

```

#include "it_api_os_specific.h"

#define IN
#define OUT

/* define IT-API 1.0 variable name mappings for
   it_rc_only_attributes_t */
#define rdma_read_inflight_incoming rdma_read_ird
#define rdma_read_inflight_outgoing rdma_read_ord

/* define IT-API 1.0 variable name mappings for it_ia_info_t */
#define ird_support ird_ord_ia_support
#define ord_support ird_ord_ia_support

/* define IT-API 1.0 name mappings for it_make_rdma_addr */
#define it_make_rdma_addr it_make_rdma_addr_absolute

/* typedefs */
typedef enum
{
    IT_SUCCESS = 0,
    IT_ERR_ABORT,
    IT_ERR_ACCESS,
    IT_ERR_ADDRESS,
    IT_ERR_AEVD_NOT_ALLOWED,
    IT_ERR_ASYNC_AFF_EVD_EXISTS,
    IT_ERR_ASYNC_UNAFF_EVD_EXISTS,
    IT_ERR_CANNOT_RESET,
    IT_ERR_CONN_QUAL_BUSY,
    IT_ERR_EP_TIMEWAIT,
    IT_ERR_EVD_BUSY,
    IT_ERR_EVD_QUEUE_FULL,
    IT_ERR_FAULT,
    IT_ERR_IA_CATASTROPHE,
    IT_ERR_INTERRUPT,
    IT_ERR_INVALID_ADDRESS,
    IT_ERR_INVALID_AEVD,
    IT_ERR_INVALID_AH,
    IT_ERR_INVALID_ETIMEOUT,
    IT_ERR_INVALID_CM_RETRY,
    IT_ERR_INVALID_CN_EST_FLAGS,
    IT_ERR_INVALID_CN_EST_ID,
    IT_ERR_INVALID_CONN_EVD,
    IT_ERR_INVALID_CONN_QUAL,
    IT_ERR_INVALID_CONVERSION,
    IT_ERR_INVALID_DTO_FLAGS,
    IT_ERR_INVALID_EP,
    IT_ERR_INVALID_EP_ATTR,
    IT_ERR_INVALID_EP_KEY,
    IT_ERR_INVALID_EP_STATE,
    IT_ERR_INVALID_EP_TYPE,
    IT_ERR_INVALID_EVD,
    IT_ERR_INVALID_EVD_STATE,
    IT_ERR_INVALID_EVD_TYPE,
    IT_ERR_INVALID_FLAGS,
    IT_ERR_INVALID_HANDLE,
    IT_ERR_INVALID_IA,

```

IT_ERR_INVALID_LENGTH,
IT_ERR_INVALID_LISTEN,
IT_ERR_INVALID_LMR,
IT_ERR_INVALID_LTIMEOUT,
IT_ERR_INVALID_MAJOR_VERSION,
IT_ERR_INVALID_MASK,
IT_ERR_INVALID_MINOR_VERSION,
IT_ERR_INVALID_NAME,
IT_ERR_INVALID_NETADDR,
IT_ERR_INVALID_NUM_SEGMENTS,
IT_ERR_INVALID_PDATA_LENGTH,
IT_ERR_INVALID_PRIVS,
IT_ERR_INVALID_PZ,
IT_ERR_INVALID_QUEUE_SIZE,
IT_ERR_INVALID_RECV_EVD,
IT_ERR_INVALID_RECV_EVD_STATE,
IT_ERR_INVALID_REQ_EVD,
IT_ERR_INVALID_REQ_EVD_STATE,
IT_ERR_INVALID_RETRY,
IT_ERR_INVALID_RMR,
IT_ERR_INVALID_RNR_RETRY,
IT_ERR_INVALID_RTIMEOUT,
~~IT_ERR_INVALID_SGID,~~
~~IT_ERR_INVALID_SLID,~~
IT_ERR_INVALID_SOFT_EVD,
IT_ERR_INVALID_SOURCE_PATH,
IT_ERR_INVALID_SPIGOT,
IT_ERR_INVALID_THRESHOLD,
IT_ERR_INVALID_UD_STATUS,
IT_ERR_INVALID_UD_SVC,
IT_ERR_INVALID_UD_SVC_REQ_ID,
IT_ERR_LMR_BUSY,
IT_ERR_MISMATCH_FD,
IT_ERR_NO_CONTEXT,
IT_ERR_NO_PERMISSION,
IT_ERR_PAYLOAD_SIZE,
IT_ERR_PDATA_NOT_SUPPORTED,
IT_ERR_PZ_BUSY,
IT_ERR_QUEUE_EMPTY,
IT_ERR_RANGE,
IT_ERR_RESOURCES,
IT_ERR_RESOURCE_IRD,
IT_ERR_RESOURCE_LMR_LENGTH,
IT_ERR_RESOURCE_ORD,
IT_ERR_RESOURCE_QUEUE_SIZE,
IT_ERR_RESOURCE_RECV_DTO,
IT_ERR_RESOURCE_REQ_DTO,
IT_ERR_RESOURCE_RRSEG,
IT_ERR_RESOURCE_RSEG,
IT_ERR_RESOURCE_RWSEG,
IT_ERR_RESOURCE_SSEG,
IT_ERR_TIMEOUT_EXPIRED,
IT_ERR_TOO_MANY_POSTS,
IT_ERR_WAITER_LIMIT,
IT_ERR_INVALID_SRQ,
IT_ERR_SOFT_HI_WATERMARK,
IT_ERR_HARD_HI_WATERMARK,

```
IT_ERR_INVALID_WATERMARK,  
IT_ERR_INVALID_RECV DTO,  
IT_ERR_INVALID SRQ SIZE,  
IT_ERR SRQ LOW WATERMARK,  
IT_ERR SRQ BUSY,  
IT_ERR SRQ NOT SUPPORTED,  
IT_ERR_INVALID ADDR MODE,  
IT_ERR_INVALID RMR TYPE,  
IT_ERR_OP NOT SUPPORTED,  
IT_ERR_EP BUSY  
} it_status_t;
```

```
typedef uint32_t it_rmr_context_t;
```

```
#ifdef IT_32BIT  
typedef uint32_t it_length_t; /* a 32-bit platform */  
#else  
typedef uint64_t it_length_t; /* a 64-bit platform */  
#endif
```

```
typedef enum  
{  
    IT_PRIV_NONE = 0x0001,0x0000, /* needed for IT-API 1.0 compat */  
    IT_PRIV_LOCAL_READ_ONLY = 0x0002 = 0x0001,  
    IT_PRIV_LOCAL_WRITE = 0x0002,  
    IT_PRIV_LOCAL = 0x0003,  
    IT_PRIV_DEFAULT = 0x0003, /* deprecated by IT_PRIV_LOCAL */  
    IT_PRIV_REMOTE_READ = 0x0004,  
    IT_PRIV_REMOTE_WRITE = 0x0008,  
    IT_PRIV_REMOTE = 0x0010x000c,  
    IT_PRIV_ALL = 0x0020,0x000f  
    IT_PRIV_DEFAULT = 0x0040  
} it_mem_priv_t;
```

```
typedef enum  
{  
    IT_LMR_FLAG_NONE = 0x0001,  
    IT_LMR_FLAG_SHARED = 0x0002,  
    IT_LMR_FLAG_NONCOHERENT = 0x0004  
} it_lmr_flag_t;
```

```
typedef enum  
{  
    IT_RMR_TYPE_DEFAULT = 0,  
    IT_RMR_TYPE_NARROW = 1,  
    IT_RMR_TYPE_WIDE = 2  
} it_rmr_type_t;
```

```
typedef enum  
{  
    IT_ADDR_MODE_ABSOLUTE = 0,  
    IT_ADDR_MODE_RELATIVE = 1  
} it_addr_mode_t;
```

```
typedef uint64_t it_ud_svc_req_identifier_t;
```

```
typedef uint64_t it_cn_est_identifier_t;
```

```
/* it_boolean_t.txt */
```

```
typedef enum
{
    IT_FALSE = 0,
    IT_TRUE = 1
} it_boolean_t;
```

```
/* it_handle_t.txt */
```

```
typedef enum
{
    IT_HANDLE_TYPE_ADDR,
    IT_HANDLE_TYPE_EP,
    IT_HANDLE_TYPE_EVD,
    IT_HANDLE_TYPE_IA,
    IT_HANDLE_TYPE_LISTEN,
    IT_HANDLE_TYPE_LMR,
    IT_HANDLE_TYPE_PZ,
    IT_HANDLE_TYPE_RMR,
    IT_HANDLE_TYPE_UD_SVC_REQ,
    IT_HANDLE_TYPE_SRQ
} it_handle_type_enum_t;
```

```
typedef void *it_handle_t;
```

```
#define IT_NULL_HANDLE _____((it_handle_t) NULL)
```

```
typedef struct it_addr_handle_s *it_addr_handle_t;
typedef struct it_ep_handle_s *it_ep_handle_t;
typedef struct it_evd_handle_s *it_evd_handle_t;
typedef struct it_ia_handle_s *it_ia_handle_t;
typedef struct it_listen_handle_s *it_listen_handle_t;
typedef struct it_lmr_handle_s *it_lmr_handle_t;
typedef struct it_pz_handle_s *it_pz_handle_t;
typedef struct it_rmr_handle_s *it_rmr_handle_t;
typedef struct it_ud_svc_req_handle_s *it_ud_svc_req_handle_t;
```

```
/* it_conn_qual_t.txt */
```

```
/* Enumerates all the possible Connection Qualifier types
supported by the API. */ typedef struct it_srq_handle_s *it_srq_handle_t;
```

```
typedef enum
{
```

```
    /* IANA (TCP/UDP) Port Number */
```

```
    IT_IANA_PORT = 0x10x01,
```

```
    /* InfiniBand Service ID, as described in section 12.7.3 of
    Volume 1 of the InfiniBand specification. */
```

```
    IT_IB_SERVICEID = 0x20x02,
```

```
    /* VIA Connection Discriminator */
```

```
    IT_VIA_DISCRIMINATOR = 0x40x04,
```

```

| /* iWARP local and remote IP (IANA) port object */
| IT_IANA_LR_PORT = 0x08
| } it_conn_qual_type_t;

| /* Defines the Connection Qualifier format for a VIA
| "connection discriminator".
| The API imposes a fixed upper bound on the discriminator size. */

| #define IT_MAX_VIA_DISC_LEN 64

typedef struct
{

    /* The total number of bytes in the array below */
    /* that are significant */
    uint16_t len;

    /* VIA connection discriminator, which is an array of bytes */
    unsigned char discriminator[IT_MAX_VIA_DISC_LEN];

} it_via_discriminator_t;

| /* This defines the Connection Qualifier for InfiniBand,
| which is the 64 bit Service ID */
typedef uint64_t it_ib_serviceid_t;

| /* This describes a Connection Qualifier suitable for input to
| several routines in the API. */
typedef struct
{
| uint16_t local;
| uint16_t remote;
| it_iana_lr_port_t;
} it_conn_qual_type_t;

| typedef struct
| {

    /* The discriminator for the union below. */
    it_conn_qual_type_t type;

    union
    {

        /* IANA Port Number, in network byte order */
        uint16_t port;

        /* InfiniBand Service ID, in network byte order */
        it_ib_serviceid_t serviceid;

        /* VIA connection discriminator. */
        it_via_discriminator_t discriminator;

| /* IANA local/remote Port numbers, in network byte order */
| it_iana_lr_port_t lr_port;

    } conn_qual;

```

```
} it_conn_qual_t;
```

```
/* it_context_t.txt */
```

```
typedef union  
{  
    void *ptr;  
    uint64_t index;  
} it_context_t;
```

```
/* it_dto_cookie_t.txt */
```

```
typedef uint64_t it_dto_cookie_t;
```

```
/* it_dto_status_t.txt */
```

```
typedef enum  
{  
    IT_DTO_SUCCESS = 0,  
    IT_DTO_ERR_LOCAL_LENGTH = 1,  
    IT_DTO_ERR_LOCAL_EP = 2,  
    IT_DTO_ERR_LOCAL_PROTECTION = 3,  
    IT_DTO_ERR_FLUSHED = 4,  
    IT_RMR_OPERATION_FAILED = 5,  
    IT_DTO_ERR_BAD_RESPONSE = 6,  
    IT_DTO_ERR_REMOTE_ACCESS = 7,  
    IT_DTO_ERR_REMOTE_RESPONDER = 8,  
    IT_DTO_ERR_TRANSPORT = 9,  
    IT_DTO_ERR_RECEIVER_NOT_READY = 10,  
    IT_DTO_ERR_PARTIAL_PACKET = 11,  
    IT_DTO_ERR_LOCAL_MM_OPERATION = 12  
} it_dto_status_t;
```

```
/* it_dto_flags_t.txt */
```

```
typedef enum  
{  
    /* If flag set, completion generates a local Eventevent */  
    IT_COMPLETION_FLAG = 0x01,  
  
    /* If flag set, completion causecauses local Notification */  
    IT_NOTIFY_FLAG = 0x02,  
  
    /* If flag set, receipt of DTO at remote will cause  
    Notification at remote */  
    IT_SOLICITED_WAIT_FLAG = 0x04,  
  
    /* If flag set, DTO processing will not start if  
    previously posted RDMA Reads are not complete. */  
    IT_BARRIER_FENCE_FLAG = 0x08,  
} it_dto_flags_t;
```

```
/* it_net_addr_t.txt */
```

```
/* Enumerates all the possible Network Address types supported  
by the API. */
```

```
typedef enum
```

```

{

/* IPv4 address */
IT_IPV4 = 0x1,

/* IPv6 address */
IT_IPV6 = 0x2,

/* InfiniBand GID */
IT_IB_GID = 0x3,

/* VIA Network Address */
IT_VIA_HOSTADDR = 0x4
} it_net_addr_type_t;

/* Defines the Network Address format for a VIA "host address".
The API has a fixed upper bound on the maximum sized VIA
address it will support */

#define IT_MAX_VIA_ADDR_LEN 64

typedef struct
{

/* The number of bytes in the array below that are
significant */
uint16_t len;

/* VIA host address, which is an array of bytes */
unsigned char hostaddr[IT_MAX_VIA_ADDR_LEN];

} it_via_net_addr_t;

/* This defines the Network Address format for the InfiniBand
GID, which is just an IPv6 address. */
typedef struct in6_addr it_ib_gid_t;

/* This describes a Network Address suitable for input to several
routines in the API. */
typedef struct
{

/* The discriminator for the union below. */
it_net_addr_type_t addr_type;

union
{

/* IPv4 address, in network byte order */
struct in_addr ipv4;

/* IPv6 address, in network byte order */
struct in6_addr ipv6;

/* InfiniBand GID, in network byte order */
it_ib_gid_t gid;


```

```

|     /* VIA Network Address- */
|     it_via_net_addr_t via;

|     } addr;

| } it_net_addr_t;

| /* it_ia_info_t.txt */
| /* Enumerates all the transport types supported by the API. */
typedef enum
{
|     /* InfiniBand Native Transport */
|     IT_IB_TRANSPORT = 1,

|     /* VIA host Interface using IP transport, supporting
|       only the Reliable Delivery reliability level */
|     IT_VIA_IP_TRANSPORT = 2,

|     /* VIA host Interface, using Fibre Channel transport, supporting
|       only the Reliable Delivery reliability level */
|     IT_VIA_FC_TRANSPORT = 3,

|     /* iWARP over TCP transport */
|     IT_IWARP_TCP_TRANSPORT = 4,

|     /* Vendor-proprietary Transport */
|     IT_VENDOR_TRANSPORT = 1000
| } it_transport_type_t;

| /* Transport Service Type definitions. */
typedef enum
{
|     /* Reliable Connected Transport Service Type */
|     IT_RC_SERVICE = 0x1,

|     /* Unreliable Datagram Transport Service Type */
|     IT_UD_SERVICE = 0x2,

| } it_transport_service_type_t;

| /* The following structure describes an Interface Adapter Spigot */
typedef struct
| {
|     /* Spigot identifier */
|     size_t spigot_id;

|     /* Maximum sized Send operation for the RC service
|       on
| ----- this Spigot.spigot. */
|     size_t max_rc_send_len;

|     /* Maximum sized RDMA Read/Write operation for the RC service on
|       this Spigot.spigot. */

```

```

size_t max_rc_rdma_len;

/* Maximum sized Send operation for the UD service
   on
   this Spigot.spigot. */
size_t max_ud_send_len;

/* Indicates whether the Spigot is online or offline. A
   An IT_TRUE
   value means online. */
it_boolean_t spigot_online;

/* A mask indicating which Connection Qualifier types
   this
   IA supports for input to it_ep_connect and
   it_ud_service_request_handle_create. The bits in the
   mask are
   an inclusive OR of the values for Connection
   Qualifier types
   that this IA supports. - */
it_conn_qual_type_t active_side_conn_qual;

/* A mask indicating which Connection Qualifier types this to
   it_listen_create. - The bits in the mask are an inclusive OR
   of
   the values for Connection Qualifier types that this IA
   supports. - */
it_conn_qual_type_t passive_side_conn_qual;

/* The number of Network Addresses associated with Spigot.spigot. */
size_t num_net_addr;

/* Pointer to array of Network Address addresses. - */
it_net_addr_t *net_addr;

} it_spigot_info_t;

/* The following structure is used to identify the vendor associated
   with an IA that uses the IB transport*/
typedef struct
{

/* The NodeInfo:VendorID as described in chapter 14
   of the
   IB spec. */
uint32_t vendor:24;

/* The NodeInfo:DeviceID as described in chapter 14
   of the
   IB spec. */
uint16_t device;

/* The NodeInfo:Revision as described in chapter 14
   of the
   IB spec. */
uint32_t revision;

```

```
} it_vendor_ib_t;
```

```
typedef struct
```

```
{
```

```
__/* The following structure is used to identify the vendor  
associated with an IA that uses a VIA transport*/
```

```
typedef struct
```

```
{
```

```
__/* The "Name" member of the VIP_NIC_ATTRIBUTES structure,  
as  
described in the VIA spec. */  
char name[64];
```

```
/* The "HardwareVersion" member of the VIP_NIC_ATTRIBUTES  
structure, as described in the VIA spec. */  
unsigned long hardware;
```

```
/* The "ProviderVersion" member of the VIP_NIC_ATTRIBUTES  
structure, as described in the VIA spec. */  
unsigned long provider;
```

```
} it_vendor_via_t;
```

```
/* The following structure is returned by the it_ia_query function */
```

```
typedef struct
```

```
{
```

```
/* Indicates whether or not vid field contains valid data */  
it_boolean_t valid_vid;
```

```
/* Vendor Identification field - strictly vendor-specific (only  
valid if valid_vid field is IT_TRUE */
```

```
unsigned char vid[64];
```

```
} it_vendor_iwarp_tcp_t;
```

```
typedef struct
```

```
{
```

```
/* Interface Adapter name, as specified in it_ia_create */  
char *ia_name;
```

```
/* The major version number of the latest version of the  
IT-API  
that this IA supports. */  
uint32_t api_major_version;
```

```
/* The minor version number of the latest version of the  
IT-API  
that this IA supports. */  
uint32_t api_minor_version;
```

```
/* The major version number for the software being used to  
control  
this IA.— The IT-API imposes no structure whatsoever  
on this  
number; its meaning is completely IA-dependent. */  
uint32_t sw_major_version;
```

```
/* The minor version number for the software being used to
```

```

control
this IA.— The IT-API imposes no structure whatsoever
on this
number; its meaning is completely IA-dependent. */
uint32_t sw_minor_version;

/* The vendor associated with the IA. —This information is useful
if the Consumer wishes to do device-specific programming. —This
union is discriminated by transport_type. —No vendor
identification is provided for transports not listed below. */
union
{

/* Used if transport_type is IT_IB_TRANSPORT */
it_vendor_ib_t ib;

/* Used if transport_type is IT_VIA_IP_TRANSPORT or
IT_VIA_FC_TRANSPORT */
it_vendor_via_t via;

/* Used if transport_type is IT_IWARP_TCP_TRANSPORT */
it_vendor_iwarp_tcp_t iwarp;

} vendor;

/* The Interface Adapter and platform provide a data alignment hint
to the Consumer to help the Consumer align their data transfer
buffers in a way that is optimal for the performance of the IA.
For example, if the best throughput is obtained by aligning
buffers to 128-byte boundaries, dto_alignment_hint will have the
value 128.— The Consumer may choose to ignore the alignment hint
without any adverse functional impact. —(There may be an adverse
performance impact.) */
uint32_t dto_alignment_hint;

/* The transport type (e.g., InfiniBand) supported by an Interface
Adapter. —An Interface Adapter supports precisely one transport
type. */
it_transport_type_t transport_type;

/* The Transport Service Types supported by this IA. —This is
constructed by doing an inclusive OR of the Transport Service
Type values. */
it_transport_service_type_t supported_service_types;

/* Indicates whether work queuesWork Queues are resizable */
it_boolean_t ep_work_queues resizable_work_queue;

/* Indicates whether the underlying transport used by this IA uses
a three-way handshake for doing Connection establishment.— Note
that if the underlying transport supports a three-way handshake
the Consumer can choose whether to use two handshakes or three
when establishing the Connection.— If the underlying transport
supports a two-way handshake for establishing a Connection, the
Consumer can only use two handshakes when establishing the
Connection. */
it_boolean_t three_way_handshake_support;

```

```
/* Indicates whether Private Data is supported on Connection
   establishment or UD service resolution operations. */
it_boolean_t private_data_support;
```

```
/* Indicates whether the max_message_size field in the
   IT_CM_REQ_CONN_REQUEST_EVENT is valid for this IA. */
it_boolean_t max_message_size_support;
```

```
/* Indicates whether the rdma_read_inflight_incoming field or not the IA
supports IRD/ORD. Affects
whether the rdma_read_ird or rdma_read_ord fields in the
IT_CM_REQ_CONN_REQUEST_EVENT is valid for this IA. */ or the
it_boolean_t ird_support;
```

```
IT_CM_MSG_CONN_ESTABLISHED_EVENT are valid for this IA.
Also affects whether IRD/ORD suppression is an option.
Deprecates IT-API 1.0 values ird_support and ord_support. */
it_boolean_t ird_ord_ia_support;
```

```
/* Indicates whether the rdma_read_inflight_outgoing field
in the IT_CM_REQ_CONN_REQUEST_EVENT IRD/ORD suppression is valid for this
IA. */ supported
for this IA. If this member has a value of IT_TRUE, the
Consumer can control IRD/ORD suppression in it_ep_connect
and it_listen_create. Otherwise they cannot. */
it_boolean_t ird_ord_supportsuppressible;
```

```
/* Indicates whether the IA generates IT_ASYNC_UNAFF_SPIGOT_ONLINE
   Events.- See it_unaffiliated_event_t for details. */
it_boolean_t spigot_online_support;
```

```
/* Indicates whether the IA generates IT_ASYNC_UNAFF_SPIGOT_OFFLINE
   Events.- See it_unaffiliated_event_t for details. */
it_boolean_t spigot_offline_support;
```

```
/* The maximum number of bytes of Private Data supported for the
   it_ep_connect routine.- This will be less than or equal to
   IT_MAX_PRIV_DATA. */
size_t connect_private_data_len;
```

```
/* The maximum number of bytes of Private Data supported for the
   it_ep_accept routine. This will be less than or equal to
   IT_MAX_PRIV_DATA. */
size_t accept_private_data_len;
```

```
/* The maximum number of bytes of Private Data supported for the
   it_reject routine. This will be less than or equal to
   IT_MAX_PRIV_DATA. */
size_t reject_private_data_len;
```

```
/* The maximum number of bytes of Private Data supported for the
   it_ep_disconnect routine. This will be less than or equal to
   IT_MAX_PRIV_DATA. */
size_t disconnect_private_data_len;
```

```
/* The maximum number of bytes of Private Data supported for the
```

```

    it_ud_service_request_handle_create routine. This will be
    less
    than or equal to IT_MAX_PRIV_DATA. */
    size_t ud_req_private_data_len;

    /* The maximum number of bytes of Private Data supported for the
    it_ud_service_reply routine. This will be less than or equal to
    IT_MAX_PRIV_DATA. */
    size_t ud_rep_private_data_len;

    /* Specifies the number of Spigots associated with this Interface
    Adapter */
    size_t num_spigots;

    /* An array of Spigot information data structures. -The array
    contains num_spigots elements. _____*/
    it_spigot_info_t *spigot_info;

    /* The Handle for the EVD that contains the affiliated async Event
    Stream.- If no EVD contains the affiliated_asyncAffiliated Async Event
    Stream,
    this member will have the distinguished value IT_NULL_HANDLE */
    it_evd_handle_t affiliated_err_evd;

    /* The Handle for the EVD that contains the unaffiliated_asyncUnaffiliated
    Async
    Event
    Stream.- If no EVD contains the unaffiliated_asyncUnaffiliated Async Event
    Stream,
    this member will have the distinguished value
    IT_NULL_HANDLE */
    it_evd_handle_t unaffiliated_err_evd;

    /* Indicates whether the IA supports the S-RQ feature */
    it_boolean_t srq_support;

    /* Indicates whether the IA supports the Endpoint Hard
    High Watermark mechanism for limiting the number of Receive
    DTOs that can be in progress on an Endpoint that has an
    associated S-RQ */
    it_boolean_t hard_hi_watermark_support;

    /* Indicates whether the IA supports the Endpoint Soft
    High Watermark mechanism for generating an Affiliated
    Asynchronous Event when the number of Receive DTOs in
    progress on an Endpoint that has an associated S-RQ exceeds
    the Endpoint Soft High Watermark. */
    it_boolean_t soft_hi_watermark_support;

    /* Indicates whether an S-RQ can be resized after it is created */
    it_boolean_t srq_resizable;

    /* Indicates that an iWARP V-RNIC supports a modified qp state
    diagram (outside the RDMAC verbs) */
    it_boolean_t extended_iwarp_qp_states;

    /* Indicates that Implementation supports socket conversion (TDI).

```

```

    This attribute is IT_TRUE if and only if transport_type is
    IT_IWARP_TCP_TRANSPORT. */
    it_boolean_t socket_conversion_support;

    /* Indicates that IA supports increasing ORD
       (decreasing ORD is mandatory for all RNICs) */
    it_boolean_t rdma_read_ord_increasable;

    /* Indicates that IA supports modifying IRD */
    it_boolean_t rdma_read_ird_modifiable;

    /* Bit set indicating which RMR types are supported.
       Possible values are IT_RMR_TYPE_NARROW, IT_RMR_TYPE_WIDE, and
       (IT_RMR_TYPE_NARROW|IT_RMR_TYPE_WIDE). See also it_rmr_type_t. */
    it_rmr_type_t rmr_types_supported;

    /* Indicates whether the IA supports Relative Addressing. See also
       it_addr_mode_t. */
    it_boolean_t addr_mode_relative_support;

    /* Indicates whether the Destination buffer for an RDMA Read DTO
       must have remote or local write permission, and whether or not
       the Endpoint to which an RDMA Read DTO is posted must have RDMA
       Write access enabled. See also it_post_rdma_read. */
    it_boolean_t rdma_read_requires_remote_write;

    /* Indicates whether the IA supports changing the RDMA enables
       after EP creation. */
    it_boolean_t ep_rdma_enables_modifiable;

    /* Indicates whether the IA supports it_post_rdma_read_to_rmr. */
    it_boolean_t rdma_read_local_extensions;

    /* Indicates whether the IA supports DTO EVD overflow detection. */
    it_boolean_t dto_evd_overflow_detection;
} it_ia_info_t;

```

```

/* it_lmr_triplet_t.txt */

```

```

typedef struct
{
    it_lmr_handle_t lmr;
    union
    {
        void *abs;
        it_length_t rel;
    } addr;
    it_length_t length;
} it_lmr_triplet_t;

```

```

/* it_path_t.txt */

```

```

/* This is the remote component of the Path information for the
   InfiniBand transport */

```

```

typedef struct
{

```

```

it_rmr_handle_t rmr;
union
{
  void *abs;
  it_length_t rel;
} addr;
it_length_t length;
} it_rmr_triplet_t;

```

```

typedef struct
{

```

```

  /* Partition Key, as defined in the REQ message for the IB
  CM protocol */

```

```

  uint16_t partition_key;

```

```

  /* Path Packet Payload MTU, as defined in the REQ message
  for the IB CM protocol */

```

```

  uint8_t path_mtu:4;

```

```

  /* PacketLifeTime, as defined in the PathRecord in IB
  specification.— This field is useful for Consumers that
  wish to use timeout values other than the default ones
  for doing Connection establishment. */

```

```

  uint8_t packet_lifetime:6;

```

```

  /* Local Port LID, as defined in the REQ message for the IB
  CM protocol.— The low order bits of this value also
  constitute the "Source Path Bits" that are used to
  create an Address Handle. —*/

```

```

  uint16_t local_port_lid;

```

```

  /* Remote Port LID, as defined in the REQ message for the
  IB CM protocol.— This is also the "Destination LID" used
  to create an Address Handle. —*/

```

```

  uint16_t remote_port_lid;

```

```

  /* Local Port GID in network byte order, as defined in the
  REQ message for the IB CM protocol.— This is also used to
  determine the appropriate "Source GID Index" to be used
  when creating an Address Handle. */

```

```

  it_ib_gid_t local_port_gid;

```

```

  /* Remote Port GID in network byte order, as defined in the
  REQ message for the IB CM protocol.— This is also the
  "Destination GID or MGID" used to create an Address
  Handle. */

```

```

  it_ib_gid_t remote_port_gid;

```

```

  /* Packet Rate, as defined in the REQ message for the IB CM
  protocol.— This is also the "Maximum Static Rate" to be
  used when creating an Address Handle. */

```

```

  uint8_t packet_rate:6;

```

```

  /* SL, as defined in the REQ message for the IB CM
  protocol.— This is also the "Service Level" to be used
  when creating an Address Handle. */

```

```

uint8_t sl:4;

/* Subnet Local, as defined in the REQ message for the IB
   CM protocol.— When creating an Address Handle, setting
   this bit causes a GRH to be included as part of any
   Unreliable Datagram sent using the Address Handle. */
uint8_t subnet_local:1;

/* Flow Label, as defined in the REQ message for the IB CM
   protocol.— This is also the "Flow Label" to be used when
   creating an Address Handle. —This is only valid if
   subnet_local is clear. */
uint32_t flow_label:20;

/* Traffic Class, as defined in the REQ message for the IB
   CM protocol.— This is also the "Traffic Class" to be
   used when creating an Address Handle. —This is only
   valid if subnet_local is clear. */
uint8_t traffic_class;

/* Hop Limit, as defined in the REQ message for the IB CM
   protocol.— This is also the "Hop Limit" to be used when
   creating an Address Handle. This is only valid if
   subnet_local is clear. */
uint8_t hop_limit;
} it_ib_net_endpoint_t;

/* This is the remote component of the Path information for the
   VIA transport */
typedef it_via_net_addr_t it_via_net_endpoint_t;

/* This is the Path data structure used by several routines in
   —typedef enum
   {
   IT_IP_VERS_IPV4 = 0x1,
   IT_IP_VERS_IPV6 = 0x2
   } it_ip_vers_t;

typedef struct
{
/* Designates the type of IP address that is found in
   both the APIladdr and raddr unions below */
it_ip_vers_t ip_vers;

/* Local path element */
union
{
struct in_addr ipv4;
struct in6_addr ipv6;
} laddr;

/* Remote path element */
union
{
struct in_addr ipv4;
struct in6_addr ipv6;
} raddr;
} it_path_t;

```

```

    } raddr;
} it_iwarp_net_endpoint_t;

typedef struct
{
    /* Identifier for the Spigot to be used on the local IA.
       Note that this data structure is always used in a
       Context where the IA associated with the Spigot can be
       deduced. */
    size_t spigot_id;

    /* The transport-independent timeout parameter for how long
       to wait, in microseconds, before timing out a Connection
       establishment attempt using this Path. The timeout
       period for establishing a Connection
       can only be specified on the Active side; the timeout
       period can not be changed on the Passive side. */
    uint64_t timeout;

    /* The remote component of the Path */
    union
    {
        /* For use with InfiniBand */
        it_ib_net_endpoint_t ib;

        /* For use with VIA */
        it_via_net_endpoint_t via;

        /* For use with iWARP */
        it_iwarp_net_endpoint_t iwarp;
    } remote;

} it_path_t;

/* it_ep_attributes_t.txt */

typedef uint32_t it_ud_ep_id_t;
typedef uint32_t it_ud_ep_key_t;

typedef enum
{
    IT_EP_PARAM_ALL = 0x00000001,
    IT_EP_PARAM_IA = 0x00000002,
    IT_EP_PARAM_SPIGOT = 0x00000004,
    IT_EP_PARAM_STATE = 0x00000008,
    IT_EP_PARAM_SERV_TYPE = 0x00000010,
    IT_EP_PARAM_PATH = 0x00000020,
    IT_EP_PARAM_PZ = 0x00000040,
    IT_EP_PARAM_REQ_SEVD = 0x00000080,
    IT_EP_PARAM_RECV_SEVD = 0x00000100,
    IT_EP_PARAM_CONN_SEVD = 0x00000200,
    IT_EP_PARAM_RDMA_RD_ENABLE = 0x00000400,
    IT_EP_PARAM_RDMA_WR_ENABLE = 0x00000800,
    IT_EP_PARAM_MAX_RDMA_READ_SEG = 0x00001000,

```

```

IT_EP_PARAM_MAX_RDMA_WRITE_SEG = 0x00002000,
IT_EP_PARAM_MAX_IRD = 0x00004000,
IT_EP_PARAM_MAX_ORD = 0x00008000,
IT_EP_PARAM_EP_ID = 0x00010000,
IT_EP_PARAM_EP_KEY = 0x00020000,
IT_EP_PARAM_MAX_PAYLOAD = 0x00040000,
IT_EP_PARAM_MAX_REQ_DTO = 0x00080000,
IT_EP_PARAM_MAX_RECV_DTO = 0x00100000,
IT_EP_PARAM_MAX_SEND_SEG = 0x00200000,
IT_EP_PARAM_MAX_RECV_SEG = 0x00400000,
IT_EP_PARAM_SRQ = 0x00800000,
IT_EP_PARAM_SOFT_HI_WATERMARK = 0x01000000,
IT_EP_PARAM_HARD_HI_WATERMARK = 0x02000000
} it_ep_param_mask_t;

```

```

/*
* the it_ep_param_mask_t value in the comment beside or
* following each attribute is the mask value used to select
* the attribute in the it_ep_query and it_ep_modify calls
*/

```

```

typedef struct
{
    it_boolean_t rdma_read_enable;
    /* IT_EP_PARAM_RDMA_RD_ENABLE */
    it_boolean_t rdma_write_enable;
    /* IT_EP_PARAM_RDMA_WR_ENABLE */
    size_t max_rdma_read_segments;
    /* IT_EP_PARAM_MAX_RDMA_READ_SEG */
    size_t max_rdma_write_segments;
    /* IT_EP_PARAM_MAX_RDMA_WRITE_SEG */
    uint32_t rdma_read_inflight_incomingird;
    /* IT_EP_PARAM_MAX_IRD */
    uint32_t rdma_read_inflight_outgoingord;
    /* IT_EP_PARAM_MAX_ORD */
    it_srq_handle_t srq;
    /* IT_EP_PARAM_SRQ */
    size_t soft_hi_watermark;
    /* IT_EP_PARAM_SOFT_HI_WATERMARK */
    size_t hard_hi_watermark;
    /* IT_EP_PARAM_HARD_HI_WATERMARK */
} it_rc_only_attributes_t;

```

```

#define IT_HARD_HI_WATERMARK_DISABLE ((size_t) -1)

```

```

typedef struct
{
    it_ud_ep_id_t ud_ep_id; /* IT_EP_PARAM_EP_ID */
    it_ud_ep_key_t ud_ep_key; /* IT_EP_PARAM_EP_KEY */
} it_remote_ep_info_t;

```

```

typedef struct
{
    it_remote_ep_info_t ep_info;
} it_ud_only_attributes_t;

```

```

typedef union

```

```

{
    it_rc_only_attributes_t rc;
    it_ud_only_attributes_t ud;
} it_service_attributes_t;

typedef struct
{
    size_t max_dto_payload_size;          /* IT_EP_PARAM_MAX_PAYLOAD */
    size_t max_request_dtos;             /* IT_EP_PARAM_MAX_REQ_DTO */
    size_t max_rcv_dtos;                 /* IT_EP_PARAM_MAX_RECV_DTO */
    size_t max_send_segments;           /* IT_EP_PARAM_MAX_SEND_SEG */
    size_t max_rcv_segments;            /* IT_EP_PARAM_MAX_RECV_SEG */

    it_service_attributes_t srv;
} it_ep_attributes_t;

/* it_event_t.txt */
#define IT_EVENT_STREAM_MASK ___ 0xff000

#define IT_TIMEOUT_INFINITE ___ ((uint64_t)(-1))

typedef enum
{
    /* DTO Completion */
    /* * Event Stream for WR/DTO completions */
    IT_DTO_EVENT_STREAM = 0x00000,
    IT_DTO_SEND_CMPL_EVENT = 0x00001,
    IT_DTO_RC_RECV_CMPL_EVENT = 0x00002,
    IT_DTO_UD_RECV_CMPL_EVENT = 0x00003,
    IT_DTO_RDMA_WRITE_CMPL_EVENT = 0x00004,
    IT_DTO_RDMA_READ_CMPL_EVENT = 0x00005,
    IT_RMR_BIND_CMPL_EVENT = 0x00006,

    /*
    * IT_RMR_LINK_CMPL_EVENT = 0x00006,

    /*
    * Event Stream for Communication Management Request Event StreamEvents
    */
    IT_CM_REQ_EVENT_STREAM = 0x01000,
    IT_CM_REQ_CONN_REQUEST_EVENT = 0x01001,
    IT_CM_REQ_UD_SERVICE_REQUEST_EVENT = 0x01002,

    /*
    * Event Stream for Communication Management Message Event StreamEvents
    */
    IT_CM_MSG_EVENT_STREAM = 0x02000,
    IT_CM_MSG_CONN_ACCEPT_ARRIVAL_EVENT = 0x02001,
    IT_CM_MSG_CONN_ESTABLISHED_EVENT = 0x02002,
    IT_CM_MSG_CONN_DISCONNECT_EVENT = 0x02003,
    IT_CM_MSG_CONN_PEER_REJECT_EVENT = 0x02004,
    IT_CM_MSG_CONN_NONPEER_REJECT_EVENT = 0x02005,
    IT_CM_MSG_CONN_BROKEN_EVENT = 0x02006,
    IT_CM_MSG_UD_SERVICE_REPLY_EVENT = 0x02007,

```

```
/* Asynchronous Affiliated */ Event Stream for Affiliated Asynchronous Events
*/
```

```
IT_ASYNC_AFF_EVENT_STREAM = 0x04000,
IT_ASYNC_AFF_SEVD_ENQUEUE_FAILURE = 0x04001,
IT_ASYNC_AFF_EP_FAILURE = 0x04002,
IT_ASYNC_AFF_EP_BAD_TRANSPORT_OPCODE = 0x04003,
IT_ASYNC_AFF_EP_LOCAL_ACCESS_VIOLATION = 0x04004,
IT_ASYNC_AFF_EP_REQ_DROPPED = 0x04005,
IT_ASYNC_AFF_EP_RDMAW_ACCESS_VIOLATION = 0x04006,
IT_ASYNC_AFF_EP_RDMAW_CORRUPT_DATA = 0x04007,
IT_ASYNC_AFF_EP_RDMAR_ACCESS_VIOLATION = 0x04008,
```

```
/* Asynchronous Non Affiliated Event Stream */
```

```
IT_ASYNC_AFF_EP_LOCAL_ACCESS_VIOLATION = 0x04020,
IT_ASYNC_AFF_EP_L_ACCESS_VIOLATION = 0x04020,
IT_ASYNC_AFF_EP_L_RECV_ACCESS_VIOLATION = 0x04021,
IT_ASYNC_AFF_EP_L_IRRQ_ACCESS_VIOLATION = 0x04022,
IT_ASYNC_AFF_EP_L_TRANSPORT_ERROR = 0x04023,
IT_ASYNC_AFF_EP_L_LLP_ERROR = 0x04024,
IT_ASYNC_AFF_EP_R_ERROR = 0x04040,
IT_ASYNC_AFF_EP_R_ACCESS_VIOLATION = 0x04041,
IT_ASYNC_AFF_EP_R_RECV_ACCESS_VIOLATION = 0x04042,
IT_ASYNC_AFF_EP_R_RECV_LENGTH_ERROR = 0x04043,
IT_ASYNC_AFF_EP_SOFT_HI_WATERMARK = 0x04060,
IT_ASYNC_AFF_SRQ_LOW_WATERMARK = 0x04100,
```

```
/* Event Stream for Unaffiliated Asynchronous Events */
```

```
IT_ASYNC_UNAFF_EVENT_STREAM = 0x08000,
IT_ASYNC_UNAFF_IA_CATASTROPHIC_ERROR = 0x08001,
/* 0x08001 is deprecated */
IT_ASYNC_UNAFF_SPIGOT_ONLINE = 0x08002,
IT_ASYNC_UNAFF_SPIGOT_OFFLINE = 0x08003,
IT_ASYNC_UNAFF_SEVD_ENQUEUE_FAILURE = 0x08004,
```

```
/* Software */ Event Stream for Software Events */
```

```
IT_SOFTWARE_EVENT_STREAM = 0x10000,
IT_SOFTWARE_EVENT = 0x10001,
```

```
/* AEVD Notification Event Stream */
```

```
/* Event Stream for AEVD Notifications */
IT_AEVD_NOTIFICATION_EVENT_STREAM = 0x20000,
IT_AEVD_NOTIFICATION_EVENT = 0x20001
} it_event_type_t;
```

```
/* it_aevd_notification_event_t.txt */
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t aevd;
    it_evd_handle_t sevd;
} it_aevd_notification_event_t;
```

```
/* it_affiliated_event_t.txt */
```

```
typedef struct
```

```

{
    it_event_type_t event_number;
    it_evd_handle_t evd;

    union
    {
        it_evd_handle_t sevd;
        it_ep_handle_t ep;
        it_srq_handle_t srq;
        } cause;
    } it_affiliated_event_t;

/* it_cm_msg_events.txt */

#define IT_MAX_PRIV_DATA 256

typedef enum
{
    IT_CN_REJ_OTHER = 0,
    IT_CN_REJ_TIMEOUT = 1,
    IT_CN_REJ_BAD_PATH = 2,
    IT_CN_REJ_STALE_CONN = 3,
    IT_CN_REJ_BAD_ORD = 4,
    IT_CN_REJ_RESOURCES = 5,
    IT_CN_REJ_BAD_CONN_PARAMS = 6,
} it_conn_reject_code_t;

typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_cn_est_identifier_t cn_est_id;
    it_ep_handle_t ep;
    uint32_t rdma_read inflight_incomingird;
    uint32_t rdma_read inflight_outgoingord;
    it_path_t dst_path;
    it_conn_reject_code_t reject_reason_code;
    unsigned char private_data[IT_MAX_PRIV_DATA];
    it_boolean_t private_data_present;
} it_connection_event_t;

typedef enum
{
    IT_UD_SVC_EP_INFO_VALID = 0,
    IT_UD_SVC_ID_NOT_SUPPORTED = 1,

    IT_UD_SVC_REQ_REJECTED = 2,
    IT_UD_NO_EP_AVAILABLE = 3,
    IT_UD_REQ_REDIRECTED = 4
} it_ud_svc_req_status_t;

typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_ud_svc_req_handle_t ud_svc;
    it_ud_svc_req_status_t status;

```

```
    it_remote_ep_info_t ep_info;
    it_path_t dst_path;
    unsigned char private_data[IT_MAX_PRIV_DATA];
    it_boolean_t private_data_present;
} it_ud_svc_reply_event_t;
```

```
/* it_cm_req_events.txt */
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_cn_est_identifier_t cn_est_id;
    it_conn_qual_t conn_qual;
    it_net_addr_t source_addr;
    size_t spigot_id;
    uint32_t max_message_size;
    uint32_t rdma_read_inflight_incomingird;
    uint32_t rdma_read_inflight_outgoingord;
    unsigned char private_data[IT_MAX_PRIV_DATA];
    it_boolean_t private_data_present;
} it_conn_request_event_t;
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_ud_svc_req_identifier_t ud_svc_req_id;
    it_conn_qual_t conn_qual;
    it_net_addr_t source_addr;
    size_t spigot_id;
    unsigned char private_data[IT_MAX_PRIV_DATA];
    it_boolean_t private_data_present;
} it_ud_svc_request_event_t;
```

```
/* it_dto_events.txt */
```

```
typedef enum
{
    IBIT UD_IB_GRH_PRESENT = 0x01
} it_dto_ud_flags_t;
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_ep_handle_t ep;
    it_dto_cookie_t cookie;
    it_dto_status_t dto_status;
    uint32_t transferred_length;
} it_dto_cmpl_event_t;
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_ep_handle_t ep;
```

```
it_dto_cookie_t cookie;
it_dto_status_t dto_status;
uint32_t transferred_length;
it_dto_ud_flags_t flags;
it_ud_ep_id_t ud_ep_id;
it_path_t src_path;
} it_all_dto_cmpl_event_t;
```

```
/* it_software_event_t.txt */
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    void *data;
} it_software_event_t;
```

```
/* it_unaffiliated_event_t.txt */
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
    it_ia_handle_t ia;

    size_t spigot_id;
} it_unaffiliated_event_t;
```

```
/* it_event_t.txt */
```

```
typedef struct
{
    it_event_type_t event_number;
    it_evd_handle_t evd;
} it_any_event_t;
```

```
typedef union
```

```
{
    /*
     * The following two union elements are
     * available for programming convenience.
     *
     * The event_number may be used to determine the
     * it_event_type_t of any event-Event. it_any_event_t
     * allows the evdEVD to be determined as well.
     */
```

```
it_event_type_t event_number;
it_any_event_t any;
```

```
/*
 * The remaining union elements correspond to
 * the various it_event_type_t types.
 */
```

```
/*
 * The following two Event structures
 * support the IT_DTO_EVENT_STREAM Event Stream.
```

```

*
* it_dto_cmpl_event_t supports
* only the following events:
*   IT_DTO_SEND_CMPL_EVENT
*   IT_DTO_RC_RECV_CMPL_EVENT
*   IT_DTO_RDMA_WRITE_CMPL_EVENT
*   IT_DTO_RDMA_READ_CMPL_EVENT
*   IT_RMR_BIND_CMPL_EVENT = IT_RMR_LINK_CMPL_EVENT
*
* it_all_dto_cmpl_event_t supports all
* possible DTO and RMR events:
*   IT_DTO_SEND_CMPL_EVENT
*   IT_DTO_RC_RECV_CMPL_EVENT
*   IT_DTO_UD_RECV_CMPL_EVENT
*   IT_DTO_RDMA_WRITE_CMPL_EVENT
*   IT_DTO_RDMA_READ_CMPL_EVENT
*   IT_RMR_BIND_CMPL_EVENT = IT_RMR_LINK_CMPL_EVENT
*/
it_dto_cmpl_event_t dto_cmpl;
it_all_dto_cmpl_event_t all_dto_cmpl;

/*
* The following two Event structures
* support the IT_CM_REQ_EVENT_STREAM Event
* stream:
*
* it_conn_request_event_t supports:
*   IT_CM_REQ_CONN_REQUEST_EVENT
*
* it_ud_svc_request_event_t supports:
*   IT_CM_REQ_UD_SERVICE_REQUEST_EVENT
*/
it_conn_request_event_t conn_req;
it_ud_svc_request_event_t ud_svc_request;

/*
* The following two Event structures
* support the IT_CM_MSG_EVENT_STREAM Event
* stream:
*
* it_connection_event_t supports:
*   IT_CM_MSG_CONN_ACCEPT_ARRIVAL_EVENT
*   IT_CM_MSG_CONN_ESTABLISHED_EVENT
*   IT_CM_MSG_CONN_PEER_REJECT_EVENT
*   IT_CM_MSG_CONN_NONPEER_REJECT_EVENT
*   IT_CM_MSG_CONN_DISCONNECT_EVENT
*   IT_CM_MSG_CONN_BROKEN_EVENT
*
* it_ud_svc_reply_event_t supports:
*   IT_CM_MSG_UD_SERVICE_REPLY_EVENT
*/
it_connection_event_t conn;
it_ud_svc_reply_event_t ud_svc_reply;

/*
* it_affiliated_event_t supports
* the following Event Stream:

```

```

    *      IT_ASYNC_AFF_EVENT_STREAM
    */
it_affiliated_event_t aff_async;

/*
 * it_unaffiliated_event_t supports
 * the following Event Stream:
 *      IT_ASYNC_UNAFF_EVENT_STREAM
 */
it_unaffiliated_event_t unaff_async;

/*
 * it_software_event_t supports
 * the following Event Stream:
 *      IT_SOFTWARE_EVENT_STREAM
 */
it_software_event_t sw;

/*
 * it_aevd_notification_event_t supports
 * the following Event Stream:
 *      IT_AEVD_NOTIFICATION_EVENT_STREAM
 */
it_aevd_notification_event_t aevd_notify;
} it_event_t;

| /* it_ep_state_t.txt */
typedef enum
{
    IT_EP_STATE_UNCONNECTED = 0,
    IT_EP_STATE_ACTIVE1_CONNECTION_PENDING = 1,
    IT_EP_STATE_ACTIVE2_CONNECTION_PENDING = 2,
    IT_EP_STATE_PASSIVE_CONNECTION_PENDING = 3,
    IT_EP_STATE_CONNECTED = 4,
    IT_EP_STATE_NONOPERATIONAL = 5,
    IT_EP_STATE_PASSIVE_WAIT_RDMA_TRANS_REQ = 6
} it_ep_state_rc_t;

typedef enum
{
    IT_EP_STATE_UD_NONOPERATIONAL = 0,
    IT_EP_STATE_UD_OPERATIONAL = 1
} it_ep_state_ud_t;

typedef union
{
    it_ep_state_rc_t rc;
    it_ep_state_ud_t ud;
} it_ep_state_t;

| /* it_dg_remote_ep_addr_t.txt */
typedef struct
{
    it_addr_handle_t addr;
    it_remote_ep_info_t ep_info;
} it_ib_ud_addr_t;

```

```

typedef enum
{
    IT_DG_TYPE_IB_UD
} it_dg_type_t;

typedef struct
{
    it_dg_type_t type;          /* IT_DG_TYPE_IB_UD */
    union
    {
        it_ib_ud_addr_t ud;
    } addr;
} it_dg_remote_ep_addr_t;

typedef enum
{
    IT_AH_PATH_COMPLETE = 0x1
} it_ah_flags_t;

typedef enum
{
    IT_ADDR_PARAM_ALL = 0x0001,
    IT_ADDR_PARAM_IA = 0x0002,
    IT_ADDR_PARAM_PZ = 0x0004,
    IT_ADDR_PARAM_PATH = 0x0008
} it_addr_param_mask_t;

typedef struct
{
    it_ia_handle_t ia;          /* IT_ADDR_PARAM_IA */
    it_pz_handle_t pz;          /* IT_ADDR_PARAM_PZ */
    it_path_t path;            /* IT_ADDR_PARAM_PATH */
} it_addr_param_t;

|
typedef struct
{
    /* Remote CM Response Timeout, as defined in the REQ
       message for the IB CM protocol */
    uint8_t remote_cm_timeout:5;

    /* Local CM Response Timeout, as defined in the REQ
       message for the IB CM protocol */
    uint8_t local_cm_timeout:5;

    /* Retry Count, as defined in the REQ message for the
       IB CM protocol */
    uint8_t retry_count:3;

    /* RNR Retry Count, as defined in the REQ message for
       the IB CM protocol */
    uint8_t rnr_retry_count:3;

    /* Max CM retries, as defined in the REQ message for
       the IB CM protocol */

```

```

uint8_t max_cm_retries:4;

/* Local ACK Timeout, as defined in the REQ message
   for the IB CM protocol */
uint8_t local_ack_timeout:5;

} it_ib_conn_attributes_t;

typedef struct
{
    /* VIA currently has no transport-specific connection
       attributes*/. A dummy entry is defined to allow ANSI
       compilation. */
    void *unused;
} it_via_conn_attributes_t;

typedef struct
{
    /* iWARP currently has no transport-specific connection
       attributes. A dummy entry is defined to allow ANSI
       compilation. */
    void *unused;
} it_iwarp_conn_attributes_t;

typedef union
{
    it_ib_conn_attributes_t ib;
    it_via_conn_attributes_t via;
    it_iwarp_conn_attributes_t iwarp;
} it_conn_attributes_t;

typedef enum
{
    IT_CONNECT_FLAG_TWO_WAY = 0x0001,
    IT_CONNECT_FLAG_THREE_WAY = 0x0002,
    IT_CONNECT_SUPPRESS_IRD_ORD = 0x0004
} it_cn_est_flags_t;

typedef struct
{
    it_ia_handle_t ia;          /* IT_EP_PARAM_IA */
    size_t spigot_id;          /* IT_EP_PARAM_SPIGOT */
    it_ep_state_t ep_state;    /* IT_EP_PARAM_STATE */
    it_transport_service_type_t service_type;
    /* IT_EP_PARAM_SERV_TYPE */
    it_path_t dst_path;        /* IT_EP_PARAM_PATH */
    it_pz_handle_t pz;         /* IT_EP_PARAM_PZ */
    it_evd_handle_t request_sevd;
    /* IT_EP_PARAM_REQ_SEVD */
    it_evd_handle_t recv_sevd; /* IT_EP_PARAM_RECV_SEVD */
    it_evd_handle_t connect_sevd;
    /* IT_EP_PARAM_CONN_SEVD */
    it_ep_attributes_t attr;    /* see it_ep_attributes_t

```

```

|                                     for mask flags for
|                                     attr */
| it_ep_param_t;
|
typedef enum
| {
|     IT_EP_NO_FLAG = 0x00,
|     IT_EP_REUSEADDR = 0x01,
|     IT_EP_SRQ = 0x02
| } it_ep_rc_creation_flags_t;
|
#define IT_THRESHOLD_DISABLE 0
|
typedef enum
| {
|     IT_EVD_DEQUEUE_NOTIFICATIONS = 0x01,
|     IT_EVD_CREATE_FD = 0x02,
|     IT_EVD_OVERFLOW_DEFAULT = 0x04,
|     IT_EVD_OVERFLOW_NOTIFY = 0x08,
|     IT_EVD_OVERFLOW_AUTO_RESET = 0x10
| } it_evd_flags_t;
|
typedef enum
| {
|     IT_EVD_PARAM_ALL = 0x000001,
|     IT_EVD_PARAM_IA = 0x000002,
|     IT_EVD_PARAM_EVENT_NUMBER = 0x000004,
|     IT_EVD_PARAM_FLAG = 0x000008,
|     IT_EVD_PARAM_QUEUE_SIZE = 0x000010,
|     IT_EVD_PARAM_THRESHOLD = 0x000020,
|     IT_EVD_PARAM_AEVD_HANDLE = 0x000040,
|     IT_EVD_PARAM_FD = 0x000080,
|     IT_EVD_PARAM_BOUND = 0x000100,
|     IT_EVD_PARAM_ENABLED = 0x000200,
|     IT_EVD_PARAM_OVERFLOWED = 0x000400
| } it_evd_param_mask_t;
|
typedef struct
| {
|     it_ia_handle_t ia;           /* IT_EVD_PARAM_IA */
|     it_event_type_t event_number; /* IT_EVD_PARAM_EVENT_NUMBER */
|     it_evd_flags_t evd_flag;     /* IT_EVD_PARAM_FLAG */
|     size_t sevd_queue_size;      /* IT_EVD_PARAM_QUEUE_SIZE */
|     size_t sevd_threshold;       /* IT_EVD_PARAM_THRESHOLD */
|     it_evd_handle_t aevd;        /* IT_EVD_PARAM_AEVD_HANDLE */
|     int fd;                      /* IT_EVD_PARAM_FD */
|     it_boolean_t evd_bound;      /* IT_EVD_PARAM_BOUND */
|     it_boolean_t evd_enabled;    /* IT_EVD_PARAM_ENABLED */
|     it_boolean_t evd_overflowed; /* IT_EVD_PARAM_OVERFLOWED */
| } it_evd_param_t;
|
typedef struct
| {
|     /* Most recent major version number of the IT-API supported by the
|        Interface */
|     uint32_t major_version;

```

```

/* Most recent minor version number of the IT-API supported by the
   Interface */
uint32_t minor_version;

/* The transport that the Interface uses, as defined in
   it_ia_info_t. */
it_transport_type_t transport_type;

/* The name of the Interface, suitable for input to it_ia_create.
   The name is a string of maximum length IT_INTERFACE_NAME_SIZE,
   including the terminating NULL character. */
char name[IT_INTERFACE_NAME_SIZE];

} it_interface_t;

typedef enum
{
    IT_LISTEN_NO_FLAG = 0x0000,
    IT_LISTEN_CONN_QUAL_INPUT = 0x0001,
    IT_LISTEN_SUPPRESS_IRD_ORD = 0x0002
} it_listen_flags_t;

typedef enum
{
    IT_LISTEN_PARAM_ALL = 0x0001,
    IT_LISTEN_PARAM_IA_HANDLE = 0x0002,
    IT_LISTEN_PARAM_SPIGOT_ID = 0x0004,
    IT_LISTEN_PARAM_CONNECT_EVD = 0x0008,
    IT_LISTEN_PARAM_CONN_QUAL = 0x0010
} it_listen_param_mask_t;

typedef struct
{
    it_ia_handle_t ia_handle; /* IT_LISTEN_PARAM_IA_HANDLE */
    size_t spigot_id; /* IT_LISTEN_PARAM_SPIGOT_ID */
    it_evd_handle_t connect_evd; /* IT_LISTEN_PARAM_CONNECT_EVD */
    it_conn_qual_t connect_qual; /* IT_LISTEN_PARAM_CONN_QUAL */
} it_listen_param_t;

typedef enum
{
    IT_LMR_PARAM_ALL = 0x000001,
    IT_LMR_PARAM_IA = 0x000002,
    IT_LMR_PARAM_PZ = 0x000004,
    IT_LMR_PARAM_ADDR = 0x000008,
    IT_LMR_PARAM_LENGTH = 0x000010,
    IT_LMR_PARAM_MEM_PRIV = 0x000020,
    IT_LMR_PARAM_FLAG = 0x000040,
    IT_LMR_PARAM_SHARED_ID = 0x000080,
    IT_LMR_PARAM_RMR_CONTEXT = 0x000100,
    IT_LMR_PARAM_ACTUAL_ADDR = 0x000200,
    IT_LMR_PARAM_ACTUAL_LENGTH = 0x000400,
    IT_LMR_PARAM_ADDR_MODE = 0x000800
} it_lmr_param_mask_t;

```

```

typedef struct
{
    it_ia_handle_t ia;           /* IT_LMR_PARAM_IA */
    it_pz_handle_t pz;         /* IT_LMR_PARAM_PZ */
    void *addr;                 /* IT_LMR_PARAM_ADDR */
    it_length_t length;        /* IT_LMR_PARAM_LENGTH */
    it_mem_priv_t privs;        /* IT_LMR_PARAM_MEM_PRIV */
    it_lmr_flag_t flags;        /* IT_LMR_PARAM_FLAG */
    uint32_t shared_id;         /* IT_LMR_PARAM_SHARED_ID */
    it_rmr_context_t rmr_context; /* IT_LMR_PARAM_RMR_CONTEXT */
    void *actual_addr;          /* IT_LMR_PARAM_ACTUAL_ADDR */
    it_length_t actual_length; /* IT_LMR_PARAM_ACTUAL_LENGTH */
    it_addr_mode_t addr_mode; /* IT_LMR_PARAM_ADDR_MODE */
} it_lmr_param_t;

```

```
typedef uint64_t it_rdma_addr_t;
```

```

typedef enum
{
    IT_PZ_PARAM_ALL = 0x01,
    IT_PZ_PARAM_IA = 0x02
} it_pz_param_mask_t;

```

```

typedef struct
{
    it_ia_handle_t ia;           /* IT_PZ_PARAM_IA */
} it_pz_param_t;

```

```

typedef enum
{
    IT_RMR_PARAM_ALL = 0x000001,
    IT_RMR_PARAM_IA = 0x000002,
    IT_RMR_PARAM_PZ = 0x000004,
    IT_RMR_PARAM_LINKED = 0x000008,
    IT_RMR_PARAM_BOUND = 0x000008,
    /* deprecated by IT_RMR_PARAM_LINKED */
    IT_RMR_PARAM_LMR = 0x000010,
    IT_RMR_PARAM_ADDR = 0x000020,
    IT_RMR_PARAM_LENGTH = 0x000040,
    IT_RMR_PARAM_MEM_PRIV = 0x000080,
    IT_RMR_PARAM_RMR_CONTEXT = 0x000100,
    IT_RMR_PARAM_TYPE = 0x000200,
    IT_RMR_PARAM_ADDR_MODE = 0x000400
} it_rmr_param_mask_t;

```

```

/* Need to use "bound" rather than "linked" in IT-API 1.0 */
#ifndef ITAPI_ENABLE_V20_BINDINGS

```

```

typedef struct
{
    it_ia_handle_t ia;           /* IT_RMR_PARAM_IA */
    it_pz_handle_t pz;         /* IT_RMR_PARAM_PZ */
    it_boolean_t bound;        /* IT_RMR_PARAM_BOUNDLINKED */
    it_lmr_handle_t lmr;        /* IT_RMR_PARAM_LMR */
    void *addr;                 /* IT_RMR_PARAM_ADDR */
    it_length_t length;         /* IT_RMR_PARAM_LENGTH */
}

```

```

it_mem_priv_t privs;          /* IT_RMR_PARAM_MEM_PRIV */
it_rmr_context_t rmr_context; /* IT_RMR_PARAM_RMR_CONTEXT */
it_rmr_type_t type;          /* IT_RMR_PARAM_TYPE */
it_addr_mode_t addr_mode;    /* IT_RMR_PARAM_ADDR_MODE */
} it_rmr_param_t;

```

```
#else
```

```
typedef struct
```

```

{
it_ia_handle_t ia;          /* IT_RMR_PARAM_IA */
it_pz_handle_t pz;          /* IT_RMR_PARAM_PZ */
it_boolean_t linked;        /* IT_RMR_PARAM_LINKED */
it_lmr_handle_t lmr;        /* IT_RMR_PARAM_LMR */
void *addr;                 /* IT_RMR_PARAM_ADDR */
it_length_t length;         /* IT_RMR_PARAM_LENGTH */
it_mem_priv_t privs;        /* IT_RMR_PARAM_MEM_PRIV */
it_rmr_context_t rmr_context; /* IT_RMR_PARAM_RMR_CONTEXT */
it_rmr_type_t type;          /* IT_RMR_PARAM_TYPE */
it_addr_mode_t addr_mode;    /* IT_RMR_PARAM_ADDR_MODE */
} it_rmr_param_t;

```

```
#endif /* #ifndef ITAPI_ENABLE_V20_BINDINGS */
```

```
typedef enum
```

```

{
IT_SC_DEFAULT = 0x0000,
IT_SC_NO_REQ_REP = 0x0001,
} it_sc_flags_t;

```

```
typedef enum
```

```

{
IT_SRQ_PARAM_ALL = 0x000001,
IT_SRQ_PARAM_IA = 0x000002,
IT_SRQ_PARAM_PZ = 0x000004,
IT_SRQ_PARAM_MAX_RECV_DTO = 0x000008,
IT_SRQ_PARAM_MAX_RECV_SEG = 0x000010,
IT_SRQ_PARAM_LOW_WATERMARK = 0x000020
} it_srq_param_mask_t;

```

```
typedef struct
```

```

{
it_ia_handle_t ia;          /* IT_SRQ_PARAM_IA */
it_pz_handle_t pz;          /* IT_SRQ_PARAM_PZ */
size_t max_recv_dtos;       /* IT_SRQ_PARAM_MAX_RECV_DTO */
size_t max_recv_segs;       /* IT_SRQ_PARAM_MAX_RECV_SEG */
size_t low_watermark;       /* IT_SRQ_PARAM_LOW_WATERMARK */
} it_srq_param_t;

```

```
typedef enum
```

```

{
IT_UD_PARAM_ALL = 0x00000001,
IT_UD_PARAM_IA_HANDLE = 0x00000002,
IT_UD_PARAM_REQ_ID = 0x00000004,
IT_UD_PARAM_REPLY_EVD = 0x00000008,
IT_UD_PARAM_CONN_QUAL = 0x00000010,
IT_UD_PARAM_DEST_PATH = 0x00000020,

```

```

IT_UD_PARAM_PRIV_DATA = 0x00000040,
IT_UD_PARAM_PRIV_DATA_LENGTH = 0x00000080
} it_ud_svc_req_param_mask_t;

```

```

typedef struct

```

```

{
    it_ia_handle_t ia;          /* IT_UD_PARAM_IA_HANDLE */
    uint32_t request_id;       /* IT_UD_PARAM_REQ_ID */
    it_evd_handle_t reply_evd; /* IT_UD_PARAM_REPLY_EVD */
    it_conn_qual_t conn_qual;  /* IT_UD_PARAM_CONN_QUAL */
    it_path_t destination_path; /* IT_UD_PARAM_DEST_PATH */
    unsigned char private_data[IT_MAX_PRIV_DATA];
    /* IT_UD_PARAM_PRIV_DATA */
    size_t private_data_length;
    /* IT_UD_PARAM_PRIV_DATA_LENGTH */
} it_ud_svc_req_param_t;

```

```

/* prototypes */

```

```

#ifdef ITAPI_ENABLE_V20_BINDINGS

```

```

/*

```

```

    Backwards compatibility mode:

```

```

    For functions whose signature changed from v1.0 to v2.0,
    <it_api.h> converts v1.0 function names to explicit v1.0
    function names. Functions such as it_lmr_create continue
    to use v1.0 signatures.

```

```

*/

```

```

#define it_lmr_create    it_lmr_create10

```

```

#define it_rmr_create    it_rmr_create10

```

```

#define it_rmr_bind      it_rmr_bind10

```

```

#else

```

```

/*

```

```

    Full v2.0 functionality:

```

```

    For functions whose signature changed from v1.0 to v2.0,
    <it_api.h> converts v1.0 function names to explicit v2.0
    function names. Functions such as it_lmr_create use the
    new v2.0 signatures.

```

```

*/

```

```

#define it_lmr_create    it_lmr_create20

```

```

#define it_rmr_create    it_rmr_create20

```

```

#define it_rmr_bind      it_rmr_link

```

```

#endif

```

```

it_status_t it_address_handle_create (IN it_pz_handle_t pz_handle,
                                     IN const it_path_t * destination_path,
                                     IN it_ah_flags_t ah_flags,
                                     OUT it_addr_handle_t * addr_handle);
it_status_t it_address_handle_free (IN it_addr_handle_t addr_handle);
it_status_t it_address_handle_modify (IN it_addr_handle_t addr_handle,
                                     IN it_addr_param_mask_t mask,
                                     IN const it_addr_param_t * params);
it_status_t it_address_handle_query (IN it_addr_handle_t addr_handle,
                                    IN it_addr_param_mask_t mask,
                                    OUT it_addr_param_t * params);
it_status_t it_convert_net_addr (IN const it_net_addr_t * source_addr,

```

```

        IN it_net_addr_type_t addr_type,
        OUT it_net_addr_t * destination_addr);
it_status_t it_ep_accept (IN it_ep_handle_t ep_handle,
        IN it_cn_est_identifier_t cn_est_id,
        IN const unsigned char *private_data,
        IN size_t private_data_length);
it_status_t it_ep_connect (IN it_ep_handle_t ep_handle,
        IN const it_path_t * path,
        IN const it_conn_attributes_t * conn_attr,
        IN const it_conn_qual_t * connect_qual,
        IN it_cn_est_flags_t cn_est_flags,
        IN const unsigned char *private_data,
        IN size_t private_data_length);
it_status_t it_ep_disconnect (IN it_ep_handle_t ep_handle,
        IN const unsigned char *private_data,
        IN size_t private_data_length);
it_status_t it_ep_free (IN it_ep_handle_t ep_handle);
it_status_t it_ep_modify (IN it_ep_handle_t ep_handle,
        IN it_ep_param_mask_t mask,
        IN const it_ep_attributes_t * ep_attr);
it_status_t it_ep_query (IN it_ep_handle_t ep_handle,
        IN it_ep_param_mask_t mask,
        OUT it_ep_param_t * params);
it_status_t it_ep_rc_create (IN it_pz_handle_t pz_handle,
        IN it_evd_handle_t request_sevd_handle,
        IN it_evd_handle_t recv_sevd_handle,
        IN it_evd_handle_t connect_sevd_handle,
        IN it_ep_rc_creation_flags_t flags,
        IN const it_ep_attributes_t * ep_attr,
        OUT it_ep_handle_t * ep_handle);
it_status_t it_ep_reset (IN it_ep_handle_t ep_handle);
it_status_t it_ep_ud_create (IN it_pz_handle_t pz_handle,
        IN it_evd_handle_t request_sevd_handle,
        IN it_evd_handle_t recv_sevd_handle,
        IN const it_ep_attributes_t * ep_attr,
        IN size_t spigot_id,
        OUT it_ep_handle_t * ep_handle);
it_status_t it_evd_create (IN it_ia_handle_t ia_handle,
        IN it_event_type_t event_number,
        IN it_evd_flags_t evd_flag,
        IN size_t sevd_queue_size,
        IN size_t sevd_threshold,
        IN it_evd_handle_t aevd_handle,
        OUT it_evd_handle_t * evd_handle, OUT int *fd);
it_status_t it_evd_dequeue (IN it_evd_handle_t evd_handle,
        OUT it_event_t * event);
it_status_t it_evd_free (IN it_evd_handle_t evd_handle);
it_status_t it_evd_modify (IN it_evd_handle_t evd_handle,
        IN it_evd_param_mask_t mask,
        IN const it_evd_param_t * params);
it_status_t it_evd_post_se (IN it_evd_handle_t evd_handle,
        IN const void *event);
it_status_t it_evd_query (IN it_evd_handle_t evd_handle,
        IN it_evd_param_mask_t mask,
        OUT it_evd_param_t * params);
it_status_t it_evd_wait (IN it_evd_handle_t evd_handle,
        IN uint64_t timeout,

```

```

        OUT it_event_t * event, OUT size_t * nmore);
it_status_t it_get_consumer_context (IN it_handle_t handle,
        OUT it_context_t * context);
it_status_t it_get_handle_type (IN it_handle_t handle,
        OUT it_handle_type_enum_t * type_of_handle);
it_status_t it_get_pathinfo (IN it_ia_handle_t ia_handle,
        IN size_t spigot_id,
        IN const it_net_addr_t * net_addr,
        IN OUT size_t * num_paths,
        OUT size_t * total_paths, OUT it_path_t * paths);
it_status_t it_handoff (IN const it_conn_qual_t * conn_qual,
        IN size_t spigot_id,
        IN it_cn_est_identifier_t cn_est_id);

```

```

uint64_t it_hton64 (uint64_t hostint);

```

```

uint64_t it_ntoh64 (uint64_t hostint);

```

```

it_status_t it_ia_create (IN const char *name,
        IN uint32_t major_version,
        IN uint32_t minor_version,
        OUT it_ia_handle_t * ia_handle);

```

```

it_status_t it_ia_free (IN it_ia_handle_t ia_handle);

```

```

void it_ia_info_free (IN it_ia_info_t * ia_info);

```

```

it_status_t it_ia_query (IN it_ia_handle_t ia_handle,
        OUT it_ia_info_t ** ia_info);

```

```

void it_interface_list (OUT it_interface_t * interfaces,
        IN OUT size_t * num_interfaces,
        IN OUT size_t * total_interfaces);

```

```

it_status_t it_listen_create (IN it_ia_handle_t ia_handle,
        IN size_t spigot_id,
        IN it_evd_handle_t connect_evd,
        IN it_listen_flags_t flags,
        IN OUT it_conn_qual_t * conn_qual,
        OUT it_listen_handle_t * listen_handle);

```

```

it_status_t it_listen_free (IN it_listen_handle_t listen_handle);

```

```

it_status_t it_listen_query (IN it_listen_handle_t listen_handle,
        IN it_listen_param_mask_t mask,
        OUT it_listen_param_t * params);

```

```

/*

```

```

it_lmr_create10 is provided for backwards-compatibility
and may be dropped in a future IT-API version.

```

```

Calls with a suffix "10" can be typically implemented through
direct inlining to the corresponding calls with suffix "20",
providing default arguments as necessary.

```

```

*/

```

```

it_status_t it_lmr_create (IN it_pz_handle_t pz_handle,
        IN void *addr,
        IN it_length_t length,
        IN it_mem_priv_t privs,
        IN it_lmr_flag_t flags,
        IN uint32_t shared_id,
        OUT it_lmr_handle_t * lmr_handle,
        IN OUT it_rmr_context_t * rmr_context);

```

```

/*

```

```

it_lmr_create20 provides the v2.0 functionality
and may be renamed to it_lmr_create in a future IT-API version.

```

```

*/

```

```

it_status_t it_lmr_create20 (IN it_pz_handle_t pz_handle,
IN void *addr,
IN it_length_t length,
IN it_addr_mode_t addr_mode,
IN it_mem_priv_t privs,
IN it_lmr_flag_t flags,
IN uint32_t shared_id,
OUT it_lmr_handle_t * lmr_handle,
IN OUT it_rmr_context_t * rmr_context);
it_status_t it_lmr_free (IN it_lmr_handle_t lmr_handle);
it_status_t it_lmr_modify (IN it_lmr_handle_t lmr_handle,
IN it_lmr_param_mask_t mask,
IN const it_lmr_param_t * params);
it_status_t it_lmr_query (IN it_lmr_handle_t lmr_handle,
IN it_lmr_param_mask_t mask,
OUT it_lmr_param_t * params);
it_status_t it_lmr_sync_rdma_read (IN const it_lmr_triplet_t * local_segments,
IN size_t num_segments);
it_status_t it_lmr_sync_rdma_write (IN const it_lmr_triplet_t *
local_segments, IN size_t num_segments);
it_rdma_addr_t it_make_rdma_addr_absolute (void *addr);
it_rdma_addr_t it_make_rdma_addr_relative (it_length_t offset);
it_status_t it_post_rdma_read (IN it_ep_handle_t ep_handle,
IN const it_lmr_triplet_t * local_segments,
IN size_t num_segments,
IN it_dto_cookie_t cookie,
IN it_dto_flags_t dto_flags,
IN it_rdma_addr_t rdma_addr,
IN it_rmr_context_t rmr_context);
it_status_t it_post_rdma_read_to_rmr (IN it_ep_handle_t ep_handle,
IN const it_rmr_triplet_t *
local_segments, IN size_t num_segments,
IN it_dto_cookie_t cookie,
IN it_dto_flags_t dto_flags,
IN it_rdma_addr_t rdma_addr,
IN it_rmr_context_t rmr_context);
it_status_t it_post_rdma_write (IN it_ep_handle_t ep_handle,
IN const it_lmr_triplet_t * local_segments,
IN size_t num_segments,
IN it_dto_cookie_t cookie,
IN it_dto_flags_t dto_flags,
IN it_rdma_addr_t rdma_addr,
IN it_rmr_context_t rmr_context);
it_status_t it_post_recv (IN it_ep_handle_t ep_handle,
IN const it_lmr_triplet_t * local_segments,
IN size_t num_segments,
IN it_dto_cookie_t cookie,
IN it_dto_flags_t dto_flags);
it_status_t it_post_recvfrom (IN it_ep_handle_t ep_handle,
IN const it_lmr_triplet_t * local_segments,
IN size_t num_segments,
IN it_dto_cookie_t cookie,
IN it_dto_flags_t dto_flags);
it_status_t it_post_send (IN it_ep_handle_t ep_handle,
IN const it_lmr_triplet_t * local_segments,
IN size_t num_segments,
IN it_dto_cookie_t cookie,

```

```

        IN it_dto_flags_t dto_flags);
it_status_t it_post_sendto (IN it_ep_handle_t ep_handle,
        IN const it_lmr_triplet_t * local_segments,
        IN size_t num_segments,
        IN it_dto_cookie_t cookie,
        IN it_dto_flags_t dto_flags,
        IN const it_dg_remote_ep_addr_t * remote_ep_addr);
it_status_t it_pz_create (IN it_ia_handle_t ia_handle,
        OUT it_pz_handle_t * pz_handle);
it_status_t it_pz_free (IN it_pz_handle_t pz_handle);
it_status_t it_pz_query (IN it_pz_handle_t pz_handle,
        IN it_pz_param_mask_t mask,
        OUT it_pz_param_t * params);
it_status_t it_reject (IN it_cn_est_identifier_t cn_est_id,
        IN const unsigned char *private_data,
        IN size_t private_data_length);

```

```

/* IT-API 1.0 prototype for B/W compatibility */
it_status_t it_rmr_bind10 (IN it_rmr_handle_t rmr_handle,
        IN it_lmr_handle_t lmr_handle,
        IN void *addr,
        IN it_length_t length,
        IN it_mem_priv_t privs,
        IN it_ep_handle_t ep_handle,
        IN it_dto_cookie_t cookie,
        IN it_dto_flags_t dto_flags,
        OUT it_rmr_context_t * rmr_context);

```

```

/*
    it_rmr_create10 is provided for backwards compatibility
    and may be dropped in a future IT-API version.

```

```

    Calls with a suffix "10" can be typically implemented through
    direct inlining to the corresponding calls with suffix "20",
    providing default arguments as necessary.

```

```

*/
it_status_t it_rmr_create10 (IN it_pz_handle_t pz_handle,
        OUT it_rmr_handle_t * rmr_handle);

```

```

/*
    it_rmr_create20 provides the v2.0 functionality
    and may be renamed to it_rmr_create in a future IT-API version.

```

```

*/
it_status_t it_rmr_create20 (IN it_pz_handle_t pz_handle,
        IN it_rmr_type_t rmr_type,
        OUT it_rmr_handle_t * rmr_handle);

```

```

it_status_t it_rmr_free (IN it_rmr_handle_t rmr_handle);
it_status_t it_rmr_link (IN it_rmr_handle_t rmr_handle,
        IN it_lmr_handle_t lmr_handle,
        IN void *addr,
        IN it_length_t length,
        IN it_addr_mode_t addr_mode,
        IN it_mem_priv_t privs,
        IN it_ep_handle_t ep_handle,
        IN it_dto_cookie_t cookie,

```

