MiddleWare y Applet

Guía para desarrollo de Software

consumiendo funcionalidades del MiddleWare

Rolosa HyJ S.A. - MICITT

20 de Enero de 2014

Resumen

El presente documento es una guía para desarrolladores de software útil para construir aplicaciones utilizando el Middleware. Con este documento es posible crear una aplicación simple que pueda acceder a la tarjeta inteligente de Firma Digital de Costa Rica. No todos los métodos y objetos son descritos en detalle. Mayor detalle se puede encontrar en la documentación del API del Middleware.

Tabla de contenido

Pre-requisitos....1

Introducción....1

Abreviaciones y Acrónimos....2

Tarjetas soportadas....2

Compatibilidad....2

Instalación....3

Windows....3

Linux....4

Mac....4

Descripción....5

Componentes....5

Inicializando /Finalizando el modulo PKCS#11....6

Accediendo a una Tarjeta Inteligente....7

Obtener lista de lectores de tarjeta....8

Verificar si la Tarjeta está presente....8

Obtener los datos - Certificados....10

Obtener los datos - Claves Privadas....11

Autenticación y Firmado digital....13

Aplicación demostrativa - C#....13

Aplicación demostrativa - Java....14

Pre-requisitos

Para entender los ejemplos que se proveen en este documento, es necesario conocimientos de C/C++, Java y/o C# para el Middleware.

También es necesario disponer de:

- Una Tarjeta Inteligente para Firma Digital de Costa Rica
- Un lector de tarjetas apropiado
- Certificados Digitales de la cadena de la confianza de la Firma Digital de Costa Rica
- Drivers necesarios para la Tarjeta Inteligente, dependiendo el sistema operativo en el que se desee realizar los trabajos.

Los drivers y certificados digitales tienen que ser obtenidos desde el sitio de Soporte de Firma Digital de Costa Rica:

https://www.soportefirmadigital.com/sfd/default.aspx

Introducción

El Middleware Es un conjunto de librerías utilizadas para acceder al contenido de las tarjetas inteligentes de Firma Digital de Costa Rica.

El SDK del que se dispone como parte del Middleware no define ningún API nuevo, más aun, utiliza el estándar de interfaz para tokens criptográficos definido por RSA - PKCS#11 v2 11 API.

El estándar RSA - PKCS#11 v2 11 API le permitirá al desarrollador verificar lectores de tarjetas y tarjetas inteligentes, y permite recuperar información de dentro de las tarjetas (como archivos completos de certificados).

Ejemplos de cómo utilizar el API son provistos en el SDK (únicamente para Windows) y aplicaciones demostrativas que se encuentran en el sitio de Google:

http://dcfd-mw-applet.googlecode.com

Es SDK es construido bajo un núcleo de C/C++, y es provisto para 3 tipos de Sistemas operativos:

- Win32
- Linux
- Mac OSX

Para facilitar el desarrollo de aplicaciones en otros leguajes diferentes a C/C++, es necesario utilizar cualquiera de los distintos Wrapper disponibles en internet para PKCS#11.

Abreviaciones y Acrónimos

OCSP (Online Certificate Status Protocol).- Protocolo utilizado para obtener el estado de revocación de un certificado digital.

UTF-8 (Unicode Transformation Format 8bit).- Codificación de longitud variable para caracteres Unicode.

Tarjetas soportadas

En la actualidad, únicamente la tarjeta de Firma Digital de Costa Rica es soportada por el Middleware. Si bien este Midleware es resultado de las modificaciones realizadas al Middleware original de Belgica, ninguna de las tarjetas inteligentes de Bélgica no son soportadas por el actual Middleware

Compatibilidad

Plataformas:

- Windows: Win32 (Windows 2000, XP, Vista, Windows 7 (32 bits), Windows 8 (32bits))
- Linux: Ubuntu, Fedora 9, Debian.
- Mac: OSX 10.4 y 10.5

Lenguajes de programación (con wrappers adecuados):

- C/C++: Windows/Linux/Mac
- Java: Windows/Linux/Mac

- dotNet: VB, C#

Compilador C++

- Windows: Microsoft Visual Studio 2010 y superior
- Linux: Compilador g++ por defecto
- Mac OSX: Compilador g++ por defecto

Instalación

Para utilizar el SDK y poder crear aplicaciones es necesario disponer de las librerías resultantes de la compilación del Middleware las cuales con conocidas como el Runtime.

Windows

Las librerías resultantes de la compilación del MiddleWare que sigue el estándar RSA - PKCS #11, están disponibles en la ruta:

trunk\mw\VS_2010\Binaries\Win32_Debug\beidpkcs11D.dll

trunk\mw\VS_2010\Binaries\Win32_Release\beidpkcs11.dll

La versión Release, puede ser instalada en: **c:\Windows\system32** y registrada dentro el sistema, desde una ventana de consola, con el comando:

regsvr32.exe beidpkcs11.dll

El registro de la librería en el sistema es opcional dependiendo del tipo de proyecto de desarrollo de software.

Los Certificados Digitales de la cadena de la confianza de la Firma Digital de Costa Rica deberán ser instalados en: **C:\Firma Digital\certificados**

Linux

Las librerías resultantes de la compilación del MiddleWare que sigue el estándar RSA - PKCS#11 (**libbeidpkcs11.so**), están disponibles en la ruta:

/usr/local/lib/

Los Certificados Digitales de la cadena de la confianza de la Firma Digital de Costa Rica deberán ser instalados en:

/usr/lib/dcfd/certificados/

Mac

Las librerías resultantes de la compilación del MiddleWare que sigue el estándar RSA - PKCS#11 (**libbeidpkcs11.dylib**), están disponibles en la ruta:

/usr/local/lib/

Los Certificados Digitales de la cadena de la confianza de la Firma Digital de Costa Rica deberán ser instalados en:

/usr/lib/dcfd/certificados/

Descripción

Componentes

C/C++			Win32	Linux	Mac
	pkcs11.h	Archivo de Cabecera que expone el API PKCS v2 11	x	x	x
	beidpkcs11.dll	Libreria dinamica para Windows	x		
	libbeidpkcs11.so	Libreria dinamica para Linux		x	
	libbeidpkcs11.dylib	Libreria dinamica para Mac			x
	asepkcs.dll	Libreria PKCS de Athena para Firma Digital del Banco Central	x		
	libASEP11.so	Libreria PKCS de Athena para Firma Digital del Banco Central		x	
	libASEP11.dylib	Libreria PKCS de Athena para Firma Digital del Banco Central			x
Java					
	iaikPkcs11Wrapper.jar	Wrapper IAIK PKCS11 para Java JNI	x	x	x
	PKCS11Wrapper.dll	Wrapper IAIK PKCS11 para Java	x		
	libpkcs11wrapper.so	Wrapper IAIK PKCS11 para Java Linux		x	
	libpkcs11wrapper.dylib	Wrapper IAIK PKCS11 para Java Mac			x
dotNet					
	Net.Pkcs11.dll	Wrapper PKCS11 para C#	x		
	BouncyCastle.Crypto.dll	Libreria auxiliar BouncyCastle para C#	x		

Los componentes listados en la tabla anterior para **Java y dotNet** son librerías de terceros y no son incluidos en el código de la solución Middleware - Applet. Son presentados en el presente documento únicamente con fines educativos y de enseñanza. El desarrollador puede utilizar con toda libertad

cualquier otra implementación de wrappers tanto para Java como dotNet dependiendo el tipo de proyecto de Desarrollo de Software al que se esté enfrentando.

Inicializando /Finalizando el modulo PKCS#11

Para C/C++ se debe incluir el archivo de cabecera estándar "**pkcs11.h**" y llamar a la función "**C_Initialize**" / "**C_Finalize**":

Ejemplo C++

```
CK_ULONG beidsdk_GetData()
{
	void *pkcs11Handle;                                          //handle to the
pkcs11 library
	CK_FUNCTION_LIST_PTR pFunctions;          //list of the pkcs11 function pointers
	CK_C_GetFunctionList pC_GetFunctionList;
	CK_RV retVal = CKR_OK;

	//open the pkcs11 library
	pkcs11Handle = dlopen("beidpkcs11.dll", RTLD_LAZY); // RTLD_NOW is slower
	if (pkcs11Handle != NULL)
	{
		// get function pointer to C_GetFunctionList
		pC_GetFunctionList = (CK_C_GetFunctionList)dlsym(pkcs11Handle,
"C_GetFunctionList");
		if (pC_GetFunctionList != NULL)
		{
			// invoke C_GetFunctionList to get the list of pkcs11 function
pointers
			retVal = (*pC_GetFunctionList) (&pFunctions);
			if (retVal == CKR_OK)
			{
				// initialize Cryptoki
				retVal = (pFunctions->C_Initialize) (NULL);
				if (retVal == CKR_OK)
				{
.....

	retVal = (pFunctions->C_Finalize) (NULL_PTR);
.....

}
```

Ejemplo C#

```
using Net.Sf.Pkcs11;
using Net.Sf.Pkcs11.Objects;
using Net.Sf.Pkcs11.Wrapper;

namespace CSmwEIDTest
{
    class PKCS11Controller
    {
        ....
        private Net.Sf.Pkcs11.Module m_Module = null;
        ....
        m_Module = Module.GetInstance("beidpkcs11.dll");
   .....
}
```

Ejemplo Java

```
package javamweidtest;

import iaik.pkcs.pkcs11.*;
import iaik.pkcs.pkcs11.objects.*;
import iaik.pkcs.pkcs11.wrapper.PKCS11Constants;
....

public class PKCS11Controller {
    ....
    private Module m_Module = null;
    m_Module = Module.getInstance("beidpkcs11.dll"););
    m_Module.initialize(null);
    ....
    m_Module.finalize(null);
    ....
}
```

Accediendo a una Tarjeta Inteligente

Luego de inicializar la librería, el sistema está listo para acceder a una tarjeta, para ello se debe seguir la siguiente secuencia:

- Obtener una lista de lectores de tarjeta para seleccionar un lector de tarjeta
- Verificar si la tarjeta está presente
- Obtener los objetos de datos

Obtener lista de lectores de tarjeta

Para obtener la lista de lectores de tarjeta en C/C++ se debe llamar a la función "**C_GetSlotList**"

Ejemplo C++

……

```
// retrieve the list of slots (cardreaders)
retVal = (pFunctions->C_GetSlotList) (CK_TRUE, slotIds, &slot_count);
```

…….

Ejemplo C#

…..

```
// GetSlotList.
Net.Sf.Pkcs11.Slot[] m_Slots = m_Slots = m_Module.GetSlotList(true);
```

…..

Ejemplo Java

…..

```
// GetSlotList
iaik.pkcs.pkcs11.Slot[] m_Slots = m_Module.getSlotList(true);
```

……

Verificar si la Tarjeta está presente

Este paso se realiza abriendo una sesión en la tarjeta ("**C_OpenSession**"), si la tarjeta está presente, será posible luego abrir la tarjeta utilizando el código PIN ("**C_Login**").

Ejemplo C++

....

```
       CK_SESSION_HANDLE session_handle;
       //open a session
       retVal = (pFunctions->C_OpenSession)(slotIds[slotIdx], CKF_SERIAL_SESSION,
NULL_PTR, NULL_PTR, &session_handle);
       if (retVal == CKR_OK)
       {
           CK_CHAR_PTR pPIN = (CK_CHAR_PTR)TEXT("1234");
           retVal = (pFunctions->C_Login)(session_handle, CKU_USER, pPIN, (CK_ULONG) 4) ;
           if (retVal == CKR_OK)
           {
               printf("LOGIN Successful\n");
           }
```

.....

Ejemplo C#

.....

```
                    Slot slot = m_Slots[in_SlotIndex];
                    Session session = slot.Token.OpenSession(false);
                    m_CurrentIndex = in_SlotIndex;
                    session.Login(UserType.USER, "1234");
```

.....

Ejemplo Java

.....

```
                Slot slot = m_Slots[in_SlotIndex];
                Session session =
slot.getToken().openSession(Token.SessionType.SERIAL_SESSION,
                        Token.SessionReadWriteBehavior.RO_SESSION, null, null);
                m_CurrentIndex = in_SlotIndex;
                session.login(Session.UserType.USER, in_PIN.toCharArray());
```

.....

Obtener los datos - Certificados

La Tarjeta Inteligente de Firma Digital de Costa Rica almacena 2 Certificados Digitales con sus respectivas claves privadas.

Uno de los certificados es utilizado para Autenticación y el otro para Firma Digital/No-Repudio

Para obtener datos que se encuentran almacenados en la tarjeta digital, es necesario realizar una búsqueda de los objetos llamando a la función "**C_FindObjects** " . Como requisito es imprescindible preparar la búsqueda indicando la plantilla de atributos que definirán la búsqueda de objetos. Esto se realiza llamando a la función "**C_FindObjectsInit** " pasando un arreglo de "**Attributes**" como parámetro.
Luego de cada búsqueda de objetos, ya sea que se hayan encontrado los objetos interesados o nó, es imprescindible finalizar el procedimiento con una llamada a la función "**C_FindObjectsFinal**".

Los siguientes ejemplos muestran como obtener los certificados digitales, especificando el Atributo **CKO_CERTIFICATE**:

Ejemplo C#

```
.....

ObjectClassAttribute classAttribute = new ObjectClassAttribute(CKO.CERTIFICATE);

session.FindObjectsInit(new P11Attribute[] {
            classAttribute
        }
        );
P11Object[] certificates = session.FindObjects(2) as P11Object[];
if (certificates.Length == 2)
{
    //Label de Certificado de Autenticacion
   ((X509PublicKeyCertificate)certificates[0]).Label.Value));
    //Label de Certificado de Firma Digital
   ((X509PublicKeyCertificate)certificates[1]).Label.Value));
}

session.FindObjectsFinal();
.....
```

Ejemplo Java

.....

```
GenericTemplate certificateSearchTemplate = new GenericTemplate();
LongAttribute objectClassAttribute = new LongAttribute(PKCS11Constants.CKA_CLASS);
objectClassAttribute.setLongValue(new Long(PKCS11Constants.CKO_CERTIFICATE));
certificateSearchTemplate.addAttribute(objectClassAttribute);
LongAttribute certificateTypeAttribute = new
LongAttribute(PKCS11Constants.CKA_CERTIFICATE_TYPE);
certificateTypeAttribute.setLongValue(new Long(PKCS11Constants.CKC_X_509));
certificateSearchTemplate.addAttribute(certificateTypeAttribute);

session.findObjectsInit(certificateSearchTemplate);
//P11Object
iaik.pkcs.pkcs11.objects.Object[] certificates = session.findObjects(2);
if (certificates.length == 2) {
    //Label de Certificado de Autenticacion
    ((X509PublicKeyCertificate) certificates[0]).getLabel().getCharArrayValue();
    //Label de Certificado de Firma Digital
    ((X509PublicKeyCertificate) certificates[1]).getLabel().getCharArrayValue();
}

session.findObjectsFinal();
```

.....

Obtener los datos - Claves Privadas

La Tarjeta Inteligente de Firma Digital de Costa Rica almacena 2 claves privadas para los Certificados Digitales que contiene.

Los siguientes ejemplos muestran cómo obtener las claves privadas, especificando el Atributo **CKO_PRIVATE_KEY**

Ejemplo C#

.....

```
ObjectClassAttribute classAttribute = new ObjectClassAttribute(CKO.PRIVATE_KEY);
ByteArrayAttribute keyLabelAttribute = new ByteArrayAttribute(CKA.LABEL);
```

```
keyLabelAttribute.Value = System.Text.Encoding.UTF8.GetBytes("Label Certificado de Firma
Digital");

session.FindObjectsInit(new P11Attribute[] {
            classAttribute,
            keyLabelAttribute
        }
        );
P11Object[] privatekeys = session.FindObjects(1) as P11Object[];
session.FindObjectsFinal();
```

.....

Ejemplo Java

.....

```
GenericTemplate privateKeySearchTemplate = new GenericTemplate();
LongAttribute classAttribute = new LongAttribute(PKCS11Constants.CKA_CLASS);
classAttribute.setLongValue(new Long(PKCS11Constants.CKO_PRIVATE_KEY));
privateKeySearchTemplate.addAttribute(classAttribute);

ByteArrayAttribute keyLabelAttribute = new ByteArrayAttribute(PKCS11Constants.CKA_LABEL);

byte[] label = ("Label Certificado de Firma Digital").getBytes("UTF-8");

keyLabelAttribute.setByteArrayValue(label);
privateKeySearchTemplate.addAttribute(keyLabelAttribute);

session.findObjectsInit(privateKeySearchTemplate);

iaik.pkcs.pkcs11.objects.Object[] privatekeys = session.findObjects(1);

session.findObjectsFinal();
```

.....

Autenticación y Firmado digital

Los procedimientos de autenticación y Firmado digital, realizando validación de certificados digitales ya sea utilizando CRL o OCSP, se encuentra fuera del alcance del presente documento puesto que el Middleware no provee de un soporte transparente para estas funcionalidades. Existe abundante literatura en Internet sobre cómo realizar estos procedimientos, los mismos que podrán ser utilizados para aprovecha las funcionalidades expuestas por el Middleware siguiendo el estándar RSA - PKCS v2#11.

Un ejemplo práctico y funcional de cómo realizar estos procedimientos tanto en C# como en Java, se puede obtener desde el sitio SVN de Google que contiene el código de la solución de MiddleWare y Applet del Micitt para el DCFD que está actualmente ubicado en:

http://dcfd-mw-applet.googlecode.com

Aplicación demostrativa - C#

El código fuente y proyecto de Microsoft Visual Studio 2010 de la aplicación demostrativa en C# **CSmwEIDTest** puede ser obtenido desde el sitio SVN de Google en la ruta:

trunk\aplicaciones_demostrativas\CS\CSmwEIDTest_VisualStudio-2010

El detalle de utilización de la aplicación demostrativa en C# puede ser encontrado en el documento denominado:

4.1 Manual Aplicación Demostrativa - CSmwEIDTest.pdf

Aplicación demostrativa - Java

El código fuente y proyecto de NetBeans 6.8 de la aplicación demostrativa en Java **JAVAmwEIDTest** puede ser obtenido desde el sitio SVN de Google en la ruta:

trunk\aplicaciones_demostrativas\JAVA\JAVAmwEIDTest_NetBeans-6.8

El detalle de utilización de la aplicación demostrativa en Java puede ser encontrado en el documento denominado:

4.2 Manual Aplicación Demostrativa - JAVAmwEIDTest.pdf