



Junta de Andalucía



Cofinanciado por
la Unión Europea



GOBIERNO
DE ESPAÑA



Fondos Europeos



Junta
de Andalucía

Plataforma @firma

Manual Integrador Cliente DSS

Versión: v01r20
Fecha: 29/07/2025

HOJA DE CONTROL

Título	Plataforma @firma		
Entregable	Plan de pruebas Cliente DSS		
Nombre del Fichero	20250729-Manual Integrador cliente DSS_v01r20 - copia.docx		
Autor	DGED		
Versión/Edición	v01r19	Fecha Versión	29/07/2025
Aprobado por		Fecha Aprobación	
		Nº Total Páginas	37

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Area	Fecha del Cambio
v01r00	Creación del documento	UTE	UTE	21/12/2012
v01R02	Revisión del documento	UTE	UTE	27/12/2012
v01R02	Revisión del documento	SCAE	DGPD	22/02/2013
v01R03	Revisión del documento	UTE	UTE	11/04/2013
v01R04	Revisión del documento	UTE	UTE	29/01/2014
v01R05	Se indica la URL del repositorio de artefactos Maven de CHAP en el apartado 3.3	UTE	UTE	12/05/2014
v01r06	Configuración avanzada	UTE	UTE	21/07/2014
v01r07	Corregido apartado 3.6	UTE	UTE	02/12/2014
v01r08	Se incluyen en el apartado 3.6 los métodos getResult() y getResultMessage() para los 'beans' de respuesta.	UTE	UTE	11/12/2014

v01r09 (afirma- dss-client- 1.1.0)	Eliminado acceso al servicio DSSAfirma ArchiveSubmit. Incluido parámetro afirma.hashalgorithm.	UTE	UTE	28/04/2015
v01r10 (afirma- dss-client- 1.1.3)	Se permite el uso del almacén de confianza por defecto de la JRE (apartado 3.4).	UTE	UTE	26/01/2016
v01r11 (afirma- dss-client- 1.1.4)	Se permite el upgrade de multifirmas en @firma 6.	UTE	UTE	07/11/2016
v01r12 (afirma- dss-client- 1.1.5)	Nuevo método en la API para hacer upgrade de un firmante (solo @firma6).	UTE	UTE	11/11/2016
v01r13 (afirma- dss-client- 1.1.7)	Nuevo método en la API para hacer upgrade de un firmante con sólo dos parámetros (solo @firma6).	UTE	UTE	02/02/2017
v01r14 (afirma- dss-client- 1.1.8)	Nuevo método en la API que comprueba las firmas Xades Explícitas. Nuevo método en la API que devuelve el algoritmo de hash de una firma dada.	UTE	UTE	05/01/2018
v01r15 (afirma- dss-client- 1.2)	Se corrige la detección del algoritmo de hash en las firmas PadES. En esta versión ya no es obligatorio que se indique el tipo de firma cuando se utilizan los métodos dssAfirmaVerify(String sign) y dssAfirmaVerify(String sign, String document, String hashDocument), de forma que si no se indica, el tipo de la firma es detectado automáticamente. Nuevo método de verificación cuyo	UTE	UTE	06/11/2018

	último parámetro sirve para indicar el algoritmo de hash utilizado para calcular el hash que se le envía en el tercer parámetro.			
v01r16 (afirma- dss-client- 1.3)	Se añade un nuevo método para separar una cadena con los dos apellidos en dos cadenas con un apellido cada una. separarApellidos(String apellidos). Cambios para que al resolver el tipo y modo de la firma, si es XAdES-Enveloping decodifique la firma. Nuevos tipos de firma añadidos para contemplar los nuevos formatos Baseline.	UTE	UTE	15/03/2021
v01r17	Se añade un nuevo formato CMSWithTST, para poder actualizar firmas CMS.	UTE	UTE	21/06/2021
v01r18	Se corrige un error en las funciones getResult y getResultMajor por el que en determinados casos se devolvía null, en lugar del mensaje correcto.	UTE	UTE	03/12/2021
v01r19	Se modifica la dependencia con la librería commons-logging. Ahora depende de la versión 1.2 ya que la v1.1 tenía dependencia con log4j.1.2.12. Se corrigen los valores de algunas constantes referentes a versiones de formatos Baseline.	UTE	UTE	11/04/2022
v01r20	Nueva versión 2.0.1 del cliente dss.	UTE	UTE	29/07/2025

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Area	Nº Copias
José Ignacio Cortés Santos	Director de Proyecto	ADA	1

ÍNDICE

1.	Introducción	8
2.	Servicios Web DSS de @firma	9
2.1	El Estándar DSS de OASIS	9
2.2	Catálogo de Servicios	10
2.3	Documentación de Referencia	11
3.	Integración con Afirma-DSS-Client	12
3.1	Dependencias Software	12
3.2	Dependencias de Servicios	12
3.3	Establecer Dependencias con Maven	13
3.4	Fichero de Propiedades	14
3.5	Interfaz del Componente afirma-dss-client	18
3.5.1	DSSAfirmaSign	19
3.5.2	DSSAfirmaCoSign	20
3.5.3	DssAfirmaCounterSign	21
3.5.4	DSSAfirmaVerify	22
3.5.5	DssAfirmaUpgrade	25
3.5.6	DSSAfirmaVerifyCertificate	27
3.5.7	DssAfirmaBatchVerifySignature	27
3.5.8	DSSAfirmaBatchVerifyCertificate	28
3.5.9	DssAfirmaAsyncRequestStatus	28
3.5.10	HashDetector	29
3.5.11	SepararApellidos	29
3.6	Clases de Respuesta	30
3.7	Ejemplos de integración	33
3.7.1	Validación de certificado (DSSAfirmaVerifyCertificate)	33
3.7.2	Actualización de Firma (DSSAfirmaUpgrade)	33

4.	Formatos Baseline y Correspondencia con Formatos Tradicionales	35
----	--	----

1. INTRODUCCIÓN

El componente **afirma-dss-client** tiene como objetivo fundamental facilitar a los integradores la interacción con los servicios DSS disponibles en la plataforma @firma. Se trata de un componente software que implementa la lógica necesaria relacionada con la mensajería (basada en los estándares DSS de OASIS) y las comunicaciones (SOAP), ofreciendo a los integradores una interfaz sencilla.

Afirma-dss-client está desarrollado en Java y está construido con Maven siguiendo las recomendaciones de MADEJA.

Funcionalmente, **afirma-dss-client** se divide en dos partes funcionales:

1. **Composición y procesamiento de mensajes XML:** Realiza la composición de la petición y el procesamiento de la respuesta XML implementando la lógica de mensajes basada en los estándares DSS de OASIS. Esta composición de los mensajes XML se ha llevado a cabo siguiendo la especificación detallada en el documento “Manual de programación de servicios web estándar avanzados DSS” de la plataforma @firma v6.
2. **Envío y recepción de mensajes SOAP:** El cliente incluye la implementación del envío y recepción del mensaje SOAP que contiene la petición y respuesta DSS respectivamente, haciendo este proceso transparente para el integrador. El cliente utiliza librerías de código abierto como Apache-CXF y Web Service Security para realizar el intercambio de información con el servicio web de @firma.

Este manual presenta la primera funcionalidad y describe con detalle la interfaz proporcionada por **afirma-dss-client**, cuya finalidad es el acceso a cada uno de los servicios DSS que ofrece @firma. Se incluye también la correspondencia con los servicios nativos de @firma.

2. SERVICIOS WEB DSS DE @FIRMA

2.1 El Estándar DSS de OASIS

El estándar **Digital Signature Service (DSS)** de OASIS define protocolos de firma, verificación y custodia basados en interfaces XML. Las especificaciones DSS están en continuo desarrollo y definen multitud de casos de uso, agrupados por perfiles, para distintos contextos. Utilizando estas interfaces, un cliente puede interactuar con cualquier servidor que cumpla con los perfiles definidos.

Aunque los perfiles DSS se encuentran en un estado de madurez óptimo, no existe uno que se adapte al cien por cien a los servicios que ofrece actualmente la plataforma @firma v6. Por este motivo, la plataforma define nuevos perfiles basados en algunos ya definidos por OASIS, añadiendo elementos nuevos para recoger aquellos aspectos no contemplados en el estándar.

Los perfiles DSS en los que se basan los servicios de @firma v6 se refieren a **Extended Signature Services (XSS)** según las especificaciones [DSS XSS] para los servicios de firma y verificación de firma, y **Signature Archive Profile** con las características recogidas en [DSS Archive] para los servicios de custodia. Estos perfiles han sido definidos e implementados por CATCERT y están disponibles en:

- **Entorno de Desarrollo:** <https://des-afirma.redsara.es/afirmaws/xsd/dss/oasis-dss-1.0-profiles-XSS-schema-wd02.xsd>
- **Entorno de Producción:** <https://afirma.redsara.es/afirmaws/xsd/dss/oasis-dss-1.0-profiles-XSS-schema-wd02.xsd>

Además de los perfiles anteriormente expuestos, las especificaciones XSS de @firma se basan en el perfil **Comprehensive Multi-Signature Verification Reports** recogido en [DSS VR] y disponible en la Web (http://www.oasis-open.org/committees/download.php/28182/2008_05_05_oasis-dss-profile-for-comprehensive-signature-verification-report.doc) para solventar las limitaciones del protocolo de verificación en los procesos de verificación de multifirmas.

El componente **afirma-dss-client** implementa aquellos elementos que se incluyen en los perfiles anteriores, formando los nuevos perfiles de aplicación para la plataforma @firma. La composición de estas estructuras XML por parte de **afirma-dss-client** abstrae al

integrador de la especificación DSS comentada y permite la interacción rápida y sencilla con los servicios web DSS que se describen a continuación.

2.2 Catálogo de Servicios

En este apartado se detallan los Servicios Web que publica la plataforma @firma v6.4, siguiendo las recomendaciones **Digital Signature Service (DSS)** de OASIS. Estos servicios son:

- **DSSAfirmaSign:** Servicio de Firma Delegada que permite la realización de operaciones de firma y multifirma de servidor en los formatos soportados por la plataforma. (Este servicio está deprecado y no se recomienda su uso ya que dejará de estar disponible en una próxima actualización de la plataforma @firma).
- **DSSAfirmaVerify:** Servicio de Verificación y Actualización de Firma. Permite la verificación de una firma electrónica y la obtención de información adicional sobre dicha firma. El servicio de verificación permitirá, además, de forma opcional, la realización de operaciones de upgrade sobre una firma electrónica, pudiendo completar dicha firma a un formato más avanzado.
- **DSSAfirmaVerifyCertificate:** Servicio de Verificación de Certificado. Mediante esta interfaz Web Service se puede solicitar la verificación de certificados X509 así como extraer la información del certificado mediante la aplicación del mapeo definido para su tipo.
- **DSSBatchVerifyCertificate:** Servicio de Validación Masiva de Certificados. Mediante este servicio se puede solicitar la verificación asíncrona de un conjunto de certificados.
- **DSSBatchVerifySignature:** Servicio de Validación Masiva de Firmas. Mediante este servicio se puede solicitar la verificación asíncrona de un conjunto de firmas.
- **DSSAsyncRequestStatus:** Servicio de Consulta de Peticiones Asíncrona. Este servicio permite la consulta y obtención de respuesta de peticiones asíncronas.

No forma parte del objetivo de este documento describir los componentes que forman las interfaces XML de petición y respuesta para cada uno de los anteriores servicios.

2.3 Documentación de Referencia

El componente **afirma-dss-client** se ha construido basándose en la especificación de los servicios DSS que se presentan en el documento **@Firma-Global-XMLSOAP_PROFILE_DSS.pdf** disponible en el kit del integrador que facilita la SGAD para @firma.

Ese documento define los perfiles para la adaptación de los servicios web de @firma al protocolo OASIS-DSS que implementa **afirma-dss-client**.

3. INTEGRACIÓN CON AFIRMA-DSS-CLIENT

3.1 Dependencias Software

El componente **afirma-dss-client** está implementado en Java y compilado con JDK 21. Son necesarias una serie de dependencias (ficheros JAR) que deben estar en el classpath de ejecución de la librería principal para su correcto funcionamiento. La librería que implementa el cliente se denomina **afirma-dss-client-2.x.x.jar** y las dependencias necesarias son las siguientes:

- commons-lang:commons-lang:jar:2.1
- commons-logging:commons-logging:jar:1.3.5
- com.fasterxml.jackson.core:jackson-core:jar:2.15.0
- com.itextpdf:itextpdf:jar:5.5.12
- iaik:iaik_jce:jar:3.181
- iaik:iaik-cms:jar:4.1
- javax.mail:mail:jar:1.4
- org.apache.cxf:cxf-rt-transports-http:jar:4.1.2
- org.apache.cxf:cxf-rt-frontend-jaxws:jar:4.1.2
- org.apache.cxf:cxf-rt-transports-http-undertow:jar:4.1.2
- org.apache.cxf:cxf-rt-transports-http-undertow:jar:4.1.2
- org.apache.ws.commons.util:ws-commons-util:jar:1.0.2
- xerces:xercesImpl:jar:2.12.2

3.2 Dependencias de Servicios

Para el correcto funcionamiento de **afirma-dss-client** es necesario que los servicios web de @firma @firma v6 sean visibles desde la máquina donde se pretende ejecutar el componente, tanto el host como el puerto.

3.3 Establecer Dependencias con Maven

El componente afirma-dss-client puede ser incluido en aplicaciones desarrolladas en Maven añadiendo la siguiente dependencia al pom.xml de la aplicación:

```
<dependency>
  <groupId>es.juntadeandalucia.afirma</groupId>
  <artifactId>afirma-dss-client</artifactId>
  <version>2.x.x</version>
</dependency>
```

Este componente, y todos los desarrollados por la Consejería de Hacienda y Administración Pública, pueden encontrarse en el Repositorio Maven del Servicio de Coordinación de Administración Electrónica (<https://ws024.juntadeandalucia.es/ae/adminelec/repositoriomaven>).

Para que Maven obtenga el artefacto que contiene el componente y resuelva automáticamente sus dependencias, es necesario incluir el siguiente fragmento XML en el fichero pom.xml correspondiente:

```
<repositories>
  <repository>
    <id>SCAERepo</id>
    <url>https://ws024.juntadeandalucia.es/maven/repository/internal</url>
  </repository>
</repositories>
```

NOTA: El Repositorio Maven del Servicio de Coordinación de Administración Electrónica de la Consejería está accesible a través del protocolo HTTPS utilizando el certificado cn=*.juntadeandalucia.es, o=Junta de Andalucía, l=Sevilla, s=sevilla c=ES de la Junta de Andalucía. Para que Maven confíe en el dominio ws024.juntadeandalucia.es a través de HTTPS y logre descargar el componente y sus dependencias, es necesario que incluya el certificado indicado en el almacén de certificados por defecto (cacerts) de la Máquina Virtual Java sobre la que se ejecute el script mvn.

3.4 Fichero de Propiedades

Con el objetivo de facilitar la integración, se establece un fichero de propiedades que contiene los argumentos estáticos de las peticiones a realizar. Dicho fichero contiene, por un lado, argumentos obligatorios como el identificador de aplicación de @firma, el usuario y la contraseña de la aplicación o los datos de autenticación mediante certificado digital, y los datos del servidor. Por otro lado, el fichero de propiedades permite incluir argumentos propios de las peticiones que no van a sufrir modificaciones. Por ejemplo, si todas las firmas que la aplicación cliente va a realizar o validar son de tipo XAdESv1.3.2, se asigna a la propiedad *afirma.signaturetype* el valor XAdESv1.3.2 y así no es necesario incluir este argumento en las llamadas a la interfaz de **afirma-dss-client** que lo requieran.

A continuación, se muestra un ejemplo del fichero *afirma.properties* que debe incluirse en el classpath donde se ejecute el cliente:

```
#####  
# Configuración del componente afirma-dss-client  
#####  
#####  
# Credenciales de la aplicación (obligatorio)  
#  
# Si se establecen las propiedades de autenticación mediante BinarySecurityToken  
# (certificado) este tipo de autenticación prevalece respecto a la autenticación  
# mediante UsernameToken (usuario / contraseña).  
#####  
afirma.idapp = IDAPP  
#####  
# BinarySecurityToken  
afirma.authorization.ks.path = PATH_PKCS12  
  
# Tipo del almacén anterior: JKS,PKCS12  
afirma.authorization.ks.type = PKCS12  
afirma.authorization.ks.password = usr  
afirma.authorization.ks.cert.alias = pas  
#####  
# UsernameToken  
afirma.user = user  
afirma.password = pass  
#####  
#####
```

```
# Datos del servidor (obligatorio)
#
# El componente siempre realizará las peticiones al siguiente endpoint:
# https://<afirma_host>/afirmaws/services/<nombre_del_servicio>
#####
afirma.host = ws028.juntadeandalucia.es
#####
#####
# Almacén de certificados de confianza (opcional)
#
# El componente presupone que el almacén de certificados de confianza está
# incluido en el CLASSPATH de la aplicación, por lo que en 'afirma.truststore'
# debe indicarse el nombre del fichero JKS (o bien, la ruta relativa al mismo
# dentro del CLASSPATH). Si se establece el valor "default" se hace uso del
# almacén de confianza de la JRE que ejecuta la aplicación (cacerts)
#
# Por ejemplo, una aplicación web (empaquetada en un WAR) cuyo almacén está
# situado en el directorio WEB-INF/classes/almacenconfianza.jks, deberá
# establecer la propiedad afirma.truststore de la siguiente manera:
#
# También se admite referenciar al almacén de confianza mediante su ruta absoluta.
#
# afirma.truststore = almacenconfianza.jks
#####
afirma.truststore = default
afirma.truststorePassword =
#####
#####
# Tipo de firma utilizado (opcional)
#
# Los tipos permitidos son: PKCS7
# v1.5, CMS, CMS-T, XMLSignature, CAdES,
# XAdESv1.3.2, XAdESv1.2.2, XAdESv1.1.1, ODF, ODF-T, PDF, PAdES
#####
afirma.signaturetype = XAdESv1.3.2
#####
#####
# Modalidad de firma XML (opcional)
#
# En el caso que la firma a generar pertenezca a algunos de los formatos de la
# familia de firmas en XML (XMLSignature, XAdES), el cliente puede incluir el
# el modo de la firma a generar.
#
# Los modos permitidos son: ENVELOPING, ENVELOPED, DETACHED
#####
```

```
afirma.xmlsignaturemode = ENVELOPING
#####
#####
# Formato de firma avanzado (opcional)
#
# Por defecto, el componente presupondrá que se trabaja con el formato BES
# (siempre que sea aplicable al tipo de firma configurado en la propiedad
# afirma.signaturetype)
#
# Los formatos permitidos son: BES, EPES, T, C, X, X-1, X-2, X-L, X-L-1,
# X-L-2, A
#####
afirma.signatureform = BES
#####
#####
# Algoritmo de hash (opcional)
#
# Los formatos permitidos son: SHA1, SHA256, SHA384, SHA512
#
# Si se omite esta propiedad, se asume SHA256
#####
afirma.hashalgorithm = SHA256
#####
# Política de firma (opcional)
#
# Indica la política de firma bajo la cual el servidor debe generar o validar
# la firma.
#
# Las políticas de firmas pueden ser identificadas mediante URI u OID
#####
#afirma.signaturePolicy = urn:oid:2.16.724.1.3.1.1.2.1.8
#####
#####
# Identificador o alias del certificado para las firmas delegadas (opcional)
#
# Por defecto, el componente presupondrá que se utiliza el certificado
# con el identificador 'default'
#####
afirma.keyname = default
#####
#####
# Timeout de conexión con @firma (en ms)
#
# Por defecto el valor es 8.000 ms
#####
```

```
afirma.connectionTimeout=8000
#####
# Timeout de respuesta con @firma (en ms)
#
# Por defecto el valor es 8.000ms
#####
afirma.receiveTimeout=8000
#####
```

Configuración Avanzada

Como alternativa al uso de un fichero .properties, se puede configurar el componente afirma-dss-client mediante un objeto de la *clase java.util.Properties*. Dicho objeto debe contener las propiedades de forma análoga a como se definen en el fichero *afirma.properties* que acabamos de describir. A modo de ejemplo, se indica a continuación cómo debe invocarse el componente con esta configuración:

```
/*
 * Definición del fichero de propiedades
 */

Properties afirmaProperties = new Properties();

afirmaProperties.put( "afirma.idapp", "IDAPP" );
afirmaProperties.put( "afirma.user", "user" );
afirmaProperties.put( "afirma.password", "pass" );

afirmaProperties.put( "afirma.host", "ws028.juntadeandalucia.es" );
afirmaProperties.put( "afirma.truststore", "trustStore.jks" );
afirmaProperties.put( "afirma.truststorePassword", "passTrustStore" );
afirmaProperties.put( "afirma.version", "6" );

// Pueden definirse el resto de propiedades incluidas en el fichero
afirma.properties

// Invocación avanzada del componente

AfirmaClient afirmaClient = new AfirmaClientImpl( new
AfirmaConfiguration().configure( afirmaProperties ));

// Llamada a la operación
```

3.5 Interfaz del Componente **afirma-dss-client**

En este apartado se van a tratar los principales aspectos de la interfaz del componente **afirma-dss-client**, con el objetivo de facilitar a los integradores la adopción de los servicios DSS disponibles en la plataforma @firma.

Como ya se ha comentado anteriormente, para poder obtener una instancia del componente, es necesario que exista en el classpath del sistema, el fichero *afirma.properties* y éste esté correctamente configurado, estableciendo al menos los parámetros obligatorios. Alternativamente, el componente también se puede haber configurado mediante un objeto de la clase *java.util.Properties*

Es importante destacar que existen métodos sobrecargados en esta interfaz que suprimen ciertos parámetros como son *signatureType*, *xmlSignatureMode* y *signatureForm*. Estos métodos sobrecargados extraen dichos parámetros del fichero *afirma.properties* con el propósito de aumentar la simplicidad en el código.

afirma-dss-client implementa aquellos servicios DSS que no están deprecados en @firma v6.4 (con excepción del servicio *DSSAfirmaSign*). Algunos de estos servicios se utilizan para diferentes operaciones dependiendo de la composición de la petición. El procedimiento que sigue **afirma-dss-client** para construir la petición y la invocación del servicio correspondiente no es objeto de conocimiento para el integrador. A pesar de ello, se reflejan a continuación a modo aclaratorio los métodos de la interfaz relacionados con los servicios de @firma que son invocados:

Servicio @firma / operación	Métodos de la interfaz
DSSAfirmaSign / sign (deprecado)	dssAfirmaSign, dssAfirmaCoSign, dssAfirmaCounterSign
DSSAfirmaVerify / verify	dssAfirmaVerify, dssAfirmaUpgrade
DSSAfirmaVerifyCertificate / verify	dssAfirmaVerifyCertificate
DSSBatchVerifySignature / verifySignatures	dssAfirmaBatchVerifySignature
DSSBatchVerifyCertificate / verifyCertificates	dssAfirmaBatchVerifyCertificate
DSSAsyncRequestStatus / getProcessResponse	dssAfirmaAsyncRequestStatus

En los siguientes apartados se describen los métodos de la interfaz del componente **afirma-dss-client**.

3.5.1 DSSAfirmaSign

Servicio de Firma Delegada que permite la realización de operaciones de firma de servidor en los formatos soportados por la plataforma. (Actualmente no se recomienda su uso ya que actualmente el servicio *DSSAfirmaSign* de @firma se encuentra deprecado)

Cabecera del método	<code>public SignResponse dssAfirmaSign(String document)</code> <code>throws AfirmaException;</code>
Parámetros de entrada	document: Cadena de texto con el contenido del documento a firmar codificado en Base64.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public SignResponse dssAfirmaSign(String document,</code> <code>String aliasCertificadoFirma) throws AfirmaException;</code>
Parámetros de entrada	document: Cadena de texto con el contenido del documento a firmar codificado en Base64. aliasCertificadoFirma: Identificador o alias del certificado que va a realizar la firma delegada.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public SignResponse dssAfirmaSign(String document, SignatureType signatureType, XmlSignatureMode xmlSignatureMode, SignatureForm signatureForm) throws AfirmaException;</code>
Parámetros de entrada	document: Cadena de texto con el contenido del documento a firmar codificado en Base64. signatureType: Tipo de firma electrónica. xmlSignatureMode: Modalidad de la firma en caso de firmas XML. signatureForm: Formato de firma avanzada.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public SignResponse dssAfirmaSign(String document, SignatureType signatureType, XmlSignatureMode xmlSignatureMode, SignatureForm signatureForm, String aliasCertificadoFirma) throws AfirmaException;</code>
Parámetros de entrada	document: Cadena de texto con el contenido del documento a firmar codificado en Base64. signatureType: Tipo de firma electrónica. xmlSignatureMode: Modalidad de la firma en caso de firmas XML. signatureForm: Formato de firma avanzada. aliasCertificadoFirma: Identificador o alias del certificado que va a realizar la firma delegada.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

3.5.2 DSSAfirmaCoSign

Servicio de Firma Delegada que permite la realización de operaciones de cofirma de servidor en los formatos soportados por la plataforma. (No se recomienda su uso ya que actualmente el servicio `DSSAfirmaSign` de @firma se encuentra deprecado)

Cabecera del método	<code>public SignResponse dssAfirmaCoSign(String documentArchiveId) throws AfirmaException;</code>
----------------------------	--

Parámetros de entrada	documentArchiveId: Valor del identificador de transacción origen.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public SignResponse dssAfirmaCoSign(String documentArchiveId, String aliasCertificadoFirma) throws AfirmaException;</code>
Parámetros de entrada	documentArchiveId: Valor del identificador de transacción origen. aliasCertificadoFirma: Identificador o alias del certificado que va a realizar la firma delegada.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

3.5.3 DssAfirmaCounterSign

Servicio de Firma Delegada que permite la realización de operaciones de contrafirma de servidor en los formatos soportados por la plataforma. (No se recomienda su uso ya que actualmente el servicio *DSSAfirmaSign* de @firma se encuentra deprecado)

Cabecera del método	<code>public SignResponse dssAfirmaCounterSign(String documentArchiveId) throws AfirmaException;</code>
Parámetros de entrada	documentArchiveId: Valor del identificador de transacción origen.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public SignResponse dssAfirmaCounterSign(String documentArchiveId, String aliasCertificadoFirma) throws AfirmaException;</code>
Parámetros de entrada	documentArchiveId: Valor del identificador de transacción origen. aliasCertificadoFirma: Identificador o alias del certificado que va a realizar la firma delegada.
Respuesta	Objeto de tipo <code>SignResponse</code> con los datos de la respuesta.

3.5.4 DSSAfirmaVerify

Servicio de validación de firmas que puede recibir como parámetros: la firma, el documento original o hash del documento (uno de los dos), el algoritmo de hash, el tipo de firma y la modalidad de firma.

Si se hace uso de un método que no incluye alguno de los parámetros mencionados, estos deben establecerse en el fichero de propiedades *afirma.properties*.

En general, todos los parámetros deben ser no nulos, a menos que se indique lo contrario explícitamente en el javadoc.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaVerify(String sign) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64.
Respuesta	Objeto de tipo <code>VerifySignatureResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaVerify(String sign, SignatureType signatureType, XmlSignatureMode signatureMode) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. signatureType: Tipo de firma electrónica. signatureMode: Modalidad de la firma en caso de firmas XML.

Respuesta	Objeto de tipo <code>VerifySignatureResponse</code> con los datos de la respuesta.
------------------	--

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaVerify(String sign, String document, String hashDocument) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. document: Datos originalmente firmados codificados en Base64. hashDocument: Hash de los datos originalmente firmados.
Respuesta	Objeto de tipo <code>VerifySignatureResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaVerify(String sign, String document, String hashDocument, String hashAlgorithm) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. document: Datos originalmente firmados codificados en Base64. hashDocument: Hash de los datos originalmente firmados. hashAlgorithm: Algoritmo de hash que se ha utilizado en el cálculo del anterior parámetro.
Respuesta	Objeto de tipo <code>VerifySignatureResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaVerify(String sign, String document, String hashDocument, SignatureType signatureType, XmlSignatureMode signatureMode) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. document: Datos originalmente firmados codificados en Base64. hashDocument: Hash de los datos originalmente firmados. signatureType: Tipo de firma electrónica. signatureMode: Modalidad de la firma en caso de firmas XML.
Respuesta	Objeto de tipo <code>VerifySignatureResponse</code> con los datos de la respuesta.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaVerifyExtended(String sign, String document, String hashDocument, String hashAlgorithm, SignatureType signatureType, XmlSignatureMode signatureMode) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. document: Datos originalmente firmados codificados en Base64. hashDocument: Hash de los datos originalmente firmados. hashAlgorithm: Algoritmo de hash con el que se ha calculado el hash indicado en el parámetro <code>hashDocument</code>. signatureType: Tipo de firma electrónica. signatureMode: Modalidad de la firma en caso de firmas XML.
Respuesta	Objeto de tipo <code>VerifySignatureResponse</code> con los datos de la respuesta.
Este método funciona de manera similar a los anteriores, pero, además, comprueba para las firmas XAdES explícitas, que el documento o hash que se le pasa como parámetro se corresponde con los datos firmados, ya que para este tipo de firmas la plataforma @firma no realiza esa comprobación.	

3.5.5 DssAfirmaUpgrade

Servicio de validación de firmas que actualiza la firma enviada como parámetro o la firma custodiada cuyo identificador se pasa como parámetro.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaUpgrade(String sign, String targetSigner) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. targetSigner: Certificado público en base64 del firmante que se desea actualizar.
Respuesta	Objeto de tipo VerifySignatureResponse con los datos de la respuesta.

Cabecera del método	<code>public VerifySignatureResponse dssAfirmaUpgrade(String sign) throws AfirmaException;</code>
Parámetros de entrada	sign: Firma electrónica codificada en Base64.
Respuesta	Objeto de tipo VerifySignatureResponse con los datos de la respuesta.

Cabecera del método	<pre>public VerifySignatureResponse dssAfirmaUpgrade(String sign, SignatureType signatureType, XmlSignatureMode signatureMode, SignatureForm signatureForm, String targetSigner) throws AfirmaException;</pre>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. signatureType: Tipo de firma electrónica. signatureMode: Modalidad de la firma en caso de firmas XML. signatureForm: Formato de firma a actualizar. targetSigner: Certificado público en base64 del firmante que se desea actualizar.
Respuesta	Objeto de tipo VerifySignatureResponse con los datos de la respuesta.

Cabecera del método	<pre>public VerifySignatureResponse dssAfirmaUpgrade(String transactionId, SignatureForm signatureForm) throws AfirmaException;</pre>
Parámetros de entrada	transactionId: Identificador de transacción de la firma electrónica custodiada. signatureForm: Formato de firma a actualizar.
Respuesta	Objeto de tipo VerifySignatureResponse con los datos de la respuesta.

Cabecera del método	<pre>public VerifySignatureResponse dssAfirmaUpgrade(String sign, SignatureType signatureType, XmlSignatureMode signatureMode, SignatureForm signatureForm) throws AfirmaException;</pre>
Parámetros de entrada	sign: Firma electrónica codificada en Base64. signatureType: Tipo de firma electrónica.

	signatureMode: Modalidad de la firma en caso de firmas XML. signatureForm: Formato de firma a actualizar.
Respuesta	Objeto de tipo VerifySignatureResponse con los datos de la respuesta.

3.5.6 DSSAfirmaVerifyCertificate

Servicio de validación de certificados que permite la verificación de certificados X509 finales.

Cabeceras del método	<pre>public VerifyCertificateResponse dssAfirmaVerifyCertificate(String certificate) throws AfirmaException;</pre>
Parámetros de entrada	certificate: Certificado a validar codificado en Base64.
Respuesta	Objeto de tipo VerifyCertificateResponse con los datos de la respuesta.

3.5.7 DssAfirmaBatchVerifySignature

Servicio de Validación por lotes (o masivo) de Firmas. Mediante este servicio se puede solicitar la verificación asíncrona de un conjunto de firmas.

Cabeceras del método	<pre>public BatchResponse dssAfirmaBatchVerifySignature(String requestId, List<VerifySignatureObject> verifySignatureObjectList, SignatureType signatureType, XmlSignatureMode xmlSignatureMode) throws AfirmaException;</pre>
Parámetros de entrada	requestId: Identificador alfanumérico que permite relacionar la petición con la respuesta asociada a la misma. Si la aplicación cliente incluye este atributo, la respuesta generada por el servidor incluirá de igual manera un atributo con el mismo valor en el elemento

	BatchResponse. verifySignatureObjectList: Lista de objetos de tipo VerifySignatureObject que encapsulan los datos de una solicitud de validación de firma. signatureType: Tipo de firma electrónica. xmlSignatureMode: Modalidad de la firma en caso de firmas XML.
Respuesta	Objeto de tipo BatchResponse con los datos de la respuesta.

3.5.8 DSSAfirmaBatchVerifyCertificate

Servicio de Validación Masiva de Certificados. Mediante este servicio se puede solicitar la verificación asíncrona de un conjunto de certificados.

Cabeceras del método	<pre>public BatchResponse dssAfirmaBatchVerifyCertificate(String requestId, List<String> certificateList) throws AfirmaException;</pre>
Parámetros de entrada	requestId: Identificador alfanumérico que permite relacionar la petición con la respuesta asociada a la misma. Si la aplicación cliente incluye este atributo, la respuesta generada por el servidor incluirá de igual manera un atributo con el mismo valor en el elemento BatchResponse. certificateList: Lista de certificados codificados en Base64.
Respuesta	Objeto de tipo BatchResponse con los datos de la respuesta.

3.5.9 DssAfirmaAsyncRequestStatus

Servicio para la consulta del estado de peticiones asíncronas. Este servicio es independientemente del proceso asíncrono a consultar (validaciones de certificados o de firmas en lote).

Cabeceras del método	<pre>public AsyncResponse dssAfirmaAsyncRequestStatus(String asyncResponseId) throws AfirmaException;</pre>
-----------------------------	---

Parámetros de entrada	asyncResponseId: Identificador del Proceso Asíncrono.
Respuesta	Objeto de tipo AsyncResponse con los datos de la respuesta.

3.5.10 HashDetector

En este caso no se trata de un servicio, sino de un método que sirve para detectar el algoritmo de hash con el que se generó el resumen de los datos de una firma.

Cabeceras del método	<code>public HashAlgorithm hashDetector(String signature) throws AfirmaException;</code>
Parámetros de entrada	signature: Firma electrónica codificada en Base64.
Respuesta	Objeto de tipo HashAlgorithm con el algoritmo utilizado en la firma.

3.5.11 SepararApellidos

Al igual que el método anterior, tampoco se trata de un servicio, sino de un método que sirve para separar una cadena formada por dos apellidos (ambos apellidos pueden ser compuestos). Como resultado devuelve cada apellido en una cadena.

Cabeceras del método	<code>public String[] separarApellidos(String apellidos) throws AfirmaException;</code>
Parámetros de entrada	apellidos: Cadena formada por dos apellidos.
Respuesta	Objeto de tipo String[] con los dos apellidos por separado.

3.6 Clases de Respuesta

La información obtenida como resultado de la llamada a los métodos de interfaz se presenta en diferentes estructuras de datos implementadas en clases de tipo “bean” que se describen a continuación:

SignResponse

Bean que encapsula la respuesta devuelta por el servicio *DSSAfirmaSign* (métodos *dssAfirmaSign*, *dssAfirmaCosign*, *dssAfirmaCounterSign*). Se compone de los siguientes métodos:

- *getResult()*: URI de acuerdo al estándar DSS que identifica el resultado de la operación (*).
- *getResultMessage()*: Mensaje de texto que describe el resultado de la operación.
- *getSignatureType()*: Obtiene el tipo de la firma generada.
- *getArchiveIdentifier()*: Devuelve el identificador de transacción de la firma generada.
- *getSignature()*: Devuelve una cadena con la firma XML enveloping.
- *getXmlSignatureForm()*: Obtiene el formato avanzado de la firma generada.
- *getAsyncResponseId()*: Obtiene el Identificador de Proceso asíncrono para obtener el resultado asociado a la petición mediante el servicio "*DSSAsyncRequestStatus*".
- *getAsyncResponseTime()*: Devuelve la fecha estimada a partir de la cual se podrá obtener la respuesta correspondiente a la petición realizada.
- *toString()*: Devuelve una cadena con los campos del objeto formateados.

VerifySignatureResponse

Bean que encapsula las respuestas devueltas por el servicio *DSSAfirmaVerify* (métodos *ddAfirmaVerify* y *dssAfirmaUpgrade*). Se compone de los siguientes métodos:

- *getResult()*: URI de acuerdo al estándar DSS que identifica el resultado de la operación (*).
- *getResultMessage()*: Mensaje de texto que describe el resultado de la operación.

- *getSignatureForm()*: Devuelve el formato avanzado de la firma validada.
- *getSignatureType()*: Obtiene el tipo de la firma validada.
- *getSignatureTimeStamp()*: Obtiene el instante de tiempo extraído del TST de la firma (en caso de que la firma tenga sello de tiempo).
- *getArchiveIdentifier()*: Obtiene el identificador de transacción asociado a la firma actualizada, en caso de upgrade.
- *getVerificationReport()*: Presenta el informe de la verificación de la firma.
- *getUpgradedSignature()*: Obtiene la firma actualizada (método *dssAfirmaUpgrade*).
- *getIndividualSignatureReportlist()*: Obtiene la lista de los informes de validación para cada una de las firmas. Cada informe incluye el certificado del firmante codificado en base64 y un listado de cada uno sus campos mapeados.
- *toString()*: Devuelve una cadena con los campos del objeto formateados.

VerifyCertificateResponse

Bean que encapsula las respuestas devueltas por el método *dssAfirmaVerifyCertificate*. Se compone de los siguientes métodos:

- *getResult()*: URI de acuerdo al estándar DSS que identifica el resultado de la operación (*).
- *getResultMessage()*: Mensaje de texto que describe el resultado de la operación.
- *getCertificateInfoList()*: Devuelve la información del certificado mapeado por @firma.
- *toString()*: Devuelve una cadena con los campos del objeto formateados.

BatchResponse

Bean que encapsula las respuestas devueltas por los métodos *dssAfirmaBatchVerifySignature* y *dssAfirmaBatchVerifyCertificate*.

- *getResult()*: URI de acuerdo al estándar DSS que identifica el resultado de la operación (*).
- *getResultMessage()*: Mensaje de texto que describe el resultado de la operación.

- *getAsyncResponseId()*: Obtiene el identificador que se ha asignado a la solicitud para posteriores consultas de estado de la misma.
- *getAsyncResponseTime()*: Devuelve la fecha y hora estimada de realización de la operación solicitada.
- *toString()*: Devuelve una cadena con los campos del objeto formateados.

AsyncResponse

Bean que encapsula la respuesta devuelta por el método *dssAfirmaAsyncRequestStatus*. Se compone de los siguientes métodos:

- *getResult()*: URI de acuerdo al estándar DSS que identifica el resultado de la operación (*).
- *getResultMessage()*: Mensaje de texto que describe el resultado de la operación.
- *getAsyncResponseTime()*: Componente que recoge la fecha estimada a partir de la cual se podrá obtener la respuesta correspondiente a la petición realizada.
- *toString()*: Devuelve una cadena con los campos del objeto formateados.

HashAlgorithm

Objeto de tipo enumerado que representa el algoritmo, posibles valores: SHA1, SHA256, SHA384, SHA512, MD2, MD5.

El javadoc del componente *afirma-dss-client* describe con mayor detalle cada una de las respuestas de las operaciones de la interfaz.

(*) Las URI's devueltas por los servicios DSS de la plataforma están descritas en el documento **@Firma-Global-XMLSOAP_PROFILE_DSS** disponible en el kit del integrador que facilita la SGAD para @firma.

Todos los objetos de respuesta del componente *afirma-dss-client* devuelven (invocando el método *getResult()*) el resultado 'ResultMajor' de la respuesta DSS original, exceptuando los objetos de la clase *VerifyCertificateResponse*, que devuelven el resultado 'ResultMinor' para indicar el estado concreto del certificado correspondiente y

el método `dssAfirmaVerifyExtended` que puede modificar la respuesta original en base al resultado de la comprobación de los datos originalmente firmados.

3.7 Ejemplos de integración

A continuación, se muestran ejemplos de integración para las operaciones de validación de certificados y actualización de firma.

3.7.1 Validación de certificado (DSSAfirmaVerifyCertificate)

El siguiente fragmento de código muestra a modo de ejemplo cómo realizar una firma delegada (firma en servidor) de un documento haciendo uso del componente `afirma-dss-client`:

```
// Se obtiene el certificado a validar

String certificadoBase64 = "MIIGBTCCA...";

// Se crea una instancia del componente AfirmaClient

AfirmaClient afirmaClient = AfirmaClientBuilder.getAfirmaClient();

// Se invoca la operación dssAfirmaVerifyCertificate con los parámetros requeridos.

VerifyCertificateResponse response =
    afirmaClient.dssAfirmaVerifyCertificate(certificado);

// Se muestra la respuesta

System.out.println(response);
```

3.7.2 Actualización de Firma (DSSAfirmaUpgrade)

El siguiente fragmento de código muestra a modo de ejemplo cómo realizar una actualización de una firma haciendo uso del componente `afirma-dss-client`:

```
// Se obtiene la firma a actualizar

String signBase64 =
    FileUtils.loadFileFromClasspathToString( "xades_enveloping.xsig" );
```

```
// Se crea una instancia del componente AfirmaClient
AfirmaClient afirmaClient = AfirmaClientBuilder.getAfirmaClient();

// Se invoca la operación dssAfirmaUpgrade incluyéndose como parámetro la firma en
// base64, el tipo de firma, el modo de firma XML
VerifySignatureResponse verifyResponse =afirmaClient.dssAfirmaUpgrade(signBase64,
AfirmaClient.SignatureType.XAdES_v132,AfirmaClient.XmlSignatureMode.ENVELOPING,
AfirmaClient.SignatureForm.T);

// Se imprime la respuesta
System.out.println(signResponse);
```

El siguiente fragmento de código muestra a modo de ejemplo cómo realizar una actualización de un solo firmante haciendo uso del componente afirma-dss-client sobre @firma6:

```
// Se obtiene la firma a actualizar
String signBase64 =
FileUtils.loadFileFromClasspathToString( "xades_enveloping.xsig" );

// Se obtiene la parte pública en Base64 del certificado del firmante que se desea
// actualizar.
String targetSignerBase64 =
FileUtils.loadFileFromClasspathToString( "signer.cer" );

// Se crea una instancia del componente AfirmaClient
AfirmaClient afirmaClient = AfirmaClientBuilder.getAfirmaClient();

// Se invoca la operación dssAfirmaUpgrade incluyéndose como parámetro la firma en
// base64, el tipo de firma, el modo de firma XML

VerifySignatureResponse verifyResponse = afirmaClient.dssAfirmaUpgrade( signBase64,
AfirmaClient.SignatureType.XAdES_v132,AfirmaClient.XmlSignatureMode.ENVELOPING,
AfirmaClient.SignatureForm.T, targetSignerBase64 );

// Se imprime la respuesta
System.out.println( signResponse );
```

4. FORMATOS BASELINE Y CORRESPONDENCIA CON FORMATOS TRADICIONALES

En la versión 6.2.2 de la plataforma @firma, se realizó una adecuación del tipo de formato de firma detectado, a la normativa de desarrollo del Reglamento (UE) nº 910/2014 del Parlamento Europeo y del Consejo de 23 de julio de 2014 relativo a la identificación electrónica y los servicios de confianza para las transacciones electrónicas en el mercado interior y por la que se deroga la Directiva 1999/93/CE (eIDAS).

Esta adecuación consistió en indicar en las respuestas de generación, actualización y validación de firmas, el formato 'Baseline' como información del formato de firma detectado, si la firma se ajusta al mismo. Esto supuso un pequeño cambio en las respuestas de @firma, ya que, a partir de entonces, la información devuelta en los campos [SignatureType] y [SignatureForm] se amplió, pudiendo venir indicados los nuevos tipos Baseline. Estos tipos son equivalentes a los formatos tradicionales siempre y cuando las firmas incluyan una serie de requisitos determinados.

El nuevo formato representa un perfil más genérico y se utiliza para garantizar la interoperabilidad transfronteriza de las firmas electrónicas en el contexto de la Directiva 2006/123/CE del Parlamento Europeo y del Consejo del 12 de diciembre de 2006 relativo a los servicios en el mercado interior. El cambio normativo viene refrendado en el Reglamento (UE) nº 910/2014 del Parlamento Europeo y del Consejo de 23 de julio de 2014 relativo a la identificación electrónica y los servicios de confianza para las transacciones electrónicas en el mercado interior, y más concretamente en la Decisión de Ejecución (UE) 2015/1506 de la Comisión de 8 de septiembre de 2015 por la que se establecen las especificaciones relativas a los formatos de las firmas electrónicas avanzadas y los sellos avanzados que deben reconocer los organismos del sector público de conformidad con los artículos 27, apartado 5, y 37, apartado 5, del Reglamento eIDAS.

En dicha Decisión de Ejecución se indica en los artículos 1 y 3 respectivamente:

1. Los Estados miembros que requieran una firma electrónica avanzada o una firma electrónica avanzada basada en un certificado cualificado como se prevé en el artículo 27, apartados 1 y 2, del Reglamento (UE) no 910/2014, reconocerán la firma electrónica avanzada XML, CMS o PDF en el nivel de conformidad B, T o LT o con un

contenedor con firma asociada donde las firmas cumplan las especificaciones técnicas que figuran en el anexo.

2. Los Estados miembros que requieran un sello electrónico avanzado o un sello electrónico avanzado basado en un certificado cualificado como se prevé en el artículo 37, apartados 1 y 2, del Reglamento (UE) no 910/2014 reconocerán el sello electrónico avanzado XML, CMS o PDF en el nivel de conformidad B, T o LT o con un contenedor con sello asociado que cumpla las especificaciones técnicas que figuran en el anexo.

Y en cuyo anexo se incluye la lista de especificaciones técnicas para las firmas electrónicas avanzadas XML, CMS o PDF y el contenedor con firma/sello asociada/o:

- Perfil de base XAdES (ETSI TS 103171 v2.1.1)
- Perfil de base CAdES (ETSI TS 103173 v2.2.1)
- Perfil de base PAdES (ETSI TS 103172 v2.2.2)
- Perfil de base del contenedor (ETSI TS 103174 v2.2.1)

Hay que tener en cuenta que no todas las firmas electrónicas a validar serán detectadas con formato del nuevo perfil base, ya que para ello tienen que cumplir con una serie de requisitos específicos que se ajusten a dicho perfil, por lo que algunas de ellas seguirán siendo detectadas con el formato tradicional. A modo informativo, se establecen las siguientes equivalencias entre los formatos tradicionales y los nuevos formatos 'Baseline':

Formato Tradicional	Formato Baseline
CAdES-BES/-EPES	CAdES B-Level
CAdES-T	CAdES T-Level
CAdES-A	CAdES LTA-Level
XAdES-BES/-EPES	XAdES B-Level
XAdES-T	XAdES T-Level
XAdES-X-L	XAdES LT-Level
XAdES-A	XAdES LTA-Level
PAdES-BES/-EPES	PAdES B-Level
PAdES-LTV	PAdES T-Level/LT-Level/LTA-Level



**Consejería de Presidencia, Interior,
Diálogo Social y Simplificación
Administrativa**

Agencia Digital de Andalucía