

Manual do SDK – Middleware do Cartão de Cidadão

Versão 2.4



Índice

1. Histórico.....	4
2. Introdução.....	5
3. Abreviaturas e acrónimos.....	6
XML Advanced Electronic Signatures.....	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
5.4 Dados pessoais do cidadão.....	12
5.4.1. Obtenção da Identificação.....	13
5.4.2. Obtenção da fotografia.....	14
5.4.3. Obtenção da morada.....	15
5.4.4. Leitura e escrita das notas pessoais.....	16
5.4.5. Leitura dos dados de identidade do Cidadão e da Morada.....	17
5.4.6. Obtenção dos dados cartão em formato XML.....	21
5.5 PINs.....	22
5.5.1. Verificação e alteração do PIN.....	22

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.....	29
5.6.4. Multi-assinatura com uma única introdução de PIN.....	30
5.7 Certificados digitais.....	31
5.7.1. Leitura dos certificados digitais presentes no cartão de cidadão.....	31
5.8 Sessão segura.....	32
6. Tratamento de erros.....	35
7. Compatibilidade com o SDK da versão 1.....	36
7.1 Métodos removidos.....	36
7.2 Diferenças no comportamento de alguns métodos.....	36
8. Notas do Utilizador.....	38

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

2. Introdução

Este documento destina-se a programadores e analistas de sistemas que tencionam desenvolver soluções informáticas com base no SDK do middleware versão 2 do Cartão de Cidadão.

Esta versão do SDK disponibiliza a mesma interface (API) que a disponibilizada na versão 1 do SDK. Desta forma, pretende-se obter a retro-compatibilidade entre as duas versões 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.

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
XAdES	XML Advanced Electronic Signatures
PDF	Portable Document Format

4. Instalação

4.1 Sistemas Operativos suportados

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

- Sistemas operativos Windows:
 - Windows Vista;
 - Windows 7;
 - Windows 8/8.1;
 - Windows 10
- Distribuições de Linux:
 - Ubuntu: 14.04 e superiores
 - OpenSuse: Leap 42.2
 - Fedora: 24 e superiores
 - Caixa Mágica: 22
- 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 7 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.

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 16.04, para outras distribuições Linux os nomes serão diferentes):

- libxerces-c-dev
- libxml-security-c-dev
- libssl-dev
- libcurl4-openssl-dev
- qtbase5-dev
- swig
- libpcsclite-dev
- qt5-qmake
- qt5-default
- libpoppler-qt5-dev
- default-jdk

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 diretoria "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();
```

5.4 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.

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

1. Exemplo C++

2. Exemplo Java

```
(...)  
PTEID_EIDCard card = context.getEIDCard();  
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();
}
```

3. Exemplo C#

```
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();
}
```

5.4.4. Leitura e escrita das notas pessoais

Para ler as notas pessoais deverá ser utilizado o método `PTEID_EId.getPersoData()`. Para a escrita de dados deverá ser utilizado o método `PTEID_EIDCard.writePersonalNotes()`, sendo necessária a introdução do pin da autenticação.

1. Exemplo C++

```
PTEID_EIDCard card;
PTEID_ByteArray pb;
bool bOk;
(...)
// leitura
string pdata = card.getID().getPersoData();
// escrita
bOk = card.writePersonalNotes( pb,
card.getPins().getPinByPinRef(PTEID_Pin.AUTH_PIN));
```

2. Exemplo Java

```
PTEID_EIDCard card;
PTEID_ByteArray pb;
boolean bOk;
(...)
// leitura
String pdata = card.getID().getPersoData();
//escrita
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
<code>getDocumentVersion()</code>	versão do documento de identificação
<code>GetDocumentType()</code>	tipo de documento - "Cartão de cidadão"
<code>getCountry()</code>	código do país no formato ISO3166
<code>getGivenName()</code>	nomes próprios do detentor do cartão
<code>getSurname()</code>	apelidos do detentor do cartão
<code>getGender()</code>	género do detentor do cartão
<code>getDateOfBirth()</code>	data de nascimento
<code>getLocationOfBirth()</code>	local de nascimento
<code>getNationality()</code>	nacionalidade (código do país no formato ISO3166)
<code>getDocumentPAN()</code>	número PAN do cartão (PAN - primary account number)
<code>getValidityBeginDate()</code>	data de emissão
<code>getValidityEndDate()</code>	data de validade
<code>getHeight()</code>	altura do detentor do cartão
<code>getDocumentNumber()</code>	número do cartão de cidadão
<code>getCivilianIdNumber()</code>	número de identificação civil
<code>getTaxNo()</code>	número de identificação fiscal
<code>getSocialSecurityNumber()</code>	número de segurança social
<code>getHealthNumber()</code>	número de utente de saúde

Método	Descrição
getIssuingEntity()	entidade emissora do cartão
getLocalofRequest()	local de pedido do cartão
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() getParents()	apelidos da mãe do detentor do cartão
getPhotoObj()	objecto que contém a foto do detentor do cartão
getCardAuthKeyObj()	chave pública do cartão
getPersoData()	notas pessoais
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

Método	Descrição
getAbbrStreetType()	abreviatura do tipo de via
getStreetType()	tipo de via
getStreetName()	nome da via
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
<code>getGeneratedAddressCode()</code>	código do endereço
<code>IsNationalAddress()</code>	retorna um booleano

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

² ETSI - TS 102 918 -

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

2. Exemplo Java

```
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";
String errors;

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

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";
string errors;
uint lerror;

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

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

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

2. 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/output_signed.pdf";
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);
(...)
```

5.7 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 efetuada a autenticação entre ambas as partes (a aplicação e o cartão). Após este processo as operações seguintes são efetuadas sobre comunicação cifrada e autenticada.

A autenticação da aplicação é efetuada através de CVCs (Card Verifiable Certificates). Estes certificados são emitidos somente a entidades que estejam autorizadas em Lei a efetuar 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 exceções qualquer que seja a linguagem utilizada: C++, Java ou C#.

Os métodos de compatibilidade com a versão 1 do Middleware usam outros mecanismos de tratamento de erros: para mais detalhes consultar o respectivo Manual Técnico.

A classe base de todas as exceções 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 classe **pteidlib_JavaWrapper** com o prefixo **EIDMW**

7. Compatibilidade com o SDK da versão 1

7.1 Métodos removidos

Nome do método da classe pteid	Notas
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.

