VOLKSWAGEN

AKTIENGESELLSCHAFT

CONFIDENTIAL
VERTRAULICH

UNECE Regulation on Software Updates – General Diagnostic Requirements

Supplementary Specification for Electronic Control Units

Development, General Project-Independent Performance Specification:
LAH.DUM.905.E

Author	Peter-Michael Hofmann (EESN/4), Dr. Markus Koch (EEY3), Arno von Querfurth (I/EE-871)
Dept./OU	EESN/4, I/EE-871, EEY3
Phone	+49 5361 9 78398, +49 711 911 83633, +49 841 89 716631
E-mail	peter.hofmann@volkswagen.de, markus.koch@porsche.de, arno.von-querfurth@audi.de

First issue	2019-11-22
Date of revision	2020-06-04
Performance Specification version	1.2
Baseline	4.2 ( )

The English translation is believed to be accurate. In case of discrepancies, the German version controls. The decimal sign in non-editable figures may be a comma or a point on the line. The decimal point is used consistently within the running text.

Contents

1 General information

1.1 Purpose

[I: F-LAH_RxSWIN-7]

This document describes technical requirements for onboard and offboard systems. UNECE software update regulations specified 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") must be complied with.

[I: F-LAH_RxSWIN-712]

All software for the implementation of vehicle functions must be developed in accordance with the Software Update Management System (SUMS). In addition, vehicle functions subject to obligatory type approval must be assigned a Regulation X Software Identification Number (RxSWIN). This applies, for example, to United Nations Economic Commission for Europe (UNECE) or GB/T regulations. These requirements are mandatory and must be met to receive type approval.

[I: F-LAH_RxSWIN-433]

The following requirements for diagnostics arise from the UNECE regulation on software updates:

[I: F-LAH_RxSWIN-732]

- Authenticated installation of software (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 or an alternative – at least equivalent – validation method and Protection of Vehicle Diagnostics (SFD)

  [I: F-LAH_RxSWIN-733]

- Introduction of an RxSWIN for vehicle functions subject to obligatory 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 data identifier (DID) in the gateway

  [I: F-LAH_RxSWIN-734]

- Introduction of a software integrity property (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]

As per the UNECE regulation on software updates, signatories to the 1958 treaty (market or country) may not perform software updates on vehicles without the technical requirements (e.g., RxSWIN, software versions) and without a SUMS certification. This renders type-approval testing for software updates and cyber security, and therefore the type approval of the vehicle, impossible.

[I: F-LAH_RxSWIN-430]

The purchaser assigns the vehicle projects to existing architectures, End-to-End Electronics 1.2 Group Architectures (E³ 1.2 architectures), or to new architectures.

[I: F-LAH_RxSWIN-432]

The following requirements apply to vehicle projects with type approval for software updates:

[I: F-LAH_RxSWIN-825]

For all programmable or configurable electronic control units (ECUs):

- ECU programming data security (EPDS) for software
- Protection of Vehicle Diagnostics (SFD end-to-end validation) for data
    - Diagnostic class (DK) 2 systems are an exception: Validation is provided by the higher-level DK4-low system.
- Integrity validation data for software and data

[I: F-LAH_RxSWIN-826]

Additionally for gateway ECUs:

- Implementation of a diagnostic filter with SFD access protection
- RxSWIN with SFD end-to-end (E2E) validation

1.2 Abbreviations and terms

[I: F-LAH_RxSWIN-21]

Table 1-1: Abbreviations and terms

Abbreviation or term	Designation	Explanation
BOM	Bill of materials	Build status documentation
BSB	Component engineer	Role designation at Audi AG, equivalent to "part owner (BTV)" 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 flash data security or 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 securing flash data
GB/T	GB stands for Guobiao, which is Chinese for "national standard."	This is the basis for the product test that the product must pass through in the course of CCC certification.
IVD	Integrity validation data	Hash values for the 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	Must not be implemented for servers commissioned by Volkswagen AG
RxSWIN	Regulation X Software Identification Number	Software identification number for each pertinent regulation
SFD	Protection of Vehicle Diagnostics	Cryptographic method for validating a) access to diagnostic objects (SFD access protection) and/or B) diagnostic contents (SFD E2E validation)
SUMS	Software Update Management System	Ensures compliance with legal requirements concerning the provisioning of software updates by way of specific manufacturer processes
SW	Software	In the terms of the UNECE regulation on software updates, software means the executable code and the data (instruction code and data).
U	User-optional	Cvt. = NoImp: Must not be implemented for servers commissioned by Volkswagen AG
UNECE	United Nations Economic Commission for Europe	United Nations Economic Commission for Europe
UNECE R SU	UNECE regulation on software updates	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
zGW	Central network gateway	ECU with central diagnostics access, e.g., gateway, In-Car Application Server ICAS1, high-performance computing platform HCP5
Reserved for Volkswagen AG		Use is reserved for future Volkswagen AG applications.

1.3 Scope

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

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

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

In the event of differing requirements between this document and the other General Project-Independent Performance Specifications (Q-LAHs) for Diagnostics, the requirements in this Performance Specification apply.

[I: F-LAH_RxSWIN-554]

The upcoming releases of the Diagnostics Q-LAH will merge the requirements from this Performance Specification with the requirements from the other documents.

[I: F-LAH_RxSWIN-521]

The color coding in the tables is intended for the reader’s convenience only.

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

The requirements marked with "Prozess-Anf." (process requirement) must not be taken into account in the course of ECU development.

2 Requirements for ECUs

2.1 ECUs in existing architectures and in the E³ 1.2 architecture

[I: F-LAH_RxSWIN-987]

The measures for ECUs in existing architectures and in the E³ 1.2 architecture must be specified in agreement with the purchaser's Safety and Security department and the Diagnostics department and documented in the Component Performance Specification (BT-LAH).

[allg. Anf.: F-LAH_RxSWIN-988]

The measures must be implemented according to the following sections in this document:

- New ECUs
- Requirements for validating vehicle diagnostics
- Integrity validation data
- Diagnostic objects

2.2 New ECUs

[I: F-LAH_RxSWIN-917]

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

	Central gateway	DK4(V)	DK3(V)	DK2F(V)	DK2(V)	DK1(V)/0
RxSWIN data identifier	X	NoImp	NoImp	NoImp	NoImp	NoImp
Diagnostic filter (OBD-port)	X	NoImp	NoImp	NoImp	NoImp	NoImp
EPDS	X	X	X[3)]	X	NoImp	NoImp
Protection of Vehicle Diagnostics (SFD) **- SFD authentication** **- SFD 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 configuration via ZDC is not possible in the application, then the use of SFD depends on the risk/security analysis.

[I: F-LAH_RxSWIN-923]

2) = Receives the data without a signature from DK4

[I: F-LAH_RxSWIN-943]

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

2.2.1 DK0/DK1 systems

[I: F-LAH_RxSWIN-709]

This document does not apply to DK0/DK1 systems since they are not calibratable.

2.2.2 DK2/DK2F/DK2FV systems

[I: F-LAH_RxSWIN-941]

This document does not apply to non-calibratable (neither programmable nor configurable) DK2/DK2F/DK2FV systems.

2.2.2.1 Programmable DK2F/DK2FV systems

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

Programmable DK2F/DK2FV systems must:

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

- support EPDS.

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

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

2.2.2.2 Configurable DK2/DK2F/DK2FV systems

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

Configurable DK2/DK2F/DK2FV systems

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

- are configured via ZDC.

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

- receive data without a signature from the DK4-low system. The DK4-low system performs the SFD E2E signature check for the DK2/DK2F/DK2FV system.

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

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

2.2.3 DK3/DK3V/DK4/DK4V systems and software clusters

[I: F-LAH_RxSWIN-942]

This document does not apply to non-calibratable (neither programmable not configurable) DK3/DK3V/DK4/DK4V systems and software clusters.

2.2.3.1 Programmable DK3/DK3V/DK4/DK4V systems and software clusters

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

Programmable DK3/DK3V/DK4/DK4V systems and software clusters must:

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

- support EPDS.

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

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

2.2.3.2 Configurable DK3/DK3V/DK4/DK4V systems and software clusters

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

Configurable DK3/DK3V/DK4/DK4V systems and software clusters

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

- are configured via ZDC.

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

- must support SFD E2E validation if they are configurable in the application. See section "SFD E2E validation."

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

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

2.2.3.3 Additional requirements for DK4-low/DK4V-low systems

2.2.3.3.1 Group data identifier

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

DK4-low/DK4V-low systems must:

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

- output the group data identifier for the integrity validation data of the DK2/DK2F/DK2FV system configuration. See section "Integrity validation data."

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

- output the group data identifier for the integrity validation data of the DK2F/DK2FV system programming. See section "Integrity validation data."

2.2.3.3.2 Routing mechanism in the DK4-low/DK4V-low system for a DK2F/DK2FV system

[I: F-LAH_RxSWIN-59]

The activation of the routing mechanism in the DK4-low/DK4V-low system is started as per document /3/ by the CommunicationControl (28hex) service using the [0x04-EnableRxAndDisableTxWithEnhancedAddressInformation] subfunction.

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

A DK4-low/DK4V-low system must protect the CommunicationControl (28hex) service for the [0x04-EnableRxAndDisableTxWithEnhancedAddressInformation] subfunction by way of SFD access protection if the DK4-low/DK4V-low system has lower-level physical DK2/DK2F systems.

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

SFD access protection must be implemented with the "Basic" role for the [0x04-EnableRxAndDisableTxWithEnhancedAddressInformation] subfunction.

[allg. Anf.: F-LAH_RxSWIN-989]

If the DK4-low/DK4V-low system has only virtual lower-level DK2V/DK2FV systems, SFD access protection for the CommunicationControl (28hex) service for the [0x04-EnableRxAndDisableTxWithEnhancedAddressInformation] must not be implemented.
Note: Only in this case may DK2FV systems use bootloader data sets.

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

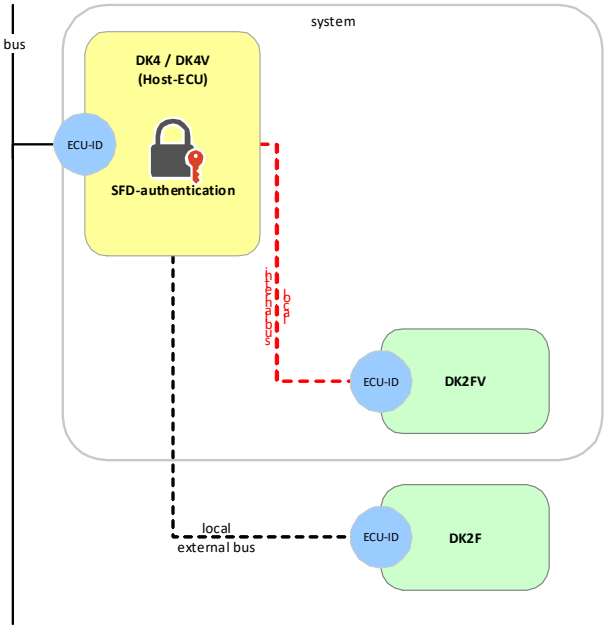
Contrary to requirement 80127-3047, the following applies up to v5.8:
It is explicitly prohibited for all DK2F systems to permanently implement diagnostic communication routing in the DK4-low system in normal operation.

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

If a DK4-low/DK4V-low system and a DK2F system are connected to the same bus segment, and this bus segment is used for diagnostic communication of both systems, 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: DK4-low routing activation, secured via SFD access protection in case of lower-level physical DK2/DK2F systems



2.2.3.4 Additional requirements for gateway ECUs

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

The following additional requirements apply to the ECU with central diagnostics access:

2.2.3.4.1 RxSWIN

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

The ECU with central diagnostics access must implement the data identifier "0xF18F-Regulation_x_software_identification_numbers" as per the section "Diagnostic objects."

2.2.3.4.2 "Diagnostic filter" function

[I: F-LAH_RxSWIN-625]

The diagnostic filter function is implemented, e.g., in a gateway ECU. It acts as a communication filter that prevents routing of diagnostic requests with write requests (coding, adaptation/calibration, data set download, programming). Additional requirements must be taken from document /9/.

2.3 Other ECUs

[I: F-LAH_RxSWIN-953]

ECUs in this category do not have to implement any additional requirements from this document.

3 Requirements for validating vehicle diagnostics

3.1 SFD E2E validation

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

Diagnostic servers that implement SFD E2E validation as per the section "Requirements for ECUs" must use document /4/ (Q-LAH for Protection of Vehicle Diagnostics, 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 using the WriteDataByIdentifier (2Ehex) service must be validated using SFD E2E validation; this must be documented accordingly in the diagnostic data tables of the BT-LAH:

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

- Coding (equipment-specific switching on/off of sub-functions)

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

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

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

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

[allg. Anf.: F-LAH_RxSWIN-991]

The following changes to document /4/ [allg. Anf.: Q-LAH_SFD_1752] must be taken into consideration:

- All adaptations and codings from the ZDC Mode K datum must be writable by a single signature check according to the process from document /4/ *[I: Q-LAH_SFD_1572]*. Deviations must be agreed upon with the appropriate department.

[I: F-LAH_RxSWIN-992]

The SFD E2E validation is not compatible with the data set download generation 1.

[allg. Anf.: F-LAH_RxSWIN-993]

An ECU with data set download generation 1 as per document /11/ must implement data set download generation 2 as per document /7/ when SFD E2E validation is used.

3.1.1 Special case for DK4-low/DK4V-low systems with lower-level DK2/DK2F/DK2FV systems

[I: F-LAH_RxSWIN-932]

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

[I: F-LAH_RxSWIN-101]

DK2/DK2F/DK2FV systems do not implement SFD with E2E validation themselves.

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

As per requirement "F-LAH_RxSWIN-341," a DK4-low/DK4V-low system must use SFD E2E validation to protect the configuration data of a lower-level DK2/DK2F/DK2FV system that is writable by the WriteDataByIdentifier (2Ehex) service against unauthorized access. Deviations must be agreed upon with the purchaser's Security department.

3.1.1.1 Writing SFD E2E validated data to a DK2/DK2F/DK2FV system

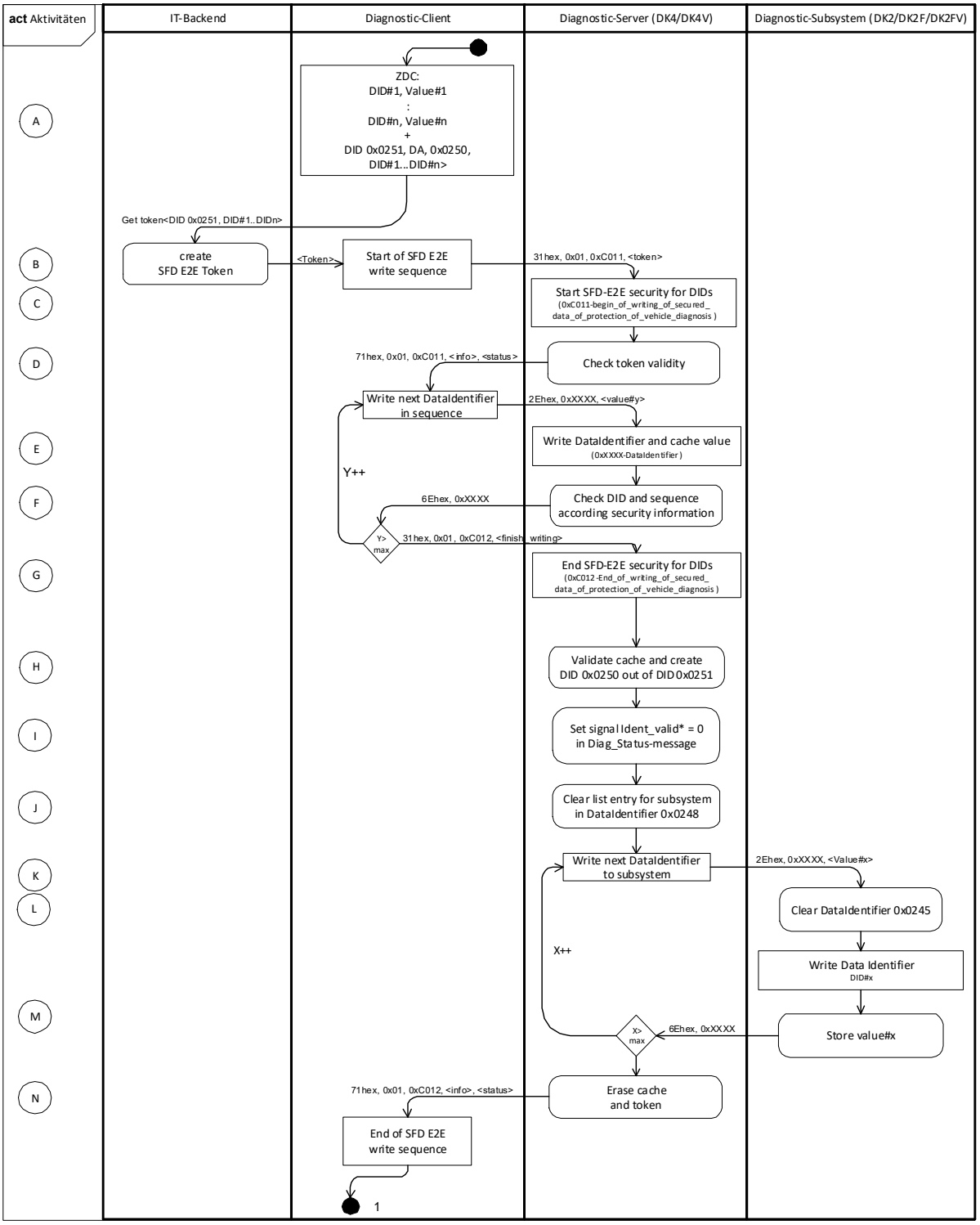
[I: F-LAH_RxSWIN-280]

A diagnostic client uses a DK4-low/DK4V-low system to write SFD E2E validated data to a lower-level DK2/DK2F/DK2FV system.

CONFIDENTIAL
VERTRAULICH

[I: F-LAH_RxSWIN-281]

Figure 3-1: Writing adaptations to a DK2/DK2F/DK2FV system – part 1/2

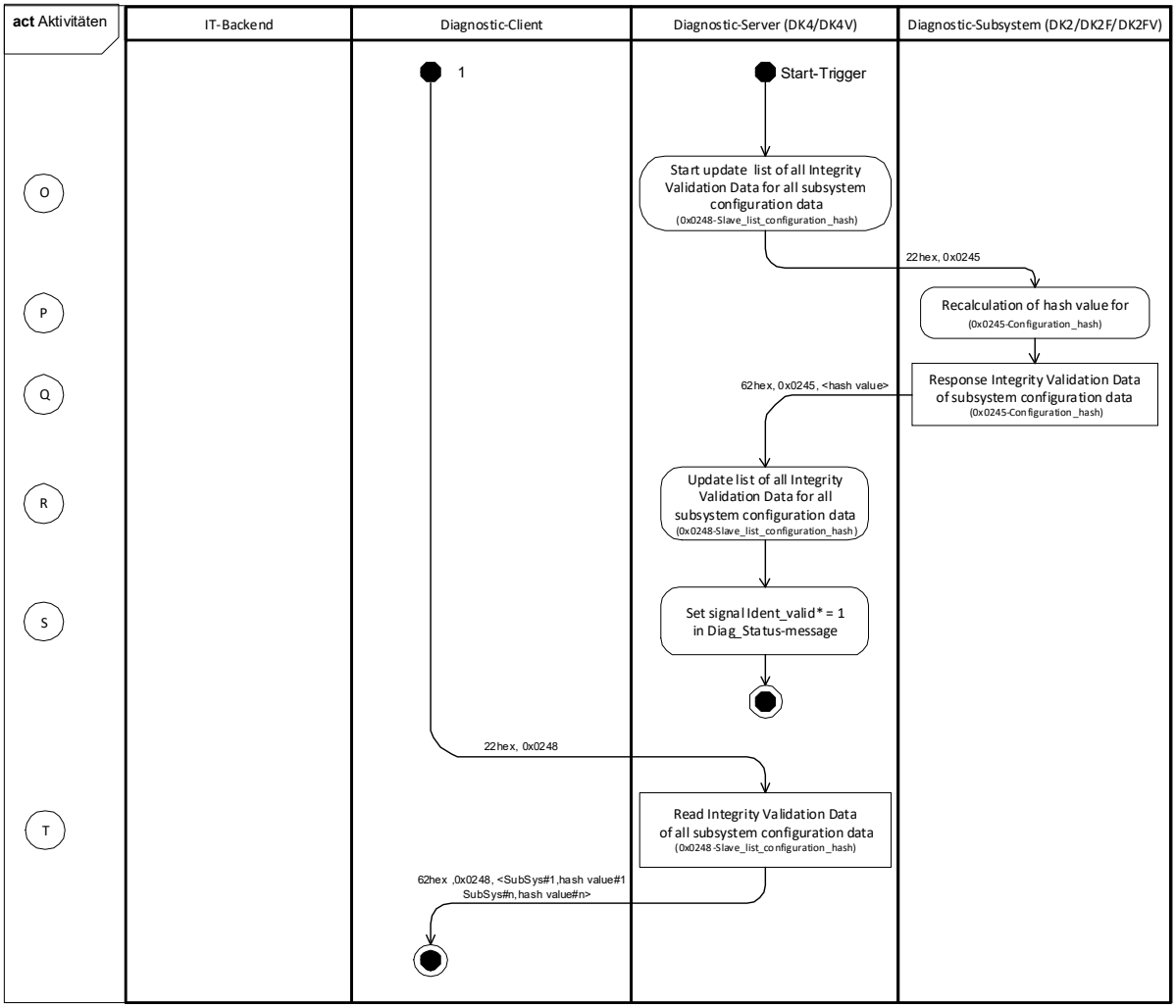


*Note: The signal Ident_valid (QLAH 80114 from version 5.6) has only informational character here and no requirement character.

act Aktivitäten	act activities

[I: F-LAH_RxSWIN-282]

Figure 3-2: Writing adaptations to a DK2/DK2F/DK2FV system – part 2/2



Note: The signal Ident_valid (QLAH 80114 from version 5.6) has only informational character here and no requirement character.

act Aktivitäten	act activities

[I: F-LAH_RxSWIN-283]

A – The data identifiers required for the ECU configuration are made available to the diagnostic client. The data identifiers result from the production order that configures the ZDC. This also results in the content of the "0x0250-Integrity_validation_data_configuration_list" data identifier for a specific sub-system. For diagnostic systems with SFD E2E validation upstream of checkpoint CP8, the diagnostic client requests a validation token for the diagnostic server from the SFD IT backend.

[I: F-LAH_RxSWIN-284]

B – The IT backend generates a validation token via the data identifiers 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 when the "0xC011-Begin_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis" SFD routine starts. This triggers the write sequence for the SFD E2E validated data.

[I: F-LAH_RxSWIN-286]

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

[I: F-LAH_RxSWIN-287]

E – All data identifiers from the configured ZDC and the "0x0251-Write_generic_to_sub_system" data identifier are transferred sequentially by the WriteDataByIdentifier (2Ehex) service to the diagnostic server, where they are cached. In case of SFD with group unlocking, the diagnostic server writes them directly to the diagnostic sub-systems.

[I: F-LAH_RxSWIN-288]

F* – In accordance with SFD E2E validation as per document /4/, the receive sequence of the individual data identifiers is checked using the validation information. If the check is passed, the diagnostic server replies with a positive response. If the check is failed, the received data identifiers and the associated data records as well as the validation information are deleted. If the check is failed, the diagnostic server responds with a negative response code (NRC) of "0x22-ConditionsNotCorrect".

[I: F-LAH_RxSWIN-289]

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

[I: F-LAH_RxSWIN-290]

H* – The authenticity of the transferred data identifier is checked based on the validation information from the validation token. The target sub-system address and the content of the "0x0250-Integrity_validation_data_configuration_list" data identifier are determined on the basis of the received "0x0251-Write_generic_to_sub_system" data identifier.

[I: F-LAH_RxSWIN-295]

I – A value of '0' in the [Ident_valid] signal of the diagnostic server's Diag_Status message indicates that the identification data is not currently valid. Saving the received data identifier means that the existing hash value for the sub-system configuration data is no longer valid and needs to be recalculated.

[I: F-LAH_RxSWIN-716]

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

[I: F-LAH_RxSWIN-293]

K* – The cached data identifiers are sequentially transmitted to the diagnostic sub-system using the WriteDataByIdentifier (2Ehex) service.

[I: F-LAH_RxSWIN-717]

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

[I: F-LAH_RxSWIN-291]

M* – The diagnostic sub-system checks the validity of the data received in the request (e.g., value range). If the check is passed, the data is saved in the sub-system. If the check is failed, 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 on the diagnostic server are erased after all received data identifiers have been transferred to the diagnostic sub-system's target memory.

[I: F-LAH_RxSWIN-297]

O* – The end of the SFD write sequence is the start trigger for the diagnostic server to update the "0x0248-Slave_list_configuration_hash" group data identifier by reading the diagnostic sub-system's "0x0245-Configuration_hash" data identifier. (For further information on the start trigger, see requirements F-LAH_RxSWIN-200 and F-LAH_RxSWIN-693.)

[I: F-LAH_RxSWIN-298]

P* – The request of the "0x0245-Configuration_hash" data identifier triggers the recalculation of the configuration data hash value. The recalculation includes precisely those data identifiers transferred in the "0x0250-Integrity_validation_data_configuration_list" in the exact order from the list.

[I: F-LAH_RxSWIN-299]

Q* – Until the recalculation of the hash value has been completed, the diagnostic sub-system responds to the bus master with an NRC of "0x78-RequestCorrectlyReceived-ResponsePending". Once the recalculation has been completed, a positive response is returned for the request.

[I: F-LAH_RxSWIN-300]

R* – The list entry for the diagnostic sub-system in the "0x0248-Slave_list_configuration_hash" group data identifier on the diagnostic server is updated on the basis of the "0x0245-Configuration_hash" data identifier.

[I: F-LAH_RxSWIN-301]

S* – A value of '1' in the [Ident_valid] signal of the diagnostic server's Diag_Status message indicates that the identification data is valid in its entirety.

[I: F-LAH_RxSWIN-695]

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

[I: F-LAH_RxSWIN-304]

*) Only relevant for diagnostic systems with SFD E2E validation. Not relevant for diagnostic systems with group unlocking in production, upstream of CP8.

3.2 ECU programming data security

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

The following applies to new architectures and the E³ 1.2 architecture:

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

- Diagnostic servers must use document /5/ (Q-LAH for Flash Data Security), version 3.1 or newer, or an alternative – at least equivalent – validation method 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 2.7 or newer.

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

- Diagnostic servers must use document /14/ (Q-LAH 80128-3), version 4.3 or newer.

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

The following applies to existing architectures:

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

- The contractor and the purchaser's E/E Security and Diagnostics departments must agree upon the version of document /5/ (Q-LAH for Flash Data Security), and thus the key length, to be used ("Documents and versions" table), or agree upon an alternative validation method.

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

- The contractor and the purchaser's E/E Security and Diagnostics departments must agree upon the version of document /6/ (Q-LAH 80126) to be used ("Documents and versions" table).

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

- The contractor and the purchaser's E/E Security and Diagnostics departments must agree upon the version of document /14/ (Q-LAH 80128-3) to be used ("Documents and versions" table).

4 Integrity validation data

[I: F-LAH_RxSWIN-786]

Integrity validation data is used:

[I: F-LAH_RxSWIN-831]

- as an integrity property of software and data

[I: F-LAH_RxSWIN-830]

- to verify the integrity of ECUs in production

[I: F-LAH_RxSWIN-829]

- to identify the configuration of ECUs via ZDC

4.1 General requirements

[I: F-LAH_RxSWIN-225]

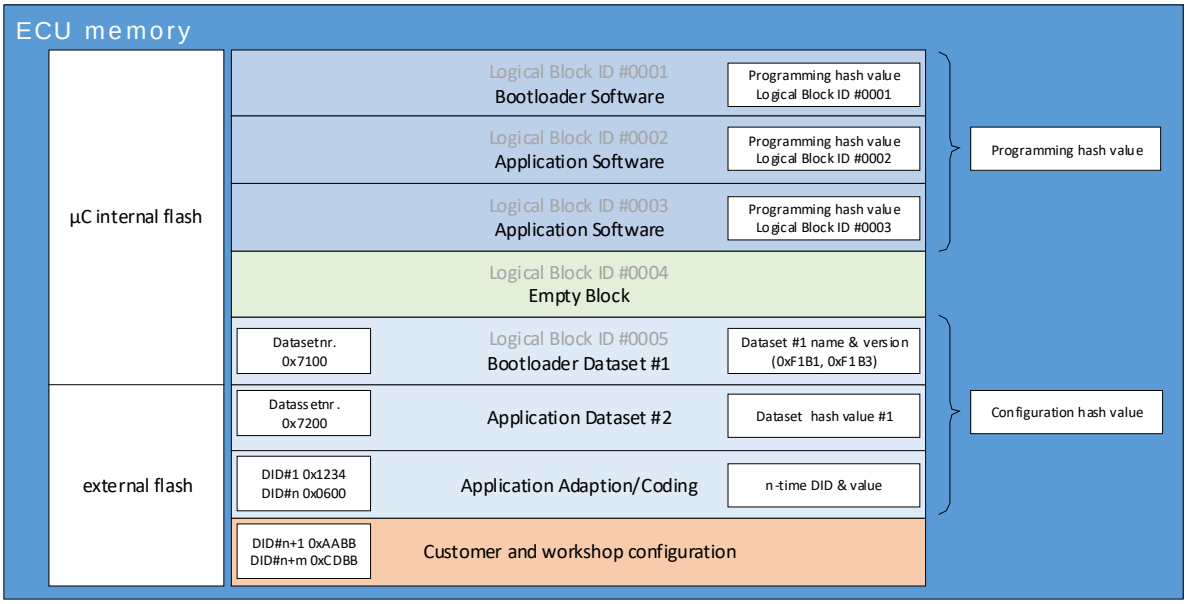
Separate hash values are calculated for instruction code (application and bootloader software) and configuration data (data identifiers and data sets from the ZDC). The hash values are calculated both on the IT systems and on the diagnostic server. This makes it possible to compare the desired state from the IT backend with the actual state on the diagnostic servers.

[I: F-LAH_RxSWIN-233]

The diagnostic server provides each of the hash values as integrity validation data. Depending on the diagnostic class, a data identifier or routine identifier (RID) is used for this purpose.

[I: F-LAH_RxSWIN-234]

Figure 4-1: Example of an ECU memory architecture



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

The SHA-256 Secure Hash Algorithm is used to calculate the hash values for programming and configuration.

[allg. Anf.: F-LAH_RxSWIN-1002]

The following applies to values used in the calculation of the hash value:

- The logical values, such as data identifiers, data set numbers, or checksum values, must be carried over as a hexadecimal byte array.
- The symbolic values or strings for identification, such as software version number or data set name, must use 7-bit ASCII encoding. In other words, 1 byte is used for each character in the ASCII value range. With regard to the character set (upper-case and lower-case letters, special characters, and numbers), ASCII strings as specified in documents /2/, /6/, and /14/ must be implemented.

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

It must take no longer than 800 ms to calculate the hash value. Deviations from this duration must be agreed upon with the Diagnostics department as well as with Production and After-Sales Service. Deviations must be documented in the component-specific diagnostics requirements of the BT-LAH.

[I: F-LAH_RxSWIN-1018]

The calculation of the hash value must be authentic. In other words, unauthorized manipulation of the instruction code for the routine for calculating the hash value must not be possible, or a manipulation of the code must be traceable. Such protection can be ensured by way of EPDS and secure boot or by way of installation in a controller-internal memory. The protection level may only be lowered in consultation with the Diagnostics department.

[I: F-LAH_RxSWIN-994]

Note: If the ECU cannot respond within 50 ms because the hash value is being calculated, it must return an NRC of "0x78-RequestCorrectlyReceived-ResponsePending".

[I: F-LAH_RxSWIN-940]

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

	Programming hash	**Configuration hash**
Programmdaten (Program data)	x	NoImp
Applikationsdaten (Calibration data)	x	NoImp
Codierung (Coding)	NoImp	x
Fahrzeugparameter (Vehicle parameters)	NoImp	x
Erstbedatungswerte (Initial calibration values)	NoImp	x
Kundenparameter (Customer parameters)	NoImp	NoImp
Werkstattparameter (Workshop parameters)	NoImp	NoImp
Prozessparameter (Process parameters)	NoImp	NoImp
Lernwerte (Learned values)	NoImp	NoImp
Analysedaten (Analysis data)	NoImp	NoImp

[I: F-LAH_RxSWIN-944]

X = included

[I: F-LAH_RxSWIN-981]

For the standardized identification data as per documents /2/ and /12/, the mapping of generic diagnostic objects to data categories must be requested from the Diagnostics department.

4.1.1 Programming hash value

4.1.1.1 Requirements for embedded systems

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

The requirements in 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 bootloader software.

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

The ECU calculates a single hash value for all logical blocks of the application and bootloader software in the memory of an ECU 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 AUTomotive Open System ARchitecture (AUTOSAR) or switch configurations), the default data, and the default data sets also count as part of the application or bootloader software.

[Prozess-Anf.: F-LAH_RxSWIN-1003]

Only data blocks from the flash container that are marked as [FLASH_DATA] type are used to calculate the hash value.

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

For ECUs that support a 2-stage boot loader update, only the logical blocks of the application are used in the calculation of the programming hash value.
Note: In the IT systems, only the flash PDX of the application software is used to calculate the programming hash value.

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

Empty logical software blocks that serve as a reserve must not be included when calculating the hash value.

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

Flash drivers, as logical blocks in RAM, must not be included when calculating the hash value.

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

Deviating from requirements *[A: 80126-A915]* and *[A: 80126-A914]* in document /6/, the optional non-empty blocks, e.g., for additional fonts, that do not directly belong to the application and do not influence the functional capability, must not be excepted from the compatibility check by the "0xFF01-Check Programming Dependencies" routine. They must be included when calculating the hash value.

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

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

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

Note: If a logical block contains data (e.g., supplier-specific information) that the supplier does not take into account when calculating the CRC32, then this data must also not be taken into account when calculating the CRC32 on the diagnostic server.

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

Only logical blocks marked with a valid software version number in the "0xF1AB-Logical Software Block Version" data identifier may be included when calculating the hash value with the "0x0253-Calculate_integrity_validation_data" routine with the [Type_of_calculation] = 0x01 control option.

[allg. Anf.: F-LAH_RxSWIN-1008]

Deviating from documents /6/ and /14/, the implementation of partial flashing is mandatory for all updatable ECUs (as per document /6/). The logical software block version must be contained in the <OWN-IDENTS> ... </OWN-IDENTS> XML tag of the respective block.

[allg. Anf.: F-LAH_RxSWIN-1009]

Deviating from requirement *[A: Q-LAH_80128_T3-435]* in document /14/, ECUs that support the integrity validation data may no longer use the value 0x30 30 30 30 (7/bit-ASCII characters = "0000") for the logical software block version number in the <OWN-IDENTS> ... </OWN-IDENTS> XML tag.

[allg. Anf.: F-LAH_RxSWIN-1010]

Deviating from requirements *[A: 80126-A936]* and *[I: Q-LAH_80128_T3-512]*, each in document /6/, the "0xF1AB-VW Logical Software Block Version" data identifier must not output the value 0x41 46 46 45 (7-bit ASCII characters = "AFFE") for logical software blocks. This ensures that the logical software block will not be skipped.

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

Immutable software components that cannot be updated by Volkswagen AG are not taken into account when calculating the programming hash value.

[I: F-LAH_RxSWIN-956]

If the software is stored redundantly, the redundantly stored portion of the software is not taken into account when calculating the programming hash value.

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

The "0x0253-Calculate_integrity_validation_data" routine with the [Type_of_calculation] = 0x01 control option recalculates the programming hash value on the ECU.

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

When starting a "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 to ensure that the logical block was transferred without any errors, the CRC32 checksum of the logical block must be recalculated.

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

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

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

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

[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)" and the "ExtendedSession (0X03)". In the bootloader, the "0x0544-Verify_partial_software_checksum" routine must be available in the "ExtendedSession (0X03)".

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

When the request for the "0x0544-Verify_partial_software_checksum" routine is received, the CRC32 checksum for the logical block addressed in the request must be recalculated and stored persistently.

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

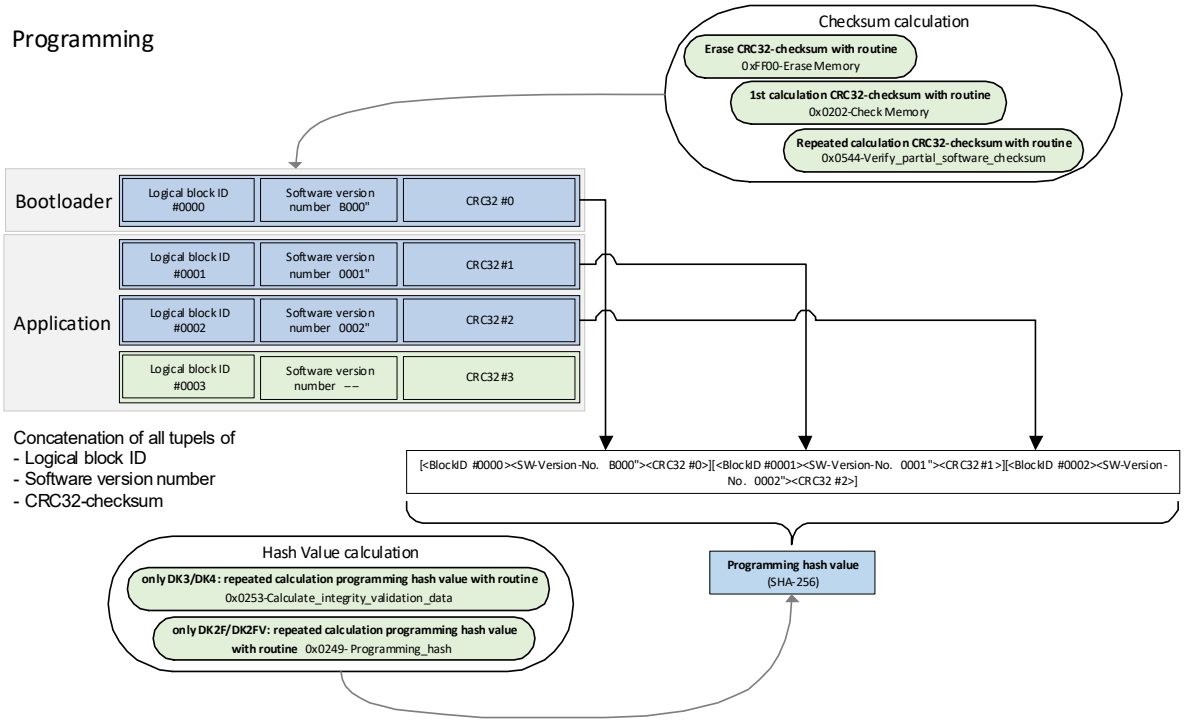
The programming hash value is calculated via the concatenated tuples consisting of the logical block ID or logical block number (2 bytes), the software version number for each logical software block (4 bytes, ASCII string), and the CRC32 checksum for each logical block (4 bytes each) as per document /6/ and as shown in figure 4-2.

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

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

[I: F-LAH_RxSWIN-235]

Figure 4-2: Concatenating the data of the logical blocks of the application and the bootloader (instruction code) to calculate the hash value



[I: F-LAH_RxSWIN-388]

Note:

- Logical block ID (length = 2 bytes) as per document /6/
- Programming hash value = hash value of the logical block of the application and bootloader software

[I: F-LAH_RxSWIN-999]

This process also applies for calculating and reading individual hash values using the "0x0254-Calculate_individual_hash_value" routine.

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

The desired programming hash value is calculated in the IT system using the same method as 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 documents /2/ and /12/ is taken into account when calculating the programming hash value under the following conditions (logical AND conjunction), if it:

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

- cannot be modified by the WriteDataByIdentifier (2Ehex) service or by the data set download (bootloader/application).

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

- can only be modified by update programming.

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

- is required according to the diagnostic class.

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

- is required by the OBD class.

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

- is required for specific use cases in accordance with the implementation requirements.

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

- is required in accordance with the system design as per document /12/.

[I: F-LAH_RxSWIN-660]

By way of example, the following data identifiers as per documents /2/ and /12/ are included in the programming hash value as a function of their OBD relevance, diagnostic class, implementation requirements, and the system design.

Only modifiable by update programming (DK2F and higher):
- 0xF187 - VW Spare Part Number
- 0xF189-VW 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

[I: F-LAH_RxSWIN-688]

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

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

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

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

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

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

[allg. Anf.: F-LAH_RxSWIN-1080]

Integrity validation data for programming (instruction code) always includes all application and bootloader software.

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

[allg. Anf.: F-LAH_RxSWIN-1119]

When flash containers as per document /14/ are used, the programming hash value for non-embedded or file-based systems can be calculated on a system-specific basis.

[allg. Anf.: F-LAH_RxSWIN-1081]

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

[Prozess-Anf.: F-LAH_RxSWIN-1082]

In the case of non-embedded or file-based systems that are programmed using Unified Diagnostic Services (UDS) as per document /6/, the hash procedure used and the desired programming hash value calculated by the supplier in the offline process must be documented in the ODX flash file and must be readable and analyzable by IT systems.

[Prozess-Anf.: F-LAH_RxSWIN-1083]

As a supplement to document /14/, the desired programming hash value calculated by the supplier in the offline process must be documented in the ODX flash file under the XPath "/ODX/FLASH/ECU-MEMS/ECU-MEM/MEM/SESSIONS/SESSION/SECURITYS/SECURITY[x]" in the "FW-CHECKSUM" data element.

[Prozess-Anf.: F-LAH_RxSWIN-1084]

As a supplement to document /14/, the hash procedure underlying the desired programming hash value must be documented in the ODX flash file under the XPath "/ODX/FLASH/ECU-MEMS/ECU-MEM/MEM/SESSIONS/SESSION/SECURITYS/SECURITY[x]" in the "SECURITY-METHOD" data element.

[Prozess-Anf.: F-LAH_RxSWIN-1085]

As a supplement to document /14/, the security method "IVD_PROGHASH_SHA256" with sub-element "FW-CHECKSUM" is requested under the following XPath:

- "/ODX/FLASH/ECU-MEMS/ECU-MEM/MEM/SESSIONS/SESSION/SECURITYS/SECURITY[x]"

[Prozess-Anf.: F-LAH_RxSWIN-1086]

Table 4-2: Security method

SECURITY-METHOD	Sub-element	Contents
IVD_PROGHASH_SHA256	FW-CHECKSUM	Programming hash value as per the SHA-256 algorithm

[I: F-LAH_RxSWIN-1087]

Exemplary security method "IVD_PROGHASH_SHA256:"

```
---------------------------------------------------------------------------------
<SECURITYS>
    <SECURITY>
        <SECURITY-METHOD TYPE="A_ASCIISTRING">IVD_PROGHASH_SHA256</SECURITY-METHOD>
        <FW-CHECKSUM TYPE="A_BYTEFIELD">8DA2…2C4B</FW-CHECKSUM>
    </SECURITY>
</SECURITYS>
---------------------------------------------------------------------------------
```

[Prozess-Anf.: F-LAH_RxSWIN-1112]

The timeout for executing the "0x029A-Calculate_module_hash_value" routine must be indicated with the following special data group (SDG) "IVD-Timeout."

[Prozess-Anf.: F-LAH_RxSWIN-1113]

If the "0x029A-Calculate_module_hash_value" routine is not completed within this time, the client must assume that an error exists, terminate the routine, and cancel the complete recalculation of the programming hash value.

[I: F-LAH_RxSWIN-1114]

Example for IVD-Timeout = 1 hour:

```
-------------------------------------------------------------------------------
<SDG>
    <SDG-CAPTION ID="EMEM_V04007001BC_0100_... USEKEEP_V001_E.SDGC_IVDTimeo">
        <SHORT-NAME>SDGC_IVDTimeo</SHORT-NAME>
        <LONG-NAME>IVD Timeout</LONG-NAME>
     </SDG-CAPTION>
     <SD SI="IVD-Timeout">3600</SD>
</SDG>
-------------------------------------------------------------------------------
```

[Prozess-Anf.: F-LAH_RxSWIN-1115]

The [IVD-Timeout] parameter must be specified in seconds.

[Prozess-Anf.: F-LAH_RxSWIN-1116]

If there is no SDG definition, the client must use the value of CP_RC78CompletionTimeout for the timer.

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

As an alternative, when using flash containers as per document /14/, the method for calculating the programming hash value as per the section "Requirements for embedded systems" may also be applied. In this case, the sub-element "FW-CHECKSUM" must not be used.

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

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

When using flash containers deviating from document /14/, the calculation can be system-specific.

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

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

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

In the case of non-embedded systems or file-based systems that are programmed via an update container as per document /21/, the desired programming hash value calculated by the supplier in the offline process must be documented via the index file of an update container as specified in document /21/ ("IdentificationData" data element) and must be readable and analyzable for IT systems.

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

Because the index file for the transmission to the IT systems is not cryptographically secured, the desired programming hash value calculated by the supplier in the offline process must additionally be stored in a cryptographically secured segment of the update container, as specified in document /21/.

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

An integrity property calculated in the vehicle must always deliver the same hash value for an ECU version, regardless of which ECU variants are installed in the vehicle. This hash value must correspond to the desired hash value calculated in the IT systems.

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

When accessing the integrity property, the diagnostic server must ensure the integrity (correctness) of the property's hash value. Ensuring the integrity means in this context that all modules associated with the integration property remain unchanged as long as the hash value remains unchanged. The algorithm for calculating a hash value for an integration property from data for individual modules must be agreed upon with the purchaser (Diagnostics department and Group IT).

[allg. Anf.: F-LAH_RxSWIN-1088]

In the case of modules with variants, the hash value for a module can be calculated using the values of the currently installed module variants and the values of the non-installed module variants, whereby:

[allg. Anf.: F-LAH_RxSWIN-1089]

- For installed module variants a hash value is calculated based on the actually programmed data, and for non-installed module variants an uploaded desired hash value is used.

[allg. Anf.: F-LAH_RxSWIN-1090]

- For modules that were not installed due to the ECU variant, the hash value can be calculated based exclusively on uploaded desired hash values for the non-installed module variants.

4.1.1.2.3 Recalculation the programming hash value

[allg. Anf.: F-LAH_RxSWIN-1092]

The following methods for determining input variables (hash values for partial modules) can be used for the hash value calculation for requesting the integrity property using the "0x0253-Calculate_integrity_validation_data" routine identifier on a DK3/DK3V/DK4/DK4V system or using the "0x0249-Programming_hash" data identifier on a DK2F/DK2FV system with a timing requirement of 800 ms as per [allg. Anf.: F-LAH_RxSWIN-707]:

[allg. Anf.: F-LAH_RxSWIN-1093]

- Calculation by determining the hash value using programmed data blocks (software)

[allg. Anf.: F-LAH_RxSWIN-1094]

- Use of precalculated hash values, provided the integrity is cryptographically secured and the hash value is checked at runtime when accessing/loading of the programmed data blocks (e.g., dm-verity with root hash check as part of a secure boot mechanism).

[allg. Anf.: F-LAH_RxSWIN-1095]

- Use of precalculated hash values if the hash value is automatically recalculated or checked at least once per driving cycle by the server.

4.1.1.2.3.1 DK3/DK4/DK3V/DK4V systems

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

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

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

- The has value could not yet be calculated from the validated data or checked against the data.
  In this case, the diagnostic server must not return a programming hash value. In the [Result_of_calculation] parameter, the value "0x04 = Calculation_incomplete" must be returned.

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

- A check result exists at runtime and the diagnostic server has completed the check of all packages (modules) to be checked.
  In this case, the diagnostic server returns the calculated overall programming hash value for all modules assigned to the integration property, regardless if this hash value corresponds to the desired value or not. In the [Result_of_calculation] parameter, the value "0x00 = Calculation_successfull" must be returned.

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

- If a check result or a hash value cannot be determined at runtime because this is prevented in the current system state (e.g., active control interventions), the value "0x01 = Calculation_failed" must be returned in the [Result_of_calculation] parameter.

4.1.1.2.4 DK2F/DK2FV systems

[allg. Anf.: F-LAH_RxSWIN-1099]

The "0x0249-Programming_hash" data identifier recalculates the programming hash value on the diagnostic server. The diagnostic server must distinguish between the following cases when doing so:

[allg. Anf.: F-LAH_RxSWIN-1100]

- The has value could not yet be calculated from the validated data or checked against the data.
  In this case, the diagnostic server must not return a programming hash value. The response to the ReadDataByIdentifier (22hex) request with a DID of "0x0249-Programming_hash" must be an NRC of "0x78-RequestCorrectlyReceived-ResponsePending".

[allg. Anf.: F-LAH_RxSWIN-1101]

- A check result exists at runtime and the diagnostic server has completed the check of all packages (modules) to be checked.
  In this case, the diagnostic server returns the calculated overall programming hash value for all modules assigned to the integration property in the positive response of the "0x0249-Programming_hash" data identifier, regardless if this hash value corresponds to the desired value or not.

[allg. Anf.: F-LAH_RxSWIN-1102]

- If a check result or a hash value cannot be determined at runtime because this is prevented in the current system state (e.g., active control interventions), the response to the ReadDataByIdentifier (22hex) request with a DID of "0x0249-Programming_hash" must be an NRC of "0x22-ConditionNotCorrect".

4.1.1.2.5 Complete recalculation of the programming hash value

[allg. Anf.: F-LAH_RxSWIN-1104]

The following applies to DK2F/DK2FV/DK3/DK3V/DK4/DK4V/software cluster systems:

- The complete recalculation of all hash values required for calculating the integrity property (programming hash value) is performed by the "0x029A-Calculate_module_hash_value" routine.

[allg. Anf.: F-LAH_RxSWIN-1106]

The "0x029A-Calculate_module_hash_value" routine does not need to meet the timing requirement of 800 ms (as per [allg. Anf.: F-LAH_RxSWIN-707]) until a positive response is returned.

[allg. Anf.: F-LAH_RxSWIN-1107]

For DK2F/DK2FV/DK3/DK3V/DK4/DK4V/software cluster systems, the "0x029A-Calculate_module_hash_value" routine performs a complete recalculation of the programming hash value on the diagnostic server using the [Type_of_calculation] = 0x01 routine control option. The diagnostic server must distinguish between the following cases when doing so:

[allg. Anf.: F-LAH_RxSWIN-1108]

- The complete recalculation of the hash value could not be performed.
  In this case, the diagnostic server must not return a programming hash value. In the [Result_of_calculation] response parameter, the value "0x04 = Calculation_incomplete" must be returned.

[allg. Anf.: F-LAH_RxSWIN-1109]

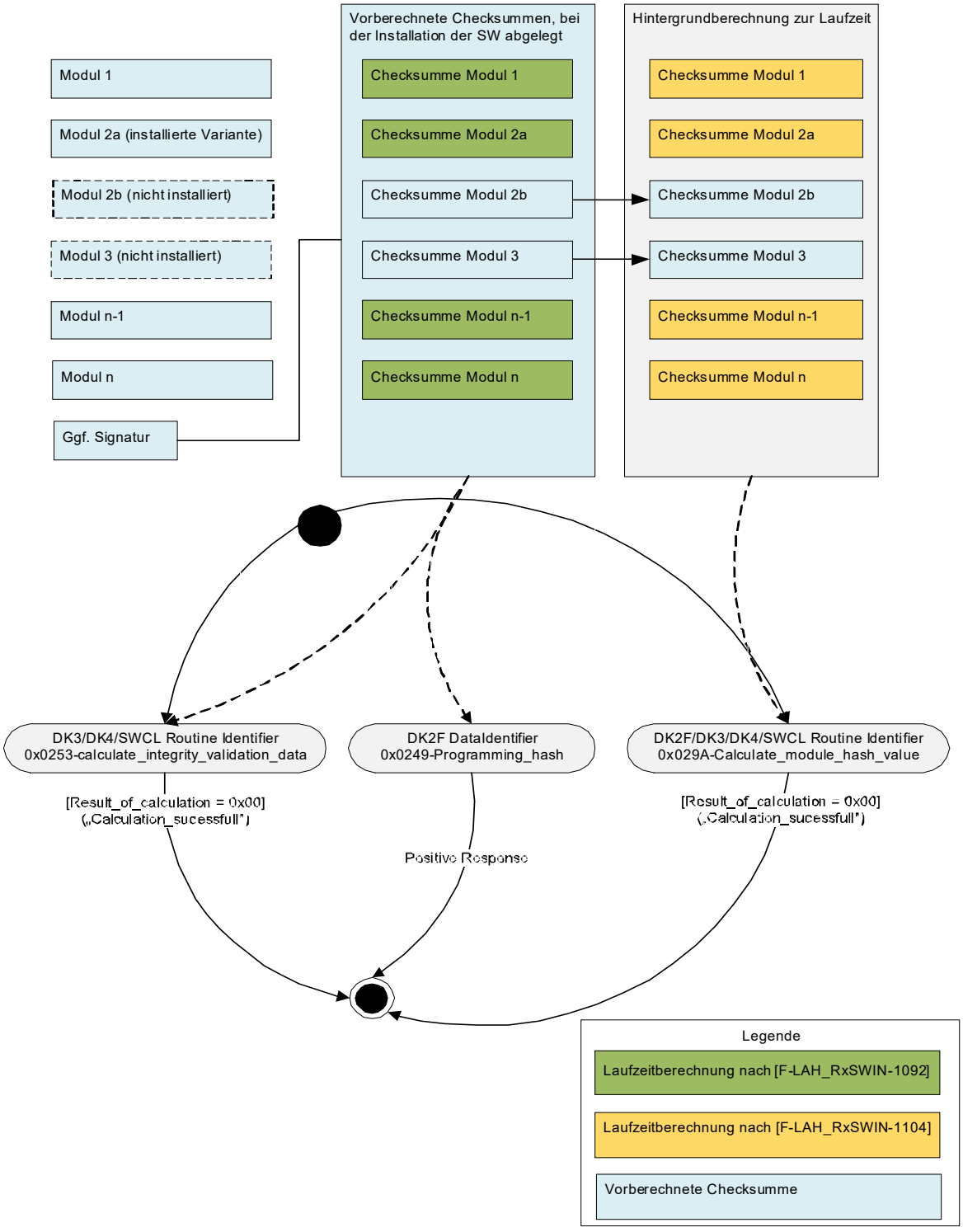
- The complete recalculation of the hash value could be completed.
  In this case, the diagnostic server returns the calculated overall programming hash value for all modules assigned to the integration property, regardless if this hash value corresponds to the desired value or not. In the [Result_of_calculation] parameter, the value "0x00 = Calculation_successfull" must be returned.

[allg. Anf.: F-LAH_RxSWIN-1110]

- If the complete recalculation of the hash value could not be performed because this is prevented in the current system state (e.g., active control interventions), the value "0x01 = Calculation_failed" must be returned in the [Result_of_calculation] parameter.

[I: F-LAH_RxSWIN-924]

Figure 4-3: Hash value calculation for LUM flash containers



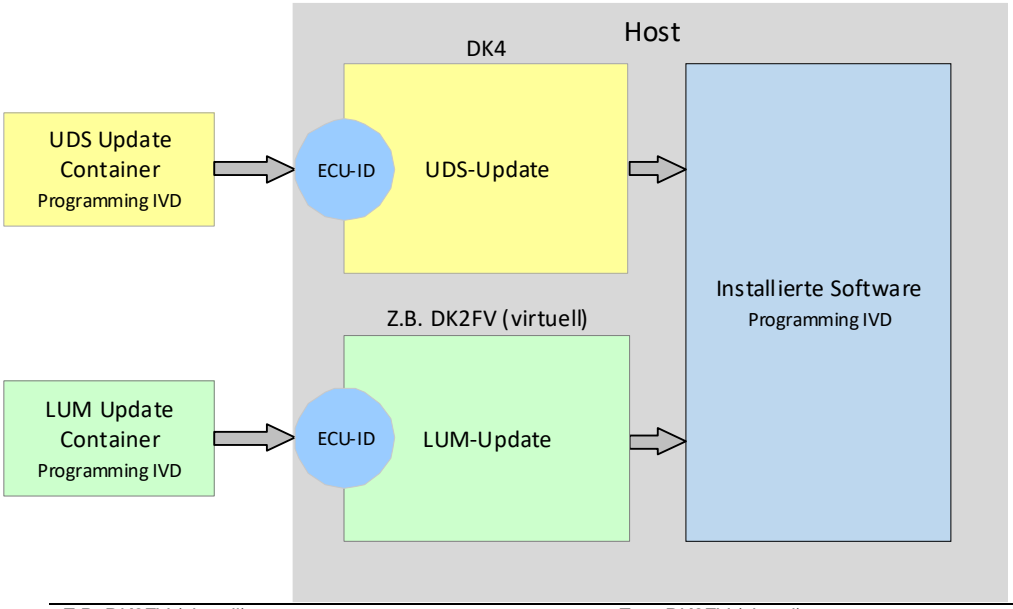
Modul 1	Module 1
Modul 2a (installierte Variante)	Module 2a (installed variant)
Modul 2b (nicht installiert) Checksumme Modul 2b	Module 2b (not installed) Module 2b checksum

Modul 3 (nicht installiert)	Module 3 (not installed)
Modul n-1	Module n-1
Modul n	Module n
Ggf. Signatur	Signature, if applicable
Vorberechnete Checksummen, bei der Installation der SW abgelegt	Precalculated checksums; stored during software installation
Checksumme Modul 1	Module 1 checksum
Checksumme Modul 2a	Module 2a checksum
Checksumme Modul 2b	Module 2b checksum
Checksumme Modul 3	Module 3 checksum
Checksumme Modul n-1	Module n-1 checksum
Checksumme Modul n	Module n checksum
Hintergrundberechnung zur Laufzeit	Background calculation at runtime
Legende	Legend
Laufzeitberechnung nach [F-LAH_RxSWIN-1092]	Runtime calculation as per [F-LAH_RxSWIN-1092]
Laufzeitberechnung nach [F-LAH_RxSWIN-1104]	Runtime calculation as per [F-LAH_RxSWIN-1104]
Vorberechnete Checksumme	Precalculated checksum

4.1.1.2.6 Special case for UDS and Local Update Manager (LUM) flashing of an ECU

[I: F-LAH_RxSWIN-1075]

Figure 4-4: Exemplary representation of a special case for ECUs with UDS and LUM update



Z.B. DK2FV (virtuell)	E.g., DK2FV (virtual)
Installierte Software Programming IVD	Installed software Programming IVD

[allg. Anf.: F-LAH_RxSWIN-957]

The following applies if an ECU implements both UDS update programming as per document /6/ and the method using the LUM:

[allg. Anf.: F-LAH_RxSWIN-958]

- A diagnostic address may only be programmed using one of the following methods:
  - UDS update programming or
  - LUM.

[allg. Anf.: F-LAH_RxSWIN-959]

- The programming hash value is always output via both diagnostic addresses, regardless which diagnostic address and which method was used to perform the last update.

[allg. Anf.: F-LAH_RxSWIN-1068]

- For both update methods and on both diagnostic addresses, the hash value is calculated based on the same algorithm and with the same hash procedure. In other words, the same programming hash value must be calculated for both diagnostic addresses.

[allg. Anf.: F-LAH_RxSWIN-960]

Configuration by means of ZDC must only be possible using the diagnostic address that supports update programming as per document /6/.

[allg. Anf.: F-LAH_RxSWIN-1069]

Performing partial software updates or distributing parts of an ECU update over multiple diagnostic addresses is not permissible. A complete software update using the respective method must be performed using each of the two diagnostic address.

[allg. Anf.: F-LAH_RxSWIN-1070]

In a server that can be programmed via two diagnostic addresses, the current identification data (logistics data as per document /6/) must be available in both diagnostic addresses at the time of system restart under all conditions after a software update and regardless of the programming method.

[allg. Anf.: F-LAH_RxSWIN-1071]

A server that can be programmed via two diagnostic addresses must supply the same information for the identification data, from physically identical memory addresses, regardless of the current operating mode of the server (application or bootloader) and regardless of the diagnostic address.

[allg. Anf.: F-LAH_RxSWIN-1072]

A server that can be programmed via two diagnostic addresses must, under all circumstances, be able to supply the identification data (logistics data as per document /6/) of the last "validly" programmed software from a separate backup (non-volatile memory) on both diagnostic addresses.

[allg. Anf.: F-LAH_RxSWIN-1073]

Suitable measures must be implemented to prevent that competing diagnostic services, e.g., write access via WriteDataByIdentifer 2Ehex, or competing sequences, e.g., software updates, are executed simultaneously via both diagnostic addresses.
Note: The objective is to ensure that sequences, in particular for the software updates are not affected by the diagnostic services of the respective other diagnostic address.

[allg. Anf.: F-LAH_RxSWIN-1074]

The competing diagnostic services that cannot be executed must be rejected with an NRC of "0x22-ConditionNotCorrect".

4.1.2 Configuration hash value

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

The requirements in 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 configuration data hash value includes logical blocks with bootloader data sets and application configuration data.

[I: F-LAH_RxSWIN-1016]

The order of the data identifiers and the data set numbers for calculating the configuration hash value (IVD) is independent of the order used during the SFD E2E validation of the configuration data.

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

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

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

- Coding (equipment-specific switching on/off of sub-functions)

[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 categories (data types) as per document /8/ must not be included when calculating the hash value for the application configuration data:

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

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

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

- Application data (equipment-specific data and characteristic curves)
  Note: Application data is part of the programming hash value.

[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 specified framework, without influencing the type approval.

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

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

[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 data identifiers must not be included when calculating the hash value for the application configuration data:

- 0xF18F-Regulation_x_software_identification_numbers

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

Note: The data identifier 0xF18F is not part of the gateway ECU's ZDC. This means that it cannot be included when calculating the desired value for the configuration hash value. It is provided via a separate data container and documented fully in the build status documentation.

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

A change of the customer parameters, workshop parameters, process parameters, learned values, and/or analysis data must NOT cause the integrity validation data of the configuration data to change.

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

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

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

Data sets that are planned to be used for calculating the hash values as per document /7/ are included when calculating the configuration data hash value; they must be referenced in the ZDC.
Note: The requirement for a valid software version number in the data identifier "0xF1AB-VW Logical Software Block Version" does not apply to reserve data sets.

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

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

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

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

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

If, for example, data relevant to Automotive Safety Integrity Levels (ASILs) is stored redundantly, the redundantly stored portion of the data is not taken into account when calculating the configuration hash value.

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

The standardized identification data as per documents /2/ and /12/ must be taken into account when calculating the configuration hash value under the following conditions (logical AND conjunction) if it:

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

- can be modified by the WriteDataByIdentifier (2Ehex) service or the data set download (bootloader/application).

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

- is required according to the diagnostic class.

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

- is required by the OBD class.

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

- is required for specific use cases in accordance with the implementation requirements.

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

- is required in accordance with the system design as per document /12/.

[I: F-LAH_RxSWIN-651]

By way of example, the following data identifiers as per documents /2/ and /12/ are assigned to the configuration hash value as a function of their diagnostic class, implementation requirements, and the system design.

For bus master systems in DK4-high only:
- 0x04A3-Gateway Component List
- 0x061A-Slave_component_list

Only for systems that support Data Set Download Generation 2:
- 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 configuration data that is included when calculating the configuration data hash value must be part of the ZDC.

[allg. Anf.: F-LAH_RxSWIN-1017]

Configuration data must only be changeable by one method (e.g., adaptation, data set) in an ECU. This also applies to ECUs with integrated units, e.g., virtual systems or software clusters.

[I: F-LAH_RxSWIN-599]

Note: The ZDC must contain all configuration data so that after-sales service has the option in the field of restoring all configuration data according to the type approval if the hash value on the ECU does not match the desired hash value in the IT system.

[allg. Anf.: F-LAH_RxSWIN-961]

The ZDC of the higher-/lower-level system must not contain the configuration data of an ECU/system. For this reason, DK2 systems and DK4 systems must contain their own individual ZDC each.
Note: This is necessary to be able to calculate the configuration hash value in the IT system.

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

Data from the immobilizer, component protection, and the Vehicle Key Management System (VKMS), as well as data from software as a product (SWAP) and features on demand must not be included when calculating the configuration data hash value.

[I: F-LAH_RxSWIN-624]

Note: Data from the immobilizer as per document /18/, component protection as per document /19/, and the VKMS as per document /15/, as well as data from SWAP as per document /16/ or features on demand as per document /17/ are uploaded to the vehicle using a validation method, and stored in secure memory areas (Secure Hardware Extension (SHE) or hardware security module (HSM)) as per document /20/ to rule out the possibility of data manipulation.

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

The data identifier "0x0250-Integrity_validation_data_configuration_list" is used to specify the list of data identifiers for adaptation/coding and the data set numbers that must be used for calculating the configuration data hash value.

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

The order of the data for calculating the hash value results from the content of data identifier "0x0250-Integrity_validation_data_configuration_list". This order must be adopted without any changes.

4.1.2.1 Calculating the individual configuration data hash values and checksums

[I: F-LAH_RxSWIN-690]

The configuration data hash value is determined with the aid of various individual hash values. These individual hash values can be read using the "0x0254-Calculate_individual_hash_value" routine identifier.

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

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

The individual hash values for adaptations/coding and the individual hash values for application data sets of the configuration data are initially calculated with the positive response for the "0xC012-End_of_writing_secured_data_of_protection_of_vehicle_diagnosis" SFD routine.

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

The individual hash value for all adaptations/coding and the individual hash values for application data sets of the configuration data are recalculated upon receiving the request for the "0x0F02-Calculate_configuration_state_fingerprint_of_protection_of_vehicle_diagnosis" SFD routine.

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

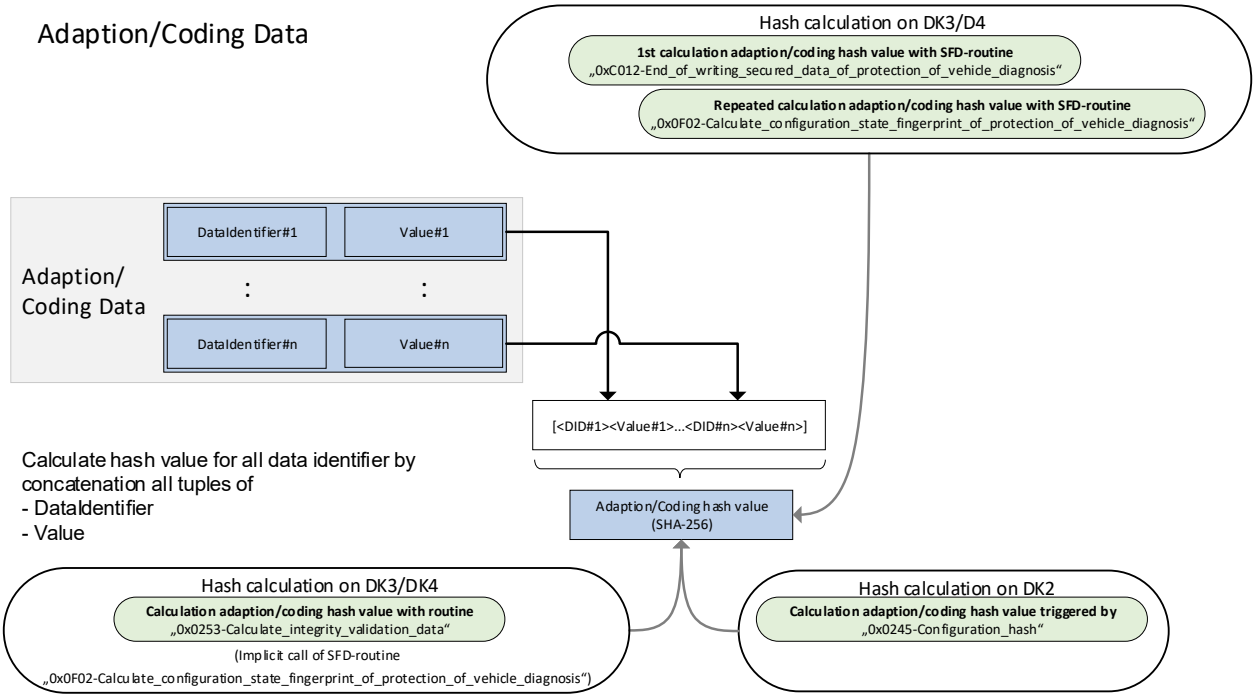
ECUs that do not implement SFD E2E calculate the individual hash value for all adaptations/coding and the individual hash values for application data sets as per document /7/ upon receiving the request for the "0x0253-Calculate_integrity_validation_data" routine with the ControlOption [Type_of_calculation] = 0x00.

[allg. Anf.: F-LAH_RxSWIN-1004]

The individual hash value for adaptations/coding is calculated from the concatenation of all tuples, consisting of data identifiers followed by the respective data record (values or byte arrays).

[I: F-LAH_RxSWIN-578]

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



[I: F-LAH_RxSWIN-995]

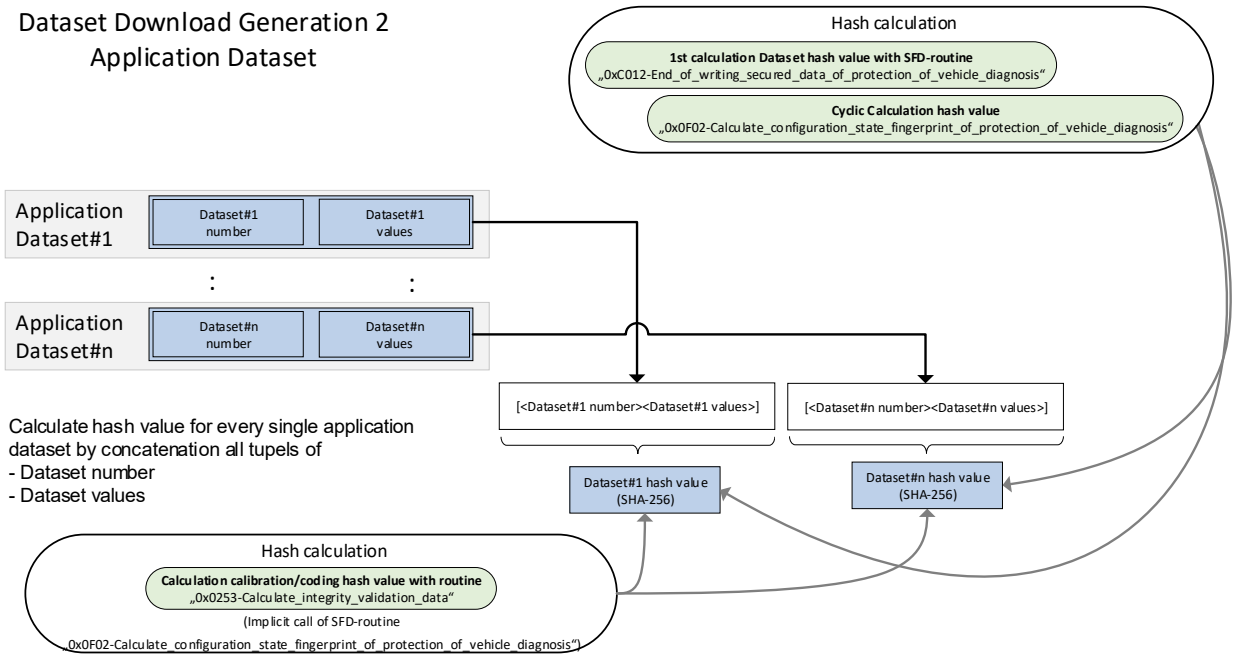
This process also applies for calculating and reading individual hash values using the "0x0254-Calculate_individual_hash_value" routine.

[allg. Anf.: F-LAH_RxSWIN-1005]

The individual hash value for an application data set is calculated using the concatenated tuple, consisting of the data set number and the respective data record (values or byte array).

[I: F-LAH_RxSWIN-585]

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



[I: F-LAH_RxSWIN-996]

This process also applies for calculating and reading individual hash values using the "0x0254-Calculate_individual_hash_value" routine.

[I: F-LAH_RxSWIN-1000]

All programming data from the data set container between the tags <DATEN> and </DATEN> are considered dataset #n values.

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

[allg. Anf.: F-LAH_RxSWIN-1121]

Deviating from document /11/, during a data set download generation 1, the logical addresses (start address) must not overlap with the data identifiers of used adaptations.

[allg. Anf.: F-LAH_RxSWIN-1122]

Deviating from document /11/, the length of the logical address (start address) is specified at 2 bytes.

[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]

The ECU additionally calculates and stores a CRC32 checksum of the entire data set container content, including the data, checksum, and memory address, for each valid data set.

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

The CRC32 checksum for a data set is erased 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 initially calculated on receiving the request for the "0x02EF-Calculate checksum" routine.

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

The CRC32 checksum for a data set is persistently stored with the positive response for the "0x02EF-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 startup.

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

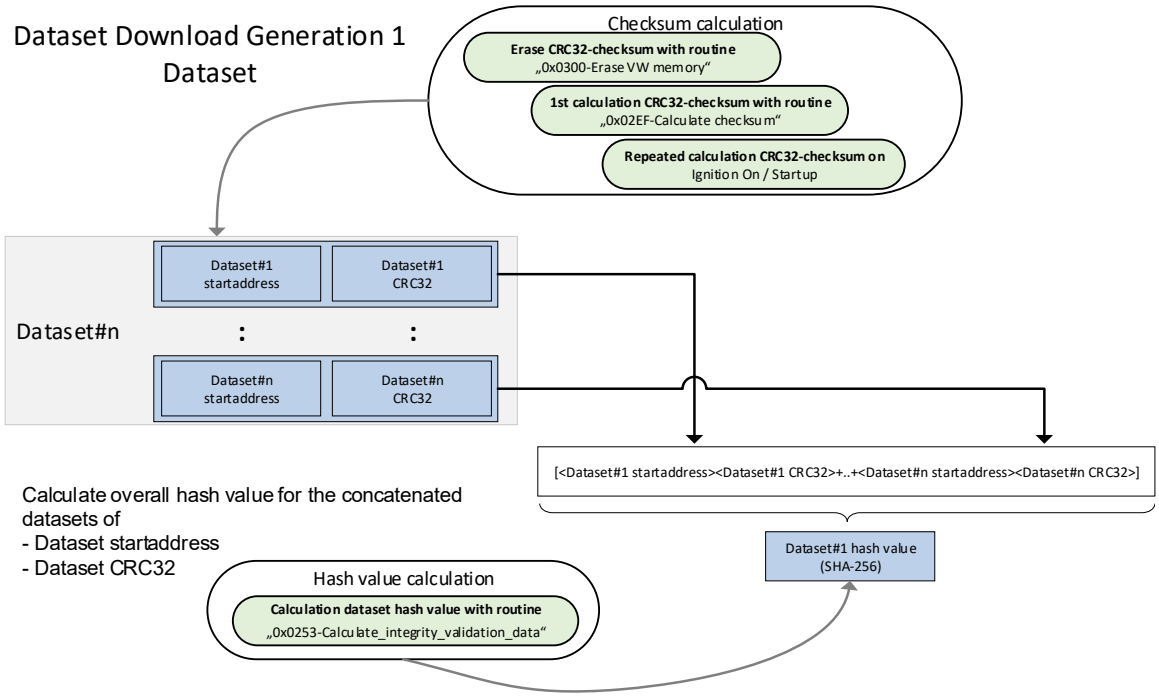
The individual hash value for a data set is calculated on receiving the request for the "0x0253-Calculate_integrity_validation_data" routine with the control option [Type_of_calculation] = 0x00.

[allg. Anf.: F-LAH_RxSWIN-1120]

An individual hash value for all data sets is calculated from the concatenated tuples of all data sets, consisting of the data set address and the CRC32 checksum as per document /11/.

[I: F-LAH_RxSWIN-583]

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



[I: F-LAH_RxSWIN-997]

This process also applies for calculating and reading individual hash values using the "0x0254-Calculate_individual_hash_value" routine.

4.1.2.1.3 Calculating the CRC32 checksum for bootloader data sets from the data set download generation 2 as per document /7/

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

The CRC32 checksum of a bootloader data set is erased on receiving the request for a "0xFF00-Erase Memory" routine that addresses the bootloader data set.

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

A "0xFF00-Erase Memory" routine that addresses a logical block with the bootloader data set must not influence the programming hash value.

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

The CRC32 checksum for a bootloader data set is initially calculated on receiving the request for the "0x0202-Check Memory" routine.

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

The CRC32 checksum for a bootloader data set is persistently stored with the positive response for the "0x0202-Check Memory" routine.

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

The CRC32 checksum for a bootloader data set is recalculated and persistently stored on each change to terminal 15 ON or startup.

[allg. Anf.: F-LAH_RxSWIN-1001]

ECUs must use the CRC32 algorithm as per document /6/ for calculating the checksum of bootloader data sets (for startup and for the "0x0202-Check Memory" routine).
Note: This means that the calculation in the ECU is identical to the calculation on the offboard side, e.g., in the IT systems.

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

The individual hash value for the bootloader data sets is calculated on receiving the request for the "0x0253-Calculate_integrity_validation_data" routing with the control option [Type_of_calculation] = 0x00.

[allg. Anf.: F-LAH_RxSWIN-1006]

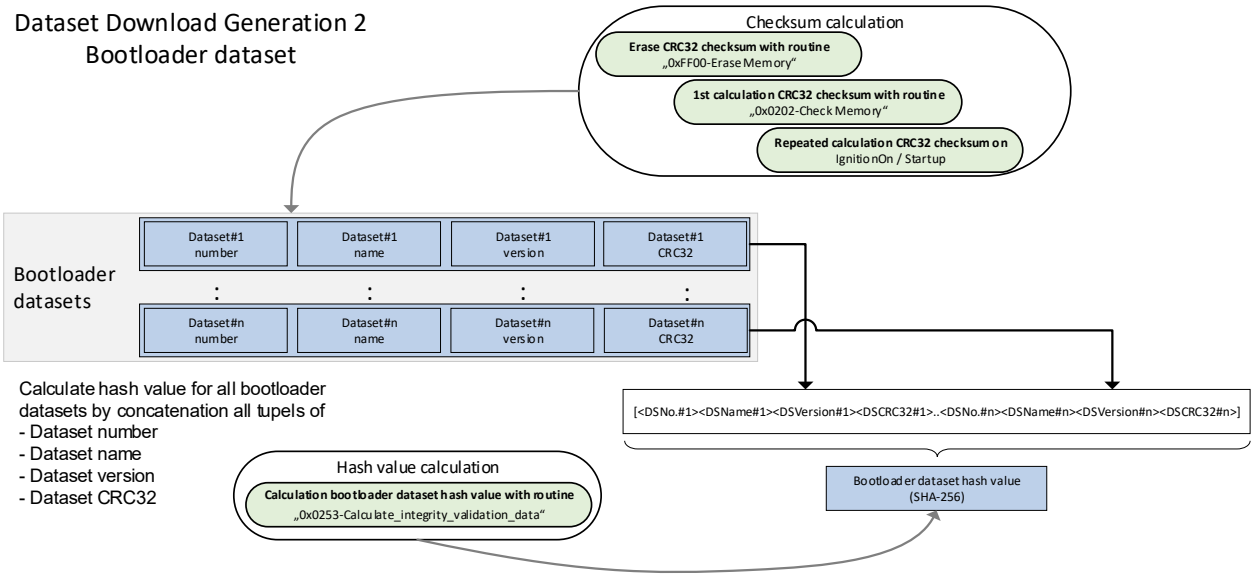
The individual hash value for all bootloader data sets is calculated from the concatenated tuples of all data sets, consisting of the data set number, the data set name, the data set version, and the CRC32 checksum as per document /7/.

[Prozess-Anf.: F-LAH_RxSWIN-1007]

The data set number is taken from the XML tag <SOURCE-START-ADDRESS> ... </SOURCE-START-ADDRESS>. Single-digit data set numbers must be expanded by 0x00 to a length of 2 bytes in the high-order byte (0x01 -> 0x00 01).

[I: F-LAH_RxSWIN-591]

Figure 4-7: Calculating the CRC32 checksum for generation 2 bootloader data sets



[I: F-LAH_RxSWIN-998]

This process also applies for calculating and reading individual hash values using the "0x0254-Calculate_individual_hash_value" routine.

4.1.2.1.4 Calculating the overall hash value of the configuration data

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

In DK2/DK2F/DK2FV/DK3/DK3V/DK4/DK4V/software cluster systems, the configuration data hash value is calculated on the basis of the list from data identifier "0x0250-Integrity_validation_data_configuration_list".

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

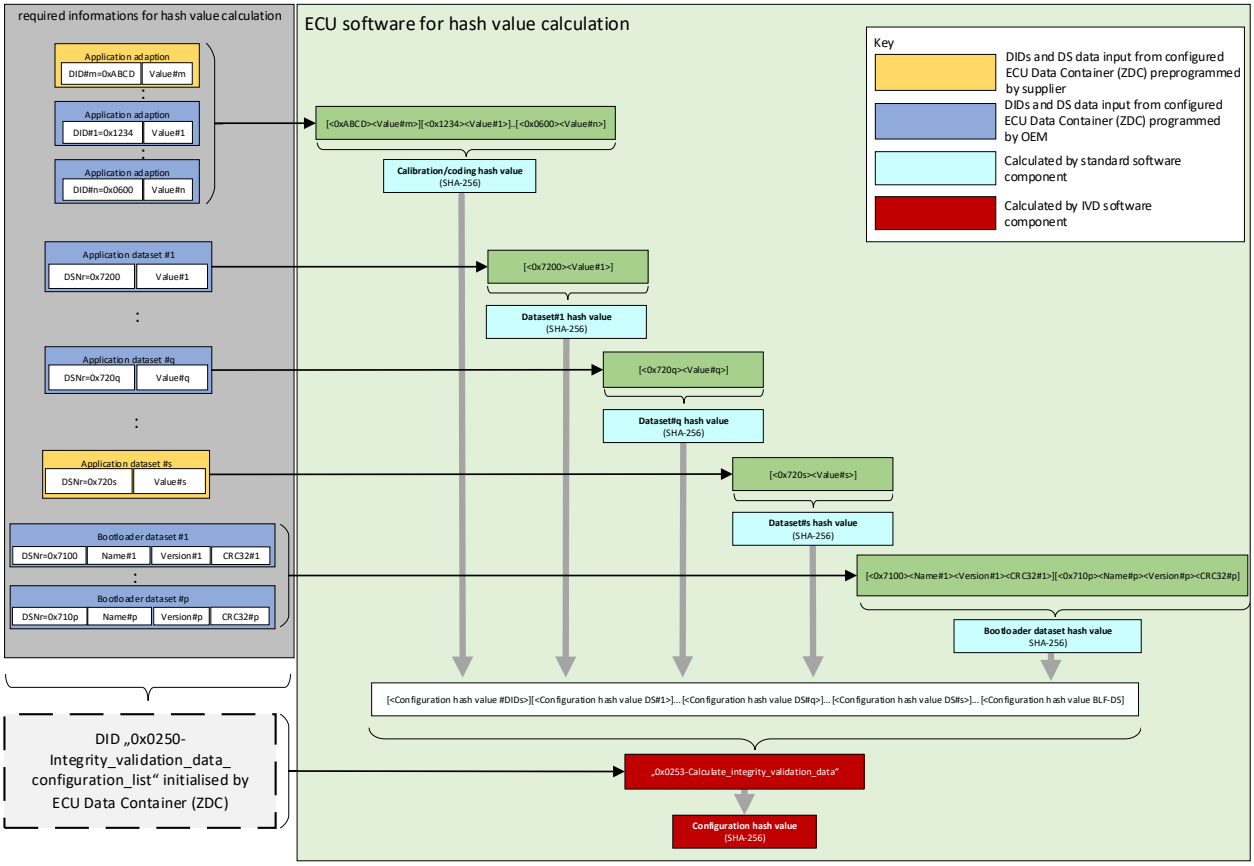
The "0x0253-Calculate_integrity_validation_data" routine with the control option [Type_of_calculation] = 0x00 triggers the calculation of the configuration data hash value for all data identifiers and data set numbers included in the list, in the order specified in the list.

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

The overall hash value of all configuration data to be output is the hash value of all concatenated individual hash values.

[I: F-LAH_RxSWIN-236]

Figure 4-8: Calculating the overall hash value of the configuration data (for Data Set Download Generation 2)

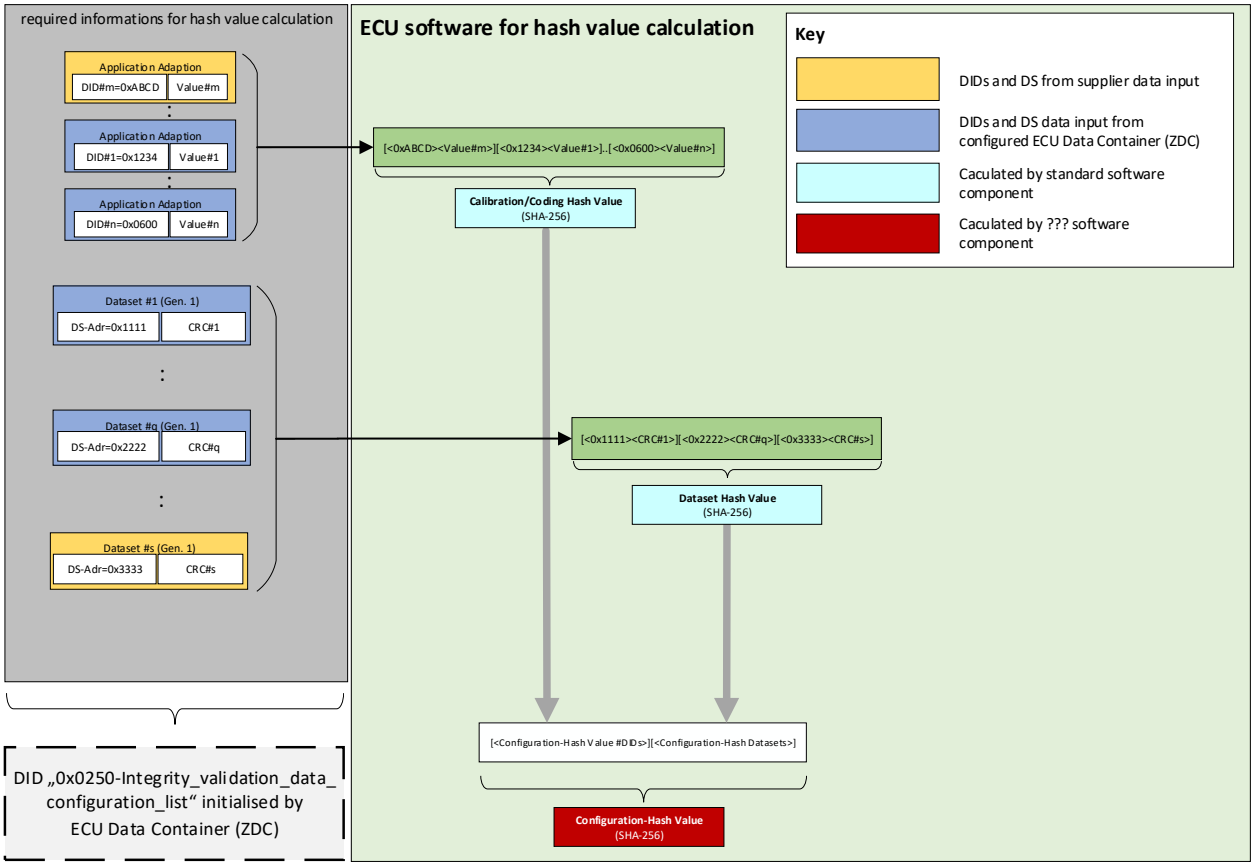


[I: F-LAH_RxSWIN-392]

DSNr = Data set number (2-byte hex) as per document /7/ 0x7100 - 0x71FF for bootloader data sets and 0x7200 - 0x72FF for application data sets
DID = Data identifier as per document /1/
Value = Data content of the adaptation or the coding or, for data sets, the programming data between the tags <DATEN> and </DATEN>
Hash = Hash value of the application data set (SHA-256, 32-byte length)
Version = Bootloader data set version as per "0xF1B1-VW_Application_data_set_identification" identification with the corresponding data set number as per document /2/
Name = Bootloader data set name as per "0xF1B3-VW_Data_set_name" identification with the corresponding data set number as per document /2/

[I: F-LAH_RxSWIN-666]

Figure 4-9: Calculating the overall hash value of the configuration data (for Data Set Download Generation 1)



[I: F-LAH_RxSWIN-667]

DS-Adr = Data set address (2-byte hex) as per document /11/
DID = Data identifier as per document /1/
Value = Data content of the adaptation or the coding or, for data sets, the programming data between the tags <DATEN> and </DATEN>
Hash = Hash value of the application data set (SHA-256, 32-byte length)

4.1.2.2 Requirements for processes

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

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

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

The order of the values for the data identifiers and for the data set numbers in the "0x0250-Integrity_validation_data_configuration_list" data identifier are determined in the ZDC calibration process and uploaded to the ECU.

[Prozess-Anf.: F-LAH_RxSWIN-1011]

The configuration data is clustered in the ZDC as follows:

- Adaptations/coding

- Application data sets
- Bootloader data sets

[Prozess-Anf.: F-LAH_RxSWIN-1012]

The appropriate part owner must specify the order of the clusters in the ZDC.

[Prozess-Anf.: F-LAH_RxSWIN-1014]

The appropriate part owner must specify the order of the data identifiers or data set numbers within a cluster in the ZDC.

[Prozess-Anf.: F-LAH_RxSWIN-1013]

The clusters of the data identifiers and data set numbers in the ZDC and their order specified in the ZDC determine the order of the data identifiers and data set numbers in the "0x0250-Integrity_validation_data_configuration_list" data identifier. This, in turn, specifies the scope and the order of the configuration hash value calculation.
Note: The real configuration order (order of the ECU write operation) may deviate from the order of data identifiers and data set numbers specified in the ZDC and must be determined during the initial setup process.

[Prozess-Anf.: F-LAH_RxSWIN-1015]

The initial entry in the "0x0250-Integrity_validation_data_configuration_list" data identifier is the 0x0250 data identifier itself.

[Prozess-Anf.: F-LAH_RxSWIN-962]

The "0x0250-Integrity_validation_data_configuration_list" DID is used to specify the order of the data identifiers and data sets for calculating the configuration hash value.

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

The desired configuration hash value is calculated in the IT system using the same calculation method as on the diagnostic server.

[Prozess-Anf.: F-LAH_RxSWIN-955]

The algorithm used to calculate the hash values must be documented with the ECU version in the IT system (version42 in this case).

4.2 Requirements for DK4/DK4V systems on the basis of Q-LAH 80127 starting from version 5.1

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

The DK4-low/DK4V-low system must retrieve the identification data for each DK2/DK2F/DK2FV system as a function of the trigger condition for group DIDs as per document /3/.

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

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

- Termination of the SFD E2E validated write operation from the DK4-low system to the DK2F system

4.2.1 Group data identifier for the integrity validation data for programming DK2F/DK2FV systems

[I: F-LAH_RxSWIN-187]

The "0x0247-Slave_list_programming_hash" data identifier (hash values for the programming of sub-bus systems) contains the integrity validation data for the instruction code of all lower-level DK2F/DK2FV systems.

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

The DK4-low/DK4V-low system must use the ReadDataByIdentifier (22hex) service via the "0x0247-Slave_list_programming_hash" group data identifier to output the integrity validation data for programming the DK2F/DK2FV systems.

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

A higher-level DK4-low/DK4V-low system uses the ReadDataByIdentifier (22hex) service with the "0x0249-Programming_hash" data identifier to collect the hash values for all lower-level DK2F/DK2FV systems assigned to it.

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

The "0x0247-Slave_list_programming_hash" group data identifier must not be writable by the WriteDataByIdentifier (2Ehex) service.

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

The implementation of the "0x0247-Slave_list_programming_hash" data identifier is mandatory for all DK4-low/DK4V-low systems with lower-level DK2F/DK2FV systems.

4.2.2 Group data identifier for the integrity validation data for configuring DK2/DK2F/DK2FV systems

[I: F-LAH_RxSWIN-480]

The "0x0248-Slave_list_configuration_hash" data identifier (hash values for the configuration of sub-bus systems) contains the integrity validation data for the configuration data of all lower-level DK2/DK2F/DK2FV systems.

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

The DK4-low system must use the ReadDataByIdentifier (22hex) service via the "0x0248-Slave_list_configuration_hash" group data identifier to output the integrity validation data for the configuration data of the DK2/DK2F/DK2FV systems.

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

A higher-level DK4-low/DK4V-low system uses the ReadDataByIdentifier (22hex) service with the "0x0245-Configuration_hash" data identifier to collect the hash values for all lower-level DK2/DK2F/DK2FV systems assigned to it.

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

The "0x0248-Slave_list_configuration_hash" group data identifier must not be writable by the WriteDataByIdentifier (2Ehex) service.

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

The implementation of the "0x0248-Slave_list_configuration_hash" group data identifier is mandatory for all DK4-low/DK4V-low systems with lower-level DK2/DK2F/DK2FV systems.

4.3 Requirements for DK4 systems on the basis of Q-LAH 80127 up to version 4.0

[I: F-LAH_RxSWIN-425]

For DK4 systems up to 80127 v4.0 (without group data identifiers), system-specific data identifiers are reserved in a separate data identifier range for the integration validation data of DK2 systems.

[I: F-LAH_RxSWIN-735]

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

DID value range for the programming hash value: 0xA800 - 0xA9FF
SubSystemNode (SSN): 0x01
Long name part 1: Control_unit_for_wiper_motor
Separator character: "_"

Long name part 2: Programming_hash

ODX generation rule: Long name combination part 1 +
separator character +
long name part 2 =
"Control_unit_for_wiper_motor_Programming_hash"

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

Note: These generation rules also apply to the configuration hash value, except for a modified offset of 0xAA00.

4.3.1 Programming hash value

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

DK4-low systems with lower-level DK2F 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 for each DK2F system with an SSN less than or equal to 0x1FF results from the following: data identifier 0xA800 + SubSystemNodeAddress (SSN).

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

DK4 systems must use the standardized "0x0249-Programming_hash" data identifier to request the data of the "Slave_x_programming_hash" DIDs from the lower-level DK2F systems.

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

For DK2F systems with an SSN less than or equal to 0x1FF, the individual data identifiers can also be used for the request.

[I: F-LAH_RxSWIN-488]

Note: For DK2F systems with an SSN greater than 0x1FF, group data identifiers must be used in the DK4-low system 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 by the WriteDataByIdentifier (2Ehex) service.

4.3.2 Configuration hash value

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

DK4-low systems with lower-level DK2/DK2F 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 for each DK2/DK2F system with an SSN less than or equal to 0x1FF results from the following: data identifier 0xAA00 + SubSystemNodeAddress (SSN).

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

DK4-low systems must use the standardized "0x0245-Configuration_hash" data identifier to request the data of the "Slave_x_configuration_hash" DIDs from the lower-level DK2/DK2F systems.

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

For DK2/DK2F systems with an SSN less than or equal to 0x1FF, the individual data identifiers can also be used for the request.

[I: F-LAH_RxSWIN-489]

Note: For DK2/DK2F systems with an SSN greater than 0x1FF, group data identifiers must be used in the DK4-low system 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 by the WriteDataByIdentifier (2Ehex) service.

4.4 Requirements for DK2/DK2F/DK2FV systems

4.4.1 Programming hash value

[I: F-LAH_RxSWIN-186]

The "0x0249-Programming_hash" data identifier is used to output the integrity validation data for the programming of a DK2F/DK2FV server.

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

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

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

Starting a "0xFF00-Erase Memory" routine that addresses a logical block with instruction code as per document /6/ makes it necessary to erase the existing integrity validation data in the DK2F/DK2FV system for the "0x0249-Programming_hash" data identifier.

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

When the request to read the "0x0249-Programming_hash" data identifier of the hash value for a DK2F/DK2FV diagnostic server is received via the ReadDataByIdentifier (22hex) service, this triggers the recalculation of the hash value on the DK2F/DK2FV system. Until the recalculation is completed, the response to the request must be an NRC of "0x78-RequestCorrectlyReceived-ResponsePending". A positive response to the request is only output after the recalculation is completed.

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

Starting a "0xFF00-Erase Memory" routine that addresses a logical block with instruction code as per document /6/ in DK2F/DK2FV systems with an SSN less than or equal to 0x1FF makes it necessary to erase the existing integrity validation data in the individual 0xA800 + SSN "VW_slave_programming_hash" data identifier.

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

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

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

Following a positive check to ensure that the logical block was transferred without any errors by the "0x0202-Check Memory" routine 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 a "0x0202-Check Memory" routine that addresses a logical block with instruction code, the calculated CRC32 checksum of the logical block is stored persistently.

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

To guarantee downward compatibility of DK2F/DK2FV systems with an SSN less than or equal to 0x1FF with DK4 systems on the basis of VW 80127 v4.0, the individual data identifier from the range 0xA800 to 0xA9FF with the same content/functionality must be supported in addition to the "0x0249-Programming_hash" data identifier.

[I: F-LAH_RxSWIN-736]

Note: The generic data identifier "0x0249-Programming_hash" in DK2F/DK2FV systems requires the "0x0247-Slave_list_programming_hash" group data identifier on the higher-level DK4-low system.

4.4.2 Configuration hash value

[I: F-LAH_RxSWIN-491]

The "0x0245-Configuration_hash" data identifier is used to output the integrity validation data for the configuration data of a DK2/DK2F/DK2FV server.

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

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

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

When the request to read the "0x0245-Configuration_hash" data identifier of the hash value for a DK2/DK2F/DK2FV diagnostic server is received via the ReadDataByIdentifier (22hex) service, this triggers the recalculation of the hash value in the DK2/DK2F/DK2FV system. Until the recalculation is completed, the response to the request must be an NRC of "0x78-RequestCorrectlyReceived-ResponsePending". A positive response to the request is only output after the recalculation is completed.

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

In DK2F/DK2FV systems with an SSN less than or equal to 0x1FF, the received request to read the individual 0xAA00 + SSN "VW_slave_configuration_hash" data identifier via the ReadDataByIdentifier (22hex) makes it necessary to recalculate the integrity validation data.

[I: F-LAH_RxSWIN-140]

The hash value is calculated for all writable data identifiers in the list of the "0x0250-Integrity_validation_data_configuration_list" data identifier. This calculation is described in detail in the section "Integrity validation data/General requirements/Configuration hash value."

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

To guarantee downward compatibility of DK2F/DK2FV systems with an SSN less than or equal to 0x1FF with DK4 systems on the basis of VW 80127 v4.0, the individual data identifier from the range 0xAA00 to 0xABFF with the same content/functionality must be supported in addition to the "0x0245-Configuration_hash" data identifier.

[I: F-LAH_RxSWIN-737]

Note: The generic data identifier "0x0245-Configuration_hash" in DK2F/DK2FV systems requires the "0x0248-Slave_list_configuration_hash" group data identifier in the higher-level DK4-low system.

4.5 Standard software module for integrity validation data

[I: F-LAH_RxSWIN-845]

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

4.6 Sequence

[I: F-LAH_RxSWIN-931]

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

4.6.1 Example: Reading all relevant identification data and integrity validation data from a diagnostic server

[I: F-LAH_RxSWIN-246]

A diagnostic client reads the identification data of a DK4 system with lower-level DK2F/DK2FV systems.
The following are read:

[I: F-LAH_RxSWIN-836]

- Vehicle- and ECU-specific identification data

[I: F-LAH_RxSWIN-835]

- Integrity validation data for the configuration data for the adaptation/coding/application/bootloader data sets of a DK4 system

[I: F-LAH_RxSWIN-834]

- Integrity validation data for the programming (instruction code) of a DK4 system

[I: F-LAH_RxSWIN-833]

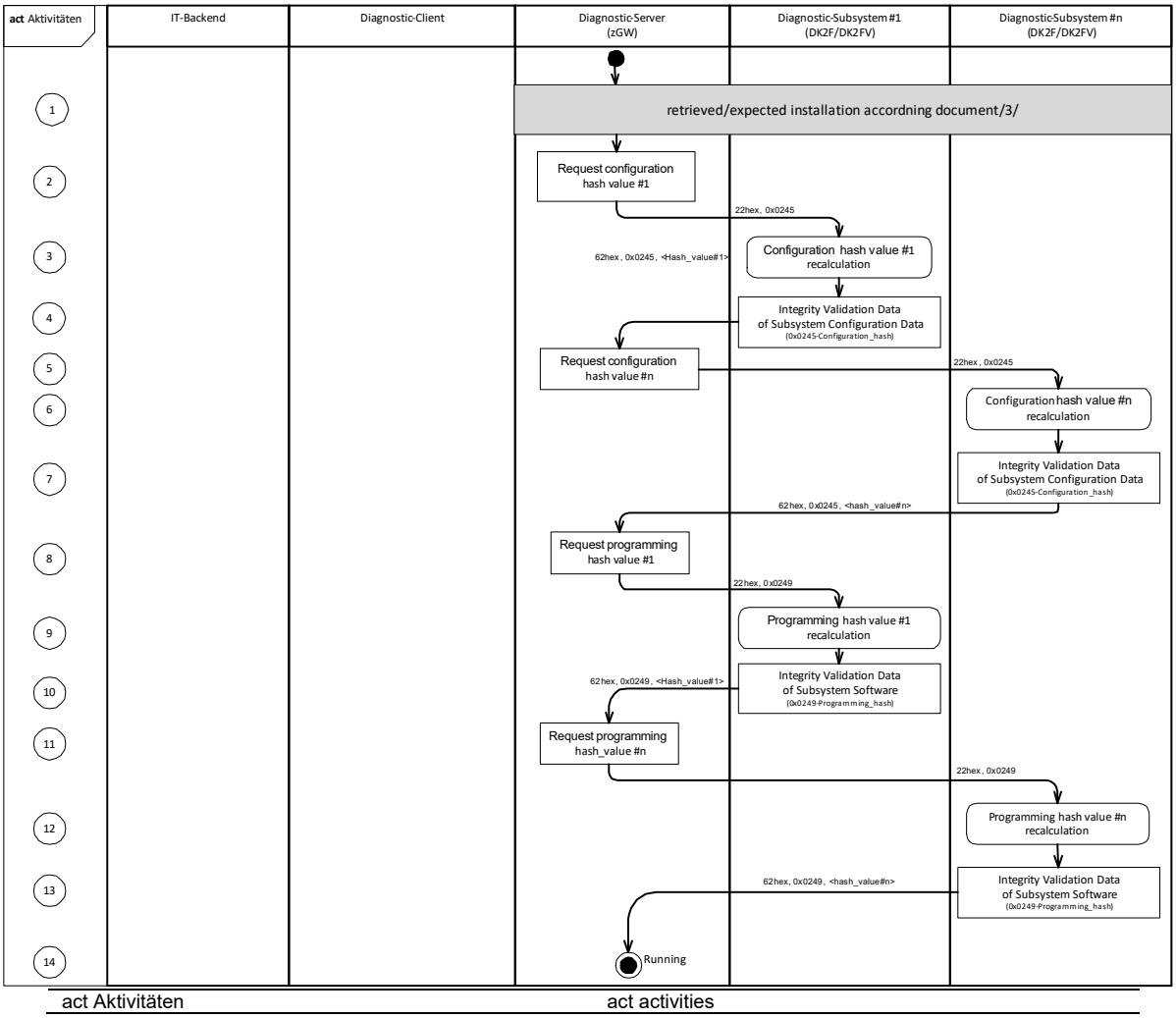
- Integrity validation data for the configuration data for the adaptation (data) of a lower-level DK2F/DK2FV system

[I: F-LAH_RxSWIN-832]

- Integrity validation data for the programming (instruction code) of a lower-level DK2F/DK2FV system

[I: F-LAH_RxSWIN-670]

Figure 4-10: Initializing the group data identifier for the integrity validation data using a gateway as an example – part 1/4



[I: F-LAH_RxSWIN-865]

1 – After the central gateway starts up, it updates the actual VBV component list with all detected, lower-level diagnostic sub-systems as per document /3/, section "Retrieving the identification data from DK1/DK2/DK2F systems."

[I: F-LAH_RxSWIN-671]

2 – After updating its VBV component list, the central gateway requests the "0x0245-Configuration_hash" data identifier for the configuration data hash value from the lower-level diagnostic sub-system#1.

[I: F-LAH_RxSWIN-672]

3 – In diagnostic sub-system#1, receiving the ReadDataByIdentifier request with the "0x0245-Configuration_hash" data identifier starts the recalculation of the configuration data hash value.

[I: F-LAH_RxSWIN-673]

4 – Diagnostic sub-system#1 sends the response with the recalculated configuration data hash value. The central gateway updates the "0x0248-Slave_list_configuration_hash" group data identifier on the basis of the received "0x0245-Configuration_hash" data identifier.

[I: F-LAH_RxSWIN-674]

5 – The central gateway requests the "0x0245-Configuration_hash" data identifier for the configuration data hash value of the lower-level diagnostic sub-system#n from the lower-level diagnostic sub-system#n.
Note: The diagnostic sub-systems#1 through #n can be requested in parallel on the buses to save time; this section only describes sequential requests.

[I: F-LAH_RxSWIN-675]

6 – In diagnostic sub-system#n, receiving the ReadDataByIdentifier request with the "0x0245-Configuration_hash" data identifier starts the recalculation of the configuration data hash value.

[I: F-LAH_RxSWIN-676]

7 – Diagnostic sub-system#n sends the recalculated configuration data hash value. The central gateway updates the "0x0248-Slave_list_configuration_hash" group data identifier on the basis of the received "0x0245-Configuration_hash" data identifier.

[I: F-LAH_RxSWIN-677]

8 – The central gateway requests the "0x0249-Programming_hash" data identifier for the programming hash value from the lower-level diagnostic sub-system#1.

[I: F-LAH_RxSWIN-710]

9 – In diagnostic sub-system#1, receiving the ReadDataByIdentifier request with the data identifier "0x0249-Programming_hash" starts the recalculation of the programming data hash value.

[I: F-LAH_RxSWIN-679]

10 – The lower-level diagnostic sub-system#1 sends the stored programming hash value. The central gateway updates the "0x0247-Slave_list_configuration_hash" group data identifier on the basis of the received "0x0249-Programming_hash" data identifier.

[I: F-LAH_RxSWIN-680]

11 – The central gateway requests the "0x0249-Programming_hash" data identifier for the programming hash value from diagnostic sub-system#n.

[I: F-LAH_RxSWIN-711]

12 – In diagnostic sub-system#n, receiving the ReadDataByIdentifier request with the data identifier "0x0249-Programming_hash" starts the recalculation of the programming data hash value.

[I: F-LAH_RxSWIN-682]

13 – Diagnostic sub-system#n sends the stored programming hash value. The central gateway updates the "0x0247-Slave_list_programming_hash" group data identifier on the basis of the received "0x0249-Programming_hash" data identifier.

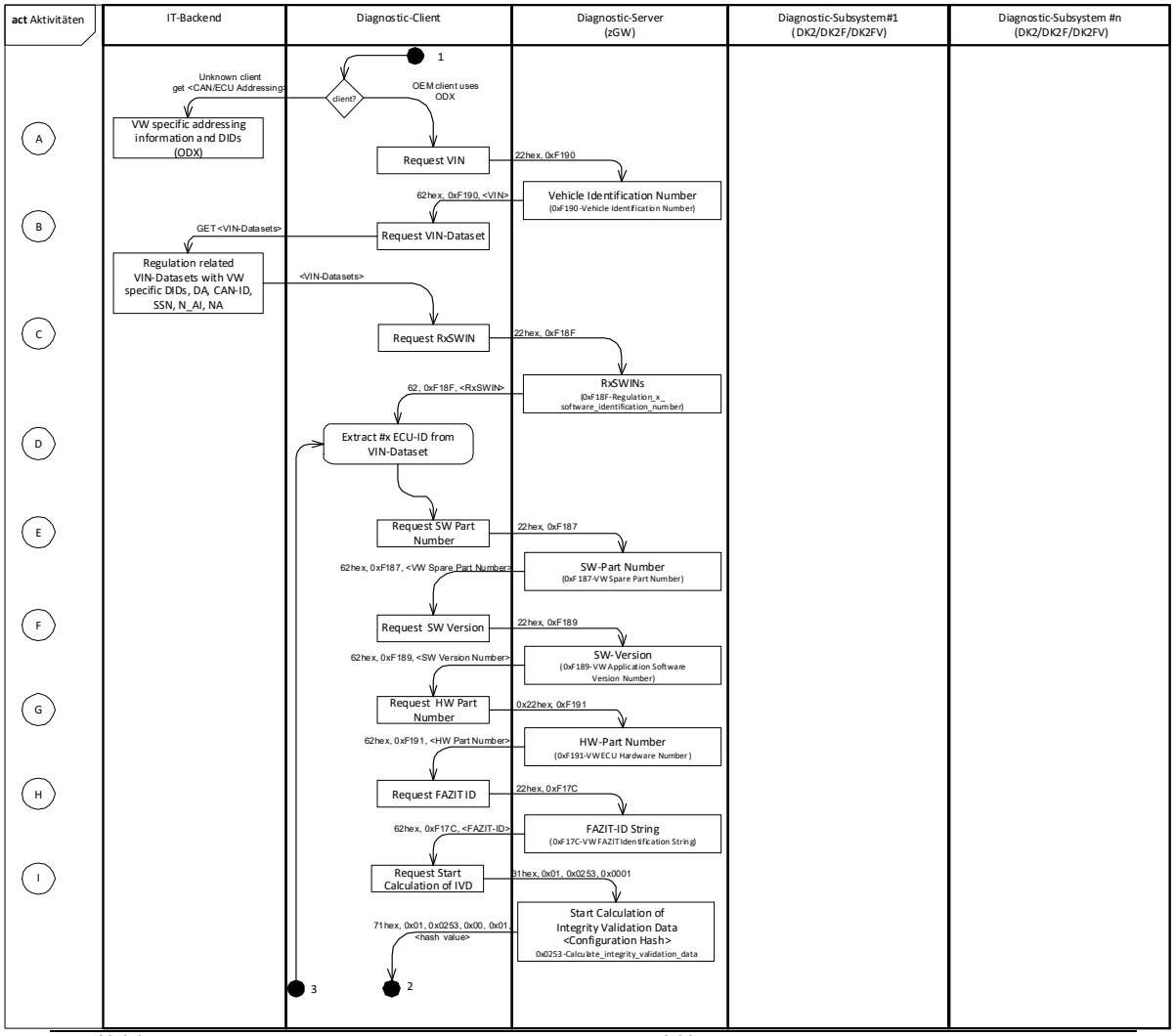
[I: F-LAH_RxSWIN-683]

14 – The central gateway updates the other group data identifiers in the diagnostic sub-systems.

CONFIDENTIAL
VERTRAULICH

[I: F-LAH_RxSWIN-245]

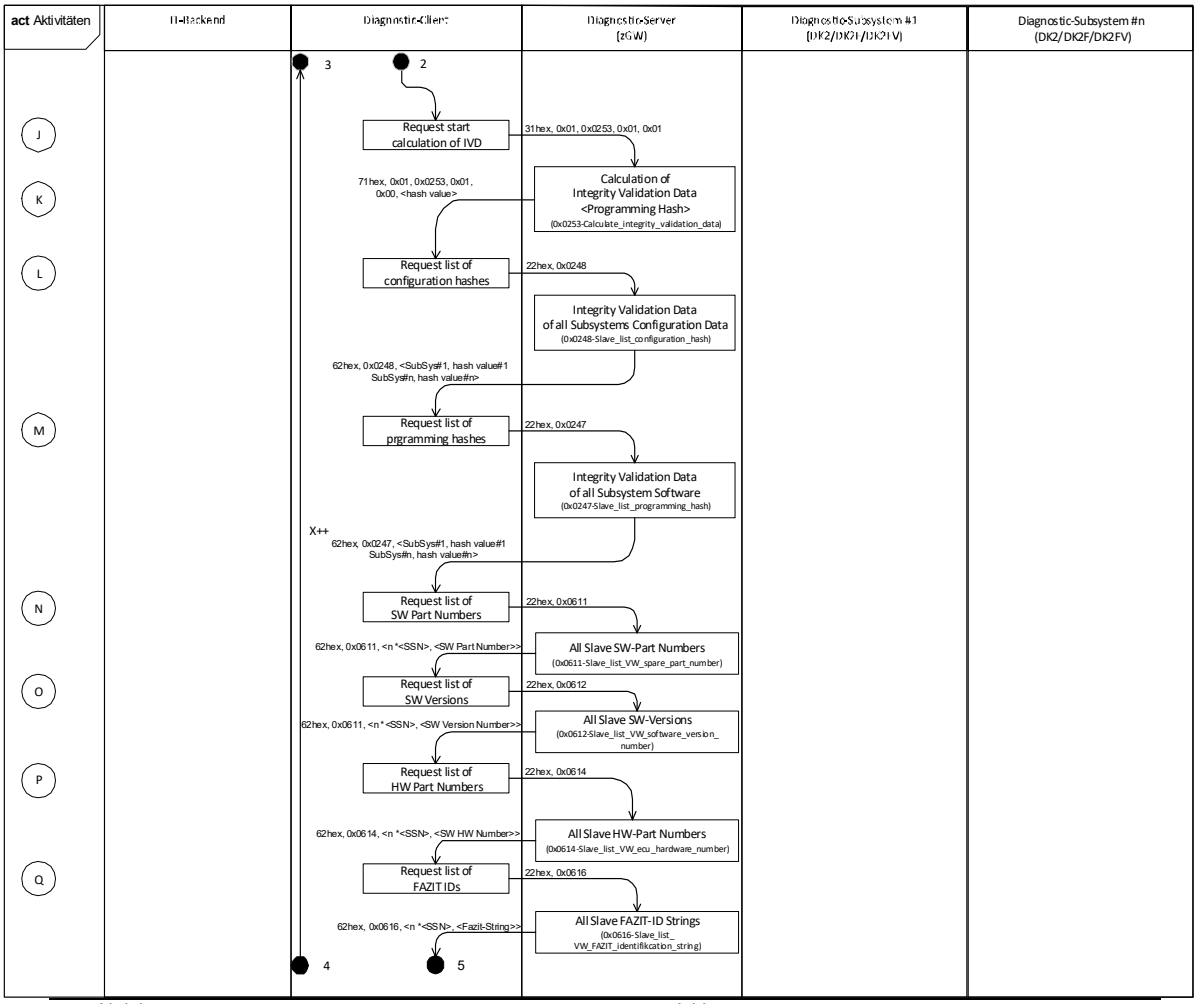
Figure 4-11: Reading all identification data and integrity validation data – part 2/4



act Aktivitäten — act activities

[I: F-LAH_RxSWIN-248]

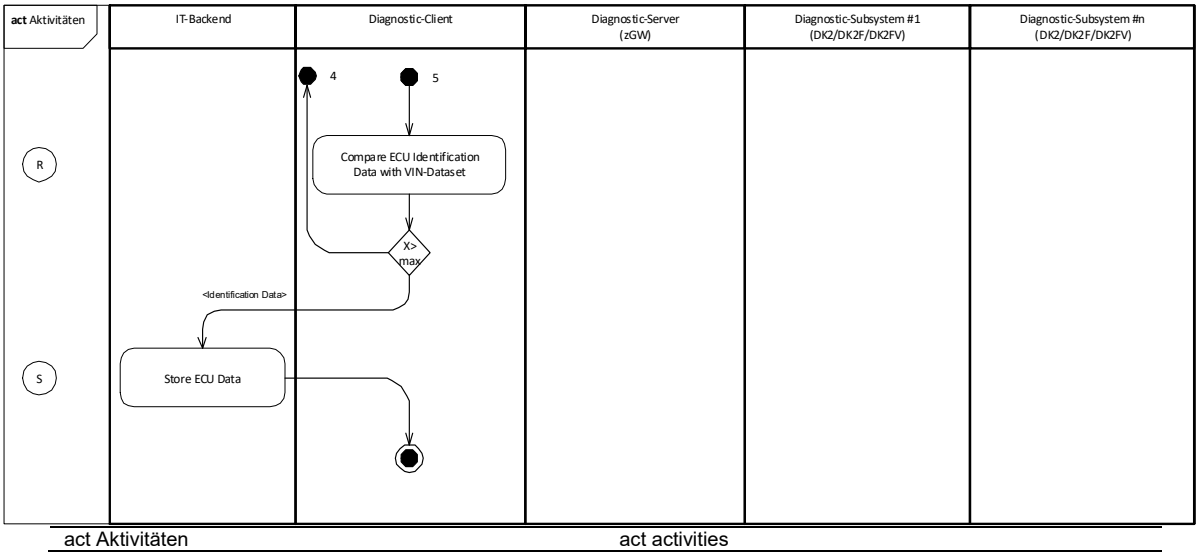
Figure 4-12: Reading all identification data and integrity validation data – part 3/4



act Aktivitäten act activities

[I: F-LAH_RxSWIN-250]

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



[I: F-LAH_RxSWIN-249]

A – The diagnostic client requests the address information required for communication with the vehicle and the data identifiers required for the vehicle's identification data from the IT backend. Address information means the diagnostic addresses (DAs), node addresses (NAs), sub-system node address (SSNs), network address information (N_AI), and the CAN identifiers for the request and response. The diagnostic client requests the vehicle identification number (VIN) for the current vehicle via the "0xF190-Vehicle Identification Number" data identifier. If the Offboard Diagnostic Information System (ODIS) is used, then all the required information is available in the ODX data.

[I: F-LAH_RxSWIN-251]

B – The diagnostic client requests the vehicle- and regulation-specific data set for the vehicle's VIN from the IT backend in order to receive additional address information, such as the diagnostic addresses, node addresses, and data identifiers.

[I: F-LAH_RxSWIN-252]

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

[I: F-LAH_RxSWIN-253]

D – Based on the data from the regulation-specific data set, the diagnostic client addresses the individual diagnostic servers installed in the vehicle in succession.

[I: F-LAH_RxSWIN-254]

E – The diagnostic client reads the software part number via the "0xF187-VW Spare Part Number" data identifier of the addressed diagnostic server.

[I: F-LAH_RxSWIN-255]

F – The diagnostic client reads the software version via the "0xF189-VW Software Version Number" data identifier of the addressed diagnostic server.

[I: F-LAH_RxSWIN-256]

G – The diagnostic client reads the hardware part number via the "0xF191-VW ECU Hardware Number" data identifier of the addressed diagnostic server.

[I: F-LAH_RxSWIN-257]

H – The diagnostic client reads the Vehicle Information and Central Identification Tool (FAZIT) ID via the "0xF17C-VW FAZIT Identification String" data identifier of the addressed diagnostic server.

[I: F-LAH_RxSWIN-259]

I – The diagnostic client requests the configuration data hash value of the addressed diagnostic server. The "0x0250-Integrity_validation_data_configuration_list" data identifier contains the list of all data identifiers and data set numbers with which the configuration data hash value must be calculated. The "0x0253-Calculate_integrity_validation_data" routine identifier returns the configuration data hash value of the diagnostic server in its positive response.

[I: F-LAH_RxSWIN-260]

J – The diagnostic client starts calculating the hash value for the overall programming for the addressed diagnostic server.

[I: F-LAH_RxSWIN-262]

K – The diagnostic client receives the recalculated programming hash value for the diagnostic server.

[I: F-LAH_RxSWIN-263]

L – The diagnostic client requests the "0x0248-Slave_list_configuration_hash" group data identifier for the configuration data hash values of all diagnostic sub-systems for the diagnostic server.

[I: F-LAH_RxSWIN-264]

M – The diagnostic client requests the "0x0247-Slave_list_programming_hash" group data identifier for the programming hash values of all diagnostic sub-systems for the diagnostic server.

[I: F-LAH_RxSWIN-266]

N – The diagnostic client uses the "0x0611-Slave_list_VW_spare_part_number" group data identifier to read the list of all software part numbers for the diagnostic sub-systems via the addressed diagnostic server.

[I: F-LAH_RxSWIN-545]

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

[I: F-LAH_RxSWIN-546]

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

[I: F-LAH_RxSWIN-267]

Q – The diagnostic client uses the "0x0616-Slave_list_VW_FAZIT_identification_string" group data identifier to read the list of all FAZIT identification data for the diagnostic sub-systems via the addressed diagnostic server.

[I: F-LAH_RxSWIN-270]

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

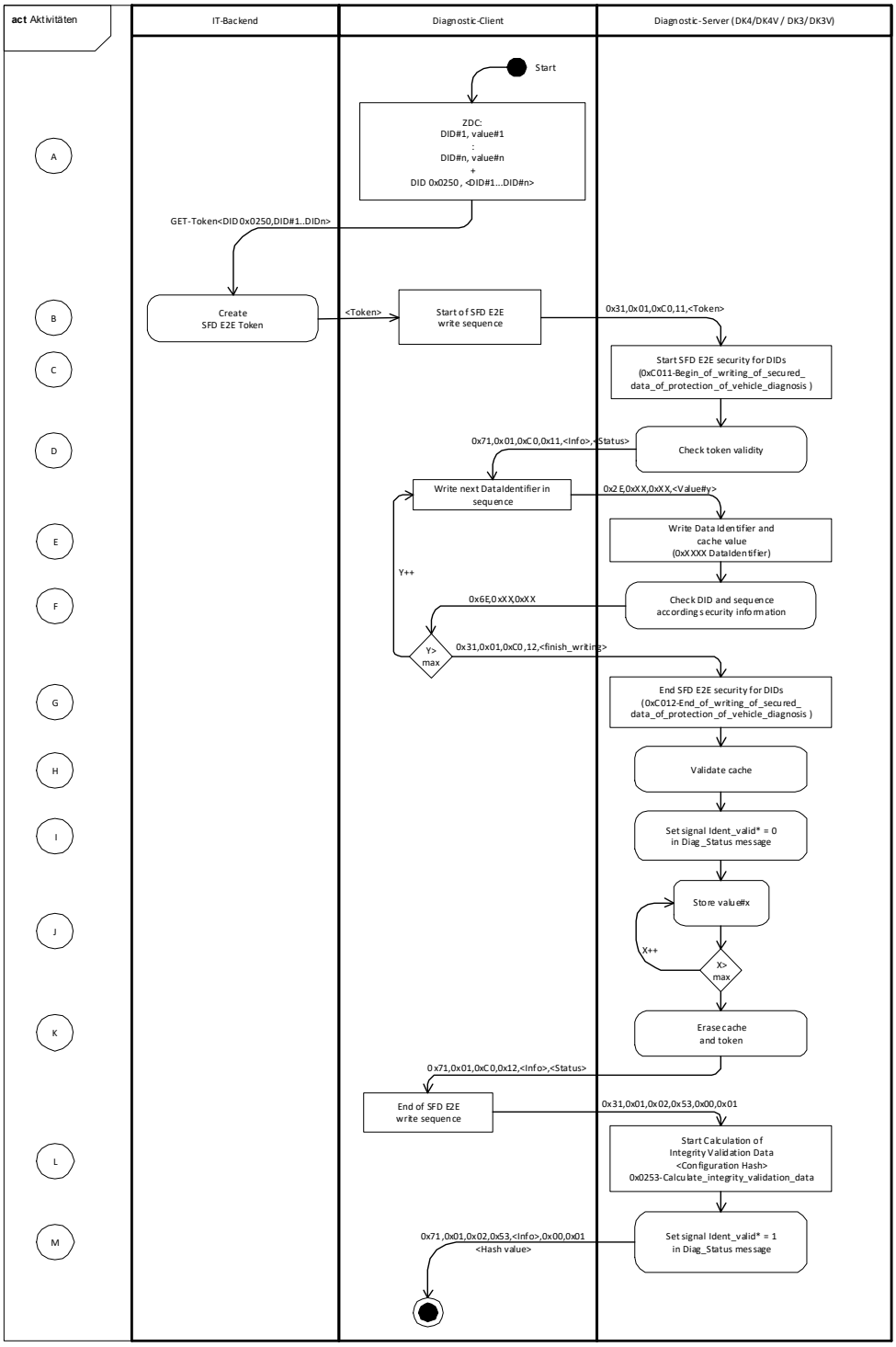
[I: F-LAH_RxSWIN-547]

S – The diagnostic client stores the identification data it read in the IT backend.

4.6.2 Example: Writing data on a SFD E2E validated DK4/DK4V or DK3/DK3V system

[I: F-LAH_RxSWIN-356]

Figure 4-14: Writing the data of a DK3/DK3V or DK4/DK4V system – part 1/1



Note: The signal Ident_valid (QLAH 80114 from version 5.6) has only informational character here and no requirement character.

act Aktivitäten	act activities

[I: F-LAH_RxSWIN-537]

A – The data identifiers required for the ECU configuration are made available to the diagnostic client. The data identifiers are defined by the production order that configures the ZDC; the content of the "0x0250-Integrity_validation_data_configuration_list" data identifier also results from this. The diagnostic client requests a validation token at the SFD IT backend for the data identifiers DID#1 to DID#n contained in the configured ZDC and the calculated data identifier "0x0250-Integrity_validation_data_configuration_list".

[I: F-LAH_RxSWIN-359]

B* – The SFD IT backend generates a validation token via the data identifiers DID#1 to DID#n, and DID 0x0250.

[I: F-LAH_RxSWIN-360]

C* – The validation token is transferred to the diagnostic server when the "0xC011-Begin_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis" SFD routine starts. This triggers the write sequence for the SFD E2E validated data.

[I: F-LAH_RxSWIN-361]

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

[I: F-LAH_RxSWIN-362]

E – The data identifiers from the configured ZDC and the "0x0250-Integrity_validation_data_configuration_list" data identifier are transferred by the WriteDataByIdentifier (2Ehex) service to the diagnostic server, where they are cached. Without SFD, the data is written directly to memory.

[I: F-LAH_RxSWIN-363]

F* – In accordance with SFD E2E validation as per document /4/, the receive sequence of the individual data identifiers is checked using the validation information. If the check is passed, the diagnostic server replies with a positive response. If the check is failed, the received data identifiers and the associated data records as well as the validation information are deleted. If the check is failed, the diagnostic server responds with a negative response code (NRC) of "0x22-ConditionsNotCorrect".

[I: F-LAH_RxSWIN-364]

G* – The SFD write sequence is concluded with the [0x01-Finish_writing_of_secured_data] routine control option when the "0xC012-End_of_writing_of_secured_data_of_protection_of_vehicle_diagnosis" SFD routine starts.

[I: F-LAH_RxSWIN-365]

H* – The authenticity of the transferred data identifiers is checked on the basis of the validation information.

[I: F-LAH_RxSWIN-368]

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

[I: F-LAH_RxSWIN-366]

J* – If the check is passed, all cached data identifiers are written sequentially from the cache to the diagnostic server's target memory. If the check is failed, then all received data identifiers, and the associated data records and validation information, are erased.

[I: F-LAH_RxSWIN-369]

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

[I: F-LAH_RxSWIN-370]

L – The diagnostic client requests the new configuration data hash value. Calling the "0x0253-Calculate_integrity_validation_data" routine starts the recalculation of the configuration data hash value on the diagnostic server. The hash value is calculated on the basis of the list of data identifiers and data set numbers transferred in the "0x0250-Integrity_validation_data_configuration_list" data identifier (see section "0x0250-Integrity_validation_data_configuration_list").

[I: F-LAH_RxSWIN-371]

M – A value of '1' in the [Ident_valid] signal of the diagnostic server's Diag_Status message indicates that the identification data is completely valid. In other words, the hash value has been updated.

[I: F-LAH_RxSWIN-527]

*Only relevant to diagnostic systems with SFD E2E validation. Not relevant to temporarily unlocked diagnostic systems with group unlocking in production, upstream of CP8

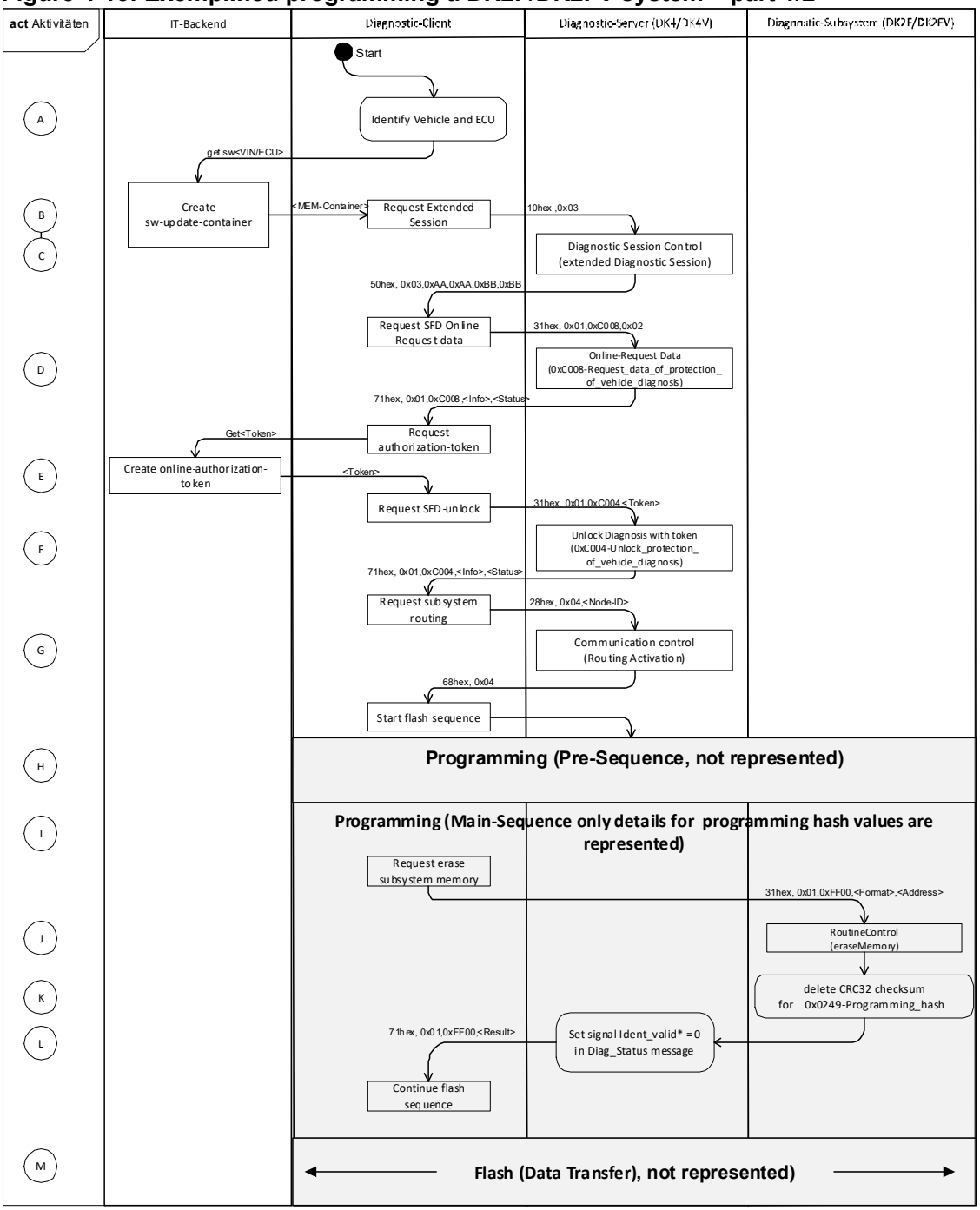
4.6.3 Exemplified programming a DK2F/DK2FV system

[I: F-LAH_RxSWIN-303]

Sequence for a software update of a DK2F/DK2FV system by a diagnostic client via a DK4/DK4V system.

[I: F-LAH_RxSWIN-305]

Figure 4-15: Exemplified programming a DK2F/DK2FV system – part 1/2

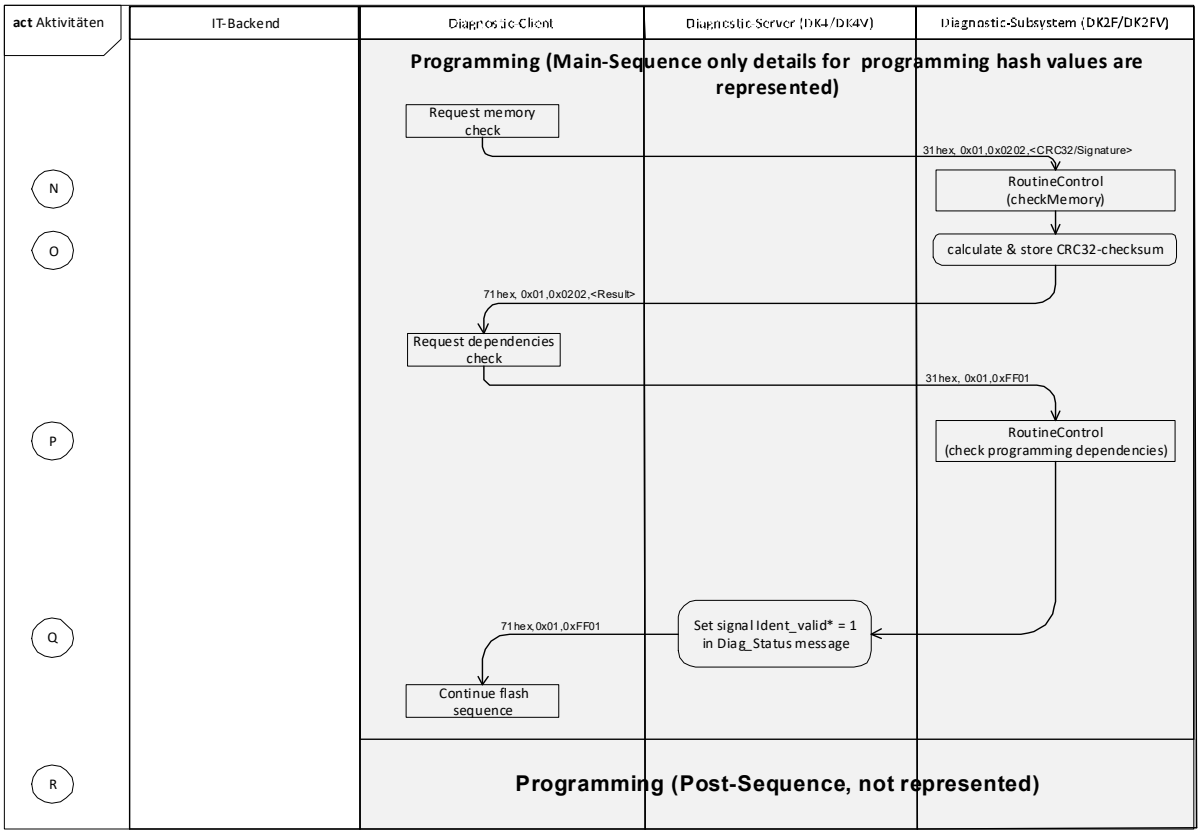


*Note: The signal Ident_valid (document QLAH 80114 from version 5.6) has only informational character here and no requirement character.

act Aktivitäten	act activities

[I: F-LAH_RxSWIN-306]

Figure 4-16: Exemplified programming a DK2F/DK2FV system – part 2/2



Note: The signal Ident_valid (QLAH 80114 from version 5.6) has only informational character here and no requirement character.

act Aktivitäten — act activities

[I: F-LAH_RxSWIN-307]

A – The diagnostic client uses the VIN to identify the current vehicle and the identification data to identify the ECU to be updated.

[I: F-LAH_RxSWIN-310]

B – The diagnostic client downloads the corresponding software update container that 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 SFD online unlocking from the diagnostic server.

[I: F-LAH_RxSWIN-313]

E – The diagnostic client uses the request data to request a unlock token from the SFD backend.

[I: F-LAH_RxSWIN-314]

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

[I: F-LAH_RxSWIN-315]

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

[I: F-LAH_RxSWIN-316]

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

[I: F-LAH_RxSWIN-317]

I – Programming main sequence as per document /6/

[I: F-LAH_RxSWIN-318]

J – The diagnostic client requests the 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 simultaneously causes the deletion of the CRC32 checksum for the block to be deleted.

[I: F-LAH_RxSWIN-523]

L – The DK4/DK4V system uses the Diag_Status message in the [Ident_valid] = 0 signal to indicate that the ECU identification data is no longer valid.

[I: F-LAH_RxSWIN-320]

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

[I: F-LAH_RxSWIN-321]

N – After the flashing sequence, the diagnostic client requests that the diagnostic sub-system verify correct receipt of the memory block.

[I: F-LAH_RxSWIN-322]

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

[I: F-LAH_RxSWIN-323]

P – After the flashing sequence, the diagnostic client requests that the diagnostic sub-system perform a compatibility/validation check of the correctly received memory block.

[I: F-LAH_RxSWIN-525]

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

[I: F-LAH_RxSWIN-324]

R – Programming post-sequence as per document /6/

5 Diagnostic objects

[I: F-LAH_RxSWIN-54]

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

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

Table 5-1: Diagnostic functions and diagnostic classes

Diagnostic object/function	DK4-high	DK4-low	DK4V-low	DK4*)	DK3	DK3V	DK2	DK2F	DK2FV	SWCL	Note
RxSWIN in the "0xF18F-Regulation_x_software_identification_numbers" DID	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	This must be implemented only once per vehicle on the central gateway.
Software part number in the "0xF187-VW Spare Part Number" DID	x	x	x	x	x	x	x	x	x	x	
Software version in the "0xF189-VW Software Version Number" DID	x	x	x	x	x	x	x	x	x	x	
Hardware part number in the "0xF191-VW ECU Hardware Number" DID	x	x	NoImp	x	x	NoImp	x	x	NoImp	NoImp	
Hardware version in the "0xF1A3-VW ECU Hardware Version Number" DID	x	x	NoImp	x	x	NoImp	x	x	NoImp	NoImp	
VIN in the "0xF190-Vehicle Identification Number" DID	(x)	(x)	(x)	(x)	(x)	(x)	NoImp	NoImp	NoImp	NoImp	VIN only for immobilizer nodes, Diagnostic over Internet Protocol (DoIP) servers, and central gateway
Configuration hash value in the "0245-Configuration_hash" DID	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	x	x	x	NoImp	Configuration hash value. Diagnostic servers that are only flashable and not configurable can use NRC 0x31 to reject this DID.
Group DID "0x0248-Slave_list_configuration_hash" for "0x0245-Configuration_hash"	NoImp	x	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	For DK4-high: only with lower-level DK2 systems
Programming hash value in the "0249-Programming_hash" DID	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	x	x	NoImp	Program hash value
Group DID "0x0247-Slave_list_programming_hash" for "0x0249-Programming_hash"	NoImp	x	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	For DK4-high: only with lower-level DK2 systems
0x0251_Write_generic_to_sub_system	NoImp	x	x	x	NoImp	NoImp	NoImp	NoImp	NoImp	NoImp	For DK4-high: only with lower-level DK2 systems
List for hash value calculation of configuration data in the "0x0250-List_of_configuration_data_identifier" DID	x	x	x	x	x	x	x	x	x	x	For DK2/DK2F: Calibration occurs via the bus master/host using 0x0251.
Calculation of the integrity validation data using the "0x0253-Calculate_integrity_validation_data" RID	x	x	x	x	x	x	NoImp	NoImp	NoImp	x	Program and data hash value
Calculation of the individual hash values using the "0x0254-Calculate_individual_hash_value" RID	x	x	x	x	x	x	NoImp	NoImp	NoImp	x	Individual hash value of the configuration data
Complete recalculation of the IVD using the "0x029A-Calculate_module_hash_value" RID	x	x	x	x	x	x	NoImp	x	x	x	Programming hash value for non-embedded or file-based systems and data
Programming hash value for Q-LAH 80127 < v5.x sub-systems (DID range 0xA800 - 0xA9FF)	NoImp	(x)	(x)	x	NoImp	NoImp	NoImp	x	NoImp	NoImp	Only for DK4 with DK2F in existing architectures (individual DID for 0608 identification as per 80127 < v5.x)
Configuration hash value for Q-LAH 80127 < v5.x sub-systems (DID range 0xAA00 - 0xABFF)	NoImp	(x)	(x)	x	NoImp	NoImp	x	x	NoImp	NoImp	

[I: F-LAH_RxSWIN-855]

*) = DK4 as per Q-LAH 80127 up to v4.0 (without being divided into DK4-low/DK4-high)

[I: F-LAH_RxSWIN-856]

X = included

[I: F-LAH_RxSWIN-857]

(x) = optional

5.1 Data identifiers

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

Table 5-2: Overview of the data identifier properties

			Diagnostic session			
			Default session	Non-default session		
				ECU programming session	Extended diagnostic session	Volkswagen end-of-line session
Diagnostic objects (DIDs)	SID	Security level	0x01	0x02	0x03	0x40
0xF1A3-VW_ECU_Hardware_version_number	22hex	NoImp	Available	Available	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0xF1B8-VW_system_firmware_versions	22hex	NoImp	NoImp	NoImp	NoImp	NoImp
	2Ehex	NoImp	NoImp	NoImp	NoImp	NoImp
0xF1A0-VW_data_set_number_or_ECU_data_container_number	22hex	NoImp	U	notAvailable	U	U
	2Ehex	SFD-E2E	NoImp	notAvailable	U	U
0xF1A1-VW_data_set_version_number	22hex	NoImp	U	notAvailable	U	U
	2Ehex	SFD-E2E	NoImp	notAvailable	U	U
0xF1B1-VW_Application_data_set_identification	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0xF1B3-VW_Data_set_name	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0x0249-Programming_hash	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0x0247-Slave_list_programming_hash	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0x0245-Configuration_hash	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0x0248-Slave_list_configuration_hash	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
0xF18F-Regulation_x_software_identification_number	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	SFD-E2E	notAvailable	notAvailable	Available	C2
0x0250-Integrity_validation_data_configuration_list	22hex	NoImp	Available	notAvailable	Available	C2
	2Ehex	SFD-E2E *)	notAvailable	notAvailable	Available	C2
0x0251-Write_generic_to_sub_system	22hex	NoImp	notAvailable	notAvailable	notAvailable	notAvailable
	2Ehex	Old platform: C1 New platform: SFD-E2E	notAvailable	notAvailable	Available	C2

[I: F-LAH_RxSWIN-1064]

*) Access protection is not required if the ECU does not support SFD E2E validation.

[I: F-LAH_RxSWIN-936]

C1 – The access protection method (if applicable) for DK4-low must be used.

[I: F-LAH_RxSWIN-1066]

C2 - Only available for diagnostic servers that implement this session as per document /1/.

5.1.1 0xF1A3-VW ECU Hardware Version Number

[I: F-LAH_RxSWIN-664]

This data identifier 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 v5.8 must be observed:

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

Removed:

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

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

New:

The following changes must be documented in the hardware version:

- Changes to active components (e.g., microcontroller, microprocessor, RAM, flash, and application-specific integrated circuit (ASIC))

- Discontinuation of passive components
- Changes to the printed circuit board (PCB), e.g., relating to electromagnetic compatibility (EMC) or current draw
- Changes to the functional behavior, e.g., reserved option is populated
- Changes affecting 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 v5.8 must be observed:

- The implementation of the "0xF1A0-VW Data Set Number Or ECU Data Container Number" data identifier on the diagnostic server is a user option.

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

The configuration data written by the ZDC is documented by the configuration hash value.

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

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

[I: F-LAH_RxSWIN-963]

The "0xF1A0-VW Data Set Number Or ECU Data Container Number" data identifier belongs to the analysis data category; it is therefore not relevant to the configuration hash value.

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 v5.8 must be observed:

- The implementation of the "0xF1A1-VW Data Set Version Number" data identifier on the diagnostic server is a user option.

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

The configuration data written by the ZDC is documented by the configuration hash value.

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

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

[I: F-LAH_RxSWIN-964]

The "0xF1A1-VW Data Set Version Number" data identifier belongs to the analysis data category; it is therefore not relevant to the configuration hash value.

5.1.4 0xF1B1-VW_Application_data_set_identification

[I: F-LAH_RxSWIN-979]

Note: The [Data_set_ID] and [Data_set_version] parameters are relevant to calculating the configuration hash value.

5.1.5 0xF1B3-VW_data_set_name

[allg. Anf.: F-LAH_RxSWIN-975]

The following change compared with the Q-LAH for Data Set Download Generation 2 up to and including v3.1 must be observed:

- The implementation of the "0xF1B3-VW_data_set_name" data identifier on the diagnostic server is mandatory.

[I: F-LAH_RxSWIN-976]

Note: The [Data_set_ID] and [Data_set_name] parameters are relevant to calculating the configuration hash value.

5.1.6 0x0249-Programming_hash

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

Table 5-3: Structure of data record for the 0x0249-Programming_hash data identifier

DID	**0x0249**
Designation	Programming_hash
Description	This data identifier contains the integrity validation data for the programming data. The hash value is calculated for the entire software. This routine identifier must be readable while the vehicle is stationary and in the state "not ready for driving." ECU-specific boundary conditions may apply in other vehicle states (e.g., active control interventions). The ready-for-driving and speed signals must be used on an architecture-specific basis as per the valid data definition.
Convention	All servers
Diagnostic class	DK2F/DK2FV
Session	APP: 0x01 (R), 0x03 (R), 0x40 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[Programming_hash] 32-byte hex>
Range	[Programming_hash] 00..00hex to FF..FFhex
Init	Not applicable, always available
Example	-
Data category	Analysis data
eBZD	M

5.1.7 0x0247-Slave_list_programming_hash

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

Table 5-4: Structure of data record for the 0x0247-Slave_list_programming_hash data identifier

DID	**0x0247**
Designation	Slave_list_programming_hash
Description	This data identifier contains all the integrity validation data for the instruction code of the lower-level DK2F/DK2FV systems. The following definitions apply: • The respective content corresponds to the [Programming_hash] parameter of the "0x0249-Programming_hash" data identifier. • The respective SubSystemNodeAddress must be prepended to the respective content.
Convention	Bus master
Diagnostic class	DK4-low/DK4V-low
Session	APP: 0x01 (R), 0x03 (R), 0x40 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[NumberOfExpectedSubSystemIdentification] 1-byte hex> + <[NumberOfRetrievedSubSystemIdentification] 1-byte hex> + <n × <[SubSystemNodeAddress] 2-byte hex> + <[Programming_hash] 32-byte hex> > (n: Number of DK2F/DK2FV 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
eBZD	NoImp

5.1.8 0x0245-Configuration_hash

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

Table 5-5: Structure of data record for the 0x0245-Configuration_hash data identifier

DID	**0x0245**
Designation	Configuration_hash
Description	This data identifier contains the integrity validation data for the configuration data. This routine identifier must be readable while the vehicle is stationary and in the state "not ready for driving." ECU-specific boundary conditions may apply in other vehicle states (e.g., active control interventions). The ready-for-driving and speed signals must be used on an architecture-specific basis as per the valid data definition.
Convention	All servers
Diagnostic class	DK2/DK2F/DK2FV
Session	APP: 0x01 (R), 0x03 (R), 0x40 (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
eBZD	M

5.1.9 0x0248-Slave_list_configuration_hash

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

Table 5-6: Structure of data record for the 0x0248-Slave_list_configuration_hash data identifier

DID	**0x0248**
Designation	Slave_list_configuration_hash
Description	This data identifier contains all integrity validation data for the configuration data of the lower-level DK2/DK2F/DK2FV systems. The following definitions apply: • The respective content corresponds to the [Configuration_hash] parameter of the "0x0245-Configuration_hash" data identifier. • The respective SubSystemNodeAddress must be prepended to the respective content.
Convention	Bus master
Diagnostic class	DK4-low/DK4V-low
Session	APP: 0x01 (R), 0x03 (R), 0x40 (R) BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	<[NumberOfExpectedSubSystemIdentification] 1-byte hex> + <[NumberOfRetrievedSubSystemIdentification] 1-byte hex> + <n × <[SubSystemNodeAddress] 2-byte hex> + <[Configuration_hash] 32-byte hex> > (n: Number of DK2/DK2F/DK2FV 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
eBZD	NoImp

5.1.10 0xF18F-Regulation_x_software_identification_numbers

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

Table 5-7: Structure of data record for the 0xF18F-Regulation_x_software_identification_numbers data identifier

DID	**0xF18F**
Designation	Regulation_x_software_identification_numbers
Description	This data identifier contains the list of software identification numbers from, e.g., UNECE, or the GB/T functions defined in Chinese standards, that are available in a vehicle. This list is vehicle-specific. All payload contents, such as [Length_of_RxSWIN], [Regulation_identification], [Separation_character], and [Software_identification], must not be validation checked by the diagnostic server, and must be output without being changed.
Convention	ECU with central diagnostic access
Diagnostic class	DK4-high
Session	APP: 0x01 (R), 0x03 (R/W), 0x40 (R/W) BLF: NoImp
SecurityLevel	SFD E2E (W)
Changing	DIAG
Format	<n × <[Length_of_RxSWIN] 1-byte hex> + <[Regulation_identification] m-byte ASCII, variable> + <[Separation_character] 1-byte ASCII> + <[Software_identification] 1 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 to 7Ahex [Separation_character] 20hex (ASCII "SPACE") [Software_identification] 30 to 39hex, 41 to 5Ahex, 61 to 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 parameters
eBZD	M

[I: F-LAH_RxSWIN-70]

The total length of the RxSWIN must be at least 50 regulations; it must be able to contain the associated regulation-specific software identification data.

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

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

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

Reading of the "0xF18F-Regulation_x_software_identification_numbers" data identifier via the ReadDataByIdentifier(22hex) service must not be protected by means of access protection methods.

[allg. Anf.: F-LAH_RxSWIN-1076]

Writing of the "0xF18F-Regulation_x_software_identification_numbers" data identifier must be performed as per document /4/ by the SFD E2E validation.

5.1.10.1 Requirements for processes

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

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

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

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

[I: F-LAH_RxSWIN-555]

Note:
This can be implemented, e.g., by a JAVA or OTX job.

5.1.10.2 Requirements for IT systems

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

The data container with the list of RxSWINs is introduced in Data Logistics by a diagnostic DK2V system assigned to the gateway ECU.

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

Note: This is not a DK2V system in the sense of documents /3/ or /12/. This DK2V system merely contains a diagnostic address required to manage the data in the logistics chain.

5.1.11 0x0250-Integrity_validation_data_configuration_list

[allg. Anf.: F-LAH_RxSWIN-1061]

ECUs that implement the integrity validation data for programming and/or for configuration, must implement the "0x0250-Integrity_validation_data_configuration_list" data identifier.

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

Table 5-8: Structure of data record for the 0x0250-Integrity_validation_data_configuration_list data identifier

DID	**0x0250**
Designation	Integrity_validation_data_configuration_list
Description	This data identifier contains the list of identifiers (data identifiers and data set numbers) used for calculating the configuration data hash value.
Convention	ZDC
Diagnostic class	2/2F/2FV, 3/3V, 4/4V, software cluster
Session	APP: 0x01 (R), 0x03 (R/W*)), 0x40 (R/W*)) BLF: NoImp
SecurityLevel	SFD-E2E *)
Changing	DIAG
Format	<<[Number_of_identifiers] 2-byte hex> + n × <[Identifier] 2-byte hex>> (n: Number of identifiers)
Range	[Number_of_identifiers] 00 00hex = No data identifier or no data set number available, e.g., if only IVD is supported for programming. 00 01hex - FF FFhex [Identifier] 00 00 - FF FFhex
Init	00 00hex
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
eBZD	M

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

*) Access protection is not required if the ECU does not support SFD E2E validation.

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

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

[allg. Anf.: F-LAH_RxSWIN-1062]

If the ECU implements the configuration hash value, the value of data identifier 0x0250 itself must also be included in the "0x0250-Integrity_validation_data_configuration_list" data identifier.

[allg. Anf.: F-LAH_RxSWIN-1063]

If the ECU implements only the programming hash value, the value of data identifier 0x0250 itself must not be included in the "0x0250-Integrity_validation_data_configuration_list" data identifier.

[allg. Anf.: F-LAH_RxSWIN-1065]

A WriteDataByIdentifier (2Ehex) request with the "0x0250-Integrity_validation_data_configuration_list" data identifier may only receive a positive response if all data identifiers and data set numbers in the ECU are identified as configuration data. Otherwise, an NRC 0x31 "RequestOutOfRange" must be used to reject the request.

5.1.12 0x0251-Write_generic_to_sub_system (writing of the 0x0250 data identifier to the lower-level DK2/DK2F/DK2FV system via the DK4-low/DK4V-low system)

[I: F-LAH_RxSWIN-494]

The "0x0250-Integrity_validation_data_configuration_list" data identifier is written via the DK4-low system using the WriteDataByIdentifier (2Ehex) service and the "0x0251-Write_generic_to_sub_system" data identifier. The SubSystemNodeAddress is used at the first two bytes in the data record in the 2Ehex request to address the lower-level target system. The next two bytes contain the 0x0250 data identifier followed by the payload.
Note: This has changed compared with Q-LAH 80124 up to v2.8 (Q-LAH_80124-7984).

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

Table 5-9: Structure of data record for the 0x0251-Write_generic_to_sub_system data identifier

DID	**0x0251**
Designation	Write_generic_to_sub_system
Description	This data identifier is used to write data for sub-systems via a DK4-low system. It is not possible to read data using this data identifier. (write only)
Convention	Bus master
Diagnostic class	DK4/DK4-low
Session	APP: 0x03 (W), 0x40 (W) BLF: NoImp
SecurityLevel	ECUs in existing architectures: If available, the DK4-low's existing access protection method must be used. ECUs in new architectures: SFD E2E (W)
Changing	DIAG
Format	<<[Target_sub_system_node_address] 2-byte hex> + <[Target_data_identifier] 2-byte hex> + <n × [Data_record_target_data_identifier] 1-byte hex>> (n: 1 to 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 00 02 13 24 97 86 Target_sub_system_node_address: 0x0141 = TV tuner card reader Target_data_identifier: 0x02 50 Data_record_target_data_identifier: 0x00 02 13 24 97 86
Data category	Process parameters
eBZD	NoImp

5.1.12.1 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

5.1.12.2 Request message parameter definition

[I: F-LAH_RxSWIN-497]

The following parameters must be implemented:

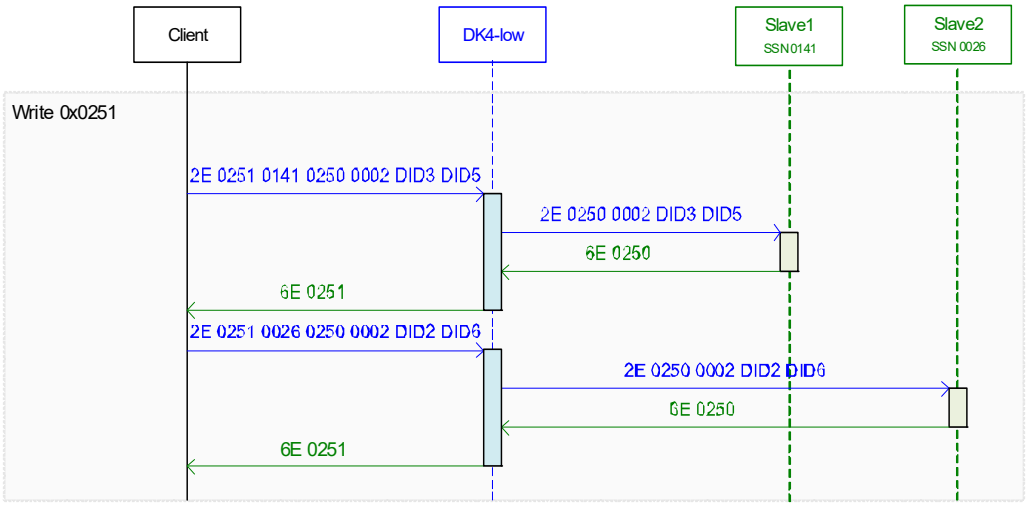
[allg. Anf.: F-LAH_RxSWIN-498]

Table 5-11: Request message parameter definition

Definition
Target_sub_system_node_address This parameter indicates the SubSystemNodeAddress of the lower-level target system.
Target_data_identifier This parameter indicates the value of the data identifier. 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 Routine identifiers

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

Table 5-12: Overview of the routine identifier properties

			Diagnostic session			
			Default session	Non-default session		
				ECU programming session	Extended diagnostic session	Volkswagen end-of-line session
Diagnostic objects (RIDs)	SID	Security level	0x01	0x02	0x03	0x03
0x0366-Reset_of_all_adaptions	31hex	SFD-Basic	NoImp	NoImp	Available	C1
0x03E7-Reset_to_factory_setting	31hex	SFD-Basic	NoImp	NoImp	Available	C1
0xC008-Request_data_of_protection_of_vehicle_diagnosis	31hex	NoImp	Available	NoImp	Available	C1
0x0253-Calculate_integrity_validation_data	31hex	NoImp	Available	notAvailable	Available	C1
0x0254-Calculate_individual_hash_value	31hex	NoImp	Available	notAvailable	Available	C1
0x029A-Calculate_module_hash_value	31hex	NoImp	Available	C2	Available	C1

[I: F-LAH_RxSWIN-1067]

C1 – Only available for diagnostic servers that implement document /1/.

[I: F-LAH_RxSWIN-1124]

C2 – Mandatory requirement for DK2F/DK2FV systems

5.2.1 Resetting configuration parameters

[I: F-LAH_RxSWIN-945]

Table 5-13: Overview of reset capability of data (categories)

	Programming hash	Configuration hash	Reset via RID	
			0x0366	0x03E7
Programmdaten (Program data)	x	NoImp	NoImp	NoImp
Applikationsdaten (Calibration data)	x	NoImp	NoImp	NoImp
Codierung (Coding)	NoImp	x	NoImp	x
Fahrzeugparameter (Vehicle parameters)	NoImp	x	NoImp	x
Erstbedatungswerte (Initial calibration values)	NoImp	x	NoImp	x
Kundenparameter (Customer parameters)	NoImp	NoImp	x	x
Werkstattparameter (Workshop parameters)	NoImp	NoImp	x	x
Prozessparameter (Process parameters)	NoImp	NoImp	NoImp	x
Lernwerte (Learned values)	NoImp	NoImp	NoImp	x
Analysedaten (Analysis data)	NoImp	NoImp	NoImp	NoImp

[I: F-LAH_RxSWIN-972]

X = included

5.2.1.1 0x0366-Reset_of_all_adaptions

[I: F-LAH_RxSWIN-970]

This routine identifier is used as per documents /1/ and /8/ to reset customer and workshop parameters to the factory defaults.

[allg. Anf.: F-LAH_RxSWIN-971]

The following change compared with Q-LAH 80125 v5.3 to v5.5 must be observed:

- SFD access protection (Basic role) for routine identifier 0x0366

5.2.1.2 0x03E7-Reset_to_factory_setting

[I: F-LAH_RxSWIN-927]

This routine identifier is used as per documents /1/ and /8/ to reset all codings, vehicle parameters, customer parameters, workshop parameters, process parameters, initial calibration values, and learned values to the factory defaults (condition as delivered by supplier).
Note: Executing the routine causes a change to the IVD-relevant portions of the configuration hash value, such as codings, vehicle parameters, and initial calibration values. After executing this routine, it is absolutely mandatory to recalibrate using a ZDC.

[allg. Anf.: F-LAH_RxSWIN-967]

The following change compared with Q-LAH 80125 v5.3 to v5.5 must be observed:

- SFD access protection (Basic role) for routine identifier 0x03E7

5.2.1.3 0xC008-Request_data_of_protection_of_vehicle_diagnosis

[I: F-LAH_RxSWIN-1059]

This routine identifier is used as per document /4/ to read the SFD request structure.

[allg. Anf.: F-LAH_RxSWIN-1060]

The following change compared with document /4/ must be observed:

- Deviating from requirement *[Server-Anf.: Q-LAH_SFD_1068]* in document /4/, the "0xC008-Request_data_of_protection_of_vehicle_diagnosis" routine must also be executable in the default session.

5.2.2 Calculating the integrity validation data

5.2.2.1 0x0253-Calculate_integrity_validation_data

[I: F-LAH_RxSWIN-451]

This routine identifier is used to calculate the integrity validation data of DK3/DK3V/DK4/DK4V/software cluster systems for the instruction code and configuration data.

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

This routine identifier must be implemented with the request/response behavior and parameters specified here. It returns its result as the positive response to RoutineControlType 0x01, and terminates automatically. Use of the sub-functions

- StopRoutine (0x02)
- RequestRoutineResults (0x03)

is not permissible for this routine identifier.

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

The response behavior described here corresponds to Q-LAH 80124 v2.x. ECUs in existing architectures that implement a version of Q-LAH 80124 v1.x, must implement the response required here for the "0x0253-Calculate_integrity_validation_data" routine identifier. Existing routines based on Q-LAH 80124 v1.x must not be adapted on the ECU.

[allg. Anf.: F-LAH_RxSWIN-965]

This routine identifier must be executable while the vehicle is stationary and in the state "not ready for driving." ECU-specific boundary conditions may apply in other vehicle states (e.g., active control interventions). The ready-for-driving and speed signals must be used on an architecture-specific basis as per the valid data definition.

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

Table 5-14: Structure of the 0x0253-Calculate_integrity_validation_data routine identifier

RID	**0x0253**
Designation	Calculate_integrity_validation_data
Description	This routine identifier is used to calculate the integrity validation data for the instruction code and configuration data.
Convention	All servers
Diagnostic class	DK3/DK3V, DK4/DK4-low/DK4V-low, DK4-high/DK4V-high, software cluster
Session	APP: 0x01, 0x03, 0x40 BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	See request/response definition
Range	See request/response definition
Init	No init on the server
Example	See request/response definition
eBZD	M

5.2.2.1.1 Request message definition

[I: F-LAH_RxSWIN-453]

The following request message must be implemented:

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

Table 5-15: 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_hash_value	M	00-FF

5.2.2.1.2 Request message parameter definition

[I: F-LAH_RxSWIN-457]

The following parameters must be implemented:

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

Table 5-16: Request message parameter definition

Definition
Type_of_calculation This parameter indicates the scope of hash value calculation. 0x00 = The hash value is calculated for the configuration data (data). The identifier information is taken from data identifier 0x0250. 0x01 = The hash value is calculated for the programming data of the application and bootloader software (instruction code). Other values are reserved by Volkswagen AG.
Type_of_hash_value This parameter indicates the hash value calculation method. 0x01 = SHA-256 Other values are reserved by Volkswagen AG.

5.2.2.1.3 Positive response message definition

[I: F-LAH_RxSWIN-460]

The following response message must be implemented:

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

Table 5-17: 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	Type_of_hash_value	M	00-FF
#7	RoutineStatusRecord#3	Hash_value [byte#1] (MSB)	C1	00-FF
:	:	:	:	:
#6+n	RoutineStatusRecord#2+n	Hash_value [byte#n]	C1	00-FF

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

C1 = Length depends on [Result_of_calculation] parameter:

- 32 bytes = If a SHA-256 hash value is successfully calculated (Type_of_hashvalue = 0x01)

- 0 byte = In case of an error (value: 0x01, 0x02, 0x03, 0x04, 0x05)

5.2.2.1.4 Positive response message parameter definition

[I: F-LAH_RxSWIN-464]

The following parameters must be implemented:

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

Table 5-18: Positive response message parameter definition

Definition
Result_of_calculation 0x00 = Calculation_successful The hash value was successfully calculated. 0x01 = Calculation_failed Calculation of the hash value failed. Note: A value of 0x01 must only be output if values 0x02 through 0x05 are not true. 0x02 = Calculation_identifier_not_found At least one data identifier or one data set number from data identifier 0x0250 does not exist on the diagnostic server. 0x03 = Calculation_no_identifier_found Data identifier 0x0250 does not contain any data identifiers or data set numbers (e.g., reserve ZDC) 0x04 = Calculation_incomplete A complete check result is not available when calculating the programming hash values for an LUM flash container. 0X05 - 0xFF = Reserved for Volkswagen AG
Type_of_hash_value 0x00 = Reserved by Volkswagen AG 0x01 = SHA-256 The hash value is calculated using the SHA-256 method. 0X02 - 0xFF = Reserved by Volkswagen AG
Hash_value n bytes <0x00 - 0xFF> Calculated hash value; n = 32 for SHA-256

5.2.2.2 0x0254-Calculate_individual_hash_value

[I: F-LAH_RxSWIN-868]

This routine identifier is used to calculate and read the individual configuration data hash values of DK3/DK3V/DK4/DK4V systems and software clusters.

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

This routine identifier must be implemented with the request/response behavior and parameters specified here. Use of the sub-functions

- StopRoutine (0x02)
- RequestRoutineResults (0x03)

is not permissible for this routine identifier. It returns its result as the positive response to RoutineControlType 0x01, and terminates automatically.

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

The response behavior described here corresponds to Q-LAH 80124 v2.x. ECUs in existing architectures that implement a version of Q-LAH 80124 v1.x, must implement the response required here for the 0x0254-Calculate_individual_hash_value" routine identifier. Existing routines based on Q-LAH 80124 v1.x must not be adapted on the ECU.

[allg. Anf.: F-LAH_RxSWIN-1019]

This routine identifier must be executable while the vehicle is stationary and in the state "not ready for driving." ECU-specific boundary conditions may apply in other vehicle states (e.g., active

control interventions). The ready-for-driving and speed signals must be used on an architecture-specific basis as per the valid data definition.

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

Table 5-19: Structure of the 0x0254-Calculate_individual_hash_value routine identifier

RID	**0x0254**
Designation	Calculate_individual_hash_value
Description	This routine identifier is used to calculate and read the individual configuration data hash values.
Convention	All servers
Diagnostic class	DK3/DK3V, DK4/DK4-low/DK4V-low, DK4-high/DK4V-high, software cluster
Session	APP: 0x01, 0x03, 0x40 BLF: NoImp
SecurityLevel	NoImp
Changing	APP
Format	See request/response definition
Range	See request/response definition
Init	No init on the server
Example	See request/response definition
eBZD	NoImp

5.2.2.2.1 Request message definition

[I: F-LAH_RxSWIN-873]

The following request message must be implemented:

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

Table 5-20: 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_hash_value	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

5.2.2.2.2 Request message parameter definition

[I: F-LAH_RxSWIN-876]

The following parameters must be implemented:

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

Table 5-21: Request message parameter definition

Definition
Type_of_hash_value 0x00 = Reserved by Volkswagen AG 0x01 = SHA-256 The hash value is calculated using the SHA-256 method. 0X02 - 0xFF = Reserved by Volkswagen AG
Type_of_hash 0x00 = Individual hash value for all data sets (for Data Set Download Generation 1) 0x01 = Individual hash value for all bootloader data sets (for Data Set Download Generation 2) 0x02 = Individual application data set hash value (equivalent to the application data set number for Data Set Download Generation 2) 0x03 = Individual hash value for all adaptations/codings as per data identifier 0x0250 0x04 – 0xFE = Reserved by Volkswagen AG 0xFF = All individual hash values as per data identifier 0x0250
Individual_hash_value_id This parameter identifies individual hash values from application data sets as per data identifier 0x0250. 0x7200 - 0x72FF = Only used for [Type_of_hash] = 0x02 0x0000 = For all other values in the [Type_of_hash] parameter All other values are reserved by Volkswagen AG.

5.2.2.2.3 Positive response message definition

[I: F-LAH_RxSWIN-879]

The following response message must be implemented:

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

Table 5-22: 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_hash_value	M	00-FF
#6	RoutineStatusRecord#2	Type_of_hash_value	M	00-FF
#7	RoutineStatusRecord#3	Type_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 = This only exists if an individual hash value is available.

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

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

5.2.2.2.4 Positive response message parameter definition

[I: F-LAH_RxSWIN-883]

The following parameters must be implemented:

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

Table 5-23: Positive response message parameter definition

Definition
State_of_hash_value 0x00 = Calculation_hash_value_valid The hash value was successfully calculated and the hash value is valid. 0x01 = Calculation_hash_value_invalid Calculation of the hash value failed and the hash value is invalid. 0x02 = Calculation_hash_value_not_found The data identifier for the hash value is invalid and does not exist in data identifier 0x0250 on the diagnostic server. 0X03 - 0xFF = Reserved for Volkswagen AG
Type_of_hash_value 0x00 = Reserved by Volkswagen AG 0x01 = SHA-256, The hash value is calculated using the SHA-256 method. 0X02 - 0xFF = Reserved by Volkswagen AG
Type_of_hash 0x00 = Individual hash value for all data sets (for Data Set Download Generation 1) 0x01 = Individual hash value for all bootloader data sets (for Data Set Download Generation 2) 0x02 = Individual application data set hash value (equivalent to the application data set number for Data Set Download Generation 2) 0x03 = Individual hash value for all adaptations/codings as per data identifier 0x0250 0x04 – 0xFE = Reserved by Volkswagen AG 0xFF = All individual hash values as per data identifier 0x0250
Individual_hash_value_id This parameter identifies individual hash values from application data sets as per data identifier 0x0250. 0x7200 - 0x72FF = Only used for [Type_of_hash] = 0x02 0x0000 = For all other values in the [Type_of_hash] parameter All other values are reserved by Volkswagen AG.
Hash_value n bytes <0x00 - 0xFF> Calculated hash value for SHA-256: n = 32

5.2.2.3 0x029A-Calculate_module_hash_value

[I: F-LAH_RxSWIN-1021]

This routine identifier is used for completely recalculating the integrity validation data for the instruction code in non-embedded or file-based DK2F/DK2FV/DK3/DK3V/DK4/DK4V/software cluster systems.

[allg. Anf.: F-LAH_RxSWIN-1022]

This routine identifier must be implemented with the request/response behavior and parameters specified here. A positive response from this routine identifier to RoutineControlType 0x01 does not provide a final result but starts the complete recalculation of the programming hash value. This routine does not end automatically. Use of the sub-functions

- StopRoutine (0x02)
- RequestRoutineResults (0x03)

must be implemented for this routine identifier as a mandatory requirement.

[allg. Anf.: F-LAH_RxSWIN-1023]

The response behavior described here corresponds to Q-LAH 80124 v2.x. ECUs in existing architectures that implement a version of Q-LAH 80124 v1.x, must implement the response required here for the " 0x029A-Calculate_module_hash_value" routine identifier. Existing routines based on Q-LAH 80124 v1.x must not be adapted on the ECU.

[allg. Anf.: F-LAH_RxSWIN-1024]

This routine identifier must be executable while the vehicle is stationary and in the state "not ready for driving." ECU-specific boundary conditions may apply in other vehicle states (e.g., active control interventions). The ready-for-driving and speed signals must be used on an architecture-specific basis as per the valid data definition.

[allg. Anf.: F-LAH_RxSWIN-1025]

Table 5-24: Structure of the 0x029A-Calculate_module_hash_value routine identifier

RID	**0x029A**
Designation	Calculate_module_hash_value
Description	This routine identifier is used to completely recalculate the integrity validation data for instruction code in case of file-based (non-embedded) systems.
Convention	All servers
Diagnostic class	DK2F/DK3/DK3V, DK4/DK4-low/DK4V-low, DK4-high/DK4V-high, software cluster
Session	DK3/DK3V, DK4/DK4-low/DK4V-low, DK4-high/DK4V-high, software cluster: APP: 0x01, 0x03, 0x40 DK2F/DK2FV: APP: 0x01 (U), 0x03 (U), 0x40 (U) BLF: 0x01, 0x02, 0x03
SecurityLevel	NoImp
Changing	APP
Format	See request/response definition
Range	See request/response definition
Init	No init on the server
Example	See request/response definition
eBZD	NoImp

5.2.2.3.1 StartRoutine (0x01)

5.2.2.3.1.1 Request message definition

[I: F-LAH_RxSWIN-1028]

The following request message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1029]

Table 5-25: Request message definition

Data	Description		Cvt.	Value (hex)
#1	Request SID	RoutineControl	M	31
#2	RoutineControlType	StartRoutine	M	01
#3	RoutineIdentifier [byte#1]	Calculate_module_hash_value [byte#1] (MSB)	M	02
#4	RoutineIdentifier [byte#2]	Calculate_module_hash_value [byte#2]	M	9A
#5	RoutineControlOption [byte#1]	Type_of_hash_value	M	00-FF
#6	RoutineControlOption [byte#2]	Type_of_calculation	M	00-FF

5.2.2.3.1.2 Request message parameter definition

[I: F-LAH_RxSWIN-1032]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1033]

Table 5-26: Request message parameter definition

Definition
Type_of_hash_value This parameter indicates the hash value calculation method. 0x01 = SHA-256 Other values are reserved by Volkswagen AG.
Type_of_calculation This parameter indicates the scope of the hash value calculation. 0x01 = The hash value is calculated using the programming data for completely recalculating the programming hash value. Other values are reserved by Volkswagen AG.

5.2.2.3.1.3 Positive response message definition

[I: F-LAH_RxSWIN-1035]

The following response message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1036]

Table 5-27: 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_module_hash_value [byte#1] (MSB)	M	02
#4	RoutineIdentifier [byte#2]	Calculate_module_hash_value [byte#2]	M	9A
#5	RoutineInfo	RoutineInfo as per document /1/	M	XX
#6	RoutineStatusRecord#1	Type_of_hash_value	M	00-FF
#7	RoutineStatusRecord#2	Type_of_calculation	M	00-FF

5.2.2.3.1.4 Positive response message parameter definition

[I: F-LAH_RxSWIN-1038]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1039]

Table 5-28: Request message parameter definition

Definition
Type_of_hash_value This parameter indicates the hash value calculation method. 0x01 = SHA-256 Other values are reserved by Volkswagen AG.
Type_of_calculation This parameter indicates the scope of the hash value calculation. 0x01 = The hash value is calculated using the programming data for completely recalculating the programming hash value. Other values are reserved by Volkswagen AG.

5.2.2.3.2 StopRoutine (0x02)

5.2.2.3.2.1 Request message definition

[I: F-LAH_RxSWIN-1041]

The following request message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1042]

Table 5-29: Request message definition

Data	Description		Cvt.	Value (hex)
#1	Request SID	RoutineControl	M	31
#2	RoutineControlType	StopRoutine	M	02
#3	RoutineIdentifier [byte#1]	Calculate_module_hash_value [byte#1] (MSB)	M	02
#4	RoutineIdentifier [byte#2]	Calculate_module_hash_value [byte#2]	M	9A

5.2.2.3.2.2 Positive response message definition

[I: F-LAH_RxSWIN-1044]

The following response message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1045]

Table 5-30: Positive response message definition

Data	Description		Cvt.	Value (hex)
#1	Response SID	RoutineControl	M	71
#2	RoutineControlType	StopRoutine	M	02
#3	RoutineIdentifier [byte#1]	Calculate_module_hash_value [byte#1] (MSB)	M	02
#4	RoutineIdentifier [byte#2]	Calculate_module_hash_value [byte#2]	M	9A

5.2.2.3.3 Request routine result (0x03)

5.2.2.3.3.1 Request message definition

[I: F-LAH_RxSWIN-1047]

The following request message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1048]

Table 5-31: Request message definition

Data	Description		Cvt.	Value (hex)
#1	Request SID	RoutineControl	M	31
#2	RoutineControlType	RequestRoutineResult	M	03
#3	RoutineIdentifier [byte#1]	Calculate_module_hash_value [byte#1] (MSB)	M	02
#4	RoutineIdentifier [byte#2]	Calculate_module_hash_value [byte#2]	M	9A

5.2.2.3.3.2 Positive response message definition

[I: F-LAH_RxSWIN-1054]

The following response message must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1055]

Table 5-32: Positive response message definition

Data	Description		Cvt.	Value (hex)
#1	Response SID	RoutineControl	M	71
#2	RoutineControlType	RequestRoutineResult	M	03
#3	RoutineIdentifier [byte#1]	Calculate_module_hash_value [byte#1] (MSB)	M	02
#4	RoutineIdentifier [byte#2]	Calculate_module_hash_value [byte#2]	M	9A
#5	RoutineInfo	RoutineInfo as per document /1/	M	XX
#6	RoutineStatusRecord#1	Progress_of_calculation	M	00-64; FF
#7	RoutineStatusRecord#2	Type_of_hash_value	M	00-FF
#8	RoutineStatusRecord#3	Type_of_calculation	M	00-FF
#9	RoutineStatusRecord#4	Result_of_calculation	M	00-FF
#10	RoutineStatusRecord#5	Hash_value [byte#1] (MSB)	C1	00-FF
:	:	:	:	:
#10+n	RoutineStatusRecord#5+n	Hash_value = [byte#n]	C1	00-FF

[allg. Anf.: F-LAH_RxSWIN-1057]

C1 = Length depends on [Result_of_calculation] parameter:
32 bytes = If a SHA-256 hash value is successfully calculated (Type_of_hashvalue = 0x01)
0 byte = In case of an error (value: ! = 0x00)

5.2.2.3.3.3 Request message parameter definition

[I: F-LAH_RxSWIN-1051]

The following parameters must be implemented:

[allg. Anf.: F-LAH_RxSWIN-1052]

Table 5-33: Request message parameter definition

Definition
Progress_of_caluclation Update progress of the installation package (block) currently being processed. 0 to 100% = progress of the update 0xFF = no information available
Type_of_hash_value This parameter indicates the hash value calculation method. 0x01 = SHA-256 Other values are reserved by Volkswagen AG.
Type_of_calculation This parameter indicates the scope of the hash value calculation. 0x01 = The hash value is calculated using the programming data for completely recalculating the programming hash value. Other values are reserved by Volkswagen AG.
Result_of_calculation 0x00 = Calculation_successful The hash value was successfully calculated. 0x01 = Calculation_failed Calculation of the hash value failed. Note: A value of 0x01 must only be output if values 0x02 through 0x05 are not true. 0x02 - 0x03 = reserved for Volkswagen AG 0x04 = Calculation_incomplete A complete check result is not available when calculating the programming hash values for an LUM flash container. 0X05 - 0xFF = Reserved for Volkswagen AG
Hash_value n bytes <0x00 - 0xFF> Calculated hash value for SHA-256: n = 32

6 RxSWIN-specific documentation

[I: F-LAH_RxSWIN-55]

Multiple ECU versions with their identification data and integrity validation data can be assigned to a single RxSWIN. The following data must be documented in the IT systems for each RxSWIN at a minimum.

6.1 Data for RxSWIN-specific documentation of a DK3/DK3V/DK4/DK4V system

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

- Gateway ECUs only: VIN from data identifier 0xF190
- Gateway ECUs only: List of RxSWINs from data identifier 0xF18F
- Volkswagen/Audi part number (VW Spare Part Number) from data identifier 0xF187
- Software version (VW Application Software Version Number) from data identifier 0xF189
- Hardware part number (VW ECU Hardware Number) from data identifier 0xF191
- List for hash value calculation of configuration data from data identifier 0x0250
- Optional: Hardware version (VW ECU Hardware Version Number) from data identifier 0xF1A3
- Optional: Part number for the ZDC(s) (VW Data Set Number Or ECU Data Container Number)
- Optional: Version of the ZDC(s) (VW Data Set Version Number)
- FAZIT identification (FAZIT Identification String) from data identifier 0xF17C
- Integrity validation data for programming (Programming_hash) from routine identifier 0x0253
- Integrity validation data for configuration (Configuration_hash) from routine identifier 0x0253
- Diagnostic address (DA)

6.2 Data for the RxSWIN-specific documentation of a software cluster

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

- Volkswagen/Audi part number (VW Spare Part Number) from data identifier 0xF187
- Software version (VW Application Software Version Number) from data identifier 0xF189
- List for hash value calculation of configuration data from data identifier 0x0250
- Optional: Part number for the ZDC(s) (VW Data Set Number Or ECU Data Container Number)
- Optional: Version of the ZDC(s) (VW Data Set Version Number)
- Integrity validation data for programming (Programming_hash) from routine identifier 0x0253
- Integrity validation data for configuration (Configuration_hash) from routine identifier 0x0253
- Diagnostic address (DA)

6.3 Data for the RxSWIN-specific documentation of a DK2/DK2F/DK2FV system

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

- Volkswagen/Audi part number (VW Spare Part Number) from data identifier 0xF187 or 0x6200 to 63FF
- Hardware part number (VW ECU Hardware Number) from data identifier 0xF191 or 6600 to 67FF
- Software version (VW Application Software Version Number) from data identifier 0xF189 or 0x6400 to 65FF
- List for hash value calculation of configuration data from data identifier 0x0250
- Optional: Part number for the ZDC(s) (VW Data Set Number Or ECU Data Container Number)
- Optional: Version of the ZDC(s) (VW Data Set Version Number)

- Optional: Hardware version (VW ECU Hardware Version Number) from data identifier 0xF1A3 or 6800 to 69FF
- Optional: FAZIT identification (FAZIT Identification String) from data identifier 0xF17C or 6E00 to 6FFF
- For DK2F/DK2FV only: Integrity validation data for programming (Programming_hash) from data identifier 0x0249 or 0xA800 to A9FF
- Integrity validation data for configuration (Configuration_hash) from data identifier 0x0245 or AA00 to ABFF
- 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

[I: F-LAH_RxSWIN-75]

/5/ LAH.DUM.000.AD – Flash Data Security for UDS Electronic Control Units

[I: F-LAH_RxSWIN-76]

/6/ LAH.DUM.906.A – Q-LAH 80126 – UDS-Compliant Programming of Electronic Control Units

[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/ Diagnostic Filter Performance Specification

[I: F-LAH_RxSWIN-543]

/10/ Removed – 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 – Vehicle Key Management System Core Functionality

[I: F-LAH_RxSWIN-702]

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

[I: F-LAH_RxSWIN-703]

/17/ FoD-LAH – Features-on-Demand Vehicle Requirements

[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

[I: F-LAH_RxSWIN-1117]

/21/ ECU Manifest (Train release) Specification