VOLKSWAGEN

AKTIENGESELLSCHAFT

CONFIDENTIAL
VERTRAULICH

UNECE Softwareupdate - General Diagnostic Requirements

Supplementary Specification for Electronic Control Units

Development, Feature Performance Specification: LAH.DUM.905.E

Author	Peter-Michael Hofmann (EESN/4), Dora Aranyi (I/EE-87), Dr. Markus Koch (EEY3)
Dept./OU	EESN/4, I/EE-87, EEY3
Phone	+49 5361 9 78398, +49 841 89 91654, +49 711 911 83633
Cell	-
Fax	-
E-mail	peter.hofmann@volkswagen.de, dora.aranyi@audi.de, markus.koch@porsche.de

First issue	2019-11-22
Date of revision	2019-12-03
Performance Specification version	1.0.1
Baseline	2.8 ( )

Contents

1 General

1.1 Purpose

[I: F-LAH_RxSWIN-7]

This document describes the technical requirements for onboard and offboard systems for compliance with the United Nations Economic Commission for Europe "UNECE Regulation on Software Updates" by the "Working Party on Automated/Autonomous and Connected Vehicles (GRVA)" from WP.29 ("Regulation on Uniform Provisions Concerning the Approval of Software Update Processes").

[I: F-LAH_RxSWIN-712]

All software for the implementation of vehicle features must be developed in accordance with the software update management system. Additionally, vehicle features subject to mandatory type approval must be assigned a Regulation X Software Identification Number (RxSWIN). This applies, for example, to UNECE or GB/T regulations. These requirements must be met as a mandatory requirement for receiving type approval.

[I: F-LAH_RxSWIN-433]

The following diagnostic requirements arise from the UNECE Regulation on Software Updates:

[I: F-LAH_RxSWIN-732]

- Authenticated software installation (instruction code and data)
  Reference: UNECE Regulation on Software Updates, 7.2.1.1. - The authenticity and integrity of software updates shall be protected to reasonably prevent their compromise and reasonably prevent invalid updates.
  Implementation: Use of flash data security (FDS) or an alternative, at least equivalent validation method and Protection of Vehicle Diagnostics (PVD)

[I: F-LAH_RxSWIN-733]

- Introduction of RxSWIN for vehicle features subject to mandatory type approval
  Reference: UNECE Regulation on Software Updates, 2.2. - "Regulation X Software Identification Number (RXSWIN)" means a dedicated identifier, defined by the vehicle manufacturer, representing information about the type approval relevant software of the Electronic Control System contributing to the Regulation N° x type approval relevant characteristics of the vehicle.
  Implementation: Introduction of an RxSWIN DataIdentifier (DID) in the gateway

[I: F-LAH_RxSWIN-734]

- Introduction of a software integrity attribute (instruction code and data)
  Reference: UNECE Regulation on Software Updates, 7.1.2.3. - For every RxSWIN, there shall be documentation describing the software relevant to the RxSWIN of the vehicle type before and after an update. This shall include information of the software versions and their integrity validation data for all relevant software for each RxSWIN.
  Implementation: Introduction of integrity validation data for the program and data

[I: F-LAH_RxSWIN-552]

For vehicles with a new type approval for the Japanese market as of January 2022 and for the EU market as of May 2022, software updates must not be performed without the technical prerequisites (RxSWIN) and without software update management system (SUMS) certification. Due to the resulting inability to perform software updates and cybersecurity type-approval testing, type approval of the vehicle is not possible! This will be extended to all existing vehicles for the Japanese market as of January 2024 and for the EU market as of May 2024.

[I: F-LAH_RxSWIN-430]

The electronic control units (ECUs) will be classified on the basis of the platform(s) to which they belong. A distinction will be made between existing platforms and new platforms:

[I: F-LAH_RxSWIN-827]

- Existing platforms include, e.g., 2nd generation modular longitudinal matrix evolution (MLBevo Gen2), modular transverse matrix (MQB), modular electric drive matrix (MEB)

[I: F-LAH_RxSWIN-828]

- New platforms include: MEB 1.1 Advanced, E³ 1.2 Premium as well as E³ 2.0, and all subsequent platforms

[I: F-LAH_RxSWIN-432]

The following requirements apply to vehicle projects with type approval as of January 2022:

[I: F-LAH_RxSWIN-825]

All programmable or configurable ECUs:

- ECU programming data security (EPDS) for software
- Protection of Vehicle Diagnostics (PVD end-to-end (E2E) security) for data
    - Exception for diagnostic class 2 system: Validation is performed via the higher-level diagnostic class 4-low system.
- Integrity validation data for software and data

[I: F-LAH_RxSWIN-826]

Gateway ECU in addition to:

- Implementation of a diagnostic filter with PVD access protection
- RxSWIN with PVD end-to-end security

[I: F-LAH_RxSWIN-746]

The same requirements apply to all vehicle projects (including existing platforms) as of January 2024:

- Fine tuning of the requirements for existing platforms will take place in the GRVA by around the end of February 2020. After the final regulation becomes available, the RxSWIN Feature Performance Specification will be adapted as necessary.

1.2 Abbreviations and terms

[I: F-LAH_RxSWIN-21]

Table 1-1: Abbreviations and terms

Abbreviation or term	Designation	Definition
BOM	Bill of materials	Build status documentation
BSB	Component engineer	Role designation at Audi AG, equivalent to "BTV – part owner" at Volkswagen AG
C	Conditional	Cvt. = C: Must be transmitted/implemented subject to specific conditions
Cvt.	Convention	Implementation rules and conventions that apply to the parameters of a service
EPDS	ECU programming data security	Generic term for a security mechanism that uses diagnostics to determine the authenticity and integrity of an ECU program version. This can be done by implementing FDS, or using an alternative, at least equivalent validation method that has been agreed upon with the purchaser's E/E Security department.
FDS	Flash data security	Cryptographic method for validating the flash data
GB/T	GB stands for Guobiao, which is Chinese for national standard.	This is the basis for the product testing that the product must pass in the course of China Compulsory Certification (CCC).
IVD	Integrity validation data	Hashes for software (instruction code) and configuration data (data)
M	Mandatory	Cvt. = M: Must be implemented or must always be transmitted for the application software
NoImp	No implementation	This must not be implemented for servers commissioned by Volkswagen AG.
RxSWIN	Regulation X Software Identification Number	Software identification number for each pertinent regulation
PVD	Protection of Vehicle Diagnostics	Cryptographic validation method a) Access to diagnostic objects (PVD access protection) b) Diagnostic content (PVD end-to-end security)
SUMS	Software update management system	Ensures compliance with legal requirements relating to the provision of software updates through corresponding processes at the manufacturer
SW	Software	According to UNECE R SU, software means the instruction code and data.
U	User-optional	Cvt. = NoImp: This must not be implemented for servers commissioned by Volkswagen AG.
UNECE	United Nations Economic Commission for Europe	
UNECE R SU	UNECE Regulation on Software Updates	Official title: "Regulation on Uniform Provisions Concerning the Approval of Software Update Processes"
ZDC	Target data container	XML file containing all parameters of a server's variants that are selected using primary properties (PR) codes.
CGW	Central (networking) gateway	ECU with central diagnostic access, e.g., a gateway, In-Car Application Server ICAS1, high-performance computing platform HCP5
Reserved by Volkswagen AG		Use is reserved for future Volkswagen AG applications.

1.3 Scope of validity

[allg. Anf.: F-LAH_RxSWIN-9]

This document applies to all programmable and configurable ECUs in existing platforms and new platforms.

[allg. Anf.: F-LAH_RxSWIN-550]

If there are any discrepancies between the requirements in this document and other Diagnostics Performance Specifications, then the requirements from this Performance Specification apply.

[I: F-LAH_RxSWIN-554]

The next releases of the General Project-Independent Performance Specification for Diagnostics will merge the requirements from this Performance Specification with the requirements from the other documents.

[I: F-LAH_RxSWIN-521]

The color coding of the tables is only for legibility.

[allg. Anf.: F-LAH_RxSWIN-918]

Requirements marked as a "process requirement" must not be taken into account in the context of ECU development.

2 Requirements for ECUs

[I: F-LAH_RxSWIN-917]

Table 2-1: Overview of the requirements from a diagnostic class perspective

	Central gateway	Diagnostic class 4	Diagnostic class 3	Diagnostic class 2F	Diagnostic class 2	Diagnostic class 1/0
RxSWIN DataIdentifier	X	NoImp	NoImp	NoImp	NoImp	NoImp
Diagnostic filter (OBD port)	X	NoImp	NoImp	NoImp	NoImp	NoImp
ECU programming data security (EPDS)	X	X	X 3)	X	NoImp	NoImp
Protection of Vehicle Diagnostics **- PVD authentication** **- PVD end-to-end**	X	X	X 1)	NoImp 2)	NoImp 2)	NoImp
Integrity validation data (IVD)	X	X 3)	X 3)	X 3)	X 3)	NoImp

[I: F-LAH_RxSWIN-922]

x = included

[I: F-LAH_RxSWIN-919]

1) = If a ZDC cannot be used for configuration purposes in the application, then the use of PVD depends on the risk/security analysis.

[I: F-LAH_RxSWIN-923]

2) = Receive data without a signature from diagnostic class 4

[I: F-LAH_RxSWIN-943]

3) = Only for calibratable (programmable and/or configurable) diagnostic servers

[I: F-LAH_RxSWIN-920]

All requirements also apply to carry-over parts.

[I: F-LAH_RxSWIN-921]

All requirements are minimum requirements from the UNECE Regulation on Software Updates. The risk/security analysis can lead to stricter requirements.

2.1 Diagnostic class 0/1 systems

[I: F-LAH_RxSWIN-709]

Diagnostic class 0/1 systems are not affected by the requirements in this document since they are not calibratable.

2.2 Diagnostic class 2/2F/2FV systems

[I: F-LAH_RxSWIN-941]

Non-calibratable (neither programmable nor configurable) diagnostic class 2/2F/2FV systems are not affected by this document.

2.2.1 Programmable diagnostic class 2F/2FV systems

[allg. Anf.: F-LAH_RxSWIN-850]

Programmable diagnostic class 2F/2FV systems must:

[allg. Anf.: F-LAH_RxSWIN-754]

- Support ECU programming data security.

[allg. Anf.: F-LAH_RxSWIN-847]

- Support integrity validation data for programming. See section "Integrity validation data."

2.2.2 Configurable diagnostic class 2/2F/2FV systems

[allg. Anf.: F-LAH_RxSWIN-851]

Configurable diagnostic class 2/2F/2FV systems:

[allg. Anf.: F-LAH_RxSWIN-755]

- Are configured using ZDCs.

[allg. Anf.: F-LAH_RxSWIN-756]

- Receive data without a signature from the diagnostic class 4-low system. The diagnostic class 4-low system performs the PVD E2E signature check for the diagnostic class 2/2F/2FV system.

[allg. Anf.: F-LAH_RxSWIN-757]

- Must support integrity validation data for the configuration. See section "Integrity validation data."

2.3 Diagnostic class 3/3V/4/4V systems and SWCLs

[I: F-LAH_RxSWIN-942]

Non-calibratable (neither programmable nor configurable) diagnostic class 3/3V/4/4V systems and software clusters (SWCLs) are not affected by this document.

2.3.1 Programmable diagnostic class 3/3V/4/4V systems and SWCLs

[allg. Anf.: F-LAH_RxSWIN-852]

Programmable diagnostic class 3/3V/4/4V systems and SWCLs must:

[allg. Anf.: F-LAH_RxSWIN-759]

- Support ECU programming data security.

[allg. Anf.: F-LAH_RxSWIN-849]

- Support integrity validation data for programming. See section "Integrity validation data."

2.3.2 Configurable diagnostic class 3/3V/4/4V systems and SWCLs

[allg. Anf.: F-LAH_RxSWIN-853]

Configurable diagnostic class 3/3V/4/4V systems and SWCLs:

[allg. Anf.: F-LAH_RxSWIN-760]

- Are configured using ZDCs.

[allg. Anf.: F-LAH_RxSWIN-761]

- Must support PVD E2E security if they are configurable in the application. See section "PVD E2E security."

[allg. Anf.: F-LAH_RxSWIN-762]

- Must support integrity validation data for the configuration. See section "Integrity validation data."

2.3.3 Additional requirements for diagnostic class 4-low/4V-low systems

2.3.3.1 Group DataIdentifier

[allg. Anf.: F-LAH_RxSWIN-804]

Diagnostic class 4-low/4V-low systems must:

[allg. Anf.: F-LAH_RxSWIN-506]

- Output the group DataIdentifier for integrity validation data of the configuration for diagnostic class 2/2F/2FV systems. See section "Integrity validation data."

[allg. Anf.: F-LAH_RxSWIN-788]

- Output the group DataIdentifier for integrity validation data of the programming for diagnostic class 2F/2FV systems. See section "Integrity validation data."

2.3.3.2 Routing mechanism in the diagnostic class 4-low/4V-low system for a diagnostic class 2F/2FV system

[I: F-LAH_RxSWIN-59]

The activation of the routing mechanism in the diagnostic class 4-low/4V-low system is started by the CommunicationControl (28hex) service with ControlType [0x04-enableRxAndDisableTxWithEnhancedAddressInformation] as per document /3/.

[allg. Anf.: F-LAH_RxSWIN-57]

A diagnostic class 4-low/4V-low system must protect the CommunicationControl (28hex) service with PVD access protection for ControlType [0x04-EnableRxAndDisableTxWithEnhancedAddressInformation].

[allg. Anf.: F-LAH_RxSWIN-78]

PVD access protection with the "Basic" role must be implemented for ControlType [0x04-enableRxAndDisableTxWithEnhancedAddressInformation].

[allg. Anf.: F-LAH_RxSWIN-79]

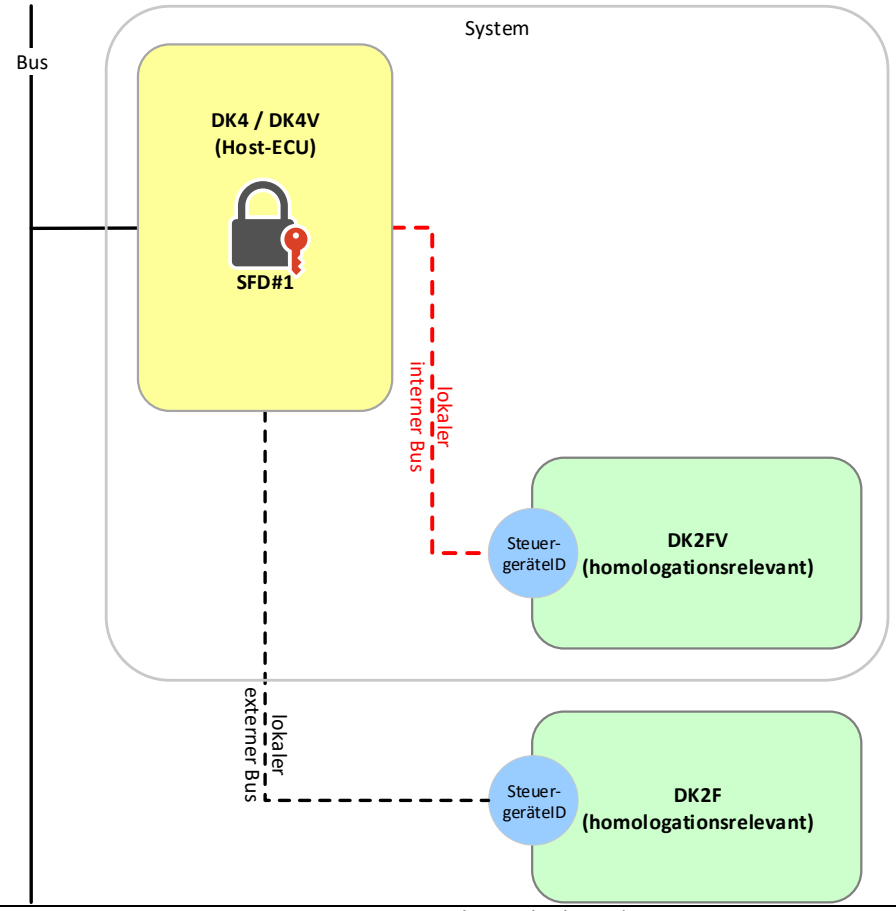
Contrary to requirement 80127-3047 up to v5.8, the following applies:
Permanently implementing the routing of diagnostic communication in normal operation in the diagnostic class 4 low system is explicitly prohibited for all diagnostic class 2F systems.

[allg. Anf.: F-LAH_RxSWIN-80]

If a diagnostic class 4-low/4V-low system and a diagnostic class 2F system are connected to the same bus segment and this bus segment is used for diagnostic communication for both systems, then the security aspects must be agreed upon with the Diagnostics department and the E/E Security department.

[I: F-LAH_RxSWIN-104]

Figure 2-1: Diagnostic class 4-low RoutingActivation, protected by PVD access protection



DK4 / DK4V (Host-ECU)	Diagnostic class 4/4V (host ECU)
SFD#1	PVD#1
lokaler interner Bus	Local internal bus
lokaler externer Bus	Local external bus
SteuergeräteID	ECU ID
DK2FV (homologationsrelevant)	Diagnostic class 2FV (relevant to homologation)
DK2F (homologationsrelevant)	Diagnostic class 2F (relevant to homologation)

2.3.4 Additional requirements for gateway ECUs

[allg. Anf.: F-LAH_RxSWIN-429]

The following additional requirements apply to ECUs with central diagnostic access:

2.3.4.1 RxSWIN

[allg. Anf.: F-LAH_RxSWIN-470]

ECUs with central diagnostic access must implement the DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" as per section "Diagnostics objects."

2.3.4.2 "Diagnostic filter" function

[I: F-LAH_RxSWIN-625]

The diagnostic filter function is implemented on the gateway ECU. It prevents the routing of diagnostics requests with write requests (coding, adaptation/calibration, data set download (DSDL), programming) during driving. Read access (e.g., identification, event memory) and generic scan tool communication is possible without restriction.

[allg. Anf.: F-LAH_RxSWIN-476]

Based on document /9/, the diagnostic filter function must be implemented on the gateway ECU with the following changes:

[allg. Anf.: F-LAH_RxSWIN-630]

- The hood contact switch must no longer be used to deactivate the diagnostic filter as of the implementation of this Performance Specification.

[allg. Anf.: F-LAH_RxSWIN-823]

- The "PVD access protection, BASIC role" SecurityLevel is required to deactivate the diagnostic filter.

[allg. Anf.: F-LAH_RxSWIN-767]

- The activated diagnostic filter must also prevent routing of diagnostics requests with write requests via service-oriented diagnostics.

[allg. Anf.: F-LAH_RxSWIN-824]

- The DataIdentifiers of the diagnostic filter function must be supported as per the definition in this document.

[allg. Anf.: F-LAH_RxSWIN-338]

- Deactivation of the diagnostic filter by the SecurityAccess (27hex) service is not permitted.

[I: F-LAH_RxSWIN-541]

Note: Only one access protection method for diagnostics objects is permissible in an ECU's application. In other words, all diagnostics objects must be protected uniformly by means of PVD access protection. As per document /4/, the implementation of PVD access protection on an ECU with central diagnostic access requires that the SecurityAccess (service 27hex) service be deleted from the application.

[allg. Anf.: F-LAH_RxSWIN-218]

Table 2-2: Overview of the properties of the diagnostics objects for the "diagnostic filter" function

			DiagnosticSession		
			Default Session	NonDefaultSession	
				ECUProgram-mingSession	Extended-Diagnostic-Session
Diagnostic objects (DID)	**SID**	**Security Level**	**0x01**	**0x02**	**0x03**
0x0BEE-Diagnostic_filter_activation	22hex	NoImp	Available	notAvailable	Available
	2Ehex	SFD-Basic	notAvailable	notAvailable	Available
0x0BEF-Diagnostic_filter_vehicle_mileage_ thresholds	22hex	NoImp	Available	notAvailable	Available
	2Ehex	at least SFD-Basic	notAvailable	notAvailable	Available
0x539B-Diagnostic_filter_status	22hex	NoImp	Available	notAvailable	Available
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable
0x539C-Diagnostic_filter_life_cycle_data	22hex	NoImp	Available	notAvailable	Available
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable
0x05EF-Diagnostic_filter_permanent_deactivation	22hex	NoImp	Available	notAvailable	Available
	2Ehex	SFD-Ext.	notAvailable	notAvailable	Available

2.3.4.2.1 0x0BEE-Diagnostic_filter_activation

[I: F-LAH_RxSWIN-731]

This DataIdentifier is used to initially activate the "diagnostic filter" function on the diagnostic server. This DataIdentifier is protected by the "PVD access protection, BASIC role" SecurityLevel.

[allg. Anf.: F-LAH_RxSWIN-696]

Table 2-3: Structure of DataRecord for 0x0BEE-Diagnostic_filter_activation DataIdentifier

DID	**0x0BEE**
Designation	Diagnostic_filter_activation
Description	This DataIdentifier is used to activate/deactivate the "diagnostic filter" function.
Convention	ECU with central diagnostic access
Diagnostic class	Diagnostic class 4-high
Session	APP: 0x01 (R), 0x03 (R/W) BLF: NoImp
SecurityLevel	PVD access protection, BASIC role
Changing	DIAG
Format	<[Diagnostic_filter_activation] 1-byte hex>
Range	[Diagnostic_filter_activation] 00hex = Function is not active. 01hex = Function is active. 02hex – FFhex = Reserved by Volkswagen AG
Init	00Hex = Function is not active.
Example	01hex = Function is active.
Data category	Workshop parameters
Electronic build status documentation	NoImp

2.3.4.2.2 0x0BEF-Diagnostic_filter_vehicle_mileage_thresholds

[I: F-LAH_RxSWIN-724]

This DataIdentifier is used to configure the mileage threshold at which the diagnostic filter is automatically activated. This DataIdentifier is at least protected by the "PVD access protection, BASIC role" SecurityLevel.

[allg. Anf.: F-LAH_RxSWIN-720]

Table 2-4: Structure of DataRecord for 0x0BEF-Diagnostic_filter_vehicle_mileage_thresholds DataIdentifier

DID	**0x0BEF**
Designation	Diagnostic_filter_vehicle_mileage_thresholds
Description	This DataIdentifier is used to configure the mileage threshold for automatic activation of the diagnostic filter.
Convention	ECU with central diagnostic access
Diagnostic class	Diagnostic class 4-high
Session	APP: 0x01 (R), 0x03 (R/W) BLF: NoImp
SecurityLevel	PVD access protection, at least BASIC role
Changing	DIAG
Format	<<[Vehicle_mileage_production] 1-byte hex> + <[Vehicle_mileage_service] 1-byte hex>>
Range	[Diagnostic_filter_production] 00hex – FFhex [Vehicle_mileage_service] 00hex – FFhex
Init	C8 14hex
Example	00 14hex
Data category	Workshop parameters
Electronic build status documentation	NoImp

2.3.4.2.3 0x05EF-Diagnostic_filter_permanent_deactivation

[I: F-LAH_RxSWIN-216]

This DataIdentifier is used to permanently deactivate the diagnostic filter. This DataIdentifier is protected by the "PVD access protection, EXTENDED role" SecurityLevel.

[allg. Anf.: F-LAH_RxSWIN-722]

Table 2-5: Structure of DataRecord for 0x05EF-Diagnostic_filter_permanent_deactivation DataIdentifier

DID	**0x05EF**
Designation	Diagnostic_filter_permanent_deactivation
Description	This DataIdentifier is used to permanently deactivate the diagnostic filter.
Convention	ECU with central diagnostic access
Diagnostic class	Diagnostic class 4-high
Session	APP: 0x01 (R), 0x03 (R/W) BLF: NoImp
SecurityLevel	PVD access protection, EXTENDED role
Changing	DIAG
Format	<[Diagnostic_filter_permanent_deactivation] 1-byte hex>
Range	[Diagnostic_filter_permanent_deactivation] 00hex = Filter not activated 01hex = Filter activated 02hex – FFhex = Reserved by Volkswagen AG
Init	01hex
Example	00hex = Filter not activated
Data category	Workshop parameters
Electronic build status documentation	NoImp

2.3.4.2.4 0x539B-Diagnostic_filter_status

[I: F-LAH_RxSWIN-665]

Deactivation of the diagnostic filter is linked to various conditions. These conditions must be output using the "0x539B-Diagnostic_filter_status" DataIdentifier. Additionally, the deactivation conditions must be documented as deactivation reasons using this DataIdentifier.

[allg. Anf.: F-LAH_RxSWIN-694]

Table 2-6: Structure of DataRecord for 0x539B-Diagnostic_filter_status DataIdentifier

DID	**0x539B**
Designation	Diagnostic_filter_status
Description	This DataIdentifier contains information about the status of the diagnostic filter. - The [Funktionsstatus] (function status) parameter indicates the function activation type. - The [Filterstatus] (filter status) parameter indicates whether the diagnostic filter is active. - The [Kilometerstand] (mileage) parameter contains the current odometer value after deactivating the filter. - The [Deaktivierungsgrund] (deactivation reason) parameter contains information on the reason for deactivating the filter.
Convention	ECU with central diagnostic access
Diagnostic class	Diagnostic class 4-high
Session	APP: 0x01 (R), 0x03 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	DIAG
Format	<<[Funktionsstatus] (function status) 1-byte hex> + <[Filterstatus] (filter status) 1-byte hex> + <[Kilometerstand] (mileage) 1-byte hex> + <[Deaktivierungsgrund] (deactivation reason) 1-byte hex>>
Range	[Funktionsstatus] (function status) 00hex = Function is not active. 01hex = Reserved by Volkswagen AG 02hex = Function active – Filter deactivated by hood 03hex = Function active – Filter deactivated by button combination 04hex = Function active – Filter deactivated by access protection method 05hex = Function active – Filter deactivated by hood and button combination 06hex – FFhex= Reserved by Volkswagen AG [Filterstatus] (filter status) 00hex = Filter is not active. 01hex = Filter is active. 02hex – FFhex = Reserved by Volkswagen AG [Kilometerstand] (mileage) 00hex – FFhex [Deaktivierungsgrund] (deactivation reason) 00hex = Not deactivated 01hex = Hood open 02hex = Electronic central electric system not available 03hex = Crash detected 04hex = Button combination 05hex = Access protection method 06hex = Production 07hex = Permanently deactivated 08hex – FFhex = Reserved by Volkswagen AG
Init	No initial value on the server
Example	00 01 11 01 [Funktionsstatus] (function status): Filter deactivated by hood [Filterstatus] (filter status): Active [Kilometerstand] (mileage): 17 m [Deaktivierungsgrund] (deactivation reason): Hood open
Data category	Analysis data
Electronic build status documentation	NoImp

2.3.4.2.5 0x539C-Diagnostic_filter_life_cycle_data

[I: F-LAH_RxSWIN-723]

This DataIdentifier can be used to read out the life cycle data.

[allg. Anf.: F-LAH_RxSWIN-721]

Table 2-7: Structure of DataRecord for 0x539C-Diagnostic_filter_life_cycle_data DataIdentifier

DID	**0x539C**
Designation	Diagnostic_filter_life_cycle_data
Description	This DataIdentifier contains life cycle data for the diagnostic filter.
Convention	ECU with central diagnostic access
Diagnostic class	Diagnostic class 4-high
Session	APP: 0x01 (R), 0x03 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	DIAG
Format	<<[Life_cycle_data_1] 31-byte ASCII (80125)> + <[Life_cycle_data_2] 31-byte ASCII (80125)> + <[Life_cycle_data_3] 31-byte ASCII (80125)> + <[Life_cycle_data_4] 31-byte ASCII (80125)> + <[Life_cycle_data_5] 31-byte ASCII (80125)>>
Range	[Life_cycle_data_1] 31-byte ASCII (80125) [Life_cycle_data_2] 31-byte ASCII (80125) [Life_cycle_data_3] 31-byte ASCII (80125) [Life_cycle_data_4] 31-byte ASCII (80125) [Life_cycle_data_5] 31-byte ASCII (80125)
Init	2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2Dhex
Example	-
Data category	Analysis data
Electronic build status documentation	NoImp

3 PVD requirements

3.1 PVD E2E security

[allg. Anf.: F-LAH_RxSWIN-82]

Diagnostic servers that implement PVD end-to-end security as per section "Requirements for ECUs" must use document /4/ (Q-LAH PVD Version 2.1 incl. Errata for Version 2.1).

[allg. Anf.: F-LAH_RxSWIN-341]

The following data categories (data types) as per document /8/ that are written via the WriteDataByIdentifier (2Ehex) service must be secured by PVD end-to-end and documented correspondingly in the diagnostic data tables of the Component Performance Specification (BT-LAH):

[allg. Anf.: F-LAH_RxSWIN-811]

- Coding (equipment-dependent switching of sub-functions ON/OFF)

[allg. Anf.: F-LAH_RxSWIN-810]

- Vehicle parameters (vehicle-specific parameters and settings) that are transferred as application data sets

[allg. Anf.: F-LAH_RxSWIN-809]

- Initial calibration values (one-time (initially) writable default values, parameters, and settings)

3.1.1 Special case for diagnostic class 4-low/4V-low systems with lower-level diagnostic class 2/2F/2FV systems

[I: F-LAH_RxSWIN-932]

The following figures show examples of sub-sequences; the sub-sequences must not be implemented.

[I: F-LAH_RxSWIN-101]

Diagnostic class 2/2F/2FV systems do not implement the PVD with end-to-end security themselves.

[allg. Anf.: F-LAH_RxSWIN-100]

A diagnostic class 4-low/4V-low system must use PVD end-to-end security to protect the configuration data, writable using the WriteDataByIdentifier (2Ehex) service, of a lower-level diagnostic class 2/2F/2FV system against unauthorized access as per the requirement "F-LAH_RxSWIN-341." Deviations must be agreed upon with the purchaser's Security department.

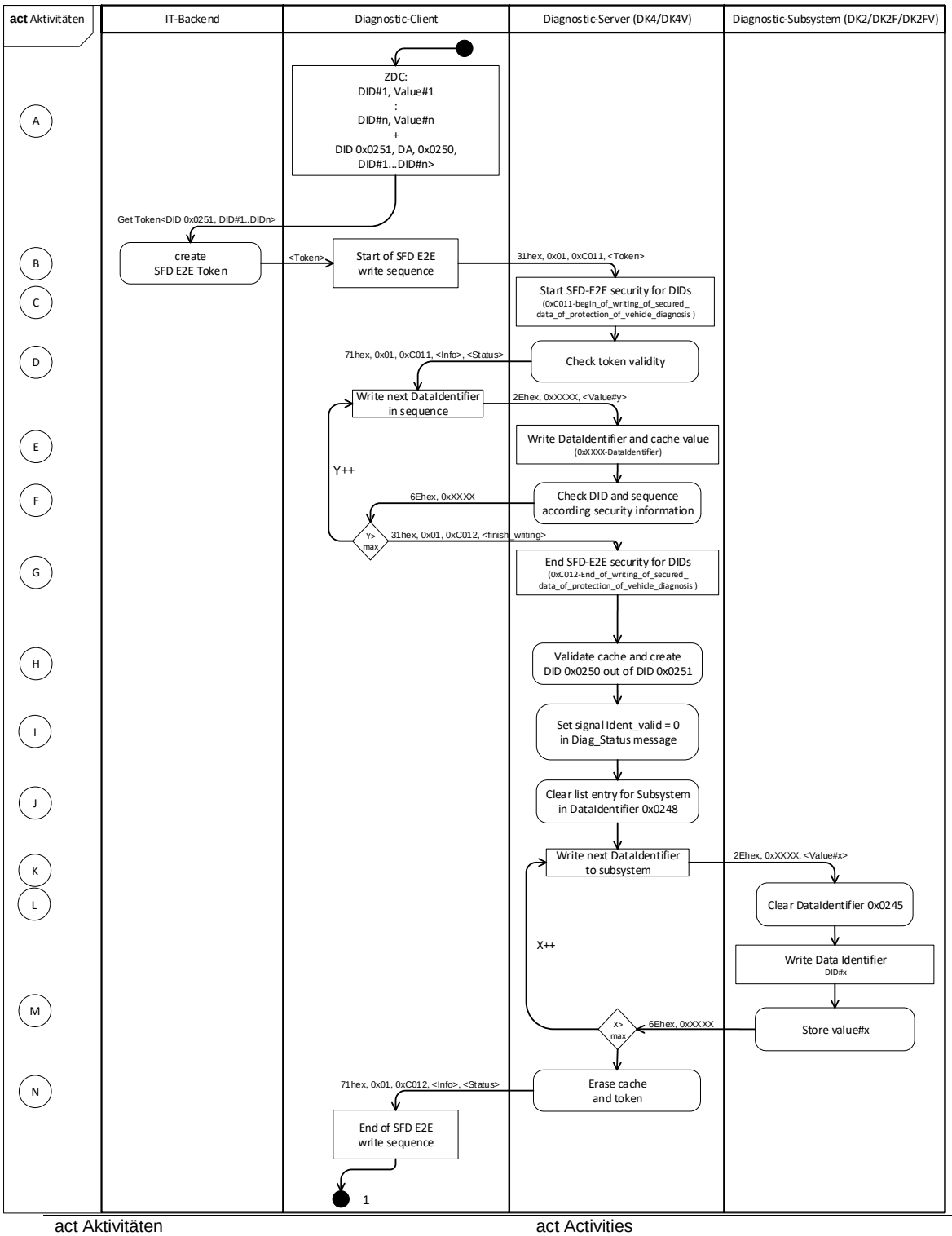
3.1.1.1 Writing PVD end-to-end secured data to a diagnostic class 2/2F/2FV system

[I: F-LAH_RxSWIN-280]

A diagnostic client uses a diagnostic class 4-low/4V-low system to write PVD end-to-end secured data to a lower-level diagnostic class 2/2F/2FV system.

[I: F-LAH_RxSWIN-281]

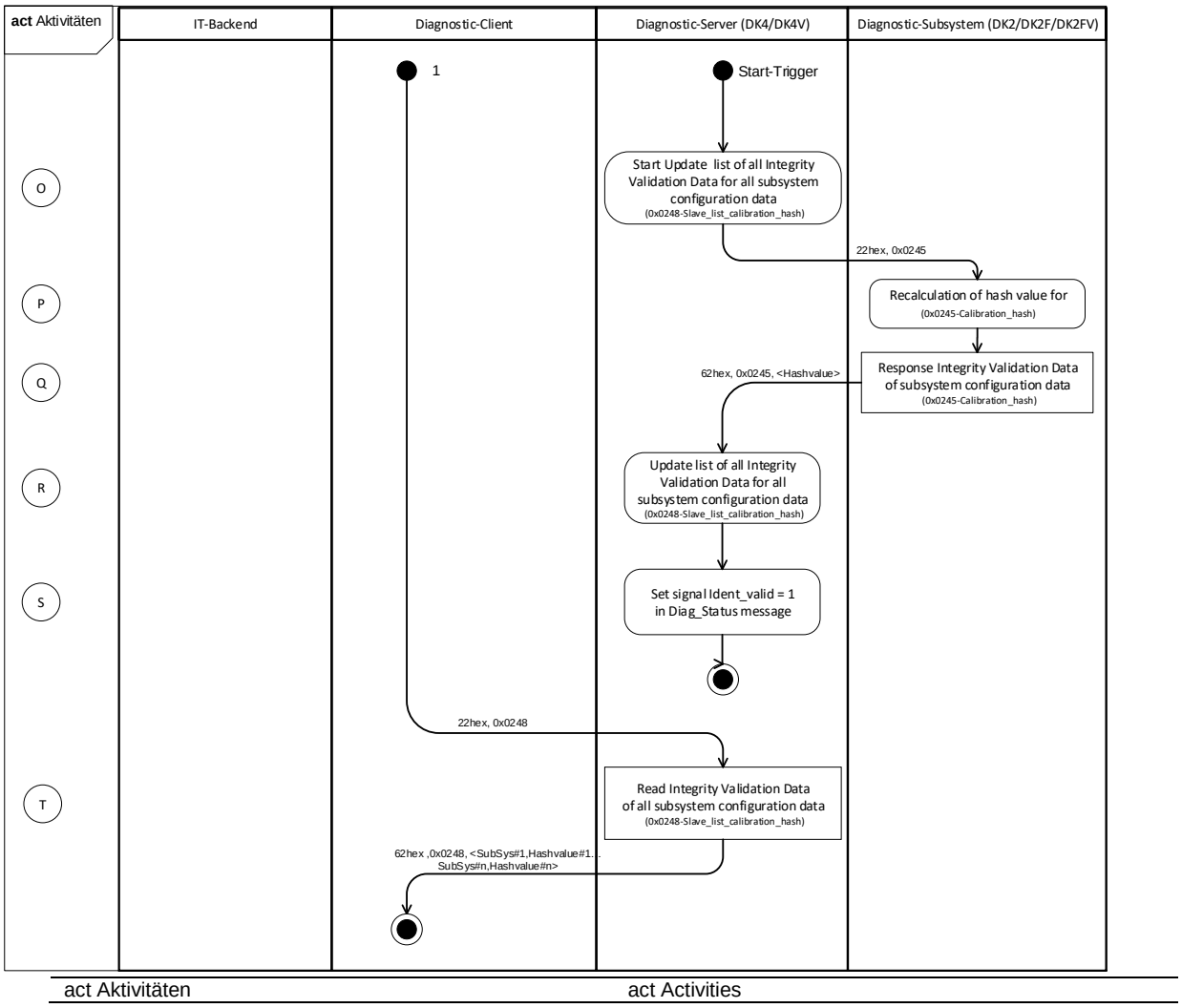
Figure 3-1: Writing adaptations to a diagnostic class 2/2F/2FV system, part 1/2



act Aktivitäten — act Activities

[I: F-LAH_RxSWIN-282]

Figure 3-2: Writing adaptations to a diagnostic class 2/2F/2FV system, part 2/2



[I: F-LAH_RxSWIN-283]

A – The DataIdentifiers required for the ECU configuration are provided to the diagnostic client. The DataIdentifiers result from the production order that configures the ZDC. The content of the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" for a specific subsystem is also defined by this. The diagnostic client requests a validation token for the diagnostic server from the PVD IT backend.

[I: F-LAH_RxSWIN-284]

B – The IT backend generates a validation token for the DataIdentifiers DID#1 to DID#n, and DID "0x0251-Write_generic_to_sub_system."

[I: F-LAH_RxSWIN-285]

C* – The validation token is transferred to the diagnostic server at the start of the PVD routine "0xC011-Begin_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis." It triggers the write sequence for the PVD end-to-end secured data.

[I: F-LAH_RxSWIN-286]

D* – The diagnostic server checks the validity of the PVD validation token it received. In accordance with PVD E2E security as per document /4/, the validation information is deleted if an invalid token is received.

[I: F-LAH_RxSWIN-287]

E – All DataIdentifiers from the configured ZDC and the DataIdentifier "0x0251-Write_generic_to_sub_system" are transferred sequentially to the diagnostic server using the WriteDataByIdentifier (2Ehex) service and cached. In the case of PVD with a group release, the diagnostic server writes the DataIdentifiers directly to the diagnostic sub-systems.

[I: F-LAH_RxSWIN-288]

F* – In accordance with PVD E2E security as per document /4/, the receive order of the individual DataIdentifiers is checked according to the validation information. If the check returns a positive result, the diagnostic server sends a positive response. If the check returns a negative result, the received DataIdentifiers, the corresponding DataRecords, and the validation information are deleted. If the check returns a negative result, the diagnostic server sends a negative response code NRC 0x22 (ConditionsNotCorrect).

[I: F-LAH_RxSWIN-289]

G* – The diagnostic client's write sequence is completed when the "0xC012-End_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis" routine starts with the RoutineControlOption [0x01-Finish_writing_of_secured_data].

[I: F-LAH_RxSWIN-290]

H* – The authenticity of the transferred DataIdentifiers is checked using the validation information from the validation token. The address of the target sub-system and the content of the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" are determined using the received "0x0251-Write_generic_to_sub_system" DataIdentifier.

[I: F-LAH_RxSWIN-295]

I – An [Ident_valid] signal with a value of '0' in the diagnostic server's Diag_Status message indicates that the identification data is currently invalid. After saving the received DataIdentifiers, the existing hash for the sub-system's configuration data is no longer valid; it must be recalculated.

[I: F-LAH_RxSWIN-716]

J – The list entry in the group DataIdentifier "0x0248-Slave_list_configuration_hash" for the diagnostic sub-system is deleted.

[I: F-LAH_RxSWIN-293]

K- The cached DataIdentifiers are sent sequentially from the cache to the diagnostic sub-systems using the WriteDataByIdentifier (2Ehex) service.

[I: F-LAH_RxSWIN-717]

L – With the request to write the configuration data, the "0x0245-Configuration_hash" DataIdentifier in the diagnostic sub-system is deleted.

[I: F-LAH_RxSWIN-291]

M – The diagnostic sub-system checks the data received in the request for validity (e.g., value range). If the check returns a positive result, the data is stored in the sub-system. If the check returns a negative result, the data is discarded and a negative response is sent to the diagnostic server. The diagnostic server sends an error code to the diagnostic client in the positive response of the "0xC012-End_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis" routine.

[I: F-LAH_RxSWIN-296]

N – The cache and the validation information are deleted on the diagnostic server after all received DataIdentifiers have been transferred to the diagnostic sub-system's target memory.

[I: F-LAH_RxSWIN-297]

O – On ending the PVD write sequence, the "Start-Trigger" of the diagnostic server must be updated to include the group DataIdentifier "0x0248-Slave_list_configuration_hash." The server reads out the diagnostic sub-system's "0x0245-Configuration_hash" DataIdentifier to do this. (For "Start-Trigger," see requirements F-LAH_RxSWIN-200 and F-LAH_RxSWIN-693.)

[I: F-LAH_RxSWIN-298]

P – Requesting the "0x0245-Configuration_hash" DataIdentifier serves as the trigger for recalculating the hash for the configuration data. To recalculate the hash, the exact DataIdentifiers from the list transferred as the "0x0250-Integrity_validation_data_configuration_list" are referenced in the exact order.

[I: F-LAH_RxSWIN-299]

Q – Until recalculation of the hash is complete, the diagnostic sub-system responds to the bus master with an NRC 0x78 (Response Pending). Once recalculation is complete, a positive response to the request is sent.

[I: F-LAH_RxSWIN-300]

R – The list entry for the diagnostic sub-system in the group DataIdentifier "0x0248-Slave_list_configuration_hash" on the diagnostic server is updated using the "0x0245-Configuration_hash" DataIdentifier.

[I: F-LAH_RxSWIN-301]

S – An [Ident_valid] signal with a value of '1' in the diagnostic server's Diag_Status message indicates that the identification data is completely valid.

[I: F-LAH_RxSWIN-695]

T – The diagnostic client requests the group DataIdentifier "0x0248-Slave_list_configuration_hash" for all diagnostic sub-systems.

[I: F-LAH_RxSWIN-304]

*) Only relevant to diagnostic systems secured by PVD E2E. Not relevant to diagnostic systems with a group release in production before checkpoint 8.

3.2 ECU programming data security

[allg. Anf.: F-LAH_RxSWIN-860]

The following applies to new platforms:

[allg. Anf.: F-LAH_RxSWIN-912]

- Diagnostic servers must use document /5/ (FDS Q-LAH), version 3.1 or newer, or an alternative, at least equivalent validation method that has been agreed upon with the purchaser's E/E Security department.

[allg. Anf.: F-LAH_RxSWIN-862]

- Diagnostic servers must use document /6/ (Q-LAH 80126), version 2.6 or newer.

[allg. Anf.: F-LAH_RxSWIN-864]

- Diagnostic servers must use document /14/ (Q-LAH 80128-3), version 4.2 or newer.

[allg. Anf.: F-LAH_RxSWIN-913]

The following applies to existing platforms:

[allg. Anf.: F-LAH_RxSWIN-914]

- Diagnostic servers must use document /5/ (FDS Q-LAH), version 2.1 or newer, or an alternative, at least equivalent validation method that has been agreed upon with the purchaser's E/E Security department.

[allg. Anf.: F-LAH_RxSWIN-915]

- Diagnostic servers must use document /6/ (Q-LAH 80126), version 2.3 or newer.

[allg. Anf.: F-LAH_RxSWIN-916]

- Diagnostic servers must use document /14/ (Q-LAH 80128-3), version 4.0 or newer.

4 Integrity validation data

[I: F-LAH_RxSWIN-786]

Integrity validation data is used:

[I: F-LAH_RxSWIN-831]

- As a software and data integrity attribute

[I: F-LAH_RxSWIN-830]

- To check the integrity of ECUs in production

[I: F-LAH_RxSWIN-829]

- To identify the configuration of ECUs by means of target data containers

4.1 General requirements

[I: F-LAH_RxSWIN-225]

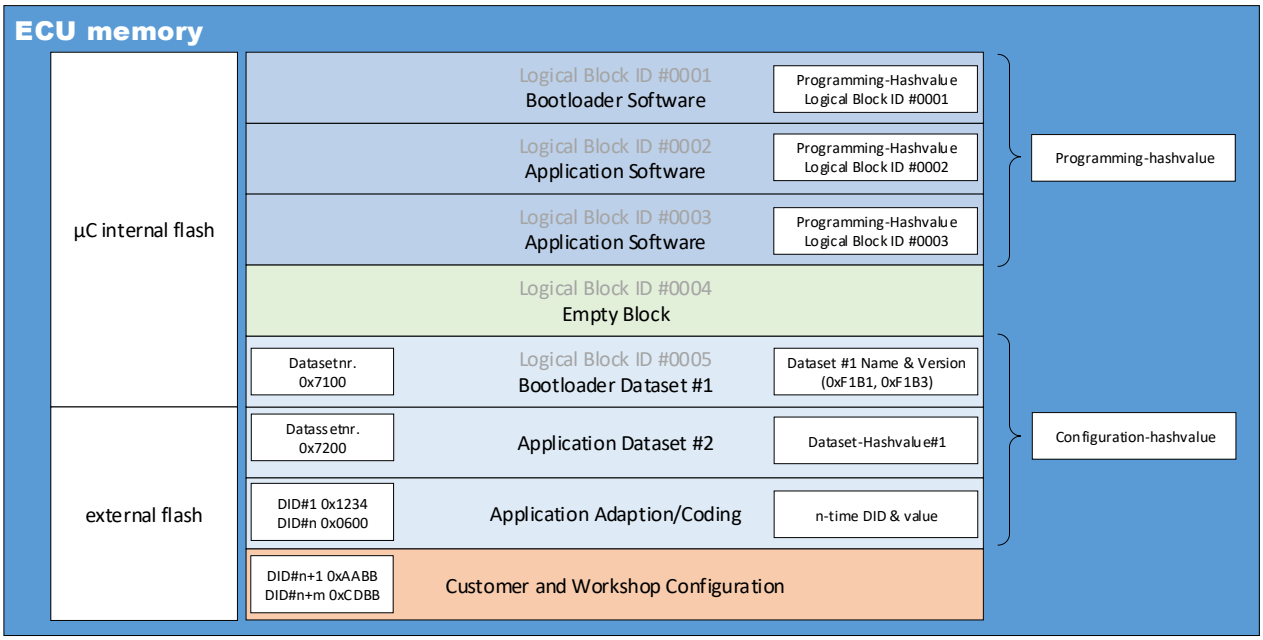
Separate hashes are calculated for instruction code (application and boot loader software) and configuration data (DataIdentifiers and data sets from the ZDC). The hashes are calculated both on IT systems and on the diagnostic server. This means that synchronization between the desired state from the IT backend and the actual state on the diagnostic servers is possible.

[I: F-LAH_RxSWIN-233]

The diagnostic server provides each of the hashes as integrity validation data. DataIdentifiers or RoutineIdentifiers (RIDs) are used for this, depending on the diagnostic class.

[I: F-LAH_RxSWIN-234]

Figure 4-1: Example of an ECU memory architecture



[allg. Anf.: F-LAH_RxSWIN-638]

The SHA-256 algorithm is used to calculate the programming and configuration hashes.

[allg. Anf.: F-LAH_RxSWIN-707]

It must not take more than 100 ms to calculate the hash.
Note: If the ECU cannot respond within 50 ms because it is calculating the hash, it must respond with a negative response code of "0x78-RequestCorrectlyReceived-ResponsePending." Deviations from this duration must be agreed upon with the Diagnostics department, Production, and After-Sales Service; they must be documented in the component-specific diagnostics requirements of the BT-LAH.

[I: F-LAH_RxSWIN-940]

Table 4-1: Mapping of data category (data type as per document /8/) to hash

	Programming hash	Configuration hash
Program data	x	NoImp
Calibration data	x	NoImp
Coding	NoImp	x
Vehicle parameters	NoImp	x
Initial calibration values	NoImp	x
Customer parameters	NoImp	NoImp
Workshop parameters	NoImp	NoImp
Process parameters	NoImp	NoImp
Learned values	NoImp	NoImp
Analysis data	NoImp	NoImp

4.1.1 Programming hash

4.1.1.1 Requirements for embedded systems

[allg. Anf.: F-LAH_RxSWIN-423]

The requirements of this section apply to all programmable diagnostic servers as per document /6/.

[allg. Anf.: F-LAH_RxSWIN-238]

Integrity validation data for programming (instruction code) always includes all logical blocks of the application and boot loader software.

[allg. Anf.: F-LAH_RxSWIN-232]

The ECU calculates a single hash for all logical blocks of the application and boot loader software in its memory as per document /6/.

[allg. Anf.: F-LAH_RxSWIN-740]

If segmented blocks as per document /6/ are used, the integrity validation data is always calculated for the entire logical block, including all segments.

[allg. Anf.: F-LAH_RxSWIN-570]

The metadata (for the software configuration, e.g., for Adaptive AUTOSAR or switch configurations), the default data, and the default data sets are also part of the application and boot loader software.

[allg. Anf.: F-LAH_RxSWIN-784]

In the case of a 2-stage boot loader update, only the logical blocks of the application are used to calculate the programming hash from the second flash PDX.

[allg. Anf.: F-LAH_RxSWIN-389]

Empty logical blocks used as a reserve must not be included in the hash calculation.

[allg. Anf.: F-LAH_RxSWIN-745]

Flash drivers, as logical blocks in RAM, must not be included in the hash calculation.

[allg. Anf.: F-LAH_RxSWIN-741]

Deviations from requirements *[A: 80126-A915]* and *[A: 80126-A914]* in document /6/ must not exclude optional populated blocks (e.g., for additional text that does not directly belong to the application, and that does not influence the functionality) from the compatibility check by the "0xFF01-Check Programming Dependencies" routine. These blocks must be included in the hash calculation.

[allg. Anf.: F-LAH_RxSWIN-768]

Deviating from requirement *[A: 80126-A146]* in document /6/, the cyclical redundancy check CRC32 value for the uncompressed and unencrypted payload in the check request must be calibrated.

[allg. Anf.: F-LAH_RxSWIN-769]

Note: If a logical block includes data (e.g., supplier-specific information) that the supplier does not include when calculating the CRC32, then this data must not be included when calculating the CRC32 on the diagnostic server either.

[allg. Anf.: F-LAH_RxSWIN-390]

When the "0x0253-Calculate_integrity_validation_data" routine calculates the hash with the "Type_of_calculation" = 0x01 ControlOption, then it must only include logical blocks labeled with a valid software version number in the "0xF1AB-Logical Software Block Version" DataIdentifier.

[allg. Anf.: F-LAH_RxSWIN-692]

Static software components that cannot be updated by Volkswagen AG are not included when calculating the programming hash.

[allg. Anf.: F-LAH_RxSWIN-567]

The "0x0253-Calculate_integrity_validation_data" routine recalculates the programming hash on the ECU with the "Type_of_calculation" = 0x01 ControlOption.

[allg. Anf.: F-LAH_RxSWIN-564]

On starting the "0xFF00-Erase Memory" routine that addresses a logical block with instruction code, the CRC32 checksum of the addressed logical block must be deleted.

[allg. Anf.: F-LAH_RxSWIN-565]

Following a positive check by the "0x0202-Check Memory" routine, confirming that the logical block was transferred without error, the CRC32 checksum of the logical block must be recalculated.

[allg. Anf.: F-LAH_RxSWIN-718]

Following a positive response for the "0x0202-Check Memory" routine that addresses a logical block with instruction code, the calculated CRC32 checksum of the logical block is persistently stored.

[allg. Anf.: F-LAH_RxSWIN-636]

The "0x0544-Verify_partial_software_checksum" routine must be implemented as per document /6/.

[allg. Anf.: F-LAH_RxSWIN-620]

Deviating from document /6/ and document /2/, the "0x0544-Verify_partial_software_checksum" routine must be available in the application in the "DefaultSession (0x01)."

[allg. Anf.: F-LAH_RxSWIN-617]

On receiving the request for the "0x0544-Verify_partial_software_checksum" routine, the CRC32 checksum must be recalculated for the logical block addressed in the request and persistently stored on the ECU.

[allg. Anf.: F-LAH_RxSWIN-345]

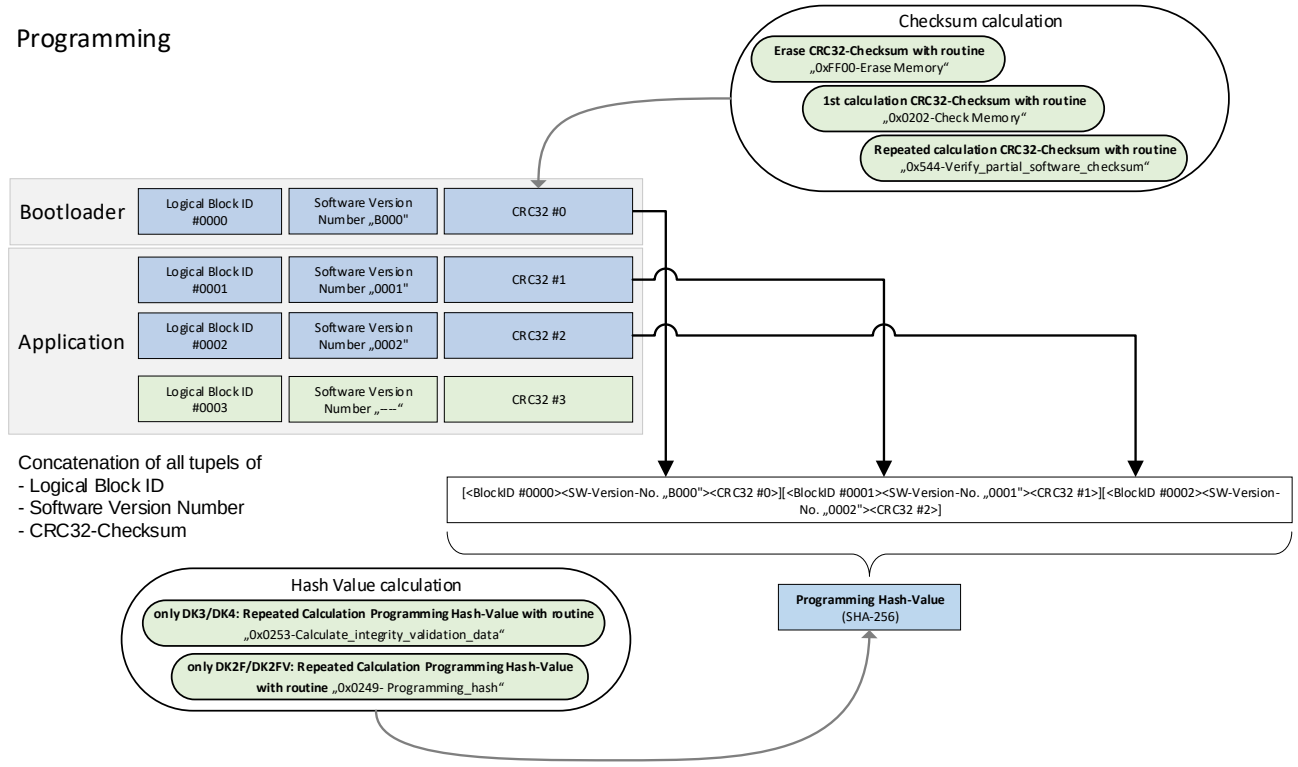
Concatenated tuples, consisting of the block ID, software version number, and CRC32 checksum of each logical block, are used to calculate the programming hash.

[allg. Anf.: F-LAH_RxSWIN-715]

The tuples for calculating the programming hash are concatenated in ascending order of the logical block ID.

[I: F-LAH_RxSWIN-235]

Figure 4-2: Concatenation of the data of the logical blocks in the application and boot loader (instruction code) for calculating the hash



[I: F-LAH_RxSWIN-388]

Note:

- Logical block ID (2 bytes) as per document /6/
- Programming hash #n = Hash for the logical block of the application and/or boot loader software

[Prozess-Anf.: F-LAH_RxSWIN-770]

The desired programming hash is calculated on the IT systems as per the calculation on the diagnostic server. The CRC32 values must be taken from the flash PDX.

[allg. Anf.: F-LAH_RxSWIN-659]

The standardized identification data as per document /2/ and document /12/ must be included under the following conditions (AND logical operator) when calculating the programming hash, if it is:

[allg. Anf.: F-LAH_RxSWIN-817]

- Modifiable neither via the WriteDataByIdentifier (2Ehex) service, nor via the data set download (boot loader/application).

[allg. Anf.: F-LAH_RxSWIN-816]

- Only modifiable via update programming.

[allg. Anf.: F-LAH_RxSWIN-815]

- Required as per the diagnostic class.

[allg. Anf.: F-LAH_RxSWIN-814]

- Required as per the on-board diagnostics (OBD) class.

[allg. Anf.: F-LAH_RxSWIN-813]

- Required for specific use cases as per the implementation requirements.

[allg. Anf.: F-LAH_RxSWIN-812]

- Required as per the system design from document /12/.

[I: F-LAH_RxSWIN-660]

By way of an example, the following DataIdentifiers as per document /2/ and document /12/ have been mapped to the programming hash as a function of the OBD relevance, the diagnostic class, the implementation requirements, and the system design.

Only modifiable via update programming. (diagnostic classes 2F and higher):

- 0xF187-Volkswagen Spare Part Number
- 0xF189-Volkswagen Application Software Version Number
- 0xF19E-ASAM ODX File Identifier
- 0xF1A2-ASAM ODX File Version

Only for systems that contain software compositions:

- 0x0441-SWCO_list_nr
- 0x011D-SWCO_list_system_name

Only for OBD-relevant systems:

- 0x02CE-OBD_type
- 0x02CF-OBD_class_description

[allg. Anf.: F-LAH_RxSWIN-688]

The following data categories (data types) as per document /8/ are included when calculating the program data hash:

[allg. Anf.: F-LAH_RxSWIN-938]

- Program data (program code or software for the ECU)
  Note: Program data is part of the flash container.

[allg. Anf.: F-LAH_RxSWIN-939]

- Calibration data (equipment-dependent data and characteristic curves)
  Note: Calibration data is part of the flash container.

4.1.1.2 Requirements for non-embedded systems or file-based systems

4.1.1.2.1 Use of flash containers "ODX flash/PDX flash"

[allg. Anf.: F-LAH_RxSWIN-641]

If flash containers as per document /14/ are used, the method for calculating the programming hash as per section "Requirements for embedded systems" must be applied.

4.1.1.2.2 Use of flash containers deviating from "ODX flash/PDX flash"

[allg. Anf.: F-LAH_RxSWIN-643]

If flash containers deviating from document /14/ are used, the calculation can be system-specific.

[allg. Anf.: F-LAH_RxSWIN-644]

The desired value of the programming hash must be included in the description files (manifest or metadata) such that the IT systems can evaluate it.

[allg. Anf.: F-LAH_RxSWIN-645]

The implementations of the hash calculation must be agreed upon with the purchaser (Diagnostics department and Group IT).

[Prozess-Anf.: F-LAH_RxSWIN-773]

If flash containers in lum-f format are used, a backend system must use the main manifest checksum file to calculate the programming hash. For this, the backend system must calculate the programming hash using the "main.mnf.cks" file.

[allg. Anf.: F-LAH_RxSWIN-774]

The diagnostic server must transfer the whole manifest file tree, including the root node ("main.mnf") and its metadata (main.mnf.cks, main.mnf.cks.sig), to the system to be flashed. It must store these together with the program files.

[allg. Anf.: F-LAH_RxSWIN-775]

Whenever the diagnostic server accesses manifest files, it must ensure the integrity of this data. For this, the system to be programmed must check the signature of the root node (the desired value is stored in the main.mnf.cks.sig file).

[allg. Anf.: F-LAH_RxSWIN-776]

For each package (module) referenced in the manifest data, the diagnostic server must regularly check the CRC32 checksum at runtime.

[allg. Anf.: F-LAH_RxSWIN-777]

A check at runtime is not necessary if the diagnostic server executes a secure boot procedure (e.g., dm_verity) to check the package (module) in question.

[allg. Anf.: F-LAH_RxSWIN-779]

The "0x0253-Calculate_integrity_validation_data" routine recalculates the programming hash on the diagnostic server with the "Type_of_calculation" = 0x01 RoutineControlOption. The diagnostic server must distinguish between the following cases when doing so:

[allg. Anf.: F-LAH_RxSWIN-780]

- A complete check result for the values at runtime does not yet exist. In this case, the diagnostic server must return the hash of the main manifest checksum file ("main.mnf.cks").

  [allg. Anf.: F-LAH_RxSWIN-781]

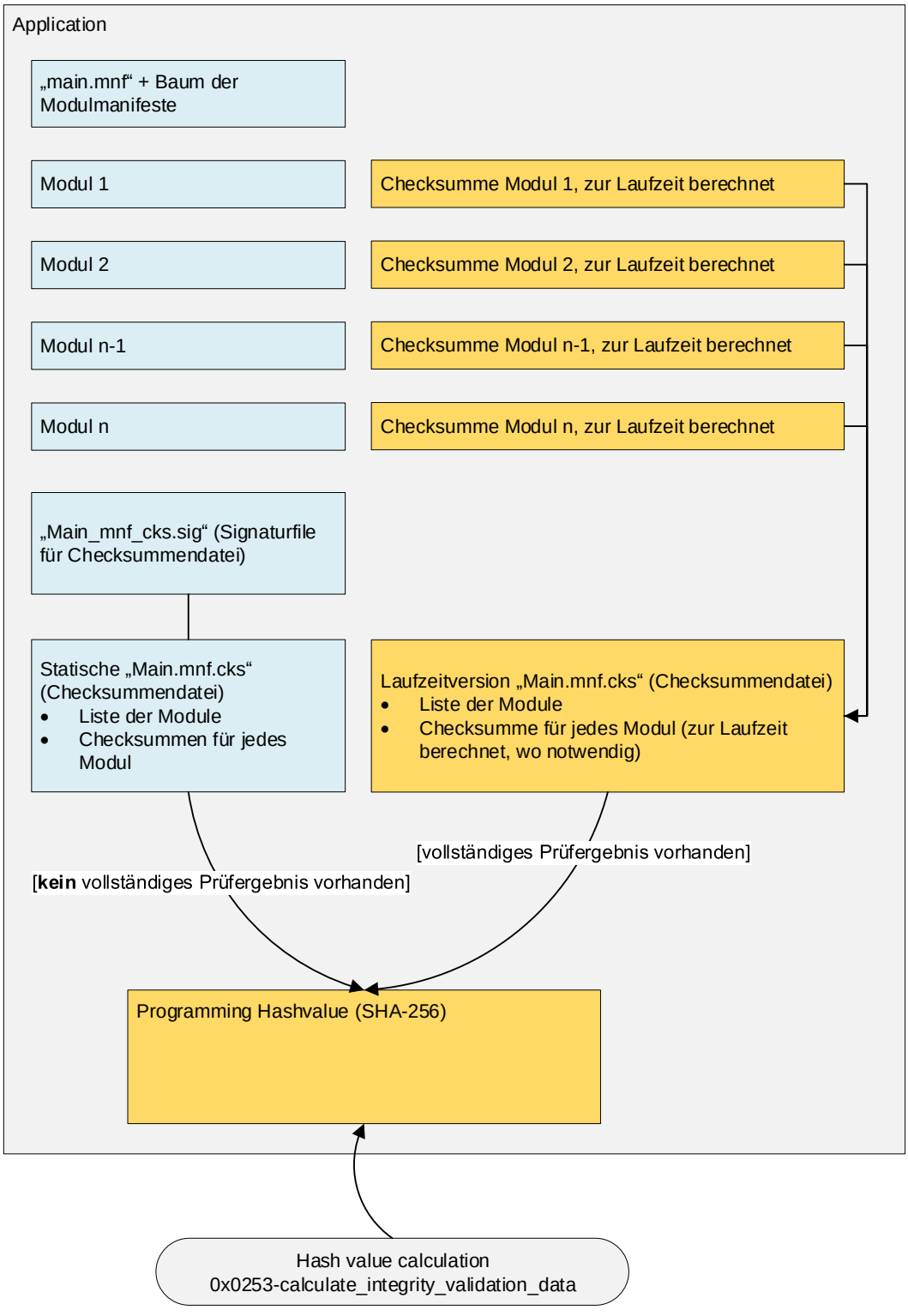
- There is a check result at runtime and the diagnostic server has completed the check of all packages (modules) to be checked with a positive result: The diagnostic server reports the result as per a).

[allg. Anf.: F-LAH_RxSWIN-782]

- There is a check result at runtime and the diagnostic server has completed the check of one or multiple packages (modules) to be checked with a negative result: The diagnostic server returns a programming hash that does not match the desired value specified on the IT backend.

[I: F-LAH_RxSWIN-924]

Figure 4-3: Calculating a hash for a lum-f container



„main.mnf“ + Baum der Modulmanifeste	"main.mnf" + tree of module manifests
Modul 1	Module 1
Checksumme Modul 1, zur Laufzeit berechnet	Checksum of module 1, calculated at runtime
Modul 2	Module 2
Checksumme Modul 2, zur Laufzeit berechnet	Checksum of module 2, calculated at runtime
Modul n-1	Module n - 1
Checksumme Modul n-1, zur Laufzeit berechnet	Checksum of module n - 1, calculated at runtime
Modul n	Module n
Checksumme Modul n, zur Laufzeit berechnet	Checksum of module n, calculated at runtime
„Main_mnf_cks.sig“ (Signaturfile für Checksummendatei)	"Main_mnf_cks.sig" (signature file for the checksum file)
Statische „Main.mnf.cks“ (Checksummendatei) • Liste der Module • Checksummen für jedes Modul	Static "Main.mnf.cks" (checksum file) • List of modules • Checksums for each module
Laufzeitversion „Main.mnf.cks“ (Checksummendatei) • Liste der Module • Checksumme für jedes Modul (zur Laufzeit berechnet, wo notwendig)	Runtime version "Main.mnf.cks" (checksum file) • List of modules • Checksum for each module (calculated at runtime where needed)
[vollständiges Prüfergebnis vorhanden]	[Complete check result exists]
[kein vollständiges Prüfergebnis vorhanden]	[Complete check result does not exist]

4.1.2 Configuration hash

[allg. Anf.: F-LAH_RxSWIN-424]

The requirements of this section apply to all configurable diagnostic servers.

[I: F-LAH_RxSWIN-646]

No distinction is made between embedded and file-based systems.

[allg. Anf.: F-LAH_RxSWIN-239]

The hash for the configuration data includes logical blocks with boot loader data sets and application configuration data.
Note: Diagnostic class 2F/2FV systems do not have application or boot loader data sets.

[allg. Anf.: F-LAH_RxSWIN-241]

Only the following data categories (data types) as per document /8/ may be included when calculating the configuration data hash:

[allg. Anf.: F-LAH_RxSWIN-791]

- Coding (equipment-dependent switching of sub-functions ON/OFF)

  [allg. Anf.: F-LAH_RxSWIN-790]

- Vehicle parameters (vehicle-specific parameters and settings)

  [allg. Anf.: F-LAH_RxSWIN-789]

- Initial calibration values (one-time (initially) writable default values, parameters, and settings)

  [allg. Anf.: F-LAH_RxSWIN-308]

The following data types as per document /8/ must <u>not</u> be included when calculating the application configuration data hash:

[allg. Anf.: F-LAH_RxSWIN-937]

- Program data (program code or software for the ECU)
  Note: Program data is part of the programming hash.

  [allg. Anf.: F-LAH_RxSWIN-797]

- Calibration data (equipment-dependent data and characteristic curves)
  Note: Program data is part of the programming hash.

  [allg. Anf.: F-LAH_RxSWIN-796]

- Customer parameters (customer-specific parameters and settings)

  [allg. Anf.: F-LAH_RxSWIN-795]

- Workshop parameters (workshop-specific parameters and settings)
  Note: Workshop parameters may only be modified within a specific framework without influencing type approval.

  [allg. Anf.: F-LAH_RxSWIN-794]

- Process parameters (process-specific parameters and settings)
  Note: Process parameter that influence type approval must be documented in the build status documentation.

  [allg. Anf.: F-LAH_RxSWIN-793]

- Learned values (parameters and settings learned independently by the ECU)

  [allg. Anf.: F-LAH_RxSWIN-792]

- Analysis data (data generated dynamically during normal vehicle operation)

[allg. Anf.: F-LAH_RxSWIN-785]

The following DataIdentifier must not be included when calculating the application configuration data hash:

- 0xF18F-Regulation_x_software_identification_numbers

[Prozess-Anf.: F-LAH_RxSWIN-787]

Note: DataIdentifier 0xF18F is not part of the gateway ECU's ZDC. It thus cannot be included when calculating the desired value for the configuration hash. It is provided via a separate data container and entered in full in the build status documentation.

[allg. Anf.: F-LAH_RxSWIN-544]

A change of customer and workshop parameters must NOT lead to a change of the integrity validation data of the configuration data.

[allg. Anf.: F-LAH_RxSWIN-568]

"Specified data sets" as per document /7/ are included when calculating the configuration data hash and must be referenced in the ZDC.

[allg. Anf.: F-LAH_RxSWIN-569]

"Reserved data sets" as per document /7/ are included when calculating the configuration data hash and must be referenced in the ZDC.
Note: Reserved data sets are not required to have a valid software version number in the "0xF1AB-Logical Software Block Version" DataIdentifier.

[allg. Anf.: F-LAH_RxSWIN-571]

"Default data sets" as per document /7/ are part of the application software and must not be included when calculating the configuration data hash.

[allg. Anf.: F-LAH_RxSWIN-772]

Reserved codings/adaptations are included when calculating the configuration data hash and must be included in the ZDC.

[allg. Anf.: F-LAH_RxSWIN-691]

If multiple instances of, e.g., ASIL-relevant data are stored, the portion of data stored redundantly is ignored when calculating the configuration hash.

[allg. Anf.: F-LAH_RxSWIN-647]

The standardized identification data as per document /2/ and document /12/ must be included under the following conditions (AND logical operator) when calculating the configuration hash, if it is:

[allg. Anf.: F-LAH_RxSWIN-822]

- Modifiable via the WriteDataByIdentifier (2Ehex) service or via the data set download (boot loader/application).

[allg. Anf.: F-LAH_RxSWIN-821]

- Required as per the diagnostic class.

[allg. Anf.: F-LAH_RxSWIN-820]

- Required as per the OBD class.

[allg. Anf.: F-LAH_RxSWIN-819]

- Required for specific use cases as per the implementation requirements.

[allg. Anf.: F-LAH_RxSWIN-818]

- Required as per the system design from document /12/.

[I: F-LAH_RxSWIN-651]

By way of an example, the following DataIdentifiers as per document /2/ and document /12/ have been mapped to the configuration hash as a function of the diagnostic class, the implementation requirements, and the system design.

Only for bus master systems in diagnostic class 4-high:
- 0x04A3-Gateway Component List
- 0x061A-Slave_component_list

Only for systems that support the 2nd generation data set download:
- 0xF1B1-VW_Application_data_set_identification
- 0xF1B3-VW_Data_set_name

Only for systems that contain software compositions:
- 0x0442-SWCO_list

[allg. Anf.: F-LAH_RxSWIN-572]

All the configuration data included when calculating the hash must be included in the target data container. The target data container must not contain other configuration data.

[I: F-LAH_RxSWIN-599]

Note: The ZDC must contain all the configuration data to allow after-sales service in the field to restore all configuration data in accordance with the type approval if the hash on the ECU does not match the desired hash on the IT backend.

[allg. Anf.: F-LAH_RxSWIN-623]

Immobilizer data, component protection data, Vehicle Key Management System (VKMS) data, and Software as a Product (SWAP) or features on demand data must not be included when calculating the configuration data hash.

[I: F-LAH_RxSWIN-624]

Note: The immobilizer data as per document /18/, the component protection data as per document /19/, VKMS data as per document /15/, and SWAP data as per document /16/ or features on demand data as per document /17/ are securely transferred to the vehicle and stored in protected memory areas (Secure Hardware Extension (SHE) or hardware security module (HSM)) as per document /20/ to be able to rule out manipulation of the data.

[allg. Anf.: F-LAH_RxSWIN-346]

The DataIdentifier "0x0250-Integrity_validation_data_configuration_list" is used to specify the list of DataIdentifiers for adaptation/coding and the data set numbers that must be used to calculate the hash for the configuration data.

[allg. Anf.: F-LAH_RxSWIN-377]

The order of the data for calculating the hash results from the content of DataIdentifier "0x0250-Integrity_validation_data_configuration_list." The order must be adopted without change.

4.1.2.1 Calculating the individual hashes or checksums of the configuration data

[I: F-LAH_RxSWIN-690]

The hash of the configuration data is determined with the aid of various individual hashes. These individual hashes can be read out using the RoutineIdentifier "0x0254-Calculate_individual_hash_value."

4.1.2.1.1 Individual hash for adaptations/codings and application data sets as per document /7/

[allg. Anf.: F-LAH_RxSWIN-576]

The individual hashes for all adaptations/codings and application data sets for the configuration data are calculated for the first time with the positive response for the PVD routine "0xC012-End_of_writing_secured_data_of_protection_of_vehicle_diagnosis."

[allg. Anf.: F-LAH_RxSWIN-577]

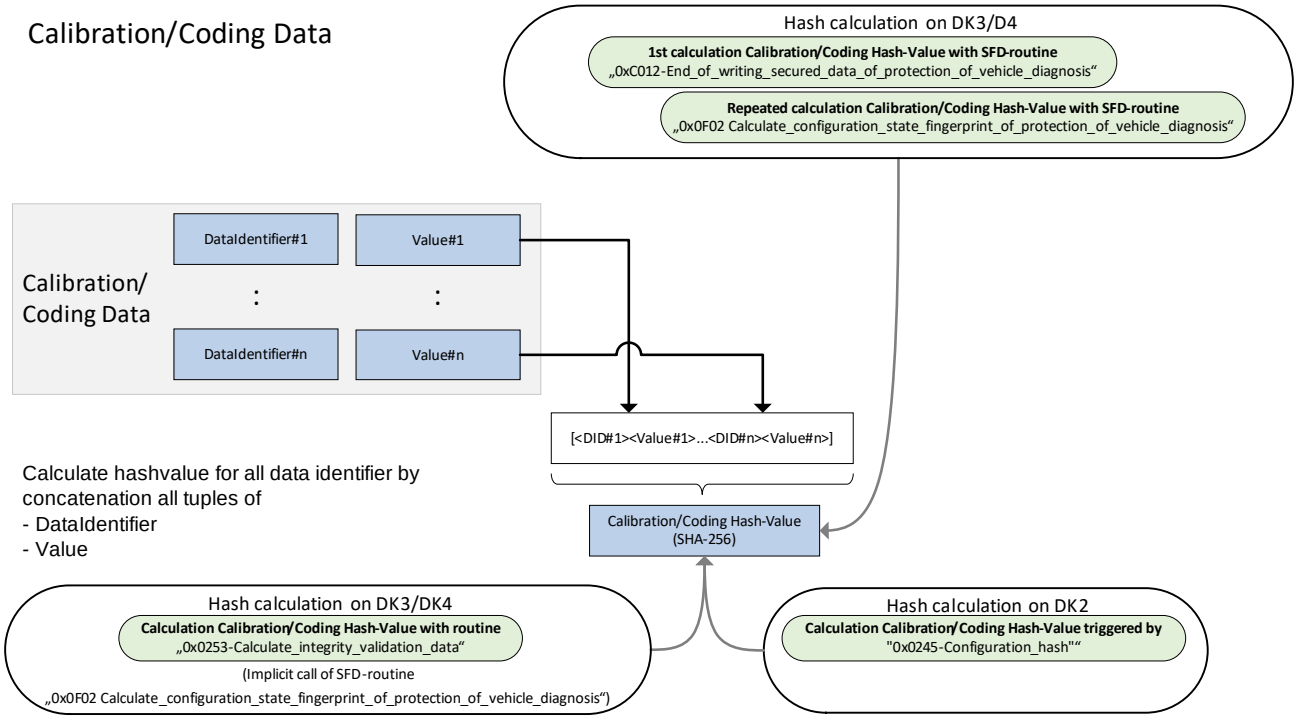
The individual hashes for all adaptations/codings and application data sets for the configuration data are recalculated on receiving the request for the PVD routine "0x0F02-Calculate_configuration_state_fingerprint_of_protection_of_vehicle_diagnosis."

[allg. Anf.: F-LAH_RxSWIN-593]

ECUs that have not implemented PVD E2E calculate the individual hashes for adaptations/codings and application data sets as per document /7/ on receiving the request for the "0x0253-Calculate_integrity_validation_data" routine with the "Type_of_calculation" = 0x00 ControlOption.

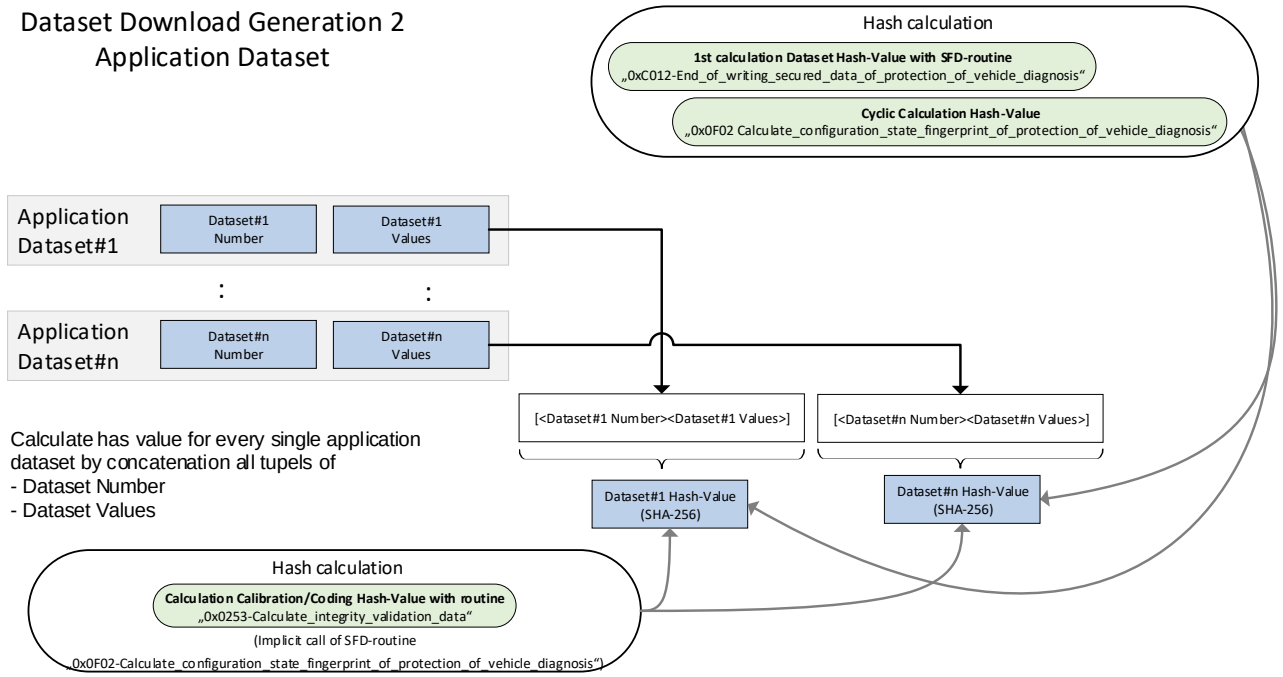
[I: F-LAH_RxSWIN-578]

Figure 4-4: Calculating an individual hash for all adaptations and codings



[I: F-LAH_RxSWIN-585]

Figure 4-5: Calculating an individual hash for application data sets



4.1.2.1.2 CRC checksum for data sets belonging to the 1st generation data set download as per document /11/

[allg. Anf.: F-LAH_RxSWIN-742]

The CRC16 or CRC32 checksum contained in the data set must not be changed.

[allg. Anf.: F-LAH_RxSWIN-743]

Additionally, a CRC32 checksum for the entire content of the data set container, including the data, checksum, and memory address, is calculated and stored on the ECU for each valid data set.

[allg. Anf.: F-LAH_RxSWIN-580]

The CRC32 checksum for a data set is deleted on receiving the request for the "0x0300-Erase VW Memory" routine.

[allg. Anf.: F-LAH_RxSWIN-581]

The CRC32 checksum for a data set is calculated for the first time on receiving the request for the "0x02FF-Calculate checksum" routine.

[allg. Anf.: F-LAH_RxSWIN-584]

The CRC32 checksum for a data set is persistently stored on receiving the positive response for the "0x02FF-Calculate checksum" routine.

[allg. Anf.: F-LAH_RxSWIN-582]

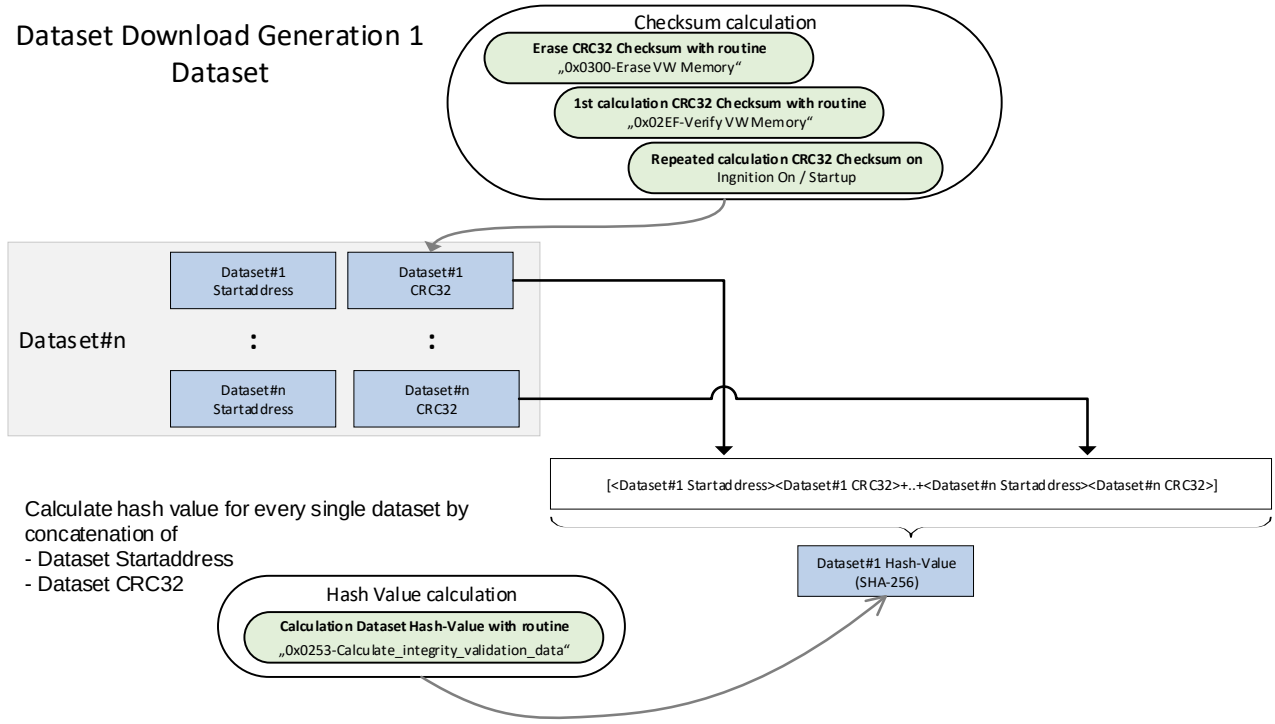
The CRC32 checksum for a data set is recalculated and persistently stored on each change to terminal 15 on or start-up.

[allg. Anf.: F-LAH_RxSWIN-661]

The individual hash for a data set is calculated on receiving the request for the "0x0253-Calculate_integrity_validation_data" routine with the "Type_of_calculation" = 0x00 ControlOption.

[I: F-LAH_RxSWIN-583]

Figure 4-6: Calculating the CRC32 checksum for data sets belonging to the 1st generation data set download



4.1.2.1.3 Calculating the CRC32 checksum for boot loader data sets belonging to the 2nd generation data set download as per document /7/

[allg. Anf.: F-LAH_RxSWIN-587]

The CRC32 checksum for a boot loader data set is deleted on receiving the request for the "0xFF00-Erase Memory" routine that addresses the boot loader data set.

[allg. Anf.: F-LAH_RxSWIN-714]

The "0xFF00-Erase Memory" routine that addresses a logical block with the boot loader data set must not influence the programming hash.

[allg. Anf.: F-LAH_RxSWIN-588]

The CRC32 checksum for a boot loader data set is calculated for the first time on receiving the request for the "0x0202-Check Memory" routine.

[allg. Anf.: F-LAH_RxSWIN-589]

The CRC32 checksum for a boot loader data set is persistently stored on receiving the positive response for the "0x0202-Check Memory" routine.

[allg. Anf.: F-LAH_RxSWIN-590]

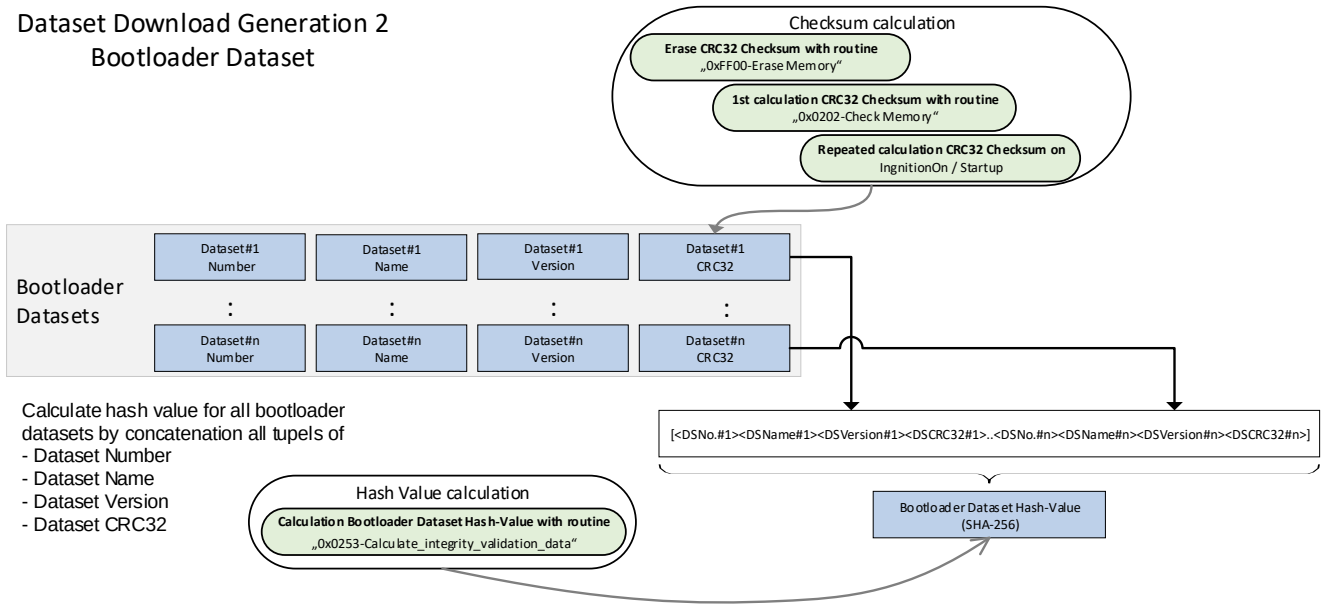
The CRC32 checksum for a boot loader data set is recalculated and persistently stored on each terminal 15 OFF/ON or start-up.

[allg. Anf.: F-LAH_RxSWIN-662]

The individual hash for the boot loader data sets is calculated on receiving the request for the "0x0253-Calculate_integrity_validation_data" routine with the "Type_of_calculation" = 0x00 ControlOption.

[I: F-LAH_RxSWIN-591]

Figure 4-7: Calculating the CRC32 checksum for 2nd generation boot loader data sets



4.1.2.1.4 Calculating the overall hash of the configuration data

[allg. Anf.: F-LAH_RxSWIN-347]

On diagnostic class 2/2F/2FV/3/3V/4/4V/SWCL systems, the list from the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" is used to calculate the hash of the configuration data.

[allg. Anf.: F-LAH_RxSWIN-622]

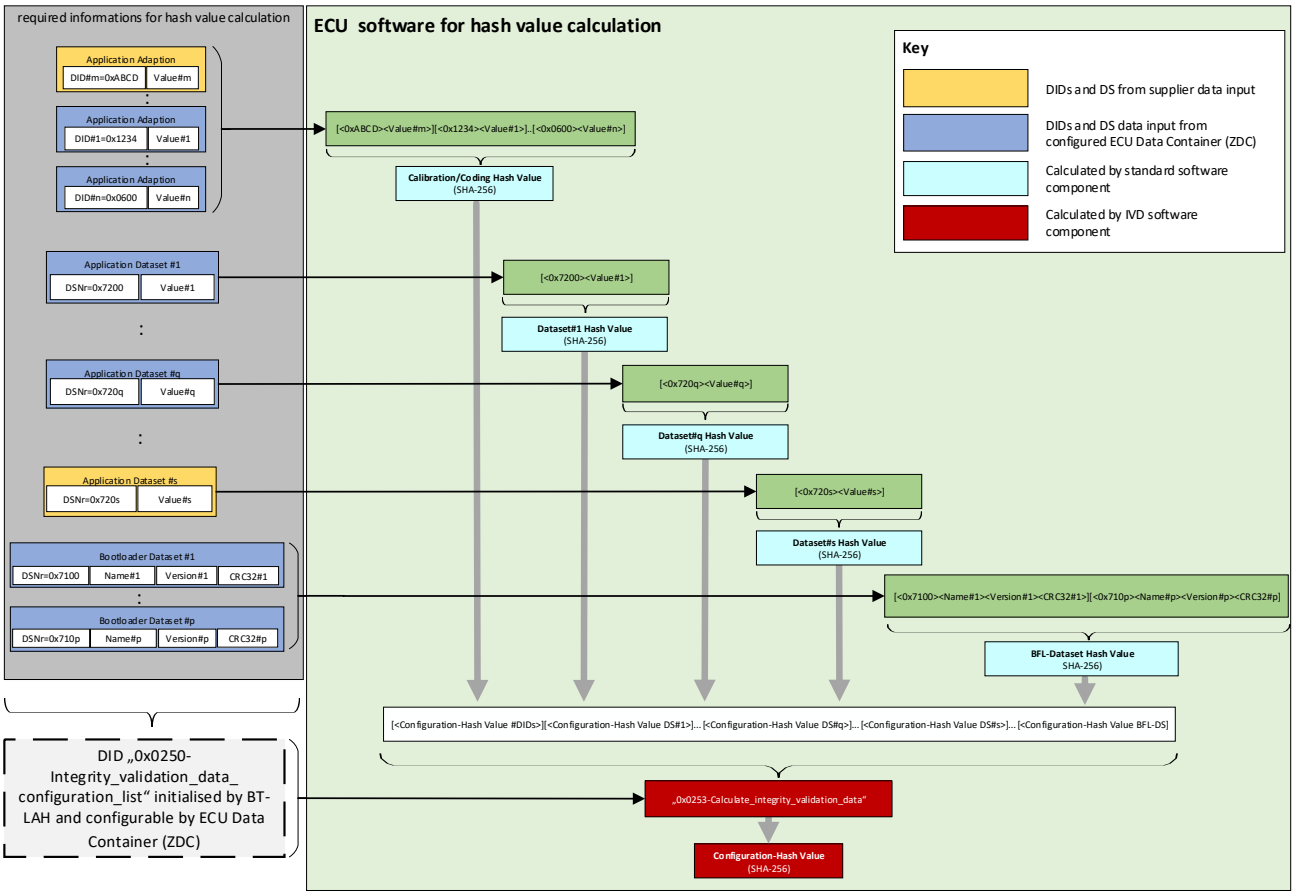
The "0x0253-Calculate_integrity_validation_data" routine with the "Type_of_calculation" = 0x00 ControlOption triggers the calculation of the configuration data hash for all DataIdentifiers and data set numbers contained in the list in the order specified by the list.

[allg. Anf.: F-LAH_RxSWIN-344]

The overall hash of all configuration data to be output is a hash calculated from the hash of all concatenated individual hashes.

[I: F-LAH_RxSWIN-236]

Figure 4-8: Calculating the overall hash of the configuration data (for 2nd gen. DSDL)

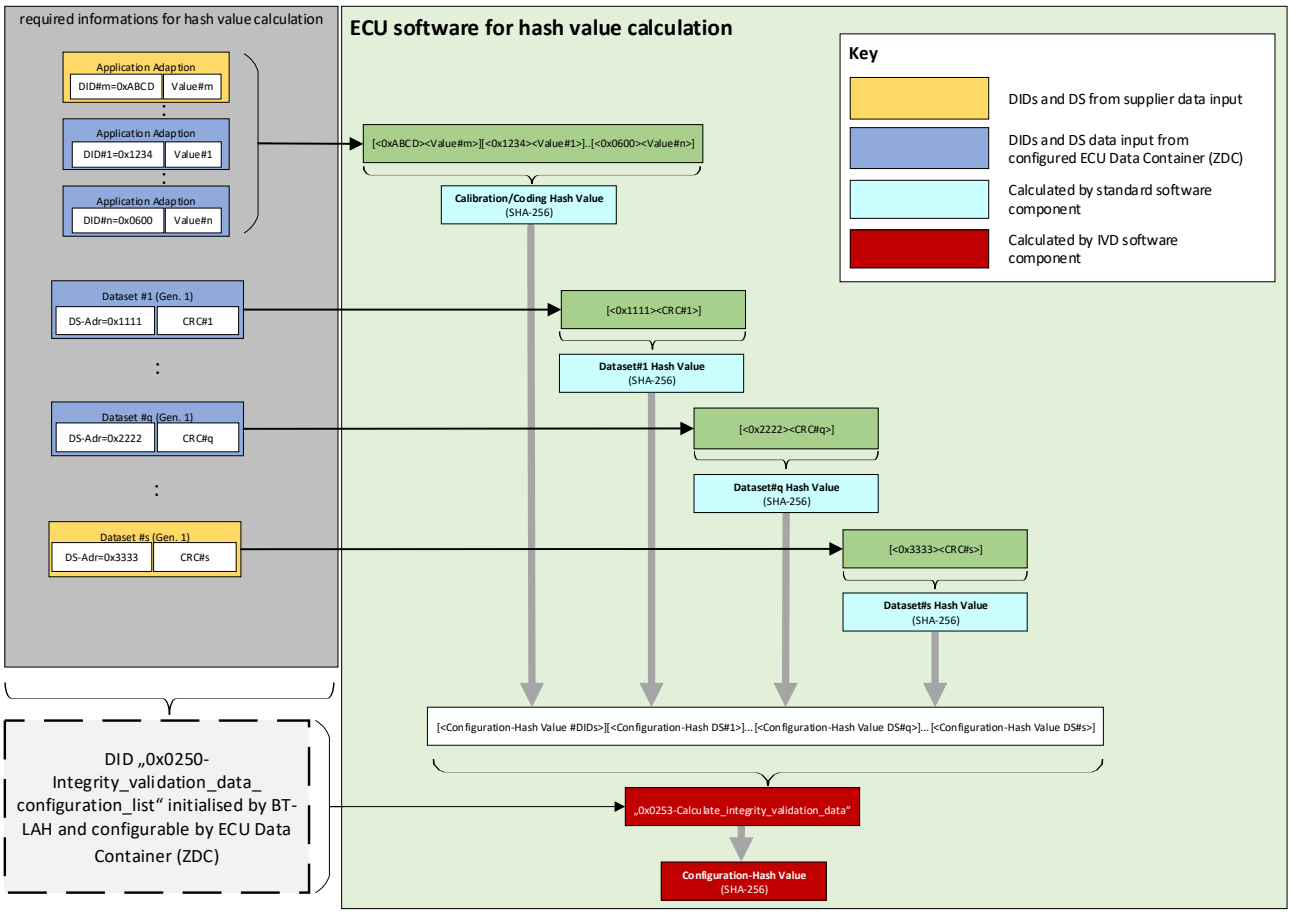


[I: F-LAH_RxSWIN-392]

DSNr = Data set number (2-byte hex) as per document /7/, 0x7100 – 0x71FF for boot loader data sets and 0x7200 – 0x72FF for application data sets
DID = DataIdentifier as per document /1/
Value = Data content of the adaptations or codings
Hash = Hash of the application data set (SHA-256 with a length of 32 bytes)
Version = Boot loader data set version as per identification "0xF1B1-VW_Application_data_set_identification" with the corresponding data set number as per document /2/
Name = Boot loader data set name as per identification " 0xF1B3-VW_Data_set_name" with the corresponding data set number as per document /2/

[I: F-LAH_RxSWIN-666]

Figure 4-9: Calculating the overall hash of the configuration data (for 1st gen. DSDL)



[I: F-LAH_RxSWIN-667]

DS-Adr = Data set address (2-byte hex) as per document /11/
DID = DataIdentifier as per document /1/
Value = Data content of the adaptations or codings
Hash = Hash of the application data set (SHA-256 with a length of 32 bytes)

4.1.2.2 Requirements for processes

[Prozess-Anf.: F-LAH_RxSWIN-663]

All configuration data relevant to calculating the configuration hash must be marked in the BT-LAH's diagnostic data tables and included in the ZDC.

[Prozess-Anf.: F-LAH_RxSWIN-538]

The BT-LAH's diagnostic data tables are used for the initial calibration of the DataIdentifier "0x0250-Integrity_validation_data_configuration_list."

[Prozess-Anf.: F-LAH_RxSWIN-560]

The initial calibration of the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" contains all DataIdentifiers for adaptations and coding marked as ZDC-relevant as well as the data set numbers of the application and boot loader data sets.

[Prozess-Anf.: F-LAH_RxSWIN-375]

The content of the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" can be changed as necessary if the initial calibration deviates from the calibration determined by a ZDC configured by a production order.

[Prozess-Anf.: F-LAH_RxSWIN-771]

The desired configuration hash is calculated on IT systems as per the calculation on the diagnostic server.

[Prozess-Anf.: F-LAH_RxSWIN-479]

The order of the values for the DataIdentifiers and the data set numbers in the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" result from the ZDC configured by the production order.

4.2 Requirements for diagnostic class 4/4V systems based on Q-LAH 80127 as of version 5.1

[allg. Anf.: F-LAH_RxSWIN-200]

For each diagnostic class 2/2F/2FV system, the diagnostic class 4-low/4V-low system must acquire the identification data for group DIDs as per document /3/ as a function of the trigger condition.

[allg. Anf.: F-LAH_RxSWIN-693]

In addition to the trigger conditions as per document /3/, the following trigger condition must also be implemented:

- Completion of a write operation secured by PVD E2E from the diagnostic class 4-low system to the diagnostic class 2F system

4.2.1 Group DataIdentifier for integrity validation data of the programming for diagnostic class 2F/2FV systems

[I: F-LAH_RxSWIN-187]

The DataIdentifier "0x0247-Slave_list_programming_hash" (hashes of the programming for sub-bus systems) contains the integrity validation data for the instruction code of all lower-level diagnostic class 2F/2FV systems.

[allg. Anf.: F-LAH_RxSWIN-196]

The diagnostic class 4-low/4V-low systems must use the ReadDataByIdentifier (22hex) service to output the integrity validation data for the programming of the diagnostic class 2F/2FV systems via the "0x0247-Slave_list_programming_hash" group DataIdentifier.

[allg. Anf.: F-LAH_RxSWIN-195]

A higher-level diagnostic class 4-low/4V-low system uses the ReadDataByIdentifier (22hex) service to collect the hashes for all of its lower-level diagnostic class 2F/2FV systems via the DataIdentifier "0x0249-Programming_hash" and calculates the overall hash of the hashes acquired from the lower-level systems.

[allg. Anf.: F-LAH_RxSWIN-197]

The "0x0247-Slave_list_programming_hash" group DataIdentifier must not be writable using the WriteDataByIdentifier (2Ehex) service.

[allg. Anf.: F-LAH_RxSWIN-199]

All diagnostic class 4-low/4V-low systems with lower-level diagnostic class 2F/2FV systems must implement the DataIdentifier "0x0247-Slave_list_programming_hash" as a mandatory requirement.

4.2.2 Group DataIdentifier for integrity validation data of the configuration for diagnostic class 2/2F/2FV systems

[I: F-LAH_RxSWIN-480]

The DataIdentifier "0x0248-Slave_list_configuration_hash" (hashes of the configuration for sub-bus systems) contains the integrity validation data for the configuration data of all lower-level diagnostic class 2/2F/2FV systems.

[allg. Anf.: F-LAH_RxSWIN-127]

The diagnostic class 4-low systems must use the ReadDataByIdentifier (22hex) service to output the integrity validation data for the configuration data of the diagnostic class 2/2F/2FV systems via group DataIdentifier "0x0248-Slave_list_configuration_hash."

[allg. Anf.: F-LAH_RxSWIN-481]

A higher-level diagnostic class 4-low/4V-low system uses the ReadDataByIdentifier (22hex) service to collect the hashes for all of its lower-level diagnostic class 2/2F/2FV systems via the DataIdentifier "0x0245-Configuration_hash" and calculates the overall hash of the hashes acquired from the lower-level systems.

[allg. Anf.: F-LAH_RxSWIN-129]

The "0x0248-Slave_list_configuration_hash" group DataIdentifier must not be writable using the WriteDataByIdentifier (2Ehex) service.

[allg. Anf.: F-LAH_RxSWIN-482]

All diagnostic class 4-low/4V-low systems with lower-level diagnostic class 2/2F/2FV systems must implement the group DataIdentifier "0x0248-Slave_list_configuration_hash" as a mandatory requirement.

4.3 Requirements for diagnostic class 4 systems based on Q-LAH 80127 up to version 4.0

[I: F-LAH_RxSWIN-425]

For diagnostic class 4 systems up to 80127 v4.0 (without a group DataIdentifier), system-specific DataIdentifiers are reserved in a separate DataIdentifier range for the integration validation data of diagnostic class 2 systems.

[I: F-LAH_RxSWIN-735]

Example of system-specific determination of the identification service (DataIdentifier and ODX designation):

DID value range for programming hash: 0xA800 – 0xA9FF
Sub-system node (SSN): 0x01
Long name part 1: Control_unit_for_wiper_motor
Separator: "_"
Long name part 2: Programming_hash

ODX formation rule:
Combination of long name part 1 + separator + long name part 2 =
"Control_unit_for_wiper_motor_Programming_hash"

DID formation rule: Offset 0xA800 + SSN = "0xA801"

Note: These formation rules also apply to the configuration hash.

4.3.1 Programming hash

[allg. Anf.: F-LAH_RxSWIN-399]

Diagnostic class 4-low systems with lower-level diagnostic class 2F systems must support the "Slave_x_programming_hash" identification service.

[allg. Anf.: F-LAH_RxSWIN-484]

The value of the concrete "Slave_x_programming_hash" identification service is yielded as follows for each diagnostic class 2F system with an SSN less than/equal to 0x1FF: DataIdentifier 0xA800 + SubSystemNodeAddress (SSN).

[allg. Anf.: F-LAH_RxSWIN-400]

Diagnostic class 4 systems must use the standardized DataIdentifier "0x0249-Programming_hash" to request the data of the "Slave_x_programming_hash" DIDs from the lower-level diagnostic class 2F systems.

[allg. Anf.: F-LAH_RxSWIN-485]

In the case of diagnostic class 2F systems with an SSN less than/equal to 0x1FF, the individual DataIdentifiers can also be used for the request.

[I: F-LAH_RxSWIN-488]

Note: In the case of diagnostic class 2F systems with an SSN greater than 0x1FF, the diagnostic class 4-low system must use group DataIdentifiers as a mandatory requirement.

[allg. Anf.: F-LAH_RxSWIN-529]

The "Slave_x_programming_hash" identification service is read only; it must not be writable using the WriteDataByIdentifier (2Ehex) service.

4.3.2 Configuration hash

[allg. Anf.: F-LAH_RxSWIN-396]

Diagnostic class 4-low systems with lower-level diagnostic class 2/2F systems must support the "Slave_x_configuration_hash" identification service.

[allg. Anf.: F-LAH_RxSWIN-486]

The value of the concrete "Slave_x_configuration_hash" identification service is yielded as follows for each diagnostic class 2/2F system with an SSN less than/equal to 0x1FF: DataIdentifier 0xAA00 + SubSystemNodeAddress (SSN).

[allg. Anf.: F-LAH_RxSWIN-397]

Diagnostic class 4-low systems must use the standardized DataIdentifier "0x0245-Configuration_hash" to request the data of the "Slave_x_configuration_hash" DIDs from the lower-level diagnostic class 2/2F systems.

[allg. Anf.: F-LAH_RxSWIN-487]

In the case of diagnostic class 2/2F systems with an SSN less than/equal to 0x1FF, the individual DataIdentifiers can also be used for the request.

[I: F-LAH_RxSWIN-489]

Note: In the case of diagnostic class 2/2F systems with an SSN greater than 0x1FF, the diagnostic class 4-low system must use group DataIdentifiers as a mandatory requirement.

[allg. Anf.: F-LAH_RxSWIN-528]

The "Slave_x_configuration_hash" identification service is read only; it must not be writable using the WriteDataByIdentifier (2Ehex) service.

4.4 Requirements for diagnostic class 2/2F/2FV systems

4.4.1 Programming hash

[I: F-LAH_RxSWIN-186]

The "0x0249-Programming_hash" DataIdentifier is used to output the integrity validation data of the programming of a diagnostic class 2F/2FV server.

[allg. Anf.: F-LAH_RxSWIN-190]

The "0x0249-Programming_hash" DataIdentifier is read only; it must not be writable using the WriteDataByIdentifier (2Ehex) service.

[allg. Anf.: F-LAH_RxSWIN-191]

Starting the "0xFF00-Erase Memory" routine that addresses a logical block with instruction code as per document /6/ means that the existing integrity validation data for the "0x0249-Programming_hash" DataIdentifier on the diagnostic class 2F/2FV system must be deleted.

[allg. Anf.: F-LAH_RxSWIN-744]

Receiving a request to read DataIdentifier "0x0249-Programming_hash" of the hash of a diagnostic class 2F/2FV diagnostic server with the ReadDataByIdentifier (22hex) service causes the hash to be recalculated on the diagnostic class 2F/2FV system. Until recalculation has been completed, a negative response of Response-Pending (NRC 0x78) must be sent for the request. A positive response to the request is not sent until recalculation has been completed.

[allg. Anf.: F-LAH_RxSWIN-738]

On diagnostic class 2F/2FV systems with an SSN less than or equal to 0x1FF, starting the "0xFF00-Erase Memory" routine that addresses a logical block with instruction code as per document /6/ means that the existing integrity validation data in the individual DataIdentifier 0xA800 + SSN "VW_slave_programming_hash" must be deleted.

[allg. Anf.: F-LAH_RxSWIN-626]

On starting the "0xFF00-Erase Memory" routine for a logical block with instruction code as per document /6/, the CRC32 checksum of the addressed block must be deleted.

[allg. Anf.: F-LAH_RxSWIN-627]

Following a positive check by the "0x0202-Check Memory" routine, confirming that the logical block was transferred without error as per document /6/, the CRC32 checksum of the logical block must be recalculated.

[allg. Anf.: F-LAH_RxSWIN-719]

Following a positive response for the "0x0202-Check Memory" routine that addresses a logical block with instruction code, the calculated CRC32 checksum of the logical block is persistently stored.

[allg. Anf.: F-LAH_RxSWIN-490]

To guarantee the backward compatibility of diagnostic class 2F/2FV systems with an SSN less than/equal to 0x1FF to diagnostic class 4 systems based on Volkswagen standard VW 80127 V4.0, the individual DataIdentifiers from the range 0xA800 to 0xA9FF must also be supported with the same content/functionality in addition to the DataIdentifier "0x0249-Programming_hash."

[I: F-LAH_RxSWIN-736]

Note: The generic DataIdentifier "0x0249-Programming_hash" on the diagnostic class 2F/2FV system requires the group DataIdentifier "0x0247-Slave_list_programming_hash" on the higher-level diagnostic class 4-low system.

4.4.2 Configuration hash

[I: F-LAH_RxSWIN-491]

The "0x0245-Configuration_hash" DataIdentifier is used to output the integrity validation data for the configuration data of a diagnostic class 2/2F/2FV server.

[allg. Anf.: F-LAH_RxSWIN-130]

The "0x0245-Configuration_hash" DataIdentifier is read only; it must not be writable using the WriteDataByIdentifier (2Ehex) service.

[allg. Anf.: F-LAH_RxSWIN-138]

Receiving a request to read DataIdentifier "0x0245-Configuration_hash" of the hash of a diagnostic class 2/2F/2FV diagnostic server with the ReadDataByIdentifier (22hex) service causes the hash to be recalculated on the diagnostic class 2/2F/2FV system. Until recalculation has been completed, a negative response of Response-Pending (NRC 0x78) must be sent for the request. A positive response to the request is not sent until recalculation has been completed.

[allg. Anf.: F-LAH_RxSWIN-739]

On diagnostic class 2F/2FV systems with an SSN less than or equal to 0x1FF, receipt of the request for reading the individual DataIdentifier 0xAA00 + SSN "VW_slave_configuration_hash" with the ReadDataByIdentifier (22hex) service means that the integrity validation data must be recalculated.

[I: F-LAH_RxSWIN-140]

The hash is calculated for all received writable DataIdentifiers contained in the "0x0250-Integrity_validation_data_configuration_list" DataIdentifier list. This calculation is described in detail in section "Integrity validation data/general requirements/configuration hash."

[allg. Anf.: F-LAH_RxSWIN-492]

To guarantee the backward compatibility of diagnostic class 2F/2FV systems with an SSN less than/equal to 0x1FF to diagnostic class 4 systems based on VW 80127 V4.0, the individual DataIdentifiers from the range 0xAA00 to 0xABFF must also be supported with the same content/functionality in addition to the DataIdentifier "0x0245-Configuration_hash."

[I: F-LAH_RxSWIN-737]

Note: The generic DataIdentifier "0x0245-Configuration_hash" on the diagnostic class 2F/2FV system requires the group DataIdentifier "0x0248-Slave_list_configuration_hash" on the higher-level diagnostic class 4-low system.

4.5 Standard software module for integrity validation data

[I: F-LAH_RxSWIN-845]

Volkswagen AG provides various standard software (SSW) modules (AUTOSAR v4.3) and a reference implementation in C for the implementation of the integrity validation data.

4.6 Sequence

[I: F-LAH_RxSWIN-931]

The following figures show examples of sub-sequences; the sub-sequences must not be implemented.

4.6.1 Example of reading out all relevant identification data and integrity validation data for a diagnostic server

[I: F-LAH_RxSWIN-246]

A diagnostic client reads out the IDs of a diagnostic class 4 system with lower-level diagnostic class 2/2F/2FV systems.
The following are read out:

[I: F-LAH_RxSWIN-836]

- Vehicle- and ECU-specific identification data

[I: F-LAH_RxSWIN-835]

- Integrity validation data of the configuration data for the adaptation/coding/application and boot loader data sets of a diagnostic class 4 system

[I: F-LAH_RxSWIN-834]

- Integrity validation data of the programming (instruction code) of a diagnostic class 4 system

[I: F-LAH_RxSWIN-833]

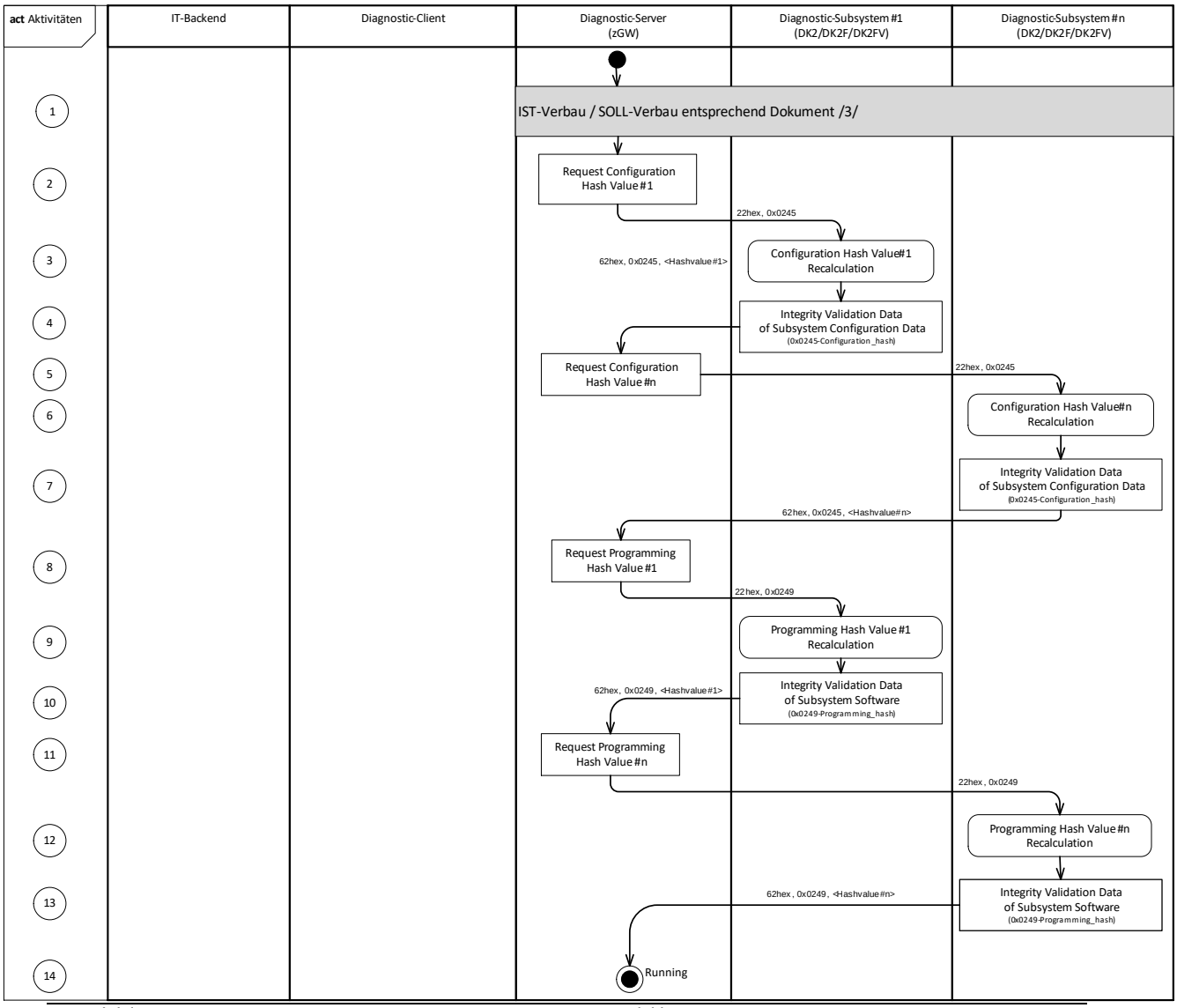
- Integrity validation data of the configuration data for the adaptation (data) of a lower-level diagnostic class 2/2F/2FV system

[I: F-LAH_RxSWIN-832]

- Integrity validation data of the programming (instruction code) of a lower-level diagnostic class 2F/2FV system

[I: F-LAH_RxSWIN-670]

Figure 4-10: Initializing the group DataIdentifiers for the integrity validation data on an example gateway, part 1/4



act Aktivitäten	act Activities
IST-Verbau / SOLL-Verbau entsprechend Dokument /3/	ACTUAL installation/DESIRED installation as per document /3/

[I: F-LAH_RxSWIN-865]

1 – After the CGW starts up, it updates the actual installation list with all detected, lower-level diagnostic sub-systems as per document /3/, section "Acquiring the identification data of diagnostic class 1/2/2F systems."

[I: F-LAH_RxSWIN-671]

2 – After updating the installation list of the CGW, the CGW requests the "0x0245-Configuration_hash" DataIdentifier for the configuration data hash from the lower-level diagnostic sub-system#1.

[I: F-LAH_RxSWIN-672]

3 – On diagnostic sub-system#1, receiving the ReadDataByIdentifier request with the DataIdentifier "0x0245-Configuration_hash" triggers the recalculation of the configuration data hash.

[I: F-LAH_RxSWIN-673]

4 – Diagnostic sub-system#1 sends the response with the recalculated configuration data hash. The CGW updates the group DataIdentifier "0x0248-Slave_list_configuration_hash" based on the received "0x0245-Configuration_hash" DataIdentifier.

[I: F-LAH_RxSWIN-674]

5 – The CGW requests diagnostic sub-system#n's "0x0245-Configuration_hash" DataIdentifier for the configuration data hash from the lower-level diagnostic sub-system#n.
Note: The requests for diagnostic sub-systems#1 to #n can occur in parallel on the buses to save time; only the sequential request is shown here.

[I: F-LAH_RxSWIN-675]

6 – On diagnostic sub-system#n, receiving the ReadDataByIdentifier request with the DataIdentifier "0x0245-Configuration_hash" triggers the recalculation of the configuration data hash.

[I: F-LAH_RxSWIN-676]

7 – Diagnostic sub-system#n sends the recalculated configuration data hash. The CGW updates the group DataIdentifier "0x0248-Slave_list_configuration_hash" based on the received "0x0245-Configuration_hash" DataIdentifier.

[I: F-LAH_RxSWIN-677]

8 – The CGW requests the "0x0249-Programming_hash" DataIdentifier for the programming hash from the lower-level diagnostic sub-system#1.

[I: F-LAH_RxSWIN-710]

9 – On diagnostic sub-system#1, receiving the ReadDataByIdentifier request with the DataIdentifier "0x0249-Programming_hash" triggers the recalculation of the programming hash.

[I: F-LAH_RxSWIN-679]

10 – The lower-level diagnostic sub-system#1 sends the stored programming hash. The CGW updates the group DataIdentifier "0x0247-Slave_list_programming_hash" based on the received "0x0249-Programming_hash" DataIdentifier.

[I: F-LAH_RxSWIN-680]

11 – The CGW requests the "0x0249-Programming_hash" DataIdentifier for the programming hash from the diagnostic sub-system#n.

[I: F-LAH_RxSWIN-711]

12 – On diagnostic sub-system#n, receiving the ReadDataByIdentifier request with the DataIdentifier "0x0249-Programming_hash" triggers the recalculation of the programming hash.

[I: F-LAH_RxSWIN-682]

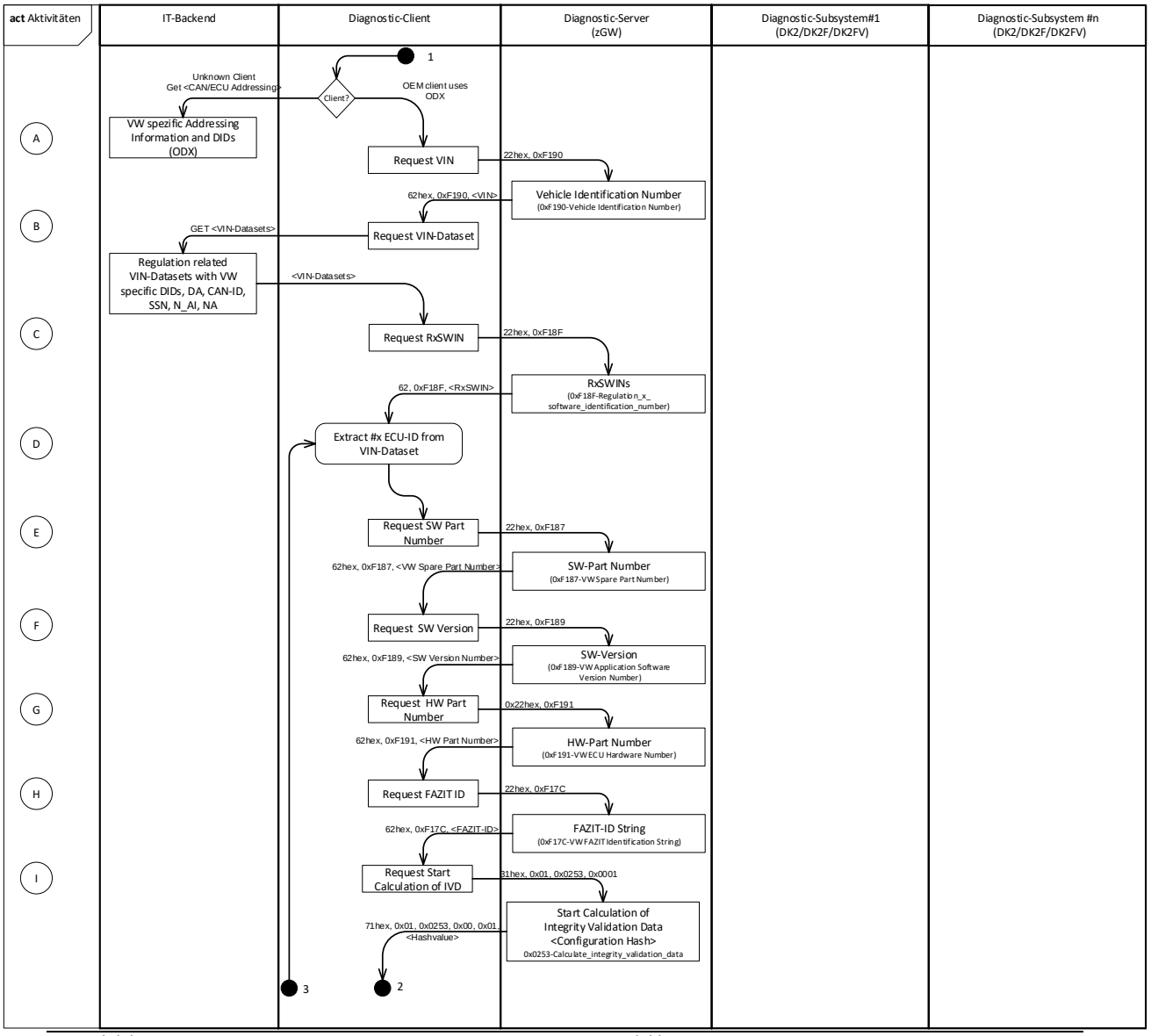
13 – Diagnostic sub-system#n sends the stored programming hash. The CGW updates the group DataIdentifier "0x0247-Slave_list_programming_hash" based on the received "0x0249-Programming_hash" DataIdentifier.

[I: F-LAH_RxSWIN-683]

14 – The CGW updates the other group DataIdentifiers of the diagnostic sub-systems.

[I: F-LAH_RxSWIN-245]

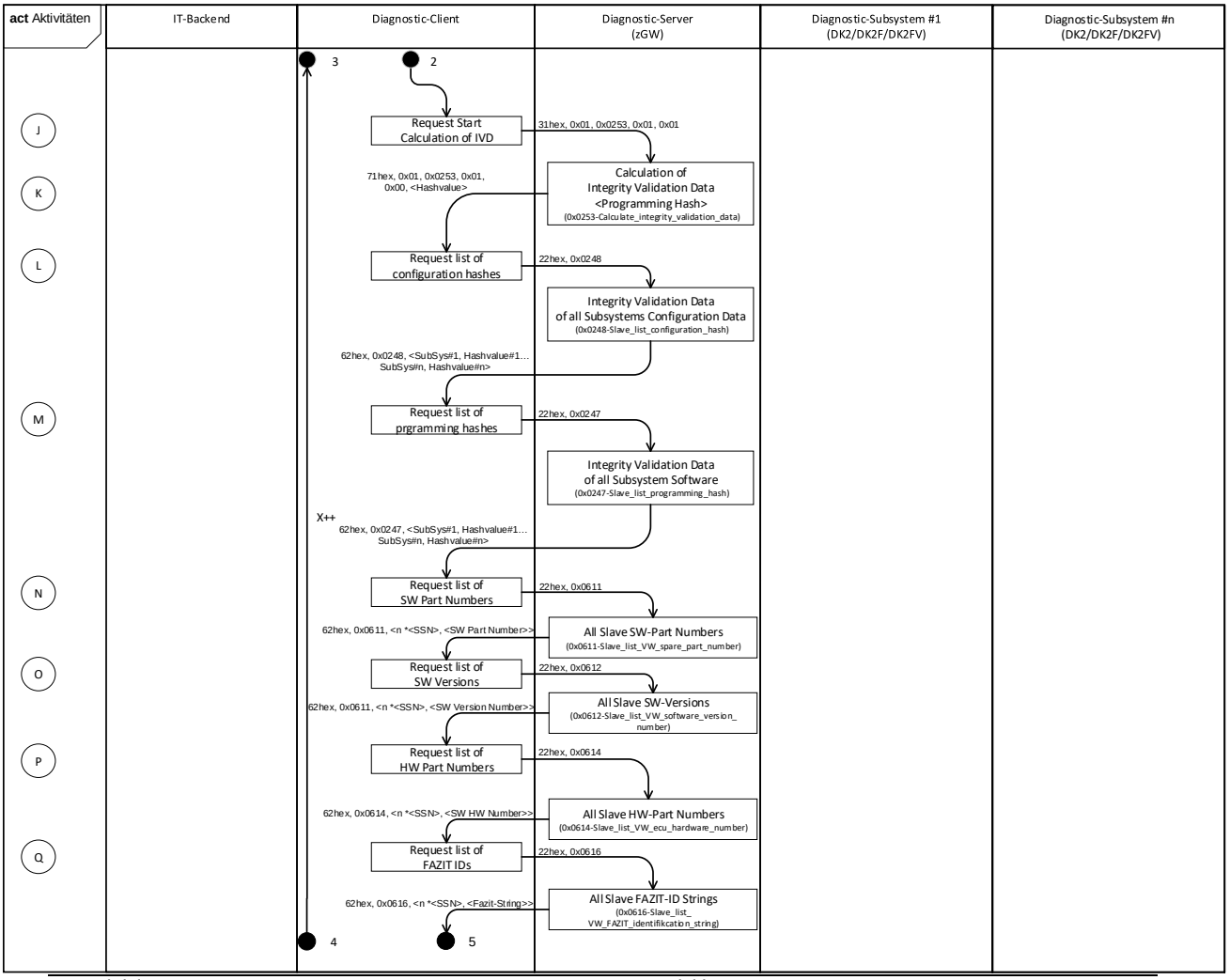
Figure 4-11: Reading out all identification data and integrity validation data, part 2/4



act Aktivitäten	act Activities
VW spezific Addressing Information and DIDs (ODX)	Addressing information and DIDs (ODX) specific to Volkswagen

[I: F-LAH_RxSWIN-248]

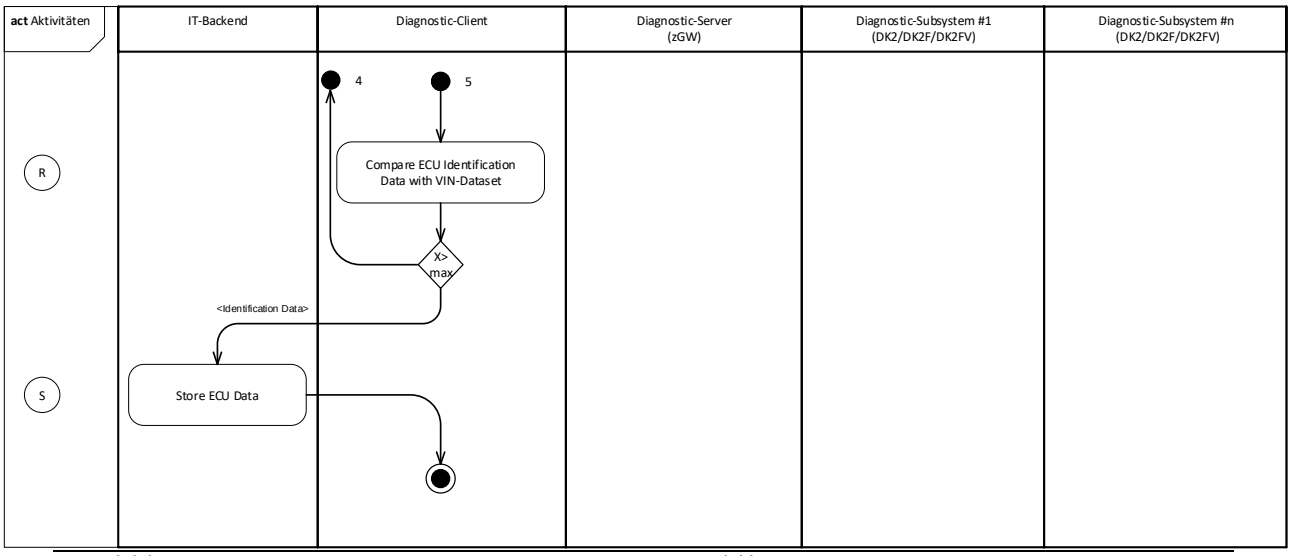
Figure 4-12: Reading out all identification data and integrity validation data, part 3/4



act Aktivitäten — act Activities

[I: F-LAH_RxSWIN-250]

Figure 4-13: Reading out all identification data and integrity validation data, part 4/4



act Aktivitäten act Activities

[I: F-LAH_RxSWIN-249]

A – The diagnostic client requests the addressing information required for communicating with the vehicle and the DataIdentifiers required for the vehicle's identification data from the IT backend. Addressing information means the diagnostic addresses (DA), node addresses (NA), sub-system node addresses (SSN), network address information (N_AI), and the CAN identifiers for requests and responses. The diagnostic client uses the DataIdentifier "0xF190-Vehicle Identification Number" to request the vehicle identification number (VIN) of the current vehicle. If the Offboard Diagnostic Information System (ODIS) is used, all necessary information is available in the ODX data.

[I: F-LAH_RxSWIN-251]

B – The diagnostic client requests the vehicle-specific and regulation-related data set for the vehicle's VIN from the IT backend to additionally receive addressing information, such as diagnostic addresses, node addressed, and DataIdentifiers.

[I: F-LAH_RxSWIN-252]

C – The diagnostic client requests the DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" with the list of all regulation numbers and software identification numbers from the central gateway.

[I: F-LAH_RxSWIN-253]

D – The diagnostic client uses the data from the regulation-related data set to successively address the individual diagnostic servers installed in the vehicle.

[I: F-LAH_RxSWIN-254]

E – The diagnostic client uses the "0xF187-VW Spare Part Number" DataIdentifier of the addressed diagnostic server to read out the software number.

[I: F-LAH_RxSWIN-255]

E – The diagnostic client uses the "0xF189-VW Software Version Number" DataIdentifier of the addressed diagnostic server to read out the software version.

[I: F-LAH_RxSWIN-256]

E – The diagnostic client uses the "0xF191-VW ECU Hardware Number" DataIdentifier of the addressed diagnostic server to read out the hardware part number.

[I: F-LAH_RxSWIN-257]

E – The diagnostic client uses the "0xF17C-VW FAZIT Identification String" DataIdentifier of the addressed diagnostic server to read out the Vehicle Information and Central Identification Tool (FAZIT) ID.

[I: F-LAH_RxSWIN-259]

I – The diagnostic client requests the configuration data hash of the addressed diagnostic server. For this purpose, the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" contains the list of all DataIdentifiers and data set numbers that must be used to calculate the configuration data hash. The "0x0253-Calculate_integrity_validation_data" RoutineIdentifier delivers the diagnostic server's configuration data hash in the positive response.

[I: F-LAH_RxSWIN-260]

J – The diagnostic client starts calculating the overall programming hash for the diagnostic servers it addressed.

[I: F-LAH_RxSWIN-262]

J – The diagnostic client receives the newly calculated programming hash for the diagnostic servers.

[I: F-LAH_RxSWIN-263]

L – The diagnostic client requests the group DataIdentifier "0x0248-Slave_list_configuration_hash" for the configuration data hashes of all diagnostic sub-systems.

[I: F-LAH_RxSWIN-264]

M – The diagnostic client requests the group DataIdentifier "0x0247-Slave_list_programming_hash" for the programming hashes of all diagnostic sub-systems for the diagnostic servers.

[I: F-LAH_RxSWIN-266]

N – The diagnostic client uses the "0x0611-Slave_list_VW_spare_part_number" group DataIdentifier of the addressed diagnostic server to read out the list of all software numbers of the diagnostic sub-systems.

[I: F-LAH_RxSWIN-545]

O – The diagnostic client uses the "0x0612-Slave_list_VW_software_version_number" group DataIdentifier of the addressed diagnostic server to read out the list of all software versions of the diagnostic sub-systems.

[I: F-LAH_RxSWIN-546]

P – The diagnostic client uses the "0x0614-Slave_list_VW_ecu_hardware_number" group DataIdentifier of the addressed diagnostic server to read out the list of all hardware part numbers of the diagnostic sub-systems.

[I: F-LAH_RxSWIN-267]

Q – The diagnostic client uses the "0x0616-Slave_list_VW_FAZIT_identification_string" group DataIdentifier of the addressed diagnostic server to read out the list of all FAZIT IDs of the diagnostic sub-systems.

[I: F-LAH_RxSWIN-270]

R – The diagnostic client compares the identification data for the RxSWIN with the data set from the IT backend and then reads out the identification data for the next diagnostic server.

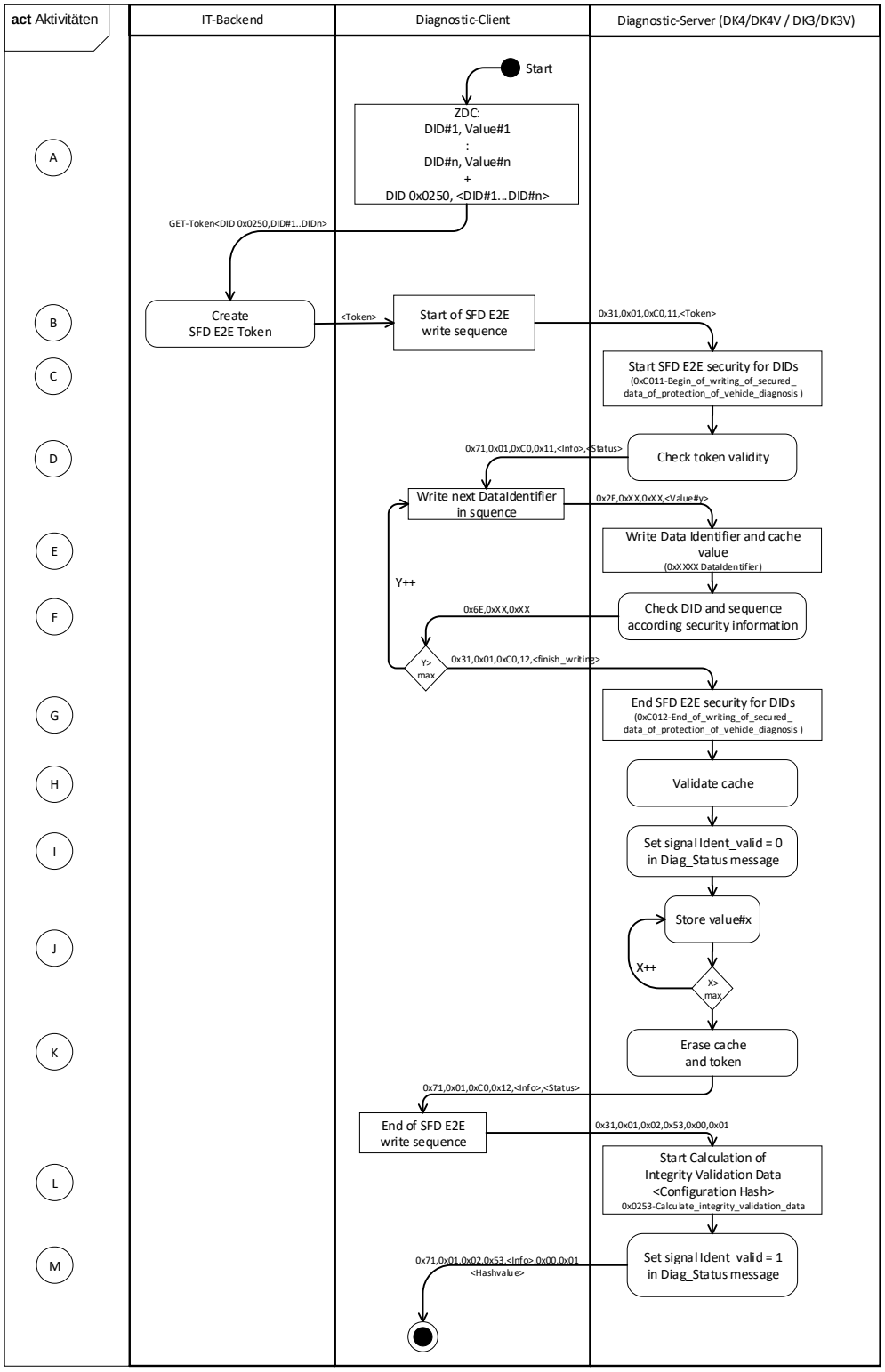
[I: F-LAH_RxSWIN-547]

S – The diagnostic client stores the identification data it has read out on the IT backend.

4.6.2 Example of writing data to a diagnostic class 4/4V or diagnostic class 3/3V system secured by PVD E2E

[I: F-LAH_RxSWIN-356]

Figure 4-14: Writing data for a diagnostic class 3/3V or diagnostic class 4/4V system, part 1/1



act Aktivitäten — act Activities

[I: F-LAH_RxSWIN-537]

A – The DataIdentifiers required for the ECU configuration are provided to the diagnostic client. The DataIdentifiers result from the production order that configures the ZDC. The result is the content of the "0x0250-Integrity_validation_data_configuration_list" DataIdentifier. The diagnostic client requests a validation token for the DataIdentifiers included in the configured ZDC, DID#1 – DID#n, and the calculated DataIdentifier "0x0250-Integrity_validation_data_configuration_list" from the PVD IT backend.

[I: F-LAH_RxSWIN-359]

B* – The PVD IT backend generates a validation token for the DataIdentifiers DID#1 – DID#n and DID 0x0250.

[I: F-LAH_RxSWIN-360]

C* – The validation token is transferred to the diagnostic server upon starting the PVD routine "0xC011-Begin_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis." This triggers the write sequence for the PVD end-to-end secured data.

[I: F-LAH_RxSWIN-361]

D* – The diagnostic server checks the validity of the PVD validation token it received. In accordance with PVD E2E security as per document /4/, the validation information is deleted if an invalid token is received.

[I: F-LAH_RxSWIN-362]

E – The DataIdentifiers from the configured ZDC and the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" are transferred to the diagnostic server using the WriteDataByIdentifier (2Ehex) service and cached. Without PVD, the data is written directly to memory.

[I: F-LAH_RxSWIN-363]

F* – In accordance with PVD E2E security as per document /4/, the receive order of the individual DataIdentifiers is checked according to the validation information. If the check returns a positive result, the diagnostic server sends a positive response. If the check returns a negative result, the received DataIdentifiers, the corresponding DataRecords, and the validation information are deleted. If the check returns a negative result, the diagnostic server sends a negative response code NRC 0x22 (ConditionsNotCorrect).

[I: F-LAH_RxSWIN-364]

G* – The PVD write sequence is completed when the "0xC012-End_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis" PVD routine starts with the RoutineControlOption [0x01-Finish_writing_of_secured_data].

[I: F-LAH_RxSWIN-365]

H* – The authenticity of the transferred DataIdentifiers is checked using the validation information.

[I: F-LAH_RxSWIN-368]

I – An [Ident_valid] signal with a value of '0' in the diagnostic server's Diag_Status message indicates that the identification data is currently not completely valid. In other words, the configuration data hashes have been deleted and have not yet been recalculated.

[I: F-LAH_RxSWIN-366]

J* – If the check is positive, all cached DataIdentifiers are sequentially written from the cache to the diagnostic server's target memory. If the check returns a negative result, all received DataIdentifiers, the corresponding DataRecords, and the validation information are deleted.

[I: F-LAH_RxSWIN-369]

K* – The cache and the validation information are deleted on the diagnostic server after all DataIdentifiers have been transferred to the target memory.

[I: F-LAH_RxSWIN-370]

L – The diagnostic client requests the new configuration data hash. Calling the "0x0253-Calculate_integrity_validation_data" routine starts recalculating the configuration data hash on the diagnostic server. The hash is calculated according to the list of DataIdentifiers and data set numbers transferred in the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" (see section "0x0250-Integrity_validation_data_configuration_list").

[I: F-LAH_RxSWIN-371]

M – An [Ident_valid] signal with a value of '1' in the diagnostic server's Diag_Status message indicates that the identification data is completely valid (i.e., the hashes were updated).

[I: F-LAH_RxSWIN-527]

*Only relevant to diagnostic systems secured by PVD E2E. Not relevant to unsecured diagnostic systems with a group release in production before checkpoint 8.

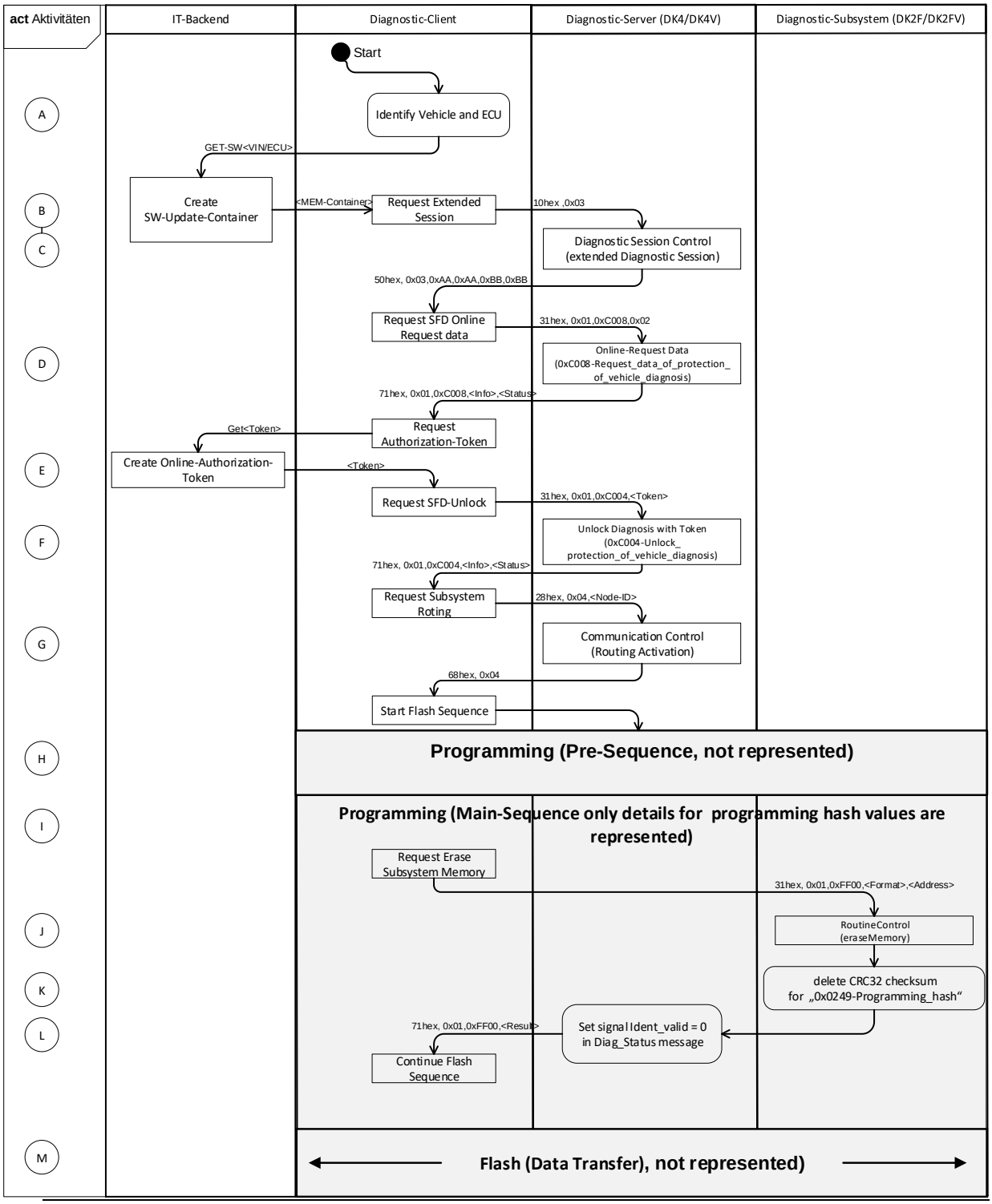
4.6.3 Example of programming a diagnostic class 2F/2FV system

[I: F-LAH_RxSWIN-303]

Sequence of a software update by a diagnostic client on a diagnostic class 2F/2FV system via a diagnostic class 4/4V system

[I: F-LAH_RxSWIN-305]

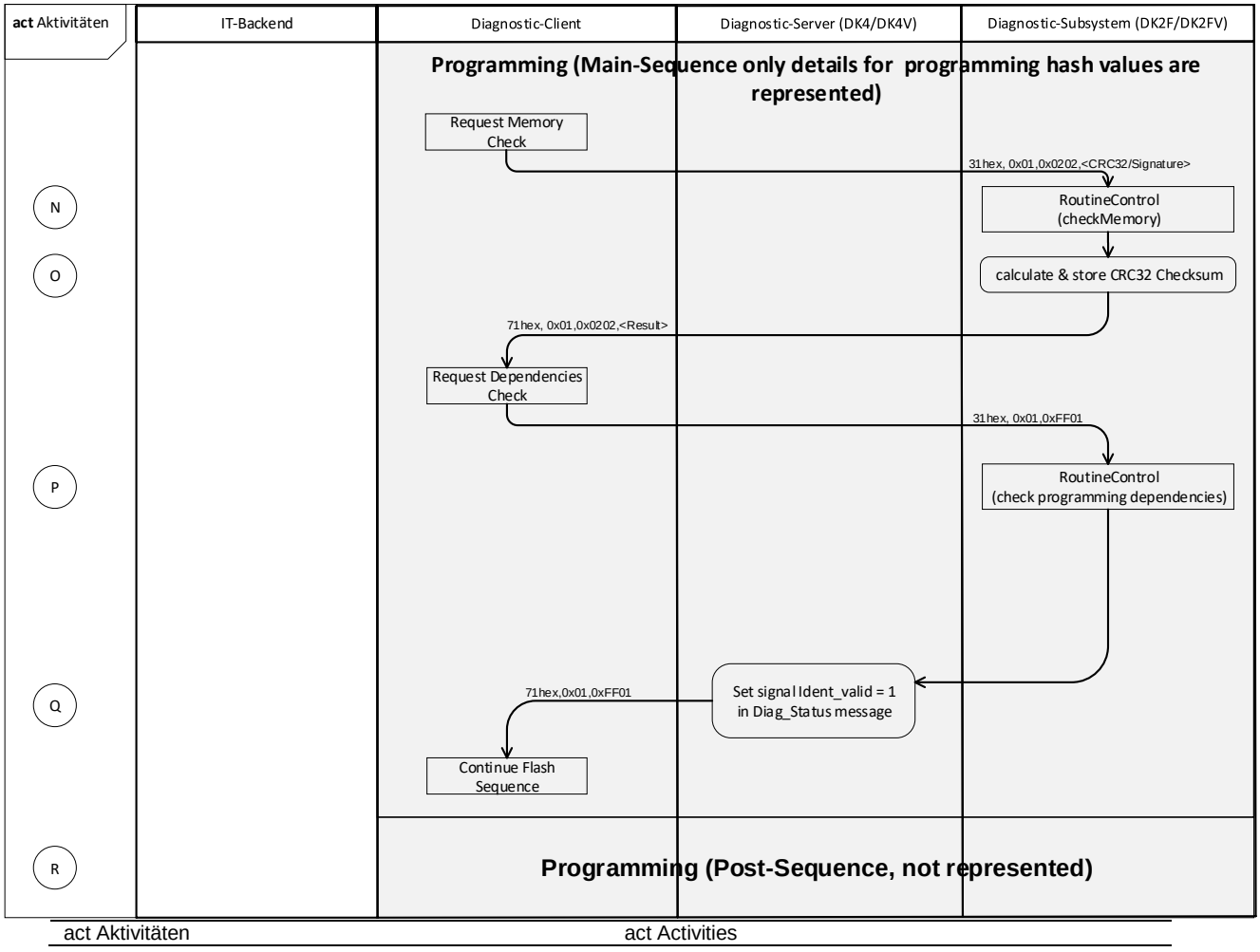
Figure 4-15: Example of programming a diagnostic class 2F/2FV system, part 1/2



act Aktivitäten act Activities

[I: F-LAH_RxSWIN-306]

Figure 4-16: Example of programming a diagnostic class 2F/2FV system, part 2/2



[I: F-LAH_RxSWIN-307]

A – The diagnostic client identifies the current vehicle using the VIN and identifies the ECU to be updated using its identification data.

[I: F-LAH_RxSWIN-310]

B – The diagnostic client downloads the appropriate software update package (MEM container), which was compiled by the IT backend.

[I: F-LAH_RxSWIN-311]

C – The diagnostic client requests a session change to the Extended Diagnostic Session 0x03 for the diagnostic server.

[I: F-LAH_RxSWIN-312]

D – The diagnostic client requests the request data for a PVD online unlocking from the diagnostic server.

[I: F-LAH_RxSWIN-313]

E – The diagnostic client uses the request data to request an unlocking token from the PVD backend.

[I: F-LAH_RxSWIN-314]

F – The diagnostic client uses the unlocking token to unlock the diagnostic server so it can use diagnostics to activate routing on the diagnostic server.

[I: F-LAH_RxSWIN-315]

G – The diagnostic client activates routing of diagnostics messages to the diagnostic sub-system on the diagnostic server.

[I: F-LAH_RxSWIN-316]

H – Programming sequence, pre-sequence as per document /6/

[I: F-LAH_RxSWIN-317]

I – Programming sequence, main sequence as per document /6/

[I: F-LAH_RxSWIN-318]

J – The diagnostic client requests deletion of a logical block of the diagnostic sub-system.

[I: F-LAH_RxSWIN-319]

K – Using the "0xFF00-Erase Memory" routine to delete the memory means that the CRC32 checksum for the block to be deleted is deleted at the same time.

[I: F-LAH_RxSWIN-523]

L – The diagnostic class 4/4V system uses the Diag_Status message in the "Ident_valid" = 0 signal to indicate that the ECU identification is no longer valid.

[I: F-LAH_RxSWIN-320]

M – The diagnostic client executes the flash sequence with the diagnostic sub-system.

[I: F-LAH_RxSWIN-321]

N – After the flash sequence, the diagnostic client sends a request to the diagnostic sub-system to check that the memory block was received correctly.

[I: F-LAH_RxSWIN-322]

O – The diagnostic server routes the request to check the memory directly to the diagnostic sub-system. The diagnostic sub-system calculates the logical block's CRC32 checksum and stores it persistently.

[I: F-LAH_RxSWIN-323]

P – After the flash sequence, the diagnostic client sends a request to the diagnostic sub-system to check the correctly received memory block for compatibility/validity.

[I: F-LAH_RxSWIN-525]

Q – The diagnostic server uses the Diag_Status message to indicate that the ECU identification is valid.

[I: F-LAH_RxSWIN-324]

H – Programming sequence, post sequence as per document /6/

5 Diagnostic objects

[I: F-LAH_RxSWIN-54]

The following table shows the overview of SUMS-relevant diagnostic functions for the respective diagnostic classes.

[allg. Anf.: F-LAH_RxSWIN-180]

Table 5-1: Diagnostic functions and diagnostic classes

	Diag-nostic class 4-high	Diag-nostic class 4-low	Diag-nostic class 4V-low	Diag-nostic class 4*)	Diag-nostic class 3	Diag-nostic class 3V	Diag-nostic class 2	Diag-nostic class 2F	Diag-nostic class 2FV	SWCL	Note
Diagnostic object/function											
0xF18F-Regulation_x_software_identification_numbers	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	Must only be implemented once per vehicle in the central gateway
Software part numbers in DID "0xF187-VW Spare Part Number"	x	x	x	x	x	x	x	x	x	x	
Software version in DID "0xF189-VW Software Version Number"	x	x	x	x	x	x	x	x	x	x	
Hardware (HW) part number in DataIdentifier (DID) "0xF191-VW ECU Hardware Number"	x	x	NoImp	NoImp	x	NoImp	x	x	NoImp	NoImp	
HW version in DID "0xF1A3-VW ECU Hardware Version Number"	x	x	NoImp	NoImp	x	NoImp	x	x	NoImp	NoImp	
VIN in DID "0xF190-Vehicle Identification Number"	(x)	(x)	(x)	(x)	(x)	(x)	NoImp	NoImp	NoImp	NoImp	VIN only for immobilizer nodes, DoIP servers, and CGW
Configuration hash in DID "0245-Configuration_hash"	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	x	x	x	NoImp	Configuration hash
Group DID 0x0248-Slave_list_configuration_hash for "0x0245-Configuration_hash"	NoImp	x	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	Only with lower-level diagnostic class 2 systems for diagnostic class 4-high
Programming hash in DID "0249-Programming_hash"	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	x	x	NoImp	Program hash
Group DID 0x0247-Slave_list_programming_hash for "0x0249-Programming_hash"	NoImp	x	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	Only with lower-level diagnostic class 2 systems for diagnostic class 4-high
0x0251_Write_generic_to_sub_system	NoImp	x	x	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	Only with lower-level diagnostic class 2 systems for diagnostic class 4-high
List of relevant DIDs for calculating hash in DID "0x0250-List_of_configuration_data_identifier"	x	x	x	x	x	x	x	x	x	x	For diagnostic class 2/2F: Calibration occurs via the bus master/host using 0x0251.
Calculation of integrity validation data with RID "0x0253-Calculate_integrity_validation_data"	x	x	x	x	x	x	NoImp	NoImp	NoImp	x	Hash of program and data
Calculation of individual hashes with RID "0x0254-Calculate_individual_hash_value"	x	x	x	x	x	x	NoImp	NoImp	NoImp	x	Individual hash of configuration data
Programming hash for Q-LAH 80127 v4.0 sub-systems (DID range 0xA800 – 0xA9FF)	NoImp	(x)	(x)	x	NoImp	NoImp	NoImp	x	NoImp	NoImp	Only for diagnostic class 4s with diagnostic class 2Fs on existing platforms (individual DID for 0608 identification as per 80127 up to v4.0)
Configuration hash for Q-LAH 80127 v4.0 sub-systems (DID range 0xAA00 – 0xABFF)	NoImp	(x)	(x)	x	NoImp	NoImp	x	x	NoImp	NoImp	

[I: F-LAH_RxSWIN-855]

*) = diagnostic class 4 as per Q-LAH 80127 up to v4.0 (without subdivision into diagnostic class 4-low/4-high)

[I: F-LAH_RxSWIN-856]

x = included

[I: F-LAH_RxSWIN-857]

(x) = optional

5.1 DataIdentifiers

[allg. Anf.: F-LAH_RxSWIN-933]

Table 5-2: Overview of DataIdentifier properties

			DiagnosticSession		
			Default Session	NonDefaultSession: ECUProgram-mingSession	Extended-Diagnostic-Session
Diagnoseobjekte (DID)	**SID**	**Security Level**	**0x01**	**0x02**	**0x03**
0xF1A3-VW_ECU_Hardware_version_number	22hex	NoImp	Available	Available	Available
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable
0xF1B8-VW_system_firmware_versions	22hex	NoImp	NoImp	NoImp	NoImp
	2Ehex	NoImp	NoImp	NoImp	NoImp
0xF1A0-VW_data_set_number_or_ECU_data_container_number	22hex	NoImp	NoImp	NoImp	NoImp
	2Ehex	NoImp	NoImp	NoImp	NoImp
0xF1A1-VW_data_set_version_number	22hex	NoImp	NoImp	NoImp	NoImp
	2Ehex	NoImp	NoImp	NoImp	NoImp
0x0249-Programming_hash	22hex	NoImp	Available	notAvailable	Available
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable
0x0247-Slave_list_programming_hash	22hex	NoImp	Available	notAvailable	Available
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable
0x0245-Configuration_hash	22hex	NoImp	Available	notAvailable	Available
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable
0x0248-Slave_list_configuration_hash	22hex	NoImp	Available	notAvailable	Available
	2Ehex	NoImp	notAvaillable	notAvailable	notAvailable
0xF18F-Regulation_x_software_identification_number	22hex	NoImp	Available	notAvailable	Available
	2Ehex	Old platform: SFD-BASIC New platform: SFD-E2E	notAvailable	notAvailable	Available
0x0250-Integrity_validation_data_configuration_list	22hex	NoImp	Available	notAvailable	Available
	2Ehex	SFD-E2E	notAvailable	notAvailable	C1
0x0251-Write_generic_to_sub_system	22hex	NoImp	notAvailable	notAvailable	notAvailable
	2Ehex	Old platform: C2 New platform: SFD-E2E	notAvailable	notAvailable	Available

[I: F-LAH_RxSWIN-935]

C1 – For systems that cannot be configured using adaptation/coding or data sets in the application, this DataIdentifier must not be writable since no PVD E2E security is required for the configuration in this case. In this case, all possible boot loader data sets must be contained in the response of the "0x0250-Integrity_validation_data_configuration_list" DataIdentifier.

[I: F-LAH_RxSWIN-936]

C2 – The diagnostic class 4-low's access protection method (if available) must be used.

5.1.1 0xF1A3-VW ECU Hardware Version Number

[I: F-LAH_RxSWIN-664]

This DataIdentifier is used to identify the hardware version of an ECU.

[allg. Anf.: F-LAH_RxSWIN-701]

The following changes compared with Q-LAH 80125 up to version 5.8 must be observed:

[allg. Anf.: F-LAH_RxSWIN-806]

Eliminated:

- Changes in the hardware version do not require a new part number suffix/version number for the vehicle production process or the spare parts business.
- Whether or not a change compatible with the software and hardware is documented by the hardware version is left to the discretion of the component engineer in consultation with the contractor.

[allg. Anf.: F-LAH_RxSWIN-805]

New:
The following changes must be documented by the hardware version:

- Changes to active components (e.g., microcontroller, microprocessor, RAM, flash, ASIC)
- End-of-life for passive components
- Changes to the printed circuit board (e.g., relating to electromagnetic compatibility, current draw)
- Changes to functional behavior (e.g., optional reserve is populated)
- Changes influence interfaces

5.1.2 0xF1A0-VW Data Set Number Or ECU Data Container Number

[allg. Anf.: F-LAH_RxSWIN-765]

The following change compared with Q-LAH 80125 up to version 5.8 must be observed:

- Implementing DataIdentifier "0xF1A0-VW Data Set Number Or ECU Data Container Number" on the diagnostic server is no longer permissible.

[Prozess-Anf.: F-LAH_RxSWIN-802]

The configuration hash is used to document the configuration data written via the ZDC.

[Prozess-Anf.: F-LAH_RxSWIN-766]

Note: The target data container's part number must still be included in the build status documentation (see section "RxSWIN-specific documentation").

5.1.3 0xF1A1-VW Data Set Version Number

[allg. Anf.: F-LAH_RxSWIN-799]

The following change compared with Q-LAH 80125 up to version 5.8 must be observed:

- Implementing DataIdentifier "0xF1A1-VW Data Set Version Number" on the diagnostic server is no longer permissible.

[Prozess-Anf.: F-LAH_RxSWIN-800]

The configuration hash is used to document the configuration data written via the ZDC.

[Prozess-Anf.: F-LAH_RxSWIN-801]

Note: The target data container's version must still be included in the build status documentation (see section "RxSWIN-specific documentation").

5.1.4 0x0249-Programming_hash

[allg. Anf.: F-LAH_RxSWIN-449]

Table 5-3: Structure of DataRecord for 0x0249-Programming_hash DataIdentifier

DID	**0x0249**
Designation	Programming_hash
Description	This DataIdentifier contains the integrity validation data for the program data. The hash is calculated for the entire software.
Convention	all servers
Diagnostic class	Diagnostic class 2F/2FV
Session	APP: 0x01 (R), 0x03 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[Programming_hash] 32-byte hex>
Range	[Programming_hash] 00..00hex – FF..FFhex
Init	Not applicable, always available
Example	-
Data category	Analysis data
Electronic build status documentation	M

5.1.5 0x0247-Slave_list_programming_hash

[allg. Anf.: F-LAH_RxSWIN-443]

Table 5-4: Structure of DataRecord for 0x0247-Slave_list_programming_hash DataIdentifier

DID	**0x0247**
Designation	Slave_list_programming_hash
Description	This DataIdentifier contains all integrity validation data for the instruction code of the lower-level diagnostic class 2F/2FV systems. The following definitions apply: • The respective content corresponds to the [Programming_hash] parameter of DataIdentifier "0x0249-Programming_hash." • The respective SubSystemNodeAddress must be prepended to the respective content.
Convention	Bus master
Diagnostic class	Diagnostic class 4-low/4V-low
Session	APP: 0x01 (R), 0x03 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[NumberOfExpectedSubSystemIdentification] 1-byte hex> + <[NumberOfRetrievedSubSystemIdentification] 1-byte hex> + <n x <[SubSystemNodeAddress] 2-byte hex> + <[Programming_hash] 32-byte hex>> (n: Number of diagnostic class 2F/2F systems)
Range	[NumberOfExpectedSubSystemIdentification]: 00hex – FFhex [NumberOfRetrievedSubSystemIdentification]: 00hex – FFhex [SubSystemNodeAddress]: 00 00hex – FF FFhex [Programming_hash]: See DID 0x0249.
Init	Not applicable, always available
Example	-
Data category	Analysis data
Electronic build status documentation	NoImp

5.1.6 0x0245-Configuration_hash

[allg. Anf.: F-LAH_RxSWIN-445]

Table 5-5: Structure of DataRecord for 0x0245-Configuration_hash DataIdentifier

DID	**0x0245**
Designation	Configuration_hash
Description	This DataIdentifier contains the integrity validation data for the configuration data.
Convention	all servers
Diagnostic class	Diagnostic class 2/2F/2FV
Session	APP: 0x01 (R), 0x03 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[Configuration_hash] 32-byte hex>
Range	[Configuration_hash] 00..00hex – FF..FFhex
Init	Not applicable, always available
Example	-
Data category	Analysis data
Electronic build status documentation	M

5.1.7 0x0248-Slave_list_configuration_hash

[allg. Anf.: F-LAH_RxSWIN-447]

Table 5-6: Structure of DataRecord for 0x0248-Slave_list_configuration_hash DataIdentifier

DID	**0x0248**
Designation	Slave_list_configuration_hash
Description	This DataIdentifier contains all integrity validation data for the configuration data of the lower-level diagnostic class 2/2F/2FV systems. The following definitions apply: • The respective content corresponds to the [Configuration_hash] parameter of DataIdentifier "0x0245-Configuration_hash." • The respective SubSystemNodeAddress must be prepended to the respective content.
Convention	Bus master
Diagnostic class	Diagnostic class 4-low/4V-low
Session	APP: 0x01 (R), 0x03 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[NumberOfExpectedSubSystemIdentification] 1-byte hex> + <[NumberOfRetrievedSubSystemIdentification] 1-byte hex> + <n x <[SubSystemNodeAddress] 2-byte hex> + <[Configuration_hash] 32-byte hex>> (n: Number of diagnostic class 2/2F/2F systems)
Range	[NumberOfExpectedSubSystemIdentification]: 00hex – FFhex [NumberOfRetrievedSubSystemIdentification]: 00hex – FFhex [SubSystemNodeAddress]: 00 00hex – FF FFhex [Configuration_hash]: See DID 0x0245.
Init	Not applicable, always available
Example	-
Data category	Analysis data
Electronic build status documentation	NoImp

5.1.8 0xF18F-Regulation_x_software_identification_numbers

[allg. Anf.: F-LAH_RxSWIN-14]

Table 5-7: Structure of DataRecord for 0xF18F-Regulation_x_software_identification_numbers DataIdentifier

DID	**0xF18F**
Designation	Regulation_x_software_identification_numbers
Description	This DataIdentifier contains the list of software identification numbers from, e.g., UNECE features or GB/T features defined in Chinese standards that are available in a vehicle. The list is vehicle-specific. The diagnostic server must not validation check any payload content, such as [Length_of_RxSWIN], [Regulation_identification], [Separation_character], and [Software_identification]; such content must be output without modification.
Convention	ECU with central diagnostic access
Diagnostic class	Diagnostic class 4-high
Session	APP: 0x01 (R), 0x03 (R/W) BLF: NoImp
SecurityLevel	ECUs on existing platforms: PVD access protection, BASIC (W) role ECUs on new platforms: PVD E2E (W)
Changing	DIAG
Format	<n x <[Length_of_RxSWIN] 1-byte hex> + <[Regulation_identification] m-byte ASCII, variable)> + <[Separation_character] 1-byte ASCII> + <[Software_identification] 9- to 11-byte ASCII, variable> > (n: Number of RxSWINs, variable; m: Number of bytes for regulation ID, variable)
Range	[Length_of_RxSWIN] 00 – FFhex [Regulation_identification] 21 – 7Ahex [Separation_character] 20hex (ASCII "SPACE") [Software_identification] 30 – 39hex, 41 – 5Ahex, 61 – 7A Other values are reserved by ISO.
Init	2D 2D 2D 2D 2Dhex ('-----')
Example	0F 52 30 37 39 20 76 30 35 37 34 31 37 35 33 61 13 47 42 2F 54 33 36 30 34 37 20 76 30 34 33 36 39 38 35 32 Number of RxSWINs = 2 RxSWIN #1: Length_of_RxSWIN: 0x0F = 15 bytes Regulation_identification: R079 Software_identification: v05741753a RxSWIN #2: Length_of_RxSWIN: 0x13 = 19 bytes Regulation_identification: GB/T36047 Software_identification: v04369852
Data category	Process parameter
Electronic build status documentation	M

[I: F-LAH_RxSWIN-70]

The total length of the RxSWIN must be able to contain at least 50 regulations and the corresponding regulation-related software identifiers.

[allg. Anf.: F-LAH_RxSWIN-71]

This results in a minimum memory size for the DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" of 1 000 bytes.

[allg. Anf.: F-LAH_RxSWIN-67]

Reading DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" via the ReadDataByIdentifier (22hex) service must not be protected by an access protection method.

[allg. Anf.: F-LAH_RxSWIN-66]

The following applies to ECUs on existing platforms: Writing DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" via the WriteDataByIdentifier (2Ehex) service must be secured using the "PVD access protection (BASIC role)" as per document /4/.

[allg. Anf.: F-LAH_RxSWIN-340]

The following applies to ECUs on new platforms: Writing DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" via the WriteDataByIdentifier (2Ehex) service must be secured using "PVD end-to-end security (PVD E2E)" as per document /4/.

5.1.8.1 Requirements for processes

[Prozess-Anf.: F-LAH_RxSWIN-72]

No RxSWIN is written for UNECE features not in the vehicle's production order. In other words, there must be no list entry in the DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" with an RxSWIN for UNECE features that are not available in the actual vehicle.

[Prozess-Anf.: F-LAH_RxSWIN-74]

The DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" has a vehicle-specific length. The calibration process must ensure that the DataIdentifier "0xF18F-Regulation_x_software_identification_numbers" is calculated as per requirement F-LAH_RxSWIN-72.

[I: F-LAH_RxSWIN-555]

Note:
This can be implemented, e.g., as a Java or OTX job.

5.1.8.2 Requirements for IT systems

[Prozess-Anf.: F-LAH_RxSWIN-220]

The data container with the list of RxSWINs is managed in terms of data logistics by a diagnostic class 2V system assigned to the gateway ECU.

[Prozess-Anf.: F-LAH_RxSWIN-803]

Note: This is not a diagnostic class 2V system as referred to in document /3/ or /12/. This diagnostic class 2V system merely contains a diagnostics address required to manage the data in the logistics chain.

5.1.9 0x0250-Integrity_validation_data_configuration_list

[allg. Anf.: F-LAH_RxSWIN-349]

Table 5-8: Structure of DataRecord for 0x0250-Integrity_validation_data_configuration_list DataIdentifier

DID	**0x0250**
Designation	Integrity_validation_data_configuration_list
Description	This DataIdentifier contains the list of identifiers (DataIdentifiers and data set numbers) used to calculate the configuration data hash. This DataIdentifier is initially populated with the list of DataIdentifiers and data set numbers marked as ZDC-relevant in the BT-LAH. This list can be changed in the course of the ZDC process.
Convention	ZDC
Diagnostic class	Diagnostic class 2/2F/2FV, diagnostic class 3/3V, diagnostic class 4/4V, SWCL
Session	APP: 0x01 (R), 0x03 (R/W*) BLF: NoImp
SecurityLevel	PVD E2E (W)
Changing	DIAG
Format	<<[Number_of_identifier] 2-byte hex> + n x <[Identifier] 2-byte hex>> (n: Number of identifiers)
Range	[Number_of_identifier] 00 00hex = No DataIdentifiers or no data set numbers present 00 01hex – FF FFhex [Identifier] 00 00 – FF FFhex
Init	Initialization with the DataIdentifiers and data set numbers marked as ZDC-relevant in the BT-LAH
Example	00 04 12 43 98 67 FE CD 71 01 Number of identifiers = 4 DID#1: 0x1243 DID#2: 0x9867 DID#3: 0xFECD DSNo: 0x7101
Data category	Vehicle parameters
Electronic build status documentation	M

[allg. Anf.: F-LAH_RxSWIN-837]

*) For systems that cannot be configured using adaptation/coding or data sets in the application, this DataIdentifier must not be writable since no PVD E2E security is required for the configuration in this case. In this case, all possible boot loader data sets must be contained in the response of the "0x0250-Integrity_validation_data_configuration_list" DataIdentifier.

[allg. Anf.: F-LAH_RxSWIN-539]

The content of the DataIdentifier "0x0250-Integrity_validation_data_configuration_list" must be persistently stored on the ECU.

[allg. Anf.: F-LAH_RxSWIN-551]

If the list of DataIdentifiers and data set numbers initially used in the "0x0250-Integrity_validation_data_configuration_list" DID deviates from the list of entries resulting from the production order, then the "0x0250-Integrity_validation_data_configuration_list" DID must be configured accordingly in the course of the ZDC process.

5.1.10 0x0251-Write_generic_to_sub_system (write the 0x0250 DataIdentifier to the lower-level diagnostic class 2/2F/2FV system via the diagnostic class 4-low/4V-low system)

[I: F-LAH_RxSWIN-494]

The DataIdentifier "0x0250-Integrity_validation_data_configuration_list" is written via the WriteDataByIdentifier (2Ehex) service and the DataIdentifier "0x0251-Write_generic_to_sub_system" via the diagnostic class 4-low system. To address the lower-level target system, the SSN is used in the first two bytes of the DataRecord in the 2Ehex request. The next two bytes contain the DataIdentifier 0x0250, followed by the payload.
Note: This is a change compared with Q-LAH 80124 up to version 2.8 (Q-LAH_80124-7984).

[allg. Anf.: F-LAH_RxSWIN-615]

Table 5-9: Structure of DataRecord for 0x0251-Write_generic_to_sub_system DataIdentifier

DID	**0x0251**
Designation	Write_generic_to_sub_system
Description	This DataIdentifier is used to write data for sub-systems via a diagnostic class 4-low system. It is not possible to read out data with this DataIdentifier. (write only)
Convention	Bus master
Diagnostic class	Diagnostic class 4/4-low
Session	APP: 0x03 (W) BLF: NoImp
SecurityLevel	ECUs on existing platforms: If present, the diagnostic class 4-low's existing access protection method must be used. ECUs on new platforms: PVD E2E (W)
Changing	DIAG
Format	<<[Target_sub_system_node_address] 2-byte hex> + <[Target_data_identifier] 2-byte hex> + <n x [Data_record_target_data_identifier] 1-byte hex>> (n: 1 – 256, variable)
Range	[Target_sub_system_node_address] 00 00 – FF FFhex [Target_data_identifier] 0x02 50 [Data_record_target_data_identifier] 00 – FFhex Other values are reserved by Volkswagen AG.
Init	No init on the server
Example	01 41 02 50 13 24 97 86 Target_sub_system_node_address: 0x0141 = TV tuner card reader Target_data_identifier: 0x02 50 Data_record_target_data_identifier: 0x13 24 97 86
Data category	Process parameter
Electronic build status documentation	NoImp

5.1.10.1 Request message definition

[I: F-LAH_RxSWIN-500]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-495]

Table 5-10: Request message definition

Data	Description		Cvt.	Value (Hex)
#1	Request SID	WriteDataByIdentifier	M	2E
#2	DataIdentifier#1	Write_generic_to_sub_system [Byte#1] MSB	M	02
#3	DataIdentifier#2	Write_generic_to_sub_system [Byte#2]	M	51
#4	DataRecord#1	Target_sub_system_node_address [Byte#1] MSB	M	00-FF
#5	DataRecord#2	Target_sub_system_node_address [Byte#2]	M	00-FF
#6	DataRecord#3	Target_data_identifier [Byte#1] MSB	M	00-FF
#7	DataRecord#4	Target_data_identifier [Byte#2]	M	00-FF
#8	DataRecord#5	Data_record_target_data_identifier#1	M	00-FF
:	:		:	:
#9+m-1	DataRecord#5+m-1	Data_record_target_data_identifier#m	U	00-FF

5.1.10.2 Request message parameter definition

[I: F-LAH_RxSWIN-497]

The following parameters must be implemented:

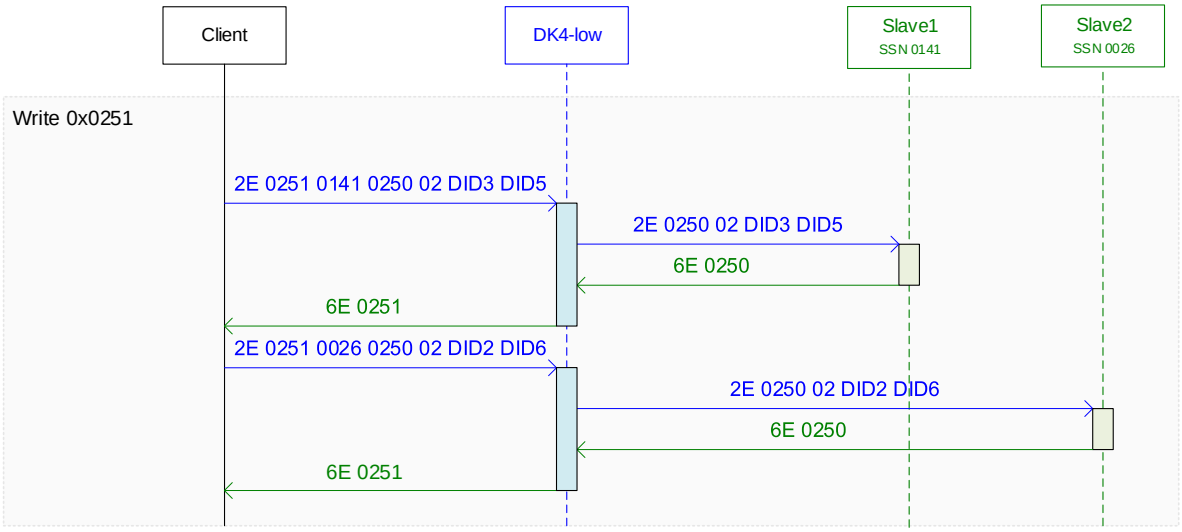
[allg. Anf.: F-LAH_RxSWIN-498]

Table 5-11: Request message parameter definition

Definition
Target_sub_system_node_address This parameter indicates the SSN of the lower-level target system.
Target_data_identifier This parameter indicates the value of the DataIdentifier. The permissible value here is 0x0250. Other values are reserved by Volkswagen AG.
Data_record_target_data_identifier This parameter indicates the DataRecord of the [Target_data_identifier] parameter.

[allg. Anf.: F-LAH_RxSWIN-475]

Figure 5-1: Writing 0x0250-Integrity_validation_data_configuration_list



5.2 RoutineIdentifiers

[allg. Anf.: F-LAH_RxSWIN-934]

Table 5-12: Overview of RoutineIdentifier properties

			DiagnosticSession		
			Default Session	NonDefaultSession	
				ECUProgram-mingSession	Extended-Diagnostic-Session
Diagnostic objects (RID)	**SID**	**Security Level**	**0x01**	**0x02**	**0x03**
0x03E7-Reset_to_factory_setting	31hex	NoImp	NoImp	NoImp	NoImp
0x0253-Calculate_integrity_validation_data	31hex	NoImp	Available	notAvaillable	Available
0x0254-Calculate_individual_hash_value	31hex	NoImp	Available	notAvaillable	Available

5.2.1 0x03E7-Reset_to_factory_setting

[I: F-LAH_RxSWIN-927]

As per document /1/, this RoutineIdentifier is used to reset all codings, vehicle, customer, workshop, and process parameters, as well as initial calibration and learned values. Running the routine leads to a change of the IVD-relevant parts of the configuration hash, such as codings, vehicle parameters, and initial calibration values.

[allg. Anf.: F-LAH_RxSWIN-928]

The following change compared with Q-LAH 80124 up to version 5.8 must be observed:

- Implementing RoutineIdentifier "0x03E7-Reset_to_factory_setting" on the diagnostic server is no longer permissible.

5.2.2 0x0253-Calculate_integrity_validation_data

[I: F-LAH_RxSWIN-451]

This RoutineIdentifier is used to calculate the integrity validation data of diagnostic class 3/3V/4/4V/SWCL systems for the instruction code and configuration data.

[allg. Anf.: F-LAH_RxSWIN-473]

This RoutineIdentifier must be implemented with the request/response behavior and parameters specified here. Use of sub-functions
• StopRoutine (0x02)
• RequestRoutineResults (0x03)
is not permissible for this RoutineIdentifier. It returns its result in the positive response to RoutineControlType 0x01 and terminates automatically.

[allg. Anf.: F-LAH_RxSWIN-597]

The response behavior described here complies with Q-LAH 80124, version 2.x. ECUs in existing projects that implemented a version of Q-LAH 80124 1.x must implement the response required here for the "0x0253-Calculate_integrity_validation_data" RoutineIdentifier. Existing routines based on Q-LAH 80124 version 1.x need not be adapted in the ECU.

[allg. Anf.: F-LAH_RxSWIN-616]

Table 5-13: Structure of RoutineIdentifier 0x0253-Calculate_integrity_validation_data

RID	**0x0253**
Designation	Calculate_integrity_validation_data
Description	This RoutineIdentifier is used to calculate the integrity validation data for instruction code and configuration data.
Convention	All servers
Diagnostic class	Diagnostic class 3/3V, diagnostic class 4/4-low/4V-low, diagnostic class 4-high/4V-high, SWCL
Session	APP: 0x01, 0x03 BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	See the request/response definition.
Range	See the request/response definition.
Init	No init on the server
Example	See the request/response definition.
Electronic build status documentation	NoImp

5.2.2.1 Request message definition

[I: F-LAH_RxSWIN-453]

The following request message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-454]

Table 5-14: Request message definition

Data	Description		Cvt.	Value (Hex)
#1	Request SID	RoutineControl	M	31
#2	RoutineControlType	StartRoutine	M	01
#3	RoutineIdentifier [Byte#1]	Calculate_integrity_validation_data [Byte#1] (MSB)	M	02
#4	RoutineIdentifier [Byte#2]	Calculate_integrity_validation_data [Byte#2]	M	53
#5	RoutineControlOption [Byte#1]	Type_of_calculation	M	00-FF
#6	RoutineControlOption [Byte#2]	Type_of_hashvalue	M	00-FF

5.2.2.2 Request message parameter definition

[I: F-LAH_RxSWIN-457]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-458]

Table 5-15: Request message parameter definition

Definition
Type_of_calculation This parameter indicates the hash calculation type. 0X00 = The hash is calculated for the configuration data (data). The details of the identifiers are taken from DataIdentifier 0x0250. 0X01 = The hash is calculated for the application and boot loader software programming data (instruction code). Other values are reserved by Volkswagen AG.
Type_of_hashvalue This parameter indicates the hash calculation method. 0x01 = SHA-256 Other values are reserved by Volkswagen AG.

5.2.2.3 Positive response message definition

[I: F-LAH_RxSWIN-460]

The following response message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-461]

Table 5-16: Positive response message definition

Data	Description		Cvt.	Value (Hex)
#1	Response SID	RoutineControl	M	71
#2	RoutineControlType	StartRoutine	M	01
#3	RoutineIdentifier [Byte#1]	Calculate_integrity_validation_data [Byte#1] (MSB)	M	02
#4	RoutineIdentifier [Byte#2]	Calculate_integrity_validation_data [Byte#2]	M	53
#5	RoutineStatusRecord#1	Result_of_calculation	M	00-FF
#6	RoutineStatusRecord#2	Typ_of_hashvalue	M	00-FF
#7	RoutineStatusRecord#3	Hashvalue [Byte#1] (MSB)	C1	00-FF
:	:	:	:	:
#6+n	RoutineStatusRecord#2+n	Hashvalue [Byte#n]	C1	00-FF

[allg. Anf.: F-LAH_RxSWIN-462]

C1 = The length depends on the [Type_of_calculation] parameter. The length is 32 bytes for 'SHA-256.'

5.2.2.4 Positive response message parameter definition

[I: F-LAH_RxSWIN-464]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-465]

Table 5-17: Positive response message parameter definition

Definition
Result_of_calculation 0x00 = Calculation_successful The hash was successfully calculated. 0x01 = Calculation_failed Calculation of the hash failed. 0x02 = Calculation_Identifier_not_found At least one DataIdentifier or data set number from DataIdentifier 0x0250 is not available on the diagnostic server. 0x03 – 0xFF = Reserved by Volkswagen AG
Type_of_hashvalue 0x00 = Reserved by Volkswagen AG 0x01 = SHA-256 The hash is calculated using the SHA-256 method. 0x02 – 0xFF = Reserved by Volkswagen AG
Hashvalue n bytes <0x00 – 0xFF> Calculated hash; this is n = 32 for SHA-256.

5.2.3 0x0254-Calculate_individual_hash_value

[I: F-LAH_RxSWIN-868]

This RoutineIdentifier is used to calculate and read out individual hashes for the configuration data from diagnostic class 3/3V/4/4V systems and SWCLs.

[allg. Anf.: F-LAH_RxSWIN-930]

This RoutineIdentifier must be implemented with the request/response behavior and parameters specified here. Use of sub-functions
• StopRoutine (0x02)
• RequestRoutineResults (0x03)
is not permissible for this RoutineIdentifier. It returns its result in the positive response to RoutineControlType 0x01 and terminates automatically.

[allg. Anf.: F-LAH_RxSWIN-929]

The response behavior described here complies with Q-LAH 80124, version 2.x. ECUs in existing projects that implement a version of Q-LAH 80124 1.x must implement the response required here for the "0x0254-Calculate_individual_hash_value" RoutineIdentifier. Existing routines based on Q-LAH 80124 version 1.x need not be adapted in the ECU.

[allg. Anf.: F-LAH_RxSWIN-871]

Table 5-18: Structure of RoutineIdentifier 0x0254-Calculate_individual_hash_value

RID	**0x0254**
Designation	Calculate_individual_hash_value
Description	This RoutineIdentifier is used to calculate and read out individual hashes for the configuration data.
Convention	All servers
Diagnostic class	Diagnostic class 3/3V, diagnostic class 4/4-low/4V-low, diagnostic class 4-high/4V-high, SWCL
Session	APP: 0x01, 0x03 BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	See the request/response definition.
Range	See the request/response definition.
Init	No init on the server
Example	See the request/response definition.
Electronic build status documentation	NoImp

5.2.3.1 Request message definition

[I: F-LAH_RxSWIN-873]

The following request message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-874]

Table 5-19: Request message definition

Data	Description		Cvt.	Value (Hex)
#1	Request SID	RoutineControl	M	31
#2	RoutineControlType	StartRoutine	M	01
#3	RoutineIdentifier [Byte#1]	Calculate_individual_hash_value [Byte#1] (MSB)	M	02
#4	RoutineIdentifier [Byte#2]	Calculate_individual_hash_value [Byte#2]	M	54
#5	RoutineControlOption [Byte#1]	Type_of_hashvalue	M	00-FF
#6	RoutineControlOption [Byte#2]	Type_of_hash	M	00-FF
#7	RoutineControlOption [Byte#3]	Individual_hash_value_id	M	00-FF
#8	RoutineControlOption [Byte#4]	Individual_hash_value_id	M	00-FF

5.2.3.2 Request message parameter definition

[I: F-LAH_RxSWIN-876]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-877]

Table 5-20: Request message parameter definition

Definition
Type_of_hashvalue 0x00 = Reserved by Volkswagen AG 0x01 = SHA-256 – The hash is calculated using the SHA-256 method. 0x02 – 0xFF = Reserved by Volkswagen AG
Type_of_hash 0x00 = Individual hash for data set (for 1st gen. data set download) 0x01 = Individual hash for boot loader data set (for 2nd gen. data set download) 0x02 = Individual hash for application data set (corresponds to application data set number of 2nd gen. data set download) 0x03 = Individual hash for all adaptations/codings as per DataIdentifier 0x0250 0x04 – 0xFE = Reserved by Volkswagen AG 0xFF = All individual hashes as per DataIdentifier 0x0250
Individual_hash_value_id This parameter identifies individual hashes for application data sets as per DataIdentifier 0x0250. 0x7200 – 0x72FF = Only used for [Type_of_hash] = 0x02 0x0000 = For all other values in the parameter [Type_of_hashvalue] All other values are reserved by Volkswagen AG.

5.2.3.3 Positive response message definition

[I: F-LAH_RxSWIN-879]

The following response message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-880]

Table 5-21: Positive response message definition

Data	Description		Cvt.	Value (Hex)
#1	Response SID	RoutineControl	M	71
#2	RoutineControlType	StartRoutine	M	01
#3	RoutineIdentifier [Byte#1]	Calculate_individual_hash_value [Byte#1] (MSB)	M	02
#4	RoutineIdentifier [Byte#2]	Calculate_individual_hash_value [Byte#2]	M	54
#5	RoutineStatusRecord#1	State_of_hashvalue	M	00-FF
#6	RoutineStatusRecord#2	Typ_of_hashvalue	M	00-FF
#7	RoutineStatusRecord#3	Typ_of_hash	M	00-FF
#8	RoutineStatusRecord#4	Individual_hash_value_id [Byte#1] (MSB)	M	00-FF
#9	RoutineStatusRecord#5	Individual_hash_value_id [Byte#2]	M	00-FF
#10	RoutineStatusRecord#6	Hash_value#1 [Byte#1] (MSB)	C1	00-FF
:	:	:	:	:
#9+n	RoutineStatusRecord#6+n	Hash_value#1 [Byte#n]	C1	00-FF
#10 + n	RoutineStatusRecord#6+n+1	Hash_value#m [Byte#1] (MSB)	C2	00-FF
:	:	:	:	:
#9+n + p	RoutineStatusRecord#6+n+p	Hash_value#m [Byte#p]	C2	00-FF

[allg. Anf.: F-LAH_RxSWIN-881]

C1 = Only present if an individual hash is available.

[allg. Anf.: F-LAH_RxSWIN-885]

C2 = Only present if the [Type_of_hash] parameter has a value of 0xFF and more than one hash is available.

5.2.3.4 Positive response message parameter definition

[I: F-LAH_RxSWIN-883]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-884]

Table 5-22: Positive response message parameter definition

Definition
State_of_hashvalue 0x00 = Calculation_hashvalue_valid The hash was successfully calculated; the hash is valid. 0x01 = Calculation_hashvalue_invalid Calculating the hash failed; the hash value is invalid. 0x02 = Calculation_hashvalue_not_found The identifier for the hash is invalid and is not available in DataIdentifier 0x0250 on the diagnostic server. 0x03 – 0xFF = Reserved by Volkswagen AG
Type_of_hashvalue 0x00 = Reserved by Volkswagen AG 0x01 = SHA-256 – The hash is calculated using the SHA-256 method. 0x02 – 0xFF = Reserved by Volkswagen AG
Type_of_hash 0x00 = Individual hash for data set (for 1st gen. data set download) 0x01 = Individual hash for boot loader data set (for 2nd gen. data set download) 0x02 = Individual hash for application data set (corresponds to application data set number of 2nd gen. data set download) 0x03 = Individual hash for all adaptations/codings as per DataIdentifier 0x0250 0x04 – 0xFE = Reserved by Volkswagen AG 0xFF = All individual hashes as per DataIdentifier 0x0250
Individual_hash_value_id This parameter identifies individual hashes for application data sets as per DataIdentifier 0x0250. 0x7200 – 0x72FF = Only used for [Type_of_hash] = 0x02 0x0000 = For all other values in the parameter [Type_of_hashvalue] All other values are reserved by Volkswagen AG.
Hash_value n bytes <0x00 – 0xFF> Computed hash; this is n = 32 for SHA-256.

6 RxSWIN-specific documentation

[I: F-LAH_RxSWIN-55]

Multiple "ECU versions" with their identification data and integrity validation data can be assigned to an RxSWIN. As a minimum, the following data must be documented for each RxSWIN on the IT systems.

6.1 Data for the RxSWIN-specific documentation of a diagnostic class 3/3V/4/4V system

[Prozess-Anf.: F-LAH_RxSWIN-149]

- Gateway ECUs only: Vehicle identification number from DataIdentifier 0xF190
- Gateway ECUs only: List of RxSWINs from DataIdentifier 0xF18F
- Volkswagen/Audi part number (VW Spare Part Number) from DataIdentifier 0xF187
- Software version (VW Application Software Version Number) from DataIdentifier 0xF189
- Hardware part number (VW ECU Hardware Number) from DataIdentifier 0xF191
- Optional: Hardware version (VW ECU Hardware Version Number) from DataIdentifier 0xF1A3
- Part number of target data container(s) (VW Data Set Number Or ECU Data Container Number)
- Version of target data container(s) (VW Data Set Version Number)
- FAZIT ID (FAZIT-Identification String) from DataIdentifier 0xF17C
- Integrity validation data of the programming (Programming_hash) from RoutineIdentifier 0x0253
- Integrity validation data of the configuration (Configuration_hash) from RoutineIdentifier 0x0253
- Netzwerk_Adress_Information (N_AI – network address information)
- Node address (NA)
- Diagnostic address (DA)

6.2 Data for the RxSWIN-specific documentation of an SWCL

[Prozess-Anf.: F-LAH_RxSWIN-428]

- Volkswagen/Audi part number (VW Spare Part Number) from DataIdentifier 0xF187
- Software version (VW Application Software Version Number) from DataIdentifier 0xF189
- Part number of target data container(s) (VW Data Set Number Or ECU Data Container Number)
- Version of target data container(s) (VW Data Set Version Number)
- Integrity validation data of the programming (Programming_hash) from RoutineIdentifier 0x0253
- Integrity validation data of the configuration (Configuration_hash) from RoutineIdentifier 0x0253
- Netzwerk_Adress_Information (N_AI – network address information)
- Diagnostic address (DA)
- Function ID (F-ID)

6.3 Data for the RxSWIN-specific documentation of a diagnostic class 2/2F/2FV system

[Prozess-Anf.: F-LAH_RxSWIN-89]

- Volkswagen/Audi part number (VW Spare Part Number) from DataIdentifier 0xF187 or 0x6200 – 63FF
- Hardware part number (VW ECU Hardware Number) from DataIdentifier 0xF191 or 6600 – 67FF
- Software version (VW Application Software Version Number) from DataIdentifier 0xF189 or 0x6400 – 65FF
- Part number of target data container(s) (VW Data Set Number Or ECU Data Container Number)
- Version of target data container(s) (VW Data Set Version Number)
- Optional: Hardware version (VW ECU Hardware Version Number) from DataIdentifier 0xF1A3 or 6800 – 69FF
- Optional: FAZIT ID (FAZIT-Identification String) from DataIdentifier 0xF17C or 6E00 – 6FFF
- For diagnostic class 2F/2FV only: Integrity validation data of the programming (Programming_hash) from DataIdentifier 0x0249 or 0xA800 – A9FF
- Integrity validation data of the configuration (Configuration_hash) from DataIdentifier 0x0245 or AA00 – ABFF
- Netzwerk_Adress_Information (N_AI – network address information) of the higher-level diagnostic class 4 system
- SubSystemNodeAddress (SSN)
- Diagnostic address (DA)

7 Applicable documents and specifications[1]

[I: F-LAH_RxSWIN-5]

/1/ LAH.DUM.909.G – Q-LAH 80124 – Unified Diagnostic Services Protocol (UDS), Emissions-Related Diagnostic Services (Legislated OBD)

[I: F-LAH_RxSWIN-48]

/2/ LAH.DUM.909.H – Q-LAH 80125 – Identification of Electronic Vehicle Systems

[I: F-LAH_RxSWIN-49]

/3/ LAH.DUM.909.B – Q-LAH 80127 – Diagnostics of Distributed Systems; Diagnostic Requirements for Bus Masters and Systems

[I: F-LAH_RxSWIN-69]

/4/ LAH.DUM.907.BD – Q-LAH for Protection of Vehicle Diagnostics (PVD)

[I: F-LAH_RxSWIN-75]

/5/ LAH.DUM.000.AD – Flash Data Security for UDS ECUs

[I: F-LAH_RxSWIN-76]

/6/ LAH.DUM.906.A – Q-LAH 80126 – UDS-Compliant Programming of ECUs

[I: F-LAH_RxSWIN-144]

/7/ LAH.DUM.907.R1 – Data Set Download, Generation 2

[I: F-LAH_RxSWIN-242]

/8/ LAH.DUM.907.BE – Q-LAH for Data Types

[I: F-LAH_RxSWIN-540]

/9/ Performance Specification for Diagnostic Filters

[I: F-LAH_RxSWIN-543]

/10/ Deleted – no description

[I: F-LAH_RxSWIN-596]

/11/ Data Set Download, Generation 1

[I: F-LAH_RxSWIN-648]

/12/ LAH.000.036.G – Q-LAH 80127ES – Supplementary Specification for Highly Integrated Systems

[I: F-LAH_RxSWIN-649]

/13/ LAH.000.036.H – Updating File-Based Systems – Supplementary Specification for Non-Embedded Systems

[I: F-LAH_RxSWIN-650]

/14/ LAH.DUM.906.B – Q-LAH 80128 Part 3 – Specification for Flash Containers, ODX-Flash/PDX-Flash

[I: F-LAH_RxSWIN-697]

/15/ LAH.000.900.AT – VKMS Core Functionality

[I: F-LAH_RxSWIN-702]

/16/ LAH.DUM.907.Q1 – SWAP Diagnostic Interface

[I: F-LAH_RxSWIN-703]

/17/ Features on Demand Performance Specification – Vehicle Requirements for Features on Demand

[I: F-LAH_RxSWIN-704]

/18/ LAH.DUM.907.xx – Immobilizer Master/Slave Performance Specification

[1] Please only refer to standard/document numbers since the document titles may vary.

[I: F-LAH_RxSWIN-705]

/19/ LAH.DUM.907.xx – Component Protection Master/Slave Performance Specification

[I: F-LAH_RxSWIN-706]

/20/ LAH.DUM.900.AB – Hardware Security Module