Kvaser Air Bridge Management Interface description

Printed Wednesday 4th September, 2024

(This page is intentionally left blank.)

Contents

1 About this document **6**

2 Introduction **7**
- 2.1 Purpose . . . 7
- 2.2 Overall operation . . . 7
- 2.3 Interface access . . . 8
- 2.4 Air Bridge device role . . . 8
- 2.5 Understanding Pairing mode . . . 8
- 2.6 Terminology . . . 10

3 Message structure **13**
- 3.1 Receiver & Sender identifiers . . . 13
- 3.2 DLC . . . 13
- 3.3 Frame format . . . 13
- 3.4 Byte order & padding . . . 15
  - 3.4.1 Byte order/Endianness . . . 15
  - 3.4.2 Padding . . . 16
- 3.5 Command, Request & Event . . . 16
- 3.6 ACK & Response . . . 16
- 3.7 NAK . . . 17

4 Service Identifier (SID) **18**
- 4.1 SID: 0x1: Reset . . . 19
- 4.2 SID: 0x2: Read data . . . 19
- 4.3 SID: 0x3: Write data . . . 19
- 4.4 SID: 0x5: Runtime status . . . 19
- 4.5 SID: 0x6: Runtime configuration . . . 19

5 Data Identifier (DID) **20**
- 5.1 DIDs for SID: 0x1 (Reset) . . . 20
  - 5.1.1 DID: 0x0100: Hard reset . . . 20
- 5.2 DIDs for SID: 0x2 (Read data) . . . 21
  - 5.2.1 DID: 0x0A00: Read user status . . . 21
  - 5.2.2 DID: 0x0D00: Read runtime session CAN speed . . . 22
  - 5.2.3 DID: 0x0D01: Read version scan result . . . 22
  - 5.2.4 DID: 0xE000: Read Air Bridge manufacturing date . . . 24
  - 5.2.5 DID: 0xE001: Read device serial number . . . 24
  - 5.2.6 DID: 0xE002: Read EAN product code . . . 25
  - 5.2.7 DID: 0xE010: Read hardware version . . . 25
  - 5.2.8 DID: 0xE011: Read Air Bridge application version . . . 26
  - 5.2.9 DID: 0xE021: Read role . . . 26
  - 5.2.10 DID: 0xE022: Read transmit power level . . . 27
  - 5.2.11 DID: 0xE040: Read local RF identifier . . . 27

5.2.12 DID: 0xE041: Read remote (pairing) RF identifier 28
5.2.13 DID: 0xE0D0: Read RF configuration 29
5.2.14 DID: 0xE0D1: Read CAN speed 30
5.2.15 DID: 0xE0D2: Read CAN filter 31
5.2.16 DID: 0xE0D3: Read custom CAN configuration 33
5.2.17 DID: 0xE100: Read Management Interface enable status . . . 34
5.2.18 DID: 0xE101: Read Management Interface receiver identifier 34
5.2.19 DID: 0xE102: Read Management Interface sender identifier . 35
5.2.20 DID: 0x:EA02: Read custom data 36
5.2.21 DID: 0x:EA03: Read heartbeat period 37
5.3 DIDs for SID: 0x3 (Write data) 38
5.3.1 DID: 0x0A00: Write user status 38
5.3.2 DID: 0xE021: Write role 39
5.3.3 DID: 0xE022: Write transmit power level 39
5.3.4 DID: 0xE0D1: Write CAN speed 40
5.3.5 DID: 0xE0D2: Write CAN filter 40
5.3.6 DID: 0xE0D3: Write custom CAN configuration 42
5.3.7 DID: 0xE100: Write Management Interface enable status . . . 43
5.3.8 DID: 0xE101: Write Management Interface receiver identifier . 44
5.3.9 DID: 0xE102: Write Management Interface sender identifier . 45
5.3.10 DID: 0xEA00: Write Primary pairing code seed 46
5.3.11 DID: 0xEA01: Write Secondary pairing code seed 46
5.3.12 DID: 0xEA02: Write custom data 47
5.3.13 DID: 0xEA03: Write heartbeat period 47
5.4 DIDs for SID: 0x5 (Runtime status) 48
5.4.1 DID: 0xAE00: Link status report 48
5.4.2 DID: 0xAE01: Version alarm 49
5.4.3 DID: 0xAE02: Device hearbeat 49
5.4.4 DID: 0xAED1: Device report 50
5.4.5 DID: 0xAED2: No pairing code seed 51
5.4.6 DID: 0xAED3: Invalid pairing code 51
5.4.7 DID: 0xAED4: Pairing session timed out 52
5.4.8 DID: 0xAED5: Pairing session completed 53
5.4.9 DID: 0xA000: Device operational mode 54
5.4.10 DID: 0xA001: Device link status 54
5.4.11 DID: 0xAD00: Reported custom data 55
5.4.12 DID: 0xAD01: Pairing code seed defined 56
5.5 DIDs for SID: 0x6 (Runtime configuration) 57
5.5.1 DID: 0xCA00: Activate link status report 57
5.5.2 DID: 0xCA01: Set link parameters 57
5.5.3 DID: 0xCA02: Activate version alarm report 58
5.5.4 DID: 0xCA03: Set runtime session heartbeat period 59
5.5.5 DID: 0xCAD0: Activate Targeted Pairing mode and pair with pre-selected device 59
5.5.6 DID: 0xCAD1: Activate Discovery Pairing mode 60

5.5.7 DID: 0xCAD2: Select pairing device/Un-pair 61

A Appendix A: Usage examples **62**
A.1 Command example for re-configuring Air Bridge role 62
A.1.1 Messages 62
A.1.2 Role re-configuration procedure summary 62
A.1.3 Message examples 62
A.2 Command examples for Pairing mode 63
A.2.1 Messages 63
A.2.2 Pairing procedure summary - Targeted Pairing 64
A.2.3 Pairing procedure summary - Discovery Pairing 64
A.2.4 Un-pairing procedure summary - Asynchronous un-pairing . . 65
A.2.5 Un-pairing procedure summary - Synchronous un-pairing . . 66
A.2.6 Message examples 66
A.3 NAK example 69

B Appendix B: LED UI examples **70**
B.1 LED behaviour in Pairing mode 70

C Document Revision History **71**

1 About this document

This document is intended for users of **Kvaser Air Bridge** (Kvaser Air Bridge M12) who will utilize the Air Bridge **Management Interface**.

NOTE! The document does not apply to any variant of **Kvaser Air Bridge Light HS**.

The content herein describes the various (CAN protocol) Management Interface messages and general instructions on how to control the Kvaser Air Bridge system of *one-to-any* single wireless CAN devices.

Three complementary documents are available:

- Kvaser Air Bridge User's Guide
- Kvaser Air Bridge Installation Guide
- Kvaser Air Bridge System Integration Guide

The Kvaser Air Bridge User's Guide provides general data about the Air Bridge product's performance and operation for end-users.

The Kvaser Air Bridge Installation Guide provides installation advice for end-users who use Kvaser Air Bridge and want to optimize radio performance and reach.

The Kvaser Air Bridge System Integration Guide provides design-in advice for system integrators who use Kvaser Air Bridge as a system component and want to make the most of its data bridging capability, not least in scenarios where multiple Kvaser Air Bridges are to be employed.

2 Introduction

2.1 Purpose

This document constitutes a description of the functions and features that are specific for the **Kvaser Air Bridge** product's **Management Interface**.

In short, the interface defines a Kvaser Air Bridge specific protocol that formats CAN frames for specialized interpretation by Kvaser Air Bridge devices.

2.2 Overall operation

The **Kvaser Air Bridge Management Interface** is an application-level **request/response protocol** that enables a user application to access the control- and monitoring services of a Kvaser Air Bridge device.

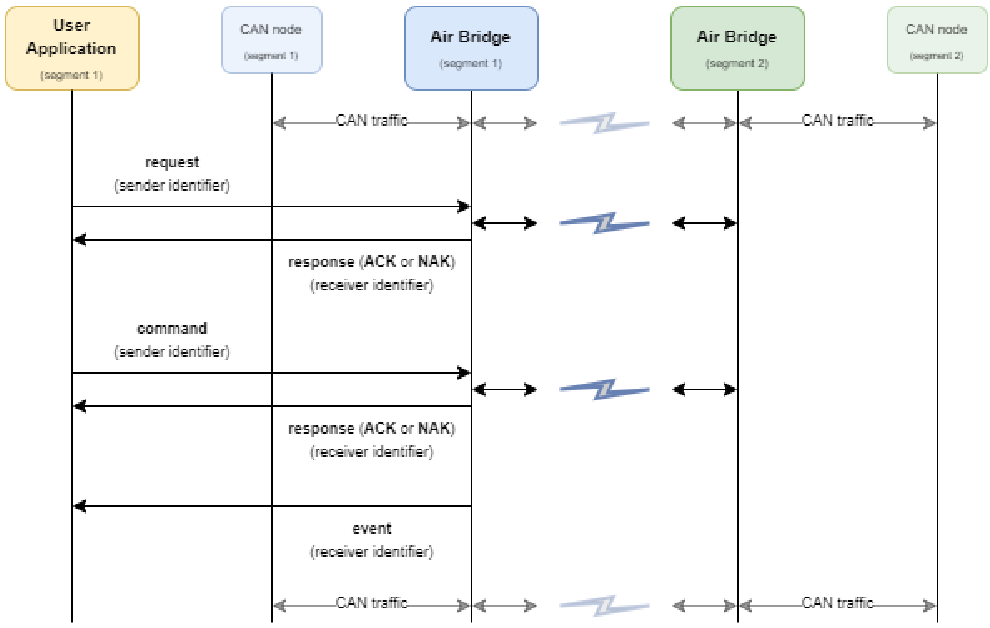


Figure 1: Interaction between a user application and Kvaser Air Bridge over the Management Interface protocol.

A client (a user application) sends a **request** to the Kvaser Air Bridge device in the form of a CAN message using a specific arbitration identifier, a **sender identifier** (see Section 3.1, Receiver & Sender identifiers, on Page 13). The Kvaser Air Bridge device distinguishes this request from any ongoing CAN traffic within the same CAN segment by recognizing this identifier. If the CAN message is formatted according to the Kvaser Air Bridge Management Interface protocol, the Kvaser Air Bridge device will handle the request based on the combination of **Service Identifier (SID)**, **Data Identifier (DID)** and possible **additional data** (see Section 3.3, Frame format, on Page 13).

The Kvaser Air Bridge **responds** to a request with a CAN message using another specific arbitration identifier, a **receiver identifier**. If the request was served, Kvaser Air Bridge responds with an acknowledgement (**ACK**) or a response message containing requested data (see Section 3.6, ACK & Response, on Page 16). If the request could not be served (for any reason), the Kvaser Air Bridge responds with a negative acknowledgement (**NAK**) (see Section 3.7, NAK, on Page 17).

2.3 Interface access

The **Kvaser Air Bridge Management Interface** enables a user to implement the services needed to control and monitor the Kvaser Air Bridge from their own application (*e.g.* for using the Pairing mode dynamically). However, it should be emphasized that *users are not required to implement this protocol*, as all commands can be executed using *e.g.* Kvaser's CAN bus monitoring application, **Kvaser CANKing**, which can be downloaded free of charge from Kvaser's website kvaser.com.

NOTE! It should also be stressed that a user that utilizes the **Kvaser Air Bridge Management Interface** in their own application only needs to implement the messages that will actually be used by that application, rather than the entire interface.

2.4 Air Bridge device role

The core function of Kvaser Air Bridge is to establish a wireless data link between two CAN segments, where one segment has an Air Bridge **Primary** device paired/associated with an Air Bridge **Secondary** device in the other CAN segment.

An Air Bridge device is factory-configured as a Secondary device. Therefore, in order to establish a paired communication between two Air Bridge devices, *one Air Bridge must be reconfigured to assume the **role** of **Primary***.

A detailed description of this role re-reconfiguration can be found in Section A.1, Command example for re-configuring Air Bridge role, on Page 62.

2.5 Understanding Pairing mode

With **Kvaser Air Bridge**, it becomes possible for a Primary device to, during operation, **discover** and **pair/associate** with different Secondary devices from various CAN segments.

This process - pairing/re-associating - is achieved when a user application places an Air Bridge Primary device into the so-called **Pairing mode** by using specific operations in the Management Interface. In this mode, any CAN traffic routed to the currently paired/associated Kvaser Air Bridge device is temporarily paused,

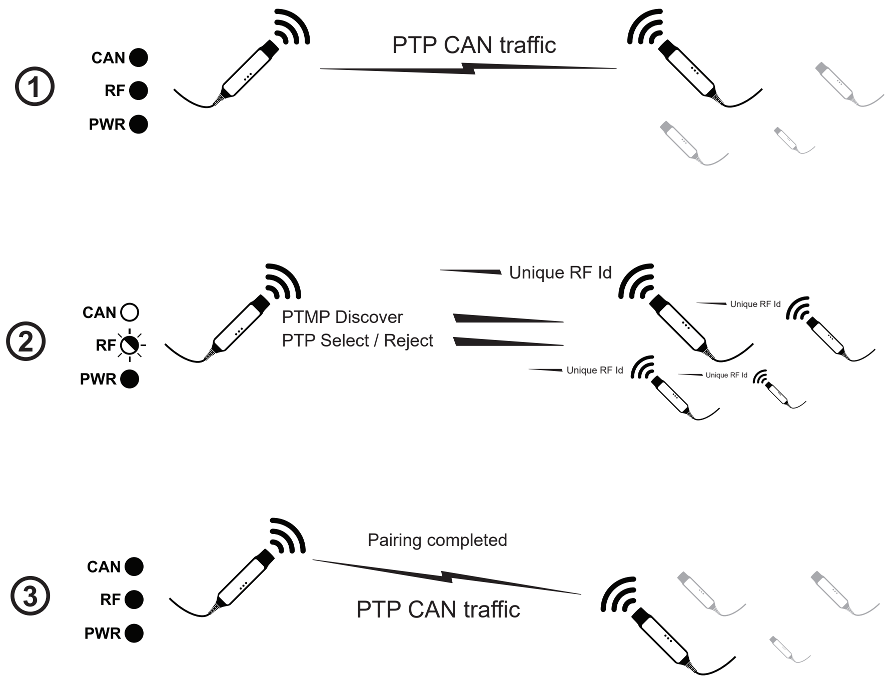


Figure 2: An established *point-to-point* connection (1) is paused and Pairing mode is entered (2) in which the Primary device pairs/re-associates with another reported Secondary device (3).

and the Primary device begins searching for Secondary devices in the vicinity (that currently do not have an established connection with any other Primary device).

An Air Bridge Secondary device can be paired/associated to an Air Bridge Primary in either of two ways:

a. **Targeted Pairing** - The Air Bridge Primary pairs with a specific or *targeted* Air Bridge Secondary device. This method is typically used when the Secondary device to be paired is determined in the context. That is, the Primary side knows the Secondary device's identifier, and the device is expected to be within radio range.

b. **Discovery Pairing** - Pairing based on reported discoveries *i.e.* a user application **selects** *one* out of possibly many reported Air Bridge Secondary devices, for pairing. This method, instead, is typically used when it is not predetermined exactly which one of a number of discovered devices should be paired.

If **Pairing mode** is initiated according to alternative **a**, with a specified, **targeted** Secondary device for pairing, the pairing/association is executed as soon as this specific Secondary device is discovered by the Primary device. Hence, if the *targeted* Secondary is not discovered, the discovery procedure proceeds until a predefined time-out condition is met, until a user application selects another

Secondary device for pairing (than the *targeted* one) or until a user application deactivates the Pairing mode, for any reason.

If **Pairing mode** is initiated according to alternative **b**, without a targeted Secondary device being specified, all discovered Secondary devices are continuously reported to the user application along with their respective unique identifier. Subsequently, the user application can command the Air Bridge Primary device to pair/re-associate with one of the discovered Secondary devices. Each discovered Air Bridge Secondary device also reports a *user-defined* **user status** value as well as a *user-defined* **custom data** to the user application. These values can *e.g.* serve as criteria for automatically selecting which Secondary device the Primary device should pair with.

To ensure that only desired Secondary devices respond to the Primary device during the discovery phase, a user can define a **code pair** in all devices intended to participate. This way, unauthorized devices (*e.g.* third-party Air Bridge devices) are excluded from a Pairing session.

Detailed descriptions of the two pairing procedures can be found in Section A.2, Command examples for Pairing mode, on Page 63.

NOTE! It is recommended to perform *only one* Pairing procedure at a time when multiple different Kvaser Air Bridge **sets** are operational within the same radio coverage area. This is because the discovery mechanism between different Kvaser Air Bridge sets may interfere with each other, thereby leading to poorer/slower discovery of Secondary devices.

2.6 Terminology

- **ACK** - Acknowledgement.
- **BCD (Binary Coded Decimal)** - Encoding method for decimal numbers where each decimal digit (0-9) is represented by its own four-bit binary sequence.
- **Command** - A Management Interface message sent *to* an Air Bridge to write some data to the device, *e.g.* a property value, mode change *etc.*
- **DID** - Data Identifier.
- **DLC** – Data Length Code. A part of the CAN message. It simply means the length of the CAN message, in bytes, and has a value between 0 and 8, inclusive.
- **Event** - A Management Interface message sent *from* an Air Bridge without prior reception of a Command or Request.
- **Identifier Extension Bit/IDE** - CAN frame indicator bit that states that a 29-bit Extended Identifier is being used rather than the 11-bit Standard Identifier.

- **Local** - Typically refers to the Air Bridge device in a pair, connected to the nearby CAN-bus segment, depending on context.
- **Local RF Id** - A Kvaser Air Bridge device's unique radio protocol identifier.
- **Primary device** - The superior device that controls the pairing in an Kvaser Air Bridge *pair* or *set* of devices.
- **NAK** - Negative acknowledgment.
- **NRC** - Negative Response Code. A specific code that further explains an error or negative acknowledgment in response to a request or command.
- **Operational mode** - Through a specific Management Interface command, an Air Bridge device can switch between various operating modes. The default mode is **CAN Traffic** where an Air Bridge pair of devices forwards CAN data between them. As for Kvaser Air Bridge, there is also the **Pairing** mode in which a Primary device can associate/pair itself with a discovered Secondary device.
- **Pairing** - The procedure in which a Kvaser Air Bridge Primary device *pairs/associates* a Kvaser Air Bridge Secondary device.
- **PTMP (Point-to-Multi-Point)** - Refers to that data is transmitted from a single source (an Air Bridge Primary device) to multiple destinations (several Air Bridge Secondary devices) simultaneously.
- **PTP (Point-to-Point)** - Refers to that data is transmitted between two endpoints directly (*one* Air Bridge Primary device and *one* Air Bridge Secondary device).
- **Remote** - Typically refers to the Air Bridge device in a pair, connected to a separated, remote CAN-bus segment.
- **Remote RF Id** - The coupling identifier (sometimes referred to as **Pairing Id**) used when a Kvaser Air Bridge Primary device *pairs/associates* with a Kvaser Air Bridge Secondary device. Typically a *pairing/association* uses the Primary device's **Local RF Id** as pairing identifier.
- **Request** - A Management Interface message sent *to* an Air Bridge to request some data from the device, *e.g.* a property value.
- **Response** - A Management Interface message sent *from* an Air Bridge as a direct response to a Command or Request.
- **RNAK** - Abbreviation for **Routing + NAK** in explanation of the message frame format.
- **Role** - The *device role* of a Kvaser Air Bridge device. The role can be either **Primary** or **Secondary**. A Kvaser Air Bridge **pair** typically consists of *one* Primary device and *one* Secondary device. A Kvaser Air Bridge **set** typically consists of *one* Primary device and *several* Secondary devices.

- **RSID** - Abbreviation for **Routing + SID** in explanation of the message frame format.
- **SID** - Service Identifier.
- **SIL** - Abbreviation for **Sequence Indicator + Length** in explanation of the message frame format.
- **Secondary device** - A subordinate device of a Kvaser Air Bridge *pair* or Kvaser Air Bridge *set* of devices.
- **Standard User** - The (non-elevated) *default* access level available for any Air Bridge user application.
- **Un-pairing** - The procedure in which a Kvaser Air Bridge Primary device *un-pairs/disassociates* from a Kvaser Air Bridge Secondary device.

3 Message structure

3.1 Receiver & Sender identifiers

Within Kvaser Air Bridge, Management Interface messages are differentiated from standard CAN load through dedicated **Receiver** and **Sender** arbitration identifiers. Hence, an application or system component that needs to access an Air Bridge device's Management interface shall use these identifiers when addressing or expecting data from the Management interface. The default identifiers are:

- Default Receiver Id: **0x1BC78FFF**
  (request messages that shall be received and handled by a Kvaser Air Bridge device)
- Default Sender Id: **0x1BFFF8F1**
  (response & event messages sent *from* a Kvaser Air Bridge device and may be handled by a user application)

NOTE[1] A user may redefine the arbitration identifier set with values tailored to the intended CAN segment.

NOTE[2] The *default* arbitration identifiers are both 29-bits **Extended Identifiers**. Hence, the **Identifier Extension Bit (IDE)** must be set in Management Interface command CAN frames when using the *default* arbitration identifier set.

3.2 DLC

The DLC is **8** for all messages of the Kvaser Air Bridge Management Interface.

3.3 Frame format

Kvaser Air Bridge Management Interface messages are typically *single*-framed messages with a payload of 0 - 4 bytes of data in addition to Service- (SID) and Data Identifiers (DID). However, greater payload can be exchanged across several consecutive frames with a maximum data length of **127 bytes** (SID and DID inclusive).

Single- and *multi*-framed messages are distinguished in how the first byte of the message (**Sequence Indicator + Length (SIL)**) is encoded:

- **Single-frame message:** Bit_7 (**n**) is always **0**, indicating that the frame represents a new message. Bit_0 - Bit_6 (**s**) specify the total length where the value range is **[3,7]** (dec), indicating that the payload (0 - 4 bytes) will fit in one frame.
- **Multi-frame message:**

- **First frame:** Bit_7 (**n**) is always **0**, indicating that the frame represents a new message. Bit_0 - Bit_6 (**s**) specify the total length where the value range is **[8,127]** (dec), indicating that the payload will span across several frames.
- **Subsequent frame:** Bit_7 (**n**) is always **1**, indicating that the frame represents a subsequent frame of a *multi*-framed message. Bit_0 - Bit_6 (**s**) specify the **remaining** length and can take any value in the range **[1,120]** (dec) depending on which part of a *multi*-frame message the respective frame represents.

Table 1 describes the format of a *single*-frame message or the first frame of a *multi*-framed message. Table 2 describes the format of the 2nd to last frame of a *multi*-framed message.

SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
0bnsssssss	**0b**000yxxxx	**0x**XX	**0x**XX	**0x**XX	**0x**XX	**0x**XX	**0x**XX

Table 1: Air Bridge Management Interface frame format for a *single*-frame message or the **first** frame in a *multi*-frame message.

SIL	D_{s-6}	D_{s-5}	D_{s-4}	D_{s-3}	D_{s-2}	D_{s-1}	D_s
0bnsssssss	**0x**XX	**0x**XX	**0x**XX	**0x**XX	**0x**XX	**0x**XX	**0x**XX

Table 2: Air Bridge Management Interface frame format for the **2nd** to **Nth** frame of a *multi*-frame message.

Byte_0: Sequence Indicator + Length (SIL)

The first byte of a *single*- as well as a *multi*-framed message frame always states if the frame is the **first** or **a sequenced** frame and also the **remaining number of bytes**, the 'Sequence Indicator + Length'-byte excluded.

- Bit_7 (**n**) $=$ Sequence Indicator [0=New *single* frame or first frame of a *multi*-framed message, 1=Concatenation frame of a *multi*-framed message]
- Bit_{6-0} (**s**) $=$ Remaining payload bytes, any 'Sequence Indicator + Length'-bytes excluded.

Hence, when intercepting a data stream with Kvaser Air Bridge Management Interface messages, the receiver shall ignore frames until the next frame with a 'Sequence Indicator + Length' value of less than **0x**80 = **0b**10000000 is recognized, meaning there is a new message (**n**-bit = 0).

Byte_1: Routing + SID (RSID)

Byte_1 holds a **routing** bit and the **SID** (Sevice Identifier). The routing bit states if the regarded command shall be routed to the *local* Air Bridge device or the *remote* side Air Bridge device. The SID points out a specific request service in the Kvaser Air Bridge Management Interface.

- $\text{Bit}_7 = 0$ (Reserved)
- $\text{Bit}_6 = 0$ (Reserved)
- $\text{Bit}_5 = 0$ (Reserved)
- $\text{Bit}_4 =$ Routing [0=local, 1=remote]
- $\text{Bit}_{3-0} =$ SID

Byte_2 - Byte_3: DID

Byte_2 and Byte_3 holds the **DID** (Data Identifier) which states the request data parameter of the specified SID.

Byte_4 - Byte_7: Data

Byte_4 to Byte_7 are used for passing parameter/value data. The **remaining length** of the data is specified by **Sequence Indicator + Length** (Byte_0) *(including the length of the SID and DID bytes (3))* .

Byte_1 - Byte_7 in a concatenated frame: Data

Byte_1 to Byte_7 are used for passing data in a *multi*-framed message. The **remaining length** of the data is specified by **Sequence Indicator + Length** (Byte_0).

3.4 Byte order & padding

3.4.1 Byte order/Endianness

The general byte order for *multi*-byte fields (*i.e.* **DID** and **Data** fields) is **Big-Endian/Network Byte Order**. Hence, the most significant byte (MSB) of a *multi*-byte data field is placed at the lowest frame byte.

Example: DID=**0304** (Select pairing device) is stored as Byte_2=0x03 and Byte_3=0x04.

3.4.2 Padding

Generally, any unused or unallocated bits of a parameter/data value are padded with zeros. For example, if only 20 bits are used to represent a value within a 4 bytes long U32 (unsigned 32-bits integer), the remaining, most significant, 12 bits will be padded with zeros.

Example: When selecting the pairing device, its 20 bits RF Id is specified as a U32 parameter (see Section 5.5.7, DID: 0xCAD2: Select pairing device/Un-pair, on Page 61). The **RF Id (0x11012)** is then encoded (blue) and padded (red) as follows:

SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
0x07	0x06	0xCA	0xD2	0x00	0x01	0x10	0x12

In a response frame with a **Sequence Indicator + Length (SIL)** value shorter than 7, unused bytes are typically padded with **0x55** (see Section A.3, NAK example, on Page 69).

3.5 Command, Request & Event

A Kvaser Air Bridge Management Interface **Command**, **Request** or **Event** is typically issued in a *single* CAN frame according to the frame format described in Table 1 on Page 14.

3.6 ACK & Response

- A Kvaser Air Bridge Management Interface **Command** is typically acknowledged with a *single* response frame similar to the Command frame (see Table 1 on Page 14) with the **Sequence Indicator + Length (SIL)** set to **3** (no data bytes).
- A Kvaser Air Bridge Management Interface **Request** is typically acknowledged with a response frame similar to the Request frame (see Table 1 on Page 14) but with the data bytes containing the requested data and the **Sequence Indicator + Length (SIL)** indicating the response data length.

When requested data is too big for a *single* CAN frame, the response will be divided into multiple CAN frames with a total payload of up to 127 bytes. The tables below give an example of a 12 bytes response (RSID + DID = 3 bytes and Payload = 9 bytes).

DLC	SIL	RSID/ACK	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0b**0000**1100**	**0b**000yxxxx	**0x**XX	**0x**XX	**0x**E1	**0x**8C	**0x**11	**0x**05

Table 3: First response frame out of two consecutive frames with total payload length of 12 bytes.

DLC	SIL	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
8	**0b1**0000**101**	**0x**00	**0x**10	**0x**41	**0x**11	**0x**EA	**0x**55	**0x**55

Table 4: Second response frame out of two consecutive frames with total length of 12 bytes. NOTE[1] The 'Sequenced + Length (SIL)'-byte states the **remaining** number of data bytes in the frame. NOTE[2] The last two *unused* bytes are padded with (red) **0x**55.

3.7 NAK

A negative respone to a Kvaser Air Bridge Management Interface Command/Request, a **NAK**, is always fitted in a *single* CAN frame according to:

DLC	SIL	RNAK	SID	NRC	—	—	—	—
8	**0b**000000**11**	**0b**000y1111	**0b**0000xxxx	**0x**00	—	—	—	—

Table 5: ACK/NAK frame format.

Byte$_0$: Sequence Indicator + Length (SIL)

Length of the frame, always **3**.

Byte$_1$: Routing + NAK (RNAK)

Byte$_1$ holds a **routing** bit (y) and the NAK discriminator 0b**1111**. The routing bit states if the concerned NAK was routed from the *local* Air Bridge device or the *remote* side Air Bridge device.

- Bit$_7$ $=$ 0 (Reserved)
- Bit$_6$ $=$ 0 (Reserved)
- Bit$_5$ $=$ 0 (Reserved)
- Bit$_4$ $=$ Routing [0=local, 1=remote]
- Bit$_{3-0}$ $=$ NAK (1111)

Byte$_2$: SID

Byte$_2$ specifies the **SID** (Sevice Identifier) of the request being negatively acknowledged.

Byte$_3$: NRC

Byte$_3$ specifies the **Negative Response Code** *i.e.* the reason for the request being negatively acknowledged:

NRC	Hex value
NRC_GENERAL_REJECT	0x10
NRC_SERVICE_NOT_SUPPORTED	0x11
NRC_INCORRECT_MESSAGE_LENGTH_OR_INVALID_FORMAT	0x13
NRC_REMOTE_UNAVAILABLE	0x14
NRC_REQUEST_OUT_OF_RANGE	0x31
NRC_SECURITY_ACCESS_DENIED	0x33
NRC_GENERAL_COMMAND_ERROR	0x72
NRC_PENDING	0x78
NRC_SUBFUNC_NOT_SUPPORTED_IN_ACTIVE_SESSION	0x7E
NRC_SERVICE_NOT_SUPPORTED_IN_ACTIVE_SESSION	0x7F
NRC_INVALID_PARAMETER	0x80
NRC_INVALID_STATE	0x81
NRC_INVALID_ROLE	0x82
NRC_NO_MATCH	0x85

Table 6: Negative Response Codes (NRC)

Byte$_4$ - Byte$_7$: Not used

4 Service Identifier (SID)

The Kvaser Air Bridge Management Interface provides a set of services that can be used to control and/or monitor an Air Bridge device. Each such service is identified with a 4 bit SID code. Hence, a user application can command or request data from an Air Bridge device by addressing it using the specific SID code. In the same manner, a user application can recognize status events sent *from* an Air Bridge device by filtering the SID code.

SID Code (Hex)	Description
0x1	Reset
0x2	Read data
0x3	Write data
0x5	Runtime status
0x6	Runtime configuration

Table 7: Available service identifiers

When using the services **Read data** (0x2), **Write data** (0x3), **Runtime status** (0x5) and **Runtime configuration** (0x6) the data to be read/written is decided by the subsequent **DID** (see Section 5, Data Identifier (DID), on Page 20). The first (hexadecimal) digit of the DID indicates which type of memory the regarded data is read/written from/to:

- **0**: Data is read/written from/to RAM (volatile).
- **E**: Data is read/written from/to EEPROM (persistent).
- **F**: Data is read/written from/to FLASH memory (persistent).

4.1 SID: 0x1: Reset

Miscellaneous levels of reset of the Air Bridge device (reset of sub-functions). Executed after acknowledging the reset on the CAN bus.

4.2 SID: 0x2: Read data

A service for reading various parameters (persistent and volatile) from an Air Bridge device. The particular parameter/data to be read is decided by the subsequent DID.

4.3 SID: 0x3: Write data

A service for writing various parameters (persistent and volatile) to an Air Bridge device. The particular parameter/data to be written is decided by the subsequent DID.

4.4 SID: 0x5: Runtime status

Monitoring/feedback service for a user application. A user application can be notified (event) as well as request data via this service.

4.5 SID: 0x6: Runtime configuration

A service for configuring various properties of an Air Bridge device's behaviour or performance, during runtime. Configurations take effect immediately or after power cycle depending on the specific request.

5 Data Identifier (DID)

Combined with a SID code, the 2-byte **Data Identifier** or DID, serves as parameter identifier and addresses a specific command or request of the Air Bridge Management interface.

5.1 DIDs for SID: 0x1 (Reset)

ID (Hex)	Role	Description	Access
0100	Primary, Secondary	Hard reset	Standard user

Table 8: Available DIDs for SID 0x1 (Reset).

5.1.1 DID: 0x0100: Hard reset

Performs a restart of the Air Bridge device. Any values stored in RAM will be discarded. No parameters required.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**01	**0x**01	**0x**00	—	—	—	—

Table 9: Message structure for DID 0x0100 (Hard reset).

5.2 DIDs for SID: 0x2 (Read data)

ID (Hex)	Role	Description	Default value	Access
0A00	Primary, Secondary	User status	0	Standard user
0D00	Primary, Secondary	Runtime session CAN speed	0	Standard user
0D01	Primary, Secondary	Version scan result	0	Standard user
E000	Primary, Secondary	Manufactoring date	-	Standard user
E001	Primary, Secondary	Device serial number	Unique	Standard user
E002	Primary, Secondary	EAN product code	014947 (hex)	Standard user
E010	Primary, Secondary	Hardware version	-	Standard user
E011	Primary, Secondary	Air Bridge application version	-	Standard user
E021	Primary, Secondary	Role	0	Standard user
E022	Primary, Secondary	Transmit power level	0	Standard user
E040	Primary, Secondary	Local RF identifier	Unique	Standard user
E041	Primary, Secondary	Remote (pairing) RF identifier	0	Standard user
E0D0	Primary, Secondary	RF configuration	0100070A (hex)	Standard user
E0D1	Primary, Secondary	CAN speed	0	Standard user
E0D2	Primary, Secondary	CAN filter	0	Standard user
E0D3	Primary, Secondary	Custom CAN configuration	0	Standard user
E100	Primary, Secondary	Management Interface enable status	1	Standard user
E101	Primary, Secondary	Management Interface receiver identifier	1BFFF8F1 (hex)	Standard user
E102	Primary, Secondary	Management Interface sender identifier	1BC78FFF (hex)	Standard user
EA02	Primary, Secondary	Custom data	0	Standard user
EA03	Primary, Secondary	Heartbeat period	0	Standard user

Table 10: Available DIDs for SID 0x2 (Read data).

5.2.1 DID: 0x0A00: Read user status

Lets an application read a previously written, *user-specified* **user status** value (0-15), from RAM (volatile).

- The **user status** value can be read repeatedly in runtime.
- The written **user status** value is reflected in the **Device report** message when in **Pairing mode** (see Section 5.4.4, DID: 0xAED1: Device report, on Page 50).
- If no **user status** value is previously written by the application (see Section 5.3.1, DID: 0x0A00: Write user status, on Page 38), the *default* value is **0**.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**03	**0x**02	**0x**0A	**0x**00	—	—	—	—

Table 11: Message structure for DID 0x0A00 (Read user status).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**02	**0x**0A	**0x**00	**0b**0000**xxxx**	—	—	—

Table 12: Message structure of a successful response for DID 0x0A00 (Read user status). **x** symbolizes the returned data.

Data	Type	Description	Value
User status (x)	1 nibble	User-specified value	0-15 (dec)

Table 13: Response data for DID 0x0A00 (Read user status)

5.2.2 DID: 0x0D00: Read runtime session CAN speed

Lets an application read the Air Bridge device's detected **CAN speed** of the current runtime session, from RAM.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**0D	**0x**00	—	—	—	—

Table 14: Message structure for DID 0x0D00 (Read runtime session CAN speed).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**02	**0x**0D	**0x**00	**0x**YY	—	—	—

Table 15: Message structure of a successful response for DID 0x0D00 (Read runtime session CAN speed). **Y** symbolizes the returned data.

Data	Type	Description	Value
CAN speed (YY)	1 byte	Reserved	0
		125 kbit/s	1
		250 kbit/s	2
		500 kbit/s	3
		1 Mbit/s	4
		Reserved	5-FF (hex)

Table 16: Response data for DID 0x0D00 (Read runtime session CAN speed)

5.2.3 DID: 0x0D01: Read version scan result

Lets an application read the Air Bridge device's current **Version scan result** of the ongoing runtime session, from RAM.

- At *start-up* Air Bridge scans the radio spectrum for any other present Air Bridge devices, both *compatible* as well as *incompatible* Air Bridge versions. This initial scan result is presented by the Air Bridge **LED**s as part of the *start-up procedure*.
- Occurrences of *compatible* and *incompatible* version Air Bridge radio frames are registered continuously in a runtime session and can be periodically sent to an application (see Section 5.4.2, DID: 0xAE01: Version alarm, on Page 49) or when requested (by the use of this service).
- The scan result presents two counter values:
  1. Number of registered *valid* version frames. A **valid version frame** is a frame from another Air Bridge device than the paired/associated one, that uses the same **radio header version**.
  2. Number of registered *invalid* version frames. An **invalid version frame** is a frame from another Air Bridge device that uses an older or newer **radio header version** than this device. Incorrectly decoded frames may also be registered as *invalid* version frames.

  Each counter *starts over* from 0 once the maximum value (0xFF) is reached.
- If *periodical reporting* of the **Version alarm** is enabled (see Section 5.5.3, DID: 0xCA02: Activate version alarm report, on Page 58), both counters are reset after every report.

DLC	**SIL**	**RSID**	**DID$_1$**	**DID$_0$**	**D$_0$**	**D$_1$**	**D$_2$**	**D$_3$**
8	**0x**03	**0x**02	**0x**0D	**0x**01	—	—	—	—

Table 17: Message structure for DID 0x0D01 (Read version scan result).

DLC	**SIL**	**RSID**	**DID$_1$**	**DID$_0$**	**D$_0$**	**D$_1$**	**D$_2$**	**D$_3$**
8	**0x**05	**0x**02	**0x**0D	**0x**01	**0xYY**	**0xZZ**	—	—

Table 18: Message structure of a successful response for DID 0x0D01 (Read version scan result). **Y** and **Z** symbolizes the returned data.

Data	**Type**	**Description**	**Value**
Valid number (Y)	1 byte	Number of *valid* version frames	0 - FF (hex)
Invalid number (Z)	1 byte	Number of *invalid* version frames	0 - FF (hex)

Table 19: Response data for DID 0x0D01 (Read version scan result)

5.2.4 DID: 0xE000: Read Air Bridge manufacturing date

Lets an application read the Air Bridge device's **manufacturing date**, from EEPROM.

- The manufacturing date is encoded using a Binary-Coded Decimal (BCD) format, where each decimal digit of the year (YY), month (MM), and day (DD) is individually represented by a 4-bit binary code. *E.g. the date '2024-02-20' is encoded as '20 24 02 20' in 4 bytes.*

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	0x03	0x02	0xE0	0x00	—	—	—	—

Table 20: Message structure for DID 0xE000 (Read manufacturing date).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	0x07	0x02	0xE0	0x00	0xYY	0xYY	0xMM	0xDD

Table 21: Message structure of a successful response for DID 0xE000 (Read Air Bridge manufacturing date). **Y**, **M** and **D** symbolizes the returned data.

Data	Type	Description	Value
Year (YYYY)	4 nibbles	Manufacturing year	0000 - 9999 (dec)
Month (MM)	2 nibbles	Manufacturing month	01 - 12 (dec)
Day (DD)	2 nibbles	Manufacturing day	01 - 31 (dec)

Table 22: Response data for DID 0xE000 (Read Air Bridge manufacturing date)

5.2.5 DID: 0xE001: Read device serial number

Lets an application read the Air Bridge device's factory preset **serial number**, from EEPROM.

- The serial number is encoded using a Binary-Coded Decimal (BCD) format, where each decimal digit is individually represented by a 4-bit binary code. *E.g. the serial number '11012' is encoded as '01 10 12' in 3 bytes.*

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	0x03	0x02	0xE0	0x01	—	—	—	—

Table 23: Message structure for DID 0xE001 (Read device serial number).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	0x07	0x02	0xE0	0x01	0x00	0xYY	0xYY	0xYY

Table 24: Message structure of a successful response for DID 0xE001 (Read device serial number). **Y** symbolizes the returned data.

Data	**Type**	**Description**	**Value**
Reserved	1 byte	-	0
SN (Y)	3 bytes	Device serial number (BCD)	0 - 999999 (dec)

Table 25: Response data for DID 0xE001 (Read device serial number)

5.2.6 DID: 0xE002: Read EAN product code

Lets an application read the Air Bridge device's factory preset **EAN product code**, from EEPROM.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**02	—	—	—	—

Table 26: Message structure for DID 0xE002 (Read EAN Product Code).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E0	**0x**02	**0x00**	**0xYY**	**0xYY**	**0xYZ**

Table 27: Message structure of a successful response for DID 0xE002 (Read EAN Product Code). **Y, Z** symbolizes the returned data.

Data	**Type**	**Description**	**Value**
EAN product code (Y)	5 nibbles	Product code within manufacturer's range	0 - FFFFF (hex)
Check digit (Z)	1 nibble	Validity check digit	0 - F (hex)

Table 28: Response data for DID 0xE002 (Read EAN Product Code)

5.2.7 DID: 0xE010: Read hardware version

Lets an application read the Air Bridge device's factory preset **hardware version**, from EEPROM.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**10	—	—	—	—

Table 29: Message structure for DID 0xE010 (Read hardware version).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E0	**0x**10	**0xYY**	**0xYY**	**0xYY**	**0xYY**

Table 30: Message structure of a successful response for DID 0xE010 (Read hardware version). **Y** symbolizes the returned data.

Data	**Type**	**Description**	**Value**
Hardware version (Y)	4 bytes	Version number	0 - FFFFFFFF (hex)

Table 31: Response data for DID 0xE010 (Read Hardware version)

5.2.8 DID: 0xE011: Read Air Bridge application version

Lets an application read the Air Bridge **application version**, from EEPROM.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**11	—	—	—	—

Table 32: Message structure for DID 0xE011 (Read Air Bridge application version).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E0	**0x**11	**0xMJ**	**0xMN**	**0xBB**	**0xBB**

Table 33: Message structure of a successful response for DID 0xE011 (Read Air Bridge application version). **MJ**, **MN** and **BB** symbolizes the returned data.

Data	**Type**	**Description**	**Value**
Major version (MJ)	1 byte	Application Major version	0-255 (dec)
Minor version (MN)	1 byte	Application Minor version	0-255 (dec)
Build number (BB)	2 bytes	Application Build number	0-65535 (dec)

Table 34: Response data for DID 0xE011 (Read Air Bridge application version)

5.2.9 DID: 0xE021: Read role

Lets an application read the **role** value, from EEPROM.

- An Air Bridge device's factory preset **role** value may be overridden by a user application (see Section 5.3.2, DID: 0xE021: Write role, on Page 39). Hence, a Primary device may be reconfigured as a Secondary and *vice versa*.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**21	—	—	—	—

Table 35: Message structure for DID 0xE021 (Read role).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**02	**0x**E0	**0x**21	**0b**0000**xxxx**	—	—	—

Table 36: Message structure of a successful response for DID 0xE021 (Read role). **x** symbolizes the returned data.

Data	Type	Description	Value
Role (x)	1 nibble	Not overridden (*default*)	0
		Overridden as **Primary**	1
		Overridden as **Secondary**	2
		Reserved	3-F (hex)

Table 37: Response data for DID 0xE021 (Read role)

5.2.10 DID: 0xE022: Read transmit power level

Lets an application read the **transmit power level** value, from EEPROM.

- An Air Bridge device's factory preset radio **transmit power level** value may be overridden by a user application (see Section 5.3.3, DID: 0xE022: Write transmit power level, on Page 39).

DLC	**SIL**	**RSID**	**DID**$_1$	**DID**$_0$	**D**$_0$	**D**$_1$	**D**$_2$	**D**$_3$
8	**0x**03	**0x**02	**0x**E0	**0x**22	—	—	—	—

Table 38: Message structure for DID 0xE022 (Read transmit power level).

DLC	**SIL**	**RSID**	**DID**$_1$	**DID**$_0$	**D**$_0$	**D**$_1$	**D**$_2$	**D**$_3$
8	**0x**07	**0x**02	**0x**E0	**0x**22	**0b**000000**xx**	—	—	—

Table 39: Message structure of a successful response for DID 0xE022 (Read transmit power level). **x** symbolizes the returned data.

Data	Type	Description	Value
Transmit power level (x)	2 bits	Not overriden (*default*)	0
		Overridden as **Maximum**	1
		Overridden as **Reduced**	2
		Overridden as **Very low**	3

Table 40: Response data for DID 0xE022 (Read transmit power level)

5.2.11 DID: 0xE040: Read local RF identifier

Lets an application read the Air Bridge device's factory preset **local RF Id**, from EEPROM.

- The Air Bridge device's **local RF Id** is a unique identifier typically derived from the device's **serial number**.
- An Air Bridge Primary device uses the **local RF Id** as *the identifier* with which it identifies itself in the transmitted radio frame headers. *An Air Bridge Secondary device however, typically uses another identifier than its **local RF Id**, namely the RF Id of its associated Primary.*
- The Air Bridge device's **local RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16).
- In **Pairing mode**, an Air Bridge Secondary device reports itself to the initiating Primary device using its *unique* identifier, in contrast to the currently set RF Id which it may have adopted from a Primary device in a previous Pairing mode session.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**03	**0x**02	**0x**E0	**0x**40	—	—	—	—

Table 41: Message structure for DID 0xE040 (Read Air Bridge local RF identifier).

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**07	**0x**02	**0x**E0	**0x**40	**0x**00	**0b**0000**yyyy**	**0xYY**	**0xYY**

Table 42: Message structure of a successful response for DID 0xE040 (Read Air Bridge local RF identifier). **y** symbolizes the returned data.

Data	Type	Description	Value
Local RF Id (y & Y)	20 bits	Air Bridge device's local RF Id	0 - FFFFF (hex)

Table 43: Response data for DID 0xE040 (Read Air Bridge local RF identifier).

5.2.12 DID: 0xE041: Read remote (pairing) RF identifier

Lets an application read the Air Bridge device's **remote RF Id**, *the pairing identifier*, from EEPROM.

- The **remote RF Id** is **0** for an *un-paired/disassociated* Primary or Secondary Air Bridge device.
- The **remote RF Id** of a *paired/associated* Primary device equals its **local RF Id**.
- The **remote RF Id** of a *paired/associated* Secondary device equals the **local RF Id** of the associated Primary device.

- The Air Bridge device's **remote RF Id** is represented with 20 bits (**y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**41	—	—	—	—

Table 44: Message structure for DID 0xE041 (Read Air Bridge remote (pairing) RF identifier).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E0	**0x**41	**0x**00	**0b**0000**yyyy**	**0xYY**	**0bYY**

Table 45: Message structure of a successful response for DID 0xE041 (Read Air Bridge remote (pairing) RF identifier). **y** and **Y** symbolizes the returned data.

Data	Type	Description	Value
Remote RF Id (y & Y)	20 bits	Air Bridge device's remote RF Id	0 - FFFFF (hex)

Table 46: Response data for DID 0xE041 (Read Air Bridge remote (pairing) RF identifier).

5.2.13 DID: 0xE0D0: Read RF configuration

Lets an application read the Air Bridge device's factory preset **RF configuration**, from EEPROM.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**D0	—	—	—	—

Table 47: Message structure for DID 0xE0D0 (Read RF configuration).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E0	**0x**D0	**0b**0000**xxxx**	**0x**00	**0x**07	**0b**00**pppmmm**

Table 48: Message structure of a successful response for DID 0xE0D0 (Read RF configuration). **x**, **p** and **m** symbolizes the returned data.

Data	**Type**	**Description**	**Value**
Role (x)	1 nibble	Reserved	0
		Primary	1
		Secondary	2
		Reserved	3-F (hex)
Reserved	2 bytes	Reserved	0007 (hex)
Transmit power level (p)	3 bits	Reserved	0
		Maximum	1
		Reduced	2
		Very low	3
Radio mode (m)	3 bits	Reserved	0-1
		LBT (Listen Before Talk)	2
		Reserved	3

Table 49: Response data for DID 0xE0D0 (Read RF configuration)

5.2.14 DID: 0xE0D1: Read CAN speed

Lets an application read the Air Bridge device's configured **CAN speed**, from EEPROM.

When a device is configured to use:

- **Auto-baud** (0), the *actual* CAN speed of a runtime session can be either of the defined speeds (value 1-4). The *actual* CAN speed of a runtime session can be retrieved using Section 5.2.2, DID: 0x0D00: Read runtime session CAN speed, on Page 22.
- **Custom** CAN speed (0xC), the baud rate and additional specified parameters used, is retrieved using Section 5.2.16, DID: 0xE0D3: Read custom CAN configuration, on Page 33.

DLC	**SIL**	**RSID**	**DID$_1$**	**DID$_0$**	**D$_0$**	**D$_1$**	**D$_2$**	**D$_3$**
8	**0x**03	**0x**02	**0x**E0	**0x**D1	—	—	—	—

Table 50: Message structure for DID 0xE0D1 (Read CAN speed).

DLC	**SIL**	**RSID**	**DID$_1$**	**DID$_0$**	**D$_0$**	**D$_1$**	**D$_2$**	**D$_3$**
8	**0x**07	**0x**02	**0x**E0	**0x**D1	**0xYY**	**0xYY**	**0xYY**	**0xYY**

Table 51: Message structure of a successful response for DID 0xE0D1 (Read CAN speed). **Y** symbolizes the returned data.

Data	Type	Description	Value
CAN speed (Y)	4 bytes	Auto-baud (*default*)	0
		125 kbit/s	1
		250 kbit/s	2
		500 kbit/s	3
		1 Mbit/s	4
		Reserved	5-B (hex)
		Custom	C (hex)
		Reserved	D-FFFF FFFF (hex)

Table 52: Response data for DID 0xE0D1 (Read CAN speed)

5.2.15 DID: 0xE0D2: Read CAN filter

Lets an application read all *user-specified* **CAN filters**, from EEPROM.

NOTE[1] For a deeper understanding of CAN filter usage, see Section 5.3.5, DID: 0xE0D2: Write CAN filter, on Page 40).

NOTE[2] A user can specify up to **4** different filters. When reading the filters from EEPROM, the number of response frames can vary depending on how many filters are defined.

- A **CAN filter** is defined by a 29-bit **filter identifier** and an equally sized **filter mask** that determines which bits of the identifier are relevant. This allows for filtering individual identifiers or ranges of identifiers. Traffic within the segment, where the applied mask to a message's identifier matches the result of the mask applied to the set filter, is accepted and transmitted to the paired Air Bridge device:
  - A binary **1** in the **mask** means the corresponding bit in the **filter identifier** is relevant.
  - A binary **0** in the **mask** means the corresponding bit in the **filter identifier** is *not* relevant.
  - A relevant binary **1** in the **filter identifier** means the corresponding bit in the **message identifier** must be **1**, for the message to be accepted and routed.
  - A relevant binary **0** in the **filter identifier** means the corresponding bit in the **message identifier** must be **0**, for the message to be accepted and routed.
- Up to **4** defined filters are returned. Hence, if one or more filters are defined and stored in EEPROM, a successful repsonse will always be a *multi-framed* response message.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E0	**0x**D2	—	—	—	—

Table 53: Message structure for DID 0xE0D2 (Read CAN filter).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**15	**0x**02	**0x**E0	**0x**D2	**0b**0000000**e**	**0b**000**iiiii**	**0xII**	**0xII**

Table 54: Message structure of the **first** frame of a successful response for DID 0xE0D2 (Read CAN filter) that returns **2** specified *Extended identifier* filters. **i** and **I** symbolizes the **filter identifier** of the first returned CAN filter.

DLC	SIL	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
8	**0x**8E	**0xII**	**0b**000**mmmmm**	**0xMM**	**0xMM**	**0xMM**	**0b**0000000**e**	**0b**000**jjjjj**

Table 55: Message structure of the **second** frame of a successful response for DID 0xE0D2 (Read CAN filter) that returns **2** specified *Extended identifier* filters. **I** symbolizes the LSB byte of the **filter identifier** of the first returned CAN filter. **m** and **M** symbolizes the **filter mask** of the first returned CAN filter. **j** symbolizes the MSB byte of the **filter identifier** of the second returned CAN filter. **NOTE!** SIL=0x8E indicates that the frame is a subsequent frame of a *multi-framed* message and that the remaining length is 14 bytes.

DLC	SIL	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}
8	**0x**87	**0xJJ**	**0xJJ**	**0xJJ**	**0b**000**nnnnn**	**0xNN**	**0xNN**	**0xNN**

Table 56: Message structure of the **third** frame of a successful response for DID 0xE0D2 (Read CAN filter) that returns **2** specified *Extended identifier* filters. **J** symbolizes the **filter identifier** of the second returned CAN filter. **n** and **N** symbolizes the **filter mask** of the second returned CAN filter. **NOTE!** SIL=0x87 indicates that the frame is a subsequent frame of a *multi-framed* message and that the remaining length is 7 bytes.

Data	Type	Description	Value
Identifier & mask type (e)	1 bit	Standard Identifier	0
		Extended Identifier	1
Filter identifier (i & I)	11 bits	User-specified arbitration identifier (Standard Identifier)	0-7FF (hex)
Filter mask (m & M)	11 bits	Bits of identifier to be considered in filter	0-07FF (hex)
or			
Filter identifier (i & I)	29 bits	User-specified filter identifier (Extended Identifier)	0-1FFFFFFF (hex)
Filter mask (m & M)	29 bits	Bits of identifier to be considered in filter	0-1FFFFFFF (hex)

Table 57: Response data for DID 0xE0D2 (Read CAN filter)

5.2.16 DID: 0xE0D3: Read custom CAN configuration

Lets an application read the *custom*, *user-specified* CAN configuration, from EEPROM (persistent).

NOTE[1] A read **custom CAN configuration** is effective only if **Custom** CAN speed is returned at the same time in Section 5.2.14, DID: 0xE0D1: Read CAN speed, on Page 30).

NOTE! This service is only intended for scenarios where the standard CAN speed options [125, 250, 500, 1000 kbps] are insufficient, and a specific, non-standard configuration is required (see Section 5.3.4, DID: 0xE0D1: Write CAN speed, on Page 40).

- The **custom configuration** includes settings for CAN speed, silent mode, and additional parameters, enabling precise control over the CAN bus operation.
- For a more detailed description of the parameters and their possible values, refer to Kvaser's web tutorials at kvaser.com:
  - CAN protocol tutorial
  - CAN Bus Bit Timing Calculator

  and the complementary document **Kvaser Air Bridge System Integration Guide**.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**03	**0x**02	**0x**E0	**0x**D3	—	—	—	—

Table 58: Message structure for DID 0xE0D3 (Read custom CAN configuration).

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**0B	**0x**02	**0x**E0	**0x**D3	**0xXX**	**0xXX**	**0xXX**	**0xXX**

Table 59: Message structure of the **first** frame of a successful response for DID 0xE0D3 (Read custom CAN configuration). **X** symbolizes the **baud rate** setting of the returned CAN configuration.

DLC	SIL	D$_4$	D$_5$	D$_6$	D$_7$	D$_8$	D$_9$	D$_{10}$
8	**0x**84	**0xYY**	**0xZZ**	**0xWW**	**0b**0000000**s**	—	—	—

Table 60: Message structure of the **second** frame of a successful response for DID 0xE0D3 (Read custom CAN configuration). **Y** symbolizes the **Bit segment 1** byte, **Z** symbolizes the **Bit segment 2** byte, **W** symbolizes the **SJW** byte and **s** symbolizes the **silent** bit of the returned CAN configuration. **NOTE!** SIL=0x84 indicates that the frame is a subsequent frame of a *multi-framed* message and that the remaining length is 4 bytes.

Data	Type	Description	Value
Baud rate (X)	4 bytes	Custom baud rate (bps)	0-FFFFFFFF (hex)
Bit segment 1 (Y)	1 byte	Propagation segment + Phase segment 1	0-FF (hex)
Bit segment 2 (Z)	1 byte	Phase segment 2	0-FF (hex)
SJW (W)	1 byte	Synchronization Jump Width	0-FF (hex)
Silent (s)	1 bit	Normal participation on bus	0
		Transmitter disabled, silent participation on bus	1

Table 61: Response data for DID 0xE0D3 (Read custom CAN configuration)

5.2.17 DID: 0xE100: Read Management Interface enable status

Lets an application read the Management Interface **enable status** of the device, from EEPROM.

- The **enable status** concerns the Management Interface **arbitration identifiers** (see Section 3.1, Receiver & Sender identifiers, on Page 13, Section 5.3.8, DID: 0xE101: Write Management Interface receiver identifier, on Page 44 and Section 5.3.9, DID: 0xE102: Write Management Interface sender identifier, on Page 45).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E1	**0x**00	—	—	—	—

Table 62: Message structure for DID 0xE100 (Read Management Interface enable status).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E1	**0x**00	**0x**00	**0x**00	**0x**00	**0b**0000000**s**

Table 63: Message structure of a successful response for DID 0xE100 (Read Management Interface enable status). **s** symbolizes the returned data.

Data	Type	Description	Value
Enable status (s)	1 bit	Disabled	0
		Enabled (default)	1

Table 64: Response data for DID 0xE100 (Read Management Interface Enable status)

5.2.18 DID: 0xE101: Read Management Interface receiver identifier

Lets an application read the Air Bridge device's *user-specified* Management Interface **receiver identifier**, from EEPROM.

- If no **receiver identifier** has been stored (see Section 5.3.8, DID: 0xE101: Write Management Interface receiver identifier, on Page 44), the factory preset identifier (**0x1BFFF8F1**) is read (see Section 3.1, Receiver & Sender identifiers, on Page 13).
- The returned identifier can be of either the 11-bit Standard Identifier type or the 29-bit Extended Identifier type.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E1	**0x**01	—	—	—	—

Table 65: Message structure for DID 0xE101 (Read Management Interface receiver identifier).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**05	**0x**02	**0x**E1	**0x**01	**0b**00000**xxx**	**0xXX**	—	—

Table 66: Message structure of a successful response for DID 0xE101 (Read Management Interface Receiver identifier) returning an 11-bit Standard Identifier. **x** & **X** symbolizes the returned data.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E1	**0x**01	**0b**000**yyyyy**	**0xYY**	**0xYY**	**0xYY**

Table 67: Message structure of a successful response for DID 0xE101 (Read Management Interface Receiver identifier). **y** & **Y** symbolizes the returned data.

Data	Type	Description	Value
Receiver id (x & X)	11 bits	Management Interface receiver identifier (Standard Identifier)	0 - 7FF (hex)
or			
Receiver id (y & Y)	29 bits	Management Interface receiver identifier (Extended Identifier)	0 - 1FFFFFFF (hex)

Table 68: Response data for DID 0xE101 (Read Management Interface receiver identifier)

5.2.19 DID: 0xE102: Read Management Interface sender identifier

Lets an application read the Air Bridge device's *user-specified* Management Interface **sender identifier**, from EEPROM.

- If no **sender identifier** has been stored (see Section 5.3.9, DID: 0xE102: Write Management Interface sender identifier, on Page 45), the factory preset identifier (**0x1BC78FFF**) is read (see Section 3.1, Receiver & Sender identifiers, on Page 13).
- The returned identifier can be of either the 11-bit Standard Identifier type or the 29-bit Extended Identifier type.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**E1	**0x**02	—	—	—	—

Table 69: Message structure for DID 0xE102 (Read Management Interface sender identifier).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**05	**0x**02	**0x**E1	**0x**02	**0b**00000**xxx**	**0xXX**	—	—

Table 70: Message structure of a successful response for DID 0xE102 (Read Management Interface Sender identifier) returning an 11-bit Standard Identifier. **x** & **X** symbolizes the returned data.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**E1	**0x**02	**0b**000**yyyyy**	**0xYY**	**0xYY**	**0xYY**

Table 71: Message structure of a successful response for DID 0xE102 (Read Management Interface Sender identifier). **y** and **Y** symbolizes the returned data.

Data	Type	Description	Value
Sender id (x & X)	11 bits	Management Interface sender identifier (Standard Identifier)	0 - 7FF (hex)
or			
Sender id (y & Y)	29 bits	Management Interface sender identifier (Extended Identifier)	0 - 1FFFFFFF (hex)

Table 72: Response data for DID 0xE102 (Read Management Interface sender identifier)

5.2.20 DID: 0x:EA02: Read custom data

Lets an application read a *user-specified* **custom data** (4 bytes) concerning the Air Bridge device, from EEPROM (persistent).

- The **custom data** can be read repeatedly in runtime.
- The **custom data** of the *initiating* Primary device can be retrieved from the Secondary devices when in **Pairing mode** (see Section 5.4.11, DID: 0xAD00: Reported custom data, on Page 55).
- The **custom data** of an *un-paired but discovered* Secondary device can be retrieved from the Primary device when in **Pairing mode** (see Section 5.4.11, DID: 0xAD00: Reported custom data, on Page 55 and Section A.2.6, Message examples, on Page 66).
- If no **custom data** is previously set by the application (see Section 5.3.12, DID: 0xEA02: Write custom data, on Page 47), the *default* value is **0**.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**EA	**0x**02	—	—	—	—

Table 73: Message structure for DID 0xEA02 (Read custom data).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**02	**0x**EA	**0x**02	**0xXX**	**0xXX**	**0xXX**	**0xXX**

Table 74: Message structure of a successful response for DID 0xEA02 (Read custom data). **X** symbolizes the returned data.

Data	Type	Description	Value
Custom data (X)	4 bytes	User-specified value	0-FFFFFFFF (hex)

Table 75: Response data for DID 0xEA02 (Read custom data)

5.2.21 DID: 0x:EA03: Read heartbeat period

Lets an application read the Air Bridge device's *user-specified* **heartbeat period**, from EEPROM (persistent).

- If no **heartbeat period** is previously set by the application (see Section 5.3.13, DID: 0xEA03: Write heartbeat period, on Page 47), the *default* value is **0** *i.e.* no **Device heartbeat** messages are sent from the device.
- If a *non-zero*, valid range **heartbeat period** is stored, the Air Bridge device will send a **Device heartbeat** message (see Section 5.4.4, DID: 0xAED1: Device report, on Page 50) every **hearbeat period** millisecond.
- The **heartbeat period** can be read repeatedly in runtime.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**02	**0x**EA	**0x**03	—	—	—	—

Table 76: Message structure for DID 0xEA03 (Read heartbeat period).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**05	**0x**02	**0x**EA	**0x**03	**0xXX**	**0xXX**	—	—

Table 77: Message structure of a successful response for DID 0xEA03 (Read heartbeat period). **X** symbolizes the returned data.

Data	Type	Description	Value
Heartbeat period (ms) (X)	2 bytes	No heartbeat (*default*)	0
		Reserved	1 - 99 (dec)
		Valid period (ms)	100 - 60000 (dec)
		Reserved	60001 - 65535 (dec)

Table 78: Response data for DID 0xEA03 (Read heartbeat period)

5.3 DIDs for SID: 0x3 (Write data)

ID (Hex)	Role	Description	Access
0A00	Primary, Secondary	User status	Standard user
E021	Primary, Secondary	Role	Standard user
E022	Primary, Secondary	Transmit power level	Standard user
E0D1	Primary, Secondary	CAN speed	Standard user
E0D2	Primary, Secondary	CAN filter	Standard user
E0D3	Primary, Secondary	Custom CAN configuration	Standard user
E100	Primary, Secondary	Management Interface enable status	Standard user
E101	Primary, Secondary	Management Interface receiver identifier	Standard user
E102	Primary, Secondary	Management Interface sender identifier	Standard user
EA00	Primary, Secondary	Primary pairing code seed	Standard user
EA01	Primary, Secondary	Secondary pairing code seed	Standard user
EA02	Primary, Secondary	Custom data	Standard user
EA03	Primary, Secondary	Heartbeat period	Standard user

Table 79: Available DIDs for SID 0x3 (Write data)

5.3.1 DID: 0x0A00: Write user status

Lets an application store a *user-specified* **user status** value (0-15), in RAM (volatile).

- The **user status** value can be updated repeatedly in runtime.
- The stored **user status** value is reflected in the **Device report** message when in **Pairing mode** (see Section 5.4.4, DID: 0xAED1: Device report, on Page 50).

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**04	**0x**03	**0x**0A	**0x**00	**0b**0000**ssss**	—	—	—

Table 80: Message structure for DID 0x0A00 (Write user status).

Parameter	Type	Description	Value
User status (s)	1 nibble	User-specified value	0-15 (dec)

Table 81: Parameters for DID 0x0A00 (Write user status)

5.3.2 DID: 0xE021: Write role

Lets an application store the **role** value, in EEPROM.

- The **role** value lets an application override the device's factory preset role (Secondary) and instead reconfigure the device as a Primary.
- A device is reset to the factory preset role (Secondary) by storing the value `0`.
- The stored value takes effect after a subsequent restart/power cycle.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**03	**0x**E0	**0x**21	**0b**0000**yyyy**	—	—	—

Table 82: Message structure for DID 0xE021 (Write role).

Data	Type	Description	Value
Role (y)	1 nibble	No override/Reset to default	0
		Override as **Primary**	1
		Override as **Secondary**	2
		Reserved	3-F (hex)

Table 83: Parameters for DID 0xE021 (Write role)

5.3.3 DID: 0xE022: Write transmit power level

Lets an application store the **transmit power level** value, in EEPROM.

- The **power level** value lets an application override the device's factory preset transmit power level (**Maximum**).
- A device is reset to the factory preset power level by storing the value `0`.
- The stored value takes effect after a subsequent restart/power cycle.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**03	**0x**E0	**0x**22	**0b**000000**yy**	—	—	—

Table 84: Message structure for DID 0xE022 (Write transmit power level).

Data	Type	Description	Value
Transmit power level (x)	2 bits	No override/Reset to default	0
		Override as **Maximum**	1
		Override as **Reduced**	2
		Override as **Very low**	3

Table 85: Parameters for DID 0xE022 (Write transmit power level)

5.3.4 DID: 0xE0D1: Write CAN speed

Lets an application configure the *default* **CAN speed** of the Air Bridge device, in EEPROM (persistent).

- If **Custom** CAN speed (0xC) is selected, the baud rate and additional parameters is specified using the **Write custom CAN configuration** service (see Section 5.3.6, DID: 0xE0D3: Write custom CAN configuration, on Page 42). Consequently, if **Custom** is selected, also the **custom CAN configuration** must have a valid set of values.
- The **CAN speed** value is stored in EEPROM and persists across power cycles.
- The stored value takes effect after a subsequent restart/power cycle.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**03	**0x**E0	**0x**D1	**0xYY**	**0xYY**	**0xYY**	**0xYY**

Table 86: Message structure for DID 0xE0D1 (Write CAN speed).

Parameter	Type	Description	Value
CAN speed (Y)	4 bytes	Auto-baud	0
		125 kbit/s	1
		250 kbit/s	2
		500 kbit/s	3
		1 Mbit/s	4
		Reserved	5-B (hex)
		Custom	C (hex)
		Reserved	D-FFFF FFFF (hex)

Table 87: Parameters for DID 0xE0D1 (Write CAN speed)

5.3.5 DID: 0xE0D2: Write CAN filter

Lets an application store a *user-specified* **CAN filter** in EEPROM (persistent).

NOTE[1] A stored filter is applied to bus traffic first after a subsequent restart/power cycle.

- A **CAN filter** is used to restrict which traffic within a CAN segment should be routed via Air Bridge to another CAN segment.
- A **CAN filter** is defined by either a *standard* 11-bit or an *extended* 29-bit **filter identifier** and an equally sized **filter mask** that determines which bits of the identifier are relevant. This allows for filtering individual identifiers or ranges of identifiers. Traffic within the segment, where the applied mask to a message's identifier matches the result of the mask applied to the set filter, is accepted and transmitted to the paired Air Bridge device:

- A binary **1** in the **mask** means the corresponding bit in the **filter identifier** is relevant.
- A binary **0** in the **mask** means the corresponding bit in the **filter identifier** is *not* relevant.
- A relevant binary **1** in the **filter identifier** means the corresponding bit in the **message identifier** must be **1**, for the message to be accepted and routed.
- A relevant binary **0** in the **filter identifier** means the corresponding bit in the **message identifier** must be **0**, for the message to be accepted and routed.

- A user can define up to **4** different filters that are used simultaneously.
- If **Enable status** is **1**, the specified filter is enabled and stored. If **0**, the filter is disabled and removed from EEPROM.
- Only one filter definition can be stored at a time.
- If **Filter identifier, Filter mask** and **Enable status** are all **0**, all stored filters are disabled and removed from EEPROM.
- The **CAN filters** are stored in EEPROM and persists across power cycles.
- **NOTE!** If Management Interface **enable status** is indeed *enabled* (see Section 5.3.7, DID: 0xE100: Write Management Interface enable status, on Page 43) and one or more filters are defined/stored, an *internally defined* filter that accepts traffic from the specified **Sender identifier** (see Section 5.2.19, DID: 0xE102: Read Management Interface sender identifier, on Page 35), is automatically activated.

NOTE² When different CAN segments use different bus speeds and bus load needs to be reduced within one segment, CAN filters can be used for load balancing.

NOTE³ Together with **custom data** (see Section 5.3.12, DID: 0xEA02: Write custom data, on Page 47), **CAN filters** can be used to create a subscription process where only data from a specific identifier is exchanged between CAN segments.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**0C	**0x**03	**0x**E0	**0x**D2	**0b**000000**s**1	**0b**000**xxxxx**	**0xXX**	**0xXX**

Table 88: Message structure for the **first** frame for DID 0xE0D2 (Write EXT CAN filter).

DLC	SIL	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
8	**0x**85	**0xXX**	**0b**000**yyyyy**	**0xYY**	**0xYY**	**0xYY**	—	—

Table 89: Message structure for the **second** frame for DID 0xE0D2 (Write EXT CAN filter).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**08	**0x**03	**0x**E0	**0x**D2	**0b**000000**s**0	**0b**00000**xxx**	**0xXX**	**0b**00000**yyy**

Table 90: Message structure for the **first** frame for DID 0xE0D2 (Write STD CAN filter).

DLC	SIL	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
8	**0x**81	**0xYY**	—	—	—	—	—	—

Table 91: Message structure for the **second** frame for DID 0xE0D2 (Write STD CAN filter).

Parameter	Type	Description	Value
Enable status (s)	1 bit	Disable/Delete	0
		Enable/Store	1
Filter identifier (x & X)	11 bits	User-specified arbitration identifier (Standard Identifier)	0-7FF (hex)
Filter mask (y & Y)	11 bits	Bits of identifier to be considered in filter	0-7FF (hex)
or			
Filter identifier (x & X)	29 bits	User-specified filter identifier (Extended Identifier)	0-1FFFFFFF (hex)
Filter mask (y & Y)	29 bits	Bits of identifier to be considered in filter	0-1FFFFFFF (hex)

Table 92: Parameters for DID 0xE0D2 (Write CAN filter)

5.3.6 DID: 0xE0D3: Write custom CAN configuration

Lets an application store a *custom* CAN configuration in EEPROM (persistent).

NOTE[1] A stored **custom CAN configuration** becomes effective only if **Custom** CAN speed is selected at the same time in Section 5.3.4, DID: 0xE0D1: Write CAN speed, on Page 40).

NOTE[2] This service is only intended for scenarios where the standard CAN speed options [125, 250, 500, 1000 kbps] are insufficient, and a specific, non-standard configuration is required (see Section 5.3.4, DID: 0xE0D1: Write CAN speed, on Page 40).

- The **custom configuration** includes settings for CAN speed, silent mode, and additional parameters, enabling precise control over the CAN bus operation.
- The maximum custom baud rate is **1,000,000 bps**.
- The Air Bridge CAN controller uses a **36 MHz** input clock frequency (a key parameter for determining the correct values for `bit segment 1`, `bit segment 2`, and `SJW` for a desired baud rate).
- For a more detailed description of the parameters and their possible values, refer to Kvaser's web tutorials at kvaser.com:

- CAN protocol tutorial
- CAN Bus Bit Timing Calculator

and the complementary document **Kvaser Air Bridge System Integration Guide**.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**0B	**0x**03	**0x**E0	**0x**D3	**0xXX**	**0xXX**	**0xXX**	**0xXX**

Table 93: Message structure for **first** frame for DID 0xE0D3 (Write custom CAN configuration).

DLC	SIL	D$_4$	D$_5$	D$_6$	D$_7$	D$_8$	D$_9$	D$_{10}$
8	**0x**84	**0xYY**	**0xZZ**	**0xWW**	**0b**0000000**s**	—	—	—

Table 94: Message structure for **second** frame for DID 0xE0D3 (Write custom CAN configuration).

Data	Type	Description	Value
Baud rate (X)	4 bytes	Custom baud rate (bps)	0 - 1000000 (dec)
		Reserved	F4241 - FFFFFFFF (hex)
Bit segment 1 (Y)	1 byte	Propagation segment + Phase segment 1	0-FF (hex)
Bit segment 2 (Z)	1 byte	Phase segment 2	0-FF (hex)
SJW (W)	1 byte	Synchronization Jump Width	0-FF
Silent (s)	1 bit	Normal participation on bus	0
		Transmitter disabled, silent participation on bus	1

Table 95: Parameters for DID 0xE0D3 (Write custom CAN configuration)

For the predefined, *standard* CAN speeds (see Section 5.3.4, DID: 0xE0D1: Write CAN speed, on Page 40) the parameter configurations are:

Baud rate (bps)	Bit segment 1	Bit segment 2	SJW
125000 (dec)	13 (dec)	2 (dec)	1 (dec)
250000 (dec)	6 (dec)	1 (dec)	1 (dec)
500000 (dec)	6 (dec)	1 (dec)	1 (dec)
1000000 (dec)	7 (dec)	1 (dec)	1 (dec)

Table 96: Standard CAN speed configuration parameters.

5.3.7 DID: 0xE100: Write Management Interface enable status

Lets an application write the Management Interface **enable status** of the device, to EEPROM.

- The **enable status** concerns the Management Interface **arbitration identifiers** (see Section 3.1, Receiver & Sender identifiers, on Page 13, Section 5.3.8, DID: 0xE101: Write Management Interface receiver identifier, on Page 44 and Section 5.3.9, DID: 0xE102: Write Management Interface sender identifier, on Page 45).
- **NOTE! enable status** indicates only whether the Air Bridge device will accept incoming request messages. If any function that sends event messages has been activated before the **enable status** is set to `0`, the Air Bridge will continue to emit such events.
- The stored value takes effect after a subsequent restart/power cycle.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**07	**0x**03	**0x**E1	**0x**00	**0x**00	**0x**00	**0x**00	**0b**0000000**s**

Table 97: Message structure for DID 0xE100 (Write Management Interface enable status).

Data	Type	Description	Value
Enable status (s)	1 bit	Disabled	0
		Enabled	1

Table 98: Parameters for DID 0xE100 (Write Management Interface enable status)

5.3.8 DID: 0xE101: Write Management Interface receiver identifier

Lets an application store one *user-specified* Management Interface **receiver identifier**, in EEPROM.

- The **receiver identifier** is stored in EEPROM and persists across power cycles.
- The stored **receiver identifier** takes effect only after a subsequent power cycle.
- **NOTE!** Invoking this service sets a *user-specified* arbitration identifier and overwrites the *default* arbitration **receiver identifier** (see Section 3.1, Receiver & Sender identifiers, on Page 13).
- The stored identifier can be of either the 11-bit Standard Identifier type or the 29-bit Extended Identifier type.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**05	**0x**03	**0x**E1	**0x**01	**0b**00000**xxx**	**0xXX**	—	—

Table 99: Message structure for DID 0xE101 (Write Management Interface Receiver identifier) with an 11-bit Standard Identifier as parameter.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**03	**0x**E1	**0x**01	**0b**000yyyyy	**0xYY**	**0xYY**	**0xYY**

Table 100: Message structure for DID 0xE101 (Write Management Interface Receiver identifier) with a 29-bit Extended Identifier as parameter.

Parameter	Type	Description	Value
Receiver Id (x & X)	11 bits	User-specified arbitration identifier (Standard Identifier)	0-7FF (hex)
or			
Receiver Id (y & Y)	29 bits	User-specified arbitration identifier (Extended Identifier)	0-1FFFFFFF (hex)

Table 101: Parameters for DID 0xE101 (Write Management Interface Receiver identifier)

5.3.9 DID: 0xE102: Write Management Interface sender identifier

Lets an application store one *user-specified* Management Interface **sender identifier**, in EEPROM.

- The **sender identifier** is stored in EEPROM and persists across power cycles.
- The stored **sender identifier** takes effect only after a subsequent power cycle.
- **NOTE!** Invoking this service sets a *user-specified* arbitration identifier and overwrites the *default* arbitration **sender identifier** (see Section 3.1, Receiver & Sender identifiers, on Page 13).
- The stored identifier can be of either the 11-bit Standard Identifier type or the 29-bit Extended Identifier type.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**05	**0x**03	**0x**E1	**0x**02	**0b**00000xxx	**0xXX**	—	—

Table 102: Message structure for DID 0xE102 (Write Management Interface Sender identifier) with an 11-bit Standard Identifier as parameter.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**03	**0x**E1	**0x**02	**0b**000yyyyy	**0xYY**	**0xYY**	**0xYY**

Table 103: Message structure for DID 0xE102 (Write Management Interface Sender identifier) with a 29-bit Extended Identifier as parameter.

Parameter	Type	Description	Value
Sender Id (x & X)	11 bits	User-specified arbitration identifier (Standard Identifier)	0-7FF (hex)
or			
Sender Id (y & Y)	29 bits	User-specified arbitration identifier (Extended Identifier)	0-1FFFFFFF (hex)

Table 104: Parameters for DID 0xE102 (Write Management Interface sender identifier)

5.3.10 DID: 0xEA00: Write Primary pairing code seed

5.3.11 DID: 0xEA01: Write Secondary pairing code seed

Lets an application store *user-specified* pairing code **seed** values concerning the Air Bridge device, in EEPROM memory (persistent).

- Each **seed** value is stored in EEPROM and persists across power cycles.
- Each **seed** value can be updated repeatedly in runtime.
- All Kvaser Air Bridge devices use *default* **seed** values until each pairing code seed is set or reset, respectively.
- The **seed** value is reset (*to default seed*) by writing the value `0x0`.
- **NOTE!** Only Kvaser Air Bridge devices using matching seed values can be discovered and selected in **Pairing mode**.

The seed values are used to generate the **Pairing Codes** used for authenticating the device when entering a Pairing session. Each Air Bridge device uses two seed values:

1. The **Primary seed** is used to generate the code that authenticates the *Primary* device on a Secondary device.
2. The **Secondary seed** is used to generate the code that authenticates a *Secondary* device on the Primary device.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	0x07	0x03	0xEA	0x00	0xYY	0xYY	0xYY	0xYY

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	0x07	0x03	0xEA	0x01	0xYY	0xYY	0xYY	0xYY

Table 105: Message structure for DID 0xEA00, 0xEA01 (Write Primary and Secondary pairing code seeds).

Parameter	Type	Description	Value
Pairing code seed (Y)	4 bytes	User-specified value	0 (reset) 1-FFFFFFFF (hex)

Table 106: Parameters for DID 0xEA00, 0xEA01 (Write Primary and Secondary pairing code seeds).

5.3.12 DID: 0xEA02: Write custom data

Lets an application store a *user-specified* **custom data** (4 bytes) concerning the Air Bridge device, in EEPROM (persistent).

- The **custom data** is stored in EEPROM and persists across power cycles.
- The **custom data** can be updated repeatedly in runtime.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**03	**0x**EA	**0x**02	**0xYY**	**0xYY**	**0xYY**	**0xYY**

Table 107: Message structure for DID 0xEA02 (Write custom data).

Parameter	Type	Description	Value
Custom data (Y)	4 bytes	User-specified value	0-FFFFFFFF (hex)

Table 108: Parameters for DID 0xEA02 (Write custom data)

5.3.13 DID: 0xEA03: Write heartbeat period

Stores a *user-specified* **heartbeat period** concerning the Air Bridge device, in EEPROM (persistent). The heartbeat period **value** decides if the Air Bridge heartbeat mechanism is activated or deactivated.

- If a *zero* value **heartbeat period** is stored, the heartbeat mechanism is **deactivated** *(default)* and no **Device heartbeat** messages are sent from the device.
- If a *non-zero*, valid range **heartbeat period** is stored, the Air Bridge device will send a **Device heartbeat** message (see Section 5.4.3, DID: 0xAE02: Device hearbeat, on Page 49) every **hearbeat period** millisecond.
- The **heartbeat period** is stored in EEPROM and persists across power cycles.
- The **heartbeat period** can be updated repeatedly in runtime.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**05	**0x**03	**0x**EA	**0x**03	**0xYY**	**0xYY**	—	—

Table 109: Message structure for DID 0xEA03 (Write heartbeat period).

Parameter	Type	Description	Value
Heartbeat period (ms) (Y)	2 bytes	No heartbeat	0
		Reserved	1 - 99 (dec)
		Valid period (ms)	100 - 60000 (dec)
		Reserved	60001 - 65535 (dec)

Table 110: Parameters for DID 0xEA03 (Write heartbeat period)

5.4 DIDs for SID: 0x5 (Runtime status)

ID (Hex)	Service type	Role	Description	Access
AE00	Event	Primary, Secondary	Link status report	Standard user
AE01	Event	Primary, Secondary	Version alarm	Standard user
AE02	Event	Primary, Secondary	Device hearbeat	Standard user
AED1	Event	Primary, Secondary	Device report	Standard user
AED2	Event	Primary, Secondary	No pairing code seed	Standard user
AED3	Event	Primary, Secondary	Invalid pairing code	Standard user
AED4	Event	Primary	Pairing session timed out	Standard user
AED5	Event	Primary, Secondary	Pairing session completed	Standard user
A000	Request	Primary, Secondary	Device operational mode	Standard user
A001	Request	Primary, Secondary	Device link status	Standard user
AD00	Request	Primary, Secondary	Reported custom data	Standard user
AD01	Request	Primary, Secondary	Pairing code seed defined	Standard user

Table 111: Available DIDs for SID 0x5 (Runtime status)

5.4.1 DID: 0xAE00: Link status report

Notifies a user application of link state changes for the concerned Air Bridge device.

NOTE! This event is only triggered if link status reports have been activated using Section 5.5.1, DID: 0xCA00: Activate link status report, on Page 57).

- The event is triggered when there is a change in *link state* for the concerned Air Bridge device.
- The reported link status concerns the *local* device of an Air Bridge pair.
- The link is considered **established** when a configured number of consecutive, valid RF frames from the paired/associated Air Bridge device is received. This threshold value is configured using the **Set link parameters** message (see Section 5.5.2, DID: 0xCA01: Set link parameters, on Page 57).
- The link is considered **lost** when a configured number of consecutive RF frames are not received. This threshold value is configured using the **Set link parameters** message (see Section 5.5.2, DID: 0xCA01: Set link parameters, on Page 57).
- The message aims to alert a user application when the link to the paired/associated Air Bridge device is lost and reestablished. *Hence, the frequency of unintentional link state changes can serve as one measure of how 'healthy'/stable the current radio environment is.*

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**05	**0x**AE	**0x**00	**0b**0000000**y**	—	—	—

Table 112: Message structure for DID 0xAE00 (Link status report).

Data	Type	Description	Value
Link status (y)	1 bit	Link is **lost**	0
		Link is **established**	1

Table 113: Data for DID 0xAE00 (Link status report).

5.4.2 DID: 0xAE01: Version alarm

Notifies a user application that *incompatible* version Air Bridge devices have been detected in the radio spectrum.

NOTE! This event is only triggered if version alarm reports have been activated using Section 5.5.3, DID: 0xCA02: Activate version alarm report, on Page 58).

- Occurrences of *compatible* and *incompatible* version Air Bridge radio frames are registered continuously in a runtime session.
- The report states the number of registered *incompatible* version frames since last report. (the counter is reset after every report).
- The report period is 5 seconds.
- The message aims to alert a user application that *incompatible* Air Bridge devices, that may cause interference, have been detected in the radio spectrum.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**05	**0x**AE	**0x**01	**0xYY**	—	—	—

Table 114: Message structure for DID 0xAE01 (Version alarm)

Data	Type	Description	Value
Invalid number (Y)	1 byte	Number of *incompatible* version frames	0 - FF (hex)

Table 115: Parameters for DID 0xAE01 (Version alarm)

5.4.3 DID: 0xAE02: Device hearbeat

Notifies a user application that an Air Bridge device in the CAN segment is powered up and responsive.

NOTE! This event is only triggered if a *non-zero* heartbeat period has previously been set (see Section 5.3.13, DID: 0xEA03: Write heartbeat period, on Page 47 and Section 5.5.4, DID: 0xCA03: Set runtime session heartbeat period, on Page 59).

- The event is triggered with the Air Bridge device's configured **heartbeat period**.
- The heartbeat concerns the *local* device of an Air Bridge pair.
- The **Operational mode** value reflects the device's current operational mode (see Section 5.4.9, DID: 0xA000: Device operational mode, on Page 54).
- The **Link status** value states whether or not the device has an established link to its paired counterpart (see Section 5.4.10, DID: 0xA001: Device link status, on Page 54).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**05	**0x**AE	**0x**02	**0b**0000**mmmm**	**0b**0000**ssss**	—	—

Table 116: Message structure for DID 0xAE02 (Device heartbeat).

Parameter	Type	Description	Value
Operational mode (s)	1 nibble	CAN traffic	0
		Pairing	1
		Reserved	2-F (hex)
Link status (s)	1 nibble	Link is **lost**	0
		Link is **established**	1
		Reserved	2-F (hex)

Table 117: Parameters for DID 0xAE02 (Device heartbeat)

5.4.4 DID: 0xAED1: Device report

Reports a detected Air Bridge **Secondary** device (from a Primary device) *or* the initiating **Primary** device (from Secondary devices), when in Pairing mode.

- The **User status** value reflects the current *user-defined* status of the referenced Air Bridge device (see Section 5.3.1, DID: 0x0A00: Write user status, on Page 38).
- The **Paired** value states whether (from a Primary Air Bridge) the referenced device is the currently paired Secondary device . **NOTE!** When reported from a Secondary device, the **Paired** value is not used (always *zero*).
- The **Device RF Id** states the referenced Air Bridge device's unique identifier (*the device's **local RF Id***)

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**05	**0x**AE	**0x**D1	**0b**0000**ssss**	**0b**000**pyyyy**	**0xYY**	**0xYY**

Table 118: Message structure for DID 0xAED1 (Device report)

Parameter	**Type**	**Description**	**Value**
User status (s)	1 nibble	User-defined status value	0 - F (hex)
Paired (p)	1 bit	**From Primary:** Device is not the currently paired Secondary device	0
		From Primary: Device *is* the currently paired Secondary device	1
		From Secondary: Not used	0
Device RF Id (y & Y)	20 bits	Device's local RF Id	0 - FFFFF (hex)

Table 119: Parameters for DID 0xAED1 (Device report)

5.4.5 DID: 0xAED2: No pairing code seed

Reports that no pairing code **seed** is set on the device.

- The event is triggered *once* (for each type of seed) if no seed is set, when a Primary or Secondary device enters Pairing mode. The message aims to alert a user application that no pairing code seed is set, and the respective Kvaser Air Bridge device will use the default seed.

DLC	**SIL**	**RSID**	**DID$_1$**	**DID$_0$**	**D$_0$**	**D$_1$**	**D$_2$**	**D$_3$**
8	**0x**04	**0x**05	**0x**AE	**0x**D2	**0b**0000000**y**	—	—	—

Table 120: Message structure for DID 0xAED2 (No pairing code seed).

Data	**Type**	**Description**	**Value**
Seed Id (y)	1 bit	Primary seed is not set	0
		Secondary seed is not set	1

Table 121: Data for DID 0xAED2 (No pairing code seed).

5.4.6 DID: 0xAED3: Invalid pairing code

Reports that an **invalid** pairing code is being used by a device referenced by **RF Id** in a Pairing session.

- On a Secondary device, the event is triggered if a Primary device is initiating a Pairing session but uses an invalid code (*compared to the code generated in the Secondary device*).
- On a Primary device, the event is triggered if a Secondary device reports itself in a Pairing session but uses an invalid code (*compared to the code generated in the Primary device*).
- The event is triggered at most once per second by either role.

- The **RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16). Each nibble (4-bit segment) within the 20-bit identifier must represent a decimal digit (0-9).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**05	**0x**AE	**0x**D3	**0x**00	**0b**0000**yyyy**	**0xYY**	**0xYY**

Table 122: Message structure for DID 0xAED3 (Invalid pairing code)

Data	Type	Description	Value
RF Id (y & Y)	20 bits	Device RF Id	00001 - 99999 (hex) (decimal digits only)

Table 123: Data for DID 0xAED3 (Invalid pairing code)

5.4.7 DID: 0xAED4: Pairing session timed out

Reports that an initiated Pairing session with a **targeted** Secondary device, has timed out.

- On a Primary device, the event is triggered if an ongoing session has exceeded (in time) the set *timeout* value.
- The **RF Id** states the concerned, **targeted** Secondary device's unique identifier (Local RF Id).
- The **RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16). Each nibble (4-bit segment) within the 20-bit identifier must represent a decimal digit (0-9).
- After a session timeout, the concerned device's **operational mode** is reverted to **CAN traffic** mode.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**05	**0x**AE	**0x**D4	**0x**00	**0b**0000**yyyy**	**0xYY**	**0bYY**

Table 124: Message structure for DID 0xAED4 (Pairing session timed out)

Data	Type	Description	Value
RF Id (y & Y)	20 bits	Targeted device's RF Id	00001 - 99999 (hex) (decimal digits only)

Table 125: Data for DID 0xAED4 (Pairing session timed out)

5.4.8 DID: 0xAED5: Pairing session completed

Reports that an initiated Pairing session has run its course, whether or not pairing was successful.

- The **RF Id** indicates the identifier of a potentially paired counterpart device at the conclusion of a session:

  On a **Primary** device:

  - A *non-zero* **RF Id** states the *targeted* Secondary device's unique identifier (local RF Id) and that *that particular device* was indeed paired.
  - A *zero* value **RF Id** states that a previously paired Secondary device was *un-paired/disassociated* from the Primary device in the session.

  On a **Secondary** device:

  - A *non-zero* **RF Id** states the *paired* Primary device's unique identifier (local RF Id) at the end of the session whether or not the concerned Secondary device was selected for pairing in the session.
  - A *zero* value **RF Id** states that the Secondary device is not paired to any Primary device.

- The **RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16). Each nibble (4-bit segment) within the 20-bit identifier must represent a decimal digit (0-9).
- After a session is completed, the concerned device's **operational mode** is reverted to **CAN traffic** mode.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**07	**0x**05	**0x**AE	**0x**D5	**0x**00	**0b**0000**yyyy**	**0x****YY**	**0b****YY**

Table 126: Message structure for DID 0xAED5 (Pairing session completed)

Data	Type	Description	Value
RF Id (y & Y)	20 bits	Unpaired	0
		Paired device's RF Id	00001 - 99999 (hex) (decimal digits only)

Table 127: Data for DID 0xAED5 (Pairing session completed)

5.4.9 DID: 0xA000: Device operational mode

Requests the Air Bridge device's current **operational mode**.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**05	**0x**A0	**0x**00	—	—	—	—

Table 128: Message structure for DID 0xA000 (Device operational mode)

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**05	**0x**A0	**0x**00	**0b**0000**xxxx**	—	—	—

Table 129: Message structure of a successful response for DID 0xA000 (Device operational mode). **x** symbolizes the returned data.

Data	Type	Description	Value
		CAN traffic	0
Operational mode (x)	1 nibble	Pairing	1
		Reserved	2-F (hex)

Table 130: Response data for DID 0xA000 (Device operational mode)

5.4.10 DID: 0xA001: Device link status

Requests the Air Bridge device's current **link status**.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**05	**0x**A0	**0x**01	—	—	—	—

Table 131: Message structure for DID 0xA001 (Device link status)

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**05	**0x**A0	**0x**01	**0b**0000**xxxx**	—	—	—

Table 132: Message structure of a successful response for DID 0xA001 (Device link status). **x** symbolizes the returned data.

Data	Type	Description	Value
		Link is **lost**	0
Link status (x)	1 nibble	Link is **established**	1
		Reserved	2-F (hex)

Table 133: Response data for DID 0xA001 (Device link status)

5.4.11 DID: 0xAD00: Reported custom data

Requests the **reported** *user-specified* **custom data** from initiating Primary or discovered Secondary devices, in Pairing mode.

NOTE! Reported custom data can only be requested when Pairing mode is activated. Hence, this service retrieves the custom data of a not yet paired device. From an already *paired* device, that device's **custom data** is retrieved with the Read service (see Section 5.2.20, DID: 0x:EA02: Read custom data, on Page 36).

Primary side request

When in Pairing mode, the initiating Air Bridge Primary device requests the **custom data** from detected Secondary devices using their respective **Device RF Id** (see Section 5.4.4, DID: 0xAED1: Device report, on Page 50) as parameter.

- The Secondary device **RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**05	**0x**AD	**0x**00	**0x**00	**0b**0000**yyyy**	**0xYY**	**0bYY**

Table 134: Message structure for DID 0xAD00 (Reported custom data (from Primary device))

Parameter	Type	Description	Value
RF Id (y)	20 bits	Secondary device's RF Id	0 - FFFFF (hex)

Table 135: Parameters for DID 0xAD00 (Reported custom data)

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**05	**0x**AD	**0x**00	**0xXX**	**0xXX**	**0xXX**	**0xXX**

Table 136: Message structure of a successful response for DID 0xAD00 (Reported custom data (to Primary device)). **X** symbolizes the returned data.

Data	Type	Description	Value
Custom data (X)	4 bytes	User-specified value	0 - FFFFFFFF (hex)

Table 137: Response data for DID 0xAD00 (Reported custom data)

Secondary side request

A Secondary device that has entered Pairing mode requests the **custom data** from the session's initiating Primary device. *No parameter needed.*

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**03	**0x**05	**0x**AD	**0x**00	—	—	—	—

Table 138: Message structure for DID 0xAD00 (Reported custom data (from Secondary device))

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**05	**0x**AD	**0x**00	**0xXX**	**0xXX**	**0xXX**	**0xXX**

Table 139: Message structure of a successful response for DID 0xAD00 (Reported custom data (to Secondary device)). **X** symbolizes the returned data.

Data	Type	Description	Value
Custom data (X)	4 bytes	User-specified value	0 - FFFFFFFF (hex)

Table 140: Response data for DID 0xAD00 (Reported custom data)

5.4.12 DID: 0xAD01: Pairing code seed defined

Requests if a *user-specified* **pairing code seed** has been stored in the Air Bridge device (see Section 5.3.10, DID: 0xEA00: Write Primary pairing code seed, on Page 46, Section 5.3.11, DID: 0xEA01: Write Secondary pairing code seed, on Page 46).

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**05	**0x**AD	**0x**01	**0b**0000000**s**	—	—	—

Table 141: Message structure for DID 0xAD01 (Pairing code seed defined)

Parameter	Type	Description	Value
Seed Id (s)	1 bit	Primary seed	0
		Secondary seed	1

Table 142: Parameters for DID 0xAD01 (Pairing code seed defined)

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**05	**0x**AD	**0x**01	**0b0000000x**	—	—	—

Table 143: Message structure of a successful response for DID 0xAD01 (Pairing code seed defined). **x** symbolizes the returned data.

Data	Type	Description	Value
Seed is set (x)	1 bit	Requested seed is *not* defined	0
		Requested seed is defined	1

Table 144: Response data for DID 0xAD01 (Pairing code seed set)

5.5 DIDs for SID: 0x6 (Runtime configuration)

ID (Hex)	Role	Description	Access
CA00	Primary, Secondary	Activate/deactivate link status report	Standard user
CA01	Primary, Secondary	Set link parameters	Standard user
CA02	Primary, Secondary	Activate/deactivate version alarm report	Standard user
CA03	Primary, Secondary	Set runtime session heartbeat period	Standard user
CAD0	Primary	Activate Targeted Pairing mode and pair with pre-selected device	Standard user
CAD1	Primary	Activate/deactivate Discovery Pairing mode	Standard user
CAD2	Primary	Select pairing device/Un-pair	Standard user

Table 145: Available DIDs for SID 0x6 (Runtime configuration)

5.5.1 DID: 0xCA00: Activate link status report

Activates/deactivates **link status** reporting depending on parameter.

- When activated, the Air Bridge device's **link status** [Link established, Link lost] is reported to a user application with the **Link status report** message (see Section 5.4.1, DID: 0xAE00: Link status report, on Page 48).
- Link status reporting is deactivated by default.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**04	**0x**06	**0x**CA	**0x**00	**0b**0000000**a**	—	—	—

Table 146: Message structure for DID 0xCA00 (Activate link status report)

Parameter	Type	Description	Value
Activate (a)	1 bit	Deactivate link status report	0
		Activate link status report	1

Table 147: Parameters for DID 0xCA00 (Activate link status report)

5.5.2 DID: 0xCA01: Set link parameters

Configures the threshold values for when a link between this Air Bridge device and its paired/associated counterpart, is considered *established* and *lost* respectively.

- **Link parameters** can be updated repeatedly in runtime.
- The set **link parameters** are *not* persistent but reset to their *default* values on every restart/power cycle.
- The parameter `up` states the number of consecutive RF frames (from paired Air Bridge device) that must be correctly received to consider the link *established*. The *default* value is **2**.
- The parameter `down` states how many consecutive RF frames (from paired Air Bridge device) that can be missing/invalid before the link is considered *lost*. The *default* value is **4**.
- *The RF LED indicator (blue) of the Air Bridge device indicates (in CAN Traffic mode) when a link is considered established (illuminated) and lost (OFF). Thus, any adjustments to these parameters will affect this LED behavior.*

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**05	**0x**06	**0x**CA	**0x**01	**0xXX**	**0xYY**	—	—

Table 148: Message structure for DID 0xCA01 (Set link parameters)

Parameter	Type	Description	Value
Down (X)	1 byte	Revert to default (4)	0
		Number of missing frames	1-FF (hex)
Up (Y)	1 byte	Revert to default (2)	0
		Number of valid frames	1-FF (hex)

Table 149: Parameters for DID 0xCA01 (Set link parameters)

5.5.3 DID: 0xCA02: Activate version alarm report

Activates/deactivates **Version alarm** reporting depending on parameter.

- When activated, the Air Bridge device periodically reports the **Version alarm** message to a user application every 5 seconds (see Section 5.4.2, DID: 0xAE01: Version alarm, on Page 49).
- Version alarm reporting is deactivated by default.

DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
8	**0x**04	**0x**06	**0x**CA	**0x**02	**0b**0000000**a**	—	—	—

Table 150: Message structure for DID 0xCA02 (Activate version alarm report)

Parameter	Type	Description	Value
Activate (a)	1 bit	Deactivate version alarm report	0
		Activate version alarm report	1

Table 151: Parameters for DID 0xCA02 (Activate version alarm report)

5.5.4 DID: 0xCA03: Set runtime session heartbeat period

Stores a *user-specified* **heartbeat period** concerning the Air Bridge device, *valid only in the current runtime session*, in RAM (volatile). The heartbeat period **value** decides if the Air Bridge heartbeat mechanism is activated or deactivated.

- A *non-zero*, valid range **heartbeat period**, **overrides** the value stored in EEPROM (see Section 5.3.13, DID: 0xEA03: Write heartbeat period, on Page 47). Hence, if an application needs to temporarily monitor the Air Bridge in the current runtime session (and not at every startup), or with a different period than the one stored in EEPROM, this service can be utilized.
- If a *non-zero*, valid range **heartbeat period** is set, the Air Bridge device will send a **Device heartbeat** message (see Section 5.4.3, DID: 0xAE02: Device hearbeat, on Page 49) every **hearbeat period** millisecond.
- If a *zero* value **heartbeat period** is set, the heartbeat value stored in EEPROM (see Section 5.3.13, DID: 0xEA03: Write heartbeat period, on Page 47) will instead apply. Hence, persistent activation of the heartbeat mechanism can not be deactivated by setting a *zero* value **heartbeat period** using this service.
- The **heartbeat period** can be updated repeatedly in runtime.

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**05	**0x**06	**0x**CA	**0x**03	**0x**YY	**0x**YY	—	—

Table 152: Message structure for DID 0xCA03 (Set runtime session heartbeat period)

Parameter	Type	Description	Value
Heartbeat period (ms) (Y)	2 bytes	No heartbeat/Use persistently stored value	0
		Reserved	1 - 99 (dec)
		Valid period (ms)	100 - 60000 (dec)
		Reserved	60001 - 65535 (dec)

Table 153: Parameters for DID 0xCA03 (Set runtime session heartbeat period)

5.5.5 DID: 0xCAD0: Activate Targeted Pairing mode and pair with pre-selected device

Activates the Targeted Pairing mode and selects the Secondary device, specified by **RF Id**, for pairing, if discovered within time frame specified by the **timeout** parameter.

- A *non-zero* **RF Id** specifies the targeted Secondary device's unique identifier (local RF Id).
- A *zero* **RF Id** means that the initiating Primary device shall instead un-pair/disassociate any currently paired Secondary device.
- The **RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16). Each nibble (4-bit segment) within the 20-bit identifier must represent a decimal digit (0-9).
- A *positive* **timeout** specifies a number of seconds **[1, 255]**. If the targeted Secondary device is not discovered by the Air Bridge Primary device within this time frame, or if a user application has not selected another device for pairing within this time frame, the Pairing mode is deactivated and a Runtime status event (**Pairing session timed out**) is triggered (see Section 5.4.7, DID: 0xAED4: Pairing session timed out, on Page 52).
- A *zero* **timeout** value means indefinite timeout, *i.e.*, the Pairing procedure does *not* timeout.

NOTE! Pairing mode can only be activated on an Air Bridge **Primary** device.

DLC	**SIL**	**RSID**	**DID**$_1$	**DID**$_0$	**D**$_0$	**D**$_1$	**D**$_2$	**D**$_3$
8	**0x**07	**0x**06	**0x**CA	**0x**D0	**0xXX**	**0b**0000**yyyy**	**0xYY**	**0xYY**

Table 154: Message structure for DID 0xCAD0 (Activate Targeted Pairing mode and pair with pre-selected device)

Parameter	**Type**	**Description**	**Value**
Timeout (X)	1 byte	Indefinite timeout	0
		Maximum discovery time (seconds)	1-FF (hex)
RF Id (y & Y)	20 bits	Un-pair from any paired Secondary device	0
		Targeted Secondary device's RF Id (20 bits)	00001 - 99999 (hex) (decimal digits only)

Table 155: Parameters for DID 0xCAD0 (Activate Targeted Pairing mode and pair with pre-selected device)

5.5.6 DID: 0xCAD1: Activate Discovery Pairing mode

Activates/deactivates the Discovery Pairing mode depending on parameter.

NOTE! Pairing mode can only be activated on an Air Bridge **Primary** device.

DLC	**SIL**	**RSID**	**DID**$_1$	**DID**$_0$	**D**$_0$	**D**$_1$	**D**$_2$	**D**$_3$
8	**0x**04	**0x**06	**0x**CA	**0x**D1	**0b**0000000**a**	—	—	—

Table 156: Message structure for DID 0xCAD1 (Activate Discovery Pairing mode)

Parameter	**Type**	**Description**	**Value**
Activate (a)	1 bit	*Deactivate* Discovery Pairing mode	0
		Activate Discovery Pairing mode	1

Table 157: Parameters for DID 0xCAD1 (Activate Discovery Pairing mode)

5.5.7 DID: 0xCAD2: Select pairing device/Un-pair

Selects a previously discovered Secondary device, specified by **RF Id**, to be paired/associated with the initiating Primary device.

- A *non-zero* **RF Id** specifies the desired Secondary device's unique identifier (local RF Id) as reported with the **Device report** message (see Section 5.4.4, DID: 0xAED1: Device report, on Page 50).
- A *zero* **RF Id** means that the initiating Primary device shall instead un-pair/disassociate any currently paired Secondary device.
- The **RF Id** is represented with 20 bits (**y** & **Y**) and the remaining, most significant bits of the U32, are padded with zeros (see Section 3.4.2, Padding, on Page 16). Each nibble (4-bit segment) within the 20-bit identifier must represent a decimal digit (0-9).

NOTE! Pairing device can only be selected on an Air Bridge **Primary** device when Pairing mode is activated.

DLC	**SIL**	**RSID**	**DID**$_1$	**DID**$_0$	**D**$_0$	**D**$_1$	**D**$_2$	**D**$_3$
8	**0x**07	**0x**06	**0x**CA	**0x**D2	**0x**00	**0b**0000**yyyy**	**0x**YY	**0b**YY

Table 158: Message structure for DID 0xCAD2 (Select pairing device/Un-pair)

Parameter	**Type**	**Description**	**Value**
RF Id (y & Y)	20 bits	Un-pair from any paired Secondary device	0 (hex)
		Selected Secondary device's RF Id (20 bits)	00001 - 99999 (hex) (decimal digits only)

Table 159: Parameters for DID 0xCAD2 (Select pairing device/Un-pair)

A Appendix A: Usage examples

A.1 Command example for re-configuring Air Bridge role

A Kvaser Air Bridge set of devices typically includes *one* Primary device and *one* or *more* Secondary devices. To facilitate easy setup, every Kvaser Air Bridge is delivered pre-configured as Secondary for minimal effort. As a consequence, in order to establish an Air Bridge link between two devices, at least *one* device has to be re-configured as Primary, *i.e.*, the **role** parameter of the device must be overridden.

A.1.1 Messages

There are two types of messages involved in re-configuration of the Air Bridge **role**:

#	SIL	SID (name)	DID	Message description
1	**0x03**	**0x02** (Read data)	**0xE021**	Read role
2	**0x04**	**0x03** (Write data)	**0xE021**	Write role

Table 160: Messages involved in the role re-configuration

A.1.2 Role re-configuration procedure summary

The following *step-by-step* instructions summarize the necessary steps to re-configure a Kvaser Air Bridge device to another **role**.

1. Connect the Air Bridge device to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge device that shall be re-configured as another **role**.
3. *(optional)* From a user application, send message **#1** (see Table 160) to read the current **role** value of the Air Bridge device.
4. From a user application, send message **#2** (see Table 160) to override the current **role** in accordance with specified message parameter.
5. Power cycle the Air Bridge device for the new **role** value to take effect.

A.1.3 Message examples

a) User application reads the current **Role** value from a connected Air Bridge device:

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**03	**0x**02	**0x**E0	**0x**21	-	-	-	-
Rx	8	**0x**04	**0x**02	**0x**E0	**0x**21	00	-	-	-

*Here, the device responds `00` which means its **role** is **not overridden** and hence, acts as **Secondary**.*

b) User application writes the **Role** value to a connected Air Bridge device and reconfigures it as **Primary**:

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**04	**0x**03	**0x**E0	**0x**21	**0x**01	-	-	-
Rx	8	**0x**03	**0x**03	**0x**E0	**0x**21	-	-	-	-

*Here, `01` is written to the device which means its **role** after a subsequent power cycle, will be **Primary**.*

A.2 Command examples for Pairing mode

The Air Bridge operational mode - **Pairing mode** - is used to pair/associate a Kvaser Air Bridge Primary device with one specific Kvaser Air Bridge Secondary device (out of possibly many). When the pairing procedure is completed, the paired Primary and Secondary devices will try to re-establish a link with one another, even after a subsequent power cycle.

NOTE! The **Pairing mode** can only be activated on Kvaser Air Bridge **Primary** devices.

A.2.1 Messages

There are 14 types of messages involved in the Pairing procedure:

#	SIL	SID (name)	DID	Message description
1	**0x04**	**0x03** (Write data)	**0x0A00**	Write user status
2	**0x07**	**0x03** (Write data)	**0xEA00**	Write Primary pairing code seed
3	**0x07**	**0x03** (Write data)	**0xEA01**	Write Secondary pairing code seed
4	**0x07**	**0x03** (Write data)	**0xEA02**	Write custom data
5	**0x07**	**0x06** (Runtime configuration)	**0xCAD0**	Activate Targeted Pairing mode and pair with pre-selected device
6	**0x04**	**0x06** (Runtime configuration)	**0xCAD1**	Activate/Deactivate Discovery Pairing mode
7	**0x07**	**0x06** (Runtime configuration)	**0xCAD2**	Select pairing device/Un-pair
8	**0x03**	**0x05** (Runtime status)	**0xA000**	Request device's current operational mode
9	**0x07**	**0x05** (Runtime status)	**0xAED1**	Device report
10	**0x04**	**0x05** (Runtime status)	**0xAED2**	No pairing code seed
11	**0x07**	**0x05** (Runtime status)	**0xAED3**	Invalid pairing code
12	**0x07**	**0x05** (Runtime status)	**0xAD00**	Request reported custom data
13	**0x07**	**0x05** (Runtime status)	**0xAED4**	Pairing session timed out
14	**0x07**	**0x05** (Runtime status)	**0xAED5**	Pairing session completed

Table 161: Messages involved in the Pairing mode

A.2.2 Pairing procedure summary - Targeted Pairing

The following *step-by-step* instructions summarize the necessary steps to pair/associate a Kvaser Air Bridge Primary device with a *targeted* Kvaser Air Bridge Secondary device.

1. Connect the Air Bridge Primary device to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Primary device that shall discover and pair with a *targeted* Air Bridge Secondary device.
3. Connect the *targeted* Air Bridge Secondary device to another CAN-bus segment.
4. **Power ON** the *targeted* Air Bridge Secondary device.
5. *(optional)* If *user-defined*, non-default **Pairing codes** will be utilised to lock out unauthorized devices (*e.g.* third-party Air Bridge devices) from the pairing session, set **Primary pairing code seed** and **Secondary pairing code seed** for both the Primary and Secondary devices respectively, in each CAN-bus segment, by sending message **#2** and **#3** (see Table 161 on Page 63). **NOTE!** Use the same seeds on all devices intended to communicate with each other.
6. From a user application, send message **#5** (see Table 161 on Page 63) with parameter bytes (timeout, targeted device RF Id) to the Primary device. *This message causes the Primary device to switch to Targeted Pairing mode and 1) any ongoing CAN-traffic is blocked from RF transmission and 2) the RF LED (blue) indicator will flash ON and OFF every half second.*
7. *All powered Secondary devices (within reach) that do not already have an established connection with another Primary device, will detect the mode change and start reporting their respective RF Identifiers & user status to the Primary device. When a Secondary device has reported itself, the PWR LED (green) indicator will be constantly lit and the RF LED (blue) indicator will flash ON and OFF typically every half second.*
8. *The Primary device discovers Secondary device(s) and when the targeted Secondary device is recognized, it is automatically paired/associated with the initiating Primary device.*
9. *When the pairing is acknowledged, all devices involved will leave Pairing mode and return to standard (CAN Traffic) mode. Message **#14** (see Table 161 on Page 63) will be transmitted on every involved device's segment to mark the end of the session. The newly paired Primary and Secondary devices will establish a connection and exchange any ongoing CAN traffic.*
10. *Secondary device(s) not selected for pairing will continue to try and establish a connection to any previously paired Primary device or become idle (if it was previously paired to the Primary device which is now being paired to another Secondary device).*

A.2.3 Pairing procedure summary - Discovery Pairing

The following *step-by-step* instructions summarize the necessary steps to pair/associate a Kvaser Air Bridge Primary device with one Kvaser Air Bridge Secondary device.

1. Connect the Air Bridge Primary device to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Primary device that shall discover Air Bridge Secondary device(s) and eventually pair with one *selected* Air Bridge Secondary device.
3. *(optional)* Connect the Air Bridge Secondary device(s) that will utilise **pairing codes**, **user status** and/or **custom data**, to a CAN-bus segment from where it can be accessed from a user application (that can assign pairing code seed/user status/custom data).

4. **Power ON** the Air Bridge Secondary device(s).
5. *(optional)* From a user application (in the Secondary device(s) segment), send message **#1** (see Table 161 on Page 63) with a *user-specified* **user status** to each Secondary device desired.
6. *(optional)* From a user application (in the Secondary device(s) segment), send message **#4** (see Table 161 on Page 63) with a *user-specified* **custom data** to each Secondary device desired.
7. *(optional)* If *user-defined*, non-default **pairing codes** will be utilised to lock out unauthorized devices from the pairing session, set **Primary pairing code seed** and **Secondary pairing code seed** for both the Primary and Secondary devices respectively, in each CAN-bus segment, by sending message **#2** and **#3** (see Table 161 on Page 63). **NOTE!** Use the same seeds on all devices intended to communicate with each other.
8. From a user application, send message **#6** (see Table 161 on Page 63) with parameter byte=`0x01` (Activate) to the Primary device. *This message causes the Primary device to switch to Discovery Pairing mode and 1) any ongoing CAN-traffic is blocked from RF transmission and 2) only the PWR LED indicator will be lit (green).*
9. *All powered Secondary devices (within reach) that do not already have an established connection with another Primary device, will detect the mode change and start reporting their respective RF Identifiers & user status to the Primary device. When a Secondary device has reported itself, the PWR LED (green) indicator will be constantly lit and the RF LED (blue) indicator will flash ON and OFF typically every half second .*
10. *The Primary device discovers Secondary device(s) and reports them with message **#9** (see Table 161 on Page 63) to a user application with approximately 200 ms period. There will be one message #9 for each discovered Secondary device, every reporting period. The CAN LED (yellow) will flash for every discovered Secondary device being reported.*
11. *(optional)* If any reported Secondary device's **custom data** is needed (in addition to the reported **user status**) for deciding the *pairing selection*, let the user application request the **custom data** using message **#12** (see Table 161 on Page 63) with concerned Secondary device's reported **RF Identifier** as parameter.
12. From a user application, select the Secondary device that shall be paired with the Primary device, by sending message **#7** (see Table 161 on Page 63) with the Secondary device's reported **RF Identifier** as parameter. ***NOTE!** The user application may utilise the reported **user status** and **custom data** as criteria to automatically select one of several Secondary devices.*
13. *When the pairing is acknowledged, all devices involved will leave Pairing mode and return to standard (CAN Traffic) mode. Message **#14** (see Table 161 on Page 63) will be transmitted on every involved device's segment to mark the end of the session. The newly paired Primary and Secondary devices will establish a connection and exchange any ongoing CAN traffic.*
14. *Secondary device(s) not selected for pairing will continue to try and establish a connection to any previously paired Primary device or become idle (if it was previously paired to the Primary device which is now being paired to another Secondary device).*

A.2.4 Un-pairing procedure summary - Asynchronous un-pairing

Prerequisite: An Air Bridge Primary device has been previously paired/associated with an Air Bridge Secondary device (see Section A.2.2, Pairing procedure summary - Targeted Pairing, on Page 64, Section A.2.3, Pairing procedure summary - Discovery Pairing, on Page 64).

1. Connect the Air Bridge Primary device that shall un-pair/disassociate from a previously paired Air Bridge Secondary device, to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Primary device.
3. From a user application, command **un-pairing** by sending message **#5** (see Table 161 on Page 63) with parameter bytes (timeout=`0`, targeted device RF Id=`00000`) to the Primary device. *This message causes the Primary device to asynchronously un-pair from a currently paired Secondary device.* Message **#14** (see Table 161 on Page 63) will be transmitted on the Primary device's segment to mark the end of the session.

A.2.5 Un-pairing procedure summary - Synchronous un-pairing

Prerequisite: An Air Bridge Primary device has been previously paired/associated with an Air Bridge Secondary device (see Section A.2.2, Pairing procedure summary - Targeted Pairing, on Page 64, Section A.2.3, Pairing procedure summary - Discovery Pairing, on Page 64).

1. Connect the Air Bridge Primary device that shall un-pair/disassociate from a previously paired Air Bridge Secondary device, to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Primary device.
3. From a user application, send message **#6** (see Table 161 on Page 63) with parameter byte=`0x01` (Activate) to the Primary device. *This message causes the Primary device to switch to Discovery Pairing mode and 1) any ongoing CAN-traffic is blocked from RF transmission and 2) only the PWR LED indicator will be lit (green).*
4. *All powered Secondary devices (within reach) that do not already have an established connection with another Primary device, will detect the mode change and start reporting their respective RF Identifiers & user status to the Primary device.*
5. From a user application, command **un-pairing** by sending message **#7** (see Table 161 on Page 63) with RF Id parameter set to `0x00000`.
6. *The Primary device un-pairs/disassociates the currently paired Secondary device, leaves the Pairing mode and returns to standard (CAN Traffic) mode. However, since the Primary device is no longer paired with any Secondary device, it will stay **radio-silent**. Message **#14** (see Table 161 on Page 63) will be transmitted on the Primary device's segment to mark the end of the session.*
7. *Any Secondary device that entered the Discovery Pairing mode (step 4) will eventually **timeout** from the session and return to standard (CAN Traffic) mode. Message **#14** (see Table 161 on Page 63) will be transmitted on their respective segments to mark the end of the session.*
8. ***NOTE!** A previously paired Secondary device that **did not** enter Discovery Pairing mode (step 4) (because it was not within reach or powered-up) will automatically un-pair/disassociate from the Primary device the next time the Secondary device detects the very same Primary device.*

A.2.6 Message examples

a) User application assigns a **user status** (value=`0xF`) to a connected Air Bridge Secondary device:

Dir	DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
Tx	8	0x04	0x03	0x0A	0x00	0x0F	-	-	-
Rx	8	0x03	0x03	0x0A	0x00	-	-	-	-

b) User application assigns a **custom data** (value=`0xB0BBAF00`) to a connected Air Bridge Secondary device:

Dir	DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
Tx	8	0x07	0x03	0xEA	0x02	0xB0	0xBB	0xAF	0x00
Rx	8	0x03	0x03	0xEA	0x02	-	-	-	-

c) User application assigns a **Primary pairing code seed** (value=`0xFFFFFFFB`) to a connected Air Bridge Primary or Secondary device:

Dir	DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
Tx	8	0x07	0x03	0xEA	0x00	0xFF	0xFF	0xFF	0xFB
Rx	8	0x03	0x03	0xEA	0x00	-	-	-	-

d) User application assigns a **Secondary pairing code seed** (value=`0x0000A6A9`) to a connected Air Bridge Primary or Secondary device:

Dir	DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
Tx	8	0x07	0x03	0xEA	0x01	0x00	0x00	0xA6	0xA9
Rx	8	0x03	0x03	0xEA	0x01	-	-	-	-

e) User application enables Discovery Pairing mode on an Air Bridge Primary device:

Dir	DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
Tx	8	0x04	0x06	0xCA	0xD1	0x01	-	-	-
Rx	8	0x03	0x06	0xCA	0xD1	-	-	-	-

f) Air Bridge Primary device **reports** a detected Secondary device with RF Identifier=**11012** to user application:

Dir	DLC	SIL	RSID	DID$_1$	DID$_0$	D$_0$	D$_1$	D$_2$	D$_3$
Rx	8	0x07	0x05	0xAE	0xD1	0x0F	0x11	0x10	0x12

Here, device 11012 is the currently paired device, with user status=`F`.

g) User application **requests** the custom data of the reported Secondary device (with RF Identifier=**11012**) from the Air Bridge Primary device:

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**07	**0x**05	**0x**AD	**0x**00	**0x**00	**0x**01	**0x**10	**0x**12
Rx	8	**0x**07	**0x**05	**0x**AD	**0x**00	**0x**B0	**0x**BB	**0x**AF	**0x**00

Here, custom data=B0BBAF00 is returned regarding device 11012.

h) User application **selects** Secondary device with RF Identifier=**11012** to be paired with the Primary device:

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**07	**0x**06	**0x**CA	**0x**D2	**0x**00	**0x**01	**0x**10	**0x**12
Rx	8	**0x**03	**0x**06	**0x**CA	**0x**D2	-	-	-	-

i) User application **un-pairs/dissociates** currently paired Secondary device from the Primary device (using RF Identifier value=`0x00000`):

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**07	**0x**06	**0x**CA	**0x**D0	**0x**00	**0x**00	**0x**00	**0x**00
Rx	8	**0x**03	**0x**06	**0x**CA	**0x**D0	-	-	-	-

j) User application **disables** Discovery Pairing mode on an Air Bridge Primary device before selecting a Secondary device for pairing:

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**04	**0x**06	**0x**CA	**0x**D1	**0x**00	-	-	-
Rx	8	**0x**03	**0x**06	**0x**CA	**0x**D1	-	-	-	-

k) User application enables Targeted Pairing mode on an Air Bridge Primary device and commands **pairing** with a **targeted** Air Bridge Secondary device (RF Identifier=**11012**) within 15 seconds (timeout=**0F**).

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**07	**0x**06	**0x**CA	**0x**D0	**0x**0F	**0x**01	**0x**10	**0x**12
Rx	8	**0x**03	**0x**06	**0x**CA	**0x**D0	-	-	-	-

l) User application enables Pairing mode on an Air Bridge Primary device and commands immediate, asynchronous **un-pairing** from the currently paired Air Bridge Secondary device (RF Identifier=**00000**, timeout=**0**).

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**07	**0x**06	**0x**CA	**0x**D0	**0x**00	**0x**00	**0x**00	**0x**00
Rx	8	**0x**03	**0x**06	**0x**CA	**0x**D0	-	-	-	-

m) User application request current **Operational mode** on an Air Bridge device. The response indicates that the Air Bridge device is in **CAN Traffic** mode.

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Tx	8	**0x**03	**0x**05	**0x**A0	**0x**00	-	-	-	-
Rx	8	**0x**04	**0x**05	**0x**A0	**0x**00	**0x**00	-	-	-

n) Air Bridge Primary device **reports** that a session is completed and that it is paired with Secondary device with RF Identifier=**11014**, to user application:

Dir	DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
Rx	8	**0x**07	**0x**05	**0x**AE	**0x**D5	**0x**00	**0x**01	**0x**10	**0x**14

Here, device 11014 is the currently paired Secondary device.

A.3 NAK example

DLC	SIL	RSID	DID_1	DID_0	D_0	D_1	D_2	D_3
8	**0x**07	**0x**06	**0x**CA	**0x**D2	**0x**00	**0x**08	**0x**01	**0x**03

Table 162: Command frame for **Select pairing device** with Rf Id=0x80103.

In **Discovery Pairing mode**, if selecting a pairing device that is **not** discovered, the response will be a NAK with NRC=0x80 (NRC_INVALID_PARAMETER) for SID=0x6 (*unused bytes of the frame are typically padded with 55*).

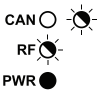
DLC	SIL	RNAK	SID	NRC	—	—	—	—
8	**0x**03	**0x**0F	**0x**06	**0x**72	**0x**55	**0x**55	**0x**55	**0x**55

Table 163: NAK with NRC=0x72 for SID=0x6.

B Appendix B: LED UI examples

B.1 LED behaviour in Pairing mode

When activating **Targeted Pairing** mode or **Discovery Pairing** mode, the Air Bridge LED light behaviour is reconfigured to signal the current state of the ongoing discovery and pairing operation as follows:



- **CAN** (yellow) will be turned OFF indicating that no CAN traffic is routed through the Air Bridge device. On the Primary device, however, the LED will blink when/if it detects and reports any Secondary devices on the CAN-bus. The LED blinks more intensely the more Secondary devices are detected and reported. Hence, the **CAN** LED only indicates data being transmitted from the Air Bridge device to the CAN bus.
- **RF** (blue) will flash ON and OFF *approx.* every half second. On the Primary device, this is a visible acknowledgment that a new Pairing session has been initiated. On the Secondary device, it is a visible acknowledgment that the Secondary device has responded to the mode transition and has also been accepted into the initiated session.

  NOTE! Should the periodic RF blinking pattern become partially irregular, this indicates that the respective Secondary device is experiencing interference from other transmitters in the vicinity, and that the reception was *momentarily* disturbed (as can occur during standard operational mode (CAN Traffic) as well).
- **PWR** (green) will be constantly illuminated indicating the device *is* powered up.

When deactivating **Discovery Pairing** mode or when a pairing procedure is concluded, the LED light behaviour will return to the behaviour of *standard operation* (see Kvaser Air Bridge User's Guide).

C Document Revision History

Version history for document: Kvaser Air Bridge Management Interface description

Revision	Date	Changes
1.0	2023-07-07	Initial version.
1.1	2023-09-06	Updated for 2nd Air Bridge X prototype release.
1.2	2023-11-13	Updated for 3rd Air Bridge X prototype release.
1.3	2023-12-18	Updated for Air Bridge release candidate.
1.4	2024-04-24	Updated for Air Bridge release.
1.5	2024-05-08	Updated for Air Bridge release.
1.6	2024-08-27	Updated for Air Bridge release.