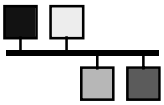
EPICS

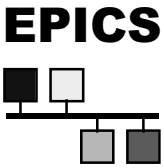


Writing Channel Access Clients

Andy Foster

Observatory Sciences Limited

Outline



- EPICS Communication Protocol
  - Client/Server Architecture
- Server is part of iocCore
- Basic Principles of Client Operation
- Clients are written using the CA Client Library – Basic Functions:
  - Initialize CA
  - Search
  - Put
  - Get
  - Monitors
  - Flush Send Buffer
  - Error reporting
- Explain 6 example clients
- See EPICS 3.12 Channel Access Reference Manual for full details of functions and macros (not changed for 3.13)
- Channel Access Performance Figures

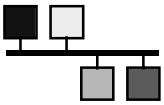
Basic Principles of Client Operation

EPICS

- CA library calls are buffered until either:
  - send buffer is full
  - a special CA library call is made
  - allows efficient operation over a network
- Error codes returned from CA library calls only check the validity of the request NOT the success of the operation.
- When the server detects an error, an exception handler is executed in the client. Default is to print a message to the screen.

Initialize

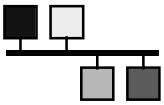
EPICS



- ca_task_initialize()
- Should be called prior to making any other CA calls.
- If you forget, it will be called from the first call to any ca_xxx function.
- Registers this client with the CA repeater.
  - This is a daemon (under Unix) which fans out UDP broadcasts received on the CA UDP port to all CA processes running on the machine. Solves the problem of the O/S only allowing one process to have access to the port.

Search

EPICS



- ca_search(“name”, &channelId)
  - Causes a UDP broadcast to be sent to each IOC on the subnet.
  - A TCP connection is then established with the IOC that contains this record.
  - Using the channel identifier, a client can retrieve lots of useful information.

Put

EPICS

- There are many CA PUT functions:
  - ca_put(field_type, channelId, &val)
  - ca_put_callback

  (field_type, channelId, &val, usrFunc, usrArg)

- “field_type” is one of DBR_DOUBLE etc. See: base/include/db_access.h.

- Conversion between types are done in the server so ask for native types for better performance at the real-time end!

- All use “channel ID” from Search to communicate with record field

- usrFunc is called after all records in the processing chain in the database have been processed.

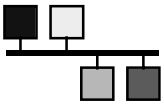
Get

EPICS

- There are many CA GET functions:
  - ca_get(field_type, channelId, &val)
  - ca_get_callback (field_type, channelId, usrFunc, usrArg)
- Arguments similar to the PUT functions.
- Notice that ca_get_callback does not get the value directly. It is available to the user defined function (see example).

Monitors

EPICS



- How can we write a client which will react to monitors raised in the database?
- Use:

  ca_add_event(field_type, channelId,
  usrFunc, usrArg, eventId
)
- usrFunc will be called when the channel's value changes by more than the dead band or when the channel's alarm state changes.
- eventId is a pointer to a user defined event ID. Often omitted by setting as NULL.

Flushing the Send Buffer

EPICS

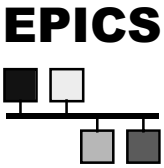
- Vital for the CA client to work!!
- All CA client calls are buffered until either the send buffer is full or one of these is explicitly called:
  - ca_pend_io(timeout)

  Flushes the send buffer and waits until the shorter of:
    - All outstanding calls to ca_get complete
    - The timeout expires

    (Note a timeout of 0 means wait forever)
  - ca_pend_event(timeout)

  Flushes the send buffer and always waits "timeout" seconds for asynchronous events (timeout of 0 means wait forever)

Error Reporting



- For portability reasons CA functions do not return status following the UNIX convention.
- Do not test return status against “0”.
- Useful macro for testing return status:

SEVCHK(status, “user string”);

but CANNOT do: “if(SEVCHK())”

So in the following examples, I have defined this:

```
#define MY_SEVCHK(status)                \
{                                        \
    if( status != ECA_NORMAL )           \
    {                                    \
        SEVCHK(status, NULL);            \
        exit(status);                    \
    }                                    \
}
```

Example Client caPut1.c

EPICS

```
#include <stdio.h>
#include <stdlib.h>
#include <cadef.h>     /* Structures and data types used by CA */

int main( int argc, char *argv[] )
{
  int        status;
  chid       channelId;
  double val;

   val = atof( argv[1] );

   status = ca_task_initialize();
   MY_SEVCHK(status);

   status = ca_search( “test:ai”, &channelId );
   MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  status = ca_put( DBR_DOUBLE, channelId, &val );
  MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  return(0);
}
```

Example Client caGet1.c

EPICS

```
#include <stdio.h>
#include <stdlib.h>
#include <cadef.h>

int main( int argc, char *argv[] )
{
  int        status;
  chid       channelId;
  double val;

   status = ca_task_initialize();
   MY_SEVCHK(status);

   status = ca_search( “test:ai”, &channelId );
   MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  status = ca_get( DBR_DOUBLE, channelId, &val );
  MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  printf(“Value is %f\n”, val);
  return(0);
}
```

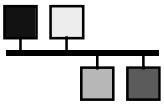
Example Client caPut2.c (main)

EPICS

```
int main( int argc, char *argv[] )
{
  int         status;
  chid        channelId;
  double val;
  info_t   info;

  info.project  = “Diamond”;
  info.numIoc = 167;
  info.current  = 3.45;
   val                = atof( argv[1] );

   status = ca_task_initialize();
   MY_SEVCHK(status);

   status = ca_search( “test:ai”, &channelId );
   MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  status = ca_put_callback( DBR_DOUBLE, channelId, &val,
                                    (void (*)())usrFunc, &info );
  MY_SEVCHK(status);

  status = ca_pend_event(0.1);   /* This is different! */
  MY_SEVCHK(status);
  return(0);
}
```

Example Client caPut2.c (the rest)

EPICS



```
#include <stdio.h>
#include <stdlib.h>
#include <cadef.h>

typedef struct info
{
  char     *project;
  int         numIoc;
  double  current;
} info_t;

void usrFunc( struct event_handler_args args )    /* cadef.h */
{
  info_t *t = (info_t *)args.usr;

  printf("UsrFunc called:\n");
  printf("User Argument Name     = %s\n",  t->project);
  printf("User Argument NumIoc = %d\n", t->numIoc);
  printf("User Argument Current  = %f\n",  t->current);
  printf("Channel Name                = %s\n", ca_name(args.chid) );
  printf("Number of Elements        = %d\n",
     ca_element_count(args.chid) );
   printf("Host Name                     = %s\n",
     ca_host_name(args.chid) );
}
```

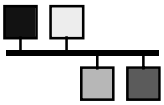
Example Client caGet2.c (main)

EPICS

```
int main( int argc, char *argv[] )
{
  int         status;
  chid        channelId;
  info_t    info;

  info.project  = “Diamond”;
  info.numIoc = 167;
  info.current  = 3.45;

   status = ca_task_initialize();
   MY_SEVCHK(status);

   status = ca_search( “test:ai”, &channelId );
   MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  status = ca_get_callback( DBR_DOUBLE, channelId,
                                  (void (*)())usrFunc, &info );
  MY_SEVCHK(status);

  status = ca_pend_event(0.1);   /* This is different! */
  MY_SEVCHK(status);
  return(0);
}
```

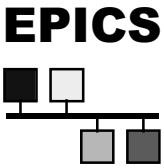
Example Client caGet2.c (the rest)

EPICS



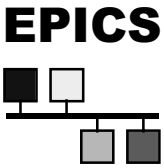
```
#include <stdio.h>
#include <stdlib.h>
#include <cadef.h>

typedef struct info
{
  char     *project;
  int       numIoc;
  double  current;
} info_t;

void usrFunc( struct event_handler_args args )    /* cadef.h */
{
  info_t *t = (info_t *)args.usr;

  printf("UsrFunc called: Value      = %f\n",  *(double *)args.dbr);
  printf("User Argument Name     = %s\n",  t->project);
  printf("User Argument NumIoc = %d\n", t->numIoc);
  printf("User Argument Current  = %f\n",  t->current);
  printf("Channel Name                = %s\n", ca_name(args.chid) );
  printf("Number of Elements        = %d\n",
     ca_element_count(args.chid) );
   printf("Host Name                     = %s\n",
     ca_host_name(args.chid) );
}
```

Example Client caMonitor.c (main)




```
int main( int argc, char *argv[] )
{
  int        status;
  chid       channelId;

   status = ca_task_initialize();
   MY_SEVCHK(status);

  /* Note new type of search call */
   status = ca_search_and_connect( “test:ai”, &channelId,
                                          (void (*)())connectFunc, NULL );
   MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  status = ca_add_event( DBR_DOUBLE, channelId,
                                 (void (*)())monitorFunc, NULL, NULL );
  MY_SEVCHK(status);

  status = ca_pend_event(0.0);   /* Wait forever */
  MY_SEVCHK(status);
  return(0);
}
```

Example Client caMonitor.c (the rest)




```
#include <stdio.h>
#include <stdlib.h>
#include <cadef.h>

void connectFunc( struct connection_handler_args args )
{
  if( ca_state(args.chid) != cs_conn )
    printf("%s" has just Disconnected\n",
     ca_name(args.chid));
  else
    printf("%s" has just Connected\n", ca_name(args.chid));
}

void monitorFunc( struct event_handler_args args )
{
  printf("Monitor on %s, new value = %f\n",
            ca_name(args.chid),  *(double *)args.dbr);
}
```

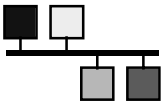
Composite Data Structures

EPICS

- Requests can be made for data related to the value field e.g.
  - Alarm Status
  - Alarm Severity
  - Precision
  - Engineering units
  - Time Stamp
  - Display Limits
  - Alarm Limits
- Many fields are fetched from the database in one access

Example Client caGet3.c (main)

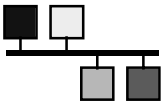
EPICS



```
int main( int argc, char *argv[] )
{
  int                              status;
  chid                             channelId;
  struct dbr_ctrl_double    data;
  struct dbr_time_double tdata;
  struct  tsDetail                T;

    status = ca_task_initialize();
   MY_SEVCHK(status);

   status = ca_search( “test:ai”, &channelId );
   MY_SEVCHK(status);

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  status = ca_get( DBR_CTRL_DOUBLE, channelId, &data );
  MY_SEVCHK(status);

  status = ca_get( DBR_TIME_DOUBLE,  channelId, &tdata );

  status = ca_pend_io(0.0);
  MY_SEVCHK(status);

  tsStampToLocal( tdata.stamp, &T );
  printResults( data, T );
  return(0);
}
```

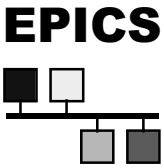
Example Client caGet3.c (the rest)

EPICS



```
#include <stdio.h>
#include <stdlib.h>
#include <cadef.h>

void printResults( struct dbr_ctrl_double data,  struct tsDetail T )
{
  printf("Channel Value           = %f\n",  data.value);
  printf("Alarm Status            = %d\n",  data.status);      /* see alarm.h */
  printf("Alarm Severity          = %d\n",  data.severity);  /* see alarm.h */
  printf("Precision                 = %d\n",  data.precision);
  printf("Engineering Units      = %s\n",  data.units);
  printf("Upper Display Limit  = %d\n",
                                        data.upper_disp_limit);
  printf("Lower Display Limit  = %d\n",
                                        data.lower_disp_limit);
  printf("Upper Alarm Limit  = %d\n",
                                        data.upper_alarm_limit);
  printf("Lower Alarm Limit  = %d\n",
                                        data.lower_alarm_limit);
  printf("Upper Warning Limit  = %d\n",
                                        data.upper_warning_limit);
  printf("Lower Warning Limit  = %d\n",
                                        data.lower_warning_limit);
  printf("Last Processed on: %d/%d/%d %d:%d:%d\n",
            T.dayMonth+1, T.monthNum+1, T.year,
            T.hours, T.minutes, T.seconds);
}
```

Channel Access Performance Figures



- External factors should be taken into account such as the network loading during the test

	PowerPC	68040
ca_search	1050/sec	840/sec
ca_put	43,500/sec	9200/sec
ca_get	17,300/sec	8500/sec

- Based on measurements at the KEK Accelerator. See:

www.aps.anl.gov/icalepcs97/paper97/p077.pdf

for the full story