

# AXI4-Stream BFM – Quick Reference

**NOTE: As of UVVM v3.x, all shared variables have been made protected. This means that any access to shared variables must be done using get- and set-methods. This documentation has not yet been updated with the methods for accessing these variables, but will be very soon.**

AXI4-Stream Master (see page 2 for AXI4-Stream Slave)

**axistream\_transmit (data\_array, [user\_array, [strb\_array, id\_array, dest\_array]], msg, clk, axistream\_if, [scope, [msg\_id\_panel, [config]]])**

**Example (tdata'length = 16) :** axistream\_transmit ( (x"D0", x"D1", x"D2", x"D3"), (x"00", x"0A"), "Send a 4 byte packet with tuser=A at the 2<sup>nd</sup> (last) word", clk, axistream\_if);  
**Example (tdata'length = 8) :** axistream\_transmit ( (x"D0", x"D1", x"D2", x"D3"), (x"00", x"00", x"00", x"0A"), "Send a 4 byte packet with tuser=A at the 4<sup>th</sup> (last) word", clk, axistream\_if);  
**Example:** axistream\_transmit(v\_data\_array(0 to v\_numBytes-1), "Send v\_numBytes bytes", clk, axistream\_if\_m, C\_SCOPE, shared\_msg\_id\_panel, axistream\_bfm\_config);  
**Example:** axistream\_transmit(v\_data\_array(0 to v\_numBytes-1)(15 downto 0), "Send 2 x v\_numBytes bytes", clk, axistream\_if\_m, C\_SCOPE, shared\_msg\_id\_panel, axistream\_bfm\_config);  
**Example:** axistream\_transmit(v\_data\_array(0 to v\_numBytes-1), v\_user\_array(0 to v\_numWords-1), "Send v\_numBytes bytes", clk, axistream\_if\_m, C\_SCOPE, shared\_msg\_id\_panel, axistream\_bfm\_config);  
**Example:** axistream\_transmit(v\_data\_array(0 to v\_numBytes-1), v\_user\_array(0 to v\_numWords-1), v\_strb\_array(0 to v\_numWords-1), v\_id\_array(0 to v\_numWords-1), v\_id\_array(0 to v\_numWords-1), "Send" ....

**init\_axistream\_if\_signals (is\_master, data\_width, user\_width, id\_width, dest\_width)**

**Example:** axistream\_if <= init\_axistream\_if\_signals(true, axistream\_if.tdata'length, axistream\_if.tuser'length, axistream\_if.tid'length, axistream\_if.tdest'length);

**BFM**



*axistream\_bfm\_pkg.vhd*



**UVVM™**

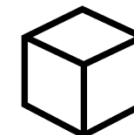
# AXI4-Stream BFM – Quick Reference

AXI4-Stream Slave (see page 1 for AXI4-Stream Master)

**axistream\_receive** (data\_array, data\_length, user\_array, strb\_array, id\_array, dest\_array, msg, clk, axistream\_if, [scope, [msg\_id\_panel, [config, [proc\_name]]]])

**Example:** axistream\_receive(v\_rx\_data\_array, v\_rx\_length, v\_rx\_user\_array, v\_rx\_strb\_array, v\_rx\_id\_array, v\_rx\_dest\_array, "Receive packet", clk, axistream\_if);

**BFM**



axistream\_bfm\_pkg.vhd

**axistream\_expect** (exp\_data\_array, [exp\_user\_array, [exp\_strb\_array, exp\_id\_array, exp\_dest\_array]], msg, clk, axistream\_if, [alert\_level, [scope, [msg\_id\_panel, [config]]]])

**Example (tdata'length = 16) :** axistream\_expect( (x"D0", x"D1", x"D2", x"D3"), (x"00", x"0A"), "Expect a 4 byte packet with tuser=A at the 2<sup>nd</sup> (last) word", clk, axistream\_if);

**Example (tdata'length = 8) :** axistream\_expect( (x"D0", x"D1", x"D2", x"D3"), (x"00", x"00", x"00", x"0A"), "Expect a 4 byte packet with tuser=A at the 4<sup>th</sup> (last) word", clk, axistream\_if);

**Example:** axistream\_expect(v\_data\_array(0 to 1), "Expect a 2 byte packet, ignoring the tuser bits", clk, axistream\_if);

**Example:** axistream\_expect(v\_data\_array(0 to v\_numBytes-1), v\_user\_array(0 to v\_numWords-1), "Expect a packet, check data and tuser, but ignore tstrb, tid, tdest", clk, axistream\_if);

**Example:** axistream\_expect(v\_data\_array(0 to v\_numBytes-1), v\_user\_array(0 to v\_numWords-1), v\_strb\_array(0 to v\_numWords-1), v\_id\_array(0 to v\_numWords-1), v\_id\_array(0 to v\_numWords-1), "Expect packet", clk, axistream\_if);

**init\_axistream\_if\_signals** (is\_master, data\_width, user\_width, id\_width, dest\_width)

**Example:** axistream\_if <= init\_axistream\_if\_signals(false, axistream\_if.tdata'length, axistream\_if.tuser'length, axistream\_if.tid'length, axistream\_if.tdest'length);

#### BFM Configuration record 't\_axistream\_bfm\_config'

| Record element                 | Type               | C_AXISTREAM_BFM_CONFIG_DEFAULT |
|--------------------------------|--------------------|--------------------------------|
| max_wait_cycles                | integer            | 100                            |
| max_wait_cycles_severity       | t_alert_level      | ERROR                          |
| clock_period                   | time               | -1 ns                          |
| clock_period_margin            | time               | 0 ns                           |
| clock_margin_severity          | t_alert_level      | TB_ERROR                       |
| setup_time                     | time               | -1 ns                          |
| hold_time                      | time               | -1 ns                          |
| bfm_sync                       | t_bfm_sync         | SYNC_ON_CLOCK_ONLY             |
| match_strictness               | t_match_strictness | MATCH_EXACT                    |
| byte_endianness                | t_byte_endianness  | FIRST_BYTE_LEFT                |
| valid_low_at_word_num          | integer            | 0                              |
| valid_low_multiple_random_prob | real               | 0.5                            |
| valid_low_duration             | integer            | 0                              |
| valid_low_max_random_duration  | integer            | 5                              |
| check_packet_length            | boolean            | false                          |
| protocol_error_severity        | t_alert_level      | ERROR                          |
| ready_low_at_word_num          | integer            | 0                              |
| ready_low_multiple_random_prob | real               | 0.5                            |
| ready_low_duration             | integer            | 0                              |
| ready_low_max_random_duration  | integer            | 5                              |
| ready_default_value            | std_logic          | '0'                            |
| id_for_bfm                     | t_msg_id           | ID_BFM                         |

| Record element           | Type             |
|--------------------------|------------------|
| tdata                    | std_logic_vector |
| tkeep                    | std_logic_vector |
| tuser, tstrb, tid, tdest | std_logic_vector |
| tvalid                   | std_logic        |
| tlast                    | std_logic        |
| tready                   | std_logic        |

#### BFM signal parameters

| Name         | Type           | Description   |
|--------------|----------------|---|
| clk          | std_logic      | The clock signal used to read and write data in/out of the AXI4-Stream BFM.   |
| axistream_if | t_axistream_if | See table "Signal record 't_axistream_if'" in page 1 and 2.<br>Note: All supported signals, including tuser, tstrb, tid, tdest are included in the record type, even when not used or connected to DUT. |

For more information on the AXI4-Stream signals, refer to "AMBA® 4 AXI4-Stream Protocol Specification", document number ARM IHI 0051A (ID030510), available from ARM

## BFM non-signal parameters

| Name           | Type                               | Example(s)                                      | Description   |
|----------------|------------------------------------|---|---|
| data_array     | t_slv_array or<br>std_logic_vector | (x"D0", x"D1")<br>(x"D0D1", x"D2D3")<br>x"D0D1" | An array of SLVs or a single SLV containing the packet data to be sent.<br><br>data_array(0) is sent/received first, while data_array(data_array'high) is sent/received last. Note that the 8 upper bits in the data word is sent/received first, and the 8 lower bits are is sent/received last.<br>For clarity, data_array is required to be ascending, for example defined by the test sequencer as follows:<br>variable v_slv_data_array : t_slv_array(0 to C_MAX_WORDS-1)(C_WORD_LENGTH-1 downto 0);                           |
| exp_data_array | t_slv_array or<br>std_logic_vector | (x"D0", x"D1")<br>(x"D0D1", x"D2D3")<br>x"D0D1" | An array of SLVs or a single SLV containing the packet of data that is expected to be received.<br>The data_array specifications listed above applies for exp_data_array as well.   |
| user_array     | t_user_array                       | x"01" & x"02"                                   | Sideband data to send or has been received via the TUSER signal.<br>The number of entries in user_array equals the number of data words, i.e. transfers <sup>1</sup> . For example, if 16 bytes shall be sent, and there are 8 bytes transmitted per transfer, the user_array has 2 entries.<br>The number of bits actually used in each user_array entry corresponds to the width of axistream_if.tuser.<br>Note: If axistream_if.TUSER is wider than 8, increase the value of the constant C_MAX_TUSER_BITS in axistream_bfm_pkg. |
| strb_array     | t_strb_array                       | "00" & "10"                                     | Sideband data to send or has been received via the TSTRB signal. The BFM transmits/receives the values without affecting TDATA.<br>The number of entries in this array equals the number of data words, i.e. transfers <sup>1</sup> .<br>The number of bits actually used in each array entry corresponds to the width of axistream_if.TSTRB.<br>Note: If axistream_if.TSTRB is wider than 32, increase the value of the constant C_MAX_TSTRB_BITS in axistream_bfm_pkg.  |
| id_array       | t_id_array                         | x"01" & x"02"                                   | Sideband data to send or has been received via the TID signal.<br>The number of entries in this array equals the number of data words, i.e. transfers <sup>1</sup> .<br>The number of bits actually used in each array entry corresponds to the width of axistream_if.TID.<br>Note: If axistream_if.TID is wider than 8, increase the value of the constant C_MAX_TID_BITS in axistream_bfm_pkg.  |
| dest_array     | t_dest_array                       | x"1" & x"2"                                     | Sideband data to send or has been received via the TDEST signal.<br>The number of entries in this array equals the number of data words, i.e. transfers <sup>1</sup> .<br>The number of bits actually used in each array entry corresponds to the width of axistream_if.TDEST.<br>Note: If axistream_if.TDEST is wider than 4, increase the value of the constant C_MAX_TDEST_BITS in axistream_bfm_pkg.  |
| data_length    | natural                            | 2   | The number of bytes received, i.e. the number of valid bytes in data_array.   |
| alert_level    | t_alert_level                      | ERROR or TB_WARNING                             | Set the severity for the alert that may be asserted by the procedure.   |
| msg            | string                             | "Send packet"                                   | A custom message to be appended in the log/alert.   |
| scope          | string                             | "AXISTREAM BFM"                                 | A string describing the scope from which the log/alert originates.<br>In a simple single sequencer typically "AXISTREAM BFM". In a verification component typically "AXISTREAM_VVC".  |
| msg_id_panel   | t_msg_id_panel                     | shared_msg_id_panel                             | Optional msg_id_panel, controlling verbosity within a specified scope. Defaults to a common message ID panel defined in the UVVM-Util adaptations package.  |
| config         | t_axistream_bfm_config             | C_AXISTREAM_BFM_CONFIG_DEFAULT                  | Configuration of BFM behaviour and restrictions. See section 2 for details.   |

<sup>1</sup> In AXI4-Stream, a transfer is defined as a TVALID/TREADY handshake.

## BFM features

This BFM supports the following subset of the AXI4-Stream protocol:

- Continuous aligned stream, as described in chapter 1.2.2 in AMBA 4 AXI4-Stream protocol Specification (ARM IHI 0051A)

The following signals are supported:

| Signal  | Source | Width              | Supported by BFM | Description  |
|---------|--------|--------------------|------------------|--|
| ACLK    | Clock  | 1                  | Yes              | Sample on the rising edge  |
| ARESETn | Reset  | -                  | No               | BFM doesn't control the reset.   |
| TVALID  | Master | 1                  | Yes              | A transfer takes place when both TVALID and TREADY are asserted  |
| TREADY  | Slave  | 1                  | Yes <sup>2</sup> | A transfer takes place when both TVALID and TREADY are asserted  |
| TDATA   | Master | n*8                | Yes              | Data word. The width must be a multiple of bytes.  |
| TUSER   | Master | 1:c_max_tuser_bits | Yes <sup>2</sup> | Sideband info transmitted alongside the data stream.<br>If <code>axistream_if.tuser</code> is wider than <code>c_max_tuser_bits</code> in <code>axistream_bfm_pkg</code> , increase the value of the latter.   |
| TSTRB   | Master | 1:c_max_tstrb_bits | Yes <sup>2</sup> | The protocol uses this signal for marking TDATA as position byte, but the BFM simply sends/receives/checks the values of TSTRB as specified by the sequencer without affecting TDATA:<br>While transmitting, the test sequencer defines what TSTRB values to send. The BFM transmits TDATA regardless of the TSTRB value.<br>While receiving, the received TSTRB values are presented to the test sequencer. The BFM presents TDATA regardless of the TSTRB value.<br>If <code>axistream_if.tstrb</code> is wider than <code>c_max_tstrb_bits</code> in <code>axistream_bfm_pkg</code> , increase the value of the latter. |
| TKEEP   | Master | TDATA'length/8     | Partly           | When TKEEP is '0', it indicates a null byte that can be removed from the stream.<br>The same limitations apply for this BFM as in the <i>Xilinx ug761 AXI Reference Guide</i> :<br>Null bytes are only used for signalling the number of valid bytes in the last data word. Leading or intermediate Null bytes are not supported.  |
| TLAST   | Master | 1                  | Yes              | When '1', it indicates that the tdata is the last word of the packet.  |
| TID     | Master | 1:c_max_tid_bits   | Yes <sup>2</sup> | Indicates different streams of data. Usually used by routing infrastructures.<br>When BFM is transmitting, the test sequencer defines what TID values to send.<br>When BFM is receiving, the received TID values are presented to the test sequencer.<br>If <code>axistream_if.tid</code> is wider than <code>c_max_tid_bits</code> in <code>axistream_bfm_pkg</code> , increase the value of the latter   |
| TDEST   | Master | 1:c_max_tdest_bits | Yes <sup>2</sup> | Provides routing info for the data stream. Usually used by routing infrastructures.<br>When BFM is transmitting, the test sequencer defines what TDEST values to send.<br>When BFM is receiving, the received TDEST values are presented to the test sequencer.<br>If <code>axistream_if.tdest</code> is wider than <code>c_max_tdest_bits</code> in <code>axistream_bfm_pkg</code> , increase the value of the latter   |

<sup>2</sup> Although defined as optional in the AXI4-Stream protocol, the signal must exist in the `axistream_if` record, even when not used / connected to DUT.

# BFM details

## 1 BFM procedure details

| Procedure                          | Description   |
|------------------------------------|---|
| <b>axistream_transmit()</b>        | <p><b>axistream_transmit (data_array, [user_array, [strb_array, id_array, dest_array]], msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])</b></p> <p>The axistream_transmit () procedure transmits a packet on the AXI interface.<br/> The packet length and data are defined by the "data_array" argument, and is either a t_slv_array or a SLV.<br/> data_array(0) is sent first. data_array(data_array'high) is sent last. The upper 8 bits are sent first and the lower 8 bits are sent last.<br/> Byte locations within the data word are defined in chapter 2.3 in "AMBA® 4 AXI4-Stream Protocol Specification", document number ARM IHI 0051A (ID030510), available from ARM.</p> <p>The values to be transmitted on the signal TUSER is defined by the optional user_array parameter. There is one user_array index per transfer (data word).<br/> If user_array is omitted in the BFM call, the BFM transmits all zeros on the TUSER signal.</p> <p>The values to be transmitted on the signals TSTRB, TID, TDEST are defined by the parameters strb_array, id_array and dest_array.<br/> There is one array index per transfer (data word).<br/> All or none of these three arrays may be omitted in the BFM call. If they are omitted, the BFM transmits all zeros on the TSTRB, TID, TDEST signals.</p> <p>At the last word, the BFM asserts the TLAST bit, and it asserts the TKEEP bits corresponding to the data bytes that are valid within the word.<br/> At all other words, all TKEEP bits are '1', thus the BFM supports only "continuous aligned stream", as described in chapter 1.2.2 in AMBA 4 AXI4-Stream protocol Specification (ARM IHI 0051A).</p> |
| <b>axistream_receive()</b>         | <p><b>axistream_receive (data_array, data_length, user_array, strb_array, id_array, dest_array, msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])</b></p> <p>The axistream_receive() procedure receives a packet on the AXI interface.<br/> The received packet data is stored in the data_array output, which is a t_slv_array(0 to C_VVC_CMD_DATA_MAX_BYTES - 1)(7 downto 0). data_array'length can be longer than the actual packet received, so that you can call receive() without knowing the length to be expected. The number of bytes received is indicated in the packet_length output.</p> <p>The sampled values of the TUSER signal are stored in user_array, which has one entry per transfer (data word).<br/> The sampled values of the TSTRB signal are stored in strb_array, which has one entry per transfer (data word).<br/> The sampled values of the TID signal are stored in id_array, which has one entry per transfer (data word).<br/> The sampled values of the TDEST signal are stored in dest_array, which has one entry per transfer (data word).</p> <p>When TLAST = '1' the TKEEP bits are used to determine the number of valid data bytes within the last word.<br/> At all other words, the BFM checks that all TKEEP bits are '1', since the BFM supports only "continuous aligned stream" described in chapter 1.2.2 in AMBA 4 AXI4-Stream protocol Specification (ARM IHI 0051A)</p>   |
| <b>axistream_expect()</b>          | <p><b>axistream_expect (exp_data_array, [exp_user_array, [exp_strb_array, exp_id_array, exp_dest_array]], msg, clk, axistream_if, [alert_level, [scope, [msg_id_panel, [config]]]])</b></p> <p>Calls the axistream_receive() procedure, then compares the received data with exp_data_array.<br/> The exp_user_array, exp_strb_array, exp_id_array, exp_dest_array are compared to the received user_array, strb_array, id_array and dest_array respectively.<br/> If some signals are unused, the checks can be skipped by filling the corresponding exp_*_array with don't cares. For example: v_dest_array := (others =&gt; (others =&gt; '-'));</p>   |
| <b>init_axistream_if_signals()</b> | <p><b>init_axistream_if_signals(is_master, data_width, user_width, id_width, dest_width)</b></p> <p>This function initializes the AXI4-Stream interface. All the BFM outputs are set to zeros ('0')</p>   |

## 2 BFM Configuration record

Type name: t\_axistream\_bfm\_config

| Record element                 | Type               | C_AXISTREAM_BFM_CONFIG_DEFAULT | Description   |
|--------------------------------|--------------------|--------------------------------|---|
| max_wait_cycles                | integer            | 100                            | Used for setting the maximum cycles to wait before an alert is issued when waiting for ready or valid signals from the DUT.   |
| max_wait_cycles_severity       | t_alert_level      | ERROR                          | The above timeout will have this severity   |
| clock_period                   | time               | -1 ns                          | Period of the clock signal.   |
| clock_period_margin            | time               | 0 ns                           | Input clock period margin to specified clock_period   |
| clock_margin_severity          | t_alert_level      | TB_ERROR                       | The above margin will have this severity  |
| setup_time                     | time               | -1 ns                          | Setup time for generated signals. Suggested value is clock_period/4.<br>An alert is reported if setup_time exceed clock_period/2.   |
| hold_time                      | time               | -1 ns                          | Hold time for generated signals. Suggested value is clock_period/4.<br>An alert is reported if hold_time exceed clock_period/2.   |
| bfm_sync                       | t_bfm_sync         | SYNC_ON_CLOCK_ONLY             | When set to SYNC_ON_CLOCK_ONLY the BFM will enter on the first falling edge, estimate the clock period, synchronise the output signals and exit ¼ clock period after a succeeding rising edge.<br>When set to SYNC_WITH_SETUP_AND_HOLD the BFM will use the configured setup_time, hold_time and clock_period to synchronise output signals with clock edges. |
| match_strictness               | t_match_strictness | MATCH_EXACT                    | Matching strictness for std_logic values in check procedures.<br>MATCH_EXACT requires both values to be the same. Note that the expected value can contain the don't care operator '-'.<br>MATCH_STD allows comparisons between 'H' and '1', 'L' and '0' and '-' in both values.  |
| byte_endianness                | t_byte_endianness  | FIRST_BYTE_LEFT                | Little-endian or big-endian endianness byte ordering.   |
| valid_low_at_word_num          | integer            | 0                              | Word index during which the Master BFM shall deassert valid while sending a packet. Can be set to multiple random indices using C_MULTIPLE_RANDOM.  |
| valid_low_multiple_random_prob | real               | 0.5                            | Probability, between 0.0 and 1.0, of how often valid shall be deasserted when using C_MULTIPLE_RANDOM.  |
| valid_low_duration             | integer            | 0                              | Number of clock cycles to deassert valid. To disable this feature set to 0. Can be set to random using C_RANDOM.  |
| valid_low_max_random_duration  | integer            | 5                              | Maximum number of clock cycles to deassert valid when using C_RANDOM.   |
| check_packet_length            | boolean            | false                          | When true, receive() will check that tlast is set at data_array/high.<br>Set to false when length of packet to be received is unknown.  |
| protocol_error_severity        | t_alert_level      | ERROR                          | severity if protocol errors are detected  |
| ready_low_at_word_num          | integer            | 0                              | Word index during which the Slave BFM shall deassert ready while receiving the packet. Can be set to multiple random indices using C_MULTIPLE_RANDOM.   |
| ready_low_multiple_random_prob | real               | 0.5                            | Probability, between 0.0 and 1.0, of how often ready shall be deasserted when using C_MULTIPLE_RANDOM.  |

|                               |           |        |  |
|-------------------------------|-----------|--------|--|
| ready_low_duration            | integer   | 0      | Number of clock cycles to deassert ready. To disable this feature set to 0. Can be set to random using C_RANDOM. |
| ready_low_max_random_duration | integer   | 5      | Maximum number of clock cycles to deassert ready when using C_RANDOM.  |
| ready_default_value           | std_logic | '0'    | Determines the ready output value while the Slave BFM is idle  |
| id_for_bfm                    | t_msg_id  | ID_BFM | The message ID used as a general message ID in the BFM   |

### 3 Additional Documentation

For additional documentation on the AXI4-Stream standard, refer to “AMBA® 4 AXI4-Stream Protocol Specification”, document number ARM IHI 0051A (ID030510), available from ARM.

### 4 Compilation

The AXI4-Stream BFM may only be compiled with VHDL 2008. It is dependent on the UVVM Utility Library (UVVM-Util), which is only compatible with VHDL 2008. See the separate UVVM-Util documentation for more info. After UVVM-Util has been compiled, the axistream\_bfm\_pkg.vhd BFM can be compiled into any desired library. See UVVM Essential Mechanisms located in uvvm\_vvc\_framework/doc for information about compile scripts.

#### 4.1 Simulator compatibility and setup

See README.md for a list of supported simulators.

For required simulator setup see UVVM-Util Quick reference.



## 5 Local BFM overloads

A good approach for better readability and maintainability is to make simple, local overloads for the BFM procedures in the TB process.

This allows calling the BFM procedures with the key parameters only

e.g.

```
axistream_transmit(v_data_array(0 to 1), "msg");
```

rather than

```
axistream_transmit(v_data_array(0 to 1), "msg", clk, axistream_if_m, C_SCOPE, shared_msg_id_panel, axistream_bfm_config);
```

By defining the local overload as e.g.:

```
procedure axistream_transmit (  
    constant data_array : in t_slv_array;  
    constant msg        : in string) is  
begin  
    axistream_transmit(data_array,  
                        msg,  
                        clk,  
                        axistream_if,  
                        C_SCOPE,  
                        shared_msg_id_panel,  
                        C_AXISTREAM_BFM_CONFIG_LOCAL);  
end;
```

-- keep as is  
-- keep as is  
-- Clock signal  
-- Signal must be visible in local process scope  
-- Just use the default  
-- Use global, shared msg\_id\_panel  
-- Use locally defined configuration or C\_AXISTREAM\_BFM\_CONFIG\_DEFAULT

Using a local overload like this also allows the following – if wanted:

- Set up defaults for constants. May be different for two overloads of the same BFM
- Apply dedicated message\_id\_panel to allow dedicated verbosity control

### IMPORTANT

This is a simplified Bus Functional Model (BFM) for AXI4-Stream. The given BFM complies with the basic AXI4-Stream protocol and thus allows a normal access towards an AXI4-Stream interface. This BFM is not AXI4-Stream protocol checker. For a more advanced BFM please contact Bitvis AS at [support@bitvis.no](mailto:support@bitvis.no)

### INTELLECTUAL PROPERTY

Disclaimer: This IP and any part thereof are provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with this IP.