Junta de Andalucía

Manual de integración

Port@firmas v3.5

Versión: v01r04
Fecha: 20/09/2024

HOJA DE CONTROL

Título	Manual de integración		
Entregable	Port@firmas v3.5.x		
Nombre del Fichero	PF-PRO-Manual_de_integración.odt		
Autor	UTE		
Versión/Edición	v01r04	**Fecha Versión**	20/09/2024
Aprobado por		**Fecha Aprobación**	20/09/2024
		Nº Total Páginas	53

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Área	Fecha del Cambio
v01r00	Creación del documento	UTE	ADA	30/10/2023
v01r02	Se añade estado ELIMINADO en las peticiones	UTE	ADA	02/05/2024
v01r03	Generación informe firma con aplicaciones con custodia por referencia	UTE	ADA	20/09/2024
v01r04	Integración con ENIDOC 2.0	UTE	ADA	31/10/2024

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Área	Nº Copias
Manuel Escobar Montes	Jefe de Servicio	ADA	1
José Ignacio Cortés Santos	Director de Proyecto	ADA	1

ÍNDICE

Sumario

1 INTRODUCCIÓN

El objetivo del presente documento es servir como guía para el uso de los servicios web de la aplicación Port@firmas Corporativo v3.5.x por parte de los integradores que estén desarrollando aplicaciones clientes que hagan uso de dichos servicios.

2 SERVICIOS WEB PORT@FIRMAS

Port@firmas dispone de dos fachadas Web Service para atacar a las API's desde terceras aplicaciones. Dichas fachadas son la de los servicios de Port@firmas v2 con una funcionalidad ampliada y una fachada para compatibilidad con los servicios de Port@firmas v1.

Cada fachada contiene varios servicios que serán descritos:

- Fachada Web Services v1
  - Servicio de envío de peticiones de firma.
- Fachada Web Services v2
  - Servicio de consulta.
  - Servicio de modificación.

Para poder hacer uso de las nuevas funcionalidades de Port@firmas v3 es necesario hacer uso de la fachada de servicios de v2, que irá evolucionando con cada nueva funcionalidad que se añada a la aplicación. La fachada de servicios de v1 no ofrece nuevas funcionalidades ni evolucionará dado que su función es únicamente ofrecer retrocompatibilidad con versiones anteriores de Port@firmas. La fachada que ofrecía la funcionalidad de firma remota se ha eliminado definitivamente.

Los servicios v1 serán discontinuados en futuras versiones de la herramienta y no se garantiza su futura presencia.

2.1 Nuevas funcionalidades

Se ha eliminado la funcionalidad del servicio de creación de una petición a partir de una firma, que se mantenía por compatibilidad con versiones previas (el servicio se llama "createRequestFromSign"). Se ha eliminado también la funcionalidad de la firma remota en la v1. También se elimina la funcionalidad del método "deleteDocument" de los servicios web versión 2. En el caso de que se añada un documento por error se tendría la opción de llamar al método "deleteRequest" y eliminar la petición entera o abortarla en su caso.

En el caso de que no se utilice estos servicios los usuarios que estén realizando una migración de una v3.0.X a la v3.3.X y que utilicen los servicios de la v2 no tendrán que realizar modificación alguna para adaptarse a la v3.3.X

A partir de la versión 3.3.30 los identificadores de las peticiones y documentos (campo C_HASH) pasan de tener 10 caracteres a 20 para nuevas aplicaciones.

Se elimina el uso de los servicios de la versión 1 de la firma en trámite, consulta externa de peticiones y la obtención de los informes de firma.

Se actualiza la librería pfirmav2CXFClient a la versión 3.4.10 la cual debe de utilizarse para futuras integraciones. Dicha librería tiene restringido el uso. Más información en el punto 2.3.4.

Para la generación de informes de firma por parte de la aplicación que se integre debe de utilizarse la librería firma-report, la cual genera el informe de firma a partir de un documento PDF original, los firmantes y fecha.

2.2 Fachada Web Services v1

Tal como ya se ha indicado, esta fachada solo se ofrece por motivos de compatibilidad y no puede utilizarse en nuevas integraciones. Las aplicaciones que aún hagan uso de esta fachada v1 de servicios web deben adaptarse a la fachada de servicios v2 ya que próximamente su uso será discontinuado.

2.3 Fachada Web Services v2

La fachada ofrece dos servicios, uno para consulta y otro para modificación.

2.3.1 Servicio de consulta

Proporciona métodos de consulta de peticiones y datos asociados, usuarios y puestos de trabajo. El WSDL descriptor del servicio se encuentra en la siguiente URL:

```
http://servidor:puerto/pfirma/servicesv2/QueryService?wsdl
```

El javadoc de la interfaz webservice en java se puede encontrar dentro del fichero ***javadocWS.zip*** el cual se encuentra en el kit de integración en la carpeta “Kit de desarrollo”.

2.3.2 Servicio de modificación

Ofrece métodos para la creación, edición y envío de peticiones. El WSDL descriptor del servicio se encuentra en la siguiente URL:

```
http://servidor:puerto/pfirma/servicesv2/ModifyService?wsdl
```

El javadoc de la interfaz webservice en java se puede encontrar dentro del fichero ***javadocWS.zip*** el cual se encuentra en el kit de integración en la carpeta "Kit de desarrollo".

2.3.3 Ejemplo de integración con Web Services

Se incluye una aplicación de test llamada ***pfirma-test-ws-v2-3.5-jar-with-dependencies*** *(*el cual puede encontrarse en https://intranet.ada.junta-andalucia.es/intranet-ada/nuestro-trabajo/servicios/portfirmas-corporativo en el enlace del kit de integración junto con el código fuente) para ilustrar el uso de los servicios ofrecidos en la fachada web service v2. Se trata de una aplicación con estructura maven/eclipse y hace uso de unos clientes generados a través de CXF.

Es posible ejecutar la aplicación de pruebas de dos formas:

- Desde Eclipse. La aplicación contiene una clase main que se encarga de lanzar las suites de tests JUnit definidos.
- Como una aplicación java (.jar).

2.3.3.1 Requisitos previos

Las especificaciones del servidor a desarrollar las pruebas deben ser las indicadas en el documento de instalación de Port@firmas v3.5.X.

En cuanto a los requisitos a nivel de equipo cliente, que tenga conexión contra el servidor donde está desplegada la aplicación y tener instalada una JRE 1.8 o superior.

2.3.3.2 Configuración del entorno de pruebas

La aplicación cliente de los servicios web contiene un fichero de propiedades que debe ser configurado para la correcta realización de las pruebas (*ws.properties*). El fichero se encuentra en la carpeta raíz.

```
##URL's servicios
```

```
ws.query.wsdl=http://server:port/pfirma/servicesv2/QueryService?wsdl

ws.mofify.wsdl=http://server:port/pfirma/servicesv2/ModifyService?wsdl


#Codigo de la aplicacion y password

aplicacion.codigo=APP_INTEGRADA

aplicacion.password=123456


#Primer firmante. Debe de tener valor obligatoriamente

createRequestByReference.firmante1=41197904M

createRequestByReference.tipoFirmafirmante1=FIRMA

#Segundo firmante. Si no hay segundo firmante dejarlo vacío

createRequestByReference.firmante2=52988516N

createRequestByReference.tipoFirmafirmante2=VISTOBUENO

#Tercer firmante. Si no hay segundo firmante dejarlo vacío

createRequestByReference.firmante3=23740909X

createRequestByReference.tipoFirmafirmante3=FIRMA

createRequestByReference.asunto=Actividad Comisión Interdepartamental para la
Sociedad de la Información

createRequestByReference.referencia=EXP-2016/00000726-PID@

createRequestByReference.texto=Solicitud SOL-2016/00002596-PID@. Este documento
ha sido puesto a su firma por REYES MOLINA RODRIGUEZ

createRequestByReference.tipoFirma=CASCADA


#Primer documento. Es obligatorio que tenga valores

createRequestByReference.pathArchivo1=C:/user/portafirmas/PDFS/dummy1.pdf

createRequestByReference.nombreArchivo1=Archivo1.pdf

createRequestByReference.csvArchivo1=EGMHRVYF4TNEKE512P4LRU9EDXZ111

createRequestByReference.mimeTypeArchivo1=application/pdf
```

```
createRequestByReference.tipoDocumentoArchivo1=TD99

createRequestByReference.firmableArchivo1=true


#Segundo documento. Si no hay segundo documento dejarlo vacío

createRequestByReference.pathArchivo2=C:/user/portafirmas/PDFS/dummy2.pdf

createRequestByReference.nombreArchivo2=Archivo2.pdf

createRequestByReference.csvArchivo2=EGMHRVYF4TNEKE512P4LRU9EDXZ222

createRequestByReference.mimeTypeArchivo2=application/pdf

createRequestByReference.tipoDocumentoArchivo2=TD99

createRequestByReference.firmableArchivo2=true


#Tercer documento. Si no hay segundo documento dejarlo vacío

createRequestByReference.pathArchivo3=C:/user/portafirmas/PDFS/dummy3.pdf

createRequestByReference.nombreArchivo3=Archivo3.pdf

createRequestByReference.csvArchivo3=EGMHRVYF4TNEKE512P4LRU9EDXZ333

createRequestByReference.mimeTypeArchivo3=application/pdf

createRequestByReference.tipoDocumentoArchivo3=TD99

createRequestByReference.firmableArchivo3=true


deleteRequest.identificador=BZ0pTM9VhsYU1KiDMfef


queryRequestList.identificador1=h8EtCvTIUMs3y1azu2CF

queryRequestList.identificador2=CPt5Fuzc1yPB7IDXCvK3

queryRequestList.identificador3=mP2sM2pX6mcHtOHc25zi


queryHistoricList.identificador1=h8EtCvTIUMs3y1azu2CF

queryHistoricList.identificador2=CPt5Fuzc1yPB7IDXCvK3
```

```
queryHistoricList.identificador3=mP2sM2pX6mcHtOHc25zi

queryRequest.identificador=h8EtCvTIUMs3y1azu2CF

queryHistoric.identificador=mP2sM2pX6mcHtOHc25zi

downloadSign.identificador=y9LfpkQAQzxoxFCrD9mM

queryCsv.identificador=BZ0pTM9VhsYU1KiDMfef

queryComments.identificador=BZ0pTM9VhsYU1KiDMfef
```

2.3.3.3 Parámetros del entorno de pruebas

A nivel de servidor no se requiere ningún requisito especial para Port@firmas. La aplicación de pruebas se lanzará a través de línea de comandos.

Las propiedades a editar en *ws.properties* son las siguientes:

ws.query.wsdl	URL del servicio de consulta.
ws.mofify.wsdl	URL del servicio de modificación.
aplicacion.codigo	Nombre de la aplicación en el sistema de Port@firmas a la cual pertenecerá la petición.
aplicacion.password	Contraseña para autenticación en servicios web.
createRequestByReference.firmante1	NIF o NIE del firmante 1
createRequestByReference.tipoFirmafirmante1	FIRMA o VISTOBUENO (El último firmante no puede ser VISTOBUENO)
createRequestByReference.firmante2	NIF o NIE del firmante 2. (En caso de no

	querer enviar firmante 2 dejarlo vacío)
createRequestByReference.tipoFirmafirmante1	FIRMA o VISTOBUENO (El último firmante no puede ser VISTOBUENO)
createRequestByReference.firmante3	NIF o NIE del firmante 3. (En caso de no querer enviar firmante 3 dejarlo vacío)
createRequestByReference.tipoFirmafirmante1	FIRMA o VISTOBUENO (El último firmante no puede ser VISTOBUENO)
createRequestByReference.asunto	Asunto de la petición
createRequestByReference.referencia	Referencia de la petición
createRequestByReference.texto	Texto de la petición
createRequestByReference.tipoFirma	Tipo de firma (CASCADA o PARALELA)
createRequestByReference.pathArchivo1	Ruta del documento 1 a adjuntar.
createRequestByReference.nombreArchivo1	Nombre del documento 1 a adjuntar.
createRequestByReference.csvArchivo1	CSV del documento 1 a adjuntar
createRequestByReference.mimeTypeArchivo1	MimeType del documento 1 a adjuntar
createRequestByReference.tipoDocumentoArchivo1	Tipo del documento 1 a adjuntar
createRequestByReference.firmableArchivo1	Indica si el documento 1 es firmable o no
createRequestByReference.pathArchivo2	Ruta del documento 2 a adjuntar. (En caso de no querer enviar documento 2 dejarlo vacío)
createRequestByReference.nombreArchivo2	Nombre del documento 2 a adjuntar.
createRequestByReference.csvArchivo2	CSV del documento 2 a adjuntar
createRequestByReference.mimeTypeArchivo2	MimeType del documento 2 a adjuntar
createRequestByReference.tipoDocumentoArchivo2	Tipo del documento 2 a adjuntar
createRequestByReference.firmableArchivo2	Indica si el documento 2 es firmable o no
createRequestByReference.pathArchivo3	Ruta del documento 3 a adjuntar. (En caso de

	no querer enviar documento 2 dejarlo vacío)
`createRequestByReference.nombreArchivo3`	Nombre del documento 3 a adjuntar.
`createRequestByReference.csvArchivo3`	CSV del documento 3 a adjuntar
`createRequestByReference.mimeTypeArchivo3`	MimeType del documento 1 a adjuntar
`createRequestByReference.tipoDocumentoArchivo3`	Tipo del documento 3 a adjuntar
`createRequestByReference.firmableArchivo3`	Indica si el documento 3 es firmable o no
`deleteRequest.identificador`	Identificador de la petición a eliminar
`queryRequestList.identificador1`	Identificador de la petición 1 para la consulta masiva de peticiones
`queryRequestList.identificador2`	Identificador de la petición 2 para la consulta masiva de peticiones
`queryRequestList.identificador3`	Identificador de la petición 3 para la consulta masiva de peticiones
`queryHistoricList.identificador1`	Identificador de la petición 1 para la consulta masiva de histórico de peticiones
`queryHistoricList.identificador2`	Identificador de la petición 2 para la consulta masiva de histórico de peticiones
`queryHistoricList.identificador3`	Identificador de la petición 3 para la consulta masiva de histórico de peticiones
`queryRequest.identificador`	Identificador de la petición a consultar en la consulta simple de peticiones
`queryHistoric.identificador`	Identificador de la petición a consultar en la consulta simple de histórico peticiones
`downloadSign.identificador`	Identificador del documento a descargar
`queryCsv.identificador`	Identificador del documento para consultar su CSV
`queryComments.identificador`	Identificador de la petición para consultar sus comentarios

Para ejecutar la aplicación de test bastará con ejecutar la siguiente instrucción desde línea de comandos:

```
java -jar pfirma-test-ws-v2-3.5-jar-with-dependencies.jar
```

Tras esto se mostrará el menú principal de la aplicación.

Existe la opción de ejecutar el proyecto de pruebas indicando por parámetro el test que se desea realizar y el número de veces que debe ejecutarse la prueba, por ejemplo:

```
java -jar pfirma-test-ws-v2-3.5-jar-with-dependencies.jar 4 3
```

Con la instrucción anterior se ejecutará el test 4 un total de 3 veces.

También se posible abrir el proyecto a través del IDE Eclipse, ejecutando el siguiente comando desde la línea de comandos y situándose en el raíz del proyecto:

```
mvn clean eclipse:eclipse
```

La clase que contiene el método main ejecutable es la siguiente:

```
es.juntadeandalucia.pfirmav2.ws.test.TestLauncher
```

2.3.3.4 Detalle de pruebas

A continuación se detallan cada uno de los casos probados en la aplicación.

La clase ***ClientManager.java*** es la encargada de generar los clientes del servicio web a través de sus respectivas URL definidas en el fichero ***ws.properties***. Dicha clase pertenece al componente pfirmav2CXFClient. Port@firmas para nuevas integraciones es imprescindible activar la autenticación mediante usuario y clave para los servicios web.

Obtención de clientes

Para obtener los clientes autenticados son necesarios además de la url de los servicios, el código de la aplicación en Port@firmas (hará las funciones de usuario) y la clase de autenticación (clase en la que se establece la contraseña para autenticar).

```
ClientManager cm = new ClientManager();



QueryService queryServiceClient =

cm.getQueryServiceClient("http://servidor:puerto/pfirma/servicesv2/QueryService?
wsdl", "app", WSAuthenticate.class);



ModifyService modifyServiceClient =

cm.getModifyServiceClient("http://servidor:puerto/pfirma/servicesv2/ModifyService?
wsdl", "app", WSAuthenticate.class);
```

El identificador de la aplicación debe coincidir con el de la aplicación a la que van asociadas las peticiones que se vayan a gestionar a través del servicio Modify.

La clase WSAuthenticate es una clase de manejo de callback en la que se establece la contraseña para que sea insertada en la petición SOAP y así poder autenticar en los servicios web. Por ello la clase debe implementar la interface CallbackHandler y en concreto el método handle, en el que se establece la contraseña. A continuación se muestra un ejemplo simple de implementación:

```
public class WSAuthenticate implements CallbackHandler {



	public void handle(Callback[] callbacks) throws IOException,

			UnsupportedCallbackException {

		WSPasswordCallback pc = (WSPasswordCallback) callbacks[0];

		pc.setPassword("12345");

	}

}
```

Una vez explicada la forma de obtención de los clientes, se pasa a detallar uno a uno los distintos ejemplos existentes en la aplicación de ejemplo.

Opción 1 – Creación de una petición por referencia – createRequestByReference

El envío de documentos por referencia es la opción de uso obligatoria para la creación de peticiones. Seleccionando la primera opción desde el menú de la aplicación se lanzará dicho test. En este ejemplo se crea una petición y se envía a los destinatarios indicados en el properties. El tipo de firma será también el definido en el properties. Cabe destacar que el método para adjuntar el fichero en este caso de uso será el envío por referencia, este método de envío se explicará más detalladamente en este mismo documento.

En este ejemplo no se explica cómo añadir un remitente a la petición ya que en las futuras integraciones solo podrán realizarse sobre aquellas aplicaciones que tengan autorización previa. Si se considera que la aplicación a integrar debe tener remitente debéis de solicitar autorización en NAOS.

El primer paso en la creación de la petición es añadirle una línea de firma que contenga a los firmantes que se definieron anteriormente. Los firmantes han de añadirse a las líneas de firma a través de listas por lo que se instanciará una lista de firmantes para añadirlas a la línea de firma y ésta será añadida también a una lista de líneas de firma. Si no se especifica el tipo de línea de firma, ésta toma por defecto el valor de firma. En este caso se coge el tipo de firma de las propiedades.

```
            SignLineList signLineList = new SignLineList();



            String firmante1 =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_FIRMANTE_1);

            String firmante2 =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_FIRMANTE_2);

            String firmante3 =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_FIRMANTE_3);



            assertNotNull(firmante1);



            if(StringUtils.isNotEmpty(firmante1)) {



                  String tipoFirma1 =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_FIRMA_FIRMANTE_1);
```

```
                    SignLine signLine = this.createSignLine(firmante1.trim(),
tipoFirma1);

                    signLineList.getSignLine().add(signLine);

            }


            if(StringUtils.isNotEmpty(firmante2)) {


                    String tipoFirma2 =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_FIRMA_FIRMANTE_2);


                    SignLine signLine = this.createSignLine(firmante2.trim(),
tipoFirma2);

                    signLineList.getSignLine().add(signLine);

            }


            if(StringUtils.isNotEmpty(firmante3)) {


                    String tipoFirma3 =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_FIRMA_FIRMANTE_3);


                    SignLine signLine = this.createSignLine(firmante3.trim(),
tipoFirma3);

                    signLineList.getSignLine().add(signLine);

            }


      private SignLine createSignLine(String firmante, String tipo) {


            User userSigner = new User();

            userSigner.setIdentifier(firmante);
```

```
            Signer signer = new Signer();

            signer.setUserJob(userSigner);


            Log.info("Creating sign line");

            SignLine signLine = new SignLine();

            signLine.setType(tipo);

            SignerList signerList = new SignerList();

            signerList.getSigner().add(signer);

            signLine.setSignerList(signerList);


            return signLine;


      }
```

Tras crear la línea de firma se procederá a indicar el tipo de documento que se añadirá a la petición. Los tipos de documentos válidos se encuentran definidos en el Anexo I del presente documento.

Y se adjuntará el documento a la petición a través del método de Envío por Referencia. Mediante el método *setSign* se especifica que el documento será firmable o anexo. Los documentos de la petición también se almacenarán en una lista de los mismos para la creación de la petición.

```
String path =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_PATH_ARCHIVO_1);


            String nombre = null;

            String csv = null;

            String tipoDocumento = null;
```

```
            String mimeType = null;

            boolean signable = false;


            if(StringUtils.isNotBlank(path)) {

                  nombre =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_NOMBRE_ARCHIVO_1);


                  assertNotNull(nombre);


                  csv =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_CSV_ARCHIVO_1);


                  assertNotNull(csv);


                  tipoDocumento =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_DOCUMENTO_ARCHIVO_1);


                  assertNotNull(tipoDocumento);


                  mimeType =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_MIMETYPE_ARCHIVO_1);


                  assertNotNull(mimeType);


                  signable =
Boolean.valueOf(properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_FIRMABLE_ARC
HIVO_1));


                  assertNotNull(signable);
```

```
                    this.addDocument(documentReferenceList, nombre, path, csv,
tipoDocumento, mimeType, signable);

             }



             path =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_PATH_ARCHIVO_2);



             if(StringUtils.isNotBlank(path)) {

                    nombre =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_NOMBRE_ARCHIVO_2);

                    csv =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_CSV_ARCHIVO_2);

                    tipoDocumento =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_DOCUMENTO_ARCHIVO_2);

                    mimeType =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_MIMETYPE_ARCHIVO_2);

                    signable =
Boolean.valueOf(properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_FIRMABLE_ARC
HIVO_2));



                    this.addDocument(documentReferenceList, nombre, path, csv,
tipoDocumento, mimeType, signable);

             }



             path =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_PATH_ARCHIVO_3);



             if(StringUtils.isNotBlank(path)) {

                    nombre =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_NOMBRE_ARCHIVO_3);

                    csv =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_CSV_ARCHIVO_3);

                    tipoDocumento =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_DOCUMENTO_ARCHIVO_3);
```

```
                    mimeType =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_MIMETYPE_ARCHIVO_3);

                    signable =
Boolean.valueOf(properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_FIRMABLE_ARC
HIVO_3));



                    this.addDocument(documentReferenceList, nombre, path, csv,
tipoDocumento, mimeType, signable);

             }

      private void addDocument(DocumentReferenceList documentReferenceList,
String nombre, String path, String csv,

                    String tipoDocumento, String mimeType, boolean signable) {



             DocumentType docType = new DocumentType();

             docType.setIdentifier(tipoDocumento);



             DocumentReference doc = new DocumentReference();

             doc.setSign(signable);

             doc.setDocumentType(docType);

             doc.setMime(mimeType);

             doc.setName(nombre);



             File file = new File(path);

             log.debug("File exists: " + file.exists());

             byte[] bytes = null;

             try {

                    FileInputStream fis = new FileInputStream(file);

                ByteArrayOutputStream bos = new ByteArrayOutputStream();

                byte[] buf = new byte[1024];

                for (int readNum; (readNum = fis.read(buf)) != -1;) {
```

```
                bos.write(buf, 0, readNum);
            }
            bytes = bos.toByteArray();
            fis.close();
        } catch (Exception ex) {

        }

        String hash = createHash(bytes, "SHA-256");

        doc.setHashToSign(hash);
        doc.setHashAlgorithm("SHA256");
        doc.setSize(new Long(file.length()).intValue());
        doc.setCsv(csv);
        doc.setSize(new Long(file.length()).intValue());

        documentReferenceList.getDocumentReferenceList().add(doc);
    }
    private String createHash(byte[] bytes, String digestAlg) {

        String hashStr = "";

        try {
            MessageDigest md = MessageDigest.getInstance(digestAlg);

            md.update(bytes);
            byte[] hash = md.digest();
```

```
                hashStr = new String(Hex.encodeHex(hash));
            } catch (NoSuchAlgorithmException ex) {
                log.error("Error in digest algorithm:" + ex);
            }
            return hashStr;
        }
```

El siguiente paso es instanciar un objeto RequestByReference y establecer sus propiedades con los datos obtenidos anteriormente además de indicar la aplicación a la que irá asociada dicha petición, el asunto, texto, referencia y tipo de firma de la misma.

```
            String signType =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TIPO_FIRMA);
            assertNotNull(signType);

            String application = properties.get(Constants.APLICACION_CODIGO);
            assertNotNull(application);

            String subject =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_ASUNTO);
            assertNotNull(subject);

            String reference =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_REFRENCIA);
            assertNotNull(reference);

            String text =
properties.get(Constants.CREATE_REQUEST_BY_REFERENCE_TEXTO);
            assertNotNull(text);

            // Create request
```

```
            RequestByReference req = new RequestByReference();

            req.setApplication(application);

            req.setDocumentReferenceList(documentReferenceList);

            req.setReference(reference);

            req.setSignLineList(signLineList);

            req.setSignType(signType);

            req.setSubject(subject);

            req.setText(text);
```

Una vez creado el objeto, se creará la petición a través del método *createRequestByReference* del servicio de modificación pasándole el objeto *RequestByReference*, devolviendo el identificador de la petición creada. Al estar obligatoriamente activado el envío automático de peticiones en las nuevas aplicaciones no es necesaria la llamada al método sendRequest de tal forma que la creación de la petición es totalmente transaccional.

```
            String requestHash =
modifyServiceClient.createRequestByReference(req);



            log.info("Petición creada con hash {}", requestHash);
```

Tras este proceso, se habrá creado y enviado la petición. En caso de producirse algún error, el servicio lanzará una *PfirmaException* con los detalles de dicho error.

Opción 2 – Eliminar petición – deleteRequest

Una vez creada una petición podemos proceder a eliminarla a través del servicio *deleteRequest* pasándole como parámetro el identificador de dicha petición de firma.

```
      public void deleteRequest() throws PfirmaException {



            ModifyService modifyServiceClient = this.getModifyServiceClient();
```

```
modifyServiceClient.deleteRequest(properties.get(Constants.DELETE_REQUEST_IDENTIFICADOR));

            System.out.println("La petición con identificador " + properties.get(Constants.DELETE_REQUEST_IDENTIFICADOR) + " se ha eliminado correctamente.");

      }
```

Si la petición se ha completado por parte de los usuarios retornará un error y no podrá eliminarse.

En el caso de si las firmas/VBs de la petición se han comenzado pero no se han completado, la petición quedará en estado “ABORTADO” y aparecerá en la bandeja de terminadas de los firmantes en este estado.

Tras este proceso, la petición quedará en estado “ELIMINADO” y no podrán ser consutadas ni firmadas por los usuarios. En caso de producirse algún error, el servicio lanzará una PfirmaException con los detalles de dicho error.

Opción 3 – Consultar peticiones – queryRequestList

Al seleccionar esta opción se podrán consultar masivamente la información sobre varias peticiones a la vez. Se recomienda no consultar más de 100 peticiones a la vez ya que podría ocasionar retardo en la respuesta.

```
            RequestIdList requestIdList = new RequestIdList();

            String id1 = properties.get(Constants.QUERY_REQUEST_LIST_IDENTIFICADOR_1);

            String id2 = properties.get(Constants.QUERY_REQUEST_LIST_IDENTIFICADOR_2);

            String id3 = properties.get(Constants.QUERY_REQUEST_LIST_IDENTIFICADOR_3);
```

```
		if(StringUtils.isNotBlank(id1)) {
			requestIdList.getRequestId().add(id1);
		}

		if(StringUtils.isNotBlank(id2)) {
			requestIdList.getRequestId().add(id2);
		}

		if(StringUtils.isNotBlank(id3)) {
			requestIdList.getRequestId().add(id3);
		}

		RequestList requestList =
queryServiceClient.queryRequestList(requestIdList);

		List<RequestObjectResponse> response =
requestList.getRequestObjectResponseList();

		for (RequestObjectResponse requestObjectResponse : response) {

			System.out.println("ESTADO " +
requestObjectResponse.getEstado());

				System.out.println("Application " +
request.getApplication());
		System.out.println("Identifier " + request.getIdentifier());
		System.out.println("Reference " + request.getReference());
```

```
            System.out.println("SignType " + request.getSignType());
            System.out.println("Subject " + request.getSubject());
            System.out.println("Text " + request.getText());
            System.out.println("HashCode " + request.hashCode());
            System.out.println("Fentry " + request.getFentry());
            System.out.println("Fexpiration " + request.getFexpiration());
            System.out.println("Fstart " + request.getFstart());

            SignLineList  signLineList = request.getSignLineList();

            for(SignLine signLine: signLineList.getSignLine()) {

                  System.out.println("SignLine type " + signLine.getType());

                  SignerList signerList = signLine.getSignerList();

                  for (Signer signer : signerList.getSigner()) {

                        System.out.println("SignLine");

                        System.out.println("      Fstate " +
signer.getFstate());
                        System.out.println("      State " +
signer.getState().getIdentifier());
                        System.out.println("      User " +
signer.getUserJob().getIdentifier());
                  }
            }
```

```
            DocumentList documentList = request.getDocumentList();


            for (Document document : documentList.getDocument()) {


                  System.out.println("Document");


                  System.out.println("      CSV " + document.getCSV());

                  System.out.println("      Type " +
document.getDocumentType().getIdentifier());

                  System.out.println("      Hash " + document.getHash());

                  System.out.println("      HashAlgoritm " +
document.getHashAlg());

                  System.out.println("      Identifier " +
document.getIdentifier());

                  System.out.println("      Mime " + document.getMime());

                  System.out.println("      Name " + document.getName());

                  System.out.println("      Type " + document.getType());

            }


            }
```

Opción 4 – Consultar petición – queryRequest

Al seleccionar esta opción se podrán consultar sobre una única petición. Se recomienda el uso de queryRequestList ya que es más eficiente consultar varias peticiones que ir de una en una.

```
QueryService queryServiceClient = this.getQueryServiceClient();
```

```
            Request                                request                                =
queryServiceClient.queryRequest(properties.get(Constants.QUERY_REQUEST_IDENTIFICA
DOR));


             System.out.println("Application " + request.getApplication());

            System.out.println("Identifier " + request.getIdentifier());

            System.out.println("Reference " + request.getReference());

            System.out.println("SignType " + request.getSignType());

            System.out.println("Subject " + request.getSubject());

            System.out.println("Text " + request.getText());

            System.out.println("HashCode " + request.hashCode());

            System.out.println("Fentry " + request.getFentry());

            System.out.println("Fexpiration " + request.getFexpiration());

            System.out.println("Fstart " + request.getFstart());


            SignLineList  signLineList = request.getSignLineList();


            for(SignLine signLine: signLineList.getSignLine()) {


                  System.out.println("SignLine type " + signLine.getType());


                  SignerList signerList = signLine.getSignerList();


                  for (Signer signer : signerList.getSigner()) {


                        System.out.println("SignLine");


                        System.out.println("      Fstate " +
signer.getFstate());
```

```
                        System.out.println("      State " +
signer.getState().getIdentifier());

                        System.out.println("      User " +
signer.getUserJob().getIdentifier());

                    }

            }


            DocumentList documentList = request.getDocumentList();


            for (Document document : documentList.getDocument()) {


                    System.out.println("Document");


                    System.out.println("      CSV " + document.getCSV());

                    System.out.println("      Type " +
document.getDocumentType().getIdentifier());

                    System.out.println("      Hash " + document.getHash());

                    System.out.println("      HashAlgoritm " +
document.getHashAlg());

                    System.out.println("      Identifier " +
document.getIdentifier());

                    System.out.println("      Mime " + document.getMime());

                    System.out.println("      Name " + document.getName());

                    System.out.println("      Type " + document.getType());

            }
```

Opción 5 – Consultar histórico peticiones – queryHistoricList

Port@firmas ofrece un servicio de consulta masiva de estado, a través del cual se podrán consultar los históricos de numerosas peticiones en una sola invocación al webservice, haciendo uso del método queryHistoricList el cual nos devolverá un listado de objetos

HistoricList es decir, un objeto que será una lista que contendrá todos los históricos de las peticiones consultadas (HistoricListList).

Lo ideal para que una aplicación se entere de que una petición ha sido finalizada es utilizar el método PUT de ENIDOC 2.0 (más información en el punto 6), sin obviar el uso de este método por si haya podido haber algún problema en la comunicación con este servicio.

Aunque su aplicación no esté integrada con ENIDOC 2.0 puede implementar el uso de este método. La URL de acceso y credenciales deben de estar configurada en la aplicación en la zona de Administración en los siguientes campos:

La llamada a este servicio no debe realizarse de forma abusiva (por ejemplo realizar llamadas cada 5 minutos). Se recomienda llamar a este método para comprobar el estado de la petición cada 30 minutos.

```
            public void queryHistoricList() throws PfirmaException {


            QueryService queryServiceClient = this.getQueryServiceClient();


            RequestIdList requestIdList = new RequestIdList();


            String id1 =
properties.get(Constants.QUERY_HISTORIC_LIST_IDENTIFICADOR_1);
            String id2 =
properties.get(Constants.QUERY_HISTORIC_LIST_IDENTIFICADOR_2);
            String id3 =
properties.get(Constants.QUERY_HISTORIC_LIST_IDENTIFICADOR_3);


            requestIdList.getRequestId().add(id1);
```

```
            requestIdList.getRequestId().add(id2);

            requestIdList.getRequestId().add(id3);



            HistoricListList historicList =
queryServiceClient.queryHistoricList(requestIdList);



            for (HistoricObjectResponse historicObjectResponse :
historicList.getHistoricObjectResponseList()) {



                  if (historicObjectResponse.getHistoricList() == null) {

                        System.out.println("No existe histórico para la
petición " + historicObjectResponse.getRequestId());

                  }else {

                        System.out.println("Petición "+
historicObjectResponse.getRequestId());

                        for (Historic historic :
historicObjectResponse.getHistoricList().getHistoric()) {

                              this.printHistoric(historic);

                        }

                  }

            }

      }



      private void printHistoric(Historic historic) {



            System.out.println("Histórico");



            System.out.println("      Comment " + historic.getComment());

            System.out.println("      Fstate " + historic.getFstate());

            System.out.println("      State " +
```

```
historic.getState().getIdentifier());

            System.out.println("      Text " + historic.getText());

            System.out.println("      User identifier " +
historic.getUserJob().getIdentifier());

            System.out.println(" ");

      }
```

Opción 6 – Consultar histórico petición – queryHistoric

Método que se utiliza para consultar el estado de una petición. Se recomienda el uso del método queryHistoricList para consultar a la vez el estado de varias peticiones.

Se obtiene el objeto HistoricList que contiene la lista de entradas en el histórico de la petición a través del servicio queryHistoric pasándole su hash y se itera sobre ellas. De cada objeto Historic se muestra el texto del mismo por consola.

```
                  public void queryHistoric() throws PfirmaException {


            QueryService queryServiceClient = this.getQueryServiceClient();


            String idPeticion =
properties.get(Constants.QUERY_HISTORIC_IDENTIFICADOR);


            HistoricList historicList =
queryServiceClient.queryHistoric(idPeticion);


            if(historicList == null) {

                  System.out.println("No existe histórico para la petición " +
idPeticion);

            }else {

                  List<Historic> historicL = historicList.getHistoric();


                  for (Historic historic : historicL) {
```

```
                        this.printHistoric(historic);

                  }

            }

      }


      private void printHistoric(Historic historic) {


            System.out.println("Histórico");



            System.out.println("      Comment " + historic.getComment());

            System.out.println("      Fstate " + historic.getFstate());

            System.out.println("      State " +
historic.getState().getIdentifier());

            System.out.println("      Text " + historic.getText());

            System.out.println("      User identifier " +
historic.getUserJob().getIdentifier());

            System.out.println(" ");

      }
```

Opción 7 – Descargar firma – downloadSign

Desde está opción se realiza la descarga de la firma de la última firma documento a partir de su identificador. Debe de invocarse una vez se tenga constancia de que la petición ha sido finalizada por todos los firmantes.

```
byte [] signContent =
queryServiceClient.downloadSign(properties.get(Constants.DOWNLOAD_SIGN_IDENTIFICA
DOR));

            if(signContent == null) {

                  System.out.println("NO se ha encontrado firma para el
documento con identificador "
```

```
                                        +
properties.get(Constants.DOWNLOAD_SIGN_IDENTIFICADOR));

            }else {

                  System.out.println("Firma descargada correctamente para el
documento con identificador "

                                        +
properties.get(Constants.DOWNLOAD_SIGN_IDENTIFICADOR));

            }
```

Opción 8 – Consultar CSV – queryCsv

La consulta del código seguro de verificación de un documento se puede realizar a través del siguiente método pasándole como parámetro el identificador del documento.

```
String csv =
queryServiceClient.queryCsv(properties.get(Constants.QUERY_CSV_IDENTIFICADOR));



System.out.println("El CSV para el documento con identificador "

                           +
properties.get(Constants.QUERY_COMMENTS_IDENTIFICADOR) + " es " + csv);
```

Opción 9 – Consultar comentarios petición – queryComments

Seleccionando la novena opción en el menú de la aplicación se lanzará dicho test. En este ejemplo se consultan los comentarios de una petición a partir de su identificador. A continuación se muestran por consola los datos obtenidos.

Una vez generado el cliente de consulta, se llama al servicio de consulta de comentarios queryComments de una petición pasándole su identificador y una vez obtenido el objeto CommentList se itera sobre la lista de comentarios contenida en caso de que ésta no venga vacía. De cada objeto Comment obtenido se muestran la fecha, el identificador del usuario que comentó y el texto del comentario.

```
QueryService queryServiceClient = this.getQueryServiceClient();
```

```
CommentList commentList =
queryServiceClient.queryComments(properties.get(Constants.QUERY_COMMENTS_IDENTIFI
CADOR));

if (commentList != null && commentList.getComment() != null

                        && !commentList.getComment().isEmpty()) {

      log.info("Comentarios para la petición con identificador"

                                  +
properties.get(Constants.QUERY_COMMENTS_IDENTIFICADOR));

                  for (Comment comment : commentList.getComment()) {

                        System.out.println(comment.getFcomment() + " - "

                                    + comment.getUser().getIdentifier() + " -
"

                                    + comment.getText());

                  }

            } else {

                  System.out.println("No hay comentarios para la petición con
identificador "

                                  +
properties.get(Constants.QUERY_COMMENTS_IDENTIFICADOR));

            }
```

2.3.3.5 Envío automático de peticiones

Como novedad en esta versión existe la posibilidad de que se realice el envío automático de peticiones. Quiere decir que cuando se cree una petición con el método “createRequest”, no es necesario llamar al método “sendRequest”. Para este cambio debe de modificarse la configuración de la aplicación por parte del usuario Administrador. Todas las aplicaciones nuevas que se integren deben utilizar el envío automático de peticiones.

2.3.4 Generación de cliente de WS con Apache CXF

En la aplicación de ejemplo de integración del apartado anterior se facilita un cliente de servicio web de Port@firmas generado con Apache CXF (**pfirmav2CXFClient.jar**).

Normalmente en integraciones con Port@firmas se suele utilizar este jar.

Para utilizar este jar, es necesario que incluya en su pom.xml la siguiente dependencia:

```
<dependency>
	<groupId>es.juntadeandalucia.pfirma</groupId>
	<artifactId>pfirmav2CXFClient</artifactId>
	<version>3.4.10</version>
</dependency>
```

Esta librería se encuentra en el repositorio Maven de SCAE en la siguiente URL:

http://deploytools01.chap.junta-andalucia.es/artifactory/ja-artifacts-deploy/

Se debe de tener en cuenta que a partir de la versión 3.4.10 de esta librería desde ella no se pueden realizar llamadas a determinados servicios los cuales no se permiten la integración para nuevas aplicaciones o realizan la integración de manera ineficiente. Si su aplicación ya está integrada no debe de utilizar la versión 3.4.10 de la librería.

Estos servicios son los siguientes:

Servicios de ModifyService

- CreateRequest.

Para nuevas integraciones el uso de este servicio no está permitido debe de utilizarse el método createRequestByReference.

- DeleteDocument

A modo de evitar modificaciones a posteriori de la creación de la petición se evita el uso de este método. Si al crear una petición hay algún error se podrá llamar al método deleteRequest y posteriormente crear una nueva petición.

- DeleteSigners

La explicación es la misma que el método anterior.

- InsertDocument

La explicación es la misma que en métodos anteriores.

- InsertSigners

La explicación es la misma que en métodos anteriores.

- UpdateRequest

La explicación es la misma que en métodos anteriores.

- SendRequest

Al ser obligatorio el envío automático de peticiones este método es innecesario ya que al crear la petición a la vez se envía haciendo la creación de la petición transaccional.

Servicios de QueryService

- DownloadDocument.

Ya que todas las aplicaciones tienen custodia por referencia no tiene sentido el uso de este método ya que Portafirmas tendría que irse al repositorio ENIDOC de la aplicación origen a obtener el documento.

- DownloadReport

Lo mismo ocurre con este método. Las aplicaciones que se integren deben de generar su propio informe de firma con la librería firma-report

- QueryDocumentTypes
  Los tipos de documentos no cambian y pueden encontrarse en este documento en el anexo I.
- QueryJobs

El uso de cargos no está permitido en Port@firmas Centralizado por lo que se descarta su uso en la librería

- QueryStates

Al igual que con los tipos de documentos los estados no varían y pueden encontrarse en este documento en el anexo II

- QueryUsers

El uso de este método no es necesario. Si un usuario no existe, a la hora de crear una petición el servicio indicará que el usuario X no existe.

El uso de esta librería no es obligatorio. La aplicación que se integre puede crear su propio cliente a partir del wsdl especificado.

Si por cualquier motivo se quiere generar los clientes desde el wsdl de la aplicación hay que tener en cuenta el siguiente aspecto.

Al crear los clientes debemos hacerlo con la línea:

```
wsdl2java -b bindings.xml
```

siendo **bindings.xml**:

```
<jaxb:bindings version="2.1"
      xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
      xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <jaxb:globalBindings generateElementProperty="false" />
</jaxb:bindings>
```

3 ENVÍO DE DOCUMENTOS POR REFERENCIA

Para nuevas integraciones los documentos se alojarán en repositorios externos a Port@firmas mediante la creación de peticiones con envío de documentos por referencia. Dichos repositorios deben ofrecer a Port@firmas un protocolo de acceso a los documentos que custodian, denominado ENIDOCWS, el cual se adecua y alinea con los requisitos definidos por el Esquema Nacional de Interoperabilidad.

Estas aplicaciones si quieren generar su propio informe de firma se aconseja utilizar la librería firma-report la cual genera el informe de firma a partir de un documento PDF original, los firmantes y fecha.

En la zona de administración de Port@firmas y para cada aplicación, se definirá su sistema de custodia:

- **Custodiado por Port@firmas**: no se admitirá el envío de documentos por referencia ya que todos los documentos deberán estar alojados en Port@firmas (aplicaciones antiguamente integradas)

- **Custodiado en repositorios externos**: solo se admitirá el envío de documentos por referencia. En este caso será necesario configurar adicionalmente la URL del servicio ENIDOCWS de acceso a los documentos custodiados por el repositorio externo, así como el usuario y la contraseña con los que Port@firmas se autenticará ante este servicio.

A través de los servicios web es posible crear peticiones con documentos alojado en este tipo de repositorio a través del método "**createRequestByReference**" publicado en el servicio de modificación perteneciente al servicio web v2.

Para invocar este método es necesario construir una lista de documentos, donde cada uno de ellos se genera a partir de:

- **Hash a firmar**: lo debe aportar la aplicación que envíe la petición a Port@firmas. Debe de ser codificado en hexadecimal y se genera con el algoritmo de hash a partir del contenido del documento.

- **Algoritmo de Hash**: algoritmo usado para generar el anterior Hash. Los algoritmos permitidos son los configurados en la administración de Portafirmas. En el Portafirmas centralizado pronto se dejará de permitir el algoritmo SHA1, por lo que es recomendable empezar a utilizar SHA256 y obligatorio en las nuevas integraciones.

- **Tamaño del fichero**

- **CSV del Documento**: debe corresponder con el Código de Verificación que se usará para la posterior verificación del documento. Éste deberá ser interpretable por la Herramienta Centralizada de Verificación (HCV):

  <5_CHARS_ID_REPOSITORIO><25_DOCUMENT_CSV>

  - 5 caracteres que identifiquen al repositorio en el que se encuentre el documento
  - 25 caracteres que identifiquen al documento

  Puede consultar más información acerca del **sistema HCV** en:

  https://intranet.ada.junta-andalucia.es/intranet-ada/nuestro-trabajo/servicios/hcv

- **Datos de la referencia del documento**:

PROTOCOLO	PARÁMETROS DE REFERENCIA
ENIDOCWS	No es necesario indicar ningún parámetro extra. Se usará el **CSV indicado** como identificador del Documento.

En cualquier caso, si a través del cliente webservice se indica un protocolo incorrecto o no se indican todos los parámetros necesarios para la localización del fichero, Port@firmas devolverá un error identificando de qué error se trata.

⚠ Para nuevas integraciones es obligatorio trabajar con Documentos ENI, los cuales están orientados a la Interoperabilidad de los Sistemas de información.

4 BUENAS PRÁCTICAS EN LA INTEGRACIÓN DE APLICACIONES

En este apartado se resumen varias recomendaciones a hacer al realizar la integración de una aplicación.

- Evitar establecer remitentes en las peticiones de firma ya que obligan a dar de alta en Port@firmas usuarios que probablemente nunca tengan que hacer uso de la herramienta. En lugar de ello, si fuera necesario que el destinatario de la petición identifique a la persona que da origen a la petición, puede incorporarse esta información en el campo de observaciones.

- Evitar la consulta previa de los usuarios que intervienen en una petición de firma antes de crearla. En lugar de ello, es preferible crear la petición de firma asumiendo que todas las personas existen y solo si se produce un error, entonces proceder a verificar si su causa no es la ausencia en Port@firmas de alguno de ellos.

- Evitar la consulta del listado completo de usuarios en la implantación ya que es una operación que será discontinuada.

- Establecer una fecha de caducidad a las peticiones y prever en la lógica de la aplicación la posibilidad de que las peticiones que se remiten no sean firmadas ni rechazadas en el plazo establecido.

- Actualizar el estado de una petición en la aplicación origen consultado su estado y no mediante el uso de las acciones de retorno ya que este último puede dar lugar a errores de comunicación y perderse la actualización.

- Autenticarse en Port@firmas tanto los servicios de “QueryService” como “ModifyService” con usuario y contraseña.

- Uso de la última versión de la librería firma-report para la generación de sus propios informes de firma, la cual genera el informe de firma a partir de un documento PDF original, los firmantes y fecha.

5 CRITERIOS OBLIGATORIOS PARA LA INTEGRACIÓN DE NUEVAS APLICACIONES CON PORT@FIRMAS CORPORATIVO

- Custodia de documentos en repositorio remoto no en Port@firmas. Con preferencia de uso de ENIDOC 2.0, más información en el punto 6**.**
- Autenticarse en Port@firmas tanto los servicios de "QueryService" como "ModifyService" con usuario y contraseña.
- Uso de algoritmo de SHA256.
- Identificadores de las peticiones y documentos (campo C_HASH) deben tener 20 caracteres.

6 INTEGRACIÓN CON ENIDOC 2.0

Para la integración con ENIDOC 2.0 la aplicación integradora deberá cumplir el contrato especificado en la siguiente URL:

https://cancanaprun1.chap.junta-andalucia.es/enidoc-svc/swagger-ui/index.html?configUrl=/enidoc-svc/v3/api-docs/swagger-config

La aplicación integradora deberá implementar los siguientes servicios:

- PUT /rest/eni/v2/enidoc/{csv}/signature

Añade una firma electrónica a un documento electrónico identificado a partir de su código seguro de verificación

A este servicio llamaría Portafirmas cuando se han firmado la petición por todos los firmantes para que su aplicación custodie la firma y no sea necesario obtenerla llamando al servicio downloadSign de los servicios web. En caso de que un usuario haya rechazado la petición devolvería la firma vacía.

En caso de que no interese esta funcionalidad se implimentaria devolviendo un 200 sin hacer nada en su aplicación.

GET /rest/eni/v2/enidoc/{csv}/content

Recupera el contenido de un documento electrónico a partir de su código seguro de verificación.

Aclarar que el CSV se envía codificado en Base64.

GET /rest/eni/v2/enidoc/{csv} Recupera un documento electrónico a partir de su código seguro de verificación

En este servicio para Portafirmas solo necesita recuperar el documento ENI

También el CSV se envía codificado en Base64.

7 GLOSARIO

Término	Descripción
Apache CXF	Framework de servicios web.
MAVEN	Herramienta para administración de proyectos.
JUnit	Framework de pruebas unitarias para clases Java.
Eclipse	Entorno de desarrollo integrado (IDE)
ENIDOC - NTI	Norma Técnica de Interoperabilidad
HCV	Herramienta Centralizada de Verificación

8 ANEXO I. TIPOS DE DOCUMENTOS

Se enumera a continuación los posibles valores que puede adoptar el metadato "Tipo documental" que todo documento electrónico ha de tener vinculado conforme a lo establecido en la Norma Técnica de Interoperabilidad de Documento Electrónico.

Término	Descripción
TD01	Resolución
TD02	Acuerdo
TD03	Contrato
TD04	Convenio
TD05	Declaración
TD06	Comunicación
TD07	Notificación
TD08	Publicación
TD09	Acuse de recibo
TD10	Acta
TD11	Certificado
TD12	Diligencia
TD13	Informe
TD99	Otros

La versión 3.5.X de Port@firmas requiere que las peticiones creadas indiquen un valor de entre los anteriores al establecer el tipo de cada uno de los documentos vinculados a una petición de firma. Por motivos exclusivamente de compatibilidad se aceptan adicionalmente los valores configurados en la administración de la herramienta, si bien las aplicaciones que hagan uso de estos otros valores configurados deben abandonar su uso y adaptarse a citada NTI de Documento Electrónico.

9 ANEXO 2. ESTADOS DE UNA PETICIÓN POR USUARIO

Estado	Descripción
ABORTADO	La petición ha sido abortada desde la petición de origen
DEVUELTO	El usuario ha rechazado o devuelto la petición.
EN ESPERA	El usuario está a la espera de que otro usuario o usuarios firma la petición previamente.
FIRMADO	La petición ha sido firmada por el usuario
LEIDO	La petición ha sido leída por el usuario
NUEVO	La petición todavía no ha sido leída por el usuario y a la espera de firmarse por este.
VISTOBUENO	El usuario ha dado el visto bueno a la petición.
ELIMINADA	La petición ha sido eliminada y no podrá consultarse ni firmarse

10 ¿Alguna duda?

Si una vez consultado el documento tiene alguna duda o problema a la hora de realizar la integración puede ponernos un tique en NAOS: https://naossuite.juntadeandalucia.es/autogestion donde atenderemos su duda o problema en el menor tiempo posible.