certicámara.

Validez y seguridad jurídica electrónica

Componente de Verificación de Firma Digital en JAVA

Documento de especificación y contexto:

Versión 1.0

IDENTIFICADOR	Verify4J.pdf
NOMBRE DEL DOCUMENTO	Documento inicial de documentación.
ESTADO DEL DOCUMENTO	-----
ÁREA	Consultoría
RESPONSABLES	**Jhon Fredy Triana Marin** (jhon.trian@certicamara.com)
REVISORES	**Carlos Orlando Peña** (carlos.pena@certicamara.com)

CONTROL DE VERSIONES DEL DOCUMENTO

Versión	Fecha creación	Fecha liberación	Descripción cambio

CONTROL DE REVISIONES Y APROBACIONES

Revisado por	Firma	Fecha

Tabla de Contenidos

Documento de Especificación y contexto del Componente de Verificación de Firma Digital en Java

1. INTRODUCCIÓN

Certicámara S.A. ha desarrollado un Componente API que permite verificar las firmas de un documento en formato PKCS#7 Attached y Detached, así como en formatos XML Enveloped, Enveloping y archivos en formato PDF.

2. OBJETIVOS

- Documentar el uso de las funcionalidades y el uso del Componente de Verificación de Firma Digital en Java.

3. VERSIÓN DEL SERVICIO

- 1.0

4. DUEÑOS DEL SERVICIO

Responsable	Certicámara – Ing. Carlos Peña.
Responsable	Área: Consultoría
Informado	

5. FUNCIONALIDADES DEL API.

Este componente API permite verificar las firmas digitales en un documento. Los tipos de documentos que puede verificar este componente API son:

- PKCS#7 Attached
- PKCS#7 Detached
- XML Enveloped
- XML Enveloping
- PDF (PaDES)

Para verificar un documento se debe especificar en una clase de parámetros toda la configuración requerida para verificar la firma, y para validar la revocación del certificado digital.

5.1. Creación de un objeto de verificación firma

La creación de un objeto de verificación de firma, se encapsuló en la clase ***VerifySignFactory***,

la cual que tiene un método estático *getVerifier* que recibe un String que indica cual tipo de verificador se requiere (PKCS#7 Attached, XML Enveloped, etc.) y recibe una variable de tipo SignParameters que es donde se encapsula toda la configuración requerida para la verificación de la firma de un documento . A continuación se presenta un ejemplo de verificación de firma sobre un documento (signParameters) de tipo PKCS#7 Attached.

```
VerifySign v =
VerifySignFactory.getVerifier(VerifySignFactory.PKCS7AttachedFile,
signParameters);
v.verifyDocument(false);
```

El booleano que recibe como parámetro el método verifyDocument, indica si se debe realizar la verificación del certificado asociado a la firma indica si se hace o no la verificación de los certificados.

5.1.1. Método *getVerifier* de la clase *VerifySignFactory*.

El método *getVerifier* de la clase ***VerifySignFactory*** recibe dos parámetros, el primero es un String que indica el tipo de firma, podemos elegir entre diferentes atributos estáticos de la clase ***VerifySignFactory***, estos atributos son:

Atributo	Valor	Tipo
PKCS7AttachedFile	pkcs7.attached.VerifyPKCS7Attached	String
PKCS7detachedFile	pkcs7.detached.VerifyPKCS7Detached	String
XMLenvelopedFile	xml.enveloped.VerifyXMLEnveloped	String
XMLenvelopingFile	xml.enveloping.VerifyXMLEnveloping	String
PDFFile	pdf.VerifyPDF	String

El parámetro SignParameters se debe definir de la siguiente manera:

```
SignParameters signParameters = new SignParameters();
signParameters.setDocumentSigned(Util.getBytesFromFile(path));
signParameters.setRevocationValidation(revocationVerify);
```

En donde la variable *signParameters* es una implementación de la clase ***SignParameters, que depende del formato de firma a verificar***. Esta variable trae los parámetros requeridos para ejecutar la validación de la firma. Por lo general para verificar una firma normalmente se requieren solo los bytes y el tipo de verificación de revocación, pero en las firmas dettached, se requieren los bytes del documento original con el fin de validar la integridad de la firma

5.1.2. SignParameters.

Esta es la clase de configuración para realizar una firma en los formatos que soporta el API. El constructor de esta clase, como ya ha sido indicado, recibe:

- **byte[]** *documentSigned*: Son los bytes de la firma.
- **RevocationVerify** *revocationVerify*: es la configuración para validar los certificados con los que se firmó el documento.

- **signedContentDetached** *signedContentDetached*: bytes del contenido firmado (en firma detached)
- **String** *pdfPasswordCypher*: es el String del password del pdf cifrado.

5.1.3. RevocationVerify

El componente API de verificación, también verifica la revocación de los certificados utilizados al momento de generar la firma. Para realizar esta verificación, se debe configurar el objeto ***RevocationVerify*** de la siguiente manera:

```
RevocationVerify p = new RevocationVerify("CRL_ONLY",new
GregorianCalendar(),"c:/temp/crl","http://oscp.certicamara.com");

p.setKeyStorePath("c:/temp/keystore/Keystore");
```

El constructor recibe 4 parámetros que son los siguientes en orden:

Atributo	Valor	Tipo
Tipo de verificación de Revocación	Tiene 4 posible valores: “CRL_ONLY” “OSCP_ONLY” “OSCP_CRL” “CRL_OSCP”	String
Calendar	Fecha para la cual se quiere validar la validez de la firma, si es null toma la fecha actual del sistema.	Calendar
Dirección CRL	Se indica la ruta absoluta de la carpeta en la cual se encuentran las crl.	String
Dirección OCSP	Se indica la dirección URL en donde se válida la revocación por OCSP.	String

Sobre este objeto, también se debe configurar la ruta absoluta en donde se encuentra el keystore que contiene los certificados públicos de las entidades de confianza, como el de la CA raíz y subordinada de Certicámara. Con esta configuración ya es posible validar los certificados utilizados para generar la firma.

5.1.4. Respuesta del VerifySign: ProcessResponseVerify processResponseVerify.

La respuesta del proceso de verificación se ve reflejada en un objeto de tipo ProcessResponseVerify, el cual tiene los siguientes elementos:

Atributo	Valor	Tipo

Firmantes	Lista de los firmantes del documento	ArrayList<String>
DocumentClear	Son los bytes del documento firmado.	Byte[]

Un ejemplo sobre como verificar un documento:

```
ProcessResponseVerify processResponseVerify = v.verifyDocument(false);
System.out.println(processResponseVerify.isExito());
if (!processResponseVerify.isExito()) {
	for (MessageResponse i : processResponseVerify.getMessageResponse()) {
		System.out.println("Código: " + i.getCodigo());
		System.out.println("Mensaje: " + i.getMensaje());
	}
}
else {
	for (String i : processResponseVerify.getFirmantes()) {
		System.out.println(i);
	}
}
```

En el anterior ejemplo, se verifican todas las firmas de un archivo, si la respuesta es exitosa, se imprimen los datos de cada uno de los firmantes, si la respuesta es negativa, se imprimen todos los errores que se presentaron durante la validación
A continuación se explica cómo se debe invocar el componente, para cada uno de los formatos de firma que se pueden verificar:

5.2. VERIFICACIÓN DE FIRMA EN FORMATO PKCS#7

. Para la verificación de una firma en este formato se requiere la ruta del archivo .p7z o .p7m Attached y Detached respectivamente, crear un objeto SingParameters, en donde se guardan todos los parámetros que necesita la verificación, y luego se construye un objeto VerifySign cuyo constructor recibe como parámetros el tipo de verificación que se quiere realizar y el objeto SingParameters, luego se llama al método ***verifyDocument***.

5.2.1. Verificación de firma en formato PKCS#7 Attached

Para realizar la verificación en formato PKCS#7 Attached se debe enviar como parámetro en el método getVerifier del VerifySignFactory el tipo de verificación igual a *PKCS7AttachedFile*. Un ejemplo de verificación es el siguiente:

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = pathResources + "/pkcs7Attached/pkcs7testAttached.p7z";
SignParameters signParameters = new SignParameters();
```

```
signParameters.setDocumentSigned(Util.getBytesFromFile(path));
RevocationVerify revocationVerify = new RevocationVerify(VerifyType.CRL_ONLY, new
GregorianCalendar(), null, "http://ocsp.certicamara.com");
revocationVerify.setKeyStorePath("C:/KeyStore/KeyStore");
signParameters.setRevocationValidation(revocationVerify);
VerifySign v = VerifySignFactory.getVerifier(VerifySignFactory.PKCS7AttachedFile,
signParameters);
ProcessResponseVerify processResponseVerify = v.verifyDocument(false);
if(!processResponseVerify.isExito()){
	for(MessageResponse i : processResponseVerify.getMessageResponse()){
		System.out.println("Código: " + i.getCodigo());
		System.out.println("Mensaje: " + i.getMensaje());
	}
}
else{
	for (String i : processResponseVerify.getFirmantes()){
		System.out.println(i);
	}
}
```

5.2.2. Verificación de firma y estampa en formato PKCS#7 Attached

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = pathResources + "/pkcs7Attached/pkcs7testAttached.p7z";
SignParameters signParameters = new SignParameters();
signParameters.setDocumentSigned(Util.getBytesFromFile(path));
RevocationVerify revocationVerify = new RevocationVerify(VerifyType.CRL_ONLY, new
GregorianCalendar(), null, "http://ocsp.certicamara.com");
revocationVerify.setKeyStorePath("C:/KeyStore/KeyStore");
signParameters.setRevocationValidation(revocationVerify);
VerifySign v = VerifySignFactory.getVerifier(VerifySignFactory.PKCS7AttachedFile,
signParameters);
ProcessResponseVerify processResponseVerify = v.verifyDocument(false);
if(!processResponseVerify.isExito()){
	for(MessageResponse i : processResponseVerify.getMessageResponse()){
		System.out.println("Código: " + i.getCodigo());
		System.out.println("Mensaje: " + i.getMensaje());
	}
}
else{
	for (String i : processResponseVerify.getFirmantes()){
		System.out.println(i);
	}
}
//Se recuperan los ceritificados publicos con los que se firmó el documento
		LinkedHashMap<String, Certificate> a = v.getSigners();
```

```
            //Se recorre la lista de certificados
            for(String key : a.keySet()){
                  System.out.println("--------------------------------");
                  //se imprime el OID del firmante.

      System.out.println("*"+((Certificate)a.get(key)).getSignerInfo().getSignerDN());
                  //se imprime la fecha en la que se firmó
                  System.out.println("-
"+((Certificate)a.get(key)).getSignerInfo().getSignDate());
                  //Se imprime el OID total
                  System.out.println("´´"+key);
                  //se imprime la vigencia del certificado
                  System.out.println("*Certficado
desde:"+((Certificate)a.get(key)).getNotBefore());
                  System.out.println("*Certficado
hasta:"+((Certificate)a.get(key)).getNotAfter());
                  if(((Certificate)a.get(key)).isTimeStamped())
                        //Se imprime la fecha de la estampa
                        System.out.println("*timeStamp:
"+((Certificate)a.get(key)).getTimeStamp());
            }
```

5.2.3. Verificación de Firma en formato PKCS#7 Detached

Para realizar el firmado en formato PKCS#7 Detached se debe enviar como parámetro en el método getVerifier del VerifySignFactory el tipo de verificación igual a *PKCS7detachedFile*. Un ejemplo de verificación es el siguiente:

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = pathResources +"/pkcs7Detached/archivo.xls.p7m";
String pathClear = pathResources + "/pkcs7Detached/archivo.xls";
SignParameters signParameters = new SignParameters();
signParameters.setDocumentSigned(Util.getBytesFromFile(path));
signParameters.setSignedContentDetached(Util.getBytesFromFile(pathClear));
String crlPath = "C:/CRL/ac_subordinada_certicamara.crl";
signParameters.setRevocationValidation(new
RevocationVerify(VerifyType.OCSP_ONLY,new
GregorianCalendar(),"C:/VerifyWebService/CRL/","http://ocsp.certicamara.com"));
      VerifySign v =
VerifySignFactory.getVerifier(VerifySignFactory.PKCS7detachedFile,
signParameters);
ProcessResponseVerify processResponseVerify = v.verifyDocument(true);
System.out.println(processResponseVerify.isExito());
if (!processResponseVerify.isExito()) {
      for (MessageResponse i : processResponseVerify.getMessageResponse()) {
            System.out.println("Código: " + i.getCodigo());
            System.out.println("Mensaje: " + i.getMensaje());
      }
```

```
}
else {
	for (String i : processResponseVerify.getFirmantes()) {
		System.out.println(i);
	}
}
```

5.3. VERIFICACIÓN DE FIRMA EN FORMATO XML

Para la verificación de una firma en este formato se requiere a ruta del archivo .xml, crear un objeto SingParameters, en donde se guardan todos los parámetros que necesita la verificación, y luego se construye un objeto VerifySign cuyo constructor recibe como parámetros el tipo de verificación que se quiere realizar y el objeto SingParameters, luego se llama al método ***verifyDocument***.

5.3.1. Firma en formato XML Enveloped

Para realizar el firmado en formato XML Enveloped se debe enviar como parámetro en el método getVerifier del VerifySignFactory el tipo de verificación igual a *XMLenvelopingFile*. A continuación se presenta un ejemplo de verificación de este formato:

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = pathResources + "/xmlEnveloped/ArchivoFirmadoEnveloping.xml";
try{
	SignParameters signParameters = new SignParameters();
	signParameters.setDocumentSigned(Util.getBytesFromFile(path));
	RevocationVerify revocationVerify = new RevocationVerify(VerifyType.OCSP_ONLY,
new GregorianCalendar(), "C:/CRL/", "http://ocsp.certicamara.com");
	revocationVerify.setKeyStorePath(pathResources+"/keystore/Keystore");
	signParameters.setRevocationValidation(revocationVerify);
	VerifySign v =
VerifySignFactory.getVerifier(VerifySignFactory.XMLenvelopingFile, signParameters);
	ProcessResponseVerify processResponseVerify = v.verifyDocument(true);
	if(processResponseVerify.isExito()){
		System.out.println("Verificación exitosa");
		System.out.println("Firmantes");
		for(String subject : processResponseVerify.getFirmantes()){
			System.out.println(subject);
		}
		System.out.println("Texto claro");
		System.out.println(new
String(processResponseVerify.getDocumentClear()));

	}
	else{
		System.out.println("Verificación no exitosa");
		for(MessageResponse messageResponse :
```

```
processResponseVerify.getMessageResponse()){
                    System.out.println("Código: " + messageResponse.getCodigo() +
"\nMensaje: " + messageResponse.getMensaje());
              }
              System.out.println();
       }
}
catch (Exception e) {
       e.printStackTrace();
}
```

5.3.2. Firma en formato XML Enveloping

Para realizar el firmado en formato XML Enveloping se debe enviar como parámetro en el método getVerifier del VerifySignFactory el tipo de verificación igual a *XMLenvelopingFile*. A continuación se presenta un ejemplo de verificación de este formato:

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = "C:/hola.xml.enveloped.xml";
try {
       SignParameters signParameters = new SignParameters();
       signParameters.setDocumentSigned(Util.getBytesFromFile(path));
       RevocationVerify revocationVerify = new RevocationVerify(VerifyType.OCSP_ONLY,
new GregorianCalendar(), "C:/VerifyWebService/CRL/", "http://ocsp.certicamara.com");
       revocationVerify.setKeyStorePath("C:/KeyStore/CAKeyStoreLUSI_ISP");
       signParameters.setRevocationValidation(revocationVerify);
       VerifySign v =
VerifySignFactory.getVerifier(VerifySignFactory.XMLenvelopedFile, signParameters);
       ProcessResponseVerify processResponseVerify = v.verifyDocument(true);
       if (processResponseVerify.isExito()) {
              System.out.println("Verificación exitosa");
              System.out.println("Firmantes");
              for (String subject : processResponseVerify.getFirmantes()) {
                     System.out.println(subject);
              }
              System.out.println("Texto claro");
              System.out.println(new
String(processResponseVerify.getDocumentClear()));
       }
       else {
              System.out.println("Verificación no exitosa");
              for (MessageResponse messageResponse :
processResponseVerify.getMessageResponse()) {
                     System.out.println("Código: " + messageResponse.getCodigo() +
"\nMensaje: " + messageResponse.getMensaje());
              }
              System.out.println();
```

```
		}
	}
	catch (Exception e) {
		e.printStackTrace();
	}
```

5.4. FIRMA EN FORMATO PDF

Para la verificación de una firma en este formato se requiere la ruta del archivo .pdf, crear un objeto SingParameters, en donde se guardan todos los parámetros que necesita la verificación, y luego se construye un objeto VerifySign cuyo constructor recibe como parámetros el tipo de verificación que se quiere realizar y el objeto SingParameters, luego se llama al método ***verifyDocument***.

5.4.1. Firma en formato PDF

Para realizar la verificacion en formato PDF se debe enviar como parámetro en el método getVerifier del VerifySignFactory el tipo de verificación igual a *PDFFile*. A continuación se presenta un ejemplo de verificación de este formato

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = pathResources + "/PDF/Documento_ValidoConEstampa.pdf";
try{
	SignParameters signParameters = new SignParameters();
	signParameters.setDocumentSigned(Util.getBytesFromFile(path));
	RevocationVerify revocationVerify = new
	RevocationVerify("CRL_ONLY",null,"C:/CRL/crl",null);
	revocationVerify.setKeyStorePath(pathResources+"/keystore/Keystore");
	signParameters.setRevocationValidation(revocationVerify);
	VerifySign v = VerifySignFactory.getVerifier(VerifySignFactory.PDFFile,
signParameters);
	ProcessResponse processResponse;
	processResponse = v.verifyDocument(true);
	if(processResponse.isExito()){
		System.out.println("Verificación exitosa");
	}
	else{
		System.out.println("Verificación no exitosa");
		for(MessageResponse i : processResponse.getMessageResponse()){
			System.out.println(i.getCodigo());
			System.out.println(i.getMensaje());
		}
	}
	LinkedHashMap<String, Certificate> a = v.getSigners();
	for(String key : a.keySet()){
		//se imprime el OID del firmante.
System.out.println(((Certificate)a.get(key)).getSignerInfo().getSignerDN());
```

```
            //se imprime la fecha en la que se firmó
System.out.println(((Certificate)a.get(key)).getSignerInfo().getSignDate());
      }
}
catch (Exception e) {
      e.printStackTrace();
}
```

5.4.2. Firma en formato PDF con estampa

```
File fileAux = new File("src/test/resources");
String pathResources = fileAux.getAbsolutePath();
String path = pathResources + "/PDF/Documento_ValidoConEstampa.pdf";
try{
      SignParameters signParameters = new SignParameters();
      signParameters.setDocumentSigned(Util.getBytesFromFile(path));
      RevocationVerify revocationVerify = new
      RevocationVerify("CRL_ONLY",null,"C:/CRL/crl",null);
      revocationVerify.setKeyStorePath(pathResources+"/keystore/Keystore");
      signParameters.setRevocationValidation(revocationVerify);
      VerifySign v = VerifySignFactory.getVerifier(VerifySignFactory.PDFFile,
signParameters);
      ProcessResponse processResponse;
      processResponse = v.verifyDocument(true);
      if(processResponse.isExito()){
            System.out.println("Verificación exitosa");
      }
      else{
            System.out.println("Verificación no exitosa");
            for(MessageResponse i : processResponse.getMessageResponse()){
                  System.out.println(i.getCodigo());
                  System.out.println(i.getMensaje());
            }
      }
      LinkedHashMap<String, Certificate> a = v.getSigners();
      for(String key : a.keySet()){
            //se imprime el OID del firmante.
System.out.println(((Certificate)a.get(key)).getSignerInfo().getSignerDN());
            //se imprime la fecha en la que se firmó
System.out.println(((Certificate)a.get(key)).getSignerInfo().getSignDate());
           //se imprime la fecha en la que se estampo
System.out.println("*Estampa: "+((Certificate)a.get(key)).getTimeStamp());
      }
}
catch (Exception e) {
      e.printStackTrace();
}
```