@Firma ::Sesión técnica @firma

Dirección General de Política Digital

Consejería de Hacienda y Administración Pública

Sevilla, 18 de Septiembre de 2013

ÍNDICE

OASIS-DSS (servicios DSS)

Índice

OASIS-DSS (servicios DSS)

¿Qué es OASIS?

Fundación, sin ánimo de lucro, para el establecimiento de estándares abiertos en la sociedad de la información.

Patrocinado por importantes empresas del sector.

OASIS-DSS (servicios DSS)

¿Qué es OASIS-DSS?

Digital Signature Services, define la interfaz para peticiones de servicios web que producen y/o verifican firmas digitales sobre unos datos dados.

Se basa en dos pares de mensajes XML petición/respuesta. A través de estos servicios un cliente puede enviar un mensaje solicitando la firma del servidor y recibiendo un XML que incluye la firma de los datos solicitados, o puede enviar una firma junto a los datos firmados y solicitar que se verifique, recibiendo una respuesta sobre si la firma es válida y corresponde con los datos enviados.

OASIS-DSS (servicios DSS)

Servicios DSS (perfil Afirma)

Servicios DSS del perfil Afirma

- **DSSAfirmaSign**. Firma y multifirma de servidor.
- **DSSAfirmaVerify**. Verificación de firma y obtención de información sobre la misma. Además permite la realización de un upgrade o actualización sobre la firma a un formato más avanzado (por ejemplo la inclusión de un sello de tiempo).
- **DSSAfirmaArchiveSubmit**. Servicio de registro de firmas.
- **DSSAfirmaArchiveRetrieval**. Servicio de obtención de firmas registradas.
- **DSSAfirmaVerifyCertificate**. Servicio de validación de certificado (a partir de @firma 5.5).
- **DSSBatchVerifyCertificate**. Servicio de validación masiva de certificados (a partir de @firma 5.5).
- **DSSBatchVerifySignature**. Servicio de validación masiva de firmas (a partir de @firma 5.5).
- **DSSAsyncRequestStatus**. Servicio de consulta del estado de peticiones asíncronas (a partir de @firma 5.5).

Limitaciones servicios DSS del perfil Afirma:

En la versión del núcleo 5.3.1, no está disponible un servicio para la validación de certificados.

- No existe servicio de firma en bloque
- No existe servicio para registrar documentos. Sólo se podrán registrar firmas.
- Los servicios nativos estarán obsoletos en la versión 5.5.

OASIS-DSS (servicios DSS)

Tipos y formatos de firma

Las diferentes estructuras de firmas compatibles por la plataforma vendrán especificadas por el tipo y el formato de la misma, ambos identificados por su URI. Los formatos compatibles son:

- **CMS**: urn:ietf:rfc:3369
- **CMS-T**: urn:afirma:dss:1.0:profile:XSS:forms:CMSWithTST
- **CAdES**: http://uri.etsi.org/01733/v1.7.3#
- **XAdES**: http://uri.etsi.org/01903/v1.3.2#
- **ODF**: urn:afirma:dss:1.0:profile:XSS:forms:ODF, firmas de Open Office, disponible en @firma 5.5
- **PDF**: urn:afirma:dss:1.0:profile:XSS:forms:PDF, firmas de PDF, disponible en @firma 5.5.

Los formatos CAdES y XAdES tienen asociado un formato determinado:

- **BES**: urn:oasis.names:tc:dss:1.0:profiles:AdES:forms:BES: formato básico.
- **T**: urn:oasis:names:tc:dss:1.0:profiles:AdES:formas:ES-T: incluye sello de tiempo.
- **EPES**: urn:oasis:names:tc:dss:1.0:profiles:AdES:formas:EPES: incluye información sobre la política utilizada, a partir de @firma 5.5.
- **C**: urn:oasis:names:tc:dss:1.0:profiles:AdES:formas:ES-C, a partir de @firma 5.5.
- **X**: urn:oasis:names:tc:dss:1.0:profiles:AdES:formas:ES-T , a partir de @firma 5.5.
- **X-L**: urn:oasis:names:tc:dss:1.0:profiles:AdES:formas:ES-X-L , a partir de @firma 5.5.
- **A**: urn:oasis:names:tc:dss:1.0:profiles:AdES:formas:ES-A, a partir de @firma 5.5.

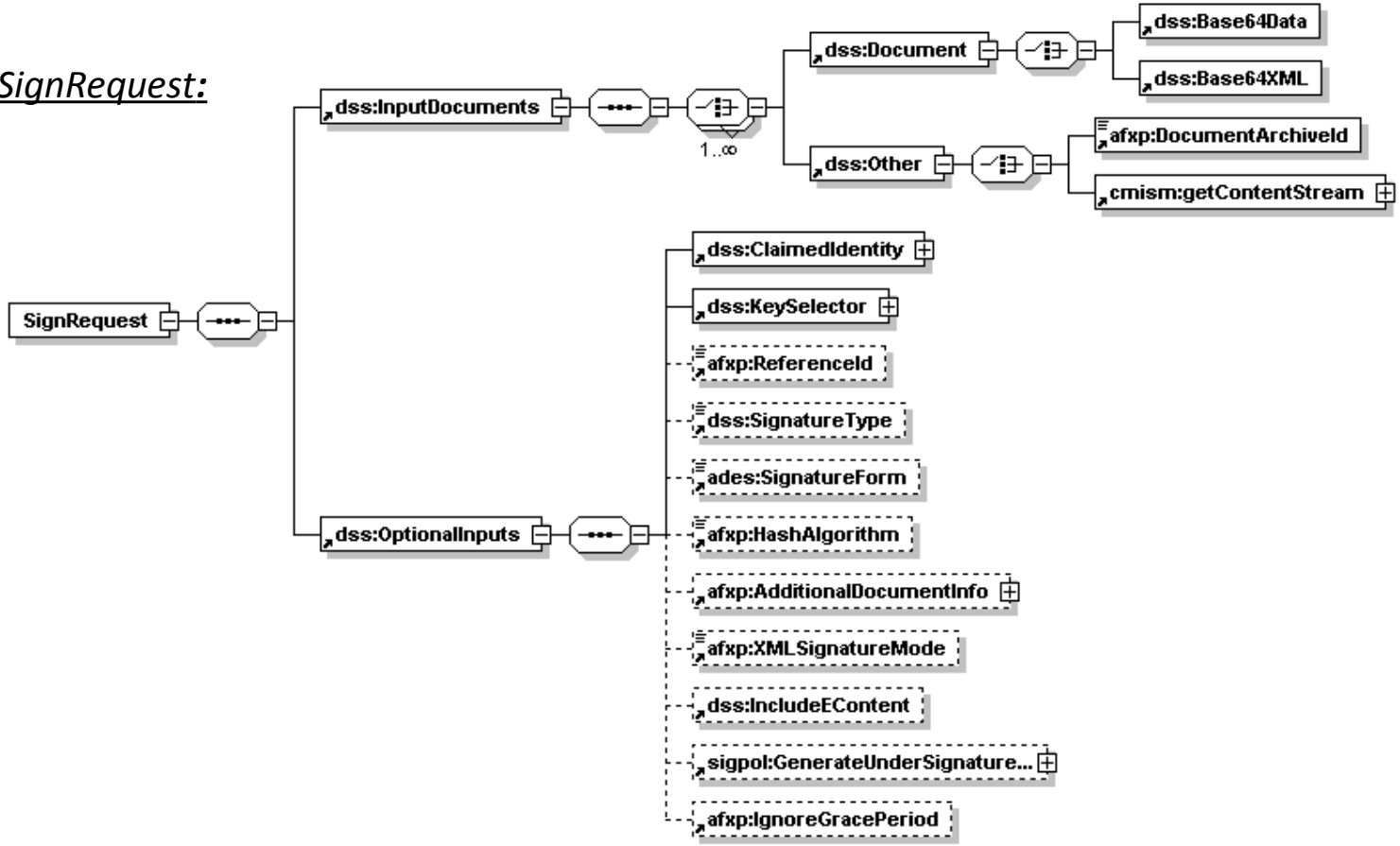
OASIS-DSS (servicios DSS)

DSSAfirmaSign - Firma de servidor

Realización de una firma con un certificado dado de alta en la plataforma sobre unos datos que previamente fueron registrados o se incluyen en la petición. Los elementos definidos en OASIS con este objetivo son:

- dss:SignRequest: Elemento XML de petición de firma.
- dss:SignResponse: Elemento XML de respuesta de firma.

dss:SignRequest:



OASIS-DSS (servicios DSS)

DSSAfirmaSign vs FirmaServidor (Petición)

Petición DSS

```
<?xml version="1.0" encoding="UTF-8"?>
<dss:SignRequest>
    <dss:InputDocuments>
        <dss:Document>
            <dss:Base64Data>
                <![CDATA[cHJ1ZWJhLnR4dA==]]>
            </dss:Base64Data>
        </dss:Document>
    </dss:InputDocuments>
    <dss:OptionalInputs>
        <dss:ClaimedIdentity>
            <dss:Name>STERIA_TEST</dss:Name>
        </dss:ClaimedIdentity>
        <dss:KeySelector>
            <ds:KeyInfo>
                <ds:KeyName>default</ds:KeyName>
            </ds:KeyInfo>
        </dss:KeySelector>
        <dss:SignatureType>
            urn:ietf:rfc:3369
        </dss:SignatureType>
        <afxp:HashAlgorithm>
            http://www.w3.org/2000/09/xmldsig#sha1
        </afxp:HashAlgorithm>
        <afxp:AdditionalDocumentInfo>
            <afxp:DocumentName>
                prueba.txt
            </afxp:DocumentName>
            <afxp:DocumentType>txt</afxp:DocumentType>
        </afxp:AdditionalDocumentInfo>
    </dss:OptionalInputs>
</dss:SignRequest>
```

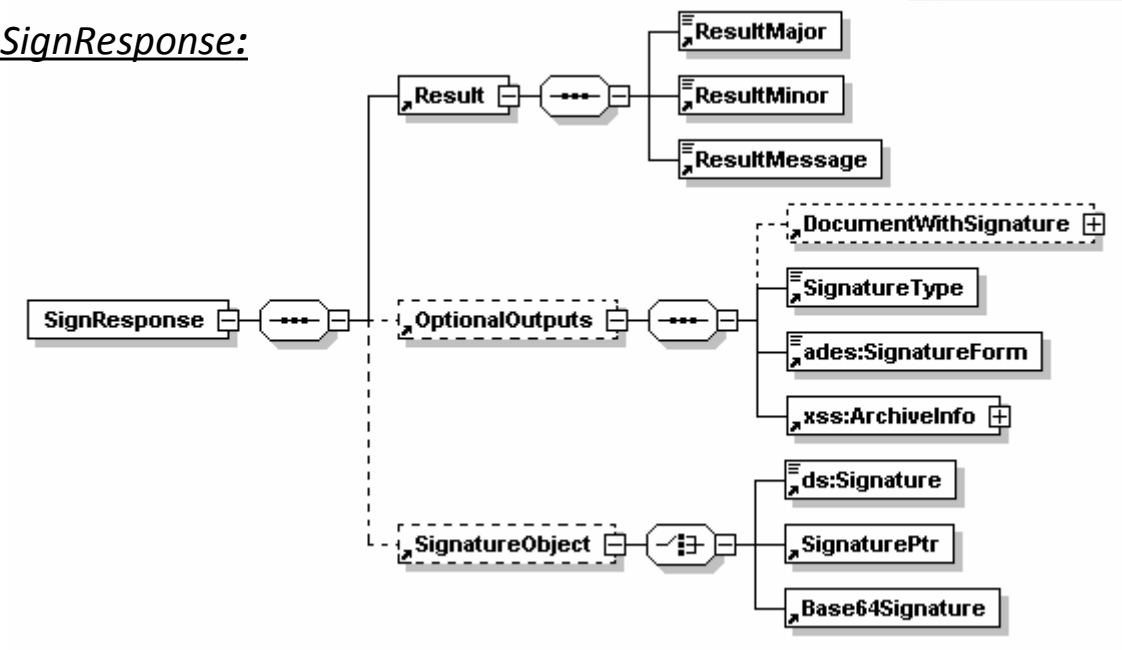
Petición nativa:

```
<?xml version="1.0" encoding="UTF-8"?>
<mensajeEntrada>
    <peticion>FirmaServidor</peticion>
    <versionMsg>1.0</versionMsg>
    <parametros>

<idAplicacion>STERIA_TEST</idAplicacion>
        <documento>
            <![CDATA[cHJ1ZWJhLnR4dA==]]>
        </documento>
        <nombreDocumento>
            prueba.txt
        </nombreDocumento>
        <tipoDocumento>txt</tipoDocumento>
        <firmante>default</firmante>
        <idReferencia />
        <algoritmoHash>SHA1</algoritmoHash>
        <formatoFirma>CMS</formatoFirma>
    </parametros>
</mensajeEntrada>
```

OASIS-DSS (servicios DSS)

DSSAfirmaSign (Respuesta)

dss:SignResponse:



OASIS-DSS (servicios DSS)

DSSAfirmaSign vs FirmaServidor (Respuesta)

Respuesta DSS:

```
<dss:SignResponse>
    <dss:Result>
        <dss:ResultMajor>
        urn:oasis:names:tc:dss:1.0:resultmajor:Success
        </dss:ResultMajor>
        <dss:ResultMessage xml:lang="es">
      Proceso de generación de firma en servidor realizado
      correctamente.
        </dss:ResultMessage>
    </dss:Result>
    <dss:OptionalOutputs>
    <dss:SignatureType>urn:ietf:rfc:3369</dss:SignatureType>
        <xss:ArchiveInfo>
            <arch:ArchiveIdentifier>
                1340100597551030
            </arch:ArchiveIdentifier>
        </xss:ArchiveInfo>
    </dss:OptionalOutputs>
    <dss:SignatureObject>
        <dss:Base64Signature Type="urn:ietf:rfc:3369">
    <![CDATA[MIIH6wYJKoZIhvcNAQcIIH3DCCB...]]>
        </dss:Base64Signature>
    </dss:SignatureObject>
</dss:SignResponse>
```

Respuesta nativa:

```
<mensajeSalida>
    <peticion>FirmaServidor</peticion>
    <versionMsg>1.0</versionMsg>
    <respuesta>
        <Respuesta>
            <estado>true</estado>
            <descripcion>
      Proceso de generación de firma en servidor
realizado correctamente.
            </descripcion>
            <idTransaccion>
                1340100597551030
            </idTransaccion>
            <firmaElectronica>
                <![CDATA[MIIH6wYJKo...]]>
            </firmaElectronica>
            <formatoFirma>CMS</formatoFirma>
        </Respuesta>
    </respuesta>
</mensajeSalida>
```

OASIS-DSS (servicios DSS)

DSSAfirmaVerify - Validación y actualización de firma

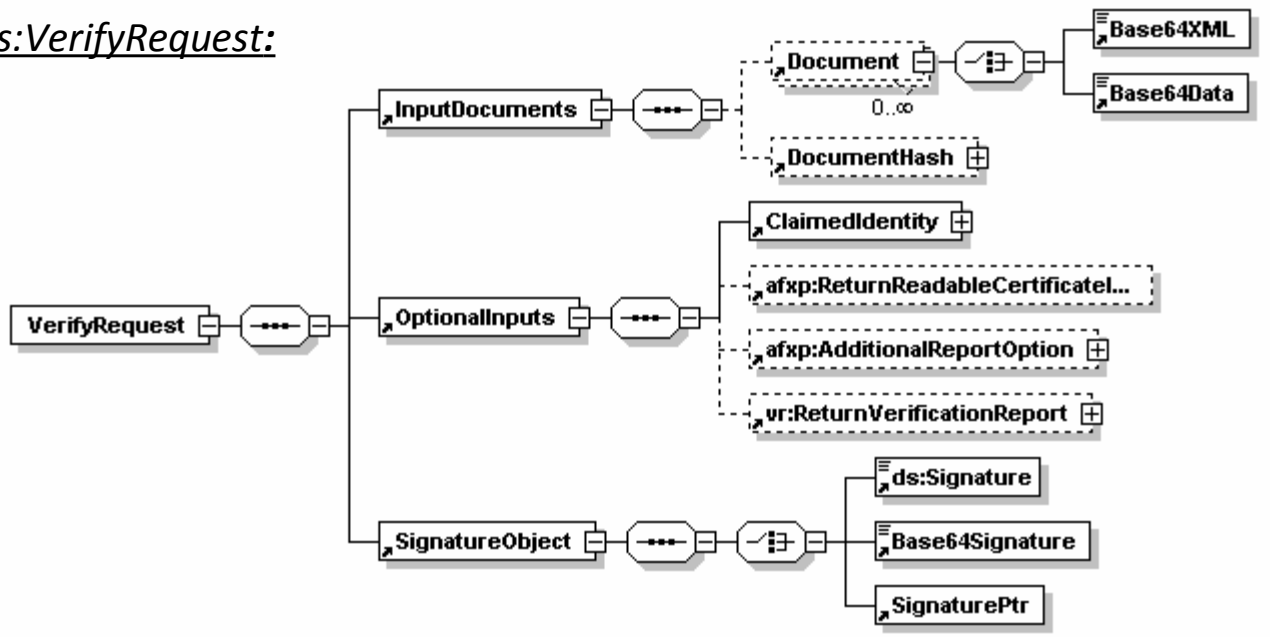
Validación de firmas electrónicas en los formatos admitidos por el sistema. Para la validación existen distintos niveles de validación (dependiendo del nivel se enviarán unos datos u otros):

- *Validación de firma electrónica.*
- *Validación de firma electrónica y el fichero firmado.*
- *Validación de firma electrónica y el hash del fichero firmado.*
- *Validación de firma electrónica, fichero firmado y su hash.*

Los elementos definidos en OASIS para este propósito son:

- dss:VerifyRequest.
- dss:VerifyResponse.

dss:VerifyRequest:



OASIS-DSS (servicios DSS)

DSSAfirmaVerify vs ValidarFirma (Petición)

Petición DSS:

```
<dss:VerifyRequest>
    <dss:InputDocuments>
        <dss:Document>
            <dss:Base64Data>
                <![CDATA[cHJ1ZWJhLnR4dA==]]>
            </dss:Base64Data>
        </dss:Document>
    </dss:InputDocuments>
    <dss:OptionalInputs>
        <dss:ClaimedIdentity>
            <dss:Name>STERIA_TEST</dss:Name>
        </dss:ClaimedIdentity>
        <vr:ReturnVerificationReport>
            <vr:ReportOptions>
                <vr:IncludeCertificateValues>
                    false
                </vr:IncludeCertificateValues>
                <vr:ReportDetailLevel>
                urn:oasis:names:tc:dss:1.0:reportdetail:noDetails
                </vr:ReportDetailLevel>
            </vr:ReportOptions>
        </vr:ReturnVerificationReport>
    </dss:OptionalInputs>
    <dss:SignatureObject>
        <dss:Base64Signature><![CDATA[MIIKhwYJKoZIhv...
            +iag3Ku74iVpXCTJBZfCEOJo3h04Ls=]]>
        </dss:Base64Signature>
    </dss:SignatureObject>
</dss:VerifyRequest>
```

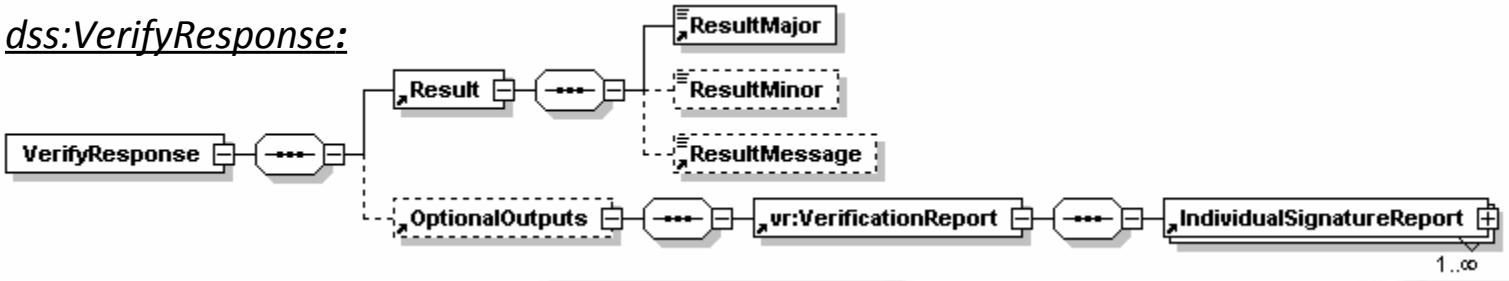
Petición nativa :

```
<mensajeEntrada>
    <peticion>ValidarFirma</peticion>
    <versionMsg>1.0</versionMsg>
    <parametros>
        <idAplicacion>STERIA_TEST</idAplicacion>
        <firmaElectronica>
            <![CDATA[MIIKhwYJKoZIhv...
            +iag3Ku74iVpXCTJBZfCEOJo3h04Ls=]]>
        </firmaElectronica>
        <formatoFirma>CADES</formatoFirma>
        <hash />
        <algoritmoHash />
        <datos>
            <![CDATA[cHJ1ZWJhLnR4dA==]]>
        </datos>
    </parametros>
</mensajeEntrada>
```

OASIS-DSS (servicios DSS)

DSSAfirmaVerify (Respuesta)

dss:VerifyResponse:



OASIS-DSS (servicios DSS)

DSSAfirmaVerify vs ValidarFirma (Respuesta)

Respuesta DSS:

```
<dss:VerifyResponse>
    <dss:Result>
        <dss:ResultMajor>
        urn:afirma:dss:1.0:profile:XSS:resultmajor:ValidSignature
        </dss:ResultMajor>
        <dss:ResultMessage xml:lang="es">La firma es
valida</dss:ResultMessage>
    </dss:Result>
    <dss:OptionalOutputs>
        <vr:VerificationReport>
            <vr:IndividualSignatureReport>
                <vr:SignatureIdentifier>
                    <vr:DigestAlgAndValue>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
                        </ds:DigestMethod>
                        <ds:DigestValue>
                        <![CDATA[udEI2RCxZmLfJQSI1Oof8LQUO4A=]]>
                        </ds:DigestValue>
                    </vr:DigestAlgAndValue>
                </vr:SignatureIdentifier>
                <dss:Result>
                    <dss:ResultMajor>
            urn:afirma:dss:1.0:profile:XSS:resultmajor:ValidSignature
                    </dss:ResultMajor>
                    <dss:ResultMessage xml:lang="es">
                        La firma es valida
                    </dss:ResultMessage>
                </dss:Result>
            </vr:IndividualSignatureReport>
        </vr:VerificationReport>
    </dss:OptionalOutputs>
</dss:VerifyResponse>
```

Respuesta nativa:

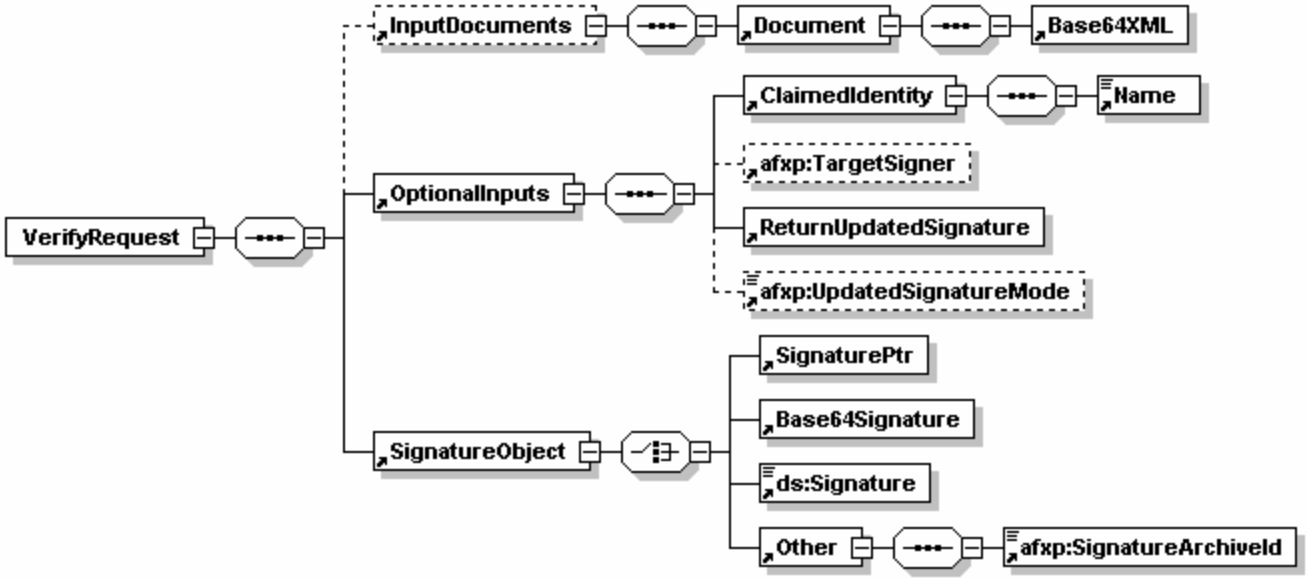
```
<mensajeSalida>
    <peticion>ValidarFirma</peticion>
    <versionMsg>1.0</versionMsg>
    <respuesta>
        <Respuesta>
            <estado>true</estado>
            <descripcion>
                <validacionFirmaElectronica>
                    <proceso>
                    Proceso de verificación de Firma Electrónica
                        completo
                    </proceso>
                    <detalle>Firma Digital correcta |
                        Firma Electrónica correcta |
                        Los certificados contenidos en la Firma
                        Electrónica sonválidos (integridad,
                        periodo de validez, estado de
                        revocación)
                    </detalle>
                    <conclusion>
                        Firma Electrónica correcta
                    </conclusion>
                </validacionFirmaElectronica>
            </descripcion>
        </Respuesta>
    </respuesta>
</mensajeSalida>
```

OASIS-DSS (servicios DSS)

DSSAfirmaVerify - Actualización de firma (Petición)

Actualización o upgrade de firmas electrónicas a un formato más avanzado (por ejemplo para la inclusión del sello de tiempo). Los pares de mensajes son los mismos que para la verificación, pero la inclusión de un elementos como dss:ReturnUpdatedSignature indicará al sistema que lo que se desea es una actualización de firma.

dss:VerifyRequest:



OASIS-DSS (servicios DSS)

DSSAfirmaVerify - Actualización de firma (Petición)

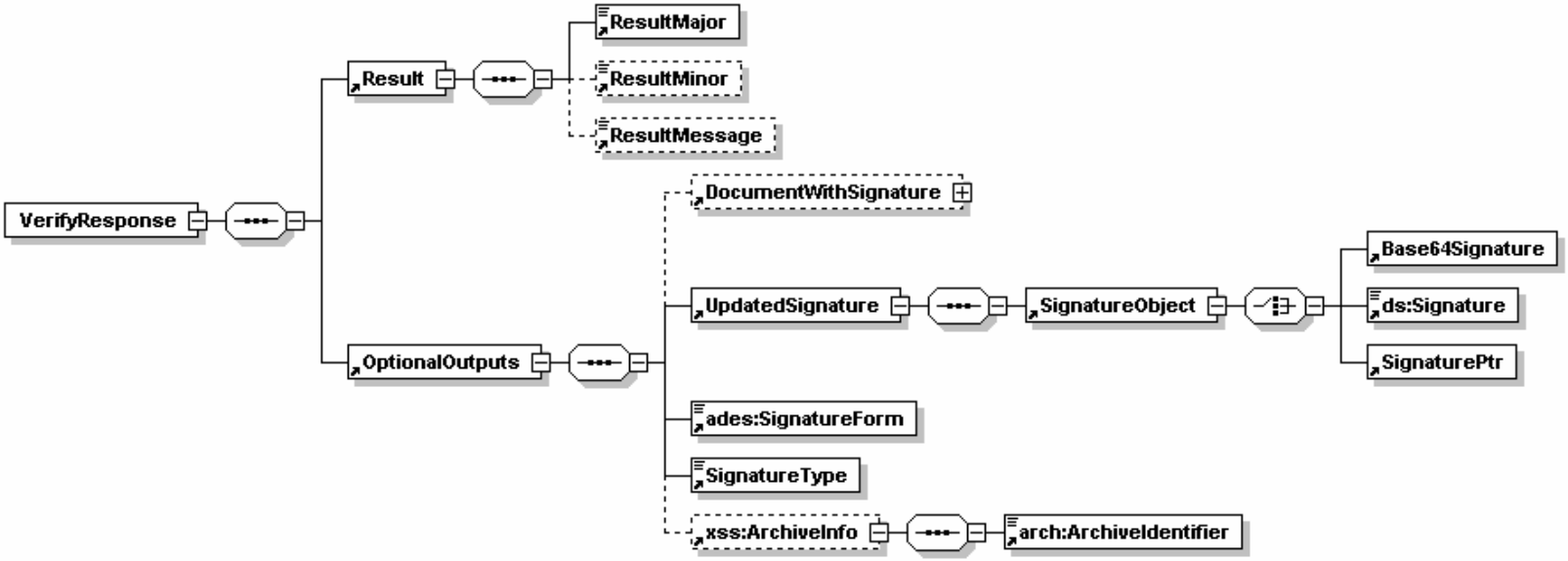
Petición dss:

```
<dss:VerifyRequest Profile="urn:afirma:dss:1.0:profile:XSS">
    <dss:OptionalInputs>
        <dss:ClaimedIdentity>
            <dss:Name>STERIA_TEST</dss:Name>
        </dss:ClaimedIdentity>
        <dss:ReturnUpdatedSignature
            Type="urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-T" />
    </dss:OptionalInputs>
    <dss:SignatureObject>
        <dss:Base64Signature><![CDATA[MIIKhwYJKo...
        EmTmkm655ywUUOIMts+3QXLnj3rbwGK
        O+Q8tgWgghh+iag3Ku74iVpXCTJBZfCEOJo3h04Ls=]]>
        </dss:Base64Signature>
    </dss:SignatureObject>
</dss:VerifyRequest>
```

OASIS-DSS (servicios DSS)

DSSAfirmaVerify - Actualización de firma (Respuesta)

dss:VerifyResponse:



OASIS-DSS (servicios DSS)

DSSAfirmaVerify - Actualización de firma (Respuesta)

Respuesta dss:

```
<dss:VerifyResponse Profile="urn:afirma:dss:1.0:profile:XSS">
    <dss:Result>
        <dss:ResultMajor>
            urn:oasis:names:tc:dss:1.0:resultmajor:Success
        </dss:ResultMajor>
        <dss:ResultMessage xml:lang="es">
        Proceso de actualización de firma realizado correctamente.
        </dss:ResultMessage>
    </dss:Result>
    <dss:OptionalOutputs>
        <dss:UpdatedSignature>
            <dss:SignatureObject>
                <dss:Base64Signature
                    Type="http://uri.etsi.org/01733/v1.7.3#">
                    <![CDATA[MIIQYJKoZIhvcNAQcCoIIQBjCCE...
                    OrnCJoI9wWvntSFerkolRO5sP4c9+xB0k4=]]>
                </dss:Base64Signature>
            </dss:SignatureObject>
        </dss:UpdatedSignature>
        <xss:ArchiveInfo>
            <arch:ArchiveIdentifier>
                1340269416581074
            </arch:ArchiveIdentifier>
        </xss:ArchiveInfo>
        <dss:SignatureType>
            http://uri.etsi.org/01733/v1.7.3#
        </dss:SignatureType>
        <ades:SignatureForm>
            urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-T
        </ades:SignatureForm>
    </dss:OptionalOutputs>
</dss:VerifyResponse>
```

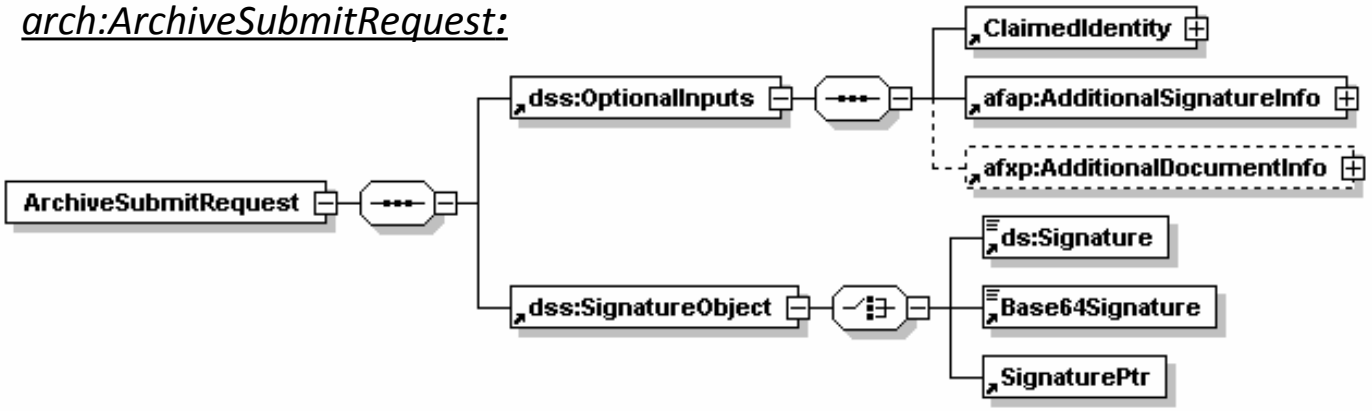
OASIS-DSS (servicios DSS)

DSSAfirmaArchiveSubmit - Registro de firma

Permite el registro o custodia en el sistema, de firmas realizadas externamente. El perfil de OASIS define los siguientes mensajes:

- arch:ArchiveSubmitRequest.
- arch:ArchiveSubmitResponse.

arch:ArchiveSubmitRequest:



OASIS-DSS (servicios DSS)

DSSAfirmaArchiveSubmit vs FirmaUsuario2FasesF2 (Petición)

Petición dss:

```
<arch:ArchiveSubmitRequest>
    <dss:OptionalInputs>
        <dss:ClaimedIdentity>
            <dss:Name>STERIA_TEST</dss:Name>
        </dss:ClaimedIdentity>
        <afap:AdditionalSignatureInfo>
            <dss:InputDocuments>
                <dss:Document>
                    <dss:Base64Data>
                        <![CDATA[cHJ1ZWJhLnR4dA==]]>
                    </dss:Base64Data>
                </dss:Document>
            </dss:InputDocuments>
            <ds:X509Data>
                <ds:X509Certificate><![CDATA[MIIFTDC...
                    yXeqlVRsWQp5e/anZHTWnaMnEb+7XQ==]]>
                </ds:X509Certificate>
            </ds:X509Data>
            <afxp:HashAlgorithm>
                http://www.w3.org/2000/09/xmldsig#sha1
            </afxp:HashAlgorithm>
            <afap:StoreDocument>false</afap:StoreDocument>
        </afap:AdditionalSignatureInfo>
        <afxp:AdditionalDocumentInfo>
            <afxp:DocumentName>prueba.txt</afxp:DocumentName>
            <afxp:DocumentType>txt</afxp:DocumentType>
        </afxp:AdditionalDocumentInfo>
    </dss:OptionalInputs>
    <dss:SignatureObject>
        <dss:Base64Signature><![CDATA[MIIG4AYJKoZ...
        Go2wdBtb8bgLMt5IdZZRGu4hGqjhCYmsKzy3+zSNafFN1Uq]]>
        </dss:Base64Signature>
    </dss:SignatureObject>
</arch:ArchiveSubmitRequest>
```

Petición nativa:

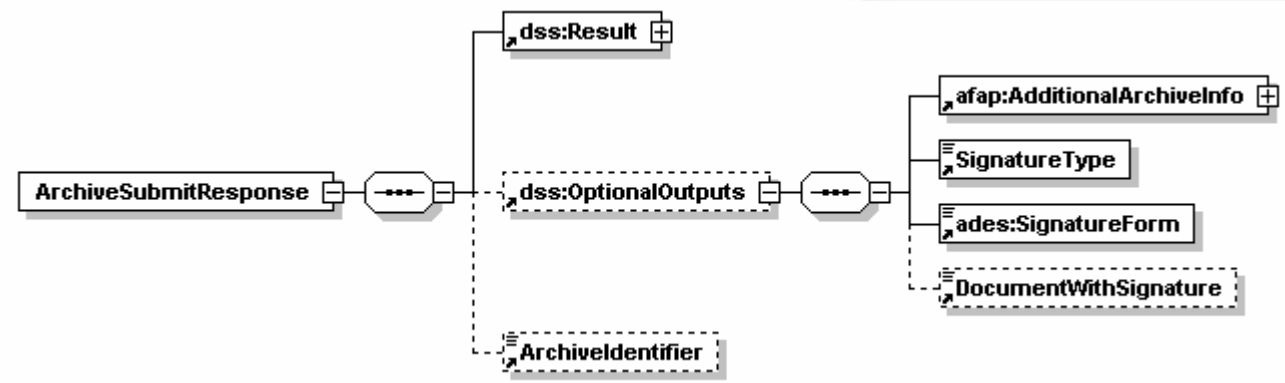
```
<?xml version="1.0" encoding="UTF-8"?>
<mensajeEntrada>
    <peticion>FirmaUsuario2FasesF2</peticion>
    <versionMsg>1.0</versionMsg>
    <parametros>
        <idAplicacion>STERIA_TEST</idAplicacion>
        <firmaElectronica>
            <![CDATA[MIIG4AYJKoZ...
Go2wdBtb8bgLMt5IdZZRGu4hGqjhCYmsKzy3+zSNafFN1Uq]]>
        </firmaElectronica>
        <certificadoFirmante>
            <![CDATA[MIIFTDC...
            yXeqlVRsWQp5e/anZHTWnaMnEb+7XQ==]]>
            </certificadoFirmante>
        <idReferencia />
        <formatoFirma><![CDATA[CMS]]></formatoFirma>
        <documento>
            <![CDATA[cHJ1ZWJhLnR4dA==]]>
        </documento>
        <nombreDocumento>prueba.txt</nombreDocumento>
        <tipoDocumento>txt</tipoDocumento>
        <algoritmoHash>
            SHA1
        </algoritmoHash>
        <custodiarDocumento>false</custodiarDocumento>
    </parametros>
</mensajeEntrada>
```

Formato de firma auto detectado.

OASIS-DSS (servicios DSS)

DSSAfirmaArchiveSubmit - Registro de firma (Respuesta)

arch:ArchiveSubmitResponse:



OASIS-DSS (servicios DSS)

DSSAfirmaArchiveSubmit vs FirmaUsuario2FasesF2 (Respuesta)

Respuesta dss:

```
<?xml version="1.0" encoding="UTF-8"?>
<arch:ArchiveSubmitResponse>
    <dss:Result>
        <dss:ResultMajor>
            urn:oasis:names:tc:dss:1.0:resultmajor:Success
        </dss:ResultMajor>
        <dss:ResultMessage xml:lang="es">
            Proceso de fase 2 de firma de usuario en 2 fases
            realizado correctamente.  Justificante [CADES-T] –
            Firma Usuario [CMS]
        </dss:ResultMessage>
    </dss:Result>
    <dss:OptionalOutputs>
        <afap:AdditionalArchiveInfo>
            <afap:EvidenceOfESignature>
                <dss:SignatureObject>
                    <dss:Base64Signature>
                        <![CDATA[MIIQ4AYJKoZ…
                        Ym2rvP+yukJ6IcnPQ1PM381w==]]>
                    </dss:Base64Signature>
                </dss:SignatureObject>
                <dss:SignatureType>
                    http://uri.etsi.org/01733/v1.7.3#
                </dss:SignatureType>
                <ades:SignatureForm>
                    urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-T
                </ades:SignatureForm>
            </afap:EvidenceOfESignature>
        </afap:AdditionalArchiveInfo>
        <dss:SignatureType>
            urn:ietf:rfc:3369
        </dss:SignatureType>
    </dss:OptionalOutputs>
    <arch:ArchiveIdentifier>1340345037383004</arch:ArchiveIdentifier>
</arch:ArchiveSubmitResponse>
```

Respuesta nativa:

```
<?xml version="1.0"?>
<mensajeSalida xmlns="https://afirmaws/ws/firma"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://afirmaws/ws/firma
https://10.90.29.30/afirmaws/xsd/mfirma/ws.xsd ">
    <peticion>FirmaUsuario2FasesF2</peticion>
    <versionMsg>1.0</versionMsg>
    <respuesta>
        <Respuesta>
            <estado>true</estado>
            <descripcion>Proceso de fase 2 de firma de usuario en 2 fases
                realizado correctamente. Justificante [CADES-T] –
                Firma Usuario  [CMS-T]
            </descripcion>
            <idTransaccion>1340346930982041</idTransaccion>
            <justificanteFirmaElectronica><![CDATA[MIIQ4AYJKoZ…
                5zItPk3Xm9PriBYIKSceOPUOiGfhRoLXuTb+42SBpg==]]>
            </justificanteFirmaElectronica>
        </Respuesta>
    </respuesta>
</mensajeSalida>
```

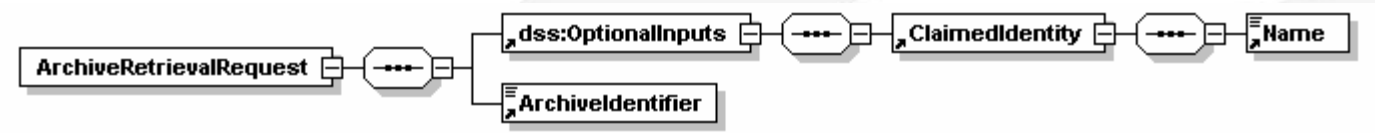
OASIS-DSS (servicios DSS)

DSSAfirmaArchiveRetrieval - Obtención de firma

El cliente incluye en la petición el identificador de transacción de la firma a recuperar y obtiene una respuesta con el resultado del proceso y la firma asociada. OASIS define la siguiente pareja de mensajes:

- arch:ArchiveRetrievalRequest.
- arch:ArchiveRetrievalResponse.

arch:ArchiveSubmitRequest:



OASIS-DSS (servicios DSS)

DSSAfirmaArchiveRetrieval vs ObtenerFirmaTransaccion (Petición)

Petición DSS:

```
<?xml version="1.0" encoding="UTF-8"?>
<arch:ArchiveRetrievalRequest
                Profile="urn:afirma:dss:1.0:profile:archive">
                <dss:OptionalInputs>
                        <dss:ClaimedIdentity>
                                <dss:Name>STERIA_TEST</dss:Name>
                        </dss:ClaimedIdentity>
                </dss:OptionalInputs>
                <arch:ArchiveIdentifier>
                        1340346930982017
                </arch:ArchiveIdentifier>
</arch:ArchiveRetrievalRequest>
```

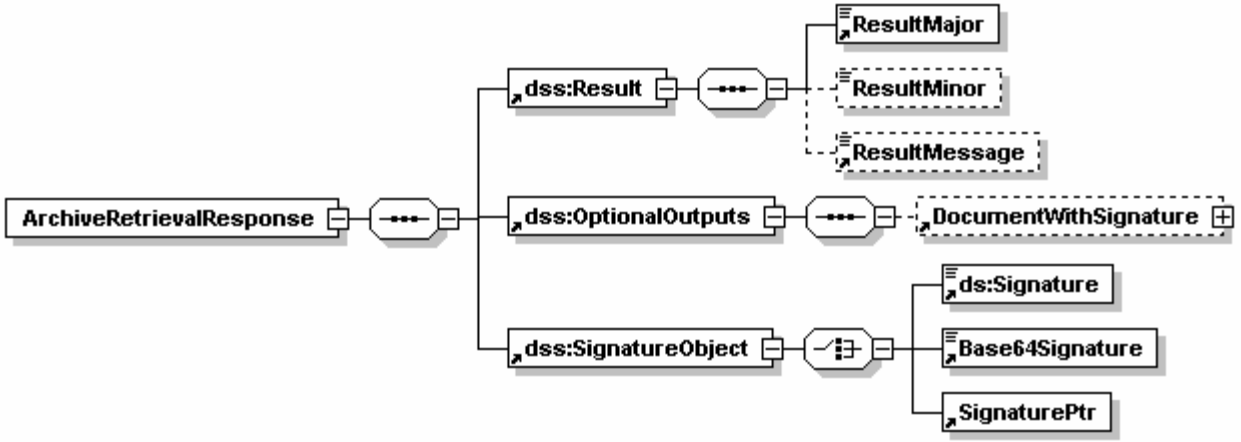
Petición nativa:

```
<?xml version="1.0" encoding="UTF-8"?>
<mensajeEntrada>
      <peticion>ObtenerFirmaTransaccion</peticion>
      <versionMsg>1.0</versionMsg>
      <parametros>
            <idAplicacion>STERIA_TEST</idAplicacion>
            <idTransaccion>
                  1340346930982017
            </idTransaccion>
      </parametros>
</mensajeEntrada>
```

OASIS-DSS (servicios DSS)

DSSAfirmaArchiveRetrieval (Respuesta)

arch:ArchiveSubmitResponse:



OASIS-DSS (servicios DSS)

DSSAfirmaArchiveRetrieval vs ObtenerFirmaTransaccion (Respuesta)

Respuesta DSS:

```
<?xml version="1.0" encoding="UTF-8"?>
<arch:ArchiveRetrievalResponse
      Profile="urn:afirma:dss:1.0:profile:archive">
      <dss:Result>
            <dss:ResultMajor>
                  urn:oasis:names:tc:dss:1.0:resultmajor:Success
            </dss:ResultMajor>
            <dss:ResultMessage xml:lang="es">
                  Proceso de obtención de la FirmaElectrónica
                  realizado correctamente.
            </dss:ResultMessage>
      </dss:Result>
      <dss:SignatureObject>
            <dss:Base64Signature>
                  <![CDATA[MIIMbgYJKoZIhvcNAQcCoI...
                  6cXTZ/SHlGwJy1f0A4mJGCfl3Im1EQMVX2O]]>
            </dss:Base64Signature>
      </dss:SignatureObject>
</arch:ArchiveRetrievalResponse>
```

Respuesta nativa:

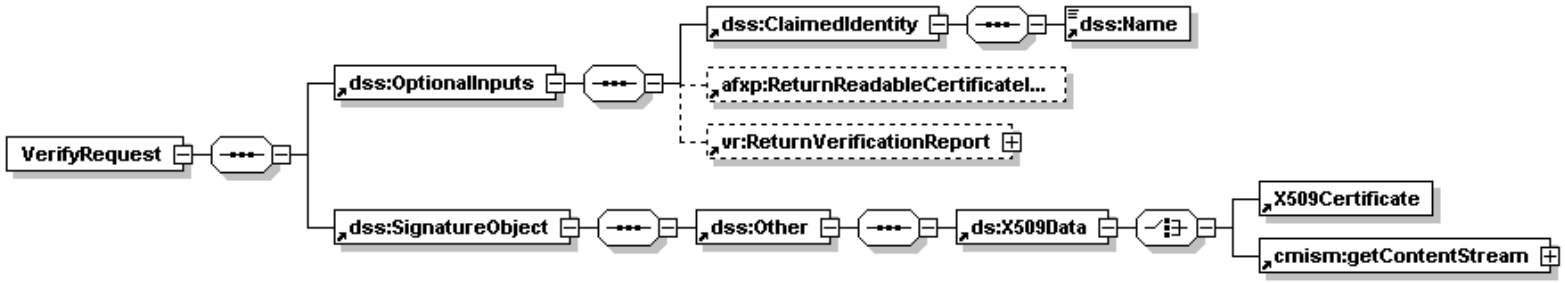
```
<?xml version="1.0"?>
<mensajeSalida>
      <peticion>ObtenerFirmaTransaccion</peticion>
      <versionMsg>1.0</versionMsg>
      <respuesta>
            <Respuesta>
                  <estado>true</estado>
                  <descripcion>
                        Proceso de obtención de la FirmaElectrónica
                        realizado correctamente.
                  </descripcion>
                  <firmaElectronica>
                        <![CDATA[MIIMbgYJKoZIhvcNAQcCoI...
                  6cXTZ/SHlGwJy1f0A4mJGCfl3Im1EQMVX2O]]>
                  </firmaElectronica>
                  <formatoFirma>CMS-T</formatoFirma>
            </Respuesta>
      </respuesta>
</mensajeSalida>
```

OASIS-DSS (servicios DSS)

DSSAfirmaVerifyCertificate - Validación de certificado

Permite la verificación de certificados X509 finales. Para el diseño de este servicio se ha partido del las especificaciones [DSS XSS] que permiten la verificación de certificados en su protocolo de verificación.
La petición de validación de certificado similar a la de validación de firma.

dss:VerifyRequest:



OASIS-DSS (servicios DSS)

DSSAfirmaVerifyCertificate vs ValidarCertificado (Petición)

Petición dss:

```
<?xml version="1.0" encoding="UTF-8"?>
<dss:VerifyRequest>
    <dss:OptionalInputs>
        <dss:ClaimedIdentity>
            <dss:Name>STERIA_TEST</dss:Name>
        </dss:ClaimedIdentity>
        <afxp:ReturnReadableCertificateInfo />
        <vr:ReturnVerificationReport>
            <vr:CheckOptions>
                <vr:CheckCertificateStatus>
                    true
                </vr:CheckCertificateStatus>
            </vr:CheckOptions>
            <vr:ReportOptions>
        <vr:IncludeCertificateValues>true</vr:IncludeCertificateValues>
        <vr:IncludeRevocationValues>true</vr:IncludeRevocationValues>
                <vr:ReportDetailLevel>
                    urn:oasis:names:tc:dss:1.0:reportdetail:allDetails
                </vr:ReportDetailLevel>
            </vr:ReportOptions>
        </vr:ReturnVerificationReport>
    </dss:OptionalInputs>
    <dss:SignatureObject>
        <dss:Other>
            <ds:X509Data>
                <ds:X509Certificate>
                    <![CDATA[MIIDP….]]>
                </ds:X509Certificate>
            </ds:X509Data>
        </dss:Other>
    </dss:SignatureObject>
</dss:VerifyRequest>
```

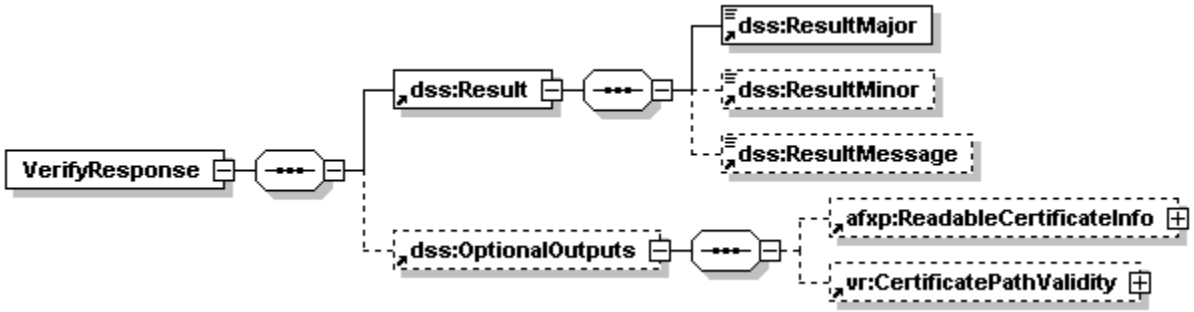
Petición nativa:

```
<?xml version="1.0" encoding=" UTF-8"?>
<mensajeEntrada>
    <peticion>ValidarCertificado</peticion>
    <versionMsg>1.0</versionMsg>
    <parametros>
        <certificado>
            <![CDATA[MIIDP….]]>
        </certificado>
        <idAplicacion>STERIA_TEST</idAplicacion>
        <modoValidacion>1</modoValidacion>
        <obtenerInfo>true</obtenerInfo>
    </parametros>
</mensajeEntrada>
```

OASIS-DSS (servicios DSS)

DSSAfirmaVerifyCertificate (Respuesta)

dss:VerifyResponse:



OASIS-DSS (servicios DSS)

DSSAfirmaVerifyCertificate vs ValidarCertificado (Respuesta)

Respuesta dss:

```
<?xml version="1.0" encoding="UTF-8"?>
<dss:VerifyResponse>
    <dss:Result>
        <dss:ResultMajor>
urn:oasis:names:tc:dss:1.0:resultmajor:Success
        </dss:ResultMajor>
        <dss:ResultMinor>
urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:valid:certificate:Definitive
        </dss:ResultMinor>
        <dss:ResultMessage xml:lang="es">El certificado es válido
        </dss:ResultMessage>
    </dss:Result>
    <dss:OptionalOutputs>
        <afxp:ReadableCertificateInfo>
            ...
        </afxp:ReadableCertificateInfo>
        <vr:CertificatePathValidity>
            ...
        </vr:CertificatePathValidity>
    </dss:OptionalOutputs>
</dss:VerifyResponse>
```

```
<mensajeSalida>
    <peticion>ValidarCertificado</peticion>
    <versionMsg>1.0</versionMsg>
    <respuesta>
        <ResultadoProcesamiento>
            <InfoCertificado>
                <Campo>
                    <idCampo>usoCertificado</idCampo>
                    <valorCampo>
                        digitalSignature | nonRepudiation
                    </valorCampo>
                </Campo>
                ...
            </InfoCertificado>
            <ResultadoValidacion>
                <resultado>0</resultado>
                <descripcion>
                    Validacion Satisfactoria
                </descripcion>
                <ValidacionSimple>
                    <codigoResultado>0</codigoResultado>
                    <descResultado>
                        Validacion Satisfactoria
                    </descResultado>
                </ValidacionSimple>
            </ResultadoValidacion>
        </ResultadoProcesamiento>
    </respuesta>
</mensajeSalida>
```

OASIS-DSS (servicios DSS)

Validación de firmas y certificados masivas (asíncronas)

Mediante estos servicios se puede solicitar la validación de múltiples firmas o certificados con una sola petición.

Estas peticiones serán procesadas de manera asíncrona por el servidor, el cual generará una respuesta del tipo «pendiente de procesado».

- DSSBatchVerifyCertificate
- DSSBatchVerifySignature

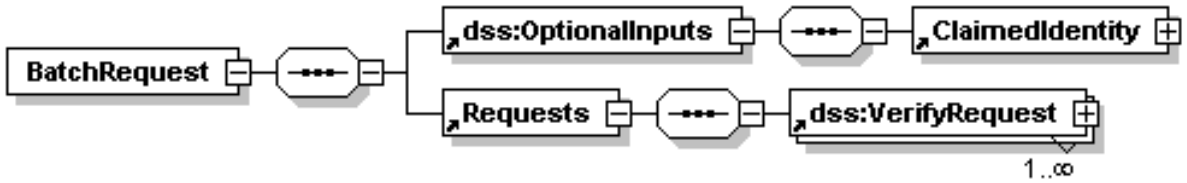
Posteriormente se podrá solicitar al servidor, utilizando el identificador de petición (recibido en la respuesta de la solicitud de verificación) junto al identificador de aplicación, el estado de la petición mediante los mensajes:

- async:PendingRequest
- afxp:BatchResponse

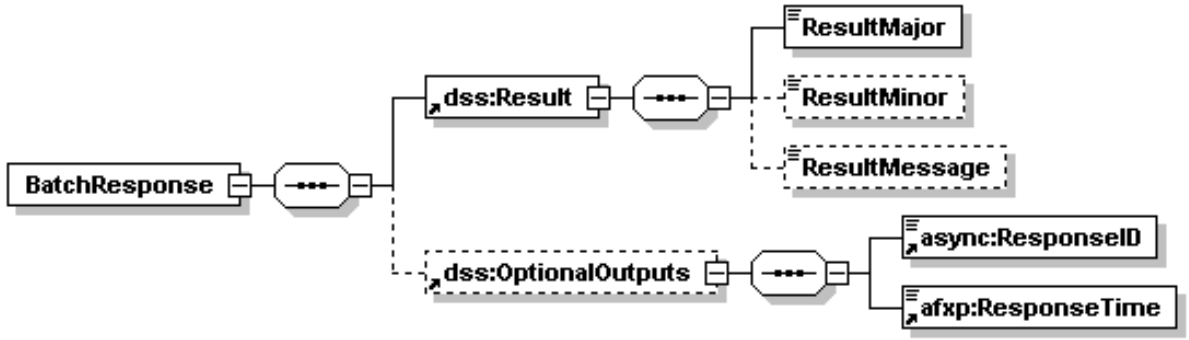
OASIS-DSS (servicios DSS)

Validaciones masivas (Petición)

BatchVerify:



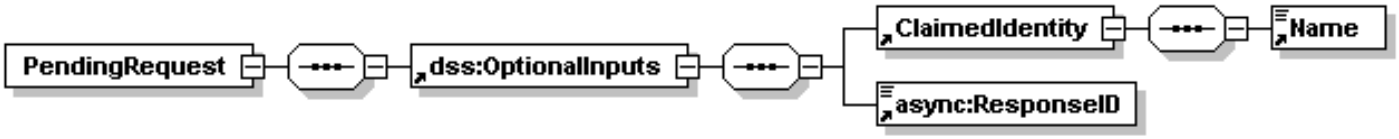
BatchResponse:



OASIS-DSS (servicios DSS)

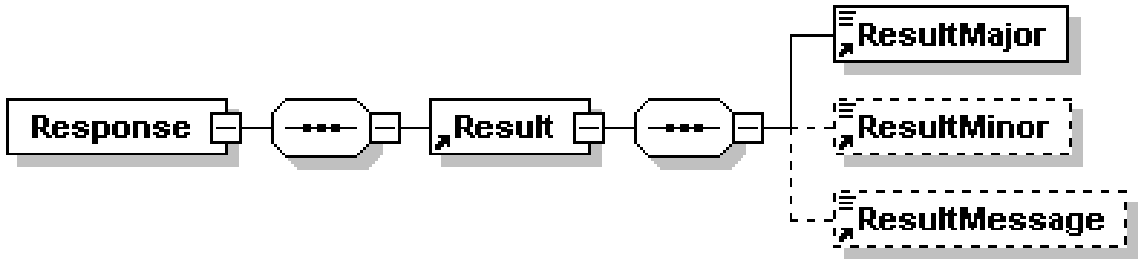
Validaciones masivas (consulta de estado y respuesta)

async:PendingRequest:



Respuesta:

- Respuesta de petición válida. Si la petición es válida y el proceso asíncrono consultado existe se devolverá una respuesta de proceso finalizado o pendiente de ejecutar acorde al servicio inicialmente invocado.
- Respuesta de petición no válida. Si la petición no es formalmente válida, no está autorizada, el identificador de procesos asíncrono no es válido o se produce otro tipo de error se devuelve al cliente una respuesta genérica como la representada en la figura.



ÍNDICE

Introducción

¿Qué es el kit de desarrollo de @firma?

- El kit de desarrollo de @firma es un **conjunto de librerías java** que permite a los integradores interactuar fácilmente con los servicios DSS, de autenticación y custodia de @firma 5.3.1 y 5.5.
- Estas librerías sustituyen a la librería utilizada hasta ahora: afirma5ServiceInvoker.jar.
- El kit de desarrollo está compuesto por tres módulos:
    - Cliente DSS (afirma-dss-client)
    - Cliente de autenticación (afirma-authentication-client)
    - Cliente de custodia (afirma-custody-client)
- Cada uno de los módulo constituye una librería java y pueden utilizarse de manera independiente.

Introducción

Requisitos mínimos

El kit de desarrollo de @firma requiere lo siguiente para su ejecución:

•Tener instalada la **JDK 1.5. o superior.**

•**Incluir las dependencias** (ficheros jar) que se facilitan con el paquete entregable en el classpath de la aplicación cliente.

•**Visibilidad de los servicios** de @firma requeridos.

•Fichero de propiedades ***afirma.properties*** correctamente configurado e incluido en el classpath.

Introducción

Limitaciones

El kit de desarrollo de @firma presenta las siguientes limitaciones:

•Cliente DSS:

- @firma 5.3.1 no implementa el servicio DSSVerifyCertificate (para este caso se hace uso del servicio nativo) ni los servicios asíncronos.
- Los servicios asíncronos de @firma 5.5. presentan deficiencias en su funcionamiento.
- El servicio DSSAfirmaVerify no devuelve ordenados los certificados de los firmantes en las multifirmas.

•Cliente de custodia:

- No permite custodiar documentos, sólo se acepta la consulta de los mismos.

Cliente DSS (afirma-dss-client)

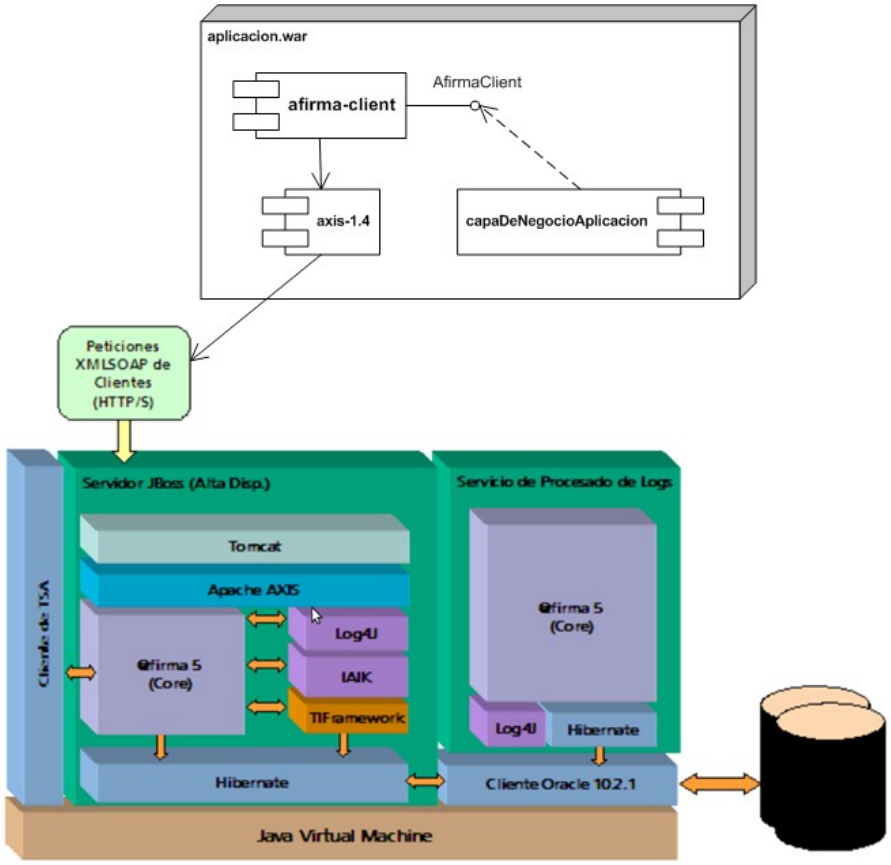
Índice

- Introducción
- Funcionalidad y servicios
- Integración
  - Interfaz
  - Paso a paso
  - Dependencias
  - Configuración
  - Ejemplos

Cliente DSS (afirma-dss-client)

Introducción

- El cliente DSS (afirma-dss-client) es un componente software que tiene como objetivo fundamental facilitar a los integradores la interacción con los servicios DSS disponibles en la plataforma @firma. Se trata de un componente 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.



Cliente DSS (afirma-dss-client)

Funcionalidad y servicios (I)

¿Qué hace?

1.Implementa la lógica de mensajería:

- Realiza la composición de la petición XML
- Procesamiento de la respuesta XML

2.Implementa el envío y recepción del mensaje SOAP:

- Transparente para integrador
- Utiliza librerías de código abierto: Axis, WS Security…

Cliente DSS (afirma-dss-client)

Funcionalidad y servicios (II)

¿Qué servicios DSS se utilizan?

A partir de @firma 5.3.1:

- **DSSAfirmaSign**: Firma de servidor.
- **DSSAfirmaVerify**: Validación de firmas.
- **DSSAfirmaArchiveSubmit**: Custodia de firmas.
- **DSSAfirmaArchiveRetrieval**: Obtención de firmas custodiadas.

A partir de @firma 5.5:

- **DSSAfirmaVerifyCertificate**: Validación de certificados.
- **DSSBatchVerifyCertificate**: Validación masiva de certificados.
- **DSSBatchVerifySignature**: Validación masiva de firmas.
- **DSSAsyncRequestStatus**: Consulta de estado de peticiones asíncronas.

Cliente DSS (afirma-dss-client)

Interfaz del componente (I)

All Classes

Packages

es.juntadeandalucia.afirma.client
es.juntadeandalucia.afirma.client.be
es.juntadeandalucia.afirma.client.be
es.juntadeandalucia.afirma.client.be
es.juntadeandalucia.afirma.client.be
es.juntadeandalucia.afirma.client.be

All Classes

AbstractRequest
AdditionalDocumentInfo
AdditionalReportOption
AdditionalSignatureInfo
AfirmaArchiveProfileSchemaNS
AfirmaClient
AfirmaClient.SignatureForm
AfirmaClient.SignatureType
AfirmaClient.XmlSignatureMode
AfirmaClientBuilder
AfirmaClientImpl
AfirmaConfiguration
AfirmaException
AfirmaXSSProfileSchemaNS
ArchiveIdentifier
ArchiveRetrievalRequest
ArchiveRetrievalRequestFactory
ArchiveRetrievalResponse
ArchiveSubmitRequest
ArchiveSubmitRequestFactory
ArchiveSubmitResponse
AsyncRequestFactory
AsyncResponse
Base64Data
Base64Signature
BatchRequest

Method Summary

ArchiveRetrievalResponse	**dssAfirmaArchiveRetrieval**(String transactionId) Servicio para la obtención de firmas electrónicas custodiadas en la plataforma.
ArchiveSubmitResponse	**dssAfirmaArchiveSubmit**(String sign, String certificate) Servicio para el registro o custodia de firmas electrónicas ya generadas.
ArchiveSubmitResponse	**dssAfirmaArchiveSubmit**(String sign, String certificate, AfirmaClient.SignatureType signatureTy AfirmaClient.XmlSignatureMode xmlSignatureMode) Servicio para el registro o custodia de firmas electrónicas ya generadas.
AsyncResponse	**dssAfirmaAsyncRequestStatus**(String asyncResponseId) Servicio para la consulta del estado de peticiones asíncronas.
BatchResponse	**dssAfirmaBatchVerifyCertificate**(String requestId, List<String> certificateList) Servicio de Validación Masiva de Certificados.
BatchResponse	**dssAfirmaBatchVerifySignature**(String requestId, List<VerifySignatureObject> verifySignatureObjectList, AfirmaClient.SignatureType signatureTyp AfirmaClient.XmlSignatureMode xmlSignatureMode) Servicio de Validación Masiva de Firmas.
SignResponse	**dssAfirmaCoSign**(String documentArchiveId) Servicio de Firma Delegada que permite la realización de operaciones de cofirma de servidor en los formatos soportad la plataforma.
SignResponse	**dssAfirmaCoSign**(String documentArchiveId, String aliasCertificadoFirma) Servicio de Firma Delegada que permite la realización de operaciones de cofirma de servidor en los formatos soportad la plataforma.
SignResponse	**dssAfirmaCounterSign**(String documentArchiveId) Servicio de Firma Delegada que permite la realización de operaciones de contrafirma de servidor en los formatos sopo por la plataforma.
SignResponse	**dssAfirmaCounterSign**(String documentArchiveId, String aliasCertificadoFirma) Servicio de Firma Delegada que permite la realización de operaciones de contrafirma de servidor en los formatos sopo por la plataforma.
SignResponse	**dssAfirmaSign**(String document) Servicio de Firma Delegada que permite la realización de operaciones de firma de servidor en los formatos soportados

Cliente DSS (afirma-dss-client)

Interfaz del componente (II)

Servicio @firma / operación	Métodos de la interfaz
DSSAfirmaArchiveRetrieval / archiveRetrieval	dssAfirmaArchiveRetrieval
DSSAfirmaArchiveSubmit / archiveSubmit	dssAfirmaArchiveSubmit
DSSAfirmaSign / sign	dssAfirmaSign, dssAfirmaCoSign, dssAfirmaCounterSign
DSSAfirmaVerify / verify	dssAfirmaVerify, dssAfirmaUpgrade
DSSAfirmaVerifyCertificate/ verify	dssAfirmaVerifyCertificate
DSSBatchVerifySignature / verifySignatures	dssAfirmaBatchVerifySignature
DSSBatchVerifyCertificate / verifyCertificates	dssAfirmaBatchVerifyCertificate
DSSAsyncRequestStatus / getProcessResponse	dssAfirmaAsyncRequestStatus

Cliente DSS (afirma-dss-client)

Integración - ¿cómo se integra en mi aplicación? (I)

- **Aplicaciones desarrolladas 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>1.x.x</version>
</dependency>
```

Del mismo modo debe incluirse la referencia al repositorio de software donde se encuentra el componente:

```
<repositories>
		<repository>
			<id>ArtifactoryRepo</id>
			<url>dav:http://[HOST]:[PUERTO]/artifactory/libs-releases</url>
		</repository>
</repositories>
```

Cliente DSS (afirma-dss-client)

Integración - ¿Cómo se integra en mi aplicación? (II)

- **Aplicaciones no Maven**

El componente afirma-dss-client está implementado en su totalidad en la librería **afirma-dss-client-x.x.x.jar**. Tan sólo hay que incluir las siguientes dependencias:

- commons-logging:commons-logging:jar:1.1
- log4j:log4j:jar:1.2.12
- logkit:logkit:jar:1.0.1
- avalon-framework:avalon-framework:jar:4.1.3
- javax.servlet:servlet-api:jar:2.3
- commons-lang:commons-lang:jar:2.1
- org.apache.ws.commons.util:ws-commons-util:jar:1.0.2
- xml-apis:xml-apis:jar:1.0.b2
- xerces:xercesImpl:jar:2.4.0
- axis:axis:jar:1.4
- org.apache.axis:axis-jaxrpc:jar:1.4
- org.apache.axis:axis-saaj:jar:1.4
- axis:axis-wsdl4j:jar:1.5.1
- commons-discovery:commons-discovery:jar:0.2
- org.apache.ws.security:wss4j:jar:1.6.7
- org.apache.santuario:xmlsec:jar:1.5.2
- org.opensaml:opensaml:jar:2.5.1-1
- org.opensaml:openws:jar:1.4.2-1
- org.opensaml:xmltooling:jar:1.3.2-1
- org.slf4j:slf4j-api:jar:1.6.1
- joda-time:joda-time:jar:1.6.2

Cliente DSS (afirma-dss-client)

Integración - Configuración

Para configurar el cliente, tan sólo hay que configurar un fichero de propiedades denominado **afirma.properties** en el CLASSPATH de la aplicación que lo utiliza.

```
############################################################
# Información de conexión
############################################################
afirma.idapp = IDAPP
afirma.host = ws028.juntadeandalucia.es
afirma.truststore = almacenconfianza.jks
afirma.truststorePassword = pass

############################################################
# Información de autenticación
############################################################
# BinarySecurityToken
#afirma.authorization.ks.path = PATH_PKCS12

# Tipo del almacen 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

############################################################
# Información sobre formatos de firma
############################################################
afirma.signaturetype = XAdESv1.3.2
afirma.xmlsignaturemode = ENVELOPING
afirma.signatureform = BES
afirma.signaturePolicy = urn:oid:2.16.724.1.3.1.1.2.1.8
afirma.keyname = default
```

Cliente DSS (afirma-dss-client)

Integración - Ejemplos (I)

Ejemplo de firma delegada (firma servidor)

```
// Se obtiene el documento a firmar
String documento = Base64.encode( "<texto atributo=\"valor\">Documento
ejemplo</texto>".getBytes() );

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

// Se invoca la operación dssAfirmaSign con los parámetros requeridos.
SignResponse signResponse =
afirmaClient.dssAfirmaSign( documento, AfirmaClient.SignatureType.XAdES_v132,
AfirmaClient.XmlSignatureMode.ENVELOPING, AfirmaClient.SignatureForm.T );

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

Cliente DSS (afirma-dss-client)

Integración - Ejemplos (II)

Ejemplo de actualización de firma

```
// 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 );
```

Cliente Autenticación (afirma-authentication-client)

Índice

- Introducción
- Funcionalidad y servicios
- Integración
  - Interfaz
  - Paso a paso
  - Dependencias
  - Configuración
  - Ejemplos

Cliente Autenticación (afirma-authentication-client)

Introducción

- **Objetivo.** Facilitar a los integradores la interacción con los servicios de autenticación disponibles en la plataforma @firma a través de la fachada de autenticación por tickets.

- **¿Qué es?** Componente software que implementa la lógica necesaria relacionada con la mensajería y las comunicaciones (SOAP), ofreciendo a los integradores una interfaz sencilla basada en servlets. El entregable lo conforman:
  - Librería core y dependencias (ficheros “jar”)
  - Manual de integrador
  - Javadoc
  - Código fuente del componente y clases de test JUnit

- **¿Qué tecnología?** Está desarrollado en java y está construido con Maven siguiendo las recomendaciones de MADEJA.

Cliente Autenticación (afirma-authentication-client)

Funcionalidad y servicios (I)

¿Qué hace?

1.Lógica de mensajería:

- Realiza la composición de la petición XML
- Procesamiento de la respuesta XML

2.Envío y recepción del mensaje SOAP:

- Transparente para integrador
- Utiliza librerías de código abierto: Axis, WS Security…

3.Módulo de servlets:

- Servlet de llamada → Punto de entrada al componente
- Servlet de retorno → Incluye el resultado en sesión

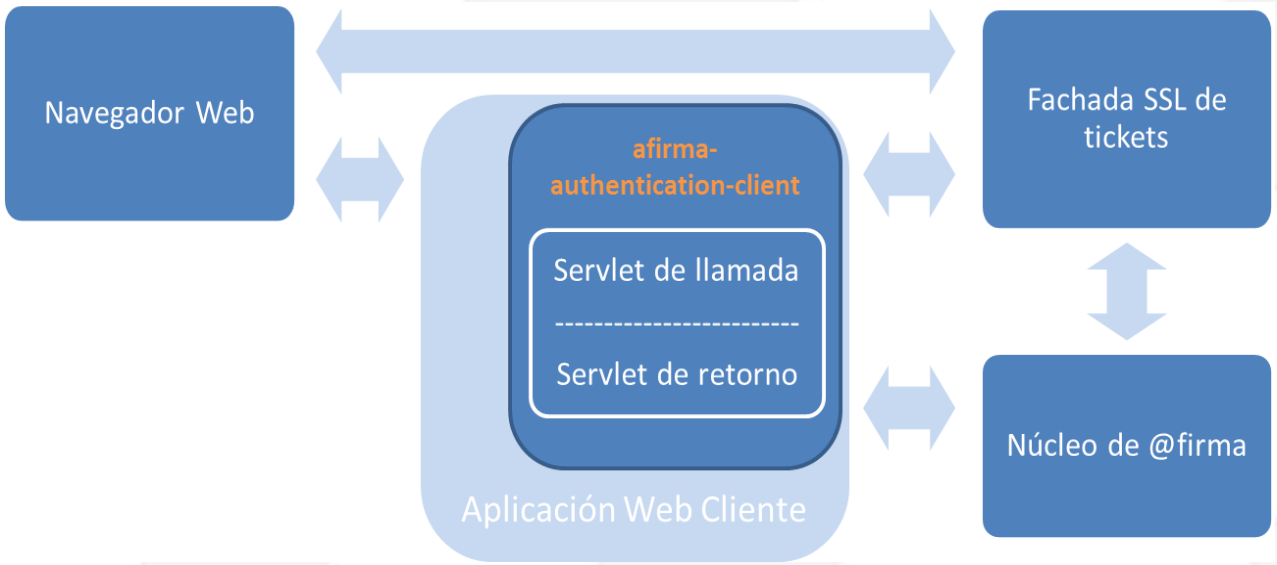
Cliente Autenticación (afirma-authentication-client)

Funcionalidad y servicios (II)

¿Qué servicios de autenticación utiliza?

- Fachada SSL de autenticación por tickets
- Servicios web de autenticación por tickets

Arquitectura lógica



Cliente Autenticación (afirma-authentication-client)

Integración e interfaz

- **CallAuthenticationServlet**: Servlet que se encarga de recopilar la información necesaria del navegador web (identificador de sesión), solicitar el ticket invocando el servicio GenerarTicket y redirigir el flujo de la aplicación a la fachada SSL de autenticación por ticket. Los parámetros que se facilitan a la fachada SSL son los siguientes:
  - identificador del ticket
  - identificador de la aplicación en @firma
  - identificador de sesión
  - URL del servlet de retorno

- **ReturnAuthenticationServlet**: Servlet que verifica la corrección de los datos facilitados por la fachada e invoca al servicio ObtenerInfoValidacionTicket, el cual provee los datos de validación del ticket y del certificado. Una vez finalizado el proceso, el componente de autenticación incluye los siguientes datos en sesión como resultado del proceso de validación del ticket:
  - resultado del proceso (éxito o error).
  - descripción del resultado del proceso.
  - datos del certificado validado.

  La información incluida en sesión debe ser recuperada por la aplicación web cliente para finalizar el proceso de autorización de acceso a la misma.

Cliente Autenticación (afirma-authentication-client)

Integración paso a paso

1. Incluir las dependencias java en el classpath de la aplicación.

2. Establecer la configuración adecuada en el fichero afirma.properties e incluirlo en el classpath de la aplicación.

3. Definir los servlets de llamada y retorno en el fichero web.xml.

4. Invocar el servlet de llamada (**CallAuthenticationServlet**) desde algún punto de la aplicación.

5. Recuperar los datos devueltos por el servlet de retorno (**ReturnAuthenticationServlet**) que se incluyen en el atributo de sesión "*afirma_authentication_client_response*".

Cliente Autenticación (afirma-authentication-client)

Integración - Dependencias

- Los servicios web de @firma 5 y el servidor SSL de la fachada de autenticación por tickets deben ser visibles desde el servidor donde se pretende ejecutar el componente.

- Dependencias software:

  - JDK 1.5 o superior
  - Librerías comunes (commons-logging, commons-discovery, commons-lang…)
  - Log4j
  - Axis 1.4
  - WS Security
  - XML (xml-apis y xerces)

Cliente Autenticación (afirma-authentication-client)

Integración - Configuración (I)

- La configuración se establece en el fichero afirma.properties, que debe incluirse en el classpath de la aplicación.

- Los parámetros de configuración son:

  - Identificador de aplicación (obligatorio)
  - Datos de autenticación:
    - UsernameToken
    - BinarySecurityToken
  - Datos del servidor (Obligatorio)
  - Almacén de certificados de confianza (SSL)
  - Host del servidor SSL de autentiación
  - URL del servlet de retorno definido en web.xml
  - URL de vuelta de la aplicación cliente tras finalizar el proceso

Cliente Autenticación (afirma-authentication-client)

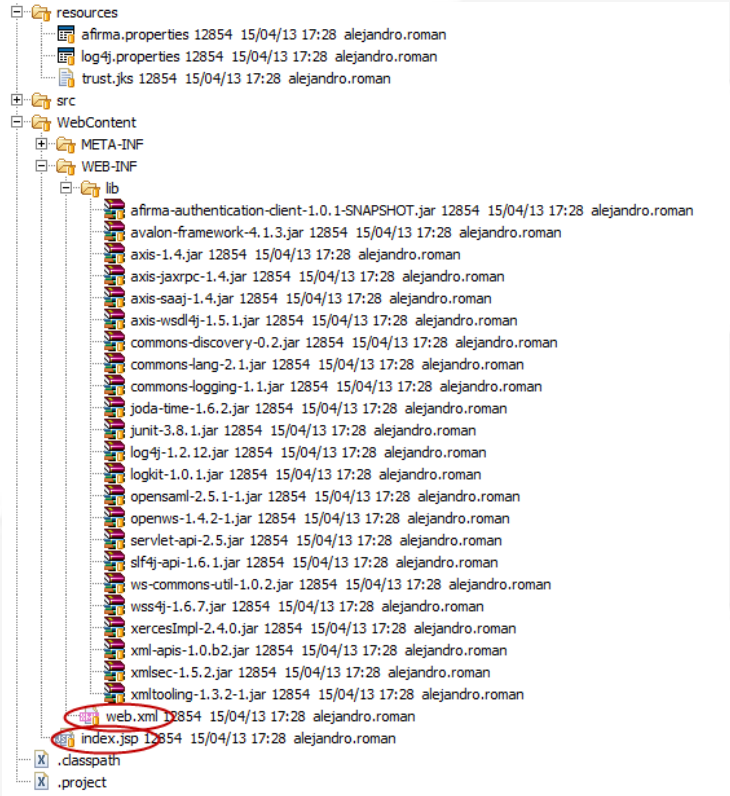
Integración - Configuración (II)

```
################################################################################
# Configuración del componente afirma-authentication-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 = STERIA_TEST
################################################################################

################################################################################
# BinarySecurityToken
################################################################################
afirma.authorization.ks.path = cert.p12
afirma.authorization.ks.type = PKCS12
afirma.authorization.ks.password = pass
afirma.authorization.ks.cert.alias = alias

################################################################################
# UsernameToken
################################################################################
afirma.user = usuario
afirma.password = password
################################################################################

################################################################################
# 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 incluído en el CLASSPATH de la aplicación, por lo que en 'afirma.truststore'
# debe indicarse el nombre del fichero JKS (o bien, la rula relativa al mismo dentro del CLASSPATH).
# 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:
# afirma.truststore = almacenconfianza.jks
# También se admite referenciar al almacén de confianza mediante su ruta absoluta.
################################################################################
afirma.truststore = trust.jks
afirma.truststorePassword = testdes
################################################################################

################################################################################
# Autenticación Fachada de Tickets
# Es necesario incluir los siguientes datos si se va a hacer uso de la autenticación mediante la fachada de tickets: servidor de autenticación y urls de retorno
################################################################################
# Host del servidor de la fachada SSL de autentiación por tickets
afirma.tickets.auth.host = ws116.juntadeandalucia.es
# URL del servlet de retorno definido en el fichero web.xml
afirma.tickets.url.servlet = https://[HOST_APP]/autFachadaTicketComponente/ReturnAuthenticationServlet
# URL de vuelta de la aplicación cliente tras finalizar el proceso de autenticación
afirma.tickets.url.app = https://[HOST_APP]/autFachadaTicketComponente/index.jsp
```

Cliente Autenticación (afirma-authentication-client)

Integración - Ejemplos (I)

- Creamos la estructura de una aplicación web de ejemplo para implementar la autenticación:



Cliente Autenticación (afirma-authentication-client)

Integración - Ejemplos (II)

- Definimos los servlets de llamada y retorno del componente en el fichero web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>AuthenticationApp</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>CallAuthenticationServlet</servlet-name>
        <servlet-class>
            es.juntadeandalucia.afirma.servlet.CallAuthenticationServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CallAuthenticationServlet</servlet-name>
        <url-pattern>/CallAuthenticationServlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>ReturnAuthenticationServlet</servlet-name>
        <servlet-class>
            es.juntadeandalucia.afirma.servlet.ReturnAuthenticationServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ReturnAuthenticationServlet</servlet-name>
        <url-pattern>/ReturnAuthenticationServlet</url-pattern>
    </servlet-mapping>

</web-app>
```

Cliente Autenticación (afirma-authentication-client)

Integración - Ejemplos (III)

- Invocamos al servlet de llamada y recuperamos los datos resultantes de la validación del objeto “session”:

```
<%@page import="es.juntadeandalucia.afirma.authentication.beans.CertificateInfo"%>
<%@page import="es.juntadeandalucia.afirma.authentication.beans.ResultAuthenticationBean"%>
<%@page import="java.util.Iterator"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Ejemplo Autenticación Web.</title>
</head>
<body>

    <div align="center">

        <a href="CallAuthenticationServlet" target="_self">Autenticaci&oacute;n
            mediante certificados digitales basado en Tickets</a>


        <%
        ResultAuthenticationBean result = new ResultAuthenticationBean();
                result = (ResultAuthenticationBean) session.getAttribute( "afirma_authentication_client_response" );
        %>
        <%
                if(result != null){

                    out.println( "\nTicket válido: " + result.isValidTicket(  ));
                    out.println( "\nDescripción resultado autenticación: " + result.getResultDescription(  ) );

                    if(result.getCertificateData(  ) != null){
                        out.println( "\nDATOS DEL CERTIFICADO:" );

                        Iterator<CertificateInfo> iterador = result.getCertificateData(  ).iterator(  );
                        while(iterador.hasNext(  )){
                            CertificateInfo cer = (CertificateInfo) iterador.next(  );

                            out.print( cer.getFieldIdentity(  ) + " - ");
                            out.print( cer.getFieldValue(  ) );
                            out.println( "\n" );
                        }
                    }
                }
        %>
    </div>

</body>
</html>
```

Cliente Custodia (afirma-custody-client)

Índice

Cliente Custodia (afirma-custody-client)

Introducción

- **Objetivo.** Facilitar a los integradores la interacción con los servicios básicos del módulo de Custodia disponibles en la plataforma @firma.

- **¿Qué es?** Componente software que implementa la lógica necesaria relacionada con la mensajería y las comunicaciones (SOAP), ofreciendo a los integradores una interfaz sencilla. El entregable lo conforman:
  - Librería core y dependencias (ficheros jar)
  - Manual de integrador
  - Javadoc
  - Código fuente del componente y clases de test JUnit

- **¿Qué tecnología?** Está desarrollado en java y está construido con Maven siguiendo las recomendaciones de MADEJA.

Cliente Custodia (afirma-custody-client)

Funcionalidad y servicios (I)

¿Qué hace?

1.Implementa la lógica de mensajería:

- Realiza la composición de la petición XML
- Procesamiento de la respuesta XML

2.Implementa el envío y recepción del mensaje SOAP:

- Transparente para integrador
- Utiliza librerías de código abierto: Axis, WS Security…

Cliente Custodia (afirma-custody-client)

Funcionalidad y servicios (II)

¿Qué servicios de custodia utiliza?

- **GetDocumentContent**: Obtiene el contenido de un documento almacenado en el módulo de Custodia, a partir del identificador de transacción de la firma del documento.

- **GetDocumentContentByDocId**: Obtiene el contenido de un documento almacenado en el módulo de Custodia, a partir de su identificador de documento.

- **GetESignature:** Obtiene la firma electrónica almacenada en el módulo de Custodia a partir del identificador de transacción.

Cliente Custodia (afirma-custody-client)

Integración - Interfaz (I)

- **getDocumentContent**
  - Parámetros de entrada:
    - **transactionId**: Identificador de la transacción de la firma del documento.
  - Respuesta:
    - Objeto de tipo **GetDocumentContentResponse** con los datos de la respuesta.
  - Cabecera del método:
    - **public GetDocumentContentResponse getDocumentContent(String transactionId) throws CustodyException;**

- **getDocumentContentByDocId**
  - Parámetros de entrada:
    - **docId**: Identificador del documento custodiado.
  - Respuesta:
    - Objeto de tipo **GetDocumentContentResponse** con los datos de la respuesta.
  - Cabecera del método:
    - **public GetDocumentContentResponse getDocumentContentByDocId(String docId) throws CustodyException;**

Cliente Custodia (afirma-custody-client)

Integración - Interfaz (II)

- **getESignature**
  - Parámetros de entrada:
    - **transactionId**: Identificador de transacción de una firma custodiada en @Firma.
  - Respuesta:
    - Objeto de tipo **GesESignatureResponse** con los datos de la respuesta.
  - Cabecera del método:
    - **public GetESignatureResponse getESignature(String transactionId) throws CustodyException;**

Cliente Custodia (afirma-custody-client)

Integración - Dependencias

- Los servicios web de @firma 5 deben ser visibles desde el servidor donde se pretende ejecutar el componente.

- Dependencias software:

  - JDK 1.5 o superior
  - Librerías comunes (commons-logging, commons-discovery, commons-lang…)
  - Log4j
  - Axis 1.4
  - WS Security
  - XML (xml-apis y xerces)

Cliente Custodia (afirma-custody-client)

Integración - Configuración (I)

- La configuración se establece en el fichero afirma.properties, que debe incluirse en el classpath de la aplicación.
- Los parámetros de configuración son:
  - Identificador de aplicación (obligatorio)
  - Datos de autenticación:
    - UsernameToken
    - BinarySecurityToken
  - Datos del servidor (Obligatorio)
  - Almacén de certificados de confianza (SSL)

Cliente Custodia (afirma-custody-client)

Integración - Configuración (II)

```
################################################################################
# Configuración del componente afirma-custody-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 = STERIA_TEST
################################################################################

################################################################################
# BinarySecurityToken
################################################################################
afirma.authorization.ks.path = cert.p12
afirma.authorization.ks.type = PKCS12
afirma.authorization.ks.password = pass
afirma.authorization.ks.cert.alias = alias
################################################################################

################################################################################
# UsernameToken
################################################################################
afirma.user = usuario
afirma.password = password
################################################################################

################################################################################
# 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 incluído en el CLASSPATH de la aplicación, por lo que en 'afirma.truststore'
# debe indicarse el nombre del fichero JKS (o bien, la rula relativa al mismo dentro del CLASSPATH).
# 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:
# afirma.truststore = almacenconfianza.jks
# También se admite referenciar al almacén de confianza mediante su ruta absoluta.
################################################################################
afirma.truststore = trust.jks
afirma.truststorePassword = testdes
################################################################################
```

Cliente Custodia (afirma-custody-client)

Integración - Ejemplos (I)

```
public class GetDocumentContentRequestTest extends TestCase
{
    public GetDocumentContentRequestTest( String testName )
    {
        super( testName );
    }

    public static Test suite()
    {
        return new TestSuite( GetDocumentContentRequestTest.class );
    }

    public void testGetDocumentContentRequest() throws Exception
    {
        // Creo una instancia del componente
        CustodyClient custodyClient = CustodyClientBuilder.getCustodyClient();

        // Invoco el servicio GetDocumentContent
        GetDocumentContentResponse response = custodyClient.getDocumentContent( "1365674096609848" );

        assertTrue( "true".equals( response.getStatus() ) );
    }

    public void testGetDocumentContentByDocIdRequest() throws Exception
    {
        // Documento a firmar
        String document = Base64.encode( "Hola mundo!".getBytes() );
        String documentName = "holaMundo.txt";
        String documentType = "txt";

        // Creo una instancia del componente
        CustodyClient custodyClient = CustodyClientBuilder.getCustodyClient();

        // Invoco el servicio StoreDocument
        StoreDocumentResponse response1 = custodyClient.storeDocument( document, documentName, documentType );

        System.out.println( response1 );

        assertTrue( "true".equals( response1.getStatus() ) );

        // Invoco el servicio GetDocumentContent
        GetDocumentContentResponse response2 = custodyClient.getDocumentContentByDocId( response1.getDocumentId() );

        assertTrue( "true".equals( response2.getStatus() ) );
    }
}
```

Cliente Custodia (afirma-custody-client)

Integración - Ejemplos (II)

```
public class GetESignatureRequestTest extends TestCase
{
    public GetESignatureRequestTest( String testName )
    {
        super( testName );
    }

    public static Test suite()
    {
        return new TestSuite( GetESignatureRequestTest.class );
    }

    public void testGetESignatureRequest() throws Exception
    {
        // Creo una instancia del componente
        CustodyClient custodyClient = CustodyClientBuilder.getCustodyClient();

        // Invoco el servicio GetESignature
        GetESignatureResponse response = custodyClient.getESignature( "1365674096609848" );

        System.out.println( response );

        assertTrue( "true".equals( response.getStatus() ) );
    }
}
```

ÍNDICE

Introducción

¿Qué es el Componente Validador?

- El **Componente Validador** es un componente software cuya finalidad es la de informar al usuario del grado de **compatibilidad** de su equipo informático con el Cliente de Firma Electrónica de la Junta de Andalucía distribuido por la Consejería de Hacienda y Administración Pública.
- El componente realiza dos operaciones principales:
    - **Obtiene información** del equipo del usuario a través de consultas estándares sobre variables definidas en su navegador web.
    - **Contrasta dicha información con una matriz de compatibilidad** del cliente de firma, definida por la Consejería de Hacienda y Administración Pública.
- Se presenta mediante una aplicación web que debe ser enlazada por las aplicaciones que hagan uso del Cliente de Firma Electrónica.

Integración

Acceso al Componente Validador (I)

- El acceso al componente validador se realiza mediante una petición HTTP a la aplicación web ***afirma-validator***. Dicha URL debe incluir los siguiente parámetros obligatorios:
  - **clientVersion:** Versión del cliente de firma a validar (3.3.1).
  - **signatureFormat:** Formato de firma que realiza la aplicación cliente (CADES, XADES…).
  - **signatureMode:** Modo de las firmas que realiza la aplicación cliente (IMPLICIT o EXPLICIT).
  - **callbackUrl:** URL de vuelta de la aplicación cliente.

- La aplicación web cliente debe incluir en su página inicial la petición HTTP para permitir al usuario la comprobación de su sistema.

Integración

Acceso al Componente Validador (II)

- El código HTML de la petición HTTP debe tener una apariencia similar a:

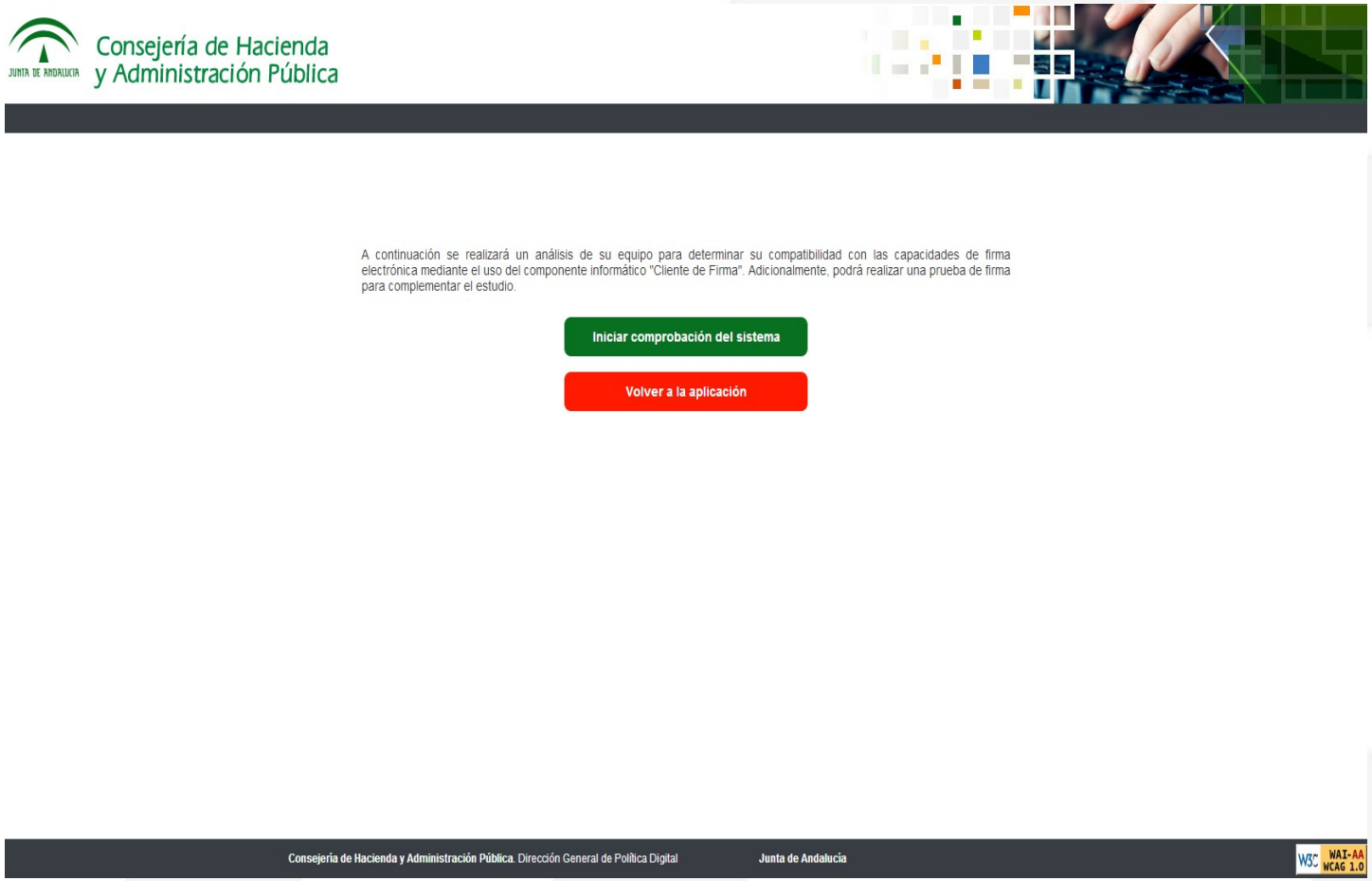
```
<html>
<body>
...
<a href="http://[HOST_AFIRMA_VALIDATOR]/validator.action?clientVersion=3.3.1&signatureFormat=XADES&signatureMode=IMPLICIT&callbackUrl=https://ws031.juntadeandalucia.es/notificaciones/snja">Compruebe si su equipo es compatible</a>
...
</body>
</html>
```

- Se permite el envío de los parámetros mediante GET y POST.

Integración

Web del Componente Validador (I)

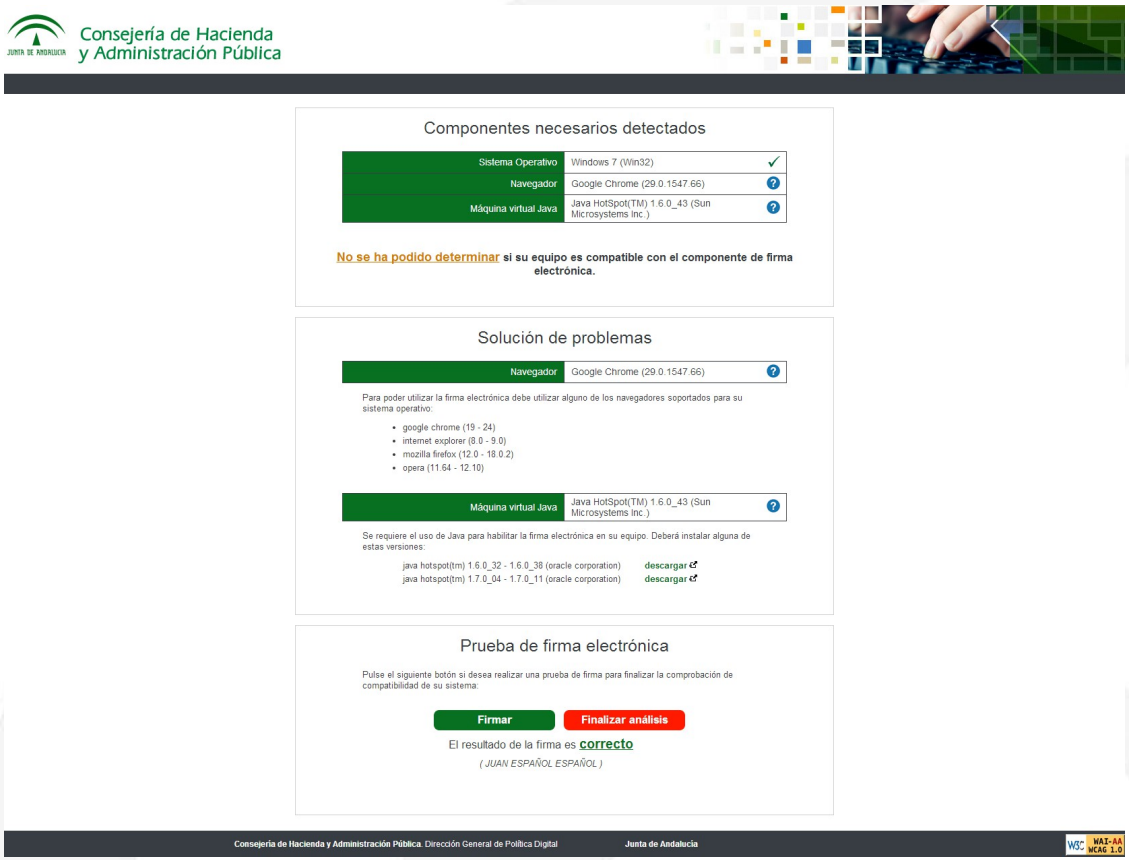
- Se muestra a continuación la pantalla inicial del componente validador:



Integración

Web del Componente Validador (II)

- Se muestra a continuación la pantalla de resumen de resultados una vez obtenidos y contrastados los datos del entorno del usuario:



Integración

Obtención de datos

- Durante la ejecución de la aplicación afirma-validator, el usuario puede retornar a la aplicación web cliente mediante los botones **“Volver a la aplicación”** o **“Finalizar análisis”**.
- Si el usuario opta por finalizar el análisis, el Componente Validador realizará la siguiente petición HTTP:

  [callbackURL]?r=[resumenDatosBase64URLEncoded]

- Opcionalmente, la aplicación cliente puede procesar los datos incluidos en el parámetro devuelto para obtener el resumen del resultado de la validación. Se muestra un ejemplo de los datos decodificados:

```
<?xml version="1.0"?>
<afirmaValidator>
  <osName>Windows</osName>
  <osVersion>7</osVersion>
  <osArch>Win32</osArch>
  <browserName>Google Chrome</browserName>
  <browserVersion>29.0.1547.66</browserVersion>
  <jreName>Java HotSpot(TM)</jreName>
  <jreVersion>1.6.0_43</jreVersion>
  <jreVendor>Sun Microsystems Inc.</jreVendor>
  <clientVersion>3.3.1</clientVersion>
  <clientDistribution>MEDIA</clientDistribution>
  <signatureFormat>CADES</signatureFormat>
  <signatureMode>IMPLICIT</signatureMode>
  <osResult>SC</osResult>
  <browserResult>ND</browserResult>
  <jreResult>ND</jreResult>
  <signTestResult>OK</signTestResult>
</afirmaValidator>
```

Integración

Próximas actuaciones

- Actualización periódica de la matriz de compatibilidad del cliente 3.3.1. en lo referente a nuevas versiones de
  - Sistemas Operativos
  - Máquina Virtual Java
  - Navegadores

- Inclusión del componente Miniapplet y de su matriz de compatibilidad.

- Corrección de problemas eventuales.

ÍNDICE

Introducción

Novedades funcionales en @firma 5.5 (I)

- Módulo de firma
  - Inclusión de nuevos formatos de firma avanzada:
    - CAdES/XAdES: -EPES, -C, –X, -XL, -A
    - PAdES-BES.
  - Política de Firma. Se permite la generación y validación de firmas electrónicas en base a políticas de firmas admitidas por la plataforma.
  - Autodetección del formato de firma en firmas de usuario en fases.

- Módulo DSS:
  - Mayor detalle en el resultado del servicio de validación de firmas mediante interfaz OASIS-DSS.
  - Incorporación del nuevo servicio de validación de certificados mediante interfaz OASIS-DSS.
  - Servicio de Upgrade de firmas mejorado mediante interfaz OASIS-DSS, incluyendo la posibilidad de realizar la actualización a nuevos formatos de firma extendidos.

Introducción

Novedades funcionales en @firma 5.5 (II)

- Módulo de auditoría actualizada para la auditoría y estadísticas de las nuevas funcionalidades de firma.

- Módulo de Administración:
  - Incluida la Administración y Gestión de Políticas.
  - Evolución del sistema de configuración de la plataforma, dando la opción de que esta ahora sea almacenada, gestionada y compartida por los nodos que conformen el clúster, mediante la base de datos.

- Fachada de Tickets 2.0.0:
  - Renombrado de paquetes: es.juntadeandalucia.afirma
  - Adaptación a dependencias de @firma 5.5.

Introducción

Novedades en @firma 5.5 (II)

Componente	@firma 5.3.1	@firma 5.5
Java	Java 5	Java 6
Jboss	Jboss 4.2.1	Jboss 4.2.1
Interfaz de administración	Iguax droy	Mejorada con Iguax 2.3.02 y TiFrameWork 2.6.6
Extensión de compatibilidad	Activa	Sólo firma de ficheros
Fachada de tickets	1.1.2	2.0.0 (adaptación)

Actualización

Actualización de @firma 5.3.1 a 5.5.

- Documentos de referencia:
  - Plan_Implantación@Firma5_5.pdf
  - @Firma-InstalacionyDespliegue-MAN.pdf
  - FirmaTickets-InstalacionyDespliegue-MAN.pdf
- Actualización del modelo de datos.
- Consideraciones sobre la implantación del servidor.
  - Trazas de auditoría y estadísticas (por lotes)
    - **productor.modo=3**
    - **consumidor.procesador.numtransacciones=50**
  - Extractor de datos de manera externa.
  - Configuración almacenada en Base de Datos no se recomienda su uso
  - Consola JMX y Administración: cambiar contraseñas
  - Formato de los almacenes de certificados: de JKS a JCE
  - Cliente TSA: Incluir la librería iaik_jce_full.jar
  - Versión de JDK: Java 6
  - Tamaño de las peticiones:
    - **parameter.AxisServletExtended.MAX_LENGTH_OF_MESSAGE=0**

Actualización

Implantación del núcleo paso a paso

1. Realizar copia de seguridad de los esquemas de BD de @firma 5.3.1

2. Actualizar el modelo de datos a la versión 5.5 ejecutando los scripts de la carpeta: **Actualizacion/5.3.1_008/ScriptsSQL**

3. Realizar la implantación de los componentes de servidor de @Firma siguiendo el documento @Firma-InstalacionyDespliegue-MAN.pdf teniendo en cuenta las consideraciones indicadas en la anterior transparencia.

4. Realizar migración de los almacenes de certificados que aún mantienen el formato JKS.

5. Realizar la importación de la última versión publicada de la política de @Firma 5.3.1.

6. Migrar las aplicaciones modificando el fichero **configuracionArrobaFirma5_0.xml**

7. Realizar pruebas de integración para validar la implantación.

Actualización

Implantación de la fachada de tickets 2.0.0

- Se plantea como una adecuación de la versión 1.1.2 para conseguir la integración con el núcleo de @Firma en su versión 5.5.

- Requisitos:

  - Disponer de una versión 1.1.2 correctamente instalada y configurada sobre un núcleo de @Firma en su versión 5.3.1 (se reutilizan todos los ficheros de configuración de dicha versión para la nueva implantación).
  - Disponer de un núcleo de @Firma 5.5 correctamente instalado y configurado.
  - Instalar la herramienta de despliegue ANT proporcionada en el CD de Instalación de la plataforma.

- Se facilita un fichero ZIP con todo lo necesario para el despliegue de los servicios de autenticación sobre el núcleo de @firma 5.5. y la nueva fachada SSL (**sslServerFacade)**.

  - Los pasos a seguir para la implantación vienen definidos con detalle en el documento **FirmaTickets-InstalacionyDespliegue-MAN.pdf**

Muchas gracias

Dirección General de Política Digital

Consejería de Hacienda y Administración Pública