Consorci
Administració Oberta
de Catalunya

Plataforma iArxiu. Guia d'integració pels clients dels serveis web

Generalitat de Catalunya

LOCALRET

Creat per: Consorci AOC
Versió: 2.9
Codi de referència: D1300 N-iArxiu/Manuals
Data: 11/04/2012

Índex

1 Introducció

iArxiu és un plataforma de preservació i arxiu de documents, el servei del qual es presenta mitjançant un motor d'arxiu anomenat *Core* utilitzant tecnologia de serveis web.

Aquesta guia pretén ser una referència per a poder desenvolupar programari de software client que pugui interactuar amb el *Core*.

Els coneixements previs necessaris són[1]:

- Interfícies del sistema iArxiu

  (Consultar els documents funcionals i tècnics del sistema iArxiu)
- Comunicacions segures (*HTTPS*) usant clients *Java* i *.NET*

  (Consultar les referències dels llenguatges per a més informació)
- Missatgeria *SOAP*

  http://www.w3.org/TR/soap12-part0/
- Llenguatge d'assercions *SAML*

  http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
- Perfils de seguretat *SAML*

  http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
- Seguretat en els serveis web

  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
- Signatura digital de missatges *SOAP*

  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

A més la guia presenta finalment uns exemples en Java i .NET que poden ser presos com punts de partida per confeccionar els programaris client.

[1] S'indiquen enllaços recomanats

2 Missatgeria iArxiu

Aquest capítol descriu únicament el protocol i no especifica els detalls particulars de les peticions.

iArxiu utilitza missatgeria de serveis web sobre canal segur segons les especificacions *SOAP* (*SOAP/HTTPS*). Ara bé, tot i que el canal de transmissió sigui segur, els missatges han de ser autenticats i autoritzats. És a dir, el *Core* iArxiu, per tal d'executar les accions incloses en els missatges, ha de conèixer la identitat i atribucions de l'usuari sol·licitant.

Els procediments utilitzats per a autenticar i autoritzar els missatges són:

- Autenticació

  Signatura Digital de missatges *SOAP* (*WSSE*)
- Autorització

  Capçaleres al missatge *SOAP* amb informació de l'usuari mitjançant llenguatge d'assercions (*SAML*)

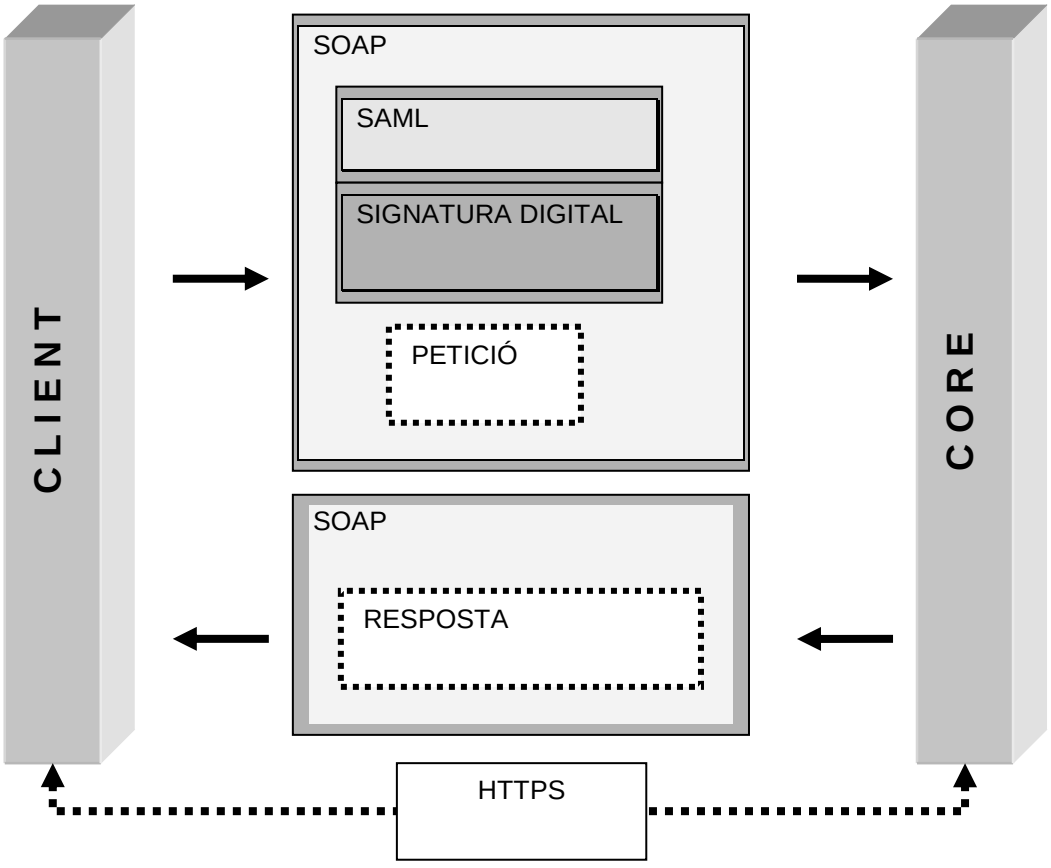

Fig. 1 - Flux de missatges

A continuació es detallen els protocols i versions que s'usen a iArxiu per tal d'autenticar i autoritzar els missatges:

- HTTPS
- SOAP 2.0
- WSSE 1.0
- SAML 2.0[2]

2.1 Autenticació

L'autenticació es realitza verificant la signatura digital del missatge. La signatura la realitza l'emissor del missatge per tal d'acreditar-se davant la plataforma iArxiu.

Per a signar s'utilitza el protocol *WSSE* i s'han de signar obligatòriament les següents parts del missatge:

- Cos (*Body*)
- Capçalera d'atribucions *SAML* (Veure Autorització)

A més la capçalera de seguretat ha d'incloure una marca de temps (*Timestamp*) que acoti temporalment la validesa del missatge.

La **Fig.2** mostra un missatge *SOAP* simplificat. El contingut complet es troba a l'annex Missatge SOAP

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
       <SOAP-ENV:Header>

              <wsse:Security xmlns:wsse="..." SOAP-ENV:mustUnderstand="1">

                     <wsu:Timestamp xmlns:wsu="..." wsu:Id="Timestamp-8120514">
                            <wsu:Created xmlns:wsu="...">
                                   2008-10-17T09:08:32.017Z
                            </wsu:Created>
                            <wsu:Expires xmlns:wsu="...">
                                   2008-10-17T09:13:32.017Z
                            </wsu:Expires>
                     </wsu:Timestamp>

                     <wsse:BinarySecurityToken xmlns:wsu="..." wsu:Id="CertId-
74556710"
                            xmlns:wsse="...">
                            MIIHgTCCBmmgAw ... u4tyELzBGEj2j+6oeTxB+rI9hxlE=
                     </wsse:BinarySecurityToken>

                     <ds:Signature xmlns:ds="http://.../xmldsig#" Id="Signature-
14965598">

                            <ds:SignedInfo xmlns:ds="http://.../xmldsig#">

                                   <ds:Reference URI="#id-14894886" ...>
                                          ...
                                   </ds:Reference>

                                   <ds:Reference URI="#id-15308417" ...>
                                          ...
                                   </ds:Reference>

                            </ds:SignedInfo>
                                   ...
                     </ds:Signature>
```

[2] iArxiu dona únicament suport a la versió *SAML 2.0* i no és compatible amb la versió *SAML 1.x*

```
                </wsse:Security>

                <ish:Context xmlns:ish="http://soap.iarxiu/headers"
                        xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-
1.0.xsd"
                        SOAP-ENV:mustUnderstand="1" wsu:Id="id-15308417">
                        <saml2:Assertion ... >
                                ...
                        </saml2:Assertion>
                </ish:Context>

        </SOAP-ENV:Header>


        <SOAP-ENV:Body
                xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
                wsu:Id="id-14894886">
                ...
        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Fig. 2 - Missatge SOAP simplificat

Com es veu a la **Fig.2** hi ha una capçalera de seguretat segons especificacions *WSSE*, la qual signa les parts crítiques del missatge, conté el segell de temps i informa de qui ha signat el missatge:

- Capçalera d'atribucions d'usuari (*id-15308417*)
- Cos del missatge (*id-14894886*)
- Segell de temps (*Timestamp-8120514*)
- Signant del missatge (*CertId-74556710*)

2.2 Autorització

L'autorització a iArxiu es realitza mitjançant la inserció de capçaleres segons les especificacions *SAML*:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header>

                <wsse:Security
                        ...
                </wsse:Security>

                <ish:Context xmlns:ish="http://soap.iarxiu/headers"
                        xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-
1.0.xsd"
                        SOAP-ENV:mustUnderstand="1" wsu:Id="id-15308417">

                        <saml2:Assertion
                                xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
                                ID="AssertId-1224234511549"
                                IssueInstant="2008-10-17T11:08:31.549+02:00"
Version="2.0">
                                <saml2:Issuer>JavaGuide</saml2:Issuer>
                                <saml2:Subject>
                                        <saml2:SubjectConfirmation

        Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches">
                                                <saml2:NameID>Username</saml2:NameID>
                                        </saml2:SubjectConfirmation>
```

```
                        </saml2:Subject>
                        <saml2:AttributeStatement>
                                <saml2:Attribute

        Name="urn:iarxiu:2.0:names:organizationAlias">
                                        <saml2:AttributeValue>
                                                organization-1
                                        </saml2:AttributeValue>
                                </saml2:Attribute>
                                <saml2:Attribute
                                        Name="urn:iarxiu:2.0:names:fondsAlias">
                                        <saml2:AttributeValue>
                                                fonds-1
                                        </saml2:AttributeValue>
                                </saml2:Attribute>
                                <saml2:Attribute
                                        Name="urn:iarxiu:2.0:names:member-of">
                                        <saml2:AttributeValue>
                                                group-1
                                        </saml2:AttributeValue>
                                </saml2:Attribute>
                        </saml2:AttributeStatement>
                </saml2:Assertion>
        </ish:Context>

        </SOAP-ENV:Header>

        <SOAP-ENV:Body...>
                ...
        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Fig. 3 - Capçalera SAML simplificada

La **fig.3** detalla una capçalera *SAML* –on s'han eliminat parts per claredat -, la qual ha de tenir els següents paràmetres obligatoris:

- Nom: ***Context***
- Espai de noms: ***http://soap.iarxiu/headers***

Dins la capçalera es troba l'asserció en format *SAML*. Es tracta d'una asserció d'atributs en format *SAML 2.0*, la qual ha de complir el següent:

- Elements obligatoris a la seva capçalera
  - ***ID***: identificador
  - ***IssueInstant***: Instant de creació
  - ***Version***: "2.0"
- Identificador de l'emissor

  Nom lliure de l'emissor. És un camp obligatori a l'especificació però iArxiu l'ignora
- Confirmació de l'usuari (*Subject* / *SubjectConfirmation*)

  El *Subject*, o usuari involucrat a la petició, no té perquè coincidir amb l'emissor del missatge. En aquest cas es tracta d'una autenticació al·legada, mitjançant la qual l'emissor actua en nom d'una tercera persona.

  El *Core* iArxiu espera que s'indiqui com s'ha autenticat l'usuari Hi ha diverses maneres d'indicar-ho utilitzant el camp *SubjectConfirmation*. En aquest cap s'especifica el mètode de confirmació de la identitat, el qual pot ser:
  - ***Propietari de la clau (Holder-Of-Key)***

L'identificador és *urn:oasis:names:tc:SAML:2.0:cm:holder-of-key*, i s'utilitza un *KeyInfo* segons les especificacions *XMLDSig* indicant el certificat de l'usuari

- ***Emissor dona testimoni (Sender-Vouches)***

  L'identificador, és *urn:oasis:names:tc:SAML:2.0:cm:sender-vouches* i s'utilitza un camp del tipus *NameID*

- Atribucions de l'usuari (*AttributeStatement*)

  S'especifiquen tantes entrades com atributs tingui l'usuari segons la següent taula:

Atribut	Descripció	URN	Tipus	Multivaluat
Ens	Ens al qual pertany l'usuari	**urn:iarxiu:2.0:names:organization**	xs:string	NO
	Nom simbòlic de l'Ens	**urn:iarxiu:2.0:names:organizationAlias**	xs:string	NO
Fons	Contracte al qual pertany l'usuari	**urn:iarxiu:2.0:names:fonds**	xs:string	NO
	Nom simbòlic del Fons	**urn:iarxiu:2.0:names:fondsAlias**	xs:string	NO
Membre Del Grup	Grups als qual l'usuari pertany	**urn:iarxiu:2.0:names:member-of**	xs:string	SI

2.3 Missatgeries addicionals

2.3.1 Servei *Upload*

La missatgeria *SOAP* està basada en *XML*. Això implica que els binaris ingressats utilitzant aquest protocol són transformats per poder ser transmesos en format *XML*. Aqueta pràctica conceptualment és correcte però a la pràctica pot donar problemes de càrrega en els sistemes, especialment en sistemes clients el dimensionament dels quals, a priori, pot no ser el suficient en el cas d'enviaments de documents grans.

En aquest cas existeix una via addicional de missatgeria la qual consisteix en utilitzar un servei del *Core* no basat en serveis web únicament sinó basat en *HTML* estàndard, mitjançant el qual es pot executar un ingrés previ o càrrega dels documents a banda de la missatgeria *SOAP –Upload-* i un cop finalitzats executar una petició final d'ingrés molt més lleugera utilitzant el protocol habitual de serveis web.

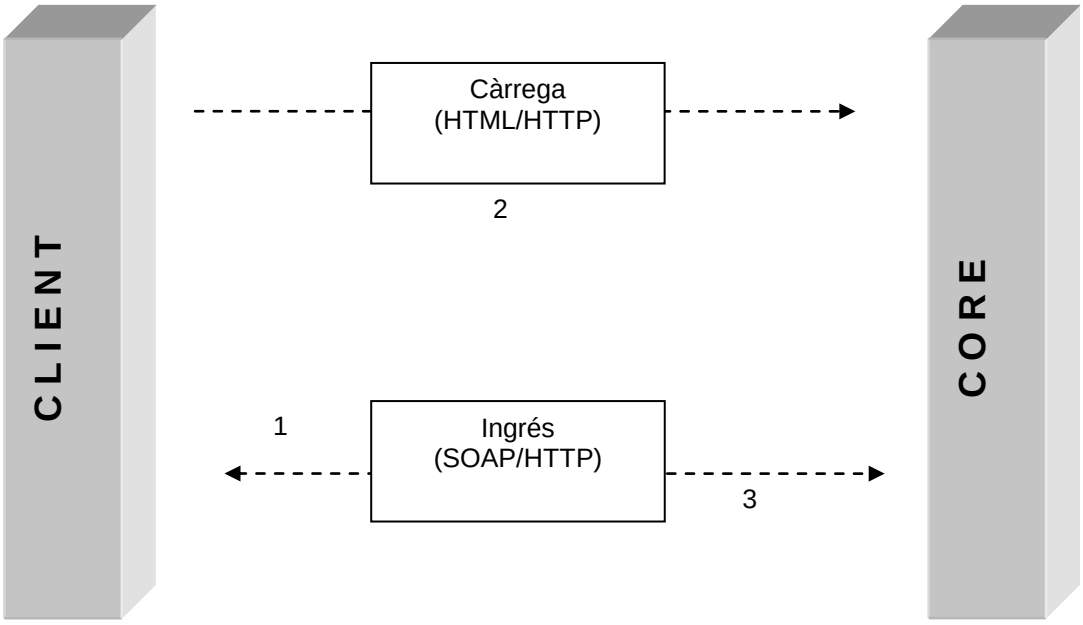

Fig. 4 - Esquema del servei Upload

El procés s'executa de la següent manera:

1. El client sol·licita un identificador de ingrés (*tiquet*) utilitzant els serveis web habituals del *Core*
2. Utilitzant aquest tiquet es fa una càrrega mitjançant protocol *HTML –Upload-*
3. Un cop finalitzades les càrregues prèvies s'invoca un altre servei web del *Core* per tal d'ingressar definitivament el paquet

3 Creació de Programari Client

Aquest capítol és una orientació per crear programari client del *Core* iArxiu. Es crearà una aplicació destinada a exemplificar interaccions d'ingrés i difusió amb un suposat *Core* iArxiu[3] utilitzant tecnologies ***Java*** i ***.NET***.

- ***El programari inclòs en aquesta guia ha de ser pres com a referència i no com a directiva***
- ***En els annexos de la guia es troben tots els codis font usats per a la seva redacció***
- ***Tot i que els codis mostrats en aquesta guia son exemples d'ús, compilen correctament i actuen com a veritables clients iArxiu***
- ***No s'han tingut en compte consideracions relatives a un disseny acurat***
- ***No hi ha tractament d'errors en els codis d'exemple***
- ***És imprescindible un coneixement funcional del sistema, en concret de les funcionalitats d'ingrés i difusió***

[3] Revisar la documentació d'interfícies iArxiu per a més detall

3.1 Comunicacions i protocols

Per a la realització de les tasques de desenvolupament i execució del client **iArxiu**, es necessari disposar d'un ordinador amb connexió a Internet i accés al punt d'entrada dels serveis Web de iArxiu.

- Entorn preproducció:
  - http://www.preproduccio.iarxiu.eacat.cat/core/soap/ingest.wsdl
  - http://www.preproduccio.iarxiu.eacat.cat/core/soap/dissemination.wsdl
  - http://www.preproduccio.iarxiu.eacat.cat/core/soap/administration.wsdl
  - https://www.preproduccio.iarxiu.eacat.cat/core/soap

- Entorn producció:
  - http://www.iarxiu.eacat.cat/core/soap/ingest.wsdl
  - http://www.iarxiu.eacat.cat/core/soap/dissemination.wsdl
  - http://www.iarxiu.eacat.cat/core/soap/administration.wsdl
  - https://www.iarxiu.eacat.cat/core/soap

Els serveis que ofereix el programari **iArxiu** s'han desenvolupat seguint en compte la definició **WSDL**. **WSDL** defineix tots els mètodes, interfícies i elements necessaris per què el programador desenvolupi el codi del seu client d'una forma estàndard i independent de la tecnologia.

Per a verificar la accessibilitat a cadascú dels serveis, des d'una finestra del navegador, es pot introduir la **URL** dels diferents fitxers de descripció dels serveis Web, depenent del entorn al qual es vol accedir.

Si la accessibilitat es correcta, s'obrirà la plana amb la descripció del serveis Web corresponent.

El *Core* iArxiu també admet serveis mitjançant protocols estàndards web (veure Missatgeries addicionals). Les seves *URL* són les mateixes que les llistades en aquest punt però únicament canviant */soap* per */servlet*.

Els serveis de ingrés o consulta no estan disponibles via /servlet. Aquest punt d'entrada està reservat per a usos particulars tal i com es descriu en altres punts d'aquesta guia

3.2 Creació de clients Java

- ***Les accions que es descriuen a continuació es poden automatitzar, o utilitzar eines com Eclipse, però en aquesta guia es descriu el procés més elemental amb intencions il·lustratives***
- ***La guia usa el framework Spring per la qual cosa és recomanable un coneixement mínim d'aquest framework. Alternatives a Spring podrien ser l'ús directe de l'api SAAJ, o la llibreria Rampart d'Apache, les quals no es contemplen en aquesta guia***

> ❖ ***El tractament de documents XML es fa amb la llibreria XMLBeans. En una aplicació real és recomanable el seu ús, però no obligatori***
>
> ❖ ***Els codis d'exemple corresponen a la versió 6 o superior de Java. Aquesta guia inclou referències a l'annex Altres versions de Java en els punts de incompatibilitat***

Per a generar un client Java seran necessàries les següents consideracions:

1. Preparar l'entorn de treball
2. Crear i preparar un projecte Java
3. Crear el codi del client iArxiu Java
4. Configurar el temps d'execució del client iArxiu Java
5. Crear el codi de servei

3.2.1 Preparació de l'entorn Java

Descarregar les següents llibreries i instal·lar-les en algun directori del sistema:

Llibreria	Versió[1]	Lloc de descàrrega
Spring	2.5.4	http://www.springframework.org/
Spring-WS	1.5.2	http://static.springframework.org/spring-ws/sites/1.5/
XMLBeans	2.3.0	http://xmlbeans.apache.org/
WSS4J	1.5.4	http://ws.apache.org/wss4j/
XMLSec	1.4.0	http://santuario.apache.org/
Commons-Logging	1.1	http://commons.apache.org/logging/
*Únicament necessari pel servei **Upload***		
Commons-HttpClient	3.1	http://hc.apache.org/httpclient-3.x/
Commons-Codec	1.3	http://commons.apache.org/codec/

[1] Versió mínima recomanada

Un cop instal·lades afegir al *path* del sistema la carpeta ***bin*** que es troba a la instal·lació de *XMLBeans*.

> ***La eina scomp de XMLBeans usada alguns punts d'aquesta guia necessita compilar codi per la qual cosa és necessari que l'entorn d'execució Java per aquesta eina sigui una JDK i no una JRE. Una manera d'assegurar aquest punt és afegir la carpeta de binaris de la JDK en el primer lloc del path d'execucions del sistema.***

3.2.1.1 Connexions segures (https)

En el cas que el servidor *Core* iArxiu es trobi en una connexió segura s'haurà d'enregistrar el servidor com a entitat de confiança.

És necessari obtenir el certificat que el *Core* iArxiu usarà per identificar-se envers el client.

Un cop obtingut hi ha vàries possibilitats:

1. Enregistrar el certificat a l'arxiu de seguretat de la màquina virtual Java

Aquesta opció és la més directe i no caldrà cap més acció per poder compilar i executar els clients iArxiu Java.

Serà suficient amb situar-se en el subdirectori *<JAVA_HOME>/jre/lib/security* (on *<JAVA_HOME>* correspon al directori d'instal·lació de la màquina virtual Java) i executar la següent comanda per a importar el certificat del servidor a l'arxiu de seguretat *–cacerts-*:

```
> keytool –importcert –keystore cacerts –alias <ALIAS> -file <PATH CERTIFICAT>
```

On es considera:

- *<ALIAS>*: nom[4] que s'assignarà al certificat dins l'arxiu *cacerts*
- *<PATH_CERTIFICAT>*: directori del sistema on es troba el certificat

2. Enregistrar el certificat en un arxiu de seguretat propi

   En aquest cas no s'utilitzarà l'arxiu de seguretat del sistema, per motius específics de cada client.

   S'ha d'actuar com s'ha descrit en el punt anterior. Únicament s'ha d'indicar el nom d'arxiu segur que es desitgi (enlloc de *cacerts*)

   En aquest cas no és suficient enregistrant el certificat. S'ha de tenir especial cura que quan s'executin les comandes per a generar *stubs*, compilar, etc... els codis de client iArxiu Java s'indiquin les propietats:

   - *javax.net.ssl.trustStore*: Ubicació de l'arxiu creat en aquest punt[5]
   - *javax.net.ssl.trustStorePassword*: Contrasenya per accedir a l'arxiu creat en aquest punt

La guia no contempla desenvolupaments especials per a connexions segures donat que com s'ha descrit en aquest punt es tracta únicament d'un problema de configuració i no afecta als codis client iArxiu Java

3.2.2 Preparació del projecte del client iArxiu Java

Es crearà una projecte d'aplicació Java. En aquesta guia es suposarà una aplicació *standalone* anomenada ***JavaGuide***.

Un cop creat el projecte s'han d'executar les següents accions prèvies a la codificació directa del client iArxiu Java:

1. Crear els *stubs* del servei iArxiu
2. Compilar els esquemes per a les *SAML Assertions*
3. Instal·lar les credencials de l'aplicació per tal de poder signar els missatges sortints
4. Actualitzar el *classpath* de l'aplicació

3.2.2.1 Creació dels *stubs* del servei

Crear una carpeta en el projecte amb el nom *WSDL* i un cop en el seu interior executar les següents comandes:

[4] Aquest nom és lliure i no s'utilitza generalment per la qual cosa no té cap implicació
[5] La ruta no pot contenir espais en blanc

1. Crear els *stubs* del servei ingrés (*ingest*):

```
> scomp –dl http://<server>:<port>/core/soap/ingest.wsdl –out ingest-service.jar
```

2. Crear els *stubs* del servei difusió (*dissemination*):

```
> scomp –dl http://<server>:<port>/core/soap/dissemination.wsdl –out
dissemination-service.jar
```

Un cop compilats els esquemes es tindran els arxius necessaris per afegir al *classpath* (veure Classpath de l'aplicació)

3.2.2.2 Compilació de l'esquema SAML

Crear una carpeta en el projecte amb el nom *SAML* i un cop en el seu interior executar les següents comandes:

1. Compilar l'esquema:

```
> scomp –dl http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-
2.0.xsd –out saml-schema-assertion-2.0.jar
```

Un cop compilat l'esquema es tindrà l'arxiu necessari per afegir al *classpath* (veure Classpath de l'aplicació)

3.2.2.3 Credencials de l'aplicació

Les credencials (en aquesta guia d'exemple) es subministraran mitjançant un arxiu extern del tipus ***pfx*** o ***p12***.

Aquest arxiu es copiarà directament a la carpeta de fonts del projecte per a que estigui disponible per *classpath* en temps d'execució. En una aplicació real aquest pot ser un sistema però no l'únic.

3.2.2.4 Classpath de l'aplicació

El *classpath* de l'aplicació estarà integrat per algunes de les llibreries descarregades (veure Preparació de l'entorn), les seves dependències, i els arxius generats en els punts anteriors (veure Creació dels stubs del servei i Compilació de l'esquema Saml)

A continuació es detallen els arxius que integren el *classpath* de l'aplicació els quals es troben dins les descarregues que s'han descrit a Preparació de l'entorn[6]:

Llibreria	Arxius	Opcional
Spring	*Spring-2.5.4.jar*	
Spring-WS	*spring-ws-core-1.5.1.jar*	
	spring-ws-security-1.5.1.jar	
Spring-oxm	*spring-oxm-1.5.1.jar*	
Spring-xml	*spring-xml-1.5.1.jar*	
XMLBeans	*xmlpublic.jar*	
	jsr173_1.0_api.jar	
	resolver.jar	

[6] Els noms d'arxius i les dependències podrien canviar si s'utilitzen versions superiors a les indicades

	xbean.jar	
	xbean_xpath.jar	
	xmlbeans-qname.jar	
WSS4J	*wss4j-1.5.4.jar*	
	opensaml-1.1.jar	SI
Commons-Logging	*commons-logging-api.jar*	
	commons-logging.jar	
XMLSec	*serializer.jar*	
	xalan-2.7.0.jar	
	xmlsec-1.4.0.jar	
Arxius de *WSDL* i *SAML*	*saml-schema-assertion-2.0.jar*	
	ingest-service.jar	
	dissemination-service.jar	
Només necessari pel servei Upload		
Commons-HttpClient	*commons-httpclient-3.1.jar*	
Commons-Codec	*commons-codec-1.3.jar*	

3.2.3 Codi del client iArxiu Java

Un cop el projecte està preparat (veure Preparació de l'entorn Java i Preparació del projecte del client iArxiu Java) es pot crear el codi comú a qualsevol client iArxiu Java. Aquesta part del codi el conformen els següents grups:

1. Interceptor *SAML*
2. *Proxy* del servei

Com s'observa no hi ha codi per a signar el missatge sortint. El motiu és que el missatge sortint es signa mitjançant configuració de l'aplicació i no hi ha codi directe d'aplicació involucrat en aquesta acció (veure Configuració de l'interceptor de signatura SOAP).

3.2.3.1 Interceptor *SAML*

L'interceptor *SAML* afegirà les *SAML Assertions* al missatge sortint mitjançant intercepció (veure el contingut íntegre a l'annex SamlInterceptor.java). Per a detalls de com afegir aquest interceptor veure Configuració del client iArxiu Java

Dins el codi de l'interceptor es troben dues grans parts. Per una banda el propi codi d'intercepció i per altre el codi que representa la *SAML Assertion*.

La **Fig.5** mostra la part d'intercepció del missatge:

```
030 public class SamlInterceptor implements ClientInterceptor {

038     @Override
039     public boolean handleRequest(MessageContext messageContext)
040                     throws WebServiceClientException {
041             QName qn = new QName("http://soap.iarxiu/headers","Context","ish");
042
043             SoapMessage soapMessage = ((SoapMessage)messageContext.getRequest());
044
045             SoapHeaderElement wsh =
soapMessage.getSoapHeader().addHeaderElement(qn);
046             wsh.setMustUnderstand(true);
```

```
047
048             AssertionDocument assertion = createAssertion();
049
050             Map<String,String> ns = new HashMap<String,String>();
051             ns.put("urn:oasis:names:tc:SAML:2.0:assertion", "saml2");
052
053             XmlOptions xmlOptions = new XmlOptions();
054             xmlOptions.setSaveSuggestedPrefixes(ns);
055             xmlOptions.setSaveAggressiveNamespaces();
056
057             XmlBeansMarshaller marshaller = new XmlBeansMarshaller();
058             marshaller.setXmlOptions(xmlOptions);
059
060             try {
061                     marshaller.marshal(assertion, wsh.getResult());
062             } catch(IOException e) {
063                     throw new SaajSoapHeaderException("Error creating SAML Header",
e);
064             }
065
066             return true;
067     }
```

Fig. 5 - Interceptor SAML

Com es veu l'interceptor deriva directament de la *interface* Spring ***ClientInterceptor*** [030]. Només és necessari el mètode ***handleRequest***, tot i que en una aplicació real podria no ser així (veure l'annex SamlInterceptor.java per saber com completar la resta de mètodes de la *interface*).

El mètode *handleRequest* genera una nova capçalera *SOAP* segons el prefix, nom i espai de noms que s'indiquen a la variable *qn* [041]. Els valors d'aquest variable no son ficticis i són els que s'han d'usar en una aplicació real. L'associació de prefix a espai de noms [050->051] és opcional i eliminar-la no afecta a l'aplicació final.

Aquest interceptor delega la creació de la *SAML Assertion* en un mètode exterior [048], el qual es mostra a la següent figura:

```
075     protected AssertionDocument createAssertion() {
076             AssertionDocument assertionDocument =
AssertionDocument.Factory.newInstance();
077             AssertionType assertion;
078
079             assertion = assertionDocument.addNewAssertion();
080
081             assertion.setVersion("2.0");
082             assertion.setID("AssertId-" + System.currentTimeMillis());
083             assertion.setIssueInstant( Calendar.getInstance() );
084
085             NameIDType issuerName = assertion.addNewIssuer();
086             issuerName.setStringValue("JavaGuide");
087
088             SubjectType subject = assertion.addNewSubject();
089
090             SubjectConfirmationType subjectConfirmation =
subject.addNewSubjectConfirmation();
091             subjectConfirmation.setMethod("urn:oasis:names:tc:SAML:2.0:cm:sender-
vouches");
092
093             NameIDType subjectName = subjectConfirmation.addNewNameID();
094             subjectName.setStringValue("Username");
095
096             AttributeStatementType attributeStatement =
assertion.addNewAttributeStatement();
097
098             addAttribute(attributeStatement,
"urn:iarxiu:2.0:names:organizationAlias", "organization-1");
```

```
099             addAttribute(attributeStatement, "urn:iarxiu:2.0:names:fondsAlias",
"fonds-1");
100             addAttribute(attributeStatement, "urn:iarxiu:2.0:names:member-of",
"archivists");
101
102             return assertionDocument;
103     }
```

Fig. 6 - Creació d'una SAML Assertion amb Java

Els paràmetres obligatoris de la *SAML Assertion* es mostren a [081->083] i en una aplicació real es poden generar de la mateixa manera que en aquesta guia.

La versió [081] ha de ser obligatòriament la 2.0.

A [086] es mostra que s'ha d'indicar el nom de l'emissor. Com ja s'ha comentat aquest nom és lliure i iArxiu l'ignora.

A [090->094] s'indica la confirmació d'usuari. En aquest exemple s'indica que l'emissor dona testimoni [091] de l'usuari i no es presenten més credencials que el nom [093->094]. Com ja s'ha comentat hi ha altres maneres de relacionar la *SAML Assertion* amb un usuari.

La part final de la creació consisteix en indicar els atributs de l'usuari [096->100]: Ens, Fons i grups als quals pertany.

Els atributs de l'usuari estan codificats directament en aquest mètode. En una aplicació real els atributs d'usuari s'haurien d'aconseguir per altres mitjans més genèrics

En aquest exemple s'ha creat un mètode d'utilitat per facilitar la tasca [105->116]:

```
105     private void addAttribute(
106                     final AttributeStatementType attributeStatement,
107                     final String name,
108                     final String value ) {
109
110             AttributeType attribute = attributeStatement.addNewAttribute();
111             attribute.setName(name);
112             XmlObject xmlNode = attribute.addNewAttributeValue();
113             Node node = xmlNode.getDomNode();
114             Text text = node.getOwnerDocument().createTextNode(value);
115             node.appendChild(text);
116     }
```

Fig. 7 - Afegir un attribut a una *SAML Assertion* amb Java

3.2.3.2 *Proxy* del servei

El *Proxy* del servei és una classe d'utilitat per a enviar i rebre missatges des d'un client iArxiu Java (veure el codi íntegre a l'annex ProxyClient.java)

```
009 public class ProxyClient extends WebServiceGatewaySupport{

011   public ProxyClient() {
012         XmlBeansMarshaller xmlBeansMarshaller = new XmlBeansMarshaller();
013         setMarshaller(xmlBeansMarshaller);
014         setUnmarshaller(xmlBeansMarshaller);
015   }

017   public IngestResponseDocument send(IngestRequestDocument request) {
018     return
(IngestResponseDocument)getWebServiceTemplate().marshalSendAndReceive(request);
```

```
019    }
…
```

Fig. 8 - Proxy del client iArxiu Java

Com es veu la classe hereta de *WebServiceGatewaySupport* [009] i declara en el seu constructor el *marshaller* [013] i *unmarshaller* [014] per a convertir classes Java a XML i l'inrevés.

Aquesta classe és una utilitat que facilita la integració. La única tasca que realitza és una comprovació de tipus i per aquest motiu conté les funcions de les *interfícies* de servei amb adaptació de tipus. Les línees [017->019] són un exemple del mètode ingrés.[7] (Veure el codi íntegre a ProxyClient.java)

3.2.4 Configuració del client iArxiu Java

Com s'ha comentat en els punts anteriors una aplicació client iArxiu Java està conformada per un conjunt de mòduls amb funcionalitats específiques.

Per a confeccionar l'encaix de totes les peces, l'aplicació ha de ser configurada. Com aquesta guia tracta d'una aplicació Spring la configuració es realitza segons el *framework* Spring amb un arxiu de context. Aquest arxiu s'anomena generalment ***applicationContext.xml*** i el seu contingut íntegre es pot trobar a l'annex ApplicationContext.xml.

3.2.4.1 Configuració de l'interceptor SAML

Aquest interceptor no necessita més configuració que la seva declaració com es veu a la següent figura:

```
026    <bean id="samlInterceptor" class="iarxiu.javaguide.SamlInterceptor"/>
```

Fig. 9 - Configuració de l'interceptor SAML amb Spring

En una aplicació real segurament s'haurien d'injectar propietats addicionals.

3.2.4.2 Configuració de l'interceptor de signatura SOAP

La següent figura mostra com es configura l'interceptor per a signar digitalment els missatges SOAP.

```
028    <bean id="wssInterceptor"
class="org.springframework.ws.soap.security.wss4j.Wss4jSecurityInterceptor">
029            <property name="securementActions" value="Signature Timestamp"/>
030            <property name="securementUsername" value="..."/>
031            <property name="securementPassword" value="..."/>
032            <property name="securementSignatureCrypto" ref="crypto"/>
033            <property name="securementSignatureKeyIdentifier"
value="DirectReference"/>
034
035            <property name="securementSignatureParts">
036
       <value>{}{http://schemas.xmlsoap.org/soap/envelope/}Body;{}{http://soap.iarxiu
/headers}Context;</value>
037            </property>
038
```

[7] Consultar la documentació Spring-WS per a més detalls sobre altres mètodes d'enviament i recepció de missatges

```
039     </bean>

041     <bean id="crypto"
class="org.springframework.ws.soap.security.wss4j.support.CryptoFactoryBean">
042             <property name="keyStoreLocation" value="classpath:....p12"/>
043             <property name="keyStoreType" value="pkcs12"/>
044             <property name="keyStorePassword" value="..."/>
045     </bean>
```

Fig. 10 – Configuració d'interceptor de signatura SOAP amb Spring

L'interceptor es declara com *wssInterceptor* [028] però s'observa que hi ha un altre declaració amb el nom *crypto* [041].

El *bean crypto* s'utilitza per accedir a arxius segurs que contenen claus privades i certificats d'usuari. Els seus paràmetres són:

- ***keyStoreLocation*** [042]: Ubicació de l'arxiu de claus. En el cas de la guia l'arxiu s'ha col·locat en un lloc accessible per *classpath* i per aquest motiu el nom s'indica com *classpath:Nomd'arxiu*
- ***keyStoreType*** [043]: El tipus pot ser pkcs12 o pfx
- ***keyStorePassword*** [044]: La contrasenya per accedir a l'arxiu de claus

El *bean crypto* es configurarà per accedir a l'arxiu de credencials de l'aplicació (veure Credencials de l'aplicació) i correspondrà a la identificació que acreditarà l'aplicació client iArxiu en el Core iArxiu.

L'interceptor *wssInterceptor* no hauria de canviar a excepció dels paràmetres propis de cada aplicació:

- ***securementActions*** [029]: El missatges que el Core iArxiu accepta han de tenir signatura digital i marca de temps (*Signature Timestamp*)
- ***securementSignatureKeyIdentifier*** [033]: Respectar *DirectReference*
- ***securementSignatureParts*** [035]: S'indiquen els noms qualificats del elements *SOAP* a signar i han de ser els indicats (no son ficticis)
- ***securementUsername*** i ***securementPassword*** [030, 031]: Corresponen a les entrades especifiques de l'arxiu de claus instal·lat a l'aplicació (*alias* i *contrasenya*). Consultar Alias d'un PFX o P12 per veure com obtenir l'*alias* d'un arxiu de claus.
- ***securementSignatureCrypto*** [032]: Nom del *bean* per accedir a l'arxiu de claus. Ha de correspondre amb el declarat ([041])

3.2.4.3 Configuració del Proxy

El *proxy* es configura segons es detalla a la següent figura:

```
006     <bean id="proxyClient" class="iarxiu.javaguide.ProxyClient">
007             <property name="defaultUri" value="http://<server:<port>/core/soap"/>
008             <property name="messageFactory" ref="messageFactory"/>
009             <property name="interceptors">
010                     <list>
011                             <ref bean="samlInterceptor"/>
012                             <ref bean="wssInterceptor"/>
013                     </list>
014             </property>
015     </bean>
```

```
017     <bean id="messageFactory"
018             class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
019             <property name="messageFactory">
020                     <bean
021
        class="com.sun.xml.internal.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Im
pl">
022                     </bean>
023             </property>
024     </bean>
```

Fig. 11 - Configuració del proxy de servei amb Spring

La classe que implementa el *proxy* [006] correspon a la implementació generada en el punt Proxy del servei i els paràmetres de configuració son els següents:

- ***defaultUri*** [007]: Adreça del *Core* iArxiu[8]
- ***interceptors*** [009->014]: Es declaren els interceptors definits a Configuració de l'interceptor Saml i Configuració de l'interceptor de signatura SOAP i l'ordre mostrat no pot ser alterat. En cas contrari el client fallarà (no es pot signar una capçalera abans de crear-la)
- ***messageFactory*** [008]: S'ha de declarar un *bean* amb la implementació *Saaj*. Com la guia fa ús de la versió 6 de Java (cas habitual) es pot prendre la mostrada a l'exemple [021]. En el cas de no utilitzar la JVM 6 consultar l'annex Implementació SAAJ usant versions anteriors a Java 6

3.3 Creació de clients .NET

- ***La guia suposa que es disposa d'algun entorn integrat com el Visual Studio però no utilitzarà cap funcionalitat específica del mateix amb motiu de clarificar els conceptes***
- ***Es pren el llenguatge C# com a base per a la generació de l'exemple, però no és obligatori. Es pot generar fàcilment l'exemple amb altres llenguatges com C++.NET o VB.NET***
- ***És un requisit utilitzar una versió de .NET Framework superior o igual a la 2.0***

Per a generar un client .NET seran necessàries les següents consideracions:

1. Preparar l'entorn de treball
2. Crear i preparar un projecte .NET
3. Crear el codi del client iArxiu .NET
4. Configurar el temps d'execució del client iArxiu .NET
5. Crear el codi de servei

[8] Per a connexions segures veure l'apartat sobre preparació de l'entorn Java

3.3.1 Preparació de l'entorn .NET

Per a confeccionar clients de serveis web i afegir signatures digitals Microsoft ha creat el paquet ***WSE*** (*Webservices Enhancements*). La versió més actual del paquet és la 3 i facilita enormement la generació de missatges SOAP signats, per la qual cosa és la versió que es prendrà com a referència en el desenvolupament del codi d'exemple de la guia.

Com a contrapartida *WSE3* necessita per a la seva eina de generació el .NET Framework SDK 2.

El .NET Framework 2 no és necessari per l'aplicació final sinó únicament per a la eina de generació de codi (veure Preparació del projecte del client iArxiu .NET)

Segons el que s'ha comentat es descarregarà i instal·larà en el sistema els següents paquets:

Paquet	Descàrrega
.NET Framework 2 SDK	http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=fe6f2099-b7b4-4f47-a244-c96d69c35dec[1]
WSE3	http://www.microsoft.com/downloads/details.aspx?FamilyID=018a09fd-3a74-43c5-8ec1-8d789091255d&displaylang=en
.NET Framework	Es pot descarregar qualsevol versió igual o superior a la 2.0 i en general ja no és necessari la descàrrega perquè el .NET Framework forma part del sistema operatiu Windows Vista i per les versions de Windows anteriors a Windows Vista s'afegeix indirectament mitjançant el servei Windows Update

[1]Tenir en compte que l'enllaç indicat és per a plataforma x86

La instal·lació de WSE3 haurà creat un conjunt d'utilitats i llibreries, a la carpeta d'Arxius de programes. És convenient, però no obligatori, afegir la carpeta *v3.0/Tools* al *path* del sistema, per així poder executar les eines des de qualsevol lloc del sistema.

3.3.1.1 Connexions segures (https)

En el cas que el servidor *Core* iArxiu es trobi en una connexió segura s'haurà d'enregistrar el servidor com a entitat de confiança.

És necessari obtenir el certificat que el *Core* iArxiu usarà per identificar-se envers el client i afegir-lo a l'arxiu segur de claus de Windows.

Hi ha vàries possibilitats per enregistrar un certificat en el sistema:

1. Accedint al menú contextual de l'arxiu de certificat i executar l'opció per a instal·lar el certificat
2. Accedint a l'assistent de configuració d'Internet; després a la pestanya de connexions; i finalment opció de certificats

La guia no contempla desenvolupaments especials per a connexions segures donat que com s'ha descrit en aquest punt es tracta únicament d'un problema de configuració i no afecta als codis client iArxiu .NET

3.3.2 Preparació del projecte del client iArxiu .NET

Es crearà un projecte d'aplicació .NET. En aquesta guia es suposarà una aplicació de consola amb el nom ***NETGuide***, per a la qual es generarà com *assembly* final l'arxiu ***iArxiu.NETGuide***.

Un cop creat el projecte s'han d'executar les següents accions prèvies a la codificació directa del client iArxiu .NET.

1. Crear els *stubs* del servei
2. Compilar els esquemes per a les *SAML Assertions*
3. Instal·lar les credencials de l'aplicació per tal de poder signar els missatges sortints
4. Afegir les referències necessàries de l'aplicació

3.3.2.1 Creació dels *stubs* del servei

Crear una carpeta en el projecte amb el nom *WSDL* i un cop en el seu interior executar la següent comanda per a sol·licitar les definicions dels serveis.

Per tal de poder executar la comanda **wsewsdl3** cal tenir a les variables d'entorn del sistema la carpeta Tools del paquet WSE v3.0 (C:\Program Files\Microsoft WSE\v3.0\Tools)

- Crear els stubs del servei ingrés (*ingest*)

```
> wsewsdl3 http://<server>:<port>/core/soap/ingest.wsdl
/namespace:iArxiu.Core.Services.Ingest  /type:webClient
```

- Crear els stubs del servei difusió (*dissemination*)

```
> wsewsdl3 http://<server>:<port>/core/soap/dissemination.wsdl
/namespace:iArxiu.Core.Services.Dissemination  /type:webClient
```

El valor del paràmetre */type* és obligatori que sigui *webClient*, mentre que el del paràmetre */namespace* és opcional. En aquesta guia s'han pres els valors indicats per a generar el codi de l'aplicació amb major claredat.

Un cop executada aquesta comanda s'hauran generat uns arxius amb el nom ***IngestService.cs*** i ***DisseminationService.cs*** els quals s'afegiran posteriorment al projecte (veure Referències de l'aplicació).

3.3.2.2 Compilació de l'esquema SAML

Crear una carpeta en el projecte amb el nom *SAML* i un cop en el seu interior executar les següents comandes:

1. Descarregar els següents esquemes:
   - http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd
   - http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd
   - http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd
2. Compilar els esquemes:

Per tal de poder executar la comanda **xsd** cal tenir a les variables d'entorn del sistema la carpeta bin del SDK v2.0 del Visual Studio (C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin)

```
> xsd /c -schema-assertion-2.0.xsd xmldsig-core-schema.xsd xenc-schema.xsd
```

Un cop executada aquesta comanda s'hauran compilat els esquemes dins un únic arxiu el nom del qual estarà format per la concatenació dels noms que conformen l'esquema. Aquest arxiu s'afegirà posteriorment al projecte (veure Referències de l'aplicació).

3.3.2.3 Credencials de l'aplicació

En aquesta aplicació d'exemple les credencials s'obtindran del magatzem segur de Windows, per la qual cosa s'haurà d'instal·lar l'arxiu p12 o pfx utilitzant les eines de Windows o utilitzar un que ja estigui instal·lat.

3.3.2.4 Referències de l'aplicació

Per a la correcte compilació i execució del projecte és necessari afegir les següents referències:

- *System*
- *System.Security*
- *System.Web.Services*
- *System.Xml*
- *Microsoft.Web.Services3*[9]

A més s'han d'afegir a l'aplicació els arxius generats en els punts Creació dels stubs del servei i Compilació de l'esquema SAML, és a dir els arxiu:

- *WSDL/IngestService.cs*
- *WSDL/DisseminationService.cs*
- *SAML/saml-schema-assertion-2_0_xmldsig-core-schema_xenc-schema.cs*

3.3.3 Codi del client iArxiu .NET

Un cop el projecte està preparat (veure Preparació de l'entorn .NET i Preparació del projecte del client iArxiu .NET) es pot crear el codi comú a qualsevol client iArxiu .NET. Aquesta part del codi el conformen els següents components:

1. Extensions SAML
2. Extensions de signatura

3.3.3.1 Extensions *SAML*

Les extensions *SAML* afegiran una capçalera *SAML* al missatge *SOAP*. Es necessiten per aquesta tasca dos mòduls:

1. Capçalera *SAML*

[9] La referència Microsoft.Web.Services3 no forma part del Framework .NET sinó del paquet WSE3

2. Extensió *SAML*

3.3.3.1.1 *Capçalera* SAML

La capçalera *SAML* es modela mitjançant codi segons es detalla a la següent figura (veure el contingut íntegre a l'annex SamlHeader.cs)

```
012     [XmlRoot("Context", Namespace = "http://soap.iarxiu/headers")]
013     public class SamlHeader : SoapHeader
014     {
015         [XmlAttribute("Id", Namespace = "http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd")]
016         public string Id = "Id-" + Guid.NewGuid().ToString();
017
018         [XmlElement("Assertion", Namespace =
"urn:oasis:names:tc:SAML:2.0:assertion")]
019         public AssertionType Assertion;
020     }
```

Fig. 12 - Creació d'una capçalera SAML amb .NET

Com es veu la capçalera té part en codi i part declarativa (*Attributes*). Així el nom i espai de noms desitjat per a la capçalera es declara mitjançant atributs [012] i estan fixats (no son noms ficticis) a:

- Nom: *Context*
- Espai de noms: *http://soap.iarxiu/headers*

Com aquesta capçalera anirà signada és obligatori que tingui un atribut segons especificacions WSSE. Per aquest motiu la propietat que actuarà com identificador [016] es declara mitjançant un atribut [015] segons el següent:

- Nom: *Id*
- Espai de noms: *http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsecurity-utility-1.0.xsd*

Finalment la capçalera inclou el contingut que en aquest cas consta únicament de la *SAML Assertion*. Aquest contingut s'aconsegueix declarant una variable del tipus *AssertionType* [019] i indicant els atributs següents:

- Nom: *Assertion*
- Espai de noms: *urn:oasis:names:tc:SAML:2.0:assertion*

3.3.3.1.2

3.3.3.1.3 *Extensió SAML*

En el .NET Framework les capçaleres de client (veure Capçalera SAML) s'afegeixen mitjançant les *Extensions*.

Les *extensions* tenen codi associat i configuració (per la configuració veure Configuració de l'aplicació):

```
010     public class SamlSoapExtension : SoapExtension
011     {

027         public override void ProcessMessage(SoapMessage message)
```

```
028         {
029             switch (message.Stage)
030             {
031                 case SoapMessageStage.BeforeSerialize:
032                     if (message is SoapClientMessage)
033                     {
034                         SamlHeader samlHeader = new SamlHeader();
035                         samlHeader.MustUnderstand = true;
036 
037                         AssertionType assertion = CreateAssertion();
038 
039                         samlHeader.Assertion = assertion;
040 
041                         message.Headers.Add(samlHeader);
042                     }
043                     break;
044             }
045         }
```

Fig. 13 - Creació d'una extensió SAML amb .NET

Com es veu les *SAML Extensions* implementen la *interface SoapExtension* [010]. La **fig.13** detalla únicament els mètodes que s'han d'implementar. Per saber com s'implementen la resta de mètodes revisar el contingut íntegre a l'annex SamlSoapExtension.cs.

El mètode requerit és *ProcessMessage* [027->045] mitjançant el qual es crea una nova capçalera [034->035], la qual s'afegeix directament al missatge [041].

La capçalera conté la *SAML Assertion* la qual es crea [037] mitjançant la invocació del mètode d'ajuda *CreateAssertion* [047->082].

La següent figura descriu el mètode d'utilitat *CreateAssertion*:

```
047         private AssertionType CreateAssertion()
048         {
049             AssertionType assertion = new AssertionType();
050 
051             assertion.Version = "2.0";
052             assertion.ID = "AssertID-" + Guid.NewGuid().ToString;
053             assertion.IssueInstant = DateTime.Now;
054 
055             NameIDType issuerName = new NameIDType();
056             issuerName.Value = "NETGuide";
057             assertion.Issuer = issuerName;
058 
059             SubjectType subject = new SubjectType();
060 
061             SubjectConfirmationType subjectCofirmation = new
SubjectConfirmationType();
062             subjectCofirmation.Method = "urn:oasis:names:tc:SAML:2.0:cm:sender-
vouches";
063 
064             NameIDType subjectName = new NameIDType();
065             subjectName.Value = "Username";
066 
067             subjectCofirmation.Item = subjectName;
068 
069             subject.Items = new Object[] { subjectCofirmation };
070 
071             assertion.Subject = subject;
072 
073             AttributeStatementType attributeStatement = new
AttributeStatementType();
074 
075             AttributeType organization =
CreateAttribute("urn:iarxiu:2.0:names:organizationAlias", "organization-1");
076             AttributeType fonds =
CreateAttribute("urn:iarxiu:2.0:names:fondsAlias", "fonds-1");
```

```
077             AttributeType memberOf =
CreateAttribute("urn:iarxiu:2.0:names:member-of", "archivists");
078
079             attributeStatement.Items = new object[] { organization, fonds,
memberOf };
080
081             assertion.Items = new AttributeStatementType[] { attributeStatement
};
082
083             return assertion;
084         }
```

Fig. 14 - Creació d'una *SAML Assertion* amb .NET

Els paràmetres obligatoris de la *SAML Assertion* es mostren a [051->053] i en una aplicació real es poden generar de la mateixa manera que en aquesta guia.

La versió [051] ha de ser obligatòriament la 2.0.

A [056] es mostra que s'ha d'indicar el nom de l'emissor. Com ja s'ha comentat aquest nom és lliure i iArxiu l'ignora.

A [060->065] s'indica la confirmació d'usuari. En aquest exemple s'indica que l'emissor dona testimoni [061] de l'usuari i no es presenten més credencials que el nom [064->065]. Com ja s'ha comentat hi ha altres maneres de relacionar la *SAML Assertion* amb un usuari.

La part final de la creació consisteix en indicar els atributs de l'usuari [073->081]: Ens, Fons i grups als quals pertany.

Els atributs de l'usuari estan codificats directament en aquest mètode. En una aplicació real els atributs d'usuari s'haurien d'aconseguir per altres mitjans més genèrics

En aquest exemple s'ha creat un mètode d'utilitat per facilitar la tasca [086->093] segons es detalla a la següent figura:

```
086         private AttributeType CreateAttribute(string name, string value)
087         {
088             AttributeType attribute = new AttributeType();
089             attribute.Name = name;
090             attribute.AttributeValue = new object[] { value };
091
092             return attribute;
093         }
```

Fig. 15 - Afegir un atribut a una *SAML Assertion* amb .NET

3.3.3.2 Extensió de signatura

En .NET les accions de seguretat reben el nom d'Assertions. No s'han de confondre amb les SAML Assertions

El codi per afegir una signatura digital al missatge *SOAP* es troba íntegre a l'annex SignaturePolicyAssertion.cs

Aquest mòdul es divideix en dues parts:

1. Definició de l'*Assertion*

2. Filtre de signatura

3.3.3.2.1 *Definició de l'*Assertion

La següent figura detalla la classe que defineix l'*Assertion*

```
026     public class SignaturePolicyAssertion : SecurityPolicyAssertion
027     {
028         private X509SecurityToken taToken;

030         public X509SecurityToken X509Token
031         {
032             get { return taToken; }
033         }

040         public override SoapFilter CreateClientOutputFilter(FilterCreationContext
context)
041         {
042             return new SecuritySignatureFilter(this);
043         }

055         public override void ReadXml(System.Xml.XmlReader reader,
IDictionary<string, Type> extensions)
056         {
               ...

091             taToken = X509TokenProvider.CreateToken(storeLocation, storeName,
subjectDN);
092         }
093     }
```

Fig. 16 - Creació d'una *SignaturePolicyAssertion* amb .NET

La classe implementa [026] la *interface SecurityPolicyAssertion* i la seva funció és crear els filtres del missatge *SOAP*.

En el cas de l'exemple només interessa filtrar els missatges sortints de client per afegir signatura digital i per aquest motiu només crea el filtre en el mètode *CreateClientOutputFilter* [042]. Els altres mètodes han de retornar el valor *null* per a indicar que no filtren (veure l'annex SignaturePolicyAssertion.cs per a verificar com s'implementen la resta de mètodes no usats).

Hi ha un mètode que de no ser implementat es provocarà un error en temps d'execució. Aquest mètode és el [055] *ReadXml* i la seva funció és la de llegir els valor de configuració del filtre. En aquest exemple llegeix les credencials de l'aplicació. Com és una implementació simbòlica i en una aplicació real pot no ser així no es detalla la seva implementació (veure el seu codi íntegre a l'annex SignaturePolicyAssertion.cs per a detalls de la implementació particular per aquesta guia)

Filtre de signatura

La següent figura detalla el filtre de signatura:

```
095     public class SecuritySignatureFilter : SendSecurityFilter
096     {
097         private X509SecurityToken taToken;
098
099         public SecuritySignatureFilter(SignaturePolicyAssertion policyAssertion)
100             : base(policyAssertion.ServiceActor, true)
101         {
102             taToken = policyAssertion.X509Token;
103         }
104
105         public override void SecureMessage(SoapEnvelope envelope, Security
security)
```

```
106         {
107             security.Tokens.Add(taToken);
108 
109             MessageSignature signature = new MessageSignature(taToken);
110             signature.SignatureOptions = SignatureOptions.IncludeSoapBody;
111 
112             XmlNode contextHeader = findContextHeader(envelope);
113             if (contextHeader != null)
114             {
115                 SignatureReference reference = new SignatureReference("#" +
contextHeader.Attributes["Id", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"].Value);
116                 reference.AddTransform(new XmlDsigExcC14NTransform());
117 
118                 signature.AddReference(reference);
119             }
120 
121             security.Elements.Add(signature);
122         }
123 
124         private XmlNode findContextHeader(SoapEnvelope envelope)
125         {
126             foreach (XmlNode child in envelope.Header.ChildNodes)
127             {
128                 if (child != null && child.LocalName == "Context" &&
child.NamespaceURI == "http://soap.iarxiu/headers")
129                 {
130                     return child;
131                 }
132             }
133 
134             return null;
135         }
136     }
```

Fig. 17 - Filtre de signatura amb .NET

Els filtres de seguretat per a missatges sortints implementen la *interface SendSecurityFilter* [095].

En aquest exemple el seu constructor rep les credencials de l'aplicació [102] i la tasca de signar es realitza al implementar el mètode *SecureMessage* [105].

Les credencials de seguretat de l'aplicació s'afegeixen al missatge [107] i es crea una classe per a signar [109].

Com ja s'ha comentat la signatura ha d'incloure el cos del missatge –*Body*- i la capçalera *SAML*.

Per a signar el *Body* s'ha d'indicar amb la bandera *SignatureOptions.IncludeSoapBody* [110][10]. Ara bé per a signar la capçalera *SAML* s'ha d'afegir codi específic, tal i com es veu a la **fig.17** [112->119].

El procés per a signar la capçalera consisteix en localitzar la capçalera *SAML* (veure SamlHeader.cs) mitjançant el mètode d'ajuda *findContextHeader* [112]. Aquest mètode explora tots els elements de la capçalera fins trobar la capçalera *SAML* [126->134]. Si es localitza la capçalera s'afegeix a la signatura [115->118].

Finalment s'afegeix la signatura al missatge [121].

[10] Per defecte .NET afegeix a més un Timestamp

3.3.4 Configuració del client iArxiu .NET

Les aplicacions .NET es configuren mitjançant els arxius *App.config*[11]. El contingut dels arxius de configuració per a l'aplicació exemple es troben al annexos App.config i iArxiuClientPolicies.config.

3.3.4.1 Configuració de l'aplicació

La següent figura mostra l'arxiu de configuració (App.config)

```
001 <?xml version="1.0" encoding="utf-8" ?>
002 <configuration>
003   <configSections>
004     <section name="microsoft.web.services3"
type="Microsoft.Web.Services3.Configuration.WebServicesConfiguration,
Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
005   </configSections>
006   <system.web>
007     <webServices>
008       <soapExtensionTypes>
009         <add type="iArxiu.NETGuide.SamlSoapExtension, iArxiu.NETGuide"
priority="1" group="0"/>
010       </soapExtensionTypes>
011     </webServices>
012   </system.web>
013   <microsoft.web.services3>
014     <policy fileName="iArxiuClientPolicies.config"/>
015   </microsoft.web.services3>
016 </configuration>
```

Fig. 18 - Arxiu de configuració del client iArxiu .NET

L'arxiu per una banda indica que s'ha d'afegir una capçalera *SAML* [006->012]. A [009] s'indica el nom del mètode per a tractar la capçalera (totalment qualificat) i l'*Assembly* on es troba[12].

Per altra banda s'indica al paquet *WSE3* on es troba l'arxiu de definició de polítiques de seguretat [013->015].

Recordar que si s'executa aquesta aplicació des de l'entorn integrat, la ruta [014] hauria de ser "..\..\iArxiuClientPolicies.config"

La configuració de *WSE3* no funcionarà si no s'afegeix com primer element de l'arxiu la secció de configuració [003->005] tal i com es descriu a la **fig.18**[13].

3.3.4.2 Configuració de la seguretat

La següent figura mostra la configuració de seguretat (iArxiuClientPolicies.config)

```
001 <?xml version="1.0" encoding="utf-8" ?>
002 <policies>
003   <extensions>
004     <extension name="SignaturePolicyAssertion"
type="iArxiu.NETGuide.SignaturePolicyAssertion, iArxiu.NETGuide"/>
005   </extensions>
006   <policy name="CorePolicy">
```

[11] En el cas d'aplicacions ASP.NET l'arxiu s'anomena Web.config
[12] Per a més detalls sobre aquesta configuració consultar la documentació de Microsoft a http://msdn.microsoft.com/en-us/library/8f7skt4f(VS.80).aspx
[13] S'ha de copiar íntegrament el contingut tal i com apareix en aquesta guia

```
007     <SignaturePolicyAssertion>
008       <X509TokenProvider
009         storeLocation="CurrentUser"
010         storeName="My"
011         subjectDN="..."/>
012     </SignaturePolicyAssertion>
013   </policy>
014 </policies>
```

Fig. 19 - Configuració de seguretat a aplicacions .NET

En aquest arxiu[14] es defineixen polítiques [006]. Aquestes polítiques son invocades directament des del codi.

Les polítiques utilitzen les extensions definides a la secció d'extensions [003->005] i el conveni de declaració és mitjançant el nom de la classe completament qualificat i l'*Assembly* on es troba.

Per tant l'arxiu de configuració de seguretat defineix polítiques i associa aquestes polítiques amb extensions.

La associació es fa mitjançant el nom. Es comprendrà millor examinant l'exemple de la **fig.19**.

En aquest exemple es veu que es defineix una política amb el nom *CorePolicy* [006], la qual té configurada l'extensió *SignaturePolicyExtension* [007]. És a dir el nom de la política coincideix amb el nom del nus *Xml* i està relacionat amb l'extensió definida a [004] la qual té el mateix nom.

Dins la política [007] en el nus de l'extensió el contingut és lliure i és el que es llegeix des dels mètodes *ReadXml en els* codis de les *Assertion* (veure Definició de l'Assertion).

En aquest exemple únicament es configura un certificat a trobar en el magatzem segur de Windows (veure l'annex SamlSoapExtension.cs per identificar aquestes dades amb la metodologia per a capturar-les en els mètodes *ReadXml*)

[14] Per a més detalls sobre aquesta configuració consultar la documentació de Microsoft a http://msdn.microsoft.com/en-us/library/aa529254.aspx

4 Exemples de clients iArxiu

Per comprendre els exemples descrits en aquest capítol és imprescindible un coneixement mínim de les funcionalitats d'ingrés i difusió iArxiu

Aquest capítol descriurà codis exemple per a executar serveis iArxiu basant-se en les indicacions per a crear clients iArxiu descrits en el punt Creació de Programari Client. Hi ha clients d'exemple tant en Java com en .NET.

Els clients iArxiu d'exemple mostren la invocació de serveis iArxiu. Cada servei està implementat en una classe d'exemple aïllada per claredat. Aquestes classes s'invoquen des d'una classe principal anomenada *TestClient*, la qual prepara els clients i els invoca separadament.

Els serveis que s'invocaran seran els següents:

Funcionalitat	Serveis iArxiu involucrats
Ingrés de paquet petit	*Ingest*
Ingrés de paquet gran	*GetUploadTicket* *Upload* *OfflineIngest* *GetOfflineIngestStatus*
Difusió	*GetPackage*

Tot i que funcionalment són idèntics es fa distinció entre ingrés de paquets petits i grans. Com ja s'ha descrit en el punt Servei Upload d'aquesta guia ingressar paquets grans per missatgeria de serveis web pot ocasionar problemes de rendiment, i fins i tot esgotar la memòria, a les aplicacions client. Per aquest motiu iArxiu té opcions especials per a ingressar paquets grans.

Les funcionalitats d'ingrés de paquets petits i difusió no ofereixen dificultats addicionals, donat que únicament impliquen invocacions simples de serveis. No obstant la funcionalitat d'ingrés de paquets grans mereix una explicació acurada.

Ingrés de paquets grans:
Aquesta funcionalitat està dividida en tres etapes:

1. Obtenir un tiquet d'enviament

   S'executa el servei *GetUploadTicket* per a obtenir un identificador que s'usarà a la resta del procés

2. Enviament dels continguts fora de banda

   Utilitzant el tiquet obtingut en el pas anterior executar els enviaments dels continguts. Els enviaments es fan mitjançant peticions web *HTTP* utilitzant el mètode *POST/Multipart*.

3. Executar un ingrés desatès

   Utilitzant un altre cop el tiquet d'enviament s'executa un ingrés desatès. L'execució d'aquest servei retorna un identificador d'ingrés –*offlineticket*-. Si es desitja conèixer l'estat d'ingrés s'utilitzarà aquest identificador i el servei *GetOfflineIngestStatus*; en cas contrari el valor retornat es pot ignorar i es dona per acabat l'enviament.

Finalment indicar que el exemples no inclouen codi per a generar els paquets mets i es suposa que els paquets mets han estat generats mitjançant sistemes o utilitats alienes als clients. No obstant, en el cas de clients .NET, s'haurien de tenir en compte els aspectes referits a l'annex Filtrat de codificació de text a .NET

4.1 Exemples de clients iArxiu Java

La classe principal dels exemples és *TestClient.java*. Un fragment es mostra a la següent figura (el codi complet es troba a l'annex TestClient.java):

```
008 public class TestClient {

010     public static void main(String[] args) throws Exception {
011             ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
012
013             String packageId = null;
014
015             SmallMetsIngestClient smallMetsIngest =
(SmallMetsIngestClient)context.getBean("smallMetsIngest");
016             packageId = smallMetsIngest.ingest("Data\\mets.xml", false,
ContentTypeHandlingType.COMPLETE WITH INTROSPECTION);
017             System.out.println("SmallMetsIngest: " + packageId);

... Altres invocacions ...

026     }
027 }
```

Fig. 20 - Test del client iArxiu Java

La **fig.20** mostra com s'invoca el client per a ingressar paquets petits. En aquesta figura es veu com es carrega el context *Spring* [011]. D'aquest context s'extreuen els diferents clients com a *beans* de *Spring*. Com exemple el client per a ingressos petits (*SmallMetsIngestClient*) [015].

Per tant aquesta classe únicament té funcions d'inicialització i no es farà més menció a la mateixa. A continuació es descriuen els exemples de clients iArxiu Java específics.

Hi ha una classe d'utilitat destinada a facilitar les tasques de lectura i escriptura de paquets Mets anomenada MetsUtils. Aquesta classe és usada en els codis que es descriuen a continuació.
El codi íntegre es troba a l'annex MetsUtils.java.

4.1.1 Client iArxiu Java per a ingressos de paquets petits

La declaració del *bean* corresponent es troba a l'arxiu de context de l'aplicació (veure l'annex ApplicationContext.xml)

Per aquest cas d'exemple la declaració és la següent:

```
050     <bean id="smallMetsIngest" class="iarxiu.javaguide.SmallMetsIngestClient">
051             <property name="proxyClient" ref="proxyClient"/>
052     </bean>
```

Fig. 21 - Declaració del client iArxiu Java SmallMetsIngestClient

Com es veu a la declaració únicament s'afegeix la propietat *proxyClient* per accedir al *Core* iArxiu [051] (Per a detalls de configuració del client veure el punt Configuració del client iArxiu Java)

```
024     public String ingest(
025                     String metsFilename,
026                     boolean preservation,
027                     ContentTypeHandlingType.Enum contentTypeHandling
028                     ) throws Exception {
029
030             String packageId = null;
031
032             File metsFile = new File(metsFilename);
033
034             Mets mets = MetsUtils.load(metsFile);
035
036             IngestRequestDocument requestDocument =
IngestRequestDocument.Factory.newInstance();
037             IngestRequest request = requestDocument.addNewIngestRequest();
038
039             request.setPreservation(preservation);
040             request.setContentTypeHandling(contentTypeHandling);
041             request.setMets( mets );
042
043             IngestResponseDocument responseDocument =
getProxyClient().send(requestDocument);
044             packageId =
responseDocument.getIngestResponse().getIngestInfo().getId();
045
046             return packageId;
047     }
```

Fig. 22 - Client iArixu Java SmallMetsIngestClient

No ofereix cap dificultat el codi presentat a la **fig.22**. El mètode *ingest* admet els paràmetres possibles per a executar un ingrés:

- Nom del paquet (*metsFilename*)
- Si es desitja o no preservació
- Com ha d'actuar el *Core* pel que respecta al tipus de contingut

El primer que es fa és crear una referència a l'arxiu [032] i a continuació es llegeix el paquet com un objecte *Mets* [034].

Es crea una petició [036->041] del tipus *Ingest* i finalment s'invoca [043].

De la resposta retornada pel *Core* [044] s'obté la identificació del paquet en la variable *packageId* i aquest valor es retorna.

4.1.2 Client iArxiu Java per a ingressos de paquets grans

Aquest és el client més complex presentat en aquesta guia. Funcionalment és equivalent al cas descrit en el punt anterior (Client iArxiu Java per a ingressos de paquets petits (SmallMetsIngestClient)) però amb grans diferències internes.

Per una banda la declaració del *bean* de *Spring* conté l'afegit de la propietat *uploadServiceUrl*, la qual conté l'adreça del servei *upload*.

El servei *upload* es troba a: *http://<server>:<port>/core/servlet/upload*, on *<server>* i *<port>* corresponen a cada cas particular. La resta de la *url* es manté igual per a totes les instal·lacions.

```
054     <bean id="bigMetsIngest" class="iarxiu.javaguide.BigMetsIngestClient">
055             <property name="proxyClient" ref="proxyClient"/>
```

```
056           <property name="uploadServiceUrl"
value="http://<server>:<port>/core/servlet/upload"/>
057     </bean>
```

Fig. 23 - Declaració del client iArxiu Java BigMetsIngestClient

Les diferències més grans es troben a la implementació. Com es veu a la següent figura, el codi està dividit en vàries parts ben diferenciades, seguint el ja descrit en el començament d'aquest capítol:

```
045     public String ingest(
046                     String metsFilename,
047                     boolean preservation,
048                     ContentTypeHandlingType.Enum contentTypeHandling
049                     )
050     throws
051             HttpException,
052             IOException {
053
054             String packageId = null;
055
056             String uploadTicket = getUploadTicket();
057
058             upload(uploadTicket, metsFilename);
059
060             String offlineTicket = offlineIngest(uploadTicket, preservation,
contentTypeHandling);
061
062             packageId = waitOffline(offlineTicket);
063
064             return packageId;
065     }
```

Fig. 24 – Client iArxiu Java BigMetsIngestClient

Primerament s'obté un tiquet de càrrega [056], el qual s'utilitza per a fer la càrrega del paquet [058].

En una aplicació real a més del paquet s'haurien de carregar altres continguts. En aquest exemple únicament es suposa un paquet simple

A continuació es fa un ingrés en modalitat *offline* i s'obté el tiquet d'ingrés a la variable *offlineTicket* [060]. Si no es desitja seguir l'estat de 'ingrés es podria retornar en aquest punt. L'exemple invoca al mètode per a seguir l'estat *waitOffline* [062]. El valor retornat per aquest mètode és la identificació del paquet ingressat, el qual s'emmagatzema a la variable *packageId* i es retorna.

4.1.2.1 Obtenció del tiquet de càrrega

```
067     private String getUploadTicket() {
068             String ticket = null;
069
070             GetUploadTicketRequestDocument requestDocument =
GetUploadTicketRequestDocument.Factory.newInstance();
071             requestDocument.addNewGetUploadTicketRequest();
072
073             GetUploadTicketResponseDocument responseDocument =
getProxyClient().send(requestDocument);
074             ticket = responseDocument.getGetUploadTicketResponse().getTicket();
075
076             return ticket;
077     }
```

Fig. 25 - Client iArxiu Java BigMetsIngestClient (Obtenció del tiquet de càrrega)

S'invoca el servei *GetUploadTicket* creant la petició [070->071]. Aquest servei no té paràmetres d'entrada. En el *Core* iArxiu els paràmetres necessaris són l'ens i el fons, els quals s'obtenen de les capçaleres de seguretat del propi missatge.

Finalment s'invoca el servei [073] i el valor retornat [074] correspon a tiquet de càrrega, el qual es retorna com a resposta del mètode.

4.1.2.2 Càrrega del paquet i binaris

Consisteix en carregar el paquet –*Mets*- indicant el tiquet de càrrega obtingut en el pas anterior. La càrrega es fa mitjançant petició web i mètode *POST/multipart*. A la petició els paràmetres obligatoris són el tiquet de càrrega i el paquet amb els següents noms:

- Tiquet de càrrega : *ticket*
- Paquet : *mets*

Si el paquet conté binaris addicionals s'han d'afegir a la petició com a nous paràmetres. Cada nou binari ha de tenir com a nom de paràmetre el nom d'arxiu segons correspongui a la referència declarada en el paquet[15].

```
079     private void upload(String ticket, String metsFilename)
080     throws
081             HttpException,
082             IOException {
083
084             File metsFile = new File(metsFilename);
085
086             HttpClient httpClient = new HttpClient();
087
088             PostMethod post = new PostMethod(getUploadServiceUrl());
089
090             Part ticketPart = new StringPart("ticket", ticket);
091             Part metsPart = new FilePart("mets", metsFile);
092
093             Part[] parts = new Part[]{
094                             ticketPart,
095                             metsPart
096             };
097
098             post.setRequestEntity( new
MultipartRequestEntity(parts,post.getParams()) );
099
100             httpClient.executeMethod(post);
101     }
```

Fig. 26 - Client iArxiu Java BigMetsIngestClient (Càrrega del paquet i binaris)

Com es veu a la **fig.26** s'han utilitzat les classes *HttpClient* d'*Apache* com a clients web, perquè faciliten molt la tasca, especialment en aquest cas d'enviaments *multipart*.

[086->098] creen el client i inicialitzen el mètode d'enviament –*Post*-. [090->096] creen les parts. La primera part es del tipus *String* i correspon al paràmetre *ticket*. La segona part correspon al paràmetre *mets* i és del tipus *File*. Aquestes dues parts són obligatòries. En un cas real, i amb binaris separats del paquet, s'haurien d'afegir tantes parts com binaris externs. Cada binari correspondria a una part del tipus *File* i amb nom de paràmetre el nom de l'arxiu segons la referència en el paquet.

Les parts es combinen en un matriu [093] i es passen al mètode [098] abans de ser executat [100].

[15] Aquesta guia no mostra com carregar binaris addicionals i únicament mostra com carregar un paquet simple

En una aplicació real s'hauria de verificar el codi de retorn de [100] i actuar en conseqüència en el cas d'error

4.1.2.3 Execució d'ingrés desatès

Un cop carregats el paquets i els binaris, si n'hi ha, s'ha d'invocar el servei ingrés desatès –*offline*-.

```
103     private String offlineIngest(
104                     String uploadTicket,
105                     boolean preservation,
106                     ContentTypeHandlingType.Enum contentTypeHandling) {
107
108             String offlineTicket = null;
109
110             OfflineUploadIngestRequestDocument requestDocument =
OfflineUploadIngestRequestDocument.Factory.newInstance();
111             OfflineUploadIngestRequest request =
requestDocument.addNewOfflineUploadIngestRequest();
112
113             request.setPreservation(preservation);
114             request.setContentTypeHandling(contentTypeHandling);
115             request.setUploadTicket(uploadTicket);
116
117             OfflineUploadIngestResponseDocument responseDocument =
getProxyClient().send(requestDocument);
118             offlineTicket = responseDocument.getOfflineUploadIngestResponse();
119
120             return offlineTicket;
121     }
```

Fig. 27 - Client iArxiu Java BigMetsIngestClient (Execució d'ingrés desatés)

Com es pot veure a la **fig.27** els paràmetres d'entrada corresponen als paràmetres d'un ingrés però on el paquet s'ha canviat pel tiquet de càrrega –*uploadTicket*-.

Aquesta part del procés consisteix en crear una petició *OfflineUploadIngest* [110->115] i invocar-la [117].

El valor retornat [118] correspon al tiquet de ingrés, el qual s'emmagatzema a la variable *offlineTicket* i es retorna.

4.1.2.4 Seguiment de l'estat d'ingrés

Com ja s'ha comentat aquesta part és opcional, però aquest guia la mostra com a exemple.

```
123     private String waitOffline(String offlineTicket) {
124             String packageId = null;
125
126             OfflineIngestInfoType ingestInfo;
127             while(true) {
128                     GetOfflineIngestStatusRequestDocument requestDocument =
GetOfflineIngestStatusRequestDocument.Factory.newInstance();
129
        requestDocument.setGetOfflineIngestStatusRequest(offlineTicket);
130
131                     GetOfflineIngestStatusResponseDocument responseDocument =
getProxyClient().send(requestDocument);
132                     ingestInfo =
responseDocument.getGetOfflineIngestStatusResponse().getOfflineIngestInfo();
133
134                     if
(!ingestInfo.getStatus().equals(OfflineIngestInfoType.Status.IN_PROCESS))
```

```
135                             break;
136
137                     try { Thread.sleep(1000);}
138                     catch (InterruptedException e) {
139                             throw new RuntimeException("Error in sleep", e);
140                     }
141             }
142
143             if (ingestInfo.getStatus().equals(OfflineIngestInfoType.Status.ERROR))
144                     throw new RuntimeException("Offline error: " +
ingestInfo.getErrorCode());
145
146             if
(ingestInfo.getStatus().equals(OfflineIngestInfoType.Status.UNKNOWN))
147                     throw new RuntimeException("Unknown offline error: " +
ingestInfo.getErrorCode());
148
149             packageId = ingestInfo.getId();
150
151             return packageId;
152     }
```

Fig. 28 - Cilent iArxiu Java BigMetsIngestClient (Seguiment de l'estat d'ingrés)

El codi presentat a la fig.28 llença excepcions del tipus Runtime. En una aplicació real aquesta no seria una bona pràctica i correspondria llençar excepcions pròpies de l'aplicació

En el bucle [127->141] s'executen peticions de consulta amb el servei *GetOfflineIngestStatus*.

Quan l'estat és diferent a en procés –*IN_PROCESS*- [138] s'abandona el bucle. En cas contrari es torna a reintentat després d'una pausa, que en aquest exemple és de 1 segon [137->140]

A [143] i [146] es verifica si l'estat no és ingrés correcte –*OK*- per generar una excepció. En el cas que el paquet ha estat ingressat correctament s'obté el valor de identificació del paquet [149] i es retorna.

4.1.3 Client iArxiu Java per a difusió de paquets

La declaració del *bean* és la següent::

```
059     <bean id="disseminationClient" class="iarxiu.javaguide.DisseminationClient">
060             <property name="proxyClient" ref="proxyClient"/>
061     </bean>
```

Fig. 29 - Declaració del client iArxiu Java DisseminationClient

Com es veu de la mateixa manera que la resta de cassos a més de la declaració únicament es necessita la injecció de la propietat *proxyClient* (Per a més detalls de configuració del client veure el punt Configuració del client iArxiu Java)

```
028     public void download(
029                     String metsFilename,
030                     String packageId,
031                     boolean includeBinaries,
032                     boolean includeDC)
033             throws
034                     IOException,
035                     XmlException,
036                     TransformerException {
```

```
037
038             File metsFile = new File(metsFilename);
039
040             GetPackageRequestDocument requestDocument =
GetPackageRequestDocument.Factory.newInstance();
041             GetPackageRequest request = requestDocument.addNewGetPackageRequest();
042
043             request.setPackageId(packageId);
044             request.setIncludeBinaries(includeBinaries);
045             request.setIncludeDC(includeDC);
046
047             GetPackageResponseDocument responseDocument =
getProxyClient().send(requestDocument);
048             Mets mets = responseDocument.getGetPackageResponse().getMets();
049
050             MetsUtils.save(mets, metsFile);
051     }
```

Fig. 30 - Client iArxiu Java DisseminationClient

Es tracta d'un altre cas molt simple. El mètode *download* admet els paràmetres:

- Nom de l'arxiu de paquet (*metsFilename*)
- Identificació del paquet (*packageId*)
- Si es desitja que el paquet contingui també els binaris o únicament metadada (*includeBinaries*)
- Si es desitja que el paquet contingui metadada en format *Dublin Core* (*includeDC*)

[040->045] crea la petició *GetPackage* i s'invoca a [047]. De la resposta es pot extreure el paquet [048], que en aquest cas s'emmagatzema en el destí [050] segons el paràmetre d'entrada.

En una aplicació real el paquet obtingut a [048] seria processat, però aquest exemple únicament mostra com salvar-ho

4.2 Exemples de clients iArxiu .NET

La classe principal dels exemples és *TestClient.cs*. Un fragment es mostra a la següent figura (el codi complet es troba a l'annex TestClient.cs):

```
011     class TestClient
012     {
013         static void Main(string[] args)
014         {
015             string packageId = null;
016
017             IngestService ingestProxy = new IngestService();
018             ingestProxy.Url = "http://localhost:8080/core/soap";
019             ingestProxy.SetPolicy("CorePolicy");
020
021             DisseminationService disseminationProxy = new DisseminationService();
022             disseminationProxy.Url = "http://localhost:8080/core/soap";
023             disseminationProxy.SetPolicy("CorePolicy");
024
025             packageId = SmallMetsIngest(ingestProxy, @"Data\mets.xml", false,
contentTypeHandlingType.completeWithIntrospection);
026             Console.WriteLine("SmallMetsIngest: {0}", packageId);

... Altres invocacions ...

033         }
034
```

```
035         private static string SmallMetsIngest(IngestService ingestProxy, string
filename, bool preservation, contentTypeHandlingType contentTypeHandling)
036         {

... Es descriu en altres punts d'aquesta guia ...

045         }

067     }
```

Fig. 31 - Test del client iArxiu .NET

La **fig.31** mostra com s'invoca el client per a ingressar paquets petits. En aquesta figura es veu com es creen els *proxies* dels serveis iArxiu (consultar els punts de seguretat en aquesta guia a Configuració de la seguretat)

També s'observa a la figura que es declara un mètode *SmallMetsIngest*. Aquest mètode, i altres similars, es descriuen posteriorment i la seva finalitat és preparar la invocació al client.

Hi ha una classe d'utilitat destinada a facilitar les tasques de lectura i escriptura de paquets Mets anomenada MetsUtils. Aquesta classe és usada en els codis que es descriuen a continuació.
El codi íntegre es troba a l'annex MetsUtils.cs.
Segons es pot veure en dit annex la classe està implementada seguint les convencions Genèrics de C#/.NET.
El motiu és perquè la estructura Mets està definida en els dos descriptors de servei – ingest.wsdl i dissemination.wsdl- i .NET genera dues implementacions de Mets idèntiques en codi però qualificades de manera diferent.
Com existeixen dues classes Mets idèntiques però en diferents espais de noms aquesta classe s'ha implementat amb Genèrics i d'aquesta manera un únic codi pot tractar les dues classes

4.2.1 Client iArxiu .NET per a ingressos de paquets petits

La classe *TestClient* prepara la invocació del client amb el mètode

```
032         private static string SmallMetsIngest(IngestService ingestProxy, string
filename, bool preservation, contentTypeHandlingType contentTypeHandling)
033         {
034             string packageId = null;
035
036             SmallMetsIngestClient smallMetsIngest = new SmallMetsIngestClient();
037             smallMetsIngest.Proxy = ingestProxy;
038
039             packageId = smallMetsIngest.Ingest(filename, preservation,
contentTypeHandling);
040
041             return packageId;
042         }
```

Fig. 32 - Preparació del client iArxiu .NET SmallMetsIngestCient

Aquest mètode crea un client *SmallMetsIngestClient* [036] i el configura amb el proxy de servei [037].

Finalment invoca el seu mètode *Ingest* per a ingressar el paquet [039].

```
014         public string Ingest(string metsFilename, bool preservation,
contentTypeHandlingType contentTypeHandling)
015         {
016             String packageId = null;
```

```
017
018             mets mets = MetsUtils<mets>.Load(metsFilename);
019
020             IngestRequest request = new IngestRequest();
021             request.mets = mets;
022             request.preservation = false;
023             request.contentTypeHandling = contentTypeHandlingType.checkAndReject;
024
025             IngestResponse response = Proxy.Ingest(request);
026             packageId = response.ingestInfo.id;
027
028             return packageId;
029         }
```

Fig. 33 - Client iArxiu .NET SmallMetsIngestClient

No ofereix cap dificultat el codi presentat a la **fig.33**. El mètode *ingest* admet els paràmetres possibles per a executar un ingrés:

- Nom del paquet (metsFilename)
- Si es desitja o no preservació
- Com ha d'actuar el Core pel que respecta al tipus de contingut

A [018] es llegeix el paquet i es crea una petició del tipus *Ingest* [020->023] que s'invoca a [025].

De la resposta retornada pel *Core* [026] s'obté la identificació del paquet en la variable *packageId* i aquest valor es retorna.

4.2.2 Client iArxiu .NET per a ingressos de paquets grans

Aquest és el paquet més complex presentat en aquesta guia. Funcionalment és equivalent al cas descrit en el punt anterior (Client iArxiu .NET per a ingressos de paquets petits) però amb grans diferències internes.

Per una banda la preparació de la invocació conté l'afegit de la propietat *UploadServiceUrl*, la qual conté l'adreça del servei *upload*.

El servei *upload* es troba a *http://<server>:<port>/core/servlet/upload*, on *<server>* i *<port>* corresponen a cada cas particular. La resta de la *url* es manté igual per a totes les instal·lacions.

```
044         private static string BigMetsIngest(IngestService ingestProxy, string
filename,  bool preservation, contentTypeHandlingType contentTypeHandling)
045         {
046             string packageId = null;
047
048             BigMetsIngestClient bigMetsIngest = new BigMetsIngestClient();
049             bigMetsIngest.Proxy = ingestProxy;
050             bigMetsIngest.UploadServiceUrl =
"http://<server>:<port>/core/servlet/upload";
051
052             packageId = bigMetsIngest.ingest(filename, preservation,
contentTypeHandling);
053
054             return packageId;
055         }
```

Fig. 34 - Preparació del client iArxiu .NET BigMetsIngestClient

Les diferències més grans es troben a la implementació. Com es veu a la següent figura, el codi està dividit en vàries parts be diferenciades, seguint el ja descrit en el començament d'aquest capítol (Exemples de clients iArxiu) de les quals la principal és la següent:

```
017         public string ingest(string metsFilename, bool preservation,
contentTypeHandlingType contentTypeHandling)
018         {
019             string packageId = null;
020
021             string uploadTicket = GetUploadTicket();
022
023             Upload(uploadTicket, metsFilename);
024
025             string offlineTicket = OfflineIngest(uploadTicket, preservation,
contentTypeHandling);
026
027             packageId = WaitOffline(offlineTicket);
028
029             return packageId;
030         }
```

Fig. 35 - Client iArxiu .NET BigMetsIngestClient

Primerament s'obté un tiquet de càrrega [021], el qual s'utilitza [023] per a fer la càrrega del paquet.

En una aplicació real a més del paquet s'haurien de carregar altres continguts. En aquest exemple únicament es suposa un paquet simple

A continuació es fa un ingrés en modalitat *offline* i s'obté el tiquet d'ingrés a la variable *offlineTicket* [025]. Si no es desitja seguir l'estat de 'ingrés es podria retornar en aquest punt. L'exemple invoca al mètode per a seguir l'estat *waitOffline* [063]. El valor retornat per aquest mètode és la identificació del paquet ingressat, el qual s'emmagatzema a la variable *packageId* [027] i es retorna.

4.2.2.1 Obtenció del tiquet de càrrega

```
032         private string GetUploadTicket()
033         {
034             string ticket = null;
035
036             GetUploadTicketRequest request = new GetUploadTicketRequest();
037
038             GetUploadTicketResponse response = Proxy.GetUploadTicket(request);
039             ticket = response.ticket;
040
041             return ticket;
042         }
```

Fig. 36 - Client iArxiu .NET BigMetsIngestClient (Obtenció del tiquet de càrrega)

S'invoca el servei *GetUploadTicket* creant la petició [036]. Aquest servei no té paràmetres d'entrada. En el *Core* iArxiu els paràmetres necessaris són l'ens i el fons, els quals s'obtenen de les capçaleres de seguretat del propi missatge.

Finalment s'invoca el servei [038] i el valor retornat [039] correspon a tiquet de càrrega, el qual es retorna com a resposta del mètode.

4.2.2.2 Càrrega del paquet i binaris

Consisteix en carregar el paquet –*Mets*- indicant el tiquet de càrrega obtingut en el pas anterior. La càrrega es fa mitjançant petició web i mètode *POST/multipart*. A la petició els paràmetres obligatoris són el tiquet de càrrega i el paquet amb els següents noms:

- Tiquet de càrrega: *ticket*
- Paquet: *mets*

Si el paquet conté binaris addicionals s'han d'afegir a la petició com a nous paràmetres. Cada nou binari ha de tenir com a nom de paràmetre el nom d'arxiu segons correspongui a la referència declarada en el paquet[16].

```
044         private void Upload(string uploadTicket, string metsFilename)
045         {
046             HttpMultiPartClient httpClient = new HttpMultiPartClient();
047             httpClient.Url = UploadServiceUrl;
048 
049             httpClient.Parts.Add(new HttpFieldPart("ticket", uploadTicket));
050             httpClient.Parts.Add(new HttpFilePart("mets", metsFilename));
051 
052             httpClient.Send();
053         }
```

Fig. 37 - Client iArxiu .NET BigMetsIngestClient (Càrrega del paquet i binaris)

Aquest mètode utilitza la classe *HttpMultipartClient*. Aquesta classe no forma part del .NET Framework i es troba en aquesta guia com a funció d'utilitat. Bàsicament és un client web que utilitza mètode *POST* i *multipart*. Per a detalls d'implementació veure l'annex HttpClient.cs i per a detalls sobre el protocol *multipart* veure l'annex POST/Multipart.

[046->047] creen i configuren el client web i a [049->050] s'afegeixen les parts. La primera part és del tipus *String* i correspon al paràmetre *ticket*. La segona part correspon al paràmetre *mets* i és del tipus *File*. Aquestes dues parts són obligatòries. En un cas real, i amb binaris separats del paquet, s'haurien d'afegir tantes parts com binaris externs. Cada binari correspondria a una part del tipus *File* i amb nopm de paràmetres el nom de l'arxiu segons la referència en el paquet.

Finalment el mètode s'executa a [052].

En una aplicació real s'hauria de verificar el codi de retorn de [052] i actuar en conseqüència en el cas d'error.

4.2.2.3 Execució d'ingrés desatès

Un cop carregats el paquets i els binaris, si n'hi ha, s'ha d'invocar el servei ingrés desatès –*offline*-

```
055         private string OfflineIngest(string uploadTicket, bool preservation,
contentTypeHandlingType contentTypeHandling)
056         {
057             string offlineTicket = null;
058 
059             OfflineUploadIngestRequest request = new
OfflineUploadIngestRequest();
060             request.preservation = false;
061             request.contentTypeHandling = contentTypeHandling;
```

[16] Aquesta guia no mostra com carregar binaris addicionals i únicament mostra com carregar un paquet simple

```
062             request.uploadTicket = uploadTicket;
063
064             offlineTicket = Proxy.OfflineUploadIngest(request);
065
066             return offlineTicket;
067         }
```

Fig. 38 - Client iArxiu .NET BigMetsIngestClient (Execució d'ingrés desatés)

Com es pot veure a la **fig.38** els paràmetres d'entrada corresponen als paràmetres d'un ingrés però on el paquet s'ha canviat pel tiquet de càrrega –*uploadTicket*-.
Aquesta part del procés consisteix en crear una petició *OfflineUploadIngest* [059->062] i invocar-la [064].

El valor retornat [064] correspon al tiquet de ingrés, el qual s'emmagatzema a la variable *offlineTicket* i es retorna.

4.2.2.4 Seguiment de l'estat d'ingrés

Com ja s'ha comentat aquesta part és opcional, però aquest guia la mostra com a exemple.

```
069         public string WaitOffline(string offlineTicket)
070         {
071             string packageId = null;
072
073             offlineIngestInfoType ingestInfo;
074             while (true)
075             {
076                 GetOfflineIngestStatusResponse response =
Proxy.GetOfflineIngestStatus(offlineTicket);
077                 ingestInfo = response.offlineIngestInfo;
078
079                 if (ingestInfo.status != offlineIngestInfoTypeStatus.inProcess)
080                     break;
081
082                 Thread.Sleep(1000);
083             }
084
085             switch (ingestInfo.status)
086             {
087                 case offlineIngestInfoTypeStatus.error:
088                     throw new ApplicationException(ingestInfo.errorCode);
089
090                 case offlineIngestInfoTypeStatus.unknown:
091                     throw new ApplicationException("Unknown status");
092             }
093
094             packageId = ingestInfo.id;
095
096             return packageId;
097         }
098     }
```

Fig. 39 - Client iArxiu .NET BigMetsIngestClient (Seguiment de l'estat d'ingrés)

El codi presentat a la fig.39 llença excepcions del tipus ApplicationException. En una aplicació real aquesta no seria una bona pràctica i correspondria llençar excepcions pròpies de l'aplicació

En el bucle [074->083] s'executen peticions de consulta amb el servei *GetOfflineIngestStatus*.

Quan l'estat és diferent a en procés –*IN_PROCESS*- [079] s'abandona el bucle. En cas contrari es torna a reintentat després d'una pausa, que en aquest exemple és de 1 segon [082]

A [085->092] es verifica si l'estat no és ingrés correcte –*OK*- per generar una excepció. En el cas que el paquet ha estat ingressat correctament s'obté el valor de identificació del paquet [094] i es retorna.

4.2.3 Client iArxiu .NET per a difusió de paquets

La classes *TestClient* prepara l'execució amb el mètode:

```
057         private static void Download(DisseminationService disseminationProxy,
string filename, string packageId, bool includeBinaries, bool includeDC)
058         {
059             DisseminationClient dissemination = new DisseminationClient();
060             dissemination.Proxy = disseminationProxy;
061
062             dissemination.Download(filename, packageId, includeBinaries,
includeDC);
063         }
```

Fig. 40 - Preparació del client iArxiu .NET DisseminationClient

Com es veu de la mateixa manera que la resta de cassos a més de la declaració únicament es necessita la injecció de la propietat *proxy*

```
014         public void Download(string metsFilename, string packageId, bool
includeBinaries, bool includeDC)
015         {
016             GetPackageRequest request = new GetPackageRequest();
017             request.packageId = packageId;
018             request.includeBinaries = includeBinaries;
019             request.includeDC = includeDC;
020
021             GetPackageResponse response = Proxy.GetPackage(request);
022             mets metsDocument = response.mets;
023
024             MetsUtils<mets>.Save(metsDocument, metsFilename);
025         }
```

Fig. 41 - Client iArxiu .NET DisseminationClient

Es tracta d'un altre cas molt simple. El mètode *download* admet els paràmetres:

- Nom de l'arxiu de paquet (*metsFilename*)
- Identificació del paquet (*packageId*)
- Si es desitja que el paquet contingui també els binaris o únicament metadada (*includeBinaries*)
- Si es desitja que el paquet contingui metadada en format *Dublin Core* (*includeDC*)

[016->0019] crea la petició *GetPackage* i s'invoca a [021]. De la resposta es pot extreure el paquet [022], que en aquest cas s'emmagatzema en el destí [024] segons el paràmetre d'entrada.

En una aplicació real el paquet obtingut a [022] seria processat, però aquest exemple únicament mostra com salvar-ho.

Nota important:

La serialització / deserialització d'objectes a .NET provoca que en el procés de conversió de documents XML al seu model d'objectes –deserialització- i posterior transformació a document XML –serialització- es perdin atributs com esquemes de noms o l'atribut schemaLocation. A més en el document XML generat en el procés de serialització apareix sempre una referència a l'esquema de noms de esquemes (xmlns:xsd=http://www.w3.org/2001/XMLSchema).
Per tant si els paquets són descarregats i s'emmagatzemen a disc per a ser validats contra els segells iArxiu, la validació serà errònia.
Si es desitja validar un paquet contra el seu segell iArxiu s'hauran d'aplicar les correccions manuals a les definicions d'espais de noms en el paquet o bé descarregar-lo utilitzant un client Java o des de la Web iArxiu.
Aquest comportament és imputable al framework .NET, la solució del qual queda fora de l'abast d'aquesta guia.

5 Annexes

5.1 Missatge SOAP

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
       <SOAP-ENV:Header>
              <wsse:Security
                     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd"
                     SOAP-ENV:mustUnderstand="1">
                     <wsu:Timestamp
                            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd"
                            wsu:Id="Timestamp-11143287">
                            <wsu:Created
                                   xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-05-
07T14:33:56.016Z
                            </wsu:Created>
                            <wsu:Expires
                                   xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-05-
07T14:38:56.016Z
                            </wsu:Expires>
                     </wsu:Timestamp>
                     <wsse:BinarySecurityToken
                            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd"
                            EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
                            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3"
                            wsu:Id="CertId-97118725"
                            xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

       MIIIaTCCB1GgAwIBAgIQIix17XF/dINIYe/OI9gZSjANBgkqhkiG9w0BAQUFADCCAT4xCzAJBgNVBA
YTAkVTMTswOQYDVQQKEzJBZ2VuY2lhIENhdGFsYW5hIGRlIENlcnRpZmljYWNpbyAoTklGIFEtMDgwMTE3Ni1
JKTE0MDIGA1UEBxMrUGFzc2F0Z2UgZGUgbGEgQ29uY2VwY2lvIDExIDA4MDA4IEJhcmNlbG9uYTEuMCwGA1UE
CxMlU2VydmVpcyBQdWJsaWNzIGRlIENlcnRpZmljYWNpbyBFQ1YtMjE3MDUGA1UECxMuVmVnZXUgaHR0cHM6L
y93d3cuY2F0Y2VydC5uZXQvdmVyQ0lDLTIgICAoYykwMzE0MDIGA1UECxMrU2VjcmV0YXJpYSBkJ0FkbWluaX
N0cmFjaW8gaSBGdW5jaW8gUHVibGljYTEdMBsGA1UEAxMUUFJFUFJPRFVDQ0lPIEVDLVNBRlAwHhcNMDgwNjI
1MDcxMjE0WhcNMTIwNjI1MDcxMTUxWjCCAWAxCzAJBgNVBAYTAkVTMQ8wDQYDVQQKEwZTQURJRUwxITAfBgNV
BAsUGMByZWEgZCdBdGVuY2nzIGFsIENsaWVudDE7MDkGA1UECxMyU2VydmVpcyBQdWJsaWNzIGRlIENlcnRpZ
mljYWNpbyBDUElTTUi0xIGFtYiBjYXJyZWMxQTA/BgNVBAsTOFZlZ2V1IGh0dHBzOi8vd3d3LmNhdGNlcnQubm
V0L3ZlckNQSVNSLTFDYXJyZWNTQUZQICAhjKTA1MRwwGgYDVQQEExNCYXJiYXJpYXMgQmVyZWNpYmFyMQ4wDAY
DVQQqEwVKb2tpbjEtMCsGA1UEDBQkQ29udGludWFkb3IgUHJvY2VkaW1lbnQgQ29udHJhY3RhY2nzMRIwEAYD
VQQFEwk0NDMzMDMxM2sxLDAqBgNVBAMTI0NQSVNSLTEgQyBKb2tpbiBCYXJiYXJpYXMgQmVyZWNpYmFyMIGfM
A0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCssAOt0/wfYd76pECg46ax3nQuiYK+v80jGsLxBucnt+50n9PhCN
n9RuZyA7aaHPlbPiEFYfv+QS4p7RRgTF5HKqTAlDYd2dIKixlRtItt2eK5uoJWBaeY5TdbGyN0vbX3Eeu8f3u
PUFB842SArgmcS2S8DAnae5k6ijuSvA1s+QIDAQABo4IDvzCCA7swNwYDVR0RBDAwLoEUamJhcmJhcmlhc0Bz
YWRpZWwuZXOkFjAUMRIwEAYDVQQFEwlRNTg1NjMzOEgwDgYDVR0PAQH/BAQDAgeAMCkGA1UdJQQiMCAGCCsGA
QUFBwMCBggrBgEFBQcDBAYKKwYBBAGCNxQCAjARBglghkgBhvhCAQEEBAMCBaAwHQYDVR0OBBYEFNChzT5zZF
ewXXzBe885F6hpaOdaMIIBPQYDVR0jBIIBNDCCATCAFGEOl4W5fdvYBUG3jeIwE16uV3DeoYIBBKSCAQAwgf0
xCzAJBgNVBAYTAkVTMTswOQYDVQQKEzJBZ2VuY2lhIENhdGFsYW5hIGRlIENlcnRpZmljYWNpbyAoTklGIFEt
MDgwMTE3Ni1JKTEuMCwGA1UECxM1U2VydmVpcyBQdWJsaWNzIGRlIENlcnRpZmljYWNpbyBFQ1YtMTE9MDsGA
1UECxM0VmVnZXUgaHR0cHM6Ly93d3cuY2F0Y2VydC5uZXQvdmVycHJlcHJvZHVjY2lvICAoYykwMzEhMB8GA1
UECxMYR2VuZXJhbGl0YXQgZGUgQ2F0YWx1bnlhMR8wHQYDVQQDExZQUkVQUk9EVUNDSU8gRUMtR0VOQ0FUghA
DaJkdPATMZ0U90npwhwqNMIIBCwYDVR0gBIIBAjCB/zCB/AYNKwYBBAH1eAEDAVECBDCB6jA4BggrBgEFBQcC
ARYsaHR0cHM6Ly93d3cuY2F0Y2VydC5uZXQvdmVyQ1BJU1ItMUNhcnJlY1NBRlAwga0GCCsGAQUFBwICMIGgG
oGdQXF1ZXN0IOlzIHVuIGNlcnRpZmljYXQgcGVyc29uYWwgcmVjb25lZ3V0IGRcJ2lkZW50aWZpY2FjafMgaS
BzaWduYXR1cmEgcmVjb25lZ3VkYSBkZSBjbGFzc2UgMSBhbWIgY+BycmVjLiBWZWdldSBodHRwczovL3d3dy5
jYXRjZXJ0Lm5ldC92ZXJDUElTTui0xQ2FycmVjU0FGUDAzBggrBgEFBQcBAQQnMCUwIwYIKwYBBQUHMAGGF2h0
dHA6Ly9vY3NwLmNhdGNlcnQubmV0MBgGCCsGAQUFBwEDBAwwCjAIBgYEAI5GAQEwdAYDVR0fBG0wazBpoGegZ
YYwaHR0cDovL2Vwc2NkLmNhdGNlcnQubmV0L2NybC9wcmVwcm9kX2VjLXNhZnAuY3JshjFodHRwOi8vZXBzY2
QyLmNhdGNlcnQubmV0L2NybC9wcmVwcm9kX2VjLXNhZnAuY3JsMA0GCSqGSIb3DQEBBQUAA4IBAQCmTHTmMDF
NCxpl4VKumtiT2jrysAToYwcoCMOsLg8Jmw5d/zJxcG+Dcrcm/7mEoh1cBxklQp9ObJXBtN++ZYfBeAVaxxt+
SggW1uR/7XLXAcNtF2Rc1hTAuD6liCZc14MSXn7+gRDRV7ZOqUkHZkDeCF/99WmarJO+uNdMbR5fWJNuwCJk4
```

```
J85hDP9XnGMisCWsb/roYq5k03FKmKuRqnvW+zsi//PJbXyosCypr3YVNgJB9twm5EVPGpTKTXQY4veLXrD+s
2KcbvdFbxuOS17Q3CfbSCol7PBCn9WUAI/YXYuX8GEk95YGArTBYLDfRHR2VuhQklnWfA04Jg+baY9
                    </wsse:BinarySecurityToken>
                    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                           Id="Signature-19321823">
                           <ds:SignedInfo
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

                                  <ds:CanonicalizationMethod
                                         Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                                  <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"

       xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                                  <ds:Reference URI="#id-18615648"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                         <ds:Transforms
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                                <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"

       xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                                         </ds:Transforms>
                                         <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"

       xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                                         <ds:DigestValue
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

       CDrnwEm+bjH4NF5zpQ/pXr7XSas=</ds:DigestValue>
                                  </ds:Reference>
                                  <ds:Reference URI="#id-30222347"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                         <ds:Transforms
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                                <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"

       xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                                         </ds:Transforms>
                                         <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"

       xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
                                         <ds:DigestValue
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

       ncGzkqR6v2JfwU4ya2+lNI1zAAw=</ds:DigestValue>
                                  </ds:Reference>
                           </ds:SignedInfo>
                           <ds:SignatureValue
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

       R4X4MtwQMrolxo0pPbB6MQss78Ql0ZjPHTrjFaOOpCWFcceRXqp2/pRKiGciN78faAK7gu51wmty

       1Rsb6mNTyzl7OhH1jn7ELLwOHANlG10SjYGo/YmmM1NNI4obk5s1bD4eqJ9TgoLL23EmQoV/147v
                                  rmDcAYRsULelqj++SnI=
</ds:SignatureValue>
                           <ds:KeyInfo Id="KeyId-12582949"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                  <wsse:SecurityTokenReference
                                         xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                                         wsu:Id="STRId-10028020"
                                         xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
                                         <wsse:Reference URI="#CertId-97118725"
                                                ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
```

```
                                                        xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" />
                                        </wsse:SecurityTokenReference>
                                </ds:KeyInfo>
                        </ds:Signature>
                </wsse:Security>
                <ish:Context xmlns:ish="http://soap.iarxiu/headers"
                        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
                        SOAP-ENV:mustUnderstand="1" wsu:Id="id-30222347">
                        <saml2:Assertion
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
                                ID="AssertId-1241706834781" IssueInstant="2009-05-
07T16:33:54.797+02:00"
                                Version="2.0">
                                <saml2:Issuer>JavaGuide</saml2:Issuer>
                                <saml2:Subject>
                                        <saml2:SubjectConfirmation

        Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches">
                                                <saml2:NameID>Username</saml2:NameID>
                                        </saml2:SubjectConfirmation>
                                </saml2:Subject>
                                <saml2:AttributeStatement>
                                        <saml2:Attribute
Name="urn:iarxiu:2.0:names:organizationAlias">
                                                <saml2:AttributeValue>organization-1
                                                </saml2:AttributeValue>
                                        </saml2:Attribute>
                                        <saml2:Attribute
Name="urn:iarxiu:2.0:names:fondsAlias">
                                                <saml2:AttributeValue>fonds-
1</saml2:AttributeValue>
                                        </saml2:Attribute>
                                        <saml2:Attribute
Name="urn:iarxiu:2.0:names:member-of">

        <saml2:AttributeValue>archivists</saml2:AttributeValue>
                                        </saml2:Attribute>
                                </saml2:AttributeStatement>
                        </saml2:Assertion>
                </ish:Context>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body
                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
                wsu:Id="id-18615648">
                <ing:IngestRequest
xmlns:ing="http://schemas.core.iarxiu.hp.com/2.0/ingest">
                        <METS:mets xmlns:METS="http://www.loc.gov/METS/"
                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
TYPE="urn:iarxiu:2.0:templates:iarxiu-ramon:PL_document"
                                xsi:schemaLocation="http://www.loc.gov/METS/
http://www.loc.gov/standards/mets/mets.xsd">

                                <!--
                                        Metadades descriptives: vocabulari expedient
Aquest vocabulari Ã©s
                                        propi de dades d'un expedient
                                -->
                                <METS:dmdSec ID="DMD EXP">
                                        <METS:mdWrap MDTYPE="OTHER" MIMETYPE="text/xml"

        OTHERMDTYPE="urn:iarxiu:2.0:vocabularies:iarxiu-ramon:Voc_document">
                                                <METS:xmlData>
                                                        <exp:expedient

        xmlns:exp="http://schemas.user.iarxiu.hp.com/2.0/Voc_document">

        <exp:codi_referencia>CAT/AACC/D1161 N-185 05
                                                                </exp:codi_referencia>
```

```
        <exp:numero_expedient>185/08</exp:numero_expedient>

        <exp:codi classificacio>G0200</exp:codi classificacio>

        <exp:titol_serie_documental>Expedients de personal

        </exp:titol_serie_documental>

        <exp:nivell descripcio>Unitat documental composta
                                                        </exp:nivell descripcio>
                                                        <exp:titol>Expedient de
contractaciÃ³ per a la compra
                                                                de quinze
ordinadors</exp:titol>
                                                        <exp:data obertura>2008-
05-17T09:30:47.0Z
                                                        </exp:data_obertura>
                                                        <exp:data_tancament>2008-
05-17T09:30:47.0Z
                                                        </exp:data tancament>

        <exp:nom_productor>AgÃ¨ncia Catalana de CertificaciÃ³
                                                        </exp:nom_productor>

        <exp:unitat productora>DirecciÃ³ General
                                                        </exp:unitat productora>

        <exp:descripcio>String</exp:descripcio>

        <exp:descriptors>String</exp:descriptors>

        <exp:documentacio relacionada>String

        </exp:documentacio_relacionada>

        <exp:tipus_relacio>String</exp:tipus_relacio>

        <exp:classificacio seguretat acces>AccÃ©s pÃºblic

        </exp:classificacio_seguretat_acces>

        <exp:sensibilitat dades LOPD>Nivell baix

        </exp:sensibilitat_dades_LOPD>
                                                                </exp:expedient>
                                                        </METS:xmlData>
                                                </METS:mdWrap>
                                        </METS:dmdSec>


                                        <METS:structMap>
                                                <METS:div DMDID="DMD_EXP" LABEL="expedient">
                                                </METS:div>
                                        </METS:structMap>

                                </METS:mets>
                                <ing:preservation>false</ing:preservation>
                                <ing:contentTypeHandling>completeWithIntrospection
                                </ing:contentTypeHandling>
                        </ing:IngestRequest>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5.2 Codi font

5.2.1 Client iArxiu Java

5.2.1.1 ApplicationContext.xml

Aquest arxiu conté dades confidencials, les quals han estat eliminades

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

       <bean id="proxyClient" class="iarxiu.javaguide.ProxyClient">
              <property name="defaultUri" value="http://<server>:<port>/core/soap"/>
              <property name="messageFactory" ref="messageFactory"/>
              <property name="interceptors">
                     <list>
                            <ref bean="samlInterceptor"/>
                            <ref bean="wssInterceptor"/>
                     </list>
              </property>
       </bean>

       <bean id="messageFactory"
              class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
              <property name="messageFactory">
                     <bean

       class="com.sun.xml.internal.messaging.saaj.soap.ver1 1.SOAPMessageFactory1 1Im
pl">
                     </bean>
              </property>
       </bean>

       <bean id="samlInterceptor" class="iarxiu.javaguide.SamlInterceptor"/>

       <bean id="wssInterceptor"
class="org.springframework.ws.soap.security.wss4j.Wss4jSecurityInterceptor">
              <property name="securementActions" value="Signature Timestamp"/>
              <property name="securementUsername" value="....."/>
              <property name="securementPassword" value="....."/>
              <property name="securementSignatureCrypto" ref="crypto"/>
              <property name="securementSignatureKeyIdentifier"
value="DirectReference"/>

              <property name="securementSignatureParts">

       <value>{}{http://schemas.xmlsoap.org/soap/envelope/}Body;{}{http://soap.iarxiu
/headers}Context;</value>
              </property>

       </bean>

       <bean id="crypto"
class="org.springframework.ws.soap.security.wss4j.support.CryptoFactoryBean">
              <property name="keyStoreLocation" value="classpath: .....p12"/>
              <property name="keyStoreType" value="pkcs12"/>
              <property name="keyStorePassword" value="....."/>
       </bean>


       <!-- Clients -->

       <bean id="smallMetsIngest" class="iarxiu.javaguide.SmallMetsIngestClient">
              <property name="proxyClient" ref="proxyClient"/>
       </bean>
```

```
        <bean id="bigMetsIngest" class="iarxiu.javaguide.BigMetsIngestClient">
                <property name="proxyClient" ref="proxyClient"/>
                <property name="uploadServiceUrl"
value="http://<server>:<port>/core/servlet/upload"/>
        </bean>

        <bean id="disseminationClient" class="iarxiu.javaguide.DisseminationClient">
                <property name="proxyClient" ref="proxyClient"/>
        </bean>

</beans>
```

5.2.1.2 MetsUtils.java

```
package iarxiu.javaguide;

import gov.loc.mets.MetsDocument;
import gov.loc.mets.MetsDocument.Mets;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.xmlbeans.XmlException;

public class MetsUtils {

        public static Mets load(File metsFile) throws XmlException, IOException {
                return MetsDocument.Factory.parse(metsFile).getMets();
        }

        public static void save(Mets mets, File metsFile) throws IOException,
XmlException {
                MetsDocument metsDocument =
MetsDocument.Factory.parse(mets.getDomNode());

                FileOutputStream output = new FileOutputStream(metsFile);
                metsDocument.save(output);
                output.close();
        }
}
```

5.2.1.3 ProxyClient.java

```
package iarxiu.javaguide;

import org.springframework.oxm.xmlbeans.XmlBeansMarshaller;
import org.springframework.ws.client.core.support.WebServiceGatewaySupport;

import com.hp.iarxiu.core.schemas.x20.dissemination.GetPackageRequestDocument;
import com.hp.iarxiu.core.schemas.x20.dissemination.GetPackageResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetOfflineIngestStatusRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetOfflineIngestStatusResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetUploadTicketRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetUploadTicketResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.IngestRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.IngestResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.OfflineUploadIngestRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.OfflineUploadIngestResponseDocument;


public class ProxyClient extends WebServiceGatewaySupport{

  public ProxyClient() {
        XmlBeansMarshaller xmlBeansMarshaller = new XmlBeansMarshaller();
        setMarshaller(xmlBeansMarshaller);
        setUnmarshaller(xmlBeansMarshaller);
  }

  public IngestResponseDocument send(IngestRequestDocument request) {
```

```
    return
(IngestResponseDocument)getWebServiceTemplate().marshalSendAndReceive(request);
  }

  public GetUploadTicketResponseDocument send(GetUploadTicketRequestDocument request)
{
       return
(GetUploadTicketResponseDocument)getWebServiceTemplate().marshalSendAndReceive(reques
t);
  }

  public OfflineUploadIngestResponseDocument send(OfflineUploadIngestRequestDocument
request) {
         return
(OfflineUploadIngestResponseDocument)getWebServiceTemplate().marshalSendAndReceive(re
quest);
  }

  public GetOfflineIngestStatusResponseDocument
send(GetOfflineIngestStatusRequestDocument request) {
         return
(GetOfflineIngestStatusResponseDocument)getWebServiceTemplate().marshalSendAndReceive
(request);
  }

  public GetPackageResponseDocument send(GetPackageRequestDocument request) {
         return
(GetPackageResponseDocument)getWebServiceTemplate().marshalSendAndReceive(request);
  }
}
```

5.2.1.4 SamlInterceptor.java

```
package iarxiu.javaguide;

import java.io.IOException;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import javax.xml.namespace.QName;

import org.apache.xmlbeans.XmlObject;
import org.apache.xmlbeans.XmlOptions;
import org.springframework.oxm.xmlbeans.XmlBeansMarshaller;
import org.springframework.ws.client.WebServiceClientException;
import org.springframework.ws.client.support.interceptor.ClientInterceptor;
import org.springframework.ws.context.MessageContext;
import org.springframework.ws.soap.SoapHeaderElement;
import org.springframework.ws.soap.SoapMessage;
import org.springframework.ws.soap.saaj.SaajSoapHeaderException;
import org.w3c.dom.Node;
import org.w3c.dom.Text;

import x0Assertion.oasisNamesTcSAML2.AssertionDocument;
import x0Assertion.oasisNamesTcSAML2.AssertionType;
import x0Assertion.oasisNamesTcSAML2.AttributeStatementType;
import x0Assertion.oasisNamesTcSAML2.AttributeType;
import x0Assertion.oasisNamesTcSAML2.NameIDType;
import x0Assertion.oasisNamesTcSAML2.SubjectConfirmationType;
import x0Assertion.oasisNamesTcSAML2.SubjectType;

public class SamlInterceptor implements ClientInterceptor {

       @Override
       public boolean handleFault(MessageContext arg0)
                     throws WebServiceClientException {
              return false;
       }

       @Override
```

```
	public boolean handleRequest(MessageContext messageContext)
			throws WebServiceClientException {
		QName qn = new QName("http://soap.iarxiu/headers","Context","ish");

		SoapMessage soapMessage = ((SoapMessage)messageContext.getRequest());

		SoapHeaderElement wsh =
soapMessage.getSoapHeader().addHeaderElement(qn);
		wsh.setMustUnderstand(true);

		AssertionDocument assertion = createAssertion();

		Map<String,String> ns = new HashMap<String,String>();
		ns.put("urn:oasis:names:tc:SAML:2.0:assertion", "saml2");

		XmlOptions xmlOptions = new XmlOptions();
		xmlOptions.setSaveSuggestedPrefixes(ns);
		xmlOptions.setSaveAggressiveNamespaces();

		XmlBeansMarshaller marshaller = new XmlBeansMarshaller();
		marshaller.setXmlOptions(xmlOptions);

		try {
			marshaller.marshal(assertion, wsh.getResult());
		} catch(IOException e) {
			throw new SaajSoapHeaderException("Error creating SAML Header",
e);
		}

		return true;
	}

	@Override
	public boolean handleResponse(MessageContext messageContext)
			throws WebServiceClientException {
		return false;
	}

	protected AssertionDocument createAssertion() {
		AssertionDocument assertionDocument =
AssertionDocument.Factory.newInstance();
		AssertionType assertion;

		assertion = assertionDocument.addNewAssertion();

		assertion.setVersion("2.0");
		assertion.setID("AssertId-" + System.currentTimeMillis());
		assertion.setIssueInstant( Calendar.getInstance() );

		NameIDType issuerName = assertion.addNewIssuer();
		issuerName.setStringValue("JavaGuide");

		SubjectType subject = assertion.addNewSubject();

		SubjectConfirmationType subjectConfirmation =
subject.addNewSubjectConfirmation();
		subjectConfirmation.setMethod("urn:oasis:names:tc:SAML:2.0:cm:sender-
vouches");

		NameIDType subjectName = subjectConfirmation.addNewNameID();
		subjectName.setStringValue("Username");

		AttributeStatementType attributeStatement =
assertion.addNewAttributeStatement();

		addAttribute(attributeStatement,
"urn:iarxiu:2.0:names:organizationAlias", "organization-1");
		addAttribute(attributeStatement, "urn:iarxiu:2.0:names:fondsAlias",
"fonds-1");
		addAttribute(attributeStatement, "urn:iarxiu:2.0:names:member-of",
"archivists");
```

```
                return assertionDocument;
        }

        private void addAttribute(
                        final AttributeStatementType attributeStatement,
                        final String name,
                        final String value ) {

                AttributeType attribute = attributeStatement.addNewAttribute();
                attribute.setName(name);
                XmlObject xmlNode = attribute.addNewAttributeValue();
                Node node = xmlNode.getDomNode();
                Text text = node.getOwnerDocument().createTextNode(value);
                node.appendChild(text);
        }
}
```

5.2.1.5 TestClient.java

```
package iarxiu.javaguide;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.hp.iarxiu.core.schemas.x20.ingest.ContentTypeHandlingType;

public class TestClient {

        public static void main(String[] args) throws Exception {
                ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

                String packageId = null;

                SmallMetsIngestClient smallMetsIngest =
(SmallMetsIngestClient)context.getBean("smallMetsIngest");
                packageId = smallMetsIngest.ingest("Data\\mets.xml", false,
ContentTypeHandlingType.COMPLETE WITH INTROSPECTION);
                System.out.println("SmallMetsIngest: " + packageId);

                BigMetsIngestClient bigMetsIngest =
(BigMetsIngestClient)context.getBean("bigMetsIngest");
                packageId = bigMetsIngest.ingest("Data\\mets.xml", false,
ContentTypeHandlingType.COMPLETE WITH INTROSPECTION);
                System.out.println("BigMetsIngest: " + packageId);

                DisseminationClient disseminationClient =
(DisseminationClient)context.getBean("disseminationClient");
                disseminationClient.download("Data\\mets2.xml", packageId, false,
false);
                System.out.println("Dissemination: OK");
        }
}
```

5.2.1.6 SmallMetsIngestClient.java

```
package iarxiu.javaguide;

import gov.loc.mets.MetsDocument.Mets;

import java.io.File;

import com.hp.iarxiu.core.schemas.x20.ingest.ContentTypeHandlingType;
import com.hp.iarxiu.core.schemas.x20.ingest.IngestRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.IngestResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.IngestRequestDocument.IngestRequest;

public class SmallMetsIngestClient {

        private ProxyClient proxyClient;
```

```
        public ProxyClient getProxyClient() {
                return proxyClient;
        }

        public void setProxyClient(ProxyClient proxyClient) {
                this.proxyClient = proxyClient;
        }

        public String ingest(
                        String metsFilename,
                        boolean preservation,
                        ContentTypeHandlingType.Enum contentTypeHandling
                        ) throws Exception {

                String packageId = null;

                File metsFile = new File(metsFilename);

                Mets mets = MetsUtils.load(metsFile);

                IngestRequestDocument requestDocument =
IngestRequestDocument.Factory.newInstance();
                IngestRequest request = requestDocument.addNewIngestRequest();

                request.setPreservation(preservation);
                request.setContentTypeHandling(contentTypeHandling);
                request.setMets( mets );

                IngestResponseDocument responseDocument =
getProxyClient().send(requestDocument);
                packageId =
responseDocument.getIngestResponse().getIngestInfo().getId();

                return packageId;
        }
}
```

5.2.1.7 BigMetsIngestClient.java

```
package iarxiu.javaguide;

import java.io.File;
import java.io.IOException;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpException;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.multipart.FilePart;
import org.apache.commons.httpclient.methods.multipart.MultipartRequestEntity;
import org.apache.commons.httpclient.methods.multipart.Part;
import org.apache.commons.httpclient.methods.multipart.StringPart;

import com.hp.iarxiu.core.schemas.x20.ingest.ContentTypeHandlingType;
import com.hp.iarxiu.core.schemas.x20.ingest.GetOfflineIngestStatusRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetOfflineIngestStatusResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetUploadTicketRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.GetUploadTicketResponseDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.OfflineIngestInfoType;
import com.hp.iarxiu.core.schemas.x20.ingest.OfflineUploadIngestRequestDocument;
import com.hp.iarxiu.core.schemas.x20.ingest.OfflineUploadIngestResponseDocument;
import
com.hp.iarxiu.core.schemas.x20.ingest.OfflineUploadIngestRequestDocument.OfflineUploa
dIngestRequest;

public class BigMetsIngestClient {

        private ProxyClient proxyClient;
        private String uploadServiceUrl;

        public ProxyClient getProxyClient() {
                return proxyClient;
```

```
        }

        public void setProxyClient(ProxyClient proxyClient) {
                this.proxyClient = proxyClient;
        }

        public String getUploadServiceUrl() {
                return uploadServiceUrl;
        }

        public void setUploadServiceUrl(String uploadServiceUrl) {
                this.uploadServiceUrl = uploadServiceUrl;
        }

        public String ingest(
                        String metsFilename,
                        boolean preservation,
                        ContentTypeHandlingType.Enum contentTypeHandling
                        )
        throws
                HttpException,
                IOException {

                String packageId = null;

                String uploadTicket = getUploadTicket();

                upload(uploadTicket, metsFilename);

                String offlineTicket = offlineIngest(uploadTicket, preservation,
contentTypeHandling);

                packageId = waitOffline(offlineTicket);

                return packageId;
        }

        private String getUploadTicket() {
                String ticket = null;

                GetUploadTicketRequestDocument requestDocument =
GetUploadTicketRequestDocument.Factory.newInstance();
                requestDocument.addNewGetUploadTicketRequest();

                GetUploadTicketResponseDocument responseDocument =
getProxyClient().send(requestDocument);
                ticket = responseDocument.getGetUploadTicketResponse().getTicket();

                return ticket;
        }

        private void upload(String ticket, String metsFilename)
        throws
                HttpException,
                IOException {

                File metsFile = new File(metsFilename);

                HttpClient httpClient = new HttpClient();

                PostMethod post = new PostMethod(getUploadServiceUrl());

                Part ticketPart = new StringPart("ticket", ticket);
                Part metsPart = new FilePart("mets", metsFile);

                Part[] parts = new Part[]{
                                ticketPart,
                                metsPart
                };

                post.setRequestEntity( new
MultipartRequestEntity(parts,post.getParams()) );
```

```
                httpClient.executeMethod(post);
        }

        private String offlineIngest(
                        String uploadTicket,
                        boolean preservation,
                        ContentTypeHandlingType.Enum contentTypeHandling) {

                String offlineTicket = null;

                OfflineUploadIngestRequestDocument requestDocument =
OfflineUploadIngestRequestDocument.Factory.newInstance();
                OfflineUploadIngestRequest request =
requestDocument.addNewOfflineUploadIngestRequest();

                request.setPreservation(preservation);
                request.setContentTypeHandling(contentTypeHandling);
                request.setUploadTicket(uploadTicket);

                OfflineUploadIngestResponseDocument responseDocument =
getProxyClient().send(requestDocument);
                offlineTicket = responseDocument.getOfflineUploadIngestResponse();

                return offlineTicket;
        }

        private String waitOffline(String offlineTicket) {
                String packageId = null;

                OfflineIngestInfoType ingestInfo;
                while(true) {
                        GetOfflineIngestStatusRequestDocument requestDocument =
GetOfflineIngestStatusRequestDocument.Factory.newInstance();

        requestDocument.setGetOfflineIngestStatusRequest(offlineTicket);

                        GetOfflineIngestStatusResponseDocument responseDocument =
getProxyClient().send(requestDocument);
                        ingestInfo =
responseDocument.getGetOfflineIngestStatusResponse().getOfflineIngestInfo();

                        if
(!ingestInfo.getStatus().equals(OfflineIngestInfoType.Status.IN PROCESS))
                                break;

                        try { Thread.sleep(1000);}
                        catch (InterruptedException e) {
                                throw new RuntimeException("Error in sleep", e);
                        }
                }

                if (ingestInfo.getStatus().equals(OfflineIngestInfoType.Status.ERROR))
                        throw new RuntimeException("Offline error: " +
ingestInfo.getErrorCode());

                if
(ingestInfo.getStatus().equals(OfflineIngestInfoType.Status.UNKNOWN))
                        throw new RuntimeException("Unknown offline error: " +
ingestInfo.getErrorCode());

                packageId = ingestInfo.getId();

                return packageId;
        }
}
```

5.2.1.8 DisseminationClient.java

```
package iarxiu.javaguide;
```

```
import gov.loc.mets.MetsDocument.Mets;

import java.io.File;
import java.io.IOException;

import javax.xml.transform.TransformerException;

import org.apache.xmlbeans.XmlException;

import com.hp.iarxiu.core.schemas.x20.dissemination.GetPackageRequestDocument;
import com.hp.iarxiu.core.schemas.x20.dissemination.GetPackageResponseDocument;
import
com.hp.iarxiu.core.schemas.x20.dissemination.GetPackageRequestDocument.GetPackageRequ
est;

public class DisseminationClient {

        private ProxyClient proxyClient;

        public ProxyClient getProxyClient() {
                return proxyClient;
        }

        public void setProxyClient(ProxyClient proxyClient) {
                this.proxyClient = proxyClient;
        }

        public void download(
                        String metsFilename,
                        String packageId,
                        boolean includeBinaries,
                        boolean includeDC)
                throws
                        IOException,
                        XmlException,
                        TransformerException {

                File metsFile = new File(metsFilename);

                GetPackageRequestDocument requestDocument =
GetPackageRequestDocument.Factory.newInstance();
                GetPackageRequest request = requestDocument.addNewGetPackageRequest();

                request.setPackageId(packageId);
                request.setIncludeBinaries(includeBinaries);
                request.setIncludeDC(includeDC);

                GetPackageResponseDocument responseDocument =
getProxyClient().send(requestDocument);
                Mets mets = responseDocument.getGetPackageResponse().getMets();

                MetsUtils.save(mets, metsFile);
        }
}
```

5.2.2 Client iArxiu .NET

5.2.2.1 App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="microsoft.web.services3"
type="Microsoft.Web.Services3.Configuration.WebServicesConfiguration,
Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
  </configSections>
  <system.web>
    <webServices>
      <soapExtensionTypes>
```

```
        <add type="iArxiu.NETGuide.SamlSoapExtension, iArxiu.NETGuide" priority="1"
group="0"/>
      </soapExtensionTypes>
    </webServices>
  </system.web>
  <microsoft.web.services3>
    <policy fileName="iArxiuClientPolicies.config"/>
  </microsoft.web.services3>
</configuration>
```

5.2.2.2 iArxiuClientPolicies.config

Aquest arxiu conté dades confidencials, les quals han estat eliminades

```
<?xml version="1.0" encoding="utf-8" ?>
<policies>
  <extensions>
    <extension name="SignaturePolicyAssertion"
type="iArxiu.NETGuide.SignaturePolicyAssertion, iArxiu.NETGuide"/>
  </extensions>
  <policy name="CorePolicy">
    <SignaturePolicyAssertion>
      <X509TokenProvider
        storeLocation="CurrentUser"
        storeName="My"
        subjectDN="..."/>
    </SignaturePolicyAssertion>
  </policy>
</policies>
```

5.2.2.3 MetsUtils.cs

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Serialization;

namespace iArxiu.NETGuide
{
    class MetsUtils<METS_TYPE>
    {
        private static XmlQualifiedName QN METS = new XmlQualifiedName("mets",
"http://www.loc.gov/METS/");

        public static METS_TYPE Load(string fileName)
        {
            METS TYPE metsDocument;

            XmlRootAttribute metsRoot = new XmlRootAttribute();
            metsRoot.ElementName = QN_METS.Name;
            metsRoot.Namespace = QN_METS.Namespace;
            metsRoot.IsNullable = true;

            XmlSerializer serializer = new XmlSerializer(typeof(METS TYPE),
metsRoot);

            TextReader reader = new StreamReader(fileName);

            XmlTextReader xmlTextReader = new XmlTextReader(reader);

            metsDocument = (METS TYPE)serializer.Deserialize(xmlTextReader);

            reader.Close();

            return metsDocument;
        }

        public static void Save(METS_TYPE metsDocument, string filename)
        {
```

```
            XmlRootAttribute metsRoot = new XmlRootAttribute();
            metsRoot.ElementName = QN_METS.Name;
            metsRoot.Namespace = QN_METS.Namespace;
            metsRoot.IsNullable = true;

            XmlSerializer serializer = new XmlSerializer(typeof(METS_TYPE),
metsRoot);

            TextWriter writer = new StreamWriter(filename);

            XmlTextWriter xmlTextWriter = new XmlTextWriter(writer);

            serializer.Serialize(xmlTextWriter, metsDocument);

            writer.Close();
        }
    }
}
```

5.2.2.4 HttpClient.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Net;

namespace iArxiu.NETGuide
{
    abstract class HttpPart
    {
        public string Name;

        public HttpPart(string name)
        {
            this.Name = name;
        }
    }

    class HttpFieldPart : HttpPart
    {
        public string Value;

        public HttpFieldPart(string name, string value)
            : base(name)
        {
            this.Value = value;
        }
    }

    class HttpFilePart : HttpPart
    {
        public string Filename;
        public string ContentType;

        public HttpFilePart(string name, string filename)
            : this(name, filename, "application/octet-stream")
        {
        }

        public HttpFilePart(string name, string filename, string contentType)
            : base(name)
        {
            this.Filename = filename;
            this.ContentType = contentType;
        }
    }

    class HttpMultiPartClient
    {
        private string BOUNDARY = Guid.NewGuid().ToString();
```

```
        public string Url;

        public List<HttpPart> Parts = new List<HttpPart>();

        public string Send()
        {
            string responseBody = null;

            byte[] postData = GetPostData();

            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(Url);

            request.Method = "POST";
            request.ContentLength = postData.Length;
            request.ContentType = String.Format("multipart/form-data; boundary={0}",
BOUNDARY);

            Stream requestStream = request.GetRequestStream();
            requestStream.Write(postData, 0, postData.Length);
            requestStream.Flush();

            HttpWebResponse response = (HttpWebResponse)request.GetResponse();

            StreamReader sr = new StreamReader(response.GetResponseStream());
            responseBody = sr.ReadToEnd();
            sr.Close();

            return responseBody;
        }

        private byte[] GetPostData()
        {
            MemoryStream postData = new MemoryStream();

            foreach (HttpPart part in Parts)
            {
                Append(postData, "--{0}\r\n", BOUNDARY);

                if (part is HttpFieldPart)
                {
                    HttpFieldPart httpFieldPart = (HttpFieldPart)part;

                    Append(postData, "Content-Disposition: form-data;
name=\"{0}\"\r\n", httpFieldPart.Name);
                    Append(postData, "\r\n");

                    Append(postData, httpFieldPart.Value);
                }
                else
                {
                    HttpFilePart httpFilePart = (HttpFilePart)part;

                    FileInfo fileInfo = new FileInfo(httpFilePart.Filename);

                    Append(postData, "Content-Disposition: form-data; name=\"{0}\";
filename=\"{1}\"\r\n", httpFilePart.Name, fileInfo.Name);
                    Append(postData, "Content-Type: {0}\r\n",
httpFilePart.ContentType);
                    Append(postData, "Content-Transfer-Encoding: binary\r\n");
                    Append(postData, "\r\n");

                    Append(postData, File.ReadAllBytes(httpFilePart.Filename));
                }

                Append(postData, "\r\n");
            }

            Append(postData, "--{0}--", BOUNDARY);

            return postData.GetBuffer();
        }
```

```
        private void Append(MemoryStream buffer, string data)
        {
            byte[] bufferData = Encoding.UTF8.GetBytes(data);
            buffer.Write(bufferData, 0, bufferData.Length);
        }

        private void Append(MemoryStream buffer, string data, params object[] args)
        {
            byte[] bufferData = Encoding.UTF8.GetBytes(String.Format(data, args));
            buffer.Write(bufferData, 0, bufferData.Length);
        }

        private void Append(MemoryStream buffer, byte[] data)
        {
            buffer.Write(data, 0, data.Length);
        }
    }
}
```

5.2.2.5 SamlHeader.cs

```
using System;

/*
 * SOAP
 */
using System.Xml.Serialization;
using System.Web.Services.Protocols;


namespace iArxiu.NETGuide
{
    [XmlRoot("Context", Namespace = "http://soap.iarxiu/headers")]
    public class SamlHeader : SoapHeader
    {
        [XmlAttribute("Id", Namespace = "http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd")]
        public string Id = "Id-" + Guid.NewGuid().ToString();

        [XmlElement("Assertion", Namespace =
"urn:oasis:names:tc:SAML:2.0:assertion")]
        public AssertionType Assertion;
    }
}
```

5.2.2.6 SamlSoapExtension.cs

```
using System;

/*
 * SOAP
 */
using System.Web.Services.Protocols;

namespace iArxiu.NETGuide
{
    class SamlSoapExtension : SoapExtension
    {
        public override object GetInitializer(LogicalMethodInfo methodInfo,
SoapExtensionAttribute attribute)
        {
            throw new NotImplementedException();
        }

        public override object GetInitializer(Type serviceType)
        {
            return null;
        }

        public override void Initialize(object initializer)
```

```
        {
            // DO NOTHING
        }

        public override void ProcessMessage(SoapMessage message)
        {
            switch (message.Stage)
            {
                case SoapMessageStage.BeforeSerialize:
                    if (message is SoapClientMessage)
                    {
                        SamlHeader samlHeader = new SamlHeader();
                        samlHeader.MustUnderstand = true;

                        AssertionType assertion = CreateAssertion();

                        samlHeader.Assertion = assertion;

                        message.Headers.Add(samlHeader);
                    }
                    break;
            }
        }

        private AssertionType CreateAssertion()
        {
            AssertionType assertion = new AssertionType();

            assertion.Version = "2.0";
            assertion.ID = "AssertID-" + Guid.NewGuid().ToString();
            assertion.IssueInstant = DateTime.Now;

            NameIDType issuerName = new NameIDType();
            issuerName.Value = "NETGuide";
            assertion.Issuer = issuerName;

            SubjectType subject = new SubjectType();

            SubjectConfirmationType subjectCofirmation = new
SubjectConfirmationType();
            subjectCofirmation.Method = "urn:oasis:names:tc:SAML:2.0:cm:sender-
vouches";

            NameIDType subjectName = new NameIDType();
            subjectName.Value = "Username";

            subjectCofirmation.Item = subjectName;

            subject.Items = new Object[] { subjectCofirmation };

            assertion.Subject = subject;

            AttributeStatementType attributeStatement = new AttributeStatementType();

            AttributeType organization =
CreateAttribute("urn:iarxiu:2.0:names:organizationAlias", "organization-1");
            AttributeType fonds = CreateAttribute("urn:iarxiu:2.0:names:fondsAlias",
"fonds-1");
            AttributeType memberOf = CreateAttribute("urn:iarxiu:2.0:names:member-
of", "archivists");

            attributeStatement.Items = new object[] { organization, fonds, memberOf
};

            assertion.Items = new AttributeStatementType[] { attributeStatement };

            return assertion;
        }

        private AttributeType CreateAttribute(string name, string value)
        {
            AttributeType attribute = new AttributeType();
```

```
            attribute.Name = name;
            attribute.AttributeValue = new object[] { value };

            return attribute;
        }
    }
}
```

5.2.2.7 SignaturePolicyAssertion.cs

```
using System;
using System.Collections.Generic;

/*
 * XML
 */
using System.Xml;
using System.Xml.Serialization;

/*
 * WSE 3.0
 */
using Microsoft.Web.Services3;
using Microsoft.Web.Services3.Design;
using Microsoft.Web.Services3.Security;
using Microsoft.Web.Services3.Security.Xml;
using Microsoft.Web.Services3.Security.Tokens;

/*
 * X509 CERTIFICATES
 */
using System.Security.Cryptography.X509Certificates;

namespace iArxiu.NETGuide
{
    class SignaturePolicyAssertion : SecurityPolicyAssertion
    {
        private X509SecurityToken taToken;

        public X509SecurityToken X509Token
        {
            get { return taToken; }
        }

        public override SoapFilter CreateClientInputFilter(FilterCreationContext
context)
        {
            return null;
        }

        public override SoapFilter CreateClientOutputFilter(FilterCreationContext
context)
        {
            return new SecuritySignatureFilter(this);
        }

        public override SoapFilter CreateServiceInputFilter(FilterCreationContext
context)
        {
            return null;
        }

        public override SoapFilter CreateServiceOutputFilter(FilterCreationContext
context)
        {
            return null;
        }

        public override void ReadXml(System.Xml.XmlReader reader, IDictionary<string,
Type> extensions)
        {
```

```
            bool isEmpty;

            StoreLocation storeLocation = StoreLocation.CurrentUser;
            StoreName storeName = StoreName.My;
            string subjectDN = null;

            isEmpty = reader.IsEmptyElement;
            reader.ReadStartElement("SignaturePolicyAssertion");
            if (!isEmpty)
            {
                if (reader.MoveToContent() == XmlNodeType.Element && reader.Name ==
"X509TokenProvider")
                {
                    string storeLocationAttribute =
reader.GetAttribute("storeLocation");
                    string storeNameAttribute = reader.GetAttribute("storeName");

                    if (storeLocationAttribute != null)
                        storeLocation =
(StoreLocation)Enum.Parse(typeof(StoreLocation), storeLocationAttribute, true);

                    if (storeNameAttribute != null)
                        storeName = (StoreName)Enum.Parse(typeof(StoreName),
storeNameAttribute, true);

                    subjectDN = reader.GetAttribute("subjectDN");

                    isEmpty = reader.IsEmptyElement;
                    reader.ReadStartElement();
                    if (!isEmpty) reader.ReadEndElement();
                }

                reader.ReadEndElement();
            }

            if (subjectDN == null)
                throw new ArgumentNullException("Missing subjectDN attribute");

            taToken = X509TokenProvider.CreateToken(storeLocation, storeName,
subjectDN);
        }
    }

    class SecuritySignatureFilter : SendSecurityFilter
    {
        private X509SecurityToken taToken;

        public SecuritySignatureFilter(SignaturePolicyAssertion policyAssertion)
            : base(policyAssertion.ServiceActor, true)
        {
            taToken = policyAssertion.X509Token;
        }

        public override void SecureMessage(SoapEnvelope envelope, Security security)
        {
            security.Tokens.Add(taToken);

            MessageSignature signature = new MessageSignature(taToken);
            signature.SignatureOptions = SignatureOptions.IncludeSoapBody;

            XmlNode contextHeader = findContextHeader(envelope);
            if (contextHeader != null)
            {
                SignatureReference reference = new SignatureReference("#" +
contextHeader.Attributes["Id", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"].Value);
                reference.AddTransform(new XmlDsigExcC14NTransform());

                signature.AddReference(reference);
            }

            security.Elements.Add(signature);
```

```
        }

        private XmlNode findContextHeader(SoapEnvelope envelope)
        {
            foreach (XmlNode child in envelope.Header.ChildNodes)
            {
                if (child != null && child.LocalName == "Context" &&
child.NamespaceURI == "http://soap.iarxiu/headers")
                {
                    return child;
                }
            }

            return null;
        }
    }
}
```

5.2.2.8 TestClient.cs

```
using System;

/*
 * iArxiu
 */
using iArxiu.Core.Services.Ingest;
using iArxiu.Core.Services.Dissemination;

namespace iArxiu.NETGuide
{
    class TestClient
    {
        static void Main(string[] args)
        {
            string packageId = null;

            IngestService ingestProxy = new IngestService();
            ingestProxy.Url = "http://localhost:8080/core/soap";
            ingestProxy.SetPolicy("CorePolicy");

            DisseminationService disseminationProxy = new DisseminationService();
            disseminationProxy.Url = "http://localhost:8080/core/soap";
            disseminationProxy.SetPolicy("CorePolicy");

            packageId = SmallMetsIngest(ingestProxy, @"Data\mets.xml", false,
contentTypeHandlingType.completeWithIntrospection);
            Console.WriteLine("SmallMetsIngest: {0}", packageId);

            packageId = BigMetsIngest(ingestProxy, @"Data\mets.xml", false,
contentTypeHandlingType.completeWithIntrospection);
            Console.WriteLine("BigMetsIngest: {0}", packageId);

            Download(disseminationProxy, @"Data\mets2.xml", packageId, false, false);
            Console.WriteLine("Dissemination: OK");
        }

        private static string SmallMetsIngest(IngestService ingestProxy, string
filename, bool preservation, contentTypeHandlingType contentTypeHandling)
        {
            string packageId = null;

            SmallMetsIngestClient smallMetsIngest = new SmallMetsIngestClient();
            smallMetsIngest.Proxy = ingestProxy;

            packageId = smallMetsIngest.Ingest(filename, preservation,
contentTypeHandling);

            return packageId;
        }
```

```
        private static string BigMetsIngest(IngestService ingestProxy, string
filename,  bool preservation, contentTypeHandlingType contentTypeHandling)
        {
            string packageId = null;

            BigMetsIngestClient bigMetsIngest = new BigMetsIngestClient();
            bigMetsIngest.Proxy = ingestProxy;
            bigMetsIngest.UploadServiceUrl =
"http://localhost:8080/core/servlet/upload";

            packageId = bigMetsIngest.ingest(filename, preservation,
contentTypeHandling);

            return packageId;
        }

        private static void Download(DisseminationService disseminationProxy, string
filename, string packageId, bool includeBinaries, bool includeDC)
        {
            DisseminationClient dissemination = new DisseminationClient();
            dissemination.Proxy = disseminationProxy;

            dissemination.Download(filename, packageId, includeBinaries, includeDC);
        }
    }
}
```

5.2.2.9 SmallMetsIngestClient.cs

```
using System;

/*
 * iArxiu
 */
using iArxiu.Core.Services.Ingest;

namespace iArxiu.NETGuide
{
    class SmallMetsIngestClient
    {
        public IngestService Proxy;

        public string Ingest(string metsFilename, bool preservation,
contentTypeHandlingType contentTypeHandling)
        {
            String packageId = null;

            mets mets = MetsUtils<mets>.Load(metsFilename);

            IngestRequest request = new IngestRequest();
            request.mets = mets;
            request.preservation = false;
            request.contentTypeHandling = contentTypeHandlingType.checkAndReject;

            IngestResponse response = Proxy.Ingest(request);
            packageId = response.ingestInfo.id;

            return packageId;
        }
    }
}
```

5.2.2.10 BigMetsIngestClient.cs

```
using System;
using System.Threading;

/*
 * iArxiu
 */
using iArxiu.Core.Services.Ingest;
```

```
namespace iArxiu.NETGuide
{
    class BigMetsIngestClient
    {
        public IngestService Proxy;

        public string UploadServiceUrl;

        public string ingest(string metsFilename, bool preservation,
contentTypeHandlingType contentTypeHandling)
        {
            string packageId = null;

            string uploadTicket = GetUploadTicket();

            Upload(uploadTicket, metsFilename);

            string offlineTicket = OfflineIngest(uploadTicket, preservation,
contentTypeHandling);

            packageId = WaitOffline(offlineTicket);

            return packageId;
        }

        private string GetUploadTicket()
        {
            string ticket = null;

            GetUploadTicketRequest request = new GetUploadTicketRequest();

            GetUploadTicketResponse response = Proxy.GetUploadTicket(request);
            ticket = response.ticket;

            return ticket;
        }

        private void Upload(string uploadTicket, string metsFilename)
        {
            HttpMultiPartClient httpClient = new HttpMultiPartClient();
            httpClient.Url = UploadServiceUrl;

            httpClient.Parts.Add(new HttpFieldPart("ticket", uploadTicket));
            httpClient.Parts.Add(new HttpFilePart("mets", metsFilename));

            httpClient.Send();
        }

        private string OfflineIngest(string uploadTicket, bool preservation,
contentTypeHandlingType contentTypeHandling)
        {
            string offlineTicket = null;

            OfflineUploadIngestRequest request = new OfflineUploadIngestRequest();
            request.preservation = false;
            request.contentTypeHandling = contentTypeHandling;
            request.uploadTicket = uploadTicket;

            offlineTicket = Proxy.OfflineUploadIngest(request);

            return offlineTicket;
        }

        public string WaitOffline(string offlineTicket)
        {
            string packageId = null;

            offlineIngestInfoType ingestInfo;
            while (true)
            {
```

```
                GetOfflineIngestStatusResponse response =
Proxy.GetOfflineIngestStatus(offlineTicket);
                ingestInfo = response.offlineIngestInfo;

                if (ingestInfo.status != offlineIngestInfoTypeStatus.inProcess)
                    break;

                Thread.Sleep(1000);
            }

            switch (ingestInfo.status)
            {
                case offlineIngestInfoTypeStatus.error:
                    throw new ApplicationException(ingestInfo.errorCode);

                case offlineIngestInfoTypeStatus.unknown:
                    throw new ApplicationException("Unknown status");
            }

            packageId = ingestInfo.id;

            return packageId;
        }
    }
}
```

5.2.2.11 DisseminationClient.cs

```
using System;

/*
 * iArxiu
 */
using iArxiu.Core.Services.Dissemination;

namespace iArxiu.NETGuide
{
    class DisseminationClient
    {
        public DisseminationService Proxy;

        public void Download(string metsFilename, string packageId, bool
includeBinaries, bool includeDC)
        {
            GetPackageRequest request = new GetPackageRequest();
            request.packageId = packageId;
            request.includeBinaries = includeBinaries;
            request.includeDC = includeDC;

            GetPackageResponse response = Proxy.GetPackage(request);
            mets metsDocument = response.mets;

            MetsUtils<mets>.Save(metsDocument, metsFilename);
        }
    }
}
```

5.3 Alias d'un PFX o P12

Els arxius p12 o pfx són repositoris (keystores) de certificats, cadenes de certificats i claus privades.

Cada entrada té un tipus, un valor i un número d'entrada (*àlies*).

- **Keytool:**

  keytool –list –keystore [NOM ARXIU] –storetype pkcs12
  (Demanarà el password)

Això generarà una sortida com aquesta:

cn=cpisr-1 c jokin barbarias berecibar, oid.2.5.4.5=44330313k, oid.2.5.4.12=cont inuador procediment contractaci¾, oid.2.5.4.42=jokin, oid.2.5.4.4=barbarias berecibar, ou=vegeu https://www.catcert.net/vercpisr-1carrecsafp (c)05, ou=serveis publics de certificacio cpisr-1 amb carrec, ou=Órea d'atenci¾ al client, o=sadiel, c=es, 22-oct-2008, PrivateKeyEntry, Huella digital de certificado (MD5): 2E:BC:DC:D6:D5:90:AA:F8:00:34:21:7C:36:45:37:BD

L'alias correspon al text marcat en negreta

- **Programa (en Java):**

```
String alias = "";

KeyStore ks = KeyStore.getInstance("PKCS12");
FileInputStream fis = new FileInputStream(new File(certificate));

try {
   ks.load(fis, keystorePassword.toCharArray());
   alias = ks.aliases().nextElement();

} finally {
   fis.close();
}

System.out.println("Alias: " + alias);
```

5.4 POST/Multipart

Un enviament multipart permet enviar múltiples continguts de diferents tipus en un única petició web. Cada contingut està demarcat per unes fronteres anomenades ***boundaries***. Les fronteres son literals textuals lliures que no haurien d'aparèixer en el contingut.

El valor de literal frontera s'indica a la capçalera. La marca de començament de frontera s'indica precedint el literal frontera amb dos signes "—", mentre que el final d'enviament s'indica amb el conveni "—" precedint i seguint el literal frontera.

Un exemple seria:

```
MIME-version: 1.0
Content-type: multipart/mixed; boundary="myboundary"

--myboundary
Content-type: text/plain

Este es el cuerpo del mensaje
--myboundary
Content-type: application/octet-stream
Content-transfer-encoding: base64

PGh0b ... +Cic=\
--myboundary--
```

En aquest exemple es veu com el literal *myboundary* actua com delimitador. Comentar que tot i que aquest enviament és correcte podria fallar si el contingut té la paraula *myboundary* entre les seves dades. En una aplicació real s'escull el valor de literal frontera amb valors *a-priori* impossibles en el contingut.

5.5 Altres versions de Java

Aquesta guia ha estat redactada considerant l'entorn de Java 6. Aquest correspon a l'entorn més actual. No obstant donat que en el moment de la redacció d'aquesta guia encara hi ha usuaris amb altres versions aquest punt descriu els canvis necessaris per a que el codi presentat en aquest document pugui funcionar també amb altres versions més antigues de Java.

5.5.1 Implementació SAAJ usant versions anteriors a Java 6

La versió 6 de Java ha internalitzat una sèrie de components. Per tant si no es disposa de l'entorn Java 6 aquests components s'han d'afegir com a llibreries externes. Aquests components són els següents:

Llibreria	Versió[1]	Lloc de descàrrega
Activation	1.1.1	http:// java.sun.com/javase/technologies/desktop/javabeans /jaf/downloads/index.html
SAAJ	1.3.3	https://saaj.dev.java.net/

Un cop descarregades les llibreries anteriors s'han d'afegir al *classpath* de l'aplicació els següents arxius:

Llibreria	Arxius	Opcional
Activation	*activation.jar*	
SAAJ	*saaj-api.jar*	
	saaj-impl.jar	

Un cop afegides les llibreries anteriors al *classpath* únicament resta modificar l'arxiu de context *Spring* per tal d'indicar la implementació *SAAJ* correcte.

Segons es descriu a l'apartat Configuració del Proxy s'ha d'indicar la implementació *SAAJ*. La versió Java 6 inclou una implementació. La versió Java 5 no la inclou però la llibreria *saaj-impl* n'inclou una. Per tant és suficient amb eliminar la part ***internal*** del *package* de la classe:

```
<bean id="messageFactory"
        class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
        <property name="messageFactory">
                <bean

class="com.sun.xml.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl">
                </bean>
        </property>
</bean>
```

5.5.2 Anotacions i genèrics

A partir de la versió Java 5 es van incorporar com a parts integrants del llenguatge Java les anotacions (*annotations*) i els genèrics (*generics*). El codi presentat en aquesta guia incorporar a l'arxiu *SamlInterceptor* aquests dos aspectes.

Pel que respecte a la compatibilitat amb versions de Java anteriors a la Java 6 s'ha de tenir en compte el següent:

- *Annotations*: La anotació *@Override* present en els mètodes no és acceptada a Java 5 per a implementació d'*interfaces*. Per tant per versions anteriors a Java 6 s'ha d'eliminar aquesta anotació del codi[17]

[17] En la situació comentada aquesta anotació tampoc és estrictament necessària a Java 6

- *Genèrics:* En versions anteriors a Java 5 no existien els *genèrics*. Per tant en el codi *SamlInterceptor* s'ha de fer el següent canvi a la línia [050]:

  Codi per Java 5 o superior:

```
Map<String,String> ns = new HashMap<String,String>();
```

  Codi per versions anteriors a Java 5:

```
Map ns = new HashMap();
```

5.6 Filtrat de codificació de text a .NET

.NET manega els objectes a memòria utilitzant per les cadenes textuals (*String*) codificació *Unicode*. Quan a partir d'aquests objectes es generen els missatges SOAP aquests usen la codificació *UTF-8*.

La codificació *UTF-8* usa generalment un *byte* per símbol però hi ha casos en els quals es requereixen més d'un *byte*.

S'ha detectat un problema que provoca un error de validació de la signatura del missatge *SOAP* quan els missatges contenen símbols la codificació dels quals en *UTF-8* ocupa més de 2 *bytes*.

Aquests casos "conflictius" corresponen generalment a cadenes textuals extretes de documents *Microsoft Office* i que contenen guions, cometes, apòstrofs i símbols especials, els quals generalment no poden ser digitats d'una manera simple des dels teclats.

El problema afecta únicament a la metadada inclosa en els paquets i per tant no afecta als continguts binaris inclosos dins els paquets ni tampoc als externs.

Per solucionar aquest problema es pot fer un filtrat de les dades incloses als camps metadada, o si es vol, es pot filtrar tot el paquet mets generat abans de ser enviat.

A mode orientatiu s'inclou a continuació una possible funció de filtre:

```
public sealed class UTF8Filter
{
    private static Dictionary<char, char> translation;

    static UTF8Filter()
    {
        translation = new Dictionary<char, char>();

        /* Conflicts */
        translation.Add('\u2013', '-');
        translation.Add('\u2019', '\'');
        translation.Add('\u201C', '"');
        translation.Add('\u201D', '"');
    }

    public string Execute(string s)
    {
        char[] chars = s.ToCharArray();

        for(int i = 0; i < chars.Length; i++)
        {
            char c = chars[i];

            if (IsUTF8Conflict(c))
            {
                c = Translate(c);

                chars[i] = c;
```

```
                }
            }

            return new String(chars);
        }

        private bool IsUTF8Conflict(char c)
        {
            return Encoding.UTF8.GetByteCount(new char[] { c }) > 2;
        }

        private char Translate(char c)
        {
            if (!translation.ContainsKey(c))
                throw new ArgumentException(String.Format("Could not translate '{0}'
= '{1:X4}'", c, (int)c));

            return translation[c];
        }
    }
```

Aquesta classe filtre pot filtrar cadenes mitjançant el seu mètode *Execute*. Es pot adaptar per filtrar altres fonts de dades com podrien ser arxius.

El funcionament del filtre es bassa en detectar els símbols conflictius y traduir-los a símbols que no provoquin els errors esmentats.

El mètode que detecta símbols conflictius es *IsUTF8Conflict* i simplement considera com a símbol conflictiu tot aquell que tingui una codificació *UTF-8* de longitud superior a 2.

Els símbols conflictius es tradueixen mitjançant el mètode *Translate*, el qual es basa en un mapa de traduccions on cada símbol conflictiu té la seva traducció. Quan un símbol no té prevista la traducció es genera una excepció i no permet generar contingut. Evidentment aquest sistema és una orientació. Les aplicacions reals utilitzaran altres sistemes.

L'exemple mostra 4 des casos típics indicant els símbols conflictius i una possible traducció:

Símbol conflictiu		Símbol correcte	
Símbol	Unicode (HEX)	Símbol	ASCII (HEX)
–	2013	-	55
’	2019	‘	47
“	201C	“	42
”	201D	“	42

Generalment aquest filtre hauria de ser inclòs en els codis generadors de paquets de tal manera que l'arxiu mets generat no tingués símbols conflictius. Ara bé es possible que això no sigui aplicable, per la qual cosa una altre opció consistiria en filtrar el paquet després de ser generat.

El següent codi d'exemple filtra un arxiu mets ja generat. No s'inclou la classe contenidora i tampoc hi ha un control acurat d'errors. Per altra banda el sistema per a indicar els noms d'entrada i sortida haurien de ser adaptats a cada necessitat:

```
            string inputMets = [INDICAR EL NOM D'ARXIU D'ENTRADA];
            string outputMets = [INDICAR EL NOM D'ARXIU DE SORTIDA];

            UTF8Filter filter = new UTF8Filter();
            string line;

            using (StreamReader input = new StreamReader(inputMets))
```

```
            {
                using (StreamWriter output = new StreamWriter(outputMets))
                {
                    while ((line = input.ReadLine()) != null)
                    {
                        line = filter.Execute(line);
                        output.WriteLine(line);
                    }
                }
            }
```

6 Annexes

6.1 Glossari

SAML	Llenguatge de marcat de documents per a expressar assercions o afirmacions sobre entitats (*Security Assertions Markup Language*)
SAAJ	Api de Java de baix nivell per a creació de missatges *SOAP* (*Soap with Attachement Api for Java*)
SOAP	Protocol de missatgeria usada per la tecnologia de serveis web (*Simple Object Access Protocol*)
WSSE	Extensions de seguretat per a serveis web (*Web Services Security Extensions*)
POST	Mètode d'enviament en missatgeria web/html en el qual els paràmetres de la petició estan inclosos en el cos de la petició.
MULTIPART	Format de transmissió web el permet enviar múltiples continguts de múltiples tipus en una única petició.

6.2 Taula d'il·lustracions