

NAME

`ibv_query_device` - query an RDMA device's attributes

SYNOPSIS

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

```
int ibv_query_device(struct ibv_context *context,
                    struct ibv_device_attr *device_attr);
```

DESCRIPTION

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

```
struct ibv_device_attr {
    char          fw_ver[64];          /* FW version */
    uint64_t      node_guid;           /* Node GUID (in network byte order) */
    uint64_t      sys_image_guid;      /* System image GUID (in network byte order) */
    uint64_t      max_mr_size;         /* Largest contiguous block that can be registered */
    uint64_t      page_size_cap;       /* Supported memory shift sizes */
    uint32_t      vendor_id;           /* Vendor ID, per IEEE */
    uint32_t      vendor_part_id;      /* Vendor supplied part ID */
    uint32_t      hw_ver;              /* Hardware version */
    int           max_qp;               /* Maximum number of supported QPs */
    int           max_qp_wr;           /* Maximum number of outstanding WR on any work queue */
    int           device_cap_flags;     /* HCA capabilities mask */
    int           max_sge;              /* Maximum number of s/g per WR for SQ & RQ of QP for non RDMA E */
    int           max_sge_rd;          /* Maximum number of s/g per WR for RDMA Read operations */
    int           max_cq;              /* Maximum number of supported CQs */
    int           max_cqe;              /* Maximum number of CQE capacity per CQ */
    int           max_mr;               /* Maximum number of supported MRs */
    int           max_pd;               /* Maximum number of supported PDs */
    int           max_qp_rd_atom;       /* Maximum number of RDMA Read & Atomic operations that can b */
    int           max_ee_rd_atom;       /* Maximum number of RDMA Read & Atomic operations that can b */
    int           max_res_rd_atom;      /* Maximum number of resources used for RDMA Read & Atomic op */
    int           max_qp_init_rd_atom; /* Maximum depth per QP for initiation of RDMA Read & Atomic */
    int           max_ee_init_rd_atom; /* Maximum depth per EEC for initiation of RDMA Read & Atomic */
    enum ibv_atomic_cap atomic_cap;     /* Atomic operations support level */
    int           max_ee;               /* Maximum number of supported EE contexts */
    int           max_rdd;              /* Maximum number of supported RD domains */
    int           max_mw;               /* Maximum number of supported MWs */
};
```

```

int          max_raw_ipv6_qp;    /* Maximum number of supported raw IPv6 datagram QPs */
int          max_raw_ethy_qp;    /* Maximum number of supported Ethertype datagram QPs */
int          max_mcast_grp;     /* Maximum number of supported multicast groups */
int          max_mcast_qp_attach; /* Maximum number of QPs per multicast group which can be attached */
int          max_total_mcast_qp_attach; /* Maximum number of QPs which can be attached to multicast */
int          max_ah;            /* Maximum number of supported address handles */
int          max_fmr;          /* Maximum number of supported FMRs */
int          max_map_per_fmr;   /* Maximum number of (re)maps per FMR before an unmap operation */
int          max_srq;          /* Maximum number of supported SRQs */
int          max_srq_wr;       /* Maximum number of WRs per SRQ */
int          max_srq_sge;      /* Maximum number of s/g per SRQ */
uint16_t     max_pkeys;        /* Maximum number of partitions */
uint8_t      local_ca_ack_delay; /* Local CA ack delay */
uint8_t      phys_port_cnt;    /* Number of physical ports */
};

```

RETURN VALUE

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

AUTHORS

Dotan Barak <dotanba@gmail.com>