PROYECTO: **Framework DNIe**

DOCUMENTO:

GUIA DE REFERENCIA. API DE DESARROLLO DEL FRAMEWORK DNIe

LENGUAJE DE PROGRAMACIÓN C SHARP

CONTROL DOCUMENTAL

Documento	GuiaRef FrameworkDNIe Csharp_2.docx
Título	Guía de referencia. API de desarrollo del Framework DNIe. Lenguaje de programación C SHARP
Tipo	Manual de referencia
Entidad	red.es
Autores	AT / PE / QA

VERSIONES

Versión	Fecha	Descripción y cambios realizados
1.0	16/05/2011	Versión inicial del documento
1.1	22/06/2011	Incorporación de logos
1.2	23/11/2011	Incorporación driver Card Module

ACRÓNIMOS

API	Application Programming Interface
CSP	Cryptographic Service Provider
DNIe	Documento Nacional de Identidad Electrónico
IDE	Integrated Development Environment
OCSP	Online Certificate Status Protocol
PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards

DOCUMENTOS DE REFERENCIA

ÍNDICE DE CONTENIDO

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

1 INTRODUCCIÓN

1.1 Descripción general del proyecto

El principal objetivo del proyecto es disponer de una **completa plataforma de fuentes abiertas (SFL) para el desarrollo rápido y sencillo de aplicaciones** basadas en el uso del DNI electrónico.

Para cubrir este objetivo la plataforma Framework DNIe[1] incluye, entre otros, los siguientes elementos:

- Incluye una API de desarrollo (Framework DNIe), implementada sobre distintos lenguajes de programación, que abstrae la necesidad de programar directamente sobre los drivers del DNIe, facilitando el desarrollo rápido y sencillo de aplicaciones basadas en el uso del DNIe.
- Incluye distintas aplicaciones que muestran, a modo de ejemplo, como se puede utilizar el Framework DNIe para la implementación de aplicaciones; a la vez que se pueden utilizar como esqueleto para nuevas aplicaciones.
- Incorporación de otros elementos que favorecen la implementación de aplicaciones: accesibilidad al código y ejecutables a través del portal de información y descarga, mecanismos de emulación del DNI físico, asistente de generación del entorno, etc.

Con todo ello se busca facilitar la incorporación al mercado de nuevas soluciones que potencien el uso de las capacidades electrónicas del DNIe, minimizando la complejidad tecnológica de este tipo de desarrollos.

1.2 Audiencia

El documento va dirigido, principalmente, a desarrolladores que vayan a implementar aplicaciones basadas en el uso del DNIe o que vayan a incluir alguna de sus capacidades.

1.3 Objeto

El objeto del presente documento es realizar la descripción y comentar la utilización de los distintos métodos contenidos en el Framework DNIe, con la finalidad de que cualquier desarrollador lo pueda utilizar en el desarrollo de aplicaciones.

[1] En el presente documento se utilizará la siguiente notación:
- Se hablará de 'Plataforma Framework DNIe' o 'Proyecto Framework DNIe' para hablar de la plataforma completa con todos los elementos mencionados.
- Se hablará de 'Portal Framework DNIe' para hablar del site de información y descarga del proyecto.
- Se hablará de 'Framework DNIe' o 'Framework (lenguaje)' para mencionar cualquiera de las APIs software que facilitan el desarrollo rápido de aplicaciones sobre cada uno de los lenguajes de programación

2 COMPONENTES

2.1 Entorno de operación

El Framework DNIe requiere de una serie de elementos software para su correcto funcionamiento. A continuación se detallan dichos recursos:

- Entorno .NET Framework 4 de Microsoft. Puede descargarse de la dirección http://msdn.microsoft.com/es-es/netframework/aa569263.
- Drivers para la comunicación con el DNIe:
  - Mediante el interfaz CSP. Puede descargarse de la siguiente dirección http://www.dnielectronico.es/descargas/.
  - Interfaz '*Smart Card Module*' (disponible mediante Windows Update).

2.2 Arquitectura de clases

El Framework DNIe está formado por la siguiente estructura de paquetes y clases:

PAQUETE	CLASES PÚBLICAS CONTENIDAS
dnieframework	➢ DNIeFramework.cs ➢ Autenticacion.cs ➢ FirmaE.cs ➢ Autochequeo.cs
dnieframework.utiles	➢ ModuloCriptografico.cs ➢ DNIeCardNotFoundException.cs ➢ DNIeDriversNotFoundException.cs ➢ DNIeException.cs ➢ DNIeKeyStoreException.cs ➢ DNIeACDNIeNotFoundException.cs

A continuación se ofrece una breve descripción de cada uno de los componentes:

- DNIeFramework.cs
  - Se utiliza para agrupar las diferentes funcionalidades ofrecidas, con la finalidad de facilitar en la medida de lo posible la utilización del Framework DNIe.
- Autenticacion.cs
  - En esta clase ofrece toda la funcionalidad referente a la autenticación mediante el DNIe, facilitándose diferentes métodos para su implementación en las diferentes arquitecturas que se pueda requerir esta funcionalidad.
- FirmaE.cs
  - En esta clase ofrece toda la funcionalidad referente a la firma mediante el DNIe. El DNIe tiene la capacidad de crear firmas reconocidas equivalentes a la firma manuscrita. Esta clase ofrece métodos que implementan diferentes formatos de firma.
- Autochequeo.cs
  - Esta clase permite realizar comprobaciones del entorno operativo del Framework.
- ModuloCriptografico.cs
  - Esta clase aglutina funciones de más bajo nivel, como comunicaciones con la tarjeta, el almacén de certificados, etc.
- DNIeCardNotFoundException.cs
  - Excepción utilizada en caso de no poder conectar con la tarjeta del DNIe.
- DNIeDriversNotFoundException.cs
  - Excepción utilizada en caso de existir algún problema con los drivers.
- DNIeException.cs
  - Excepción utilizada en caso de existir algún problema con el Framework DNIe.
- DNIeKeyStoreException.cs
  - Excepción utilizada en caso de existir algún problema con el almacén de certificados.
- DNIeACDNIeNotFoundException.cs
  - Excepción utilizada en caso de no encontrarse los Certificados de Autoridad Intermedia.

3 API DE REFERENCIA

3.1 Carga en entorno de desarrollo

El Framework DNIe se distribuye como un archivo *"dnieframework.dll",* que deberemos importar como librería en el proyecto que queramos desarrollar. A su vez, deberemos importar al proyecto que desarrollemos las librerías que utiliza el Framework DNIe para su propio funcionamiento. A continuación se detallan las librerías que necesita el Framework DNIe:

LIBRERÍA	ARCHIVO / DESCARGA
Bouncy Castle. Librería criptográfica.	Está incluida en la librería iTextSharp, por lo que no debemos incluirla.
iTextSharp. Librería para manejo de PDF.	"iTextSharp.dll". Puede descargarse de: http://sourceforge.net/projects/itextsharp/files/itextsharp/
log4net. Logger de CS.	"log4net.dll". Puede descargarse de: http://logging.apache.org/log4net/download.html

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

3.2 API

3.2.1 Package dnieframework

3.2.1.1 public class DNIeFramework

public class DNIeFramework

Esta clase aglutina todas las funcionalidades ofrecidas por el DNIe. Para utilizar dichas funcionalidades debemos instanciar la clase DNIeFramework, que ofrecerá estas funcionalidades en respectivos métodos.

Constructor

- void **DNIeFramework()**

Métodos Públicos

- public void **cargarDatosDNIe**()
- public String **obtenerNombre**()
- public String **obtenerApellidos**()
- public String **obtenerNIF**()
- public byte[] **obtenerSerialNumber**()
- public DateTime **obtenerNotAfter**()
- public DateTime **obtenerNotBefore**()
- public X509Certificate2 **obtenerCertificado** (String tipoCertificado)
- public boolean **autenticar**(byte[] reto)
- public boolean **comprobarRetoFirmado**(byte[] retoFirmado, byte[] reto, X509Certificate2 certificado)
- public byte[] **firmarReto**(byte[] reto)
- public byte[] **firmar**(byte[] datos)
- public boolean **verificarFirma**(byte[] datosFirmados,byte[] datos, X509Certificate2 certificado)
- public void **firmarPAdES**(String pdfOrigen, String pdfDestino, String motivo, String localizacion, String contacto, float llx, float lly, float urx, float ury)
- public boolean **verificarFirmaPAdES**(String ficheroPDFfirmado)
- public String **validacionCertificadoOCSP** (X509Certificate2 cert, String ACSubPath)

- public String **RealizarAutochequeo**()

3.2.1.1.1 cargarDatosDNIe

public void **cargarDatosDNIe** ()

Descripción

Este método carga los datos del certificado de autenticación del DNIe para que puedan ser accedidos posteriormente.

Parámetros

Retorno

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.2 obtenerNombre

public String **obtenerNombre** ()

Descripción

Obtiene el nombre del titular del DNIe.

Parámetros

Retorno

- (String) Nombre del titular

Excepciones

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

3.2.1.1.3 obtenerApellidos

public String **obtenerApellidos**()

Descripción

Obtiene los apellidos del titular del DNIe.

Parámetros

Retorno

- (String) Apellidos del titular

Excepciones

3.2.1.1.4 obtenerNIF

public String **obtenerNIF**()

Descripción

Obtiene el NIF del titular del DNIe.

Parámetros

Retorno

- (String) NIF del titular

Excepciones

3.2.1.1.5 obtenerSerialNumber

public byte[] **obtenerSerialNumber**()

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

Descripción
Obtiene el Serial Number del certificado del DNIe.

Parámetros

Retorno
• (byte[]) Serial Number del certificado

Excepciones

3.2.1.1.6 obtenerNotAfter

public DateTime **obtenerNotAfter**()

Descripción
Obtiene el campo NotAfter del certificado, que indica la fecha en que finaliza el periodo de validez del certificado.

Parámetros

Retorno
• (DateTime) Campo NotAfter del certificado-

Excepciones

3.2.1.1.7 obtenerNotBefore

public DateTime **obtenerNotBefore**()

Descripción
Obtiene el campo NotBefore del certificado, que indica la fecha en que comienza el periodo de validez del certificado.

Parámetros

Retorno

- (DateTime) Campo NotBefore del certificado.

Excepciones

3.2.1.1.8 obtenerCertificado

public X509Certificate2 **obtenerCertificado** (String tipoCertificado)

Descripción

Obtiene el certificado del DNIe del tipo que indiquemos.

Parámetros

- **tipoCertificado** – (String) Puede ser "AUTENTICACION" o "FIRMA".

Retorno

- (X509Certificate2) Certificado del DNIe.

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.9 autenticar

public boolean **autenticar** (byte[] reto)

Descripción

Este método permite realizar la autenticación completa con el DNIe. Principalmente orientado a aplicaciones de escritorio.

Parámetros

- **reto –** (byte[]) El reto que debe firmar el DNIe para autenticarse.

Retorno

- (boolean) Devuelve true si la autenticación es correcta, false en otro caso.

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.10 comprobarRetoFirmado

public boolean **comprobarRetoFirmado** (byte[] retoFirmado, byte[] reto, X509Certificate2 certificado)

Descripción

Este método permite comprobar la validez del reto firmado con la clave privada asociada al certificado de autenticación del DNIe.

Parámetros

- **retoFirmado –** (byte[]) El reto firmado cuya validez se debe comprobar.
- **reto –** (byte[]) El reto original con el que se hará la comprobación.
- **certificado –** (X509Certificate2) El certificado de autenticación del DNIe que queremos autenticar.

Retorno

- (boolean) Devuelve true si la autenticación es correcta, false en otro caso.

Excepciones

- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.11 firmarReto

```
public byte[] firmarReto (byte[] reto)
```

Descripción

Este método permite obtener un reto firmado con la clave privada asociada al certificado de autenticación del DNIe.

Parámetros

- **reto** -(byte[]) El reto que debe firmar el DNIe para autenticarse.

Retorno

- (byte[]) Devuelve el reto firmado.

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.12 firmar

```
public byte[] firmar (byte[] datos)
```

Descripción

Este método utiliza la clave privada asociada al certificado de firma del DNIe para realizar la firma de los datos que recibe como entrada.

Parámetros

- **datos** - (byte[]) Datos para firmar.

Retorno

- (byte[]) Devuelve la firma del fichero en formato RAW.

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.13 verificarFirma

public boolean **verificarFirma** (byte[] datosFirmados, byte[] datos, X509Certificate2 certificado)

Descripción

Este método verifica la validez de la firma que se toma como entrada.

Parámetros

- **datosFirmados** – (byte[])Los datos firmados que debemos verificar.
- **datos** – (byte[])Los datos que utilizaremos para verificar la firma.
- **certificado** – (X509Certificate2) certificado del que extraemos la clave pública para verificar la firma.

Retorno

- (boolean) true en caso de que la firma sea válida, false en otro caso.

Excepciones

- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.14 firmarPAdES

```
public void firmarPAdES (String pdfOrigen,
                         String pdfDestino,
                         String motivo,
                         String localizacion,
                         String contacto,
                         float llx,
                         float lly,
                         float urx,
                         float ury)
```

Descripción

Este método realiza la firma PAdES de un documento pdf, generando automáticamente el fichero firmado.

Parámetros

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

- **pdfOrigen** – (String) El path del documento a firmar.
- **pdfDestino** – (String) El path del documento firmado.
- **motivo** – (String) Datos a incluir en la firma.
- **localizacion** – (String) Datos a incluir en la firma.
- **contacto** – (String) Datos a incluir en la firma
- **llx** – (float) Coordenada x inferior para el sello visible de la firma.
- **lly** – (float) Coordenada y inferior.
- **urx** – (float) Coordenada x superior.
- **ury** – (float) Coordenada y superior.

Retorno

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.1.15 verificarFirmaPAdES

public boolean **verificarFirmaPAdES** (String ficheroPDFfirmado)

Descripción

Verifica la validez de la firma PAdES que se toma como entrada.

Parámetros

- **ficheroPDFfirmado** – (String) El path del documento firmado a verificar.

Retorno

- (Boolean) true en caso de que la firma sea válida, false en otro caso

Excepciones

- **DNIeException** - Si hay problemas de cifrado.

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

3.2.1.1.16 validacionCertificadoOCSP

public String **validacionCertificadoOCSP** (X509Certificate cert, String ACSubPath)

Descripción

Este método realiza la validación OCSP del certificado pasado como parámetro.

Parámetros

- **cert** – (X509Certificate) Certificado a validar.
- **ACSubPath** – (String) Path de la AC del DNIe (donde se encuentran los certificados de la autoridad intermedia)

Retorno

- (String) Resultado de la validación [‘DESCONOCIDO’, ‘REVOCADO’, ‘VALIDO’]

Excepciones

- **DNIeACDNIeNotFoundException.** No se encuentra la AC
- org.bouncycastle.ocsp.**OCSPException** - Si hay algún problema con la petición OCSP.

3.2.1.1.17 RealizarAutochequeo

public String **RealizarAutochequeo()**

Descripción

Este método realiza comprobaciones de entorno

Parámetros

Retorno

- (String) Resultado del chequeo

Excepciones

3.2.1.2 public class Autenticacion

public class Autenticacion extends Object

Esta clase aglutina el comportamiento y los datos involucrados en el proceso de autenticación con el DNIe.

Constructor

- void **Autenticacion()**

Métodos Públicos

- public boolean **autenticar**(byte[] reto)
- public boolean **comprobarRetoFirmado**(byte[] retoFirmado, byte[] reto, X509Certificate2 cert)
- public byte[] **firmarReto**(byte[] reto)

3.2.1.2.1 autenticar

public boolean **autenticar** (byte[] reto)

Descripción

Este método permite utilizar el DNIe para autenticarse de manera que se requerirá el código PIN para realizar la firma de un reto o desafío con la clave privada asociada al certificado de autenticación del DNIe. Después comprueba la validez de la firma con la clave pública contenida en el certificado de autenticación. El certificado de autenticación permite garantizar la identidad de una persona en una transacción telemática.

Parámetros

- **reto** – (byte[])El reto que debe firmar el DNIe para autenticarse.

Retorno

- (Boolean) Devuelve true si la autenticación es correcta, false en otro caso.

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

- **DNIeException** - Si hay problemas de cifrado.

3.2.1.2.2 comprobarRetoFirmado

public boolean **comprobarRetoFirmado** (byte[] retoFirmado, byte[] reto, X509Certificate2 cert)

Descripción

Recibe un reto firmado con la clave privada del DNIe y comprueba que la firma es válida con el certificado recibido, que debería ser el certificado de autenticación del DNIe.

Parámetros

- **retoFirmado** – (byte[]) El reto firmado cuya validez se debe comprobar.
- **reto** –(byte[]) El reto original con el que se hará la comprobación.
- **certificado** – (X509Certificate2) El certificado de autenticación del DNIe que queremos autenticar.

Retorno

- (Boolean) Devuelve true si la comprobación es correcta, false en otro caso.

Excepciones

- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.2.3 firmarReto

public byte[] **firmarReto** (byte[] reto)

Descripción

Este método permite firmar un reto utilizando la clave privada asociada al certificado de autenticación del DNIe.

Parámetros

- **reto** – (byte[])El reto que debe firmar el DNIe para autenticarse.

Retorno

- (byte[]) Devuelve el reto firmado

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

Excepciones

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.3 public class FirmaE

public class FirmaE extends Object

Esta clase aglutina todas las funcionalidades de firma ofrecidas por el DNIe.

Los métodos incluidos en la clase DNIeFramework, con el mismo nombre, realizan llamadas a los métodos de esta clase.

Constructor

- void **FirmaE()**

Métodos Públicos

- public byte[] **firmar**(byte[] datos)
- public boolean **verificarFirma**(byte[] datosFirmados,byte[] datos, X509Certificate2 certificado)
- public void **firmarPAdES**(String pdfOrigen, String pdfDestino, String motivo, String localizacion, String contacto, float llx, float lly, float urx, float ury)
- public boolean **verificarFirmaPAdES**(String ficheroPDFfirmado)

3.2.1.3.1 firmar

public byte[] **firmar** (byte[] datos)

Descripción

Este método utiliza la clave privada asociada al certificado de firma del DNIe para realizar la firma de los datos que recibe como entrada.

Parámetros

- **datos** – (byte[]) Datos para firmar. El certificado de firma garantiza la integridad del documento y el no repudio de origen.

Retorno

- (byte[]) Devuelve la firma del fichero en formato RAW.

Excepciones

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

- **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe.
- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.3.2 verificarFirma

public boolean **verificarFirma** (byte[] datosFirmados, byte[] datos, X509Certificate2 certificado)

Descripción

Este método verifica la validez de la firma que se toma como entrada, en formato RAW

Parámetros

- **datosFirmados** – (byte[]) Los datos firmados que debemos verificar.
- **datos** – (byte[])Los datos que utilizaremos para verificar la firma.
- **certificado** – (X509Certificate2) Certificado del que extraemos la clave pública para verificar la firma.

Retorno

- (boolean) true en caso de que la firma sea válida, false en otro caso.

Excepciones

- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.
- **DNIeException** - Si hay problemas de cifrado.

3.2.1.3.3 firmarPAdES

```
public void firmarPAdES (String pdfOrigen,
                         String pdfDestino,
                         String motivo,
                         String localizacion,
                         String contacto,
                         float llx,
                         float lly,
                         float urx,
                         float ury)
```


(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

Descripción
Este método realiza la firma PAdES (detached) de un documento pdf, generando automáticamente el fichero firmado.

Parámetros
• **pdfOrigen** – (String) El path del documento a firmar. • **pdfDestino** – (String) El path del documento firmado. • **motivo** – (String) Datos a incluir en la firma. • **localizacion** – (String) Datos a incluir en la firma. • **contacto** – (String) Datos a incluir en la firma • **llx** – (float) Coordenada x inferior para el sello visible de la firma. • **lly** – (float) Coordenada y inferior. • **urx** – (float) Coordenada x superior. • **ury** – (float) Coordenada y superior.

Retorno

Excepciones
• **DNIeCardNotFoundException** - Si existe algún problema al conectar con la tarjeta del DNIe. • **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados. • **DNIeDriversNotFoundException** - Si existe algún problema con los drivers del DNIe. • **DNIeException** - Si hay problemas de cifrado.

3.2.1.3.4 verificarFirmaPAdES

public boolean **verificarFirmaPAdES** (String ficheroPDFfirmado)

Descripción
Verifica la validez de la firma PAdES que se toma como entrada.

Parámetros
• **ficheroPDFfirmado** – (String) El path del documento firmado a verificar.

Retorno
• (Boolean) true en caso de que la firma sea válida, false en otro caso

Excepciones
• **DNIeException** - Si hay problemas de cifrado.

3.2.1.4 public class Autochequeo

public class Autochequeo extends Object

Esta clase aglutina todas las funcionalidades ofrecidas por la clase Autochequeo.

Los métodos utilizados no incluyen retorno: se considera que las funcionalidades de chequeo son correctas si no se produce una excepción.

Constructor

- void **Autochequeo()**

Métodos Públicos

- public void **comprobarSO**()

3.2.1.4.1 comprobarSO

public void **comprobarSO**()

Descripción

Este método comprueba el SO en el que se ejecuta la aplicación.

Parámetros

Retorno

Excepciones

- **AutochequeoException** - Si hay problemas en el proceso de autochequeo del entorno.

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

3.2.2 Package dnieframework.utiles

3.2.2.1 public class ModuloCriptografico

public class ModuloCriptografico

Esta clase aglutina todas las funcionalidades ofrecidas por la clase ModuloCriptografico.

Constructor

- void **ModuloCriptografico()**

Métodos Públicos

- public static Boolean **ConectarTarjetaDNIe**()
- public static bool **libreriasDNIeCargadas**()
- public static X509Certificate2 **obtenerCertificadoDNIe**(String tipoCertificado)
- public static void **AbrirAlmacenCertificados**()
- public static void **CerrarAlmacenCertificados**()
- public static byte[] **ReadToEnd**(System.IO.Stream stream)
- public static OCSPStatus **OCSP_validation**(X509Certificate clientCertificate, out String, String ACSubPath)

3.2.2.1.1 ConectarTarjetaDNIe

public static Boolean **ConectarTarjetaDNIe**()

Descripción

Este método comprueba que la tarjeta inteligente del DNIe ha sido insertada previamente en el lector, si encuentra un DNIe insertado comprueba que el estado del DNIe es correcto.

Parámetros

Retorno

- (Boolean) devuelve true en caso de conexión OK, false en otro caso.

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

Excepciones

3.2.2.1.2 obtenerCertificadoDNIe

public static X509Certificate2 **obtenerCertificadoDNIe** (String tipoCertificado)

Descripción

Este método se encarga de obtener una referencia al certificado del tipo especificado.

Parámetros

- **tipoCertificado** – (String) El tipo de certificado. Puede tratarse del certificado de firma "FIRMA" o el de autenicación "AUTENTICACION".

Retorno

- (X509Certificate2) Certificado de tipo X509.

Excepciones

- **DNIeKeyStoreException** - Si existe algún problema con el almacén de certificados.

3.2.2.1.3 AbrirAlmacenCertificados

public static void **AbrirAlmacenCertificados**()

Descripción

Este método abre el almacén de certificados

Parámetros

Retorno

Excepciones

3.2.2.1.4 CerrarAlmacenCertificados

public static void **CerrarAlmacenCertificados**()

Descripción

Este método cierra el almacén de certificados

Parámetros

Retorno

Excepciones

3.2.2.1.5 ReadToEnd

public static byte[] **ReadToEnd** (System.IO.Stream stream)

Descripción

Función auxiliar que transforma un stream en un array de bytes.

Parámetros

- **stream** – (Stream) Stream a modificar.

Retorno

- (byte[]) Array de bytes obtenido.

Excepciones

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)

3.2.2.1.6 OCSP_validation

public static OCSPStatus **OCSP_validation** (X509Certificate clientCertificado, out string respMsg, String ACSubPath)

Descripción

Obtiene la respuesta OCSP del certificado pasado como parámetro.

Parámetros

- **clientCertificado** – (X509Certificate) Certificado a validar.
- **respMsg** - (String) Respuesta obtenida.
- **ACSubPath** – (String) Path de la AC

Retorno

- (OCSPStatus) Respuesta OCSP.

Excepciones

(*) LD: Libre distribución
DI: Sólo distribución interna
ND: Bajo acuerdo de no revelación (NDA)