# Specification of Module Vin

| | |
|---|---|
| Project | BMW AUTOSAR Core 4 Rel. 2 |
| Author | BMW AG |
| Release Date | 2017-02-23 |
| Version | 3.5.0 |
| Status | Released |
| Hotline | +49 89 382 - 32233 |
| Contact | bac@bmw.de<br>https://asc.bmw.com/jira/browse/BSUP (extern)<br>https://asc.bmwgroup.net/jira/browse/BSUP (intern) |

## Revision History

| Version | Date | Changed by | Description |
|---|---|---|---|
| 3.5.0 | 2017-02-23 | Björn Sachsenberg | No changes - only version update |
| 3.4.2 | 2016-10-27 | Björn Sachsenberg | No changes - only version update |
| 3.4.1 | 2016-08-25 | Björn Sachsenberg | No changes - only version update |
| 3.4.0 | 2016-03-17 | Björn Sachsenberg | Added SWS IDs starting with SWS_Vin_0121, changed SWS_Vin_0120 |
| 3.3.0 | 2015-12-11 | Björn Sachsenberg | No changes - only version update |
| 3.2.0 | 2015-07-10 | Björn Sachsenberg | Added SI adapter, added SWS IDs from SWS_Vin_0109 on |
| 3.1.0 | 2015-03-13 | Björn Sachsenberg | Added SSV functionality from Fscsm, added SWS IDs from SWS_Vin_0034 on |
| 3.0.0 | 2014-10-29 | Björn Sachsenberg | Initial Release for SP2018. |

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

# Table of Contents

# 1 Introduction and functional overview

The Vin module is used to request the VIN over the bus, set the qualifier and hand it over to application software components.

# 2 Acronyms and Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| AUTOSAR | Automotive Open System Architecture |
| BNDB | Bordnetzdatenbank BNE ist der aktuelle Begriff; BNDB ist veraltet, wird aber gelegentlich noch gebraucht. |
| BNE | Bord Netz Engineer |
| CAN | Controller Area Network |
| DTC | Diagnostic Trouble Code -> Fehlercode des Fehlerspeichereintrages |
| ECU | Electronic Control Unit |
| EFS | LH Eigenschafts- / Funktions- / Systemlastenheft |
| FAT | Flash-Absicherungs-Tool |
| FZG | Fahrzeug |
| GMT | Greenwich Mean Time |
| HW | Hardware |
| IEEE | Institute of Electrical and Electronics Engineers Technisches Normungsgremium |
| IETF | Internet Engineering Task Force Normungsgremium für Internet-Standards |
| IP | Internet Protocol Netzwerkebene des TCP/IP Protokolls |
| ISO/OSI | Schichtenmodell der Kommunikationsprotokolle |
| OS | Operating System |
| PTP | Precision Time Protocol |
| PWF | Parken Wohnen Fahren Energie-Management Konzept bei BMW. |
| RTE | Runtime Environment |
| SG | Steuergerät |
| SGID | Steuergeräte-ID, Diagnoseadresse, Steuergeräte-Adresse |
| SID | Service Identifier |
| SW | Software |
| SW-C | Software Component |
| UDS | Universal Diagnostic Services |
| VIN | Vehicle Identification Number |

All abbreviations used throughout this document -- except the ones listed here -- can be found in the official AUTOSAR glossary [5].

## 3.1 BMW Specifications

# 4 Constraints and assumptions

**[SWS_Vin_0001]** *d*There shall be only one Vin module available per ECU. *c*()

# 5 Dependencies to other Modules

## 5.1 Dlog

The Dlog module [1] is used to get the internal VIN.

## 5.2 Fscsm

The Fscsm module [2] is needed for receiving the secure VIN.

## 5.3 RTE

The module Vin is realized as a software component and is using RTE services [6] for client/server as well as sender/receiver communication to communicate with other SWCs.

# 6 Requirements traceability

The Requirements are taken from [3] and [4].

| Requirement | Description | Satisfied by |
|---|---|---|
| **[FP4_6292]** | No description | [SWS_Vin_0002] |
| **[FsCSM_1496]** | No description | [SWS_Vin_0048] |
| **[FsCSM_1497]** | No description | [SWS_Vin_0048] |
| **[FsCSM_1707]** | No description | [SWS_Vin_0039] |
| **[FsCSM_1708]** | No description | [SWS_Vin_0039] |
| **[FsCSM_1709]** | No description | [SWS_Vin_0041] |
| **[FsCSM_1710]** | No description | [SWS_Vin_0039] |
| **[FsCSM_391]** | No description | [SWS_Vin_0078] |
| **[FsCSM_4320]** | No description | [SWS_Vin_0048] |
| **[FsCSM_4336]** | No description | [SWS_Vin_0049] |
| **[FsCSM_4338]** | No description | [SWS_Vin_0050] |
| **[FsCSM_4418]** | No description | [SWS_Vin_0056] |
| **[FsCSM_4450]** | No description | [SWS_Vin_0068] |
| **[FsCSM_4451]** | No description | [SWS_Vin_0069] |
| **[FsCSM_4469]** | No description | [SWS_Vin_0055] |
| **[FsCSM_848]** | No description | [SWS_Vin_0048] |
| **[FsCSM_859]** | No description | [SWS_Vin_0048] |
| **[FsCSM_937]** | No description | [SWS_Vin_0048] |
| **[FsCSM_956]** | No description | [SWS_Vin_0048] |

# 7 Functional specification

## 7.1 Functional behavior

**[SWS_Vin_0002]** dThe VIN shall be requested according to FP4_6292 [3]. c(FP4_6292)

**[SWS_Vin_0003]** dThe Vin Qualifier shall be set according to Figure 7.1, “Vin Qualifier State”. c()

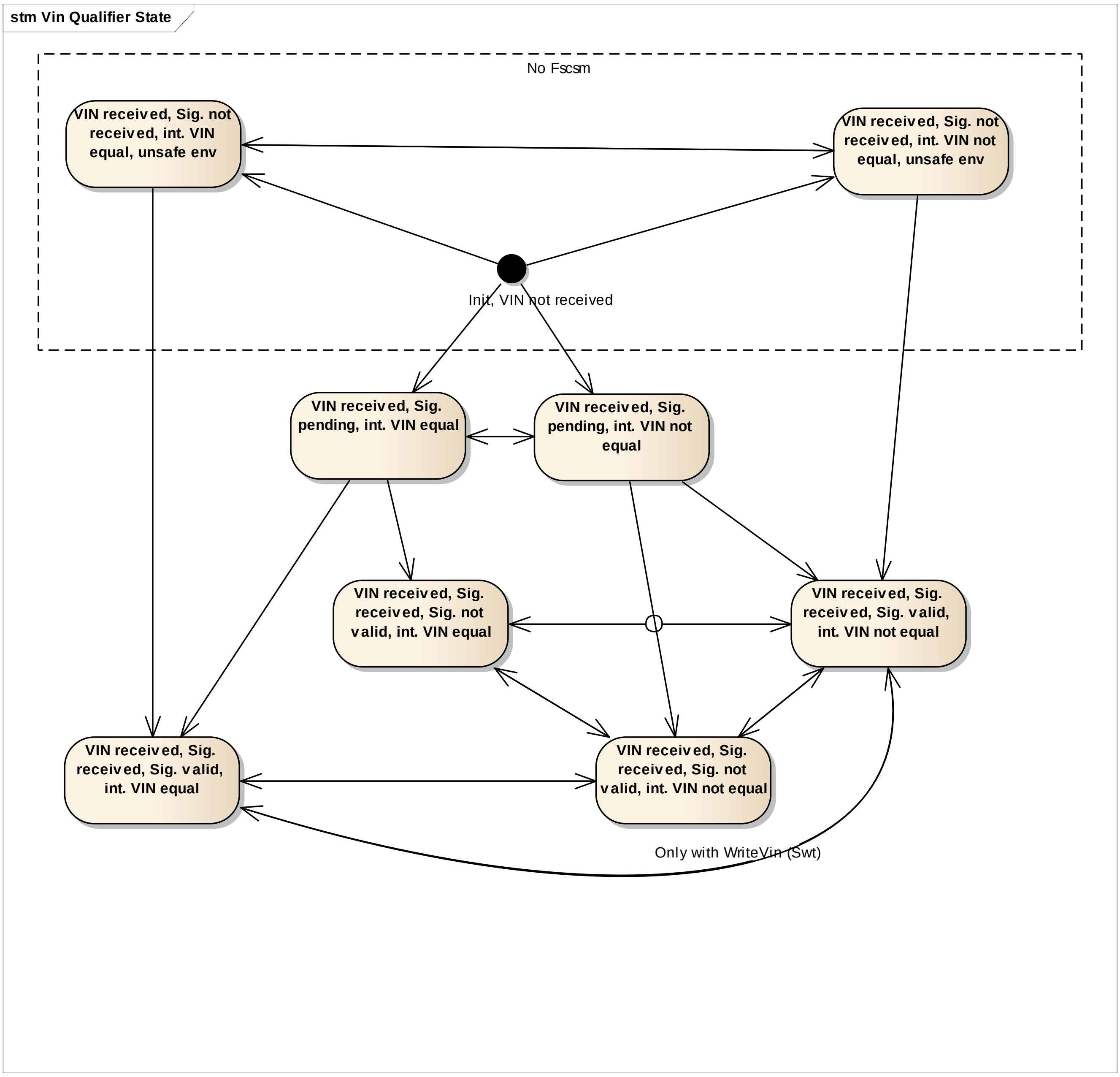

stm Vin Qualifier State
No Fscsm
VIN received, Sig. not received, int. VIN equal, unsafe env
VIN received, Sig. not received, int. VIN not equal, unsafe env
Init, VIN not received
VIN received, Sig. pending, int. VIN equal
VIN received, Sig. pending, int. VIN not equal
VIN received, Sig. received, Sig. not valid, int. VIN equal
VIN received, Sig. received, Sig. valid, int. VIN not equal
VIN received, Sig. received, Sig. valid, int. VIN equal
VIN received, Sig. received, Sig. not valid, int. VIN not equal
Only with WriteVin (Swt)

**Figure 7.1: Vin Qualifier State**

**[SWS_Vin_0034]** dThe general workflow is shall be done according to Figure 7.2, “SSV State Machine”. c()

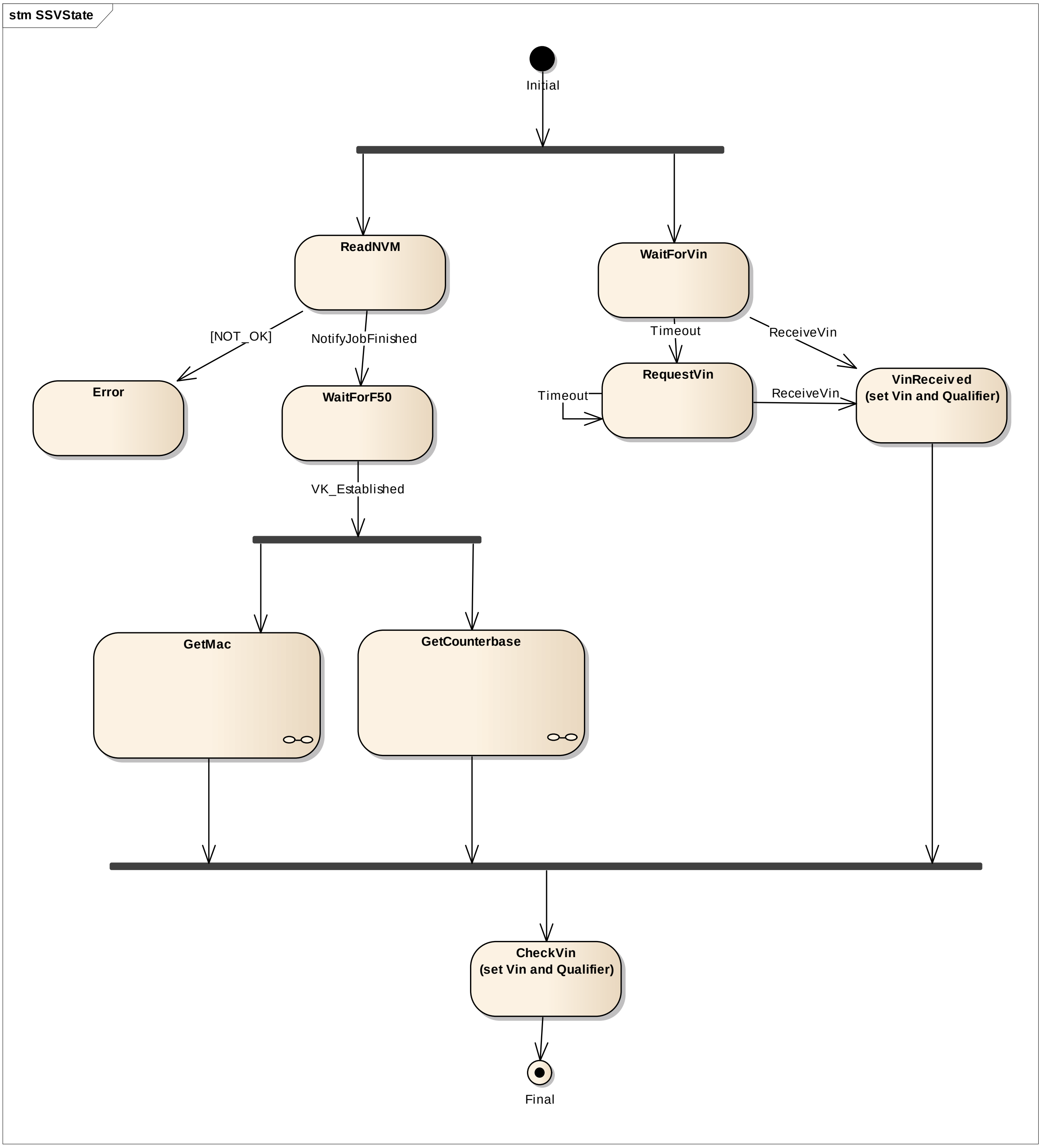


**Figure 7.2: SSV State Machine**

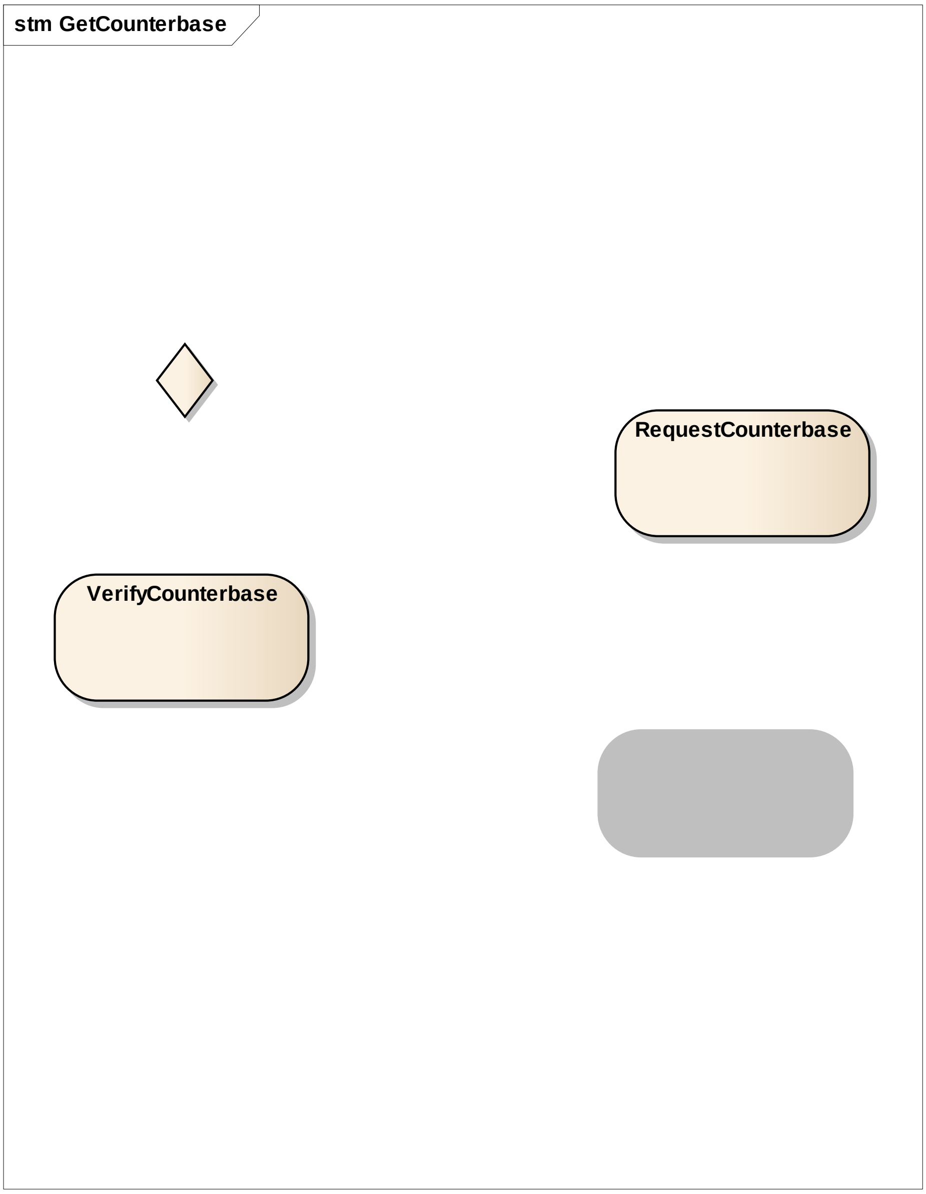


**Figure 7.3: Get Counterbase, SSV State Machine**

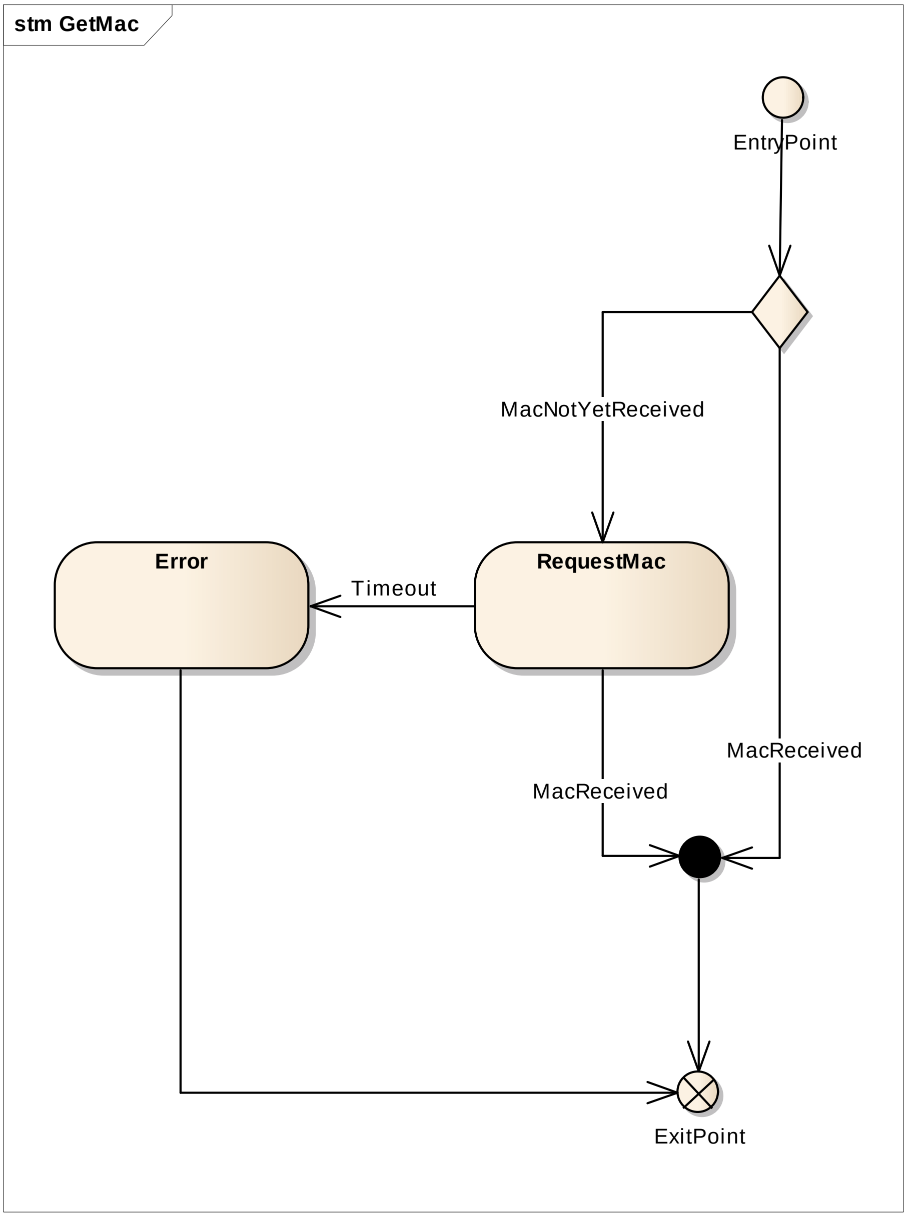


**Figure 7.4: Get Mac, SSV State Machine**

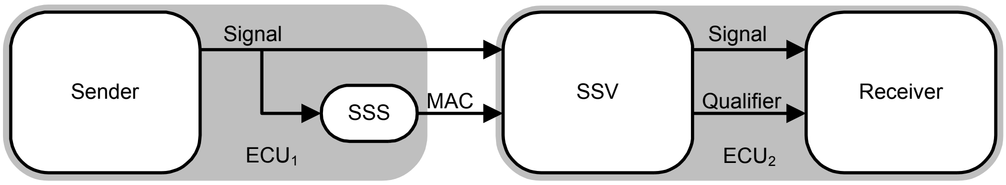
## 7.2 SSV

SSV is the receiver of the secure signal that shall be verified. It shall recognize that the transmitted signal was not manipulated or transmitted by an untrustworthy sender. The key necessary for SSV functionality are transmitted by the MSM with the functionality F50 of the Fscsm module, see [2] for details.

**[SWS_Vin_0035]** *d*A secure signal always consists of the actual (unprotected) signal and a Message Authentication Code (MAC), which is sent in a separate signal. A receiver can only classify a secure signal as secure when it has checked the MAC against the unprotected signal. *c*()

**[SWS_Vin_0036]** *d*The signal source (sender) does not know whether the signal is transferred securely.

- SSV receives both signal and MAC
- SSV always passes the actual signal and qualifier on to Application
- Task of SSV is to generate a correct qualifier for data



c()

**[SWS_Vin_0037]** *d*An SSV receives exactly one signed message and has a unique SSV-ID. *c*()

**[SWS_Vin_0038]** *d*The Fscsm only supports one SSV for the VIN. *c*()

### 7.2.1 Secure environment

**[SWS_Vin_0039]** *d*The SSV can be found in a secure or unsecured environment with regard to the signal it receives. SSV saves this state in a non-volatile flag. Initial value of the flag is SecEnv = false. *c*(FsCSM_1707, FsCSM_1708, FsCSM_1710)

**[SWS_Vin_0040]** *d*An SSV only then goes into the secure environment when it has received the secure signal successfully once, i.e. the SSV has the qualifier SecEnv_VerificationSuccess as a result after evaluation. *c*()

**[SWS_Vin_0041]** *d*Once a SSV is in a secure environment, it cannot be reset to an unsecured environment, even by programming the control unit again. *c*(FsCSM_1709)

### 7.2.2 Challenge/response and CounterBase

**[SWS_Vin_0042]** *d*For challenge/response, a common CounterBase is used between a SSS and its SSVs. CounterBase is defined in SSS. *c*()

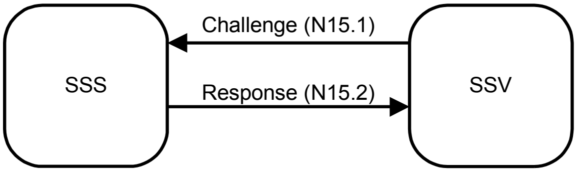
**[SWS_Vin_0043]** *d*A CounterBase has a size of 24 bits. *c*()

**[SWS_Vin_0044]** *d*If the SSV cannot verify a signal, it shall request the current CounterBase with challenge/response process immediately. *c*()

**[SWS_Vin_0045]** *d*If F50 was not successfully executed, the SSV does not execute the ChallengeResponse protocol. *c*()

**[SWS_Vin_0046]** *d*Challenge/response protocol is only needed by an SSV in a secure environment. *c*()

**[SWS_Vin_0047]** *d*Challenge/response process is done with N15.1 and N15.2.



*c*()

**[SWS_Vin_0048]** ⌈The messages for updating the counter base between a SSV and a SSS are described in [FsCSM_937], ff. ⌋(FsCSM_937, FsCSM_4320, FsCSM_956, FsCSM_848, FsCSM_1496, FsCSM_859, FsCSM_1497)

**[SWS_Vin_0049]** ⌈Operating sequence of challenge/response is described by [FsCSM_4336]: ⌋(FsCSM_4336)

**[SWS_Vin_0050]** ⌈If an SSV receives a response after a challenge is sent, it shall calculate the MAC and compares this with the received MAC. The counter base may only be considered valid and saved after successful verification. ⌋(FsCSM_4338)

**[SWS_Vin_0051]** ⌈If N15.1 message is sent and SSV receives no response after configured TimeoutCounterbase (see [SWS_Vin_0105]) or if verification of the received MAC fails, an error counter is incremented. ⌋()

**[SWS_Vin_0053]** ⌈The challenge/response mechanism shall be repeated until it has been executed successfully or the error counter has exceeded the configured value ErrorlimitCounterbase (see [SWS_Vin_0106]). In this case, an error memory entry with the error `ERC_FSCSM_SSV_ERROR_STATE` shall be set. ⌋()

**[SWS_Vin_0055]** ⌈Error counter is stored volatile. Initial value is 0. ⌋(FsCSM_4469)

### 7.2.3 Request/verify secure signal

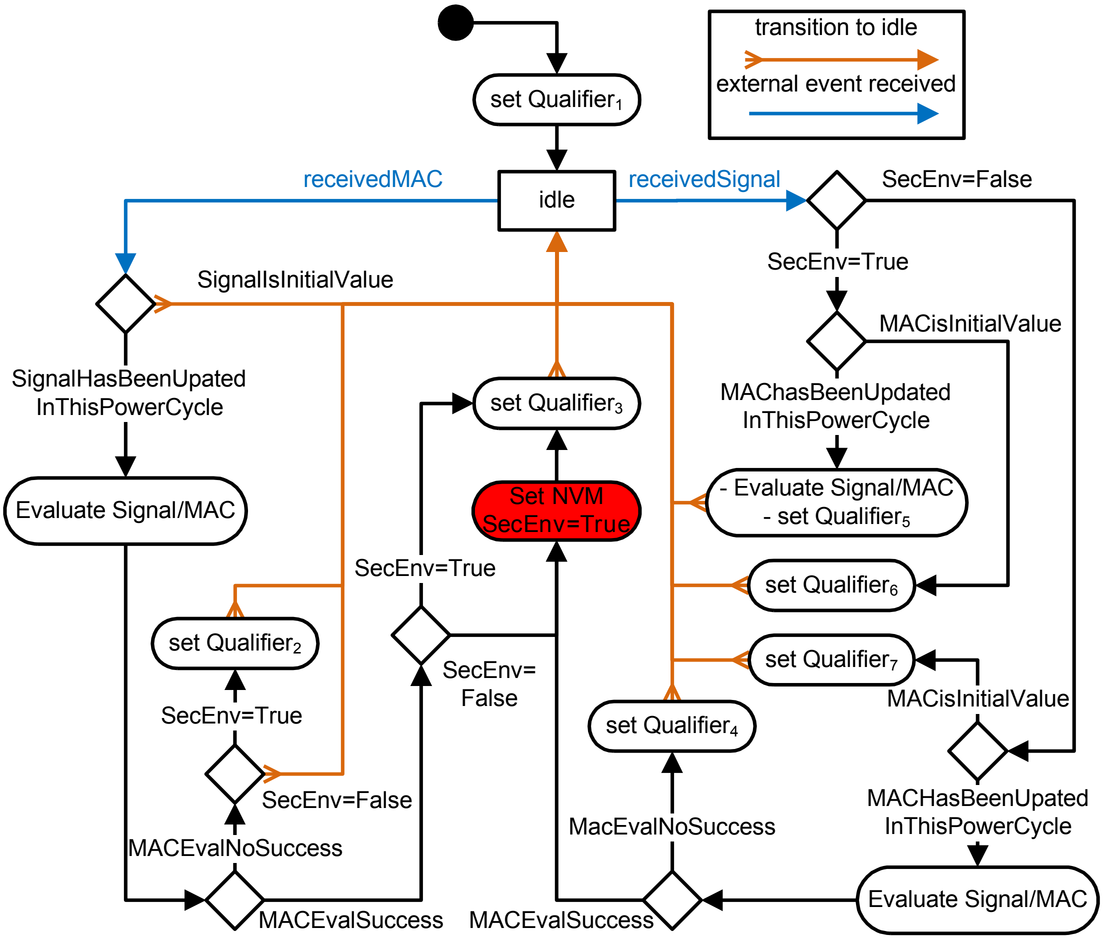
**[SWS_Vin_0056]** ⌈If functionality F50 is not successfully executed (flag VK_Established not set), SSV only requests the signal but not the MAC. In the other case it requests signal and MAC. ⌋(FsCSM_4418)

**[SWS_Vin_0057]** ⌈If SSV requires a MAC for a signal, it sends a request message N12.1 to SSS. ⌋()

**[SWS_Vin_0058]** ⌈If SSV in a secured environment requests a MAC and does not receive it after configured time (TimeoutMac), the SSV shall make an error memory entry with the error code `ERC_FSCSM_SSV_MESSAGE_TIMEOUT_REQ`. ⌋()

**[SWS_Vin_0059]** ⌈As long as an SSV has received neither a signal nor an MAC in a life cycle, signal qualifier shall be set to `FSCSM_SAFE_ENV_SSV_NOT_INITIALIZED`. ⌋()

**[SWS_Vin_0060]** ⌈In all cases, SSV passes the received signal to application. Meta information, which qualifies the signal (qualifier), shall be added. It is task of receiving application to use or not use the signal. ⌋()



| | | gültig bei setQualifier$_x$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Qualifier** | **Wert** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| NotInitialized | 0x00 | X | | | | | | |
| UnsecEnv | 0x01 | | | | X | | | X |
| SecEnv_Untrusted | 0x02 | | | | | | X | |
| SecEnv_VerificationSuccess | 0x03 | | | X | | X | | |
| SecEnv_VerificationFailed | 0x04 | | X | | | X | | |
| SecEnv_Error | 0x05 | | X | | | X | | |

**Figure 7.5: Evaluation of a secure signal by the SSV**

**[SWS_Vin_0061]** ⌈If SSV receives a plain signal and F50 was not executed yet, signal qualifier shall be set to `FSCSM_UNSAFE_ENV_MESSAGE_PLAIN` in an unsecure environment. ⌋()

**[SWS_Vin_0062]** ⌈If SSV receives a plain signal but no MAC and F50 was executed, signal qualifier shall be set to `FSCSM_SAFE_ENV_MESSAGE_PLAIN` in a secure enviroment. ⌋()

**[SWS_Vin_0063]** ⌈If SSV receives a plain signal and a correct MAC and F50 was executed, signal qualifier shall be set to `FSCSM_SAFE_ENV_MESSAGE_VERIFIED` in a secure enviroment. ⌋()

**[SWS_Vin_0064]** *d*If SSV receives a plain signal and an incorrect MAC and F50 was executed, signal qualifier shall be set to `FSCSM_SAFE_ENV_MESSAGE_NOT_VERIFIED` in a secure enviroment. *c*()

**[SWS_Vin_0065]** *d*If an error occurs while writing to NvM, signal qualifier shall be set to `FSCSM_SAFE_ENV_ERROR_STATE` in a secure enviroment. *c*()

**[SWS_Vin_0066]** *d*If an SSV receives a MAC and has not yet received an associated signal. nothing is sent to application. *c*()

**[SWS_Vin_0067]** *d*If an SSV receives a MAC and has yet received an associated signal. a verification shall be done. *c*()

**[SWS_Vin_0068]** *d*If the verification produces a negative result or if the SSV has not yet received a counter base, a new counter base shall be requested. *c*(FsCSM_4450)

**[SWS_Vin_0069]** *d*If the request for a new counter base produces a positive result, the signal verification against the MAC is started again. *c*(FsCSM_4451)

### 7.2.4 SSV ports and interfaces

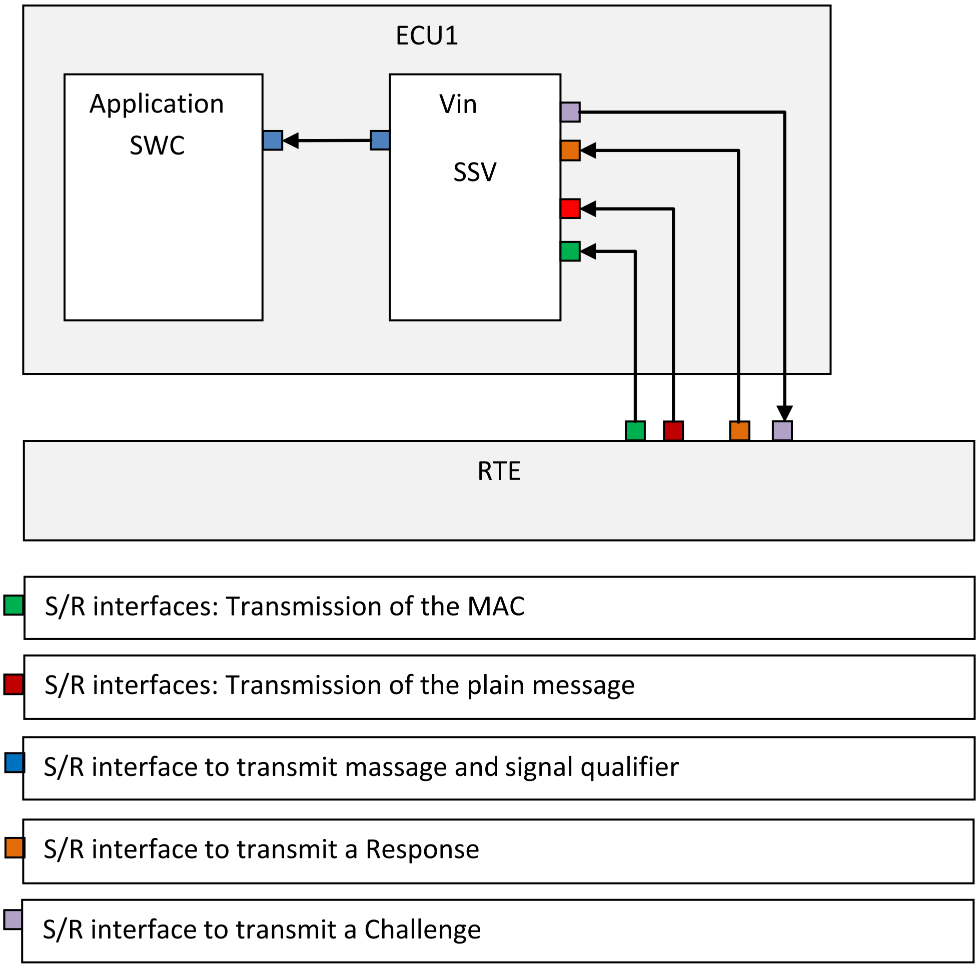

ECU1
Application
SWC
Vin
SSV
RTE
S/R interfaces: Transmission of the MAC
S/R interfaces: Transmission of the plain message
S/R interface to transmit massage and signal qualifier
S/R interface to transmit a Response
S/R interface to transmit a Challenge

**Figure 7.6: Overview of interfaces provided by SSV**

Figure 7.6, “Overview of interfaces provided by SSV” shows the interfaces provided by Fscsm functionality SSV:

- port to receive an unsigned message
- port to receive the corresponding signed message
- port to send message and qualifier to the Application SWC
- port to send a Challenge
- port to receive a Response

### 7.2.5 Message format

**[SWS_Vin_0070]** dAES and the vehicle key VK together with a CBC-MAC scheme are used for signing a message. c()

**[SWS_Vin_0071]** dIf (Information <= 64 bit), the following block is encrypted with AES and a CBC-MAC scheme

Block:

| Information (padded with zeros) | CounterBase | Delta (padded with zeros) |
|---|---|---|
| 64-Bit | 24-Bit | 40-Bit |

Signature = MSB(AES-CBC-MAC(VK,IV,Block) j Length of Signature in bits). IV = 0. c()

**[SWS_Vin_0072]** dIf (Information > 64 bit), the following block is encrypted with AES and a CBC-MAC scheme. The block length is a multiple of 128 bits. Block length = (128-Bit)*n, n>=2, n= ceil((Len(Information)-64)/128)) + 1.

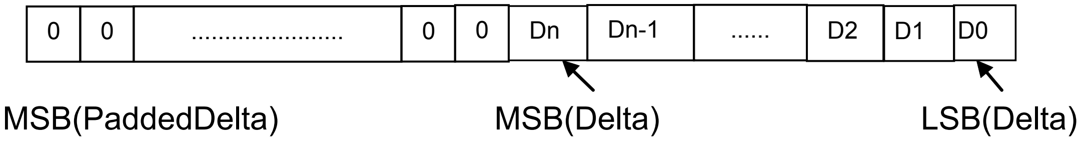
Block:

| Information (padded with zeros) | CounterBase | Delta (padded with zeros) |
|---|---|---|
| (n-1)*128-Bit + 64-Bit | 24-Bit | 40-Bit |

Signature = MSB(AES-CBC-MAC(VK,IV,Block) j Length of Signature in bits). IV = 0. The most significant bits are taken from the last 128-bit block which was encrypted with AES and the CBC-MAC scheme. c()

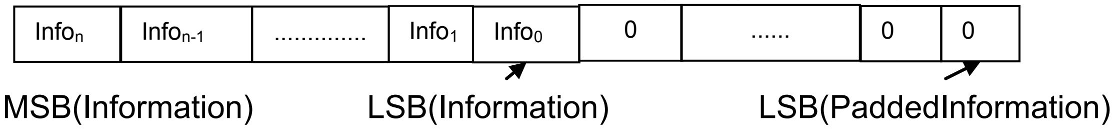
**[SWS_Vin_0073]** dThe byte order of Information, CounterBase and Delta is big endian. c()

**[SWS_Vin_0074]** dWhen padding Delta the most significant bits are filled with zeros.



c()

**[SWS_Vin_0075]** dInformation has to be padded by filling the least significant bits with zeros.



c()

**[SWS_Vin_0076]** *d*Each SSS uses a counter to sign message. The counter is incremented by 1 after a message was signed. *c*()

**[SWS_Vin_0077]** *d*Each SSV must ensure that the counter is increased by one with each received signed message. If this is not the case, the FSCSM considers the message verification as failed. *c*()

## 7.3 Error classification

**[SWS_Vin_0078]** *d*The Vin module shall provide the following errors.

| **Type of error** | **Relevance** | **Related error code** | **Value [hex]** |
|---|---|---|---|
| Fscsm Error | Production | FSCSM_ERROR | 0x800000 + 0x100 * SGID + 0x30 |

*c*(FsCSM_391)

## 7.4 Error detection

**[SWS_Vin_0079]** *d*The detection of production code errors cannot be switched off. *c*()

## 7.5 Error notification

**[SWS_Vin_0080]** *d*Production errors shall be reported to the Diagnostic Event Manager. *c*()

# 8 API Specification

## 8.1 Imported types

| Header File | Imported Type |
|---|---|
| Std_Types.h | Std_ReturnType |

## 8.2 Type definitions

No type definitions required.

## 8.3 Function definitions

### 8.3.1 Vin_Main

**[SWS_Vin_0004]** *d*

| | |
|---|---|
| **Service name** | Vin_Main |
| **Syntax** | `void Vin_Main(`<br>`   void`<br>`)` |
| **Service ID[HEX]** | – |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Description** | Main function. |

*c*()

**Note:** This function is cyclically triggered every 0.1s, when life cycle mode is running and VIN communication mode is on.

### 8.3.2 Vin_LifeCycleModeRequest

**[SWS_Vin_0005]** *d*

| | |
|---|---|
| **Service name** | Vin_LifeCycleModeRequest |
| **Syntax** | `void Vin_LifeCycleModeRequest(`<br>`   void`<br>`)` |
| **Service ID[HEX]** | – |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Description** | Evaluates the requested mode and switches accordingly. |

*c*()

**Note:** This is a RTE runnable which runs inside the exclusive area VinState.

### 8.3.3 Vin_SsvOnVkEstablished

**[SWS_Vin_0081]** *d*

| | |
|---|---|
| **Service name** | Vin_SsvOnVkEstablished |
| **Syntax** | `void Vin_SsvOnVkEstablished(`<br>`void`<br>`)` |
| **Service ID[HEX]** | – |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Description** | Called, when a new VKEstablished flag is available. |

*c*()

**Note:** This is a RTE runnable which runs inside the exclusive area VinState.

### 8.3.4 Vin_SsvReceiveResponseFromSss

**[SWS_Vin_0082]** *d*

| | |
|---|---|
| **Service name** | Vin_SsvReceiveResponseFromSss |
| **Syntax** | `void Vin_SsvReceiveResponseFromSss(`<br>`void`<br>`)` |
| **Service ID[HEX]** | – |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Description** | Called on a data received event when the response has been received from the SSS. |

*c*()

### 8.3.5 Vin_SsvReceiveMac

**[SWS_Vin_0083]** *d*

| | |
|---|---|
| **Service name** | Vin_SsvReceiveMac |
| **Syntax** | `void Vin_SsvReceiveMac(`<br>`void`<br>`)` |
| **Service ID[HEX]** | – |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Description** | Called on a data received event when the MAC has been received. |

*c*()

### 8.3.6 Vin_SsvStateGet

#### [SWS_Vin_0084] d

| | | |
|---|---|---|
| **Service name** | Vin_SsvStateGet | |
| **Syntax** | `Std_ReturnType Vin_SsvStateGet(`<br>`uint8 * mac,`<br>`uint8 * errorCounter,`<br>`uint32 * counterbase`<br>`)` | |
| **Service ID[HEX]** | – | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (out)** | `mac` | current MAC |
| | `errorCounter` | current error counter |
| | `counterbase` | current counterbase |
| **Description** | Returns the current SSV state. | |

*c*()

**Note:** This function is needed by the Fscsm module for the FAT test

### 8.3.7 Modes, Types and Mappings

#### [SWS_Vin_0006] d

```
CompuMethod CM_LifeCycleRequest {
   Category: TEXTTABLE
   VIN_INITIALIZED = 0
   VIN_RUNNING = 1
   VIN_STOPPED = 2
}
```

*c*()

#### [SWS_Vin_0008] d

```
CompuMethod CM_VinQualifier {
   Category: TEXTTABLE
   VIN_QUAL_INIT = 0
   VIN_QUAL_RECEIVED = 1
   VIN_QUAL_EQ_INTERNAL = 2
   VIN_QUAL_SECURE_PENDING = 4
   VIN_QUAL_SECURE_FINISHED = 8
   VIN_QUAL_SECURE_OK = 16
   VIN_QUAL_SECURE_MASK = 28
}
```

*c*()

#### [SWS_Vin_0009] d

```
ImplementationDataType VinArrayType {
   Category: ARRAY uint8[7]
   SizeSemantics: FIXED-SIZE
}
```

*c*()

### [SWS_Vin_0085] *d*

```
ImplementationDataType Vin_ChallengeArrayType {
   Category: ARRAY uint8[5]
   SizeSemantics: FIXED-SIZE
}
```

*c*()

### [SWS_Vin_0086] *d*

```
ImplementationDataType Vin_ChallengeRecordType {
   Category: STRUCTURE
   Elements:
   {
      TYPE_REFERENCE uint8 SssId;
      TYPE_REFERENCE uint8 SsvId;
      TYPE_REFERENCE Vin_ChallengeArrayType Challenge;
   }

}
```

*c*()

### [SWS_Vin_0010] *d*

```
ImplementationDataType Vin_ComVinType {
   Category: STRUCTURE
   Elements:
   {
      TYPE_REFERENCE uint8 Vin1;
      TYPE_REFERENCE uint8 Vin2;
      TYPE_REFERENCE uint8 Vin3;
      TYPE_REFERENCE uint8 Vin4;
      TYPE_REFERENCE uint8 Vin5;
      TYPE_REFERENCE uint8 Vin6;
      TYPE_REFERENCE uint8 Vin7;
   }

}
```

*c*()

### [SWS_Vin_0109] *d*

```
ImplementationDataType Vin_CounterBaseArrayType {
   Category: ARRAY uint8[3]
   SizeSemantics: FIXED-SIZE
}
```

*c*()

### [SWS_Vin_0011] *d*

```
ImplementationDataType Vin_LifeCycleRequestType {
   implementationDataType = uint8
   compuMethod = CM_LifeCycleRequest
}
```

*c*()

### [SWS_Vin_0012] *d*

```
ImplementationDataType Vin_QualifierType {
   implementationDataType = uint8
   compuMethod = CM_VinQualifier
}
```

c()

**[SWS_Vin_0087]** *d*

```
ImplementationDataType Vin_ResponseRecordType {
   Category: STRUCTURE
   Elements:
   {
      TYPE_REFERENCE uint8 SsvId;
      TYPE_REFERENCE uint32 CounterBase;
      TYPE_REFERENCE uint32 Signature;
   }

}
```

c()

**[SWS_Vin_0088]** *d*

```
ImplementationDataType Vin_SSVMACArrayType {
   Category: ARRAY uint8[8]
   SizeSemantics: FIXED-SIZE
}
```

c()

**[SWS_Vin_0089]** *d*

```
ImplementationDataType Vin_SSVMACRecordType {
   Category: STRUCTURE
   Elements:
   {
      TYPE_REFERENCE Vin_SSVMACArrayType Mac;
   }

}
```

c()

**[SWS_Vin_0014]** *d*

```
ImplementationDataType Vin_VinType {
   Category: STRUCTURE
   Elements:
   {
      TYPE_REFERENCE VinArrayType Vin;
      TYPE_REFERENCE Vin_QualifierType Qualifier;
   }

}
```

c()

**[SWS_Vin_0015]** *d*

```
ModeDeclarationGroup Vin_ChangeIndicator {
   {
      VIN_CI_INIT = 0,
      VIN_CI_NOCHANGE = 1,
      VIN_CI_CHANGED = 2
   }
   initialMode = VIN_CI_INIT
}
```

*c*()

**[SWS_Vin_0016]** *d*

```
ModeDeclarationGroup Vin_LifeCycle {
   {
      VIN_INITIALIZED = 0,
      VIN_RUNNING = 1,
      VIN_STOPPED = 2
   }
   initialMode = VIN_STOPPED
}
```

*c*()

**[SWS_Vin_0017]** *d*

```
ModeDeclarationGroup Vin_VinComMode {
   {
      VIN_VINCOMON = 0,
      VIN_VINCOMOFF = 1
   }
   initialMode = VIN_VINCOMOFF
}
```

*c*()

### 8.3.8 Provided Interfaces

**[SWS_Vin_0110]** *d*

```
ClientServerInterface ChassisNumberAuthentication {
   PossibleErrors {

   }
   generateAuthenticationCode(
      IN uint8 secureSignalVerifierId,
      IN Vin_ChallengeArrayType challenge,
      OUT Vin_CounterBaseArrayType counterBase,
      OUT Vin_CounterBaseArrayType authCodeCounterBase,
      OUT Vin_SSVMACArrayType authCodeVIN,
      ERR{}
   );
}
```

*c*()

**[SWS_Vin_0121]** *d*

```
ClientServerInterface FieldGetterSetterChassisNumber {
   PossibleErrors {

   }
   FieldGetterChassisNumber(
      OUT Vin_ComVinType Getter,
      ERR{}
   );
}
```

*c*()

**[SWS_Vin_0093]** *d*

```
ClientServerInterface SSVState {
   PossibleErrors {
      E_NOT_OK = 1
   }
   Get(
      OUT Vin_SSVMACArrayType mac,
      OUT uint8 errorCounter,
      OUT uint32 counterbase,
      ERR{E_NOT_OK}
   );
}
```

*c*()

**[SWS_Vin_0090]** *d*

```
SenderReceiverInterface Challenge {
   Vin_ChallengeRecordType Challenge;
}
```

*c*()

**[SWS_Vin_0018]** *d*

```
SenderReceiverInterface ComVin {
   Vin_ComVinType ComVin;
}
```

*c*()

**[SWS_Vin_0020]** *d*

```
SenderReceiverInterface ILifeCycleRequest {
   Vin_LifeCycleRequestType requestMode;
}
```

*c*()

**[SWS_Vin_0091]** *d*

```
SenderReceiverInterface Response {
   Vin_ResponseRecordType Response;
}
```

*c*()

**[SWS_Vin_0092]** *d*

```
SenderReceiverInterface SSVErrorCode {
   uint8 ErrorCode;
}
```

*c*()

**[SWS_Vin_0094]** *d*

```
SenderReceiverInterface SSVVinMac {
   Vin_SSVMACRecordType Mac;
}
```

*c*()

**[SWS_Vin_0021]** *d*

```
SenderReceiverInterface Vin {
   Vin_VinType Vin;
}
```

c()

**[SWS_Vin_0023]** d

```
SenderReceiverInterface VinRequest {
   uint16 RequestMessageIdentifier;
}
```

c()

**[SWS_Vin_0112]** d

```
SenderReceiverInterface Vin_InternalCheckVinTrigger {
   uint8 Trigger;
}
```

c()

**[SWS_Vin_0019]** d

```
ModeSwitchInterface ILifeCycle {
   Vin_LifeCycle Mode;
}
```

c()

**[SWS_Vin_0111]** d

```
ModeSwitchInterface IVinChangeIndicator {
   Vin_ChangeIndicator ChangeIndicator;
}
```

c()

**[SWS_Vin_0022]** d

```
ModeSwitchInterface VinCom {
   Vin_VinComMode Mode;
}
```

c()

### 8.3.9 Expected Interfaces

**[SWS_Vin_0095]** d

```
ClientServerInterface Vin_DiagnosticMonitor {
   PossibleErrors {
      E_NOT_OK = 1
   }
   SetEventStatus(
      IN Dem_EventStatusType EventStatus,
      ERR{E_NOT_OK}
   );
}
```

c()

**[SWS_Vin_0096]** d

```
ClientServerInterface DETService {
   PossibleErrors {
      E_NOT_OK = 1
   }
   ReportError(
      IN uint8 InstanceId,
      IN uint8 ApiId,
      IN uint8 ErrorId,
      ERR{E_NOT_OK}
   );
}
```

*c*()

**[SWS_Vin_0024]** *d*

```
ClientServerInterface EcuInfo {
   PossibleErrors {
      E_NOT_OK = 1
   }
   GetEcuId(
      OUT uint8 ecuId,
      ERR{E_NOT_OK}
   );
   GetVin(
      OUT Dlog_VinArrayType Vin,
      ERR{E_NOT_OK}
   );
}
```

*c*()

**[SWS_Vin_0098]** *d*

```
ClientServerInterface FscsmApplSwcInterface {
   PossibleErrors {
      FSCSM_E_NOT_OK = 1,
      FSCSM_E_MISSING_KEYS = 2,
      FSCSM_E_NOT_VERIFIED = 3,
      FSCSM_E_DOES_NOT_DECRYPT = 4,
      FSCSM_E_DOES_NOT_ENCRYPT = 5,
      FSCSM_E_NOT_EXPORTABLE = 6
   }
   VerifyMessage(
      IN Fscsm_MessageArrayType messageToVerify,
      IN uint16 messageLength,
      IN Fscsm_MessageArrayType macToVerify,
      IN uint16 macLength,
      ERR{FSCSM_E_NOT_OK,FSCSM_E_NOT_VERIFIED,FSCSM_E_MISSING_KEYS}
   );
}
```

*c*()

**[SWS_Vin_0099]** *d*

```
ClientServerInterface RandomNumberGenerator {
   PossibleErrors {
      E_NOT_OK = 1
   }
   GenerateRandomNumber(
      OUT Fscsm_MessageArrayType buffer,
      IN uint32 length,
      ERR{E_NOT_OK}
   );
}
```

*c*()

### [SWS_Vin_0100] *d*

```
SenderReceiverInterface VK_Established {
   boolean Flag;
}
```

*c*()

### [SWS_Vin_0101] *d*

```
ClientServerInterface Vin_NvMNotifyJobFinished {
   PossibleErrors {
      E_OK = 0
   }
   JobFinished(
      IN uint8 ServiceId,
      IN NvM_RequestResultType JobResult,
      ERR{E_OK}
   );
}
```

*c*()

### [SWS_Vin_0102] *d*

```
ClientServerInterface Vin_NvMService {
   PossibleErrors {
      E_NOT_OK = 1
   }
   GetErrorStatus(
      OUT NvM_RequestResultType RequestResultPtr,
      ERR{E_NOT_OK}
   );
   ReadBlock(
      IN NvM_DstPtrType DstPtr,
      ERR{E_NOT_OK}
   );
   WriteBlock(
      IN NvM_SrcPtrType SrcPtr,
      ERR{E_NOT_OK}
   );
}
```

*c*()

## 8.3.10 Service Definition

### [SWS_Vin_0026] *d*

```
Service Vin
{
   ProvidePort  Vin_NvMNotifyJobFinished NvMNotifyJobFinished_Vin
   ProvidePort  Challenge SSVChallengeToSSS
   ProvidePort  SSVErrorCode SSVErrorCode
   ProvidePort  SSVState SSVState
   ProvidePort  Vin Vin
   ProvidePort  IVinChangeIndicator VinChangeIndicator
   ProvidePort  VinRequest VinRequest
   RequirePort  ComVin ComVin
   RequirePort  DETService DETService_Vin
   RequirePort  Vin_DiagnosticMonitor DiagnosticMonitor_FscsmErrorEvent
   RequirePort  EcuInfo DlogEcuInfo
   RequirePort  FscsmApplSwcInterface FscsmCryptographicFunctions
   RequirePort  VK_Established FscsmVK_Established
   RequirePort  ILifeCycleRequest LifeCycleRequest
   RequirePort  Vin_NvMService NvMService_Vin
   RequirePort  RandomNumberGenerator RandomNumberGenerator
   RequirePort  Response SSVResponseFromSSS
   RequirePort  SSVVinMac SSVVinMacFromSSS
   RequirePort  VinCom VinCom
   ProvideRequirePort  Vin_InternalCheckVinTrigger InternalCheckVinTrigger
   ProvideRequirePort  ILifeCycle LifeCycle
}
```

c()

### 8.3.11 Runnables and Entry Points

### [SWS_Vin_0027] d

```
Internal behavior of Vin
{
   RunnableEntity Vin_LifeCycleModeRequest
      Symbol "Vin_LifeCycleModeRequest"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_Main
      Symbol "Vin_Main"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_NvmJobFinished
      Symbol "Vin_NvmJobFinished"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_OnComOff
      Symbol "Vin_OnComOff"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_OnComOn
      Symbol "Vin_OnComOn"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_OnVinChangeIndicatorAck
      Symbol "Vin_OnVinChangeIndicatorAck"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_ReceiveFromCom
      Symbol "Vin_ReceiveFromCom"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SsvCheckVin
      Symbol "Vin_SsvCheckVin"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SsvOnVkEstablished
      Symbol "Vin_SsvOnVkEstablished"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SsvReceiveMac
      Symbol "Vin_SsvReceiveMac"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SsvReceiveResponseFromSss
      Symbol "Vin_SsvReceiveResponseFromSss"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SsvStateGet
      Symbol "Vin_SsvStateGet"
      canbeInvokedConcurrently = false
}
```

c()

### [SWS_Vin_0113] *d*

```
Service VinSIAdapter
{
   ProvidePort  ComVin ComVin
   ProvidePort  Response SSVResponseFromSSS
   ProvidePort  SSVVinMac SSVVinMacFromSSS
   RequirePort  Challenge Challenge
   RequirePort  FieldGetterSetterChassisNumber ChassisNumber
   RequirePort  ChassisNumberAuthentication ChassisNumberAuthentication
   RequirePort  ComVin ChassisNumberNotifier
   RequirePort  VinRequest VinRequest
}
```

*c*()

## 8.3.12 Runnables and Entry Points

### [SWS_Vin_0114] *d*

```
Internal behavior of VinSIAdapter
{
   RunnableEntity Vin_SIAChallenge
      Symbol "Vin_SIAChallenge"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SIAGetterChassisNumberReturn
      Symbol "Vin_SIAGetterChassisNumberReturn"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SIANotifyChassisNumber
      Symbol "Vin_SIANotifyChassisNumber"
      canbeInvokedConcurrently = false
   RunnableEntity Vin_SIARequest
      Symbol "Vin_SIARequest"
      canbeInvokedConcurrently = false
}
```

*c*()

# 9 Sequence Diagrams

None

# 10 Configuration

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension. Chapter 10.2 specifies the structure (containers) and the parameters of the module Vin.

## 10.1 How to read this chapter

In addition to this section, it is highly recommended to read the document

- AUTOSAR Layered Software Architecture [7]
- AUTOSAR ECU Configuration Specification [8]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

### 10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term "Configuration class" (of a parameter) shall be used in order to refer to a specific configuration point in time.

### 10.1.2 Variants

The SWT module has the following variants:

**[SWS_StbP_0008]** *d*VARIANT-PRE-COMPILE: Only parameters with "Pre-compile time" configuration are allowed. *c*()

### 10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- all configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

## 10.2 Containers and configuration parameters

| | | |
|---|---|---|
| **Module Name** | Vin | |
| **Module Description** | Vin Module | |
| **Included Containers** | | |
| **Container Name** | **Multiplicity** | **Scope / Dependency** |
| CommonPublishedInformation | 1 | Common container, aggregated by all modules. It contains published information about vendor and versions. |
| VinGeneral | 1 | This container contains the configuration (parameters) of the Vin module. |
| SecureVin | 0..1 | |
| MultiConfig | 1 | |

### 10.2.1 CommonPublishedInformation

| | |
|---|---|
| **SWS Item** | SWS_Vin_0115 |
| **Container Name** | CommonPublishedInformation |
| **Description** | Common container, aggregated by all modules. It contains published information about vendor and versions. |
| **Configuration Parameters** | |

**[SWS_Vin_0116]** *d*

| | | | |
|---|---|---|---|
| **Name** | SwMajorVersion | | |
| **Description** | Major version number of the vendor specific implementation of the module. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Default Value** | 3 | | |
| **Configuration Class** | **Pre-compile time** | -- | |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

**[SWS_Vin_0117]** *d*

| | | | |
|---|---|---|---|
| **Name** | SwMinorVersion | | |
| **Description** | Minor version number of the vendor specific implementation of the module. The numbering is vendor specific. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Default Value** | 4 | | |
| **Configuration Class** | **Pre-compile time** | -- | |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

**[SWS_Vin_0118]** *d*

| | | | |
|---|---|---|---|
| **Name** | SwPatchVersion | | |
| **Description** | Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Default Value** | 1 | | |
| **Configuration Class** | **Pre-compile time** | -- | |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

### 10.2.2 VinGeneral

| | |
|---|---|
| **SWS Item** | SWS_Vin_0028 |
| **Container Name** | VinGeneral |
| **Description** | This container contains the configuration (parameters) of the Vin module. |
| **Configuration Parameters** | |

**[SWS_Vin_0029]** *d*

| | | | |
|---|---|---|---|
| **Name** | VinDevErrorDetect | | |
| **Description** | Activate/Deactivate the Development Error Detection and Notification. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucBooleanParamDef | | |
| **Default Value** | false | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

**[SWS_Vin_0031]** *d*

| | | | |
|---|---|---|---|
| **Name** | VinRequestMessageIdentifier | | |
| **Description** | Defines the ID which must use to request the VIN signal. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Range** | 0..65535 | | |
| **Default Value** | 32771 | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

**[SWS_Vin_0032]** *d*

| | | | |
|---|---|---|---|
| **Name** | TimeoutInitialVinRequest | | |
| **Description** | Timeout between init an first VIN request in s. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucFloatParamDef | | |
| **Range** | 0.0..25.0 | | |
| **Default Value** | 0.3 | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

c()

**[SWS_Vin_0033]** d

| | | | |
|---|---|---|---|
| **Name** | TimeoutSubsequentVinRequests | | |
| **Description** | Timeout between a request and the next request in s. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucFloatParamDef | | |
| **Range** | 0.1..25.0 | | |
| **Default Value** | 1.0 | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

c()

**[SWS_Vin_0119]** d

| | | | |
|---|---|---|---|
| **Name** | MaxNumberVinRequests | | |
| **Description** | Maximum number of VIN requests. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Range** | 1..255 | | |
| **Default Value** | 4 | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

c()

**[SWS_Vin_0120]** d

| | |
|---|---|
| **Name** | EnableSIAdapter |
| **Description** | Enables the Service Interface Adapter. This is needed for Ethernet communication. |
| **Multiplicity** | 1 |
| **Type** | EcucBooleanParamDef |
| **Default Value** | 0 |

| | | | |
|---|---|---|---|
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

c()

### 10.2.3 SecureVin

| | |
|---|---|
| **SWS Item** | SWS_Vin_0030 |
| **Container Name** | SecureVin |
| **Description** | |
| **Configuration Parameters** | |

**[SWS_Vin_0103]** *d*

| | | | |
|---|---|---|---|
| **Name** | SecureVinRequestMessageIdentifier | | |
| **Description** | The parameter defines the ID which the SSV must use to request a signed VIN from its connected SSS. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Range** | 0..65535 | | |
| **Default Value** | 57857 | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

c()

**[SWS_Vin_0104]** *d*

| | | | |
|---|---|---|---|
| **Name** | TimeoutMac | | |
| **Description** | This parameter defines the time span in seconds in which the SSV expects to receive the MAC message for the VIN from the SSS after requesting it. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucFloatParamDef | | |
| **Range** | 0.1..3 | | |
| **Default Value** | 1.0 | | |
| **Configuration Class** | **Pre-compile time** | X | PRECONFIGURED-CONFIGURATION |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

c()

**[SWS_Vin_0105]** *d*

| | | | |
|---|---|---|---|
| **Name** | TimeoutCounterbase | | |
| **Description** | This parameter defines the time span in seconds in which the SSV expects to receive an answer from a SSS after sending a challenge to the SSS. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucFloatParamDef | | |
| **Range** | 0.1..3 | | |
| **Default Value** | 1.0 | | |
| **Configuration Class** | **Pre-compile time** | X | PRECONFIGURED-CONFIGURATION |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

### [SWS_Vin_0106] *d*

| | | | |
|---|---|---|---|
| **Name** | ErrorlimitCounterbase | | |
| **Description** | This parameter defines the maximum number of times the CounterBase of an SSS can be requested by an SSV. If the SSV could not obtain a valid CounterBase within FscsmSSVChallengeFailedLimit times, the SSV enters an error state. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Range** | 1..10 | | |
| **Default Value** | 3 | | |
| **Configuration Class** | **Pre-compile time** | X | PRECONFIGURED-CONFIGURATION |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

### [SWS_Vin_0108] *d*

| | | | |
|---|---|---|---|
| **Name** | RemoteSSSId | | |
| **Description** | This parameter defines the ID of the SSS to which this SSV is connected. This information is needed to be able to perform the ChallengeResponse protocol correctly. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Range** | 0..255 | | |
| **Default Value** | 0 | | |
| **Configuration Class** | **Pre-compile time** | X | VARIANT-PRE-COMPILE |
| | **Link time** | -- | |
| | **Post-build time** | -- | |
| **Scope / Dependency** | | | |

*c*()

### 10.2.4 MultiConfig

| | |
|---|---|
| **SWS Item** | |
| **Container Name** | MultiConfig |
| **Description** | |
| **Configuration Parameters** | |

**[SWS_Vin_0107]** *d*

| | | | |
|---|---|---|---|
| **Name** | SSVId | | |
| **Description** | The unique ID of the SSV. Only relevant, if SecureVin is configured. | | |
| **Multiplicity** | 1 | | |
| **Type** | EcucIntegerParamDef | | |
| **Range** | 0..255 | | |
| **Default Value** | 0 | | |
| **Configuration Class** | **Pre-compile time** | -- | |
| | **Link time** | -- | |
| | **Post-build time** | X | VARIANT-POST-BUILD |
| **Scope / Dependency** | | | |

*c*()