Creación de nuevos atributos de firma

Control documental

|  |  |  |  |
| --- | --- | --- | --- |
| Estado formal | Elaborado por:  Ramsés Corporales | | Aprobado por:  Nombre y apellidos |
| Fecha de creación | 13/07/2020 | | |
| Control de versiones | Fecha: | 13/07/2020 | |
| Descripción: | Creación del documento | |
| **Nivel acceso información** |  | | |
| **Título** | procedimiento | | |
| **Fichero** | Creación de nuevos atributos de firma.docx | | |
| **Control de copias** | Solamente las copias disponibles e Ubicación de las copias controladas garantizan la actualización de los documentos. Toda copia impresa o dejada en ubicaciones diferentes se considerarán copias no controladas. | | |
| **Derechos de autor** | Atribución-NoComercial-SinDerivadas 2.5 España (CC BY-NC-ND 2.5 ES)  <https://creativecommons.org/licenses/by-nc-nd/2.5/es/deed.es> | | |

Contenido del documento

[Creación de nuevos atributos de firma 1](#_Toc113941437)

[Control documental 2](#_Toc1752927832)

[Contenido del documento 3](#_Toc1615868010)

[1. Introducción 5](#_Toc1142493281)

[1.1 Abstract 5](#_Toc124464291)

[1.2 Estructura del documento 5](#_Toc803868639)

[1.3 Notas sobre el documento 5](#_Toc235047137)

[2. Serialización 6](#_Toc916001930)

[2.1 Implementación de la serialización 6](#_Toc1614083083)

[2.1.1 XAdES 6](#_Toc502218340)

[2.1.2 CAdES/CMS/ESS 7](#_Toc795045413)

[2.2 Generación de interfaces para la abstracción sobre la serialización 7](#_Toc528610389)

[2.2.1 XAdES 7](#_Toc221616442)

[2.2.2 CAdES/CMS/ESS 8](#_Toc635029296)

[2.3 Configuración de beans para las transformaciones 9](#_Toc1599784516)

[2.3.1 XAdES 9](#_Toc373447647)

[2.3.2 CAdES/CMS/ESS 12](#_Toc67611477)

[3. Definición del atributo para los servicios 13](#_Toc2119149427)

[3.1 Generación de interfaces e implementaciones 13](#_Toc1355925548)

[3.1.1 XAdES 13](#_Toc615591902)

[3.1.2 CAdES/CMS/ESS 14](#_Toc1202372624)

[3.2 Configuración del atributo 14](#_Toc1843587854)

[4. Validación del nuevo atributo 18](#_Toc694504088)

[4.1 Creación del adapter y los parameters 18](#_Toc1808458897)

[4.2 Creación del validador 19](#_Toc1915940717)

[4.3 Configurar beans para las validaciones 20](#_Toc367971133)

[5. Creación del nuevo atributo 21](#_Toc1330211890)

[5.1 Creación del adapter y los parameters 21](#_Toc1962146886)

[5.2 Creación del creator 23](#_Toc1939099445)

[5.3 Creación del procesor 24](#_Toc1125148921)

[6. Particularidades 26](#_Toc1045699487)

[6.1 Afecta a la forma 26](#_Toc2053040882)

[6.1.1 XAdES 27](#_Toc730593105)

[6.1.2 CAdES 27](#_Toc514083849)

# Introducción

## Abstract

El objeto de este documento es definir las modificaciones que son necesarias hacer en PSIS para la adición de nuevos atributos de firma. A lo largo del documento se detallará el procedimiento a realizar tanto genérico como el específico para firmas XML y Binarias.

Cada sección vendrá completada con un texto en azul que representa un caso práctico sobre la adición del atributo SigningCertificateV2 dentro del estándar ya soportado XAdES 1.3.2. Dicho atributo viene a actualizar al antiguo SigningCertificate. Además, nos presenta un nuevo tipo, CertIDV2, que también tendremos que tener en cuenta.

## Estructura del documento

En el punto 2 Serialización trataremos el tema de cómo implementar el mecanismo de serialización y deserialización al formato final, así como el mecanismo de interfaces y transformers que abstraerán al resto de servicios de la implementación final.

En el punto 3 Definición trataremos el tema de cómo configurar el atributo para que los distintos servicios de PSIS sepan quién es y cómo tratar con él. Esto incluye la generación de nuevas interfaces para la abstracción sobre la serialización.

En el punto 4 Validación trataremos los pasos a dar para validar un atributo. Explicando la creación de validadores del propio atributo, así como la asociación de estos con el atributo en sí.

En el punto 5 Creación trataremos el tema de la creación del atributo desde cero para la actualización o creación de firmas. Veremos el sistema de Adapters, Parameters, Creators y Procesors que abstraen a cada etapa de la anterior, enfocando cada una al propósito que se definen en su propio nombre. Así mismo, también veremos cómo asociar todo este sistema con el atributo en sí.

En el punto 6.1 Afecta a la forma veremos los cambios necesarios cuando un atributo forma parte de la definición de la forma de una firma.

## Notas sobre el documento

El documento posee varios subdocumentos embebidos con ejemplos de código que no serán exportables en formatos distintos de Word. Son renderizados como cuadros con contorno azul y una previsualización que, al hacer doble clic, abrirá el documento entero en una nueva ventana.

# Serialización

## Implementación de la serialización

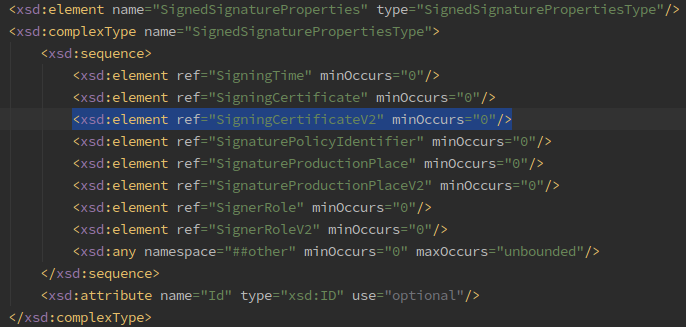
### XAdES

El punto de entrada para la serialización de atributos de documentos en XML son los **xmlbeans**. En PSIS dicho proceso se hace automáticamente mediante el plugin de maven xmlbeans-maven-plugin. Debemos localizar el XSD correspondiente a nuestro estándar dentro del módulo **psis-xmlbeans**. Tenemos dos versiones de XAdES implementadas, por lo que deberemos actualizar el XSD correspondiente XAdES122.xsd o XAdES132.xsd por el nuevo, que descargaremos de la web oficial del estándar. Una vez realizado el cambio en el XSD, debemos recompilar el proyecto para poder tener disponibles las nuevas clases de mapeo **xmlbeans** generadas.

* Añadimos el atributo SigningCertificateV2 al fichero XAdES132.xsd. Vemos como añadimos tanto el elemento como su tipo.



* También modificamos el tipo del padre que lo contendrá añadiendo un nuevo hijo.



### CAdES/CMS/ESS

Como implementación de firmas binarias usamos como proveedor a bouncycastle. Debemos analizar si nuestra versión soporta el nuevo atributo. En caso contrario, habría que revisar si existe una versión posterior a la que hay en uso en PSIS y si contiene dicho atributo. En caso de no ser posible la actualización de bouncycastle habría que implementar manualmente dicho atributo, teniendo como referencia el **rfc** correspondiente. Para ello debemos crear una clase en el paquete psis.structuredmessage.atlantis.transformer.bc, que extienda de ASN1Object. Podemos tomar como referencia la clase psis.structuredmessage.atlantis.transformer.bc.cms.SignedDataImpl.

En este caso también debemos crear una clase Builder que se encargará de serializar nuestra interfaz a formato binario. Dicho builder estará ubicado en el paquete psis.structuredmessage.atlantis.transformer.builder.bc.

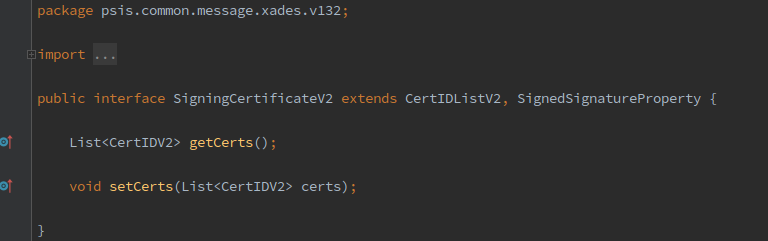
## Generación de interfaces para la abstracción sobre la serialización

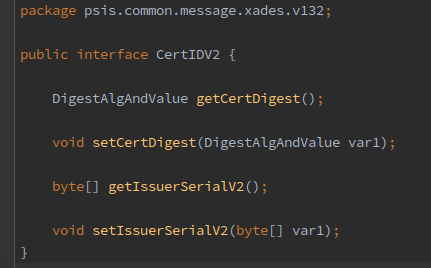
PSIS utiliza una serie de interfaces, que definen a los atributos, para realizar su trabajo. Esto es así para que la codificación de PSIS sea independiente de la tecnología de serialización de los atributos, manteniendo un código homogéneo ajeno al proveedor. Tanto para firmas XML como Binarias el proceso es el mismo, la única diferencia es la localización de dichas interfaces.

### XAdES

Debemos crear una interfaz que exponga los getters y los setters del contenido del nuevo atributo, así como de los nuevos hijos que existan. Dicha interfaz estará localizada en git:/psis-jar/src/main/java/psis/common/message/xades/vXXX, donde XXX es la versión de XAdES donde se implementa el nuevo atributo. Hay que tener en cuenta que debemos actualizar también las interfaces de los atributos padre al nuevo que estamos añadiendo.

* Creamos las interfaces para la mensajería. Implementamos el propio atributo y los hijos que contiene, en este caso CertIDV2. Al ser un atributo firmado, la interfaz debe extender de SignedSignatureProperty para que PSIS lo tome como tal a la hora de realizar las validaciones sobre la forma.





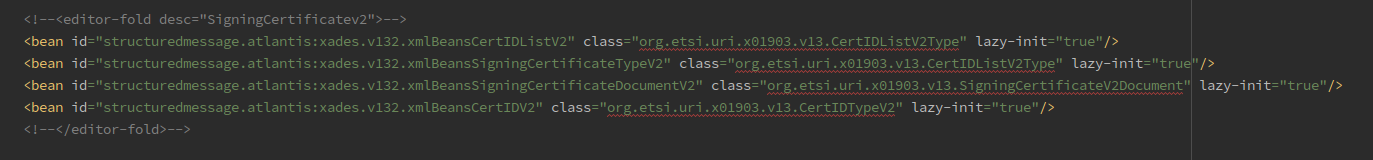
### CAdES/CMS/ESS

Debemos crear una interfaz que exponga los getters y los setters del contenido del nuevo atributo. Dicha interfaz estará localizada en git:/psis-jar/src/main/java/psis/common/message/[cades,cms,ess]/. Hay que tener en cuenta que debemos actualizar también las interfaces de los atributos padre al nuevo que estamos añadiendo.

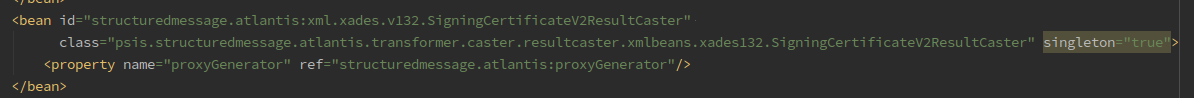
## Configuración de beans para las transformaciones

Localizar los xml de definición de beans para los transformers en git:/psis-jar/src/main/java/psis/structuredmessage/atlantis/transformer/beans y modificar el correspondiente a la versión de nuestro estándar. Debemos añadir un bean por atributo, asignando la clase a la nueva mapeada en la sección 2.1. Importante dejar la inicialización como lazy. Debemos también añadir un nuevo caster en la parte final del fichero si fuera necesario.

* En este caso, tenemos que actualizar el fichero transformer-beans-xades132.xml



* En este mismo fichero, añadimos un caster que será usado por el getter del padre.



Una vez tenemos la instanciación de los beans, debemos crear una configuración que nos permita gestionar la transformación de éstos en instancias de las interfaces para la mensajería. En este caso debemos acudir al fichero Core-v?-YYYYMMDD-xml dentro del módulo **psis-config** y añadir BeanConfigurations por cada uno de los atributos. Esta configuración se encarga de mapear, y transformar si es necesario, cada método que hemos expuesto en la interfaz con los métodos reales del objeto. Así mismo, debemos modificar la de los padres que contendrán al nuevo atributo.

### XAdES

Debemos crear un BeanConfiguration como hijo de xmlBeansHandler y otro como hijo de xmlBeansDSS.

* Primero configuramos el nuevo getter del padre.



* Después creamos un BeanConfiguration hijo de xmlBeansHandler. Habrá que crear otro igual hijo de xmlBeansDSS

<psis:child psis:name="HandlerConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="handlerImplementationId">  
 <psis:stringNodeValue>structuredmessage.atlantis:xmlHandler</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="parameterCasterId">  
 <psis:stringNodeValue>structuredmessage.atlantis:xmlSetCaster</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="resultCasterId">  
 <psis:stringNodeValue>structuredmessage.atlantis:xmlGetCaster</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>xmlBeansHandler</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 <psis:children>  
 <!--<editor-fold desc="SigningCertificatev2">-->  
 <psis:child psis:name="BeanConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="representedImplementationId">  
 <psis:stringNodeValue>structuredmessage.atlantis:xades.v132.xmlBeansSigningCertificateDocumentV2</psis:stringNodeValue>  
 <psis:stringNodeValue>structuredmessage.atlantis:xades.v132.xmlBeansSigningCertificateTypeV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="interfaceTypeName">  
 <psis:stringNodeValue>psis.common.message.xades.v132.SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 <psis:children>  
 <psis:child psis:name="MethodConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="proxyMethod">  
 <psis:stringNodeValue>getCerts</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="methodName">  
 <psis:stringNodeValue>getCertList</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="MethodConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="proxyMethod">  
 <psis:stringNodeValue>setCerts</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="parameterCasterId">  
 <psis:stringNodeValue>structuredmessage.atlantis:ListToArrayParameterCaster</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="methodName">  
 <psis:stringNodeValue>setCertArray</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 </psis:children>  
 </psis:child>  
 <psis:child psis:name="BeanConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="interfaceTypeName">  
 <psis:stringNodeValue>psis.common.message.xades.v132.CertIDListV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="representedImplementationId">  
 <psis:stringNodeValue>structuredmessage.atlantis:xades.v132.xmlBeansCertIDListV2</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 <psis:children>  
 <psis:child psis:name="MethodConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="methodName">  
 <psis:stringNodeValue>getCertList</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="proxyMethod">  
 <psis:stringNodeValue>getCerts</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="MethodConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="parameterCasterId">  
 <psis:stringNodeValue>structuredmessage.atlantis:ListToArrayParameterCaster</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="proxyMethod">  
 <psis:stringNodeValue>setCerts</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="methodName">  
 <psis:stringNodeValue>setCertArray</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 </psis:children>  
 </psis:child>  
 <psis:child psis:name="BeanConfiguration">  
 <psis:attributes>  
 <psis:attribute psis:name="representedImplementationId">  
 <psis:stringNodeValue>structuredmessage.atlantis:xades.v132.xmlBeansCertIDV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="interfaceTypeName">  
 <psis:stringNodeValue>psis.common.message.xades.v132.CertIDV2</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <!--</editor-fold>-->

### CAdES/CMS/ESS

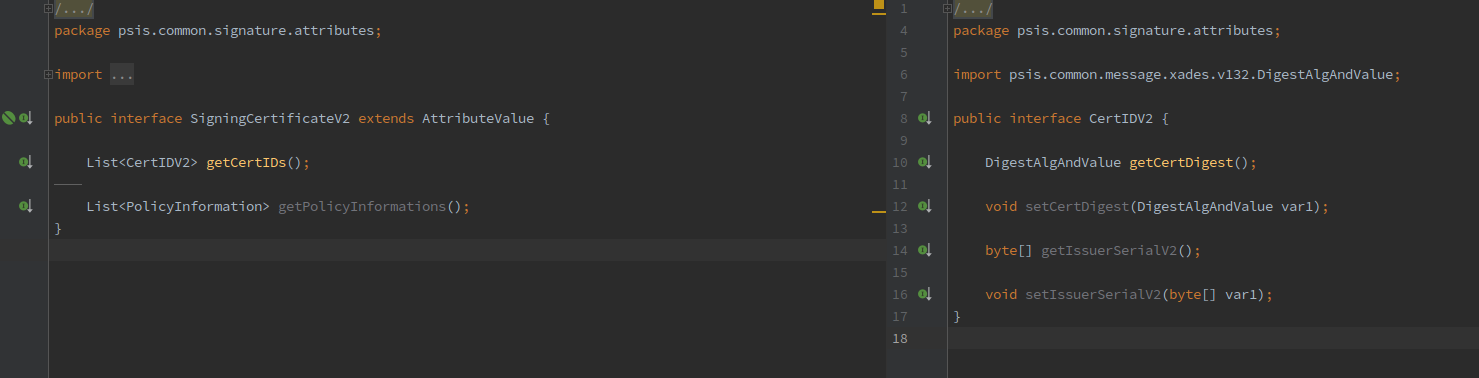
Debemos crear un BeanConfiguration como hijo de derHandler, otro como hijo de xmlBeansHandler y otro como hijo de xmlBeansDSS.

# Definición del atributo para los servicios

## Generación de interfaces e implementaciones

Al igual que con las interfaces para la mensajería interna, ver punto 2.2, debemos crear otras similares en psis.common.signature.attributes. Como diferencia notable tenemos que la implementación esta vez corre a cargo de nuestra cuenta. Dicha implementación actuará como un simple wrapper de los objetos de mensajería, cuyos getters y setters obtendrán sus datos de éste. Debemos luego configurar la instanciación en forma de beans de sus implementaciones en el fichero git:/psis-jar/src/main/java/psis/signaturebuilder/atlantis/beans.xml

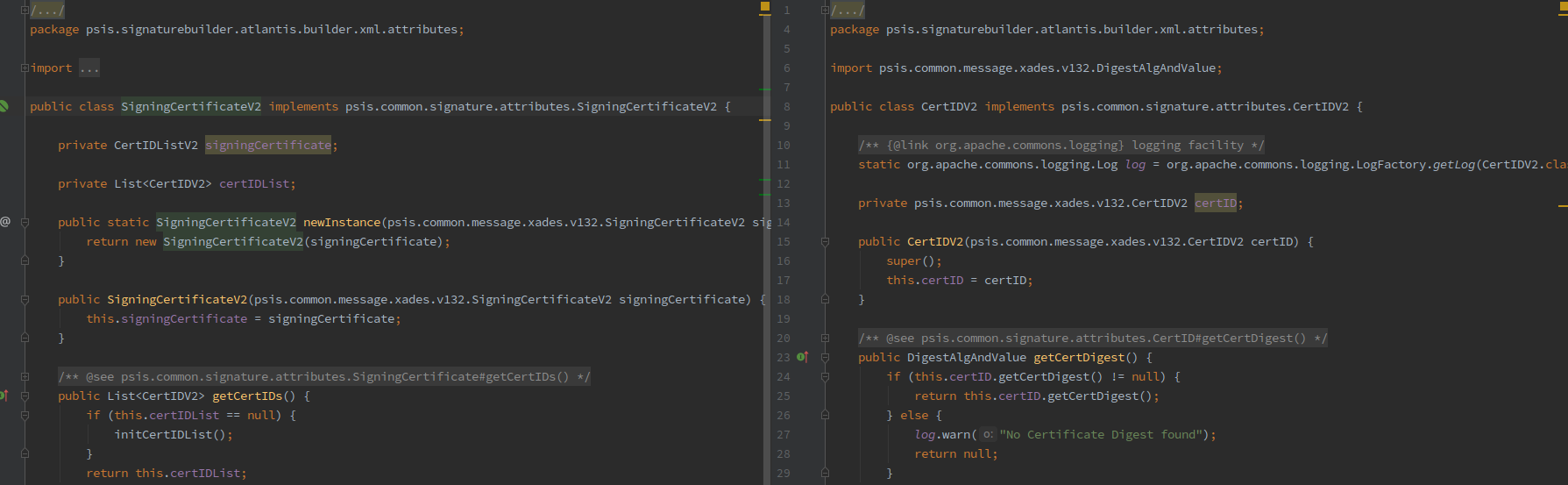
* Generamos dichas interfaces.



### XAdES

En el paquete psis.signaturebuilder.atlantis.builder.xml.attributes debemos implementar la interfaz creada, que tendrá como parámetro del constructor la interfaz usada para la mensajería, así como un método factory llamado “newInstance”

* Generamos las implementaciones.



* Configuramos los beans



### CAdES/CMS/ESS

En el paquete psis.signaturebuilder.atlantis.builder.asn1.attributes debemos implementar la interfaz creada, que tendrá como parámetro del constructor la interfaz usada para la mensajería, así como un método factory llamado “newInstance”

## Configuración del atributo

Psis posee servicios de alto nivel para la validación, creación

Debemos actualizar el registro con una nueva entrada para dicho atributo. Este cambio se hace en el fichero Organization-v?-YYYYMMDD.xml dentro del módulo **psis-config**. Debemos añadir una entrada en Registry con el “link” de nuestro nuevo atributo. Este link hará referencia a una nueva entrada que también debemos crear, esta vez bajo Identifiers/Signatures. Esta será la configuración que asociará nuestro atributo a su URI en xml o a su OID en binario. Además de esto aquí definiremos su cardinalidad, si es firmado o no firmado y los wrappers que usaremos para XAdES y CAdES.

* Añadimos la entrada en Registry, con link al atributo que crearemos justo después



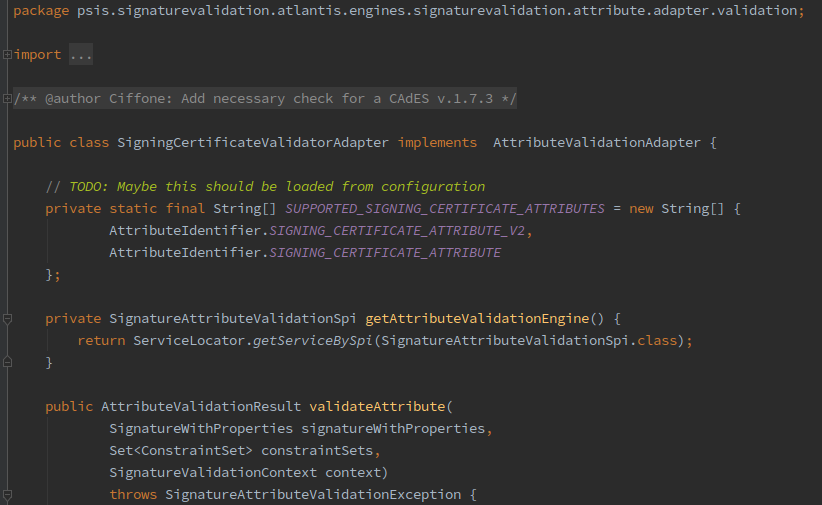
<psis:child psis:name="Key">  
 <psis:attributes>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>psis.common.attribute.AttributeIdentifier</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 <psis:children>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>psis.common.datatype.def.OID</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>oid</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>true</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>1.2.840.113549.1.9.16.2.47</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>uri</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.net.URI</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>true</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>http://uri.etsi.org/01903/v1.3.2#SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>false</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>structuredmessage.atlantis:asn1.ess.SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.String</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>cadesImpl</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>cadesWrapper</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>false</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>signaturebuilder.attribute:asn1.SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.String</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>signaturebuilder.attribute:xml.SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>xadesWrapper</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>false</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.String</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>false</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.String</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>SIGNED</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>verification</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>UNIQUE</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>cardinality</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>false</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.String</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>false</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>1</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>weight</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.Integer</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 <psis:child psis:name="Property">  
 <psis:attributes>  
 <psis:attribute psis:name="name">  
 <psis:stringNodeValue>xadesName</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="identifier">  
 <psis:stringNodeValue>true</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="value">  
 <psis:stringNodeValue>http://uri.etsi.org/01903#:SigningCertificateV2</psis:stringNodeValue>  
 </psis:attribute>  
 <psis:attribute psis:name="class">  
 <psis:stringNodeValue>java.lang.String</psis:stringNodeValue>  
 </psis:attribute>  
 </psis:attributes>  
 </psis:child>  
 </psis:children>  
</psis:child>

# Validación del nuevo atributo

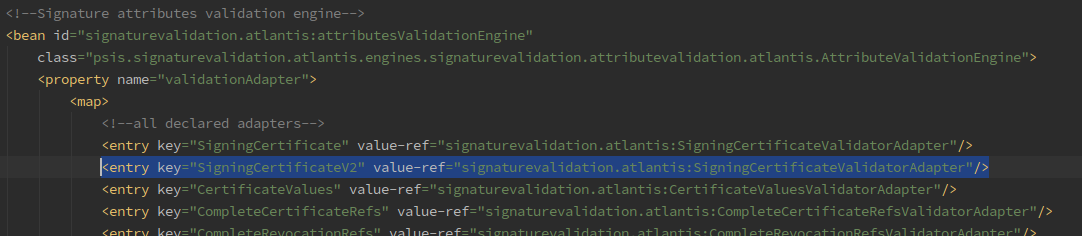
## Creación del adapter y los parameters

Debemos crear un adapter en el paquete psis.signaturevalidation.atlantis.engines.signaturevalidation.attribute.adapter.validation que implemente AttributeValidationAdapter. Dicho adapter será el encargado de extraer del contexto los datos necesarios para la validación del atributo al que referencia y dejarlos en una nueva clase parameters en psis.signatureattributemanager.creation.atlantis.parameters, que será compartida con el validador. Tenemos que instanciar dicho Adapter en el fichero git:/psis-jar/src/main/java/psis/signaturevalidation/atlantis/engines/signaturevalidation/attribute/adapter/adapters.xml. Debemos registrar dicho adapter en el servicio de actualización de firmas AttributeValidationEngine configurado en git:/psis-jar/src/main/java/psis/signaturevalidation/atlantis/engines/signaturevalidation/beans.xml.

* Creamos el adapter



* Lo registramos en el motor de validación



## Creación del validador

Para la validación de un nuevo atributo debemos crear un validator en git:/psis-jar/src/main/java/psis/signatureattributemanager/validation/atlantis/validators. Dicho validator deberá extender de SignedValidator o UnsignedValidator según el atributo sea firmado o no. Si la validación de dicho atributo es diferente en XAdES que en CAdES debemos crear dicho validador como abstracto, intentando contener en él toda la lógica común, y crear dos validadores en los subpaquetes xml y cms.

* Creamos los validadores, tanto el padre abstracto como la implementación XAdES y CAdES. (se adjuntan las clases completas)

Clase abstracta padre:



Implementaciones:

Implementaciones



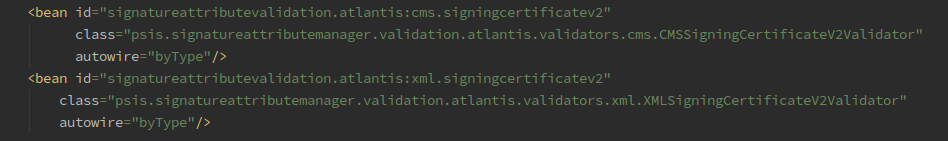


## Configurar beans para las validaciones

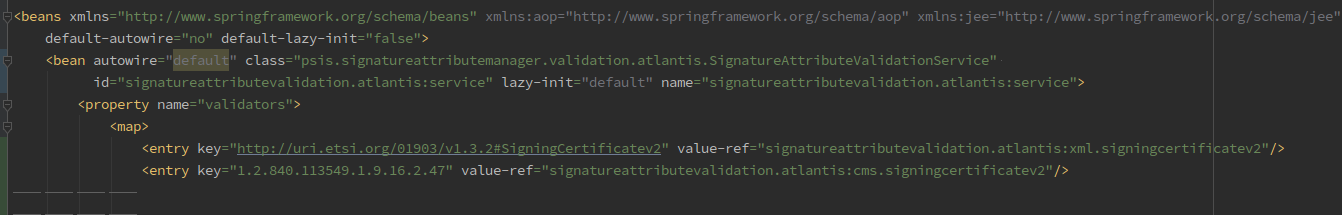
Debemos definir los beans de los validadores en el fichero beans.xml de git:/psis-jar/src/main/java/psis/signatureattributemanager/validation/atlantis

En ese mismo directorio en el fichero signature-attribute-validation-engine-beans.xml debemos asociar los beans con el atributo. Ésto se hace relacionando el bean con el OID del atributo en CAdES o con el URI en XAdES.

* Añadimos los beans de los validadores.



* Mapeamos los validadores con sus atributos, usando como clave su URI y OID, dentro de la propiedad “validators” del SignatureAttributeValidationService. Vemos las 2 entry, la de la URI y la del OID.

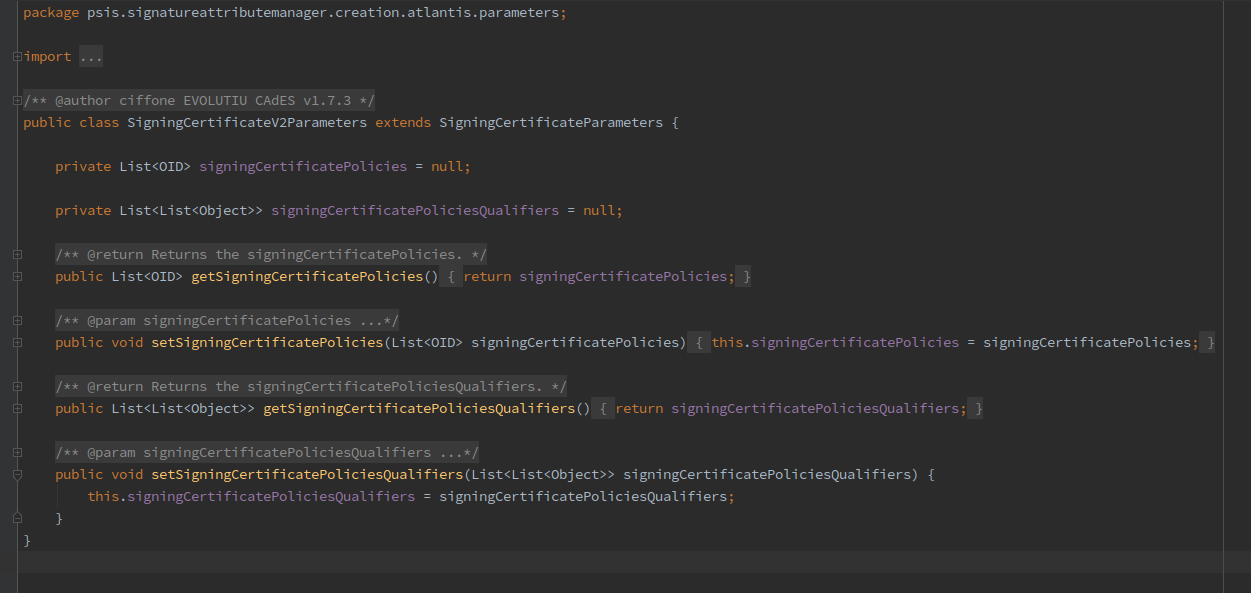


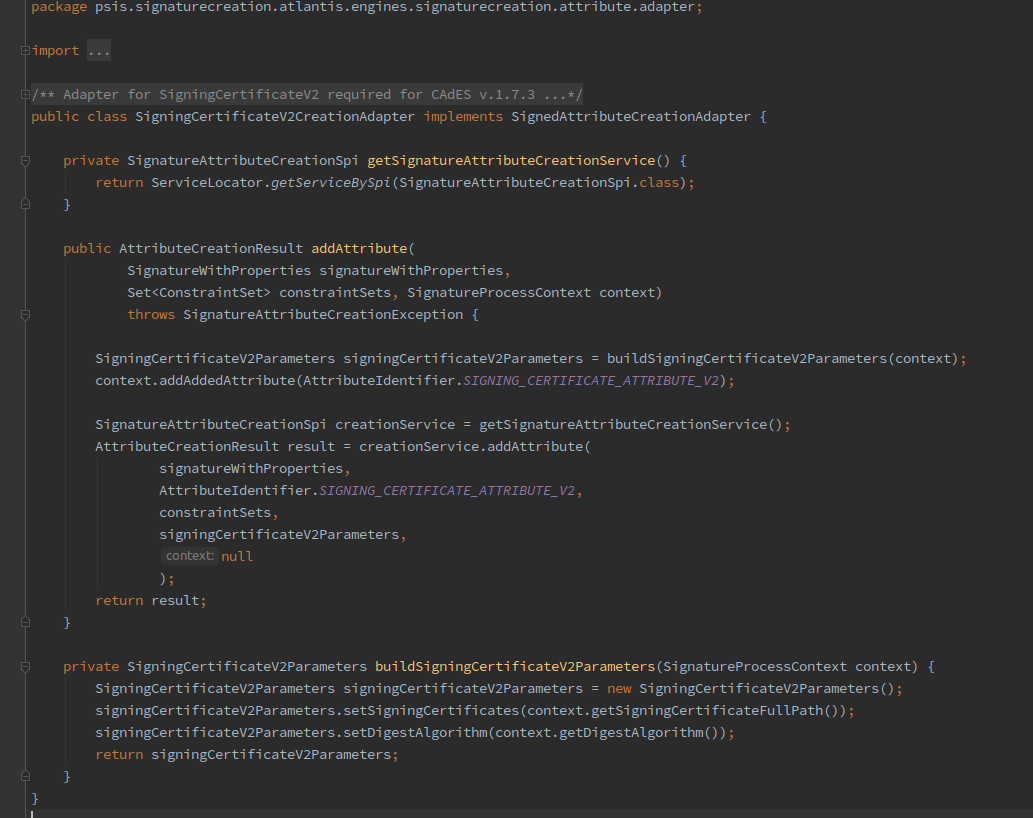
# Creación del nuevo atributo

## Creación del adapter y los parameters

Debemos crear un adapter en el paquete psis.signaturecreation.atlantis.engines.signaturecreation.attribute.adapter que implemente SignedAttributeCreationAdapter o UnsignedAttributeCreationAdapter según corresponda. Dicho adapter será el encargado de extraer del contexto los datos necesarios para la construcción del atributo al que referencia y dejarlos en una nueva clase parameters en psis.signatureattributemanager.creation.atlantis.parameters, que será compartida con el creator. Importante, normalmente se usa la misma clase de parameters que para el validator. Si el atributo es firmado, tenemos que instanciar dicho Adapter en el fichero git:/psis-jar/src/main/java/psis/signaturecreation/atlantis/engines/signaturecreation/attribute/adapter/beans.xml. Si el atributo es no firmado, tenemos que instanciar dicho Adapter en el fichero git:/psis-jar/src/main/java/psis/signaturevalidation/atlantis/engines/signaturevalidation/attribute/adapter/adapters.xml. Si el atributo es firmado, debemos registrar dicho adapter en los dos servicios de actualización de firmas SignatureUpdatingEngine configurados en git:/psis-jar/src/main/java/psis/signaturecreation/atlantis/beans.xml. Si el atributo es no firmado, debemos registrar dicho adapter en el servicio de actualización de firmas SignatureUpdatingEngine configurado en git:/psis-jar/src/main/java/psis/signaturevalidation/atlantis/engines/signaturevalidation/beans.xml.

* Creamos el Parameters y el adapter.

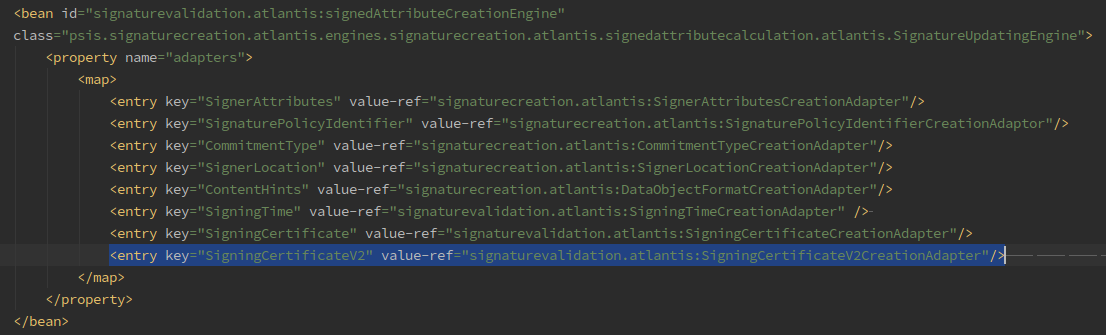


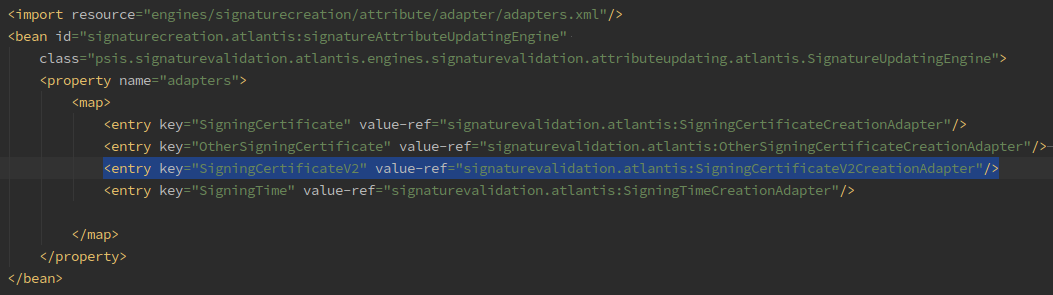


* Añadimos el bean.



* Lo registramos en los servicios de actualización de firma.

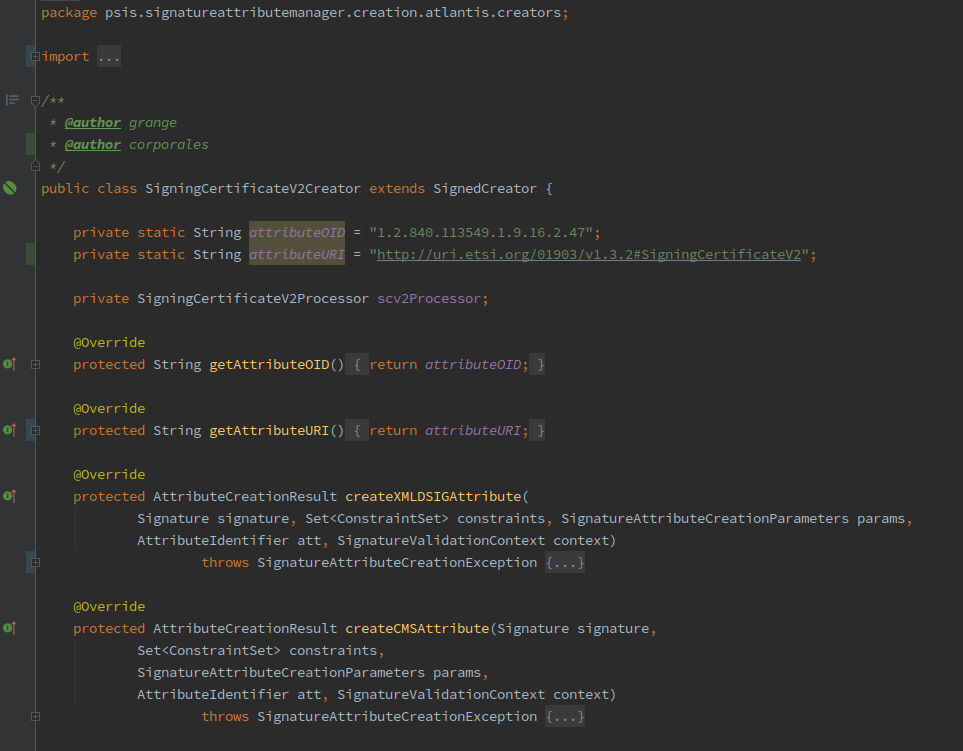




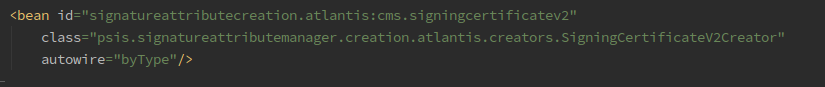
## Creación del creator

El creator será la clase encargada de crear el atributo final a partir de los datos que el adapter nos ha dejado en los parameters. Este creator, que estará ubicado en psis.signatureattributemanager.creation.atlantis.creators deberá extender de SignedCreator o UnsignedCreator según corresponda. Dicho creator tendrá dos grandes puntos de entrada: createXMLDSIGAttribute y createCMSAttribute encargados de crear el atributo en xml o en binario. Tenemos que instanciar el creator en el fichero git:/psis-jar/src/main/java/psis/signatureattributemanager/creation/atlantis/beans.xml. El creator deberemos registrarlo como tal en el servicio SignatureAttributeCreationService en el fichero git:/psis-jar/src/main/java/psis/signatureattributemanager/creation/atlantis/signature-attribute-creation-engine-beans.xml como una entrada del mapa de la property creators. Dicha entrada tendrá como clave su URI en el caso de XAdES y su OID en el caso de CAdES/CMS/ESS.

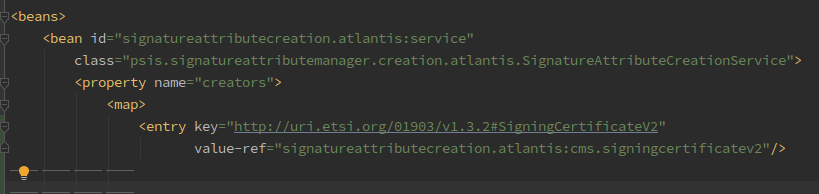
* Creamos la clase del creator.



* Añadimos el bean.



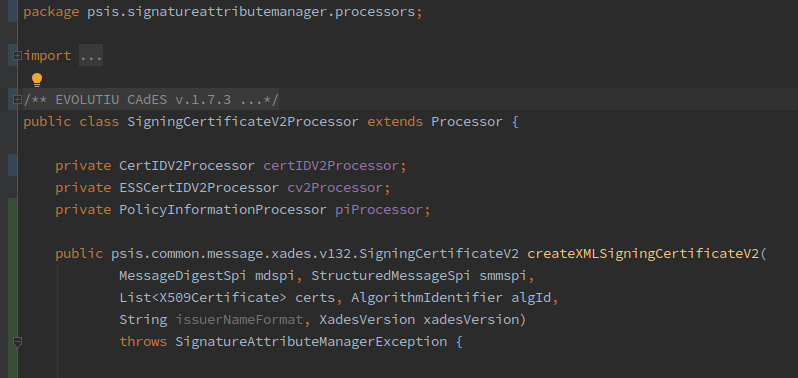
* Lo registramos usando como clave su URI.

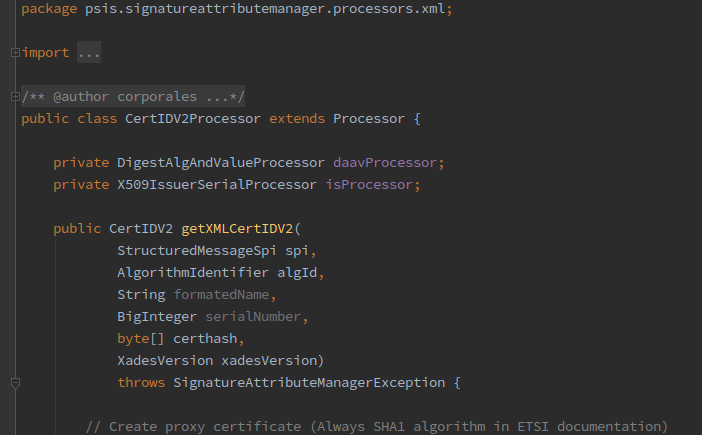


## Creación del procesor

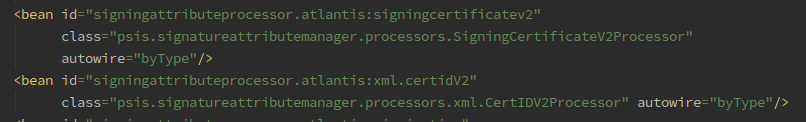
El procesor se encagra de crear cada atributo específico, encargando a otros procesor los atributos hijos que puedan existir. Este procesor estará ubicado en psis.signatureattributemanager.processors y deberá extender de Processor. Dichos procesor debemos instanciarlos en el fichero git:/psis-jar/src/main/java/psis/signatureattributemanager/processors/beans.xml.

* Creamos el procesor para SigningCertificateV2, así como para CertIDV2.





* Añadimos los beans



# Particularidades

A continuación, se detallan una serie de particularidades que pueden tener el atributo y que añaden o modifican comportamientos a la hora de su implementación.

## Afecta a la forma

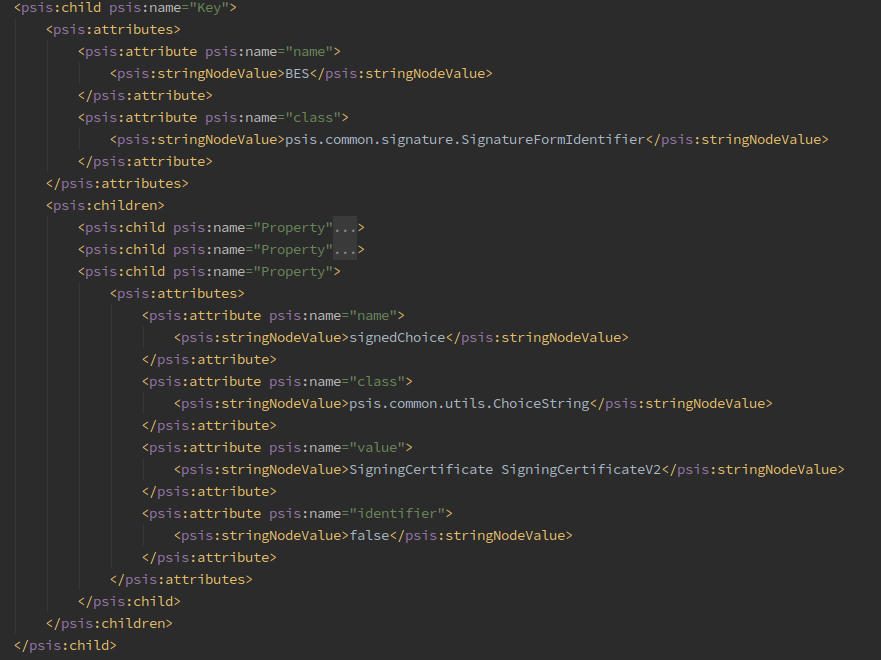
En el fichero Organization-v?-YYYYMMDD.xml dentro del módulo **psis-config**. debemos añadir entradas por cada forma afectada definida en el fichero. En caso de que actualice a otro atributo existente, debemos sustituir este por un signedChoice/unsignedChoice según corresponda, para que psis, a la hora de identificar la forma, reintente con el nuevo atributo en caso de no encontrar el antiguo.

* Modificamos las formas para incluir el SigningCertificateV2. Mostramos solamente ejemplo para la BES, del antes y el después.

Antes:



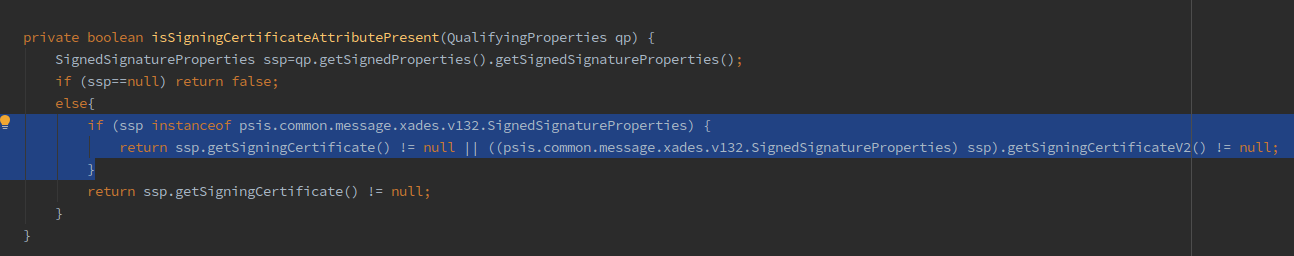
Después:



### XAdES

En caso de que un mismo atributo funcional cambie entre versiones del estándar es posible que debamos tenerlo en cuenta en el validador semántico. Si es el caso, debemos adaptar el método psis.common.signature.xades.AttributeSemanticValidator.validateXadesAttributes() para que tenga en cuenta la nueva versión del atributo.

* Añadimos, a la aserción de que exista el antiguo SigningCertificate, que pueda existir también el nuevo SigningCertificateV2



### CAdES

En caso de que un mismo atributo funcional cambie entre versiones del estándar es posible que debamos tenerlo en cuenta en el validador semántico. Si es el caso, debemos adaptar el método psis.common.signature.cades.AttributeSemanticValidator.validateCADESAttributes() para que tenga en cuenta la nueva versión del atributo.