ANEXA 102

Specificații de interfațare cu SIUI+SIPE+CEAS pentru aplicațiile de raportare ale furnizorilor de servicii medicale şi farmaceutice

Specificații eCard.SDK

ISTORICUL REVIZIILOR DOCUMENTULUI

Versiune	Data	Descriere
1.0 (PROIECT)	04.12.2012	Versiunea propusa a documentului
1.1 (PUBLICATĂ)	05.12.2012	Revizuire text
1.2 (PROIECT)	06.12.2012	Revizuire text
2.0 (PROIECT)	06.12.2012	Eliminare capitol 5.2
3.0 (PROIECT)	03.07.2014	Includere modificari din versiunea 1.0.53 a SDK-ului.
4.0 (PUBLICATĂ)	25.07.2014	Includere modificari din versiunea 1.1.54 a SDK-ului.
5.0 (PUBLICATĂ)	14.10.2020	Includerea versiunii 1.2.0.0 a SDK-ului.
5.1 (PUBLICATĂ)	09.12.2020	Includerea versiunii 1.2.0.1 a SDK-ului.

CUPRINS

1. INTRODUCERE

Acest document descrie din punct de vedere tehnic modalitatile de interfatare cu SDK-ul implementat in cadrul CEAS pentru cititoarele de smartcard-uri in vederea.

eCard.SDK permite dezvoltatorilor de aplicatii medicale sa interactioneze cu terminalele pentru citirea si scrierea informatiilor pe cardurile de sanatate.

2. FUNCTIONALITATI

Principalele functionalitati ale eCard.SDK sunt descrise mai jos:

2.1. Interfata cu aplicatia desktop

Aceasta interfata expune metode si evenimente ce pot fi accesate de aplicatia desktop.

2.2. Obtinere token

Accesul la citirea si scrierea datelor pe card, precum si executia altor operatii specifice pe card se realizeaza pe baza profilului utilizatorului logat in aplicatia desktop. Acestui profil ii corespunde o matrice de drepturi pe card. eCard.SDK permite utilizarea urmatoarelor profile :

1. ***MedicFamilie*** -rol dedicat medicului de familie;
2. ***Specialist***-rol dedicat medicului specialist;
3. ***Farmacist***- rol dedicat farmacistului;
4. ***Urgenta***- rol dedicat furnizorilor de servicii medicale de urgenta;
5. ***EmitentCard***- rol dedicat emitentului CEAS / autoritatii de sanatate indreptatite sa actualizeze anumite date inscrise pe CEAS;
6. ***ProviderSubventie***-rol dedicat anumitor autoritati (usual din afara sistemului de sanatate) care pot oferi subventii titularului de CEAS;
7. ***FurnizorServiciiSubventie***- rol decdicat anumitor entitati (companii, etc) care pot oferi servicii titularului CEAS in contul subventilor oferite de Provider Subventie.

Atunci cand aplicatia desktop se conecteaza la terminal, se initieaza o sesiune de lucru. Pentru a putea accesa metodele expuse de eCard.SDK este necesara obinerea unui token, token care contine si profilul corespunzator al utilizatorului. Tokenul se obtine prin apelul metodei ***ObtineToken*** si este valabil pe toata durata sesiunii de lucru. Tokenul se transmite ca argument la fiecare metoda din eCard.SDK. Daca terminalul este deconectat, tokenul se resteaza si este necesara obtinerea unui token nou.

2.3. Activare card

Atunci cand cardul este utilizat prima data, este necesara activarea acestuia. Procesul de activare consta in schimbarea pinului de transport cu pinul de autentificare. Pinul de transport este 000. Pinul de autentificare trebuie sa contina 4 numere. Nici o alta operatie nu este posibila pe card pana cand cardul nu este activat. Activarea cardului se face prin apelul metodei ***ActiveazaCard***;

2.4. Citire date de pe card

Operatia de citire permite citirea infomatiilor de pe un card. Aceasta operatie este posibila numai pentru un card activ. Datele de pe card se pot citi corespunzator profilului de utilizator, conform matricei de drepturi. Citirea datelor se realizeaza prin apelul metodei ***CitesteDate*** si se poate face atat in mod online, cat si in mod offline (UM indisponibil).

2.5. Scriere date pe card

Operatia de scriere permite editarea datelor pe un card. Acesta operatie este posibila numai pentru un card activ. Datele de pe card se pot scrie corespunzator profilului de utilizator, conform matricei de drepturi. Scrierea datelor se realizeaza prin apelul metodei ***EditeazaDate*** si se poate face numai in mod online.

2.6. Semnatura digitala

Operatia de semnare digitala se utilizeaza pentru semnarea unei tranzactii folosind certificatul digital de pe card. Operatia de semnare se poate realiza numai daca cardul este activ si se face prin apelul metodei ***ComputeHash***.

2.7. Schimbare PIN

Schimbarea pinului este operatia de schimbare a pinului de autentificare. Acesta operatie este posibila numai pentru un card activ si numai in mod online. Operatia de schimbare pin este initiata prin apelul metodei ***SchimbaPIN*** si consta in editarea pinului actual, a pinului nou si confirmarea pinului nou.

2.8. Resetare PIN

Resetarea pinului se face in doua cazuri, atunci cand posesorul cardului a uitat pinul de autentificare si in cazul in cardul a fost blocat ca urmare a introducerii repetate (de 5 ori) a unui pin gresit. Operatia este posibila numai daca in prealabil a fost anuntat help-deskul si s-a primit confirmarea ca resetarea pinului este permisa. Se poate executa numai in mod online. Operatia de resetare a pinului este initiata prin apelul metodei ***ReseteazaPIN*** si consta in introducerea pinului de reset (0000).

2.9. Eveniment schimbare stare card in terminal

Acest eveniment este aruncat de eCard.SDK atunci cand cardul este inserat sau retras din terminal. Sunt posibile urmatoarele stari ale cardului in terminal:

```
/// <summary>
/// Enumerator stari card in terminal
/// </summary>
public enum StareCardInTerminal
{
    /// <summary>
    /// Card inserat in terminal
    /// </summary>
    CardInserat = 2,
    /// <summary>
    /// Card scos din terminal
    /// </summary>
    CardRetras = 1
}
```

2.10. Eveniment schimbare stare card

Acest eveniment este aruncat de eCard.SDK atunci cand se schimba starea cardului (de exemplu din activ in blocat). Sunt posibile urmatoarele stari ale cardului:

```
/// <summary>
 /// Enumerator pentru stari card
 /// </summary>
 public enum StareCard
 {

     Propus=1,
     InrolareInCurs=2,
     Transport=3,
     Neconform=4,
     Inactiv=5,
     DateIncorecte=6,
     Activ=7,
     ActualizareInCurs=8,
     Expirat=9,
     Blocat=10,
     Suspendat=11,
     Decedat=12
 }
```

2.11. Eveniment schimbare stare autentificare

Acest eveniment este aruncat de eCard.SDK atunci cand se schimba starea de autentificare. Sunt posibile urmatoarele stari de autentificare:

```
/// <summary>
/// Enumerator moduri autentificare
/// </summary>
public enum StariAutentificare
{
    /// <summary>
    /// Stare valabila atat timp cat utilizatorul nu s-a autentificat
    /// prin introducerea pinului corect
    /// </summary>
    NEAUTENTIFICAT = 0,
    /// <summary>
    /// Autentificat offline, numai pe card, fara confirmarea UM
    /// </summary>
    AUTENTIFICAT_OFFILINE = 1,
    /// <summary>
    /// Autentificat pe card si cu confirmarea UM
    /// </summary>
    AUTENTIFICAT = 2,
    /// <summary>
    ///Stare valabila atunci cand utilizatorul a incercat sa se autentifice, dar autentificarea a esuat
    /// </summary>
    AUTENTIFICARE_ESUATA = 3
}
```

2.12. Eveniment schimbare stare comunicatie cu UM

Acest eveniment este aruncat atunci cand se schimba starea comunicatiei cu UM. Sunt posibile urmatoarele stari ale comunicatiei cu UM:

```
/// <summary>
/// Enumerator stari comunicatie cu UM
/// </summary>
public enum StareComunicatieCuUM
{
    /// <summary>
    ///Comunicatia cu UM este Ok
    /// </summary>
    COMUNICATIE_OK = 0,
    /// <summary>
    /// UM disponibil, dar intoarce timeout
    /// </summary>
    TIMEOUT = 1,
```

```
        /// <summary>
        /// UM disponibil, dar nu poate interoga CA
        /// </summary>
        CA_INDISPONIBIL = 2,
        /// <summary>
        /// UM disponibil, dar nu poate intoroga eCard
        /// </summary>
        ECARD_INDISPONIBIL = 3,
        /// <summary>
        /// Serviciul de comunicatie al SDK cu UM este indisponibil
        /// </summary>
        SDK_SERVICIU_COMUNICATIE_INDISPONIBIL = 4,
        /// <summary>
        ///UM indisponibil
        /// </summary>
        UM_INDISPONIBIL = 5
    }
```

2.13. Eveniment schimbare stare editare pe terminal

Acest eveniment se arunca atunci cand se schimba starea de editare pe terminal. Sunt posibile urmatoarele stari de editare pe terminal:

```
    /// <summary>
    /// Enumerator moduri editare pe terminal
    /// </summary>
    public enum ModEditarePeTerminal
    {
        /// <summary>
        /// Nu se face editare pe terminal
        /// </summary>
        NU_SE_EDITEAZA_PE_TERMINAL=0,
        /// <summary>
        /// Editare pe terminal pentru autentificare pe card
        /// </summary>
        AUTENTIFICARE=1,
        /// <summary>
        /// Editare pe terminal pentru schimbare pin
        /// </summary>
        SCHIMBARE_PIN=2,
        /// <summary>
        /// Editare pe terminal pentru resetare pin
        /// </summary>
        RESETARE_PIN=3,
        /// <summary>
        /// Schimbare pin transport
        /// </summary>
        SCHIMBARE_PIN_TRANSPORT=4
    }
```

2.14. Inrolare terminale

Inainte de utilizare, fiecare terminal trebuie sa treaca prin procesul de inrolare. Acest proces se realizeaza automat, prin comunicarea cu Unitatea de Management. Procesul de auto-inrolare este declansat atunci cand se solicita obtinerea unui token si pe statia locala nu exista inca datele de inrolare (cheie criptare PIN si profile) corespunzatoare. In cadrul procesului, SDK contacteaza Unitatea de Management pentru obtinerea fisierului de inrolare si il salveaza pe discul local.
La urmatoarea folosire a SDK cu aceeasi parametri (cod fiscal, tip furnizor, numar contract, data contract, casa asigurare) se va folosi fisierul salvat pe discul local. La schimbarea unuia dintre acesti parametri, se reia procesul de inrolare.

3. INSTALARE SI CONFIGURARE

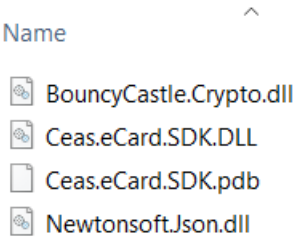
3.1. Prerechizite

Pe statia unde se foloseste eCard.SDK trebuie sa fie instalate urmatoarele pachete software:

- .Net Framework 2.0
- CardMan_SPE_API_V1_1_0_11 – pentru folosirea cititorului HID OmniKey 3821
- Drivere Omnikey 3821 – pentru folosirea cititorului HID OmniKey 3821
- Drivere ACR83 – pentru folosirea cititorului ACR83 PINEasy Smart Card Reader

3.2. Instalare

eCard.SDK este livrat sub forma unui director ce contine urmatoarele fisiere:



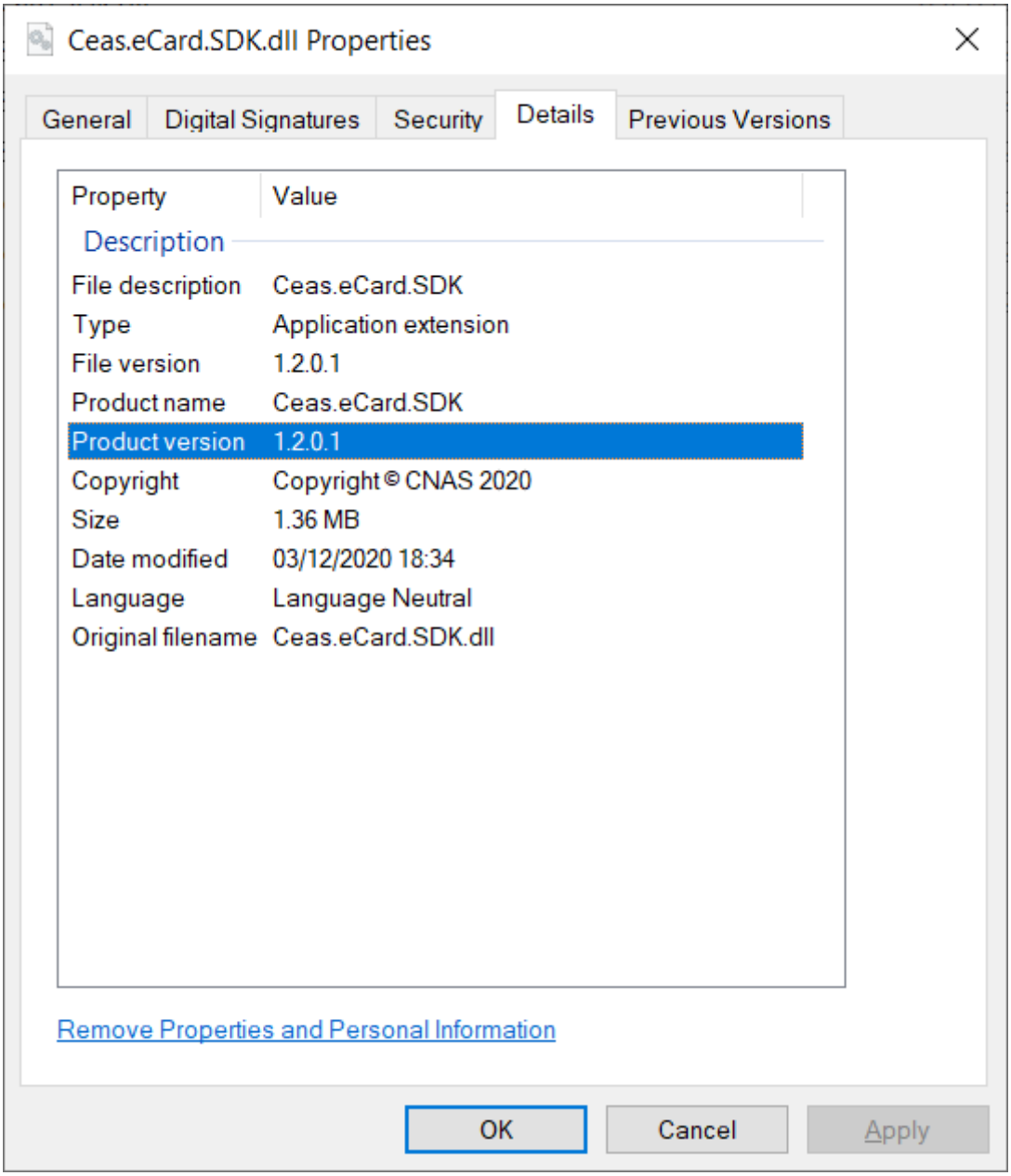
- Ceas.eCard.SDK.dll – assembly .Net ce implementeaza metodele expuse de SDK
- Ceas.eCard.SDK.pdb – informatii suplimentare, folosite la depanarea SDK-ului
- BouncyCastle.Crypto.dll – librarie externa folosita de SDK
- Newtonsoft.Json.dll - librarie externa folosita de SDK

Pentru folosirea eCard.SDK nu este necesara realizarea vreunei operatii de instalare. Este sufficient ca aplicatia apelanta sa:

- Faca referinta la Ceas.eCard.SDK.dll si sa apeleze metodele descrise in specificatii
- Distribuie cele 4 fisiere ale SDK-ului impreuna cu aplicatia

3.3. Actualizare

eCard.SDK nu include mecanisme de actualizare a versiunii. Dezvoltatorii aplicatiilor ce folosesc eCard.SDK sunt responsabili de distribuirea celei mai recente versiuni a SDK-ului impreuna cu aplicatia proprie.
Versiunea curenta a SDK-ului este indicata de fisierul Ceas.eCard.SDK.dll:



3.4. Directoare lucru

eCard.SDK creeaza si foloseste directorul:

- C:\Documents and Settings\All Users\Application Data\Novensys.eCard.SDK (pe sisteme de operare Windows XP)
- C:\ProgramData\Novensys.eCard.SDK (pe sisteme de operare Windows Vista, Windows 7, Windows 8)

In cadrul acestui director, se folosesc urmatoarele subdirectoare:

- TerminalData – pastreaza unul sau mai multe fisiere .tdx obtinute in cadrul procesului de inrolare.
- <NumeAplicatieApelanta> - pentru fiecare aplicatie apelanta, se creeaza un subdirector cu numele aplicatiei. In subdirector, se salveaza logurile de folosire a SDK-ului.

3.5. Configurare

Inainte de a utiliza eCard.SDK, este obligatorie configurarea parametrilor de conectare la Unitatea de Management. Configurarea se poate face in doua moduri:

1. Modificand fisierul app.config al aplicatiei .Net apelante astfel:
   a. Se adauga elemental cu numele `Novensys.eCard.SDK.Properties.Settings` in /configuration/configSections/sectionGroup
   b. Se adauga elementul cu numele `Novensys.eCard.SDK.Properties.Settings` in /configuration/applicationSettings
   c. Se modifica valoarea setarilor `IPUnitateManagement` si `PortUnitateManagement`

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <configSections>
                <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup,
System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
                        <section name="Novensys.eCard.SDK.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
                </sectionGroup>
        </configSections>
        <applicationSettings>
                <Novensys.eCard.SDK.Properties.Settings>
                        <setting name="IPUnitateManagement" serializeAs="String">
                                <value>testumceas.siui.ro</value>
                        </setting>
                        <setting name="PortUnitateManagement" serializeAs="String">
                                <value>443</value>
                        </setting>
                </Novensys.eCard.SDK.Properties.Settings>
        </applicationSettings>
</configuration>
```

2. Se apeleaza metoda statica `Novensys.eCard.SDK.ManagerSesiuniCard.SetAdresaUnitateManagement` si se paseaza ca parametrii adresa si portul UM. Metoda trebuie apelata inainte de crearea primei sesiuni de lucru cu cardul:

```
Novensys.eCard.SDK.ManagerSesiuniCard.SetAdresaUnitateManagement("testumceas.siui.ro", 443);
```

Nota! Adresa *testumceas.siui.ro* si portul *443* sunt trecute cu scop de exemplu. Inainte de a folosi eCard.SDK, configurati adresa si portul corecte pentru Unitatea de Management.

3.6. Dezinstalare

Pentru dezinstalarea eCard.SDK, se sterg cele 4 fisiere distribuite impreuna cu aplicatia apelanta.

4. PROGRAMARE CU ECARD.SDK

4.1. Pornirea unei sesiuni de lucru cu terminalul

Semnaturi metode

```
/// <summary>
/// Creeaza o sesiune noua care va folosi primul cititor permis.
/// Dupa conectarea la primul cititor, sesiunea nu se va muta la alt cititor.
/// </summary>
SesiuneCard = ManagerSesiuniCard.StartSesiuneNoua();

/// <summary>
/// Creeaza o sesiune noua de card pentru cititorul cu numele specificat.
/// Pentru lista completa de cititoare permise, vezi metoda <see cref="GetSupportedTerminalNames"/>
/// </summary>
/// <param name="desiredTerminalName"></param>
SesiuneCard = ManagerSesiuniCard.StartSesiuneNoua("OMNIKEY CardMan 3821");
```

Cele doua metode controleaza tipul de cititor de carduri ce urmeaza a fi folosit. In scenariile in care exista un singur tip de cititor de card conectat la calculator, se poate apela prima forma a metodei. In scenariile in care exista mai multe tipuri de cititoare de card, se poate apela a doua metoda pentru a indica in mod explicit tipul de cititor ce urmeaza a fi folosit. Lista cititoarelor permise se poate obtine prin apelarea metodei:

```
/// <summary>
/// Intoarce lista cu terminalele suportate.
/// Orice valoare returnata poate fi folosita in apelarea metodei <see cref="StartSesiuneNoua"/>
/// </summary>
string[] listaCititoare = ManagerSesiuniCard.GetSupportedTerminalNames();
```

Exemplu de implementare

```
public OperatiiCardForm()
{
    InitializeComponent();

    //instantieaza dictionarul de mesaje
    MesajeRaspuns = new MesajeRaspunsCard();

    //instantieaza o sesiune card
    SesiuneCard = ManagerSesiuniCard.StartSesiuneNoua();

    //subscrie la evenimentul de schimbare stare card in terminal
    SesiuneCard.StareCardInTerminalSchimbata += new
StareCardInTerminalSchimbataEventHandler(SesiuneCard_StareCardInTerminalSchimbata);

    //subscrie la evenimentul dupa schimbarea starii cardului in terminal
    SesiuneCard.DupaStareCardInTerminalSchimbata += new
DupaStareCardInTerminalSchimbataEventHandler(SesiuneCard_DupaStareCardInTerminalSchimbata);

    //subscrie la evenimentul de schimbare stare autentificare
    SesiuneCard.StareAutentificareSchimbata += new
StareAutentificareSchimbataEventHandler(SesiuneCard_StareAutentificareSchimbata);

    //subscrie la evenimentul de schimbare stare card
    SesiuneCard.StareCardSchimbata += new StareCardSchimbataEventHandler(SesiuneCard_StareCardSchimbata);

    //subscrie la evenimentul de schimbare stare editare pe terminal
```

```
    SesiuneCard.StareEditarePeTerminalSchimbata += new
StareEditarePeTerminalSchimbataEventHandler(SesiuneCard_StareEditarePeTerminalSchimbata);

    //subscrie la evenimentul de schimbare stare comunicatie cu UM
    SesiuneCard.StareComunicatieCuUMSchimbata += new
StareComunicatieCuUMSchimbataEventHandler(SesiuneCard_StareComunicatieCuUMSchimbata);

    // subscribe la evenimentul de citire pin de la terminale fara tastatura
    SesiuneCard.UserInputRequired += SesiuneCard_UserInputRequired;
}
```

4.2. Obtinere token

Semnatura metoda

```
/// <summary>
/// Intoarce un token care va fi utilizat atunci cand se fac apelurile de activare, citire sau scriere pe
card
/// </summary>
/// <param name="cif">Cod identificare furnizor servicii medicale</param>
/// <param name="identificatorDrepturi">Identificator drepturi</param>
/// <returns>Intoarce un token valid daca identificarea s-a facut cu succes, null pentru identificare
nereusita</returns>
string ObtineToken(string cif, IdentificatorDrepturi identificatorDrepturi);
```

Exemplu de implementare

```
private void btnGetToken_Click(object sender, EventArgs e)
        {
            ObtineToken();

        }

        private void ObtineToken()
        {
            string cif = "5678";

            try
            {
                //creeaza identificator drepturi pe baza caruia se va obtine tokenul
                IdentificatorDrepturi identificatorDrepturi = new IdentificatorDrepturi();
                identificatorDrepturi.NumarContract = "101";
                identificatorDrepturi.DataContract = new DateTime(2012, 12, 31, 22, 0, 0);
                identificatorDrepturi.CasaAsigurare = "CAS-B";
                identificatorDrepturi.TipFurnizor = (comboBoxTipContract.SelectedItem as TipContract).Cod;
                identificatorDrepturi.CUI = "123456";

                //se obtine tokenul
                string token = SesiuneCard.ObtineToken(cif, identificatorDrepturi);

                if (token == null)
                    throw new Exception("Token invalid.\r\n");

                //memoreaza tokenul obtinut la nivelul sesiunii
                this.Token = token;

                //afiseaza tokenul obtinut
                textBoxToken.Text = (this.SesiuneCard.Token != null) ? this.SesiuneCard.Token : string.Empty;

                //afiseaza profilul curent
                if (this.SesiuneCard.ProfilId != null)
                    textBoxProfilCurent.Text = ((ProfileCard)this.SesiuneCard.ProfilId).ToString();

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
```

4.3. Activare card

Semnatura metoda

```
/// <summary>
/// Activeaza card (numai in mod online)
/// </summary>
/// <param name="token">Tokenul obtinut anterior pe baza identificatorului de drepturi de scriere</param>
/// <returns>Intoarce 0 pentru operatie cu succes sau cod eroare in caz de esec</returns>
int ActiveazaCard(string token);
```

Exemplu de implementare

```
private void btnActiveazaCard_Click(object sender, EventArgs e)
        {

            try
            {
                //se apeleaza metoda de activare card
                int rez = SesiuneCard.ActiveazaCard(this.Token);

                //se afiseaza mesajul corespunzator codului de raspuns
                MessageBox.Show(MesajeRaspuns[(CoduriRaspunsOperatieCard)Enum.Parse(
                    typeof(CoduriRaspunsOperatieCard), rez.ToString())], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
```

4.4. Schimbare PIN

Semnatura metoda

```
/// <summary>
/// Initiaza schimbarea PIN-ului
/// </summary>
/// <param name="token">Token obtinut pe baza identificatorului de drepturi</param>
/// <returns></returns>
int SchimbaPIN(string token);
```

Exemplu de implementare

```
private void buttonSchimbarePIN_Click(object sender, EventArgs e)
{
        try
        {
                //se apeleaza metoda de activare card
                int rez = this.SesiuneCard.SchimbaPIN(this.Token);

                //se afiseaza mesajul corespunzator codului de raspuns
                MessageBox.Show(MesajeRaspuns[(CoduriRaspunsOperatieCard)Enum.Parse(
                        typeof(CoduriRaspunsOperatieCard), rez.ToString())], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

        }
        catch (Exception ex)
        {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
}
```

4.5. Resetare PIN

Semnatura metoda

```
/// <summary>
/// Initiaza resetarea PIN-ului
/// </summary>
/// <param name="token">Token obtinut pe baza identificatorului de drepturi</param>
/// <returns></returns>
int ReseteazaPIN(string token);
```

Exemplu de implementare

```
private void buttonResetarePIN_Click(object sender, EventArgs e)
{
        try
        {
                //se apeleaza metoda de activare card
                int rez = this.SesiuneCard.ReseteazaPIN(this.Token);

                //se afiseaza mesajul corespunzator codului de raspuns
                MessageBox.Show(MesajeRaspuns[(CoduriRaspunsOperatieCard)Enum.Parse(
                        typeof(CoduriRaspunsOperatieCard), rez.ToString())], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

        }
        catch (Exception ex)
        {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
}
```

4.6. Citeste date

Semnatura metoda

```
/// <summary>
/// Citeste date de pe card
/// Functioneaza in mod online si mod offline (in mod offline tokenul nu mai este verificat)
/// </summary>
/// <param name="token">Tokenul obtinut anterior pe baza identificatorului de drepturi de scriere</param>
/// <param name="campuriDeCitit">Lista campuri de citit de pe CIP</param>
/// <param name="cardData">Obiect populat cu datele citite de pe CIP</param>
/// <param name="rezultatOperatie">Dictionar care contine perechi cod camp-cod raspuns operatie camp, indica
daca operatia a reusit sau nu pentru un anumit camp</param>
/// <returns>Intoarce 0 pentru operatie cu succes sau cod eroare in caz de esec</returns>
int CitesteDate(string token, List<CoduriCampuriCard> campuriDeCitit, ref CardData cardData, ref
Dictionary<CoduriCampuriCard, CoduriRaspunsOperatieCamp> rezultatOperatie);
```

Exemplu de implementare

```
private void btnCitesteDate_Click(object sender, EventArgs e)
        {
            CardData cardData = null;

            try
            {
                //creeaza lista de campuri de citit
                List<CoduriCampuriCard> campuriDeCitit = new List<CoduriCampuriCard>();
                campuriDeCitit.Add(CoduriCampuriCard.A1);   //Numar card
                campuriDeCitit.Add(CoduriCampuriCard.A3);   //Versiune
                campuriDeCitit.Add(CoduriCampuriCard.B1);   //Nume
                campuriDeCitit.Add(CoduriCampuriCard.B2);   //Prenume
                campuriDeCitit.Add(CoduriCampuriCard.B3);   //Data nasterii
                campuriDeCitit.Add(CoduriCampuriCard.B4);   //CNP
                campuriDeCitit.Add(CoduriCampuriCard.C1);   //Numar asigurat
                campuriDeCitit.Add(CoduriCampuriCard.C4);   //Nume medic familie
                campuriDeCitit.Add(CoduriCampuriCard.C5);   //Prenume medic familie
```

```
                campuriDeCitit.Add(CoduriCampuriCard.C6);   //Cod medic familie
                campuriDeCitit.Add(CoduriCampuriCard.C7);   //Telefon medic familie
                campuriDeCitit.Add(CoduriCampuriCard.C8);   //Persoane contact
                campuriDeCitit.Add(CoduriCampuriCard.D1);   //Grupa sanguina
                campuriDeCitit.Add(CoduriCampuriCard.D2);   //RH
                campuriDeCitit.Add(CoduriCampuriCard.D4);   //Status donator organe
                campuriDeCitit.Add(CoduriCampuriCard.D6);   //Diagnostice
                campuriDeCitit.Add(CoduriCampuriCard.D7);   //Boli cronice
                campuriDeCitit.Add(CoduriCampuriCard.G1);   //Certificat

                //instantieaza obiectul cardData
                cardData = new CardData();

                //creeaza dictionar pentru rezultat operatie
                Dictionary<CoduriCampuriCard, CoduriRaspunsOperatieCamp> rezultatOperatie =
                    new Dictionary<CoduriCampuriCard, CoduriRaspunsOperatieCamp>();

                //se apeleaza metoda de citire de pe card
                int rez = SesiuneCard.CitesteDate(this.Token, campuriDeCitit, ref cardData, ref
rezultatOperatie);

                AfiseazaRezultatOperatie(rezultatOperatie);
                if (rez == (int)CoduriRaspunsOperatieCard.OK)
                {
                    //afiseaza datele in controale
                    AfiseazaDateCitite(cardData);
                }

                //se afiseaza mesajul corespunzator codului de raspuns
                MessageBox.Show(MesajeRaspuns[(CoduriRaspunsOperatieCard)Enum.Parse(
                    typeof(CoduriRaspunsOperatieCard), rez.ToString())], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
```

4.7. Scriere date pe card

Semnatura metoda

```
/// <summary>
/// Editeaza date pe card (doar in mod online)
/// </summary>
/// <param name="token">Token obtinut pe baza identificatorului de drepturi</param>
/// <param name="cid">Numar de asigurat</param>
/// <param name="campuriDeEditat">Lista campuri de editat pe CIP</param>
/// <param name="cardData">Obiect card populat cu datele editate</param>
/// <param name="rezultatOperatie">Dictionar care contine perechi cod camp-cod raspuns operatie camp, indica
daca operatia a reusit sau nu pentru un anumit camp</param>
/// <returns>Intoarce 0 pentru operatie cu succes sau cod eroare in caz de esec</returns>
int EditeazaDate(string token, string cid, List<CoduriCampuriCard> campuriDeEditat, ref CardData cardData,
ref Dictionary<CoduriCampuriCard, CoduriRaspunsOperatieCamp> rezultatOperatie);
```

Exemplu de implementare

```
      private void btnScrieDate_Click(object sender, EventArgs e)
        {
            //citeste cid
            string cid = "1234";

            try
            {
                //se creeaza cardData si se seteaza valorile editate
                CardData cardData = new CardData();

                //se creeaza lista de campuri de editat
```

```
                List<CoduriCampuriCard> campuriDeEditat = new List<CoduriCampuriCard>();

                if (checkBoxNumeMF.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.C4);   //Nume medic familie
                    if (textBoxNumeMF.Text.Trim().Length > 0)
                    {
                        cardData.DateAdministrative.NumeMedicFamilie.Valoare = textBoxNumeMF.Text;
                    }
                }

                if (checkBoxPrenumeMF.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.C5);   //Prenume medic familie

                    if (textBoxPrenumeMF.Text.Trim().Length > 0)
                    {
                        cardData.DateAdministrative.PrenumeMedicFamilie.Valoare = textBoxPrenumeMF.Text;
                    }
                }

                if (checkBoxCodMF.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.C6);   //Cod medic familie

                    if (textBoxCodMF.Text.Trim().Length > 0)
                    {
                        cardData.DateAdministrative.CodMedicFamilie.Valoare = textBoxCodMF.Text;
                    }
                }

                if (checkBoxTelefonMF.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.C7);   //Telefon medic familie

                    if (textBoxTelefonMF.Text.Trim().Length > 0)
                    {
                        cardData.DateAdministrative.TelefonMedicFamilie.Valoare = textBoxTelefonMF.Text;
                    }
                }


                cardData.DateAdministrative.PersoaneContact.Valoare = new List<PersoanaContact>();
                if (checkBoxPersoaneContact.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.C8);              //Persoane contact
                    PersoanaContact persoanaContact = new PersoanaContact(textBoxNumePC1.Text,
textBoxTelefonPC1.Text);
                    (cardData.DateAdministrative.PersoaneContact.Valoare as
List<PersoanaContact>).Add(persoanaContact);

                    persoanaContact = new PersoanaContact(textBoxNumePC2.Text, textBoxTelefonPC2.Text);
                    (cardData.DateAdministrative.PersoaneContact.Valoare as
List<PersoanaContact>).Add(persoanaContact);


                }

                if (checkBoxGrupaSanguina.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.D1);   //Grupa sanguina

                    if (textBoxGrupaSanguina.Text.Trim().Length > 0)
                    {
                        cardData.DateClinicePrimare.GrupaSanguina.Valoare = textBoxGrupaSanguina.Text;
                    }
                }

                if (checkBoxRH.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.D2);   //RH

                    if (textBoxRH.Text.Trim().Length > 0)
                    {
                        cardData.DateClinicePrimare.RH.Valoare = textBoxRH.Text;
                    }
```

```
                }

                if (checkBoxStatusDonatorOrgane.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.D4);   //Status donator organe

                    if (textBoxStatusDonatorOrgane.Text.Trim().Length > 0)
                    {
                        cardData.DateClinicePrimare.StatusDonatorOrgane.Valoare =
textBoxStatusDonatorOrgane.Text;
                    }
                }

                if (checkBoxDiagnostice.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.D6);   //Diagnostice

                    List<string> diagnostice = new List<string>();
                    string[] dg = new string[checkedListBoxDiagnostice.CheckedItems.Count];
                    checkedListBoxDiagnostice.CheckedItems.CopyTo(dg, 0);
                    diagnostice.AddRange(dg);

                    cardData.DateClinicePrimare.DiagnosticeMedicaleCuRiscVital.Valoare = diagnostice;
                }

                if (checkBoxBoliCronice.Checked)
                {
                    campuriDeEditat.Add(CoduriCampuriCard.D7);   //Boli cronice

                    List<string> boli = new List<string>();
                    string[] bl = new string[checkedListBoxBoliCronice.CheckedItems.Count];
                    checkedListBoxBoliCronice.CheckedItems.CopyTo(bl, 0);
                    boli.AddRange(bl);

                    cardData.DateClinicePrimare.BoliCronice.Valoare = boli;
                }


                //se creeaza dictionar pentru rezultat operatie
                Dictionary<CoduriCampuriCard, CoduriRaspunsOperatieCamp> rezultatOperatie =
                    new Dictionary<CoduriCampuriCard, CoduriRaspunsOperatieCamp>();

                //se apeleaza metoda de scriere pe card
                int rez = SesiuneCard.EditeazaDate(this.Token, cid, campuriDeEditat, ref cardData, ref
rezultatOperatie);

                AfiseazaRezultatOperatie(rezultatOperatie);
                if (rez == (int)CoduriRaspunsOperatieCard.OK)
                {
                    ReseteazaStareControale();
                }

                //se afiseaza mesajul corespunzator codului de raspuns
                MessageBox.Show(MesajeRaspuns[(CoduriRaspunsOperatieCard)Enum.Parse(
                    typeof(CoduriRaspunsOperatieCard), rez.ToString())], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);



            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

4.8. Semnatura digitala

Semnaturi metode

```
/// <summary>
/// Computes the hash value for the specified byte array.
/// </summary>
```

```
/// <param name="buffer">A byte array input to compute the hash code for.</param>
/// <returns>The computed hash code.</returns>
byte[] ComputeHash(byte[] buffer);

/// <summary>
/// Computes the hash value for the specified byte array.
/// </summary>
/// <param name="buffer">A byte array input to compute the hash code for.</param>
/// <param name="offset">The offset into the byte array from which to begin using data.</param>
/// <param name="count">The number of bytes in the array to use as data.</param>
/// <returns>The computed hash code.</returns>
byte[] ComputeHash(byte[] buffer, int offset, int count);
```

Exemple de implementare

```
  private void buttonSemnaturaDigitala1_Click(object sender, EventArgs e)
        {
            try
            {
                byte[] data = new byte[100];
                Random random = new Random();
                random.NextBytes(data);

                byte[] dateSemnate = this.SesiuneCard.ComputeHash(data);

                textBoxDate.Text = Convert.ToBase64String(data);
                textBoxDateSemnate.Text = Convert.ToBase64String(dateSemnate);


                MessageBox.Show(MesajeRaspuns[CoduriRaspunsOperatieCard.OK], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);



            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }

    private void buttonSemnaturaDigitala2_Click(object sender, EventArgs e)
        {
            try
            {
                byte[] data = new byte[100];
                Random random = new Random();
                random.NextBytes(data);

                byte[] dateSemnate = this.SesiuneCard.ComputeHash(data, 50, 50);

                textBoxDate.Text = Convert.ToBase64String(data);
                textBoxDateSemnate.Text = Convert.ToBase64String(dateSemnate);

                MessageBox.Show(MesajeRaspuns[CoduriRaspunsOperatieCard.OK], this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
```

4.9. Eveniment schimbare stare card in terminal

Subscriere la eveniment

```
            //subscrie la evenimentul de schimbare stare card in terminal
```

```
            SesiuneCard.StareCardInTerminalSchimbata += new
StareCardInTerminalSchimbataEventHandler(SesiuneCard_StareCardInTerminalSchimbata);
```

Exemplu de implementare

```
  void SesiuneCard_StareCardInTerminalSchimbata(StareCardInTerminal stareCardInTerminal)
        {
            if (textBoxStareCardInTerminal.InvokeRequired)
            {
                this.Invoke(new RefreshStareCardInTerminalDelegate(RefreshStareCardInTerminal),
stareCardInTerminal);
                return;
            }

            RefreshStareCardInTerminal(stareCardInTerminal);
        }

  private void RefreshStareCardInTerminal(StareCardInTerminal stareCardInTerminal)
        {
            string status = (stareCardInTerminal == StareCardInTerminal.CardInserat) ? "Card inserat" : "Card
retras";

            textBoxStareCardInTerminal.Text = status;

            if (stareCardInTerminal == StareCardInTerminal.CardRetras)
            {
                textBoxToken.Text = SesiuneCard.Token;
                textBoxStareAutentificare.Clear();
                textBoxStareCard.Clear();
                textBoxIncercariRamase.Clear();
                textBoxStareComunicatieCuUM.Clear();

                textBoxNume.Clear();
                textBoxPrenume.Clear();
                textBoxVersiuneCIP.Clear();
                textBoxDataNasterii.Clear();
                textBoxNumarAsigurat.Clear();
                textBoxCNP.Clear();
                textBoxNumarCard.Clear();
                textBoxCertificat.Clear();
                textBoxRezultatOperatie.Clear();
                textBoxDate.Clear();
                textBoxDateSemnate.Clear();
                textBoxNecesitaActualizare.Clear();

                ReseteazaStareControale();
            }
            else
            {
                string stareAutentificareAfisata = (SesiuneCard.StareAutentificare ==
StariAutentificare.NEAUTENTIFICAT||
                    SesiuneCard.StareAutentificare==StariAutentificare.AUTENTIFICARE_ESUATA) ?
                 "Neautentificat" : (SesiuneCard.StareAutentificare ==
StariAutentificare.AUTENTIFICAT_OFFILINE) ?
                 "Autentificat offline" : "Autentificat";

                textBoxStareAutentificare.Text = stareAutentificareAfisata;
                textBoxStareCard.Text = this.SesiuneCard.StareCard.ToString();

                ObtineToken();
            }
        }
```

4.10. Eveniment schimbare stare card

Subscriere la eveniment

```
//subscrie la evenimentul de schimbare stare card
SesiuneCard.StareCardSchimbata += new StareCardSchimbataEventHandler(SesiuneCard_StareCardSchimbata);
```

Exemplu de implementare

```
   void SesiuneCard_StareCardSchimbata(StareCard stareCard)
        {
            if (textBoxStareCard.InvokeRequired)
            {
                this.Invoke(new RefreshStareCardDelegate(RefreshStareCard),stareCard);
                return;
            }

            RefreshStareCard(stareCard);
        }

  private void RefreshStareCard(StareCard stareCard)
        {
            this.textBoxStareCard.Text = stareCard.ToString();
            this.textBoxIncercariRamase.Text = SesiuneCard.NumarIncercariRamase.ToString();
        }
```

4.11. Eveniment schimbare stare autentificare

Subscriere la eveniment

```
//subscrie la evenimentul de schimbare stare autentificare
SesiuneCard.StareAutentificareSchimbata += new
StareAutentificareSchimbataEventHandler(SesiuneCard_StareAutentificareSchimbata);
```

Exemplu de implementare

```
    void SesiuneCard_StareAutentificareSchimbata(StariAutentificare stareAutentificare)
        {
            if (textBoxStareAutentificare.InvokeRequired)
            {
                this.Invoke(new RefreshStareAutentificareDelegate(RefreshStareAutentificare),
stareAutentificare);
                return;
            }

            RefreshStareAutentificare(stareAutentificare);
        }

private void RefreshStareAutentificare(StariAutentificare stareAutentificare)
        {
            string stareAutentificareAfisata = (stareAutentificare == StariAutentificare.NEAUTENTIFICAT||
                stareAutentificare==StariAutentificare.AUTENTIFICARE_ESUATA) ?
                "Neautentificat" : (stareAutentificare == StariAutentificare.AUTENTIFICAT_OFFILINE) ?
                "Autentificat offline" : "Autentificat";

            textBoxStareAutentificare.Text = stareAutentificareAfisata;
            textBoxIncercariRamase.Text = Convert.ToString(SesiuneCard.NumarIncercariRamase);
            textBoxNecesitaActualizare.Text = (SesiuneCard.NecesitaActualizare) ? "Da" : "Nu";
       }
```

4.12. Eveniment schimbare stare comunicatie cu UM

Subscriere la eveniment

```
            //subscrie la evenimentul de schimbare stare comunicatie cu UM
            SesiuneCard.StareComunicatieCuUMSchimbata += new
StareComunicatieCuUMSchimbataEventHandler(SesiuneCard_StareComunicatieCuUMSchimbata);
```

Exemplu de implementare

```
   void SesiuneCard_StareComunicatieCuUMSchimbata(StareComunicatieCuUM stareComunicatieCuUM)
        {
            if (textBoxStareComunicatieCuUM.InvokeRequired)
```

```
            {
                this.Invoke(new RefreshStareComunicatieCuUMDelegate(RefreshStareComunicatieCuUM),
stareComunicatieCuUM);
                return;
            }

            RefreshStareComunicatieCuUM(stareComunicatieCuUM);
        }

  private void RefreshStareComunicatieCuUM(StareComunicatieCuUM stareComunicatieCuUM)
        {
            this.textBoxStareComunicatieCuUM.Text = stareComunicatieCuUM.ToString();
        }
```

4.13. Eveniment schimbare stare editare pe terminal

Subscriere la eveniment

```
            //subscrie la evenimentul de schimbare stare editare pe terminal
            SesiuneCard.StareEditarePeTerminalSchimbata += new
StareEditarePeTerminalSchimbataEventHandler(SesiuneCard_StareEditarePeTerminalSchimbata);
```

Exemplu de implementare

```
  void SesiuneCard_StareEditarePeTerminalSchimbata(ModEditarePeTerminal editarePeTerminal)
        {
            if (labelEditarePeTerminal.InvokeRequired)
            {
                this.Invoke(new RefreshStareEditarePeTerminalDelegate(RefreshStareEditarePeTerminal),
editarePeTerminal);
                return;
            }

            RefreshStareEditarePeTerminal(editarePeTerminal);
        }

    private void RefreshStareEditarePeTerminal(ModEditarePeTerminal editarePeTerminal)
        {
            string operatie=string.Empty;

            if (editarePeTerminal == ModEditarePeTerminal.SCHIMBARE_PIN)
            {
                operatie= "Schimbare PIN";
            }
            if (editarePeTerminal == ModEditarePeTerminal.SCHIMBARE_PIN_TRANSPORT)
            {
                operatie = "Schimbare PIN Transport";
            }
            else if (editarePeTerminal == ModEditarePeTerminal.RESETARE_PIN)
            {
                operatie = "Resetare PIN";
            }
            else if (editarePeTerminal == ModEditarePeTerminal.AUTENTIFICARE)
            {
                operatie = "Autentificare";
            }
            string msg = string.Format("Editare pe terminal: {0}", operatie);
            labelEditarePeTerminal.Text = msg;

            labelEditarePeTerminal.Visible =
(editarePeTerminal!=ModEditarePeTerminal.NU_SE_EDITEAZA_PE_TERMINAL);
            labelEditarePeTerminal.Refresh();
        }
```

4.14. Eveniment dupa schimbarea starii cardului in terminal

Subscriere la eveniment

```
            //subscrie la evenimentul dupa schimbarea starii cardului in terminal
            SesiuneCard.DupaStareCardInTerminalSchimbata += new
DupaStareCardInTerminalSchimbataEventHandler(SesiuneCard_DupaStareCardInTerminalSchimbata);
```

Exemplu de implementare

```
private void DupaSchimbareStareCardInTerminal(StareCardInTerminal stareCardInTerminal,
CoduriRaspunsOperatieCard raspunsOperatieCard)
        {

            if ((stareCardInTerminal==StareCardInTerminal.CardInserat )
                && (raspunsOperatieCard != CoduriRaspunsOperatieCard.OK))
            {
                MessageBox.Show(MesajeRaspuns[(CoduriRaspunsOperatieCard)Enum.Parse(
                     typeof(CoduriRaspunsOperatieCard), raspunsOperatieCard.ToString())], this.Text,
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
```

4.15. Eveniment solicitare PIN de la PC

Pentru cititoarele de carduri care nu dispun de tastatura, SDK-ul solicita PIN-ul prin evenimentul **UserInputRequired**. Aplicatia apelanta trebuie sa subscribe la acest eveniment si sa deschida o fereastra care solicita posesorului de card sa introduca PIN-ul. Evenimentul este declansat doar atunci cand este necesara introducerea unui cod PIN (ex: la autentificare, schimbare PIN, resetare PIN).

Subscriere la eveniment

```
        // Adaug metoda de citire pin
        SesiuneCard.UserInputRequired += SesiuneCard_UserInputRequired;
```

Exemplu de implementare

```
private void SesiuneCard_UserInputRequired(object sender, ReadCardTextEventArgs e)
{
        using (FormUserInput form = new FormUserInput())
        {
                form.Prompt = e.Prompt;
                form.Password = e.Password;

                if (form.ShowDialog() != System.Windows.Forms.DialogResult.OK)
                        e.Cancel = true;
                else
                        e.Result = form.Result;
        }
}
```

4.16. Terminarea unei sesiuni de lucru cu terminalul

Apel metoda

```
SesiuneCard.Stop();
```

Exemplu de implementare

```
  private void OperatiiCardForm_FormClosing(object sender, FormClosingEventArgs e)
```

```
        {
SesiuneCard.Stop();
        }
```

4.17. Coduri de raspuns operatie card

```
    /// <summary>
    /// Enumerator coduri raspuns pentru operatii cu cardul
    /// </summary>
    public enum CoduriRaspunsOperatieCard
    {
        /// <summary>
        /// Operatie executata cu succes
        /// </summary>
        OK = 0,
        /// <summary>
        /// Token lipsa
        /// </summary>
        ERR_TOKEN_LIPSA = -1,
        /// <summary>
        /// Token invalid
        /// </summary>
        ERR_TOKEN_INVALID = -2,
        /// <summary>
        /// Serviciul de comunicatie SDK-UM nu functioneaza
        /// </summary>
        ERR_COM_SERVICE = -3,
        /// <summary>
        /// Eroare handshake raportata de unitatea de management
        /// </summary>
        ERR_HANDSHAKE = -4,
        /// <summary>
        /// Stare card invalida pentru a exacuta operatia in UM
        /// </summary>
        ERR_UM_STARE_CARD_INVALIDA = -5,
        /// <summary>
        /// Eroare la scriere pe card
        /// </summary>
        ERR_CARD_SCRIERE = -6,
        /// <summary>
        /// Eroare la citire de pe card
        /// </summary>
        ERR_CARD_CITIRE = -7,
        /// <summary>
        /// Card lipsa din terminal
        /// </summary>
        ERR_CARD_LIPSA = -8,
        /// <summary>
        /// Eroare terminal deconectat
        /// </summary>
        ERR_TERMINAL_DECONECTAT = -9,
        /// <summary>
        /// Eroare mai mult de 1 terminal conectat la pc
        /// </summary>
        ERR_TERMINAL_MAI_MULT_DE_1 = -10,
        /// <summary>
        /// Eroare autentificare
        /// </summary>
        ERR_AUTENTIFICARE = -11,
        /// <summary>
        /// Token resetat ca urmare a scoaterii cardului din terminal
        /// </summary>
        ERR_TOKEN_RESETAT = -12,
        /// <summary>
        /// Terminal invalid (neinrolat)
        /// </summary>
        ERR_INVALID_TERMINAL = -13,
        /// <summary>
        /// Invalid pin
        /// </summary>
        ERR_INVALID_PIN = -14,
```

```
        /// <summary>
        /// Card blocat
        /// </summary>
        ERR_CARD_BLOCKED = -15,
        /// <summary>
        /// UM Time out
        /// </summary>
        ERR_UM_TIME_OUT = -16,
        /// <summary>
        /// Card invalid
        /// </summary>
        ERR_INVALID_CARD = -17,
        /// <summary>
        /// Cardul este deja activat
        /// </summary>
        ERR_CARD_ALREADY_ACTIVATED = -18,
        /// <summary>
        /// Eroare generala la activarea cardului
        /// </summary>
        ERR_CARD_ACTIVARE = -19,
        /// <summary>
        /// Abandon operratie activare card
        /// </summary>
        ERR_CARD_ACTIVARE_ABANDON = -20,
        /// <summary>
        /// Cardul nu este activat
        /// </summary>
        ERR_CARD_NEACTIVAT = -21,
        /// <summary>
        ///Eroare la schimbare pin
        /// </summary>
        ERR_SCHIMBARE_PIN = -22,
        /// <summary>
        /// Abandon la schimbare PIN
        /// </summary>
        ERR_SCHIMBARE_PIN_ABANDON = -23,
        /// <summary>
        /// Eroare de sistem raportata de unitatea de management
        /// </summary>
        ERR_UM_SYSTEM_ERROR = -24,
        /// <summary>
        /// Eroare mesaj receptionat format invalid raportata de unitatea de management
        /// </summary>
        ERR_UM_MESAJ_FORMAT_INVALID = -26,
        /// <summary>
        /// Eroare la obtinerea tokenului raportata de unitatea de management
        /// </summary>
        ERR_UM_TOKEN = -27,
        /// <summary>
        /// Operatie de autentificare abandonata
        /// </summary>
        ERR_AUTENTIFICARE_ABANDON = -28,
        /// <summary>
        /// Unitatea de management indisponibila
        /// </summary>
        ERR_UM_INDISPONIBILA = -29,
        /// <summary>
        /// Depasire numar maxim persoane contact
        /// </summary>
        ERR_CARD_PERSOANE_CONTACT_PESTE_MAX = -30,
        /// <summary>
        /// Depasire numar maxim diagnostice
        /// </summary>
        ERR_CARD_DIAGNOSTICE_PESTE_MAX = -31,
        /// <summary>
        /// Depasire numar maxim boli
        /// </summary>
        ERR_CARD_BOLI_PESTE_MAX = -32,
        /// <summary>
        /// Eroare accesare card
        /// </summary>
        ERR_CARD_ACCESARE = -33,
        /// <summary>
        /// Eroare executie comanda APDU
        /// </summary>
        ERR_CARD_EXECUTIE_APDU = -34,
```

```
        /// <summary>
        /// Eroare generica resetare pin
        /// </summary>
        ERR_RESETARE_PIN = -35,
        /// <summary>
        /// Reset pin neconfirmat
        /// </summary>
        ERR_RESETARE_PIN_NECONFIRMAT = -36,
        /// <summary>
        /// Eroare cand se incearca reset pin pe un card neblocat
        /// </summary>
        ERR_RESETARE_PIN_CARD_NEBLOCAT = -37,
        /// <summary>
        /// Abandon operatie resetare pin
        /// </summary>
        ERR_RESETARE_PIN_ABANDON = -38,
        /// <summary>
        /// Nu are drepturi pentru resetare pin
        /// </summary>
        ERR_RESETARE_PIN_DREPTURI_INSUFICIENTE = -39,
        /// <summary>
        /// Lungime invalida pin
        /// </summary>
        ERR_PIN_LUNGIME_INVALIDA = -40,
        /// <summary>
        /// Pin reset invalid
        /// </summary>
        ERR_PIN_RESET_INVALID = -41,
        /// <summary>
        /// PIN transport invalid
        /// </summary>
        ERR_PIN_TRANSPORT_INVALID = -42,
        /// <summary>
        /// Card neinregistrat in sistem
        /// </summary>
        ERR_CARD_NEINREGISTRAT = -43,
        /// <summary>
        /// UM procesare esuata
        /// </summary>
        ERR_UM_PROCESARE = -44,
        /// <summary>
        /// Cerere invalida catre UM
        /// </summary>
        ERR_UM_CERERE_INVALIDA = -45,
        /// <summary>
        /// Eroare autentificare UM
        /// </summary>
        ERR_UM_AUTENTIFICARE = -46,
        /// <summary>
        /// Profil invalid
        /// </summary>
        ERR_ACTIVARE_PROFIL_INVALID = -47,
        /// <summary>
        /// Drepturi insuficiente pentru schimbare PIN
        /// </summary>
        ERR_SCHIMBARE_PIN_DREPTURI_INSUFIECIENTE = -48,
        /// <summary>
        /// Tranzactie invalida
        /// </summary>
        ERR_UM_TRANZACTIE_INVALIDA = -49,
        /// <summary>
        /// Eroare la procesare raspuns din UM
        /// </summary>
        ERR_PROCESARE_RASPUNS_UM = -50,
        /// <summary>
        /// Eroare citire certificat
        /// </summary>
        ERR_CITIRE_CERTIFICAT = -51,
        /// <summary>
        /// PIN neconfirmat
        /// </summary>
        ERR_PIN_NECONFIRMAT = -52,
        /// <summary>
        /// Eroare generala la executia operatiei pe card
        /// </summary>
        ERR_OPERATIE_CARD = -53,
```

```
        /// <summary>
        /// Eroare de timeout la executia operatiei pe card
        /// </summary>
        ERR_CARD_TIMEOUT = -54,
        /// <summary>
        /// Driver duplicat pentru acelasi terminal
        /// </summary>
        ERR_CARD_TERMINAL_DUPLICAT = -55,
        /// <summary>
        /// Eroare la verificarea terminalului
        /// </summary>
        ERR_TERMINAL_VERIFICARE = -56,
        /// <summary>
        /// Eroare semnatura digitala
        /// </summary>
        ERR_SEMNATURA = -57,
        /// <summary>
        /// Eroare drepturi insuficiente pentru semnare digitala
        /// </summary>
        ERR_SEMNATURA_DREPTURI_INSUFICIENTE = -58,
        /// <summary>
        /// Eroare network la interogarea CA din UM
        /// </summary>
        ERR_UM_CA_NETWORK = -59,
        /// <summary>
        /// Eroare network la interogare eCard din UM
        /// </summary>
        ERR_UM_ECARD_NETWORK = -60,
        /// <summary>
        /// Cardul a fost schimbat in terminal
        /// </summary>
        ERR_CARD_SCHIMBAT_IN_TERMINAL = -61,
        /// <summary>
        /// Eroare la schimbare pin transport
        /// </summary>
        ERR_SCHIMBARE_PIN_TRANSPORT = -62,
        /// <summary>
        /// Eroare scriere cu rollback esuat
        /// </summary>
        ERR_CARD_SCRIERE_ROLLBACK = -63,
        /// <summary>
        /// Eroare verificare card activat
        /// </summary>
        ERR_VERIFICARE_CARD_ACTIVAT = -64,
        /// <summary>
        /// Eroare generica operatie pe terminal
        /// </summary>
        ERR_OPERATIE_TERMINAL = -65,
        /// <summary>
        /// Eroare la detectarea starii cardului in terminal
        /// </summary>
        ERR_DETECTARE_STARE_CARD_IN_TERMINAL = -66,
        /// <summary>
        /// Eroare la afisarea mesajului pe terminal
        /// </summary>
        ERR_TERMINAL_AFISARE_MESAJ = -67,
        /// <summary>
        /// Eroare instalare certificat MAI
        /// </summary>
        ERR_CITIRE_CERTIFICAT_MAI = -68,
        /// <summary>
        /// Eroare citire fisier TECH
        /// </summary>
        ERR_CITIRE_FISIER_TECH = -69,
        /// <summary>
        /// Eroare scriere fisier TECH
        /// </summary>
        ERR_SCRIERE_FISIER_TECH = -70,
        /// <summary>
        /// Eroare citire fisier Terminal Data de la UM
        /// </summary>
        ERR_UM_TERMINAL_DATA = -71
    }
```

4.18. Mesaje de raspuns operatie card

```
/// <summary>
/// Dictionar cu mesaje posibile pentru operatiile cu cardul si terminalul
/// </summary>
public class MesajeRaspunsCard : Dictionary<CoduriRaspunsOperatieCard, string>
{
    /// <summary>
    /// Constructor clasa
    /// </summary>
    public MesajeRaspunsCard()
    {
        this.Add(CoduriRaspunsOperatieCard.OK, "Operatia s-a executat cu succes.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TOKEN_LIPSA, "Tokenul sesiunii lipseste.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TOKEN_RESETAT, "Tokenul sesiunii a fost resetat. Obtineti alt
token valid.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TOKEN_INVALID, "Tokenul sesiunii este invalid.");
        this.Add(CoduriRaspunsOperatieCard.ERR_COM_SERVICE, "Serviciul de comunicatie al SDK cu UM nu
functioneaza.");
        this.Add(CoduriRaspunsOperatieCard.ERR_HANDSHAKE, "Eroare handshake raportata de unitatea de
management");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_STARE_CARD_INVALIDA, "Operatie esuata in UM. Starea
cardului este invalida.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_LIPSA, "Operatie esuata. Cardul nu este prezent in
terminal.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_CITIRE, "Eroare in timpul operatiei de citire.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_SCRIERE, "Eroare in timpul operatiei de scriere.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TERMINAL_DECONECTAT, "Operatie esuata. Terminalul nu este
conectat la PC.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TERMINAL_MAI_MULT_DE_1, "Operatie esuata. Mai multe terminale
conectate la PC.");
         this.Add(CoduriRaspunsOperatieCard.ERR_INVALID_TERMINAL,
@"Terminalul curent nu poate fi folosit deoarece nu este inrolat.
Verificati conexiunea la Internet si parametrii de configurare ai aplicatiei si reincercati.");
 this.Add(CoduriRaspunsOperatieCard.ERR_INVALID_PIN, "PIN gresit.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_BLOCKED, "Card blocat.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_TIME_OUT, "Timpul asteptare in comunicatia cu UM a
expirat.");
        this.Add(CoduriRaspunsOperatieCard.ERR_INVALID_CARD, "Card invalid.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_ALREADY_ACTIVATED, "Cardul a fost deja activat.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_ACTIVARE, "Operatie esuata. Eroare la activare card.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_ACTIVARE_ABANDON, "Operatia de activare a cardului a fost
anulata.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_NEACTIVAT, "Cardul nu este activat.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SCHIMBARE_PIN, "Eroare la schimbarea PIN-ului.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SCHIMBARE_PIN_ABANDON, "Operatia de schimbare PIN a fost
anulata.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_SYSTEM_ERROR, "Eroare de sistem raportata de unitatea de
management.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_MESAJ_FORMAT_INVALID, "Format invalid pentru mesaj in
unitatea de management");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_TOKEN, "Eroare la obtinerea tokenului raportata de unitatea
de management.");
        this.Add(CoduriRaspunsOperatieCard.ERR_AUTENTIFICARE, "Autentificare esuata.");
        this.Add(CoduriRaspunsOperatieCard.ERR_AUTENTIFICARE_ABANDON, "Operatia de autentificare a fost
anulata.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_INDISPONIBILA, "Unitatea de management indisponibila.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_PERSOANE_CONTACT_PESTE_MAX, "Puteti adauga maxim 2
persoane contact.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_DIAGNOSTICE_PESTE_MAX, "Puteti adauga maxim 10
diagnostice.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_BOLI_PESTE_MAX, "Puteti adauga maxim 20 de boli.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_ACCESARE, "Eroare la accesarea cardului.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_EXECUTIE_APDU, "Eroare executie comanda APDU.");
        this.Add(CoduriRaspunsOperatieCard.ERR_RESETARE_PIN, "Eroare la resetarea PIN-ului.");
        this.Add(CoduriRaspunsOperatieCard.ERR_RESETARE_PIN_NECONFIRMAT, "Resetare PIN
neconfirmata.\r\nAnuntati helpdesk de blocarea cardului.");
        this.Add(CoduriRaspunsOperatieCard.ERR_RESETARE_PIN_CARD_NEBLOCAT, "Aceasta operatie se executa numai
pe un card blocat.");
        this.Add(CoduriRaspunsOperatieCard.ERR_RESETARE_PIN_ABANDON, "Operatia de resetare pin a fost
anulata.");
        this.Add(CoduriRaspunsOperatieCard.ERR_RESETARE_PIN_DREPTURI_INSUFICIENTE, "Nu aveti drepturi
suficiente pentru a reseta PIN-ul pe acest terminal.");
```

```
        this.Add(CoduriRaspunsOperatieCard.ERR_PIN_LUNGIME_INVALIDA, "PIN invalid. PIN-ul trebuie sa fie
format din 4 cifre.");
        this.Add(CoduriRaspunsOperatieCard.ERR_PIN_RESET_INVALID, "PIN de reset invalid.");
        this.Add(CoduriRaspunsOperatieCard.ERR_PIN_TRANSPORT_INVALID, "PIN de transport invalid.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_NEINREGISTRAT, "Cardul acesta nu este inregistrat in
eCard.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_PROCESARE, "UM nu poate procesa cererea.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_CERERE_INVALIDA, "Cerere invalida catre UM.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_AUTENTIFICARE, "UM autentificare esuata.");
        this.Add(CoduriRaspunsOperatieCard.ERR_ACTIVARE_PROFIL_INVALID, "Nu aveti drepturi suficiente pentru
a executa operatia de activare.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SCHIMBARE_PIN_DREPTURI_INSUFIECIENTE, "Nu aveti drepturi
suficiente pentru aceasta operatie.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_TRANZACTIE_INVALIDA, "UM tranzactie invalida.");
        this.Add(CoduriRaspunsOperatieCard.ERR_PROCESARE_RASPUNS_UM, "Eroare la procesarea raspunsului de la
UM.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CITIRE_CERTIFICAT, "Eroare la citirea certificatului.");
        this.Add(CoduriRaspunsOperatieCard.ERR_PIN_NECONFIRMAT, "PIN-ul nu se confirma.");
        this.Add(CoduriRaspunsOperatieCard.ERR_OPERATIE_CARD, "Eroare la executia operatiei pe card.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_TIMEOUT, "Eroare de timeout la executia operatiei.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_TERMINAL_DUPLICAT, "Driver duplicat pentru acelasi
terminal.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TERMINAL_VERIFICARE, "Eroare la verificarea terminalului.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SEMNATURA, "Eroare la semnarea digitala.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SEMNATURA_DREPTURI_INSUFICIENTE, "Nu aveti drepturi suficiente
pentru semnatura.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_CA_NETWORK, "Eroare la interogarea CA din UM.");
        this.Add(CoduriRaspunsOperatieCard.ERR_UM_ECARD_NETWORK, "Eroare la interogarea eCard din UM.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_SCHIMBAT_IN_TERMINAL, "Operatie esuata. Cardul a fost
schimbat in terminal.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SCHIMBARE_PIN_TRANSPORT, "Eroare la schimbarea pinului de
transport.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CARD_SCRIERE_ROLLBACK, "Eroare la scriere cu rollback
esuat.");
        this.Add(CoduriRaspunsOperatieCard.ERR_VERIFICARE_CARD_ACTIVAT, "Eroare la verificare card activ.");
        this.Add(CoduriRaspunsOperatieCard.ERR_DETECTARE_STARE_CARD_IN_TERMINAL, "Eroare la detectarea starii
cardului in terminal.");
        this.Add(CoduriRaspunsOperatieCard.ERR_TERMINAL_AFISARE_MESAJ, "Eroare la afisarea mesajului pe
terminal.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CITIRE_CERTIFICAT_MAI, "Eroare la citirea certificatului
MAI.");
        this.Add(CoduriRaspunsOperatieCard.ERR_CITIRE_FISIER_TECH, "Eroare la citirea starii de activare.");
        this.Add(CoduriRaspunsOperatieCard.ERR_SCRIERE_FISIER_TECH, "Eroare la scrierea starii de
activare.");
this.Add(CoduriRaspunsOperatieCard.ERR_UM_TERMINAL_DATA, "Eroare la obtinerea datelor de inrolare de la
unitatea de management.");
    }
}
```