Kit de integración de firma con FIRe

Manual del integrador

Versión: 2.3

CONTROL DE VERSIONES			
Título		Manual del integrador	
Autor		Secretaría General de Administración Digital Ministerio de Hacienda y Función Pública	
Fecha versión 2.3		11 de ocubre de 2018	
Versión	Fecha	Responsable	Cambios introducidos
1.0	9-02-2016	DTIC	Creación del documento
1.0 Rev 1	17-02-2016	DTIC	Adaptación al cambio de nombre a Clave Firma
1.0 Rev 2	24-05-2016	DTIC	Actualización del cliente Java y PHP
1.0 Rev 3	30-05-2016	DTIC	Se amplía la información de todos los clientes.
1.0 Rev 4	15-07-2016	DTIC	Actualización de los clientes PHP y Java
1.0 Rev 5	30-08-2016	DTIC	Actualización de las dependencias del cliente Java
1.0 Rev 6	18-01-2017	SGAD	Actualización del cliente .NET
1.1	7-02-2017	SGAD	Mejoras generales en el API
1.1 Rev 1	4-04-2017	SGAD	Se amplía la información sobre la autenticación de aplicaciones.
2.0	12-05-2017	SGAD	Se actualiza la información a la de FIRe
2.1	10-10-2017	SGAD	Generación de firmas PKCS#1, el uso de clases gestoras de documentos, nuevo API para la coexistencia de aplicaciones con configuración propia, códigos de error y correcciones varias.
2.2	16-05-2018	SGAD	Sistema multiproveedor, se integra el conector de la FNMT, mejoras funcionales y configuración del Cliente @firma.
2.3	11-10-2018	SGAD	Sistema cifrado contraseñas, nuevo sistema de logs

ÍNDICE

1. OBJETO DEL DOCUMENTO

El presente manual detalla el API del componente distribuido de FIRe para su integración en aplicaciones que necesiten firma de ciudadanos, ya sea mediante certificado local o mediante alguno de los proveedores soportados de firma en la nube.

2. INTRODUCCIÓN

FIRe es un sistema para la generación de firmas electrónicas con certificado de usuario. Las aplicaciones web que deseen integrar la firma electrónica de datos como parte de su flujo de operación pueden utilizar FIRe para tal fin.

FIRe permite el uso tanto de certificados locales del usuario, como el uso de los certificados de proveedores de firma en la nube, como es el caso de Cl@ve Firma. Gracias a eso, al integrar el API de FIRe en una aplicación, se consigue que el usuario pueda usar certificados locales y remotos sin necesidad de crear un flujo de trabajo distinto para cada uno de ellos.

FIRe se componen principalmente de un componente "centralizado" servidor encargado de la firma electrónica de documentos con certificados de usuario y un API "distribuido" para la integración de aplicaciones con ese componente centralizado. Las aplicaciones que deseen utilizar las funcionalidades de firma de FIRe sólo tendrán integrar que utilizar este API.

Las funcionalidades de firma ofrecidas por el API de FIRe son:

- Firma electrónica individual:
  - Firma o multifirma electrónica de un documento
- Firma de lotes
  - Firma o multifirma de múltiples documentos simultáneamente.

Para que el componente central atienda las peticiones de nuestra aplicación, esta debe haberse registrado previamente. Este registro puede hacerlo un administrador a través del módulo de administración de FIRe, mediante el cual dará de alta la nueva aplicación, establecerá el certificado con el que deberá autenticarse y obtendrá como resultado el código alfanumérico que deberá utilizar el componente distribuido como identificador de aplicación (`AppId`). A partir de entonces, podrá realizar peticiones a FIRe utilizando este certificado e identificador de aplicación.

Si queremos que nuestros usuarios, además de poder realizar firmas con certificado local, puedan firmar con certificados en la nube, por regla general, será necesario registrar nuestra aplicación ante cada uno de los proveedores de firma en la nube a los que se quiera tener acceso. Para esto, será necesario contactarlos para que registren la aplicación y configurar en la llamada a FIRe los parámetros que estos hayan indicado. Consulte con el administrador del componente centralizado de FIRe para conocer los proveedores qué tiene dados de alta y como habilitar nuestra aplicación para su uso.

FIRe incorpora por defecto un proveedor de prueba que emula el funcionamiento de Cl@ve Firma y para cuyo uso no es necesario haberse registrado previamente ante ningún proveedor. Este componente suele habilitarse en los entornos distintos al de producción en los que se despliega el componente central de FIRe. Si no puede ver este proveedor en estos entornos, consúltelo con el administrador de FIRe.

Se distribuyen tres implementaciones diferentes del componente distribuido (Java, .NET y PHP) para permitir su integración en las aplicaciones web con tecnologías más comunes.

3. ADECUACIÓN AL ESQUEMA NACIONAL DE SEGURIDAD

FIRe permite el uso de algoritmos de firma no recomendados por la Guía 807 del Esquema Nacional de Seguridad (ENS; editada por el Centro Criptológico Nacional, CCN) vigente en el momento de publicación de este documento. Por lo que queda bajo la responsabilidad de las aplicaciones que hacen uso de FIRe el configurarlo adecuadamente para generar el resultado esperado, válido y adecuado para ese momento y el nivel de seguridad deseado, utilizando para ello algoritmos de la familia SHA-2 tal y como especifica dicha norma para la generación de firmas electrónicas.

Puede consultar la norma vigente desde el siguiente enlace:

https://www.ccn-cert.cni.es/series-ccn-stic/800-guia-esquema-nacional-de-seguridad/513-ccn-stic-807-criptologia-de-empleo-en-el-ens/file.html

4. OPERACIONES DEL COMPONENTE DISTRIBUIDO

4.1. FLUJOS DE OPERACIÓN

FIRe permite a las aplicaciones integrar 2 flujos de operación distintos:

- Firma de un único documento:
    - Siguiendo este flujo de operación una aplicación puede enviar a firmar un documento indicando la configuración de firma que desea utilizar (operación de firma, algoritmo, formato, configuración específica del formato y formato avanzado al que actualizar la firma) y seguidamente obtener la firma resultante.
- Firma de un lote de documentos:
    - Siguiendo este flujo de operación una aplicación puede crear un lote de firma indicando la configuración de firma que desea utilizar (operación de firma, algoritmo, formato, configuración específica del formato y formato avanzado al que actualizar la firma). Seguidamente puede agregar tantos documentos como desee al lote de firma, indicando opcionalmente si debe aplicarse una configuración de firma distinta a la del lote y finalmente puede ordenar la firma de todo el lote. Una vez hecho puede recuperar el resultado de la firma del lote y a continuación puede recuperar cada firma individual.

4.1.1. FIRMA DE UN ÚNICO DOCUMENTO

Para la firma de un único documento una aplicación deberá seguir la siguiente secuencia de pasos:

1. Llamar al método `sign` del componente distribuido.
    - A este método deberemos proporcionarle:
        - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.
        - Identificador de usuario. Comúnmente el DNI. Si se restringe el uso para sólo utilizar certificados locales, puede indicarse cualquier valor.
        - Configuración de firma. Esto es la operación criptográfica, el algoritmo, formato y la configuración del formato. Consulte el apartado Configuración de la operación de firma para más información.
        - Datos a firmar.
        - La configuración del conector con el proveedor de firma. Aquí se debe indicar información como las URL de las páginas o servicios de su aplicación web a las que redirigir al usuario en caso de que la operación tenga éxito o falle, el origen de los certificados, etc. Consulte el apartado Configuración de los proveedores y el componente central para más información.

- Como resultado se obtendrá un objeto con el identificador de la transacción y una URL de redirección.

2. Redirigir al usuario a la URL de redirección obtenida que será la de una página web del componente central.
    - A partir de ese momento la aplicación perderá el control de la navegación del usuario que pasará a navegar por las páginas del componente central y de la GISS hasta completar la operación de firma.
    - Una vez finalizada la operación, el usuario será redirigido automáticamente a la página de éxito o a la página de error de nuestra aplicación, según haya terminado la operación. Estas direcciones son las que debieron definirse a través del parámetro de configuración que indicamos en la llamada al método `sign` del API.
3. En la página resultante, se deberá llamar al método apropiado del API para recoger el resultado de la operación:
    - En caso de ser redirigidos a la página de éxito, se deberá llamar al método `recoverSignResult` proporcionándole:
        - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.
        - Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método `sign`.
        - Formato al que actualizar la firma electrónica por medio de la Plataforma @firma. Consulte el apartado Formato mejorado de firma para más información.

      Este nos devolverá un objeto con la firma resultante de la operación o, en caso de encontrarse un error en el último momento, el error detectado.
    - En caso de ser redirigidos a la página de error, se deberá llamar al método `recoverErrorResult`. Este método nos devolverá un objeto con el error detectado.
4. Una vez tenemos el resultado de la operación, continuaremos con nuestro flujo de trabajo habitual.

4.1.2. FIRMA DE UN LOTE DE DOCUMENTOS

Para la firma de un lote de documentos una aplicación deberá seguir la siguiente secuencia de pasos:

1. Llamar al método `createBatchProcess` del componente distribuido.
    - A este método deberemos proporcionarle:
        - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.
        - Identificador de usuario. Comúnmente el DNI. Si se restringe el uso para sólo utilizar certificados locales, puede indicarse cualquier valor.

- Configuración de firma. Esto es la operación criptográfica, el algoritmo, formato, la configuración del formato y el formato al que mejorar la firma. Consulte el apartado Configuración de la operación de firma para más información.
- Configuración de los conectores con los proveedores de servicios firma. Aquí se debe indicar información como las URL de las páginas o servicios de su aplicación web a las que redirigir al usuario en caso de que la operación tenga éxito o falle, el origen de los certificados, etc. Consulte el apartado Configuración de los proveedores y el componente central para más información.

- Como resultado se obtendrá un objeto con un identificador de transacción.

2. Llamar al método `addDocumentToBatch` para agregar un documento al lote de firma.
   - A este método deberemos proporcionarle, al menos:
     - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.
     - Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método `createBatchProcess`.
     - Identificador de los datos a firmar. Este identificador puede ser cualquier cadena alfanumérica compuesta por caracteres ASCII básicos imprimibles, además de los caracteres guion ('-') y guion bajo ('_'). La firma del documento se devolverá identificada mediante esta misma cadena para que se sepa a qué datos corresponde. No se puede usar un mismo identificador más de una vez en el mismo lote.
     - Datos a firmar.
     - Configuración de los conectores con los proveedores de servicios firma. Consulte el apartado Configuración de los proveedores y el componente central para más información.
   - Opcionalmente, se podría llamar a otra versión del método en el que se incorporase una nueva configuración de firma (a excepción del algoritmo de firma). En caso de usarse esta versión del método, se aplicará la configuración de firma indicada en él en lugar de la indicada durante la creación del lote para la firma de los datos proporcionados. Consulte el apartado Configuración de la operación de firma para más información.
   - Este método no devuelve ningún resultado.
3. Repetir el paso anterior por cada documento que se desee agregar al lote.
4. Llamar al método `signBatch` para iniciar el proceso de firma del lote.
   - A este método deberemos proporcionarle:
     - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.
     - Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método `createBatchProcess`.

- Indicador de si deseamos que, una vez detectado un error en una de las firmas del lote, se detengan las operaciones de firma del lote que no hayan finalizado todavía.

- Como resultado se obtendrá un objeto con el identificador de la transacción (el mismo que se tenía de la operación de crear lote) y una URL de redirección.

5. Redirigir al usuario a la URL de redirección obtenida que será la de una página web del componente central.

   - A partir de ese momento la aplicación perderá el control de la navegación del usuario que pasará a navegar por las páginas del componente central y de la GISS hasta completar la operación de firma.

   - Una vez finalizada la operación, el usuario será redirigido automáticamente a la página de éxito o a la página de error de nuestra aplicación, según haya terminado la operación. Estas direcciones son las que debieron definirse a través del parámetro de configuración que indicamos en la llamada al método `sign` del API.

6. En la página resultante, se deberá llamar al método apropiado del API para recoger el resultado de la operación:

   - En caso de ser redirigidos a la página de éxito, se deberá llamar al método `recoverBatchResult` proporcionándole:

     - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.

     - Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método `createBatchProcess`.

     Este método nos devolverá un objeto con un XML del que se obtiene el listado de identificadores de los documentos agregados al lote, el resultado de la operación y, si falló esa firma concreta, una descripción del error detectado. En caso de encontrarse un error general en el último momento, el objeto se incluirá el error detectado.

   - En caso de ser redirigidos a la página de error, se deberá llamar al método `recoverErrorResult`. Este método nos devolverá un objeto con el error detectado.

7. Una vez sabemos cómo ha terminado cada una de las firmas del lote, podemos llamar al método `recoverBatchSign` para recuperar las firmas del lote que finalizaron correctamente.

   - A este método deberemos proporcionarle:

     - Identificador de nuestra aplicación frente al componente central. Este identificador nos lo deberá pasar el administrador después de dar de alta nuestra aplicación en el componente central.

     - Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método `createBatchProcess`.

- Identificador del documento del que queramos recuperar la firma. Este será alguno de los identificadores que utilizamos al agregar los documentos al lote mediante la llamada al método `addDocumentToBatch` y que hemos confirmado mediante el resultado obtenido por el método `recoverBatchResult` que la firma se generó correctamente.

- La llamada a este método sólo funcionará con los identificadores de los documentos que se firmaron correctamente y sólo la primera vez que se llame para cada uno de esos identificadores.
- No es posible recuperar una firma de un lote, sin haber recuperado previamente el resultado del lote.

8. Una vez hemos recuperar el resultado de todas las firmas que deseábamos obtener, podemos continuar con el flujo de trabajo de nuestra aplicación.

4.2. LISTADO DE OPERACIONES

Aquí se listan todas las operaciones disponibles a través del componente distribuido. Estas operaciones son compartidas por todos los componentes distribuidos de FIRe, independientemente de la implementación (Java, PHP o .NET). Gran parte de las operaciones indicadas a continuación forman parte de los flujos de operación descritos en el apartado anterior:

- Firma de datos:
  - Inicia una operación de firma mediante la carga de los datos y de los parámetros de configuración de una operación de firma. Devuelve el identificador de transacción con el que referenciar a la operación y una URL a la que redirigir al usuario.
  - Tras esta operación el usuario debe ser redirigido por la aplicación que integra el API a la URL proporcionada que estará alojada en el servidor del componente central. Desde esta página podrá seleccionar el origen del certificado de firma (local o remoto), seleccionarlo el certificado y autorizar la operación de firma.
- Recuperación de la firma:
  - Completa la operación de firma, la envía a actualizar si se configuró para ello y la devuelve como resultado de la transacción. Debe recibir el identificador de la transacción de firma.
- Creación de lote de firma:
  - Crea un lote de firma con una configuración firma específica. Devuelve el identificador de transacción con el que referenciar a la operación.
- Agregar documento al lote de firma:
  - Agrega un documento al lote de firma previamente creado y al que se hará referencia mediante su identificador.
  - Existen dos versiones de este método. Una de ellas únicamente carga el documento para que se le aplique la configuración de firma establecida al crear el lote. La otra, establece una configuración de firma específica para ese documento

- Todos los documentos del lote se firmarán con el mismo algoritmo de firma que se será el establecido al crear el lote.

- Firmar lote:
  - Inicia la firma de todos los documentos agregados al lote.
  - Tras esta operación el usuario debe ser redirigido por la aplicación que integra el API a la URL proporcionada que estará alojada en el servidor del componente central. Desde esta página podrá seleccionar el origen del certificado de firma (local o remoto), seleccionarlo el certificado y autorizar la operación de firma.
- Recuperación del resultado del lote:
  - Completa las firmas del lote, las manda a actualizar si se configuraron para ello y devuelve un XML con resultado de la transacción. En el XML resultado aparece el identificador de cada uno de los documentos agregados al lote, como finalizó la firma de cada uno de los documentos y, en caso de error, el motivo.
- Recuperación de una firma de un lote:
  - Descarga la firma de uno de los documentos que finalizase correctamente del lote. Se le debe indicar tanto el identificador de transacción del lote como el identificador del documento.
- Comprobación de progreso de la firma de un lote:
  - Este método permite saber el porcentaje de avance de la firma de un lote. Debe usarse concurrentemente después de llamar al método de recuperación del resultado del lote.
- Recuperación de error:
  - En caso de ser redirigido a una página de error por el componente central, esta función permite descargar el mensaje de error asociado.

5. INTEGRACIÓN DEL COMPONENTE DISTRIBUIDO JAVA

La implementación Java del componente distribuido está desarrollada para la integración de aplicaciones JSP. El componente se distribuye en forma de archivo JAR, al que deben acompañar otros dos (para el soporte de JSON) que son dependencias de este. El JAR del componente distribuido se entrega firmado para asegurar su integridad.

Las bibliotecas son:

- Componente distribuido Java:
  - `fire-client-2.3.jar`
- Dependencias para el soporte de ficheros JSON:
  - `javax.json-api-1.0.jar`
  - `javax.json-1.0.4.jar`
- Biblioteca para la escritura de logs:
  - `slf4j-api-1.7.25.jar`
  - `(Biblioteca puente del sistema de log que se desee)`

Estas bibliotecas deben estar dentro del CLASSPATH de la aplicación que desee realizar las operaciones de firma en la nube.

Todas las llamadas al API se realizan por medio de la clase:

```
es.gob.fire.client.FireClient
```

5.1. CONFIGURACIÓN

Para la configuración del componente distribuido Java se deberá construir el objeto `FireClient` con el identificador de la aplicación y un `Properties` con las propiedades a configurar.

Las propiedades que se permiten configurar son:

- `fireUrl`
  - URL del servicio del componente central.
- `javax.net.ssl.keyStore` (Opcional)
  - Ruta del almacén de claves para la autenticación mediante certificado con el componente central. Este certificado debe estar dado de alta en la base de datos del componente central, asignado al identificador de la aplicación cliente en la que se esté integrando el componente distribuido.
  - Si se omite este parámetro se usará la configuración establecida a nivel global en la JRE.
- `javax.net.ssl.keyStorePassword` (Opcional)
  - Contraseña del almacén de claves de autenticación SSL.
- `javax.net.ssl.keyStoreType` (Opcional)

- Tipo del almacén de claves del certificado de autenticación SSL: "JKS" (almacén de Java) o "PKCS12" (almacén PKCS12/PFX).
- Por defecto, se considera que el almacén es de tipo JKS.

- `javax.net.ssl.trustStore` (Opcional)
  - Ruta del almacén de certificados de confianza SSL. Esto se usa cuando el certificado con el que se ha montado el SSL del componente central no está en el almacén de confianza de Java y se desea establecer un almacén de confianza alternativo.
  - En caso de querer desactivar la comprobación del certificado SSL del servidor, se puede configurar el valor "all".
  - Si se omite este parámetro se usará la configuración establecida a nivel global en la JRE. Por defecto, se confiará en los certificados dados de alta en el almacén "cacerts".
- `javax.net.ssl.trustStorePassword` (Opcional)
  - Contraseña del almacén de confianza.
- `javax.net.ssl.trustStoreType` (Opcional)
  - Tipo del almacén de confianza: "JKS" (almacén de Java) o "PKCS12" (almacén PKCS12/PFX).
  - Por defecto, se considera que el almacén es de tipo JKS.

Ejemplo de `Properties` de configuración:

```
fireUrl=https://localhost:8443/fire-signature/fireService
javax.net.ssl.keyStore=app_fire.jks
javax.net.ssl.keyStorePassword=11111111
javax.net.ssl.keyStoreType=JKS
```

En caso de no proporcionarse un objeto de configuración en el constructor, se utilizará el fichero "`client_config.properties`" localizado en el directorio configurado mediante la propiedad "`fire.config.path`" de Java. Si no se hubiese configurado esta propiedad, se buscará el mismo fichero en el CLASSPATH de la aplicación.

El uso del fichero "`client_config.properties`" acarrea el inconveniente de que todas las aplicaciones desplegadas en el mismo servidor que utilicen el componente distribuido usarán el mismo fichero. Esto impediría que se identificasen cada cual con su propio certificado ante el componente central. Si dispone en su servidor de varias aplicaciones que utilizan FIRe, deberá proporcionar a cada una de ellas el objeto de configuración en el constructor del objeto `FIReClient`.

5.1.1. CIFRADO DE CONTRASEÑAS

El componente distribuido Java permite que el integrador configure las contraseñas de los almacenes cifradas y codificadas en base64. Esto posibilita que las claves de los almacenes no queden directamente expuestas en el fichero o el objeto de configuración. El mecanismo de cifrado no se proporciona desde FIRe, es el propio integrador el que debe proporcionarlo.

Para poder utilizar contraseñas cifradas en la configuración del componente Java se deberá:

1. Establecer las contraseñas cifradas y codificadas en base 64 en las propiedades "javax.net.ssl.keyStorePassword" y/o "javax.net.ssl.trustStorePassword" del fichero u objeto de configuración. Esto se hará mediante la cadena:

   ```
   {@ciphered: CONTRASENA_CIFRADA_B64 }
   ```

   Por ejemplo:

   ```
   javax.net.ssl.keyStorePassword={@ciphered: aDbb+4nmBHk7ifT= }
   ```

2. Implementar la interfaz Java "es.gob.fire.client.PasswordDecipher" y su método "decipher". Este método recibe el binario resultando de decodificar el Base64 configurado en el fichero u objeto de configuración y debe descifrar ese binario para obtener y devolver la contraseña de los almacenes en claro. El mecanismo de descifrado puede ser cualquiera y utilizar cualquier número de claves, certificados o recursos externos.

3. Modificar la aplicación cliente para, en la construcción del objeto FireClient, especificar una instancia de PasswordDecipher con el que el componente distribuido pueda descifrar las contraseñas.

   Por ejemplo:

```
…
    MiPasswordDecipher decipher = new MiPasswordDecipher();
    decipher.initializarKeys();

    FireClient client = new FireClient("MiAppCode", config, decipher);
…
```

4. Importar en nuestra aplicación el componente distribuido Java de FIRe, junto con sus dependencias y, si no está incluida ya la clase en el proyecto, el módulo con la nueva clase para el descifrado.

5.1.2. CONFIGURACIÓN DE *LOGS*

El componente distribuido Java utiliza SLF4J para la impresión de logs. Esta biblioteca sirve de fachada para permitir que el componente distribuido utilice el mismo sistema de logs que se utilice en la aplicación que lo importa (*log4J*, *logback*, *Java Logging API*, etc). Para ello, deberá importar en su aplicación la biblioteca puente entre SLF4J y la biblioteca de logs que utilice en su aplicación. Considerando que se utilice la versión 1.7.25 de SLF4J (versión importada por defecto) y que se utilice Maven para construir la aplicación, las dependencias que se deben agregar a la aplicación son:

- Java Logging API (biblioteca de logs utilizada en FIRe v2.2 y anteriores)

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.25</version>
</dependency>
```

- Log4J 2

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j-impl</artifactId>
    <version>2.11.1</version>
</dependency>
```

- Log4J 1.2

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.25</version>
</dependency>
```

- Logback
    - Logback es la biblioteca utilizada nativamente por SLF4J. Sólo es necesario importar en el proyecto la propia biblioteca de Logback para que se utilice.

5.2. OPERACIONES

Las distintas operaciones y los métodos disponibles en el componente distribuido son los siguientes:

5.2.1. CONSTRUCCIÓN DEL CLIENTE

La clase `FireClient` dispone de dos constructores:

- `public FireClient(String appId, Properties config)`
    - Construye el objeto con el que realizar las llamadas al componente central.
    - **Parámetros:**
        - `appId`
            - Identificador de aplicación con el que debe autenticarse frente al componente central.
        - `config`
            - Propiedades de configuración.
            - La configuración de este parámetro sustituye a la del fichero de configuración.
    - **Lanza:**
        - `es.gob.fire.client.ClientConfigFilesNotFoundException`

- Si no se ha proporcionado ni encontrado en el sistema el fichero de configuración `client_config.properties`.

- `public FireClient(String appId)`
  - Construye el objeto con el que realizar las llamadas al componente central. La configuración para la conexión y los datos del certificado de autenticación se tomarán del fichero "`client_config.properties`".
  - **Parámetros**:
    - `appId`
      - Identificador de aplicación con el que debe autenticarse frente al componente central.
  - **Lanza:**
    - `es.gob.fire.client.ClientConfigFilesNotFoundException`
      - Si no se ha encontrado en el sistema el fichero de configuración `client_config.properties`.

5.2.2. FIRMA DE DATOS

El API distribuido Java dispone de dos métodos de firma equivalentes. Estos métodos nos permiten enviar a firmar datos indicando la configuración de firma a aplicar mediante cadenas de texto (muy útil para usar campos obtenidos externamente a Java (como formularios Web) o en base a constantes enumeradas que nos permiten establecer los valores sin riesgo de error durante la codificación.

Las opciones de configuración relativas a la operación firma se pueden consultar en el apartado Configuración de la operación de firma.

Los métodos de firma que se incluyen en el API de FIRe son los siguientes:

5.2.2.1. FIRMA A PARTIR DE ENUMERADOS

```
public static SignOperationResult sign(
      String subjectId,
      HttpSignProcessConstants.SignatureOperation op,
      HttpSignProcessConstants.SignatureFormat ft,
      HttpSignProcessConstants.SignatureAlgorithm algth,
      Properties prop,
      byte[] d,
      Properties config
)
            throws
                  java.io.IOException,
```

```
es.gob.fire.client.HttpNetworkException,
es.gob.fire.client.HttpForbiddenException,
es.gob.fire.client.HttpOperationException
```

Inicia una operación de firma proporcionando los datos a firmar y la configuración de firma que debe aplicarse.

- **Parámetros**:
  - `subjectId`
    - Identificador del titular de los certificados.
  - `op`
    - Tipo de operación a realizar (firma, cofirma o contrafirma).
  - `ft`
    - Formato de firma.
  - `algth`
    - Algoritmo de firma.
  - `prop`
    - Propiedades extra a añadir a la firma (puede ser `null`).
  - `d`
    - Datos a firmar.
  - `config`
    - Configuración a indicar al servicio remoto. Su uso depende de la implementación del conector que utilice el componente centralizado.
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
- **Retorno**:
  - Objeto con el identificador de transacción y URL a la que redireccionar al usuario.
- **Lanza**:
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Cuando no se tiene acceso al servicio remoto.
  - `es.gob.fire.client.HttpNetworkException`
    - Cuando se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si se produce un error en el servicio remoto.

5.2.2.2. FIRMA A PARTIR DE CADENAS DE TEXTO

```
public static SignOperationResult sign(
        String subjectId,
        String op,
        String ft,
        String algth,
        String propB64,
```

```
        String dataB64,
        Properties config
)
                throws
                        java.io.IOException,
                        es.gob.fire.client.HttpForbiddenException,
                        es.gob.fire.client.HttpNetworkException,
                        es.gob.fire.client.HttpOperationException,
```

Carga datos e inicia una operación de firma en el para ser posteriormente firmados.

- **Parámetros**:
  - `subjectId`
    - Identificador del titular de los certificados.
  - `op`
    - Tipo de operación a realizar (firma, cofirma o contrafirma).
  - `ft`
    - Formato de firma.
  - `algth`
    - Algoritmo de firma.
  - `propB64`
    - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
  - `dataB64`
    - Datos a firmar en base 64.
  - `config`
    - Configuración a indicar al servicio remoto. Su uso depende de la implementación del conector que utilice el componente centralizado.
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
- **Retorno:**
  - Objeto con el identificador de transacción y URL a la que redireccionar al usuario.
- **Lanza**:
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Cuando no se tiene acceso al servicio remoto.
  - `es.gob.fire.client.HttpNetworkException`
    - Cuando se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.

5.2.3. RECUPERACIÓN DE FIRMA

Este método nos permite completar la construcción y actualización de la firma solicitada anteriormente y recuperar la propia firma y el proveedor de certificados utilizado.

```
public static TransactionResult recoverSignResult(
	String transactionId,
	String subjectId,
	String upgrade
)
	throws
		java.io.IOException,
		es.gob.fire.client.HttpForbiddenException,
		es.gob.fire.client.HttpNetworkException,
		es.gob.fire.client.HttpOperationException,
		es.gob.fire.client.InvalidTransactionException
```

Completa un proceso de firma haciendo uso del servicio de firma en la nube. Los parámetros de configuración "`op`", "`ft`", "`algth`" y "`prop`" deben coincidir con los proporcionados en la operación de carga de datos.

- **Parámetros**:
  - `transactionId`
    - Identificador de la transacción.
  - `subjectId`
    - Identificador del titular de los certificados.
  - `upgrade`
    - Formato al que queremos mejorar la firma (puede ser `null`).
- **Retorno:**
  - Objeto `TransactionResult` con la firma electrónica generada y, si es posible, el nombre del proveedor de certificados seleccionado por el usuario. Si se seleccionó un certificado local, el nombre de proveedor será "local".
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Cuando no se tiene acceso al servicio remoto.
  - `es.gob.fire.client.HttpNetworkException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.

5.2.4. Recuperación de error

Método para obtener la información sobre el error ocurrido en una operación previa. Este método debería usarse en las páginas a las que el sistema redirige al usuario cuando se produce un error.

```
public static TransactionResult recoverErrorResult(
```

```
        String transactionId,
        String subjectId
)
                throws
                        java.io.IOException,
                        es.gob.fire.client.HttpForbiddenException,
                        es.gob.fire.client.HttpNetworkException,
                        es.gob.fire.client.HttpOperationException,
                        es.gob.fire.client.ClientConfigFilesNotFoundException,
                        es.gob.fire.client.InvalidTransactionException
```

Recupera un error identificador en el proceso de firma remota.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción de la operación de firma o firma de lote.
  - `subjectId`
    - Identificador del titular de los certificados.
- **Retorno:**
  - Objeto `TransactionResult` con el código, el mensaje de error y, si es posible, el nombre del proveedor de certificados seleccionado por el usuario. Si se seleccionó un certificado local, el nombre de proveedor será "local".
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.ClientConfigFilesNotFoundException`
    - Si no se ha encontrado en el sistema el fichero de configuración `client_config.properties`.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.

5.2.5. CREACIÓN DE LOTE DE FIRMA

Este método permite crear un lote de firma al que posteriormente podremos asignar documentos y firmar. La configuración de firma especificada se aplicará a todos los documentos del lote, salvo a aquellos que especifiquen su propia configuración.

```
public static CreateBatchResult createBatchProcess(
        String subjectId,
        String op,
```

```
        String ft,
        String algth,
        String propB64,
        String upgrade,
        Properties config
)
                throws
                        java.io.IOException,
                        es.gob.fire.client.HttpForbiddenException,
                        es.gob.fire.client.HttpNetworkException,
                        es.gob.fire.client.HttpOperationException,
```

Crea un lote de firma.

- **Parámetros:**
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmará el lote.
  - `op`
    - Tipo de operación a realizar (firma, cofirma o contrafirma).
  - `ft`
    - Formato de firma.
  - `algth`
    - Algoritmo de firma.
  - `propB64`
    - Propiedades extra en base 64 que aplicar a las firmas del lote (puede ser `null`).
  - `upgrade`
    - Formato avanzado de firma electrónica al que actualizar las firmas.
  - `config`
    - Configuración a indicar al servicio remoto para ejecutar la operación (dependiente de la implementación).
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
- **Retorno:**
  - Objeto `CreateBatchResult` con el identificador de la transacción de firma del lote.
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Cuando se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.

5.2.6. AGREGAR DOCUMENTO AL LOTE DE FIRMA

Estos métodos permiten agregar documentos a un lote de firma. Según el método utilizado, el documento que se agregue se firmará según la configuración establecida al crear el lote o según la configuración particular proporcionada junto al documento.

El algoritmo de firma siempre será el indicado en la creación del lote.

5.2.6.1. AGREGAR DOCUMENTO CON LA CONFIGURACIÓN DE FIRMA DEL LOTE

```
public static void addDocumentToBatch(
        String transactionId,
        String subjectId,
        String documentId,
        byte[] document,
        Properties config
)
                throws
                        java.io.IOException,
                        es.gob.fire.client.HttpForbiddenException,
                        es.gob.fire.client.HttpNetworkException,
                        es.gob.fire.client.HttpOperationException,
                        es.gob.fire.client.NumDocumentsExceededException,
                        es.gob.fire.client.InvalidTransactionException,
                        es.gob.fire.client.DuplicateDocumentException
```

Agrega un documento a un lote para que se firme con la configuración de firma indicada durante la creación del lote.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelta por la operación de creación del lote.
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmará el lote.
  - `documentId`
    - Identificador único del documento que se adjunta al lote.
    - Este es el identificador por el cual identificaremos la firma de este documento.
  - `document`
    - Datos a firmar como parte del lote.
    - Si se ha configurado una clase gestora de documentos a medida:
      - Si se proporciona este dato, se pasará al gestor de documentos para que recupere el documento que hay que firmar.
      - Si no se proporciona este dato, lo que se pasará al gestor de documentos es el identificador de documento configurado.
  - `config`

- Configuración a indicar al conector del proveedor para la gestión del documento.
- Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.

- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.NumDocumentsExceededException`
    - Cuando se intentan agregar más documentos de los permitidos al lote.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.
  - `es.gob.fire.client.DuplicateDocumentException`
    - Cuando se el identificador de documento ya se usó para otro documento del lote.

5.2.6.2. AGREGAR DOCUMENTO CON SU PROPIA CONFIGURACIÓN DE FIRMA

```
public static void addDocumentToBatch(
        String transactionId,
        String subjectId,
        String documentId,
        byte[] document,
        Properties config,
        String op,
        String ft,
        String propB64,
        String upgrade
)
                throws
                        java.io.IOException,
                        es.gob.fire.client.HttpForbiddenException,
                        es.gob.fire.client.HttpNetworkException,
                        es.gob.fire.client.HttpOperationException,
                        es.gob.fire.client.NumDocumentsExceededException,
                        es.gob.fire.client.InvalidTransactionException,
                        es.gob.fire.client.DuplicateDocumentException
```

Agrega un documento a un lote para que se firme con la configuración expresada en la llamada a este mismo método y el algoritmo indicado durante la creación del lote.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelta por la operación de creación del lote.
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmará el lote.
  - `documentId`
    - Identificador único del documento que se adjunta al lote.
    - Este es el identificador por el cual identificaremos la firma de este documento.
  - `document`
    - Datos a firmar como parte del lote.
    - Si se ha configurado una clase gestora de documentos a medida:
      - Si se proporciona este dato, se pasará al gestor de documentos para que recupere el documento que hay que firmar.
      - Si no se proporciona este dato, lo que se pasará al gestor de documentos es el identificador de documento configurado.
  - `config`
    - Configuración a indicar al proveedor del conector para la gestión del documento.
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
  - `op`
    - Operación criptográfica a realizar (firma, cofirma o contrafirma).
  - `ft`
    - Formato de firma.
  - `propB64`
    - Configuración adicional del formato de firma.
  - `upgrade`
    - Nombre del formato actualizado para la mejora de la firma antes de recuperarla.
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.NumDocumentsExceededException`
    - Cuando se intentan agregar más documentos de los permitidos al lote.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.
  - `es.gob.fire.client.DuplicateDocumentException`
    - Cuando se el identificador de documento ya se usó para otro documento del lote.

5.2.7. FIRMAR LOTE

Firma un lote previamente creado y obtiene como resultado un objeto con la URL a la que se debe redirigir al usuario para que seleccione el origen del certificado de firma, el propio certificado y autorice la operación.

```
public static SignOperationResult signBatch(
        String transactionId,
        String subjectId,
        boolean stopOnError
)
                throws
                        java.io.IOException,
                        es.gob.fire.client.HttpForbiddenException,
                        es.gob.fire.client.HttpNetworkException,
                        es.gob.fire.client.HttpOperationException,
                        es.gob.fire.client.InvalidTransactionException
```

Ejecuta el proceso de firma sobre todos los documentos de un lote previamente creado.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelta por la operación de creación del lote.
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmará el lote.
  - `stopOnError`
    - Indica si se debe detener el proceso de firma al fallar una de las firmas.
- **Retorno:**
  - Objeto `SignOperationResult` con el ID de transacción y la URL de redirección para la firma del lote.
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.

5.2.8. Recuperación del resultado del lote

Recupera el resultado de un lote ya firmado. Este resultado incluye el nombre del proveedor de certificados utilizado, la relación de identificadores de los documentos firmados y si cada uno de ellos se firmó correctamente o no.

```
public static BatchResult recoverBatchResult(
       String transactionId,
       String subjectId
)
              throws
                     java.io.IOException,
                     es.gob.fire.client.HttpOperationException,
                     es.gob.fire.InvalidTransactionException
```

Recupera el resultado de una operación de firma de un lote de documentos.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelta por la operación de creación del lote.
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmó el lote.
- **Retorno:**
  - Objeto `BatchResult` con el nombre del proveedor de certificados utilizado, el listado de identificadores de los documentos procesados, el resultado de firmar cada uno de ellos. Por cada documento que se no haya firmado correctamente, se incluye un código de error que identifica el tipo de error producido.
    - Puede consultar estos códigos en el anexo Códigos de error de las firmas de un lote.
  - Se indica cómo terminó la firma de cada documento del lote, pero no se incluye la propia firma. Para ello debe usarse el método **`recoverBatchSign`**.
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.

5.2.9. COMPROBACIÓN DE PROGRESO DE LA FIRMA DE UN LOTE

Consulta el porcentaje de avance de una operación de firma de lote.

```
public static float recoverBatchResultState(
       String transactionId,
       String subjectId
)
```

```
        throws
                java.io.IOException,
                es.gob.fire.client.HttpForbiddenException,
                es.gob.fire.client.HttpNetworkException,
                es.gob.fire.client.HttpOperationException,
                es.gob.fire.client.InvalidTransactionException
```

Recupera el porcentaje actual de progreso de la firma de un lote de firma.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelta por la operación de creación del lote.
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmó el lote.
- **Retorno:**
  - Avance del proceso del lote.
  - Es un valor decimal que va de cero (sin empezar) a uno (terminado). Por ejemplo, si se obtiene el valor "0'3" es que ha procesado ya el 30% del lote.
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.

5.2.10. RECUPERACIÓN DE UNA FIRMA DE UN LOTE

Recupera una firma electrónica generada exitosamente como parte de un lote de firma.

```
public static TransactionResult recoverBatchSign(
        String transactionId,
        String subjectId,
        String docId
)
        throws
                java.io.IOException,
                es.gob.fire.client.HttpForbiddenException,
                es.gob.fire.client.HttpNetworkException,
                es.gob.fire.client.HttpOperationException,
                es.gob.fire.client.InvalidTransactionException,
```

`es.gob.fire.client.InvalidBatchDocumentException`,
`es.gob.fire.client.BatchNoSignedException`

Recupera una firma obtenida como parte del proceso de firma de lote.

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelta por la operación de creación del lote.
  - `subjectId`
    - Identificador del titular de los certificados con los que se firmó el lote.
  - `docId`
    - Identificador del documento de cuya firma queremos recuperar.
- **Retorno:**
  - Objeto `TransactionResult` con la firma electrónica de un documento concreto del lote.
- **Lanza:**
  - `java.io.IOException`
    - Si hay problemas en la llamada al servicio de red.
  - `es.gob.fire.client.HttpForbiddenException`
    - Si el usuario no está dado de alta o no tiene permisos para ejecutar la operación.
  - `es.gob.fire.client.HttpNetworkException`
    - Si se produce un error de red.
  - `es.gob.fire.client.HttpOperationException`
    - Si hay problemas con el servicio remoto.
  - `es.gob.fire.client.InvalidTransactionException`
    - Cuando la transacción no existe o está caducada.
  - `es.gob.fire.client.InvalidBatchDocumentException`
    - Cuando se indica el identificador de un documento que no existe en el lote o que no se firmó correctamente.
  - `es.gob.fire.client.BatchNoSignedException`
    - Cuando se solicita recuperar una firma del lote antes de firmarlo.

6. INTEGRACIÓN DEL COMPONENTE DISTRIBUIDO .NET

Para la integración de Clave Firma en aplicativos .NET se proporciona un componente distribuido en este lenguaje.

Para el uso de las funciones de este componente distribuido se deberá importar la biblioteca "`fire-client.dll`" en la aplicación cliente y hacer uso del espacio de nombres "`FIRe`".

6.1. CONFIGURACIÓN

El servicio del componente central al que accederá esta biblioteca y la configuración de acceso pueden establecerse directamente por medio de un diccionario con las propiedades para la configuración del componente distribuido o a través del registro de Windows dentro de la clave "`HKEY_CURRENT_USER\Software\FIRe`". Las propiedades que deberán pasarse en el diccionario y las claves de registro que se pueden configurar comparten los mismos nombres. Estos son:

- `fire_service`: URL del servicio que procesa y realiza el tipo de operación solicitado.
- `admit_all_certs`: Indicador de admisión de cualquier tipo de certificado ("`false`" por defecto).
- `ssl_client_pkcs12`: Ruta del almacén de claves PKCS#12 para la autenticación cliente SSL.
- `ssl_client_pass`: Contraseña del almacén de claves definido anteriormente.

Todas las claves de registro son de tipo cadena ("REG_SZ").

6.2. OPERACIONES

Las distintas operaciones y los métodos disponibles en el componente distribuido son los siguientes:

6.2.1. CONSTRUCCIÓN DEL CLIENTE

La clase `FireClient` dispone de dos constructores:

- `public FireClient(string appId, Dictionary<string, string> config)`
    - Construye el objeto con el que realizar las llamadas al componente central. Proporciona la configuración para la conexión en un diccionario. Las propiedades que no se encuentren en este se buscarán en la clave "`HKEY_CURRENT_USER\Software\FIRe`" del registro de Windows.
    - **Parámetros:**
        - `appId`
            - Identificador de aplicación con el que debe autenticarse frente al componente central.
        - `config`

- Diccionario con las propiedades de configuración.

- **Lanza:**
  - `ConfigureException`
    - Si no se ha encontrado la propiedad `fire_service` ni en el diccionario de configuración ni en el registro de Windows.

- `public FireClient(String appId)`
  - Construye el objeto con el que realizar las llamadas al componente central. La configuración para la conexión y los datos del certificado de autenticación se tomarán del registro de Windows.
  - **Parámetros**:
    - `appId`
      - Identificador de aplicación con el que debe autenticarse frente al componente central.
  - **Lanza:**
    - `ConfigureException`
      - Si no se ha encontrado la propiedad `fire_service` en el registro de Windows.

6.2.2. FIRMA DE DATOS

Este método nos permite enviar datos a firmar junto con la configuración de firma que se desea aplicar.

Las opciones de configuración relativas a la operación firma se pueden consultar en el apartado Configuración de la operación de firma.

```
public static FireLoadResult sign(string subjectId, string op, string ft, string algth,
string propB64, string dataB64, string confB64)
```

- **Parámetros:**
  - `subjectId`
    - Identificador del usuario propietario de los certificados.
  - `op`
    - Operación de firma.
  - `ft`
    - Formato de firma.
  - `algth`
    - Algoritmo de firma.
  - `propB64`
    - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
  - `dataB64`
    - Datos a firmar en base 64.
  - `confB64`

  - Configuración a indicar al servicio remoto para ejecutar la operación (dependiente de la implementación). Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por "\n". Ejemplo: `clave1=valor1\nclave2=valor2\nclave3=valor3`
  - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
  - Esta cadena debe proporcionarse codificada en base 64.
- **Retorno:**
  - Objeto FireLoadResult con el identificador de transacción y la URL de redirección para la identificación del usuario.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

6.2.3. RECUPERACIÓN DE FIRMA

Este método nos permite obtener el resultado de una firma realizada anteriormente.

```
public static FireTransactionResult recoverSign(string transactionId, string subjectId, string upgrade)
```

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelto por la operación de firma.
  - `subjectId`
    - Identificador del titular de los certificados.
  - `upgrade`
    - Formato al que queremos mejorar la firma (puede ser `null`).
- **Retorno:**
  - Objeto `FireTransactionResult` con el nombre del proveedor de firma utilizado y la firma electrónica generada.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

6.2.4. RECUPERACIÓN DE ERROR

Este método nos permite recuperar la información de error de una operación anterior.

```
public static FireTransactionResult recoverError(string transactionId, string subjectId)
```

- **Parámetros:**
  - `transactionId`
    - Identificador de la transacción devuelto por la operación de firma.
  - `subjectId`
    - Identificador del titular de los certificados.
- **Retorno:**
  - Objeto FireTransactionResult con el nombre del proveedor utilizado y el mensaje y código de error producido durante la firma.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido o la respuesta del servidor no es correcta.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

6.2.5. CREACIÓN DE LOTE DE FIRMA

Este método permite recuperar un certificado de firma que hayamos solicitado recientemente para el usuario y del que dispongamos del identificador de trámite de la solicitud.

```
public static FireTransactionIdResult createBatchProcess(string subjectId, string op,
string ft, string algth, string propB64, string upgrade, string confB64)
```

- **Parámetros:**
  - `subjectId`
    - Identificador del titular de los certificados.
  - `op`
    - Operación de firma.
  - `ft`
    - Formato de firma.
  - `algth`
    - Algoritmo de firma.
  - `propB64`
    - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
  - `upgrade`
    - Parámetros de actualización.
  - `confB64`

  - Configuración a indicar al servicio remoto para ejecutar la operación (dependiente de la implementación). Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por "\n". Ejemplo: `clave1=valor1\nclave2=valor2\nclave3=valor3`
  - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
  - Este parámetro se proporciona de esta manera para permitir que en futuras versiones se puedan agregan nuevos parámetros sin necesidad de modificar la declaración del método.
  - Esta cadena debe proporcionarse codificada en base 64.
- **Retorno:**
  - Objeto `FireTransactionIdResult` con el identificador de transacción.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

6.2.6. AGREGAR DOCUMENTO AL LOTE DE FIRMA

Estos métodos permiten agregar documentos a un lote de firma. Según el método utilizado, el documento que se agregue se firmará según la configuración establecida al crear el lote o según la configuración particular proporcionada junto al documento.

El algoritmo de firma siempre será el indicado en la creación del lote.

6.2.6.1. AGREGAR DOCUMENTO CON LA CONFIGURACIÓN DE FIRMA DEL LOTE

```
public static void addDocumentToBatch(string transactionId, string subjectId, string documentId, string documentB64, string confB64)
```

- **Parámetros:**
  - `transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `subjectId`
    - Identificador del titular de los certificados.
  - `documentId`
    - Identificador único del documento que se adjunta al lote.
    - Este es el identificador por el cual identificaremos la firma de este documento.
  - `documentB64`
    - Datos del documento en base 64.
    - Si se ha configurado una clase gestora de documentos a medida:
      - Si se proporciona este dato, se pasará al gestor de documentos para que recupere el documento que hay que firmar.

- Si no se proporciona este dato, lo que se pasará al gestor de documentos es el identificador de documento configurado.

  - `confB64`
    - Configuración a indicar al conector del proveedor para la gestión del documento.
    - Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por "\n". Ejemplo: `clave1=valor1\nclave2=valor2\nclave3=valor3`
    - Esta cadena debe proporcionarse codificada en base 64.
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.

- **Retorno:**
  - Vacío.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
  - `NumDocumentsExceededException`
    - Cuando se intentan agregar más documentos de los permitidos al lote.
  - `DuplicateDocumentException`
    - Cuando se el identificador de documento ya se usó para otro documento del lote.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

6.2.6.2. AGREGAR DOCUMENTO CON SU PROPIA CONFIGURACIÓN DE FIRMA

```
public static void addDocumentToBatch(string transactionId, string subjectId, string documentId, string documentB64, string op, string ft, string propB64, string upgrade, string confB64)
```

- **Parámetros:**
  - `transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `subjectId`
    - Identificador del titular de los certificados.
  - `documentId`
    - Identificador único del documento que se adjunta al lote.
    - Este es el identificador por el cual identificaremos la firma de este documento.
  - `documentB64`
    - Datos del documento en base 64.
    - Si se ha configurado una clase gestora de documentos a medida:
      - Si se proporciona este dato, se pasará al gestor de documentos para que recupere el documento que hay que firmar.

        - Si no se proporciona este dato, lo que se pasará al gestor de documentos es el identificador de documento configurado.
    - `op`
        - Operación de firma.
    - `ft`
        - Formato de firma.
    - `propB64`
        - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
    - `upgrade`
        - Parámetros de actualización.
    - `confB64`
        - Configuración a indicar al conector del proveedor para la gestión del documento.
        - Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por "\n". Ejemplo: `clave1=valor1\nclave2=valor2\nclave3=valor3`
        - Esta cadena debe proporcionarse codificada en base 64.
        - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
- **Retorno:**
    - Vacío.
- **Lanza**:
    - `ArgumentException`
        - Cuando se proporciona un parámetro no válido.
    - `HttpForbiddenException`
        - Cuando falla la autenticación con el componente central.
    - `HttpNetworkException`
        - Cuando se produce un error de conexión con el componente central.
    - `InvalidTransactionException`
        - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
    - `NumDocumentsExceededException`
        - Cuando se intentan agregar más documentos de los permitidos al lote.
    - `DuplicateDocumentException`
        - Cuando se el identificador de documento ya se usó para otro documento del lote.
    - `HttpOperationException`
        - Cuando se produce un error interno del servidor.

6.2.7. FIRMAR LOTE

Este método permite recuperar un certificado de firma que hayamos solicitado recientemente para el usuario y del que dispongamos del identificador de trámite de la solicitud.

```
public static FireLoadResult signBatch(string transactionId, string subjectId, string stopOnError)
```

- **Parámetros:**
    - `transactionId`
        - Identificador de transacción devuelto por el método de creación de lote.
    - `subjectId`

        - Identificador del titular de los certificados.
    - `stopOnError`
        - Indicador de si debe detenerse al producirse un error en la firma.
- **Retorno:**
    - Objeto `FireLoadResult` con el identificador de transacción y la URL de redirección.
- **Lanza**:
    - `ArgumentException`
        - Cuando se proporciona un parámetro no válido.
    - `HttpForbiddenException`
        - Cuando falla la autenticación con el componente central.
    - `HttpNetworkException`
        - Cuando se produce un error de conexión con el componente central.
    - `InvalidTransactionException`
        - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
    - `HttpOperationException`
        - Cuando se produce un error interno del servidor.

6.2.8. RECUPERACIÓN DEL RESULTADO DEL LOTE

Este método nos permite recuperar el resultado de la firma de un lote realizada anteriormente.

```
public static FireBatchResult recoverBatchResult(string transactionId, string subjectId)
```

- **Parámetros:**
    - `transactionId`
        - Identificador de transacción devuelto por el método de creación de lote.
    - `subjectId`
        - Identificador del titular de los certificados.
- **Retorno:**
    - Objeto `FireBatchResult` con el nombre del proveedor de firma utilizado y los resultados de firmar cada documento, incluyendo su identificador de documento y el código de error en caso de haber fallado la firma.
        - Puede consultar los códigos de error de la firma de un documento de un lote en el anexo Códigos de error de las firmas de un lote.
    - Se indica cómo terminó la firma de cada documento del lote, pero no se incluye la propia firma. Para ello debe usarse el método **`recoverBatchSign`**.
- **Lanza**:
    - `ArgumentException`
        - Cuando se proporciona un parámetro no válido o la respuesta del servidor no es correcta.
    - `HttpForbiddenException`
        - Cuando falla la autenticación con el componente central.
    - `HttpNetworkException`
        - Cuando se produce un error de conexión con el componente central.
    - `InvalidTransactionException`
        - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
    - `HttpOperationException`
        - Cuando se produce un error interno del servidor.

6.2.9. COMPROBACIÓN DE PROGRESO DE LA FIRMA DE UN LOTE

Este método nos permite recuperar el progreso actual del proceso de firma de un lote.

```
public static float recoverBatchResultState(string transactionId, string subjectId)
```

- **Parámetros:**
  - `transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `subjectId`
    - Identificador del titular de los certificados.
- **Retorno:**
  - Número decimal entre 0 y 1 indicando el porcentaje de progreso de firma.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido o la respuesta del servidor no es correcta.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

6.2.10. RECUPERACIÓN DE UNA FIRMA DE UN LOTE

Este método nos permite recuperar la firma individual de uno de los documentos incluidos en el lote.

```
public static FireTransactionResult recoverBatchSign(string transactionId, string subjectId, string docId)
```

- **Parámetros:**
  - `transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `subjectId`
    - Identificador del titular de los certificados.
  - `docId`
    - Identificador del documento cuya firma quiere ser recuperada.
- **Retorno:**
  - Objeto `FireTransactionResult` con la firma electrónica de un documento concreto del lote.
- **Lanza**:
  - `ArgumentException`
    - Cuando se proporciona un parámetro no válido o la respuesta del servidor no es correcta.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.

- `InvalidTransactionException`
  - Cuando se intenta operar sobre una transacción inexistente o ya caducada.
- `InvalidBatchDocumentException`
  - Cuando se indica el identificador de un documento que no existe en el lote o que no se firmó correctamente.
- `BatchNoSignedException`
  - Cuando se solicita recuperar una firma del lote antes de firmar el propio lote.
- `HttpOperationException`
  - Cuando se produce un error interno del servidor.

7. INTEGRACIÓN DEL COMPONENTE DISTRIBUIDO PHP

Para la integración de FIRe en aplicativos PHP, se proporciona un componente distribuido en este lenguaje. Para el uso de las funciones de este componente distribuido será necesario tener habilitada la extensión de cURL en nuestro servidor e importar "`fire_api.php`" en nuestra página.

7.1. CONFIGURACIÓN

En "`fire_api.php`" se deberá configurar la variable `SERVICEURL` que se define al principio del fichero con la URL del componente central que se desee utilizar:

- `SERVICEURL`: URL de los servicios de firma.

También se deberán establecer las propiedades de configuración SSL cliente de cURL para la conexión con el componente central mediante la variable "$client_ssl_curl_options". Para más información sobre las propiedades que se pueden establecer en esta variable, consulte las propiedades que comienzan por "CURLOPT_SSL" del listado de propiedades de cURL (https://curl.haxx.se/libcurl/c/easy_setopt_options.html).

La configuración que se proporciona como ejemplo en el fichero es:

```
$client_ssl_curl_options = array(
        CURLOPT_SSLCERT => "ssl_client_cert.crt", // Certificado de conexión
        CURLOPT_SSLCERTTYPE => "PEM", // Tipo de certificado
        CURLOPT_SSLKEY => "ssl_client_key.pem", // Clave privada de conexión
        CURLOPT_SSLKEYTYPE => "PEM", // Tipo de clave privada
        CURLOPT_SSLKEYPASSWD => "password", // Contraseña de la clave privada
        CURLOPT_SSL_VERIFYPEER => 0 // Verificar certificado SSL del servidor
);
```

7.2. OPERACIONES

Las distintas operaciones y los métodos disponibles en el componente distribuido son los siguientes:

7.2.1. FIRMA DE DATOS

Este método nos permite enviar los datos necesarios para realizar una firma en servidor.

Las opciones de configuración relativas a la operación firma se pueden consultar en el apartado Configuración de la operación de firma.

```
function sign($appId, $subjectId, $op, $ft, $algth, $propB64, $dataB64, $confB64)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$subjectId`
    - Identificador del usuario propietario de los certificados.
  - `$op`
    - Operación de firma.
  - `$ft`
    - Formato de firma.
  - `$algth`
    - Algoritmo de firma.
  - `$propB64`
    - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
  - `$dataB64`
    - Datos a firmar en base 64.
  - `$confB64`
    - Configuración a indicar al servicio remoto para ejecutar la operación (dependiente de la implementación). Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por el carácter '\n'.
    - **Importante:** Para la correcta codificación del carácter '\n', es necesario que la cadena completa se proporcione rodeada con comillas dobles o, en caso de usarse comillas simples se inserte un salto de línea en el propio código. Ejemplos:
      - `"clave1=valor1\nclave2=valor2\nclave3=valor3"`
      - ```
        'clave1=valor1
        clave2=valor2
        clave3=valor3'
        ```
    - Esta cadena debe proporcionarse codificada en base 64.
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
- **Retorno:**
  - Objeto `SignOperationResult` con los siguientes campos:
    - `transactionId`: Identificador de transacción.
    - `redirectUrl`: URL a la que redirigir al usuario.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

7.2.2. RECUPERACIÓN DE FIRMA

Este método nos permite obtener el resultado de una firma realizada anteriormente.

```
function recoverSign($appId, $transactionId, $upgrade)
```
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de la transacción devuelto por la operación de firma.
  - `$upgrade`
    - Formato al que queremos mejorar la firma (puede ser `null`).
- **Retorno:**
  - Objeto `TransactionResult` con los siguientes campos:
    - `state`: Si el resultado es correcto (0) o si se ha producido algún error (-1).
    - `providerName`: Nombre del proveedor de firma utilizado.
    - `errorCode`: Código de error (si se produjo alguno).
    - `errorMessage`: Mensaje de error (si se produjo alguno).
    - `result`: Firma generada (si no se produjo un error).
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se solicita operar con una transacción no valida o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

7.2.3. RECUPERACIÓN DE ERROR

Este método nos permite recuperar la información de error de una operación anterior.

```
function recoverError($appId, $transactionId)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de la transacción devuelto por la operación de firma.
- **Retorno:**
  - Objeto `TransactionResult` con los siguientes campos:
    - `state`: Si el resultado es correcto (0) o si se ha producido algún error (-1).
    - `providerName`: Nombre del proveedor de firma utilizado.
    - `errorCode`: Código de error.
    - `errorMessage`: Mensaje de error.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.

- `InvalidTransactionException`
  - Cuando se solicita operar con una transacción no valida o ya caducada.
- `HttpOperationException`
  - Cuando se produce un error interno del servidor.

7.2.4. CREACIÓN DE LOTE DE FIRMA

Este método crea un lote de firma con el que poder firmar múltiples documentos simultáneamente.

```
function createBatchProcess($appId, $subjectId, $op, $ft, $algth, $propB64, $upgrade, $confB64)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$subjectId`
    - Identificador del usuario propietario de los certificados.
  - `$op`
    - Operación de firma.
  - `$ft`
    - Formato de firma.
  - `$algth`
    - Algoritmo de firma.
  - `$propB64`
    - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
  - `$upgrade`
    - Actualización.
  - `$confB64`
    - Configuración a indicar al servicio remoto para ejecutar la operación (dependiente de la implementación). Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por el carácter '\n'.
    - **Importante:** Para la correcta codificación del carácter '\n', es necesario que la cadena completa se proporcione rodeada con comillas dobles o, en caso de usarse comillas simples se inserte un salto de línea en el propio código. Ejemplos:
      - `"clave1=valor1\nclave2=valor2\nclave3=valor3"`
      - ```
        'clave1=valor1
        clave2=valor2
        clave3=valor3'
        ```
    - Esta cadena debe proporcionarse codificada en base 64.
    - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
- **Retorno:**
  - Objeto `TransactionIdResult` con las propiedades:
    - `transactionId`: Identificador de transacción.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
        ```

        - Cuando se produce un error de conexión con el componente central.
    - `HttpOperationException`
        - Cuando se produce un error interno del servidor.

7.2.5. AGREGAR DOCUMENTO AL LOTE DE FIRMA

Estos métodos permiten agregar documentos a un lote de firma. Según el método utilizado, el documento que se agregue se firmará según la configuración establecida al crear el lote o según la configuración particular proporcionada junto al documento.

El algoritmo de firma siempre será el indicado en la creación del lote.

7.2.5.1. AGREGAR DOCUMENTO CON LA CONFIGURACIÓN DE FIRMA DEL LOTE

Para agregar un documento a un lote y que se firme con la configuración de firma establecida durante la creación del lote, se deberá utilizar el siguiente método:

```
function addDocumentToBatch($appId, $transactionId, $documentId, $documentB64, $confB64)
```

- **Parámetros:**
    - `$appId`
        - Identificador de la aplicación que desea acceder al componente central.
    - `$transactionId`
        - Identificador de transacción devuelto por el método de creación de lote.
    - `$documentId`
        - Identificador único del documento que se adjunta al lote.
        - Este es el identificador por el cual identificaremos la firma de este documento.
    - `$documentB64`
        - Datos en base 64 a firmar.
        - Si se ha configurado una clase gestora de documentos a medida:
            - Si se proporciona este dato, se pasará al gestor de documentos para que recupere el documento que hay que firmar.
            - Si no se proporciona este dato, lo que se pasará al gestor de documentos es el identificador de documento configurado.
    - `$confB64`
        - Configuración a indicar al conector del proveedor para la gestión del documento.
        - Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por el carácter '\n'.
        - **Importante:** Para la correcta codificación del carácter '\n', es necesario que la cadena completa se proporcione rodeada con comillas dobles o, en caso de usarse comillas simples se inserte un salto de línea en el propio código. Ejemplos:
            - `"clave1=valor1\nclave2=valor2\nclave3=valor3"`
            - ```
              'clave1=valor1
              clave2=valor2
              clave3=valor3'
              ```
        - Esta cadena debe proporcionarse codificada en base 64.
        - Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.
              ```

- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `NumDocumentsExceededException`
    - Cuando se agregan más documentos de los permitidos a un lote.
  - `DuplicateDocumentException`
    - Cuando se agrega a un lote un documento con un identificador ya utilizado en el lote.
  - `InvalidTransactionException`
    - Cuando se solicita operar con una transacción no valida o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

7.2.5.2. AGREGAR DOCUMENTO CON SU PROPIA CONFIGURACIÓN DE FIRMA

Para agregar un documento a un lote y que se firme con una configuración de firma específica, se deberá utilizar el siguiente método. El algoritmo de firma utilizado será el definido durante la creación del lote.

```
function addCustomDocumentToBatch($appId, $transactionId, $documentId, $documentB64, $op, $ft, $propB64, $upgrade, $confB64)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `$documentId`
    - Identificador único del documento que se adjunta al lote.
    - Este es el identificador por el cual identificaremos la firma de este documento.
  - `$documentB64`
    - Datos en base 64 a firmar.
    - Si se ha configurado una clase gestora de documentos a medida:
      - Si se proporciona este dato, se pasará al gestor de documentos para que recupere el documento que hay que firmar.
      - Si no se proporciona este dato, lo que se pasará al gestor de documentos es el identificador de documento configurado.
  - `$op`
    - Operación de firma.
  - `$ft`
    - Formato de firma.
  - `$propB64`
    - Propiedades extra en base 64 que aplicar a la firma (puede ser `null`).
  - `$upgrade`
    - Formato longevo al que actualizar la firma (puede ser `null`).
  - `$confB64`
    - Configuración a indicar al conector del proveedor para la gestión del documento.

- Este será el resultado de codificar en base 64 una cadena compuesta por tuplas "`clave=valor`" separadas por el carácter '\n'.
- **Importante:** Para la correcta codificación del carácter '\n', es necesario que la cadena completa se proporcione rodeada con comillas dobles o, en caso de usarse comillas simples se inserte un salto de línea en el propio código. Ejemplos:
  - `"clave1=valor1\nclave2=valor2\nclave3=valor3"`
  - ```
    'clave1=valor1
    clave2=valor2
    clave3=valor3'
    ```
- Esta cadena debe proporcionarse codificada en base 64.
- Para saber más detalles sobre qué propiedades hay que configurar en este parámetro, consulte el apartado Configuración de los proveedores y el componente central.

- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `NumDocumentsExceededException`
    - Cuando se agregan más documentos de los permitidos a un lote.
  - `DuplicateDocumentException`
    - Cuando se agrega a un lote un documento con un identificador ya utilizado en el lote.
  - `InvalidTransactionException`
    - Cuando se solicita operar con una transacción no valida o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

7.2.6. FIRMAR LOTE

Este método nos permite enviar una petición de firma del proceso batch creado anteriormente.

```
function signBatch($appId, $transactionId, $stopOnError)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `$stopOnError`
    - Establece si se debe parar la operación al producirse un error (`true`) o no (`false`).
- **Retorno:**
  - Objeto `SignOperationResult` con los siguientes campos:
    - `transactionId`: Identificador de transacción.
    - `redirectUrl`: URL a la que redirigir al usuario.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
```

- `HttpForbiddenException`
  - Cuando falla la autenticación con el componente central.
- `HttpNetworkException`
  - Cuando se produce un error de conexión con el componente central.
- `InvalidTransactionException`
  - Cuando se solicita operar con una transacción no valida o ya caducada.
- `HttpOperationException`
  - Cuando se produce un error interno del servidor.

7.2.7. RECUPERACIÓN DEL RESULTADO DEL LOTE

Este método nos permite recuperar el resultado de firmar un lote.

```
function recoverBatchResult($appId, $transactionId)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
- **Retorno:**
  - Listado con los resultados de firmar cada documento. Cada elemento del listado tendrá las propiedades:
    - `id`: Identificador del documento firmado.
    - `ok`: Indicador de si la firma se completó correctamente (`true`) o no (`false`).
    - `dt`: Código de tipo de error en caso de haberse producido uno. Los códigos existentes son:
      - Puede consultar estos códigos en el anexo Códigos de error de las firmas de un lote.
    - `providerName`: Nombre del proveedor con el que se realizó la firma.
  - Se indica cómo terminó la firma de cada documento del lote, pero no se incluye la propia firma. Para ello debe usarse el método **`recoverBatchSign`**.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se solicita operar con una transacción no valida o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

7.2.8. COMPROBACIÓN DE PROGRESO DE LA FIRMA DE UN LOTE

Este método nos permite recuperar el progreso actual del proceso de firma del batch.

```
function recoverBatchResultState($appId, $transactionId)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de la transacción devuelto por el método de creación de lote.
- **Retorno:**
  - Número decimal entre 0 y 1 indicando el porcentaje de documentos del lote que se han procesado hasta el momento. Por ejemplo, si el método devuelve 0'3, sabremos que se han procesado el 30% de los documentos del lote.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidTransactionException`
    - Cuando se solicita operar con una transacción no valida o ya caducada.
  - `HttpOperationException`
    - Cuando se produce un error interno del servidor.

7.2.9. RECUPERACIÓN DE UNA FIRMA DE UN LOTE

Este método nos permite recuperar la firma individual de uno de los documentos incluidos en el proceso batch.

```
function recoverBatchSign($appId, $transactionId, $docId)
```

- **Parámetros:**
  - `$appId`
    - Identificador de la aplicación que desea acceder al componente central.
  - `$transactionId`
    - Identificador de transacción devuelto por el método de creación de lote.
  - `$docId`
    - Identificador del documento cuya firma quiere ser recuperada.
- **Retorno:**
  - Objeto `TransactionResult` con la firma electrónica de un documento concreto del lote.
- **Excepciones:**
  - `InvalidArgumentException`
    - Cuando se proporcionó algún parámetro no válido.
  - `HttpForbiddenException`
    - Cuando falla la autenticación con el componente central.
  - `HttpNetworkException`
    - Cuando se produce un error de conexión con el componente central.
  - `InvalidBatchDocumentException`
    - Cuando se indica el identificador de un documento que no existe en el lote o que no se firmó correctamente.
  - `BatchNoSignedException`
    - Cuando se solicita recuperar una firma del lote antes de firmarlo.
  - `InvalidTransactionException`
    - Cuando se solicita operar con una transacción no valida o ya caducada.

- `HttpOperationException`
  - Cuando se produce un error interno del servidor.

8. CONFIGURACIÓN DE LA OPERACIÓN DE FIRMA

La firma electrónica de los datos se realiza en base a las bibliotecas de FIRe, de tal forma que muchas de las propiedades de firma (especialmente los "Parámetros de firma" se configuran en base a los valores que estas bibliotecas establecen.

8.1. OPERACIÓN CRIPTOGRÁFICA

Se refiere a la operación criptográfica que se desea realizar. Los valores posibles son:

- `sign`
  - Indica que se desea realizar una operación de firma sobre los datos proporcionados.
- `cosign`
  - Indica que se desea agregar una nueva versión en paralelo sobre la firma especificada.
- `countersign`
  - Indica que se desea agregar una nueva versión en cascada sobre la firma especificada.

8.2. ALGORITMO DE FIRMA

Los algoritmos de firma soportados por FIRe son:

- `SHA1withRSA`
- `SHA256withRSA`
- `SHA384withRSA`
- `SHA512withRSA`

8.3. FORMATO DE FIRMA

Formato de la firma electrónica que se desea realizar. Los valores permitidos son:

- `CAdES`
  - Formato de firma avanzada CAdES.
- `XAdES`
  - Formato de firma avanzada XAdES.
- `FacturaE`
  - Formato de firma para facturas electrónicas.
- `PAdES`
  - Formato de firma avanzada para documentos PDF.
- `CAdES-ASiC-S`
  - Formato de firma avanzada ASiC de tipo CAdES.
- `XAdES-ASiC-S`
  - Formato de firma avanzada ASiC de tipo XAdES.
- `NONE`
  - Firma PKCS#1.

8.4. PARÁMETROS DE FIRMA

Los parámetros de firma ayudan a ajustar la estructura o el contenido de las firmas electrónicas. Estos parámetros varían según el formato de firma elegido y se heredan de los parámetros de configuración del Cliente @firma. Si requiere que su firma electrónica incluya metadatos adicionales o una estructura de firma distinta a la por defecto, consulte el anexo Configuración de los formatos de firma o el manual del integrador del Cliente @firma.

8.5. FORMATO MEJORADO DE FIRMA

Es el formato de firma longevo al que se debe actualizar la firma después de generarse. La actualización de las firmas generadas con FIRe se realiza mediante la conexión con una instancia de la Plataforma @firma. Las firmas se actualizan después de haberlas generado, pero antes de devolverlas a la aplicación que la solicitó.

Los valores de este parámetro se heredan de la configuración del software Integr@ y son dependientes de la instancia de la Plataforma @firma a la que se conecte el componente central.

Cuando se solicita la actualización a un formato de firma, FIRe envía a actualizar la firma a la Plataforma @firma y comprueba que el resultado obtenido se corresponda con el formato de firma al que se ha pedido actualizar. Si la firma no se pudiese actualizar a ese formato y en su lugar se devolviese uno intermedio (que tuviese sello de tiempo, pero no la información de revocación, por ejemplo), FIRe lanzaría un error indicando que el formato recibido no era el esperado.

Tenga en cuenta qué, si la instancia de la Plataforma @firma a la que se conecta el componente central actualiza a los nuevos perfiles de firma (Plataforma @firma v6.2 y superiores), es posible que, por ejemplo, al solicitar una firma “ES-T” se le devuelva una firma “T-Level”. En ese caso, FIRe también lanzaría un error durante la actualización de firma, ya que el formato solicitado no es el que indica la Plataforma @firma que ha devuelto (aunque sean iguales).

Consulte cuáles son los perfiles de firma admitidos por la instancia de la Plataforma @firma de su componente central y utilice los perfiles correspondientes en su aplicación.

Los perfiles admitidos por la Plataforma @firma son:

- `ES-T`
  - Para la actualización del formato CAdES a CAdES-T.
  - Para la actualización del formato XAdES a XAdES-T.
  - A partir de la versión 6.2 de la Plataforma @firma, debe usarse “`T-LEVEL`”.
- `ES-C`
  - Para la actualización del formato CAdES a CAdES-C.
  - Para la actualización del formato XAdES a XAdES-C.
- `ES-X`
  - Para la actualización del formato CAdES a CAdES-X.
  - Para la actualización del formato XAdES a XAdES-X.
- `ES-X-1`
  - Para la actualización del formato CAdES a CAdES-X1.
  - Para la actualización del formato XAdES a XAdES-X1.
- `ES-X-2`

  - Para la actualización del formato CAdES a CAdES-X2.
  - Para la actualización del formato XAdES a XAdES-X2.
- `ES-X-L-1`
  - Para la actualización del formato CAdES a CAdES-XL1.
  - Para la actualización del formato XAdES a XAdES-XL1.
- `ES-X-L-2`
  - Para la actualización del formato CAdES a CAdES-XL2.
  - Para la actualización del formato XAdES a XAdES-XL2.
- `ES-LTV`
  - Para la actualización del formato PAdES a PAdES-LTV.
- `T-LEVEL`
  - Para la actualización del formato CAdES a T-LEVEL.
  - Para la actualización del formato XAdES a T-LEVEL.
  - Para la actualización del formato PAdES a T-LEVEL.
  - Sólo disponible a partir de la versión 6.2 de la Plataforma @firma.
- `LT-LEVEL`
  - Para la actualización del formato CAdES a LT-LEVEL.
  - Para la actualización del formato XAdES a LT-LEVEL.
  - Para la actualización del formato PAdES a LT-LEVEL.
  - Sólo disponible a partir de la versión 6.2 de la Plataforma @firma.
- `LTA-LEVEL`
  - Para la actualización del formato CAdES a LTA-LEVEL.
  - Para la actualización del formato XAdES a LTA-LEVEL.
  - Para la actualización del formato PAdES a LTA-LEVEL.
  - Sólo disponible a partir de la versión 6.2 de la Plataforma @firma.

8.6. SELECCIÓN DEL CERTIFICADO DE FIRMA

Las operaciones de firma realizadas a través de un proveedor en la nube de confianza sólo se podrán realizar comúnmente mediante certificados válidos emitidos por el custodio de las claves, por lo cual, los certificados que usaremos siempre serán de confianza.

En las operaciones de firma con certificados locales, sin embargo, comúnmente no podemos prever qué certificados tiene instalados el usuario. Para restringir el conjunto de certificados con los que el usuario puede firmar, podremos configurar a través del parámetro `extraParams` de los métodos de firma una serie de filtros con los cuales restringir los certificados que puede utilizar el usuario para firmar. Estos filtros son los que se definen en el manual del integrador del Cliente @firma correspondiente a la versión del Cliente @firma integrada en esta versión de FIRe. Puede consultar el extracto correspondiente de este manual en el anexo Configuración de los filtros de certificados locales.

9. CONFIGURACIÓN DE LOS PROVEEDORES Y EL COMPONENTE CENTRAL

FIRe permite a los usuarios de las aplicaciones que hacen uso de sus servicios el realizar firmas con un certificado local o con cualquier de los proveedores de firma en la nube que integra. Tanto el propio FIRe como cada uno de los proveedores que integra permiten configurar su comportamiento frente a las llamadas realizadas por cada aplicación.

Los métodos `sign`, `createBatchProcess` y `addDocumentToBatch` del componente distribuido incluyen el parámetro `config`, a través del cual las aplicaciones pueden establecer las propiedades que configurarán el comportamiento de FIRe y los distintos proveedores. Cada proveedor de firma tiene sus propios requisitos y es probable que necesiten distinta información por parte del usuario o el integrador. La aplicación deberá proporcionar las opciones de configuración para los conectores de todos los proveedores y, cada uno de ellos, tomará de la llamada los parámetros que considere oportunos e ignorará el resto.

Las cadenas de texto con las propiedades de configuración siempre deberán utilizar la codificación UTF-8.

9.1. CONFIGURACIÓN DEL COMPONENTE CENTRAL

La configuración que siempre interpreta el componente central, independientemente del proveedor seleccionado por el usuario, es:

- Métodos `sign` y `createBatchProcess`:
  - `certOrigin`: Origen del certificado de firma.
    - Si no se establece esta propiedad, se le permitirá al usuario seleccionar entre todos los proveedores configurados en el componente central de FIRe.
    - Si se establece el nombre de un proveedor y este está dado de alta en el componente central, se utilizará directamente el proveedor seleccionado, sin dar posibilidad al usuario de seleccionar cualquier otro. Por ejemplo: `certOrigin=test`
    - Si se establece un listado de proveedores separados por comas (‘,’), se filtrará la lista de proveedores configurados en el componente central para mostrar sólo aquellos que se haya indicado, respetando el orden con el que se han proporcionado. Por ejemplo: `certOrigin=local,clavefirma`
    - Para configurar el proveedor para el uso de certificados locales, instalados en el almacén del navegador o dispositivo criptográfico, se usará el nombre “`local`”. Consulte con el administrador de FIRe qué proveedores tiene dados de alta y el nombre con el que configurar cada uno de ellos.
    - El administrador de FIRe puede haber configurado que uno o más proveedores concretos son imprescindibles. En dicho caso, el usuario siempre podrá seleccionar estos proveedores, incluso cuando la aplicación no los incluya en su listado de proveedores o, incluso, si configuró un único proveedor.

- `appName`: Nombre de la aplicación.
  - Este nombre se usará en las páginas del componente central para informar al usuario de la aplicación que solicita la operación de firma. Este parámetro es opcional.
- `docManager`: Identificador del `DocumentManager` que se desea utilizar.
  - Si no se indica, se utilizará por el defecto.
  - Consulte el apartado Gestión de datos desde el componente central para más información.

9.2. CONFIGURACIÓN DEL CONECTOR DE FIRMA CON CERTIFICADO LOCAL

FIRe incorpora la función de firma con certificado local, que puede ser cualquier certificado instalado en el almacén del navegador utilizado por el usuario o un certificado en tarjeta inteligente.

Para configurar expresamente este conector se debe utilizar el nombre "`local`".

La configuración admitida por este proveedor es igual a la admitida por el conector de Cl@ve Firma y su simulador. De esta forma, un despliegue en el que se configuren las propiedades para el uso de esos conectores también funcionará correctamente cuando el usuario seleccione el uso de un certificado local. Adicionalmente a las propiedades de aquellos proveedores, el conector para la firma local admite la propiedad "`afirmaWS`", que permite configurar la carga de AutoFirma WebStart.

El listado completo de propiedades admitidas es:

- Métodos `sign`:
  - `redirectOkUrl`: URL a la que redirigir al usuario en caso de terminar correctamente la operación.
  - `redirectErrorUrl`: URL a la que redirigir al usuario en caso de error.
    - **Advertencia:** En futuras versiones de FIRe, se evolucionarán los conectores para admitir sólo una URL de redirección a la que se llevará al usuario en todos los casos. Las nuevas integraciones de FIRe deberían configurar la misma URL en caso de éxito y error para facilitar la integración de futuras versiones.
  - `docName`: Nombre del documento que se firma. Sirve para que se muestre el nombre del documento en la pantalla de firma. Este parámetro es opcional.
  - `docTitle`: Título del documento que se firma. Sirve para que se muestre el título del documento en la pantalla de firma. Este parámetro es opcional.
  - `afirmaWS`: Configura si, para realizarse la firma con AutoFirma, debe cargarse AutoFirma WebStart o utilizar la versión nativa de AutoFirma instalada en el sistema. Este parámetro es opcional. Si no se indica, se lanzará la aplicación configurada desde el componente central.
- Métodos `createBatchProcess`:
  - `redirectOkUrl`: URL a la que redirigir al usuario en caso de terminar correctamente la operación.
  - `redirectErrorUrl`: URL a la que redirigir al usuario en caso de error.

- **Advertencia:** En futuras versiones de FIRe, se evolucionarán los conectores para admitir sólo una URL de redirección a la que se llevará al usuario en todos los casos. Las nuevas integraciones de FIRe deberían configurar la misma URL en caso de éxito y error para facilitar la integración de futuras versiones.

- Método `addDocumentToBatch`:
  - `docName`: Nombre del documento que se firma. Sirve para que se muestre el nombre del documento en la pantalla de firma. Este parámetro es opcional.
  - `docTitle`: Título del documento que se firma. Sirve para que se muestre el título del documento en la pantalla de firma. Este parámetro es opcional.

IMPORTANTE: Para la firma con certificados locales a través del Cliente @firma, es necesario que el propio cliente se conecte a algunos servicios del componente central localizados en el contexto "`fire-signature/public`". Hay que tener en cuenta que el Cliente @firma valida los certificados SSL de los servicios a los que se conecta, así que, si el certificado SSL de este dominio no es válido, no se ha emitido para el dominio en el que se utiliza o si no está reconocido por defecto por Java, el Cliente no conectará con ellos. Esto es un tema a tratar por el administrador de FIRe. Sin embargo, es común que en entornos de desarrollo se utilicen certificados SSL no válidos o no reconocidos por Java. Para permitir que el Cliente @firma se conecte correctamente a esos servicios, el usuario que se conecte al servicio deberá dar de alta la URL del componente central de FIRe en el listado de sitios seguros del panel de configuración de Java. Por ejemplo, podría dar de alta "`https://DOMINIO/fire-signature/public/`" donde "`DOMINIO`" sería el dominio o IP en el que se ha realizado el despliegue. De esta forma, el Cliente @firma no tendrá problemas para acceder a los servicios del componente central.

9.3. CONFIGURACIÓN DEL CONECTOR DE CL@VE FIRMA

El conector de Cl@ve Firma permite acceder al servicio de Cl@ve Firma para realizar firmas con los certificados de firma de Cl@ve Permanente de la GISS.

Para configurar expresamente este conector se suele utilizar el nombre "`clavefirma`", pero este puede variar según lo haya establecido el administrador del componente central. Consulte con su administrador acerca del nombre de cada uno de los proveedores dados de alta en el sistema.

El proveedor de Cl@ve Firma, comúnmente, sólo estará disponible en el entorno de producción de FIRe. Para hacer pruebas, utilice el proveedor del simulador de Cl@ve Firma.

La configuración que puede establecer a través de cada uno de los métodos del componente distribuido es:

- Métodos `sign`:
  - `redirectOkUrl`: URL a la que redirigir al usuario en caso de terminar correctamente la operación.
  - `redirectErrorUrl`: URL a la que redirigir al usuario en caso de error.

        - **Advertencia:** En futuras versiones de FIRe, se evolucionarán los conectores para admitir sólo una URL de redirección a la que se llevará al usuario en todos los casos. Las nuevas integraciones de FIRe deberían configurar la misma URL en caso de éxito y error para facilitar la integración de futuras versiones.
    - `procedureName`: Nombre del procedimiento que se ejecuta. Es el código numérico SIA que se incluye en el campo "Procedimientos asociados" del formulario de alta de aplicaciones.
    - `docName`: Nombre del documento que se firma. Sirve para que se muestre el nombre del documento en la pantalla de firma. Este parámetro es opcional.
    - `docTitle`: Título del documento que se firma. Sirve para que se muestre el título del documento en la pantalla de firma. Este parámetro es opcional.
- Métodos `createBatchProcess`:
    - `redirectOkUrl`: URL a la que redirigir al usuario en caso de terminar correctamente la operación.
        - **Advertencia:** En futuras versiones de FIRe, se evolucionarán los conectores para admitir sólo una URL de redirección a la que se llevará al usuario en todos los casos. Las nuevas integraciones de FIRe deberían configurar la misma URL en caso de éxito y error para facilitar la integración de futuras versiones.
    - `redirectErrorUrl`: URL a la que redirigir al usuario en caso de error.
    - `procedureName`: Nombre del procedimiento que se ejecuta. Es el código numérico SIA que se incluye en el campo "Procedimientos asociados" del formulario de alta de aplicaciones.
- Método `addDocumentToBatch`:
    - `docName`: Nombre del documento que se firma. Sirve para que se muestre el nombre del documento en la pantalla de firma. Este parámetro es opcional.
    - `docTitle`: Título del documento que se firma. Sirve para que se muestre el título del documento en la pantalla de firma. Este parámetro es opcional.

Un ejemplo de configuración para la carga de datos cuando se usa el conector de Cl@ve Firma sería:

```
redirectOkUrl=URL_OK
redirectErrorUrl=URL_ERROR
procedureName=PROCEDIMIENTO
```

Según el componente distribuido y el método utilizado, esta información podría pasarse como un objeto de propiedades o como una cadena de texto. Para la configuración de estos parámetros a modo de cadena, concatenaríamos las propiedades separándolas con la partícula "`\n`":

```
redirectOkUrl=URL_OK\nredirectErrorUrl=URL_ERROR\nprocedureName=PROCEDIMIENTO
```

Esta cadena, se proporcionaría al método codificada en base 64:

```
cmVkaXJlY3RPa1VybD1VUkxfT0tcbnJlZGlyZWN0RXJyb3JVcmw9VVJMX0VSUk9SXG5wcm9jZWR1cmVOYW1lPVBST0NFRElNSUVOVE8=
```

9.4. CONFIGURACIÓN DEL CONECTOR SIMULADOR DE CL@VE FIRMA

FIRe dispone de un servicio que emula el comportamiento del proveedor Cl@ve Firma. Ya que comúnmente no es posible acceder a un entorno de desarrollo de Cl@ve Firma, los integradores podría utilizar este simulador para realizar las pruebas de comportamiento.

Para configurar expresamente este conector se suele utilizar el nombre "`clavefirmatest`", pero este puede variar según lo haya establecido el administrador del componente central. Consulte con su administrador acerca del nombre de cada uno de los proveedores dados de alta en el sistema.

El conector simulador de Cl@ve Firma admite las mismas opciones de configuración que el conector del auténtico Cl@ve Firma, a excepción del nombre de procedimiento ("`procedureName`") que no es necesario. Consulte el apartado anterior para ver las propiedades de configuración.

9.5. CONFIGURACIÓN DEL CONECTOR DE LA FNMT

Con FIRe se distribuye el conector para el acceso al proveedor de firma en la nube de la FNMT para empleados públicos.

Para configurar expresamente este conector se suele utilizar el nombre "`fnmt`", pero este puede variar según lo haya establecido el administrador del componente central. Consulte con su administrador acerca del nombre de cada uno de los proveedores dados de alta en el sistema.

La configuración que puede establecer a través de cada uno de los métodos del componente distribuido es:

- Métodos `sign`:
  - `redirectOkUrl`: URL a la que redirigir al usuario una vez terminado el proceso de firma. El usuario será redirigido a esta página sea cual sea el resultado.
- Métodos `createBatchProcess`:
  - `redirectOkUrl`: URL a la que redirigir al usuario una vez terminado el proceso de firma. El usuario será redirigido a esta página sea cual sea el resultado.

Al contrario que ocurre con el conector de Cl@ve Firma y el conector de firma local, este conector no permite configurar una URL a la que redirigir al usuario en caso de error. El usuario será dirigido a la URL indicada con "`redirectOkUrl`" sea cual sea el resultado. Si vamos a permitir el uso de este conector en nuestra aplicación, es recomendable que las URL de éxito y error configuradas sean la misma y desde esa URL identifiquemos si el resultado obtenido es realmente un resultado correcto o erróneo para proporcionar al usuario de la información adecuada.

9.6. CONFIGURACIÓN DE OTROS CONECTORES

Es posible que el administrador de FIRe dé de alta otros proveedores de firma en la nube. Dado el caso, consulte con él el nombre de estos proveedores y las opciones de configuración que admiten los conectores de los mismos.

10. GESTIÓN DE DATOS DESDE EL COMPONENTE CENTRAL

El funcionamiento por defecto de FIRe requiere que la aplicación cliente cargue los datos que desea firmar, los envíe al componente central para solicitar su firma y, posteriormente, recupere la firma electrónica generada para almacenarla o tratarla como corresponda. Sin embargo, FIRe también soporta el escenario en el que una aplicación cliente le indica al componente central qué datos son los que se deben firmar y es el propio componente central el que carga los datos y trata y guarda la firma para, finalmente, devolver un resultado a la aplicación que le indique como ha terminado el proceso. Como diferencias clave entre el uso del escenario ahora descrito y el por defecto están las siguientes ventajas y desventajas:

- Ventajas:
  - Se reduce significativamente el tráfico de datos entre las aplicaciones cliente y el componente central.
  - Cuando existen varias aplicaciones que realizan el mismo tratamiento de datos antes y después de firmar, se puede implementar este comportamiento una única vez en el componente central en lugar de hacerlo en cada una de las aplicaciones.
- Desventajas:
  - Se traslada al componente central la carga de procesar los datos y la firma, lo cual puede repercutir negativamente en su rendimiento.

Para el uso de este escenario es necesario implementar una “clase gestora de documentos”. Esta clase será la que defina de dónde se deben obtener los datos indicados por la aplicación cliente y cómo tratar y almacenar la firma resultante.

El desarrollo de la clase gestora de documentos deberá realizarlo el desarrollador de la aplicación cliente, ya que es el que conoce la lógica intrínseca de su aplicación y los sistemas en la que se encuentran los datos y se almacena la firma. Sin embargo, esta lógica deberá ejecutarse desde el componente central, para lo cual el administrador del sistema FIRe deberá desplegar y configurar esta clase de tal forma que el componente central tenga acceso a ella.

El componente central FIRe permite la configuración de múltiples clases gestoras de documentos por lo que una aplicación cliente podría utilizar una de ellas mientras que el resto usa otras o adoptan el escenario por defecto de FIRe (envían directamente los datos a firmar y recuperan la firma). No se permite definir más de una clase gestora para una operación, por lo que se utilizaría la última configurada. Tampoco es posible configurar una clase gestora para cargar los datos y otra distinta para guardar la firma.

A modo de guía, el integrador que desee conectarse con FIRe y utilizar una clase gestora de documentos deberá seguir los siguientes pasos:

1. Ponerse en contacto con el administrador del componente central FIRe al que desee conectarse, ya que este deberá valorar la conveniencia de hacer uso de una clase gestora, dar su visto bueno y, posiblemente, habilitar los accesos de red para que el componente central pueda comunicarse con los sistemas desde los que se cargarán los datos y en los que se guardarán las firmas generada.
2. Implementar la clase gestora de documentos. Para conocer los detalles sobre este punto consulte el apartado Implementación de una clase gestora de documentos.

3. Proporcionar al administrador de FIRe la clase gestora de documentos ya compilada, sus dependencias, su nombre completo, los ficheros de configuración necesarios y el listado de direcciones URL a las que debe acceder el componente central para cargar, tratar y almacenar los datos y la firma.
4. Adaptar la aplicación cliente para la selección de datos a firmar mediante su identificador y la recuperación de la información resultado de la operación de firma. Para hacer esto, siga las explicaciones del apartado Uso de una clase gestora de documentos.
5. Configurar en las operaciones de firma (`sign`) y creación de lotes (`createBatchProcess`) de la aplicación cliente el uso del gestor de documentos que desee utilizar. Para ello se deberá usar el nombre de gestor que le haya proporcionado el administrador del componente central. Para hacer esto siga las explicaciones del apartado Configuración de la clase gestora de documentos.

Una vez realizados estos pasos, la aplicación cliente debe funcionar correctamente y hacer uso de la clase gestora de documentos.

10.1. IMPLEMENTACIÓN DE UNA CLASE GESTORA DE DOCUMENTOS

Las clases gestoras de documentos deben implementarse en Java para poder utilizarse de forma nativa desde el componente central de FIRe. Esta clase gestora debe implementar la interfaz "`es.gob.fire.server.document.FIReDocumentManager`". Esta interfaz se encuentra definida en el archivo "`fire-signature-document.jar`", que se puede encontrar entre los componentes de FIRe.

La interfaz `FIReDocumentManager` define los siguientes métodos que será necesario implementar:

- `void init(Properties config) throws IOException;`
  - Método para la inicialización de la clase gestora. Se ejecutará una única vez antes del primer uso de alguno de sus otros métodos.
  - Recibe como parámetro las propiedades extraídas del fichero de configuración

- `byte[] getDocument(byte[] docId, String appId, String format, Properties extraParams) throws IOException;`
  - Método para la obtención del documento que se desea firmar.
  - Los parámetros recibidos en este método serán:
    - `docId`: Identificador de los datos que se deben recuperar. Es el resultado de decodificar el parámetro base 64 que se proporcionó en el parámetro de datos del método de firma del API distribuido de FIRe. Puede ser una cadena de texto que se reconstruya a partir del propio binario:
      - `String id = new String(docId,StandardCharsets.UTF_8);`
    - `appId`: Identificador de la aplicación que solicita el documento. Es el mismo identificador de aplicación utilizado frente al componente central de FIRe.

- `format`: Formato de firma.
- `extraParams`: Parámetros adicionales de firma indicados a través del método de firma del componente distribuido. Además de los parámetros soportados por FIRe para la configuración de la firma (los mismos que utiliza el Cliente @firma y se describen en su manual de integración), se pueden incluir parámetros personalizados que no entren en conflicto con ellos y que nos sirvan para proporcionar mayor información a la clase gestora de documentos.

- Esta función debe devolver el binario del documento a firmar.
- En caso de producirse un error al recuperar el documento, la clase gestora deberá lanzar una excepción de tipo `IOException`.

- `byte[] storeDocument(byte[] docId, String appId, byte[] data, X509Certificate cert, String format, Properties extraParams) throws IOException;`
  - Método para el tratamiento y guardado de la firma generada.
  - Los parámetros recibidos en este método serán:
    - `docId`: Identificador de los datos que se firmaron. Es el resultado de decodificar el parámetro base 64 que se proporcionó en el parámetro de datos del método de firma del API distribuido de FIRe. Puede ser una cadena de texto que se reconstruya a partir del propio binario:
      - `String id = new String(docId, StandardCharsets.UTF_8);`
    - `appId`: Identificador de la aplicación que solicitó el guardado de la firma. Es el mismo identificador de aplicación utilizado frente al componente central de FIRe.
    - `data`: Firma electrónica generada y actualizada. Son los datos que se deberán tratar y guardar.
    - `cert`: Certificado utilizado para firmar. **ADVERTENCIA:** En el caso de la operación de firma de lote con certificado cliente, no se recibirá este valor.
    - `format`: Formato de firma.
    - `extraParams`: Parámetros adicionales de firma indicados a través del método de firma del componente distribuido. Además de los parámetros soportados por FIRe para la configuración de la firma (los mismos que utiliza el Cliente @firma y se describen en su manual de integración), se pueden incluir parámetros personalizados que no entren en conflicto con ellos y que nos sirvan para proporcionar mayor información a la clase gestora de documentos.
  - Este método debe devolver una respuesta que, al recuperarse por medio del componente distribuido, permita a la aplicación conocer el resultado de la operación de firma. En caso de querer enviar una cadena de texto, podría decodificarse esta de la forma:
    - `return respuesta.getBytes(StandardCharsets.UTF_8);`

- En caso de producirse un error al tratar y guardar el documento, la clase gestora deberá lanzar una excepción de tipo `IOException`.

La clase gestora puede hacer uso de otras clases y recursos que se distribuyan junto a ella. También puede hacer uso de bibliotecas externas. Sin embargo, en este último caso, habrá que consultar con el administrador del despliegue de FIRe que las bibliotecas que deseamos utilizar no introducen incompatibilidades con FIRe, las bibliotecas del servidor de aplicaciones en el que se despliega o las bibliotecas de alguna otra clase gestora de documentos.

Tenga en cuenta que si desde su clase gestora accede a sistemas externos al componente central, deberá notificarlo al administrador de FIRe para que disponga el acceso a eso a los servicios y recursos necesarios.

10.2. USO DE UNA CLASE GESTORA DE DOCUMENTOS

En el caso en que una aplicación desee utilizar una clase gestora de documentos para que sea el componente central el que cargue los datos a firmar y trate y guarde las firmas, el integrador deberá indicar en las llamadas de firma y de creación de lote cuál es la clase gestora de documentos a utilizar. Esto se puede realizar a través del parámetro de configuración de ambos métodos, agregándoles la propiedad "`docManager`".

Así pues, si deseamos configurar que una operación de firma utilizase una clase gestora de documentos llamada "`BBDD`", deberíamos configurar:

```
Properties confProperties = new Properties();
confProperties.setProperty("redirectOkUrl", REDIRECT_SUCCESS_PAGE);
confProperties.setProperty("redirectErrorUrl", REDIRECT_ERROR_PAGE);
confProperties.setProperty("appName", "Aplicación prueba");
confProperties.setProperty("docManager", "BBDD");

FireClient client = new FireClient(appId);
SignOperationResult signResult = client.sign(
                        userId,
                        op,
                        format,
                        algorithm,
                        extraparams,
                        docIdBase64,
                        confProperties);
```

El nombre de la clase gestora de documentos la establece el administrador de FIRe al configurarla en el componente central. Tras hacerlo, deberá transmitir este nombre al integrador que desee utilizarla para que pueda configurarla.

Adicionalmente, cuando se utiliza una clase gestora de documentos, en lugar de proporcionar los datos en las funciones de firma (`sign`) y agregar documento a un lote (`addDocumentToBatch`), se deberán proporcionar los identificadores necesarios para que la clase gestora de documentos cargue el documento correspondiente. Este parámetro es de uso exclusivo de la clase gestora por lo que el integrador podrá proporcionar utilizar cualquier tipo de valor (siempre codificado en base 64) y utilizarlo (ya decodificado) en los métodos de la clase `FIReDocumentManager` implementada.

En el método de firma del API, el identificador se proporcionará en lugar del parámetro de datos, sin embargo, en los métodos de agregar documento a un lote, existen 2 posibilidades:

- Puede establecerse el identificador remoto del documento en base 64 como parámetro que sustituye a los datos.
    - Por ejemplo:

```
addDocumentToBatch("JHAS7-HAS2V-HVGSG", "11111111T", "1",
                   Base64.encode(id.getBytes()), null);
```

- Puede omitirse el dato, en cuyo caso se deberá proporcionar el identificador remoto del documento codificado en base 64 como identificador de documento dentro del lote (parámetro `documentId`). El identificador que se le proporcionará a la clase gestora será el valor decodificado de este parámetro.
    - Por ejemplo:

```
addDocumentToBatch("JHAS7-HAS2V-HVGSG", "11111111T",
                   Base64.encode(id.getBytes()), null, null);
```

El utilizar un único identificador para designar al documento dentro del lote y el documento que debe recuperar la clase gestora, simplifica la lógica. Sin embargo, puede darse el caso en el que se desee firmar dos veces un mismo documento, por ejemplo, para firmarlos con distinto formato. En ese caso, el identificador del documento para la clase gestora podría ser el mismo, pero no se puede utilizar dos veces el mismo identificador para dos documentos de un lote (aunque realmente los dos sean el mismo), ya que entonces no se sabría en el listado con el resultado a que firma corresponde cada uno de los resultados.

10.3. CONFIGURACIÓN DE LA CLASE GESTORA DE DOCUMENTOS

Toda clase gestora de documentos se puede inicializar con las propiedades de un fichero de configuración residente en el componente central.

El fichero de configuración debe seguir la estructura definida para los ficheros `Properties` de texto plano de Java (no `Properties` XML). Esto es un conjunto de entradas en forma "`clave=valor`" en donde cada una de ellas termina con un salto de línea. Por ejemplo:

```
MiPropiedad1=Hola
```

```
MiPropiedad2=Mundo!!
…
```

En caso de que la clase gestora necesite cualquier configuración que pueda variar según el entorno en el que se despliega (por ejemplo, el uso de un directorio para almacenar temporales o el acceso a servicios externos en distintos entornos) el desarrollador puede hacer que esta información se cargue desde el fichero de configuración y dar instrucciones al administrador de FIRe para que lo configure como sea necesario.

Las propiedades del fichero de configuración serán cargadas a través del método `init(Properties)` de la clase gestora de documentos. La carga se realizará una única vez y siempre antes del uso de cualquiera de los otros métodos de la clase gestora.

El desarrollador de la clase gestora deberá proporcionar el fichero de configuración al administrador del componente central para que este lo renombre y almacene junto con el resto de ficheros de configuración de FIRe. Si no se crea el fichero de configuración correspondiente a una clase gestora, el método de inicialización de la clase gestora recibirá un `null` por parámetro.

ANEXO I EJEMPLO DE APLICACIÓN CLIENTE

Con la distribución de FIRe se entrega, además de los componentes básicos del sistema, una serie de servicios y aplicaciones para ilustrar y facilitar el proceso de integración:

- Aplicación de pruebas (`fire-test-jsp.war`): Aplicación web que ilustra el uso del componente distribuido. Está pensado para servir únicamente de ejemplo de cómo integrar todo el sistema, no para su uso en producción. En esta aplicación se utiliza el componente distribuido Java.
- Servicios de pruebas (`clavefirma-test-service.war`): Conjunto de servicios que emulan el comportamiento de Cl@ve Firma, permitiendo a los desarrolladores hacer pruebas de sus aplicaciones sin necesidad de conectar con la pasarela de la GISS. La conexión con estos servicios de pruebas se puede realizar configurando el conector de pruebas en el fichero "`config.properties`" del componente central.

Adicionalmente, junto a FIRe se distribuye un servidor de aplicaciones con los distintos servicios desplegados y preparados para hacer pruebas. Consulte el apartado Despliegue de demostración sobre Apache Tomcat del manual de instalación y despliegue de FIRe para saber más sobre este despliegue.

I.1 USUARIOS DE PRUEBA

Los servicios de pruebas que emulan el comportamiento de Cl@ve Firma tienen dados de alta los siguientes usuarios:

Usuario con certificado válido

- Usuario: 00001
- Contraseña del almacén: 1111

Usuario sin certificado (permite la generación de uno)

- Usuario: 00002
- Contraseña del almacén: 1111

Usuario con certificado bloqueado

- Usuario: 00003

Usuario con registro no fehaciente /débil

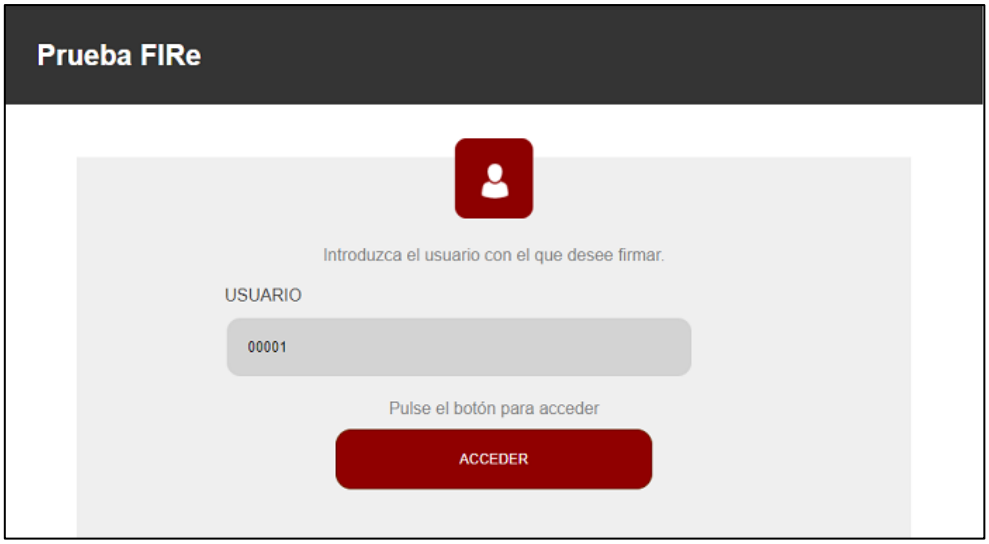
- Usuario: 00004

Por defecto, para un despliegue local, podrá acceder a la aplicación de pruebas desde la URL *https://localhost:8443/fire-test-jsp/Login.jsp*

NOTA: Tenga en cuenta que AutoFirma (utilizado para las firmas con certificados locales), por motivos de seguridad, no permite su invocación cuando se realiza desde `localhost` o `127.0.0.1`. Para hacer pruebas en local es preferible que utilice su IP de red o que establezca un alias para `127.0.0.1` en el fichero `hosts` de su equipo.

I.2 GUÍA DE LA PÁGINA DE PRUEBA

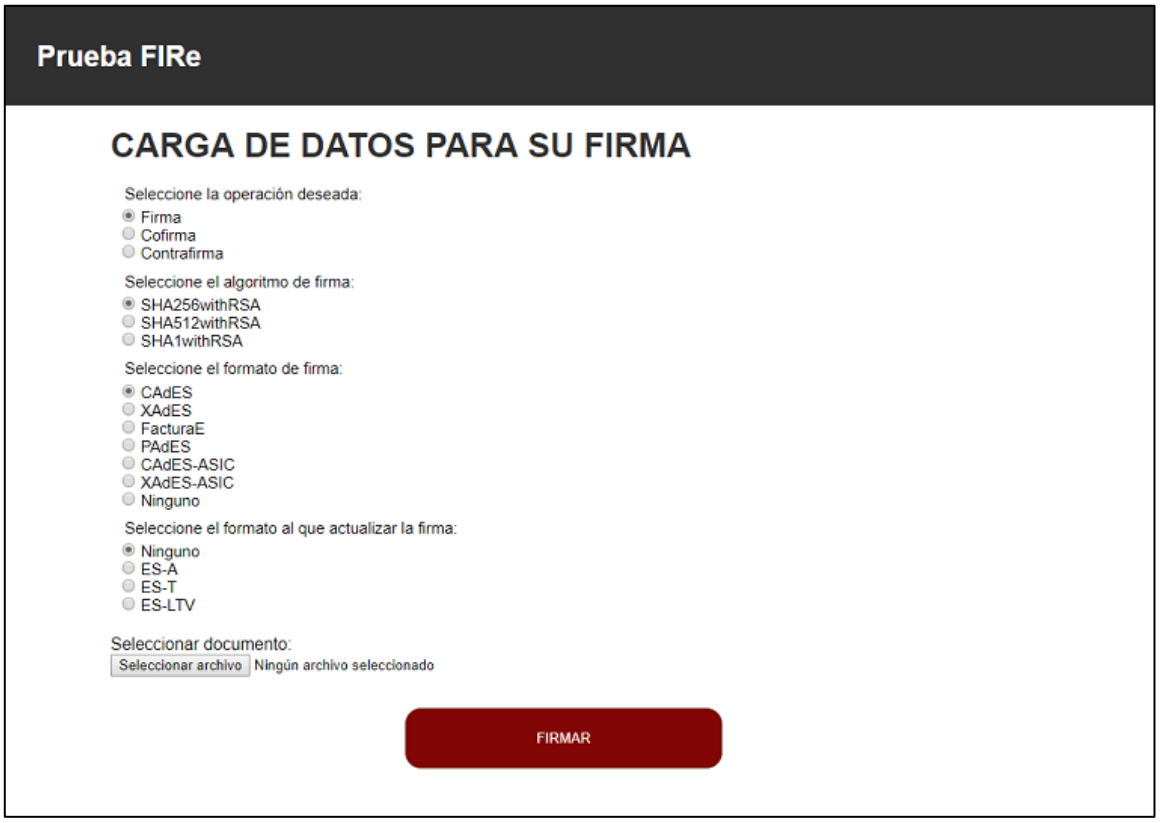
La página de pruebas requiere que el usuario inserte nombre de usuario. Si se ha configurado el acceso real a Cl@ve Firma o cualquier otro operador de firma en la nube, se deberá utilizar el DNI de la persona. Si se ha configurado el uso de los servicios de prueba, se deberá utilizar cualquiera de los usuarios del apartado anterior.



Una vez accedemos con el usuario deseado, se nos presentará una página en la que podremos elegir si queremos probar una operación de firma simple o una operación de firma de lote:

I.2.1 OPERACIÓN DE FIRMA

En el caso de la operación de firma simple, se nos presentará una página desde la que elegir las opciones de firma: operación, algoritmo, formato y formato al que actualizar (para usar esto último será necesario configurar la conexión con la Plataforma @firma); y poder seleccionar el fichero a cargar:



Al configurar las opciones deseadas y pulsar el botón “Firmar”, se realiza la llamada al método `sign` del API del cliente FIRe. A este se le proporcionan todas las opciones seleccionadas excepto el formato de actualización que se guarda para después usarlo en el método de recuperación de firma. También se le pasan las opciones de configuración del formato de firma (la aplicación de prueba siempre utiliza las mismas) y las opciones de configuración para el componente central y los conectores de los distintos proveedores (consulte el apartado Configuración de los proveedores y el componente central). Entre estas propiedades se encuentran las URL a las que redirigir en caso de éxito o error en la operación de firma. Este método nos devuelve un identificador de transacción y una URL a la que redirigir al usuario.

La URL obtenida será de una página del componente central en la que se podrá elegir el origen del certificado (proveedores de firma en la nube configurados y/o certificado local). En caso de haber seleccionado en la llamada un origen de certificado concreto, se omitiría esta nueva pantalla.

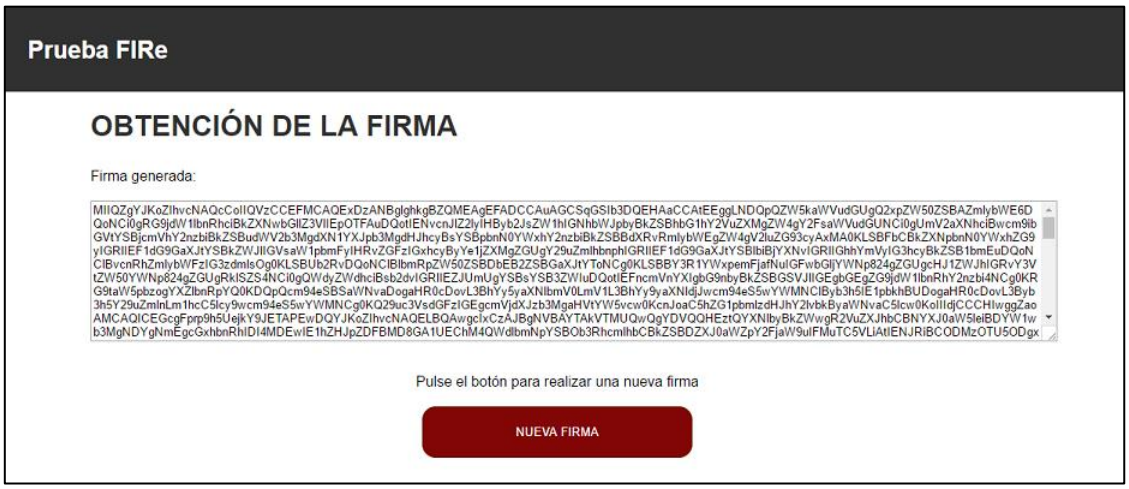
En caso de haber seleccionado el uso de los certificados de un proveedor de firma en la nube, seremos re-dirigidos a otra página del componente central que nos permitirá seleccionar el certificado de firma que deseamos utilizar:

A continuación, se nos redirigiría a la pasarela del proveedor de firma en la nube. En esta página, podemos insertar la clave del almacén en la nube del usuario (y clave OTP de ser necesario) para autorizar la operación de firma. En caso de seleccionar el proveedor de los servicios de prueba, utilice la contraseña asociada al usuario insertado inicialmente.

En caso de haber seleccionado el uso de un certificado local, seremos redirigidos a otra página del componente central en la que se cargará el Cliente @firma y se nos mostrará un botón de firma con el que iniciar la operación.

Independientemente de que se utilice un certificado de un proveedor en la nube o un certificado local, al finalizar correctamente la operación seremos redirigidos a la página de nuestra aplicación que hubiésemos indicado en la configuración proporcionada al método de firma. En esta página recuperamos la firma resultante utilizando el método `recoverSignResult` del API, al que se le debe proporcionar el identificador de transacción, el identificador del usuario y, si aplica, el formato de firma al que actualizar.

En la aplicación de prueba se muestra este resultado imprimiendo su base 64 en la propia página.



I.2.2 OPERACIÓN DE FIRMA DE LOTE

Cuando se solicita la operación de firma de lote, se nos presentará una página desde la que elegir las opciones de firma: operación, algoritmo, formato y formato de actualización. Esta será la configuración general que debe aplicar al lote.

Al pulsar el botón “Agregar documentos al lote” se llamará a la función del API `createBatchProcess`, a la que se le pasará la configuración de firma establecida en la página, además de la configuración propia de FIRe y los proveedores (consulte el apartado Configuración de los proveedores y el componente central), entre la que estarán las URL de las páginas de nuestra aplicación a las que redirigir en caso de éxito o error en la operación. Como resultado, se devolverá el identificador de la transacción de firma de lote.

A continuación, seremos redirigidos a una pantalla desde la que poder agregar un documento al lote.

En esta pantalla deberemos introducir un identificador de documento, que puede ser cualquier cadena y será con el que se identificará la firma del propio documento, y el fichero que deseemos firmar. Opcionalmente, se puede habilitar la casilla “Configuración particular” para establecer los valores necesarios para que esta firma se firme en base a esos valores y no los establecidos en el método de creación del lote. Al pulsar el botón “Agregar documento al lote” se llamará a la función del API `addDocumentToBatch` para agregar el documento el lote.

Seguidamente se volverá a cargar la misma página web, a la que se habrá agregado el botón “Firmar lote” y “Detener en caso de error”. Así, podremos seguir agregando tantos documentos como deseemos al lote.

En el momento de pulsar el botón “Firmar lote” se llamará a la función del API `signBatch` indicando el identificador de transacción del lote y si debe detenerse la ejecución en caso de detectarse un error. Como resultado, obtendremos una URL.

La URL obtenida será de una página del componente central en la que se podrá elegir el origen del certificado (certificado de Cl@ve Firma/servicios de pruebas o certificado local). En caso de haber seleccionado en la llamada un origen de certificado concreto, se omitiría esta pantalla.

En caso de haber seleccionado el uso de los certificados de Cl@ve Firma, seremos redirigidos a otra página del componente central que nos permitirá seleccionar el certificado que deseamos utilizar:

A continuación, se nos redirigiría a la pasarela de la GISS o una página similar de los servicios de prueba si son estos los que se han configurado. En esta página, podemos insertar la clave del almacén en la nube del usuario (y clave OTP de ser necesario) para autorizar la operación de firma.

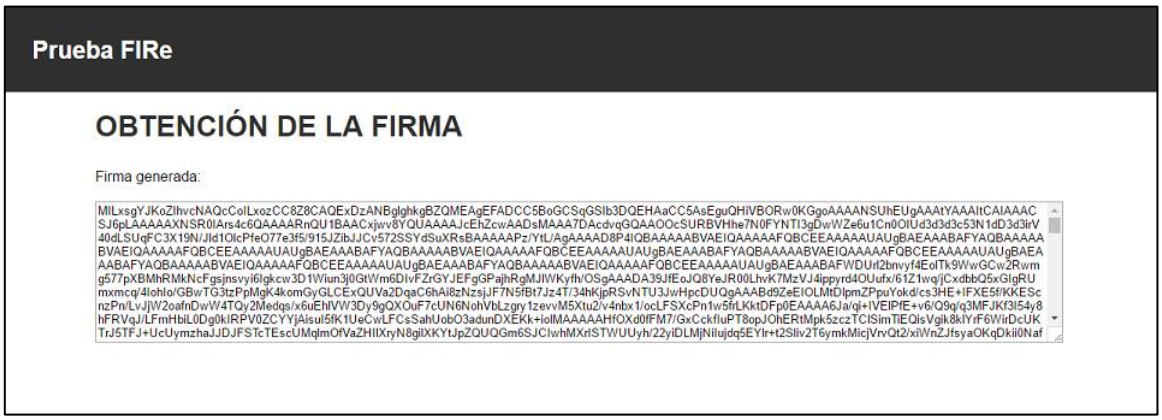
En caso de haber seleccionado el uso de un certificado local, seremos redirigidos a otra página del componente central en la que se cargará el Cliente @firma y se nos mostrará un botón de firma con el que iniciar la operación.

Independientemente de que se utilice un certificado de Cl@ve Firma/Servicio de pruebas o un certificado local, al finalizar correctamente la operación seremos redirigido a la página de nuestra aplicación que hubiésemos indicado en la configuración proporcionada al método de creación del lote. En esta página recuperamos el resultado del procesado del lote utilizando el método `recoverBatchResult` del API, al que se le debe proporcionar el identificador de transacción.

En la página de pruebas se ilustra este resultado mediante una pantalla como la que sigue, en la que se muestran los identificadores de los documentos, como finalizaron las firmas de esos documentos y, si finalizaron correctamente, un enlace para la descarga de la firma.

A modo de ejemplo, después de la llamada al método de recuperación del lote, se inicia a llamar reiteradamente al método `recoverBatchResultState` del API para conocer el estado del proceso de firma del lote. Hasta que no detectamos que este método devuelve el resultado "1", mostraremos un diálogo de carga en el que se mostrará el estado generar de avance del lote. Este proceso de comprobación del estado del lote es opcional.

Al pulsar sobre el enlace "Recuperar firma" de alguna de las firmas generadas se abrirá una nueva página desde la que se llama a la función `recoverBatchSign` del API indicándole el identificador del documento del que queremos recuperar la firma. Como resultado, nos proporcionará la firma del documento indicado del lote. La página de prueba mostrará esta firma mediante una pantalla como la que sigue:



ANEXO II CÓDIGOS DE ERROR DE LAS FIRMAS DE UN LOTE

Los mensajes que pueden devolverse como resultado de la firma de un documento de lote cuando se produce un error, son los siguientes:

- **NO_PROCESSED**: La firma del lote se detuvo antes de procesar este documento.
- **DATA_NOT_FOUND**: Error al cargar el documento.
- **PRESIGN_ERROR**: Error durante la prefirma del documento.
- **POSTSIGN_ERROR**: Error durante la postfirma del documento.
- **UPGRADE_ERROR**: Error al actualizar la firma al formato avanzado solicitado.
- **ERROR_SAVING_DATA**: Error durante el guardado de la firma.
- **INVALID_SIGNATURE_OPERATION**: Se configuró una operación de firma no válida.
- **ABORTED**: Se abortó la operación al detectar un error en otro documento de un lote. Este estado podría establecerse incluso después de haber terminado correctamente la firma del documento.
- **ERROR_RECOVERING**: Error al recuperar una firma generada.

ANEXO III CÓDIGOS DE ERROR

A continuación, se listan los distintos códigos de error que pueden devolver las operaciones de FIRe, así como su mensaje asociado y la posible causa del error:

Código	Mensaje	Motivo
1	El usuario no está dado de alta en el sistema	El usuario trató de acceder a sus certificados de firma en la nube pero no estaba dado de alta en proveedor seleccionado (Cl@ve Firma, *backend* de pruebas…)
2	Estado inválido	No se encontró en la sesión algún dato imprescindible para continuar la operación o se ha alcanzado algún punto de error al que no se debería haber llegado sin que hubiese detenido la ejecución otro error anterior. Es probable que este error derive de un error previo.
3	Error desconocido durante la operación	Se ha producido un error pero no se ha podido identificar cuál.
4	Operación cancelada por el usuario	El usuario pulsó alguno de los botones de cancelar que se muestran en las páginas de FIRe.
6	La sesión no es válida o ha caducado	Se ha indicado el identificador de una transacción que no existe, que ya ha terminado o que caducó.
7	Error interno del servidor	Se ha producido un error del componente central. Comúnmente, será un problema de configuración o conexión con sistemas externos. Consulte con el administrador del componente central.
8	Error detectado después de llamar a la pasarela externa	Se produjo un error fuera del contexto de FIRe después de redirigir al usuario a la pasarela de alguno de los proveedores de firma en la nube. Esto puede deberse a un error de la propia pasarela o un comportamiento válido en respuesta a una acción del usuario (como pulsar el botón cancelar en la página de inserción de PIN).
101	Error en la obtención de los certificados	No se pueden obtener los certificados del usuario del proveedor por un problema indeterminado.
102	Error al conectar con el servicio para la recuperación de certificados	No se puede conectar con el servicio del proveedor al intentar obtener los certificados del usuario.
103	Los certificados del usuario están bloqueados	Los certificados del usuario están bloqueados. Según el proveedor, esto puede requerir esperar un tiempo determinado o que el usuario realice alguna acción directamente contra la entidad que los gestiona.
104	El usuario no puede poseer certificados de firma por	El usuario no tiene certificados emitidos por el proveedor seleccionado, ni podrá tenerlos hasta que no se cumpla algu-

	haber realizado un registro débil	na acción ajena a la aplicación. Por ejemplo, personarse en las oficinas de expedición de la entidad emisora para que se verifique su identidad, cumplir la mayoría de edad, etc.
105	El usuario no tiene certificados de firma y el conector no permite generarlos al vuelo	El usuario no tiene certificados emitidos por el proveedor seleccionado y el conector de este no permite generarlos al vuelo. Es posible que el usuario deba personarse en alguna oficina para expedirlo, que tenga que hacerlo a través de alguna herramienta web o que se deba hacer a través del conector integrado en FIRe pero no se haya configurado en este.
201	Error en la obtención de la firma de los datos	Se produce un error desconocido durante una operación de firma. Sólo se produce cuando ocurre un error relacionado con el proveedor de certificados.
202	Error al ejecutar la prefirma de los datos	Comúnmente se debe a una configuración de firma errónea. Por ejemplo, solicitar una firma PAdES sobre un documento no PDF. Sólo se produce cuando la firma se realiza con certificado en la nube.
205	Error al conectar con el servicio para la generación de la firma con la clave remota	No se puede conectar con el proveedor de los certificados al intentar realizar una operación de firma.
250	[Mensaje variable]	Ocurre cuando se produce un error al realizar una firma local con el Cliente @firma y el usuario no podía seleccionar otro proveedor, comúnmente porque la aplicación indicó que sólo se podía realizar la firma con un certificado local.

ANEXO IV CONFIGURACIÓN DE LOS FORMATOS DE FIRMA

En este apartado se reproducen los listados de propiedades que se pueden utilizar en el parámetro `extraParams` de los métodos `sign`, `createBatchProcess` y `addDocumentToBatch` para cada uno de los formatos de firma soportados. Estos parámetros se heredan de las bibliotecas del Cliente @firma y permiten configurar las firmas generadas, pero no son obligatorios en ningún caso. Para obtener más información sobre el significado de estos parámetros, consulte el manual del integrador del Cliente @firma.

IV.1 FORMATO CADES

IV.1.1 FIRMA Y COFIRMA

Nombre del parámetro	Valores posibles	Descripción
`mode`	`explicit`	La firma resultante no incluirá los datos firmados. Si no se indica el parámetro `mode` se configura automáticamente este comportamiento.
	`implicit`	La firma resultante incluirá internamente una copia de los datos firmados. El uso de este valor podría generar firmas de gran tamaño.
`contentTypeOid`	OID	Identificador del tipo de dato firmado.
`contentDescription`	[Texto]	Descripción textual del tipo de datos firmado.
`policyIdentifier`	[OID o URN de tipo OID]	Identificador de la política de firma, necesario para generar firmas CAdES-EPES.
`policyIdentifierHash`	[Base64]	Huella digital de la política de firma. Es obligatorio indicar este parámetro si de indicó también `policyIdentifier`, al igual que es obligatorio también dar valor al parámetro `policyIdentifierHashAlgorithm`.
`policyIdentifierHashAlgorithm`	`SHA1`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA1.
	`SHA-256`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-256.
	`SHA-384`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-284.
	`SHA-512`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-512.
`policyQualifier`	[URL hacia documento]	URL (universalmente accesible) hacia el documento (normalmente PDF) que contiene una descripción textual de la política

		de firma. Este parámetro es opcional incluso si se desea generar firmas CAdES-EPES.
includeOnlySignningCertificate	true	Indica que debe incluirse en la firma únicamente el certificado del firmante.
	false	Indica que debe incluirse en la firma toda la cadena de certificación del certificado firmante. Valor por defecto.
signatureProductionCity	[Texto]	Agrega a la firma un campo con la ciudad en la que se realiza la firma. La codificación debe ser UTF-8.
signatureProductionPostalCode	[Texto]	Agrega a la firma un campo con el código postal en donde se realiza la firma. La codificación debe ser UTF-8.
signatureProductionCountry	[Texto]	Agrega a la firma un campo con el país en la que se realiza la firma. La codificación debe ser UTF-8.
commitmentTypeIndications	[Entero]	Indica el número de CommitmentTypeIndications que se van a declarar. Estos son los motivos que se declaran para la firma. Los valores concretos se especifican con commitmentTypeIndication***n***Identifier y commitmentTypeIndication***n***Description, donde '***n***' va desde 0 hasta el valor indicado en esta propiedad menos 1.
commitmentTypeIndication***n***Identifier	1	Establece que el CommitmentTypeIndications número ***n*** (contando desde cero) es "**Prueba de origen**".
	2	Establece que el CommitmentTypeIndications número ***n*** (contando desde cero) es "**Prueba de recepción**".
	3	Establece que el CommitmentTypeIndications número ***n*** (contando desde cero) es "**Prueba de entrega**".
	4	Establece que el CommitmentTypeIndications número ***n*** (contando desde cero) es "**Prueba de envío**".
	5	Establece que el CommitmentTypeIndications número ***n*** (contando desde cero) es "**Prueba de aprobación**".
	6	Establece que el CommitmentTypeIndications número ***n*** (contando desde cero) es "**Prueba de creación**".
commitmentTypeIndication***n***CommitmentTypeQualifiers	[Texto]	Lista de indicadores textuales separados por el carácter '|' que se aportan como calificadores adicionales del CommitmentTypeIndication número n (atributo opcional). Normalmente son OID. Los elementos de la lista no pueden contener el carácter '|' (ya que este se usa como separador).
signingCertificateV2	[Booleano]	Si se indica a true se utilizará SigningCertificateV2, si se indica cualquier otra cosa

		SigningCertificateV1. Si no se indica nada, se utilizará V1 para las firmas SHA1 y V2 para el resto.

IV.1.2 CONTRAFIRMA

Nombre del parámetro	Valores posibles	Descripción
`policyIdentifier`	[OID o URN de tipo OID]	Identificador de la política de firma, necesario para generar firmas CAdES-EPES.
`policyIdentifierHash`	[Base64]	Huella digital de la política de firma. Es obligatorio indicar este parámetro si de indicó también `policyIdentifier`, al igual que es obligatorio también dar valor al parámetro `policyIdentifierHashAlgorithm`.
`policyIdentifierHashAlgorithm`	`SHA1`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA1.
	`SHA-256`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-256.
	`SHA-384`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-284.
	`SHA-512`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-512.
`policyQualifier`	[URL]	URL (universalmente accesible) hacia el documento (normalmente PDF) que contiene una descripción textual de la política de firma. Este parámetro es opcional incluso si se desea generar firmas CAdES-EPES.
`includeOnlySignningCertificate`	`true`	Indica que debe incluirse en la firma únicamente el certificado del firmante.
	`false`	Indica que debe incluirse en la firma toda la cadena de certificación del certificado firmante. Valor por defecto.
`signatureProductionCity`	[Texto]	Agrega a la firma un campo con la ciudad en la que se realiza la firma. La codificación debe ser UTF-8.
`signatureProductionPostalCode`	[Texto]	Agrega a la firma un campo con el código postal en donde se realiza la firma. La codificación debe ser UTF-8.
`signatureProductionCountry`	[Texto]	Agrega a la firma un campo con el país en la que se realiza la firma. La codificación debe ser UTF-8.
`commitmentTypeIndications`	`[Entero]`	Indica el número de CommitmentTypeIndications que se van a declarar. Estos son los motivos que se declaran para la firma. Los valores concretos se especifican con `commitmentTypeIndication`***n***`Identifier` y `commitmentTypeIndication`***n***`Description`, donde '***n***' va desde 0 hasta el valor menos 1 indicado en esta propiedad.

commitmentTypeIndication*n*Identifier	1	Establece que el CommitmentTypeIndication número *n* es "**Prueba de origen**".
	2	Establece que el CommitmentTypeIndication número *n* es "**Prueba de recepción**".
	3	Establece que el CommitmentTypeIndication número *n* es "**Prueba de entrega**".
	4	Establece que el CommitmentTypeIndication número *n* es "**Prueba de envío**".
	5	Establece que el CommitmentTypeIndication número *n* es "**Prueba de aprobación**".
	6	Establece que el CommitmentTypeIndication número *n* es "**Prueba de creación**".
commitmentTypeIndication*n*CommitmentTypeQualifiers	[OID]	Lista de OID separados por el caracter '|' que se aportan como calificadores adicionales del CommitmentTypeIndication número n (atributo opcional).

IV.2 FORMATO XADES

IV.2.1 FIRMA Y COFIRMA

Nombre del parámetro	Valores posibles	Descripción
insertEnvelopedSignatureOnNodeByXPath	[Texto (expresión XPath v1)]	Indica, mediante una expresión XPath (v1), el nodo bajo el cual debe insertarse el nodo de firma en el caso de una firma *Enveloped*. Si la expresión devuelve más de un nodo, se usa solo el primero. Si la expresión no devuelve nodos o está mal construida se lanzará una excepción. Este parámetro solo tiene efecto en firmas *Enveloped*.
useManifest	true	Usa un Manifest de XMLDSig con las referencias de firma en vez de firmar directamente estas referencias. Esto permite que sea opcional la comprobación del destino y huellas digitales de las referencias.
	false	Genera las firmas normalmente, sin Manifest (comportamiento por defecto)
addKeyInfoKeyValue	true	Incluye el nodo KeyValue dentro de KeyInfo de XAdES (comportamiento por defecto).
	false	No incluye el nodo KeyValue dentro de KeyInfo de XAdES.
addKeyInfoKeyName	true	Incluye el nodo KeyName dentro de KeyInfo de XAdES.
	false	No incluye el nodo KeyName dentro de KeyInfo de XAdES (comportamiento por defecto).
avoidXpathExtraTransformsOnEnveloped	true	Evita la inclusión de la transformación XPATH2 que normalmente se añade para posibilitar las

		cofirmas y que elimina todas las firmas del documento para dejar únicamente el contenido. **ADVERTENCIA:** La cofirma de un documento en el que al menos una de las firmas no incluye la transformación XPATH, dará lugar a un documento de firma que potencialmente será validado incorrectamente por los validadores de firma. Por este motivo, sólo se permite el uso de este parámetro en la operación de firma (no en la de cofirma).
	`false`	Incluye la transformación XPATH2 posibilita las cofirmas eliminando todas las firmas del documento para dejar únicamente el contenido (comportamiento por defecto).
`format`	`XAdES Enveloping`	Genera firmas en formato *Enveloping*. Este es el formato que se utiliza por defecto cuando no se indica ninguno.
	`XAdES Enveloped`	Genera firmas en formato *Enveloped*.
	`XAdES Detached`	Genera firmas en formato *Internally Detached*.
`includeOnlySignningCertificate`	`true`	Indica que debe incluirse en la firma únicamente el certificado del firmante.
	`false`	Indica que debe incluirse en la firma toda la cadena de certificación del certificado firmante. Valor por defecto.
`policyIdentifier`	[URL]	Identificador de la política de firma (normalmente una URL hacia la política en formato XML procesable), necesario para generar firmas XAdES-EPES.
`policyIdentifierHash`	[Base64]	Huella digital de la política de firma. Es obligatorio indicar este parámetro si el valor indicado en `policyIdentifier` no es universalmente accesible. Si se da valor a este parámetro es obligatorio también dar valor al parámetro `policyIdentifierHashAlgorithm`.
`policyIdentifierHashAlgorithm`	`SHA1`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA1.
	`SHA-256`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-256.
	`SHA-384`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-384.
	`SHA-512`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-512.
`policyQualifier`	[URL hacia documento]	URL (universalmente accesible) hacia el documento (normalmente PDF) que contiene una descripción textual de la política de firma. Este parámetro es opcional incluso si se desea generar firmas XAdES-EPES.
`policyDescription`	[Texto]	Descripción textual de la política de firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado. Este parámetro es opcional incluso si se desea generar firmas XAdES-EPES.

signerClaimedRoles	[Texto]	Agrega a la firma campos con los cargos atribuidos al firmante. Deben separarse los cargos con el carácter "|" (y este no puede estar en el propio texto de ningún cargo). En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionCity	[Texto]	Agrega a la firma un campo con la ciudad en la que se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionProvince	[Texto]	Agrega a la firma un campo con la provincia en la que se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionPostalCode	[Texto]	Agrega a la firma un campo con el código postal en donde se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionCountry	[Texto]	Agrega a la firma un campo con el país en el que se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
referencesDigestMethod	http://www.w3.org/2000/09/xmldsig#sha1	Usa el algoritmo SHA1 para el cálculo de las huellas digitales de las referencias XML firmadas.
	http://www.w3.org/2001/04/xmlenc#sha256	Usa el algoritmo SHA-256 para el cálculo de las huellas digitales de las referencias XML firmadas. Este es el valor recomendado.
	http://www.w3.org/2001/04/xmlenc#sha512	Usa el algoritmo SHA-512 para el cálculo de las huellas digitales de las referencias XML firmadas.
mimeType	[Texto en formato MIME-Type]	MIME-Type de los datos a firmar. Si no se indica este parámetro el sistema intenta auto-detectar el tipo, estableciendo el más aproximado (que puede no ser el estrictamente correcto).
encoding	[URI]	Codificación de los datos a firmar (ver la documentación del elemento Object de XMLDSig para más información). Un uso incorrecto de este parámetro puede provocar la generación de una firma inválida. Si se proporcionan datos a firmar previamente codificados en base 64 pero se desea sean considerados como su forma descodificada, debe establecerse este valor a http://www.w3.org/2000/09/xmldsig#base64 y especificarse el tipo real en el parámetro mimeType. Por ejemplo, para firmar una imagen PNG haciendo que la firma se refiera a su forma binaria directa, puede proporcionarse la imagen directamente codificada en base 64 indicando el encoding como

		`http://www.w3.org/2000/09/xmldsig#base64` y el `mimeType` como `image/png`. El valor debe ser siempre una URI.
`outputXmlEncoding`	[Texto]	Codificación del XML de salida. Si no se indica este valor se intenta autodetectar a partir del XML de entrada (si los datos a firmar son un XML).
`contentTypeOid`	[OID o URN de tipo OID]	Identificador del tipo de dato firmado. Este parámetro es complementario (que no excluyente) al parámetro `mimeType`.
`canonicalizationAlgorithm`	`http://www.w3.org/TR/2001/REC-xml-c14n-20010315`	Se firma el XML con canonizado XML 1.0 inclusivo (valor por defecto).
	`http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments`	Se firma el XML con canonizado XML 1.0 inclusivo con comentarios.
	`http://www.w3.org/2001/10/xml-exc-c14n#`	Se firma el XML con canonizado XML 1.0 exclusivo.
	`http://www.w3.org/2001/10/xml-exc-c14n#WithComments`	Se firma el XML con canonizado XML 1.0 exclusivo con comentarios.
`xadesNamespace`	[URL]	URL de definición del espacio de nombres de XAdES (el uso de este parámetro puede condicionar la declaración de versión de XAdES). Si se establece este parámetro es posible que se necesite establecer también el parámetro `signedPropertiesTypeUrl` para evitar incoherencias en la versión de XAdES.
`signedPropertiesTypeUrl`	[URL]	URL de definición del tipo de las propiedades firmadas (*Signed Properties*) de XAdES. Si se establece este parámetro es posible que se necesite establecer también el parámetro `xadesNamespace` para evitar incoherencias en la versión de XAdES. Si no se establece se usa el valor por defecto: `http://uri.etsi.org/01903#SignedProperties`.
`ignoreStyleSheets`	`true`	Si se firma un XML con hojas de estilo, ignora éstas dejándolas sin firmar.
	`false`	Si se firma un XML con hojas de estilo, firma también las hojas de estilo (valor por defecto, consultar notas adicionales sobre firma de hojas de estilo).
`avoidBase64Transforms`	`true`	No declara transformaciones base 64 incluso si son necesarias.
	`false`	Declara las transformaciones base 64 cuando se han codificado internamente los datos a firmar en base 64 (valor por defecto).
`headless`	`true`	Evita que se muestren diálogos gráficos adicionales al usuario (como por ejemplo, para la *dereferenciación* de hojas de estilo enlazadas con rutas relativas).

	false	Permite que se muestren diálogos gráficos adicionales al usuario.
xmlTransforms	[Número]	Número de transformaciones a aplicar al contenido firmado. Debe indicarse posteriormente igual número de parámetros xmlTransformnType, sustituyendo *n* por un ordinal consecutivo, comenzando en 0 (ver notas adicionales sobre indicación de transformaciones adicionales).
xmlTransform***n***Type	http://www.w3.org/2000/09/xmldsig#base64	Indica que los datos que se proporcionan para firmar ya están codificados en base 64 y se debe declarar esta transformación adicional para que se decodifiquen antes de firmarlos. Esta transformación base 64 es adicional a la transformación necesaria para pasar los datos a través de los métodos de firma del cliente.
	http://www.w3.org/TR/1999/REC-xpath-19991116	El contenido se debe procesar mediante esta transformación XPATH antes de ser firmado. Únicamente es aplicable cuando se firma contenido XML.
	http://www.w3.org/2002/06/xmldsig-filter2	El contenido se debe procesar mediante esta transformación XPATH2 antes de ser firmado. Únicamente es aplicable cuando se firma contenido XML.
xmlTransform***n***Subtype	[Texto]	Subtipo de la transformación ***n***. Los valores aceptados y sus funcionalidades dependen del valor indicado en xmlTransformnType.
xmlTransform***n***Body	[Texto]	Cuerpo de la transformación ***n***. Los valores aceptados y sus funcionalidades dependen del valores indicados en xmlTransform***n***Type y en xmlTransform***n***Subtype.
nodeToSign	[Texto]	Identificador del nodo (establecido mediante el atributo "Id") que se desea firmar dentro de un XML.
commitmentTypeIndications	[Entero]	Indica el número de CommitmentTypeIndications que se van a declarar. Estos son los motivos que se declaran para la firma. Los valores concretos se especifican con commitmentTypeIndication***n***Identifier y commitmentTypeIndication***n***Description, donde '***n***' va desde 0 hasta el valor menos 1 indicado en esta propiedad.
commitmentTypeIndication***n***Identifier	1	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de origen**".
	2	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de recepción**".
	3	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de entrega**".
	4	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de envío**".
	5	Establece que el CommitmentTypeIndications

		número ***n*** es "**Prueba de aprobación**".
	6	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de creación**".
`commitmentTypeIndication`***n***`Description`	[Texto]	Establece la descripción del CommitmentTypeIndications número ***n***.. Este atributo es opcional.
`commitmentTypeIndication`***n***`DocumentationReferences`	[Texto]	Lista de URL separadas por el carácter '|' que se aportan como referencias documentales del CommitmentTypeIndication número ***n*** (atributo opcional). Las URL de la lista no pueden contener el carácter '|' (ya que este se usa como separador).
`commitmentTypeIndication`***n***`–CommitmentTypeQualifiers`	[Texto]	Lista de indicadores textuales separados por el carácter '|' que se aportan como calificadores adicionales del CommitmentTypeIndication número n (atributo opcional). Normalmente son OID. Los elementos de la lista no pueden contener el carácter '|' (ya que este se usa como separador).

IV.2.2 CONTRAFIRMA

Nombre del parámetro	Valores posibles	Descripción
`addKeyInfoKeyValue`	`true`	Incluye el nodo KeyValue dentro de KeyInfo de XAdES (comportamiento por defecto).
	`false`	No incluye el nodo KeyValue dentro de KeyInfo de XAdES.
`addKeyInfoKeyName`	`true`	Incluye el nodo KeyName dentro de KeyInfo de XAdES.
	`false`	No incluye el nodo KeyName dentro de KeyInfo de XAdES (comportamiento por defecto).
`policyIdentifier`	[URL]	Identificador de la política de firma (normalmente una URL hacia la política en formato XML procesable), necesario para generar firmas XAdES-EPES.
`policyIdentifierHash`	[Base64]	Huella digital de la política de firma. Es obligatorio indicar este parámetro si el valor indicado en `policyIdentifier` no es universalmente accesible. Si se da valor a este parámetro es obligatorio también dar valor al parámetro `policyIdentifierHashAlgorithm`.
`policyIdentifierHashAlgorithm`	`SHA1`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA1.
	`SHA-256`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-256.
	`SHA-384`	Indica que la huella digital indicada en el parámetro `policyIdentifierHash` se calculó mediante el algoritmo SHA-384.

	SHA-512	Indica que la huella digital indicada en el parámetro policyIdentifierHash se calculó mediante el algoritmo SHA-512.
policyQualifier	[URL hacia documento]	URL (universalmente accesible) hacia el documento (normalmente PDF) que contiene una descripción textual de la política de firma. Este parámetro es opcional incluso si se desea generar firmas XAdES-EPES.
policyDescription	[Texto]	Descripción textual de la política de firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado. Este parámetro es opcional incluso si se desea generar firmas XAdES-EPES.
signerClaimedRoles	[Texto]	Agrega a la firma campos con los cargos atribuidos al firmante. Deben separarse los cargos con el carácter "|" (y este no puede estar en el propio texto de ningún cargo). En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionCity	[Texto]	Agrega a la firma un campo con la ciudad en la que se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionProvince	[Texto]	Agrega a la firma un campo con la provincia en la que se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionPostalCode	[Texto]	Agrega a la firma un campo con el código postal en donde se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML firmado.
signatureProductionCountry	[Texto]	Agrega a la firma un campo con el país en el que se realiza la firma. En el caso de que se firme un XML, la codificación del texto usado debe adecuarse al XML contrafirmado.
encoding	[Texto]	Fuerza una codificación para la firma resultante. Un uso incorrecto de este parámetro puede provocar la generación de una firma inválida.
commitmentTypeIndications	[Entero]	Indica el número de CommitmentTypeIndications que se van a declarar. Estos son los motivos que se declaran para la firma. Los valores concretos se especifican con commitmentTypeIndicationnIdentifier y commitmentTypeIndicationnDescription, donde 'n' va desde 0 hasta el valor menos 1 indicado en esta propiedad.
commitmentTypeIndication-***n***Identifier	1	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de origen**".
	2	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de recepción**".
	3	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de entrega**".
	4	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de envío**".
	5	Establece que el CommitmentTypeIndications número

		n es "**Prueba de aprobación**".
	6	Establece que el CommitmentTypeIndications número ***n*** es "**Prueba de creación**".
`commitmentTypeIndication`***n***`Description`	[Texto]	Establece la descripción del CommitmentTypeIndications número ***n***. Este atributo es opcional.
`commitmentTypeIndication`***n***`DocumentationReferences`	[Texto]	Lista de URL separadas por el carácter '|' que se aportan como referencias documentales del CommitmentTypeIndication número ***n*** (atributo opcional). Las URL de la lista no pueden contener el carácter '|' (ya que este se usa como separador).
`commitmentTypeIndication`***n***`–CommitmentTypeQualifiers`	[Texto]	Lista de indicadores textuales separados por el carácter '|' que se aportan como calificadores adicionales del CommitmentTypeIndication número n (atributo opcional). Normalmente son OID. Los elementos de la lista no pueden contener el carácter '|' (ya que este se usa como separador).

IV.3 FORMATO FACTURAE

Nombre del parámetro	Valores posibles	Descripción
`signatureProductionCity`	[Texto]	Agrega a la firma un campo con la ciudad en la que se realiza la firma.
`signatureProductionProvince`	[Texto]	Agrega a la firma un campo con la provincia en la que se realiza la firma.
`signatureProductionPostalCode`	[Texto]	Agrega a la firma un campo con el código postal en donde se realiza la firma.
`signatureProductionCountry`	[Texto]	Agrega a la firma un campo con el país en el que se realiza la firma.

IV.4 FORMATO PADES

Nombre del parámetro	Valores posibles	Descripción
`includeOnlySignningCertificate`	`true`	Se incluye en la firma únicamente el certificado del firmante.
	`false`	Se incluye en la firma la cadena de certificación completa si es posible. Valor por defecto.
`alwaysCreateRevision`	`true`	Crea una revisión del PDF al realizar la firma. Requiere que los documentos de entrada cumplan estrictamente la especificación PDF 1.7 (ISO 32000-1:2008), y puede crear incompatibilidades con documentos PDF acordes a la especificación 1.3
	`false`	Creará una revisión del PDF sólo si ya contenía una firma previa.

image	[Base64]	Imagen que se desea insertar en el PDF antes de que este sea firmado. La imagen debe proporcionarse en formato JPEG codificado en base 64. Si el documento ya contiene firmas es posible que se invaliden, por lo que conviene usarlo únicamente en documentos sin firmas previas.
imagePage	[Entero]	Página donde desea insertarse la imagen indicada mediante el parámetro image. La numeración de las páginas comienza en uno. Si se indica -1 como número de página se inserta la imagen en la última página del documento. Si se indica 0 como número de página se inserta la imagen en todas las páginas del documento. Este parámetro es obligatorio, si no se indica una página válida no se insertará la imagen.
imagePositionOnPageLowerLeftX	[Entero]	Coordenada horizontal inferior izquierda de la posición de la imagen (indicada mediante el parámetro image) dentro de la página.
imagePositionOnPageLowerLeftY	[Entero]	Coordenada vertical inferior izquierda de la posición de la imagen (indicada mediante el parámetro image) dentro de la página.
imagePositionOnPageUpperRightX	[Entero]	Coordenada horizontal superior derecha de la posición de la imagen (indicada mediante el parámetro image) dentro de la página.
imagePositionOnPageUpperRightY	[Entero]	Coordenada superior derecha de la posición de la imagen (indicada mediante el parámetro image) dentro de la página.
attach	[Base64]	Contenido a añadir como adjunto al PDF. Este parámetro requiere que se haya establecido también el parámetro attachFileName.
attachFileName	[Texto]	Nombre de fichero para adjuntar el contenido binario indicado mediante attach.
attachDescription	[Texto]	Descripción del contenido binario indicado mediante attach.
certificationLevel	0	Firma sin certificar. Esta sería una firma de aprobación. Es el valor por defecto.
	1	Firma certificada de autor. Tras este tipo de firma certificada, no se permite ningún cambio posterior en el documento (no se pueden agregar firmas, ni rellenar formularios).
	2	Firma certificada de autor para formularios. Tras este tipo de firma certificada, sólo se permite el relleno de los campos de formulario (no se pueden agregar firmas).
	3	Firma certificada común. Tras este tipo de firma certificada, sólo se permite el relleno de los campos de formulario y la creación de firmas de aprobación.
compressPdf	true	Se comprime el PDF de salida (firmado). Se omite si el PDF no lo permite. Valor por defecto.
	false	El PDF resultante no se comprime.
pdfVersion	2	El PDF de salida se declara como PDF 1.2.
	3	El PDF de salida se declara como PDF 1.3.
	4	El PDF de salida se declara como PDF 1.4.
	5	El PDF de salida se declara como PDF 1.5.
	6	El PDF de salida se declara como PDF 1.6.
	7	El PDF de salida se declara como PDF 1.7.

signatureSubFilter	adbe.pkcs7.detached	Subfiltro para la firma PAdES básica. Este es el valor por defecto.
	ETSI.CAdES.detached	Para generar una firma PAdES B-Level/PAdES-BES.
signatureField	[Texto]	Nombre del campo de firma creado anteriormente en donde insertarla.
signaturePage	[Entero]	Inserta una firma visible PDF en la página indicada. Requiere indicar la posición de la firma.
	LAST_PAGE	Inserta una firma visible PDF en la última página. Requiere indicar la posición de la firma.
signaturePositionOnPageLowerLeftX	[Entero]	Coordenada horizontal inferior izquierda de la posición del recuadro visible de la firma dentro de la página.
signaturePositionOnPageLowerLeftY	[Entero]	Coordenada vertical inferior izquierda de la posición del recuadro visible de la firma dentro de la página.
signaturePositionOnPageUpperRightX	[Entero]	Coordenada horizontal superior derecha de la posición del recuadro visible de la firma dentro de la página.
signaturePositionOnPageUpperRightY	[Entero]	Coordenada vertical superior derecha de la posición del recuadro visible de la firma dentro de la página.
signatureRubricImage	[Base64]	Imagen JPEG que se desea mostrar en el campo de firma visible.
layer2Text		Texto a mostrar en la firma visible PDF. Este texto puede incluir una serie de palabras clave que serán sustituidas por los textos apropiados del titular o emisor del certificado de firma: • **$$SUBJECTCN$$** Nombre común (CN, Common Name) dentro del X.500 Principal del titular del certificado de firma. • **$$ISSUERCN$$** Nombre común (CN, Common Name) dentro del X.500 Principal del emisor del certificado de firma. • **$$CERTSERIAL$$** Número de serie del certificado de firma. • **$$SIGNDATE=PATRÓN$$** Fecha de la firma, donde PATRÓN debe indicar el formato en el que debe mostrarse la fecha, siguiendo el esquema definido por Oracle para la clase SimpleDateFormat.
layer2FontFamily	0	Texto con fuente Courier (Por defecto)
	1	Texto con fuente Helvética
	2	Texto con fuente Times Roman
	3	Texto con fuente Symbol
	4	Texto con fuente ZapfDingBats
layer2FontSize	[Entero]	Tamaño de letra a usar en el texto de la firma visible
layer2FontStyle	0	Texto de la firma visible sin estilo (Por defecto)
	1	Texto de la firma visible en negrita
	2	Texto de la firma visible en cursiva
	3	Texto de la firma visible en negrita y cursiva
	4	Texto de la firma visible subrayado
	8	Texto de la firma visible tachado
layer2FontColor	black	Texto de la firma visible en color negro (Por defecto)
	white	Texto de la firma visible en color blanco
	gray	Texto de la firma visible en color gris
	lightGray	Texto de la firma visible en color gris claro
	darkGray	Texto de la firma visible en color gris oscuro

	red	Texto de la firma visible en color rojo
	pink	Texto de la firma visible en color rosa
signReason	[Texto]	Razón por la que se realiza la firma
signatureProductionCity	[Texto]	Ciudad en la que se realiza la firma
signerContact	[Texto]	Contacto del firmante, usualmente una dirección de correo electrónico
policyIdentifier	[URL]	Identificador de la política de firma (normalmente una URL hacia la política en formato XML procesable), necesario para generar firmas EPES.
policyIdentifierHash	[Base64]	Huella digital de la política de firma. Es obligatorio indicar este parámetro si el valor indicado en policyIdentifier no es universalmente accesible. Si se da valor a este parámetro es obligatorio también dar valor al parámetro policyIdentifierHashAlgorithm.
policyIdentifierHashAlgorithm	SHA1	Indica que la huella digital indicada en el parámetro policyIdentifierHash se calculó mediante el algoritmo SHA1.
	SHA-256	Indica que la huella digital indicada en el parámetro policyIdentifierHash se calculó mediante el algoritmo SHA-256.
	SHA-384	Indica que la huella digital indicada en el parámetro policyIdentifierHash se calculó mediante el algoritmo SHA-384.
	SHA-512	Indica que la huella digital indicada en el parámetro policyIdentifierHash se calculó mediante el algoritmo SHA-512.
policyQualifier	[URL]	URL (universalmente accesible) hacia el documento (normalmente PDF) que contiene una descripción textual de la política de firma.
ownerPassword	[Texto]	Contraseña de apertura del PDF (contraseña del propietario) si este estaba cifrado. No se soporta la firma de documentos PDF cifrados con certificados o con algoritmo AES256.
headless	false	Se pide a usuario, de forma interactiva, la información imprescindible para realizar la firma. Por ejemplo, la contraseña del PDF si está cifrado. Este es el comportamiento por defecto.
	true	Se omite cualquier interacción con el usuario.
allowSigningCertifiedPdfs	[Sin establecer]	Consultará al usuario si desea realizar la firma cuando se encuentre un PDF certificado. Firmar un PDF certificado puede generar un documento inválido.
	false	Se lanzará un error al intentar firmar un PDF certificado.
	true	Se firmarán los PDF certificados. Firmar un PDF certificado puede generar un documento inválido.
signingCertificateV2	[Sin establecer]	Se usará signingCertificateV2 para las firmas SHA2 y signingCertificateV1 para las SHA1.
	false	Se usará siempre signingCertificateV1.
	true	Se usará siempre signingCertificateV2.

ANEXO V CONFIGURACIÓN DE LOS FILTROS DE CERTIFICADOS LOCALES

En este apartado se reproducen las opciones de configuración admitidas por FIRe para el filtrado de los certificados locales del usuario. Estos filtros permiten restringir cuáles son los certificados locales que podrá utilizar el usuario para firmar cuando seleccione el proveedor de firma con certificados locales.

Estos filtros se heredan de las bibliotecas del Cliente @firma. Para obtener más información sobre los filtros de certificados, consulte el manual del integrador del Cliente @firma.

Los filtros de certificados se configurarán a través del parámetro `extraParams` de los métodos `sign`, `createBatchProcess` y `addDocumentToBatch`. Las claves que nos permiten establecer filtros de certificados son:

- *filters*: Esta clave permite establecer uno o más de los filtros de certificados que se listan más adelante en este apartado. Los certificados deberán cumplir las condiciones establecidas en todos los filtros listados, o de lo contrario no se mostrarán. Los distintos filtros se deben separar mediante el carácter punto y coma (';'). Ejemplos:
    - *filters=nonexpired:*
        - Certificados no caducados
    - *filters*=issuer.rfc2254: (O=DIRECCION GENERAL DE LA POLICIA);keyusage.nonrepudiation:true
        - Certificados de firma del DNIe
    - *filters*=issuer.rfc2254: (O=DIRECCION GENERAL DE LA POLICIA);keyusage.nonrepudiation:true;*nonexpired:*
        - Certificados de firma del DNIe no caducados.
- *filters.X*: En esta clave 'X' será un entero igual o mayor que 1. Primero se procesará la clave *filters.1*, a continuación, *filters.2* y así hasta que no encuentre una de las claves de la secuencia. Al contrario que con la clave *filters*, basta con que el certificado cumpla uno de estos filtros para que se muestre. No es necesario cumplirlos todos. Cada uno de estas claves puede declarar varios filtros separados por punto y coma (';') de tal forma que sí se deberán cumplir todos ellos para satisfacer ese sub-filtro concreto. Ejemplo:
    - *filters.1*=issuer.rfc2254:(O=DIRECCION GENERAL DE LA POLICIA);keyusage.nonrepudiation:true
    - *filters.2*=issuer.rfc2254:(O=FNMT)
        - La conjunción de estas dos claves en una operación de firma hará que sólo se muestren al usuario los certificados CERES y el de firma del DNIe.

Estas claves de definición de filtros son excluyentes y tienen la prioridad según la que se listan (*filters* y *filters.X*). Es decir, si se establece la propiedad *filters*, no se procesará la propiedad *filters.1*, por ejemplo.

Los filtros disponibles son:

- Filtro DNIe: Filtra los certificados del almacén para que sólo se muestren los certificados de firma de los DNIe disponibles desde ese almacén.
    - Para establecer este filtro de certificados se indicará la propiedad *filter* con el valor *dnie:* en el parámetro de configuración de la operación de firma, cofirma o contrafirma.
    - Ejemplo:
        - `filters=dnie:`
- Filtro de certificados de firma: Filtra los certificados del almacén para que no se muestren los considerados certificados de autenticación. Esta exclusión no se realiza mediante KeyUsage para evitar que queden excluidos certificados mal identificados. Un ejemplo de certificado que no se mostrará en el diálogo es el de autenticación del DNIe.
    - Para establecer este filtro de certificados se indicará la propiedad *filter* con el valor *signingCert:* en el parámetro de configuración de la operación de firma, cofirma o contrafirma.
    - Ejemplo:
        - `filters=signingCert:`
- Filtro de certificados de autenticación: Filtra los certificados del almacén para que no se muestren los específicos para firma avanzada reconocida. Esta exclusión no se realiza mediante KeyUsage para evitar que queden excluidos certificados mal identificados. Un ejemplo de certificado que no se mostrará en el diálogo es el de firma del DNIe.
    - Para establecer este filtro de certificados se indicará la propiedad *filter* con el valor *authCert:* en el parámetro de configuración de la operación de firma, cofirma o contrafirma.
    - Ejemplo:
        - `filters=authCert:`
- Filtro de certificados SSCD: Filtra los certificados del almacén para que se muestren sólo aquellos emitidos por medio de un dispositivo SSCD (dispositivo seguro de creación de firma), como es el caso de los certificados del DNIe. Hay que tener en cuenta que el filtrado se realiza a partir de un atributo QCStatement declarado en el propio certificado. Si la autoridad de certificación no incluye este atributo, no será posible realizar la distinción.
    - Para establecer este filtro de certificados se indicará la propiedad *filter* con el valor *sscd:* en el parámetro de configuración de la operación de firma, cofirma o contrafirma.
    - Ejemplo:
        - `filters=sscd:`
- Filtro de certificados cualificados de firma: Filtra los certificados del almacén para que sólo se muestre aquellos con un número de serie concreto (comúnmente sólo será uno). En el caso de que este certificado no esté cualificado para firma, se buscará un certificado parejo que sí lo esté en el

almacén. Si se encontrase se seleccionaría este nuevo certificado y, si no, se seleccionará el certificado al que corresponde el número de serie.

  - Para establecer este filtro de certificados se indicará la propiedad *filter* con el valor *qualified:*, seguido por el número de serie del certificado, en el parámetro de configuración de la operación de firma, cofirma o contrafirma. Esto es: *filter=qualified:Nº_serie*. El número de serie se debe indicar en hexadecimal:
  - Ejemplos:
    - `filters=qualified:45553a61`
    - `filters=qualified:03ea`

- Filtro de certificados caducados: Filtra aquellos certificados que se encuentran fuera de su periodo de validez para que sólo se muestren los certificados vigentes, que son los únicos que pueden generar una firma válida.

  - Para establecer este filtro se usará la palabra clave *nonexpired:*
  - Ejemplo:
    - `filters=nonexpired:`

- Filtro por huella digital (Thumbprint): Filtra los certificados de tal forma que sólo se mostrará aquel que tenga la huella digital indicada. Hay que tener en cuenta que esta huella digital no debe calcularse en base a un fichero (por ejemplo, un ".cer"), sino que es la huella digital de la codificación del certificado.

  - Para establecer este filtro se usará la palabra clave *thumbprint:*, seguida del algoritmo de huella digital utilizado y, separado por el carácter dos puntos (':') la huella digital que se busque en hexadecimal.
  - Ejemplo:
    - `filters=thumbprint:SHA1:30 3a bb 15 44 3a fd d7 c5 a2 52 dc a5 54 f4 c5 ee 8a a5 4d`
      - Este filtro sólo mostrará el certificado cuya huella digital en SHA1 sea la indicada.

- Filtro RFC2254 en base al *Subject* del certificado: Filtra los certificados a partir de una expresión regular creada según la RFC2254 que se aplica sobre el *Subject* del certificado.

  - Para establecer este filtro se usará el valor *subject.rfc2254:* seguido de la expresión RFC2254.
  - Puede revisarse la normativa RFC 2254 en http://www.faqs.org/rfcs/rfc2254.html
  - Ejemplo:
    - `filters=subject.rfc2254:(CN=*12345678z*)`
      - Este filtro mostrará sólo aquellos certificados en los que aparezca la cadena "12345678z" en el *CommonName* de su *Subject*.

- Filtro RFC2254 en base al *Issuer* del certificado: Filtra los certificados a partir de una expresión regular creada según la RFC2254 que se aplica sobre el *Issuer* del certificado.

  - Para establecer este filtro se usará el valor *issuer.rfc2254:* seguido de la expresión RFC2254.

- Puede revisarse la normativa RFC 2254 en http://www.faqs.org/rfcs/rfc2254.html
- Ejemplo:
  - `filters=issuer.rfc2254:(|(O=FNMT)(O=DIRECCION GENERAL DE LA POLICIA))`
    - Este filtro mostrará sólo aquellos certificados cuyo *Issuer* tenga establecido como organización "FNMT" o "DIRECCION GENERAL DE LA POLICIA", es decir, sólo mostrará los certificados del DNIe y los de CERES.
- Este filtro puede aplicarse de forma recursiva, de tal forma que permitirá el uso del certificado si cualquier de los certificados de la cadena de certificación por encima de él mismo cumple con la expresión indicada. Para utilizar recursivamente este filtro se usará el valor *issuer.rfc2254.recurse:* seguido de la expresión RFC2254.
- Ejemplo:
  - `filters=issuer.rfc2254.recurse:(CN=*FNMT*)`
    - Este filtro mostrará sólo aquellos certificados en los que alguno de los certificados de su cadena de certificación tenga la partícula "FNMT" en el nombre común

- Filtro de texto en base al *Subject* del certificado: Filtra los certificados según si contienen o no una cadena de texto en el *Principal* de su *Subject*.
  - Para establecer este filtro se usará el valor *subject.contains:* seguido de la cadena de texto que debe contener.
  - Ejemplo:
    - `filters=subject.contains:JUAN ESPAÑOL ESPAÑOL`
      - Este filtro mostrará sólo aquellos certificados en los que aparezca la cadena "JUAN ESPAÑOL ESPAÑOL" en el *Subject*.
- Filtro de texto en base al *Issuer* del certificado: Filtra los certificados según si contienen o no una cadena de texto en el *Principal* de su *Issuer*.
  - Para establecer este filtro se usará el valor *issuer.contains:* seguido de la cadena de texto que debe contener.
  - Ejemplo:
    - `filters=issuer.contains:O=EMPRESA`
      - Este filtro mostrará sólo aquellos certificados en los que el *Principal* del *Issuer* muestre el texto "O=EMPRESA".
- Filtros por uso declarado de los certificados (*KeyUsage*): Colección de filtros que permiten filtrar según el uso declarado de los certificados.
  - Para establecer estos filtros usaremos las siguientes claves según los usos que se quieran comprobar. Las claves irán seguidas de los valores "true" o "false", según se desee que el uso esté habilitado o no lo esté, respectivamente:
    - *keyusage.digitalsignature:*
    - *keyusage.nonrepudiation:*
    - *keyusage.keyencipherment:*

        - *keyusage.dataencipherment:*
        - *keyusage.keyagreement:*
        - *keyusage.keycertsign:*
        - *keyusage.crlsign:*
        - *keyusage.encipheronly:*
        - *keyusage.decipheronly:*
    - Los *KeyUsages* que no se declaren en el filtro no se tendrán en cuenta.
    - Ejemplos:
        - `filters=keyusage.digitalsignature:true;keyusage.keyencipherment:true`
            - Este filtro mostrará sólo aquellos certificados que tengan establecidos a `true` los *KeyUsage* `digitalsignature` (autenticación) y `keyencipherment` (sobres electrónicos), ignorando el valor del resto de *KeyUsages*. Este filtro mostrará, por ejemplo, los certificados de la FNMT.
        - `filters=keyusage.nonrepudiation:true`
            - Este filtro mostrará sólo aquellos certificados que tengan establecidos a `true` el `nonrepudiation` (firma avanzada). Este filtro mostrará, por ejemplo, el certificado de firma del DNIe.
- Filtro por identificador de directiva: Filtra los certificados por aquellos que poseen un identificador de directiva concreto. Esto es útil para mostrar sólo determinado tipo de certificados de una autoridad de certificación.
    - Para establecer este filtro se usará el valor *policyid:* seguido por listado de OIDs, separados por comas (‘,’), por los que se quieran filtrar.
    - Ejemplo:
        - `filters=policyid:1.3.6.1.4.1.18332.3.4.1.2.11`
            - Este filtro mostrará sólo aquellos certificados con el identificador de directiva “`1.3.6.1.4.1.18332.3.4.1.2.11`”.
- Filtro de seudónimo: Filtra los certificados, listando únicamente los certificados de pseudónimo y aquellos que no tienen un certificado de seudónimo asociado. Así, quedan ocultos los certificados que tienen un certificado equivalente de seudónimo, lo que evita que aparezca su nombre real en los datos de firma.
    - Para establecer este filtro se usará la palabra clave *pseudonym:*
    - Ejemplo:
        - `filters=pseudonym:`
- Filtro de almacenes externos: Permite deshabilitar el botón de carga de almacenes PKCS#12 en el diálogo de selección de certificados. De esta forma sólo podrán usarse los certificados del almacén seleccionado por el integrador o los por defecto del navegador en caso de que el integrador ni especificase ningún almacén.
    - Para establecer este filtro se usará la palabra clave *disableopeningexternalstores*
    - Ejemplo:
        - `filters=disableopeningexternalstores`

- Filtro en base a certificado codificado: Filtra los certificados para seleccionar uno concreto proporcionado a través del filtro. Esto es de utilidad cuando, después de una operación realizada con un certificado, se quiere restringir futuras operaciones para que se realicen con el mismo certificado.
    - Para establecer este filtro se usará el valor *encodedcert:* seguido del certificado codificado en base 64. Esto es, tal como se devuelve a trasvés del callback en los métodos de firma y selección de certificado.
    - Ejemplo:
        - `filters=encodedcert:MIIIcjCCBlqgAwIB……radvEjJ=`
            - Este filtro mostrará sólo aquel certificado que hemos proporcionado en el filtro, en caso de que exista en el almacén.

Se ignorará cualquier valor establecido como filtro de certificados distinto a los que se han listado.

Si ningún certificado cumple los criterios de filtrado, se lanzará una excepción indicando que no se ha encontrado ningún certificado que cumpla con los criterios indicados y se cancelará la operación.

Si más de un certificado cumple los criterios de filtrado, se mostrarán todos ellos en el diálogo de selección de certificados.

ANEXO VI MIGRACIÓN A FIRE 2.3

VI.1 MIGRACIÓN DE APLICACIONES CON FIRE 2.0 / 2.1 / 2.1.1 / 2.2

Las aplicaciones que utilicen FIRe 2.0, 2.1 o 2.1.1 deberán actualizar el componente distribuido que estén utilizando (Java, .NET o PHP) por el correspondiente de FIRe 2.3. Para ello, deberán sustituir su componente actual (JAR, DLL o fichero PHP) por el nuevo.

Las aplicaciones que utilicen FIRe 2.2 podrán seguir usando el componente distribuido de esa versión, aunque no podrán disfrutar de las ventajas del componente distribuido de FIRe 2.3.

Las aplicaciones que utilicen el componente distribuido Java de FIRe 2.3, adicionalmente, deberán importar a su proyecto las bibliotecas puente entre SLF4J y las bibliotecas de *log* que utilice en su aplicación (Log4J, Log4J 2, Java Logging API…). Consulte el apartado Configuración de *logs* del componente distribuido Java para más información.

VI.2 MIGRACIÓN DE APLICACIONES CON CL@VE FIRMA

Las aplicaciones que utilicen directamente el API de Cl@ve Firma no deberán realizar ningún cambio.