ArtOfTanning, Overview Report

Art Weiss, Jr.

The following is a description of the ArtOfTanning tanning salon software and how it was developed/planned to be interfaced with smartcards. The ArtOfTanning software is a custom application written solely by Art Weiss with the exception of small sections of code. These sections, which are noted below, are external pre-written libraries used to efficiently development software. This document discusses the smartcard component primarily. Many of the other details, such as how connections and communications to the external database, tanning bed control unit, and other modules are not within the scope of this document.

The ArtOfTanning tanning salon software was written in Visual C++. All data (customer, employee, & product info, etc) was stored in an external Access database. Crystal Reports was used for report creation. The ArtOfTanning integrated with the Sundash CCS3 Tanning bed control unit. The CCS3 unit allowed complete control of all tanning beds within the salon from one location via the ArtOfTanning. Customization of the SCEZ – Smart Card Library[1] achieved the required smartcard communications. ZeitControl[2] BasicCard Compact 1.1 was used as the smartcard media, in addition, the ZeitControl IDE was used to create/compile the code written to the smartcard.

When a customer entered the salon, they put their smartcard into the reader. The tanning salon employee clicks the button “Get ID from Card”. ArtOfTanning would then reset the smartcard, check to ensure it is an ArtOfTanning smartcard, and retrieve the ID from the card. Once the ID is retrieved from the smartcard, the software would locate and retrieve that customer’s information from the external database. At that time, the customer can add more tanning credits to their account, purchase products, or spend tanning credits by tanning. The customer’s other info, such as skin type, tanning history, etc, could also be displayed. Upon tanning, the software deducts the appropriate tanning credits from the customers account and sets up the appropriate tanning booth via the CCS3 tanning control unit.

If the customer does not have a card and is a new customer then an account is created and a smart card assigned. Returning customers who have misplaced or forgotten their smart card, or perhaps never assigned one, could be issued a new one via a manual search of the account. Once the account is found (or created) and verified, a new smartcard can be issued to that customer. The card would be inserted into the reader and the employee clicks “Set Card ID”. The ArtOfTanning would then reset the smartcard, check to ensure it is an ArtOfTanning smartcard, and set the ID on the card to the customer’s ID.

ArtOfTanning did not utilize PCSC protocol. No device drivers are required for the smartcard reader to function. There were NO SCHEMATICS created, used or needed in this application. At no time did any

[1] http://www.franken.de/crypt/scez.html
[2] http://www.basiccard.com

electronics need modification. This application is entirely software-based control. Communications to the smartcard were made directly through the serial port using T=1 and APDU transmission.

At the time ArtOfTanning development was terminated, the only smartcard features implemented was storing and retrieving the customer's ID. The ability of relocating data from the external database to the smartcard was desired, but never fully investigated, nor implemented. In addition, since this software was never released into production, there will be bugs, as well as debugging and unexecuted (commented out) code included. The main graphical interface was also never perfected.

As mentioned before, the code used to interface with the smartcard was based upon the SCEZ – Smart Card Library. Most of the commands in the source code included within call subroutines that were unmodified from the SCEZ library. These commands usually have an 'sc' before the command, such as "`scBasiccardCmdGetApplId`". Since these subroutines did not require modification, they were not included. The SCEZ library can be downloaded from the above link. Although the current version available will be newer than the one used in this software, the functions and theory behind the library is the same.

The application utilized a dumbmouse reader as the smartcard hardware interface. Plans to use an Unlooper to repair the BasicCard ArtOfTanning smartcards and to use the unlooper as a secondary smartcard hardware interface were planned, but never completed. The smartcard hardware interface connects directly to a serial port on the computer via a standard serial cable. No other hardware, or software drivers were needed for the ArtOfTanning to communicate with the smartcard hardware interface.

Source code written to the the ZeitControl BasicCard

```
Declare ApplicationID = "ArtOfTanning Card"

Eeprom ID As Long = -1

Command &H80 &H10 SetID(CustID As Long)
   ID=CustID
End command

Command &H80 &H20 GetID(CustID As Long)
   CustID=ID
End command
```

Description of code on smartcard

Line 1 sets the Card's ApplicationID to an ArtOfTanning Card, as described in the overview, this prevents accidental usage of an incorrect smartcard in this application.

Line 2 creates an EEPROM variable, ID, which will store the customer's ID. It also sets the initial value of the ID to –1, meaning unassigned.

Lines 5-7 is the command that sets the EEPROM variable ID with the parameter CustID sent to it.

Lines 9-11 is the command that retrieves the EEPROM variable ID and returns it to the calling function

Excerpts of source code from ArtOfTanning tanning salon software

Entire source code of IdInterface.cpp:

```
#define printarray( name, length, array ); \
        TRACE(name); \
        for( i=0; i<length; i++ ) TRACE(" %.2X",array[i]); \
        TRACE("\n");
#define printarrayc( name, length, array ); \
        TRACE(name); \
        for( i=0; i<length; i++ ) TRACE(" %c",array[i]); \
        TRACE("\n");


#include "stdafx.h"
#include <math.h>
#include "scgeneral.h"
#include "scsmartcard.h"
#include "screader.h"
#include "scdumbmouse.h"
#include "scbasiccard.h"
#include "tdumbmouse.h"

#include <ctype.h>


#define checkreturn(f); if( ret!=0 ) { TRACE(f); goto exit; }

#define READER_TYPE SC_READER_DUMBMOUSE
#define READER_SLOT 1
#define READER_PORT "COM2"

long GetIdFromCard()
{
        SC_READER_INFO *ri;
        SC_CARD_INFO *ci;
        SC_READER_CONFIG rc;

        long tannerID=-1;

        int ret;
        int i;
        BYTE rbuffer[256];
        int resplen;

        BYTE data[]={0x01,0x02,0x03,0x04,0x05};
        BYTE keyc0[]={0x57,0x03,0xCF,0x4C,0xC7,0xD5,0x62,0x1F,
                0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};

        do {

                if( scInit() ) { TRACE("Exit.\n"); return(-1); }

                rc.type=READER_TYPE;
                rc.slot=READER_SLOT;
                rc.param=READER_PORT;

                ret = scReaderGetConfig( 0, NULL, &rc );
                if( ret!=SC_EXIT_OK ) {
                        TRACE( "Error getting reader configuration.\n" );
                        scEnd();
                        return(-1);
                };

                ri = scGeneralNewReader( rc.type, rc.slot );
                if( ri==NULL ) { TRACE("Exit.\n"); scEnd(); return(-1); };

                ci = scGeneralNewCard( );
                if( ci==NULL ) { TRACE("Exit.\n"); scEnd(); return(-1); };
```

```
		scBasiccardGetCardData(ci);

		// Init Reader
		ret = scReaderInit( ri, rc.param );
		checkreturn("Error: scReaderInit\n");

		// Activate Card
		ret = scReaderActivate( ri );
		checkreturn("Error: scReaderActivate\n");

		// Get Card Status
		ret = scReaderCardStatus( ri );
		checkreturn("Error: scReader\n");
		if( !(ri->status&SC_CARD_STATUS_PRESENT) )
		{ TRACE("Error: No Card.\n"); break; }

		// Reset Card
		ret= scReaderResetCard( ri, ci );
		checkreturn("Error: scReaderResetCard\n");

		// Get Card Type
		ret = scSmartcardGetCardType( ci );
		checkreturn("Error: scReaderGetCardType\n");
		if( (ci->type&0xFFFFFF00)!=SC_CARD_BASICCARD )
		{ TRACE("Error: Wrong Card.\n"); break; }

		// Get Application ID
		TRACE("scBasiccardCmdGetApplId:");
		ret = scBasiccardCmdGetApplId( ri, ci, rbuffer, &resplen );
		scSmartcardSimpleProcessSW( ci, &i, NULL );
		if( !( (ret==0) && (i==SC_SW_OK) ) )
		{ TRACE(" Error.\n"); break; }
		TRACE(" OK\n");
		printarrayc("  Application ID:", resplen, rbuffer );
		if (strncmp("ArtOfTanning Card",(char *)rbuffer,resplen))  // wrong card type
			return (-1);


		// Get Card ID
		TRACE("Get Card ID");
		ret = scBasiccardCmdGetTannerId( ri, ci, rbuffer, &resplen );
		scSmartcardSimpleProcessSW( ci, &i, NULL );
		if( !( (ret==0) && (i==SC_SW_OK) ) )
		{ TRACE(" Error.\n"); break; }
		TRACE(" OK\n");
		printarray("  Tanner ID:", resplen, rbuffer );

		tannerID=(long)(rbuffer[0]*pow(16,6)+rbuffer[1]*pow(16,4)+rbuffer[2]*pow(16,2)+rbuffer[3]);

		char msg[2049];
		sprintf (msg, "ID = %ld",tannerID);
		//AfxMessageBox(msg);
//UpdateData(false);
	} while( 0 );



	TRACE("ret: %d, SW: %.2X%.2X\n",ret,ci->sw[0],ci->sw[1]);

exit:

	ret = scReaderDeactivate( ri );
	if( ret!=0 ) TRACE("Error: scReaderDeactivate\n");

	ret = scReaderShutdown( ri );
	if( ret!=0 ) TRACE("Error: scReaderShutdown\n");

	scGeneralFreeCard( &ci );
	scGeneralFreeReader( &ri );

	scEnd();
	return (tannerID);

}


long SetIdToCard(long tannerID)
{
	SC_READER_INFO *ri;
```

```
SC_CARD_INFO *ci;
SC_READER_CONFIG rc;

int ret,myret=-1;
int i;
BYTE rbuffer[256];
int resplen;


BYTE data[]={0x01,0x02,0x03,0x04, 0x05};
BYTE keyc0[]={0x57,0x03,0xCF,0x4C,0xC7,0xD5,0x62,0x1F,
        0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};

do {

        if( scInit() ) { TRACE("Exit.\n"); return(-1); }

        rc.type=READER_TYPE;
        rc.slot=READER_SLOT;
        rc.param=READER_PORT;

        ret = scReaderGetConfig( 0, NULL, &rc );
        if( ret!=SC_EXIT_OK ) {
                TRACE( "Error getting reader configuration.\n" );
                scEnd();
                return(-1);
        };

        ri = scGeneralNewReader( rc.type, rc.slot );
        if( ri==NULL ) { TRACE("Exit.\n"); scEnd(); return(-1); };

        ci = scGeneralNewCard( );
        if( ci==NULL ) { TRACE("Exit.\n"); scEnd(); return(-1); };
        scBasiccardGetCardData(ci);

        // Init Reader
        ret = scReaderInit( ri, rc.param );
        checkreturn("Error: scReaderInit\n");

        // Activate Card
        ret = scReaderActivate( ri );
        checkreturn("Error: scReaderActivate\n");

        // Get Card Status
        ret = scReaderCardStatus( ri );
        checkreturn("Error: scReader\n");
        if( !(ri->status&SC_CARD_STATUS_PRESENT) )
        { TRACE("Error: No Card.\n"); break; }

        // Reset Card
        ret= scReaderResetCard( ri, ci );
        checkreturn("Error: scReaderResetCard\n");

        // Get Card Type
        ret = scSmartcardGetCardType( ci );
        checkreturn("Error: scReaderGetCardType\n");
        if( (ci->type&0xFFFFFF00)!=SC_CARD_BASICCARD )
        { TRACE("Error: Wrong Card.\n"); break; }

        // Get Application ID
        TRACE("scBasiccardCmdGetApplId:");
        ret = scBasiccardCmdGetApplId( ri, ci, rbuffer, &resplen );
        scSmartcardSimpleProcessSW( ci, &i, NULL );
        if( !( (ret==0) && (i==SC_SW_OK) ) )
        { TRACE(" Error.\n"); break; }
        TRACE(" OK\n");
        TRACE("ret: %d, SW: %.2X%.2X\n",ret,ci->sw[0],ci->sw[1]); \
        printarrayc("  Application ID:", resplen, rbuffer );
        if (strncmp("ArtOfTanning Card",(char *)rbuffer,resplen))  // wrong card type
                return (-1);


        // Get Card ID
        TRACE("Get Card ID");
        ret = scBasiccardCmdGetTannerId( ri, ci, rbuffer, &resplen );
        scSmartcardSimpleProcessSW( ci, &i, NULL );
        if( !( (ret==0) && (i==SC_SW_OK) ) )
        { TRACE(" Error.\n"); break; }
        TRACE(" OK\n");
```

```
       printarray("  Tanner ID:", resplen, rbuffer );
              TRACE("ret: %d, SW: %.2X%.2X\n",ret,ci->sw[0],ci->sw[1]);

//            UpdateData(true);
              long curID=tannerID;
              ldiv_t q;

              // Set Card ID
              TRACE("Set Card ID");
              memset(rbuffer,0,10);
              q=ldiv(curID,(long)pow(16,6));
              rbuffer[0]=(unsigned char)q.quot;
              q=ldiv(q.rem,(long)pow(16,4));
              rbuffer[1]=(unsigned char)q.quot;
              q=ldiv(q.rem,(long)pow(16,2));
              rbuffer[2]=(unsigned char)q.quot;
              rbuffer[3]=(unsigned char)q.rem;

              printarray(" NewID=",4,rbuffer);

              ret = scBasiccardCmdSetTannerId( ri, ci, rbuffer, 4, rbuffer, &resplen );

              scSmartcardSimpleProcessSW( ci, &i, NULL );
              if( !( (ret==0) && (i==SC_SW_OK) ) )
              { TRACE(" Error.\n"); break; }
              TRACE(" OK\n");
              printarray("  Set Tanner ID:", resplen, rbuffer );
              TRACE("ret: %d, SW: %.2X%.2X\n",ret,ci->sw[0],ci->sw[1]);


//ci->t1.ifsreq=0;
               //Reset Card
               ret= scReaderResetCard( ri, ci );
               checkreturn("Error: scReaderResetCard\n");
               myret=tannerID;
       } while( 0 );

       TRACE("ret: %d, SW: %.2X%.2X\n",ret,ci->sw[0],ci->sw[1]);
       //TRACE(msg);
exit:

       ret = scReaderDeactivate( ri );
       if( ret!=0 ) TRACE("Error: scReaderDeactivate\n");

       ret = scReaderShutdown( ri );
       if( ret!=0 ) TRACE("Error: scReaderShutdown\n");

       scGeneralFreeCard( &ci );
       scGeneralFreeReader( &ri );

       scEnd();
       return(myret);
}
```

Description of IdInterface.cpp:

IdInterface.cpp is the main interface the ArtOfTanning software uses to communicate with the smartcard subroutines contained in the SCEZ library from the main application. Both the `SetIdToCard` and `GetIdFromCard()` subroutines are essentially the same. The subroutines initialize the reader, reset the smartcard, and check to make sure the smartcard is both a BasicCard and an ArtOfTanning card. If everything is still ok, the subroutines set or get the ID accordingly. Finally, the subroutines shutdown the reader

Function scBasiccardCmdGetTannerID from scbasiccard.c :

```
int scBasiccardCmdGetTannerId( SC_READER_INFO *ri, SC_CARD_INFO *ci,
       BYTE *applid, int *length )
{
       BYTE cmd[ 5+10 ];
       BYTE rsp[ SC_GENERAL_SHORT_DATA_SIZE+2 ];
       SC_APDU apdu;
```

```
        int ret;

        ci->sw[0]=0x00; ci->sw[1]=0x00;

        apdu.cse=SC_APDU_CASE_2_SHORT;
        apdu.cmd=cmd;
        apdu.cmdlen=5;
        apdu.rsp=rsp;
        apdu.rsplen=0;

        memset( cmd, 0, sizeof(cmd) );

        cmd[0]=0x80;
        cmd[1]=0x20;
        /* For ACR20 compatibility not 0x00. */
        cmd[4]=0x04;


        if( (ret=scBasiccardEncrCAPDU( ci, &apdu )) !=SC_EXIT_OK ) return( ret );
        ret=scReaderSendAPDU( ri, ci, &apdu );
        memset( cmd, 0, sizeof(cmd) );
        if( ret!=SC_EXIT_OK ) return( ret );
        if( (ret=scBasiccardDecrRAPDU( ci, &apdu )) !=SC_EXIT_OK ) return( ret );

        if( apdu.rsplen<2 ) return( SC_EXIT_BAD_SW );

        ci->sw[0] = rsp[apdu.rsplen-2];
        ci->sw[1] = rsp[apdu.rsplen-1];

        memcpy( applid, rsp, apdu.rsplen-2 );
        *length = apdu.rsplen-2;

        memset( rsp, 0, sizeof(rsp) );

        return( SC_EXIT_OK );
}
```

Description of Function scBasiccardCmdGetTannerID from scbasiccard.c :

Scbasiccard.c is included in the SCEZ Smart Card Library. Source code was added to fit the application's needs. `ScBasiccardCmdGetTannerId` creates the actual command to be sent to the smartcard, tells the reader to send the command, and retrieves the result from the smartcard. The command being sent to the smartcard, `cmd` is set to [0x80, 0x20,0x00,0x00,0x04,0x00]. Review of the source code written to an ArtOfTanning smartcard, cmd (80, 20) is the GetID command that retrieves the customer ID from EEPROM. The `ScBasiccardCmdGetTannerId` subroutine then retrieves the result from the smartcard. The result is checked as a successful command. The remaining data section of the result is then copied to the memory location the customer id is to be stored (`memcpy( applid, rsp, apdu.rsplen-2 );`).

Function scBasiccardCmdSetTannerID from scbasiccard.c :

```
int scBasiccardCmdSetTannerId( SC_READER_INFO *ri, SC_CARD_INFO *ci,
        const BYTE *data, int datalen, BYTE *resp, int *resplen )
{
        BYTE cmd[ SC_GENERAL_SHORT_DATA_SIZE+5+1 ];
        BYTE rsp[ SC_GENERAL_SHORT_DATA_SIZE+2 ];
        SC_APDU apdu;
        int ret;

        ci->sw[0]=0x00; ci->sw[1]=0x00;

        if( datalen>255 ) return( SC_EXIT_BAD_PARAM );

        apdu.cse=SC_APDU_CASE_4_SHORT;
        apdu.cmd=cmd;
        apdu.cmdlen=5+1+datalen;
        apdu.rsp=rsp;
        apdu.rsplen=0;

        memset( cmd, 0, sizeof(cmd) );

        cmd[0]=0x80;
        cmd[1]=0x10;
        //cmd[2]=incr;
        cmd[4]=datalen;
        cmd[5+datalen]=datalen;

        memcpy( cmd+5, data, datalen );

        if( (ret=scBasiccardEncrCAPDU( ci, &apdu )) !=SC_EXIT_OK ) return( ret );
        ret=scReaderSendAPDU( ri, ci, &apdu );
        memset( cmd, 0, sizeof(cmd) );
        if( ret!=SC_EXIT_OK ) return( ret );
        if( (ret=scBasiccardDecrRAPDU( ci, &apdu )) !=SC_EXIT_OK ) return( ret );

        if( apdu.rsplen<2 ) return( SC_EXIT_BAD_SW );

        ci->sw[0] = rsp[apdu.rsplen-2];
        ci->sw[1] = rsp[apdu.rsplen-1];

        memcpy( resp, rsp, apdu.rsplen-2 );
        *resplen = apdu.rsplen-2;

        memset( rsp, 0, sizeof(rsp) );

        return( SC_EXIT_OK );
}
```

Description of Function scBasiccardCmdSetTannerID from scbasiccard.c :

As with the previous subroutine, `ScBasiccardCmdSetTannerId` creates the actual command to be sent to the smartcard, tells the reader to send the command, and retrieves the result from the smartcard. Unlike the previous subroutine, this command being sent to the smartcard will differ from time to time. The command being sent to the smartcard, `cmd` is set to [0x80, 0x10, 0x00, 0x00, the length of the data in bytes, the data (taking up as many bytes as indicated in previous byte), the length of the data in bytes again]. Review of the source code written to an ArtOfTanning smartcard, cmd (80, 10) is the SetID command that sets the customer ID in EEPROM. The `ScBasiccardCmdSetTannerId` subroutine then retrieves the result from the smartcard. The result is checked as a successful command.

Sample actual communication log between smartcard and ArtOfTanning

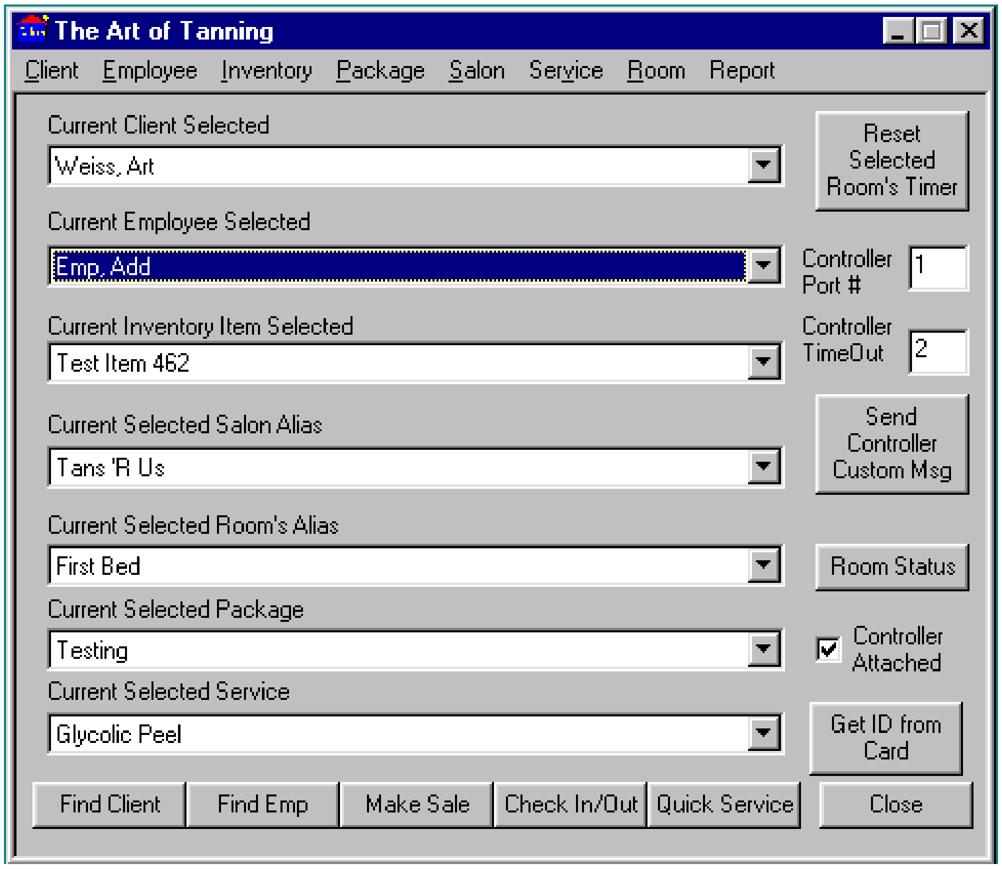
```
 1. <- 3B EF 00 FF 81 31 50 45 42 61 73 69 63 43 61 72 64 20 5A 43 31 2E 31 CC
 2. -> 00 00 05 C0 0E 00 00 00 CB
 3. <- 00 00 13 41 72 74 4F 66 54 61 6E 6E 69 6E 67 20 43 61 72 64 61 11 4C
 4. -> 00 40 0A 80 20 00 00 04 00 00 00 00 04 EA
 5. <- 00 40 06 FF FF FF FF 90 00 D6
 6. -> 00 00 0A 80 10 00 00 04 00 00 01 DA 04 41
 7. <- 00 00 06 00 00 01 DA 90 00 4D
```

Description of communications

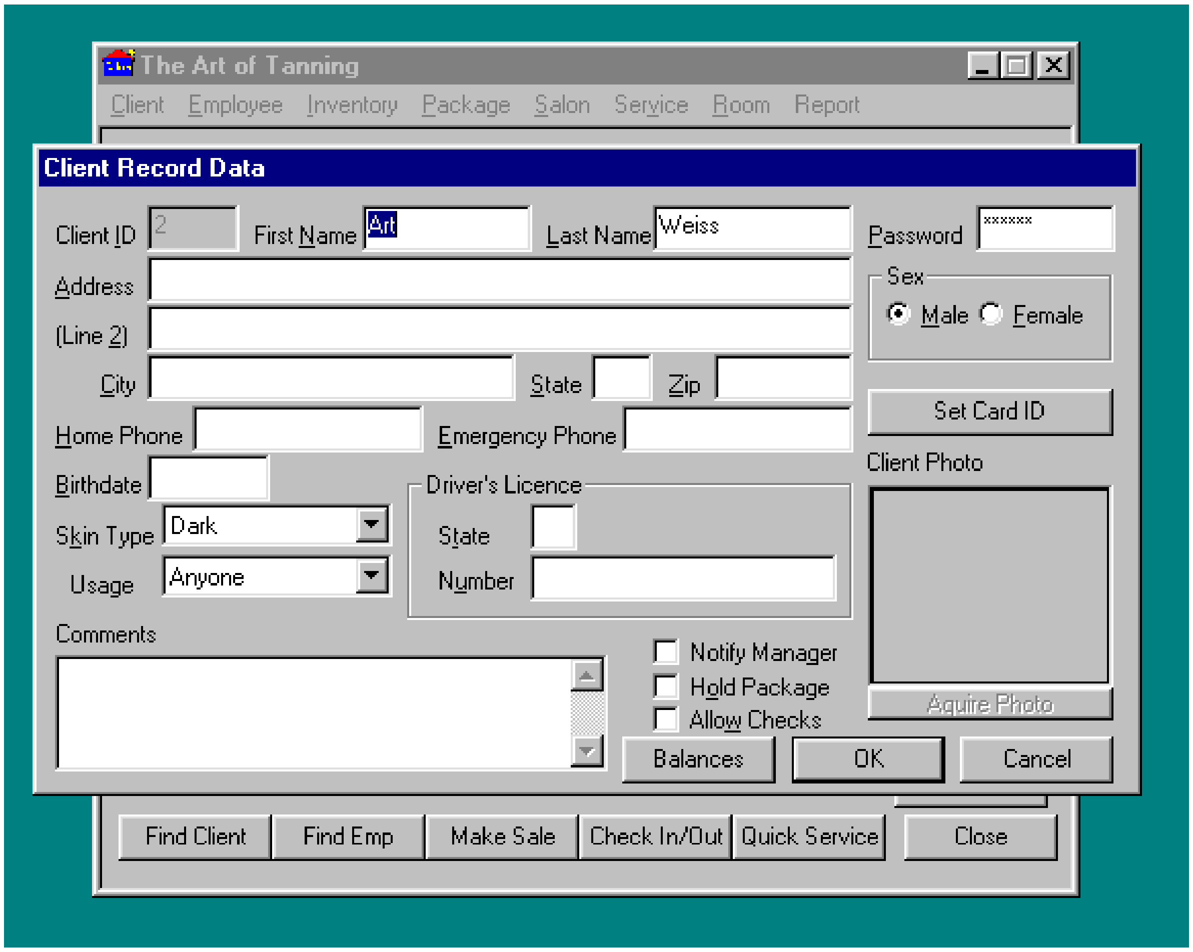
In each of the above lines `->` signifies data being sent to the smartcard and `<-` signifies data from the smart card. As you will notice, in the communications above, there are more bytes than which was indicated in the source code of ArtOfTanning. The SCEZ Smart Card Library automatically adds the bytes prior to the commands created by ArtOfTanning as well as the checksum appended to the end of the command. These extra bytes form the APDU transmission for use in the T=1 protocol.

Line 1 is the ATR returned from the smartcard upon reset. This is the standard unmodified ATR of a ZeitControl BasicCard Compact 1.1.
Line 2 sends the standard “Get Application ID” command to the smartcard.
Line 3 retrieves the result. Bytes 4-20 = “ArtOfTanning Card” in hex (41=’A’…)
Line 4 requests the ID from the smartcard, notice the [`80 20 00 00 04 00`] as was indicated in the `ScBasiccardCmdGetTannerId` description above.
Line 5 retrieves the result. Bytes 2 & 3 [`40 06`] indicate a successful command. Bytes 4-7 contain the actual ID, in this case, -1 [`FF FF FF FF`].
Line 6 sets the ID on the card to a new value. As mentioned in the description of `ScBasiccardCmdGetTannerId`, the core command of [`80 10 00 00 04 00 00 01 DA 04`] sets the new value to 474 [`00 00 01 DA`].
Line 7 acknowledges the command and returns the new value in bytes 4-7 [`00 00 01 DA`].

Screenshots from the ArtOfTanning


The Art of Tanning
Client Employee Inventory Package Salon Service Room Report
Reset Selected Room's Timer
Send Controller Custom Msg
Room Status
Get ID from Card
Find Client
Find Emp
Make Sale
Check In/Out
Quick Service
Close

Current Client Selected: Weiss, Art
Current Employee Selected: Emp, Add
Current Inventory Item Selected: Test Item 462
Current Selected Salon Alias: Tans 'R Us
Current Selected Room's Alias: First Bed
Current Selected Package: Testing
Current Selected Service: Glycolic Peel
Controller Port #: 1
Controller TimeOut: 2
[✓] Controller Attached



The Art of Tanning

Client Record Data

Client ID 2
First Name Art
Last Name Weiss
Password ******
Address
(Line 2)
City
State
Zip
Home Phone
Emergency Phone
Birthdate
Skin Type Dark
Usage Anyone
Sex: [✓ Male] Female
Driver's Licence
State
Number
Client Photo
Comments
Notify Manager ☐
Hold Package ☐
Allow Checks ☐