**Kit de integración de firma con FIRe**

Manual del integrador

Versión: 2.4

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | |
| Título | |  | |
| Autor | | Secretaría General de Administración Digital  Ministerio de Asuntos Económicos y Transformación Digital | |
| Fecha versión 2.4 | | 14 de mayo de 2021 | |
|  | | | |
| 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 |
| 2.4 | 14-05-2021 | SGAD | Conectores reutilizables, validación de firmas, sistema de logs, nueva administración |
|  |  |  |  |

ÍNDICE

[1. Objeto del documento 6](#_Toc75159168)

[2. Introducción 7](#_Toc75159169)

[3. Adecuación al Esquema Nacional de Seguridad 8](#_Toc75159170)

[4. Alta de aplicaciones 9](#_Toc75159171)

[5. Operaciones del componente distribuido 10](#_Toc75159172)

[5.1. Flujos de operación 10](#_Toc75159173)

[5.1.1. Firma de un único documento 10](#_Toc75159174)

[5.1.2. Firma de un lote de documentos 12](#_Toc75159175)

[5.1.3. Recuperar firma asíncrona 14](#_Toc75159176)

[5.2. Listado de operaciones 15](#_Toc75159177)

[6. Compatibilidad con dispositivos móviles 17](#_Toc75159178)

[7. Integración del componente distribuido Java 18](#_Toc75159179)

[7.1. Configuración 18](#_Toc75159180)

[7.1.1. Cifrado de contraseñas 19](#_Toc75159181)

[7.1.2. Configuración de *logs* 20](#_Toc75159182)

[7.2. Operaciones 21](#_Toc75159183)

[7.2.1. Construcción del cliente 21](#_Toc75159184)

[7.2.2. Firma de datos 23](#_Toc75159185)

[7.2.3. Recuperación de firma 26](#_Toc75159186)

[7.2.4. Recuperación de error 28](#_Toc75159187)

[7.2.5. Creación de lote de firma 29](#_Toc75159188)

[7.2.6. Agregar documento al lote de firma 31](#_Toc75159189)

[7.2.7. Firmar lote 34](#_Toc75159190)

[7.2.8. Recuperación del resultado del lote 35](#_Toc75159191)

[7.2.9. Comprobación de progreso de la firma de un lote 36](#_Toc75159192)

[7.2.10. Recuperación de una firma de un lote 38](#_Toc75159193)

[7.2.11. Recuperación asíncrona de una firma 39](#_Toc75159194)

[8. Integración del componente distribuido .NET 41](#_Toc75159195)

[8.1. Configuración 41](#_Toc75159196)

[8.2. Operaciones 41](#_Toc75159197)

[8.2.1. Construcción del cliente 41](#_Toc75159198)

[8.2.2. Firma de datos 42](#_Toc75159199)

[8.2.3. Recuperación de firma 43](#_Toc75159200)

[8.2.4. Recuperación de error 45](#_Toc75159201)

[8.2.5. Creación de lote de firma 46](#_Toc75159202)

[8.2.6. Agregar documento al lote de firma 47](#_Toc75159203)

[8.2.7. Firmar lote 50](#_Toc75159204)

[8.2.8. Recuperación del resultado del lote 51](#_Toc75159205)

[8.2.9. Comprobación de progreso de la firma de un lote 52](#_Toc75159206)

[8.2.10. Recuperación de una firma de un lote 53](#_Toc75159207)

[8.2.11. Recuperación asíncrona de una firma 54](#_Toc75159208)

[9. Integración del componente distribuido PHP 56](#_Toc75159209)

[9.1. Configuración 56](#_Toc75159210)

[9.2. Operaciones 56](#_Toc75159211)

[9.2.1. Construcción del cliente 56](#_Toc75159212)

[9.2.2. Firma de datos 56](#_Toc75159213)

[9.2.3. Recuperación de firma 58](#_Toc75159214)

[9.2.4. Recuperación de error 59](#_Toc75159215)

[9.2.5. Creación de lote de firma 59](#_Toc75159216)

[9.2.6. Agregar documento al lote de firma 61](#_Toc75159217)

[9.2.7. Firmar lote 64](#_Toc75159218)

[9.2.8. Recuperación del resultado del lote 64](#_Toc75159219)

[9.2.9. Comprobación de progreso de la firma de un lote 65](#_Toc75159220)

[9.2.10. Recuperación de una firma de un lote 66](#_Toc75159221)

[9.2.11. Recuperación asíncrona de una firma 67](#_Toc75159222)

[10. Configuración de la operación de firma 69](#_Toc75159223)

[10.1. Operación criptográfica 69](#_Toc75159224)

[10.2. Algoritmo de firma 69](#_Toc75159225)

[10.3. Formato de firma 69](#_Toc75159226)

[10.4. Parámetros de configuración de formato 70](#_Toc75159227)

[10.5. Formato mejorado de firma y validación 70](#_Toc75159228)

[10.6. Selección del certificado de firma 72](#_Toc75159229)

[11. Configuración de las peticiones 73](#_Toc75159230)

[11.1. Configuración de proveedores de firma 73](#_Toc75159231)

[11.1.1. Configuración del conector de firma con certificado local 74](#_Toc75159232)

[11.1.2. Configuración del conector de CL@VE FIRMA 76](#_Toc75159233)

[11.1.3. Configuración del conector Simulador de CL@VE FIRMA 77](#_Toc75159234)

[11.1.4. Configuración del conector de la FNMT 77](#_Toc75159235)

[11.2. Configuración de la actualización y validación de firmas 78](#_Toc75159236)

[11.2.1. Configuración del conector con @firma 78](#_Toc75159237)

[11.3. Configuración de otros elementos 79](#_Toc75159238)

[12. Gestión de datos desde el componente central 80](#_Toc75159239)

[12.1. Implementación de una clase gestora de documentos 81](#_Toc75159240)

[12.2. Uso de una clase gestora de documentos 83](#_Toc75159241)

[12.3. Configuración de la clase gestora de documentos 85](#_Toc75159242)

[ANEXO I Ejemplo de aplicación cliente 87](#_Toc75159243)

[I.1 Aplicación Java de prueba 87](#_Toc75159244)

[I.1.1 Usuarios de prueba 87](#_Toc75159245)

[I.1.2 Guía de la página de prueba 88](#_Toc75159246)

[ANEXO II Códigos de error de las firmas de un lote 99](#_Toc75159247)

[ANEXO III Códigos de error 100](#_Toc75159248)

[ANEXO IV Configuración de los formatos de firma 102](#_Toc75159249)

[IV.1 Formato CAdES 102](#_Toc75159250)

[IV.1.1 Firma y cofirma 102](#_Toc75159251)

[IV.1.2 Contrafirma 103](#_Toc75159252)

[IV.2 Formato XAdES 105](#_Toc75159253)

[IV.2.1 Firma y cofirma 105](#_Toc75159254)

[IV.2.2 Contrafirma 111](#_Toc75159255)

[IV.2.3 Firma manifest 113](#_Toc75159256)

[IV.3 Formato FacturaE 114](#_Toc75159257)

[IV.4 Formato PAdES 116](#_Toc75159258)

[ANEXO V Configuración de los filtros de certificados locales 120](#_Toc75159259)

[ANEXO VI Migración a FIRe 2.4 126](#_Toc75159260)

[VI.1 Migración de aplicaciones integradas con FIRe 126](#_Toc75159261)

[VI.1.1 Migración de aplicaciones PHP 126](#_Toc75159262)

[VI.2 Migración de aplicaciones integradas con Cl@ve Firma 127](#_Toc75159263)

# Objeto del documento

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

# Introducción

FIRe es un sistema para la generación de firmas electrónicas con certificado de persona física. FIRe funciona exclusivamente como pasarela de firma a la que el usuario accede desde la aplicación web que solicita firmar.

FIRe permite firmar con certificados residentes en el equipo local del usuario y con certificados de proveedores de firma en la nube, como Cl@ve Firma. Gracias a ello, se hace posible firmar con unos u otros certificados sin que la aplicación deba preocuparse del origen de los mismos.

Un integrador debe conectar su aplicación con FIRe cuando desee que un usuario firme electrónicamente como parte del flujo de trabajo de su aplicación. De esta forma, un usuario podrá acceder a su aplicación, realizar el trámite oportuno y, en el momento de firmar, será redirigido automáticamente a FIRe, donde firmará los datos proporcionados por la aplicación. Una vez firmados los datos, FIRe redirigirá al usuario automáticamente de vuelta a la aplicación, que podrá mostrar el resultado al usuario y solicitar a FIRe las firmas recién realizadas.

FIRe se compone principalmente de un API cliente (Componente distribuido) y el servicio de firma (Componente Central):

* El API cliente permite hacer las llamadas al servicio de firma para la firma de uno o varios documentos (firma de lote) y posteriormente recuperar el resultado de estas operaciones. Se distribuyen tres implementaciones diferentes de este API (Java, .NET y PHP)
* El servicio de firma es el que incluye la lógica de firma y los recursos necesarios para funcionar como pasarela de firma. No existe un despliegue de este servicio al que se puedan conectar todas las aplicaciones. Cada entidad debe desplegar su propia instancia del componente central de FIRe, a la que podrán conectarse cada una de sus aplicaciones.

Para poder hacer uso de un despliegue del componente central de FIRe, una aplicación deberá estar dada de alta en el mismo y autenticarse mediante un certificado y un identificador de aplicación.

Adicionalmente, FIRe se distribuye con otro conjunto de componentes que pueden ser necesarios según el uso que se haga de FIRe o el entorno en el que se despliegue:

* Módulo de administración: Permite gestionar el alta de aplicaciones, sus certificados y responsables, visualizar estadísticas y facilitar la consulta de logs. No es obligatorio el despliegue de este componente si el sistema sólo va a permitir el uso de una aplicación y no se quiere hacer uso del resto de sus funcionalidades. El despliegue de este módulo hace necesario el uso de base de datos.
* Simulador de Cl@ve Firma: Es un servicio que emula el aspecto y funcionalidad de Cl@ve Firma. Puede desplegarse en entornos no productivos para probar la integración de las aplicaciones antes de conectar el componente central a los proveedores reales.
* Componentes adicionales: Algunos proveedores de firma en la nube pueden proporcionar servicios propios para el funcionamiento de su conector. Por ejemplo, FIRe se distribuye con un servicio de la FNMT necesario poder utilizar este proveedor. También se proporciona un servicio para facilitar la consulta de logs del componente central desde la interfaz de administración de FIRe.

# 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>

# Alta de aplicaciones

Para que las aplicaciones hagan uso de FIRe, es necesario que se hayan registrado previamente en el componente central. Contacte con el administrador del componente central de FIRe que desee utilizar para que dé de alta su aplicación. Para el proceso de alta deberá proporcionar los datos del responsable de la aplicación, así como la parte pública del certificado con el que su aplicación se identificará frente a FIRe.

Tras el alta, el administrador de FIRe le proporcionará un identificador de aplicación. El API distribuido deberá utilizar este identificador y el certificado de autenticación en las llamadas al componente central de FIRe.

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 contactar a cada uno de los proveedores deseados para que registren su aplicación y configurar en las llamadas de las operaciones de firma los parámetros que estos hayan indicado. Consulte con el administrador 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 no productivos 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.

# Operaciones del componente distribuido

## Flujos de operación

FIRe permite a las aplicaciones integrar los siguientes flujos de operación:

* Firma de un único documento:
  + Una aplicación puede enviar un documento a firmar indicando la configuración de firma que desea utilizar (operación de firma, algoritmo, formato, configuración específica del formato y formato longevo al que actualizar la firma) y seguidamente obtener la firma resultante.
* Firma de un lote de documentos:
  + 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, ya agregados todos los documentos, se ordena la firma del lote. Finalmente, se puede obtener el resultado de la firma del lote y recuperar cada firma individual.
* Recuperación asíncrona de firma.
  + Recuperación de una firma que se generó y envió a mejorar a formato longevo, pero para la que era necesario esperar un periodo de gracia antes de recuperarla.

### Firma de un único documento

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

1. Construir el objeto cliente proporcionándole el identificador de aplicación. Este identificador nos lo proporcionará el administrador de FIRe una vez dada de alta la aplicación en el servicio.
   * Alternativamente, el componente distribuido proporciona un objeto que permite realizar llamadas al servicio mediante métodos estáticos. En caso de usarlo, el identificador de aplicación se deberá proporcionar en cada una de las llamadas al servicio.
2. Llamar al método sign del componente distribuido.
   * A este método deberemos proporcionarle:
     + Identificador de usuario que desea firmar (comúnmente su DNI). Si se restringe el uso para sólo utilizar certificados locales, este valor no se utilizará por lo que podría proporcionarse 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 las peticiones para más información.
   * Como resultado se obtendrá un objeto con el identificador de la transacción y una URL de redirección.
3. Redirigir al usuario a la URL obtenida que será la de una página web de FIRe.
   * A partir de ese momento la aplicación perderá el control de la navegación del usuario, que pasará a navegar por las páginas del servicio FIRe y, si cabe, del proveedor de firma en la nube 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 del conector que indicamos en la llamada al método sign del API.
4. 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 la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método sign.
     + Identificador de usuario que solicitó la operación.
     + Formato al que actualizar la firma electrónica y, opcionalmente, configuración adicional para la actualización. Consulte el apartado Formato mejorado de firma para más información.

Este nos devolverá un objeto con alguno de los siguientes contenidos:

* + - Si la operación finalizó y completo correctamente, la firma resultante de la operación.
    - Si la firma se realizó correctamente, se envió a actualizar/validar y la plataforma de actualización determina que se debe esperar un periodo de gracia antes de obtener la firma, se obtendrá la información sobre el periodo de gracia que habrá que esperar. Esta firma deberá ser recuperada una vez transcurrido el periodo de gracia conforme se describe en el apartado Recuperar firma asíncrona.
    - Si se produjo algún error, el error detectado.
  + En caso de ser redirigidos a la página de error, se deberá llamar al método recoverErrorResult.
    - Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método sign.
    - Identificador de usuario que solicitó la operación.

Este método nos devolverá un objeto con el error detectado.

1. Una vez obtenido el resultado de la operación, continuaremos con nuestro flujo de trabajo habitual.

### Firma de un lote de documentos

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

1. Construir el objeto cliente proporcionándole el identificador de aplicación. Este identificador nos lo proporcionará el administrador de FIRe una vez dada de alta la aplicación en el servicio.
   * Alternativamente, el componente distribuido proporciona un objeto que permite realizar llamadas al servicio mediante métodos estáticos. En caso de usarlo, el identificador de aplicación se deberá proporcionar en cada una de las llamadas al servicio.
2. Llamar al método createBatchProcess del componente distribuido.
   * A este método deberemos proporcionarle:
     + Identificador de usuario que desea firmar (comúnmente su DNI). Si se restringe el uso para sólo utilizar certificados locales, este valor no se utilizará por lo que podría proporcionarse cualquier valor.
     + Configuración de firma por defecto. Esto es la operación criptográfica, el algoritmo, el formato, la configuración del formato y el formato longevo que aplicar a cada uno de los documentos del lote cuando no se establezca una configuración específica para el documento. 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 las peticiones para más información.
   * Como resultado se obtendrá un objeto con un identificador de transacción.
3. Llamar al método addDocumentToBatch para agregar un documento al lote de firma.
   * A este método deberemos proporcionarle, al menos:
     + Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método createBatchProcess.
     + Identificador de usuario que solicitó la operación.
     + Identificador del documento 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 las peticiones 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 proporcionada durante la creación del lote. Consulte el apartado Configuración de la operación de firma para más información.
   * Este método no devuelve ningún valor.
4. Repetir el paso anterior por cada documento que se desee agregar al lote.
5. Llamar al método signBatch para iniciar el proceso de firma del lote.
   * A este método deberemos proporcionarle:
     + Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método createBatchProcess.
     + Identificador de usuario que solicitó la operación.
     + Indicador de si deseamos que, una vez detectado un error en una de las firmas del lote, se detengan las operaciones de firma 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.
6. 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 createBatch del API.
7. 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 la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método createBatchProcess.
     + Identificador de usuario que solicitó la operación.

Este método nos devolverá un objeto del que obtener el listado de identificadores de los documentos agregados al lote y el resultado asociado a cada uno de ellos que puede ser:

* + - Si la operación finalizó y completo correctamente, un indicador de que ha sido así.
    - Si la firma se realizó correctamente, se envió a actualizar/validar y la plataforma de actualización determinó que se debe esperar un periodo de gracia antes de obtener la firma, se obtendrá la información sobre el periodo de gracia que habrá que esperar. Esa firma deberá ser recuperada tras el periodo de gracia con el método recoverAsyncSign.
    - Si se produjo algún error, el tipo de error detectado.

de la operación o si es necesario dejar pasar un periodo de gracia para obtenerlo 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.

1. Una vez sabemos cómo ha terminado cada una de las firmas del lote, podemos llamar al método recoverBatchSign para recuperar las firmas que finalizaron correctamente.
   * A este método deberemos proporcionarle:
     + Identificador de la transacción. Este deberá ser el identificador proporcionado como resultado de la llamada al método createBatchProcess.
     + Identificador de usuario que solicitó la operación.
     + 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 que se firmó correctamente mediante el resultado obtenido por el método recoverBatchResult.
   * 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.
2. Una vez hemos recuperado el resultado de todas las firmas que deseábamos obtener, podemos continuar con el flujo de trabajo de nuestra aplicación.

### Recuperar firma asíncrona

Como resultado de una operación de firma o de firma de lote en la que se haya solicitado validar una firma o actualizarla a formato longevo, es posible que, en lugar de obtener la firma resultante, obtengamos la información de un periodo de gracia. Este periodo es un tiempo que el sistema de actualización y validación de firmas solicita esperar antes de recuperar la firma electrónica. Si se obtuvo un periodo de gracia como resultado de una operación, nuestra aplicación debería esperar el tiempo solicitado antes de volver a pedir el resultado de firma a FIRe. Para recuperar la firma una vez cumplido a el periodo de gracia se hará uso de la operación de recuperación asíncrona. Esta es una operación que nuestra aplicación puede ejecutar sin necesidad del usuario o su identificador, por lo que podría completarse el trámite de cara al usuario y finalizarlo en un proceso en segundo plano. El periodo de gracia depende de la plataforma de validación, pero podría llegar a ser de varios días.

Algunos sistemas de validación y actualización de firma pueden no requerir periodo de gracia, hacerlo siempre o sólo en determinados casos. FIRe utiliza por defecto @firma para las operaciones de validación y actualización de firmas. Cuando @firma se encuentra configurado, es posible configurar la petición de firma a través del API de FIRe para que el sistema nunca solicite periodo de gracia. Para más información, consulte el apartado Configuración de la actualización y validación de firmas con @firma.

Para recuperar una firma para la que requiere la espera de un periodo de gracia, se hará uso de la operación de recuperación asíncrona. Esta podría, por ejemplo, lanzarse a través de una tarea programada en el servidor. Los pasos a seguir para solicitar la firma a través del API son los siguientes:

1. Construir el objeto cliente proporcionándole el identificador de aplicación. Este identificador nos lo proporcionará el administrador de FIRe una vez dada de alta la aplicación en el servicio.
   * Alternativamente, el componente distribuido proporciona un objeto que permite realizar llamadas al servicio mediante métodos estáticos. En caso de usarlo, el identificador de aplicación se deberá proporcionar en cada una de las llamadas al servicio.
2. Llamar al método recoverAsyncSign del componente distribuido.
   * A este método deberemos proporcionarle:
     + El identificador del documento notificado en el periodo de gracia.
     + El formato al que se solicitó actualizar la firma si se desea advertir de posibles actualizaciones parciales o nulo si no es necesario.
     + La configuración adicional para el sistema de actualización de firmas si fuese necesario.
     + Si se admite una actualización parcial de la firma o si debe realizarse por completo.
   * Como resultado se obtendrá la firma generada o, si aún no está disponible, un nuevo periodo de gracias.

## Listado de operaciones

Aquí se listan todas las operaciones disponibles a través del API distribuido de FIRe. Estas operaciones son compartidas por todas sus implementaciones (Java, PHP o .NET) y gran parte de las mismas se han podido ver como 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 cliente a la URL proporcionada. Desde la página que se cargará podrá seleccionar el origen del certificado de firma (local o remoto), seleccionar el certificado y autorizar la operación de firma.
* Recuperación de la firma:
  + Completa la operación de firma, la envía a validar o 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 de firma. 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. Desde esta página podrá seleccionar el origen del certificado de firma (local o remoto), seleccionar el certificado y autorizar la operación de firma.
* Recuperación del resultado del lote:
  + Completa las firmas del lote, las manda a validar o actualizar si se configuraron para ello y devuelve un objeto con el resultado de la transacción. En el objeto proporcionado se indica el identificador de cada uno de los documentos agregados al lote, como finalizó la firma de cada uno de ellos y, en caso de error, el paso de firma en el que se ha producido.
* 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.
* Recuperación asíncrona de firma:
  + Este método recupera una firma para la que anteriormente se obtuvo que debía esperarse un periodo de gracia.

# Compatibilidad con dispositivos móviles

FIRe es compatible con dispositivos móviles a nivel general, pero pueden existir requisitos de compatibilidad concretos para los distintos proveedores de certificados.

Entre los proveedores incluidos de serie en FIRe, se identifican los siguientes requisitos de compatibilidad:

* **Firma con certificado local**:
  + Requiere tener instalado el Cliente de firma Android 1.7 o el Cliente de firma iOS 1.7.
  + Sólo está disponible para las operaciones de firma simple. No está disponible la operación de firma de múltiples documentos (firma de lote). En caso de iniciar una firma de lote desde un dispositivo móvil, FIRe mostrará inactivo el proveedor de firma local. En caso de forzar al uso de este proveedor, FIRe redirigirá automáticamente a la página de error indicada por la aplicación.

Las aplicaciones móviles son especialmente restrictivas con respecto a la confianza en los certificados SSL de las páginas y servicios a los que se accede. Si desea que su despliegue de FIRe sea compatible con dispositivos móviles, utilice para cifrar su conexión un certificado SSL reconocido internacionalmente. Una alternativa para los despliegues orientados a las aplicaciones utilizadas por un conjunto de usuarios gestionados es instalar el certificado SSL en el almacén de confianza de cada uno de los dispositivos de los usuarios.

# Integración del componente distribuido Java

La implementación Java del componente distribuido está desarrollada para facilitar la integración de aplicaciones Java con FIRe. El componente se distribuye en forma de archivo JAR, junto a las dependencias de este.

Las bibliotecas son:

* Componente distribuido Java:
  + fire-client-2.4.jar
* Dependencias para el soporte de ficheros JSON:
  + javax.json-api-1.1.3.jar
  + javax.json-1.1.3.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.

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

es.gob.fire.client.FireClient

## 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.

En caso de no proporcionarse el objeto Properties en el constructor, se buscará el fichero “client\_config.properties” en el directorio configurado mediante la propiedad del sistema “fire.config.path”. Si no se hubiese configurado esta propiedad, se buscará el 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.

Las propiedades que se permiten configurar, ya sea a través del objeto Properties o del fichero de configuración, 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.certAlias (Opcional)
  + Alias del certificado del almacén que se debe utilizar para la autenticación cliente.
  + Sólo es necesario indicarlo si en el almacén hay más de un certificado.
  + Por defecto, se utiliza el único certificado que haya o el primero que se encuentre si hay más de uno.
* 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.certAlias=

javax.net.ssl.keyStoreType=JKS

### Cifrado de contraseñas

El componente distribuido Java permite que el integrador establezca cifradas las contraseñas en el Properties de configuración. 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= }

1. Implementar la interfaz Java “es.gob.fire.client.PasswordDecipher” y su método “decipher”. Este método recibe el binario resultante de decodificar el Base 64 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.
2. 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.initializeKeys();  FireClient client = **new** FireClient("MiAppCode", config, decipher);  … |

1. 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.

### 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, a continuación se muestran las dependencias que se deben agregar a la aplicación para usar las bibliotecas de gestión de logs más comunes:

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

|  |
| --- |
| <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-jdk14</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.

## Operaciones

Los métodos disponibles en el componente distribuido Java son los siguientes:

### Construcción del cliente

La clase FireClient dispone de los siguientes constructores:

public **FireClient**(String appId) throws ClientConfigFilesNotFoundException

* 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 con el que la aplicación se autenticará frente al servicio de FIRe.
* **Excepciones:**
  + es.gob.fire.client.ClientConfigFilesNotFoundException
    - Si no se ha encontrado en el sistema el fichero de configuración client\_config.properties.
  + java.lang.SecurityExceptions
    - Si se encuentra alguno error en la configuración de la conexión con el servicio de FIRe.

public **FireClient**(String appId, Properties config)

* **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.
* **Excepciones:**
  + java.lang.SecurityExceptions
    - Si se encuentra alguno error en la configuración de la conexión con el servicio de FIRe.

public **FireClient**(String appId, Properties config, PasswordDecipher decipher)

* **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.
  + decipher
    - Objeto para el descifrado de contraseñas del objeto config.
* **Excepciones:**
  + java.lang.SecurityExceptions
    - Si se encuentra alguno error en la configuración de la conexión con el servicio de FIRe.

public **FireClient**(String appId, PasswordDecipher decipher) throws ClientConfigFilesNotFoundException

* 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.
  + decipher
    - Objeto para el descifrado de contraseñas del objeto config.
* **Excepciones:**
  + es.gob.fire.client.ClientConfigFilesNotFoundException
    - Si no se ha encontrado en el sistema el fichero de configuración client\_config.properties.
  + java.lang.SecurityExceptions
    - Si se encuentra alguno error en la configuración de la conexión con el servicio de FIRe.

### 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 (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:

#### Firma a partir de enumerados

Inicia el proceso de firma de los datos proporcionados. La configuración de firma especificada determina el tipo de firma que se debe generar. Este método también permite establecer la configuración de la operación (páginas de retorno, identificación de la aplicación frente a los proveedores de firma en la nube, etc.).

public SignOperationResult **sign**(

String subjectId,

HttpSignProcessConstants.SignatureOperation op,

HttpSignProcessConstants.SignatureFormat ft,

HttpSignProcessConstants.SignatureAlgorithm algth,

Properties prop,

byte[] data,

Properties config

)

throws

java.io.IOException,

es.gob.fire.client.HttpNetworkException,

es.gob.fire.client.HttpForbiddenException,

es.gob.fire.client.HttpOperationException

* **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).
  + data
    - 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 las peticiones.
* **Valor de retorno**:
  + Objeto con el identificador de transacción y URL a la que redireccionar al usuario.
* **Excepciones**:
  + 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.

#### Firma a partir de cadenas de texto

Inicia el proceso de firma de los datos proporcionados. La configuración de firma especificada determina el tipo de firma que se debe generar. Este método también permite establecer la configuración de la operación (páginas de retorno, identificación de la aplicación frente a los proveedores de firma en la nube, etc.).

public 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,

* **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 las peticiones.
* **Valor de retorno:**
  + Objeto con el identificador de transacción y URL a la que redireccionar al usuario.
* **Excepciones**:
  + 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.

### Recuperación de firma

Este método nos permite completar la construcción y actualización de la firma solicitada anteriormente y recuperar la propia firma, el proveedor seleccionado y el certificado utilizado para firmar. Hay dos métodos sobrecargados para esto.

#### Recuperación de firma simple

Recupera la firma generada, pudiéndola enviar a validar o actualizar antes de obtenerla.

public 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

* **Parámetros**:
  + transactionId
    - Identificador de la transacción.
  + subjectId
    - Identificador del titular de los certificados.
  + upgrade
    - Formato mejorado de firma o validación (puede ser null para no hacer ni una cosa ni la otra).
* **Valor de retorno:**
  + Objeto TransactionResult con la firma electrónica generada o, si es necesario esperar un periodo de gracia, el tiempo a esperar y los datos para su recuperación. Si es posible, se devuelve también el nombre del proveedor de certificados seleccionado por el usuario y el certificado de firma utilizado.
* **Excepciones:**
  + 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.

#### Recuperación de firma con configuración de plataforma de actualización

Recupera la firma generada, pudiéndola enviar a validar o actualizar antes de obtenerla. Permite proporcionar una serie de parámetros adicionales que se enviarán a la plataforma de validación.

public TransactionResult **recoverSignResult**(

String transactionId,

String subjectId,

String upgrade,

Properties upgradeConfig

)

throws

java.io.IOException,

es.gob.fire.client.HttpForbiddenException,

es.gob.fire.client.HttpNetworkException,

es.gob.fire.client.HttpOperationException,

es.gob.fire.client.InvalidTransactionException

* **Parámetros**:
  + transactionId
    - Identificador de la transacción.
  + subjectId
    - Identificador del titular de los certificados.
  + upgrade
    - Formato mejorado de firma o validación (puede ser null para no hacer ni una cosa ni la otra).
  + upgradeConfig
    - Configuración adicional para la plataforma de actualización y validación.
* **Valor de retorno:**
  + Objeto TransactionResult con la firma electrónica generada o, si es necesario esperar un periodo de gracia, el tiempo a esperar y los datos para su recuperación. Si es posible, se devuelve también el nombre del proveedor de certificados seleccionado por el usuario (si se seleccionó un certificado local, el nombre de proveedor será “local”) y el certificado de firma utilizado.
* **Excepciones:**
  + 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.

### Recuperación de error

Recupera 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** 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.InvalidTransactionException

* **Parámetros:**
  + transactionId
    - Identificador de la transacción de la operación de firma o firma de lote.
  + subjectId
    - Identificador del titular de los certificados.
* **Valor de retorno:**
  + Objeto TransactionResult con el código, el mensaje de error y, si es posible, el nombre del proveedor de certificados utilizado y el certificado seleccionado por el usuario.
* **Excepciones:**
  + 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.

### Creación de lote de firma

Crea 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,

* **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 de configuración de firma en base 64 que aplicar a las firmas del lote (puede ser null).
  + upgrade
    - Formato mejorado de firma o validación (puede ser null para no hacer ni una cosa ni la otra).
  + 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 las peticiones.
* **Valor de retorno:**
  + Objeto CreateBatchResult con el identificador de la transacción de firma del lote.
* **Excepciones:**
  + 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.

### 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 durante la creación del lote.

#### Agregar documento con la configuración de firma del lote

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

* **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 las peticiones.
* **Excepciones:**
  + 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.

#### Agregar documento con su propia configuración de firma

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.

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

* **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 las peticiones.
  + 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.
* **Excepciones:**
  + 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.

### 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 certificado que desee utilizar y, si es necesario, para que 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

* **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.
* **Valor de retorno:**
  + Objeto SignOperationResult con el ID de transacción y la URL de redirección para la firma del lote.
* **Excepciones:**
  + 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.

### Recuperación del resultado del lote

Recupera el resultado de un lote ya firmado. Este resultado incluye el nombre del proveedor de certificados seleccionado, el certificado utilizado para firmar, 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.HttpForbiddenException,

es.gob.fire.client.HttpNetworkException,

es.gob.fire.client.HttpOperationException,

es.gob.fire.InvalidTransactionException

* **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.
* **Valor de retorno:**
  + Objeto BatchResult con el nombre del proveedor y el certificado de firma utilizado, el listado de identificadores de los documentos procesados y el resultado de firmar cada uno de ellos. Por cada documento que se no haya firmado correctamente, se incluye también 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.**
* **Excepciones:**
  + 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.

### Comprobación de progreso de la firma de un lote

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

La recuperación del resultado de un lote (iniciada mediante la llamada a recoverBatchResult), puede requerir el acceso al proveedor de firma en la nube para recuperar el resultado de la firma, componer las firmas electrónicas y actualizarlas a formatos longevos. Este proceso puede requerir un tiempo al que también afectará el número de operaciones de firma solicitadas en el lote.

Después de llamar al método de recuperación de resultado del lote, una aplicación puede llamar recurrentemente a este método de comprobación de estado para saber el progreso de la operación. La utilidad de este método es puramente informativa y opcional. Las llamadas a este método deben realizarse recurrentemente y en paralelo a la recuperación del resultado del lote, actualizando al usuario la información de estado tras cada llamada. El periodo de actualización del progreso dependerá del tiempo que se deje transcurrir entre las llamadas a este método. Debe calcularse un tiempo razonable entre llamadas para evitar la saturación del servidor.

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

* **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.
* **Valor de 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.
* **Excepciones:**
  + 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.

### 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

* **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.
* **Valor de retorno:**
  + Objeto TransactionResult con la firma electrónica de un documento concreto del lote , si es necesario esperar un periodo de gracia, el tiempo a esperar y los datos para su recuperación.
* **Excepciones:**
  + 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.

### Recuperación asíncrona de una firma

Recupera una firma solicitada y tratada de recuperar anteriormente, pero para la que se estableció la espera de un periodo de gracia.

public static TransactionResult **recoverAsyncSign** (

String docId,

String upgrade,

Properties upgradeConfig,

boolean allowPartial

)

throws

java.io.IOException,

es.gob.fire.client.HttpForbiddenException,

es.gob.fire.client.HttpNetworkException,

es.gob.fire.client.HttpOperationException,

es.gob.fire.client.InvalidTransactionException

* **Parámetros:**
  + docId
    - Identificador de documento remitida en la anterior solicitud de recuperación.
  + upgrade
    - Formato longevo al que se solicitó actualizar la firma. Si se indica, se usará para comprobar que la firma se actualizó al formato solicitado.
  + upgradeConfig
    - Configuración adicional para la plataforma de actualización de firmas.
  + allowPartial
    - Indica si debe permitirse la devolución de la firma, aunque no se consiguiese promocionar al formato longevo solicitado. Si no se indica el parámetro upgrade este valor se ignorará.
* **Valor de retorno:**
  + Objeto TransactionResult con la firma electrónica del documento o la nueva información de periodo de gracia que se debe esperar.
* **Excepciones:**
  + 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 al recuperar la firma.

# Integración del componente distribuido .NET

Para facilitar el uso de FIRe desde aplicativos .NET se proporciona la biblioteca “FIRe.dll”.

Para el uso de las funciones de esta biblioteca se deberá importar en la aplicación cliente y hacer uso del espacio de nombres “FIRe”.

## 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 si se debe confiar en cualquier certificado SSL servidor (“true”) o sólo en aquellos de confianza para el sistema (“false”, valor 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.
* ssl\_client\_alias: Alias del certificado del almacén que se debe utilizar para la autenticación cliente. Si no se indica, se tomará el primer certificado que se encuentre.

Todas las entradas en el registro son de tipo cadena (“REG\_SZ”).

## Operaciones

Los métodos disponibles en el componente distribuido .NET son los siguientes:

### Construcción del cliente

La clase FireClient dispone de dos constructores:

public **FireClient**(String appId)

* Construye el objeto con el que realizar las llamadas al componente central de FIRe. La configuración para la conexión y los datos del certificado de autenticación se tomarán de las claves del registro de Windows descritas en el apartado Configuración.
* **Parámetros**:
  + appId
    - Identificador con el que la aplicación se autenticará frente al componente central de FIRe.
* **Excepciones:**
  + ConfigureException
    - Si no se ha encontrado la propiedad fire\_service en el registro de Windows.

public **FireClient**(string appId, Dictionary<string, string> config)

* Construye el objeto con el que realizar las llamadas al componente central FIRe. Proporciona la configuración para la conexión en un diccionario. Las propiedades que no se encuentren en este se buscarán en el registro de Windows.
* **Parámetros:**
  + appId
    - Identificador con el que la aplicación se autenticará frente al componente central de FIRe.
  + config
    - Diccionario con las propiedades de configuración.
* **Excepciones:**
  + ConfigureException
    - Si no se ha encontrado la propiedad fire\_service ni en el diccionario de configuración ni en el registro de Windows.

### 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 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 las peticiones.
    - 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.
* **Excepciones:**
  + 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.

### Recuperación de firma

Estos métodos nos permiten obtener el resultado de una firma realizada anteriormente.

#### Recuperación de firma simple

Recupera la firma generada, pudiéndola enviar a validar o actualizar antes de obtenerla.

public 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, el certificado seleccionado y la firma electrónica generada o el periodo de gracia que hay que esperar. Si se utilizó un certificado local, el nombre de proveedor será “local”.
* **Excepciones:**
  + 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.

#### Recuperación de firma con configuración de plataforma de actualización

Recupera la firma generada, pudiéndola enviar a validar o actualizar antes de obtenerla. Permite proporcionar una serie de parámetros adicionales que se enviarán a la plataforma de validación.

public FireTransactionResult **recoverSign**(

string transactionId,

string subjectId,

string upgrade,

upgradeConfigB64)

* **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).
  + upgradeConfigB64
    - Configuración adicional para la plataforma de actualización y validación en forma de *properties* codificado en Base 64 (puede ser null).
* **Retorno:**
  + Objeto FireTransactionResult con el nombre del proveedor de firma utilizado, el certificado seleccionado y la firma electrónica generada o el periodo de gracia que hay que esperar. Si se utilizó un certificado local, el nombre de proveedor será “local”.
* **Excepciones:**
  + 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.

### Recuperación de error

Recupera 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 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.
* **Excepciones:**
  + 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.

### Creación de lote de firma

Crea 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 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 las peticiones.
    - 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.
* **Excepciones:**
  + 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.

### 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.

#### Agregar documento con la configuración de firma del lote

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

public 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 las peticiones.
* **Retorno:**
  + Vacío.
* **Excepciones:**
  + 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.

#### Agregar documento con su propia configuración de firma

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.

public 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 las peticiones.
* **Retorno:**
  + Vacío.
* **Excepciones:**
  + 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.

### 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 certificado que desee utilizar y, si es necesario, para que autorice la operación.

public 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.
* **Excepciones:**
  + 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.

### Recuperación del resultado del lote

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

public 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**.**
* **Excepciones:**
  + 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.

### Comprobación de progreso de la firma de un lote

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

La recuperación del resultado de un lote (iniciada mediante la llamada a recoverBatchResult), puede requerir el acceso al proveedor de firma en la nube para recuperar el resultado de la firma, componer las firmas electrónicas y actualizarlas a formatos longevos. Este proceso puede requerir un tiempo al que también afectará el número de operaciones de firma solicitadas en el lote.

Después de llamar al método de recuperación de resultado del lote, una aplicación puede llamar recurrentemente a este método de comprobación de estado para saber el progreso de la operación. La utilidad de este método es puramente informativa y opcional. Las llamadas a este método deben realizarse recurrentemente y en paralelo a la recuperación del resultado del lote, actualizando al usuario la información de estado tras cada llamada. El periodo de actualización del progreso dependerá del tiempo que se deje transcurrir entre las llamadas a este método. Debe calcularse un tiempo razonable entre llamadas para evitar la saturación del servidor.

public 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.
* **Excepciones:**
  + 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.

### Recuperación de una firma de un lote

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

public 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.
* **Excepciones:**
  + 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.

### Recuperación asíncrona de una firma

Recupera una firma solicitada y tratada de recuperar anteriormente, pero para la que se estableció la espera de un periodo de gracia.

public FireTransactionResult **recoverAsyncSign**(

string docId,

string upgrade,

string configB64,

bool allowPartial)

* **Parámetros:**
  + docId
    - Identificador del documento cuya firma quiere ser recuperada. Este identificador es proporcionado junto al periodo de gracia.
  + upgrade
    - Formato longevo al que se solicitó actualizar la firma. Si se indica, se usará para comprobar que la firma se actualizó al formato solicitado.
  + configB64
    - Configuración adicional proporcionada para la plataforma de actualización y validación de firmas, codificada en Base 64.
  + allowPartial
    - Indica si debe permitirse la devolución de la firma, aunque no se consiguiese promocionar al formato longevo solicitado. Si no se indica el parámetro upgrade este valor se ignorará.
* **Retorno:**
  + Objeto FireTransactionResult con la firma electrónica del documento o un periodo de gracia actualizado.
* **Excepciones:**
  + 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.
  + HttpOperationException
    - Cuando se produce un error durante la operación.

# 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\_client.php” en nuestra página.

## Configuración

En “fire\_client.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

**);**

## Operaciones

Los métodos disponibles en el componente distribuido PHP son los siguientes:

### Construcción del cliente

La clase FireClient dispone de un constructor:

function \_\_construct ($id)

* **Parámetros:**
  + $id
    - Identificador de aplicación con el que debe autenticarse frente al componente central de FIRe.

### Firma de datos

Este método nos permite enviar los datos necesarios para iniciar el proceso de firma.

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

function **sign**($subjectId, $op, $ft, $algth, $propB64, $dataB64, $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 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 las peticiones.
* **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.

### Recuperación de firma

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

function **recoverSign**($subjectId, $transactionId, $upgrade=null, $upgradeConfigB64=null)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $transactionId
    - Identificador de la transacción devuelto por la operación de firma.
  + $upgrade
    - Formato al que queremos mejorar la firma (puede ser null).
  + $upgradeConfigB64
    - Configuración adicional para la plataforma de actualización codificada en Base 64 (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).
    - gracePeriod: Periodo de gracia que deberá proporcionarse antes de recuperar la firma resultante de la operación.
* **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.

### Recuperación de error

Recupera 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.

function **recoverError**($subjectId, $transactionId)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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.

### Creación de lote de firma

Crea 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.

function **createBatchProcess**($subjectId, $op, $ft, $algth, $propB64, $upgrade, $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).
  + $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 las peticiones.
* **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.

### 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.

#### Agregar documento con la configuración de firma del lote

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

function **addDocumentToBatch**($subjectId, $transactionId, $documentId, $documentB64, $confB64)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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 las peticiones.
* **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.

#### Agregar documento con su propia configuración de firma

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.

function **addCustomDocumentToBatch**($subjectId, $transactionId, $documentId, $documentB64, $op, $ft, $propB64, $upgrade, $confB64)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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 las peticiones.
* **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.

### 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 certificado que desee utilizar y, si es necesario, para que autorice la operación.

function **signBatch**($subjectId, $transactionId, $stopOnError)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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.

### Recuperación del resultado del lote

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

function **recoverBatchResult**($subjectId, $transactionId)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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.
    - signingCert: Certificado usado para firmar codificado en base 64.
  + 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.

### Comprobación de progreso de la firma de un lote

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

La recuperación del resultado de un lote (iniciada mediante la llamada a recoverBatchResult), puede requerir el acceso al proveedor de firma en la nube para recuperar el resultado de la firma, componer las firmas electrónicas y actualizarlas a formatos longevos. Este proceso puede requerir un tiempo al que también afectará el número de operaciones de firma solicitadas en el lote.

Después de llamar al método de recuperación de resultado del lote, una aplicación puede llamar recurrentemente a este método de comprobación de estado para saber el progreso de la operación. La utilidad de este método es puramente informativa y opcional. Las llamadas a este método deben realizarse recurrentemente y en paralelo a la recuperación del resultado del lote, actualizando al usuario la información de estado tras cada llamada. El periodo de actualización del progreso dependerá del tiempo que se deje transcurrir entre las llamadas a este método. Debe calcularse un tiempo razonable entre llamadas para evitar la saturación del servidor.

function **recoverBatchResultState**($subjectId, $transactionId)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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.

### Recuperación de una firma de un lote

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

function **recoverBatchSign**($subjectId, $transactionId, $docId)

* **Parámetros:**
  + $subjectId
    - Identificador del usuario propietario de los certificados.
  + $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.

### Recuperación asíncrona de una firma

Recupera una firma solicitada y tratada de recuperar anteriormente, pero para la que se estableció la espera de un periodo de gracia.

function **recoverAsyncSign** ($docId, $upgrade, $confB64, $partial)

* **Parámetros:**
  + $docId
    - Identificador del documento cuya firma quiere ser recuperada. Es el identificador que se proporciona con el periodo de gracia.
  + $upgrade
    - Formato longevo al que se solicitó actualizar la firma. Si se indica, se usará para comprobar que la firma se actualizó al formato solicitado.
  + $confB64
    - Configuración adicional proporcionada para la plataforma de actualización y validación de firmas, codificada en Base 64.
  + $partial
    - Indica si debe permitirse la devolución de la firma, aunque no se consiguiese promocionar al formato longevo solicitado. Si no se indica el parámetro $upgrade este valor se ignorará.
* **Retorno:**
  + Objeto TransactionResult con la firma electrónica del documento o un periodo de gracia actualizado.
* **Excepciones:**
  + InvalidArgumentException
    - Cuando no se indica un parámetro obligatorio.
  + 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 durante la operación.

# Configuración de la operación de firma

En este apartado se describen los parámetros que deben proporcionarse a los métodos de firma para generar cada una de las firmas soportadas por FIRe.

## 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 firma en paralelo sobre la firma especificada como datos.
* countersign
  + Indica que se desea agregar una nueva firma en cascada sobre la firma especificada como datos.

## Algoritmo de firma

Los algoritmos de firma soportados por FIRe son:

* SHA1withRSA (Deprecado)
* SHA256withRSA
* SHA384withRSA
* SHA512withRSA

## 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.

FIRe no incluye un modo concreto para la generación de firmas *baseline*. Sin embargo, en la mayoría de los casos, las firmas básicas que genera serán *baseline* (perfil B-Level). Las consideraciones a tener en cuenta para los distintos formatos son las siguientes:

* CAdES
  + Las firmas generadas siempre serán *baseline*.
  + Las contrafirmas generadas no se consideran *baseline*, ya que incluyen el atributo “content-hints”.
* XAdES
  + No deben utilizarse los parámetros de configuración “xadesNamespace” y “signedPropertiesTypeUrl”. El espacio de nombres por defecto es el correcto (XAdES 1.3.2) para las firmas *baseline*.
  + Las contrafirmas generadas no se consideran *baseline*, ya que incluyen el atributo “DataObjectFormat”.
* PAdES
  + Debe utilizarse el parámetro de configuración “signatureSubFilter” para establecer el subfiltro “ETSI.CAdES.detached”.
  + No debe utilizarse el parámetro de configuración "commitment-type-indication", salvo que también se indique una política de firma.
  + No debe utilizarse el parámetro de configuración “policyReason” cuando se utilice el parámetro "commitment-type-indication" o cuando se indique una política de firma.
* CAdES-ASiC-S
  + Las firmas siempre serán *baseline*.
* XAdES-ASiC-S
  + Las firmas nunca no serán *baseline*, ya que las firmas actualmente generadas por este formato inclumplirán uno de los requisitos impuestos por la última versión del formato.

## Parámetros de configuración de formato

Los parámetros de configuración de formato de firma ayudan a ajustar la estructura o el contenido de las firmas electrónicas que se generan. 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.

## Formato mejorado de firma y validación

FIRe puede conectarse a una plataforma de validación y actualización de firmas para validar y promocionar las firmas generadas a formatos longevos antes de la devolverlas a la aplicación que las solicitó. El administrador de FIRe puede conectar el sistema a cualquier plataforma de actualización, aunque FIRe sólo proporciona nativamente el conector necesario para utilizar la Plataforma @firma. Si el administrador no conectase FIRe con una plataforma de validación, las operaciones de validación y promoción de firmas no estarían disponibles. Consulte con su administrador de FIRe que opciones y perfiles están disponibles para la promoción y validación de firmas en su despliegue.

Cuando se solicita la actualización a un formato de firma longevo o la validación de la firma, FIRe la envía a la plataforma de validación después de generarla. Si la firma es válida y se promociona al formato solicitado, FIRe la devuelve a la aplicación como resultado de la operación de firma solicitada. Si la firma no es válida o no se puede promocionar, FIRe devolverá un error.

En el caso de la promoción de firmas a formatos longevos, es posible que la plataforma de validación sólo realice una promoción parcial de la firma, es decir, que no le agregue toda la información longeva requerida por el formato indicado. Si se diese el caso, FIRe devolverá un error o la firma parcialmente promocionada según la configuración utilizada. Consulte el apartado Configuración básica de la actualización y validación de firmas para más detalles sobre las opciones de configuración.

Tenga en cuenta qué, si la plataforma de actualización soporta los perfiles *baseline* (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 nunca lanzará un error, aunque el formato devuelto no sea exactamente el solicitado.

**IMPORTANTE:** La promoción de una firma a un formato longevo implica siempre la validación de la firma. Es decir, si desea comprobar que su firma es válida y que se promocione al perfil “LTA-LEVEL”, por ejemplo, indique únicamente la opción “LTA-LEVEL”.

Los perfiles admitidos por FIRe son:

* VERIFY
  + Para la validación de la firma.
* ES-T
  + Para la actualización del formato CAdES a CAdES-T.
  + Para la actualización del formato XAdES a XAdES-T.
  + Debería sustituirse por “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-A
  + Para la actualización del formato CAdES a CAdES-A.
  + Para la actualización del formato XAdES a XAdES-A.
* 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.
  + 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.
  + 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.
  + Disponible a partir de la versión 6.2 de la Plataforma @firma.

## Selección del certificado de firma

Los proveedores de firma en la nube de confianza sólo nos permitirán firmar con certificados válidos (vigentes y activos).

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 locales con los que el usuario puede firmar, podremos configurar a través del mismo parámetro de configuración del formato una serie de filtros con los cuales restringir los certificados que puede utilizar el usuario para firmar. Puede consultar el listado de filtros en el anexo Configuración de los filtros de certificados locales.

En muchas ocasiones el usuario sólo tendrá disponible un certificado de firma por cada proveedor de firma en la nube o instalado en local. Para evitar que el usuario tenga que seleccionar este certificado, se puede configuar la operación para que FIRe omita el paso de selección de certificado y se agilice así el proceso de firma. Para establecer si se debe omitir o no el paso de selección de certificado, deberemos establecer la propiedad “skipCertSelection” con el valor “true” en la llamada al método de firma o de creación de lote.

Si no se establece valor para esta propiedad, se saltará o no la selección de certificado según la configuración establecida en el componente central de FIRe. Nunca se saltará la selección de certificado cuando el usuario tenga más de un certificado entre los que elegir en del proveedor de firma en la nube o almacén local seleccionado.

# Configuración de las peticiones

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 que desee soportar y, cada uno de ellos, tomará de la llamada los parámetros que considere oportunos e ignorará el resto.

Los métodos recoverSign y recoverAsyncSign, por su parte, incluyen el parámetro upgradeConfig, que permite proporcionar parámetros de configuración al sistema de validación y actualización de firmas. Estos parámetros también se pueden agregar al parámetro config del método createBatchProcess.

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

Por ejemplo,

En los siguientes subapartados se proporcionan opciones de configuración genéricos de FIRe y otras específicas para algunos proveedores de firma, plataformas de actualización y otros elementos.

## Configuración de proveedores de firma

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.
  + skipCertSelection: Indica si se debe saltar o no la selección del certificado de firma cuando sólo haya un certificado para firmar en el proveedor de firma en la nube o en el almacén local seleccionado.
    - Si se configura a true, se saltará la selección de certificado cuando el usuario sólo tenga un certificado del proveedor o en el almacén local.
    - Si se configura a false, se mostrará siempre el listado de certificados disponibles para que el usuario seleccione el que desea utilizar, incluso cuando sólo haya uno.
    - Si no se indica este parámetro, se saltará o no la selección de certificado según la configuración establecida en el componente central de FIRe.
  + 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.

### 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 la firma con certificados locales, FIRe hará uso de las distintas aplicaciones del Cliente @firma:

* **Entornos de escritorio**: AutoFirma
* **Entornos Android**: Cliente @firma móvil para Android
* **Entornos iOS**: Cliente @firma móvil para iOS

En el caso de las aplicaciones cliente (AutoFirma y clientes móviles), el usuario deberá instalarlas de antemano para poder utilizar sus certificados.

Para configurar expresamente este conector a través de la propiedad “certorigin” 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.

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.
* 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. El uso de un certificado SSL válido es un tema que debe gestionar el administrador del sistema.

A pesar de lo indicado, es común que en entornos de desarrollo se utilicen certificados SSL no válidos o no reconocidos por Java. En los entornos de escritorio, 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. En los dispositivos móviles no es posible este acercamiento. Se debería instalar el certificado SSL en el almacén de confianza del dispositivo y usar certificados cuyo nombre de dominio se corresponda con el dominio de despliegue de FIRe.

### 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 a través de la propiedad “certorigin” 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=

### 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 a través de la propiedad “certorigin” 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.

### 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 a través de la propiedad “certorigin” 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.

## Configuración de la actualización y validación de firmas

FIRe permite la configuración de las siguientes propiedades para definir el comportamiento general del proceso de actualización y validación de firmas:

* Métodos recoverSign, recoverAsyncSign y createBatchProcess:
  + updater.allowPartialUpgrade: Indica que la aplicación quiere que se le devuelva la firma, incluso si no se consigue promocionar al perfil longevo solicitado (true) o si debe fallar la recuperación de firma si no se obtiene la firma con este perfil (false). Puede ser útil para la aplicación obtener la firma, aunque no esté completamente actualizada, para evitar que le falle al usuario la operación de firma. Posteriormente, la propia aplicación podría conectar directamente con la plataforma de actualización de firmas para terminar de promocionarla.

### Configuración del conector con @firma

FIRe permite actualizar y validar mediante un sistema orientado a tal fin. Por defecto y siempre que lo haya configurado el administrador de FIRe, se utilizará la Plataforma @Firma.

Se permite la configuración de las siguientes propiedades para definir el comportamiento de la Plataforma @firme durante las operaciones de validación y actualización de firmas:

* Métodos recoverSign, recoverAsyncSign y createBatchProcess:
  + updater.ignoreGracePeriod: Indica que se debe ignorar la configuración de periodo de gracia establecida por el sistema (true), de tal forma que siempre se devolverán de forma inmediata las firmas actualizadas o validadas, o si debe devolverse un periodo de gracia (false) si el sistema lo considera oportuno por seguridad.

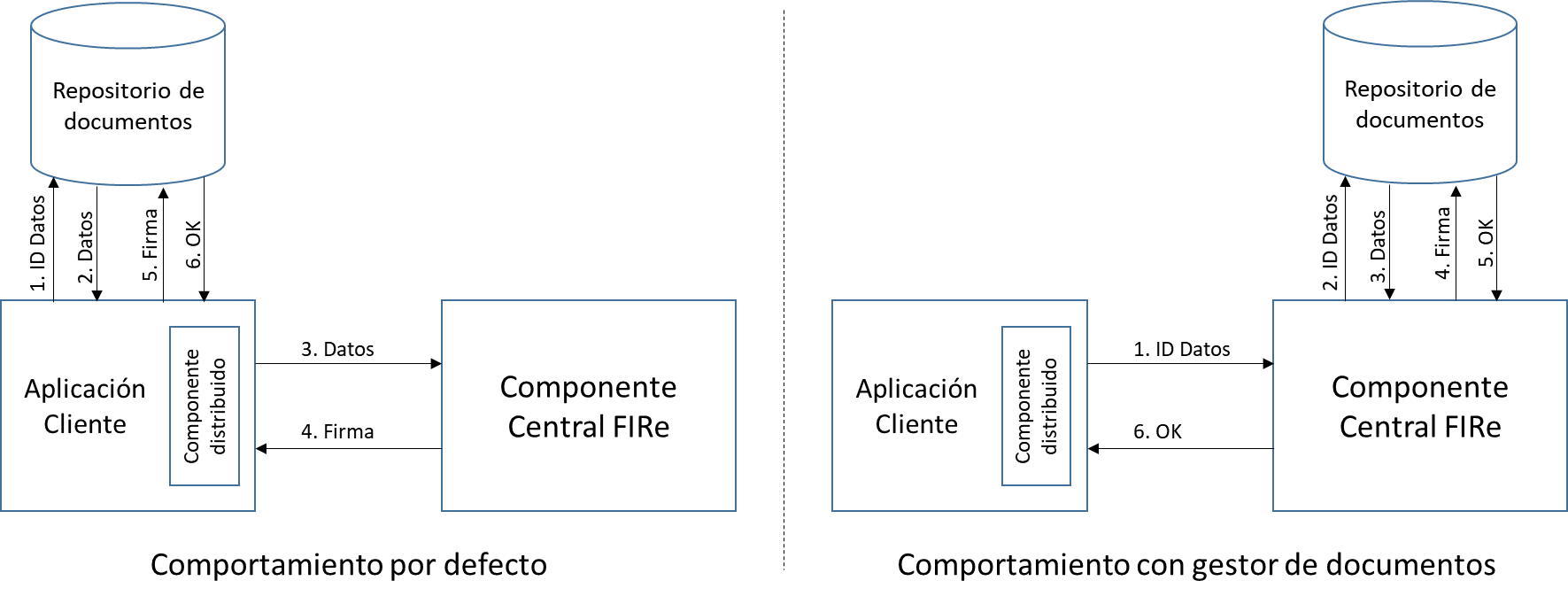
## Configuración de otros elementos

Puede ocurrir que el administrador de FIRe dé de alta otros proveedores de firma en la nube o habilite la actualización de firma mediante una plataforma distinta de @firma. Dado el caso, consulte con él el nombre de estos proveedores y las opciones de configuración que admiten los conectores de los mismos.

# 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.

A continuación, se muestran dos diagramas simplificados en los que se muestra el envío de datos en cada uno de los escenarios presentados:



Como diferencias clave entre el escenario por defecto y el que utiliza una clase gestora de documentos, el uso de la clase gestora conlleva las siguientes ventajas y desventajas:

* Ventajas:
  + Se reduce significativamente el tráfico de datos entre las aplicaciones cliente y el componente central ya que, aunque se producen el mismo número de conexiones, los datos y la firma no viajan entre la aplicación cliente y el componente central de FIRe.
  + 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 en Java 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, si el cliente envía el nombre de varias clases, se utilizaría la última indicada. 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 de 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, el nombre completo de la clase principal, sus dependencias, 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), creación de lotes (createBatchProcess) y recuperación asíncrona de firma (recoverAsyncSign) 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.

## 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 “docmanager.NOMBRE\_DOC\_MANAGER.properties”.
* 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 los servicios y recursos necesarios.

## 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, recuperación de firma asíncrona y 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.

## 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.

1. 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:

* Servicios de prueba (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.
* Aplicación Java de prueba (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.
* Páginas de prueba ASP.
* Páginas de prueba PHP.

Adicionalmente, junto a FIRe se distribuye un servidor de aplicaciones Tomcat con la aplicación Java de prueba y los servicios de FIRe 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.

* 1. Aplicación Java de prueba
     1. Usuarios de prueba

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

**Usuario con certificado válido**

* Usuario: 00001
* Contraseña: 1111

**Usuario sin certificado (permite la generación de uno durante la firma)**

* Usuario: 00002
* Contraseña: 1111

**Usuario con certificado bloqueado**

* Usuario: 00003

**Usuario con registro no fehaciente /débil**

* Usuario: 00004

**Usuario con dos certificados válidos**

* Usuario: 00005
* Contraseña: 1111

Adicionalmente, los servicios de prueba incluyen los mismos usuarios de prueba de utilizados en Cl@ve Permanente, con lo cual es posible que una aplicación haga pruebas completas de autenticación y firma utilizando estos usuarios. Los usuarios de prueba de Cl@ve Permanente son:

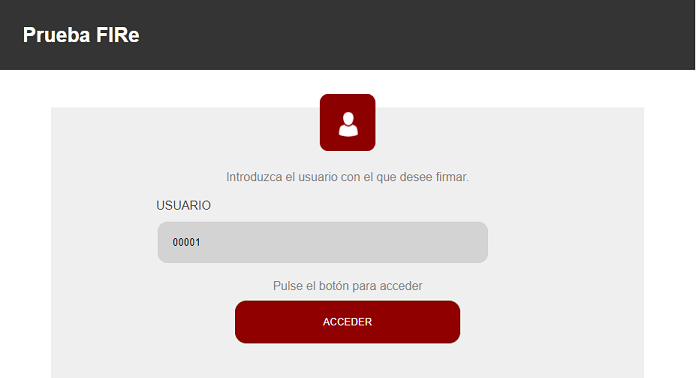
* Usuarios: **54500442X**, **24540447E**
  + Contraseña: **Modernizacion1!**
* Usuario: **92920000T**
  + Contraseña: **Entra001**
* Usuarios: **Y5010343Z**, **Y5010344S**, **Y5010345Q**, **Y5010346V**, **Y5010347H**
  + Contraseña: **Entra001**

Por defecto, para un despliegue local, podrá acceder a la aplicación de pruebas desde la URL [*https://localhost:8443/fire-test-jsp/Login.jsp*](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. También es probable que tenga que desactivar la comprobación de certificados SSL de AutoFirma a través de su panel de preferencias.

* + 1. Guía de la página de prueba

La página de pruebas requiere que se inserte el usuario. Si se ha configurado en el componente central 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.

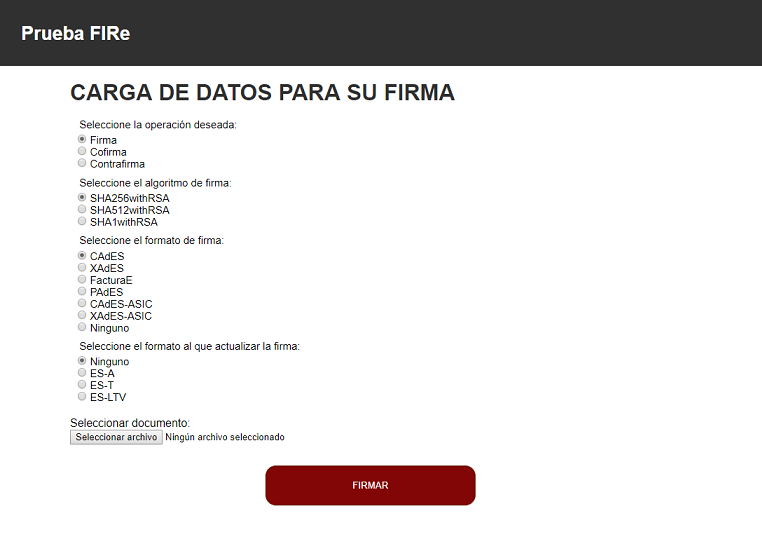


Una vez accedemos con el usuario deseado, se nos presentará una página en la que podremos elegir si queremos probar una operación de firma simple o una operación de firma de lote:



* + - 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 que el componente central se conecte con una plataforma de validación de firmas, como 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 las peticiones). 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 de FIRe 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 pantalla.



En caso de haber seleccionado el uso de los certificados de un proveedor de firma en la nube, seremos redirigidos 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.

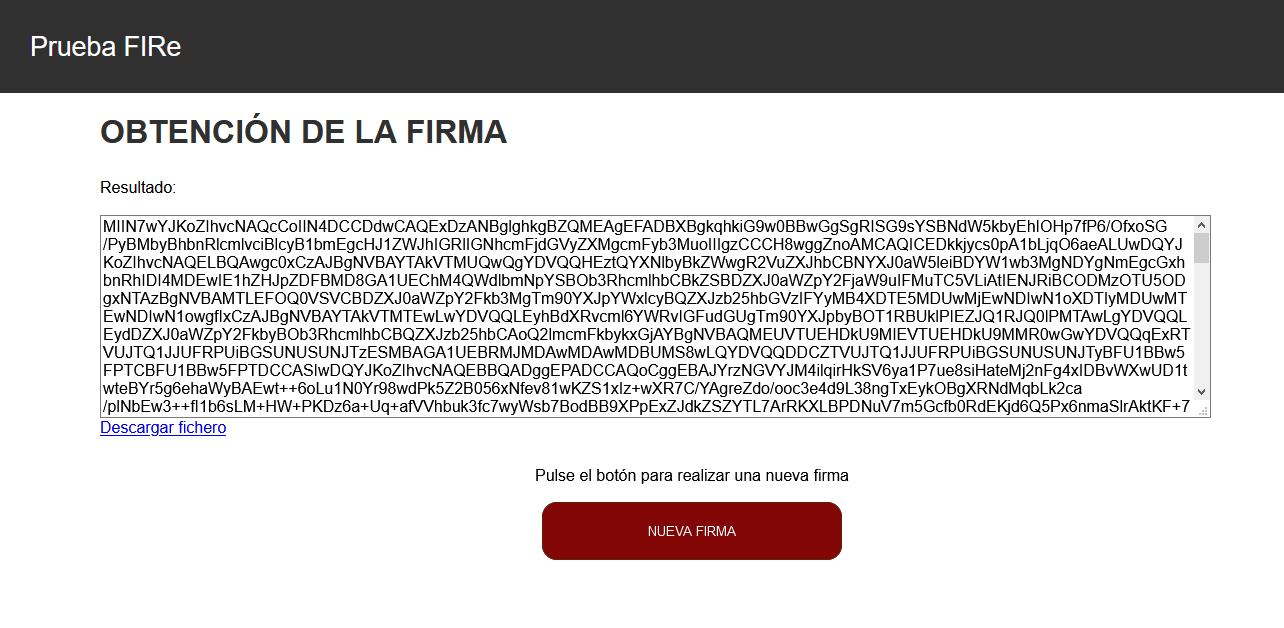


En caso de haber seleccionado el uso de un certificado local, seremos redirigidos a otra página en la que se cargará el Cliente @firma y se nos mostrará un botón de firma con el que iniciar la operación.



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, salvo que el tamaño de la firma haya excedido un tamaño predefinido, en cuyo caso se omite para evitar bloquear la página al imprimir este dato en el área de texto.



Independientemente de que se muestre o no la firma, se proporciona el enlace “Descargar fichero” para facilitar el guardado de la firma.

* + - 1. 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 las peticiones), 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. En caso de haber seleccionado la casilla de “Configuración particular”, se llamará a la version del método que proporciona una configuración particular de la configuración de firma. En caso contrario, se llamará a la versión que no proporciona la configuración, para que se aplique la configuración general del lote.

Tras agregar un documento al lote, 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 de la pasarela de FIRe desde la que se podrá elegir el origen del certificado (certificado de Cl@ve Firma, certificado local, etc.). 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 un proveedor de firma en la nube, 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 Cl@ve Firma o del proveedor que corresponda, desde la que deberemos autorizar la operación de firma. Los datos solicitados pueden variar de un proveedor de otro, pero comúnmente se nos pedirá la contraseña del usuario y, quizás, una clave OTP que se recibirá en el móvil del usuario.



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



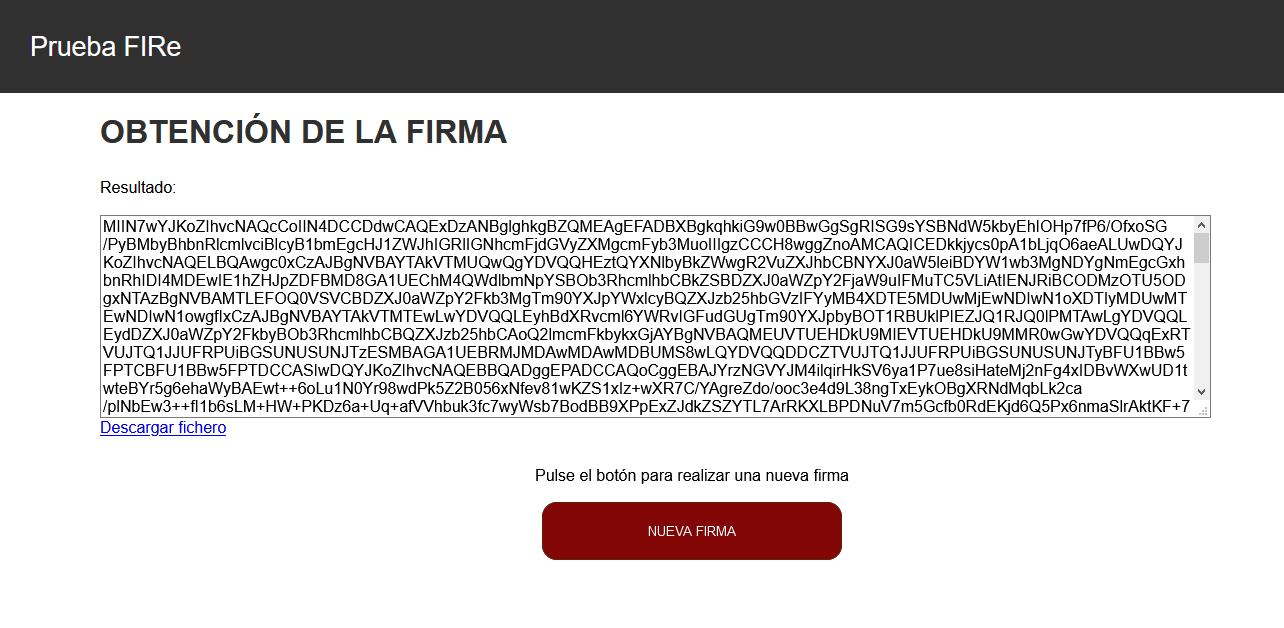
Independientemente de que se utilice un certificado de Cl@ve Firma/Servicio de pruebas o un certificado local, al finalizar correctamente la operación seremos redirigidos a la página de nuestra aplicación que hubiésemos indicado en la configuración proporcionada al método de creación del lote. En esta 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 general 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:



1. 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.
* **INVALID\_SIGNATURE**: La firma generada no es válida, probablemente debido al uso de un certificado no válido o a que el documento ya contenía una firma no válida.
* **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.

1. 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). |
| 9 | El gestor de documentos indicado no es valido | Se solicitó el uso de un gestor de documentos no válido o para el que no se tiene permiso de uso. |
| 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 haber realizado un registro débil | El usuario no tiene certificados emitidos por el proveedor seleccionado, ni podrá tenerlos hasta que no se cumpla alguna 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. |
| 106 | El usuario ya dispone de un certificado del tipo que se está solicitando generar | Se ha ejecutado la lógica de generación de certificados de firma para el usuario por medio de un proveedor que lo soporta, pero el proceso ha fallado porque el usuario ya disponía de ese tipo de certificados. Se debería reintentar la operación de firma, ya que en esta ocasión sí deben aparecerle al usuario sus certificados de firma. |
| 107 | Error en la generación de un nuevo certificado | Se ha tratado de generar un nuevo certificado de firma para el usuario por medio de un proveedor que lo soporta, pero ha fallado la generación del certificado. Se debería intentar completar la firma con un certificado local o de otro proveedor y el usuario debería contactar con el proveedor de firma para identificar el problema. |
| 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. |
| 251 | No se completó correctamente la firma del lote con certificado local | Ocurre cuando se produce un error al realizar una firma de lote con el Cliente @firma. |

1. Configuración de los formatos de firma

En este apartado se reproducen los listados de propiedades que se pueden utilizar en el parámetro prop/propB64 de los métodos sign, createBatchProcess y addDocumentToBatch para cada uno de los formatos de firma soportados. Estos parámetros permiten configurar las firmas generadas, pero no son obligatorios en ningún caso.

* 1. Formato CAdES
     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. |

* + 1. 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 | [Valor en 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. |
| 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). |
| 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). |

* 1. Formato XAdES
     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. Se ignora en la operación de cofirma.  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) |
| uri***n*** | [URI] | URI que referencia a los datos que se desean firma dentro de una firma *manifest*. ‘***n***’ indica el número de referencia de entre las que se quieren firmar, empezando en ‘1’. |
| md***n*** | [Texto Base64] | MIME-Type de los datos asociados a la referencia ‘***n***’ en una firma *manifest*. Si no se indica este parámetro, se utilizará el tipo ‘application/octet-stream’. |
| mimeType***n*** | [Texto en formato MIME-Type] | MIME-Type de los datos asociados a la referencia ‘***n***’ en una firma *manifest*. Si no se indica este parámetro, se utilizará el tipo ‘application/octet-stream’. |
| contentTypeOid***n*** | [OID o URN de tipo OID] | Identificador del tipo de dato firmado para la referencia número ‘***n***’ en una firma *manifest*. Este parámetro es complementario (que no excluyente) al parámetro mimeTypeX. |
| encoding***n*** | [URI] | Codificación de los datos asociados a la referencia número ‘***n***’ en una firma manifest. Un uso incorrecto de este parámetro puede provocar la generación de una firma inválida. |
| precalculatedHashAlgorithm | SHA1 | Indica que las huellas digitales de los datos referenciados en el *manifest* se calcularon mediante el algoritmo SHA1. |
| SHA-256 | Indica que las huellas digitales de los datos referenciados en el *manifest* se calcularon mediante el algoritmo SHA-256. |
| SHA-384 | Indica que las huellas digitales de los datos referenciados en el *manifest* se calcularon mediante el algoritmo SHA-384. |
| SHA-512 | Indica que las huellas digitales de los datos referenciados en el *manifest* se calcularon mediante el algoritmo SHA-512. |
| 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*. |
| XAdES Externally Detached | Genera firmas en formato *Externally 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 | [Valor en 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. |
| <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. Este es el comportamiento por defecto. |
| 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 Base64 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 Base64 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 auto-detectar 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#](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 Base64 incluso si son necesarias. |
| false | Declara las transformaciones Base64 cuando se han codificado internamente los datos a firmar en Base64 (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 Base64 y se debe declarar esta transformación adicional para que se decodifiquen antes de firmarlos. Esta transformación Base64 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). |

* + 1. 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 | [Texto 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). |

* + 1. Firma manifest

FIRe es compatible con las firmas XAdES con *manifest*. Este tipo de firma no contienen los datos, sino que guarda referencia a ellos mediante una URI (ya sea una URN o URL) junto con su huella digital. Las firmas *manifest* soportadas por FIRe, además, pueden guardar referencias a múltiples datos simultáneamente.

Una particularidad de las firmas *manifest* es que no requieren los datos ni en el momento de la firma ni en el de validación, sólo la referencia y el hash de los mismos. Es responsabilidad del integrador garantizar que la referencia y la huella son correctas en el momento de realizar la firma y, en el momento de la validación, que esos datos no han cambiado. Con esto, FIRe no es necesario enviar a FIRe los datos a firmar. Sin embargo, por compatibilidad con versiones anteriores y seguridad, no es posible proporcionar datos nulos a los métodos de firma y de agregar nuevos documentos a lotes. En caso de querer hacer una firma manifest, habrá que proporcionar un dato cualquiera que luego se obviará durante la firma. Por ejemplo, podríamos pasar la cadena “OK” al método de firma del cliente distribuido.

Las firmas *manifest* no requieren datos, por lo que no les afecta el gestor de documentos que pueda tener configurado FIRe. Si se configurase un gestor de documentos distinto al por defecto, idóneamente este debería ser capaz de obviar los datos que no deben recuperarse por tratarse de una firma *manifest* (identificando la cadena enviada a firmar o algún otro parámetro que lo señalice), para así evitar que se intenten recuperar datos innecesarios.

Incluso si incluye referencias a múltiples datos, una firma *manifest* genera una única firma y, por lo tanto, deberá generarse utilizando el procedimiento de firma simple. La firma de lotes sólo se utilizará si queremos generar múltiples firmas, ya sea de una o de varias referencias.

Para configurar el uso de firmas *manifest* se utilizarán algunas de las propiedades de configuración de firma listadas en el apartado IV.2.1 Firma y cofirma.

La información de las distintas referencias deberá aportarse por cada dato referenciado, para lo que se sustituirá la ‘X’ del nombre de cada uno de los siguientes parámetros por el número de referencia en cuestión (empezando en 1):

* uriX
  + Obligatorio. URI que referencia a los datos.
* mdX
  + Obligatorio. Huella digital (hash) en Base64 de los datos. El algoritmo de huella digital se indicará mediante el parámetro “precalculatedHashAlgorithm”.
* mimeTypeX
  + Opcional. MimeType correspondiente a los datos referenciados. Si no se indica, se usará “application/octet-stream”.
* contentTypeOidX
  + Opcional. OID correspondiente al tipo de dato referenciado. Si no se indica, se intentará extrapolar a partir del MimeType. En caso de no conseguirlo, no se usará ninguno.
* encodingX
  + Opcional. URI identificadora de la codificación de los datos si estuviesen codificados. Por defecto, no se usa ninguna.
* precalculatedHashAlgorithm
  + Algoritmo de huella digital que se ha utilizado para calcular la huella de los datos. Este parámetro aplica a todas las referencias insertadas. Si no se indica, se hereda la configuración del parámetro “referencesDigestMethod”. Si este otro parámetro tampoco se indicase se interpretará que las huellas digitales se han calculado con el algoritmo SHA-512. Este parámetro determina el algoritmo usado en todas las referencias.

Como ejemplo, podríamos hacer una firma *manifest* de un par de datos usando las siguientes propiedades en el método de firma:

|  |
| --- |
| format=XAdES Enveloping  useManifest=true  uri1=urn:id:3086  md1=4hrn/3Y9c/fn/uyq12w+D9A2aKc=  mimeType1=plain/xml  uri2=https://administracionelectronica.gob.es/  md2=/U5j6SLFdchtPHw6vlety2qLz/M=  precalculatedHashAlgorithm=SHA-1 |

Como resultado, obtendremos una firma XAdES Enveloping de *manifest* con las referencias y huellas indicadas.

* 1. 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. |
| 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. |
| signerClaimedRoles | emisor | Declara que el firmante es el emisor de la factura. Este es el valor por defecto. |
| receptor | Declara que el firmante es el receptor de la factura. |
| tercero | Declara que el firmante es un tercero con respecto a la factura. |
| supplier | Declara que el firmante es el emisor de la factura. Este es el valor por defecto. |
| customer | Declara que el firmante es el receptor de la factura. |
| third party | Declara que el firmante es un tercero con respecto a la factura. |
| policyIdentifier | http://www.facturae.es/politica\_de\_firma\_formato\_facturae/politica\_de\_firma\_formato\_facturae\_v3\_1.pdf | Identificador de la política de firma 3.1. Este es el valor por defecto. |
| http://www.facturae.es/politica de firma formato facturae/politica de firma formato facturae v3\_0.pdf | Identificador de la política de firma 3.0. |
| policyIdentifierHash | Ohixl6upD6av8N7pEvDABhEL6hM= | Huella digital para configurar la política de firma 3.1. Este es el valor por defecto. |
| xmfh8D/Ec/hHeE1IB4zPd61zHIY= | Huella digital para configurar la política de firma 3.0. |
| policyIdentifierHashAlgorithm | SHA1 | Indica que la huella digital indicada en el parámetro policyIdentifierHash se calculó mediante el algoritmo SHA1. |

* 1. Formato PAdES

|  |  |  |
| --- | --- | --- |
| **Nombre del parámetro** | **Valores posibles** | **Descripción** |
| 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. |
| alwaysCreateRevision | true | Siempre creará una revisión al firmar. Requiere que el documento cumpla la especificación PDF 1.7 (ISO 32000-1:2008) |
| false | No creará revisión en la primera firma y sí en las siguientes. |
| image | [Texto Base64] | Imagen JPEG que insertar en el PDF. |
| imagePage | [Entero positivo] | Insertar imagen en el número de página indicado. |
| 0 | Insertar en todas las páginas. |
| -1 | Insertar imagen en la última página. |
| imagePositionOnPageLowerLeftX | [Entero positivo] | Coordenada horizontal desde la esquina inferior izquierda de la página a la esquina inferior izquierda de la imagen. |
| imagePositionOnPageLowerLeftY | [Entero positivo] | Coordenada vertical desde la esquina inferior izquierda de la página a la esquina inferior izquierda de la imagen. |
| imagePositionOnPageUpperRightX | [Entero positivo] | Coordenada horizontal desde la esquina inferior izquierda de la página a la esquina superior derecha de la imagen. |
| imagePositionOnPageUpperRightY | [Entero positivo] | Coordenada vertical desde la esquina inferior izquierda de la página a la esquina superior derecha de la imagen. |
| attach | [Texto Base64] | Contenido a añadir como adjunto al PDF. Requiere establecer attachFileName. |
| attachFileName | [Texto] | Nombre del que asignar al fichero adjunto. |
| attachDescription | [Texto] | Descripción del documento adjunto. |
| 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 | Comprime el PDF firmado para que ocupe menos tamaño. Sólo se aplica si se trata de un PDF v4 o superior. Este es el valor por defecto. |
| false | Nunca se comprime el PDF firmado. |
| pdfVersion | 2 | Se declara que la versión del PDF de salida es 1.2. |
| 3 | Se declara que la versión del PDF de salida es 1.3. |
| 4 | Se declara que la versión del PDF de salida es 1.4. |
| 5 | Se declara que la versión del PDF de salida es 1.5. |
| 6 | Se declara que la versión del PDF de salida es 1.6. |
| 7 | Se declara que la versión del PDF de salida es 1.7. |
| signatureSubFilter | [Texto] | Subfiltro declarado. Por defecto se utiliza el de las firmas básicas (adbe.pkcs7.detached). Puede usarse la cadena “ETSI.CAdES.detached” para crear firmas BES y *baseline* B-Level. |
| signatureField | [Texto] | Nombre del campo de firma preexistente en el que insertar la firma. |
| signaturePage | [Entero positivo] | Se genera una firma visible PDF en el número de página indicado. |
| -1 | Se genera una firma visible PDF en la última página. Este es el comportamiento por defecto. |
| -2 | Agrega una nueva página al documento y agrega la firma visible PDF en ella. |
| signaturePositionOnPageLowerLeftX | [Entero positivo] | Coordenada horizontal desde la esquina inferior izquierda de la página a la esquina inferior izquierda del campo de firma visible. |
| signaturePositionOnPageLowerLeftY | [Entero positivo] | Coordenada vertical desde la esquina inferior izquierda de la página a la esquina inferior izquierda del campo de firma visible. |
| signaturePositionOnPageUpperRightX | [Entero positivo] | Coordenada horizontal desde la esquina inferior izquierda de la página a la esquina superior derecha del campo de firma visible. |
| signaturePositionOnPageUpperRightY | [Entero positivo] | Coordenada vertical desde la esquina inferior izquierda de la página a la esquina superior derecha del campo de firma visible. |
| signatureRubricImage | [Texto Base64] | Imagen JPEG que mostrar en el campo de firma visible. |
| layer2Text | [Texto] | Texto que mostrar en el campo de firma visible. |
| layer2FontFamily | 0 | El texto de la firma visible se mostrará con fuente Courier. Este es el valor por defecto. |
| 1 | El texto de la firma visible se mostrará con fuente Helvética. |
| 2 | El texto de la firma visible se mostrará con fuente Times Roman. |
| 3 | El texto de la firma visible se mostrará con fuente Symbol. |
| 4 | El texto de la firma visible se mostrará con fuente ZapfDingBats. |
| layer2FontSize | [Entero positivo] | Tamaño de fuente del texto de la firma visible. |
| layer2FontStyle | 0 | Texto de la firma visible sin estilo. Valor por defecto. |
| 1 | Texto de la firma visible en negrita. |
| 2 | Texto de la firma visible en cursiva. |
| 4 | Texto de la firma visible subrayado. |
| 8 | Texto de la firma visible tachado. |
| layer2FontColor | black | El texto de la firma visible será de color negro. Este es el valor por defecto. |
| white | El texto de la firma visible será de color blanco. |
| gray | El texto de la firma visible será de color gris. |
| lightGray | El texto de la firma visible será de color gris claro. |
| darkGray | El texto de la firma visible será de color gris oscuro. |
| red | El texto de la firma visible será de color rojo. |
| pink | El texto de la firma visible será de color rosa. |
| signReason | [Texto] | Razón por la que se realiza la firma. |
| signatureProductionCity | [Texto] | Ciudad en la que se realiza la firma. |
| signerContact | [Texto] | Información de contacto del firmante. |
| signerClaimedRoles | [Texto] | Listado de roles declarados por el firmante (separados por “|”) |
| 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 | [Texto 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 hacia el documento que contiene una descripción textual de la política de firma. |
| ownerPassword | [Texto] | Contraseña de apertura del PDF. No se soporta la firma de documentos PDF cifrados con certificados o algoritmo A256. |
| headless | true | No interrumpe el proceso de firma solicitando interacción del usuario. |
| false | Muestra diálogos al usuario si requiere de su autorización o algún dato adicional para firmar. Este es el valor por defecto. |
| allowSigningCertifiedPdfs | true | Permite la firma de documentos PDF certificados. |
| false | Produce un error al firmar documentos PDF certificados. |
| Se omite | Consulta al usuario si desea firmar. |
| allowCosigningUnregisteredSignatures | true | Permite firmar documentos PDF con firmas previas no registradas. |
| false | No permite firmar documentos PDF con firmas previas no registradas. |
| signingCertificateV2 | true | Se utiliza el atributo signingCertificateV2 en las firmas. |
| false | Se utiliza el atributo signingCertificateV1 en las firmas. |
| Se omite | Se utiliza el atributo signingCertificateV1 en las firmas SHA1withRSA y signingCertificateV2 en el resto. |

1. 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.

Los filtros aquí expuestos configuran el comportamiento de AutoFirma para que muestre sólo aquellos certificados que se deseen. Aunque la mayoría de estos filtros son comunes a las distintas versiones de AutoFirma, puede haber pequeñas diferencias entre algunas de ellas.

Estos filtros no se tendrán en cuenta cuando se realice la firma con certificado local a través de dispositivos móviles y las apps “Cliente de firma Android” y “Cliente de Firma iOS”. En estos casos, siempre se mostrarán todos los certificados disponibles por la aplicación.

Los filtros de certificados se configurarán a través del parámetro de configuración del formato de los métodos sign y createBatchProcess. 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:false*
    - Mostrar también certificados 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 mostrará un diálogo de selección vacío desde el que el usuario podrá seleccionar un nuevo almacén de claves local en el que buscar.

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

1. Migración a FIRe 2.4
   1. Migración de aplicaciones integradas con FIRe

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 2.4. Para ello, deberán sustituir su componente actual (JAR, DLL o fichero PHP) por el nuevo.

Las aplicaciones que utilicen FIRe 2.2 o 2.3 y los componentes distribuidos .NET y Java, no tendrán que actualizar al nuevo componente distribuido, aunque se recomienda para el uso de las nuevas capacidades de este.

Las aplicaciones que utilicen el componente distribuido PHP deberían actualizar para incorporar el nuevo API “fire\_client.php”. Las aplicaciones que utilicen el componente distribuido Java de FIRe 2.2, 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.

* + 1. Migración de aplicaciones PHP

FIRe 2.4 introduce el componente distribuido PHP “fire\_client.php”. Esto obligará a construir el objeto FireClient y realizar las peticiones de firma a través del mismo pasando en cada llamada el identificador del usuario.

Quien desee incorporar el nuevo componente deberá realizar la integración tal como se describe en el apartado 7 Integración del componente distribuido PHP.

Las aplicaciones que deseen mantener el API “fire\_api.php” deberán realizar cambios en todas las llamadas salvo la de firma y creación de lote de firma. En cada una de las llamadas se deberá incorporar un parámetro con el identificador del usuario, situado siempre tras el parámetro del identificador de la aplicación. Los métodos modificados quedan como sigue:

* function recoverSign($appId, $subjectId, $transactionId, $upgrade=null)
* function recoverError($appId, $subjectId, $transactionId)
* function addDocumentToBatch($appId, $subjectId, $transactionId, $documentId, $documentB64, $confB64)
* function addCustomDocumentToBatch($appId, $transactionId, $documentId, $documentB64, $op, $ft, $propB64, $upgrade, $confB64)
* function signBatch($appId, $subjectId, $transactionId, $stopOnError)
* function recoverBatchResult ($appId, $subjectId, $transactionId)
* function recoverBatchResultState($appId, $subjectId, $transactionId)
* function recoverBatchSign($appId, $subjectId, $transactionId, $docId)

Las aplicaciones que mantengan el uso del API “fire\_api.php” no podrán hacer uso de las nuevas funcionalidades de FIRe 2.4 y superiores.

* 1. Migración de aplicaciones integradas con Cl@ve Firma

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