



Kit de Bienvenida SmartWrapper

© Izenpe s.a. 2009

Este documento es propiedad de Izenpe, s.a. y su contenido es confidencial. Este documento no puede ser reproducido, en su totalidad o parcialmente, ni mostrado a otros, ni utilizado para otros propósitos que los que han originado su entrega, sin el previo permiso escrito de Izenpe, s.a.. En el caso de ser entregado en virtud de un contrato, su utilización estará limitada a lo expresamente autorizado en dicho contrato. Izenpe, s.a. no podrá ser considerada responsable de eventuales errores u omisiones en la edición del documento.

ÍNDICE

1.	INTRODUCCIÓN	3
2.	REQUISITOS	4
3.	IMPORTACIÓN DE LOS EJEMPLOS	5
4.	ARCHIVO DE CONFIGURACIÓN	7
5.	CONCLUSIONES FINALES	9
6.	EJECUCIÓN DE LOS EJEMPLOS	10
7.	EXPLICACIÓN DE LOS EJEMPLOS	11
8.	ANEXO 1. FORMATOS DE FIRMA	15
8.1	FORMATO DE FIRMA ELECTRÓNICA NO AVANZADA	15
8.1.1	NO XADES.....	15
8.2	FORMATO DE FIRMA ELECTRÓNICA AVANZADA.....	15
8.2.1	XADES- BES.....	15
8.2.2	XADES- T.....	16
8.2.3	XADES- C	16
8.2.4	XADES- XL.....	16
8.2.5	XADES- A	17
8.2.6	XADES- EPES.....	17
9.	ANEXO 2. PERFILES SOPORTADOS	18
9.1	FIRMA CMS/ PKCS#7	18
9.2	FIRMA S/ MIME.....	18
9.3	FIRMA PDF	18
9.4	FIRMA XML	19
9.5	ESTADO DEL CERTIFICADO.....	19
9.6	SELLO DE TIEMPO.....	19
9.7	NO REPUDIO	20
10.	GLOSARIO	21
10.1	XML- DSIG	21
10.2	TIPOS DE FIRMA	21
10.3	FORMATO RFC3161.....	22



1 INTRODUCCIÓN

En el presente documento se explicarán todas las pautas necesarias para que los ejemplos de SmartWrapper suministrados puedan ejecutarse sin ningún tipo de problema en la plataforma Eclipse o compatible.





zain



2 REQUISITOS

Los ejemplos han sido desarrollados bajo una versión de la plataforma Eclipse y una versión del compilador concretos. A fin de evitar posibles errores a la hora de ejecutarlos, se recomienda utilizar lo siguiente:

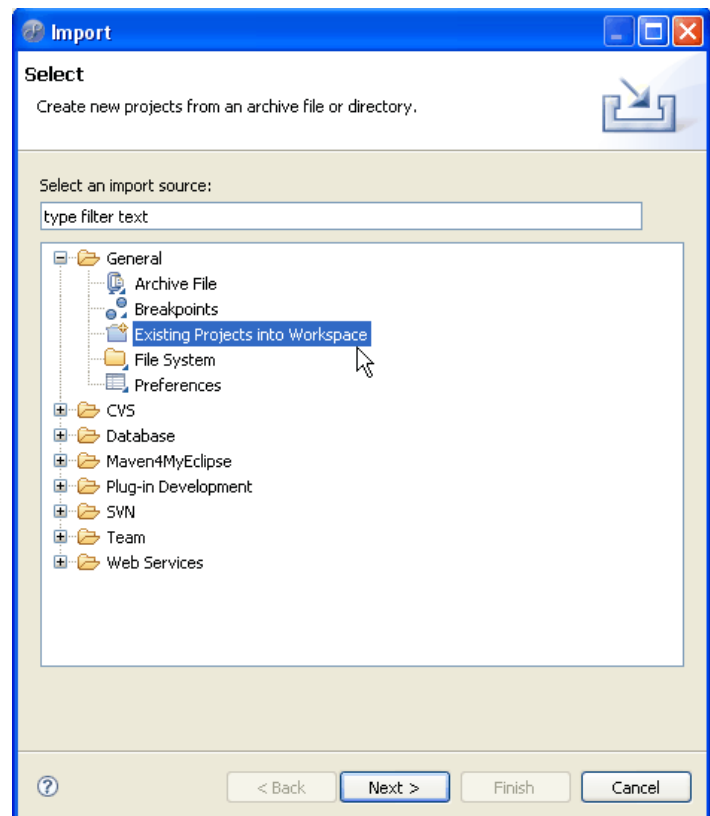
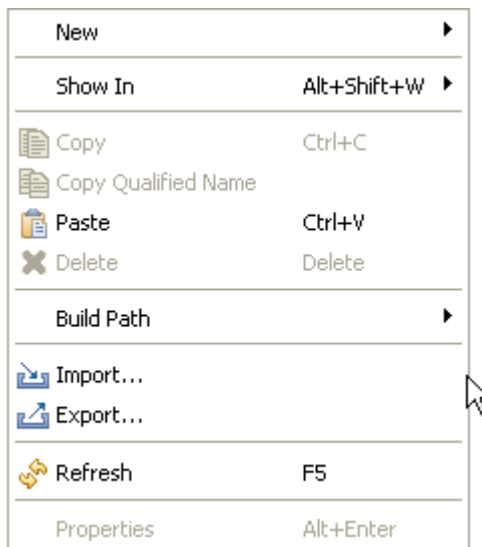
 Eclipse o basada en eclipse

 Java 2 SDK 1.4.2

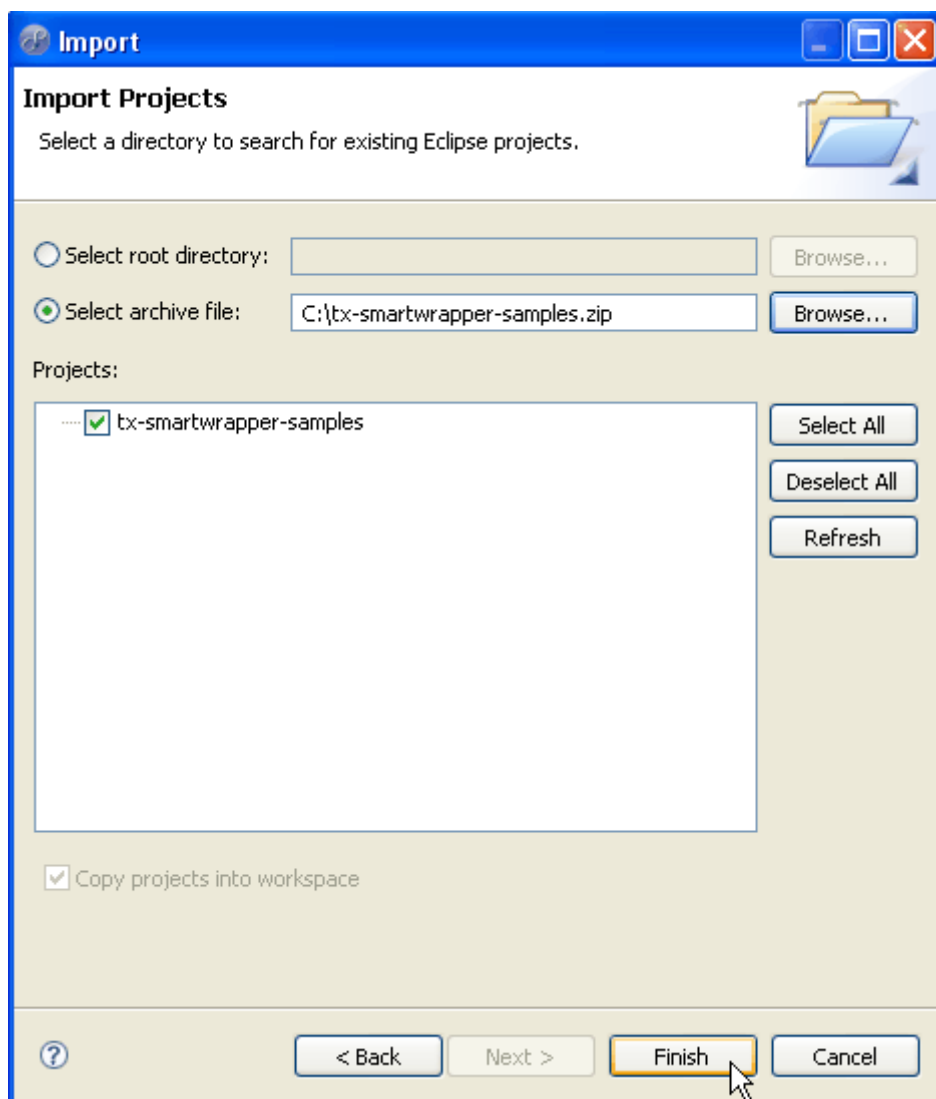
3 IMPORTACIÓN DE LOS EJEMPLOS

A continuación, se muestran los pasos a seguir para la correcta instalación de los ejemplos. Todos estos pasos se mostrarán con imágenes, intentando que sea lo más claro posible.

1. Tras abrir la aplicación Eclipse, habrá que darle a importar un proyecto existente al *workspace*.



2. Se selecciona el fichero .ZIP correspondiente a los ejemplos de SmartWrapper y se pulsa Finalizar.



4 ARCHIVO DE CONFIGURACIÓN

En este apartado se explicarán todos y cada uno de los parámetros más relevantes que conforman el archivo de configuración *smartwrapper.properties*. Por otra parte, en el punto 5. *Conclusiones Finales* de la documentación se pueden encontrar, a modo de recordatorio, aquellos parámetros de este fichero que habrá que configurar según Izenpe.

Este fichero se utiliza para poder configurar determinados parámetros de la parte cliente de un servicio Web. De esta forma, para que se procesen estos parámetros es necesario que este fichero se halle disponible en el classpath de la aplicación. La configuración de los diferentes parámetros que forman el fichero está pensada para intentar sacar el máximo rendimiento al transporte de los mensajes SOAP.

Por lo tanto, la estructura del archivo de configuración *smartwrapper.properties* presenta los siguientes elementos:

- ✚ *Truststore.active*: Tiene como valores true o false. Indica si el almacén de certificados definido en el parámetro *Truststore.path* es el que se utilizará para la conexión HTTP. Si vale false, entonces se utilizará el almacén de certificados que esté configurado (si lo hay) en la máquina virtual de Java. Por defecto su valor es false.
- ✚ *Truststore.path*: El almacén de certificados a utilizar en la conexión HTTP si el parámetro anterior, *Truststore.active*, vale true. A modo de ejemplo se utiliza el path *resources/application.truststore*. Se deberá definir por Izenpe. (Ver punto 5)
- ✚ *Truststore.password*: La contraseña del almacén de certificados definido en *Truststore.path*, a definir por Izenpe. (Ver punto 5)
- ✚ *Keystore.active*: Tiene como valores true o false. Indica si el almacén de claves definido en el parámetro *Keystore.path* es el que se utilizará para la conexión HTTP. Si vale false, entonces se utilizará el almacén de claves que esté configurado (si lo hay) en la máquina virtual de Java. Por defecto su valor es false.
- ✚ *Keystore.path*: El almacén de claves a utilizar en la conexión HTTP si el parámetro anterior, *Keystore.active*, vale true. A definir por Izenpe. (Ver punto 5)
- ✚ *Keystore.password*: La contraseña del almacén de claves definido en *Keystore.path*. A definir por Izenpe. (Ver punto 5)
- ✚ *Proxy.active*: Tiene como valores true o false. Indica si se va a utilizar un servidor proxy para la conexión HTTP.
- ✚ *Proxy.host*: El servidor proxy a utilizar si el parámetro *Proxy.active* vale true.
- ✚ *Proxy.port*: El puerto del servidor proxy.
- ✚ *Proxy.username*: Nombre de usuario para acceder al proxy si éste requiere autenticación HTTP básica.

- ✚ *Proxy.password*: Contraseña para acceder al proxy si éste requiere autenticación HTTP básica.
- ✚ *Ssl.allowCriticalExts*: Tiene como valores true o false. Indica si se permiten durante la conexión SSL certificados de servidor con cualquier extensión marcada como crítica. Valor por defecto false.
- ✚ *Ssl.noValidation*: Tiene como valores true o false. Indica si se anula la validación del certificado del servidor durante una conexión SSL. Valor por defecto false.
- ✚ *Ssl.validation*: El algoritmo de validación del certificado del servidor que se empleará durante una conexión SSL. Si se usa, debe ser un algoritmo implementado por un proveedor criptográfico registrado.
- ✚ *Timeout*: Tiempo de espera de la conexión HTTP, expresado en milisegundos. En este proyecto, se utiliza un timeout de 95.000 milisegundos, que equivale a 2 minutos más o menos.
- ✚ *Request.loadPath*: El directorio base de los archivos creados para enviar en las peticiones cuando se va a tratar con datos de gran tamaño.
- ✚ *Response.savePath*: El directorio base de los archivos creados (si así se indica en la petición) al recibir las respuestas cuando se va a tratar con datos de gran tamaño.
- ✚ *req-log.active*: Tiene como valores true o false. Indica si se guardarán archivos de log con el contenido de las peticiones SOAP enviadas a TrustedX. Por defecto su valor es false.
- ✚ *req-log.savePath*: El directorio en el que se guardarán los archivos de log con las peticiones enviadas a TrustedX si el parámetro anterior, *req-log.active*, vale true.
- ✚ *res-log.active*: Tiene como valores true o false. Indica si se guardarán archivos de log con el contenido de las respuestas enviadas por TrustedX. Por defecto su valor es false.
- ✚ *res-log.savePath*: El directorio en el que se guardarán los archivos de log con las respuestas enviadas por TrustedX si el parámetro anterior, *res-log.active*, vale true.
- ✚ *authN.policy*: La política de autenticación solicitada (opcional). Tiene como valor esperado el el identificador la política de autenticación que se solicita en la petición de servicio. Habrá que seleccionar una política en concreto, definida por Izenpe. (Ver punto 5)

5 CONCLUSIONES FINALES

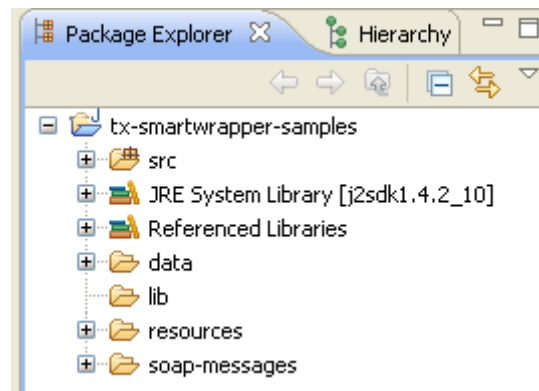
Seguir los pasos que se describen a continuación:

1. Copiar en la carpeta *resources/* el PKCS12 que usará la aplicación para autenticarse (SSL Mútuo).
2. Editar el fichero *src/smartwrapper.properties*:
 - a. En la línea que contiene el parámetro 'Keystore.path', Indicar la ruta al fichero PKCS12 copiado en el paso 1.
 - b. En la línea que contiene el parámetro 'Keystore.password', indicar el password del PKCS12 copiado en el paso 1.
 - c. En la línea que contiene el parámetro 'Truststore.path', indicar la ruta al fichero del truststore.
 - d. En la línea que contiene el parámetro 'Truststore.password', indicar el password del truststore.
 - e. En la línea que contiene el parámetro 'authN.policy', indicar la política de autenticación asignada por Izenpe a la aplicación.
3. En la clase */src/com/izenpe/trustedx/cliente/util/Propiedades.java*, asignar a la constante 'TRUSTEDX_SIGNER' el campo ASUNTO del certificado custodiado por la Plataforma para hacer firma en servidor. Será usado en los ejemplos de generación de firma.

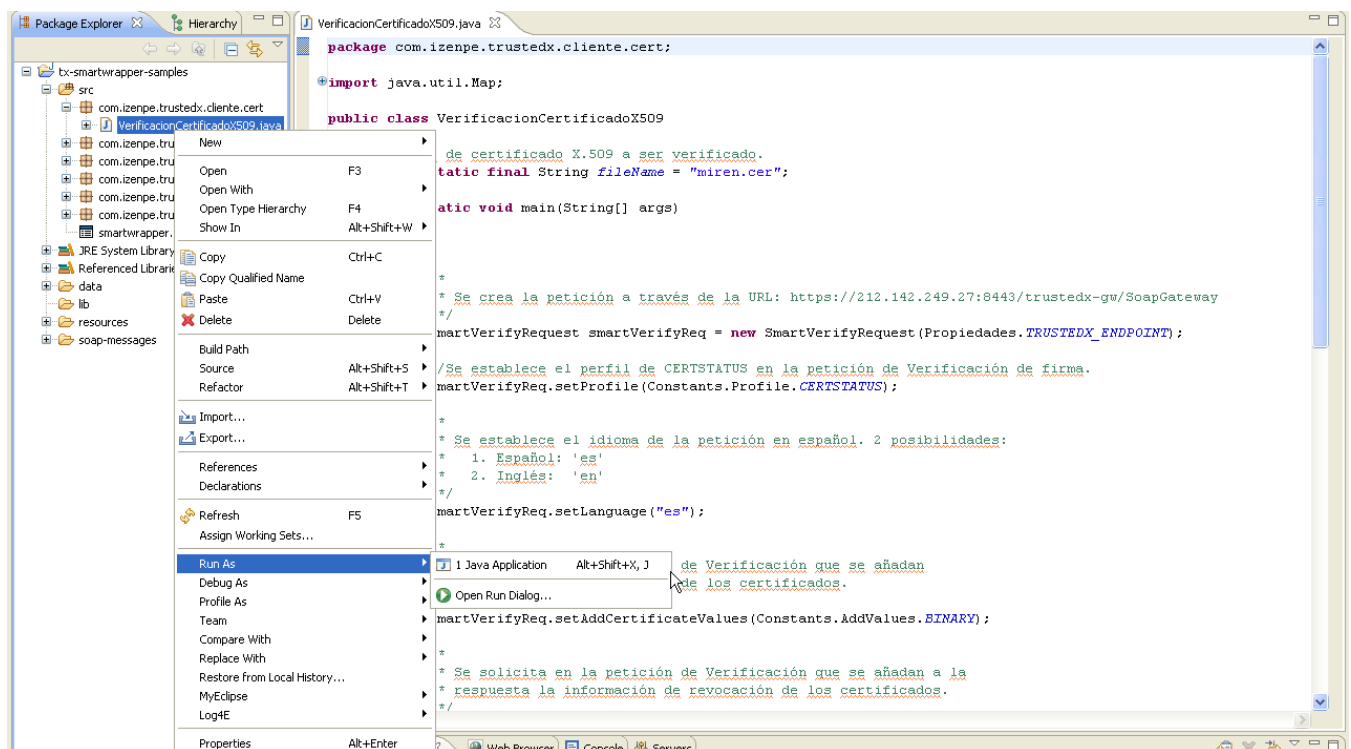
6 EJECUCIÓN DE LOS EJEMPLOS

Los pasos a seguir son estos:

1. En el explorador de paquetes aparecerá el proyecto con todos los ejemplos preparados para ser ejecutados.



2. Bastará con seleccionar un ejemplo (ver punto 7 para la ejecución ordenada de los diferentes ejemplos) y ejecutarlo como una aplicación Java para el funcionamiento.



3. Observar la consola para entender mejor el comportamiento de los ejemplos, tanto en la petición como en la respuesta.

7 EXPLICACIÓN DE LOS EJEMPLOS

En este apartado, se mostrará tanto una pequeña explicación de lo que hacen los ejemplos como la dinámica de ejecución de éstos, ya que hay algunos que dependen directamente de ficheros creados previamente por otros ejemplos.

Además de los diferentes paquetes de los que se componen los ejemplos, también se hará referencia a las distintas carpetas que se utilizan para la ejecución de éstos.

Paquete *com.izenpe.trustedx.cliente.cert*

En este paquete se muestra un ejemplo de verificación de un certificado X.509, utilizando para ello el perfil de Estado de Certificado correspondiente a la plataforma TrustedX. El ejemplo se llama *VerificacionCertificadoX509.java*. A partir de un certificado X.509 situado en *data/input/* se realizará la verificación de éste con el perfil ya comentado.

Se puede ejecutar sin ningún problema ya que no depende de otros ejemplos ubicados en el proyecto.

Paquete *com.izenpe.trustedx.cliente.pdf*

En este paquete se muestra un ejemplo de generación de firma en PDF, utilizando para ello el perfil de Firma PDF correspondiente a la plataforma TrustedX. El ejemplo se llama *GeneracionFirmaPDF.java*. A partir de un documento con extensión .pdf ubicado en *data/input/* generará otro documento .pdf con la correspondiente firma en *data/output/*.

Se puede ejecutar sin ningún problema ya que no depende de otros ejemplos ubicados en el proyecto.

Paquete *com.izenpe.trustedx.cliente.pkcs7*

En este paquete se muestran ejemplos sobre la generación, verificación y actualización de firmas PKCS#7 en el perfil de Firma CMS/PKCS7. Para ello, se usan tres perfiles distintos suministrados por la plataforma TrustedX.

En el ejemplo *GeneracionFirmaPKCS7.java* se usa el perfil de Firma CMS y tanto en el ejemplo *VerificacionFirmaPKCS7.java* como en el de *VerificacionFirmaPKCS7_ESC.java* el perfil de Verificación CMS. En cambio, para la operación de actualización, realizada por el ejemplo *ActualizacionFirmaPKCS7.java*, se utiliza el perfil de No Repudio.

Los ejemplos de este paquete se pueden dividir en dos grupos. Por un lado, están las clases *GeneracionFirmaPKCS7.java* y *VerificacionFirmaPKCS7.java*. Por otra parte, se pueden encontrar los ejemplos de *ActualizacionFirmaPKCS7.java* y *VerificacionFirmaPKCS7_ESC.java*.

En el primer grupo, el ejemplo inicial que hay que ejecutar es *GeneracionFirmaPKCS7.java*, ya que el fichero que se generará en la carpeta *data/output/* llamado 'HelloWorld.txt.p7d' será utilizado por el ejemplo *VerificacionFirmaPKCS7.java* para validarlo.

En el segundo grupo, el primer ejemplo a ejecutar es de *ActualizacionFirmaPKCS7.java*, ya que a través de la lectura de un fichero ubicado en *data/input/* llamado 'HelloWorld-activex.txt.p7s' guardará en disco (*data/output/*) un fichero con la actualización de la firma PKCS#7 llamado 'HelloWorld-activex-ESC.txt.p7s'. Después, habrá que ejecutar el ejemplo de *VerificacionFirmaPKCS7_ESC.java*, que hace uso del fichero recién generado (HelloWorld-activex-ESC.txt.p7s) para comprobar la validez de la firma.

Paquete *com.izenpe.trustedx.cliente.smime*

En este paquete se muestran dos ejemplos de generación y verificación de la firma PKCS7, utilizando para ello el perfil SMIME, soportado por la plataforma TrustedX.

El primer ejemplo que hay que ejecutar es el de *GeneracionFirmaSMIME.java*, ya que, a partir del fichero 'mime.txt' creado en tiempo de ejecución (ubicado en *data/input/*), depositará un fichero firmado llamado 'smime-v2.txt' en *data/output/*. Después, éste será utilizado por el ejemplo de *VerificacionFirmaSMIME.java*, que se encargará de validar este fichero.

Se ha de comentar que el ejemplo *GeneracionFirmaSMIME.java* hace uso del perfil de Firma SMIME y el de *VerificacionFirmaSMIME.java* utiliza el perfil de Verificación SMIME.

Paquete *com.izenpe.trustedx.cliente.xades*

En este paquete se muestran los ejemplos sobre la generación de una firma XML Enveloping, XML Enveloped, XML con Sello de Tiempo y XML con firmas múltiples. Además, se puede encontrar un ejemplo de verificación de la firma XML y ejemplo de actualización de la firma XML. Para ello, se usan los perfiles de XAdES (para firma y verificación) y de No Repudio (para la actualización).

Los primeros ejemplos que hay que ejecutar, independientemente del orden de ejecución de éstos, son *GeneracionFirmaXMLEnveloping.java*, *GeneracionFirmaXMLEnveloped.java* y *GeneracionFirmaXMLTS.java*. Se generarán tres ficheros que serán utilizados por los demás ejemplos correspondientes a este paquete: 'DemoSignedEnveloping.xml', 'DemoSignedEnveloped.xml' y 'DemoSignedEnvelopingTS.xml' (situados en *data/output/*), respectivamente. En este caso, tal y como se ha comentado, el orden de ejecución no importa. Estos ejemplos se establecen con el perfil de Firma XAdES proporcionado por la plataforma TrustedX.

Después de haber ejecutado los tres ejemplos anteriores el orden de ejecución de los restantes ejemplos da igual.

De esta forma, el ejemplo *GeneracionFirmaMultipleXML.java* hará uso del fichero 'DemoSignedEnveloping.xml' para realizar la operación de firma múltiple. Después, escribirá en disco un fichero llamado 'DemoSignedEnvelopingMulti.xml' con la firma múltiple (en *data/output/*). Al igual que los ejemplos ya mencionados, hace uso del perfil de Firma XAdES.

El ejemplo *VerificacionFirmaXML.java* hace del fichero 'DemoSignedEnveloped.xml' generado por la clase *GeneracionFirmaXMLEnveloped.java*. El fichero de entrada puede cambiarse por otro generado con alguno de los ejemplos anteriores. El perfil usado es de Verificación XAdES.

Por último, indicar que el ejemplo *ActualizacionFirmaXML.java* hace uso del fichero 'DemoSignedEnvelopingTS.xml', el cual ha sido creado previamente por el ejemplo

GeneracionFirmaXMLTS.java. En este caso, el perfil usado ha sido el de No Repudio. Éste, guardará en disco un fichero llamado 'DemoSignedEnvelopingTSArchive.xml'.

Paquete *com.izenpe.trustedx.cliente.util*


En este paquete se encuentran tres clases que son utilizadas constantemente por todos los ejemplos de SmartWrapper. Son las siguientes: *Constantes.java*, *Propiedades.java* y *TxSmartWrapperUtil.java*.


La clase *Constantes.java* está compuesta por dos atributos que sirven para representar una petición con éxito:

 RMAJOR_SUCCESS

 RMINOR_SUCCESS

La clase *Propiedades.java* contiene los atributos utilizados para la configuración de los ejemplos:


 La dirección URL donde están ubicados todos los servicios necesarios para la ejecución de los ejemplos.


 El campo *Asunto* del certificado custodiado por la plataforma para hacer la firma en el servidor (aplicable a los ejemplos de generación de firma).

Por último, está la clase *TxSmartWrapperUtil.java* que contiene los métodos necesarios para leer de disco y escribir a disco, así como para obtener por consola la petición y la respuesta en cadena de texto.

Carpeta *data*

En este directorio se pueden encontrar dos carpetas en las que se encuentran todos los ficheros que serán utilizados en el proyecto.

 Carpeta *input*: En esta carpeta se encuentran todos los ficheros que serán utilizados en los diversos servicios.

 Carpeta *output*: En esta carpeta aparecerán aquellos ficheros generados por los diferentes ejemplos.

Carpeta *resources*

En esta carpeta se encuentran dos ficheros diferentes. Por un lado, está el fichero de configuración del almacén de certificados (que se entrega con los ejemplos) y, por otro lado, el fichero de configuración del almacén de claves, incluido por el usuario.



Carpeta *soap-messages*

En este directorio se pueden encontrar dos carpetas en las que se encuentran todos los ficheros que se generan con los diferentes ejemplos:

- 📁 Carpeta *request*: En esta carpeta se guardan, en ficheros con extensión .txt, las peticiones SOAP generadas por los ejemplos.
- 📁 Carpeta *response*: En esta carpeta se guardan, en ficheros con extensión .txt, las respuestas SOAP generadas por los ejemplos.

8 ANEXO 1. FORMATOS DE FIRMA

Antes que nada, hay que decir que esta plataforma de servicios de firma soporta dos grupos de formatos de firma: el formato de firma electrónica no avanzada (No AdES) y los formatos de firma electrónica avanzada (XAdES- BES, XAdES- T, XAdES- C, XAdES- XL y XAdES- A).

8.1 Formato de firma electrónica no avanzada

8.1.1 No XAdES

Se trata de un formato de firma electrónica no avanzada, por lo tanto, la firma no se realiza con técnicas de criptografía asimétrica.

8.2 Formato de firma electrónica avanzada

La mayoría de los formatos de firma que se pueden utilizar en los ejemplos provienen de la especificación de XAdES, que no es más que un conjunto de extensiones a las recomendaciones XML- DSig. De esta forma, XAdES es muy adecuado para la firma electrónica avanzada.

Así pues, XAdES está basado en perfiles específicos de XML- DSig para ser usados con firma electrónica reconocida, cumpliendo con la directiva de la Unión Europea. Una ventaja importante de utilizar estos formatos es que los documentos firmados electrónicamente pueden seguir teniendo validez durante largos periodos de tiempo, incluyendo el caso en que los algoritmos criptográficos subyacentes se rompan.

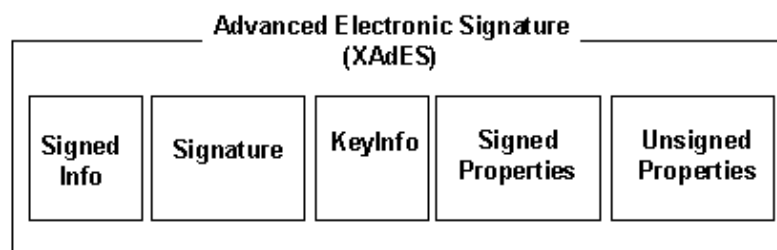
Los tipos de firma XAdES que se pueden utilizar son los siguientes, según el nivel de seguridad que se quiera ofrecer. Cada tipo de firma incluye y extiende al anterior:

8.2.1 XAdES- BES

Formato básico de firma electrónica avanzada que únicamente cumple con los requisitos legales de la Directiva para firma electrónica avanzada. No posee *timestamp*, por lo que la firma electrónica no está protegida contra el problema de No Repudio.

Por lo tanto, este formato de firma electrónica incluye el resultado de la operación de hash y la clave privada, identificando los algoritmos utilizados y el certificado asociado a la clave privada del firmante. A su vez puede ser 'attached' o 'detached', 'enveloped' y 'enveloping'.

En la siguiente figura puede observarse la estructura inicial de una firma electrónica avanzada XAdES.



8.2.2 XAdES- T

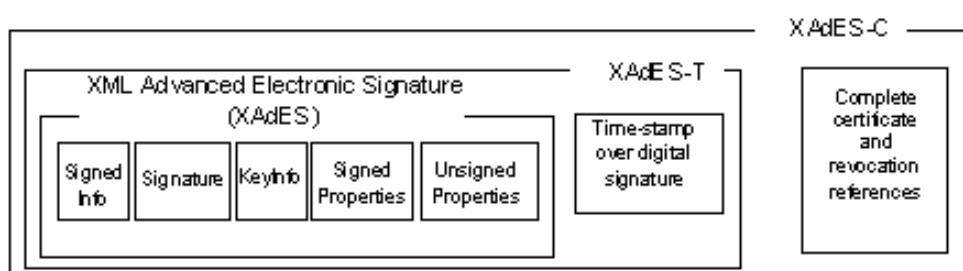
Formato de firma electrónica avanzada en el que se ha añadido un campo de sellado de tiempo para proteger contra el No Repudio. De esta forma, se consiguen [validaciones de firmas para largos periodos de tiempo](#).

8.2.3 XAdES- C

En este formato, se han añadido referencias de los diferentes certificados y listas de revocación a los documentos firmados para permitir, en un futuro, la verificación y validación off- line. Sin embargo, no almacena los datos en sí mismo.

Básicamente, a una firma XAdES- T se le incorpora información sobre los certificados y sus correspondientes datos de revocación (referencias).

En la figura de abajo se puede ver una comparación entre estos tres formatos de firma electrónica avanzada:

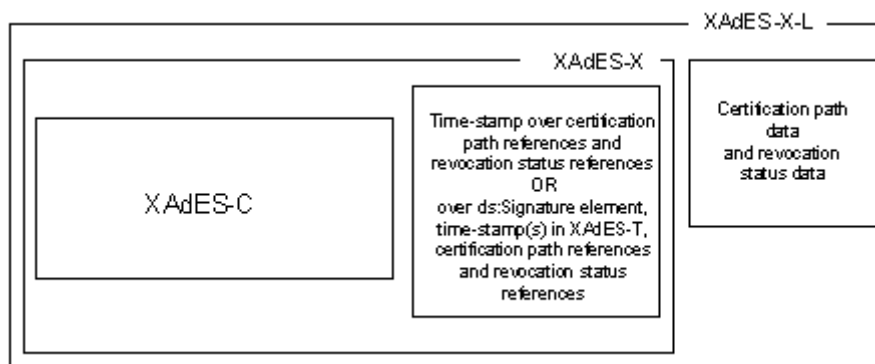


8.2.4 XAdES- XL

Formato de firma electrónica avanzada en el que se ha añadido un sello de tiempo sobre una firma XAdES- C y XAdES- X. Además, se adjuntan los certificados y los valores de revocación necesarios para realizar la validación de la firma.

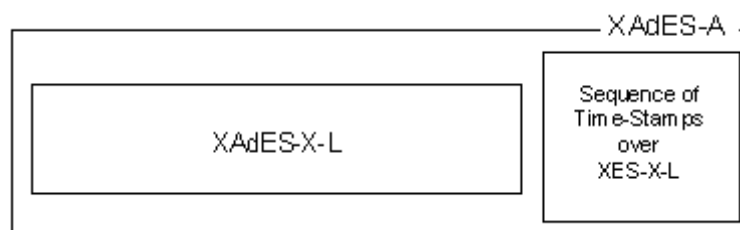
De esta forma, se permite la verificación de las firmas en el futuro incluso si las fuentes originales no estuvieran ya disponibles.

En la figura se puede ver la relación de inclusión entre los formatos XAdES- C y XAdES- XL:



8.2.5 [XAdES- A](#)

En este formato se añade la opción del *timestamp* periódico de documentos archivados a fin de prevenir que puedan ser comprometidos debido a la debilidad de la firma durante un periodo largo de tiempo.



En el siguiente formato de firma electrónica que se muestra a continuación, no se sigue la relación de inclusión de los otros cinco formatos.

8.2.6 [XAdES- EPES](#)

Este formato de firma electrónica avanzada con política explícita está construido sobre el formato XAdES- BES, incorporando propiedades de firma. Estas propiedades indicarán la obligación de validar la firma contra una política de firmas especificada con anterioridad. Por supuesto, este atributo de firma estará protegido mediante la firma del sujeto en cuestión.

9 ANEXO 2. PERFILES SOPORTADOS

Este apartado hace referencia al uso de los distintos servicios de firma digital (TWS- DS, TWS- DR y TWS- DSV) a través de perfiles adaptados a los diferentes escenarios de TrustedX. Los perfiles que se permiten son los que se presentan a continuación:

9.1 Firma CMS/ PKCS#7

Establece el formato de los mensajes de petición/ respuesta para los servicios de generación (TWS-DS) y verificación (TWS-DSV) de firmas digitales en formato CMS o PKCS #7.

Este perfil soporta los siguientes tipos de firma:

- ✓ Firmas múltiples (en serie y paralelo).
- ✓ Firmas embedded (attached) o separadas (detached). (Ver punto 10)
- ✓ Firmas con un sello de tiempo emitido por una autoridad externa.

9.2 Firma S/ MIME

Establece el formato de los mensajes de petición/ respuesta para los servicios de generación (TWS-DS) y verificación (TWS-DSV) de firmas digitales en formato S/MIME.

Este perfil soporta los siguientes tipos de firma:

- ✓ Firmas múltiples (en serie y paralelo).
- ✓ Firmas embedded (attached) o separadas (detached).
- ✓ Firmas con un sello de tiempo emitido por una autoridad externa.

9.3 Firma PDF

Establece el formato de los mensajes de petición/ respuesta para los servicios de generación (TWS-DS) y verificación (TWS-DSV) de firmas digitales en formato PDF.

El perfil de firma PDF de TrustedX permite generar y verificar firmas digitales interoperables con el mecanismo nativo incluido en la versión 6 de Adobe Acrobat

Soporta los siguientes tipos de firmas:

- ✓ Formatos definidos por Adobe.
 - Firma de tipo detached.
 - Firma con SHA1.

✓ Formatos avanzados:

- XAdES- BES.
- XAdES- T: Firmas con sello de tiempo. El sello de tiempo sólo puede ser verificado por aplicaciones Adobe Acrobat a partir de la versión 7 (aunque los atributos de la firma básica sean compatibles con la versión 6).
- XAdES- A: Firmas de archivo. El sello de archivo es soportado por las aplicaciones Adobe Acrobat (aunque los atributos de la firma básica puedan ser verificados por dichas aplicaciones).

Por último, la plataforma de TrustedX permite realizar y verificar firmas visibles e invisibles.

9.4 Firma XML

Establece el formato de los mensajes de petición/ respuesta para los servicios de generación (TWS-DS) y verificación (TWS-DSV) de firmas digitales en formato XML- DSig.

Este perfil de firma soporta los formatos de firma digital avanzada definidos en XAdES.

9.5 Estado del Certificado

Establece el formato de los mensajes de petición/ respuesta para los servicios de validación (TWS-DV) de certificados X.509 y cadenas de certificados PKCS #7.

Este perfil soporta certificados simples y contenedores de certificados en formato PKCS #7/ CMS.

9.6 Sello de Tiempo

Establece el formato de los mensajes de petición/ respuesta para solicitar la generación (TWS-DS) y verificación (TWS-DSV) de sellos de tiempo en formato RFC3161 (ver punto 10).

En concreto, el sello de tiempo es una forma de firma digital con las siguientes propiedades:

- ✓ Protege la integridad del documento.
- ✓ Evidencia la existencia de unos datos en un momento determinado del tiempo.

9.7 No Repudio

Establece el formato de los mensajes de petición/ respuesta para la generación (TWS-DR) y verificación (TWS-DSV) de evidencias de no repudio. Las evidencias se generan sobre la firma digital del documento de entrada:

- ✓ Para firmas digitales de tipo XML- DSig, las evidencias que se generan son de tipo XAdES-T (evidencia XML de firma) y XAdES-A (evidencia XML de firma longeva).
- ✓ Para firmas digitales de tipo CMS, S/ MIME y PDF, las evidencias que se generan son de tipo ES-T (evidencia CMS de firma) y ES-A (evidencia CMS de firma longeva).

El perfil de no-repudio limita la flexibilidad para verificar y actualizar firmas con elementos de confianza. Tal como se indica en XAdES, se trata de un proceso en cadena:

1. Una primera firma (ejemplo: una firma generada mediante los perfiles CMS/PKCS #7 o XML de TWS-DS) se completa mediante información de terceras partes de confianza (información de revocación y sellos de tiempo).
2. Se genera una firma avanzada longeva CMS o XAdES.

Hay que tener en cuenta que si la entrada de este perfil ya es una firma longeva, se obtiene de nuevo información de las terceras partes de confianza para crear una firma longeva (más reciente y duradera) sobre la anterior.

Además, el servicio definido en este perfil establece explícitamente que las firmas de entrada deben ser válidas; de otro modo, no se actualizará la firma.

10 GLOSARIO

En este apartado se explicarán aquellos conceptos que pueden no haber quedado demasiado claros a lo largo de este documento.

10.1 XML- DSig

El XML-DSig es una recomendación del W3C que define un tipo de sintaxis XML destinada a poder utilizar la firma digital en documentos XML.

Es el formato de mayor expansión actualmente, y el más comúnmente usado en aplicaciones online.

10.2 Tipos de firma

A la hora de realizar una firma electrónica hay que tener en cuenta el formato de firma que se utiliza. De esta forma, tendrá un significado u otro:

- ✓ Formato CMS/ PKCS#7: Puede representarse de dos formas distintas.
 - *Firma attached*: La firma electrónica y el documento forman un fichero único.
 - *Firma detached*: La firma electrónica y el documento son documentos separados.
- ✓ Formato XML: Según la posición de la firma electrónica en el documento XML, se distinguen tres formas para firmar un documento XML.
 - *Firma enveloped*: La firma se añade al final del documento XML como un elemento más. Se firma todo lo inmediatamente anterior al documento.
 - *Firma enveloping*: El documento se incluye dentro de la firma en la que se referencia lo firmado como objeto insertado en la firma. Ya que se referencian los objetos, este modelo permitiría distinguir lo que se firma, pudiendo firmar el objeto entero o partes de él (asignando un identificador unívoco).
 - *Firma detached*: La firma y el documento se separan en dos archivos. Una referencia a donde se encuentra el documento puede aparecer en la propia firma.



10.3 Formato RFC3161

En este documento se describe el formato de una solicitud enviada a una TSA (Autoridad de Sellado de Tiempo) y de la respuesta que se devuelve. También, se establecen varios requisitos de seguridad pertinentes para las operaciones de la TSA, en lo que respecta a la tramitación de solicitudes para generar respuestas.