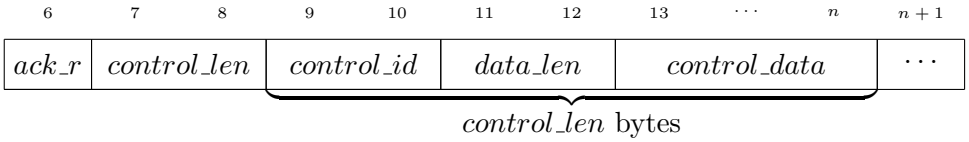
Protocol

Control-Header

The control header is prepended to each frame. It can be used to store informations like the name of the frame or its length. In addition it can be used to acknowledge that a frame has been received successfully or it can be used to request missing fragments. Table 1 describes the structure of the control header and table 2 lists the available control IDs.



ack_r	Defines whether a frame has to be acknowledged: 0 → No ACK required. 1 → The sender expects an ACK (see table 2 *ACKED_FRAME*). 2 → The sender expects a list of missing fragments (see table 2 *MISSING_FRAGMENTS*).
control_len	Number of bytes of all control informations.
control_id	Control information ID (see table 2).
data_len	Length of the following control data.
control_data	Control data encoded as a character string.

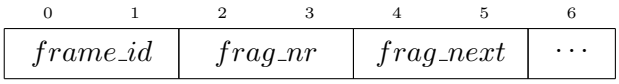
Tabelle 1: Control-Header Elements

Name	**ID**	**Description**
PORT_NAME	1	Contains the name of the frame. Within a Rock environment the name of the according port is used to identify the frame.
ACKED_FRAME	2	Allows to acknowledge the receiving of a frame. In this case the control_data contains the ID of the received frame.
PAKET_LEN	3	Stores the length of the frame data (encoded as a character string) following the control header.
MISSING_FRAGMENTS	4	In this case the control data contains the ID of the received frame followed by the IDs of the missing fragments. If no fragments are missing the frame has been received successfully.

Tabelle 2: Control Informations

Fragment-Header

SequencedComm splits the data (frame) into single fragments acoording to the configuration parameter *max_fragment_size* which is required if the size of the data exceeds the maximum size of an UDP/TCP datagram package (65507 byte). The fragment header is prepended to each single fragment.[1]



$frame\_id$	Identifies a complete frame. All fragments of a frame are assigned the same frame ID (starting with 1).
$frag\_nr$	Identifies a single fragment (starting with 0). For each additional fragment the fragment number is increased by one.
$frag\_next$	Identifies the next fragment ($frag\_nr$ + 1). If this value is 0 the last fragment of the frame has been found.

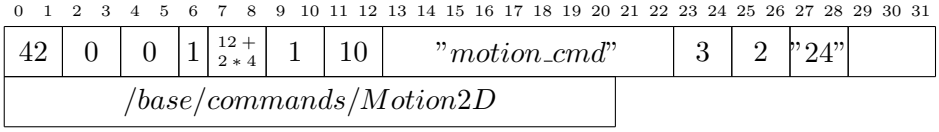
Tabelle 3: Fragment-Header

[1] There is always one control header, the frame data is optional and the number of fragment headers depends on the overall frame size.

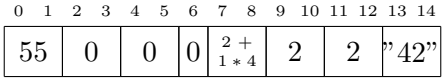
Examples

Sending an ACK frame

In the following example frame (ID 42) is sent which is not fragmented and which should be acknowledged by the receiver. Therefore the ACK byte *ack_r* is set to 1 / true. The length of both control informations is 20 Byte:
2 (control ID) + 2 (control length) + 10 (control content) + 2 + 2 + 2 = 12 + 2 * 4 = 20 Byte.

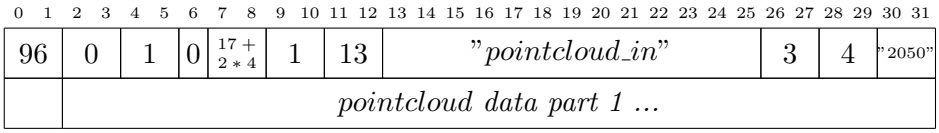


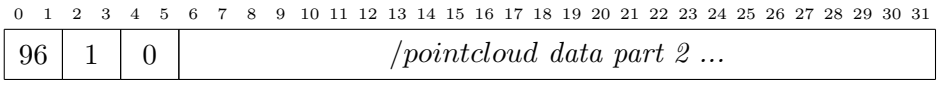
The receiver replies with an ACK frame which just contains the control header and no additional data. The ID of the frame is 55, it is not fragmented, the control length is 6 Byte and it acknowledges the receiving of frame 42.



Sending of a fragmented frame

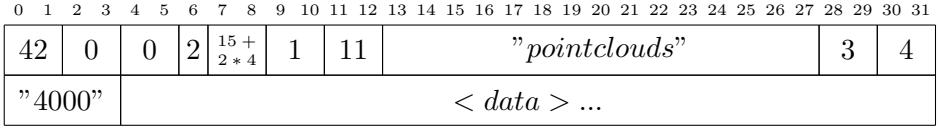
In this example a poincloud is sent which size exceeds the maximum frame size defined by the configuration parameter *max_fragment_size*. Therefore the frame is split into two fragments. The frame ID for both fragments is 96, the fragment-ID and the next-fragment-ID for the first fragment are 0 and 1 and for the second fragment 1 and 0 (0 identifies the last fragment of the frame). The fragment header is prepended to both fragments.





Request missing fragments

In this example *ack_r* is set to 2 which tells the receiver that it can request missing fragments. The data is split into three fragments which are stored by the sender.



We assume that the middle fragment has been lost during sending so the receiver requests for frame

42 fragment 1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
55		0		0		0	$4 + 1 * 4$		4		4		"42 1"			

After the missing fragment has been resent and received successfully the receiver tells the sender that all fragments have been received by sending another *MISSING_FRAGMENTS* frame just containing the ID of the received frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
58		0		0		0	$2 + 1 * 4$		4		2		"42"	