



Kvaser Air Bridge Management Interface description



Copyright 2024 Kvaser AB, Mölndal, Sweden
<https://www.kvaser.com>

Printed Wednesday 8th May, 2024

We believe that the information contained herein was accurate in all respects at the time of printing. Kvaser AB cannot, however, assume any responsibility for errors or omissions in this text. Also note that the information in this document is subject to change without notice and should not be construed as a commitment by Kvaser AB.

(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 Discovery/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: 0xE100: Read Management Interface enable status . . .	31
5.2.16	DID: 0xE101: Read Management Interface receiver identifier	31
5.2.17	DID: 0xE102: Read Management Interface sender identifier .	32
5.2.18	DID: 0xEA02: Read custom value	32
5.2.19	DID: 0xEA03: Read heartbeat period	33
5.3	DIDs for SID: 0x3 (Write data)	34
5.3.1	DID: 0x0A00: Write user status	34
5.3.2	DID: 0xE021: Write role	35
5.3.3	DID: 0xE022: Write transmit power level	35
5.3.4	DID: 0xE0D1: Write CAN speed	36
5.3.5	DID: 0xE100: Write Management Interface enable status . . .	36
5.3.6	DID: 0xE101: Write Management Interface receiver identifier .	37
5.3.7	DID: 0xE102: Write Management Interface sender identifier .	37
5.3.8	DID: 0xEA00: Write Master pairing code seed	38
5.3.9	DID: 0xEA01: Write Slave pairing code seed	38
5.3.10	DID: 0xEA02: Write custom value	39
5.3.11	DID: 0xEA03: Write heartbeat period	39
5.4	DIDs for SID: 0x5 (Runtime status)	40
5.4.1	DID: 0xAE00: Link status report	40
5.4.2	DID: 0xAE01: Version alarm	41
5.4.3	DID: 0xAE02: Device hearbeat	41
5.4.4	DID: 0xAED1: Device report	42
5.4.5	DID: 0xAED2: No pairing code seed	43
5.4.6	DID: 0xAED3: Invalid pairing code	43
5.4.7	DID: 0xAED4: Discovery/Pairing session timed out	44
5.4.8	DID: 0xA000: Device operational mode	44
5.4.9	DID: 0xA001: Device link status	45
5.4.10	DID: 0xAD00: Slave device custom value	45
5.4.11	DID: 0xAD01: Pairing code seed defined	46
5.5	DIDs for SID: 0x6 (Runtime configuration)	47
5.5.1	DID: 0xCA00: Activate link status report	47
5.5.2	DID: 0xCA01: Set link parameters	48
5.5.3	DID: 0xCA02: Activate version alarm report	48
5.5.4	DID: 0xCA03: Set runtime session heartbeat period	49
5.5.5	DID: 0xCAD0: Activate Discovery/Pairing mode and pair with preferred device	49
5.5.6	DID: 0xCAD1: Activate Discovery/Pairing mode	50
5.5.7	DID: 0xCAD2: Select pairing device/Un-pair	51
A	Appendix A: Usage examples	52
A.1	Command example for re-configuring Air Bridge role	52
A.1.1	Messages	52

A.1.2	Role re-configuration procedure summary	52
A.1.3	Message examples	52
A.2	Command examples for Discovery/Pairing mode	53
A.2.1	Messages	53
A.2.2	Pairing procedure summary - Preferred pairing	53
A.2.3	Pairing procedure summary - Indecisive pairing	54
A.2.4	Un-pairing procedure summary - Asynchronous un-pairing	55
A.2.5	Un-pairing procedure summary - Indecisive un-pairing	56
A.2.6	Message examples	56
A.3	NAK example	58
B	Appendix B: LED UI examples	60
B.1	LED behaviour in Discovery/Pairing mode	60
C	Document Revision History	61

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.

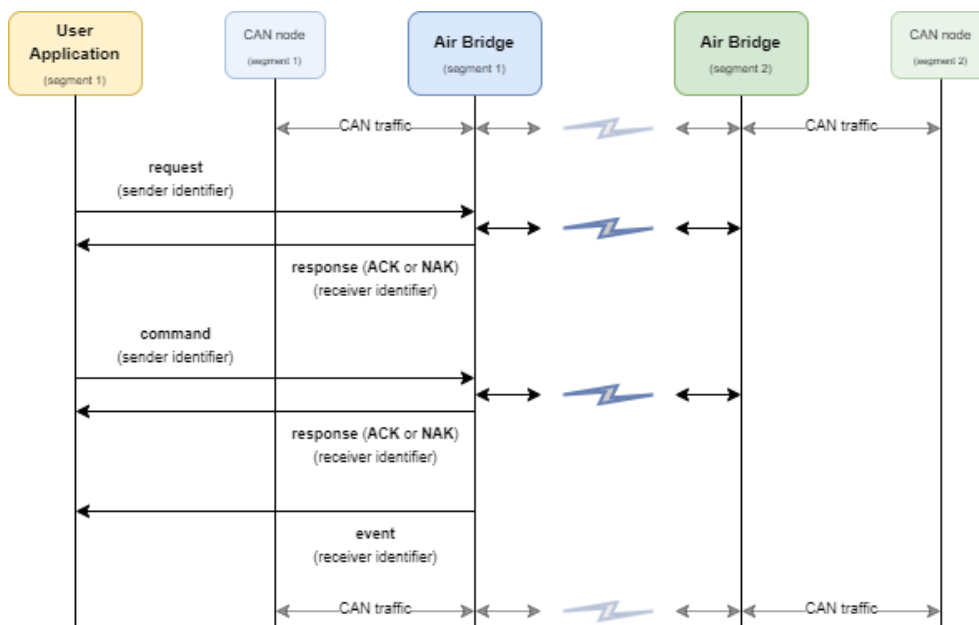


Figure 1: Interaction between a user application and Kvaser Air Bridge over the Management Interface protocol.

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 Discovery/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 **Master** device paired/associated with an Air Bridge **Slave** device in the other CAN segment.

An Air Bridge device is factory-configured as a Slave 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 Master*.

A detailed description of this role re-configuration can be found in Section A.1, Command example for re-configuring Air Bridge role, on Page 52.

2.5 Understanding Discovery/Pairing mode

With **Kvaser Air Bridge**, it becomes possible for a Master device to, during operation, **discover** and **re-associate** with different Slave devices from various CAN segments.

This process - pairing/re-associating - is achieved when a user application places an Air Bridge Master device into the so-called **Discovery/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, and the Master device begins searching for Slave devices in the vicinity

(that currently do not have an established connection with any other Master device).

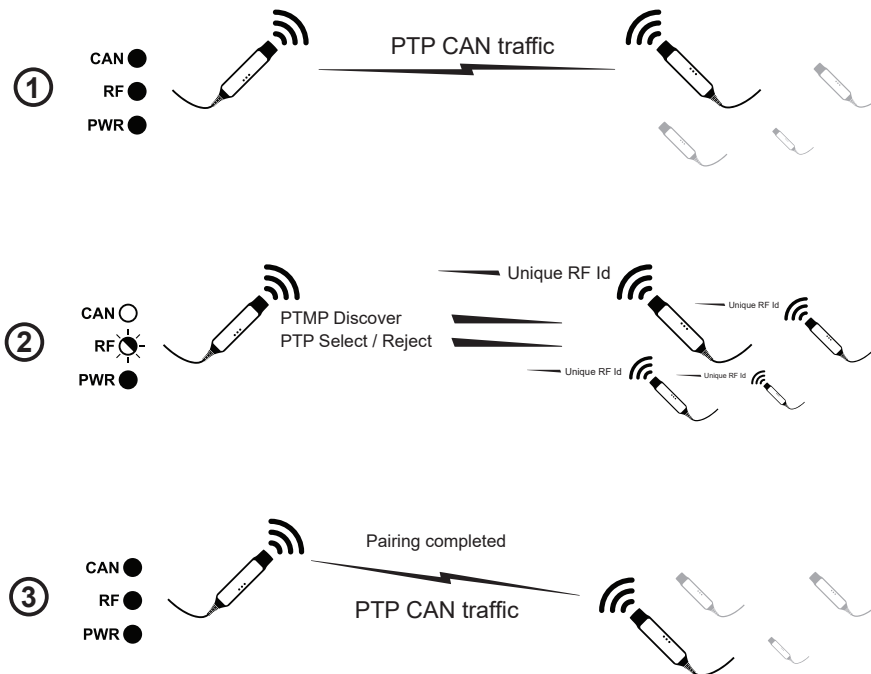


Figure 2: An established *point-to-point* connection (1) is paused and Discovery/Pairing mode is entered (2) in which the Master device pairs/re-associates with another reported Slave device (3).

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

- The Air Bridge Master **pairs with a preferred** Air Bridge Slave device. This method is typically used when the Slave device to be paired is determined in the context. That is, the Master side knows the Slave device's identifier, and the device is expected to be within radio range.
- Pairing based on reported discoveries *i.e.* a user application **selects one** out of possibly many reported Air Bridge Slave 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 **Discovery/Pairing mode** is initiated according to alternative **a**, with a predefined, **preferred** Slave device for pairing, the pairing/association is executed as soon as this specific Slave device is discovered by the Master device. Hence, if the *preferred* Slave is not discovered, the discovery procedure proceeds until a predefined time-out condition is met, until a user application selects another Slave device for pairing (than the *preferred* one) or until a user application deactivates the Discovery/Pairing mode, for any reason.

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

To ensure that only desired Slave devices respond to the Master 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 Discovery/Pairing session.

Detailed descriptions of the two pairing procedures can be found in Section A.2, Command examples for Discovery/Pairing mode, on Page 53.

NOTE! It is recommended to perform *only one* Discovery/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 Slave devices.

2.6 Terminology

- **ACK** - Acknowledgement.
- **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.
- **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.
- **Master 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 **Discovery/Pairing** mode in which a Master device can associate/pair itself with a discovered Slave device.
- **Pairing** - The procedure in which a Kvaser Air Bridge Master device *pairs/associates* a Kvaser Air Bridge Slave device.
- **PTMP (Point-to-Multi-Point)** - Refers to that data is transmitted from a single source (an Air Bridge Master device) to multiple destinations (several Air Bridge Slave devices) simultaneously.
- **PTP (Point-to-Point)** - Refers to that data is transmitted between two endpoints directly (*one* Air Bridge Master device and *one* Air Bridge Slave 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 Master device *pairs/associates* with a Kvaser Air Bridge Slave device. Typically a *pairing/association* uses the Master 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 **Master** or **Slave**. A Kvaser Air Bridge **pair** typically consists of *one* Master device and *one* Slave device. A Kvaser Air Bridge **set** typically consists of *one* Master device and *several* Slave 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.

- **Slave 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 Master device *un-pairs/disassociates* from a Kvaser Air Bridge Slave 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! A user may redefine the arbitration identifier set with values tailored to the intended CAN segment.

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:** Bit₇ (**n**) is always **0**, indicating that the frame represents a new message. Bit₀ - Bit₆ (**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:** Bit₇ (**n**) is always **0**, indicating that the frame represents a new message. Bit₀ - Bit₆ (**s**) specify the total length where the value range is [**8,127**] (dec), indicating that the payload will span across several frames.
- **Subsequent frame:** Bit₇ (**n**) is always **1**, indicating that the frame represents a subsequent frame of a *multi*-framed message. Bit₀ - Bit₆ (**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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
0bnsysssss	0b000yxxxx	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

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
0bnsysssss	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

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

Byte₀: 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.

- Bit₇ (**n**) = Sequence Indicator [0=New *single* frame or first frame of a *multi*-framed message, 1=Concatenation frame of a *multi*-framed message]
- Bit₆₋₀ (**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 **0x80** = **0b10000000** is recognized, meaning there is a new message (**n**-bit = 0).

Byte₁: Routing + SID (RSID)

Byte₁ holds a **routing** bit and the **SID** (Service 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.

- Bit₇ = 0 (Reserved)
- Bit₆ = 0 (Reserved)
- Bit₅ = 0 (Reserved)
- Bit₄ = Routing [0=local, 1=remote]
- Bit₃₋₀ = SID

Byte₂ - Byte₃: DID

Byte₂ and Byte₃ holds the **DID** (Data Identifier) which states the request data parameter of the specified SID.

Byte₄ - Byte₇: Data

Byte₄ to Byte₇ are used for passing parameter/value data. The **remaining length** of the data is specified by **Sequence Indicator + Length** (Byte₀) (*including the length of the SID and DID bytes (3)*).

Byte₁ - Byte₇ in a concatenated frame: Data

Byte₁ to Byte₇ are used for passing data in a *multi*-framed message. The **remaining length** of the data is specified by **Sequence Indicator + Length** (Byte₀).

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 Byte₂=0x03 and Byte₃=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 51). The **RF Id (0x11012)** is then encoded (**blue**) and padded (**red**) as follows:

SIL	RSID	DID ₀	DID ₁	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 58).

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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0b00001100	0b000yxxxx	0xXX	0xXX	0xE1	0x8C	0x11	0x05

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	0b10000101	0x00	0x10	0x41	0x11	0xEA	0x55	0x55

Table 4: Second response frame out of two consecutive frames with total length of 12 bytes. NOTE¹ The 'Sequenced + Length (SIL)'-byte states the **remaining** number of data bytes in the frame. NOTE² The last two *unused* bytes are padded with (red) 0x55.

3.7 NAK

A negative response 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	0b00000011	0b000y1111	0b0000xxxx	0x00	—	—	—	—

Table 5: ACK/NAK frame format.

Byte₀: Sequence Indicator + Length (SIL)

Length of the frame, always 3.

Byte₁: Routing + NAK (RNAK)

Byte₁ holds a **routing** bit (y) and the NAK discriminator 0b1111. The routing bit states if the concerned NAK was routed from the *local* Air Bridge device or the *remote* side Air Bridge device.

- Bit₇ = 0 (Reserved)
- Bit₆ = 0 (Reserved)
- Bit₅ = 0 (Reserved)
- Bit₄ = Routing [0=local, 1=remote]
- Bit₃₋₀ = NAK (1111)

Byte₂: SID

Byte₂ specifies the **SID** (Service Identifier) of the request being negatively acknowledged.

Byte₃: NRC

Byte₃ 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

Table 6: Negative Response Codes (NRC)

Byte₄ - Byte₇: 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	Master, Slave	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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x01	0x01	0x00	—	—	—	—

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	Master, Slave	User status	0	Standard user
0D00	Master, Slave	Runtime session CAN speed	0	Standard user
0D01	Master, Slave	Version scan result	0	Standard user
E000	Master, Slave	Manufacturing date	-	Standard user
E001	Master, Slave	Device serial number	Unique	Standard user
E002	Master, Slave	EAN product code	014947 (hex)	Standard user
E010	Master, Slave	Hardware version	-	Standard user
E011	Master, Slave	Air Bridge application version	-	Standard user
E021	Master, Slave	Role override	0	Standard user
E022	Master, Slave	Transmit power level override	0	Standard user
E040	Master, Slave	Local RF identifier	Unique	Standard user
E041	Master, Slave	Remote (pairing) RF identifier	0	Standard user
E0D0	Master, Slave	RF configuration	0100070A (hex)	Standard user
E0D1	Master, Slave	CAN speed	0	Standard user
E100	Master, Slave	Management Interface enable status	1	Standard user
E101	Master, Slave	Management Interface receiver identifier	1BFFF8F1 (hex)	Standard user
E102	Master, Slave	Management Interface sender identifier	1BC78FFF (hex)	Standard user
EA02	Master, Slave	Custom value	0	Standard user
EA03	Master, Slave	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 of a **Slave** device is reflected in the **Device report** message when in **Discovery/Pairing mode** (see Section 5.4.4, DID: 0xAED1: Device report, on Page 42).
- If no **user status** value is previously written by the application (see Section 5.3.1, DID: 0x0A00: Write user status, on Page 34), the *default* value is **0**.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0x0A	0x00	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x02	0x0A	0x00	0b0000xxxx	—	—	—

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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0x0D	0x00	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x02	0x0D	0x00	0xYY	—	—	—

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 **LEDs** 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 41) 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 48), both counters are reset after every report.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0x0D	0x01	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x05	0x02	0x0D	0x01	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 <i>valid</i> version frames	0 - FF (hex)
Invalid number (Z)	1 byte	Number of <i>invalid</i> 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 ₀	DID ₁	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 ₀	DID ₁	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.

DLC	SIL	RSID	DID ₀	DID ₁	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 ₀	DID ₁	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	0 - FFFFFFFF (hex)

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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x02	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x02	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 - FFFFFF (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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x10	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x10	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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x11	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x11	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 35). Hence, a Master device may be reconfigured as a Slave and *vice versa*.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x21	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x21	0b0000xxxx	—	—	—

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

Data	Type	Description	Value
Role override (x)	1 nibble	Not overridden (<i>default</i>)	0
		Overridden as Master	1
		Overridden as Slave	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 35).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x22	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x22	0b000000xx	—	—	—

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 overridden (<i>default</i>)	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 Master device uses the **local RF Id** as *the identifier* with which it identifies itself in the transmitted radio frame headers. *An Air Bridge Slave device however, typically uses another identifier than its **local RF Id**, namely the RF Id of its associated Master.*
- The Air Bridge device's **local 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).
- In **Discovery/Pairing mode**, an Air Bridge Slave device reports itself to the initiating Master device using its *unique* identifier, in contrast to the currently set RF Id which it may have adopted from a Master device in a previous Discovery/Pairing mode session.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x40	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x40	0x00	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

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)	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* Master or Slave Air Bridge device.
- The **remote RF Id** of a *paired/associated* Master device equals its **local RF Id**.
- The **remote RF Id** of a *paired/associated* Slave device equals the **local RF Id** of the associated Master 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0x41	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0x41	0x00	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

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

Data	Type	Description	Value
Remote RF Id (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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0xD0	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0xD0	0b0000xxxx	0x00	0x07	0b00ppppmm

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
		Master	1
		Slave	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.

NOTE! 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.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE0	0xD1	—	—	—	—

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE0	0xD1	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 (<i>default</i>)	0
		125 kbit/s	1
		250 kbit/s	2
		500 kbit/s	3
		1 Mbit/s	4
		Reserved	5-FFFF FFFF (hex)

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

5.2.15 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 5.3.6, DID: 0xE101: Write Management Interface receiver identifier, on Page 37 and Section 5.3.7, DID: 0xE102: Write Management Interface sender identifier, on Page 37).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE1	0x00	—	—	—	—

Table 53: Message structure for DID 0xE100 (Read Management Interface enable status).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE1	0x00	0x00	0x00	0x00	0b0000000s

Table 54: 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 55: Response data for DID 0xE100 (Read Management Interface Enable status)

5.2.16 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 5.3.6, DID: 0xE101: Write Management Interface receiver identifier, on Page 37), the factory preset identifier (**0x1BFFF8F1**) is read (see Section 3.1, Receiver & Sender identifiers, on Page 13).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE1	0x01	—	—	—	—

Table 56: Message structure for DID 0xE101 (Read Management Interface receiver identifier).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE1	0x01	0xYY	0xYY	0xYY	0xYY

Table 57: Message structure of a successful response for DID 0xE101 (Read Management Interface Receiver identifier). **Y** symbolizes the returned data.

Data	Type	Description	Value
Receiver id (Y)	4 bytes	Management Interface receiver identifier	0 - FFFFFFFF (hex)

Table 58: Response data for DID 0xE101 (Read Management Interface receiver identifier)

5.2.17 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 5.3.7, DID: 0xE102: Write Management Interface sender identifier, on Page 37), the factory preset identifier (**0x1BC78FFF**) is read (see Section 3.1, Receiver & Sender identifiers, on Page 13).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xE1	0x02	—	—	—	—

Table 59: Message structure for DID 0xE102 (Read Management Interface sender identifier).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xE1	0x02	0xYY	0xYY	0xYY	0xYY

Table 60: Message structure of a successful response for DID 0xE102 (Read Management Interface sender identifier). **Y** symbolizes the returned data.

Data	Type	Description	Value
Sender id (Y)	4 bytes	Management Interface Sender identifier	0 - FFFFFFFF (hex)

Table 61: Response data for DID 0xE102 (Read Management Interface sender identifier)

5.2.18 DID: 0xEA02: Read custom value

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

- The **custom value** can be read repeatedly in runtime.
- The **custom value** of an *un-paired but discovered* Slave device can be retrieved from the Master device when in **Discovery/Pairing mode** (see Section A.2.6, Message examples, on Page 56).
- If no **custom value** is previously set by the application (see Section 5.3.10, DID: 0xEA02: Write custom value, on Page 39), the *default* value is **0**.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xEA	0x02	—	—	—	—

Table 62: Message structure for DID 0xEA02 (Read custom value).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x02	0xEA	0x02	0xXX	0xXX	0xXX	0xXX

Table 63: Message structure of a successful response for DID 0xEA02 (Read custom value). X symbolizes the returned data.

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

Table 64: Response data for DID 0xEA02 (Read custom value)

5.2.19 DID: 0xEA03: 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 5.3.11, DID: 0xEA03: Write heartbeat period, on Page 39), 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 42) every **heartbeat period** millisecond.
- The **heartbeat period** can be read repeatedly in runtime.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x02	0xEA	0x03	—	—	—	—

Table 65: Message structure for DID 0xEA03 (Read heartbeat period).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x05	0x02	0xEA	0x03	0xXX	0xXX	—	—

Table 66: 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 (<i>default</i>)	0
		Reserved	1 - 99 (dec)
		Valid period (ms)	100 - 60000 (dec)
		Reserved	60001 - 65535 (dec)

Table 67: Response data for DID 0xEA03 (Read heartbeat period)

5.3 DIDs for SID: 0x3 (Write data)

ID (Hex)	Role	Description	Access
0A00	Master, Slave	User status	Standard user
E021	Master, Slave	Role override	Standard user
E022	Master, Slave	Transmit power level override	Standard user
E0D1	Master, Slave	CAN speed	Standard user
E100	Master, Slave	Management Interface enable status	Standard user
E101	Master, Slave	Management Interface receiver identifier	Standard user
E102	Master, Slave	Management Interface sender identifier	Standard user
EA00	Master, Slave	Master pairing code seed	Standard user
EA01	Master, Slave	Slave pairing code seed	Standard user
EA02	Master, Slave	Custom value	Standard user
EA03	Master, Slave	Heartbeat period	Standard user

Table 68: 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 of a **Slave** device is reflected in the **Device report** message when in **Discovery/Pairing mode** (see Section 5.4.4, DID: 0xAED1: Device report, on Page 42).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x03	0x0A	0x00	0b0000ssss	—	—	—

Table 69: Message structure for DID 0x0A00 (Write user status).

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

Table 70: 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 (Slave) and instead reconfigure the device as a Master.
- A device is reset to the factory preset role (Slave) by storing the value 0.
- The stored value takes effect after a subsequent restart/power cycle.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x03	0xE0	0x21	0b0000yyyy	—	—	—

Table 71: Message structure for DID 0xE021 (Write role).

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

Table 72: 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x03	0xE0	0x22	0b000000yy	—	—	—

Table 73: 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 74: 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).

- 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xE0	0xD1	0xYY	0xYY	0xYY	0xYY

Table 75: 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-FFFF FFFF (hex)

Table 76: Parameters for DID 0xE0D1 (Write CAN speed)

5.3.5 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 5.3.6, DID: 0xE101: Write Management Interface receiver identifier, on Page 37 and Section 5.3.7, DID: 0xE102: Write Management Interface sender identifier, on Page 37).
- **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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xE1	0x00	0x00	0x00	0x00	0b0000000s

Table 77: Message structure for DID 0xE100 (Write Management Interface enable status).

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

Table 78: Parameters for DID 0xE100 (Write Management Interface enable status)

5.3.6 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).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xE1	0x01	0xYY	0xYY	0xYY	0xYY

Table 79: Message structure for DID 0xE101 (Write Management Interface Receiver identifier).

Parameter	Type	Description	Value
Receiver Id (Y)	4 bytes	User-specified arbitration identifier	0-FFFFFFFF (hex)

Table 80: Parameters for DID 0xE101 (Write Management Interface Receiver identifier)

5.3.7 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).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xE1	0x02	0xYY	0xYY	0xYY	0xYY

Table 81: Message structure for DID 0xE102 (Write Management Interface sender identifier).

Parameter	Type	Description	Value
Sender Id (Y)	4 bytes	User-specified arbitration identifier	0-FFFFFFFF (hex)

Table 82: Parameters for DID 0xE102 (Write Management Interface sender identifier)

5.3.8 DID: 0xEA00: Write Master pairing code seed

5.3.9 DID: 0xEA01: Write Slave 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 **Discovery/Pairing mode**.

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

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xEA	0x00	0xYY	0xYY	0xYY	0xYY

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xEA	0x01	0xYY	0xYY	0xYY	0xYY

Table 83: Message structure for DID 0xEA00, 0xEA01 (Write Master and Slave pairing code seeds).

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

Table 84: Parameters for DID 0xEA00, 0xEA01 (Write Master and Slave pairing code seeds).

5.3.10 DID: 0xEA02: Write custom value

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

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x03	0xEA	0x02	0xYY	0xYY	0xYY	0xYY

Table 85: Message structure for DID 0xEA02 (Write custom value).

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

Table 86: Parameters for DID 0xEA02 (Write custom value)

5.3.11 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 heartbeat, on Page 41) every **heartbeat 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x05	0x03	0xEA	0x03	0xYY	0xYY	—	—

Table 87: 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 88: Parameters for DID 0xEA03 (Write heartbeat period)

5.4 DIDs for SID: 0x5 (Runtime status)

ID (Hex)	Service type	Role	Description	Access
AE00	Event	Master, Slave	Link status report	Standard user
AE01	Event	Master, Slave	Version alarm	Standard user
AE02	Event	Master, Slave	Device heartbeat	Standard user
AED1	Event	Master	Device report	Standard user
AED2	Event	Master, Slave	No pairing code seed	Standard user
AED3	Event	Master, Slave	Invalid pairing code	Standard user
AED4	Event	Master	Discovery/Pairing session timed out	Standard user
A000	Request	Master, Slave	Device operational mode	Standard user
A001	Request	Master, Slave	Device link status	Standard user
AD00	Request	Master	Slave device custom value	Standard user
AD01	Request	Master, Slave	Pairing code seed defined	Standard user

Table 89: 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 47).

- 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 48).
- 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 48).
- 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xAE	0x00	0b0000000y	—	—	—

Table 90: 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 91: 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 48).

- 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xAE	0x01	0xYY	—	—	—

Table 92: Message structure for DID 0xAE01 (Version alarm)

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

Table 93: Parameters for DID 0xAE01 (Version alarm)

5.4.3 DID: 0xAE02: Device heartbeat

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 5.3.11, DID: 0xEA03: Write heartbeat period, on Page 39 and Section 5.5.4, DID: 0xCA03: Set runtime session heartbeat period, on Page 49).

- 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 5.4.8, DID: 0xA000: Device operational mode, on Page 44).
- The **Link status** value states whether or not the device has an established link to its paired counterpart (see Section 5.4.9, DID: 0xA001: Device link status, on Page 45).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x05	0xAE	0x02	0b0000mmmm	0b0000ssss	—	—

Table 94: Message structure for DID 0xAE02 (Device heartbeat).

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

Table 95: Parameters for DID 0xAE02 (Device heartbeat)

5.4.4 DID: 0xAED1: Device report

Reports a detected Air Bridge **Slave** device when in Discovery/Pairing mode.

- The **User status** value reflects the current *user-defined* status of the referenced Air Bridge Slave device (see Section 5.3.1, DID: 0x0A00: Write user status, on Page 34).
- The **Paired** value states whether the referenced Slave device is the currently paired Slave device.
- The **Device RF Id** states the referenced Slave device’s unique identifier (*the device’s local RF Id*)

NOTE! Device report is only transmitted by an Air Bridge **Master** device.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x05	0xAE	0xD1	0b0000ssss	0b0000pyyyy	0byyyyyyyy	0byyyyyyyy

Table 96: 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	Device is not the paired Slave device	0
		Device <i>is</i> the currently paired Slave device	1
Device RF Id (y)	20 bits	Slave device's local RF Id	0 - FFFFF (hex)

Table 97: 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 Master or Slave device enters Discovery/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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xAE	0xD2	0b0000000y	—	—	—

Table 98: Message structure for DID 0xAED2 (No pairing code seed).

Data	Type	Description	Value
Seed Id (y)	1 bit	Master seed is not set	0
		Slave seed is not set	1

Table 99: 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 Discovery/Pairing session.

- On a Slave device, the event is triggered if a Master device is initiating a Discovery/Pairing session but uses an invalid code (*compared to the code generated in the Slave device*).
- On a Master device, the event is triggered if a Slave device reports itself in a Discovery/Pairing session but uses an invalid code (*compared to the code generated in the Master device*).
- The event is triggered at most once per second by either role.
- The **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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x05	0xAE	0xD3	0x00	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

Table 100: Message structure for DID 0xAED3 (Invalid pairing code)

Data	Type	Description	Value
RF Id (y)	20 bits	Device RF Id	0 - FFFFF (hex)

Table 101: Data for DID 0xAED3 (Invalid pairing code)

5.4.7 DID: 0xAED4: Discovery/Pairing session timed out

Reports that an initiated Discovery/Pairing session with a **preferred** Slave device, has timed out.

- On a Master device, the event is triggered if an ongoing session has exceeded (in time) the set *timeout* value.
- The **RF Id** states the concerned, **preferred** Slave device's unique identifier (Local RF Id).
- The **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).
- After a session timeout, the concerned device's **operational mode** is reverted to **CAN traffic** mode.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x05	0xAE	0xD4	0x00	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

Table 102: Message structure for DID 0xAED4 (Pairing session timed out)

Data	Type	Description	Value
RF Id (y)	20 bits	Preferred device's RF Id	1 - FFFFF (hex)

Table 103: Data for DID 0xAED4 (Pairing session timed out)

5.4.8 DID: 0xA000: Device operational mode

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x05	0xA0	0x00	—	—	—	—

Table 104: Message structure for DID 0xA000 (Device operational mode)

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xA0	0x00	0b0000xxxx	—	—	—

Table 105: Message structure of a successful response for DID 0xA000 (Device operational mode). **x** symbolizes the returned data.

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

Table 106: Response data for DID 0xA000 (Device operational mode)

5.4.9 DID: 0xA001: Device link status

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x03	0x05	0xA0	0x01	—	—	—	—

Table 107: Message structure for DID 0xA001 (Device link status)

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xA0	0x01	0b0000xxxx	—	—	—

Table 108: Message structure of a successful response for DID 0xA001 (Device link status). **x** symbolizes the returned data.

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

Table 109: Response data for DID 0xA001 (Device link status)

5.4.10 DID: 0xAD00: Slave device custom value

Requests the *user-specified* **custom value** of a detected Air Bridge **Slave** device when in Discovery/Pairing mode.

- The Slave device **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).

NOTE¹ Slave device custom value can only be requested from an Air Bridge **Master** device.

NOTE² From an already *paired* device, that device's **custom value** is retrieved with the Read service (see Section 5.2.18, DID: 0xEA02: Read custom value, on Page 32).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x05	0xAD	0x00	0x00	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

Table 110: Message structure for DID 0xAD00 (Slave device custom value)

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

Table 111: Parameters for DID 0xAD00 (Slave device custom value)

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x05	0xAD	0x00	0xXX	0xXX	0xXX	0xXX

Table 112: Message structure of a successful response for DID 0xAD00 (Slave device custom value). **X** symbolizes the returned data.

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

Table 113: Response data for DID 0xAD00 (Slave device custom value)

5.4.11 DID: 0xAD01: Pairing code seed defined

Requests if a *user-specified pairing code seed* has been stored in the Air Bridge device (see Section 5.3.8, DID: 0xEA00: Write Master pairing code seed, on Page 38, Section 5.3.9, DID: 0xEA01: Write Slave pairing code seed, on Page 38).

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xAD	0x01	0b0000000s	—	—	—

Table 114: Message structure for DID 0xAD01 (Pairing code seed defined)

Parameter	Type	Description	Value
Seed Id (s)	1 bit	Master seed	0
		Slave seed	1

Table 115: Parameters for DID 0xAD01 (Pairing code seed defined)

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x05	0xAD	0x01	0b0000000x	—	—	—

Table 116: 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 <i>not</i> defined	0
		Requested seed is defined	1

Table 117: Response data for DID 0xAD01 (Pairing code seed set)

5.5 DIDs for SID: 0x6 (Runtime configuration)

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

Table 118: 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 40).
- Link status reporting is deactivated by default.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x06	0xCA	0x00	0b0000000a	—	—	—

Table 119: 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 120: 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x05	0x06	0xCA	0x01	0xXX	0xYY	—	—

Table 121: 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 122: 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 41).
- Version alarm reporting is deactivated by default.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x06	0xCA	0x02	0b0000000a	—	—	—

Table 123: 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 124: 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 5.3.11, DID: 0xEA03: Write heartbeat period, on Page 39). 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 heartbeat, on Page 41) every **heartbeat period** millisecond.
- If a *zero* value **heartbeat period** is set, the heartbeat value stored in EEPROM (see Section 5.3.11, DID: 0xEA03: Write heartbeat period, on Page 39) 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x05	0x06	0xCA	0x03	0xYY	0xYY	—	—

Table 125: 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 126: Parameters for DID 0xCA03 (Set runtime session heartbeat period)

5.5.5 DID: 0xCAD0: Activate Discovery/Pairing mode and pair with preferred device

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

- A *non-zero RF Id* specifies the preferred Slave device’s unique identifier (local RF Id).
- A *zero RF Id* means that the initiating Master device shall instead un-pair/disassociate any currently paired Slave device.
- The **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).
- A *positive timeout* specifies a number of seconds [**1, 255**]. If the preferred Slave device is not discovered by the Air Bridge Master device within this time frame, or if a user application has not selected another device for pairing within this time frame, the Discovery/Pairing mode is deactivated and a Runtime status event (**Pairing session timed out**) is triggered (see Section 5.4.7, DID: 0xAED4: Discovery/Pairing session timed out, on Page 44).
- A *zero timeout* value means indefinite timeout, *i.e.*, the Discovery/Pairing procedure does *not* timeout.

NOTE! Discovery/Pairing mode can only be activated on an Air Bridge **Master** device.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x06	0xCA	0xD0	0xXX	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

Table 127: Message structure for DID 0xCAD0 (Activate Discovery/Pairing mode and pair with preferred device)

Parameter	Type	Description	Value
Timeout (X)	1 byte	Indefinite timeout	0
		Maximum discovery time (seconds)	1-FF (hex)
RF Id (y)	20 bits	Un-pair from any paired Slave device	0
		Preferred Slave device’s RF Id (20 bits)	1-FFFFF (hex)

Table 128: Parameters for DID 0xCAD0 (Activate Discovery/Pairing mode and pair with preferred device)

5.5.6 DID: 0xCAD1: Activate Discovery/Pairing mode

Activates/deactivates the Discovery/Pairing mode depending on parameter.

NOTE! Discovery/Pairing mode can only be activated on an Air Bridge **Master** device.

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x04	0x06	0xCA	0xD1	0b0000000a	—	—	—

Table 129: Message structure for DID 0xCAD1 (Activate Discovery/Pairing mode)

Parameter	Type	Description	Value
Activate (a)	1 bit	<i>Deactivate</i> Discovery/Pairing mode	0
		<i>Activate</i> Discovery/Pairing mode	1

Table 130: Parameters for DID 0xCAD1 (Activate Discovery/Pairing mode)

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

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

- A *non-zero* **RF Id** specifies the desired Slave 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 42).
- A *zero* **RF Id** means that the initiating Master device shall instead un-pair/disassociate any currently paired Slave device.
- The **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).

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

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x06	0xCA	0xD2	0x00	0b0000yyyy	0byyyyyyyy	0byyyyyyyy

Table 131: Message structure for DID 0xCAD2 (Select pairing device/Un-pair)

Parameter	Type	Description	Value
RF Id (y)	20 bits	Un-pair from any paired Slave device	0 (hex)
		Selected Slave device's RF Id (20 bits)	1-FFFFFF (hex)

Table 132: 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* Master device and *one* or *more* Slave devices. To facilitate easy setup, every Kvaser Air Bridge is delivered pre-configured as Slave 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 Master, *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 override
2	0x04	0x03 (Write data)	0xE021	Write role override

Table 133: 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 133) to read the current **role** value of the Air Bridge device.
4. From a user application, send message **#2** (see Table 133) 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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x03	0x02	0xE0	0x21	-	-	-	-
Rx	8	0x04	0x02	0xE0	0x21	00	-	-	-

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

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x04	0x03	0xE0	0x21	0x01	-	-	-
Rx	8	0x03	0x03	0xE0	0x21	-	-	-	-

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

A.2 Command examples for Discovery/Pairing mode

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

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

A.2.1 Messages

There are 13 types of messages involved in the Discovery/Pairing procedure:

#	SIL	SID (name)	DID	Message description
1	0x04	0x03 (Write data)	0x0A00	Write user status
2	0x07	0x03 (Write data)	0xEA00	Write Master pairing code seed
3	0x07	0x03 (Write data)	0xEA01	Write Slave pairing code seed
4	0x07	0x03 (Write data)	0xEA02	Write custom value
5	0x07	0x06 (Runtime configuration)	0xCAD0	Activate Discovery/Pairing mode and pair with preferred 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	Slave device custom value
13	0x04	0x05 (Runtime status)	0xAED4	Discovery/Pairing session timed out

Table 134: Messages involved in the Discovery/Pairing mode

A.2.2 Pairing procedure summary - Preferred pairing

The following *step-by-step* instructions summarize the necessary steps to pair/associate a Kvaser Air Bridge Master device with a *preferred* Kvaser Air Bridge Slave device.

1. Connect the Air Bridge Master device to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Master device that shall discover and pair with a *preferred* Air Bridge Slave device.
3. Connect the *preferred* Air Bridge Slave device to another CAN-bus segment.
4. **Power ON** the *preferred* Air Bridge Slave 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 **Master pairing code seed** and **Slave pairing code seed** for both the Master and Slave devices respectively, in each CAN-bus segment, by sending message **#2** and **#3** (see Table 134 on Page 53). **NOTE!** Use the same seeds on all devices intended to communicate with each other.
6. From a user application, send message **#5** (see Table 134 on Page 53) with parameter bytes (timeout, preferred device RF Id) to the Master device. *This message causes the Master device to switch to Discovery/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 Slave devices (within reach) that do not already have an established connection with another Master device, will detect the mode change and start reporting their respective RF Identifiers & User status to the Master device. When a Slave 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 Master device discovers Slave device(s) and when the preferred Slave device is recognized, it is automatically paired/associated with the initiating Master device.*
9. *When the pairing is acknowledged, all devices involved will leave Discovery/Pairing mode and return to standard (CAN Traffic) mode. The newly paired Master and Slave devices will establish a connection and exchange any ongoing CAN traffic.*
10. *Slave device(s) not selected for pairing will continue to try and establish a connection to any previously paired Master device or become idle (if it was previously paired to the Master device which is now being paired to another Slave device).*

A.2.3 Pairing procedure summary - Indecisive pairing

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

1. Connect the Air Bridge Master device to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Master device that shall discover Air Bridge Slave device(s) and eventually pair with one *selected* Air Bridge Slave device.
3. (*optional*) Connect the Air Bridge Slave device(s) that will utilise **Pairing codes**, **User status** and/or **Custom value**, to a CAN-bus segment from where it can be accessed from a user application (that can assign Pairing code seed/User status/Custom value).
4. **Power ON** the Air Bridge Slave device(s).
5. (*optional*) From a user application (in the Slave device(s) segment), send message **#1** (see Table 134 on Page 53) with a *user-specified* **User status** to each Slave device desired.
6. (*optional*) From a user application (in the Slave device(s) segment), send message **#4** (see Table 134 on Page 53) with a *user-specified* **Custom value** to each Slave device desired.

7. (optional) If user-defined, non-default **Pairing codes** will be utilised to lock out unauthorized devices from the pairing session, set **Master pairing code seed** and **Slave pairing code seed** for both the Master and Slave devices respectively, in each CAN-bus segment, by sending message **#2** and **#3** (see Table 134 on Page 53). **NOTE!** Use the same seeds on all devices intended to communicate with each other.
8. From a user application, send message **#6** (see Table 134 on Page 53) with parameter byte=0x01 (Activate) to the Master device. *This message causes the Master 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 Slave devices (within reach) that do not already have an established connection with another Master device, will detect the mode change and start reporting their respective RF Identifiers & User status to the Master device. When a Slave 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 Master device discovers Slave device(s) and reports them with message **#9** (see Table 134 on Page 53) to a user application with approximately 200 ms period. There will be one message #9 for each discovered Slave device, every reporting period.
11. (optional) If any reported Slave device's **Custom value** is needed (in addition to the reported **User status**) for deciding the *pairing selection*, let the user application request the **Custom value** using message **#12** (see Table 134 on Page 53) with concerned Slave device's reported **RF Identifier** as parameter.
12. From a user application, select the Slave device that shall be paired with the Master device, by sending message **#7** (see Table 134 on Page 53) with the Slave device's reported **RF Identifier** as parameter. **NOTE!** The user application may utilise the reported **User status** and **Custom value** as criteria to automatically select one of several Slave devices.
13. When the pairing is acknowledged, all devices involved will leave Discovery/Pairing mode and return to standard (CAN Traffic) mode. The newly paired Master and Slave devices will establish a connection and exchange any ongoing CAN traffic.
14. Slave device(s) not selected for pairing will continue to try and establish a connection to any previously paired Master device or become idle (if it was previously paired to the Master device which is now being paired to another Slave device).

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

Prerequisite: An Air Bridge Master device has been previously paired/associated with an Air Bridge Slave device (see Section A.2.2, Pairing procedure summary - Preferred pairing, on Page 53, Section A.2.3, Pairing procedure summary - Indecisive pairing, on Page 54).

1. Connect the Air Bridge Master device that shall un-pair/disassociate from a previously paired Air Bridge Slave device, to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Master device.
3. From a user application, command **un-pairing** by sending message **#5** (see Table 134 on Page 53) with parameter bytes (timeout=0, preferred device RF Id=00000) to the Master device. *This message causes the Master device to asynchronously un-pair from a currently paired Slave device.*

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

Prerequisite: An Air Bridge Master device has been previously paired/associated with an Air Bridge Slave device (see Section A.2.2, Pairing procedure summary - Preferred pairing, on Page 53, Section A.2.3, Pairing procedure summary - Indecisive pairing, on Page 54).

1. Connect the Air Bridge Master device that shall un-pair/disassociate from a previously paired Air Bridge Slave device, to a CAN-bus segment from where it can be accessed from a user application.
2. **Power ON** the Air Bridge Master device.
3. From a user application, send message **#6** (see Table 134 on Page 53) with parameter byte=0x01 (Activate) to the Master device. *This message causes the Master 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 Slave devices (within reach) that do not already have an established connection with another Master device, will detect the mode change and start reporting their respective RF Identifiers & User status to the Master device.*
5. From a user application, command **un-pairing** by sending message **#7** (see Table 134 on Page 53) with RF Id parameter set to 0x00000.
6. *The Master device un-pairs/disassociates the currently paired Slave device, leaves the Discovery/Pairing mode and returns to standard (CAN Traffic) mode. However, since the Master device is no longer paired with any Slave device, it will stay **radio-silent**.*
7. *Any Slave device that entered the Discovery/Pairing mode (step 4) will eventually **timeout** from the session and return to standard (CAN Traffic) mode.*
8. **NOTE!** *A previously paired Slave 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 Master device the next time the Slave device detects the very same Master device.*

A.2.6 Message examples

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	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 value** (value=0xB0BBAF00) to a connected Air Bridge Slave device:

Dir	DLC	SIL	RSID	DID ₀	DID ₁	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 **Master pairing code seed** (value=0xFFFFFFFFB) to a connected Air Bridge Master or Slave device:

Dir	DLC	SIL	RSID	DID ₀	DID ₁	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 **Slave pairing code seed** (value=0x0000A6A9) to a connected Air Bridge Master or Slave device:

Dir	DLC	SIL	RSID	DID ₀	DID ₁	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 Master device:

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x04	0x06	0xCA	0xD1	0x01	-	-	-
Rx	8	0x03	0x06	0xCA	0xD1	-	-	-	-

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	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 value of the reported Slave device (with RF Identifier=**11012**) from the Air Bridge Master device:

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x07	0x05	0xAD	0x00	0x00	0x01	0x10	0x12
Rx	8	0x07	0x05	0xAD	0x00	0xB0	0xBB	0xAF	0x00

Here, Custom value=B0BBAF00 is returned regarding device 11012.

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x07	0x06	0xCA	0xD2	0x00	0x01	0x10	0x12
Rx	8	0x03	0x06	0xCA	0xD2	-	-	-	-

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x07	0x06	0xCA	0xD0	0x00	0x00	0x00	0x00
Rx	8	0x03	0x06	0xCA	0xD0	-	-	-	-

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x04	0x06	0xCA	0xD1	0x00	-	-	-
Rx	8	0x03	0x06	0xCA	0xD1	-	-	-	-

k) User application enables Discovery/Pairing mode on an Air Bridge Master device and commands **pairing** with a **preferred** Air Bridge Slave device (RF Identifier=**11012**) within 15 seconds (timeout=**0F**).

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x07	0x06	0xCA	0xD0	0x0F	0x01	0x10	0x12
Rx	8	0x03	0x06	0xCA	0xD0	-	-	-	-

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

Dir	DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x07	0x06	0xCA	0xD0	0x00	0x00	0x00	0x00
Rx	8	0x03	0x06	0xCA	0xD0	-	-	-	-

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 ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
Tx	8	0x03	0x05	0xA0	0x00	-	-	-	-
Rx	8	0x04	0x05	0xA0	0x00	0x00	-	-	-

A.3 NAK example

DLC	SIL	RSID	DID ₀	DID ₁	D ₀	D ₁	D ₂	D ₃
8	0x07	0x06	0xCA	0xD2	0x00	0x08	0x01	0x03

Table 135: 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	0x03	0x0F	0x06	0x72	0x55	0x55	0x55	0x55

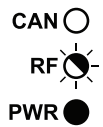
Table 136: NAK with NRC=0x72 for SID=0x6.

B Appendix B: LED UI examples

B.1 LED behaviour in Discovery/Pairing mode

When activating the **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.
 - **RF** (blue) will flash ON and OFF *approx.* every half second. On the Master device, this is a visible acknowledgment that a new Discovery/Pairing session has been initiated. On the Slave device, it is a visible acknowledgment that the Slave device has responded to the mode transition and has also been accepted into the initiated session.
 - **PWR** (green) will be constantly illuminated indicating the device *is* powered up.
- NOTE!** Should the periodic RF blinking pattern become partially irregular, this indicates that the respective Slave 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).



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.