Consorci
Administració Oberta
de Catalunya

DI - Signador Centralitzat (SSC)

Generalitat
de Catalunya

LOCALRET

Realitzat per: Consorci AOC – Servei Eines de Signatura

Versió: 1.3.3

Fecha: 30/05/2017

Consorci
Administració Oberta
de Catalunya

Control del document

Informació general

Título:	GuiaIntegracioSSC.v1.3.3.docx
Creat per:	Consorci AOC – Servei Eines de Signatura
Nom del documento:	GuiaIntegracioSSC.v1.3.3.docx

Histórico de revisiones

Versió	Data	Autor	Comentaris
1.3.2	26/10/2016	Servei Eiunes de Signatura	Actualització de contingut.

Consorci
Administració Oberta
de Catalunya

Index

1 Introducció

El Signador Centralitzat és la solució que dóna cobertura al concepte de sistema de signatura electrònica per a l'actuació administrativa automatitzada, tal com es descriu a la LAECSP (art.18, Llei 11/2007). Permet la custòdia i ús posterior de segells electrònics de les administracions públiques, òrgans o entitats de dret públic de Catalunya que ho sol·licitin, agilitzant la posada en marxa d'un sistema automatitzat de signatura electrònica de documents des d'un dispositiu remot. En aquesta URL es pot consultar la informació del servei : https://www.aoc.cat/serveis-aoc/signador-centralitzat/

Les claus associades al segell electrònic, resideixen en magatzems de claus protegits per un dispositiu criptogràfic segur (HSM). El dispositiu està certificat sota els criteris de Common Critera EAL4+, oferint les màximes garanties en matèria de seguretat i custòdia de claus.

La guia bàsica d'integració al Servei de Signatura Centralitzada (SSC) és un document que va dirigit a desenvolupadors que vulguin integrar les seves aplicacions de gestió amb l'SSC de l'AOC.

El lector d'aquest document ha de ser un professional amb coneixements en programació avançada en el llenguatge Java. És molt recomanable disposar també de coneixement sobre Webservices, missatgeria SOAP, WS-Security, ús de certificats i signatures digitals.

L'objectiu d'aquest document és oferir una guia bàsica d'integració amb l'SSC mitjançant tecnologia Java

La disposició de capitols és la següent:

Al capítol 2 s'indiquen les URLs d'accés als Webservices de signatura de l'SSC. Al capítol 3 es descriu l'SmartWrapper, API d'integració a l'SSC. Al capítol 4 es descriu com accedir als serveis de creació de signatura mitjançant SmartWrapper, amb codi d'exemple. També s'indica com parametritzar la signatura visible de documents PDF. Finalment, al capítol 5 es descriu la gestió de dades de gran volum amb l'API SmartWrapper

L'accés al servei es fa mitjançant canal segur SSL amb autenticació de client. Les següent són les URLs per accedir al Servei de Signatura Centralitzada pels entorns de PREPRODUCCIÓ i PRODUCCIÓ.

2 URLS de l'SSC

L'accés al servei es fa mitjançant canal segur SSL amb autenticació de client. Les següent són les URLs per accedir al Servei de Signatura Centralitzada pels entorns de PREPRODUCCIÓ i PRODUCCIÓ.

2.1 PREPRODUCCIÓ

La URL d'accés al Webservice de signatura de l'SSC a l'entorn de PREPRODUCCIÓ mitjançant certificat és:

https://ssc.preproduccio.catcert.cat:8443/trustedx-sgw/SignCert

2.2 PRODUCCIÓ

La URL d'accés al Webservice de signatura de l'SSC a l'entorn de PRODUCCIÓ mitjançant certificat és:

https://ssc.catcert.cat:8443/trustedx-sgw/SignCert

2.3 Connexió

Els protocols utilitzats per realitzar les connexions SSL depenen de la versió de Java amb la qual s'executa l'aplicació. Per defect SmartWrapper utilitza protocols segurs per xifrar la comunicació amb el servidor utilitzant TLS amb la versió 1.1 o 1.2. D'altra banda, les versions de TLS per defecte que s'utilitzen en cada versió de Java varien, tal com mostra la següent taula:

	JDK 8 (March 2014 to present)	JDK 7 (July 2011 to present)	JDK 6 (2006 to end of public updates 2013)
TLS Protocols	TLSv1.2 (default) TLSv1.1 TLSv1 SSLv3	TLSv1.2 TLSv1.1 TLSv1 (default) SSLv3	TLSv1 (default) SSLv3
JSSE Ciphers:	Ciphers in JDK 8	Ciphers in JDK 7	Ciphers in JDK 6
Reference:	JDK 8 JSSE	JDK 7 JSSE	JDK 6 JSSE
Java Cryptography Extension, Unlimited Strength (explained later)	JCE for JDK 8	JCE for JDK 7	JCE for JDK 6

Per tal motiu, per a versions anteriors de Java 1.7.XX cal habilitar l'ús de TLSv1.0 per a realitzar les comunicacions SSL. Hi ha dues formes d'habilitar aquest protocol, mitjançant una variable global al sistema o amb una variable a la màquina virtual de Java (JVM).

2.3.1 Habilitar la propietat de JVM

Abans de fer la connexió amb TrustedX (mètode `send`) cal introduir el següent codi:

```
System.setProperty("PropertyAllowTls10AsClient", "true");
```

Aquesta propietat habilitarà l'ús de TLSv1.0 a la màquina virtual de Java per a l'aplicació de SmartWrapper.

2.3.2 Habilitar la propietat com a variable d'entorn

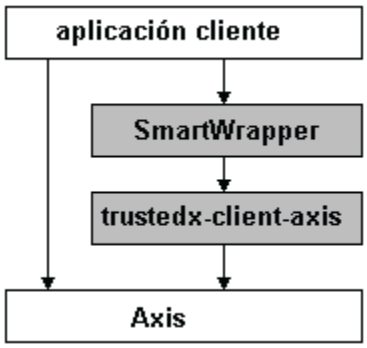
Depenent del sistema operatiu, la configuració de variables d'entorn difereix. Cal afegir la variable d'entorn `PropertyAllowTls10AsClient` amb valor `true`.

3 SmartWrapper

En aquest capítol oferim una versió general de l'API SmartWrapper proporcionada per accedir a l'SSC.

3.1 L'API SmartWrapper

SmartWrapper és una API desenvolupada sobre Axis per crear aplicacions client sense necessitat d'aprofundir en la complexitat d'Axis (permet també l'accés a estructures Axis d'ús avançat). SmartWrapper permet generar aplicacions Java senzilles, i amb poques línies de codi.



3.2 Accés a l'SSC mitjançant SmartWrapper

A continuació resumim els passos necessaris per enviar una petició a l'SSC, obtenir la resposta i consultar si s'ha processat correctament.

Per invocar el servei i avaluar la resposta:

1. Creem una instància de la classe `SmartSignRequest` corresponent al servei invocat.
2. Recuperem l'objecte `Stub` de la classe Axis mitjançant el mètode `getStub`.
3. Utilitzem el mètode `setHeader` per a l'objecte Stub afegint els tags (`Rol` i `Dispositari`).
4. Creem una instància de la classe `Request` corresponent al servei invocat.
5. Realitzem les operacions set necessàries per construir una petició vàlida.

Important: A la petició hem de definir tant el perfil escollit (mitjançant el mètode `setProfile`) com els paràmetres adequats a aquell perfil.

6. Enviem la petició (mitjançant la operació `send`) per obtenir un objecte del tipus Response.
7. Consultem, mitjançant els mètodes de l'objecte `Response`, el resultat de la petició. Si la petició ha sigut processada correctament, obtenim el valor dels paràmetres retornats amb les mètodes get del mateix objecte `Response`.

3.3 Classes de l'API SmartWrapper

En aquest punt descrivim les classes utilitzades per l'API SmartWrapper.

3.3.1 Classe per gestionar peticions SmartWrapper de signatura

La classe de l'API SmartWrapper que serveix per gestionar peticions de signatura, és la següent:

- Serveis de signatura digital: `SmartSignRequest`

Aquest classe inclou els següents mètodes:

- Mètode `send` per enviar la petició. Aquest mètode retorna un objecte de la classe associada amb la resposta corresponent.
- Mètodes per assignar valors als paràmetres necessaris en cada petició. Els noms d'aquests mètodes segueixen la sintaxi `set<Param>`, on `<Param>` és el nom del paràmetre al que assignen valor (amb la primera lletra en majúscules).

3.3.2 Classe per gestionar respostes SmartWrapper de signatura

La classe de l'API SmartWrapper per gestionar respostes de signatura, és la següent:

- Serveis de signatura digital: `SmartSignResponse`

Aquesta classe inclou mètodes per consultar els paràmetres de la resposta. Els noms d'aquests mètodes segueixen la sintaxi `get<Param>`, on `<Param>` és el nom del paràmetre el valor del qual es vol consultar (amb la primera lletra en majúscules).

3.3.3 Classe SmartHeader per gestionar la capçalera

La classe `SmartHeader` gestiona la capçalera de les peticions i inclou informació per la correcta autenticació al servei. En concret, suporta autenticació amb certificat.

Tanmateix, l'autenticació la portarem a terme sempre mitjançant certificat digital per l'aplicació en concret que vol consumir els serveis de l'SSC.

Cal emfatitzar que el servei inclou dos paràmetres nous a la capçalera: *Rol* i *Dispositari*. Ambdós elements son presents al fitxer de propietats.

3.3.4 Classe Constants per gestionar constants

La classe `Constants` defineix una sèrie de constants per facilitar l'ús de l'API i reduir el risc d'error tipogràfics.

3.4 Configuració d'SmartWrapper

3.4.1 JDK

El sistema on s'executin ha de tenir instal•lada la **Java JDK1.4 (o superior).**

3.4.2 Llibreries del client

Les llibreries necessàries pel correcte funcionament del client basat en SmartWrapper són:

- smartwrapper.jar
- trustedx-client-axis-stub.jar
- trustedx-client-axis.jar
- trustedx-provider.jar
- activation.jar
- axis.jar

Durant la compilació, aquest arxiu només és necessari per codificar i descodificar valors en Base64 mitjançant la classe org.apache.axis.encoding.Base64.

- commons-discovery-0.2.jar
- commons-httpclient-3.0.1.jar
- commons-logging-1.0.4.jar
- jaxrpc.jar
- log4j-1.2.8.jar
- saaj.jar
- wsdl4j-1.5.1.jar
- xercesImpl.jar
- xml-apis.jar
- commons-collections-3.2.jar
- commons-configuration.jar
- commons-lang-2.6.jar
- sfly-ssl.jar

3.5 Arxiu smartwrapper.properties

El directori des d'on s'executi el client, haurà d'incloure l'arxiu de configuració **smartwrapper.properties**.
La següent taula defineix els paràmetres inclosos en l'arxiu smartwrapper.properties:

Paràmetre	Valor
Truststore.active	true: per la connexió HTTP es farà servir el magatzem de certificats definit al paràmetre Truststore.path. false (valor por defecte): s'utilitzarà el magatzem de certificats configurat en la màquina virtual de Java (si n'hi ha).
Truststore.path	Magatzem de certificats que s'utilitzarà a la connexió HTTP (si el valor de Truststore.active és true).
Comptabilitzada l'obligació reconeguda	Contrasenya del magatzem de certificats definit a Truststore.path.
Pagada	true: per la connexió HTTPS s'utilitzarà el magatzem de claus definit al paràmetre Keystore.path.

	false (valor por defecte): s'utilitzarà el magatzem de claus configurat en a màquina virtual de Java (si n'hi ha).
Keystore.path	Magatzem de claus utilitzat en la connexió HTTPS (si el valor de keystore.active és true).
Keystore.password	Contrasenya del magatzem de claus definit a Keystore.path.
Keystore.Type	Tipus de keystore: JKS (valor per defecte), JCEKS, o PKCS12
Proxy.active	true: per la connexió HTTP s'utilitzarà un servidor Proxy. false: no s'utilitzarà un servidor proxy.
Proxy.host	Servidor proxy utilitzat (si proxy.active és true).
Proxy.port	Port del servidor proxy.
Proxy.username	Nom d'usuari per accedir al servidor proxy (si requereix autenticació HTTP bàsica).
Proxy.password	Contrasenya per accedir al servidor proxy (si requereix autenticació HTTP bàsica).
Timeout	Temps d'espera de la connexió HTTP (en mil·lisegons).
Request.loadPath	Directori base dels arxius creats per enviar amb les peticions quan es maneguen dades de gran tamany.
Response.savePath	Directori base dels arxius creats en rebre les respostes quan es maneguen dades de gran tamany.
req-log.active	true: las peticions SOAP enviades es guardaran en arxius de log. false (valor por defecte): les peticions SOAP enviades no es guardaran en arxius de log.
req-log.savePath	Directori on es guardaran els arxius de log amb les peticions enviades (si el valor de req-log.active és true).
res-log.active	true: es guardaran arxius de log amb el contingut de les respostes rebudes. false (valor por defecte): no se guardaran arxius de log amb el contingut de les respostes rebudes.
res-log.savePath	Directori on es guardaran els arxius de log amb les respostes rebudes (si el valor de res-log.active és true).
authN.policy	Política d'autenticació sol·licitada (opcional).
Ssl.allowCriticalExts	true: durant la connexió SSL es permeten certificats de servidor amb qualsevol extensió marcada com a crítica. false (valor per defecte): durant la connexió SSL no es permeten certificats de servidor amb extensions crítiques.
Ssl.noValidation	true: s'anul·la la validació del certificat del servidor durant les connexions SSL. false (valor per defecte): es valida el certificat del servidor durant les connexions SSL.
Ssl.validation	Algorisme per validar el certificat del servidor durant connexions SSL. L'algorisme utilitzat ha d'estar implementat per un proveïdor criptogràfic registrat.

4 Serveis de Signatura

En aquest capítol explicarem, fent ús de casos pràctics, com accedir als serveis de creació de signatura mitjançant SmartWrapper.
En tots aquests exemples farem servir l'autenticació de client mitjançant certificat. A tal efecte, caldrà configurar l'arxiu smartwrapper.properties amb les dades del keystore i el truststore. Pels exemples farem servir el keystore i el truststore de proves que distribuïm dins el pack d'integració:

```
# truststore
Truststore.active = true
Truststore.path = resources/truststore/catcert.truststore
Truststore.password = catcert

# keystore
Keystore.active = true
Keystore.path =
resources/keystore/apptest/20100428_111540_CDA_1_cda_c1_policy.p12
Keystore.password = 1234
Keystore.type = PKCS12
```

4.1 Signatura CMS attached

Amb el codi següent enviem una petició SOAP de creació d'una signatura CMS attached (adjunta), i consultem la resposta obtinguda.

```
import com.safelayer.trustedx.client.smartwrapper.*;
import org.apache.axis.encoding.Base64;
import org.apache.axis.message.SOAPHeaderElement;

public class CmsSignature {

	static final String TRUSTEDX_URL =
"https://ssc.preproduccio.catcert.cat:8443/trustedx-sgw/SignCert";
	static final String SIGNER_DN = "CN=Dummy";
	static final String RESULTMAJOR_SUCCESS =
"urn:oasis:names:tc:dss:1.0:resultmajor:Success";
	static final String DIPOSITARI = "O=0000000000";
	static final String ROL  = "TEST";

	public static byte[] signCms(byte[] dataToSign) throws Exception {

		SmartSignRequest sReq = new SmartSignRequest(TRUSTEDX_URL);
		//afegeix la capçalera a la petició
		Stub stub = (Stub) sReq.getStub();
		stub.setHeader(null, "Rol", ROL);
		stub.setHeader(null, "Dipositari", DIPOSITARI);

		//signatura CMS
		sReq.setProfile(Constants.Profile.CMSPKCS7);
		//signatura attached
		sReq.setEnvelopingSignature(true);
		//dades a signar
		sReq.setInputBase64Data(Base64.encode(dataToSign));
		//selecció de la clau de signatura
		sReq.setKeySubjectName(SIGNER_DN);


		//enviament de la petició
		SmartSignResponse sResp = sReq.send();
```

```
            if (RESULTMAJOR_SUCCESS.equals(sResp.getResultMajor())
                  && sResp.getResultMinor() == null) {
                  //recuperació de la signatura
                  String signatureBase64 = sResp.getSignatureBase64();
                  return Base64.decode(signatureBase64);
            } else {
                  throw new Exception("Error signing:" +
sResp.getResultMessage());
            }
      }
      public static void main(String args[]) throws Exception {
      String data = "data to sign...";
      byte[] signature = signCms(data.getBytes());
      }

}
```

Per construir la petició, invoquem els mètodes de la classe `SmartSignRequest` i `SmartHeader` descrits a la taula següent:

La classe `SmartHeader` permet afegir paràmetres al element `Header` de la petició SOAP. Aquesta classe es comporta igual a tots els següents exemples.

Mètode	Paràmetres	Mètode	Paràmetres
getStub	Obté l'objecte Stub de baix nivell Axis.	getStub	Obté l'objecte Stub de baix nivell Axis.
setHeader	Afegeix l'element al node Header.	setHeader	Afegeix l'element al node Header.

D'altra banda, la classe `SmartSignRequest` construeix l'element `Body` de la petició SOAP.

Mètode	Paràmetres
setProfile	Constants.Profile.CMSPKCS7 per seleccionar el tipus de signatura CMS/PKCS#7.
setEnvelopingSignature	true per signatura attached (adjunta).
setInputBase64Data	Dades a signar, codificades en Base64.

Amb els mètodes de la classe `SmartSignResponse` obtenim la resposta a la nostra petició de signatura:

Mètode	Paràmetres
getResultMajor	Indica si s'ha pogut processar la petició, independentment del resultat d'aquesta.
getResultMinor	Detalls de la operació.
getSignatureBase64	Signatura CMS codificada en Base64.

4.2 Signatura XML enveloped

Amb el codi següent enviem una petició SOAP de creació d'una signatura XML enveloped (embolcallada), i consultem la resposta obtinguda.

```
import com.safelayer.trustedx.client.smartwrapper.*;
import org.apache.axis.encoding.Base64;
import org.apache.axis.message.SOAPHeaderElement;

public class XmlEnvelopedSignature {
	static final String TRUSTEDX_URL =
"https://ssc.preproduccio.catcert.cat:8443/trustedx-sgw/SignCert";
	static final String SIGNER_DN = "CN=Dummy";
	static final String RESULTMAJOR_SUCCESS =
"urn:oasis:names:tc:dss:1.0:resultmajor:Success";
	static final String DIPOSITARI = "O=0000000000";
	static final String ROL  = "TEST";

	public static byte[] signEnvelopedXml(byte[] dataToSign) throws Exception {

		SmartSignRequest sReq = new SmartSignRequest(TRUSTEDX_URL);
		//afegeix la capçalera a la petició
		Stub stub = (Stub) sReq.getStub();
		stub.setHeader(null, "Rol", ROL);
		stub.setHeader(null, "Dipositari", DIPOSITARI);

		//signatura XML simple
		sReq.setProfile(Constants.Profile.XADES);
		sReq.setSignatureFormat(Constants.SignatureFormat.NOADES);
		//signatura enveloped
		sReq.setSignaturePlacement(Constants.SignaturePlacement.ENVELOPED);
		//posicionament de la signatura dins el document
		sReq.setXmlEnvelopedXPathFirstChildOf("/");
		//dades a signar
		sReq.setInputXmlBase64(Base64.encode(dataToSign));
		//selecció de la clau de signatura
		sReq.setKeySubjectName(SIGNER_DN);

		//enviament de la petició
		SmartSignResponse sResp = sReq.send();

		if (RESULTMAJOR_SUCCESS.equals(sResp.getResultMajor())
		&& sResp.getResultMinor() == null) {
			//recuperació de la signatura
			String signature = sResp.getDocumentWithSignatureXml();
			return signature.getBytes();
		} else {
			throw new Exception("Error signing:" +
sResp.getResultMessage());
		}
	}

	public static void main(String args[]) throws Exception {
		String data = "<library Id='c123tpe4ryta6di24'>"
		+"<book Id='1'>"
		+" <title>The dark house</title>"
		+"</book>"
		+"<book Id='2'>"
		+" <title>The laundry</title>"
		+"</book>"
		+"</library>";
		byte[] signature = signEnvelopedXml(data.getBytes());
		System.out.println("signature=" + new String(signature));
	}
```

```
}
```

Per construir la petició, invoquem els mètodes de la classe `SmartSignRequest` descrits a la taula següent:

Mètode	Paràmetres
setProfile	Constants.Profile.XADES per seleccionar el perfil de signatura XML (tant per signatures simples com avançades).
setSignatureFormat	Format de la signatura. Pot tenir els següents valors: Signatura simple: Constants.SignatureFormat.NOADES Signatura AdES-BES: Constants.SignatureFormat.BES Signatura AdES-T: Constants.SignatureFormat.ES_T Signatura AdES-C: Constants.SignatureFormat.ES_C Signatura AdES-A: Constants.SignatureFormat.ES_A
setSignaturePlacement	Constants.SignaturePlacement.ENVELOPED per seleccionar el tipus de signatura XML enveloped. Per seleccionar una signatura detached, farem servir el següent paràmetre: Constants.SignaturePlacement.DETACHED I per seleccionar una signatura del tipus enveloping, farem servir el paràmetre: Constants.SignaturePlacement.ENVELOPING
setXmlEnvelopedXPathFirstChildOf	Expressió XPath que indica on volem que s'incrusti la signatura generada. Per exemple, "/" per ubicar la signatura a continuació del node arrel del document XML.
setXmlReturnBase64	true per obtenir la signatura codificada en Base64 (i evitar així problemes amb Axis a l'hora de serialitzar i deserialitzar signatures XML).
setKeySubjectName	Certificat amb el que generar la signatura.
setInputXmlBase64	Dades a signar, codificades en Base64.

Amb els mètodes de la classe `SmartSignResponse` obtenim la resposta a la nostra petició de signatura:

Mètode	Paràmetres
getResultMajor	Indica si s'ha pogut processar la petició, independentment del resultat d'aquesta.
getResultMinor	Detalls de la operació.
getDocumentWithSignatureXml	Signatura XML.
getDocumentWithSignatureXmlBase64	Signatura XML codificada en Base64.

4.3 Signatura XAdES-T detached a partir del hash d'un document

En aquest cas, exposem codi d'exemple per generar una signatura XML avançada amb segell de temps, a partir del resum criptogràfic (hash) d'un document.

```
import com.safelayer.trustedx.client.smartwrapper.*;
import org.apache.axis.encoding.Base64;
import org.apache.axis.message.SOAPHeaderElement;

public class XAdESTDetachedSignature {

	static final String TRUSTEDX_URL =
"https://ssc.preproduccio.catcert.cat:8443/trustedx-sgw/SignCert";
	static final String SIGNER_DN = "CN=Dummy";
	static final String RESULTMAJOR_SUCCESS =
"urn:oasis:names:tc:dss:1.0:resultmajor:Success";
	static final String DIPOSITARI = "O=0000000000";
	static final String ROL  = "TEST";

	public static byte[] signDetachedXAdEST(byte[] dataToSign) throws
Exception {

		SmartSignRequest sReq = new SmartSignRequest(TRUSTEDX_URL);
		//afegeix la capçalera a la petició
		Stub stub = (Stub) sReq.getStub();
		stub.setHeader(null, "Rol", ROL);
		stub.setHeader(null, "Dipositari", DIPOSITARI);

		//signatura XML simple
		sReq.setProfile(Constants.Profile.XADES);
		sReq.setSignatureFormat(Constants.SignatureFormat.ES_T);
		//dades a signar
		sReq.setInputHashDigest(dataToSign.toString());
		sReq.setInputHashAlgorithm("sha1");
		//selecció de la clau de signatura
		sReq.setKeySubjectName(SIGNER_DN);

		//enviament de la petició
		SmartSignResponse sResp = sReq.send();

		if (RESULTMAJOR_SUCCESS.equals(sResp.getResultMajor())
		&& sResp.getResultMinor() == null) {
			//recuperació de la signatura
			String signature = sResp.getSignatureXml();
			return signature.getBytes();
		} else {
			throw new Exception("Error signing:" +
sResp.getResultMessage());
		}
	}

	public static void main(String args[]) throws Exception {
		String data = "gYbYj9w6DofPvCfwqKKwXitsErA=";
		byte[] signature = signDetachedXAdEST(data.getBytes());
		System.out.println("signature=" + new String(signature));
	}
}
```

Per construir la petició, invoquem els mètodes:

Mètode	Paràmetres
setInputHashDigest	Hash de les dades a signar, codificat en Base64.
setInputHashAlgorithm	Algorisme de hash amb que s'ha calculat el resum anterior

4.4 Signatura de documents PDF

Amb el codi següent enviem una petició de signatura d'un document PDF, fent signatura visible, i consultem la resposta obtinguda.

Important: L'SSC només suporta la signatura d'arxius PDF que utilitzin la versió 1.5 del format PDF.

```
import com.safelayer.trustedx.client.smartwrapper.*;
import java.io.*;
import org.apache.axis.encoding.Base64;
import org.apache.axis.message.SOAPHeaderElement;

public class PdfSignature {
	static final String TRUSTEDX_URL =
"https://ssc.preproduccio.catcert.cat:8443/trustedx-sgw/SignCert";
	static final String SIGNER_DN = "CN=Dummy";
	static final String RESULTMAJOR_SUCCESS =
"urn:oasis:names:tc:dss:1.0:resultmajor:Success";
	static final String DIPOSITARI = "O=0000000000";
	static final String ROL  = "TEST";

	public static byte[] signPdf(byte[] pdfToSign, String signatureInfo) throws
Exception{

		SmartSignRequest sReq = new SmartSignRequest(TRUSTEDX_URL);
		//afegeix la capçalera a la petició
		Stub stub = (Stub) sReq.getStub();
		stub.setHeader(null, "Rol", ROL);
		stub.setHeader(null, "Dipositari", DIPOSITARI);

		//signatura d'un document PDF
		sReq.setProfile(Constants.Profile.PDF);
		//tipus de signatura
		sReq.setSignatureType(Constants.SignatureType.CMS);
		//selecció de la clau de signatura
		sReq.setKeySubjectName(SIGNER_DN);
		//PDF a signar
		sReq.setInputPdfBase64Data(Base64.encode(pdfToSign));
		//configuració signatura visible
		sReq.setPdfSignatureInfo(signatureInfo);

		//enviament de la petició
		SmartSignResponse sResp = sReq.send();

		if (RESULTMAJOR_SUCCESS.equals(sResp.getResultMajor()) &&
sResp.getResultMinor() == null) {
			return Base64.decode(sResp.getDocumentWithSignaturePdf());
		} else {
			throw new Exception("Error signing:" + sResp.getResultMessage());
		}
	}

	public static void main(String args[]) throws Exception {
```

```
            String signatureInfo =
            "<css:PdfSignatureInfo xmlns:css='http://www.safelayer.com/TWS'>" +
            "<css:PdfAttributes><css:signatureAlg>SHA1</css:signatureAlg>" +
            " <css:validationMethod>PPKMS</css:validationMethod>" +
            " <css:signaturePosition>ADDLAST</css:signaturePosition>" +
            " <css:params><css:reason>Author</css:reason></css:params>" +
            "</css:PdfAttributes>" +
            "<css:Appearance>" +
            " <css:Rect x0='100' y0='50' x1='520' y1='150'/>" +
            " <css:Background><css:image encodeType='base64'>" +

    "<css:data>/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAAD/2wBD" +

    "AAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hy" +

    "c5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy" +

    "MjIyMjIyMjIyMjIyMjIyMjIyMjL/wAARCAB+AMcDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQ" +

    "EAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1Fh" +

    "ByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1" +

    "hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLD" +

    "xMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAA" +

    "AAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEI" +

    "FEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZG" +

    "VmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbH" +

    "yMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwD3+iiigAooooAKKK" +

    "KACiiigAooooAKKKKACiiigAooooAKKKKACiiigAqOeeO2t5biZtkUSF3Y" +

    "9lAyTT6z/EH/ACLeqf8AXnL/AOgGgFuUx4w0MjP2qX/wFl/+Jo/4TDQ/+fqX/wABZf8A4muWjk" +

    "8u3QseAgyeuOM9qT7TGSMllz03Arkfj/XFc3t32PSWATSdzqv+Ex0P/n6l/wDAWX/4mj/hMdD/" +

    "AOfqX/wFl/8Aia5rP+cmkyN3X+dSsT5/F+iEgC6lyen+jS/wC9cf8AgL" +

    "L/APE1y/+H/AE1oc5nf2/p/8AeuP/AAF+l/wDiadHrunyOtO9GBHKHdY8+iSoSVS7BF328igknAGStX+veszWcCG1P/AE" +

    "+Qfn5goA1aKKKACiiigAooooAKKKKAErP8Qf8AIt6p/wBecv8A6Aa0O9Z+" +

    "v/8AIt6p/wBekv8A6AaXQFucE4DC2Rzhdh2k9A2Bg/WobbbAWtbuJUkERJkXpIAeT9elXURZLW" +

    "NSMqVBx74pFtYVbd5YjmuBux9BztPbKUlrrSU6btuyvZyTS2kUlx5ZZmRgE/ulhXcl9S/fbY4mwf3eeOM/4Vx2BhQBgBl" +

    "HTHGRXoKgbRXVQd4s4sw/iL0O3kUXN+NzKsePMC+i54FjtHJEhcMPmJwCuf54q9jjFGOtn" +

    "CUma/wDNTaI9vmEx/D2qjfm6MMQuggD0D2Ax49PMFbh+tZesnsnMVqMZ/0yD/0MUAatFFFABRRRQ" +

    "AUUUAFFFFABUF7bLe2FxaMxVZ4mjLDqAwIz+tT0UAc2PCQYyL" +

    "0/Kl/wCEU/6iM3/ftP8A4mub6iGroTLBwwZ1+Qjf9/oIL/eERdkYR4LRbiDo+xRkxwHn7oDr56U9XrFo" +

    "j2cew/rFX+Y53/AIRMdQEUHrw9eSjIjYvERmoV6tgzOCiQRGpnDlxxZtvWcb4wo0/g6DA9dAe1ZPXs6++" +

    "/a/4UfZrn/n+f/v2v2v2+FW6KZJU+y3X/P+P/v2P+wD37X/C/CoZdNkuDF594zrHKkoGwDlTmtGigAoooo" +

    "AKKKKACiiigAooooAKKKKACkOaKX3oAiuJktoGlcMQMcKpYknAAA5JJIAA6k0w/Lu4VRk" +
```

```
        "1zMWrX1/rUCRNsiznZjqueSfw/nTvEF/51wLWN/3acyY7n0qTwvZkyT3r9ztjz2Hf+WPwrPnbl" +

        "ZHV7JRpc73OlopD1pa0OUKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKA" +

        "CqmpXYsrGWY/eAwo9W7CrVYGsebqOqQadCflX55G/u/5B/Wk9tCoK71KWi6e99dNcTZMSOST2d" +

        "u/4Zrq0RY1CooVR2FMtreO1t0hjGEQYFSmlGPKiqtRzlcO9FAoqjMKKKKACiiigAooooAKKKKA" +

        "CiiigAooooAKKKKACiiigAooooAKKKKAENU9PtDAsk8nM9w2+Q+noPwHFXCKWgAooooAKKKKAC" +

        "iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA" +
                "KKKKACiiigAooooAKKKKACiiigD/2Q==</css:data>" +
                " <css:imageSize width='200' height='100'/>" +
                " <css:position x='0' y='0'/></css:image></css:Background>" +
                " <css:Foreground>" +
                " <css:image
encodeType='base64'><css:data>/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgGBgcGBQgHB" +

        "wcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL" +

        "/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyM" +

        "jIyMjIyMjIyMjL/wAARCAABAAEDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAE" +

        "CAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII" +

        "0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGl" +

        "qc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1" +

        "NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgc" +

        "ICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzU" +

        "vAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ" +

        "3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2" +
                "Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwD3+iiigD//2Q==</css:data>" +
                " <css:imageSize width='1' height='1'/>" +
                " <css:position x='0' y='0'/></css:image>" +
                " <css:text><css:properties color='0 0 0' fontSize='10'/>" +
                " <css:position x='12' y='12'/>" +
                " <css:SignatureInfos>" +
                " <css:signatureInfo title=' ' id='Subject'/>" +
                " <css:signatureInfo title=' ' id='Date'/>" +
                " </css:SignatureInfos></css:text></css:Foreground>" +
                " </css:Appearance></css:PdfSignatureInfo>";

                //lectura del document PDF
                File file = new File("unsigned.pdf");
                FileInputStream fis = new FileInputStream(file);
                byte[] pdfToSign = new byte[(int)file.length()];
                fis.read(pdfToSign);
                fis.close();
                byte[] signature = signPdf(pdfToSign, signatureInfo);
                //guardem el PDF signat en un fitxer
                java.io.FileOutputStream fos = new java.io.FileOutputStream("signed.pdf");
                if (fos!=null) {
                        fos.write(signature);
                        fos.close();
                }
        }
}
```

Per construir la petició, invoquem els mètodes de la classe `SmartSignRequest` descrits a la taula següent:

Mètode	Paràmetres
setProfile	Constants.Profile.PDF per seleccionar el perfil de signatura PDF.
setSignatureType	Format de la signatura: Constants.SignatureType.CMS
setPdfSignatureInfo	Tipus de signatura desitjat. En aquest exemple incloem informació de l'aparença de la signatura per tal que sigui visible.
setInputPdfBase64Data	Document PDF a signar codificat en Base64.

4.4.1 Paràmetres de signatura visible

Passarem a descriure els diferents paràmetres que permeten la configuració de la signatura visible en documents PDF.

4.4.1.1 Element <css:PdfSignatureInfo>

Aquest element conté informació sobre els atributs de signatura propis d'Adobe. L'estructura d'aquest element és la següent:

```
<xs:element name="PdfSignatureInfo" type="css:PdfSignatureInfoType"
maxOccurs="unbounded"/>
<xs:complexType name="PdfSignatureInfoType">
 <xs:all>
 <xs:element name="PdfAttributes" type="css:PdfAttributesType"/>
 <xs:element name="Appearance" type="css:AppearanceType" minOccurs="0"/>
 </xs:all>
 <xs:attribute name="name" type="xs:string" use="optional"/>
 <xs:attribute name="order" type="xs:integer" use="optional"/>
</xs:complexType>
```

On podem observat que l'element `<css:PdfSignatureInfo>` inclou els següents elements:

- `<css:PdfAttributes>`: Configuració del mode de signatura.
- `<css:Appearance>` [Opcional]. Paràmetres de l'aparença de la signatura. Si aquest element no està present, es genera una signatura de tipus invisible.

Element <css:PdfAttributes>

L'element `<css:PdfAttributes>` es pot fer servir en dos contexts bàsics:

- Configuració dels paràmetres relacionats amb el mode de signatura.
- Obtenció d'informació sobre el mode de signatura en operacions de verificació.

La seva definició és la següent:

```
<xs:element name="PdfAttributes" type="css:PdfAttributesType"/>
<xs:complexType name="PdfAttributesType">
      <xs:sequence>
            <xs:element name="certified" minOccurs="0">
```

```
                    <xs:simpleType>
                        <xs:restriction base="xs:integer">
                            <xs:enumeration value="0"/>
                            <xs:enumeration value="1"/>
                            <xs:enumeration value="2"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="signatureAlg">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="SHA1"/>
                            <xs:enumeration value="DETACHED"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="validationMethod">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="PPKMS"/>
                            <xs:enumeration value="PPKLITE"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="signaturePosition">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="FIRST"/>
                            <xs:enumeration value="LAST"/>
                            <xs:enumeration value="ADDLAST"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="params" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element     name="location"     type="xs:string"
minOccurs="0"/>
                            <xs:element      name="reason"      type="xs:string"
minOccurs="0"/>
                            <xs:element   name="contactInfo"   type="xs:string"
minOccurs="0"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
</xs:complexType>
```

On es pot observar que l'element `<css:PdfAttributes>` conté una seqüència dels següents elements, que veurem tot seguit:

- `<css:signatureAlg>`
- `<css:validationMethod>`
- `<css:signaturePosition>`
- `<css:params> [Opcional]`

Element <css:signatureAlg>

Indica el tipus de signatura PDF. La següent taula descriu els valors suportats per aquest element:

Valor	Tipus de signatura
SHA1	Signatura PKCS#7 attached sobre un resum criptogràfic SHA1 de les dades del PDF (adbe.pkcs7.sha1)
DETACHED	Signatura PKCS#7 detached sobre les dades d'un PDF (adbe.pkcs7.detached). Adobe recomana utilitzar l'algorisme DETACHED per a un millor compliment dels estàndards.

Element <css:validationMethod>

Selecciona el plug-in amb el que validar la signatura PDF en AdobeReader. La següent taula descriu els valors suportats per aquest element:

Valor	Tipus de signatura
PPKMS	Validador de Microsoft
PPKLITE	Validador d'Adobe

Element <css:signaturePosition>

Indica el posicionament de l'aparença de la signatura. La següent taula descriu els valors suportats per aquest element:

Valor	Tipus de signatura
FIRST	Validador de Microsoft
LAST	Validador d'Adobe
ADDLAST	Pàgina afegida expressament al final del document.

Element <css:params>

Defineix les propietats d'una signatura PDF (que corresponen a entrades del seu diccionari) mitjançant els següents elements:

- `<css:location>`: Lloc de la signatura, indicat mitjançant una cadena de text. Correspon a l'entrada de clau *Location* del diccionari de la signatura.
- `<css:reason>`: Compromís de signatura, indicat mitjançant una cadena de text. Correspon a l'entrada de clau *Reason* del diccionari de signatura.
- `<css:contactInfo>`: Informació de contacte del signatari (p.ex. el seu número de telèfon), indicat mitjançant una cadena de text. Correspon a l'entrada de clau *ContactInfo* del diccionari de signatura.

Exemple d'element <css:PdfAttributes>

```
<css:PdfAttributes>
	<css:validationMethod>PPKMS</css:validationMethod>
	<css:signatureAlg>SHA1</css:signatureAlg>
	<css:signaturePosition>LAST</css:signaturePosition>
	<css:params>
		<css:location>Barcelona</css:location>
		<css:reason>Razon de la firma</css:reason>
		<css:contactInfo>+34 555 666 777</css:contactInfo>
	</css:params>
</css:PdfAttributes>
```

Element <css:Appearance>

Defineix el format de l'aparença de la signatura. Si aquest element no està present, es genera una signatura invisible.

L'estructura d'aquest element és la següent:

```
<xs:element name="Appearance" type="css:AppearanceType" minOccurs="0"/>
<xs:complexType name="AppearanceType">
	<xs:sequence>
		<xs:element name="Rect" type="css:RectType" minOccurs="0"/>
		<xs:element name="Background" type="css:BackgroundType" minOccurs="0"/>
		<xs:element name="Foreground" type="css:ForegroundType" minOccurs="0"/>
	</xs:sequence>
</xs:complexType>
```

Conté els següents elements:

- `<css:Rect>`: Espai on s'incrustarà l'aparença si és necessari crear un nou camp de signatura. Si es treballa amb un camp de signatura buit, aquest element no té cap efecte.
- `<css:Background>`: Imatge de fons de l'aparença de la signatura.
- `<css:Foreground>`: Imatge en primer pla de l'aparença de la signatura.

Element <css:Rect>

L'element `<css:Rect>` indica on col·locar una signatura quan aquesta és visible. La seva definició és la següent:

```
<xs:complexType name="RectType">
	<xs:attribute name="x0" type="xs:positiveInteger"/>
	<xs:attribute name="y0" type="xs:positiveInteger"/>
	<xs:attribute name="x1" type="xs:positiveInteger"/>
	<xs:attribute name="y1" type="xs:positiveInteger"/>
</xs:complexType>
```

Els atributs `x0`, `y0`, `x1`, i `y1` defineixen les coordenades del rectangle al que es mostra la signatura. Les coordenades origen (0,0) es situen a la cantonada inferior esquerra de la pàgina.

- `x0, y0`: Punt inferior esquerre (expressat en píxels)
- `x1, y1`: Punt superior dret (expressat en píxels)

Element <css:Background>

L'element `<css:Background>` defineix la imatge de fons (en segon pla) d'una signatura visible. La seva definició és:

```
<xs:element name="Background" type="css:BackgroundType"/>
<xs:complexType name="BackgroundType">
	<xs:all>
		<xs:element name="image" type="css:ImageType"/>
	</xs:all>
</xs:complexType>
```

L'element `<css:Background>` només conté l'element `<css:image>`, que descriurem més endavant.

Element <css:Foreground>

L'element `<css:Foreground>` defineix la imatge (en primer pla) i el text de la signatura. La seva definició és:

```
<xs:element name="Foreground" type="css:ForegroundType" minOccurs="0"/>
<xs:complexType name="ForegroundType">
	<xs:all>
		<xs:element name="image" type="css:ImageType"/>
```

```
            <xs:element name="text" type="css:TextType"/>
      </xs:all>
</xs:complexType>
```

L'element <css:Foreground> conté els dos elements següents:

- <css:image>: Imatge de la signatura.
- <css:text>: Text amb informació sobre la signatura.

Element <css:image>

L'element <css:image> defineix la imatge d'una signatura. La seva definició és:

```
<xs:element name="image" type="css:ImageType"/>
<xs:complexType name="ImageType">
      <xs:all>
            <xs:element name="data" type="xs:string"/>
            <xs:element name="imageSize" minOccurs="0">
                  <xs:complexType>
                        <xs:attribute name="width" type="xs:positiveInteger"/>
                        <xs:attribute name="height" type="xs:positiveInteger"/>
                  </xs:complexType>
            </xs:element>
            <xs:element name="position" type="css:PositionType" minOccurs="0"/>
      </xs:all>
      <xs:attribute name="encodeType">
            <xs:simpleType>
                  <xs:restriction base="xs:string">
                        <xs:enumeration value="uri"/>
                        <xs:enumeration value="base64"/>
                  </xs:restriction>
            </xs:simpleType>
      </xs:attribute>
</xs:complexType>
```

L'element <css:image> conté els següents atributs:

- //image/@encodeType: Indica la codificació utilitzada per l'element <css:data> per definir la imatge de la signatura. La taula següent defineix els valors suportats per aquest atribut:

Valor d'encodeType	Contingut de <css:data>
uri	Referència al fitxer de la imatge
base64	Imatge codificada en Base64

I els següents elements:

- <css:data>: Imatge de la signatura codificada de la manera que indica l'atribut //image/@encodeType. El format de la imatge ha de ser JPEG.
- <css:imageSize>: Indica, mitjançant els següents atributs, el tamany de visualització de la imatge:
  - //imageSize/@width: Amplada de la imatge (en píxels).
  - //imageSize/@height: Alçada de la imatge (en píxels).

Si aquest element s'omet, el tamany de visualització serà el real de la imatge.

- <css:Position>: Indica la posició relativa de la imatge dins l'espai que defineix l'element <css:Rect>. En concret, aquest element estableix la coordenada inferior esquerra de la imatge mitjançant els següents atributs:

- `//position/@x`: Desplaçament horitzontal (en píxels).

- `//position/@y`: Desplaçament vertical (en píxels).

Element <css:text>

L'element `<css:text>` conté un text amb informació sobre la signatura. La definició és:

```
<xs:element name="text" type="css:TextType"/>
<xs:complexType name="TextType">
	<xs:all>
		<xs:element name="properties" minOccurs="0">
			<xs:complexType>
				<xs:attribute name="color" type="xs:string"/>
				<xs:attribute name="fontSize" type="xs:positiveInteger"/>
			</xs:complexType>
		</xs:element>
		<xs:element name="position" type="css:PositionType" minOccurs="0"/>
		<xs:element	name="SignatureInfos"	type="css:SignatureInfosType"
minOccurs="0"/>
	</xs:all>
</xs:complexType>
<xs:element name="SignatureInfos" type="css:SignatureInfosType" minOccurs="0"/>
<xs:complexType name="SignatureInfosType">
	<xs:sequence>
		<xs:element name="signatureInfo" maxOccurs="unbounded">
			<xs:complexType>
				<xs:attribute name="title" type="xs:string"/>
				<xs:attribute name="id">
					<xs:simpleType>
						<xs:restriction base="xs:string">
							<xs:enumeration value="Subject"/>
							<xs:enumeration value="Issuer"/>
							<xs:enumeration value="SerialNumber"/>
							<xs:enumeration value="Reason"/>
							<xs:enumeration value="Location"/>
							<xs:enumeration value="ContactInfo"/>
							<xs:enumeration value="Date"/>
						</xs:restriction>
					</xs:simpleType>
				</xs:attribute>
			</xs:complexType>
		</xs:element>
	</xs:sequence>
</xs:complexType>
```

L'element `<css.text>` conté els següents elements:

- `<css:properties>` [Opcional] Defineix, mitjançant els següents atributs, el color i el tamany del text:

  - `//properties/@color`: Color del text (en notació RGB, amb valors entre 0 i 1).

  - `//properties/@fontSize`: Tamany de la font.

- `<css:position>` [Opcional] Indica la posició relativa del text dins l'espai que defineix l'element `<css:Rect>`. En concret, aquest element indica la coordenada inferior esquerra del text mitjançant els següents atributs:

  - `//position/@x`: Desplaçament horitzontal (en píxels).

- `//position/@y`: Desplaçament vertical (en píxels).

o `<css:SignatureInfos>` [Opcional]: Inclou una seqüència d'elements `<css:SignatureInfo>`, cadascun dels quals correspon a un camp textual que s'integra en el text de la signatura, i té els següents atributs:

- `//SignatureInfo/@title`: Etiqueta que es mostra al costat del camp de text en la representació gràfica de la signatura.

- `//SignatureInfo/@id`: Identificador del camp de text. Aquest pot correspondre tant a camps del certificat (*Subject*, *Issuer*, *SerialNumber*), com a propietats de la signatura del PDF (*Reason*, *Location*, *ContactInfo*, *Date*) que estan contingudes al seu diccionari. Els valors que pot tenir són:

Camp	Descripció
Subject	Titular del certificat de la signatura
Issuer	Autoritat de certificació (CA) que ha emès el certificat de la signatura
SerialNumber	Número de sèrie del certificat de la signatura
Reason	Raó de la generació de la signatura
Location	Lloc on s'ha portat a terme la signatura
ContactInfo	Dades de contacte de la persona/entitat que ha realitzat al signatura
Date	Data de la signatura

Element <css:StoreSignatureField>

Aquest element selecciona el camp de signatura al qual s'ha d'incrustar la signatura generada. La definició d'aquest element és la següent:

```
<xs:element name="StoreSignatureField">
	<xs:complexType>
		<xs:attribute name="name" type="xs:string" use="required"/>
		<xs:attribute name="create" type="xs:boolean" use="required"/>
	</xs:complexType>
</xs:element>
```

L'element `<css:StoreSignatureField>` conté els següents atributs:

o `//StoreSignatureField/@name`: Identificador del camp de signatura.
o `//StoreSignatureField/@create`: Booleà que indica si cal crear el camp de signatura indicat per `//StoreSignatureField/@name`, si no existeix al PDF.

4.4.1.2 Plantilla de signatures PDF

Una plantilla de signatura PDF és un document XML que té la següent estructura:

```
<xs:element name="PdfTemplate" type="css:PdfTemplateType"/>
<xs:complexType name="PdfTemplateType">
	<xs:sequence>
		<xs:element name="PdfDocument" maxOccurs="unbounded">
			<xs:complexType>
				<xs:sequence>
					<xs:element name="SignatureField"
type="css:PdfSignatureInfoType" maxOccurs="unbounded"/>
				</xs:sequence>
				<xs:attribute name="title" type="xs:string"
use="optional"/>
			</xs:complexType>
		</xs:element>
	</xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="PdfSignatureInfoType">
      <xs:all>
            <xs:element name="PdfAttributes" type="css:PdfAttributesType"/>
            <xs:element name="Appearance" type="css:AppearanceType" minOccurs="0"/>
      </xs:all>
      <xs:attribute name="name" type="xs:string" use="optional"/>
      <xs:attribute name="order" type="xs:integer" use="optional"/>
</xs:complexType>
```

<css:PdfTemplate> és l'element arrel de qualsevol plantilla de signatura PDF. Aquest element consisteix en una seqüència d'elements <css:PdfDocument>, cadascun dels quals defineix una variació de la plantilla (és a dir, plantilles diferents en la pràctica). D'aquesta manera, quan s'hagi de signar un document PDF, fent ús d'una plantilla determinada, s'utilitzarà la "variant" que contingui l'element /css:PdfTemplate/css:PdfDocument, l'atribut /css:PdfTemplate/css:PdfDocument/@title del qual coincideixi amb el títol del document PDF. És a dir, amb el valor que estigui associat a la clau Title del diccionari d'informació del document. Cal tenir en compte que sempre hi hauran correspondències (matching) entre un document PDF i un element /css:PdfTemplate/css:PdfDocument que manqui de l'atribut title.

Per un altre costat, en aquells casos en que:

- El títol del document PDF no coincideixi amb l'atribut title de cap dels elements /css:PdfTemplate /css:PdfDocument de la plantilla.
- I, a més, no hi hagi cap element /css:PdfTemplate /css:PdfDocument que no tingui l'atribut title.

Aleshores, s'haurà d'utilitzar la plantilla de signatures PDF per defecte, és a dir, la plantilla urn:Safelayer:TWS:policies:common:templates:pdf:default. Aquesta plantilla es descriu més endavant en Plantilla de signatures PDF per defecte.

- Cada element <css:PdfDocument> conté un nombre il·limitat d'elements <css:SignatureField>, cadascun dels quals descriu els atributs i l'aparença d'un camp de signatura del document PDF.
- Cada element <css:SignatureField> conté els següents elements:

  - <css:PdfAttributes>, Que determina les característiques de la signatura. Vegi Element <css:PdfAttributes> per obtenir més detalls sobre aquest element.

  - <css:Appearance>, Que determina l'aparença "visual" de la signatura. Vegi Element <css:Appearance> per a més detalls sobre aquest element. Si aquest element s'omet, la signatura que es generi serà invisible.

- I els següents atributs:

  - //SignatureField/@name, que identifica (pel seu nom) el camp de signatura del document PDF a què es refereixen els atributs i l'aparença que defineix aquest element.

  - //SignatureField/@order, que identifica (pel seu nombre d'ordre) el camp de signatura del document PDF a què es refereixen els atributs i l'aparença que defineix aquest element, tenint en compte que el valor 0 identifica el "següent" camp de signatura (sigui quin sigui el seu número d'ordre).

L'element /css:PdfTemplate/css:PdfDocument/css:SignatureField que s'utilitzarà per establir els atributs i l'aparença de la signatura PDF es determinarà de la manera següent:

a) Si en la petició s'ha especificat el nom del camp de signatura (element `//DSS:OptionalInputs/css:StoreSignatureField`) aleshores s'utilitzarà l'element `/css:PdfTemplate/css:PdfDocument/css:SignatureField` l'atribut `//SignatureField/@name` del qual contingui el mateix valor.

b) Si l'atribut `//SignatureField/@name` no és present, la petició no conté l'element `//DSS:OptionalInputs/css:StoreSignatureField`, o bé no hi ha coincidència entre un i altre, llavors s'ha d'utilitzar l'element `/css:PdfTemplate/css:PdfDocument/css:SignatureField` l'atribut `//SignatureField/@order` del qual sigui 0 o coincideixi amb el resultat d'incrementar (en un) el nombre de camps de signatura que contingui el document PDF.

c) Si no s'ha pogut seleccionar cap element `/css:PdfTemplate/css:PdfDocument/css:SignatureField` en base als criteris anteriors, llavors cal utilitzar el (únic) camp `/css:PdfTemplate/css:PdfDocument/css:SignatureField` de la plantilla per defecte del sistema (és a dir, la plantilla `urn:Safelayer:TWS:policies:common:templates:pdf:default`). Aquesta plantilla es descriu seguidament.

Plantilla de signatures PDF per defecte

TrustedX conté una plantilla per defecte per realitzar signatures PDF, que s'identifica per `urn:Safelayer:TWS:policies:common:templates:pdf:default` i que té la següent definició:

```
<css:PdfTemplate xmlns:css="http://www.safelayer.com/TWS">
	<css:PdfDocument>
		<css:SignatureField>
			<css:PdfAttributes>
				<css:signatureAlg>DETACHED</css:signatureAlg>
				<css:validationMethod>PPKMS</css:validationMethod>
				<css:signaturePosition>LAST</css:signaturePosition>
			</css:PdfAttributes>
		</css:SignatureField>
	</css:PdfDocument>
</css:PdfTemplate>
```

On es pot observar que, quan es genera una signatura en un document PDF, utilitzant la plantilla per defecte de TrustedX, aquesta signatura tindrà les següents característiques:

- Serà invisible i s'ubicarà a l'última pàgina.
- El seu format serà el d'una signatura PKCS#7 de tipus detached. És a dir, s'ha d'associar el valor adbe.pkcs.detached a la clau *SubFilter* del diccionari de la signatura.
- El signature-handler que, de manera preferent, s'ha d'utilitzar per a verificar la signatura és el de Microsoft. És a dir, s'ha d'associar el valor Adobe.PPKMS a la clau Filter del diccionari de la signatura.

Exemple de plantilla de signatures PDF

El següent codi XML és un exemple de plantilla de signatures PDF:

```
<?xml version="1.0" encoding="UTF-8"?>
<css:PdfTemplate                                xmlns:css="http://www.safelayer.com/TWS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
	<css:PdfDocument title="Document1">
		<css:SignatureField name="Signature1">
			<css:PdfAttributes>
				<css:signatureAlg>SHA1</css:signatureAlg>
				<css:validationMethod>PPKMS</css:validationMethod>
				<css:signaturePosition>ADDLAST</css:signaturePosition>
				<css:params>
					<css:location>Barcelona</css:location>
					<css:reason>Author</css:reason>
```

```
                                    <css:contactInfo>+345818090</css:contactInfo>
                              </css:params>
                        </css:PdfAttributes>
                  </css:SignatureField>
                  <css:SignatureField order="0">
                        <css:PdfAttributes>
                              <css:signatureAlg>SHA1</css:signatureAlg>
                              <css:validationMethod>PPKMS</css:validationMethod>
                              <css:signaturePosition>ADDLAST</css:signaturePosition>
                              <css:params>
                                    <css:location>Barcelona</css:location>
                                    <css:reason>Author</css:reason>
                                    <css:contactInfo>+345818090</css:contactInfo>
                              </css:params>
                        </css:PdfAttributes>
                        <css:Appearance>
                              <css:Rect x0="31" y0="200" x1="557" y1="40"/>
                              <css:Background>
                                    <css:image encodeType="base64">
                                          <css:data>/9j/4...Q==</css:data>
                                          <css:imageSize width="400" height="40"/>
                                          <css:position x="50" y="35"/>
                                    </css:image>
                              </css:Background>
                              <css:Foreground>
                                    <css:image encodeType="base64">
                                          <css:data>/9j/...ZCg==</css:data>
                                          <css:imageSize width="80" height="30"/>
                                          <css:position x="223" y="1"/>
                                    </css:image>
                                    <css:text>
                                          <css:properties color="0 0 0" fontSize="5"/>
                                          <css:position x="150" y="35"/>
                                          <css:SignatureInfos>
                                                <css:signatureInfo title="Autor de la
firma: " id="Subject"/>
                                                <css:signatureInfo  title="Emisor  del
certificado: " id="Issuer"/>
                                                <css:signatureInfo   title="Número   de
serie: " id="SerialNumber"/>
                                                <css:signatureInfo   title="Razón:   "
id="Reason"/>
                                                <css:signatureInfo
title="Localització: " id="Location"/>
                                                <css:signatureInfo  title="Información
de contacto: " id="ContactInfo"/>
                                                <css:signatureInfo   title="Fecha   de
firma: " id="Date"/>
                                          </css:SignatureInfos>
                                    </css:text>
                              </css:Foreground>
                        </css:Appearance>
                  </css:SignatureField>
            </css:PdfDocument>
            <css:PdfDocument>
                  <css:SignatureField order="0">
                        <css:PdfAttributes>
                              <css:signatureAlg>SHA1</css:signatureAlg>
                              <css:validationMethod>PPKMS</css:validationMethod>
                              <css:signaturePosition>LAST</css:signaturePosition>
                        </css:PdfAttributes>
                  </css:SignatureField>
            </css:PdfDocument>
</css:PdfTemplate>
```

5 Gestió de dades de volum gran

Si els documents XML inclouen elements amb valors de volum gran, Axis experimenta problemes de rendiment. Per pal·liar aquests problemes, podem guardar en arxiu aquests valors, deixant a l'XML una referència a aquest arxiu. D'aquesta manera, el tamany de l'XML que gestiona Axis es redueix considerablement. Més endavant, en la capa de transport, es substitueixen les referències per les dades guardades en arxiu.

Per defecte, les dades no es tracten com a arxiu.

5.1 Gestió de peticions amb dades de gran tamany

Quan el document a signar tingui un volum considerable, i cal enviar-lo sencer en la petició de signatura, podem experimentar problemes de rendiment d'Axis.

La classe `SmartSignRequest` ofereix certs mètodes que permeten gestionar com a arxius les dades de gran volum.

Mètodes:

- `setInputBase64Data(fileName, format)`: Per documents binaris.
- `setInputXmlBase64(fileName, format)`: Per documents XML.
- `setInputPdfBase64Data(fileName, format)`: Per documents PDF.

On:

`fileName` és el nom del fitxer que conté les dades a signar

`format` és el format de les guardades en arxiu, i que pot prendre els següents valors:

- `Constants.SourceFormat.BASE64`
- `Constants.SourceFormat.RAW`

La classe `SmartSignRequest` també ofereix certs mètodes que permeten que en la resposta les dades de gran volum puguin tractar-se també com a arxius:

- `enableSignatureBase64(format)`: Per signatures binàries.
- `enableSignatureXmlBase64(format)`: Per signatures XML.
- `enableDocumentWithSignaturePdf(format)`: Per documents PDF signats.
- `enableDocumentWithSignatureXmlBase64(format)`: Per signatures XML enveloped.

On:

`format` és el format de les guardades en arxiu, i que pot prendre els següents valors:

- `Constants.SourceFormat.BASE64`
- `Constants.SourceFormat.RAW`

Els següents mètodes deshabilitarien l'habilitat amb els corresponents mètodes anteriors:

- `disableSignatureBase64(format)`: Per signatures binàries.

- `disableSignatureXmlBase64(format)`: Per signatures XML.
- `disableDocumentWithSignaturePdf(format)`: Per documents PDF signats.
- `disableDocumentWithSignatureXmlBase64(format)`: Per signatures XML enveloped.

En fer un *disable* després d'un *enable*, deshabilitarem el tractament com a arxiu de les dades corresponents. Recordem que per defecte el tractament com a arxiu està deshabilitat.

NOTA 1: En cas de que habilitem la gestió de dades de la resposta en fitxer, aquests es guardaran al directori indicat al paràmetre `Response.savePath` de la configuració de l'SmartWrapper. Si aquest paràmetre no és present, els arxius es guardaran al directori d'execució.

NOTA 2: Si volem que les dades d'entrada s'agafin directament d'un directori base concret, cal indicar el paràmetre `Request.loadPath` a la configuració d'SmartWrapper. Si no s'especifica, el directori base serà el d'execució.

5.2 Gestió de respostes amb dades de gran tamany

Si s'ha habilitat en la petició de signatura el tractament com a fitxers de dades de la resposta, en cas de voler recuperar aquestes dades per codi, haurem de fer-ho mitjançant els mètodes que veurem a continuació.

La classe `SmartSignResponse` ofereix doncs mètodes que permeten obtenir les dades de la resposta quan aquestes s'han gestionat com a arxius. En aquest cas es tractarà de signatures o documents signats.

Mètodes:

- `getDocumentWithSignaturePdf()`: Per obtenir la referència a un document PDF signat.
- `getDocumentWithSignatureXml()`: Per obtenir la referència a un document que conté la referència a una signatura XML enveloped.
- `getSignatureBase64()`: Per obtenir la referència a una signatura binària.
- `getSignatureXml()`: Per obtenir la referència a una signatura XML.

Quan parlem de referència volem dir el nom del fitxer al directori de sortida.

A partir de els referències obtingudes amb els mètodes anteriors, podem recuperar per codi el contingut dels fitxers que contenen les dades de la resposta:

- `getReferenceFileContent(String reference, boolean deleteFile)`

On:

- `reference` és el nom del fitxer.
- `deleteFile` és un booleà amb el qual podem esborrar el document guardat en fitxer:

  - `true`: L'arxiu que conté el valor, s'esborrarà després de llegir el seu contingut

  - `false`: L'arxiu no s'esborrarà.

5.3 Exemple de gestió en arxius per generació d'una signatura XAdES-BES enveloping

```
import org.apache.axis.encoding.Base64;
import org.apache.axis.message.SOAPHeaderElement;
import org.apache.commons.configuration.Configuration;
import utils.*;
import com.safelayer.trustedx.client.smartwrapper.*;

public class CreateSignature_XAdES_BES_Enveloped_BigFile {
	private static final String path_in = "in/docs_to_sign/BigFiles/";
	private static final String path_out = "out/Signatures/BigFiles/";
	private static final String filename = "Demo-BigFile.xml";

	public static void main(String[] args) {
		try {
			Configuration conf = TrustedXConfiguration.getConfiguration();
			String host = conf.getString("host");
			String distinguishedName = conf.getString("distinguishedname");
			String dipositari = conf.getString("dipositari");
			String rol = conf.getString("rol");
			// Definition of Signature Request endpoint
			SmartSignRequest ssr = new SmartSignRequest(host);
			// Add customized header elements for SOAP
			Stub stub = (Stub) ssr.getStub();
			stub.setHeader(null, "Rol", rol);
			stub.setHeader(null, "Dipositari", dipositari);
			// XML Signature (XAdES)
			ssr.setProfile(Constants.Profile.XADES);
			// Referència al fitxer que conté les dades a signar
			// Paràmetres: path (codificat en Base64) del fitxer + tipus de
dades del fitxer (Raw o Base64)
			ssr.setInputXmlBase64File(Base64.encode((path_in +
filename).getBytes()), Constants.SourceFormat.RAW);
			// Habilitem la gestió en arxiu de la signatura en la resposta

	ssr.enableDocumentWithSignatureXmlBase64File(Constants.SourceFormat.RAW);
			// XAdES signature form
			ssr.setSignatureFormat(Constants.SignatureFormat.BES);
			// Signature Placement

	ssr.setSignaturePlacement(Constants.SignaturePlacement.ENVELOPED);
			ssr.setXmlEnvelopedXPathAfter("//*[local-name()='ssc']//*[local-
name()='description']");
			// Key selection using Distinguished Name
			ssr.setKeySubjectName(distinguishedName);
			// Sending request
			SmartSignResponse ssrs = ssr.send();
			// Retrieveing signature
			if (UtilTrustedX.checkSW(ssrs.getResultMajor(),
ssrs.getResultMinor(), ssrs.getResultMessage())) {
				String destFilename = path_out + filename.substring(0,
filename.lastIndexOf("."))
				+ "_Signature_XAdES_BES_Enveloped.xml";
				Util.writeBinaryFile(destFilename,
ssrs.getDocumentWithSignatureXml().getBytes());
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

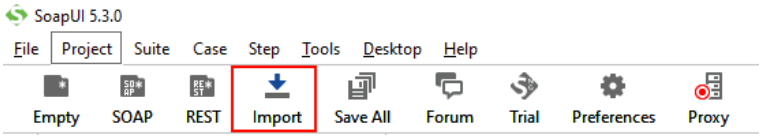
6 Exemple d'ús amb SoapUI

A l'apartat de Documentació de Integració d'aquest servei es disposa d'uns exemples de SOAPUI amb un certificat en format pkcs#12 (pin : 1234) per realitzar l'autenticació al servei.

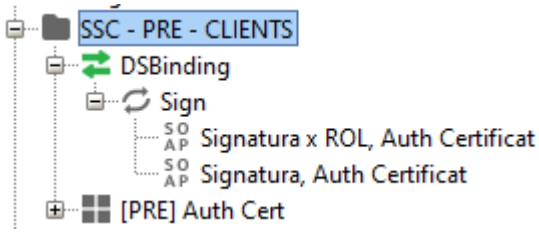
SoapUI és una eina per a realitzar proves d'APIs del tipus SOAP o REST. Es pot descarregar gratuïtament a la web https://www.soapui.org/. Aquest breu guía ha sigut creada amb la versió SoapUI 5.3.0.

6.1 Importar un projecte

Obrir SoapUI i utilitzar l'opció *Import* situada al menú de la part superior.



Seleccionar el projecte proporcionat a la web. L'eina crearà una estructura semblant a la seguent:

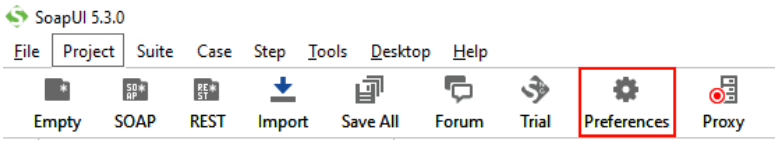


Els missatges SOAP disponibles són:

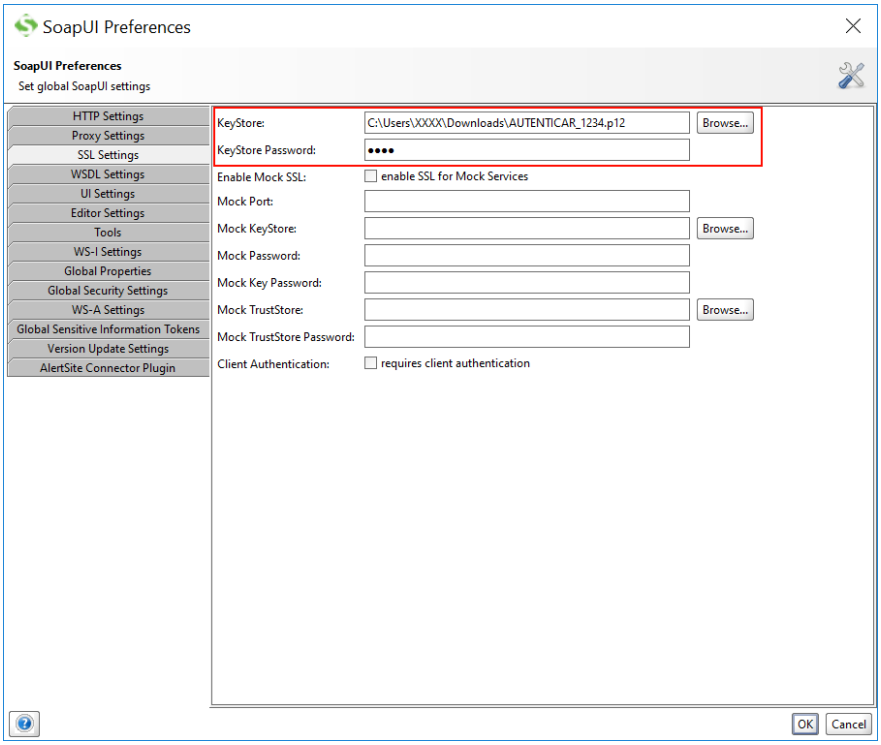
- Signatura amb ROL i autenticació amb certificat
- Signatura sense ROL i amb autenticació amb certificat

6.2 Afegir certificat PKCS#12

Per utilitzar un certificat personal per autenticar amb el servei de signatura, prémer el botó *Preferences* situat al menú superior de l'eina.



A l'apartat *SSL Setting* seleccionar el certificate PKCS#12 proporcionat i posar la contrasenya tal com s'indica a la següent figura:


SoapUI Preferences
SoapUI Preferences
Set global SoapUI settings
HTTP Settings
Proxy Settings
SSL Settings
WSDL Settings
UI Settings
Editor Settings
Tools
WS-I Settings
Global Properties
Global Security Settings
WS-A Settings
Global Sensitive Information Tokens
Version Update Settings
AlertSite Connector Plugin
KeyStore:
C:\Users\XXXX\Downloads\AUTENTICAR_1234.p12
Browse...
KeyStore Password:
Enable Mock SSL:
enable SSL for Mock Services
Mock Port:
Mock KeyStore:
Browse...
Mock Password:
Mock Key Password:
Mock TrustStore:
Browse...
Mock TrustStore Password:
Client Authentication:
requires client authentication
OK
Cancel