NAME

ibv_query_device_ex - query an RDMA device's attributes

SYNOPSIS

#include <infiniband/verbs.h>

int ibv_query_device_ex(struct ibv_context **context*,
                struct ibv_device_attr_ex **attr***);**

DESCRIPTION

ibv_query_device_ex() returns the attributes of the device with context *context*. The argument *attr* is a pointer to an ibv_device_attr_ex struct, as defined in <infiniband/verbs.h>.

```
struct ibv_device_attr_ex {
        struct ibv_device_attr orig_attr;
        uint32_t        comp_mask;         /* Compatibility mask that defines which of the following variables
        struct ibv_odp_caps   odp_caps;          /* On-Demand Paging capabilities */
        uint64_t        completion_timestamp_mask;  /* Completion timestamp mask (0 = unsupported) */
        uint64_t        hca_core_clock;        /* The frequency (in kHZ) of the HCA (0 = unsupported) */
        uint64_t        device_cap_flags_ex;    /* Extended device capability flags */
        struct ibv_tso_caps   tso_caps;          /* TCP segmentation offload capabilities */
        struct ibv_rss_caps   rss_caps;          /* RSS capabilities */
        uint32_t        max_wq_type_rq;        /* Max Work Queue from type RQ */
        struct ibv_packet_pacing_caps packet_pacing_caps; /* Packet pacing capabilities */
        uint32_t        raw_packet_caps;       /* Raw packet capabilities, use enum ibv_raw_packet_caps */
};

struct ibv_odp_caps {
    uint64_t general_odp_caps;    /* Mask with enum ibv_odp_general_cap_bits */
    struct {
        uint32_t rc_odp_caps; /* Mask with enum ibv_odp_tranport_cap_bits to know which operations are supporte
        uint32_t uc_odp_caps; /* Mask with enum ibv_odp_tranport_cap_bits to know which operations are supporte
        uint32_t ud_odp_caps; /* Mask with enum ibv_odp_tranport_cap_bits to know which operations are supporte
    } per_transport_caps;
};

enum ibv_odp_general_cap_bits {
    IBV_ODP_SUPPORT = 1 << 0, /* On demand paging is supported */
};
```

```
enum ibv_odp_transport_cap_bits {
        IBV_ODP_SUPPORT_SEND     = 1 << 0, /* Send operations support on-demand paging */
        IBV_ODP_SUPPORT_RECV     = 1 << 1, /* Receive operations support on-demand paging */
        IBV_ODP_SUPPORT_WRITE    = 1 << 2, /* RDMA-Write operations support on-demand paging */
        IBV_ODP_SUPPORT_READ     = 1 << 3, /* RDMA-Read operations support on-demand paging */
        IBV_ODP_SUPPORT_ATOMIC   = 1 << 4, /* RDMA-Atomic operations support on-demand paging */
};

struct ibv_tso_caps {
        uint32_t max_tso;        /* Maximum payload size in bytes supported for segmentation by TSO engine.*/
        uint32_t supported_qpts; /* Bitmap showing which QP types are supported by TSO operation. */
};

struct ibv_rss_caps {
        uint32_t supported_qpts;                   /* Bitmap showing which QP types are supported RSS */
        uint32_t max_rwq_indirection_tables;       /* Max receive work queue indirection tables */
        uint32_t max_rwq_indirection_table_size;   /* Max receive work queue indirection table size */
        uint64_t rx_hash_fields_mask;              /* Mask with enum ibv_rx_hash_fields to know which incoming packet’s f
        uint8_t  rx_hash_function;                 /* Mask with enum ibv_rx_hash_function_flags to know which hash functio
};

struct ibv_packet_pacing_caps {
       uint32_t qp_rate_limit_min; /* Minimum rate limit in kbps */
       uint32_t qp_rate_limit_max; /* Maximum rate limit in kbps */
       uint32_t supported_qpts;    /* Bitmap showing which QP types are supported. */
};

enum ibv_raw_packet_caps {
                IBV_RAW_PACKET_CAP_CVLAN_STRIPPING        = 1 << 0, /* CVLAN stripping is supported */
                IBV_RAW_PACKET_CAP_SCATTER_FCS            = 1 << 1, /* FCS scattering is supported */
                IBV_RAW_PACKET_CAP_IP_CSUM            = 1 << 2, /* IP CSUM offload is supported */
};
```

RETURN VALUE

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

NOTES

The maximum values returned by this function are the upper limits of supported resources by the device. However, it may not be possible to use these maximum values, since the actual number of any resource that can be created may be limited by the machine configuration, the amount of host memory, user permissions, and the amount of resources already in use by other users/processes.

SEE ALSO

ibv_query_device(3), **ibv_open_device**(3), **ibv_query_port**(3), **ibv_query_pkey**(3), **ibv_query_gid**(3)

AUTHORS

Majd Dibbiny <majd@mellanox.com>