Manual do SDK – Middleware da Identificação Eletrónica em Cabo Verde

Versão 1.9.7

04/11/2021

Tabela de Conteúdos

1 Introdução

Este documento destina-se a programadores e analistas de sistemas que tencionam desenvolver soluções informáticas com base no SDK do middleware versão 1 da Identificação Eletrónica em Cabo Verde.

O SDK da Identificação Eletrónica em Cabo Verde consiste num conjunto de bibliotecas utilizadas no acesso e suporte ao Cartão Nacional de Identificação (CNI) e ao Título de Residência para Estrangeiros (TRE). 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 os documentos de identificação.

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

Para questões, sugestões ou comentários relativos à utilização do SDK, envie um e-mail para casadocidadao@casadocidadao.gov.cv.

2 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

3 Instalação

3.1 Sistemas Operativos oficialmente 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: 20.04 e superiores
- Sistemas operativos Apple MacOS:
  - Versões MacOS Mojave (10.14) e superiores.

3.2 Linguagens de programação

A lista de linguagens de programação suportadas são:

- C++
- C#
- Java

3.3 Compiladores

A lista de compiladores suportados são:

- C++:
  - Windows: Visual Studio 2013
- C#:
  - Windows: Visual Studio 2013
- Java
  - Oracle JDK 8 ou superior

3.4 Instalação do Middleware

3.4.1 Windows

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

As bibliotecas Java e C# ficarão disponíveis por defeito em `C:\Program Files\Cabo Verde eID\sdk\` ou na directoria seleccionada durante a instalação da aplicação, neste caso a SDK estará disponível em `{directoria_seleccionada}\sdk\` .

3.4.2 Linux

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

As bibliotecas C++ (libcvmwlib.so) está disponivel em `/usr/lib/cveid-mw/` os respectivos C++ *header files* estão em `/usr/include/cveid-mw/`. A biblioteca Java (cvmwlibj.jar) em `/usr/share/cveid-mw/cveid_jni/`.

3.4.3 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/cveid-mw/cvmw_jni` . No que diz respeito ao SDK C++, os *header files* ficam localizados em `/usr/local/include/cveid-mw` e a biblioteca à qual as aplicações deverão linkar está no caminho `/usr/local/lib/cveid-mw/libcvmwlib.dylib` .

4 Procedimentos

4.1 Pré-condições

1. **Java**
   - Incluir o ficheiro **cvmw-libj.jar** como biblioteca no projecto e adicionar à library path do Java a localização das bibliotecas nativas do SDK (se necessário). As classes estão no *package* **cv.sniac.eid**.
2. **C#**
   - Adicionar a biblioteca **cvmw-lib_dotnet.dll** às *references* do projecto Visual Studio.
   - As classes e métodos estão no namespace **cv.sniac.eid**.

4.2 Inicialização / Finalização do SDK

A biblioteca **cvmw-lib** é inicializada através da invocação do método **CVMW_initSDK()** (não é contudo obrigatório efectuar a inicialização). A finalização do SDK (é obrigatória) deve ser efectuada através da invocação do método **CVMW_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 Java

```java
package cvmwsample;
import cv.sniac.eid.*;
(...)
/* NOTA: o bloco estático seguinte é estritamente necessário uma vez
que é preciso carregar explicitamente a biblioteca JNI que implementa
as funcionalidades disponível pelo wrapper Java.*/

static {
    try {
        System.loadLibrary("cvmw-libj");
    } catch (UnsatisfiedLinkError e) {
        System.err.println("Native code library failed to load. \n" + e);
        System.exit(1);
    }
}
public class SampleCVMW {
    public static void main(String[] args) {
        CVMW_ReaderSet.initSDK();
        (...)
        CVMW_ReaderSet.releaseSDK();
    }
}
```

2. Exemplo em C#

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

4.3 Configurar modo teste

Para alterar as configurações de forma a utilizar o modo teste, para usar cartões de teste, deve usar-se a função estática **SetTestMode(*bool* bTestMode)** da classe **CVMW_Config**.

Com o valor do parâmetro *bTestMode* a *true*, os seguintes exemplos ativam o modo de teste.

1. Exemplo Java

```
(...)
CVMW_Config.SetTestMode(true);
```

2. Exemplo C#

```
(...)
CVMW_Config.SetTestMode(true);
```

4.4 Acesso ao cartão

Para aceder ao cartão de identificação eletrónica 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 contêm um cartão;
- Obter o objecto que fornece acesso ao cartão;
- Obter o objecto que contêm os dados pretendidos;

A classe **CVMW_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 especifico 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 **CVMW_ReaderContext.getEIDCard()** que devolve um objecto do tipo **CVMW_EIDCard**.

1. Exemplo Java

```
CVMW_EIDCard card;
CVMW_ReaderContext context;
CVMW_ReaderSet readerSet;
readerSet = CVMW_ReaderSet.instance();
for( int i=0; i < readerSet.readerCount(); i++){
    context = readerSet.getReaderByNum(i);
    if (context.isCardPresent()){
        card = context.getEIDCard();
        (...)
    }
}
```

2. Exemplo C#

```
CVMW_EIDCard card;
CVMW_ReaderContext context;
CVMW_ReaderSet readerSet;
readerSet = CVMW_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.

- Java

  `CVMW_ReaderContext readerContext = CVMW_ReaderSet.instance().getReader();`

- C#

  `CVMW_ReaderContext readerContext = CVMW_ReaderSet.instance().getReader();`

4.4.1 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 **CVMW_ReaderContext** associado ao leitor que se pretende monitorizar.

A função de *callback* definida deve ter a seguinte assinatura:

- Java

  `public void onEvent(long lRet, long ulState, Object callbackData)`, numa classe que implemente a interface `CVMW_EventsCallback`

- C#

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

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()**.

1. Exemplo Java:

```
class CardEventsCallback implements CVMW_EventsCallback {
       @Override
      public void onEvent(long lRet, long ulState, Object callbackData) {
        int cardState = (int) ulState & 0x0000FFFF;
        int eventCounter = ((int) ulState) >> 16;
        System.out.println("[Card Event] CardState: " + cardState + "
            EventCounter: " + eventCounter);
        try {
            if (readerContext.isCardPresent()) {
                System.err.println("Card inserted");
            } else {
```

```
                System.err.println("Card removed");
            }
        } catch (CVMW_Exception e) {
            System.out.println("Got Exception on isCardPresent" + e.GetError());
            e.printStackTrace();
        }
    }
}

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

2. 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");
       }
   }

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

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

4.5 Dados pessoais do titular

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

- ficheiro de identificação - contém os dados do titular/cartão impressos nas faces do cartão, incluindo a foto);
- ficheiros de certificados do titular – contêm os certificados de assinatura e autenticação do titular.
- ficheiros de certificados CA's.

4.5.1 Obtenção da Identificação

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

1. Exemplo Java

```
(...)
CVMW_EIDCard card = context.getEIDCard();
CVMW_EId eid = card.getID();

String nome = eid.getGivenName();
String nrDoc = eid.getDocumentNumber();
(...)
```

2. Exemplo C#

```
(...)
CVMW_EIDCard card = context.getEIDCard();
CVMW_EId eid = card.getID();

string nome = eid.getGivenName();
string nrDoc = eid.getDocumentNumber();
(...)
```

4.5.2 Obtenção da fotografia

A fotografia do titular está disponível apenas no formato JPEG2000, o SDK disponibiliza a fotografia no formato original e em formato PNG.

1. Exemplo Java

```
(...)
CVMW_EIDCard card = context.getEIDCard();
CVMW_EId eid = card.getID();
CVMW_Photo photoObj = eid.getPhotoObj();
CVMW_ByteArray praw = photoObj.getphotoRAW();   // formato jpeg2000
CVMW_ByteArray ppng = photoObj.getphoto();  // formato PNG
```

2. Exemplo C#

```
(...)
CVMW_EIDCard card = context.getEIDCard();
CVMW_EId eid = card.getID();
CVMW_Photo photoObj = eid.getPhotoObj();
CVMW_ByteArray praw = photoObj.getphotoRAW();   // formato jpeg2000
CVMW_ByteArray ppng = photoObj.getphoto();  // formato PNG
(...)
```

4.5.3 Leitura dos dados de identidade do titular

Para estes métodos da classe **CVMW_EId** não apresentamos exemplos já que estes dados apenas são responsáveis pelas tarefas de obtenção dos campos específicos dentro do ficheiro de identidade e todos eles devolvem resultados do tipo String.

CVMW_Eid

Método	Descrição
getDocumentVersion()	versão do documento de identificação
getDocumentType()	tipo de documento - ”TRE” ou ”CNI”
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 (apenas TRE)
getPlaceOfBirth()	local de nascimento
getNationality()	nacionalidade (código do país no formato ISO3166)
getResidence()	morada de residência (apenas TRE)
getKindOfTitle()	tipo do título (apenas TRE)
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 documento de identificação
getCivilianIdNumber()	número de identificação civil (apenas TRE)
getIssuingEntity()	entidade emissora 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()	apelidos da mãe do detentor do cartão
getParents()	filiação do detentor do cartão sobre na forma seguinte ”nome e apelido do pai nome e apelido da mãe”
getPhotoObj()	objecto que contêm 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

4.5.4 Leitura e escrita das notas pessoais

Para ler as notas pessoais deverá ser utilizado o método **CVMW_EIDCard.readPersonalNotes()**, passando como parâmetro o tipo de notas (enum **CVMW_Notes**). Para a leitura de notas do bloco privado (**CVMW_NOTES_PRIVATE**) é necessária a introdução do PIN de autenticação.

Para a escrita de dados deverá ser utilizado o método **CVMW_EIDCard.writePersonalNotes()**, passando como parâmetro o tipo de notas (enum **CVMW_Notes**). Para a escrita de notas é necessária a introdução do PIN de autenticação, tanto para o bloco público como para o privado.

Neste momento, as notas pessoais têm um limite de 1000 bytes (codificação recomendada: UTF-8).

1. Exemplo Java

```
CVMW_EIDCard card;
(...)
// Leitura
String pdata = card.readPersonalNotes(CVMW_Notes.CVMW_NOTES_PUBLIC);
// Escrita
String notes = "a few notes";
CVMW_ByteArray pb = new CVMW_ByteArray(notes.getBytes(), notes.getBytes().length
    );
boolean bOk = card.writePersonalNotes(pb, CVMW_Notes.CVMW_NOTES_PUBLIC);
(...)
```

2. Exemplo C#

```
CVMW_EIDCard card;
(...)
// Leitura
String pdata = card.readPersonalNotes(CVMW_Notes.CVMW_NOTES_PUBLIC);
// Escrita
String notes = "a few public notes";
byte[] notesBytes = Encoding.UTF8.GetBytes(notes);
CVMW_ByteArray pb = new CVMW_ByteArray(notesBytes, (uint)notesBytes.Length);
bool ok = card.writePersonalNotes(pb, CVMW_Notes.CVMW_NOTES_PUBLIC);
(...)
```

4.5.5 Obtenção dos dados do cartão em formato XML

Os dados do titular existentes no cartão podem ser extraídos em formato XML. A fotografia que consta no XML está no formato aberto PNG e codificada em Base-64

1. Exemplo em Java

```
String resultXml;
CVMW_EIDCard card;
CVMW_ulwrapper triesLeft = new CVMW_ulwrapper(-1);
(...)
CVMW_XmlUserRequestedInfo requestedInfo = new CVMW_XmlUserRequestedInfo();
requestedInfo.add(XMLUserData.XML_CIVIL_PARISH);
(...)
requestedInfo.add(XMLUserData.XML_GENDER);
CVMW_EIDXML_Doc result = idCard.getXmlEIDDoc(requestedInfo);
resultXml = result.getEIDXML();
```

2. Exemplo em C#

```
string resultXml;
CVMW_EIDCard card;
uint triesLeft;
(...)
CVMW_XmlUserRequestedInfo requestedInfo = new CVMW_XmlUserRequestedInfo();
requestedInfo.add(XMLUserData.XML_CIVIL_PARISH);
(...)
requestedInfo.add(XMLUserData.XML_GENDER);
CVMW_EIDXML_Doc result = idCard.getXmlEIDDoc(requestedInfo);
resultXml = result.getEIDXML();
```

4.6 PINs

4.6.1 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 Java

```
CVMW_EIDCard card;
CVMW_ulwrapper triesLeft = new CVMW_ulwrapper(-1);
(...)
CVMW_Pins pins = card.getPins();
CVMW_Pin pin = pins.getPinByPinRef(CVMW_Pin.AUTH_PIN);
if (pin.verifyPin("", triesLeft, true){
    bool bResult = pin.changePin("","", triesLeft, pin.getLabel());
    if (!bResult && -1 == triesLeft) return;
}
```

2. Exemplo C#

```
CVMW_EIDCard card;
uint triesLeft;
(...)
CVMW_Pins pins = card.getPins();
CVMW_Pin pin = pins.getPinByPinRef(CVMW_Pin.AUTH_PIN);
if (pin.verifyPin("", ref triesLeft, true){
    bool bResult = pin.changePin("","", triesLeft, pin.getLabel());
    if (!bResult && -1 == triesLeft) return;
}
```

4.7 Assinatura Digital

Com o SDK é possível assinar ficheiros em formato PDF, noutros formatos ou blocos de dados.

A data da assinatura de ficheiros pode ser certificada usando timestamps conforme descritos nos exemplos. Para tal, é ainda necessário configurar a *Time Stamping Authority*.

4.7.1 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.

1. 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";

//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)
```

```
15 card.SignXadesA( destino, ficheiros, ficheiros.length );
16 (...)
```

2. Exemplo C#

```
1  string ficheiros[] = new string[4];
2  ficheiros[0]=@"c:\teste\Ficheiro1";
3  ficheiros[1]=@"c:\teste\Ficheiro2";
4  ficheiros[2]=@"c:\teste\Ficheiro3";
5  ficheiros[3]=@"c:\teste\Ficheiro4";
6  string destino = @"c:\teste\ficheiros_assinados.zip";
7
8  //assinar (1 única assinatura para todos os ficheiros)
9  card.SignXades( destino, ficheiros, ficheiros.length );
10 (...)
11 //assinar com selo temporal (1 única assinatura para todos os ficheiros)
12 card.SignXadesT( destino, ficheiros, ficheiros.length );
13 // assinar (1 única assinatura tipo A (archival) para todos os ficheiros)
14 card.SignXadesA( destino, ficheiros, ficheiros.length );
15 (...)
```

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

- Sem selo temporal
  - Java/C#

    ```
    card.SignXadesIndividual( dirDestino, ficheiros, ficheiros.length );
    ```

- Com selo temporal
  - Java/C#

    ```
    card.SignXadesTIndividual( dirDestino, ficheiros, ficheiros.length );
    ```

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

4.7.2 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 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).

A localização da assinatura, na página do documento a assinar, é definida a partir 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 Java visto que, para esta funcionalidade, os wrappers Java e C# contêm exactamente as mesmas classes e métodos necessários **CVMW_PdfSignature()** e **CVMW_EIDCard.SignPDF()**.

Exemplo Java:

```
import cv.sniac.eid.*;
(...)
CVMW_EIDCard card = readerContext.getEIDCard();

//Ficheiro PDF a assinar
CVMW_PDFSignature signature = new  CVMW_PDFSignature("/home/user/input.pdf");

/* Adicionar uma imagem customizada à assinatura visível
   O array de bytes image_data deve ser uma imagem em formato
   JPEG com as dimensões recomendadas (185x41 px) */
CVMW_ByteArray jpeg_data = new CVMW_ByteArray(imageBytes, imageBytes.length);
signature.setCustomImage(jpeg_data);

//Selo de assinatura reduzido
signature.enableSmallSignatureFormat();

//Assinatura com selo temporal
signature.enableTimestamp();

String location = "Lisboa, Portugal";
String reason = "Concordo com o conteudo do documento";

/* Assinatura utilizando localização precisa: os parâmetros pos_x e pos_y
   indicam a localização em percentagem da largura e altura da página */
int page = 1;
//Para páginas A4 verticais este valor pode variar no intervalo [0-1]
double pos_x = 0.1;
//Para páginas A4 verticais este valor pode variar no intervalo [0-1]
double pos_y = 0.1;

String output = "/home/user/output_signed.pdf";
card.SignPDF(signature,  page, pos_x, pos_y, location, reason, output_file);
```

4.7.3 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 **CVMW_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 Java

```
CVMW_ByteArray data_to_sign;
(...)
CVMW_EIDCard card = context.getEIDCard();
(...)
CVMW_ByteArray output= card.Sign(data_to_sign, true);
(...)
```

2. Exemplo C#

```
CVMW_ByteArray data_to_sign, output;
```

```
(...)
CVMW_EIDCard card = readerContext.getEIDCard();
CVMW_ByteArray output;
output = card.Sign(data_to_sign, true);
(...)
```

4.7.4 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 Java visto que, para esta funcionalidade, os wrappers Java e C# contêm exactamente as mesmas classes e métodos necessários **CVMW_PdfSignature()**.

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

Exemplo Java

```
import cv.sniac.eid.*;
(...)
CVMW_EIDCard card = readerContext.getEIDCard();

//Ficheiro PDF a assinar
CVMW_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;
boolean isLandscape = false;
String location = "Lisboa, Portugal";
String reason = "Concordo com o conteudo do documento";

//Para uma assinatura em batch, este parâmetro aponta para a directoria de
    destino
String output = "/home/user/";
card.SignPDF(signature,  page, sector, isLandscape, location, reason, output);
(...)
```

4.7.5 Configurar o servidor de selo temporal

O SDK permite assinar com selo temporal conforme expresso nos exemplos acima. Para tal, tem que configurar o servidor usado da seguinte forma:

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

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

Nota: Como este parâmetro é configurado através de uma chave de registo, a configuração mantém-se mesmo após término da aplicação que definiu o servidor de *timestamp*. Além disso, a configuração irá aplicar-se a outras aplicações que também usem este SDK.

4.8 Certificados digitais

4.8.1 Leitura dos certificados digitais presentes

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 Java

```
CVMW_EIDCard card = context.getEIDCard();

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

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

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

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

2. Exemplo C#

```
CVMW_EIDCard card = context.getEIDCard();

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

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

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

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

4.9 Informação biométrica

4.9.1 Leitura das impressões digitais

Os cartões de identificação eletrónica de Cabo Verde (CNI e TRE) têm registadas no seu circuito integrado 2 imagens das impressões digitais do titular em formato WSQ (Wavelet scalar quantization). Através do SDK uma aplicação pode obter essas imagens de um cartão inserido mediante introdução do PIN de autenticação.

Posteriormente estes ficheiros WSQ podem ser analisados com qualquer software que suporte o formato WSQ por exemplo a aplicação Fingerprint Minutiae Viewer desenvolvida pelo NIST:

https://www.nist.gov/services-resources/software/fingerprint-minutiae-viewer-fpmv

1. Exemplo Java

```
1
2  CVMW_EIDCard idCard = context.getEIDCard();
3
4  /* Optional argument pinCode: can be used for application PIN handling
5     If auth_pinCode is null the authentication PIN is requested in a SDK dialog
6   */
7  String auth_pinCode = null;
8  CVMW_Fingerprints fingerprints = idCard.getFingerprintInfo(auth_pinCode);
9  if (fingerprints.containsRightFingerprint()) {
10
11   CVMW_ByteArray wsq_data_right = fingerprints.getRightFingerprint();
12 }
13 else {
14   System.out.println("Card has no right fingerprint!");
15 }
16
17 if (fingerprints.containsLeftFingerprint()) {
18
19   CVMW_ByteArray wsq_data_left = fingerprints.getLeftFingerprint();
20 }
21 else {
22   System.out.println("Card has no left fingerprint!");
23 }
```

2. Exemplo C#

```
1  CVMW_EIDCard idCard = context.getEIDCard();
2
3  /* Optional argument pinCode: can be used for application PIN handling
4     If auth_pinCode is null the authentication PIN is requested in a SDK dialog
5   */
6  string auth_pinCode = null;
7  CVMW_Fingerprints fingerprints = idCard.getFingerprintInfo(auth_pinCode);
8  if (fingerprints.containsRightFingerprint()) {
9
10   CVMW_ByteArray wsq_data_right = fingerprints.getRightFingerprint();
11 }
12 else {
13   Console.WriteLine("Card has no right fingerprint!");
14 }
15
16 if (fingerprints.containsLeftFingerprint()) {
17
18   CVMW_ByteArray wsq_data_left = fingerprints.getLeftFingerprint();
19 }
20 else {
21   Console.WriteLine("Card has no left fingerprint!");
22 }
```

5 Tratamento de erros

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

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

Existem algumas subclasses de **CVMW_Exception** para erros específicos como **CVMW_ExNoCardPresent** ou **CVMW_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 **CVMW_Exception**.

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

- C#: membros públicos da classe **cvmwlib_dotNet** com o prefixo EIDMW
- Java: membros públicos da interface **cvmwJava_WrapperConstants** com o prefixo EIDMW

6 Notas do Utilizador