

Manual do SDK – Middleware do Cartão de Cidadão

Versão 3.0



Índice

1. Histórico.....	4
2. Introdução.....	5
3. Abreviaturas e acrónimos.....	6
4. Instalação.....	7
4.1 Sistemas Operativos suportados.....	7
4.2 Linguagens de programação.....	7
4.3 Compiladores.....	7
4.4 Instalação do Middleware.....	8
4.4.1. Windows.....	8
4.4.2. Linux.....	8
4.4.3. MacOS.....	8
5. Procedimentos.....	9
5.1 Pré-condições.....	9
5.2 Inicialização / Finalização do SDK.....	9
5.3 Acesso ao <i>smartcard</i> Cartão de Cidadão.....	10
5.3.1. Eventos de inserção / remoção de cartões.....	12
5.4 Dados pessoais do cidadão.....	14
5.4.1. Obtenção da Identificação.....	14
5.4.2. Obtenção da fotografia.....	16
5.4.3. Obtenção da morada.....	17
5.4.4. Leitura e escrita das notas pessoais.....	18
5.4.5. Leitura dos dados de identidade do Cidadão e da Morada.....	19
5.4.6. Obtenção dos dados cartão em formato XML.....	22
5.5 PINs.....	23
5.5.1. Verificação e alteração do PIN.....	23
5.6 Assinatura Digital.....	24
5.6.1. Formato XML Advanced Electronic Signatures (XAdES).....	24
5.6.2. Ficheiros PDF.....	26

5.6.3. Bloco de dados.....	28
5.6.4. Multi-assinatura com uma única introdução de PIN.....	30
5.6.5. Configurar o servidor de selo temporal.....	31
5.7 Certificados digitais.....	32
5.7.1. Leitura dos certificados digitais presentes no cartão de cidadão.....	32
5.8 Sessão segura.....	33
6. Tratamento de erros.....	36
7. Compatibilidade com o SDK da versão 1.....	37
7.1 Métodos removidos.....	37
7.2 Diferenças no comportamento de alguns métodos.....	37
7.3 Códigos de Erro.....	38
8. Notas do Utilizador.....	40

1. Histórico

Versão	Autor	Descrição	Data
1.0	Luiz Lemos	Versão inicial	2017-01-27
1.1	André Guerreiro	Remover referências ao pteidlibJava_Wrapper e outras melhorias	2017-05-15
1.2	André Guerreiro	Corrigir formatação da secção 5	2017-09-11
1.3	André Guerreiro	Revisão geral e mais detalhe nas questões de compatibilidade	2017-10-02
1.4	André Guerreiro	Adicionada secção para o método SetEventCallback()	2018-02-08
1.5	Veniamin Craciun	Actualizada a lista de dependências em sistemas Ubuntu e adicionado os códigos de erro da SDK para linguagem C	2018-12-17
1.5	Veniamin Craciun	Correcção de erros ortográficos e pontuais melhorias nos exemplos	2019-01-14

2. Introdução

Este documento destina-se a programadores e analistas de sistemas que tencionam desenvolver soluções informáticas com base no SDK do middleware versão 3 do Cartão de Cidadão.

Esta versão do SDK disponibiliza a mesma interface (API) que a disponibilizada na versão 1 e 2 do SDK. Desta forma, pretende-se obter a retro-compatibilidade entre versões anteriores do SDK. Embora a API anterior continue disponível, esta é desaconselhada pois a nova API cobre a maior parte dos casos de uso da anterior e tem funcionalidades novas.

Para obter informação detalhada sobre a API do middleware da versão 1 deverá consultar o “*Manual técnico do Middleware Cartão de Cidadão*” versão 1.61¹.

Na secção 6 listamos as diferenças existentes na implementação desta API.

A secção 7 contém informações relativamente continuação da compatibilidade com a versão 1, relata as diferenças comportamentais, os métodos removidos e os códigos de erro disponíveis.

O SDK do cartão de cidadão consiste num conjunto de bibliotecas utilizadas no acesso e suporte ao cartão de cidadão. Este SDK foi desenvolvido em C++, sendo providenciado o suporte a três diferentes tipos de sistemas operativos de 32/64 bits:

- Windows;
- Linux;
- MacOS;

Através dos exemplos presentes neste documento será possível desenvolver uma aplicação simples que interaja com o Cartão de Cidadão.

O desenvolvimento aplicacional utilizando o SDK pode ser realizado em C++ ou alternativamente em Java ou C# através de *wrappers* providenciados com o SDK.

¹ Manual disponível na seguinte localização:

<https://www.autenticacao.gov.pt/documents/10179/11463/Manual+T%C3%A9cnico+do+Middleware+do+Cart%C3%A3o+de+Cidad%C3%A3o/07e69665-9f1a-41c8-b3f5-c6b2e182d697>

3. Abreviaturas e acrónimos

Acrónimos / abreviaturas	Definição
API	Application Programming Interface
SDK	Software Development Kit
PDF	Portable Document Format
XML	Extensible Markup Language
XAdES	XML Advanced Electronic Signatures
PAdES	PDF Advanced Electronic Signatures

4. Instalação

4.1 Sistemas Operativos suportados

A lista de sistemas operativos suportados, arquitecturas de 32 e 64 bits, são:

- Sistemas operativos Windows:
 - Windows 7;
 - Windows 8/8.1;
 - Windows 10
- Distribuições de Linux:
 - Ubuntu: 18.04 e superiores
 - OpenSuse: Leap 42.2
 - Fedora: 24 e superiores
- Sistemas operativos Apple MacOS:
 - Versões Yosemite (10.10) e superiores.

4.2 Linguagens de programação

A lista de linguagens de programação suportadas são:

- C++: Em Windows, Linux e MacOS;
- Java: Em Windows, Linux e MacOS;
- C#: Apenas em Windows;

4.3 Compiladores

A lista de compiladores suportados são:

- C++:
 - Windows: Visual Studio 2013
 - Linux: GCC ou LLVM (clang);
 - MacOS: Compilador distribuído pela Apple. Dependendo da versão pode ser GCC ou LLVM (clang).
- Java - Oracle JDK 8 ou superior

4.4 Instalação do Middleware

4.4.1. Windows

Para instalar o SDK basta efectuar o *download* do ficheiro MSI de instalação e executar.

As bibliotecas C++ (pteidlibCpp.lib e respectivos *header files*), Java e C# ficarão disponíveis em

C:\ Program Files\Portugal Identity Card\sdk

4.4.2. Linux

Para instalar o SDK é necessário efectuar o *download* do pacote em formato deb ou rpm conforme a distribuição Linux que utiliza.

As bibliotecas C++ (libpteidlib.so) e Java (pteidlibj.jar) estão disponíveis em **/usr/local/lib** e os respectivos C++ *header files* estão em **/usr/local/include**

Se a instalação for feita a partir do código fonte disponível em <https://svn.gov.pt/projects/ccidadao> será necessário instalar as seguintes dependências (pacotes Ubuntu 18.04, para outras distribuições Linux os nomes serão diferentes):

- | | |
|-------------------------|--------------------------------|
| ◦ libpcsc-lite-dev | ◦ zlib1g-dev |
| ◦ libpoppler-qt5-dev | ◦ libpng-dev |
| ◦ openjdk-8-jdk | ◦ libopenjp2-7-dev |
| ◦ qtbase5-dev | ◦ libzip-dev |
| ◦ qt5-qmake | ◦ qt5-default |
| ◦ qtbase5-private-dev | ◦ qtdeclarative5-dev |
| ◦ libxerces-c-dev | ◦ qtquickcontrols2-5-dev |
| ◦ libxml-security-c-dev | ◦ qml-module-qtquick-controls2 |
| ◦ swig | ◦ libssl1.0-dev |
| ◦ libcurl4-openssl-dev | |

4.4.3. MacOS

Para instalar o SDK é necessário efectuar o *download* do pacote de instalação e executar. O SDK Java ficará disponível em **/usr/local/lib/pteid_jni**. No que diz respeito ao SDK C++, os *header files* ficam localizados em **/usr/local/include** e a biblioteca à qual as aplicações deverão linkar está no caminho **/usr/local/lib/libpteidlib.dylib**

5. Procedimentos

5.1 Pré-condições

1. C/C++

Windows/Visual Studio

Adicionar a import library **pteidlibCpp.lib** ao projecto.

Por forma a conseguir incluir os *header files* do SDK adicionar a directoria “C:\Program Files\ Portugal Identity Card\sdk” nas propriedades do projecto em “C/C++” → “General” → “Additional Include Directories”

2. Java

Incluir o ficheiro **pteidlibj.jar** como biblioteca no projecto e adicionar à library path do java a localização das bibliotecas nativas do SDK (se necessário).

De notar que as classes e métodos de compatibilidade estão disponíveis no package **pteidlib** enquanto que as novas classes estão no *package* **pt.gov.cartaodecidadao**

3. C#

Adicionar a biblioteca pteidlib_dotnet.dll às *references* do projecto Visual Studio.

As classes e métodos de compatibilidade estão no namespace **eidpt** enquanto que as novas classes estão no namespace **pt.portugal.eid**

5.2 Inicialização / Finalização do SDK

A biblioteca **pteidlib** é inicializada através da invocação do método PTEID_initSDK() (não é contudo obrigatório efectuar a inicialização). A finalização do SDK (é obrigatória) deve ser efectuada através da invocação do método PTEID_releaseSDK(), a invocação deste método garante que todos os processos em segundo plano são terminados e que a memória alocada é libertada.

1. Exemplo em C++

```
#include "eidlib.h"

(...)

int main(int argc, char **argv){

    PTEID_InitSDK();

    (...)

    PTEID_ReleaseSDK();

}
```

2. Exemplo em Java

```
package pteidsample;
import pt.gov.cartaodecidadao.*;
(...)
static {
    try {
        System.loadLibrary("pteidlibj");
    } catch (UnsatisfiedLinkError e) {
        System.err.println("Native code library failed to load. \n" + e);
        System.exit(1);
    }
}
public class SamplePTEID {
    public static void main(String[] args) {
        PTEID_ReaderSet.initSDK();
        (...)
        PTEID_ReaderSet.releaseSDK();
    }
}
```

Nota: o bloco estático a **vermelho** é estritamente necessário uma vez que é preciso carregar explicitamente a biblioteca JNI que implementa as funcionalidades disponível pelo wrapper Java.

3. Exemplo em C#

```
namespace PTEIDSample {
    class Sample{
        (...)
        public static void Main(string[] args){
            PTEID_ReaderSet.initSDK();
            (...)
            PTEID_ReaderSet.releaseSDK();
        }
    }
}
```

5.3 Acesso ao *smartcard* Cartão de Cidadão

Para aceder ao Cartão de Cidadão programaticamente devem ser efectuados os seguinte passos:

- Obter a lista de leitores de *smartcards* no sistema;
- Seleccionar um leitor de *smartcards*;
- Verificar se o leitor contém um cartão;

- Obter o objecto que fornece acesso ao cartão;
- Obter o objecto que contém os dados pretendidos;

A classe `PTEID_ReaderSet` representa a lista de leitores de cartões disponíveis no sistema, esta classe disponibiliza uma variedade de métodos relativos aos leitores de cartões disponíveis. Através da lista de leitores, um leitor de cartões pode ser seleccionado resultando na criação de um objecto de contexto específico ao leitor em questão, a partir do qual é possível aceder ao cartão.

O objecto de contexto do leitor faculta o acesso ao cartão (se este estiver presente no leitor). O acesso ao cartão é obtido através do método `PTEID_ReaderContext.getEIDCard()` que devolve um objecto do tipo `PTEID_EIDCard`.

1. Exemplo C++

```
PTEID_ReaderSet& readerSet = PTEID_ReaderSet::instance();
for( int i=0; i < readerSet.readerCount(); i++){
    PTEID_ReaderContext& context = readerSet.getReaderByNum(i);
    if (context.isCardPresent()){
        PTEID_EIDCard &card = context.getEIDCard();
        (...)
    }
}
```

2. Exemplo Java

```
PTEID_EIDCard card;
PTEID_ReaderContext context;
PTEID_ReaderSet readerSet;
readerSet = PTEID_ReaderSet.instance();
for( int i=0; i < readerSet.readerCount(); i++){
    context = readerSet.getReaderByNum(i);
    if (context.isCardPresent()){
        card = context.getEIDCard();
        (...)
    }
}
```

3. Exemplo C#

```
PTEID_EIDCard card;
PTEID_ReaderContext context;
PTEID_ReaderSet readerSet;
readerSet = PTEID_ReaderSet.instance();
for( int i=0; i < readerSet.readerCount(); i++){
    context = readerSet.getReaderByNum(i);
    if (context.isCardPresent()){
        card = context.getEIDCard();
        (...)
    }
}
```

NOTA: Uma forma rápida de obter um objecto de contexto será utilizar o método `getReader()`. Este método devolve o objecto de contexto do primeiro leitor com cartão que for encontrado no sistema. Alternativamente caso não existam cartões inseridos devolverá o primeiro leitor que encontrar no sistema.

- C++

PTEID_ReaderContext

&readerContext = PTEID_ReaderSet.instance().getReader();

- Java

PTEID_ReaderContext

readerContext = PTEID_ReaderSet.instance().getReader();

- C#

PTEID_ReaderContext

readerContext = PTEID_ReaderSet.instance().getReader();

5.3.1. Eventos de inserção / remoção de cartões

O SDK oferece uma forma de obter notificações dos eventos de cartão inserido e removido através de uma função callback definida pela aplicação.

Para tal é necessário invocar o método `SetEventCallback()` no objecto `PTEID_ReaderContext` associado ao leitor que se pretende monitorizar.

A função de callback definida deve ter a seguinte assinatura:

`void callback (long lRet, unsigned long ulState, void *callbackData)` em C++

O parâmetro *ulState* é a combinação de dois valores:

- 1 - contador de eventos no leitor em causa
- 2 - flag que indica se foi inserido ou removido um cartão

O parâmetro *lRet* é um código de erro que em caso de sucesso no acesso ao leitor terá sempre o valor 0.

O parâmetro *callbackData* é uma referência/ponteiro para o objecto que terá sido associado ao callback na função `SetEventCallback()`

Exemplo Java:

```
class CardEventsCallback implements Callback {
    @Override
    public void getEvent(long lRet, long ulState, Object callbackData) {
        int cardState = (int)ulState & 0x0000FFFF;
        int eventCounter = ((int)ulState) >> 16;

        System.err.println("DEBUG: Card Event:" +
            " cardState: "+cardState + " Event Counter: "+ eventCounter);
        if ((cardState & 0x0100) != 0)
        {
            System.out.println("Card inserted");
        }
        else {
            System.out.println("Card removed");
        }
    }
}

PTEID_ReaderSet readerSet = PTEID_ReaderSet.instance();
PTEID_ReaderContext context = readerSet.getReader();
long callbackId = context.SetEventCallback(new CardEventsCallback(), null);
(...)
context.StopEventCallback(callbackId);
```

Exemplo C#:

```
public static void CardEventsCallback(int lRet, uint ulState, IntPtr callbackData) {

    uint cardState = ulState & 0x0000FFFF;
    uint eventCounter = (ulState) >> 16;

    Console.WriteLine("DEBUG: Card Event: cardState: {1} Event Counter: {2}",
cardState, eventCounter);

    if ((cardState & 0x0100) != 0) {
        Console.WriteLine("Card inserted");
    }
    else {
        Console.WriteLine("Card removed");
    }
}

PTEID_ReaderSet readerSet = PTEID_ReaderSet.instance();
IntPtr callbackData = (IntPtr)0;

PTEID_ReaderContext context = readerSet.getReader();
context.SetEventCallback(CardEventsCallback, callbackData);
```

5.4 Dados pessoais do cidadão

Os dados do cidadão e do cartão estão armazenados no cartão em múltiplos ficheiros. Destacam-se os seguintes ficheiros:

- ficheiro de identificação - contém os dados do cidadão/cartão impressos nas faces do cartão, incluindo a foto);
- ficheiro de morada - contém a morada do cidadão, este ficheiro é de acesso condicionado
- ficheiros de certificados do cidadão - contém os certificados de assinatura/autenticação do cidadão.
- ficheiros de certificados CA's.
- ficheiro de notas pessoais - é um ficheiro de leitura livre e de escrita condicionada onde o cidadão pode colocar até 1000 bytes.

5.4.1. Obtenção da Identificação

Para obter o conteúdo do ficheiro de identificação, o método `PTEID_EIDCard.getID()` deverá ser utilizado.

1. Exemplo C++

```
(...)  
PTEID_EIDCard& card = context.getIDCard();  
PTEID_EID& eid = card.getID();  
  
std::string nome = eid.getGivenName();  
std::string nrCC = eid.getDocumentNumber();  
(...)
```

2. Exemplo Java

```
(...)  
PTEID_EIDCard card = context.getIDCard();  
PTEID_EID eid = card.getID();  
  
String nome = eid.getGivenName();  
String nrCC = eid.getDocumentNumber();  
(...)
```

3. Exemplo C#

```
(...)  
PTEID_EIDCard card = context.getEIDCard();  
PTEID_EId eid = card.getID();  
string nome = eid.getGivenName();  
string nrCC = eid.getDocumentNumber();  
(...)
```

5.4.2. Obtenção da fotografia

A fotografia do cidadão está no formato jpeg2000, o SDK disponibiliza a fotografia no formato original e em formato PNG.

1. Exemplo C++

```
(...)  
PTEID_EIDCard& card = context.getEIDCard();  
PTEID_EId& eid = card.getID();  
PTEID_Photo& photoObj = eid.getPhotoObj();  
PTEID_ByteArray& praw = photoObj.getphotoRAW(); // formato jpeg2000  
PTEID_ByteArray& ppng = photoObj.getphoto();    // formato PNG  
(...)
```

2. Exemplo Java

```
(...)  
PTEID_EIDCard card = context.getEIDCard();  
PTEID_EId eid = card.getID();  
PTEID_Photo photoObj = eid.getPhotoObj();  
PTEID_ByteArray praw = photoObj.getphotoRAW(); // formato jpeg2000  
PTEID_ByteArray ppng = photoObj.getphoto();    // formato PNG  
(...)
```

3. Exemplo C#

```
(...)  
PTEID_EIDCard card = context.getEIDCard();  
PTEID_EId eid = card.getID();  
PTEID_Photo photoObj = eid.getPhotoObj();  
PTEID_ByteArray praw = photoObj.getphotoRAW(); // formato jpeg2000  
PTEID_ByteArray ppng = photoObj.getphoto();    // formato PNG  
(...)
```


5.4.3. Obtenção da morada

O ficheiro da morada só pode ser lido após a inserção do pin da morada correcto.

Para obter os dados da morada deverá ser utilizado o método `PTEID_EIDCard.getAddr()`.

1. Exemplo C++

```
PTEID_EIDCard &card;
unsigned long triesLeft;

(...)
PTEID_Pins pins = card.getPins();
PTEID_Pin pin = pins.getPinByPinRef(PTEID_Pin.ADDR_PIN);
if (pin.verifyPin("", &triesLeft, true){
    PTEID_Address &addr = card.getAddr();
    const char * municipio = addr.getMunicipality();
}
```

2. Exemplo Java

```
PTEID_EIDCard card;
PTEID_ulwrapper triesLeft = new PTEID_ulwrapper(-1);
PTEID_Address addr;
(...)
PTEID_Pins pins = card.getPins();
PTEID_Pin pin = pins.getPinByPinRef(PTEID_Pin.ADDR_PIN);
if (pin.verifyPin("", triesLeft, true){
    addr = card.getAddr();
    String municipio = addr.getMunicipality();
}
```

```
PTEID_EIDCard card;
uint triesLeft;
PTEID_Address addr;
(...)
PTEID_Pins pins = card.getPins();
PTEID_Pin pin = pins.getPinByPinRef(PTEID_Pin.ADDR_PIN);
if (pin.verifyPin("", ref triesLeft, true){
    addr = card.getAddr();
    string municipio = addr.getMunicipality();
}
```

3. Exemplo C#

5.4.4. Leitura e escrita das notas pessoais

Para ler as notas pessoais deverá ser utilizado o método `PTEID_EIDCard.readPersonalNotes()`. Para a escrita de dados deverá ser utilizado o método `PTEID_EIDCard.writePersonalNotes()`, sendo necessária a introdução do PIN de autenticação. Neste momento, as notas pessoais têm um limite de 1000 caracteres.

1. Exemplo C++

```
PTEID_EIDCard &card = readerContext.getEIDCard();

std::string notes("a few notes");

PTEID_ByteArray personalNotes(notes.c_str(), notes.size() + 1);
bool bOk;
(...)
// leitura
string pdata = card.readPersonalNotes();
// escrita
bOk = card.writePersonalNotes(personalNotes,
card.getPins().getPinByPinRef(PTEID_Pin.AUTH_PIN));
```

2. Exemplo Java

```
PTEID_EIDCard card;
(...)
// leitura
String pdata = card.readPersonalNotes();
//escrita
String notes = "a few notes";
PTEID_ByteArray pb = new PTEID_ByteArray(notes.getBytes(),
notes.getBytes().length);
boolean bOk = card.writePersonalNotes(pb,
card.getPins().getPinByPinRef(PTEID_Pin.AUTH_PIN));
(...)
```

3. Exemplo C#

```
PTEID_EIDCard card;
PTEID_ByteArray pb;
boolean bOk;
(...)
// leitura
string pdata = card.readPersonalNotes();

//escrita
bOk = card.writePersonalNotes( pb,
card.getPins().getPinByPinRef(PTEID_Pin.AUTH_PIN));
(...)
```

5.4.5. Leitura dos dados de identidade do Cidadão e da Morada

Para estes métodos das classes PTEID_Eid, PTEID_Address não apresentamos exemplos já que estes dados apenas são responsáveis pelas tarefas de obtenção dos campos específicos dentro dos ficheiros de identidade e morada e todos eles devolvem resultados do tipo String (no caso de Java/C#) ou const char * (no caso da biblioteca C++)

PTEID_Eid

Método	Descrição
getDocumentVersion()	versão do documento de identificação
getDocumentType()	tipo de documento - "Cartão de cidadão"
getCountry()	código do país no formato ISO3166
getGivenName()	nomes próprios do detentor do cartão
getSurname()	apelidos do detentor do cartão
getGender()	género do detentor do cartão
getDateOfBirth()	data de nascimento
getNationality()	nacionalidade (código do país no formato ISO3166)
getDocumentPAN()	número PAN do cartão (PAN - primary account number)
getValidityBeginDate()	data de emissão
getValidityEndDate()	data de validade
getLocalofRequest()	local de pedido do cartão
getHeight()	altura do detentor do cartão
getDocumentNumber()	número do cartão de cidadão
getCivilianIdNumber()	número de identificação civil
getTaxNo()	número de identificação fiscal
getSocialSecurityNumber()	número de segurança social
getHealthNumber()	número de utente de saúde
getIssuingEntity()	entidade emissora do cartão

Método	Descrição
getGivenNameFather()	nomes próprios do pai do detentor do cartão
getSurnameFather()	apelidos do pai do detentor do cartão
getGivenNameMother()	nomes próprios da mãe do detentor do cartão
getSurnameMother()	apelidos da mãe do detentor do cartão
getParents()	filiação do detentor do cartão sobre na forma seguinte “nome e apelido do pai * nome e apelida da mãe”
getPhotoObj()	objecto que contém a foto do detentor do cartão
getCardAuthKeyObj()	chave pública do cartão
getValidation()	indica se cartão se encontra válido
getMRZ1()	primeira linha do campo MRZ
getMRZ2()	segunda linha do campo MRZ
getMRZ3()	terceira linha do campo MRZ
getAccidentalIndications()	indicações eventuais

PTEID_Address

Método	Descrição
getCountryCode()	código do país no formato ISO3166
getDistrict()	nome do distrito
getDistrictCode()	código do distrito
getMunicipality()	nome do município
getMunicipalityCode()	código do município
getCivilParish()	nome da freguesia
getCivilParishCode()	código da freguesia
getAbbrStreetType()	abreviatura do tipo de via
getStreetType()	tipo de via
getStreetName()	nome da via

Método	Descrição
getAbbrBuildingType()	abreviatura do tipo de edifício
getBuildingType()	tipo do edifício
getDoorNo()	número da entrada
getFloor()	número do piso
getSide()	lado
getLocality()	localidade
getPlace()	lugar
getZip4()	código postal
getZip3()	código postal
getPostalLocality()	localidade postal

PTEID_Address - Apenas aplicável a moradas estrangeiras

Método	Descrição
getForeignCountry()	país
getForeignAddress()	endereço
getForeignCity()	cidade
getForeignRegion()	região
getForeignLocality()	localidade
getForeignPostalCode()	código postal

PTEID_Address - Aplicável a ambas as moradas (nacionais e estrangeiras)

Método	Descrição
getGeneratedAddressCode()	código do endereço
isNationalAddress()	valor de retorno, em booleano, indica se é uma morada nacional

5.4.6. Obtenção dos dados cartão em formato XML

Os dados do cidadão existentes no cartão podem ser extraídos em formato xml. A fotografia é retornada em base-64 no formato aberto PNG. Para além dos dados do cidadão é possível incluir também a área de notas pessoais.

1. Exemplo em C++

```
unsigned long triesLeft;
PTEID_EIDCard *card;
(...)
card->getPins().getPinByPinRef(PTEID_Pin::ADDR_PIN).verifyPin("", triesLeft, true);
PTEID_XmlUserRequestedInfo requestedInfo;
requestedInfo.add(XML_CIVIL_PARISH);
(...)
requestedInfo.add(XML_GENDER);
PTEID_CCXML_Doc &ccxml = card.getXmlCCDoc(requestedInfo);
const char * resultXml = ccxml.getCCXML();
```

2. Exemplo em Java

```
String resultXml;
PTEID_EIDCard card;
PTEID_ulwrapper triesLeft = new PTEID_ulwrapper(-1);
(...)
card.getPins().getPinByPinRef(PTEID_Pin.ADDR_PIN).verifyPin("", triesLeft, true);
PTEID_XmlUserRequestedInfo requestedInfo = new PTEID_XmlUserRequestedInfo();
requestedInfo.add(XMLUserData.XML_CIVIL_PARISH);
(...)
requestedInfo.add(XMLUserData.XML_GENDER);
PTEID_CCXML_Doc result = idCard.getXmlCCDoc(requestedInfo);
resultXml = result.getCCXML();
```

3. Exemplo em C#

```
string resultXml;
PTEID_EIDCard card;
uint triesLeft;
(...)
card.getPins().getPinByPinRef(PTEID_Pin.ADDR_PIN).verifyPin("", ref triesLeft,
true);
PTEID_XmlUserRequestedInfo requestedInfo = new PTEID_XmlUserRequestedInfo();
requestedInfo.add(XMLUserData.XML_CIVIL_PARISH);
(...)
requestedInfo.add(XMLUserData.XML_GENDER);
PTEID_CCXML_Doc result = idCard.getXmlCCDoc(requestedInfo);
resultXml = result.getCCXML();
```

5.5 PINs

5.5.1. Verificação e alteração do PIN

Para verificação do PIN deverá ser utilizado o método `verifyPin()`. Para a sua alteração, deverá ser utilizado o método `changePin()`.

1. Exemplo C++

```
PTEID_EIDCard& card;
unsigned long triesLeft;
(...)
PTEID_Pins pins = card.getPins();
PTEID_Pin pin = pins.getPinByPinRef(PTEID_Pin.ADDR_PIN);
if (pin.verifyPin("", &triesLeft, true){
    bool bResult = pin.changePin("", "", triesLeft, pin.getLabel());
    if (!bResult && -1 == triesLeft) return;
}
```

2. Exemplo Java

```
PTEID_EIDCard card;
PTEID_ulwrapper triesLeft = new PTEID_ulwrapper(-1);
(...)
PTEID_Pins pins = card.getPins();
PTEID_Pin pin = pins.getPinByPinRef(PTEID_Pin.ADDR_PIN);
if (pin.verifyPin("", triesLeft, true){
    bool bResult = pin.changePin("", "", triesLeft, pin.getLabel());
    if (!bResult && -1 == triesLeft) return;
}
```

3. Exemplo C#

```
PTEID_EIDCard card;
uint triesLeft;
(...)
PTEID_Pins pins = card.getPins();
PTEID_Pin pin = pins.getPinByPinRef(PTEID_Pin.ADDR_PIN);
if (pin.verifyPin("", ref triesLeft, true){
    bool bResult = pin.changePin("", "", triesLeft, pin.getLabel());
    if (!bResult && -1 == triesLeft) return;
}
```

5.6 Assinatura Digital

5.6.1. Formato XML Advanced Electronic Signatures (XAdES)

Esta funcionalidade permite a assinar um ou múltiplos ficheiros em qualquer formato utilizando ou não selos temporais.

Os métodos SignXades/SignXadesT produzem um ficheiro .zip que contém os ficheiros assinados e um ficheiro XML com a assinatura. O formato deste ficheiro .zip segue a norma europeia ASIC para *containers* de assinatura ²

1. Exemplo C++

```
unsigned long n_errors = 200;
char errors[n_errors];
const char *ficheiros[] = {"teste/Ficheiro1",
                           "teste/Ficheiro2",
                           "teste/Ficheiro3",
                           "teste/Ficheiro4"};

const char *destino = "teste/ficheiros_assinados.zip";
int n_paths = 4; // tamanho do array ficheiros

// assinar (1 única assinatura para todos os ficheiros)
card.SignXades( destino, ficheiros, n_paths );
(...)
// assinar com selo temporal (1 única assinatura para todos os ficheiros)
card.SignXadesT( destino, ficheiros, n_paths );
(...)
// assinar (1 única assinatura tipo A (archival) para todos os ficheiros)
card.SignXadesA( destino, ficheiros, n_paths );
(...)
```

2. Exemplo Java

² ETSI - TS 102 918 -

http://www.etsi.org/deliver/etsi_ts/102900_102999/102918/01.01.01_60/ts_102918v010101p.pdf


```
String ficheiros[] = new String[4];
ficheiros[0]="teste/Ficheiro1";
ficheiros[1]="teste/Ficheiro2";
ficheiros[2]="teste/Ficheiro3";
ficheiros[3]="teste/Ficheiro4";
String destino = "teste/ficheiros_assinados.zip";

//assinar (1 única assinatura para todos os ficheiros)
card.SignXades( destino, ficheiros, ficheiros.length );
(...)
//assinar com selo temporal (1 única assinatura para todos os ficheiros)
card.SignXadesT( destino, ficheiros, ficheiros.length );
(...)
// assinar (1 única assinatura tipo A (archival) para todos os ficheiros)
card.SignXadesA( destino, ficheiros, ficheiros.length );
(...)
```

3. Exemplo C#

```
string ficheiros[] = new string[4];
ficheiros[0]="c:\teste\Ficheiro1";
ficheiros[1]="c:\teste\Ficheiro2";
ficheiros[2]="c:\teste\Ficheiro3";
ficheiros[3]="c:\teste\Ficheiro4";
string destino = "c:\teste\ficheiros_assinados.zip";

//assinar (1 única assinatura para todos os ficheiros)
card.SignXades( destino, ficheiros, ficheiros.length );
(...)
//assinar com selo temporal (1 única assinatura para todos os ficheiros)
card.SignXadesT( destino, ficheiros, ficheiros.length );
// assinar (1 única assinatura tipo A (archival) para todos os ficheiros)
card.SignXadesA( destino, ficheiros, ficheiros.length );
(...)
```

NOTA: Alternativamente é possível assinar individualmente cada ficheiro da seguinte forma:

- Sem selo temporal
 - C++
card.SignXadesIndividual(dirDestino, ficheiros, n_paths);
 - Java/C#
card.SignXadesIndividual(dirDestino, ficheiros, ficheiros.length);

- Com selo temporal
 - C++
`card.SignXadesTIndividual(dirDestino, ficheiros, n_paths);`
 - Java/C#
`card.SignXadesTIndividual(dirDestino, ficheiros, ficheiros.length);`

O parâmetro **dirDestino** contém a directoria destino onde serão colocados os ficheiros assinados.

5.6.2. Ficheiros PDF

O SDK fornece métodos para assinatura de ficheiros PDF de acordo com os standards PAdES (ETSI TS 102 778-1) e com o standard mais antigo implementado pelo Adobe Reader e Acrobat (*ISO 32000*)

As assinaturas produzidas pelas bibliotecas do SDK podem ser validadas com os referidos produtos da Adobe ou alternativas *opensource* como a biblioteca iText (<http://itextpdf.com>)

Os métodos de assinatura de PDF fornecem as seguintes opções:

- Assinatura com *timestamp* de modo a garantir que a validade da assinatura não se limita à validade do certificado do Cartão de Cidadão
- Assinatura de vários ficheiros em *batch* (com apenas uma introdução de PIN)
- Inclusão de detalhes adicionais como a localização ou motivo da assinatura
- Personalização do aspecto da assinatura no documento (página, localização na mesma e tamanho da assinatura)

Quanto à localização da assinatura estão disponíveis duas abordagens:

1. Definir a localização indicando um sector assumindo que a página estão dividida em grelha de rectângulos. Neste caso a localização assume uma página de tamanho A4 em formato vertical ou horizontal.

Apresentam-se a seguir as grelhas que são assumidas para páginas A4 em formato horizontal ou vertical:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18

- Definindo a localização precisa do canto superior esquerdo do rectângulo de assinatura através de coordenadas (x,y) expressas em percentagem da largura/altura da página em que o ponto (0,0) se situa no canto superior esquerdo da página. De notar que usando este método existem localizações que produzem uma assinatura truncada na página já que o método de assinatura não valida se a localização é válida para o “selo de assinatura” a apresentar.

Será apresentado apenas um exemplo C++ para esta funcionalidade embora os wrappers Java e C# contenham exactamente as mesmas classes e métodos necessários `PTEID_PdfSignature()` e `PTEID_EIDCard.SignPDF()`

Exemplo C++:

```
#include "eidlib.h"
(...)
PTEID_EIDCard &card = readerContext.getEIDCard();
//Ficheiro PDF a assinar
PTEID_PDFSignature signature("/home/user/input.pdf");
signature.enableSmallSignatureFormat();
//Assinatura com selo temporal
signature.enableTimestamp();

// Adicionar uma imagem customizada à assinatura visível
// O pointer image_data deve apontar para uma imagem em formato JPEG de dimensões
recomendadas (185x41 px)
signature.setCustomImage(unsigned char *image_data, unsigned long image_length);

//Assinatura utilizando localização por sector.
// É necessário o parâmetro is_landscape para indicar que grelha de sectores
pretendemos utilizar
//Numero de sector, ver as grelhas apresentadas acima
int sector = 1;
int page = 1;
bool is_landscape = false;
const char * location = "Lisboa, Portugal";
const char * reason = "Concordo com o conteúdo do documento";

//No caso de assinatura em batch este parâmetro deve apontar para a directoria de
destino
const char * output = "/home/user/destination_directory";
card.SignPDF(signature, page, sector, is_landscape, location, reason,
output_file);

//Assinatura utilizando localização precisa: os parâmetros pos_x e pos_y indicam
a localização em percentagem da largura e altura da página
double pos_x = 0.1; //Para páginas A4 verticais este valor pode variar no
intervalo [0-1]
double pos_y = 0.1; //Para páginas A4 verticais este valor pode variar no
intervalo [0-1]
card.SignPDF(signature, page, pos_x, pos_y, location, reason, output_file);
```

5.6.3. Bloco de dados

Esta funcionalidade permite assinar um bloco de dados usando ou não o certificado de assinatura.

Para isso deverá ser utilizado o método Sign() da classe PTEID_EIDCard.

O Algoritmo de assinatura suportado é o RSA-SHA256 mas o *smartcard* apenas implementa o algoritmo RSA e como tal o bloco de input deve ser o *hash* SHA-256 dos dados que se pretende assinar.

1. Exemplo C++

```
PTEID_ByteArray data_to_sign;
(...)
PTEID_EIDCard &card = readerContext.getEIDCard();
(...)
PTEID_ByteArray output = card.Sign(data_to_sign, true);
(...)
```

2. Exemplo Java

```
PTEID_ByteArray data_to_sign;
(...)
PTEID_EIDCard card = context.getEIDCard();
(...)
PTEID_ByteArray output= card.Sign(data_to_sign, true);
(...)
```

3. Exemplo C#

```
PTEID_ByteArray data_to_sign, output;
(...)
PTEID_EIDCard card = readerContext.getEIDCard();
PTEID_ByteArray output;
output = card.Sign(data_to_sign, true);
(...)
```

5.6.4. Multi-assinatura com uma única introdução de PIN

Esta funcionalidade permite assinar vários ficheiros introduzindo o PIN somente uma vez. Deverá ser utilizado o método `addToBatchSigning()`.

Será apresentado apenas um exemplo C++ para esta funcionalidade embora os wrappers Java e C# contenham exactamente as mesmas classes e métodos necessários `PTEID_PDFSignature()`.

Exemplo C++

```
#include "eidlib.h"
(...)
PTEID_EIDCard &card = readerContext.getEIDCard();
//Ficheiro PDF a assinar
PTEID_PDFSignature signature("/home/user/input.pdf");

//Para realizar uma assinatura em batch adicionar todos os ficheiros
usando o seguinte método antes de invocar o card.SignPDF()
signature.addToBatchSigning( "Other_File.pdf" );
signature.addToBatchSigning( "Yet_Another_FILE.pdf" );
(...)
int sector = 1;
int page = 1;
bool is_landscape = false;
const char * location = "Lisboa, Portugal";
const char * reason = "Concordo com o conteúdo do documento";

//Para uma assinatura em batch, este parâmetro aponta para a directoria
de destino
const char * output = "/home/user/output_signed.pdf";
card.SignPDF(signature, page, sector, is_landscape, location, reason,
output_file);
(...)
```

5.6.5. Configurar o servidor de selo temporal

O SDK permite seleccionar um servidor diferente para a obtenção de selos temporais, uma vez que o servidor por defeito do Cartão do Cidadão (<http://ts.cartaodecidadao.pt/tsa/server>) tem um limite máximo de 20 pedidos em cada período de 20 minutos que se podem efectuar (para mais informações sobre o serviço de selo temporal/timestamps do Cartão do Cidadão, consulte a página <https://pki.cartaodecidadao.pt/>).

Para usar um servidor diferente basta criar uma nova configuração, da seguinte forma

```
PTEID_Config config = new PTEID_Config(PTEID_PARAM_XSIGN_TSAURL);  
config.setString("http://sha256timestamp.ws.symantec.com/sha256/timestamp");
```

Após esta configuração tanto as assinaturas de documentos PDF (PAdES) bem como as assinaturas em formato XAdES vão usar este novo servidor configurado para obter os selos temporais ao assinar.

5.7 Certificados digitais

5.7.1. Leitura dos certificados digitais presentes no cartão de cidadão

Para a obtenção do certificado *root* , deverá ser utilizado o método `getRoot()`.

Para a obtenção do certificado *CA*, deverá ser utilizado o método `getCA()`.

Para a obtenção do certificado *de assinatura* , deverá ser utilizado o método `getSignature()`.

Para a obtenção do certificado *de autenticação* , deverá ser utilizado o método `getAuthentication()`.

1. Exemplo C++

```
PTEID_EIDCard &card = readerContext.getEIDCard();
// Get the root certificate from the card
PTEID_Certificate &root=card.getRoot();

// Get the ca certificate from the card
PTEID_Certificate &ca=card.getCA();

// Get the signature certificate from the card
PTEID_Certificate &signature=card.getSignature();

// Get the authentication certificate from the card
PTEID_Certificate &authentication=card.getAuthentication();
```

2. Exemplo Java

```
PTEID_EIDCard card = context.getEIDCard();
// Get the root certificate from the card
PTEID_Certificate root=card.getRoot();
// Get the ca certificate from the card
PTEID_Certificate ca=card.getCA();

// Get the signature certificate from the card
PTEID_Certificate signature=card.getSignature();

// Get the authentication certificate from the card
PTEID_Certificate authentication=card.getAuthentication();
```

3. Exemplo C#


```
PTEID_EIDCard card = context.getEIDCard();
PTEID_Eid eid = card.getID();

// Get the root certificate from the card
PTEID_Certificate root=card.getRoot();

// Get the ca certificate from the card
PTEID_Certificate ca=card.getCA();

// Get the signature certificate from the card
PTEID_Certificate signature=card.getSignature();
// Get the authentication certificate from the card
PTEID_Certificate authentication=card.getAuthentication();
```

5.8 Sessão segura

O Cartão de Cidadão permite o estabelecimento de sessões seguras. É efectuada a autenticação entre ambas as partes (a aplicação e o cartão). Após este processo as operações seguintes são efectuadas sobre comunicação cifrada e autenticada.

A autenticação da aplicação é efectuada através de CVCs (Card Verifiable Certificates). Estes certificados são emitidos somente a entidades que estejam autorizadas em Lei a efectuar operações privilegiadas no cartão.

Existem duas operações privilegiadas que obrigam ao estabelecimento prévio de uma sessão segura:

- Leitura da morada sem introdução de PIN.
- Alteração da morada.

1. Exemplo em C para a leitura da morada sem introdução do PIN (utilizando a biblioteca OpenSSL para implementar a assinatura do desafio enviado pelo cartão).

Foram omitidos do bloco de código seguinte as declarações de *#include* necessárias para utilizar as funções do OpenSSL. Para mais informações sobre OpenSSL, consultar a wiki do projecto em: <https://wiki.openssl.org/>

Esta funcionalidade está apenas disponível nos métodos de compatibilidade com a versão 1 do Middleware.

Em seguida é apresentado o exemplo em C mas a sequência de métodos do SDK a utilizar em Java ou C# será a mesma, isto é:

1. pteid.CVC_Init()
2. pteid.CVC_Authenticate()
3. pteid.CVC_ReadFile() ou pteid.CVC_GetAddr()

```
//Função auxiliar para carregar a chave privada associada ao certificado CVC
RSA * loadPrivateKey(char * file_path) {
    FILE * fp = fopen(file_path, "r");
    if (fp == NULL) {
        fprintf(stderr, "Failed to open private key file: %s!\n", file_path);
        return NULL;
    }
    RSA * key = PEM_read_RSAPrivateKey(fp, NULL, NULL, NULL);
    if (key == NULL) {
        fprintf(stderr, "Failed to load private key file!\n");
    }
    return key;
}
```

```
//Init OpenSSL
OpenSSL_add_all_algorithms();
ERR_load_crypto_strings();
(...)
unsigned char challenge[128];
// challenge that was signed by the private key corresponding to the CVC
unsigned char signature[128];
unsigned char fileBuffer[2000];
long ret;
ret = PTEID_CVC_Init( cvcCert, cvcCert_len, challenge,
sizeof(challenge));
if ( ret != 0 ){
    PTEID_Exit(0);
    return 1;
}
// private_key_path - path for private key file in
RSA* rsa_key = loadPrivateKey(private_key_path);
RSA_private_encrypt( sizeof(challenge), challenge, signature, rsa_key,
RSA_NO_PADDING);
ret = PTEID_CVC_Authenticate( signature, sizeof(signature) );
if ( ret != 0 ){
    PTEID_Exit(0);
    return 1;
}
unsigned char fileID[] = { /* address for file */ };
unsigned long outlen = sizeof(fileBuffer);
ret = PTEID_CVC_ReadFile( fileID, sizeof(fileID), fileBuffer, &outlen );
if ( ret != 0 ){
    PTEID_Exit(0);
    return 1;
}
(...)
PTEID_ADDR addrData; //For CVC_GetAddr()
ret = PTEID_CVC_GetAddr( &addrData );
if ( ret != 0 ){
    PTEID_Exit(0);
    return 1;
}
```

6. Tratamento de erros

O SDK do middleware trata os erros através do lançamento de excepções qualquer que seja a linguagem utilizada: C++, Java ou C#.

Os métodos de compatibilidade com a versão 1 do Middleware (MW) usam outros mecanismos de tratamento de erros: para mais detalhes consultar a secção 7.3.

A classe base de todas as excepções do MW é a classe *PTEID_Exception*.

Existem algumas subclasses de *PTEID_Exception* para erros específicos como *PTEID_ExNoCardPresent* ou *PTEID_ExCardBadType*.

Em todos os casos é sempre possível obter um código de erro numérico para todos os erros que estão tipificados nos métodos do MW através da chamada ao método *GetError()* da classe *PTEID_Exception*.

As constantes numéricas dos códigos de erro estão expostas às aplicações em:

- C++: no ficheiro de include **eidErrors.h**
- C#: membros públicos da classe **pteidlib_dotNet** com o prefixo EIDMW
- Java: membros públicos da interface **pteidlib_JavaWrapperConstants** com o prefixo EIDMW

7. Compatibilidade com o SDK da versão 1

7.1 Métodos removidos

Nome do método da classe pteid	Descrição
ChangeAddress	Métodos relacionados com alteração de morada, na versão 2 do SDK são substituídos pelo método ChangeAddress() da classe PTEID_EIDCard
CancelChangeAddress	
GetChangeAddressProgress	
GetLastWebErrorCode	
GetLastWebErrorMessage	
CVC_Authenticate_SM101	Métodos destinados a cartões que já não estão em circulação
CVC_Init_SM101	
CVC_WriteFile	
CVC_WriteAddr	
CVC_WriteSOD	
CVC_WriteFile	
CVC_R_DH_Auth	
CVC_R_Init	

7.2 Diferenças no comportamento de alguns métodos

- Método pteid.GetCertificates()

Na versão anterior 1.26, o certificado «*Baltimore CyberTrust Root*» não está a ser retornado, ao contrário desta versão que obtém tal certificado.

- Método pteid.GetPINs()

As flags dos PINs retornadas possuem valores diferentes. A versão anterior 1.26, neste momento, retorna o valor 47_{10} ($0011\ 0001$)₂ e esta versão retorna o valor 17_{10} ($0001\ 0001$)₂.

- Método pteid.ReadFile()

O tamanho do *buffer* retornado com o conteúdo do ficheiro lido tem tamanhos diferentes. A versão 1, retorna em blocos de 240 bytes, enquanto esta versão retorna o tamanho total do ficheiro, que neste momento é de 1000 bytes (para o caso do ficheiro de notas).

- Métodos `pteid.WriteFile()` / `pteid.WriteFile_inOffset()`

Quando é necessário escrever no ficheiro `PersoData` (Notas) do Cartão de Cidadão, o pedido de PIN é diferente. Na versão 1, o PIN é pedido uma vez dentro de uma sessão, podendo ser efectuada várias escritas, sem ser pedido novamente o PIN. Nesta versão, o PIN é sempre pedido quando é feita uma nova escrita no ficheiro.

- Métodos `pteid.VerifyPIN()`

Na versão 1, quando um PIN é introduzido incorrectamente, é lançada uma excepção de imediato, enquanto que nesta versão tal não acontece. A excepção é apenas lançada se o utilizador escolher a opção “Cancelar” no diálogo de PIN errado que é mostrado.

7.3 Códigos de Erro

Em vez de lançar excepções, a *SDK* para linguagem C mantém a compatibilidade com versões anteriores em que são devolvidos os códigos descritos nas seguintes tabelas. Os códigos retornados pela SDK estão apresentados na seguinte tabela, também presentes no ficheiro *eidlibcompat.h*.

Código de retorno	Valor	Descrição
PTEID_OK	0	Função executou com sucesso
PTEID_E_BAD_PARAM	1	Parâmetro inválido
PTEID_E_INTERNAL	2	Ocorreu um erro de consistência interna
PTEID_E_NOT_INITIALIZED	9	A biblioteca não foi inicializada

Foi removida a dependência da biblioteca *pteidlibopensc* contudo um conjunto de códigos de retorno, descritos na tabela abaixo, continuam a ser mantidos para garantir retrocompatibilidade.

Código de retorno	Valor	Notas
SC_ERROR_NO_READERS_FOUND	-1101	Não existe um leitor conectado
SC_ERROR_CARD_NOT_PRESENT	-1104	O cartão de cidadão não está inserido no leitor

SC_ERROR_KEYPAD_TIMEOUT	-1108	Expirou o tempo para introduzir o PIN
SC_ERROR_KEYPAD_CANCELLED	-1109	O utilizador cancelou a acção de introduzir o PIN
SC_ERROR_AUTH_METHOD_BLOCKED	-1212	O cartão tem o método de autenticação bloqueado
SC_ERROR_PIN_CODE_INCORRECT	-1214	O código PIN introduzido está incorrecto
SC_ERROR_INTERNAL	-1400	Ocorreu um erro interno
SC_ERROR_OBJECT_NOT_VALID	-1406	A consistência da informação presente no cartão está comprometida

8. Notas do Utilizador

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.