package es.mityc.firmaJava.libreria.xades ;

import java.io.ByteArrayInputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.io.StringWriter;

import java.io.UnsupportedEncodingException;

import java.io.Writer;

import java.net.URI;

```
import java.net.URISyntaxException;
import java.net.URLEncoder;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.Provider;
import java.security.Security;
import java.security.cert.CertPath;
import java.security.cert.CertStore;
import java.security.cert.Certificate;
import java.security.cert.CertificateEncodingException;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Map.Entry;
import java.util.Properties;

import javax.security.auth.x500.X500Principal;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
```

```
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.bouncycastle.asn1.ocsp.ResponderID;
import org.bouncycastle.cms.CMSException;
import org.bouncycastle.cms.CMSSignedData;
import org.bouncycastle.ocsp.BasicOCSPResp;
import org.bouncycastle.ocsp.OCSPException;
import org.bouncycastle.ocsp.OCSPResp;
import org.bouncycastle.tsp.TimeStampResponse;
import org.bouncycastle.tsp.TimeStampToken;
import org.w3c.dom.Attr;
import org.w3c.dom.DOMException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import adsi.org.apache.xml.security.Init;
import adsi.org.apache.xml.security.algorithms.JCEMapper;
import adsi.org.apache.xml.security.signature.ObjectContainer;
import adsi.org.apache.xml.security.signature.XMLSignature;
import adsi.org.apache.xml.security.transforms.Transforms;
import adsi.org.apache.xml.security.utils.Constants;
import adsi.org.apache.xml.security.utils.IgnoreAllErrorHandler;
```

```
import adsi.org.apache.xml.security.utils.resolver.ResourceResolverSpi;


import es.mityc.firmaJava.libreria.ConstantesXADES;
import es.mityc.firmaJava.libreria.errores.ClienteChainNotFoundError;
import es.mityc.firmaJava.libreria.errores.ClienteError;
import es.mityc.firmaJava.libreria.excepciones.AddXadesException;
import es.mityc.firmaJava.libreria.utilidades.Base64;
import es.mityc.firmaJava.libreria.utilidades.Base64Coder;
import es.mityc.firmaJava.libreria.utilidades.I18n;
import es.mityc.firmaJava.libreria.utilidades.NombreNodo;
import es.mityc.firmaJava.libreria.utilidades.UtilidadCertificados;
import es.mityc.firmaJava.libreria.utilidades.UtilidadFechas;
import es.mityc.firmaJava.libreria.utilidades.UtilidadFicheros;
import es.mityc.firmaJava.libreria.utilidades.UtilidadFirmaElectronica;
import es.mityc.firmaJava.libreria.utilidades.UtilidadTratarNodo;
import es.mityc.firmaJava.libreria.xades.DataToSign.XADES_X_TYPES;
import es.mityc.firmaJava.libreria.xades.elementos.xades.CRLRef;
import es.mityc.firmaJava.libreria.xades.elementos.xades.CRLRefs;
import es.mityc.firmaJava.libreria.xades.elementos.xades.CRLValues;
import es.mityc.firmaJava.libreria.xades.elementos.xades.CertificateValues;
import es.mityc.firmaJava.libreria.xades.elementos.xades.DataObjectFormat;
import es.mityc.firmaJava.libreria.xades.elementos.xades.EncapsulatedX509Certificate;
import es.mityc.firmaJava.libreria.xades.elementos.xades.ObjectIdentifier;
import es.mityc.firmaJava.libreria.xades.elementos.xades.SignatureProductionPlace;
import es.mityc.firmaJava.libreria.xades.elementos.xades.SigningTime;
import es.mityc.firmaJava.libreria.xades.errores.BadFormedSignatureException;
import es.mityc.firmaJava.libreria.xades.errores.FirmaXMLError;
import es.mityc.firmaJava.libreria.xades.errores.InvalidInfoNodeException;
import es.mityc.firmaJava.libreria.xades.errores.PolicyException;
import es.mityc.firmaJava.role.IClaimedRole;
import es.mityc.javasign.ConstantsXAdES;
```

```
import es.mityc.javasign.EnumFormatoFirma;
import es.mityc.javasign.asn1.ASN1Utils;
import es.mityc.javasign.certificate.CertStatusException;
import es.mityc.javasign.certificate.ICertStatus;
import es.mityc.javasign.certificate.IOCSPCertStatus;
import es.mityc.javasign.certificate.IX509CRLCertStatus;
import es.mityc.javasign.certificate.ocsp.OCSPLiveConsultant;
import es.mityc.javasign.exception.SignMITyCException;
import es.mityc.javasign.i18n.I18nFactory;
import es.mityc.javasign.i18n.II18nManager;
import es.mityc.javasign.pkstore.IPKStoreManager;
import es.mityc.javasign.trust.TrustAbstract;
import es.mityc.javasign.trust.TrustFactory;
import es.mityc.javasign.ts.HTTPTimeStampGenerator;
import es.mityc.javasign.ts.TSPAlgoritmos;
import es.mityc.javasign.tsa.ITimeStampGenerator;
import es.mityc.javasign.tsa.TimeStampException;
import es.mityc.javasign.xml.refs.AbstractObjectToSign;
import es.mityc.javasign.xml.refs.InternObjectToSign;
import es.mityc.javasign.xml.refs.ObjectToSign;
import es.mityc.javasign.xml.refs.SignObjectToSign;
import es.mityc.javasign.xml.resolvers.IPrivateData;
import es.mityc.javasign.xml.resolvers.IResourceData;
import es.mityc.javasign.xml.resolvers.MITyCResourceResolver;
import es.mityc.javasign.xml.resolvers.ResolverPrivateData;
import es.mityc.javasign.xml.resolvers.XAdESResourceResolverSpi;
import es.mityc.javasign.xml.transform.Transform;
import es.mityc.javasign.xml.transform.TransformEnveloped;
import es.mityc.javasign.xml.xades.IStoreElements;
import es.mityc.javasign.xml.xades.LocalFileStoreElements;
import es.mityc.javasign.xml.xades.policy.IFirmaPolicy;
```

```
import es.mityc.javasign.xml.xades.policy.PoliciesManager;


/**
 * <p>Clase principal para la firma de documentos XML.</p>
 *
 */
public class FirmaXML {


   private static Log log = LogFactory.getLog(FirmaXML.class);
   private static II18nManager i18n =
I18nFactory.getI18nManager(ConstantsXAdES.LIB_NAME);


   /** <p>Validación de elementos de validación. Un valor <code>true</code>
    * incluye OCSPs de la TSA, OCSPResponder, etc... en firmas >= XAdES-C. Con valor
    * <code>false</code> sólo se valida el certificado firmante.</p> */
   private final static boolean ADD_VALIDATION_OCSP = true;


   String          profileDirectory                =
ConstantesXADES.CADENA_VACIA;
   private String          xadesNS                 =
ConstantsXAdES.DEFAULT_NS_XADES;
   private String          xadesSchema             = null;
   private String          xmldsigNS               =
ConstantsXAdES.DEFAULT_NS_XMLSIG;


   // Almacena las id´s para el esquema 1.1.1
   private ArrayList<String> idNodoSelloTiempo = new ArrayList<String>();
   private String idNodoCertificateRefs = null;
   private String idNodoRevocationRefs = null;
   private String idSigProperties = null;
   private String idSignatureValue = null;
   /** Listado de resolvers aplicados. */
```

```
  private ArrayList<ResourceResolverSpi> resolvers;


  /**
   * Crea una nueva instancia de FirmaXML
   */
  public FirmaXML() {
  }


  /**
   * <p>Establece el namespace que se aplicará a los nodos de XML Signature.</p>
   * @param namespace Namespace aplicado a XMLSig
   */
  public void setDefaultNSXmlSig(String namespace) {
          this.xmldsigNS = namespace;
  }


          /**
           * <p>Establece el Locale del sistema antiguo de internacionalización.</p>
           * @param locale Localización a aplicar
           */
          public void setLocale(String locale) {
                  I18n.setLocale(locale, locale.toUpperCase());
          }



          /**
           * Añade una instancia encargada de resolver los accesos a elementos firmados
en la firma cuyo contenido es privado.
           *
           * @param resolver objeto que implementa la interfaz IPrivateDate para el acceso
a elementos privados
           */
```

```
	public void addResolver(IPrivateData resolver) {
		addResolver(new ResolverPrivateData(resolver));
	}

	/**
	 * Añade una instancia encargada de resolver accesos a información.
	 *
	 * @param resolver resolver
	 */
	public void addResolver(MITyCResourceResolver resolver) {
		if (resolvers == null) {
			resolvers = new ArrayList<ResourceResolverSpi>();
		}
		resolvers.add(resolver);
	}

	/**
	 * Añade una instancia encargada de resolver los accesos a elementos firmados en la firma que requieran un acceso especial.
	 *
	 * @param resolver objeto que implementa la interfaz IResourceData para el acceso a elementos
	 */
	public void addResolver(IResourceData resolver) {
		addResolver(new XAdESResourceResolverSpi(resolver));
	}

	/**
	 *
	 * @param firmaCertificado
	 * @param xml
```

```
         * @param storeManager
         * @param salida
         * @return Retorna el id del
         * @throws Exception
         */
  public String sign2Stream(
      X509Certificate firmaCertificado,
      DataToSign xml,
      IPKStoreManager storeManager,
      OutputStream salida) throws Exception{
        PrivateKey pk = storeManager.getPrivateKey(firmaCertificado);
                return signFile(firmaCertificado, xml, pk, salida,
storeManager.getProvider(firmaCertificado));
  }


  /**
   * Firma un fichero XML
   * @param pk Clave privada del certificado firmante
   * @param firmaCertificado Certificado firmante
   * @param xml Fichero XML a firmar
   * @param directorioPerfil Directorio de configuracion de Firefox
   * @throws java.lang.Exception En caso de error
   * @return Array de bytes con el XML firmado
   */
  public boolean signFile(X509Certificate firmaCertificado,
      DataToSign xml,
      IPKStoreManager storeManager,
      String destino,
      String nombreArchivo) throws Exception{
        PrivateKey pk = storeManager.getPrivateKey(firmaCertificado);
```

```
        return signFile(firmaCertificado, xml, pk, destino, nombreArchivo,
storeManager.getProvider(firmaCertificado));
  }


        private String signFile(X509Certificate certificadoFirma, DataToSign xml,
      PrivateKey pk, OutputStream salida, Provider provider) throws Exception {


        Object[] res = signFile(certificadoFirma, xml, pk, provider);


        if (res[1] != null)
        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4
3));


    try
    {
                UtilidadFicheros.writeXML((Document)res[0], salida);
        return (String) res[2];
    }
    catch (Throwable t)
    {
        if (t.getMessage() != null &&
t.getMessage().startsWith(ConstantesXADES.JAVA_HEAP_SPACE))
                throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3));
        else
                throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4));
    }
  }


        private boolean signFile(X509Certificate certificadoFirma, DataToSign xml,
```

```
                PrivateKey pk, String destino, String nombreArchivo, Provider provider)
throws Exception {
                if (destino == null || nombreArchivo == null) {
                        // No se proporcionaron los datos de firma
                        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_31));
                }

        Object[] res = signFile(certificadoFirma, xml, pk, provider);

//      // Si se firma XADES-C exclusivamente, se guardan las respuestaOCSP y los
certificados
//      // con un nombre asociado al fichero de firma y en la misma ruta temporal
        Document doc = (Document) res[0];

    // Se guarda la firma en su destino
    File fichero = new File(destino, nombreArchivo);
    FileOutputStream f = new FileOutputStream(fichero);

    try
    {
                Writer out = new OutputStreamWriter(f, ConstantesXADES.UTF8);

                Transformer xformer =
TransformerFactory.newInstance().newTransformer();
                Properties props = new Properties();
                props.setProperty(OutputKeys.METHOD, "XML");
                props.setProperty(OutputKeys.ENCODING, ConstantesXADES.UTF8);
                props.setProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
                xformer.setOutputProperties(props);

                StringWriter salida = new StringWriter();
```

```
                xformer.transform(new DOMSource(doc), new StreamResult(salida));
        out.write(salida.toString());
        out.flush();
                out.close();
    }
    catch (Throwable t) {
        if (t.getMessage() != null &&
t.getMessage().startsWith(ConstantesXADES.JAVA_HEAP_SPACE))
                throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3));
        else
                throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4));
    } finally {
        f.flush();
        f.close();
    }

    return true ;
  }

        public Object[] signFile(X509Certificate certificadoFirma, DataToSign dataToSign,
      PrivateKey pk, Provider provider) throws Exception {

                ArrayList<RespYCerts> respuestas = new ArrayList<RespYCerts>();
    ArrayList<X509Certificate> certificadosConOCSP = new ArrayList<X509Certificate>();

    Init.init();

    Document doc = dataToSign.getDocument();
    if (doc == null) {
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

```
        dbf.setNamespaceAware(true);
        DocumentBuilder db = dbf.newDocumentBuilder();
        db.setErrorHandler(new IgnoreAllErrorHandler());
                try {
                    InputStream is = dataToSign.getInputStream();
                    if (is != null) {
                            InputSource isour = new InputSource(is);
                            String encoding = dataToSign.getXMLEncoding();
                            isour.setEncoding(encoding);
                            doc = db.parse(isour);
                    } else {
                              doc = db.newDocument();
                    }
                } catch (IOException ex) {
                    throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_50),
ex);
                }
    }

    // Se toman las variables de configuración de DataToSign
    XAdESSchemas esquemaTemp = dataToSign.getEsquema();
    if (esquemaTemp != null) {
        xadesSchema = esquemaTemp.getSchemaUri();
//configuracion.getValor(ConstantesXADES.XADES_SCHEMA);
    } else {
        xadesSchema = XAdESSchemas.XAdES_132.getSchemaUri();
    }

    String algDigestXML = (dataToSign.getAlgDigestXmlDSig() != null) ?
dataToSign.getAlgDigestXmlDSig() : UtilidadFirmaElectronica.DIGEST_ALG_SHA1;
```

```
    // Consulta si puede resolver el algoritmo antes de continuar. Si no puede, lanza
excepción
    if (JCEMapper.translateURItoJCEID(algDigestXML) == null) {
        throw new
SignMITyCException(i18n.getLocalMessage(ConstantsXAdES.I18N_SIGN_1,
algDigestXML));
    }

    XMLSignature.setDefaultPrefix(Constants.SignatureSpecNS, xmldsigNS);
    XMLSignature firma = new XMLSignature(doc,
                dataToSign.getBaseURI(),
                XMLSignature.ALGO_ID_SIGNATURE_RSA_SHA1);
    firma.setId(UtilidadTratarNodo.newID(doc,
ConstantesXADES.SIGNATURE_NODE_ID));
    firma.getSignedInfo().setId(UtilidadTratarNodo.newID(doc,
ConstantesXADES.SIGNED_INFO_NODE_ID));
                if (resolvers != null) {
                    Iterator<ResourceResolverSpi> it = resolvers.iterator();
                    while (it.hasNext()) {
                        firma.addResourceResolver(it.next());
                    }
                }

    firma.setXPathNamespaceContext(xmldsigNS, ConstantesXADES.SCHEMA_DSIG);
    EnumFormatoFirma tipoFirma = dataToSign.getXadesFormat();
    boolean xadesActivo = (tipoFirma.compareTo(EnumFormatoFirma.XAdES_BES)>=0);

    if(xadesActivo){
      firma.setXPathNamespaceContext(xadesNS, xadesSchema);
    }

    Element elementoPrincipal = null;
    // TODO: permitir utilizar cadenas XPATH para indicar el nodo que contendrá la firma
```

```
    if (!dataToSign.isEnveloped()) {
        doc.appendChild(firma.getElement());
    } else {
        String nodoRaizXml = dataToSign.getParentSignNode();
            if (nodoRaizXml == null) { // Si no se indicó, se toma el primero del documento
                elementoPrincipal = doc.getDocumentElement();//(Element)
doc.getFirstChild();
            } else { // Sí se indicó
                NodeList nodos = doc.getElementsByTagName(nodoRaizXml);
              if(nodos.getLength() != 0) // Si se encuentra el/los nodos se toma el primero
                elementoPrincipal = (Element)nodos.item(0);
              else { // Si no se encuentra el nodo, se realiza la busqueda con un
identificador en lugar de TagName
                Node nodo = UtilidadTratarNodo.getElementById(doc, nodoRaizXml);
                    if(nodo != null) // Si se encuentra el nodo, se toma como nodo padre
de la firma
                      elementoPrincipal = (Element)nodo;
                    else
                      throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_2)) ;
              }
            }
            elementoPrincipal.appendChild(firma.getElement());
    }

    idSigProperties = UtilidadTratarNodo.newID(doc,
ConstantesXADES.GUION_SIGNED_PROPERTIES);
    if(xadesActivo) {

        String tipoEsquema =
UtilidadFirmaElectronica.obtenerTipoReference(xadesSchema);

      firma.addDocument(ConstantesXADES.ALMOHADILLA +  firma.getId()
```

```
                + idSigProperties, null, algDigestXML,
                UtilidadTratarNodo.newID(doc,
ConstantesXADES.SIGNED_PROPERTIES_ID), tipoEsquema);
    }

    firma.addKeyInfo(certificadoFirma);
    firma.addKeyInfo(certificadoFirma.getPublicKey()) ;
    String idCert = UtilidadTratarNodo.newID(doc, ConstantesXADES.CERTIFICATE1);
    firma.getKeyInfo().setId(idCert);

    // TODO: cambiar la manera de indicar los elementos que se firman a una manera más
estándar a través de objetos externos,
    // objetos internos en firma y objetos internos fuera de firma.
    // Añadimos los elementos propios de la firma en XADES

    // Firma el certificado de firma
    firma.addDocument(ConstantesXADES.ALMOHADILLA + idCert, null, algDigestXML,
null, null);

    // Incluye los objetos que se quiere firmar
    ArrayList<ObjectToSign> objects = dataToSign.getObjects();
    if (objects != null) {
        Iterator<ObjectToSign> it = objects.iterator();
        while (it.hasNext()) {
                ObjectToSign obj = it.next();
                AbstractObjectToSign objToSign = obj.getObjectToSign();

                        String refId = UtilidadTratarNodo.newID(doc, "Reference-ID-");
                        List<ObjectContainer> containers = objToSign.getObjects(doc);
                        if (containers != null) {
                                for (ObjectContainer objectContainer : containers) {
                                        firma.appendObject(objectContainer);
```

```
                                        }
                        }

        String objId = objToSign.getReferenceURI();
        // Construye las transformadas que se aplicarán al objeto
        Transforms trans = null;
        List<Transform> list = objToSign.getTransforms();
        // Si la firma está dentro de lo firmado indica que la transformada debe ser de
enveloped
        if (dataToSign.isEnveloped()) {
          if (objId != null) {
            Element aFirmar = UtilidadTratarNodo.getElementById(doc, objId);
            if (UtilidadTratarNodo.isChildNode(firma.getElement(), aFirmar)) {
              list.add(new TransformEnveloped());
            }
          }
        }
        if (list.size() > 0) {
          trans = new Transforms(doc);
          for (Transform transform : list) {
            trans.addTransform(transform.getAlgorithm(), transform.getExtraData(doc));
          }
        }

        MITyCResourceResolver resolver = objToSign.getResolver();
                        if (resolver != null) {
                                firma.addResourceResolver(resolver);
                        }

                        String typeInfo = objToSign.getType();
                        firma.addDocument(objId, trans, algDigestXML, refId, typeInfo);
```

```
                        obj.setId(ConstantesXADES.ALMOHADILLA + refId);
            }
        }

        XAdESSchemas schema = null;
        if(xadesActivo){

            schema = XAdESSchemas.getXAdESSchema(xadesSchema);
            if (schema == null) {

            log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_44)
                                    + ConstantesXADES.ESPACIO + xadesSchema);
                    throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_45));
            }
            addXades(doc,
                firma.getId(),
                certificadoFirma,
                firma.getElement(),
                schema,
                dataToSign,
                algDigestXML);
            if (dataToSign.hasPolicy())  {
              addXadesEPES(firma.getElement(), dataToSign.getPolicyKey());
            } else if (XAdESSchemas.XAdES_111.equals(schema)) {
              // Se escribe una política implícita
              addXadesEPES(firma.getElement(),
ConstantesXADES.LIBRERIAXADES_IMPLIEDPOLICY_MANAGER);
            }
        }
```

```
    try
    {
      if(provider!=null && Security.getProvider(provider.getName()) == null)
      {
        JCEMapper.setProviderSignatureThread(provider);
      }
      firma.sign(pk);
    }
    catch(Exception ex)
    {
      log.error(I18n.getResource("libreriaxades.firmaxml.error4"), ex);
      throw ex;
    }
    finally
    {
      JCEMapper.removeProviderSignatureThread();
    }



    // Añadimos el Id al nodo signature value


        Element elementoValorFirma = null ;


    NodeList nodoValorFirma =
firma.getElement().getElementsByTagNameNS(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE_VALUE);
    if(nodoValorFirma.getLength() != 0)
      elementoValorFirma = (Element)nodoValorFirma.item(0);
    else
      throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_5));
    // Le añadimos el elemento ID
```

```
    Attr idValorFirma = doc.createAttributeNS(null, ConstantesXADES.ID);
    idSignatureValue = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SIGNATURE_VALUE);
    idValorFirma.setValue(idSignatureValue);
    NamedNodeMap elementoIdAtributosValorFirma =
        elementoValorFirma.getAttributes();
    elementoIdAtributosValorFirma.setNamedItem(idValorFirma);



    //Comprobamos si se debe de firmar añadiendo el elemento de XADES-T


    boolean xadesT =(tipoFirma.compareTo(EnumFormatoFirma.XAdES_T)>=0);



log.debug(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_DEBUG_1) +
xadesT);


    byte[] selloTiempo = null;
    if(xadesT) {
      try {
        // Añadimos XADES-T


        ITimeStampGenerator timeStampGenerator =
dataToSign.getTimeStampGenerator();
        if (timeStampGenerator == null) {
                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_6)
);
                } else {
                // Obtenemos la respuesta del servidor TSA
                // Se añaden los elementos propios de la firma XADES-T
                byte[] byteSignature =
UtilidadTratarNodo.obtenerByteNodo(firma.getElement(),
```

```
ConstantesXADES.SCHEMA_DSIG, ConstantesXADES.SIGNATURE_VALUE,
CanonicalizationEnum.C14N_OMIT_COMMENTS, 5);

                    selloTiempo = timeStampGenerator.generateTimeStamp(byteSignature);

                    addXadesT(firma.getElement(), firma.getId(), selloTiempo);

          }

        } catch (AddXadesException e) {

           throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_7)
+ e.getMessage()) ;

        }

      }


      // Comprobamos si se debe de firmar añadiendo los elementos de XADES-C

      boolean xadesC = (tipoFirma.compareTo(EnumFormatoFirma.XAdES_C)>=0);


log.debug(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_DEBUG_2) +
xadesC);


      if(xadesC){

          try {

                    // Comprobamos si se ha realizado antes la firma XADES-T. En caso
contrario se le avisa

                    // al usuario que no puede realizarse la firma XADES-C

                    if(xadesT){

                              if (dataToSign.getCertStatusManager() == null) {

                                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_5
3));

                              }

                              // Añadimos XADES-C


          log.info(i18n.getLocalMessage(ConstantsXAdES.I18N_VALIDATE_18));

                              convertICertStatus2RespYCerts(
```

```
dataToSign.getCertStatusManager().getCertChainStatus(certificadoFirma),
certificadosConOCSP, respuestas);


                    if (ADD_VALIDATION_OCSP) {
                        if (log.isDebugEnabled()) {
                            log.debug("Se incluyen las referencias OCSP de la
propia VA");
                        }

                        X509Certificate ocspCert = null;
                        try {
                            // Se extrrae el certificado emisor de la respuesta
OCSP
                            IOCSPCertStatus respOcsp = (IOCSPCertStatus)
respuestas.get(0).getCertstatus();
                            OCSPResp resp = new
OCSPResp(respOcsp.getEncoded());
                            BasicOCSPResp respuestaBasica =
(BasicOCSPResp)resp.getResponseObject();
                            if (log.isDebugEnabled()) {
                                ResponderID respID =
respuestaBasica.getResponderId().toASN1Object();
                                log.debug("Extracción del certificado OCSP:
" + ASN1Utils.getResponderID(respID).toString());
                            }

                            X509Certificate[] ocspCerts =
respuestaBasica.getCerts(ConstantesXADES.SUN);
                            if (ocspCerts != null && ocspCerts.length > 0) {
                                if (log.isDebugEnabled()) {
                                    log.debug("Se regenera la cadena y
se lanza la validación");
                                }
```

```
                                        CertPath cpOcsp =
UtilidadCertificados.orderCertPath(Arrays.asList(ocspCerts));
                                        ocspCert =
(X509Certificate)cpOcsp.getCertificates().get(0);
                                        //ArrayList<RespYCerts> respuestasOCSP =
new ArrayList<RespYCerts>();
                                        if
(!ocspCert.getIssuerX500Principal().equals(certificadoFirma.getIssuerX500Principal())) {
                                            List<ICertStatus> re =
dataToSign.getCertStatusManager().getCertChainStatus(ocspCert);
                                            convertICertStatus2RespYCerts(re,
certificadosConOCSP, respuestas);
                                        } else {
                                            ICertStatus respOCSP =
dataToSign.getCertStatusManager().getCertStatus(ocspCert);
                                            ArrayList<ICertStatus> re = new
ArrayList<ICertStatus>(1);
                                            re.add(respOCSP);
                                            convertICertStatus2RespYCerts(re,
certificadosConOCSP, respuestas);
                                        }
                                        //respuestas.addAll(respuestasOCSP);
                                    } else {
                                        log.error("No se pudo recuperar el
certificado de la VA");
                                    }
                                } catch (Exception e1) {
                                    log.error(e1);
                                }

                                if (log.isDebugEnabled()) {
                                    log.debug("Se incluyen las referencias OCSP del
sello de tiempo");
                                }
                                TimeStampToken tst = null;
```

```
                try {
                        tst = new TimeStampToken(new
CMSSignedData(selloTiempo));
                } catch (CMSException e) {
                        // Intenta obtenerlo como
org.bouncycastle.tsp.TimeStampResponse
                        try {
                                TimeStampResponse tsr = new
TimeStampResponse(selloTiempo);
                                tst = tsr.getTimeStampToken();
                        } catch (Exception ex) {
                                log.error(ex);
                        }
                } catch (Exception e) {
                        log.error(e);
                }
                X509Certificate certTSA = null;
                try {
                        CertStore cs =
tst.getCertificatesAndCRLs("Collection", null);
                        Collection<? extends Certificate> certs =
cs.getCertificates(null);
                        if (certs != null && certs.size() > 0) {
                                if (log.isDebugEnabled()) {
                                        log.debug("Se regenera la cadena de
certificados firmante del sello de tiempo y se lanza su validación");
                                }
                                // Se regenera el CertPath para asegurar el
orden correcto
                                try {
                                        Iterable<X509Certificate>
iterableCerts = null;
                                        if (certs instanceof Iterable<?>) {
```

```
                                                iterableCerts =
(Iterable<X509Certificate>) certs;
                                        } else {
                                                throw new Exception("El
certificado no es del tipo esperado: " + certs.getClass());
                                        }
                                        CertPath cpTsa =
UtilidadCertificados.orderCertPath(iterableCerts);
                                        certTSA =
(X509Certificate)cpTsa.getCertificates().get(0);
                                } catch (Exception e) {
                                        // si el token no indica el nombre del
firmante, intenta extraerlo por el certificado
                                        Certificate cert =
certs.iterator().next();
                                        if (cert instanceof X509Certificate) {
                                                certTSA = (X509Certificate)
cert;
                                        }
                                }
                        } else {
                                log.error("No se pudo recuperar el
certificado del sello de tiempo");
                        }
                } catch (Exception e) {
                        log.error(e);
                }

                if (certTSA != null) {
                        if (log.isDebugEnabled()) {
                                log.debug("Certificado de TSA obtenido " +
certTSA.getSubjectX500Principal());
                        }
                        ArrayList<RespYCerts> respuestasTSA = new
ArrayList<RespYCerts>();
```

```
                                        // Si los certificados emisores ya han sido validados,
se valida sólo el certificado final
                                        if
(certTSA.getIssuerX500Principal().equals(certificadoFirma.getIssuerX500Principal())
                                                || (ocspCert != null &&
certTSA.getIssuerX500Principal().equals(ocspCert.getIssuerX500Principal())) ) {
                                                ICertStatus respTSA =
dataToSign.getCertStatusManager().getCertStatus(certTSA);
                                                ArrayList<ICertStatus> re = new
ArrayList<ICertStatus>(1);
                                                re.add(respTSA);
                                                convertICertStatus2RespYCerts(re,
certificadosConOCSP, respuestasTSA);
                                        } else {
                                                convertICertStatus2RespYCerts(

        dataToSign.getCertStatusManager().getCertChainStatus(certTSA),
certificadosConOCSP, respuestasTSA);
                                        }


                                        respuestas.addAll(respuestasTSA);


                                        if (log.isDebugEnabled()) {
                                                log.debug("TSA Validada. Se valida la VA del
propio sello");
                                        }
                                        try {
                                                IOCSPCertStatus respOcspTsa =
(IOCSPCertStatus) respuestasTSA.get(0).getCertstatus();
                                                OCSPResp resp = new
OCSPResp(respOcspTsa.getEncoded());
                                                BasicOCSPResp respuestaBasica =
(BasicOCSPResp)resp.getResponseObject();
                                                if (log.isDebugEnabled()) {
                                                        ResponderID respID =
respuestaBasica.getResponderId().toASN1Object();
```

```
                                                log.debug("Extracción del certificado
OCSP para el sello de tiempo: " + ASN1Utils.getResponderID(respID).toString());
                                        }


                                        X509Certificate[] ocspTsaCerts =
respuestaBasica.getCerts(ConstantesXADES.SUN);
                                        if (ocspTsaCerts != null &&
ocspTsaCerts.length > 0) {
                                                //ArrayList<RespYCerts>
respuestasOCSPTSA = new ArrayList<RespYCerts>();
                                                X509Certificate tsaOCSPCert = null;
                                                // Se regenera el CertPath para
asegurar el orden correcto
                                                CertPath cpTsa =
UtilidadCertificados.orderCertPath(Arrays.asList(ocspTsaCerts));
                                                try {
                                                        tsaOCSPCert =
(X509Certificate)cpTsa.getCertificates().get(0);
                                                } catch (Exception e) {
                                                        // si el token no indica el
nombre del firmante, intenta extraerlo por el certificado
                                                        Certificate cert =
cpTsa.getCertificates().get(0);
                                                        if (cert instanceof
X509Certificate) {
                                                                tsaOCSPCert =
(X509Certificate) cert;
                                                        }
                                                }


                                                if (log.isDebugEnabled()) {
                                                        log.debug("Certificado VA del
sello de tiempo obtenido: " + tsaOCSPCert.getSubjectX500Principal());
                                                }
```

```
                                                    // Si los certificados emisores ya han
sido validados, se valida sólo el certificado final
                                                    if
(tsaOCSPCert.getIssuerX500Principal().equals(certificadoFirma.getIssuerX500Principal())
                                                            || (ocspCert != null &&
tsaOCSPCert.getIssuerX500Principal().equals(ocspCert.getIssuerX500Principal()))
                                                            || (certTSA != null &&
tsaOCSPCert.getIssuerX500Principal().equals(certTSA.getIssuerX500Principal())) ) {
                                                            ICertStatus respOCSP =
dataToSign.getCertStatusManager().getCertStatus(tsaOCSPCert);
                                                            ArrayList<ICertStatus> re =
new ArrayList<ICertStatus>(1);
                                                            re.add(respOCSP);

        convertICertStatus2RespYCerts(re, certificadosConOCSP, respuestas);
                                                    } else {

        convertICertStatus2RespYCerts(dataToSign.getCertStatusManager().getCertChain
Status(tsaOCSPCert), certificadosConOCSP, respuestas);
                                                    }



        //respuestas.addAll(respuestasOCSPTSA);
                                                } else {
                                                        log.error("No se pudo recuperar el
certificado de la VA del sello de tiempo");
                                                }
                                            } catch (Exception e1) {
                                                log.error(e1);
                                            }
                                        }
                                    }

                                    // TODO: si alguna de las respuestas de la cadena de certificación
es revocado se deberá parar la firma
```

```
                        addXadesC(firma.getElement(), respuestas, schema,
algDigestXML);


                } else {

                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_2
4)) ;

            }

        } catch (CertStatusException e) {

        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_5
6)

                        + ConstantesXADES.NUEVA_LINEA + e.getMessage()) ;

        } catch (AddXadesException e) {

                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_1
0) + ConstantesXADES.DOS_PUNTOS

                        + ConstantesXADES.NUEVA_LINEA + e.getMessage()) ;

        }

    }


    // Comprobamos si se debe de firmar añadiendo los elementos de XADES-X

    boolean xadesX = (tipoFirma.compareTo(EnumFormatoFirma.XAdES_X)>=0);


log.debug(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_DEBUG_3) +
xadesX);


    boolean xadesXL = (tipoFirma.compareTo(EnumFormatoFirma.XAdES_XL)==0);


log.debug(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_DEBUG_4) +
xadesXL);


    //xadesXL = xadesX;         // Si es XAdES-X, se pone xades-XL a true para redondear

    // Si se firma XADES-C exclusivamente, se guardan las respuestaOCSP y los
certificados
```

```
    // con un nombre asociado al fichero de firma y en la misma ruta temporal
    // TODO: solucionar nombre de los ficheros OCSP
    if (xadesC && !xadesXL) {
        try {
                doc = addURIXadesC(firma.getElement(), saveOCSPFiles(respuestas,
dataToSign.getElementsStorer()), dataToSign.getBaseURI());
        } catch (FirmaXMLError ex) {
                throw new ClienteError("Error al guardar ficheros de estados de
certificados", ex);
        }
    }

    if(xadesX) {
        // Para realizar la firma XADES-XL se deben completar antes los
        // formatos de firmas XADES-T y XADES-C

        if (xadesT && xadesC ) {

                // Se obtiene el nodo raíz de la firma
                Element signatureElement = firma.getElement();
                if (!(new NombreNodo(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE).equals(
                        new NombreNodo(signatureElement.getNamespaceURI(),
signatureElement.getLocalName())))) {
                        // No se encuentra el nodo Signature
                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3
3) +
                                ConstantesXADES.ESPACIO +
ConstantesXADES.SIGNATURE);
                }

                // A partir del nodo raíz de la firma se obtiene el nodo
UnsignedSignatureProperties
```

```
		Element unsignedSignaturePropertiesElement = null;
		NodeList unsignedSignaturePropertiesNodes =
			signatureElement.getElementsByTagNameNS(xadesSchema,
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);

		if (unsignedSignaturePropertiesNodes.getLength() != 1) {
			// El nodo UnsignedSignatureProperties no existe o no es único

	log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_36) + ConstantesXADES.ESPACIO +

	ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES +
	ConstantesXADES.ESPACIO +

	I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_37) +
ConstantesXADES.ESPACIO +

	unsignedSignaturePropertiesNodes.getLength());
			// El sistema no soporta nodos UnsignedSignatureProperties
múltiples
			throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4
1));
		} else
			unsignedSignaturePropertiesElement =
(Element)unsignedSignaturePropertiesNodes.item(0);

		// Se añaden los elementos propios de la firma XADES-X
		switch (dataToSign.getXAdESXType()) {
			case TYPE_2:
				addXadesX2(unsignedSignaturePropertiesElement,
dataToSign.getTimeStampGenerator());
				break;
			case TYPE_1:
			default:
```

```
                                addXadesX(unsignedSignaturePropertiesElement,
dataToSign.getTimeStampGenerator());
                    }
        }
        else
            {
                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_2
5)) ;
            }
    }

    if(xadesXL) {
        // Para realizar la firma XADES-XL se deben completar antes los
        // formatos de firmas XADES-T, XADES-C y XADES-X

            if (xadesT && xadesC && xadesX) {
              try {
                // Añadimos XADES-XL

                addXadesXL(firma.getElement(), respuestas, schema);
              }
              catch (Exception e) {
                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_1
2) + e.getMessage(), e) ;
              }
            }
            else {
                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_1
3)) ;
            }
    }
```

```
    Object[] res = new Object[3];
    res[0] = doc;
    if (xadesC && !xadesXL)
        res[1] = respuestas;
    else
        res[1] = null;

    res[2]= idSignatureValue;

    return res;
  }

        private void convertICertStatus2RespYCerts(List<ICertStatus> status,
ArrayList<X509Certificate> certificadosConOCSP, ArrayList<RespYCerts> resps) {
            //Hashtable<X509Certificate,RespYCerts> resps = new
Hashtable<X509Certificate,RespYCerts>((status != null) ? status.size() : 0);
                if (status != null) {
                        Iterator<ICertStatus> itStatus = status.iterator();
                        while (itStatus.hasNext()) {
                                RespYCerts resp = new RespYCerts();
                                resp.setCertstatus(itStatus.next());

        if(!certificadosConOCSP.contains(resp.getCertstatus().getCertificate())) {

certificadosConOCSP.add(resp.getCertstatus().getCertificate());
                                   resps.add(resp);
                                }
                        }
                }
                //return resps;
        }
```

```
  /**
   * Este método realiza la implementación de la firma XADES-BES
   * @param doc Documento de firma
   * @param firmaID Identificador del nodo de firma
   * @param firmaCertificado Certificado que realiza la firma
   * @param elementoPrincipalFirma Elemento principal del nodo de firma
   * @param schemaXades Esquema XAdES a utilizar
   * @param dataToSign Objeto con la información de los datos a firmar
   * @param algDigestXML Valor del atributo que indica el algoritmo de digest
   * @return Documento de firma con formato XADES-BES
   * @throws Exception
   */
  private Document addXades(Document doc,
        String firmaID,
        X509Certificate firmaCertificado,
        Element elementoPrincipalFirma,
        XAdESSchemas schemaXades,
        DataToSign dataToSign,
        String algDigestXML) throws AddXadesException {

      // Añadimos el objeto
      Element elementoObjeto = doc.createElementNS(ConstantesXADES.SCHEMA_DSIG,
xmldsigNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.OBJECT);
      elementoObjeto.setAttributeNS(null, ConstantesXADES.ID,
UtilidadTratarNodo.newID(doc, firmaID + ConstantesXADES.GUION_OBJECT ));
      elementoPrincipalFirma.appendChild(elementoObjeto) ;

      // Creamos el QualifyingProperties
      Element elemntQualifyingProperties = doc.createElementNS(xadesSchema, xadesNS
+ ConstantesXADES.DOS_PUNTOS + ConstantesXADES.QUALIFYING_PROPERTIES);
      elementoObjeto.appendChild(elemntQualifyingProperties);
```

```
// Creamos los atributos de QualifyingProperties
elemntQualifyingProperties.setAttributeNS(null, ConstantesXADES.TARGET ,
ConstantesXADES.ALMOHADILLA + firmaID);

// Creamos el elemento SignedProperties
Element propiedadesFirmadasElemento = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.SIGNED_PROPERTIES);
elemntQualifyingProperties.appendChild(propiedadesFirmadasElemento);

// Creamos los atributos de SignedProperties
propiedadesFirmadasElemento.setAttributeNS(null, ConstantesXADES.ID, firmaID  +
idSigProperties);

// Creamos el xades:SignedSignatureProperties
Element propiedadesFirmadasElementoFirma = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.SIGNED_SIGNATURE_PROPERTIES);
propiedadesFirmadasElemento.appendChild(propiedadesFirmadasElementoFirma);

// Creamos el xades:SigningTime
Date signingDate = dataToSign.getSignDate();
if ((signingDate == null) && (schemaXades.equals(XAdESSchemas.XAdES_111)))
    throw new AddXadesException("SigningTime es requerido");
if (signingDate != null) {
        SigningTime tiempoFirma = new SigningTime(schemaXades, signingDate);
        Element tiempoFirmaElemento = null;
                try {
                        tiempoFirmaElemento = tiempoFirma.createElement(doc,
xadesNS);
                } catch (InvalidInfoNodeException e) {
                        throw new AddXadesException(e.getMessage(), e);
                }
```

```
            propiedadesFirmadasElementoFirma.appendChild(tiempoFirmaElemento);
    }

    // Creamos el xades:SigningCertificate
    Element certificadoFirmaElemento = doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.SIGNING_CERTIFICATE);
    propiedadesFirmadasElementoFirma.appendChild(certificadoFirmaElemento);

    // Creamos el xades:Cert
    Element certificadoElemento = doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CERT);
    File signCertFile = dataToSign.getSigningCert();
    if (signCertFile != null) {
        String uri = UtilidadFicheros.relativizeRute(dataToSign.getBaseURI(), signCertFile);
        certificadoElemento.setAttributeNS(null, ConstantesXADES.URI_MAYUS, uri);
    }

    // Creamos el xades:CertDigest
    Element resumenCertificadoElemento = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CERT_DIGEST);

    // Creamos el xades:DigestMethod
    Element metodoResumenElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.DIGEST_METHOD);
    metodoResumenElemento.setAttributeNS(null, ConstantesXADES.ALGORITHM,
algDigestXML);

    // Creamos el xades:DigestValue
    String resumenCertificado =ConstantesXADES.CADENA_VACIA;

    try {
```

```
                    MessageDigest resumenCertificadoTemp =
UtilidadFirmaElectronica.getMessageDigest(algDigestXML);

                    if (resumenCertificadoTemp == null)

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_16));

      byte[] byteMessageDigest
=resumenCertificadoTemp.digest(firmaCertificado.getEncoded());

      resumenCertificado = new String(Base64Coder.encode(byteMessageDigest));

   }

   catch (CertificateEncodingException cee)

   {

       throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_17));

   }


   Element elementDigestValue =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.DIGEST_VALUE);

   elementDigestValue.appendChild(doc.createTextNode(resumenCertificado));


   // Creamos el xades:IssuerSerial

   Element elementoEmisorSerial = doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.ISSUER_SERIAL );


   // Creamos el xades:X509IssuerName

   Element elementoX509EmisorNombre =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.X_509_ISSUER_NAME);


   String issuerName = null;

   try { // Se comprueba si la cadena puede ser codificada en UTF8 o contiene caracteres
especiales

       issuerName = firmaCertificado.getIssuerX500Principal().getName();
```

```
        if (log.isTraceEnabled()) {
                log.trace("Certificado emisor obtenido del X500: " + issuerName);
        }
        Charset charsetUtf = Charset.forName(ConstantesXADES.UTF8);
        issuerName =
charsetUtf.decode(ByteBuffer.wrap(issuerName.getBytes(charsetUtf))).toString();
        if (log.isTraceEnabled()) {
                log.trace("Emisor decodificado en UTF8:" + issuerName);
        }
    } catch (Exception e1) {
        if (log.isDebugEnabled()) {
                log.error("Error al codificar el emisor en UTF-8. Se toma su valor con el
charset original.", e1);
        }
                        issuerName = firmaCertificado.getIssuerDN().getName();
    }
    if (log.isTraceEnabled()) {
                log.debug("Certificado emisor: " + issuerName);

                Charset charset = null;
                Iterator<Entry<String, Charset>> charsets =
Charset.availableCharsets().entrySet().iterator();
                log.debug("Charsets disponibles encontrados: " +
Charset.availableCharsets().size());
                while (charsets.hasNext()) {
                        charset = charsets.next().getValue();
                        //log.debug("Codificando en " + charset);
                        //log.debug("Nombre codificado: " + new
String(issuerName.getBytes(), charset));
                        byte[] data1 = null;
                        byte[] data2 = null;
                        try {
                                data1 = issuerName.getBytes(charset);
```

```
                                data2 =
firmaCertificado.getIssuerX500Principal().getName().getBytes(charset);
                        } catch (Exception e) {
                                log.error("No se puede codificar la cadena en " +
charset.displayName(), e);
                                continue;
                        }
                        if (data1.length == data2.length) {
                                for (int i = 0; i < data1.length; i++) {
                                        if (data1[i] != data2[i])
                                                log.debug("El nombre del Issuer leído y el
X500 original no coinciden en formato " + charset.displayName() + ": " + data1[i] + " - " +
data2[i]);
                                        continue;
                                }
                        } else {
                                log.debug("No coincide el tamaño en " + charset);
                        }
                }
        }
    elementoX509EmisorNombre.appendChild(doc.createTextNode(issuerName));

    // Creamos el xades:X509SerialNumber
    Element elementoX509NumeroSerial =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.X_509_SERIAL_NUMBER);

elementoX509NumeroSerial.appendChild(doc.createTextNode(firmaCertificado.getSerial
Number().toString()));

    // Creamos el xades:SignatureProductionPlace.
    String[] produc = dataToSign.getProductionPlace();
    if (produc != null) {
```

```
        SignatureProductionPlace spp = new SignatureProductionPlace(schemaXades,
produc[0], produc[1], produc[2], produc[3]);
            Element productionPlaceElemento = null;
                    try {
                        productionPlaceElemento = spp.createElement(doc,
xadesNS);
                    } catch (InvalidInfoNodeException e) {
                        throw new AddXadesException(e.getMessage(), e);
                    }

propiedadesFirmadasElementoFirma.appendChild(productionPlaceElemento);
    }

    resumenCertificadoElemento.appendChild(metodoResumenElemento);
    resumenCertificadoElemento.appendChild(elementDigestValue);

    certificadoElemento.appendChild(resumenCertificadoElemento);

    elementoEmisorSerial.appendChild(elementoX509EmisorNombre);
    elementoEmisorSerial.appendChild(elementoX509NumeroSerial);

    certificadoElemento.appendChild(elementoEmisorSerial);



    certificadoFirmaElemento.appendChild(certificadoElemento);

    // Se crea el xades:SignerRole. Para ello se consulta el fichero de propiedades
    ArrayList<IClaimedRole> rolesFirmante = dataToSign.getClaimedRoles();
    if (rolesFirmante != null) {
      Element elementoRoleFirmanteElemento = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.SIGNER_ROLE);

propiedadesFirmadasElementoFirma.appendChild(elementoRoleFirmanteElemento);
```

```
        Element elementoRolesDemandadosElementos =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.CLAIMED_ROLES);


        elementoRoleFirmanteElemento.appendChild(elementoRolesDemandadosEleme
ntos);
      Iterator<IClaimedRole> it = rolesFirmante.iterator();
      while (it.hasNext()) {
                Element elementClaimedRoleElement =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.CLAIMED_ROLE);


        elementClaimedRoleElement.appendChild(it.next().createClaimedRoleContent(d
oc));


        elementoRolesDemandadosElementos.appendChild(elementClaimedRoleEleme
nt);
      }
    }


    // Se agrega información sobre los objetos firmados
    ArrayList<ObjectToSign> objects = dataToSign.getObjects();
    if (objects != null && objects.size() > 0) {


        // Se crea el elemento SignedDataObjectProperties
        Element signedDataObjectProperties = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.XADES_SIGNED_DATA_OBJECT_PROPERTIES);


        Iterator<ObjectToSign> it = objects.iterator();
        DataObjectFormat dof = null;  // Nodo a escribir
        ObjectIdentifier oi = null;


        ObjectToSign obj = null;                    // Objeto de datos
```

```
String id = null;                    // Identificador del objeto
String desc = null;                          // Descripción del formato del objeto
String tipoMIME = null;              // Tipo MIME del objeto firmado
URI encoding = null;                 // Codificación del objeto

URI referenceId = null;
while (it.hasNext()) {
        obj = it.next();

        if (obj != null) {
                // Se recupera el identificador
                id = obj.getId();
                // Se recoge la descripción
        desc = obj.getDescription();
        // Se recoge el tipo MIME
        tipoMIME = obj.getMimeType();
        // Se recoge la codificación
        encoding = obj.getEncoding();
        // Se recupera ObjectIdentifier
        oi = obj.getObjectIdentifier();
        }

        if ((desc == null) && (tipoMIME == null) && (oi == null))
                continue;

        try {
                                desc = new String(desc.getBytes(),
ConstantesXADES.UTF8);
                        } catch (UnsupportedEncodingException e1) {
                                if (log.isDebugEnabled()) {
                                        log.debug(e1);
```

```
                                        }
                                        try {
                                                desc = URLEncoder.encode(desc,
ConstantesXADES.UTF8);
                                        } catch (UnsupportedEncodingException e) {
                                                log.error(e.getMessage(), e);
                                                desc = "Unknown";
                                        }
                                }

                // Identificador del objeto
                if (id != null) {
                        try {
                                referenceId = new URI(id);
                        } catch (URISyntaxException e) {
                                log.error(e.getMessage(), e);
                        }
                }

                if ((referenceId == null) || (schemaXades == null))
                {
                        log.error("No se puede incluir el objeto DataObjectFormat porque
faltan datos");
                        throw new
AddXadesException(i18n.getLocalMessage(ConstantsXAdES.I18N_SIGN_2));
                }

                dof = new DataObjectFormat(schemaXades,
                                referenceId,
                                desc,
                                tipoMIME);
```

```
                if (encoding != null)
                        dof.setEncoding(encoding);

                if (oi != null)
                        dof.setObjectIdentifier(oi);

                try {

        signedDataObjectProperties.appendChild(dof.createElement(doc, xadesNS));
                                } catch (DOMException e) {
                                        throw new AddXadesException(e.getMessage(), e);
                                } catch (InvalidInfoNodeException e) {
                                        log.error(e.getMessage(), e);
                                        throw new
AddXadesException(i18n.getLocalMessage(ConstantsXAdES.I18N_SIGN_2));
                                }
        }

        // Se añade el nodo generado, si contiene información
        if (signedDataObjectProperties.getChildNodes().getLength() > 0)

        propiedadesFirmadasElemento.appendChild(signedDataObjectProperties);
    }

    return null;
  }

  private void addXadesEPES(Element elementoPrincipalFirma, String
confPolicyManager) throws AddXadesException {
        if (confPolicyManager == null) {
                confPolicyManager =
ConstantesXADES.LIBRERIAXADES_IMPLIEDPOLICY_MANAGER;
        }
```

```
        // Se obtiene el manager para la política indicada
        IFirmaPolicy policyManager =
PoliciesManager.getInstance().getEscritorPolicy(confPolicyManager);
        if (policyManager == null) {
                // PolicyManager buscado no disponible

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_46) +
                                ConstantesXADES.ESPACIO + confPolicyManager);
                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_47));
        }

        XAdESSchemas schema = XAdESSchemas.getXAdESSchema(xadesSchema);
        if (schema == null) {

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_44) +
                                ConstantesXADES.ESPACIO + xadesSchema);
                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_45));
        }

        try {
                policyManager.writePolicyNode(elementoPrincipalFirma, xmldsigNS,
xadesNS, schema);
        } catch (PolicyException ex) {
                // Error escribiendo políticas

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_48) +
                                ConstantesXADES.ESPACIO + ex.getMessage(), ex);
                throw new AddXadesException(ex.getMessage(), ex);
```

```
        }
    }

    /**
     * Este método añade la implementación para XADES-T
     * @param doc Documento de firma con formato XADES-BES
     * @param firmaID Identificador del nodo de firma
     * @param selloTiempo Respuesta del servidor TSA con el sello de tiempo en formato binario
     * @return Documento de firma con formato XADES-T
     * @throws Exception
     */
    private Document addXadesT(Element firma, String firmaID, byte[] selloTiempo)
    throws AddXadesException   {

        Document doc = firma.getOwnerDocument();
        Element elementoPrincipal = null ;
        NodeList nodos = firma.getElementsByTagNameNS(xadesSchema, ConstantesXADES.QUALIFYING_PROPERTIES);
        if(nodos.getLength() != 0)
            elementoPrincipal = (Element)nodos.item(0);
        else
            throw new AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_18)) ;

        // Se comprueba si existen ya los nodos Unsigned o se deben crear nuevos
        Element propiedadesElementosNoFirmados = null;
        ArrayList<Element> nodosUnsigendProp = null;
            try {
                nodosUnsigendProp = UtilidadTratarNodo.obtenerNodos(elementoPrincipal, 1,
```

```
                                        new NombreNodo(xadesSchema,
ConstantesXADES.UNSIGNED_PROPERTIES));
                } catch (FirmaXMLError e) { /** No se hace nada. */}

        if (nodosUnsigendProp != null && nodosUnsigendProp.size() == 1) {
                        propiedadesElementosNoFirmados = nodosUnsigendProp.get(0);
        } else {
                propiedadesElementosNoFirmados =
                        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.UNSIGNED_PROPERTIES);

                // Creamos los atributos de UnSignedProperties
                Attr propiedadesNoFirmadasId = doc.createAttributeNS(null,
ConstantesXADES.ID);
                propiedadesNoFirmadasId.setValue(UtilidadTratarNodo.newID(doc,
                                firmaID  +
ConstantesXADES.GUION_UNSIGNED_PROPERTIES));
                NamedNodeMap atributosSinFirmarPropiedadesElemento =
                        propiedadesElementosNoFirmados.getAttributes();

        atributosSinFirmarPropiedadesElemento.setNamedItem(propiedadesNoFirmadas
Id);
        }

        Element propiedadesSinFirmarFirmaElementos = null;
        ArrayList<Element> nodosUnsigendSigProp = null;
                try {
                        nodosUnsigendSigProp =
UtilidadTratarNodo.obtenerNodos(elementoPrincipal, 2,
                                        new NombreNodo(xadesSchema,
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES));
                } catch (FirmaXMLError e) { /** No se hace nada. */}

        if (nodosUnsigendSigProp != null && nodosUnsigendSigProp.size() == 1) {
```

```
                propiedadesSinFirmarFirmaElementos = nodosUnsigendSigProp.get(0);


        } else {

                propiedadesSinFirmarFirmaElementos =

                        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);

        }


        // Se buscan otros sellos de tiempo en la firma y se les asigna una Id si no la tienen

        NodeList sellosPreexistentes = doc.getElementsByTagNameNS(xadesSchema,
ConstantesXADES.SIGNATURE_TIME_STAMP);

        int numSellos = sellosPreexistentes.getLength();

        for (int i = 0; i < numSellos; ++i) {

                Element sello = (Element) sellosPreexistentes.item(i);

                String selloId = sello.getAttribute(ConstantesXADES.ID);

                if (selloId == null) {

                        Attr informacionElementoSigTimeStamp =
doc.createAttributeNS(null, ConstantesXADES.ID);

                        selloId = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SELLO_TIEMPO);

                informacionElementoSigTimeStamp.setValue(selloId);

                sello.getAttributes().setNamedItem(informacionElementoSigTimeStamp);

                }

                // Se almacena su nombre de Id por si es preciso referenciarlos

                idNodoSelloTiempo.add(selloId);

        }


        // Se crea el nodo de sello de tiempo

        Element tiempoSelloElementoFirma =

                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.SIGNATURE_TIME_STAMP);


        // Se escribe una Id única
```

```
	Attr informacionElementoSigTimeStamp = doc.createAttributeNS(null,
ConstantesXADES.ID);

	String idSelloTiempo = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SELLO_TIEMPO);

	informacionElementoSigTimeStamp.setValue(idSelloTiempo);

	idNodoSelloTiempo.add(idSelloTiempo);


	tiempoSelloElementoFirma.getAttributes().setNamedItem(informacionElementoS
igTimeStamp);


	// Se incluye un nodo que referencia a la Id de SignatureValue

	if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)

			|| ConstantesXADES.SCHEMA_XADES_122.equals(xadesSchema)) {


		String nombreNodoUri = null;

		String tipoUri = null;

		if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)) {

			nombreNodoUri = ConstantesXADES.HASH_DATA_INFO;

			tipoUri = ConstantesXADES.URI_MINUS;

		} else {

			nombreNodoUri = ConstantesXADES.INCLUDE;

			tipoUri = ConstantesXADES.URI_MAYUS;

		}


	Element informacionElementoHashDatos = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + nombreNodoUri);


	ArrayList<Element> listElements = new ArrayList<Element>();

			try {

				listElements = UtilidadTratarNodo.obtenerNodos(firma, 2,
new NombreNodo(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE_VALUE));

			} catch (FirmaXMLError e) {
```

```
        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_5), e);

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_5));

                        }

        if (listElements.size() != 1) {


        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_5));

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_5));

        }

        idSignatureValue = listElements.get(0).getAttribute(ConstantesXADES.ID);

        if (idSignatureValue == null) {


        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_5));

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_5));

        }


                Attr informacionElementoHashDatosUri = doc.createAttributeNS(null,
tipoUri);


        informacionElementoHashDatosUri.setValue(ConstantesXADES.ALMOHADILLA +
idSignatureValue);


                NamedNodeMap informacionAtributosElementoHashDatos =
informacionElementoHashDatos.getAttributes();


        informacionAtributosElementoHashDatos.setNamedItem(informacionElementoH
ashDatosUri);
```

```
        tiempoSelloElementoFirma.appendChild(informacionElementoHashDatos) ;
        }

        // Se crea el nodo canonicalizationMethod en los esquemas 1.2.2 y 1.3.2
        if (!ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)) {
                Element canonicalizationElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CANONICALIZATION_METHOD);

                Attr canonicalizationAttribute = doc.createAttributeNS(null,
ConstantesXADES.ALGORITHM);

        canonicalizationAttribute.setValue(Transforms.TRANSFORM_C14N_OMIT_COMME
NTS);

        canonicalizationElemento.getAttributes().setNamedItem(canonicalizationAttribut
e);

                tiempoSelloElementoFirma.appendChild(canonicalizationElemento);
        }

                // Se crea el nodo del sello de tiempo
                Element tiempoSelloEncapsulado =
                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.ENCAPSULATED_TIME_STAMP);

        tiempoSelloEncapsulado.appendChild(
                        doc.createTextNode(new
String(Base64Coder.encode(selloTiempo))));
        Attr tiempoSelloEncapsuladoId = doc.createAttributeNS(null,
ConstantesXADES.ID);
        String idEncapsulated = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SELLO_TIEMPO_TOKEN);
        tiempoSelloEncapsuladoId.setValue(idEncapsulated);
```

```
        tiempoSelloEncapsulado.getAttributes().setNamedItem(tiempoSelloEncapsulado
Id);



        tiempoSelloElementoFirma.appendChild(tiempoSelloEncapsulado);


        propiedadesSinFirmarFirmaElementos.appendChild(tiempoSelloElementoFirma);

        propiedadesElementosNoFirmados.appendChild(propiedadesSinFirmarFirmaEle
mentos);
        elementoPrincipal.appendChild(propiedadesElementosNoFirmados);
        return doc;
  }


  /**
   * Este método añade la implementacion para XADES-C
   * @param doc Documento de firma con formato XADES-T
   * @param tiempoRespuesta Fecha y hora de la respuesta del servidor OCSP
   * @param mensajeRespuesta Valor del OCSPResponse
   * @param certRefs Cadena de Certificación del certificado de firma
   * @return Documento de firma con formato XADES-C
   * @throws Exception
   */
  private Document addXadesC(Element firma,
                  ArrayList<RespYCerts> respuestas,
                  XAdESSchemas schema,
                  String algDigestXML)
  throws AddXadesException
  {
        Document doc = firma.getOwnerDocument();
        // Recogemos el nodo UnsignedSignatureProperties del cual dependen los nodos
```

```
        // que hay que añadir para completar la firma XADES-C
        Element elementoPrincipal = null ;
        ArrayList<X509Certificate> certRefs = null;

                String tipoUri = null;
                if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)) {
                        tipoUri = ConstantesXADES.URI_MINUS;
                } else {
                        tipoUri = ConstantesXADES.URI_MAYUS;
                }

        NodeList nodos = firma.getElementsByTagNameNS(xadesSchema,
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);
        if(nodos.getLength() != 0)
        {
                elementoPrincipal = (Element)nodos.item(0);
    }
    else
    {
       throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_19)) ;
    }

    // Aqui vienen las llamadas para los certificados
    Element certificadosElementosFirma =
        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.COMPLETE_CERTIFICATE_REFS);
    Element revocacionesElementoFirma =
      doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.COMPLETE_REVOCATION_REFS);
```

```
// Construye las referencias del certificado
int size = respuestas.size();
if (size > 0) {
    certRefs = new ArrayList<X509Certificate> (size);
    for(int x = 0; x < size; ++x) {

if(!(respuestas.get(x)).getCertstatus().getStatus().equals(ICertStatus.CERT_STATUS.valid))
{
        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_56) + " " + (respuestas.get(x)).getCertstatus().getCertificate().getSubjectDN() + " " +
(respuestas.get(x)).getCertstatus().getCertificate().getSerialNumber()) ;
      }
          certRefs.add((respuestas.get(x)).getCertstatus().getCertificate());
    }
} else {
    throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_56)) ;
}

if(certRefs != null)
{
    // Se le agrega una Id única
    Attr informacionElementoCertRef = doc.createAttributeNS(null,
ConstantesXADES.ID);
    idNodoCertificateRefs = UtilidadTratarNodo.newID(doc,
ConstantesXADES.COMPLETE_CERTIFICATE_REFS);
    informacionElementoCertRef.setValue(idNodoCertificateRefs);

    certificadosElementosFirma.getAttributes().setNamedItem(informacionElemento
CertRef);

    Element elementoCertRefs =
```

```
        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CERT_REFS);


        certificadosElementosFirma.appendChild(elementoCertRefs);

        int longitud = certRefs.size();


        // Se agrega una id al certificado de firma

        String idNueva = UtilidadTratarNodo.newID(doc,
ConstantesXADES.LIBRERIAXADES_CERT_PATH);

        respuestas.get(0).setIdCertificado(idNueva);


        for (int i = 1; i < longitud; i++) // Se salta el primero porque es el certificado firmante

        {

                X509Certificate firmaCertificado = (X509Certificate) certRefs.get(i);

                Element elementCertRef = doc.createElementNS(xadesSchema, xadesNS
+ ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CERT);


                // Creamos los atributos de UnSignedProperties

        Attr uris = doc.createAttributeNS(null, tipoUri);

                                // AppPerfect: Falso positivo. No son expresiones
constantes

        idNueva = UtilidadTratarNodo.newID(doc,
ConstantesXADES.LIBRERIAXADES_CERT_PATH);

        uris.setValue( ConstantesXADES.ALMOHADILLA + idNueva );

        respuestas.get(i).setIdCertificado(idNueva);

        NamedNodeMap atributosURI = elementCertRef.getAttributes();

        atributosURI.setNamedItem(uris);


            Element resumenElementoCert = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CERT_DIGEST);


            // Creamos el xades:DigestMethod
```

```
        Element metodoResumenElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.DIGEST_METHOD);


        // Creamos los atributos de DigestMethod

        Attr propiedadesFirmaAlgoritmo = doc.createAttributeNS(null,
ConstantesXADES.ALGORITHM);

        propiedadesFirmaAlgoritmo.setValue(algDigestXML);

        NamedNodeMap cualidadesMetodoResumenElemento =

            metodoResumenElemento.getAttributes();


cualidadesMetodoResumenElemento.setNamedItem(propiedadesFirmaAlgoritmo);


        // Creamos el xades:DigestValue

        String resumenCertificado = ConstantesXADES.CADENA_VACIA;

        try

        {

                    MessageDigest resumenCertificadoTemp =
UtilidadFirmaElectronica.getMessageDigest(algDigestXML);

                    if (resumenCertificadoTemp == null)

            throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_16));

            byte[] resumenMensajeByte
=resumenCertificadoTemp.digest(firmaCertificado.getEncoded());

            resumenCertificado = new
String(Base64Coder.encode(resumenMensajeByte));

        } catch (CertificateEncodingException e) {

            log.error(e);

            throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_23));

        }


        Element elementDigestValue =
```

```
                doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.DIGEST_VALUE);
            elementDigestValue.appendChild(
                doc.createTextNode(resumenCertificado));

            // Creamos el xades:IssuerSerial
            Element elementoEmisorSerial =
                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.ISSUER_SERIAL);
            // Creamos el xades:X509IssuerName
            Element elementoX509EmisorNombre =
                doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.X_509_ISSUER_NAME);
            String issuerName = null;
            try { // Se comprueba si la cadena puede ser codificada en UTF8 o contiene
caracteres especiales
                issuerName = firmaCertificado.getIssuerX500Principal().getName();
                if (log.isTraceEnabled()) {
                        log.trace("Certificado emisor obtenido del Issuer X500: " +
issuerName);
                }
                Charset charsetUtf = Charset.forName(ConstantesXADES.UTF8);
                issuerName =
charsetUtf.decode(ByteBuffer.wrap(issuerName.getBytes())).toString();
                if (log.isTraceEnabled()) {
                        log.trace("Emisor decodificado en UTF8:" + issuerName);
                }
            } catch (Exception e1) {
                if (log.isDebugEnabled()) {
                        log.error("Error al codificar el emisor en UTF-8. Se toma su valor
con el charset original.", e1);
                }
                        issuerName = firmaCertificado.getIssuerDN().getName();
            }
```

```
            if (log.isDebugEnabled()) {
                        log.debug("Certificado emisor: " + issuerName);
                }

elementoX509EmisorNombre.appendChild(doc.createTextNode(issuerName));

            // Creamos el xades:X509SerialNumber
            Element elementoX509NumeroSerial =
                doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.X_509_SERIAL_NUMBER);
            elementoX509NumeroSerial.appendChild(
                doc.createTextNode(firmaCertificado.getSerialNumber().toString()));

            //Add references
            elementoEmisorSerial.appendChild(elementoX509EmisorNombre);
            elementoEmisorSerial.appendChild(elementoX509NumeroSerial);

            resumenElementoCert.appendChild(metodoResumenElemento);
            resumenElementoCert.appendChild(elementDigestValue);

            elementCertRef.appendChild(resumenElementoCert);
            elementCertRef.appendChild(elementoEmisorSerial);

            elementoCertRefs.appendChild(elementCertRef);
        }
    }

        Element elementOCSPRef = null;
        String tiempoRespuesta = null;
        byte[] mensajeRespuesta = null;
```

```
if (size > 0) {

        // Se le agrega una Id única
        Attr informacionElementoCertRef = doc.createAttributeNS(null,
ConstantesXADES.ID);
        idNodoRevocationRefs = UtilidadTratarNodo.newID(doc,
ConstantesXADES.COMPLETE_REVOCATION_REFS);
        informacionElementoCertRef.setValue(idNodoRevocationRefs);

        revocacionesElementoFirma.getAttributes().setNamedItem(informacionElemento
CertRef);

        int nOCSPRefs = 0;
        int nCRLRefs = 0;

      // Construye el valor de la respuesta del servidor OCSP
      // bajo el nodo completo de la referencia de la revocación
        Element elementOCSPRefs =
        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.OCSP_REFS);
        CRLRefs elementCRLRefs = new CRLRefs(schema);

        for(int x = 0; x < size; ++x) {

                RespYCerts respYCert = respuestas.get(x);
                ICertStatus certStatus = respYCert.getCertstatus();
                if (certStatus instanceof IOCSPCertStatus) {
                        nOCSPRefs++;
                        IOCSPCertStatus respOcsp = (IOCSPCertStatus) certStatus;

                        tiempoRespuesta =
UtilidadFechas.formatFechaXML(respOcsp.getResponseDate());
                        IOCSPCertStatus.TYPE_RESPONDER tipoResponder =
respOcsp.getResponderType();
```

```
                    String valorResponder = respOcsp.getResponderID();
                    mensajeRespuesta = respOcsp.getEncoded();

                    elementOCSPRef = doc.createElementNS(xadesSchema, xadesNS
+ ConstantesXADES.DOS_PUNTOS + ConstantesXADES.OCSP_REF);

                    // Creamos los atributos de UnSignedProperties
                    String idNueva = UtilidadTratarNodo.newID(doc,
ConstantesXADES.OCSP);
                    respYCert.setIdRespStatus(idNueva);

            Element identificadorElementoOCSP =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.OCSP_IDENTIFIER);
            Attr uris = doc.createAttributeNS(null, tipoUri);
            uris.setValue( ConstantesXADES.ALMOHADILLA + idNueva );
            NamedNodeMap atributosURI =
identificadorElementoOCSP.getAttributes();
            atributosURI.setNamedItem(uris);

                    // Creamos el xades:DigestMethod
                    Element elementoRespondedorId =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.RESPONDER_ID);

                    Element responderFinal = elementoRespondedorId;
                    if
(!(ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)) &&
!(ConstantesXADES.SCHEMA_XADES_122.equals(xadesSchema))) {
                            Element hijo = null;
                            if
(tipoResponder.equals(IOCSPCertStatus.TYPE_RESPONDER.BY_NAME)) {
                                    hijo = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.BY_NAME);
```

```
                }
                else {
                        hijo = doc.createElementNS(xadesSchema,
xadesNS + ConstantesXADES.DOS_PUNTOS + ConstantesXADES.BY_KEY);
                }
                // TODO: tener en cuenta que podria no ser ninguno de
estos valores en un futuro
        elementoRespondedorId.appendChild(hijo);
        responderFinal = hijo;
        }

        // Se codifica según UTF-8
        try {
                valorResponder = new String(valorResponder.getBytes(),
ConstantesXADES.UTF8);
                } catch (UnsupportedEncodingException e1) {
                        if (log.isDebugEnabled()) {
                                log.debug(e1);
                        }
                        try {
                                valorResponder =
URLEncoder.encode(valorResponder, ConstantesXADES.UTF8);
                        } catch (UnsupportedEncodingException e) {
                                throw new AddXadesException("No
se pudo construir las referencias OCSP", e);
                        }
                }

        responderFinal.appendChild(doc.createTextNode(valorResponder));
```

```
                Element elementoProdujoEn =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.PRODUCE_AT);




        elementoProdujoEn.appendChild(doc.createTextNode(tiempoRespuesta));



        identificadorElementoOCSP.appendChild(elementoRespondedorId);

                identificadorElementoOCSP.appendChild(elementoProdujoEn);

                Element valorYResumenElemento =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.DIGEST_ALG_AND_VALUE);


                // Creamos el xades:DigestMethod

                Element metodoResumenElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.DIGEST_METHOD);


                // Creamos los atributos de DigestMethod

                Attr propiedadesAlgoritmoFirmado = doc.createAttributeNS(null,
ConstantesXADES.ALGORITHM);

                propiedadesAlgoritmoFirmado.setValue(algDigestXML);

                NamedNodeMap atributosMetodoResumenElemento =
metodoResumenElemento.getAttributes();


        atributosMetodoResumenElemento.setNamedItem(propiedadesAlgoritmoFirmad
o);


                // Creamos el xades:DigestValue

                // El mensaje de la respuesta es el OCSPResponse

                String digestCertificado =ConstantesXADES.CADENA_VACIA;

                MessageDigest resumenCertificadoTemp =
UtilidadFirmaElectronica.getMessageDigest(algDigestXML);

                if (resumenCertificadoTemp == null)
```

```
                                    throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_20));
                                byte[] resumenMensajeByte =
resumenCertificadoTemp.digest(mensajeRespuesta);
                                digestCertificado = new
String(Base64Coder.encode(resumenMensajeByte));


                                Element valorResumenElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.DIGEST_VALUE);



        valorResumenElemento.appendChild(doc.createTextNode(digestCertificado));



        valorYResumenElemento.appendChild(metodoResumenElemento);
                        valorYResumenElemento.appendChild(valorResumenElemento);


                        elementOCSPRef.appendChild(identificadorElementoOCSP);
                        elementOCSPRef.appendChild(valorYResumenElemento);


                        elementOCSPRefs.appendChild(elementOCSPRef);
                }
                else if (certStatus instanceof IX509CRLCertStatus) {
                        nCRLRefs++;
                        IX509CRLCertStatus respCRL = (IX509CRLCertStatus) certStatus;
                        try {
                                                CRLRef crlRef = new CRLRef(schema,
algDigestXML, respCRL.getX509CRL());


                                                String idNueva =
UtilidadTratarNodo.newID(doc, ConstantesXADES.CRL);


        crlRef.getCrlIdentifier().setUri(ConstantesXADES.ALMOHADILLA + idNueva);
```

```
                                        respYCert.setIdRespStatus(idNueva);


                                        elementCRLRefs.addCRLRef(crlRef);
                        } catch (InvalidInfoNodeException ex) {
                                        throw new AddXadesException("No se pudo
construir las referencias a CRLs", ex);
                                        }
                } else if (log.isDebugEnabled()) {
                        log.debug("Se salta el elemento número " + x);
                }


        }


        if (nCRLRefs > 0) {
                try {
                                        Element el = elementCRLRefs.createElement(doc,
xmldsigNS, xadesNS);
                                        revocacionesElementoFirma.appendChild(el);
                                } catch (InvalidInfoNodeException ex) {
                                throw new AddXadesException("No se pudo construir las
referencias a CRLs", ex);
                                }
        }


        if (nOCSPRefs > 0)
        revocacionesElementoFirma.appendChild(elementOCSPRefs);


    }


    elementoPrincipal.appendChild(certificadosElementosFirma);
    elementoPrincipal.appendChild(revocacionesElementoFirma);
```

```
    return doc;
  }

  /**
   * Este metodo añade la implementación del sello de tiempo de tipo 1 (implícito) para
   * XADES-X según los esquemas 1.2.2 y 1.3.2.
   * Los elementos sobre los que se calcula el sello son los siguientes:
         *                - SignatureValue
         *                - SignatureTimestamp
         *                - CompleteCertificateRefs
         *                - CompleteRevocationRefs
         *        Opcionalmente en el esquema 1.2.2 y 1.3.2:
         *                - AttributeCertificateRefs
         *                - AttributeRevocationRefs
         *
   * @param Element UnsignedSignatureProperties Nodo a partir del cual se añade el
nodo SigAndRefsTimeStamp
   * @param timeStampGenerator Generador de sellos de tiempo
   * @return Documento de firma con formato XADES-X
   * @throws AddXadesException En caso de error
   */
  private Document addXadesX(Element UnsignedSignatureProperties,
ITimeStampGenerator timeStampGenerator)
        throws AddXadesException {

        // Se obtiene el formato de la constante URI en función del esquema
                String tipoUri = null;
                String nombreNodoUri = null;
                if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)){
                        nombreNodoUri = ConstantesXADES.HASH_DATA_INFO;
                        tipoUri = ConstantesXADES.URI_MINUS;
                } else {
```

```
                nombreNodoUri = ConstantesXADES.INCLUDE;
                tipoUri = ConstantesXADES.URI_MAYUS;
        }

        // Se obtiene el documento que contiene al nodo UnsignedSignatureProperties
        Document doc = UnsignedSignatureProperties.getOwnerDocument();

        // Se obtiene el nodo Signature que contiene al nodo UnsignedSignatureProperties
(es el 4º padre, según esquema XAdES)
        Node padre = UnsignedSignatureProperties.getParentNode();
        for (int i = 0; i < 3; ++i) {
                if (padre != null)
                        padre = padre.getParentNode();
                else
                        // No se encuentra el nodo Signature
                        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_33) +
                                        ConstantesXADES.ESPACIO +
ConstantesXADES.SIGNATURE);
        }

        Element signatureElement = null;
        if (padre != null && ConstantesXADES.SIGNATURE.equals(padre.getLocalName()))
                signatureElement = (Element)padre;
        else
                // No se encuentra el nodo Signature
                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_33) +
                                ConstantesXADES.ESPACIO +
ConstantesXADES.SIGNATURE);

        // Se crea el nodo SigAndRefsTimeStamp
```

```
Element sigAndRefsTimeStampElement =
        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.SIG_AND_REFS_TIME_STAMP);

// Se escribe una Id única
        Attr informacionElementoSigTimeStamp = doc.createAttributeNS(null,
ConstantesXADES.ID);
        String idSelloTiempo = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SELLO_TIEMPO);
        informacionElementoSigTimeStamp.setValue(idSelloTiempo);
        idNodoSelloTiempo.add(idSelloTiempo);

        sigAndRefsTimeStampElement.getAttributes().setNamedItem(informacionElemen
toSigTimeStamp);

// Se coloca el nodo creado al final del nodo UnsignedSignatureProperties
UnsignedSignatureProperties.appendChild(sigAndRefsTimeStampElement);

// Se obtiene el listado de elementos de un sello de tiempo XAdES X de tipo 1
ArrayList<Element> elementosSelloX = null;
try {
                elementosSelloX =
UtilidadXadesX.obtenerListadoXADESX1imp(xadesSchema, signatureElement,
sigAndRefsTimeStampElement);
            } catch (BadFormedSignatureException e) {
                throw new AddXadesException(e.getMessage(), e);
            } catch (FirmaXMLError e) {
                throw new AddXadesException(e.getMessage(), e);
            }

            // Se añaden nodos de referencia a los nodos obtenidos para el cálculo del
sello (sólo para esquemas 1.2.2 y 1.1.1)
            if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema) ||
```

```
        ConstantesXADES.SCHEMA_XADES_122.equals(xadesSchema)) {
                        // Se obtienen las Ids de los nodos del sello de tiempo X
                        ArrayList<String> elementosIdSelloX =
UtilidadTratarNodo.obtenerIDs(elementosSelloX);


                        // Se crea una estructura con los nodos Include (1.2.2) o
HashDataInfo (1.1.1) que contienen las URIs que apuntan a estas IDs
                        ArrayList<Element> nodosUriReferencia = new ArrayList<Element>
(elementosIdSelloX.size());
                        Iterator<String> itIds = elementosIdSelloX.iterator();
                        while (itIds.hasNext()) {
                                String id = itIds.next();
                                Element uriNode =
                                        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + nombreNodoUri);
                Attr includeNodeUri = doc.createAttributeNS(null, tipoUri);
                includeNodeUri.setValue(ConstantesXADES.ALMOHADILLA + id);
                NamedNodeMap atributosNodo = uriNode.getAttributes();
                atributosNodo.setNamedItem(includeNodeUri);


                nodosUriReferencia.add(uriNode);
                        }


            // Se escribe en el nodo SigAndRefsTimeStamp el listado obtenido por orden
                        Iterator<Element> itUrisReferencia = nodosUriReferencia.iterator();
                        while (itUrisReferencia.hasNext()) {
                                Element includeNode = itUrisReferencia.next();


                                sigAndRefsTimeStampElement.appendChild(includeNode);
                        }
                }
```

```
// Se obtiene el Array de bytes de los nodos obtenidos
byte[] byteData = null;
try {
    byteData = UtilidadTratarNodo.obtenerByte(elementosSelloX,
CanonicalizationEnum.C14N_OMIT_COMMENTS);
} catch (FirmaXMLError e) {
    throw new AddXadesException(e.getMessage(), e);
}

if (timeStampGenerator == null) {
    throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_14)) ;
} else {
    try {
        byteData =
timeStampGenerator.generateTimeStamp(byteData);
    } catch (TimeStampException e) {
        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_11) + e.getMessage()) ;
    }
    String hashSelloX = new String(Base64Coder.encode(byteData));

    // Se crea el nodo canonicalizationMethod en los esquemas 1.2.2 y
1.3.2
    if
(!ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)) {
        Element canonicalizationElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CANONICALIZATION_METHOD);

        Attr canonicalizationAttribute = doc.createAttributeNS(null,
ConstantesXADES.ALGORITHM);
```

```
        canonicalizationAttribute.setValue(Transforms.TRANSFORM_C14N_OMIT_COMME
NTS);

        canonicalizationElemento.getAttributes().setNamedItem(canonicalizationAttribut
e);



        sigAndRefsTimeStampElement.appendChild(canonicalizationElemento);
                        }


                        // Escribimos el resultado en el nodo EncapsulatedTimeStamp
                        Element encapsulatedTimeStampNode =
                                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.ENCAPSULATED_TIME_STAMP);

        encapsulatedTimeStampNode.appendChild(doc.createTextNode(hashSelloX));

        sigAndRefsTimeStampElement.appendChild(encapsulatedTimeStampNode);
                }


                return doc;
  }


  /**
   * Este metodo añade la implementación del sello de tiempo de tipo 2 (explícito) para
   * XADES-X según los esquemas 1.1.1, 1.2.2 y 1.3.2.
   * Los elementos sobre los que se calcula el sello son los siguientes:
        *               - CompleteCertificateRefs
        *               - CompleteRevocationRefs
        *       Opcionalmente en el esquema 1.2.2 y 1.3.2:
        *               - AttributeCertificateRefs
        *               - AttributeRevocationRefs
        *
```

```
    * @param Element UnsignedSignatureProperties Nodo a partir del cual se añade el nodo RefsOnlyTimeStamp
    * @param timeStampGenerator Generador de sellos de tiempo
    * @return Documento de firma con formato XADES-X
    * @throws AddXadesException En caso de error
    */
   private Document addXadesX2(Element UnsignedSignatureProperties, ITimeStampGenerator timeStampGenerator)
            throws AddXadesException
            {
            // Se obtiene el formato de la constante URI en función del esquema
                    String tipoUri = null;
                    String nombreNodoUri = null;
                    if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)){
                            nombreNodoUri = ConstantesXADES.HASH_DATA_INFO;
                            tipoUri = ConstantesXADES.URI_MINUS;
                    } else {
                            nombreNodoUri = ConstantesXADES.INCLUDE;
                            tipoUri = ConstantesXADES.URI_MAYUS;
                    }

            // Se obtiene el documento que contiene al nodo UnsignedSignatureProperties
            Document doc = UnsignedSignatureProperties.getOwnerDocument();

            // Se obtiene el nodo Signature que contiene al nodo UnsignedSignatureProperties (es el 4º padre, según esquema XAdES)
            Node padre = UnsignedSignatureProperties.getParentNode();
            for (int i = 0; i < 3; ++i) {
                    if (padre != null)
                            padre = padre.getParentNode();
                    else
                            // No se encuentra el nodo Signature
```

```
                        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_33) +
                                                ConstantesXADES.ESPACIO +
ConstantesXADES.SIGNATURE);
        }

        Element signatureElement = null;
        if (padre != null && ConstantesXADES.SIGNATURE.equals(padre.getLocalName()))
                signatureElement = (Element)padre;
        else
                // No se encuentra el nodo Signature
                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_33) +
                                        ConstantesXADES.ESPACIO +
ConstantesXADES.SIGNATURE);

        // Se crea el nodo RefsOnlyTimeStamp
    Element refsOnlyTimeStampElement =
        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.REFS_ONLY_TIME_STAMP);

    // Se escribe una Id única
        Attr informacionElementoSigTimeStamp = doc.createAttributeNS(null,
ConstantesXADES.ID);
        String idSelloTiempo = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SELLO_TIEMPO);
        informacionElementoSigTimeStamp.setValue(idSelloTiempo);
        idNodoSelloTiempo.add(idSelloTiempo);

        refsOnlyTimeStampElement.getAttributes().setNamedItem(informacionElemento
SigTimeStamp);

    // Se coloca el nodo creado al final del nodo UnsignedSignatureProperties
```

```
UnsignedSignatureProperties.appendChild(refsOnlyTimeStampElement);


// Se obtiene el listado de elementos de un sello de tiempo XAdES X de tipo 2
ArrayList<Element> elementosSelloX = null;
try {
                elementosSelloX =
UtilidadXadesX.obtenerListadoXADESX2exp(xadesSchema, signatureElement,
refsOnlyTimeStampElement);
        } catch (BadFormedSignatureException e) {
                throw new AddXadesException(e.getMessage(), e);
        } catch (FirmaXMLError e) {
                throw new AddXadesException(e.getMessage(), e);
        }


        // Se añaden nodos de referencia a los nodos obtenidos para el cálculo del
sello (sólo para esquemas 1.2.2 y 1.1.1)
        if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema) ||

    ConstantesXADES.SCHEMA_XADES_122.equals(xadesSchema)) {
                // Se obtienen las Ids de los nodos del sello de tiempo X
                ArrayList<String> elementosIdSelloX =
UtilidadTratarNodo.obtenerIDs(elementosSelloX);


                // Se crea una estructura con los nodos Include (1.2.2) o
HashDataInfo (1.1.1) que contienen las URIs que apuntan a estas IDs
                ArrayList<Element> nodosUriReferencia = new ArrayList<Element>
(elementosIdSelloX.size());
                Iterator<String> itIds = elementosIdSelloX.iterator();
                while (itIds.hasNext()) {
                        String id = itIds.next();
                        Element uriNode =
                                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + nombreNodoUri);
        Attr includeNodeUri = doc.createAttributeNS(null, tipoUri);
```

```
                includeNodeUri.setValue(ConstantesXADES.ALMOHADILLA + id);
                NamedNodeMap atributosNodo = uriNode.getAttributes();
                atributosNodo.setNamedItem(includeNodeUri);

                nodosUriReferencia.add(uriNode);
                        }

        // Se escribe en el nodo RefsOnlyTimeStamp el listado obtenido, por orden
                        Iterator<Element> itUrisReferencia = nodosUriReferencia.iterator();
                        while (itUrisReferencia.hasNext()) {
                                Element includeNode = itUrisReferencia.next();

                                refsOnlyTimeStampElement.appendChild(includeNode);
                        }
                }

                // Se obtiene el Array de bytes de los nodos obtenidos
                byte[] byteData = null;
                try {
                        byteData = UtilidadTratarNodo.obtenerByte(elementosSelloX,
CanonicalizationEnum.C14N_OMIT_COMMENTS);
                } catch (FirmaXMLError e) {
                        throw new AddXadesException(e.getMessage(), e);
                }


                if (timeStampGenerator == null) {
                        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_14)) ;
                } else {
                        try {
```

```
                        byteData =
timeStampGenerator.generateTimeStamp(byteData);
                    } catch (TimeStampException e) {
                        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_11) + e.getMessage()) ;
                    }
                    String hashSelloX = new String(Base64Coder.encode(byteData));

                    // Se crea el nodo canonicalizationMethod en los esquemas 1.2.2 y
1.3.2
                    if
(!ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema)) {
                        Element canonicalizationElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CANONICALIZATION_METHOD);

                        Attr canonicalizationAttribute = doc.createAttributeNS(null,
ConstantesXADES.ALGORITHM);

        canonicalizationAttribute.setValue(Transforms.TRANSFORM_C14N_OMIT_COMME
NTS);

        canonicalizationElemento.getAttributes().setNamedItem(canonicalizationAttribut
e);


        refsOnlyTimeStampElement.appendChild(canonicalizationElemento);
                    }

                    // Escribimos el resultado en el nodo EncapsulatedTimeStamp
                    Element encapsulatedTimeStampNode =
                        doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.ENCAPSULATED_TIME_STAMP);

        encapsulatedTimeStampNode.appendChild(doc.createTextNode(hashSelloX));
```

```
            refsOnlyTimeStampElement.appendChild(encapsulatedTimeStampNode);
                    }

      return doc;
    }

    /**
     * Este metodo añade la implementacion para XADES-XL
     * @param doc Documento de firma con formato XADES-X
     * @param valorCertificado
     * @param valorRevocacion
     * @return Documento de firma con formato XADES-XL
     * @throws Exception
     */
    private Document addXadesXL(Element firma, ArrayList<RespYCerts> respuestas,
XAdESSchemas schema)
            throws AddXadesException
            {
            // Recogemos el nodo UnsignedSignatureProperties del cual dependen los nodos
            // que hay que añadir para completar la firma XADES-XL
            Document doc = firma.getOwnerDocument();
        Element elementoPrincipal = null ;

        NodeList nodosUnsignedSignatureProperties =
firma.getElementsByTagNameNS(schema.getSchemaUri(),
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);
        if(nodosUnsignedSignatureProperties.getLength() != 0)
          elementoPrincipal = (Element)nodosUnsignedSignatureProperties.item(0);
        else
            // No se encuentra el nodo UnsignedSignatureProperties
```

```
        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_19));


    // Se añaden los certificados referenciados en el nodo CertificateValues
    if(respuestas != null) {
        EncapsulatedX509Certificate encapsulatedX509certificate = null;
        ArrayList<EncapsulatedX509Certificate> certs = new
ArrayList<EncapsulatedX509Certificate> ();
            Iterator<RespYCerts> itResp = respuestas.iterator();
            boolean hasNext = itResp.hasNext();


            // Se salta el primero de la lista, que se corresponde con el certificado firmante
            // Ya que esta contenido dentro del nodo ds:KeyInfo
            if (hasNext) {
                itResp.next();
                hasNext = itResp.hasNext();
            }


            while (hasNext) {
                RespYCerts resp = itResp.next();
                hasNext = itResp.hasNext();
                encapsulatedX509certificate = new EncapsulatedX509Certificate(schema,
resp.getIdCertificado());
                try {


        encapsulatedX509certificate.setX509Certificate(resp.getCertstatus().getCertificat
e());
                                } catch (CertificateException e) {
                                    log.error(e.getMessage(), e);
                                    throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_23));
                                }
```

```
                certs.add(encapsulatedX509certificate);
            }

            CertificateValues certificateValues = new CertificateValues(schema, certs);
            Element certificateValuesElement = null;
            try {
                certificateValuesElement = certificateValues.createElement(doc,
xadesNS);
                    } catch (InvalidInfoNodeException e) {
                        log.error(e.getMessage(), e);
                        throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_23));
                    }

                    // Se escribe una Id única
                Attr atributoCertVal = doc.createAttributeNS(null, ConstantesXADES.ID);
                String idCertVal = UtilidadTratarNodo.newID(doc,
ConstantesXADES.CERTIFICATE_VALUES);
                atributoCertVal.setValue(idCertVal);
                certificateValuesElement.getAttributes().setNamedItem(atributoCertVal);

                    elementoPrincipal.appendChild(certificateValuesElement);

    // Se añade la respuesta del servidor OCSP
    Element valoresElementosRevocados =
      doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.REVOCATION_VALUES);

    Element valorElementOCSP =
      doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.OCSP_VALUES);
```

```
CRLValues valorElementoCRL = new CRLValues(schema);


        int nOcspResps = 0;

        int nCRLSResps = 0;

    itResp = respuestas.iterator();

        hasNext = itResp.hasNext();

        while (hasNext) {

            RespYCerts resp = itResp.next();

            hasNext = itResp.hasNext();

            ICertStatus respStatus = resp.getCertstatus();

            if (respStatus instanceof IOCSPCertStatus) {

                    nOcspResps++;

                    IOCSPCertStatus respOCSP = (IOCSPCertStatus) respStatus;

                    Element valorElementoEncapsuladoOCSP =
doc.createElementNS(xadesSchema, xadesNS + ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.ENCAPSULATED_OCSP_VALUE);

                    valorElementoEncapsuladoOCSP.appendChild(

                                doc.createTextNode(new
String(Base64Coder.encode(respOCSP.getEncoded()))));

                    valorElementoEncapsuladoOCSP.setAttributeNS(null,
ConstantesXADES.ID, resp.getIdRespStatus());


      valorElementOCSP.appendChild(valorElementoEncapsuladoOCSP);

                }

                else if (respStatus instanceof IX509CRLCertStatus) {

                    nCRLSResps++;

                    IX509CRLCertStatus respCRL = (IX509CRLCertStatus) respStatus;

                    try {

                        valorElementoCRL.addCRL(respCRL.getX509CRL(),
resp.getIdRespStatus());

                    } catch (InvalidInfoNodeException ex) {

                        throw new AddXadesException("No se pudo generar nodo
EncapsulatedCRLValue", ex);

                    }
```

```
                    }
                }

        if (nCRLSResps > 0) {
          try {
                                        Element el = valorElementoCRL.createElement(doc,
xadesNS);
                                        valoresElementosRevocados.appendChild(el);
                                    } catch (InvalidInfoNodeException ex) {
                                        throw new AddXadesException("No se pudo generar
nodo CRLValues", ex);
                                    }
        }

        if (nOcspResps > 0)
          valoresElementosRevocados.appendChild(valorElementOCSP);

                        // Se escribe una Id única
                Attr atributoRevVal = doc.createAttributeNS(null, ConstantesXADES.ID);
                String idRevVal = UtilidadTratarNodo.newID(doc,
ConstantesXADES.REVOCATION_VALUES);
                atributoRevVal.setValue(idRevVal);

          valoresElementosRevocados.getAttributes().setNamedItem(atributoRevVal);

        elementoPrincipal.appendChild(valoresElementosRevocados);
      }

      return doc;
    }

    private Document addXadesA (Element firma, byte[] selloTiempo, ArrayList<String> inc)
throws Exception {
```

```
        Document doc = firma.getOwnerDocument();


        ArrayList<Element> unsignedSignaturePropertiesNodes =
UtilidadTratarNodo.obtenerNodos(firma, 4,
                                        new NombreNodo(xadesSchema,
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES));
        Element unsignedSignaturePropertiesNode = null;
        if(unsignedSignaturePropertiesNodes.size() == 1)
                unsignedSignaturePropertiesNode =
(Element)unsignedSignaturePropertiesNodes.get(0);
        else
                // No se encuentra el nodo UnsignedSignatureProperties
                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_19)) ;


        Element archiveTimeStamp =
                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.ARCHIVE_TIME_STAMP);


        // Creamos los atributos de ArchiveTimeStamp (Id)
        Attr archiveTimeStampId = doc.createAttributeNS(null, ConstantesXADES.ID);
        archiveTimeStampId.setValue(UtilidadTratarNodo.newID(doc,
                        ConstantesXADES.ARCHIVE_TIME_STAMP +
ConstantesXADES.GUION));
        NamedNodeMap archiveTimeStampAttributesElement =
                archiveTimeStamp.getAttributes();
        archiveTimeStampAttributesElement.setNamedItem(archiveTimeStampId);


        // Se agrega el nodo EncapsulatedTimeStamp, con Id y Encoding como atributos
        Element encapsulatedTimeStamp =
                doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS +
```

```
                                ConstantesXADES.ENCAPSULATED_TIME_STAMP);

        // Se escribe una Id única
        Attr informacionElementoSigTimeStamp = doc.createAttributeNS(null,
ConstantesXADES.ID);
        String idSelloTiempo = UtilidadTratarNodo.newID(doc,
ConstantesXADES.SELLO_TIEMPO_TOKEN);
        informacionElementoSigTimeStamp.setValue(idSelloTiempo);
        idNodoSelloTiempo.add(idSelloTiempo);

        encapsulatedTimeStamp.getAttributes().setNamedItem(informacionElementoSigT
imeStamp);

        // Se agrega el CanonicalizationMethod
        Element canonicalizationElemento =
doc.createElementNS(ConstantesXADES.SCHEMA_DSIG, xmldsigNS +
ConstantesXADES.DOS_PUNTOS + ConstantesXADES.CANONICALIZATION_METHOD);

                Attr canonicalizationAttribute = doc.createAttributeNS(null,
ConstantesXADES.ALGORITHM);

        canonicalizationAttribute.setValue(Transforms.TRANSFORM_C14N_OMIT_COMME
NTS);

        canonicalizationElemento.getAttributes().setNamedItem(canonicalizationAttribut
e);

                archiveTimeStamp.appendChild(canonicalizationElemento);

    encapsulatedTimeStamp.appendChild(doc.createTextNode(new
String(Base64Coder.encode(selloTiempo))));

    // Se agregan, si existen, los nodos include
    if (inc != null) {
        Element includeNode = null;
        for (int i = 0; i < inc.size(); ++i) {
```

```
                includeNode = doc.createElementNS(xadesSchema, xadesNS +
ConstantesXADES.DOS_PUNTOS +
                                ConstantesXADES.INCLUDE);
                includeNode.setAttributeNS(null, ConstantesXADES.URI_MAYUS,
inc.get(i));
                archiveTimeStamp.appendChild(includeNode);
            }
        }

        archiveTimeStamp.appendChild(encapsulatedTimeStamp);

        // Se agrega el sello creado a las propiedades no firmadas
        unsignedSignaturePropertiesNode.appendChild(archiveTimeStamp);

            return doc;
    }

    /**
     * Contrafirma una firma según esquema XAdES y la deja en un fichero
     *
     * @param firmaCertificado Certificado con el que realizar la contrafirma
     * @param xml Objeto de la clase DataToSign con la información a contrafirmar
     * @param storeManager Almacén de certificados
     * @param nodoAFirmarId Identificador del nodo a contrafirmar
     * @param destino Directorio donde dejar la contrafirma
     * @param nombreArchivo Nombre del archivo que contendrá la contrafirma.
     * @return Identificador de la firma generada
     * @throws Exception
     */
    public String countersignFile(X509Certificate firmaCertificado,
            DataToSign xml, IPKStoreManager storeManager,
            String nodoAFirmarId, String destino, String nombreArchivo) throws Exception {
```

```
        PrivateKey pk = storeManager.getPrivateKey(firmaCertificado);
        Object[] res = countersign(firmaCertificado, xml, nodoAFirmarId, pk,
storeManager.getProvider(firmaCertificado));

        // Se guarda la firma en su destino
        File fichero = new File(destino, nombreArchivo);
        FileOutputStream f2 = new FileOutputStream(fichero);

        try {
                UtilidadFicheros.writeXML((Document)res[0], f2);
                return (String)res[2];
        } catch (Throwable t) {
                if (t.getMessage() != null &&
t.getMessage().startsWith(ConstantesXADES.JAVA_HEAP_SPACE))
                        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3));
                else
                        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4));
        } finally {
                f2.close();
        }
  }

  /**
   * Contrafirma una firma según esquema XAdES y la deja en un OutputStream
   *
   * @param firmaCertificado Certificado con el que realizar la contrafirma
   * @param xml Objeto de la clase DataToSign con la información a contrafirmar
   * @param storeManager Almacén de certificados
   * @param nodoAFirmarId Identificador del nodo a contrafirmar
   * @param salida OutputStream donde dejar la contrafirma
```

```
    * @return Identificador de la firma generada
    * @throws Exception
    */
   public String countersign2Stream(X509Certificate firmaCertificado,
         DataToSign xml, IPKStoreManager storeManager,
         String nodoAFirmarId, OutputStream salida) throws Exception {
            PrivateKey pk = storeManager.getPrivateKey(firmaCertificado);
            Object[] res = countersign(firmaCertificado, xml, nodoAFirmarId, pk,
storeManager.getProvider(firmaCertificado));

            // Se guarda la firma en su destino
            try {
                  UtilidadFicheros.writeXML((Document)res[0], salida);
                  return (String) res[2];
            } catch (Throwable t) {
                  if (t.getMessage() != null &&
t.getMessage().startsWith(ConstantesXADES.JAVA_HEAP_SPACE))
                        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3));
                  else
                        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4));
            }
   }

   /**
    * Contrafirma una firma según esquema XAdES
    *
    * @param certificadoFirma .- Certificado de firma
    * @param xml .- Contenido a firmar
    * @param nodoAFirmarId Identificador del nodo a contrafirmar
    * @param pk .- Clave privada del certificado
```

```
     * @param provieder .- Ruta donde guardar la firma generada
     * @param provider .- Proveedor criptográfico a utilizar
     */
    private Object[] countersign(X509Certificate certificadoFirma,
                        DataToSign xml, String nodoAFirmarId, PrivateKey pk,
                        Provider provider) throws Exception {

            es.mityc.javasign.utils.Utils.addBCProvider();
            Document doc = xml.getDocument();
            if (doc == null) {
                try {
                    InputStream is = xml.getInputStream();
                    if (is != null) {
                    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
                    dbf.setNamespaceAware(true);
                    DocumentBuilder db = dbf.newDocumentBuilder();
                    db.setErrorHandler(new IgnoreAllErrorHandler());
                            InputSource isour = new InputSource(is);
                            doc = db.parse(isour);

                    }
                } catch (IOException ex) {
                    throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_50));
                }
            }
        xml.setXMLEncoding(doc.getXmlEncoding());

            // Si no se indica nodo a contrafirmar se contrafirma la última firma disponible
            Node nodePadreNodoFirmar = null;
            if (nodoAFirmarId != null) {
```

```
	Element nodoAFirmar = UtilidadTratarNodo.getElementById(doc, nodoAFirmarId);
	if(nodoAFirmar == null) {

	log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_33)
					+ ConstantesXADES.ESPACIO + nodoAFirmarId);
		throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_51));
	}

	// Se indique el signatureValue o el signature se obtiene el mismo padre
	if (ConstantesXADES.SIGNATURE_VALUE.equals(nodoAFirmar.getLocalName())) {
		idSignatureValue = nodoAFirmarId;
		nodePadreNodoFirmar = nodoAFirmar.getParentNode();
	} else if (ConstantesXADES.SIGNATURE.equals(nodoAFirmar.getLocalName())) {
		nodePadreNodoFirmar = nodoAFirmar;
	} else {

	log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_33)
					+ ConstantesXADES.ESPACIO + nodoAFirmarId);
		throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_51));
	}
	} else {
		// Busca la última firma
		NodeList list =
doc.getElementsByTagNameNS(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE);
		if (list.getLength() < 1) {

	log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_33)
```

```
                                + ConstantesXADES.ESPACIO + nodoAFirmarId);

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_51));

                } else {

                        nodePadreNodoFirmar = list.item(list.getLength() - 1);

                }

        }

        String idSignatureValue = null;

        Element padreNodoFirmar = null;

        if ((nodePadreNodoFirmar != null) && (nodePadreNodoFirmar.getNodeType() ==
Node.ELEMENT_NODE)) {

                padreNodoFirmar = (Element)nodePadreNodoFirmar;

        ArrayList<Element> listElements =
UtilidadTratarNodo.obtenerNodos(padreNodoFirmar, 2, new
NombreNodo(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE_VALUE));

        if (listElements.size() != 1) {

                // TODO: indicar un error específico (No se puede tener más de un nodo
SignatureValue por firma XmlDSig)


        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_33)

                                + ConstantesXADES.ESPACIO + nodoAFirmarId);

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_51));

        }

        idSignatureValue = listElements.get(0).getAttribute(ConstantesXADES.ID);

        // TODO: Si este nodo no tiene id, identificarlo vía XPATH

        if (idSignatureValue == null) {

                // TODO: indicar un error específico (No se puede identificar nodo
SignatureValue en firma XmlDSig)


        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_33)
```

```
                                + ConstantesXADES.ESPACIO + nodoAFirmarId);

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_51));

        }

        }


        // Se busca si existe el path hasta el nodo raíz CounterSignature. Si no existe, se
crea.

        ArrayList<Element> listElements =
UtilidadTratarNodo.obtenerNodos(padreNodoFirmar, 2,
ConstantesXADES.QUALIFYING_PROPERTIES);

        if (listElements.size() != 1) {

                // TODO: indicar un error específico (No se puede tener más de un nodo
Qualifying por firma XAdES


        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_33)

                                + ConstantesXADES.ESPACIO + nodoAFirmarId);

                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_51));

        }

        String esquemaOrigen = listElements.get(0).getNamespaceURI();

        NodeList nodosUnsigSigProp =
(padreNodoFirmar).getElementsByTagNameNS(esquemaOrigen,

                        ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);


        Element nodoRaiz = null;

        if (nodosUnsigSigProp != null && nodosUnsigSigProp.getLength() != 0)

                nodoRaiz = (Element)nodosUnsigSigProp.item(0); // Se toma el primero de
la lista

        else { // Se busca el nodo QualifyingProperties

                NodeList nodosQualifying =
(padreNodoFirmar).getElementsByTagNameNS(esquemaOrigen,
ConstantesXADES.QUALIFYING_PROPERTIES);
```

```
        if (nodosQualifying != null && nodosQualifying.getLength() != 0) {
                Element nodoQualifying = (Element)nodosQualifying.item(0);
                Element unsignedProperties = null;
                if (nodoQualifying.getPrefix() != null) {
                        unsignedProperties =
                                doc.createElementNS(esquemaOrigen,
nodoQualifying.getPrefix() +
                                                ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.UNSIGNED_PROPERTIES);
                        nodoRaiz = doc.createElementNS(esquemaOrigen,
nodoQualifying.getPrefix() +
                                        ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);
                } else {
                        unsignedProperties =
                                doc.createElementNS(esquemaOrigen,
ConstantesXADES.UNSIGNED_PROPERTIES);
                        nodoRaiz = doc.createElementNS(esquemaOrigen,
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);
                }

                unsignedProperties.appendChild(nodoRaiz);
                nodosQualifying.item(0).appendChild(unsignedProperties);
        } else
                throw new
AddXadesException(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_E
RROR_52));
        }

        // Se genera un nuevo nodo Countersignature donde irá la firma
        Element counterSignature = null;
                if (nodoRaiz.getPrefix() != null) {
```

```
                counterSignature = doc.createElementNS(esquemaOrigen,
nodoRaiz.getPrefix() +
                        ConstantesXADES.DOS_PUNTOS +
ConstantesXADES.COUNTER_SIGNATURE);
            } else {
                counterSignature = doc.createElementNS(esquemaOrigen,
ConstantesXADES.COUNTER_SIGNATURE);
            }
            nodoRaiz.appendChild(counterSignature);

            // Se toman la variable de esquema de DataToSign
    XAdESSchemas esquemaTemp = xml.getEsquema();
    if (esquemaTemp != null) {
        xadesSchema = esquemaTemp.getSchemaUri();
    } else {
        xadesSchema = XAdESSchemas.XAdES_132.getSchemaUri();
    }

        // Se escribe una Id única
        Attr counterSignatureAttrib = doc.createAttributeNS(null, ConstantesXADES.ID);
        String counterSignatureId = UtilidadTratarNodo.newID(doc,
ConstantesXADES.COUNTER_SIGNATURE + ConstantesXADES.GUION);
        counterSignatureAttrib.setValue(counterSignatureId);
        counterSignature.getAttributes().setNamedItem(counterSignatureAttrib);

        // Se reemplaza el documento original por el documento preparado para
contrafirma
        xml.setDocument(doc);

    // Se incluye la referencia a la contrafirma
    AbstractObjectToSign obj = null;
            if (XAdESSchemas.XAdES_132.getSchemaUri().equals(xadesSchema)) {
                obj = new SignObjectToSign(idSignatureValue);
```

```
                } else {
                        obj = new InternObjectToSign(idSignatureValue);
                }

    xml.addObject(new ObjectToSign(obj, null, null, null, null));


    // Se firma el documento generado, indicando el nodo padre y el identificador del
nodo a firmar
    xml.setParentSignNode(counterSignatureId);
        Object[] res = signFile(certificadoFirma, xml, pk, provider);

        // Se elimina el identificador del nodo CounterSignature
        counterSignature = UtilidadTratarNodo.getElementById(doc, counterSignatureId);
        counterSignature.removeAttribute(ConstantesXADES.ID);

        return res;
  }

  /**
   * <p>Comprueba la fecha de firma con respecto a la fecha actual del sistema.</p>
   * @param padre padre Nodo raíz del cual pende el nodo buscado.
   * @param idNode Id del nodo desde el cual se extraerá el nodo SigningTime.
   * @return 0 en caso de que sean iguales o en caso de error.
   *       < 0 en caso de que la fecha de firma sea anterior.
   *       > 0 en caso de una firma realizada en el furuto.
   */
  public long getMilisDiffSigningTime(Element padre, String idNode) {
        try {
                return getMilisDiffSigningTime(UtilidadTratarNodo.getElementById(padre,
idNode));
        } catch (Exception e) {
```

```
                log.warn("No se pudo obtener la fecha de la firma: " + e.getMessage(), e);
                return 0;
        }
    }

    /**
     * <p>Comprueba la fecha de firma con respecto a la fecha actual del sistema.</p>
     * @param node Nodo padre desde el cual se extraerá el nodo SigningTime
     * @return 0 en caso de que sean iguales.
     *       < 0 en caso de que la fecha de firma sea anterior.
     *       > 0 en caso de una firma realizada en el furuto.
     */
    public long getMilisDiffSigningTime(Element node) {
        if (node == null) {
                log.debug("No se recibió ningun parámetro");
                return 0;
        }

        Date fechaFirma = null;
        try {
                ArrayList<Element> nodos = UtilidadTratarNodo.obtenerNodos(node, 5,
"SigningTime");
                if (nodos != null && nodos.size() == 1) {
                        String fecha = nodos.get(0).getFirstChild().getNodeValue();
                        if (fecha != null) {
                                fechaFirma = UtilidadFechas.parseaFechaXML(fecha);
                        } else {
                                log.warn("No se pudo obtener la fecha del nodo");
                        return 0;
                        }
                }
```

```
                long now = System.currentTimeMillis();


        return now - fechaFirma.getTime();
        } catch (Exception e) {
                log.warn("No se pudo obtener la fecha de la firma: " + e.getMessage(), e);
                return 0;
        }
  }


  /**
   *  Sube el nivel XAdES de un InputStream de firma y lo guarda en la dirección indicada
   *
        *  @param InputStream Stream que contiene la firma
   *  @param EnumFormatoFirma nivelDeseado de firma deseado
   *  @param String path Ruta bajo la que se guarda el fichero generado
   *  @param String nombreArchivo Nombre bajo el que se guarda el fichero generado
   *  @param id Id de la firma a subir de nivel
   *  @param ocspUrl URL del servidor OCSP
   *  @param tsaUrl URL del servidor de la Autoridad de sellado de tiempo
   *  @param trusterId Clave del validador de confianza a utilizar
   */
  public boolean raiseLevel(InputStream firma, EnumFormatoFirma nivelDeseado, String
path, String nombreArchivo,
                String id, String ocspUrl, String tsaUrl, String trusterId) throws Exception {


        if (firma == null || nivelDeseado == null || path == null || nombreArchivo == null
                        || id == null || ocspUrl == null || tsaUrl == null) {
                throw new
Exception(i18n.getLocalMessage(ConstantsXAdES.I18N_SIGN_11));
        }


                DataToSign data2Sign = new DataToSign();
```

```
                // Nivel de firma deseado
                data2Sign.setXadesFormat(nivelDeseado);
                data2Sign.setXAdESXType(XADES_X_TYPES.TYPE_1);

    // Condiciones de entorno
                data2Sign.setBaseURI(path);
                data2Sign.setXMLEncoding("UTF-8");
                data2Sign.setEsquema(XAdESSchemas.XAdES_132);

        data2Sign.setAlgDigestXmlDSig(UtilidadFirmaElectronica.DIGEST_ALG_SHA256);

    // Documento que contiene la firma a subir de nivel
                DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    dbf.setNamespaceAware(true);
    Document docToRaise = null;
    try {
        DocumentBuilder db = dbf.newDocumentBuilder();
        db.setErrorHandler(new IgnoreAllErrorHandler());
        if (firma != null) {
                InputSource isour = new InputSource(firma);
                isour.setEncoding("UTF-8");
                docToRaise = db.parse(isour);
        }
    } catch (IOException e) {
        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_50),
e);
    } catch (ParserConfigurationException e) {
        throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_50),
e);
                } catch (SAXException e) {
```

```
                    throw new
Exception(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_50),
e);
                }


    data2Sign.setDocument(docToRaise);


    // Generador de sellos de tiempo
    data2Sign.setTimeStampGenerator(new HTTPTimeStampGenerator(tsaUrl,
TSPAlgoritmos.SHA1));


     //Validador de confianza de certificados
    TrustAbstract truster = TrustFactory.getInstance().getTruster(trusterId);
    if (truster == null) {
       System.out.println("No se ha encontrado el validador de confianza");
    }
    data2Sign.setCertStatusManager(new OCSPLiveConsultant(ocspUrl, truster));


    // Handler de ficheros para firmas XAdES-C
    data2Sign.setElementsStorer(new LocalFileStoreElements());
                data2Sign.setXadesFormat(nivelDeseado);


    Document doc = raiseLevel(data2Sign, id);


    // Se salva el fichero generado
          File fichero = null;
          FileOutputStream f = null;
    try
    {
        fichero = new File(path + nombreArchivo);
        f = new FileOutputStream(fichero);
                UtilidadFicheros.writeXML(doc, f);
```

```
    }
    catch (Throwable t)
    {
        if (t.getMessage() != null &&
t.getMessage().startsWith(ConstantesXADES.JAVA_HEAP_SPACE))
                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3)
);
        else
                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4)
);
    } finally {
        try {
                if (f != null)
                        f.close();
                        } catch (IOException e) { // No ocurre
                                log.error(e.getMessage(), e);
                        }
    }

    return true;
  }

  /**
   *  <p>Sube el nivel XAdES de la firma indicada.</p>
   *
   *      @param signData Objeto con la información de la firma
   *  @param signatureID Id de la firma a subir de nivel. Si es vacío se contrafirma la
última firma.
   *  @return Documento de firma generado
   */
```

```
  public Document raiseLevel(DataToSign signData, String signatureID) throws
ClienteError {


    ArrayList<RespYCerts> respuestas = new ArrayList<RespYCerts>();

    ArrayList<X509Certificate> certificadosConOCSP = new ArrayList<X509Certificate>();


        // Se validan los parámetros de entrada

        if (signatureID == null || signData == null) {

                        // No se proporcionaron los datos de firma

                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3
1));

                }


        // Se recupera el Doc con la firma

        Document doc = signData.getDocument();


        // Se recupera el nivel deseado

        EnumFormatoFirma nivelDeseado = signData.getXadesFormat();


                // Se obtiene la firma indicada

                NodeList listaFirmas =
doc.getElementsByTagNameNS(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.LIBRERIAXADES_SIGNATURE);

                int listaFirmasLength = listaFirmas.getLength();

                if (listaFirmasLength < 1)        {


        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_VALIDARFIRMA_E
RROR2));

                        // Error en la validación. No se pudo encontrar el nodo de firma

                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_VALIDARFIRMA_ERRO
R2));

                }
```

```
		Element signature=null;
		if ( signatureID.equals("")) {
			// Busca la última firma
			signature = (Element)listaFirmas.item(listaFirmasLength - 1);
		} else {
			signature = UtilidadTratarNodo.getElementById(doc, signatureID);
		}
		// Si no existe ninguna firma con el Id indicado
		if (signature == null) {
			// No se puede subir el nivel de firma. Firmas presentes en el
documento:

	log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_49) +
					ConstantesXADES.ESPACIO +  listaFirmasLength);
			throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_VALIDARFIRMA_ERRO
R2));
	}

		if (!(new NombreNodo(ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE).equals(
				new NombreNodo(signature.getNamespaceURI(),
signature.getLocalName())))) {
			// No se encuentra el nodo Signature
			throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_3
3) +
					ConstantesXADES.ESPACIO +
ConstantesXADES.SIGNATURE);
		}

		// Se obtiene el certificado firmante
	Element certificateNode = (Element) signature.getElementsByTagNameNS(
			ConstantesXADES.SCHEMA_DSIG, "X509Certificate").item(0);
```

```
	byte[] certificateContent =
Base64Coder.decode(certificateNode.getFirstChild().getNodeValue());

	X509Certificate certificate = null;

		try {

			certificate = (X509Certificate)
CertificateFactory.getInstance("X.509")

				.generateCertificate(new
ByteArrayInputStream(certificateContent));

		} catch (CertificateException e) {

			log.error(e.getMessage(), e);

			throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_6)
, e);

		}


	  // Validamos el documento XADES. Como mínimo será XAdES-BES.

		ResultadoValidacion resultado = null;

	try {

		ValidarFirmaXML validacion = new ValidarFirmaXML();

		resultado = validacion.validar((Node)signature,


	signature.getBaseURI(),

                                                                    null);


		xadesSchema = resultado.getDatosFirma().getEsquema().getSchemaUri();

	} catch (Exception e) {

		throw new ClienteError(e.getMessage(), e);

	}


	if (resultado != null && resultado.getDatosFirma() != null) {

		if ((resultado.isValidate() ||
resultado.getDatosFirma().getDatosNodosNoSignFirmados().size() > 0))

		{ // Se continúa sólo si el documento de firma es válido o es firma
dettached
```

```
                    // Obtenemos los niveles a subir
                    ArrayList<EnumFormatoFirma> mejora = new
ArrayList<EnumFormatoFirma>(); // Niveles a subir
                    EnumFormatoFirma nivel = resultado.getEnumNivel(); // Nivel
actual de la firma, del cual se parte
                    if (nivel == null &&
resultado.getDatosFirma().getDatosNodosNoSignFirmados().size() > 0) {
                        nivel = EnumFormatoFirma.XAdES_BES;
                    }
                    while (true) {
                        if ((nivelDeseado).equals(nivel)) { // Si la firma ya tiene el
nivel deseado
                            break;
                        } else {
                            if (EnumFormatoFirma.XAdES_BES.equals(nivel)) {
                                nivel = EnumFormatoFirma.XAdES_T;
                            } else if (EnumFormatoFirma.XAdES_T.equals(nivel))
{
                                nivel = EnumFormatoFirma.XAdES_C;
                            } else if (EnumFormatoFirma.XAdES_C.equals(nivel))
{
                                nivel = EnumFormatoFirma.XAdES_X;
                            } else if (EnumFormatoFirma.XAdES_X.equals(nivel))
{
                                nivel = EnumFormatoFirma.XAdES_XL;
                            } else {
                                break;
                            }
                            mejora.add(nivel);
                        }
                    }
```

```
                    // Se define un booleano para indicar si es preciso salvar las
respuestas OCSP
                    // o si por el contrario las respuestas se incluirán en la firma
(XAdES-XL)
                    boolean isXadesXL =
mejora.contains(EnumFormatoFirma.XAdES_XL);


                    // Se incluyen los niveles de firma correspondientes. Están
ordenados de menor a mayor
                    Iterator<EnumFormatoFirma> niveles = mejora.iterator();
                    boolean hasNext = niveles.hasNext();
                    byte[] selloTiempo = null;
                    while(hasNext) {
                        nivel = niveles.next(); // Nivel actual a incluir
                        hasNext = niveles.hasNext();


                        if ((EnumFormatoFirma.XAdES_T).equals(nivel)) {
                            try {
                                ITimeStampGenerator timeStampGenerator
= signData.getTimeStampGenerator();
                                if (timeStampGenerator == null) {
                                    throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_6)
);
                                } else {
                                    // Se añaden los elementos propios
de la firma XADES-T
                                    try {
                                        byte[] byteSignature =
UtilidadTratarNodo.obtenerByteNodo(signature, ConstantesXADES.SCHEMA_DSIG,
ConstantesXADES.SIGNATURE_VALUE, CanonicalizationEnum.C14N_OMIT_COMMENTS,
5);
                                        selloTiempo =
timeStampGenerator.generateTimeStamp(byteSignature);

                    addXadesT((Element)signature, signatureID, selloTiempo);
```

```
                                        } catch (FirmaXMLError e) {
                                                log.error(e.getMessage(), e);
                                                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_6)
, e);
                                        } catch (TimeStampException e) {
                                                log.error(e.getMessage(), e);
                                                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_6)
, e);
                                        }
                                }
                        } catch (AddXadesException e) {
                                log.error(e.getMessage(), e);
                                throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_7)
+ e.getMessage(), e) ;
                        }
                } else if((EnumFormatoFirma.XAdES_C).equals(nivel)) {
                        try {

        log.info(i18n.getLocalMessage(ConstantsXAdES.I18N_VALIDATE_18));

        convertICertStatus2RespYCerts(signData.getCertStatusManager().getCertChainSt
atus(certificate), certificadosConOCSP, respuestas);

                                if (ADD_VALIDATION_OCSP) {
                                        if (log.isDebugEnabled()) {
                                                log.debug("Se
incluyen las referencias OCSP de la propia VA");
                                        }

                                        //X509Certificate ocspCert = null;
                                        try {
```

```
                                                    // Se extrrae el certificado
emisor de la respuesta OCSP
                                                    IOCSPCertStatus respOcsp =
(IOCSPCertStatus) respuestas.get(0).getCertstatus();
                                                    OCSPResp resp = new
OCSPResp(respOcsp.getEncoded());
                                                    BasicOCSPResp
respuestaBasica = (BasicOCSPResp)resp.getResponseObject();

        convertICertStatus2RespYCerts(getOCSPfromOCSP(respuestaBasica,
certificadosConOCSP, null, signData), certificadosConOCSP, respuestas);
                                                } catch (CertStatusException ec) {
                                                  log.error(ec);
                                                  throw new
ClienteChainNotFoundError(ec, "OCSP");
                                                } catch (Exception e1) {
                                                    log.error(e1);
                throw new ClienteError(e1);
                                                }

                                                if (log.isDebugEnabled()) {
                                                        log.debug("Se
incluyen las referencias OCSP del sello de tiempo");
                                                    }
                                                TimeStampToken tst = null;
                                                try {

                                                  if (selloTiempo == null) {
                                                    ArrayList<Element>
nodosSignatureTimeStamp = UtilidadTratarNodo.obtenerNodos(signature, 6,
                                                new
NombreNodo(signData.getEsquema().getSchemaUri(),
ConstantesXADES.LIBRERIAXADES_SIGNATURETIMESTAMP));
                                                    Element nodoSigTimeStamp =
(Element)nodosSignatureTimeStamp.get(0);
```

NodeList nodesEncapsulatedTimeStamp = nodoSigTimeStamp.getElementsByTagNameNS(signData.getEsquema().getSchemaUri(), ConstantesXADES.LIBRERIAXADES_ENCTIMESTAMP);

Element encapsulatedTimeStampElement = (Element)nodesEncapsulatedTimeStamp.item(0);

String encapsulatedTS = encapsulatedTimeStampElement.getFirstChild().getNodeValue() ;

selloTiempo = Base64.decode(encapsulatedTS) ;

}

tst = new TimeStampToken(new CMSSignedData(selloTiempo));

} catch (CMSException e) {

// Intenta obtenerlo como org.bouncycastle.tsp.TimeStampResponse

try {

TimeStampResponse tsr = new TimeStampResponse(selloTiempo);

tst = tsr.getTimeStampToken();

} catch (Exception ex) {

log.error(ex);

throw new ClienteError(ex);

}

} catch (Exception e) {

log.error(e);

throw new ClienteError(e);

}

X509Certificate certTSA = null;

try {

```
                                                    CertStore cs =
tst.getCertificatesAndCRLs("Collection", null);
                                                    Collection<? extends
Certificate> certs = cs.getCertificates(null);
                                                    if (certs != null && certs.size()
> 0) {
                                                        if
(log.isDebugEnabled()) {
                                                            log.debug("Se
regenera la cadena de certificados firmante del sello de tiempo y se lanza su validación");
                                                        }
                                                        // Se regenera el
CertPath para asegurar el orden correcto
                                                        try {
        Iterable<X509Certificate> iterableCerts = null;
                                                            if (certs
instanceof Iterable<?>) {
        iterableCerts = (Iterable<X509Certificate>) certs;
                                                            } else {
                                                                throw
new Exception("El certificado no es del tipo esperado: " + certs.getClass());
                                                            }
        CertPath cpTsa = UtilidadCertificados.orderCertPath(iterableCerts);
                                                            certTSA =
(X509Certificate)cpTsa.getCertificates().get(0);
                                                        } catch (Exception e) {
                                                            // si el token
no indica el nombre del firmante, intenta extraerlo por el certificado
                                                            Certificate cert
= certs.iterator().next();
                                                            if (cert
instanceof X509Certificate) {
        certTSA = (X509Certificate) cert;
```

```
                                                                }
                                                            }

                                                        } else {
                                                            log.error("No se pudo
recuperar el certificado del sello de tiempo");
                                                            throw new
ClienteError("No se pudo recuperar el certificado del sello de tiempo");
                                                        }
                                                    } catch (Exception e) {
                                                        log.error(e);
                    throw new ClienteError(e);
                                                    }

                                                    if (certTSA != null) {
                                                      try {
                                                        if (log.isDebugEnabled()) {

        log.debug("Certificado de TSA obtenido " + certTSA.getSubjectX500Principal());
                                                        }
                                                        ArrayList<RespYCerts>
respuestasTSA = new ArrayList<RespYCerts>();
                                                        // Si los certificados emisores
ya han sido validados, se valida sólo el certificado final
                        if (certificadosConOCSP.contains(certTSA.getIssuerX500Principal())) {
                                                            ICertStatus respTSA =
signData.getCertStatusManager().getCertStatus(certTSA);

        ArrayList<ICertStatus> re = new ArrayList<ICertStatus>(1);
                                                            re.add(respTSA);

        convertICertStatus2RespYCerts(re, certificadosConOCSP, respuestasTSA);
                                                        } else {
```

```
convertICertStatus2RespYCerts(signData.getCertStatusManager().getCertChainStatus(certTSA), certificadosConOCSP, respuestasTSA);
                                                                    }


        respuestas.addAll(respuestasTSA);


                                                                    if (log.isDebugEnabled()) {
                                                                            log.debug("TSA Validada. Se valida la VA del propio sello");
                                                                    }
                                                                    try {
                                                                      if(respuestasTSA.size()>0) {
                                                                            IOCSPCertStatus respOcspTsa = (IOCSPCertStatus) respuestasTSA.get(0).getCertstatus();
                                                                            OCSPResp resp = new OCSPResp(respOcspTsa.getEncoded());
                                                                            BasicOCSPResp respuestaBasica = (BasicOCSPResp)resp.getResponseObject();


        convertICertStatus2RespYCerts(getOCSPfromOCSP(respuestaBasica, certificadosConOCSP, null, signData), certificadosConOCSP, respuestas);
                                                                      } else {
                                log.error("No se ha podido obtener información de revocación de la cadena del sello de tiempo.");
                                throw new ClienteError("No se ha podido obtener información de revocación de la cadena del sello de tiempo.");

                                                                      }
                                } catch (CertStatusException ec) {
                                  log.error(ec);
                                  throw new ClienteChainNotFoundError(ec, "OCSP");
                                                                    } catch (Exception e1) {
```

```
                                                                log.error(e1);
                    throw new ClienteError(e1);
                                                                }
                } catch (CertStatusException e) {
                    log.error(e.getMessage(), e);
                    throw new
ClienteChainNotFoundError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMA
XML_ERROR_10), e, "TSA") ;
                }
            }
                                        }
                                    } catch (CertStatusException e) {
                                        log.error(e.getMessage(), e);
                                        throw new
ClienteChainNotFoundError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMA
XML_ERROR_10), e, "FIRMANTE") ;
                                    }
                                    // Se añaden los elementos propios de la firma
XADES-C tomando las
                                    // variables de configuración de DataToSign
                                    String algDigestXML =
(signData.getAlgDigestXmlDSig() != null) ? signData.getAlgDigestXmlDSig() :
UtilidadFirmaElectronica.DIGEST_ALG_SHA1;

                                    try {
                                        addXadesC(signature, respuestas,
XAdESSchemas.getXAdESSchema(xadesSchema), algDigestXML);
                                    } catch (AddXadesException e) {

log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_10) +
e.getMessage(), e);
                                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_1
0) + e.getMessage(), e) ;
                                    }
```

```
                                        // Si se firma XAdES-C ó XAdES-X exclusivamente, se
guardan los ficheros adjuntos
                                        // TODO: solucionar nombre de los ficheros OCSP

                                        if (!isXadesXL) {
                                                try {
                                                        doc = addURIXadesC(signature,
saveOCSPFiles(respuestas, signData.getElementsStorer()), signData.getBaseURI());
                                                } catch (FirmaXMLError ex) {
                                                  log.error("Error al guardar ficheros de
estados de certificados", ex);
                                                        throw new ClienteError("Error al
guardar ficheros de estados de certificados", ex);
                                                }
                                        }
                                } else if((EnumFormatoFirma.XAdES_X).equals(nivel)) {

                                        // A partir del nodo raíz de la firma se obtiene el nodo
UnsignedSignatureProperties
                                        Element unsignedSignaturePropertiesElement =
null;
                                        NodeList unsignedSignaturePropertiesNodes =

        signature.getElementsByTagNameNS(xadesSchema,
ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES);

                                        if (unsignedSignaturePropertiesNodes.getLength() <
1) {
                                                // El nodo UnsignedSignatureProperties no
existe o no es único

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_36) + ConstantesXADES.ESPACIO +

        ConstantesXADES.UNSIGNED_SIGNATURE_PROPERTIES +
        ConstantesXADES.ESPACIO +
```

```
I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_37) +
ConstantesXADES.ESPACIO +

unsignedSignaturePropertiesNodes.getLength());
                                        // El sistema no soporta nodos
UnsignedSignatureProperties múltiples
                                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_4
1));
                                    } else
                                        unsignedSignaturePropertiesElement =
(Element) unsignedSignaturePropertiesNodes.item(0);

                                    // Se añaden los elementos propios de la firma
XADES-X
                                    try {

addXadesX(unsignedSignaturePropertiesElement,
signData.getTimeStampGenerator());
                                    } catch (AddXadesException e) {
                                        log.error(e.getMessage(), e);
                                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_1
2) + e.getMessage()) ;
                                    }

                                } else if((EnumFormatoFirma.XAdES_XL).equals(nivel)) {
                                    try {
                                        addXadesXL(signature, respuestas,
XAdESSchemas.getXAdESSchema(xadesSchema));
                                    } catch (Exception e) {
                                        log.error(e.getMessage(), e);
                                        throw new
ClienteError(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERROR_1
2) + e.getMessage()) ;
```

```
                                        }
                                    }
                                } // fin del bucle
                            } // Fin del if isValidate
                        }

                        return doc;
                    }

                    /**
                     * Realiza la comprobación OCSP de un certificado de un OCSP de forma recursiva si es necesario
                     * @param respuestaBasica
                     * @param certificadosConOCSP
                     * @param certificadosAnteriores
                     * @param signData
                     * @return
                     * @throws CertStatusException
                     * @throws NoSuchProviderException
                     * @throws OCSPException
                     */
                    private List<ICertStatus> getOCSPfromOCSP(BasicOCSPResp respuestaBasica, List<X509Certificate> certificadosConOCSP, List<X500Principal> certificadosAnteriores, DataToSign signData) throws CertStatusException, NoSuchProviderException, OCSPException {
                        if (log.isDebugEnabled()) {
                            ResponderID respID = respuestaBasica.getResponderId().toASN1Object();
                            log.debug("Extracción del certificado OCSP: " + ASN1Utils.getResponderID(respID).toString());
                        }
                        if(certificadosAnteriores == null) {
                            certificadosAnteriores = new ArrayList<X500Principal>();
                            for(X509Certificate currentCert : certificadosConOCSP) {
```

```
            certificadosAnteriores.add(currentCert.getSubjectX500Principal());
          }
        }
        List<ICertStatus> re = new ArrayList<ICertStatus>();

        X509Certificate[] ocspCerts = respuestaBasica.getCerts(ConstantesXADES.SUN);
        if (ocspCerts != null && ocspCerts.length > 0) {
          if (log.isDebugEnabled()) {
            log.debug("Se regenera la cadena y se lanza la validación");
          }
          CertPath cpOcsp = UtilidadCertificados.orderCertPath(Arrays.asList(ocspCerts));
          X509Certificate ocspCert = (X509Certificate)cpOcsp.getCertificates().get(0);

          if(!certificadosConOCSP.contains(ocspCert) &&
!certificadosAnteriores.contains(ocspCert.getSubjectX500Principal())) {
            if(!certificadosAnteriores.contains(ocspCert.getIssuerX500Principal())) {
              re = signData.getCertStatusManager().getCertChainStatus(ocspCert);
            } else {
              ICertStatus respOCSP =
signData.getCertStatusManager().getCertStatus(ocspCert);
              re = new ArrayList<ICertStatus>(1);
              re.add(respOCSP);

            }

            if (re == null || re.size() == 0 || re.get(0) == null) {
              log.error("No se ha podido obtener respuestas de validación para la cadena de
certificación de: " + ocspCert.getSubjectX500Principal() + ". La cadena tiene: " +
(cpOcsp.getCertificates()!=null?cpOcsp.getCertificates().size():0));
              throw new CertStatusException("No se ha podido obtener respuestas de
validación para la cadena de certificación de: " + ocspCert.getSubjectX500Principal() + ".
La cadena tiene: " + (cpOcsp.getCertificates()!=null?cpOcsp.getCertificates().size():0));
            }
```

```
            certificadosAnteriores.add(ocspCert.getSubjectX500Principal());

            X509Certificate newOCSPCert = re.get(0).getCertificate();

            if(!certificadosConOCSP.contains(newOCSPCert) &&
!certificadosAnteriores.contains(newOCSPCert.getSubjectX500Principal())){

               re.addAll(getOCSPfromOCSP(respuestaBasica, certificadosConOCSP,
certificadosAnteriores, signData));

            }


         }


      } else {

         log.error("No se pudo recuperar el certificado de la VA del OCSP");

         throw new CertStatusException("No se pudo recuperar el certificado de la VA del
OCSP");

      }

      return re;


   }

   /**

    * Este método se encarga de insertar las URIs de XADES-C en la firma

    *

    * @param firma, Documento con la firma xml

    * @param listaArchivos, Lista de nombres de la respuestaOCSP y el path de
certificación

    * @param baseUri

    * @return Document doc, Documento firmado con las nuevas URI´s

    */

   public Document addURIXadesC(Element firma, ArrayList<NombreElementos>
listaArchivos, String baseUri) throws FirmaXMLError

   {


            Document doc = firma.getOwnerDocument();
```

```
        NodeList completeCertificateRefs = null;
        NodeList completeRevocationRefs = null;

        // TODO: se estan buscando por todo el documento
        completeCertificateRefs = firma.getElementsByTagNameNS(xadesSchema,
ConstantesXADES.COMPLETE_CERTIFICATE_REFS);
        completeRevocationRefs = firma.getElementsByTagNameNS(xadesSchema,
ConstantesXADES.COMPLETE_REVOCATION_REFS);

        if (completeCertificateRefs.getLength() == 0 ||
completeRevocationRefs.getLength() == 0) {

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_29));
                return doc;
        }

    String tipoUri = null;
    if (ConstantesXADES.SCHEMA_XADES_111.equals(xadesSchema))
                tipoUri = ConstantesXADES.URI_MINUS;
        else
                tipoUri = ConstantesXADES.URI_MAYUS;

    // TODO: recodificar relacionando correctamente nombres de ficheros con CRL/OCSP
ref relacionado

    // A continuación se sacan las referencias OCSP del nodo OCSPRefs
    NodeList ocspRefs = null;
    NodeList crlRefs = null;
    try {
                ArrayList<Element> listOcspRefs =
UtilidadTratarNodo.obtenerNodos((Element)completeRevocationRefs.item(0), null, new
NombreNodo(xadesSchema, ConstantesXADES.OCSP_REFS));
```

```
                    ArrayList<Element> listCRLRefs =
UtilidadTratarNodo.obtenerNodos((Element)completeRevocationRefs.item(0), null, new
NombreNodo(xadesSchema, ConstantesXADES.CRL_REFS));
                    if (listOcspRefs.size() > 1) {
                    throw new FirmaXMLError("hay demasiados elementos ocsprefs");
                    }
                    if (listCRLRefs.size() > 1) {
                    throw new FirmaXMLError("hay demasiados elementos crlrefs");
                    }
                    if (listOcspRefs.size() > 0)
                            ocspRefs = listOcspRefs.get(0).getChildNodes();
                    if (listCRLRefs.size() > 0)
                            crlRefs = listCRLRefs.get(0).getChildNodes();
            } catch (FirmaXMLError ex) {
            throw new FirmaXMLError("error obteniendo elementos ocsprefs y crlrefs");
            }

        // Si ha encontrado el nodo OCSPRefs, se pasa a capturar su contenido
            Iterator<NombreElementos> it = listaArchivos.iterator();
            int indexOCSP = 0;
            int indexCRL = 0;
            while (it.hasNext()) {
                    NombreElementos nf = it.next();
                    if (it.hasNext()) {
                            String nameFile = nf.getNameFileCRLResp();
                            Node el;
                            if (nameFile == null) {
                                    nameFile = nf.getNameFileOCSPResp();
                                    if (nameFile == null)
                                            throw new FirmaXMLError("Fichero de status
(OCSP o CRL) sin nombre");
```

```
                                        if ((ocspRefs == null) || (ocspRefs.getLength() <=
indexOCSP))
                                        throw new FirmaXMLError("Fichero de status no
relacionable con crlref");
                                        el =
((Element)ocspRefs.item(indexOCSP++)).getElementsByTagNameNS(xadesSchema,
ConstantesXADES.OCSP_IDENTIFIER).item(0);
                                }
                                else {
                                        if ((crlRefs == null) || (crlRefs.getLength() <=
indexCRL))
                                                throw new FirmaXMLError("Fichero de status
no relacionable con crlref");
                                        el =
((Element)crlRefs.item(indexCRL++)).getElementsByTagNameNS(xadesSchema,
ConstantesXADES.XADES_TAG_CRL_IDENTIFIER).item(0);
                                }
                                Attr uri = doc.createAttributeNS(null, tipoUri);
                                uri.setValue(nameFile);

                                NamedNodeMap nodo = el.getAttributes();
                                nodo.setNamedItem(uri);
                        }
                }

        // A continuación se sacan los Certificados del nodo CertRefs
        Node certRefs = (Node)completeCertificateRefs.item(0).getFirstChild();

        // Si ha encontrado el nodo CertRefs, se pasa a capturar su contenido
        if (certRefs != null)
        {
                // Se saca la lista de certificados
                NodeList certs = certRefs.getChildNodes();
                int l = certs.getLength();
```

```
                if (l!=(listaArchivos.size()-1)) {

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_30));

                }


                for (int i=0; i<l; i++)

                {

                        // Sacamos los nodos Cert uno por uno

                        Node certificado = certs.item(i); // Sacamos cert

                        if (certificado != null) {

                                // incluimos la uri


                                Attr uri = doc.createAttributeNS(null, tipoUri);

                                uri.setValue(listaArchivos.get(i+1).getNameFileX509Cert());
// La posicion 0 es del certificado firmante


                                NamedNodeMap nodoCertificado =
certificado.getAttributes();

                                nodoCertificado.setNamedItem(uri);

                }

        }

    }


    return doc;

  }


  /**

   * Este método se encarga de guardar los archivos OCSP

   * @return un ArrayList con la lista de archivos guardados

   */
```

```
  public ArrayList<NombreElementos> saveOCSPFiles(ArrayList<RespYCerts> respuesta,
IStoreElements storer)

  {

        ArrayList<NombreElementos> listaArchivos = new
ArrayList<NombreElementos>();


        if ((respuesta != null) && (respuesta.size() > 0)) {

                int i = 0;

                Iterator<RespYCerts> it = respuesta.iterator();

                while (it.hasNext()) {

                        RespYCerts respAndCert = it.next();


                        // Datos del certificado

                        X509Certificate certificate = null;

                        String certFile = null;

                        if (i > 0) {

                                certFile = respAndCert.getX509CertFile();

                                if ((certFile == null) || (certFile.trim().length() == 0))

                                        certificate =
respAndCert.getCertstatus().getCertificate();

                        }

                        // Datos del estado del certificado

                        ICertStatus respCert = null;

                        if (i < respuesta.size() - 1) {

                                respCert = respAndCert.getCertstatus();

                        }

                        // Almacena los datos

                        String[] names = storer.storeCertAndStatus(certificate, respCert);


                        NombreElementos nombreElemento = new NombreElementos();

                        if ((certFile != null) && (certFile.trim().length() > 0)) {

                                nombreElemento.setNameFileX509Cert(certFile);
```

```
                        } else {
                                nombreElemento.setNameFileX509Cert(names[0]);
                        }
                        if (respCert instanceof IOCSPCertStatus) {
                                nombreElemento.setNameFileOCSPResp(names[1]);
                        } else if (respCert instanceof IX509CRLCertStatus) {
                                nombreElemento.setNameFileCRLResp(names[1]);
                        }
                        listaArchivos.add(nombreElemento);

                        i++;
                }
        } else {

        log.error(I18n.getResource(ConstantesXADES.LIBRERIAXADES_FIRMAXML_ERRO
R_27));
                return null;
        }

        return listaArchivos;
   }
}
```