AIX Version 6.1

Remote Direct Memory Access

AIX Version 6.1

Remote Direct Memory Access

Note
Before using this information and the product it supports, read the information in "Notices" on page 15.

This edition applies to AIX Version 6.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2013, 2014.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2013, 2014

About this document

This document provides experienced C programmers with detailed information about programming by using Open Fabrics Enterprise Distribution (OFED) verbs over Internet Wide Area RDMA Protocol (iWARP) or RDMA Network Interface Controller (RNIC) fabrics in the AIX® operating system.

To use the document effectively, you should be familiar with commands, system calls, subroutines, file formats, and special files.

Highlighting

The following highlighting conventions are used in this document:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Case-sensitivity in AIX

Everything in the AIX operating system is case-sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type `LS`, the system responds that the command `is not found`. Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

© Copyright IBM Corp. 2013, 2014

Remote Direct Memory Access

Experienced C programmers can find detailed information about programming with Remote Direct Memory Access (RDMA) verbs and Open Fabrics Enterprise Distribution (OFED) verbs in the AIX operating system.

To use the information effectively, you must be familiar with commands, system calls, subroutines, file formats, and special files.

What's new in Remote Direct Memory Access

Read about new or significantly changed information for the Remote Direct Memory Access topic collection.

How to see what's new or changed

In this PDF file, you might see revision bars (|) in the left margin that identify new and changed information.

November 2013

The following information is a summary of the updates made to this topic collection:

- All the topic collection included in the Remote Direct Memory Access is new.

Open Fabrics Enterprise Distribution (OFED)

Learn how to get started with Open Fabrics Enterprise Distribution (OFED) verbs programming in the AIX operating system. The OFED verbs allows applications that require high throughput and low latency to use the Remote Direct Memory Access (RDMA) feature.

Concepts for OFED

The verb layer for Open Fabrics Enterprise Distribution (OFED) verbs are common for InfiniBand, RDMA over Converged Ethernet (RoCE), Internet Wide Area RDMA Protocol (iWARP), and the verbs that are derived from the InfiniBand architecture.

Hardware requirements

The AIX operating system supports RDMA over Converged Ethernet (RoCE) adapters. The hardware that supports RoCE RDMA in AIX is called the PCIe2 10 GbE RoCE adapter.

Software requirements

The AIX OFED Verbs is based on the OFED 1.5 code of OpenFabrics Alliance. The 32-bit, and 64-bit user applications of the OFED code are supported on AIX operating system. The following libraries are delivered with the installation of RDMA:

- Librdmacm
- Libibverbs

Verbs API

An AIX application can determine the verbs API that is either the Open Fabrics Enterprise Distribution (OFED) verbs or the AIX InfiniBand (IB) verbs that must communicate with a specific destination.

© Copyright IBM Corp. 2013, 2014

The following example in pseudocode tests the result of the **rdma_resolve_addr** command on the required remote address to determine the OFED verbs that can be used.

The program returns the following values:

- **0**- if the communication with the destination can be established by using the OFED verbs.
- **error**- if the communication with the destination cannot be established through an OFED supported device, but communication can be established by using the InfiniBand architecture.

```
/*The following check_ofed_verbs_support routine does:
/*- Call rdma_create_event_channel to open a channel event     */
/*- Calls rdma_create_id() to get a cm_id                        */
/*- And then calls rdma_resolve_addr()             */
/*- Get the communication event                 */
/*- Returns the event status:                  */
/* 0: OK                               */
/* error: NOK  output device may be not a RNIC device     */
/*- Calls rdma_destroy_id() to delete the cm_id created     */
/*- Call rdma_destroy_event_channel  to close a channel event  */

 int check_ofed_verbs_support (struct sockaddr *remoteaddr)
   {
    struct rdma_event channel *cm_channel;
    struct rdma_cm_id *cm_id;
    int ret=0;
    cm_channel = rdma_create_event_channel();
    if (!cm_channel)  {
      fprintf(stderr,"rdma_create_event_channel error\n");
      return -1;
      }
      ret = rdma_create_id(cm_channel, &cm_id, NULL, RDMA_PS_TCP);
      if (ret) {
        fprintf(stderr,"rdma_create_id: %d\n", ret);
        rdma_destroy_event_channel(cm_channel);
        return(ret);
      }
      ret = rdma_resolve_addr(cm_id, NULL, remoteaddr, RESOLVE_TIMEOUT_MS);
      if (ret) {
        fprintf(stderr,"rdma_resolve_addr: %d\n", ret);
        goto out;
      }
      ret = rdma_get_cm_event(cm_channel, &event);
      if (ret) {
        fprintf(stderr," rdma_get_cm_event() failed\n");
        goto out;
      }
      ret = event->status;
      rdma_ack_cm_event(event);
         out:
      rdma_destroy_id(cm_id);
      rdma_destroy_event_channel(cm_channel);
      return(ret);
   }
```

Libibverbs library

The **Libibverbs** library enables user-space processes to use Remote Direct Memory Access (RDMA) verbs.

The **Libibverbs** library is described in the InfiniBand architecture specification and the RDMA protocol verbs specification.

Several `/dev/rdma/uverbsN` character device nodes are used to handle communication between the **Libibverbs** library and the `ib_uverbs` kernel layer. Every RDMA network interface controller (RNIC) adapter has one device that is registered with the Open Fabrics Enterprise Distribution (OFED) core such as the uverbs1 and uverbs2 devices. To run on the appropriate device, the library writes commands corresponding to the verb.

Related information:

InfiniBand

RDMA protocol verbs

Librdmacm library

The **librdmacm** library provides the communication manager (CM) function and a generic set of Remote Direct Memory Access (RDMA) CM interfaces that runs on different fabrics such as, InfiniBand (IB), RDMA over Converged Ethernet (RoCE), or Internet Wide Area RDMA Protocol (iWARP).

A single `/dev/rdma/rdma_cm` device node is used by the user space to communicate with the kernel, regardless of the number of adapters or ports that are present.

The **librdmacm** library is used by applications that must be run on any RDMA device.

RDMA network interface controller (RNIC)

A network I/O adapter or embedded controller with Internet Wide Area RDMA Protocol (iWARP), and Verbs function.

RDMA_CM communication manager

The Remote Direct Memory Access communication manager (RDMA_CM) is used to set up reliable connection for transferring data.

The communication manager provides an RDMA transport neutral interface for establishing connections. The API is based on sockets, but is adapted for queue pair (QP) based semantics. The communication is over a specific RDMA device, and data transfers are message-based.

The RDMA CM uses the **librdmacm** library to provide the communication management to set up and teardown the connection of an RDMA API. The communication manager works with the verbs API by using the **libibverbs** library for data transfers.

Resource managed by using OFED verbs

Lists the resources that are managed by using the OFED verbs.

Completion Queue (CQ):
A first-in-first-out (FIFO) queue that contains Completion queues (CQ). The CQ is associated with a queue pair, which are used to receive completion notifications and events.

Completion Queue Entry (CQE):
An entry in the CQ that describes the information about the completed Work request (WR) such as the status and size.

Event Channel:
Used to report communication events. Each event channel is mapped to a file descriptor. The associated file descriptor can be used and manipulated like any other file descriptor to change its behavior. You can make the file descriptor perform one of the following actions:

- non-block the file descriptor
- poll the file descriptor
- select the file descriptor

Memory Region (MR):
A set of memory buffers that are registered with the access permissions. Tho use the memory buffers with the network adapters, the memory regions must be registered.

Protection Domain (PD):
Enables a client to associate multiple resources, such as queue pairs and memory regions, within a domain. The client then grants access rights to send or receive data within the protection domain to other domains that are located on the RDMA fabric.

Queue Pair (QP):
Queue pairs (QPs) contain a send and a receive queue. The send queue sends outbound messages requesting for the RDMA operations. The receive queue receives incoming messages or immediate data.

Scatter or Gather Elements (SGE):
An entry to a pointer to a full or a part of a local registered memory block. The element holds the start address of the block, size, and the lkey with the associated permissions.

Scatter or Gather Array:
An array of scatter or gather elements that exists in a work request (WR). The array works according to the operation code that either collects data from multiple buffers and sends them as a single stream or takes a single stream and separates the data to numerous buffers.

Work Queue (WQ):
A work queue consists of the Send Queue or the Receive Queue. The work queue is used to send or receive messages.

Work Queue Element (WQE):
Work Queue Element is an element in a work queue.

Work Request (WR):
Work Request is a request that is posted by a user to a work queue.

Communication operations

Lists the communication operations that are available for an RDMA device.

Send and send with immediate operation:

The send operation sends data to the receive queue of a remote Queue Pair (QP).

To receive the data, the receiver must post data into a receive buffer. The sender does not have any control over the data that is in the remote host.

An immediate 4-byte value is transmitted with the data buffer. This immediate value is presented to the receiver as part of the receive notification, and it is not contained in the data buffer.

Receive operation:

The receive operation is the corresponding operation to a send operation.

The receiving host is notified that a data buffer is received with an inline immediate value. The receiving application maintains the receive buffer and posts information.

RDMA read operation:

The `RDMA read` operation reads a memory region from the remote host.

You must specify the remote virtual address and a local memory address where the read information is copied. Before you run the Remote Direct Memory Access (RDMA) operations, the remote host must provide appropriate permissions to access its memory. After the permissions are set, the RDMA read operations are run without any notification to the remote host.

Atomic operation:

Atomic operation is not supported by the Remote Direct Memory Access (RDMA) hardware available for the AIX operating system.

RDMA write or RDMA write with immediate operation:

The RDMA write operation is similar to the RDMA read operation, but the data is written to the remote host.

The RDMA write operations are run with no notification to the remote host. RDMA write with immediate operations do notify the remote host about the immediate value.

Transport modes

The transport modes establish a connection for the queue pair.

The following transport modes are supported

- Reliable connection (RC)
  - Each queue pair (QP) is associated with another QP
  - Messages that are transmitted by the send queue of one QP are reliably delivered to the receive queue of another QP.
  - Packets are delivered in order.
  - An RC is similar to a TCP connection.
- Unreliable datagram (UD)
  - No actual connection is formed between the QPs.
  - The UD mode is similar to a UDP connection.

Planning for Open Fabrics enterprise Distribution (OFED)

A configuration file must exist in the `/etc/libibverbs.d/` directory for every Remote Direct Memory Access (RDMA) adapter that is installed on the system.

The configuration file enables the **libibverbs** library to use the driver for the RDMA devices. For example, to use a **Mellanox ConnectX-2 RoCE** adapter, the `mx2.driver` file must exist in the `/etc/libibverbs.d/` directory. The `mx2.driver` file must contain the following code:

```
# cat /etc/libibverbs.d/mx2.driver
driver mx2
```

To use any other directory, except the `/etc/libibverbs.d/` directory, use the `IBV_CONFIG_DIR` environment variable. To establish communication between the two nodes, the adapters must have IPv4 or IPv6 addresses configured.

Creating connections by using the communication manager (RDMA_CM)

The Remote Direct Memory Access (RDMA) `RDMA_CM` communication manager provides the communication management that includes connection setup and tear down for an RDMA application programming interface (API).

The `RDMA_CM` communication manger works with the verbs API defined by the **libibverbs** library. The **libibverbs** library provides the interfaces that are required to send and receive data.

Client Operation

Learn about the overview of the basic operation for the active or client communication.

A general connection flow follows:

rdma_create_event_channel
: Creates a channel to receive events.

rdma_create_id
Allocates an `rdma_cm_id` identifier that is conceptually similar to a socket.

rdma_resolve_addr
Obtains a local Remote Direct memory Access (RDMA) device to reach the remote address.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_ADDR_RESOLVED` event.

rdma_ack_cm_event
Acknowledges the received event.

rdma_create_qp
Allocates a queue pair (QP) for the communication.

rdma_resolve_route
Determines the route to the remote address.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_ROUTE_RESOLVED` event.

rdma_ack_cm_event
Acknowledges the received event.

rdma_connect
Connects to the remote server.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_ESTABLISHED` event.

rdma_ack_cm_event
Acknowledges the received event.

ibv_post_send()
Performs data transfer over the connection.

rdma_disconnect
Tears down the connection.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_DISCONNECTED` event.

rdma_ack_cm_event
Acknowledges the event.

rdma_destroy_qp
Destroys the QP.

rdma_destroy_id
Releases the `rdma_cm_id` identifier.

rdma_destroy_event_channel
Releases the event channel.

Note: In the example, the client initiated the disconnect. However, either the client or server operation can initiate the disconnect process.

Server operation

Learn about the basic operation that can be run for the passive or server communication.

A general connection flow follows:

rdma_create_event_channel
Creates a channel to receive events.

rdma_create_id
Allocates an `rdma_cm_id` identifier that is conceptually similar to a socket.

rdma_bind_addr
Sets the local port number on which the event listens.

rdma_listen
Starts listening to the connection requests.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_CONNECT_REQUEST` event with a new `rdma_cm_id` identifier.

rdma_create_qp
Allocates a queue pair (QP) for the communication on the new `rdma_cm_id` identifier.

rdma_accept
Accepts the connection request.

rdma_ack_cm_event
Acknowledges the event.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_ESTABLISHED` event.

rdma_ack_cm_event
Acknowledges the event.

ibv_post_send()
Performs the data transfer over the connection.

rdma_get_cm_event
Waits for the `RDMA_CM_EVENT_DISCONNECTED` event.

rdma_ack_cm_event
Acknowledges the event.

rdma_disconnect
Tears down the connection.

rdma_destroy_qp
Destroys the QP.

rdma_destroy_id
Releases the connected **rdma_cm_id** identifier.

rdma_destroy_id
Releases the listening `rdma_cm_id` identifier.

rdma_destroy_event_channel
Releases the event channel.

RDMA_CM communication manager examples

Learn about an example that was presented to the Open Fabrics Enterprise Distribution (OFED) community during the LinuxConf.Europe 2007 conference.

Related information:

Example presented to the OFED community

Example of an active client

An example of the communication operation where the client is active.

```
/*
 * build:
 * cc -o client client.c -lrdmacm -libverbs
 *
```

```
 * usage:
 * client <servername> <val1> <val2>
 *
 * connects to server, sends val1 via RDMA write and val2 via send,
 * and receives val1+val2 back from the server.
 */
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

#include <rdma/rdma cma.h>
enum   {
        RESOLVE TIMEOUT MS        = 5000,
};
struct pdata {
        uint64_t buf va;
        uint32_t buf rkey;
};

int main(int argc, char   *argv[ ])
{
   struct pdata         *server pdata;
   struct rdma_event channel  *cm_channel;
   struct rdma_cm_id      *cm_id;
   struct rdma_cm_event     *event;
   struct rdma_conn_param    conn_param = { };
   struct ibv_pd        *pd;
   struct ibv_comp_channel   *comp_chan;
   struct ibv_cq        *cq;
   struct ibv_cq        *evt_cq;
   struct ibv_mr        *mr;
   struct ibv_qp_init_attr   qp attr = { };
   struct ibv_sge        sge;
   struct ibv_send_wr      send_wr = { };
   struct ibv_send_wr      *bad send wr;
   struct ibv_recv_wr      recv wr = { };
   struct ibv_recv_wr      *bad recv wr;
   struct ibv_wc        wc;
   void             *cq context;
   struct addrinfo       *res, *t;
   struct addrinfo       hints = { .ai_family    = AF INET,
                           .ai_socktype  = SOCK STREAM
                                       };
 int              n;
 uint32_t          *buf;
 int              err;

       /* Set up RDMA CM structures */
   cm_channel = rdma_create_event_channel();
   if (!cm_channel)  return 1;
   err = rdma_create_id(cm_channel, &cm_id, NULL, RDMA_PS_TCP);
   if (err)
     return err;
   n = getaddrinfo(argv[1], "20079", &hints, &res);
   if (n < 0)
     return   1;

   /* Resolve server address and route */
   for (t = res; t; t = t->ai next) {
     err = rdma_resolve_addr(cm_id, NULL, t->ai_addr, RESOLVE_TIMEOUT_MS);
     if (!err)
       break;
```

```
}
if (err)
  return err;
err = rdma_get_cm_event(cm_channel, &event);
if (err)
  return err;
if (event->event != RDMA_CM_EVENT_ADDR_RESOLVED)
  return 1;
rdma_ack_cm_event(event);
err = rdma_resolve_route(cm_id, RESOLVE_TIMEOUT_MS);
if (err)
  return err;
err = rdma_get_cm_event(cm_channel, &event);
if (err)
  return err;
if (event->event != RDMA_CM_EVENT_ROUTE_RESOLVED)
  return 1;
rdma_ack_cm_event(event);

/* Create verbs objects now that we know which device to use */
pd = ibv_alloc_pd(cm_id->verbs);
if (!pd)
  return 1;
comp chan = ibv_create_comp_channel(cm_id->verbs);
if (!comp_chan)
  return 1;
cq = ibv_create_cq(cm_id->verbs, 2,NULL, comp_chan, 0);
if (!cq)
  return 1;
 if (ibv_req_notify_cq(cq,  0))
  return 1;
 buf = calloc(2, sizeof (uint32_t));
if (!buf)
  return 1;
mr = ibv_reg_mr(pd, buf,2 * sizeof(uint32_t), IBV_ACCESS_LOCAL_ WRITE);
if (!mr)
  return 1;
qp_attr.cap.max        send_wr = 2;
qp_attr.cap.max        send_sge = 1;
qp_attr.cap.max        recv_wr = 1;
qp_attr.cap.max        recv_sge = 1;
qp_attr.send_cq        = cq;
qp_attr.recv_cq        = cq;
qp_attr.qp_type        = IBV_QPT_RC;
err = rdma_create_qp(cm_id, pd, &qp_attr);
if (err)
  return err;
conn_param.initiator_depth = 1;
conn_param.retry_count     = 7;

/* Connect to server */
err = rdma_connect(cm_id, &conn_param);
if (err)
  return err;
err = rdma_get_cm_event(cm_channel,&event);
if (err)
  return err;
if (event->event != RDMA_CM_EVENT_ESTABLISHED)
  return 1;
memcpy(&server_pdata, event->param.conn.private_data, sizeof server_pdata);
rdma_ack_cm_event(event);

/* Prepost receive */
sge.addr    =       (uintptr_t) buf;
sge.length  =       sizeof (uint32_t);
sge.lkey    =       mr->lkey;
recv_wr.wr_id =     0;
```

```
recv_wr.sg_list =   &sge;
recv_wr.num_sge =   1;

if (ibv_post_recv(cm_id->qp, &recv_wr, &bad_recv_wr))
  return 1;

/* Write/send two integers to be added */
buf[0] = strtoul(argv[2], NULL, 0);
buf[1] = strtoul(argv[3], NULL, 0);
printf("%d + %d = ", buf[0], buf[1]);
buf[0] = htonl(buf[0]);
buf[1] = htonl(buf[1]);

sge.addr                      = (uintptr_t) buf;
sge.length                    = sizeof (uint32_t);
sge.lkey                      =  mr->lkey;
send_wr.wr_id                 = 1;
sendwr.opcode                 = IBV_WR_RDMA_WRITE;
send _wr.sg_list              = &sge;
send_wr.num_sge               = 1;
send_wr.wr.rdma.rkey          = ntohl(server_pdata.buf_rkey);
send_wr.wr.rdma.remote_addr   = ntohll(server_pdata.buf_va);

  if (ibv post send(cm id->qp, &send_wr,  &bad_send_wr))
return 1;
sge.addr                      = (uintptr_t) buf + sizeof (uint32_t);
sge.length                    = sizeof (uint32_t);
sge.lkey                      = mr->lkey;
send_wr.wr_id                 = 2;
send_wr.opcode                = IBV_WR_SEND;
send_wr.send_flags            = IBV_SEND_SIGNALED;
send_wr.sg_list               =&sge;
send_wr.num_sge               = 1;

if (ibv_post_send(cm_id->qp, &send_wr,&bad_send_wr))
 return 1;

 /* Wait for receive completion */
 while (1) {
    if (ibv_get_cq_event(comp_chan,&evt_cq, &cq_context))
   return 1;
 if (ibv_req_notify_cq(cq, 0))
   return 1;
 if (ibv_poll_cq(cq, 1, &wc) != 1)
   return 1;
 if (wc.status != IBV_WC_SUCCESS)
   return 1;
 if (wc.wr_id == 0) {
   printf("%d\n", ntohl(buf[0]));
   return 0;
 }
  }
  return 0;
}
```

OFED commands

Learn about the Open Fabrics Enterprise Distribution (OFED) commands, including syntax statements, descriptions of flags, and usage examples.

ibv_devices command

Lists the Remote Direct Memory Access (RDMA) devices available for use from the user space.

ibv_devinfo command

Prints information about RDMA network interface controller (RNIC) devices available for use from user space.

Syntax

```
ibv_devinfo [-v] { [-d <dev>] [-i <port>] } | [-l]
```

Flags

Item	Description
-d *dev*	Uses the *dev* RDMA device. By default, the first device that is found is used.
-i *port*	Uses the *port* port of RDMA device. By default, all ports are used.
-l	Prints only the RDMA devices name.
-v	Prints all the attributes of the RDMA devices.

ofedctrl command

Loads and unloads the **ofed_core** kernel extension.

Syntax

ofedctrl { [-k *KernextName*] -l|u|q } | [-c | -p *ParameterName=Value*] | -h

Flags

Item	Description
-c	Reloads the configuration file if the file was edited.
-h	Specifies the usage.
-k *KernextName*	Specifies the kernel extension path. By default, the /usr/lib/drivers/ofed_core path is used.
-l	Loads the kernel extension.
-p *ParameterName=Value*	Sets the value of a parameter directly on the command line. **Note:** The values that are set by using the -p option are not persistent. The -p option changes only the current configuration. It does not update the configuration file. Changes that are made by using the -p option does not apply after the system is restarted.
-q	Indicates whether the kernel extension is loaded or not.
-u	Unloads the kernel extension.

rping command

Tests the connection of the RDMA communication manager (RDMA_CM) by using the RDMA ping-pong test.

Syntax

```
rping -s [-v] [-V] [-d] [-P] [-a address] [-p port] [-C message_count] [-S message_size]
rping -c [-v] [-V] [-d] -a address [-p port] [-C message_count] [-S message_size]
```

Description

The **rping** command establishes a reliable Remote Direct Memory Access (RDMA) connection between two nodes by using the **librdmacm** library. Optionally, the **rping** command also performs RDMA transfers between the nodes, and then disconnects the connection. The **rping** command sets an RDMA_CM connection and performs an RDMA ping-pong test. For information on the **rping** command, see the Open Source OpenFabrics Alliance OFED 1.4 at http://www.openfabrics.org .

Flags

Item	Description
-a *address*	Specifies the network address to bind the connection on the server and specifies the server address to connect to the client.
-c	Runs as the client.
-C *message_count*	Specifies the number of messages to transfer over each connection. The default value is infinite.
-d	Displays the debug information.
-p	Specifies the port number for the listening server.
-P	Runs the server in persistent mode. This allows multiple **rping** clients to connect to a single server instance and the server runs until the instance is killed.
-v	Displays the ping data.
-V	Validates the ping data.
-s	Runs as the server.
-S *message_size*	Specifies the size of each message transferred, in bytes. The default value is 100.

Related information:

Openfabrics

User-level Direct Access Programming Library (uDAPL)

User Direct Access Programming Library (uDAPL) is a direct-access framework to be run on transports that support direct data access, such as InfiniBand and RDMA network interface controller (RNIC).

The DAT Collaborative specifies the uDAPL application programming interface (API). The uDAPL codebase from Open Fabrics is ported to AIX operating system and is supported over GX++ HCA and 4X DDR Expansion card (CFFh) InfiniBand adapters.

Related concepts:

"Supported uDAPL APIs in the AIX operating system" on page 13
The User Direct Access Programming Library (uDAPL) APIs that are specified by the DAT Collaborative are all not supported by theAIX operating system.

"Vendor-specific attributes for uDAPL" on page 14
Learn about the vendor-specific attributes that are supported by the AIX operating system. The `delayed_ack_supported`, `vendor_extension`, `vendor_ext_version`, `debug_query`, and `debug_modify` attributes are supported.

Related information:

Datcollaborative

Installing uDAPL

The User-level Direct Access Programming Library (uDAPL) version 2.0 is supported by the AIX operating system.

The uDAPL installation image is shipped on the expansion pack as **udapl.rte**. This image ships the DAT header files, which are located in the `/usr/include/dat` directory. The installation image also ships the **libdat.a** and **libdapl.a** libraries.

Applications include the DAT header files and link with the **libdat.a** DAT library in the `/usr/include/dat` directory. The DAT layer determines the appropriate underlying transport-specific libraries.

An AIX uDAPL provider registers itself with the DAT registry by using the `dat.conf` file entries. The `/etc/dat.conf` file is shipped with default entries and the file has details on the format of the entry.

The uDAPL libraries support the AIX system trace for debugging events. The uDAPL system trace connects ID that includes 5C3 (for DAPL events), 5C4 (for DAPL error events), 5C7 (for DAT events), and 5C8 (for DAT error events). The initial trace level is modified by using the `DAT_TRACE_LEVEL` and

DAPL_TRACE_LEVELenvironment variables. These environmental variables accept values in the 0 - 10 range. The number of events and amount of data traced increases with the key trace levels as follows:

```
TRC_LVL_ERROR  = 1
TRC_LVL_NORMAL = 3
TRC_LVL_DETAIL = 7
```

Other standard AIX serviceability features, such as the AIX error log are used to identify problems when tracing an event. The serviceability features of the underlying transport layer, such as the **ibstat** command and InfiniBand component trace, are also helpful for analyzing the issues.

The DAT APIs return the standard return codes that can be decoded by using the `/usr/include/dat/dat_error.h` file. The detailed explanation about the return codes is provided in the uDAPL specification from the DAT Collaborative.

Supported uDAPL APIs in the AIX operating system

The User Direct Access Programming Library (uDAPL) APIs that are specified by the DAT Collaborative are all not supported by theAIX operating system.

The following APIs are supported by the common industry uDAPL implementations, and that are supported by the AIX operating system.

The following APIs are not supported by the common industry uDAPL implementations, and is not supported by the AIX operating system.

API	Version
dat_cr_handoff	// In DAT 2.0
dat_ep_create_with_srq	// In DAT 2.0
dat_ep_recv_query	// In DAT 2.0
dat_ep_set_watermark	// In DAT 2.0
dat_srq_create	// In DAT 2.0
dat_srq_post_recv	// In DAT 2.0
dat_srq_resize	// In DAT 2.0
dat_srq_set_lw	// In DAT 2.0
dat_srq_free	// In DAT 2.0
dat_srq_query	// In DAT 2.0

The following additional APIs that AIX operating system does not support:

- `dat_lmr_sync_rdma_read`
- `dat_lmr_sync_rdma_write`
- `dat_registry_add_provider`
- `dat_registry_add_provider`

For all unsupported APIs, the AIX operating system follows the specific mechanisms that are described in the DAT specification to identify the unsupported API list. These include `max_srq` attribute values that are zero and specific `DAT_MODEL_NOT_SUPPORTED` return codes. According to the industry implementation and the DAT specification, `DAT_NOT_IMPLEMENTED` code can be returned for a function, which is not supported.

Support of remote memory region (RMR)-related APIs such as *dat_rmr_create*, *dat_rmr_bind*, *dat_rmr_free*, and *dat_rmr_query* is dependent on the underlying host channel adapter (HCA) capability, and success or failure is determined by the underlying InfiniBand framework. Currently the GX++ HCA and 4X DDR Expansion card (CFFh) InfiniBand adapters do not support the RMR operations.

Related concepts:

"User-level Direct Access Programming Library (uDAPL)" on page 12
User Direct Access Programming Library (uDAPL) is a direct-access framework to be run on transports

that support direct data access, such as InfiniBand and RDMA network interface controller (RNIC).

"Vendor-specific attributes for uDAPL"
Learn about the vendor-specific attributes that are supported by the AIX operating system. The `delayed_ack_supported`, `vendor_extension`, `vendor_ext_version`, `debug_query`, and `debug_modify` attributes are supported.

Vendor-specific attributes for uDAPL

Learn about the vendor-specific attributes that are supported by the AIX operating system. The `delayed_ack_supported`, `vendor_extension`, `vendor_ext_version`, `debug_query`, and `debug_modify` attributes are supported.

The AIX operating system is a transport provider for the InfiniBand (IB) framework, which includes a vendor-specific interface adapter (IA®) and the `delayed_ack_supported` attribute. The value of the `delayed_ack_supported` attribute is either **true** or **false**. When the value is **true**, the endpoints that are associated with the IA have a modifiable provider-specific `delayed_ack` attribute. When the `delayed_ack_supported` attribute is **false**, the endpoints of the provider-specific`delayed_ack` attribute cannot be changed. The default value of an endpoint of the provider-specific `delayed_ack` attribute is **false**. The `delayed_ack` attribute is set to **true** by using the `dat_ep_modify` option that enables the delayed acknowledge feature of the underlying InfiniBand (IB) host channel adapter (HCA) for the specific InfiniBand queue pair that is associated with the endpoint. This hardware feature is not implemented by all HCAs, thus it is not available for all IAs. When this feature is enabled, the acknowledgement sent by HCA is delayed until a data transfer operation is detected in the system memory of a server. This process causes a small latency increase.

For debugging errors, the uDAPL libraries support AIX system trace. The initial trace level can be changed by using the `DAT_TRACE_LEVEL` and `DAPL_TRACE_LEVEL` environment variables. To change these trace levels dynamically by using an API, use the dynamic trace level support on AIX. To verify whether the library has dynamic trace level support, applications can query for the vendor-specific IA `vendor_extension` attribute. The presence of the `vendor_extension` attribute indicates the dynamic trace level that is supported. When the `vendor_extension` attribute is present, applications can access the `dat_trclvl_query()` and `dat_trclvl_modify()` function pointers by querying for the`debug_query` and `debug_modify` vendor-specific IA attributes. The value of these attributes points to the corresponding functions. To make this `vendor_extension` interface available for future, the `vendor_extension` vendor-specific IA attribute must be used. Currently, the `vendor_extension` attribute is set to 1.0 and it is the only version that is supported. If the `vendor_extension` attribute does not exist, applications cannot modify the trace levels dynamically.

An example of how to change these attributes is included in the uDAPL sample code that is installed with the AIX implementation.

Related concepts:

"Supported uDAPL APIs in the AIX operating system" on page 13
The User Direct Access Programming Library (uDAPL) APIs that are specified by the DAT Collaborative are all not supported by theAIX operating system.

"User-level Direct Access Programming Library (uDAPL)" on page 12
User Direct Access Programming Library (uDAPL) is a direct-access framework to be run on transports that support direct data access, such as InfiniBand and RDMA network interface controller (RNIC).

Notices

This information was developed for products and services offered in the U.S.A. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 2013, 2014

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM® prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

OpenFabrics.org BSD License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as the customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

INFINIBAND, InfiniBand Trade Association, and the INFINIBAND design marks are trademarks and/or service marks of the INFINIBAND Trade Association.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Index

C

L

O

R

T

U

V

© Copyright IBM Corp. 2013, 2014

Printed in USA