https://verifone.cloud/docs/application-development-kit-version-48/crdsync-demo_8cpp-example
Updated: 05-Mar-2025

crdsync-demo.cpp

This is an example how to use ADK-CRD-SYNC

```
/****************************************************************************** *
 #include <stdlib.h>#include <stdio.h>#include <string
.h>#include <stdarg.h>#include <unistd.h>#include <signal.h>#include
"pthread.h"#include "crdsync.h"  #define UNUSED(x) (void) x #define MENU
"ADK-CRD-SYNC testapp:\n\1 Open\n\2 IsCardPresent\n\3 PowerUp AT24C01SC\n\4 Powe
 #define TRACE(str) printf("%s", str);#define sleep_ms(a) usleep(1000
*a); #define CRDSYNC_DBG(format, ...) { trace("CRD-SYNC-TEST: "
); trace(format, ## __VA_ARGS__); trace("\n"
); }#define CRDSYNC_DBG_BYTES(format, data, len, ...) { int i; trace(
"CRD-SYNC-TEST: "); trace(format, ## __VA_ARGS__); for (i = 0; i < len
; i++) trace("%02x ", data[i]); trace("\n"
); } static pthread_mutex_t traceMutex = PTHREAD_MUTEX_INITIALIZER;static void t
const char *format, ...); static const char *return_table[] = {
"CRDSYNC_Success",        "CRDSYNC_Failure",
"CRDSYNC_Error_Power_Action",        "CRDSYNC_Error_Not_Supported",
"CRDSYNC_Error_BadParam",        "CRDSYNC_Error_Protocol"};static const
 char *mapReturn(CRDSYNC_RET r){    if(r < 0
 || (unsigned)r >= sizeof(return_table)/sizeof(char*))        return
"unknown";    else        return
 return_table[r];} static void callbackfunctionAtExit(int
 signal){    signal = 0;    exit(signal);} void traceCallback(const
 char *str, void *data){    UNUSED(data);    trace("ADK-CRD-SYNC: "
);    trace(str);    trace("\n");} int main(int
 argc, char *argv[]){    CRDSYNC_RET ret;    int key = 0xFF;    int
 c;     UNUSED(argc);    UNUSED(argv);     signal(SIGTERM, callbackfunctionAtExi
64];    unsigned char len = sizeof(vers);    crdSync_Version(vers,len
);    CRDSYNC_DBG("CRD-SYNC library version: %s",vers);     while (key != 0
)    {        unsigned char AtrBuf[256];        unsigned char AtrLen = 255
;         printf("\n\n%s", MENU);        c = getc(stdin);        printf(
"\n\n");        key = c - '1' + 1;        while (c != '\n'
 && c != EOF)            c = getc(stdin);         if ((key >= 1 && key <= 9
) || (key > 48 && key < 53))        {            switch
 (key) {                case 0x01:                    ret = crdSync_Open(0
);                    CRDSYNC_DBG("Open(0): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x02
:                    ret = crdSync_IsCardPresent();                    CRDSYNC_D
"crdSync_IsCardPresent(): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x03
:                    ret = crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C01SC, AtrBuf, &
"crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C01SC): %d (%s)"
```

```
, ret, mapReturn(ret));                    break;                case 0x04
:                   ret = crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C02SC, AtrBuf, &A
"crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C02SC): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x05
:                   ret = crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C04SC, AtrBuf, &A
"crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C04SC): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x06
:                   ret = crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C08SC, AtrBuf, &A
"crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C08SC): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x07
:                   ret = crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C16SC, AtrBuf, &A
"crdSync_PowerUp(CRDSYNC_CARDTYPE_AT24C16SC): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x08
:                   ret = crdSync_PowerUp(CRDSYNC_CARDTYPE_ST14C02SC, AtrBuf, &A
"crdSync_PowerUp(CRDSYNC_CARDTYPE_ST14C02SC): %d (%s)"
, ret, mapReturn(ret));                    break;                case 0x09
:                {                    unsigned char TxBuf[]="12345678"
;                   unsigned short TxLen = sizeof(TxBuf);                    ret
32,TxBuf,TxLen);                    CRDSYNC_DBG(
"crdSync_WriteData(12345678): %d (%s)"
, ret, mapReturn(ret));                    break
;                }                case 49
:                {                    unsigned char TxBuf[]="99999999"
;                   unsigned short TxLen = sizeof(TxBuf);                    ret
32,TxBuf,TxLen);                    CRDSYNC_DBG(
"crdSync_WriteData(99999999): %d (%s)"
, ret, mapReturn(ret));                    break
;                }                case 50
:                {                    unsigned char RxBuf[64]={0
};                    unsigned short RxLen = sizeof(RxBuf);                     re
32,RxBuf,RxLen);                    CRDSYNC_DBG(
"crdSync_ReadData(): %d (%s)"
, ret, mapReturn(ret));                    CRDSYNC_DBG_BYTES("Read Data: "
,RxBuf,RxLen);                    break;                }
case 51
:                {                    ret = crdSync_PowerDown();
"crdSync_PowerDown(): %d (%s)", ret, mapReturn(ret));
break;                }                 case 52
:                {                    ret = crdSync_Close(0
);                    CRDSYNC_DBG("crdSync_Close(): %d (%s)"
, ret, mapReturn(ret));                    break
;                }                default:                    break
;            }        }    }    return 0;}   static void trace(const
 char *format, ...){    int size = 128;    int len = 0
;    char *buf = NULL;    char *bufr = NULL;    va_list arg;    char save;    cha
if (buf)    {        va_start(arg, format);        len
 = vsnprintf(buf, size, format, arg);        if (len
 >= size)        {            size = len + 1
;            bufr = (char *) realloc(buf, size);            if
 (bufr)            {                buf = bufr;                len
 = vsnprintf(buf, size, format, arg);            }            else
            {                free(buf);                buf = NULL;            }
if (buf)    {        ptr = buf;        while (len > 99
)        {            save = ptr[99];            ptr[99] = '\0'
;            TRACE(ptr);            ptr[99] = save;            ptr += 99
;            len -= 99
;        }        TRACE(ptr);        free(buf);    }    pthread_mutex_unlock(&t
```