **Responsibility**

Function developer (FD) is responsible for the correctness of his/her signal (conversion, length, limitation, OBD configuration)

CE department is responsible for completeness (workshop, OBD)

- **Testing concept (MCD-3D)**

Using MCD-3D to test individual signals

- OBD: By testing with I15031 (=CARB) services for output in the scan tool
- Workshop: By testing with customer-specific services and DiaP

Agenda

- Chapter 1 General: DiagInf
 - Function
 - Configuration (BCT)

- Chapter 2 Service diagnostics
 - ATS (Advanced Test Service) – actuators
 - AVS (Adjustment Value Service) – adjustments
 - ETC (Engine Test Coordinator) – engine tests
 - Signals

- **Chapter 3 OBD – Diagnostics**
 - **Signals/OBD**
 - I15031



OBD Component

OBD relationship of a signal:

a) Configuration

Standard configuration of the signal + assignment of a signal to a particular PID results in an OBD signal.

OBD Component

OBD relationship of a signal:

ISO 15031-5 Requirement –SW messages

- Most important requirement is that the message used for a PID has to be a real living value, no substitute or default values are allowed.
- Clipping of ISO 15031-5.4, section 7.1.1
 - All data values returned for sensor readings must be actual readings, not default or substitute values used by the system because of a fault with that sensor.
- The SW messages have to fulfill this requirement.



OBD Component

OBD relationship of a signal :

Please observe:

- For an easier calibration of the signals to the PID numbers, the signals should be named as:
- PID<PIDNumber>h_<SWmessage>, e.g. PID1Eh_PTOSwt_st
- This naming convention is valid for Diesel and Gasoline projects





OBD Component

OBD relationship of a signal :

b) Calibration

Assignment of signals to PIDs using:

- Signals_Mode1PID_CA
- Signals_Mode1SigNum_CA



OBD Component - Description

Signals provides the following calibration labels:

- Signals_Mode1Pid_CA
- Signals_Mode1SigNum_CA

These two can only be used in combination.

The assignment of a signal to a PID is carried out in both arrays.

In doing so, the PID is entered in a specific Index X in

Signals_Mode1Pid_CA, in Signals_Mode1SigNum_CA, the assigned signal number is entered in the same Index X.

Signals_Mode1Pid_CA must be sorted in ascending order.

Example:

Signals_Mode1Pid_CA	2	4	13	13	13	42
Signals_Mode1SigNum_CA	Signals_SigA	Signals_SigB	Signals_SigQ	Signals_SigE	Signals_SigX	Signals_SigZ

OBD Component - Description

If several signals are to be assigned to one PID, the respective PID is repeated at the position X ... X+n in Signals_Mode1Pid_CA, and the corresponding signals are entered at the position X ... X+n in Signals_Mode1SigNum_CA.

Example:

Signals_Mode1Pid_CA	2	4	13	13	13	42
Signals_Mode1SigNum_CA	Signals_SigA	Signals_SigB	Signals_SigQ	Signals_SigE	Signals_SigX	Signals_SigZ





Agenda

- Chapter 1 General: DiagInf
 - Function
 - Configuration (BCT)

- Chapter 2 Service diagnostics
 - ATS (Advanced Test Service) – actuators
 - AVS (Adjustment Value Service) – adjustments
 - ETC (Engine Test Coordinator) – engine tests
 - Signals

- Chapter 3 OBD – Diagnostics
 - Signals/OBD
 - **I15031**



Overview

I15031: Carb/OBD Services

- ➔ I15031 specifies communication services between the external Tester and the ECU.

- ➔ CM (Configuration Management) storage:
 - ClearCase: medc17/core/com/diagsrv/i15031
 - Nestor: COCOMP: I15031 / CONFIG: I15031
 - SDOM: BC: DiaBas / MC: I15031 MC: I15031Appl



Overview of the Services

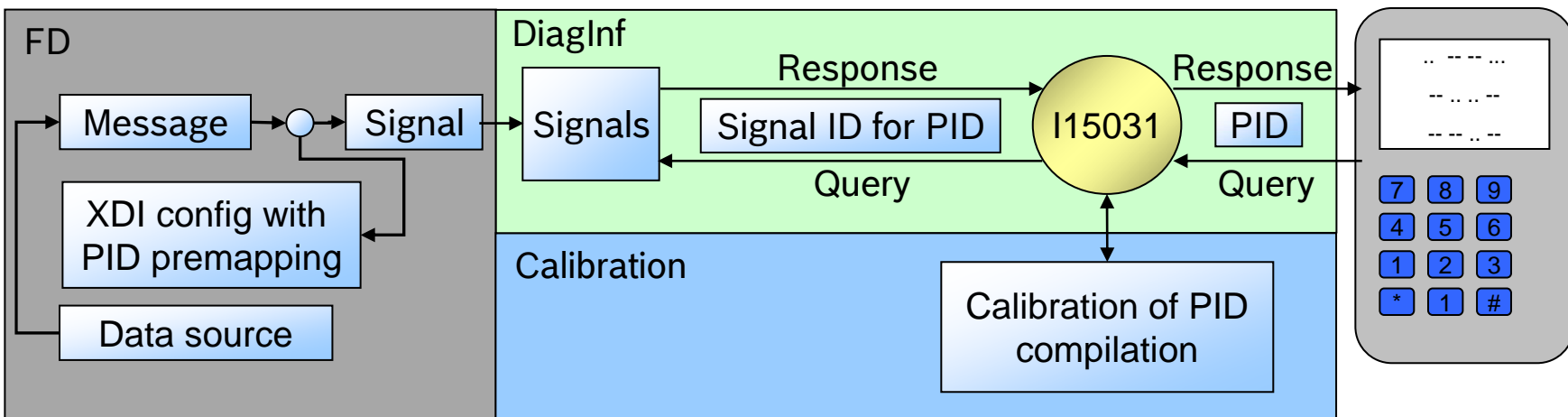
Service...

- ... \$01: Request current powertrain diagnostic data
- ... \$02: Request powertrain freeze frame data
- ... \$03: Request emission-related diagnostic trouble codes
- ... \$04: Clear/reset emission-related diagnostic trouble codes
- ... \$05: Request oxygen sensor monitoring test results (not available)
- ... \$06: Request on-board monitoring test results for specific monitored systems
- ... \$07: Request emission-related diagnostic trouble codes detected during current or last completed driving cycle
- ... \$08: Request control of on-board system, test or component
- ... \$09: Request vehicle information (e.g. VIN, CALID, CVN...)
- ... \$0A: Request emission-related diagnostic trouble codes with permanent status



Data Flow, Interfaces in the System - Service\$01

Output of actual values to the scan tool



Example for PID - Service\$01

TABLE B35 - PID \$2F DEFINITION

PID (hex)	Description	Data Byte	Min. Value	Max. Value	Scaling/Bit	External Test Equipment SI (Metric) / English Display
2F	Fuel Level Input	A	0 % no fuel	100 % max. fuel capacity	100/255 %	FLI: xxx.x %
	FLI shall indicate nominal fuel tank liquid fill capacity as a percent of maximum, if utilized by the control module for OBD monitoring. FLI may be obtained directly from a sensor, may be obtained indirectly via the vehicle serial data communication bus, or may be inferred by the control strategy using other sensor inputs. Vehicles that use gaseous fuels shall display the percent of useable fuel capacity. If there are two tanks in a bi-fuel car, one for each fuel type, the Fuel Level Input reported shall be from the tank, which contains the fuel type the engine is running on.					

Function Description, Module Features Service \$01

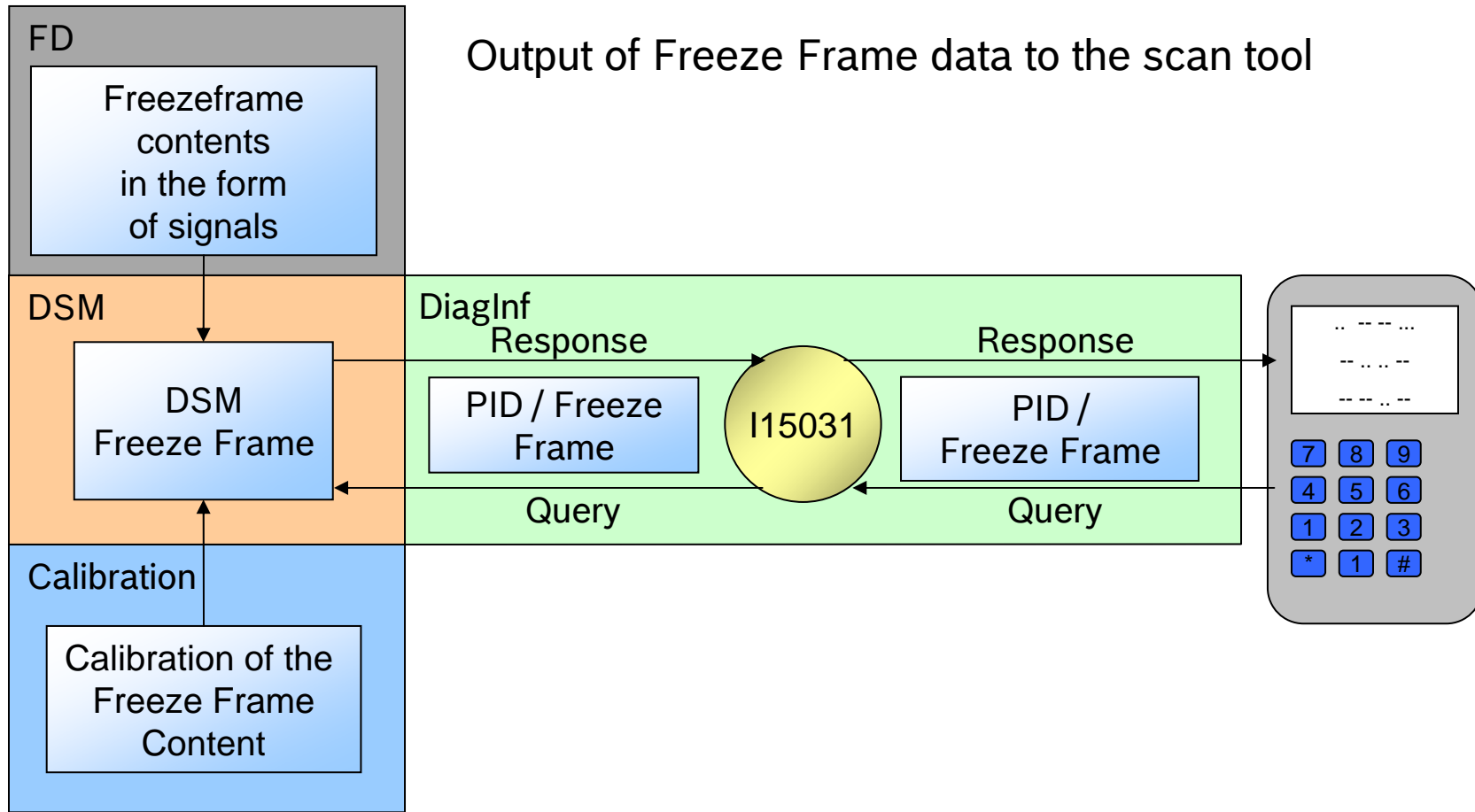
→ Output of actual values to the scan tool

Sequence:

- Tester queries I15031 for PIDs
- I15031 queries Signals for the signals corresponding to the PID
- Signals provides the signals
- I15031 sends the PIDs to the tester



Data Flow, Interfaces in the System - Service \$02



Function Description, Module Features, Service \$02

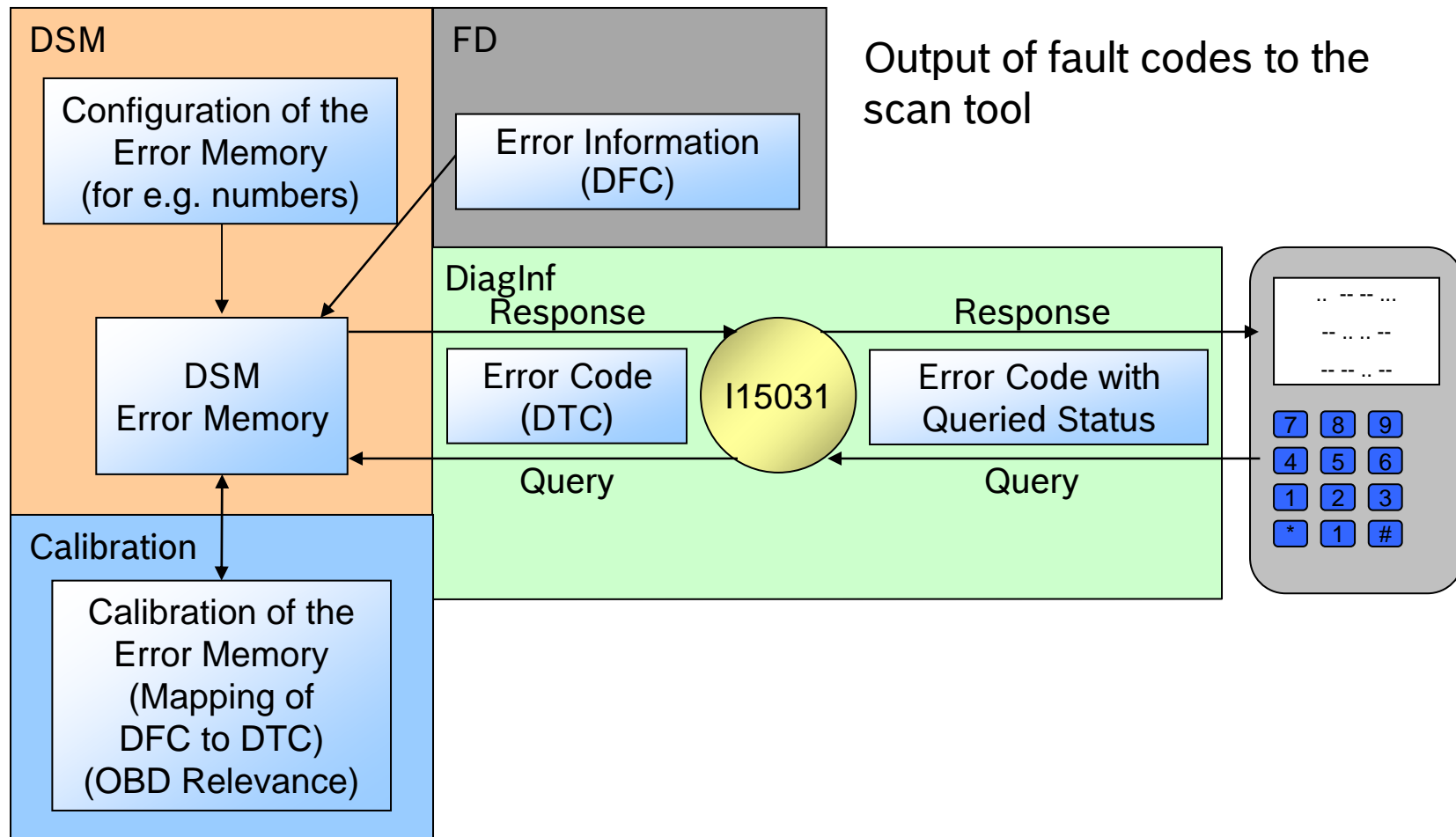
→ Output of Freeze Frame data to the scan tool

Sequence:

- Tester queries I15031 for PID/Freeze Frame number
- I15031 queries DSM for the signals corresponding to the PID/Freeze Frame
- Signals/DSM provides the signals data
- I15031 samples the data and sends this to the tester



Data Flow, Interfaces in the System - Service \$03/\$07/\$0A



Function Description, Module Features, Service \$03/\$07/\$0A

- Output of fault codes to the scan tool

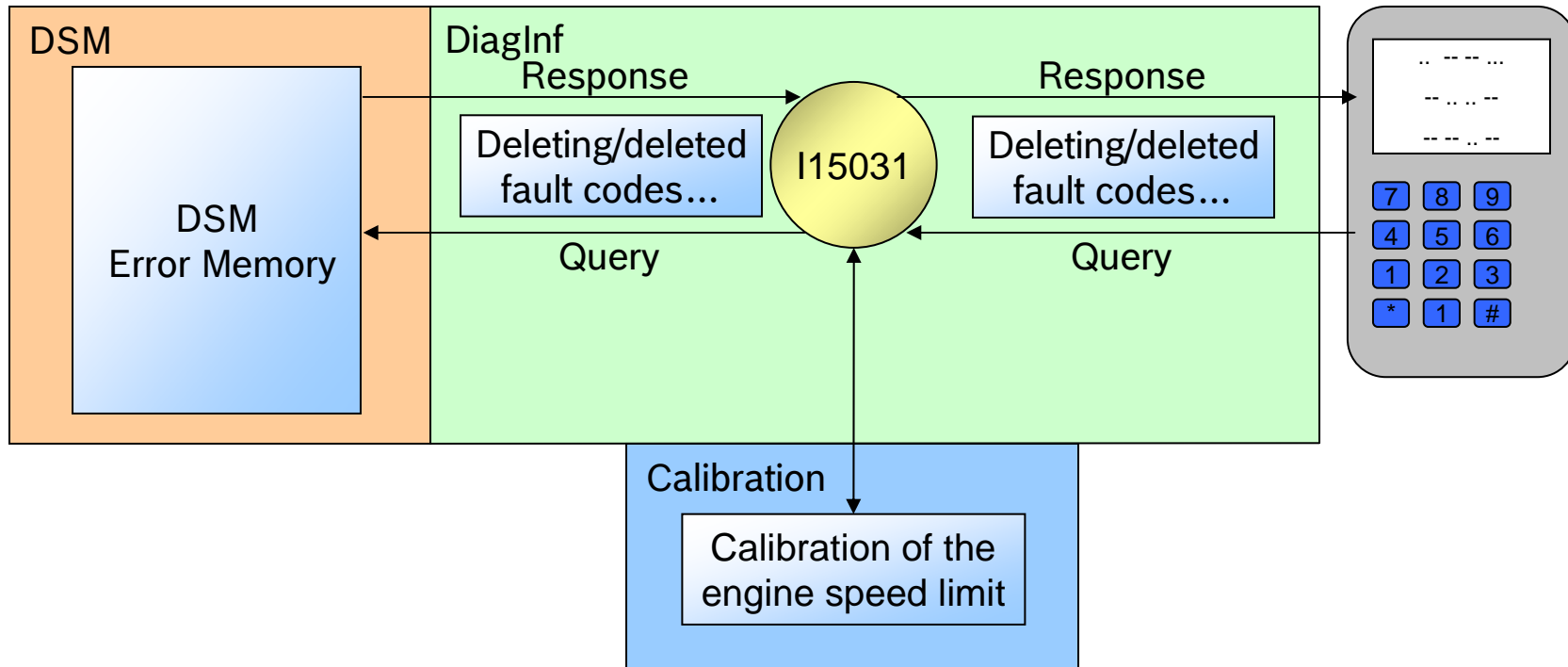
Sequence:

- Tester queries I15031 for the fault codes using the status that corresponds to the service
- I15031 sets the filter according to the visibility (which relates to the service) and queries the fault codes
- DSM provides the fault codes
- I15031 sends the fault codes to the tester



Data Flow, Interfaces in the System - Service \$04

Deleting the fault codes



Function Description, Module Features, Service \$04

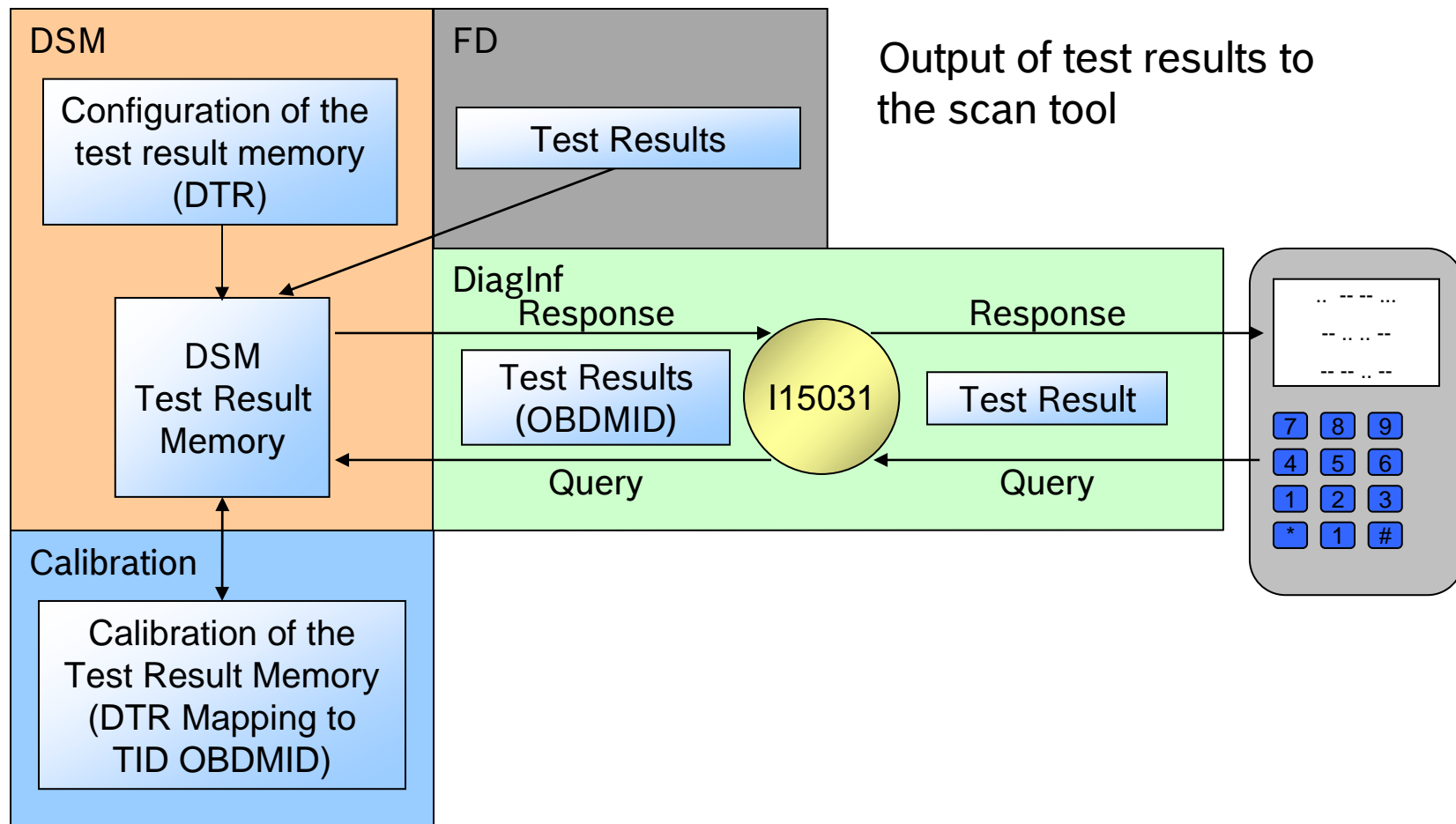
- Deleting the fault codes

Sequence:

- Tester demands deletion of the fault codes and other values in the DSM
- I15031 sets visibility and demands deletion of the errors in the DSM
- DSM deletes and provides positive response
- I15031 reports to the tester



Data Flow, Interfaces in the System - Service \$06



Function Description, Module Features, Service \$06

→ Output of test results to the scan tool

Sequence:

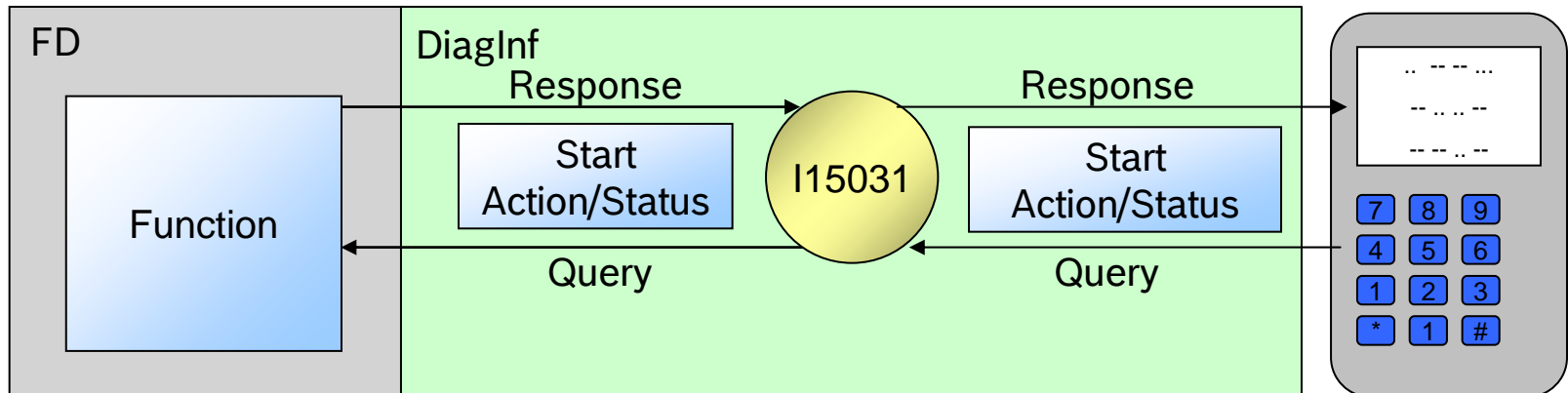
- Tester queries I15031 for test results
- I15031 queries DSM for the requested test results
- DSM provides the test results
- I15031 sends the test results to the tester



Data Flow, Interfaces in the System - Service \$08

Check of the scan tool via test or components

Implemented only for few customers



Function Description, Module Features, Service \$08

- Check of the scan tool via test or components

Sequence:

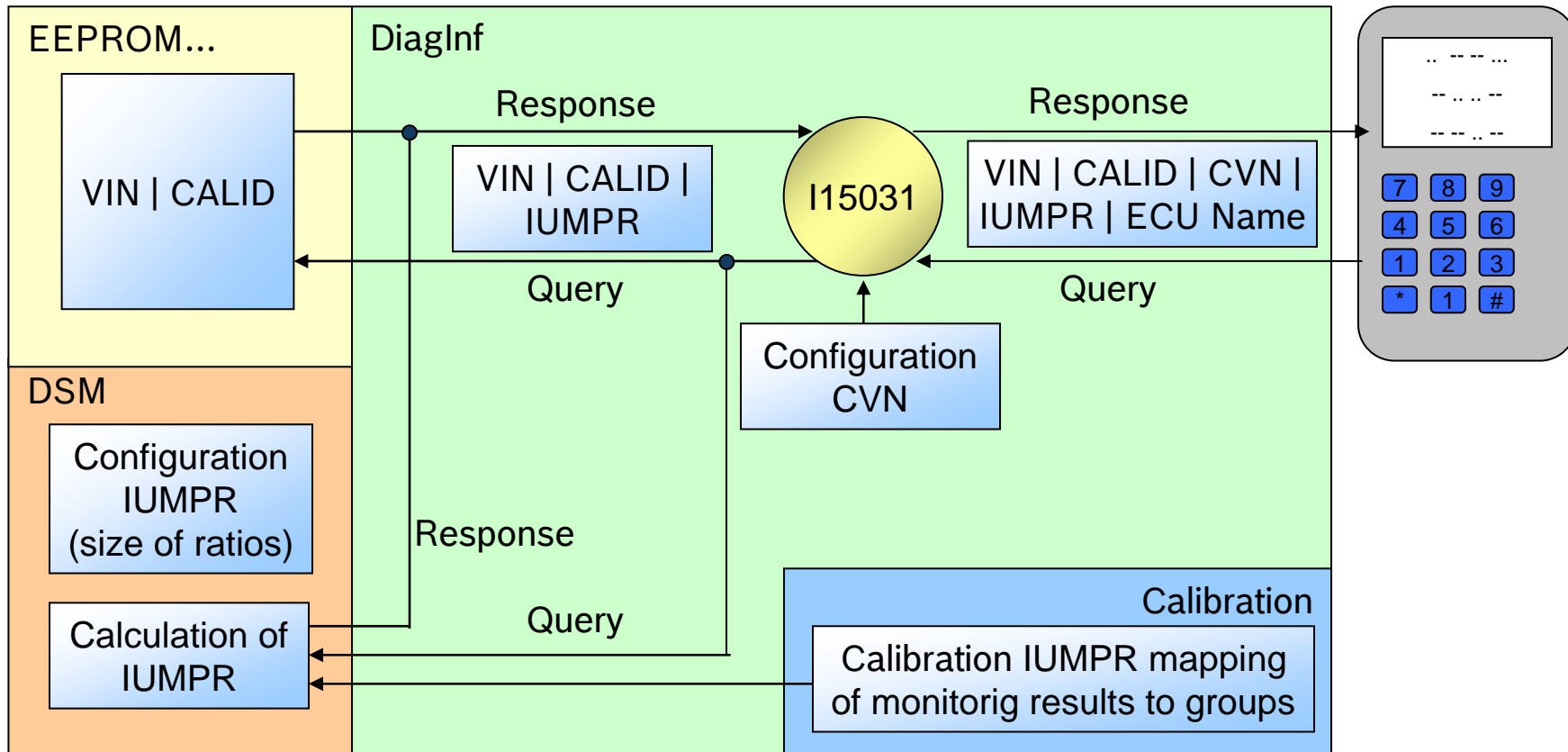
- Tester requires action from I15031
- I15031 forwards the request
- Function carries out action and provides results to I15031
- I15031 sends the results to the tester





Data Flow, Interfaces in the System - Service \$09

Output of vehicle information to the scan tool



Function Description, Module Features, Service \$09

→ Output of vehicle information to the scan tool

Sequence:

- Tester queries I15031 for one of the following pieces of information:
VIN, CALID, CVN, IUMPR, ECU Name
- I15031 queries as follows:
 - VIN, CALID from EEPROM...
 - IUMPR from DSM
 - CVN and ECU Name is calculated internally by/provided by I15031
- Required information is provided by DSM, I15031 and EEPROM...
- I15031 sends the requested information to the tester



Test Procedure

→ **Provided:**

- Carb services and the corresponding information from FD, DSM, Signals, EEPROM

→ **Sought:**

- Call of the individual services via a tester or a test tool (e.g. CANalyzer)
- Check depending on the service
 - Correct support information and correct content (Service\$01/\$02/\$06/\$09)
 - Correct content (Service\$03/\$07/\$0A)
 - Correct reaction (Service\$04/\$08)



Test Procedure

→ Setup of the testing environment

→ The following is required:

- Service \$01: PIDs to with valid signals
- Service \$02: A filled Freeze Frame in the DSM
- Service \$03/\$07/\$0A: DTCs which corresponding status in the DSM
- Service \$06: OBDMIDs with corresponding content
- Service \$09: Available vehicle information
- INCA may be required to display the message value
- Connection to diagnostic interface
(ODX-Link, KTS tester or SamDia/CANalyser for simulation of tester)



Test Procedure

→ Testing the calibration 1

→ Service\$01:

- Are all required PIDs calibrated with the correct signals?
- Query of the support information => any missing PIDs?
Are unwanted PIDs supported?
- Query of the PIDs => Are the correct contents transferred?

→ Service\$02:

- Are all PIDs required for Service\$02 available?
- Query of the support information => any missing PIDs?
Are unwanted PIDs supported?



Test Procedure

→ Testing the calibration 2

→ Service\$04:

- Is the upper engine speed limit for deleting errors correct?
- Deleting the errors below the limit => deletion possible?
- Deleting the errors above the limit => deletion not possible?

→ Service\$09:

- Are all required Infotypes are calibrated correctly?
- Request for support information -> only the Infotypes which are required are supported?
- Request of a not supported Infotype -> no answer?
- Request of the required Infotyps-> answer with correct content?



Overview about I15031 calibration

- General
 - Definition of terms
 - Range of functions I15031
- Legal principles (extract)
- Calibration notes
 - Service\$01/\$02
 - for I15031 packages according to model year up to 2009
 - for I15031 packages as of model year 2010
 - Service\$04
 - Service\$09



Definition of Terms

→ PID	Parameter ID
→ Element	Value within a PID corresponding to the standard
→ Signal	Includes a message and a conversion
→ Dummy message	Signal for filling non-supported elements
→ Service	Function for querying and issuing data
→ Lean Application	Calibration simplification for GS
→ Supported	Supported PIDs/elements
→ Support Byte	Shows the elements of a PID that are supported
→ Infotype	Designation for specific values that are issued via Service\$09, e.g. VIN



Range of Functions I15031

- ➔ I15031 is the interface between the OBD tester and the control unit
- ➔ Data can be read out, error memories can be deleted and actions can be initiated

- ➔ The following is relevant to the calibration:
 - Selection of PIDs
 - Allocation of data to parameter IDs
 - Support information within a PID
 - Threshold for deleting the error memory

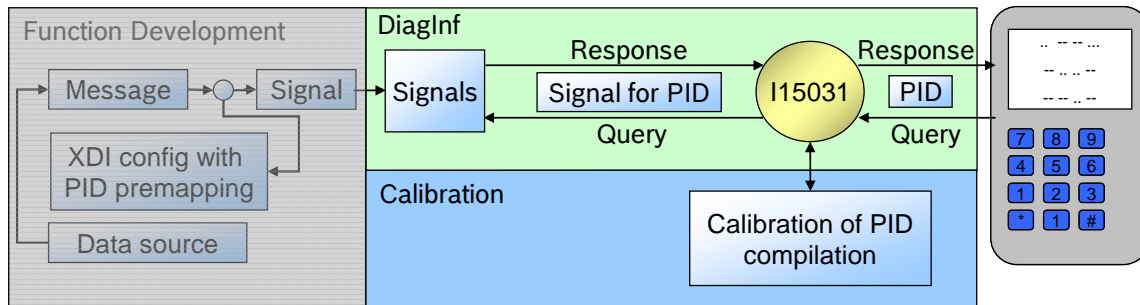
Diesel Gasoline Systems



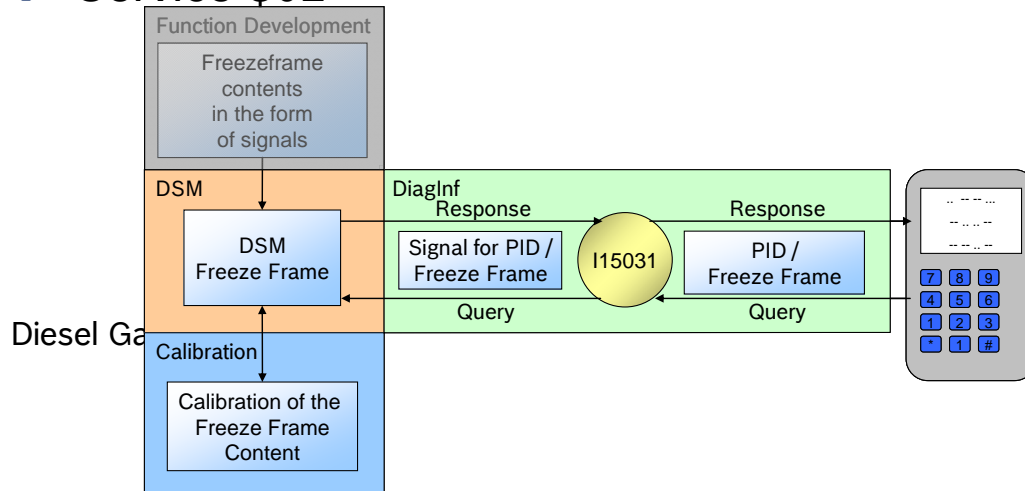
BOSCH

Range of Functions I15031

→ Service \$01



→ Service \$02



Legal Principles

- ➔ PIDs are defined in the standard ISO15031 / SAE J1979
- ➔ Which PIDs are required in the project depends on the context of the project (target market, model year and equipment) and the customer requirements. Consultation from GS/EPD1 (system department)
- ➔ Which values are issued and in which format is defined in the standard ISO15031 /SAE J1979
- ➔ A PID must always be fully available. Non-supported elements must be filled with a dummy message.
- ➔ The following dummy messages are available:
 - for one byte elements: I15031_FILLSIG1
 - for two byte elements: I15031_FILLSIG2
 - for four byte elements: I15031_FILLSIG4



Calibration Notes

→ General

- The standards SAE J1979 and ISO 15031 are constantly evolving
- Legislators, when determining his requirements for a model year, refer to the respective standard version, usually the latest version
- To implement the standard, a software update is generally required for the package I15031 as well as a respective calibration

→ Mapping between standard and SW version

- ISO 15031-5:2006
 - up to package I15031 version 4.x.x
 - up to MY2009
- SAE J1979_AffirmationBallot_APR032007Package I15031
 - as of version 7.0.4
 - as of MY2010



Calibration Notes

New for MY2010 (SAE J1979)	Available as of
Introduction of Service\$09 Infotype B	B_15031.5.0.0 E09/07
Introduction of Service\$0A	B_15031.5.0.0 E09/07
Minimum changes to Service\$09 Infotype A (ECU Name)	B_15031.5.0.0 E09/07
Changes in PID\$1C Introduction of PID\$5F	B_15031.5.0.0 E09/07
Introduction of dummy messages for the PID configuration	B_15031.5.0.0 E09/07
Introduction of composite PIDs with support byte	B_15031.5.0.0 E09/07



Calibration of the PIDs up to MY2009

→ Service \$01

- Purpose: current values from diagnostic data are issued via PIDs.
- Notes concerning the calibration and set of rules
 1. The PIDs are entered in **Signals_Mode1Pid_CA** in ascending order. This determines which PIDs are available
 2. The corresponding signals are entered at the corresponding positions in the array **Signals_Mode1SigNum_CA**. This determines which signals are assigned to the PIDs and in what order.
 3. If a PID consists of several signals, the PID must be entered multiple times in **Signals_Mode1Pid_CA**.
 4. If not all elements are supported in a PID, then **dummy messages** must be entered at the respective positions.



Calibration of the PIDs up to MY2009

→ Example

Signals_Mode1PiD_CA

PID\$A	PID\$B	PID\$B	PID\$B	PID\$C	PID\$C	PID\$D
--------	--------	--------	--------	--------	--------	--------

Signals_Mode1SigNum_CA

Signal1	Signal2	FillSig	Signal3	Signal4	Signal5	Signal Default
---------	---------	---------	---------	---------	---------	-------------------

Element 2 of
the PID\$B
not
supported

PID\$D not
supported

Calibration of the PIDs up to MY2009

→ Explanations to the example

- PID\$A, according to the standard, consists of one element. It is available in the program baseline as "Signal1". The PID\$A has been entered and is issued to the OBD Scan Tool as "supported". "Signal1" must present the characteristics that are requested in SAEJ1979
- PID\$B, according to the standard, consists of 3 elements. However – as determined by the system configuration – the 2nd element is not/cannot be supported. This is why a "Fillsig" is calibrated here instead. The length of the dummy message and all characteristics of "Signal2" and "Signal3" must comply with the standard
- PID\$D is not supported. Alternatively, PID\$D could also be removed from the array Signals_Mode1PiD_CA



Calibration of the PIDs up to MY2009

→ Service \$02

- Purpose: Freeze Frame data is issued via PIDs.
- Notes concerning the calibration and set of rules
 - The PIDs are entered in the array ***I15031_stSrv02SuppPid_CA*** in ascending order. This determines which PIDs are available in Service \$02.
 - Each PID is only entered once.
 - The empty positions are filled in with zeros at the beginning of the array.



Calibration of the PIDs up to MY2009

→ Example

I15031_stSrv02SuppPid_CA

0	0	0	0	PID\$A	PID\$B	PID\$C
---	---	---	---	--------	--------	--------

Beginning of the array should only be filled with zero

PID\$B is a composite PID and is nevertheless only entered once.

Calibration of the PIDs up to MY2009

→ Explanations to the example

- The assignment of PIDs to signals for Service\$02 is taken from the Service\$01 calibration. (see example of Service\$01)
- In Service\$02, PID\$A, PID\$B and PID\$C are supported.



Calibration of the PIDs as of MY2010

→ DS/GS calibration concepts

- GS: "Lean Application"
 - The simplification of the calibration "Lean Application" means that the assignment between signals and PIDs cannot be calibrated. It must come correctly out of the configuration.
 - Only PIDs can be switched off. The array **Signals_stSrv01SuppPID_CA** is used for this
- DS
 - The "Lean Application" is switched off via system constants
 - **Signals_stSrv01SuppPID_CA** is not available
 - PIDs are entered in **Signals_Mode1Pid_CA**.
 - The corresponding signals are entered in the array **Signals_Mode1SigNum_CA**.



Calibration of the PIDs as of MY2010

→ Service \$01 **without** "Lean Application" (DS)

- Notes concerning the calibration and set of rules without "Lean Application"
 - The PIDs are entered in **Signals_Mode1Pid_CA** in ascending order. This determines which PIDs are available
 - The corresponding signals are entered at the corresponding positions in the array **Signals_Mode1SigNum_CA**. This determines which signals are assigned to the PIDs and in what order.
 - If a PID consists of several signals, the PID must be entered multiple times in **Signals_Mode1Pid_CA** .
 - If not all elements are supported in a PID, then **dummy messages** must be entered at the respective positions.



Calibration of the PIDs as of MY2010

→ Example **without** "Lean Application"

Signals_Mode1PiD_CA

PID\$A	PID\$B	PID\$B	PID\$B	PID\$C	PID\$C	PID\$D
--------	--------	--------	--------	--------	--------	--------

Signals_Mode1SigNum_CA

Signal1	Signal2	FillSig	Signal3	Signal4	Signal5	Signal Default
---------	---------	---------	---------	---------	---------	----------------

Element 2 of
the PID\$B
not
supported

PID\$D not
supported

Calibration of the PIDs as of MY2010

→ Notes on the example **without** "Lean Application"

- PID\$A, according to the standard, consists of one element. It is available in the program baseline as "Signal1". The PID\$A has been entered and is issued to the OBD Scan Tool as "supported". "Signal1" must present the characteristics that are requested in SAEJ1979
- PID\$B, according to the standard, consists of 3 elements. However – as determined by the system configuration – the 2nd element is not/cannot be supported. This is why a "Fillsig" is calibrated here instead. The length of the dummy message and all characteristics of "Signal2" and "Signal3" must comply with the standard
- PID\$D is not supported. Alternatively, PID\$D could also be removed from the array Signals_Mode1PiD_CA



Calibration of the PIDs as of MY2010

- Service \$02 **without** "Lean Application"
 - ***i15031_stSrv02SuppPid_CA*** is omitted. The Service\$01 arrays and the Freeze Frame array ***DFES_xAsgnFrzFrSig_CA*** are evaluated in its place. A PID is only supported in Service\$02 if all signals assigned in Service\$1 are also available in the Freeze Frame.
 - Notes concerning the calibration and set of rules
 - In the DSM Freeze Frame array ***DFES_xAsgnFrzFrSig_CA***, all signals are entered that are relevant to the PIDs supported in Service\$2. This also includes the dummy messages.
 - If a PID is not supported, then at least one signal is not entered in the Freeze Frame array.



Calibration of the PIDs as of MY2010

→ Example **without** "Lean Application"

DFES_xAsgnFrzFrSig_CA

Signal1	Signal2	FillSig	Signal3	Signal4	Signal5
---------	---------	---------	---------	---------	---------

The dummy
message must be
entered in order to
support the PID\$B

Calibration of the PIDs as of 2010

→ Notes on the example **without** "Lean Application"

- The assignment of PIDs to signals for Service\$02 is taken from the Service\$01 calibration. (see example of Service\$1)
- Since the Freeze Frame array contains all signals that are assigned to the PIDs \$A, \$B and \$C, PID\$A, PID\$B and PID\$C are supported.



Calibration of the PIDs as of MY2010

- Service \$01 **with** "Lean Application" (GS)
 - Since the PID signal assignment is predetermined, the arrays **Signals_Mode1Pid_CA** and **Signals_Mode1SigNum_CA** are only available as read only.
 - Notes concerning the calibration and set of rules with "Lean Application"
 - All PIDs that are available in the project are automatically entered in **Signals_stSrv01SuppPID_CA** and can be switched off there. This determines which PIDs are available.



Calibration of the PIDs as of MY2010

→ Example **with** "Lean application"

Signals_stSrv01SuppPid_CA

PID\$A	PID\$B	PIDx00
--------	--------	--------

PID\$C is switched off

Calibration of the PIDs as of 2010

→ Explanations to the example **with** "Lean Application"

- The assignment of PIDs to signals is determined in the configuration.
- Since the PID\$C is replaced by PIDx00 in the array **Signals_stSrv01SuppPid_CA**, it is not supported.



Calibration of the PIDs as of MY2010

- Service \$02 **with** "Lean Application"
 - ***i15031_stSrv02SuppPid_CA*** is omitted. The Service\$01 arrays and the Freeze Frame array ***DFES_xAsgnFrzFrSig_CA*** are evaluated in its place. A PID is only supported in Service\$02 if all signals assigned in Service\$1 are also available in the Freeze Frame.
 - Notes concerning the calibration and set of rules
 - In the Freeze Frame array ***DFES_xAsgnFrzFrSig_CA***, all signals are entered that are relevant to the PIDs supported in Service\$2. This also includes the dummy messages.
 - If a PID is not supported, then at least one signal is not entered in the Freeze Frame array.



Calibration of the PIDs as of MY2010

→ Example **with** "Lean application"

DFES_xAsgnFrzFrSig_CA

Signal1	Signal2	FillSig	Signal3	Signal4	Signal5
---------	---------	---------	---------	---------	---------

The dummy
message must be
entered in order to
support the PID\$B

Calibration of the PIDs as of MY 2010

→ Explanations to the example **with** "Lean Application"

- The assignment of PIDs to signals for Service\$02 is taken from the Service\$01 calibration. (see example of Service\$1)
- Since the Freeze Frame array contains all signals that are assigned to the PIDs \$A and \$B, PID\$A and PID\$B are supported. Since PID\$C is not supported in Service\$01, it is also not supported in Service\$2



Calibration of the Support Bytes as of MY 2010

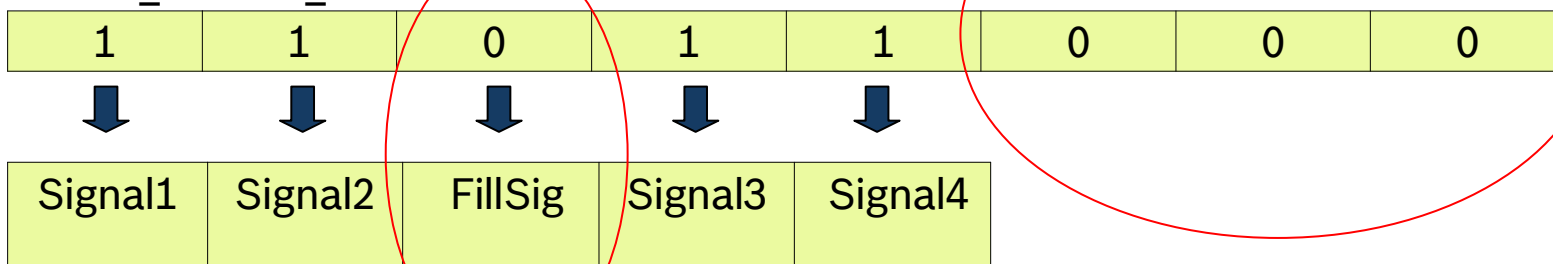
- Support bytes show the elements of a PID that are supported.
- Not all PIDs have a support byte.
- With the exception of PID\$8B, the support information, if it is available, is at Byte\$A
- As a rule, a bit of the support byte corresponds to an element, ie. if the bit is set, then the corresponding element is supported.
- The support information must be calibrated.
- For each support byte, there is a calibration label, e.g. I15031_PIDxxA_C



Calibration of the Support Bytes as of MY 2010

→ Example of the support byte

I15031_PIDxxA_C



PIDxx structure

Element not
supported

Elements
that are not
available,
according to
the standard

Calibration of the Services \$01/\$02 as of MY 2010

General notes concerning the calibration:

- 1.) After the calibration, an Ini Phase must be completed.
- 2.) All signals of a PID must be present in the FreezFrame array (DFES_xAsgnFrzFrSig_CA).



Calibration of the Engine-speed threshold for Service\$04

- In Service\$04, an engine-speed threshold can be calibrated for which a deletion is not possible.
- The calibration label is: **I15031_nMaxSrv4_C**
- The minimum value is: **EPM_N_ZERO**
- The maximum value is: **EPM_N_MAX**
- The initial value of the preferred solution is: **EPM_N_MAX**



Calibration of the Infotypes for Service\$09

- ➔ For Service\$09, infotypes can be switched on or off per calibration.
- ➔ The calibration labels are:
 - **I15031_srv9PIDTable.PID0_C**
 - **I15031_srv9PIDTable.PID1_C**
 - ...
 - **I15031_srv9PIDTable.PIDB_C**
- ➔ An infotype is active if the calibration label has the corresponding value (value corresponds to infotype)



Calibration of the Infotypes for Service\$09

Label Name	Value (example)	Infotype value (example)
I15031_srv9PIDTable.PID0_C	0	Support information
I15031_srv9PIDTable.PID1_C	1	MessageCount VIN
I15031_srv9PIDTable.PID2_C	2	VIN
I15031_srv9PIDTable.PID3_C	3	MessageCount CALID
I15031_srv9PIDTable.PID4_C	4	CALID
I15031_srv9PIDTable.PID5_C	0	MessageCount CVN
I15031_srv9PIDTable.PID6_C	0	CVN
I15031_srv9PIDTable.PID7_C	0	MessageCount In-use Performance Tracking
I15031_srv9PIDTable.PID8_C	0	In-use Performance Tracking
I15031_srv9PIDTable.PID9_C	9	MessageCount ECUName
I15031_srv9PIDTable.PIDA_C	10	ECU Name
I15031_srv9PIDTable.PIDB_C	0	In-use Performance Tracking

Elements
not
supported



Calibration of the Infotypes for Service\$09

- ➔ Explanations to the Infotypes example
- ➔ In this example, the following infotypes are supported:
 - Support Information
 - MessageCount VIN
 - VIN
 - MessageCount CALID
 - CALID
 - MessageCount ECUName
 - ECU Name



Notes Regarding Integration

- The services must be entered in the Service Distributor "diagcom_distrib_confdata.xml."
- Sytem constants for I15031:
 - **DIABAS_VAR_SY** is used to differentiate different customer specific variants
 - 999 Diesel Reference Software
 - 147 Hyundai specific version of I15031
 - 22 Ford specific version of I15031
 - 11 BMW specific version of I15031
 - 5 Audi specific version of I15031
 - 3 VWW specific version of I15031
 - 1 Daimler specific version of I15031



Notes Regarding Integration

→ System constants for I15031:

- **CMBTYP_SY** to select Gasoline or Diesel version within a customer
 - CMBTYP_DS : select diesel version
 - CMBTYP_GS : select gasoline version
- **I15031_SRV8_SY** is used to enable Service 8
Currently only VW / Audi GS specific version of Service 8 is available.
 - 0 Service 8 disabled
 - 1 Service 8 enabled
- **SIG_I15031_LeanAppl_SY** is used to enable the feature “lean application”. Only used for Gasoline projects.
 - 0 feature disabled
 - 1 enabled



Notes Regarding Integration

→ System constants for I15031:

- **DIABAS_SCRCTL_RCNT_RESET_SY** is used to enable the feature of resetting the restriction counter in I15031
 - 0 feature disabled
 - 1 enabled
- additional system constants for I15031 (**only for VW/Audi Diesel**):
 - I15031APPL_NUMSIZETEXHFLD_SY
is used to determine the size of the exhaust temperature fields.
→ 0 all other projects
 - I15031APPL_NOXSENSNUM_VAG_SY
 - NOMCATDS_CALID_SY
 - NOCAT2DS_CALID_SY
 - HEGN_CVN_SENS1_SY
 - HEGN_CVN_SENS2_SY



Process

- **CE department**
 - Ordering of OBD signals in function development (FD)
 - Systems test, SAE J 1699
- **Function development**
 - Configuration of all OBD signals
- **Calibration**
 - Disabling of all OBD signals which are not supported
 - If necessary, temporary correction of the PID assignment (permanent change is done by the function developer (FD))



Process

- **Responsibility**
 - CE department is responsible for completeness (OBD)





Glossary of Terms

AA	= A utomotive A ftermarket
DSM	= D iagnostic S ystem M anager
DTR	= D iagnostic T est R esult
DFC	= D iagnostic F ault C heck
DTC	= D iagnostic T rouble C ode
ECU	= E lectronic C ontrol U nit
FD	= F unction D evelopment
IMA	= I njector M ass A justment
IUMPR	= I n U se M onitoring P erformance R atio
MY	= M odel y ear
OBD	= O n- B oard D iagnosis
OBDMID	= O n- B oard D iagnosis M onitor I D
ODX	= O pen D iagnostic D ata e Xchange
OEM	= O riginal E quipment M anufacturer
PID	= P arameter I dentification
TID	= T est I dentification
UDS	= U niversal D iagnostic S ervices
XDI	= E xtended D iagnostic I nformation



Information in the Intranet

- DGS-EC Training Centre

http://www.intranet.bosch.com/ds/esq/100_topics/500_fit/

- Seminar topic
Information in the intranet



Useful Information – Contacts DGS-EC/ESC

→ **ATS** and **ETC**: Gangolf Mansmann

Telephone: 0711 / 811 43556

E-mail: Gangolf.Mansmann@de.bosch.com

→ **I15031**: Frank Jach

Telephone: 0711 / 811 49257

E-mail: Frank.Jach@de.bosch.com

→ **Signals** and **AVS**: Michael Heinzelmann

Telephone: 0711 / 811 34645

E-mail: Michael.Heinzelmann3@de.bosch.com



Addendum, list of report examples

- **ATS**
- **AVS**
- **ETC**
- **Signals**





Configuration – Report File & Script Error Messages

ATS (Advanced Test Service) (version B_ATS.1.8.0 (eASEE version)) configuration report, run @ 2008-07-25 14:05:51

- 1) Total amount of configured actuator and sensor values
- 2) Total amount of configured actuator values
- 3) Total amount of configured sensor values
- 4) A quick overview of all actuators that are overwritten by freeze configuration
- 5) A quick overview of all sensors that are overwritten by freeze configuration
- 6) A quick overview of all created actuators
- 7) List of all actuators that are configured properly
- 8) A quick overview of all created sensors
- 9) List of all sensors that are configured properly
- 10) A quick overview of all actuators with warnings
- 11) List of all actuators with warnings
- 12) A quick overview of all sensors with warnings
- 13) List of all sensors with warnings
- 14) A quick overview of all not created actuators
- 15) List of all actuators that are not configured
- 16) A quick overview of all not created sensors
- 17) List of all sensors that are not configured

- 1) Total amount of configured actuator and sensor values
-

Number of actuator and sensor values :48

- 2) Total amount of configured actuator values
-

Number of actuator values :47





Configuration – Report File & Script Error Messages

7) List of all actuators that are configured properly

Actuator Name	:ATS_InjCrv_phiMI1Des
Description	:
Assigned ID	:2
Assigned ID in Hex	:0x02
Class	:analogue
Datatype	:SINT16
Type	:V
CompuMethod	:AngleCrS
Limit Type Mask	:0x00
Lower Limit	:-455
Upper Limit	:1820
Position	:No Signal number configured
Limit	:16
Offset	:0
Factor	:1
Norm	:0
Standby	:0
Deactivated	:FALSE
Callback function	:NULL
Generate systemconst	:FALSE
Configured in file	:C:/priv/mng2si/u_ATS.6.0.0_mng2si.prv/tmp/coreproc/corecfg/injcrv_confdata_conf.xml
Overwritten by freeze configuration	:NO
Freeze configuration file	:---



Configuration – Report File & Script Error Messages

11) List of all actuators with warnings

Actuator Name	:EGRVlv_O_P_ATS
Reason	:Default value outside the limits. Default value set to the lower limit by the script!



Addendum, list of report examples

- ATS
- AVS**
- ETC
- Signals



Configuration – Report File & Script Error Messages

```
-----
| AVS REPORT (configuration run at 17:02:05, 2009-09-17) |
|-----|
Total amount of configured adjustment values: 22
-----
| EEP-BLOCK "AvsPostDrive" |
|-----|
Please always cross-check the file "eep_auto_conf.h" if order is really correct.

...
-----
| EEP-BLOCK "AvsNoPostDrive" |
|-----|
Please always cross-check the file "eep_auto_conf.h" if order is really correct.

...
```



Configuration – Report File & Script Error Messages

DETAILED LIST OF CONFIGURED ADJUSTMENT VALUES

Adjustment value: AIRCTL_TRM_VAL

Access Macro:	AVS_AIRCTL_TRM_VAL_S16
ID (value for this configuration run):	Avs_IdAIRCTL_TRM_VAL_cu8 (1)
Adjustment value of type "flag":	false
Is element non-volatile?:	true
Writeback Postdrive?:	true
Module set function:	AirCtl_SetAdj_TrimVal
Module get function:	AirCtl_GetAdj_TrimVal
Module callback function:	NULL
Data type:	SINT16
Blocksize:	2
Number of control bytes:	0
Initialisationvalues for EEPROM:	0

REMARKS:
none

File reference:
O:/medcl7/asw/eng/gssys/airsys/egrctl/airctl/_conf/airctl_conf.xml



Addendum, list of report examples

- ATS
- AVS
- ETC**
- Signals



Configuration – Report File & Script Error Messages

ETC (version B_ETC.6.0.0) configuration report, run @ 2008-10-09 14:44:13

- 1) A quick overview of all created tests
- 2) All tests that were configured properly
- 3) All tests that were not configured because buffer size of return values greater than max buffer size
- 4) All tests that were configured more than once and therefore can't be used
- 5) All tests that were not configured properly and therefore can't be used
- 6) List of all tests that were missing the number of arguments. (Number of arguments was set to 0)
- 7) List of all tests that were missing the number of return values. (Number of arguments was set to 0)
- 8) List of all tests that were missing the total size of the return value buffer.
- 9) List of all double declared ETC_FUNCTIONS.)
- 10) List of all abort reasons
- 11) List of all erroneous abort reasons
- 12) List of all abort reasons with warnings
- 13) Locking matrix
- 14) Resource report
- 15) General information



Configuration – Report File & Script Error Messages

2) All tests that were configured properly

```
-----
Test ID                               :1
Name of test                          :DSD
Description                           :function diagnostic engine speed demand
Number of arguments                   :12
Number of return vals per set         :2
Total Number of return values         :2
Functions of test component           :
  Start permission                    :DSD_StrtSetP
  Stop permission                     :DSD_StopSetP
  Get status                          :DSD_GetStSetP
  Get abort reason                    :DSD_GetAbortReasonSetP
Verbal conversion in C- code          :EtclId_DSD
Configured in file                    :
  C:/views/snapshot/u_DiagInf_snap_mng2si/medc17/tmp/coreproc/corecfg/dsd_setpoint_confdata_conf.xml
Bit coded abort reasons                : Bitposition - Description  :0 - "FID"
  Bitposition - Description            :1 - "CLUTCH"
  Bitposition - Description            :2 - "SPEED_NEZ"
  Bitposition - Description            :3 - "BRAKE_PRESS"
Value coded abort reasons              :
Faulty value abort reasons            :
Warnings                              :Return value expected but ETC_USED_SIGNAL and ETC_RETURNVALUE
  empty
```



Addendum, list of report examples

- ATS
- AVS
- ETC
- Signals**



Configuration – Report File & Script Error Message

Example

Signals (version B_SIGNALS.4.0.0) configuration report, run @ 2008-06-16 09:54:10

-
- 1) Quick info
 - 2) List of all signals that are configured properly
 - 3) List of all virtual signals that are configured properly
 - 4) List of all messages that are not implemented
 - 5) Conversion formulae
 - 6) Carb Services PID list in INCA
 - 7) Error list
 - 8) Common information
 - 9) Switch off information

1) Quick info

Number of correct configured Signals	:16
Number of standard Signals	:16
Number of virtual Signals	:0
Number of overwritten Implementations	:0
Number of overwritten Messages	:0
Faulty (and therefore unused) Signals	:2 (See chapter 7, Error list)
Unique identifier are used	:FALSE
Switch off configuration is used	:FALSE



Configuration – Report File & Script Error Message

Chapter 2, detailed description of all correctly configured signals:

2) List of all signals that are configured correctly

Signal name	:EpmCaS_phiOfsCorr
Description	:filtered value of camshaft offset
Assigned message	:sint16 EpmCaS_phiOfsCorr
Assigned internal signal number	:1
Verbal conversion in C-Code	:SIGNALS_EpmCaS_phiOfsCorr
Verbal conversion in INCA	:SIGNALS_EpmCaS_phiOfsCorr
External signal number	:---
Assigned PID	:---
Offset in this PID	:---
Index of assigned conversion for DIA	:0
Length in Bits for DIA	:16
Index of assigned conversion for CAN	:0
Length in Bits for CAN	:16
Configuration file of message	:C:/priv/hmn2fe/.../epm_prot_confdata_conf.xml
Configuration file of implementation	:C:/priv/hmn2fe/.../epm_prot_confdata_conf.xml
Support of signed values	:FALSE
DIAP switch:	:NO
List of warnings	:No warnings





Configuration – Report File & Script Error Message

Detailed description of all generated conversion formulas:

5) Conversion formulae

Linear conversion formulae(used as $\text{external} = \text{internal} * (\text{Factor} \gg \text{Norm}) + \text{Offset}$)

Index	:0	→ Please note: The index is used for referencing in a signal.
Offset	:32767	→ Offset as integer value, as it is used in the ECU software
Factor	:32790	→ Slope factor as integer value, as it is used in the ECU software
Normalization of factor	:15	→ Quantization basis for the slope factor
Resolution of factor	:0.000031	→ Resolution of the slope factor based on the quantization basis
Real factor	:1.000680	→ For accuracy check: The slope factor as realistic value
Calculated factor	:1.000671	→ Here is the slope factor in the integer/quantization representation in order to show the deviation of the actual realistic value

Index	:1
Offset	:0
Factor	:8192
Normalization of factor	:12
Resolution of factor	:0.000244
Real factor	:2.000000
Calculated factor	:2.000000



Configuration – Report File & Script Error Message

List of all errors:

List of all errors which occurred during the configuration:

- Signals which were discarded for any reason
(incl. description of the reasons)
- Signals which have been used in the software, but for which fundamental corrections have been carried out
(e.g. removal of a PID used twice – since this can be corrected via calibration, the signal is retained)



Configuration – Report File & Script Error Message

7) Error list

```
-----  
                                     Faulty configured standard signals  
-----  
Signal name:PID4FAh  
Reason      :(NO FAULT) Switched off by master configuration  
File        :M:/u_siz8484_coreintegration_hmn2fe/medc17/core/inf/diaginf/../../../../com/diagsrv/i15031/  
              _conf/i15031_conf.xml
```

This example shows a signal which has been removed, but only because of the master configuration, which permits the disabling of correctly configured signals in a project.



Configuration – Report File & Script Error Message

General information:

As an example, here are signals which have been overwritten. This means, there are 2 identical configurations of a signal in one project, where one had a higher priority than the other and the lower priority configuration was thus discarded. This information is given directly in chapter 1) for the affected signal, in addition to the overview in this chapter.

8) Common information

```
-----
Configured, but overwritten Implementations
-----
:No overwritten Implementations

Configured, but overwritten ECU-variables
-----
:No overwritten ECU- variables

"External" declared compu methods
-----
:No external compumethods found
```

