## NAME

**ibv_read_counters** – Read counter values

## SYNOPSIS

```
#include <infiniband/verbs.h>

int ibv_read_counters(struct ibv_counters *counters,
                      uint64_t *counters_value,
                      uint32_t ncounters,
                      uint32_t flags);
```

## DESCRIPTION

**ibv_read_counters**() returns the values of the chosen counters into *counters_value* array of which can accumulate *ncounters*. The values are filled according to the configuration defined by the user in the **ibv_attach_counters_point_xxx** functions.

## ARGUMENTS

*counters*
: Counters object to read.

*counters_value*
: Input buffer to hold read result.

*ncounters*
: Number of counters to fill.

*flags* Use enum ibv_read_counters_flags.

*flags* **Argument**

**IBV_READ_COUNTERS_ATTR_PREFER_CACHED**
: Will prefer reading the values from driver cache, else it will do volatile hardware access which is the default.

## RETURN VALUE

**ibv_read_counters**() returns 0 on success, or the value of errno on failure (which indicates the failure reason)

## EXAMPLE

Example: Statically attach counters to a new flow

This example demonstrates the use of counters which are attached statically with the creation of a new flow. The counters are read from hardware periodically, and finally all resources are released.

```
/* create counters object and define its counters points        */
/* create simple L2 flow with hardcoded MAC, and a count action */
/* read counters periodically, every 1sec, until loop ends      */
/* assumes user prepared a RAW_PACKET QP as input               */
/* only limited error checking in run time for code simplicity  */

#include <inttypes.h>
#include <infiniband/verbs.h>

/* the below MAC should be replaced by user */
#define FLOW_SPEC_ETH_MAC_VAL {
    .dst_mac = { 0x00, 0x01, 0x02, 0x03, 0x04,0x05},
    .src_mac = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
    .ether_type = 0, .vlan_tag = 0, }
#define FLOW_SPEC_ETH_MAC_MASK {
    .dst_mac = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
    .src_mac = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
    .ether_type = 0, .vlan_tag = 0, }
```

```
void example_create_flow_with_counters_on_raw_qp(struct ibv_qp *qp) {
    int idx = 0;
    int loop = 10;
    struct ibv_flow *flow = NULL;
    struct ibv_counters *counters = NULL;
    struct ibv_counters_init_attr init_attr = {0};
    struct ibv_counter_attach_attr attach_attr = {0};

    /* create single coutners handle */
    counters = ibv_create_counters(qp->context, &init_attr);

    /* define counters points */
    attach_attr.counter_desc = IBV_COUNTER_PACKETS;
    attach_attr.index = idx++;
    ret = ibv_attach_counters_point_flow(counters, &attach_attr, NULL);
    if (ret == ENOTSUP) {
        fprintf(stderr, "Attaching IBV_COUNTER_PACKETS to flow is not \
supported");
        exit(1);
    }
    attach_attr.counter_desc = IBV_COUNTER_BYTES;
    attach_attr.index = idx++;
    ibv_attach_counters_point_flow(counters, &attach_attr, NULL);
    if (ret == ENOTSUP) {
        fprintf(stderr, "Attaching IBV_COUNTER_BYTES to flow is not \
supported");
        exit(1);
    }

    /* define a new flow attr that includes the counters handle */
    struct raw_eth_flow_attr {
        struct ibv_flow_attr              attr;
        struct ibv_flow_spec_eth          spec_eth;
        struct ibv_flow_spec_counter_action spec_count;
    } flow_attr = {
        .attr = {
                .comp_mask  = 0,
                .type       = IBV_FLOW_ATTR_NORMAL,
                .size       = sizeof(flow_attr),
                .priority   = 0,
                .num_of_specs = 2, /* ETH + COUNT */
                .port       = 1,
                .flags      = 0,
            },
        .spec_eth = {
                .type = IBV_EXP_FLOW_SPEC_ETH,
                .size = sizeof(struct ibv_flow_spec_eth),
                .val  = FLOW_SPEC_ETH_MAC_VAL,
                .mask = FLOW_SPEC_ETH_MAC_MASK,
            },
        .spec_count = {
                .type   = IBV_FLOW_SPEC_ACTION_COUNT,
                .size   = sizeof(struct ibv_flow_spec_counter_action),
                .counters = counters, /* attached this counters handle
```

```
        to the newly created ibv_flow */ } };

            /* create the flow */
            flow = ibv_create_flow(qp, &flow_attr.attr);

            /* allocate array for counters value reading */
            uint64_t *counters_value = malloc(sizeof(uint64_t) * idx);

            /* periodical read and print of flow counters */
            while (--loop) {
                sleep(1);

                /* read hardware counters values */
                ibv_read_counters(counters, counters_value, idx,
                          IBV_READ_COUNTERS_ATTR_PREFER_CACHED);

                printf("PACKETS = %"PRIu64", BYTES = %"PRIu64 \n",
                    counters_value[0], counters_value[1] );
            }

            /* all done, release all */
            free(counters_value);

            /* destroy flow and detach counters */
            ibv_destroy_flow(flow);

            /* destroy counters handle */
            ibv_destroy_counters(counters);

            return;
        }
```

## SEE ALSO

**ibv_create_counters**, **ibv_destroy_counters**, **ibv_attach_counters_point_flow**, **ibv_create_flow**

## AUTHORS

Raed Salem <raeds@mellanox.com>

Alex Rosenbaum <alexr@mellanox.com>