Kit de integración de firma con FIRe

Manual de instalación y despliegue

Versión: 2.3

CONTROL DE VERSIONES	
Título	Manual de instalación y despliegue
Autor	Secretaría General de Administración Digital Ministerio de Política Territorial y Función Pública
Fecha versión 2.3	11 de octubre de 2018

Versión	Fecha	Responsable	Cambios introducidos
1.0	11-01-2016	DTIC	Creación del documento
1.0 Rev 1	16-02-2016	DTIC	Actualización de las propiedades de configuración
1.0 Rev 2	17-02-2016	DTIC	Adaptación al cambio de nombre a Cl@ve Firma
1.0 Rev 3	24-02-2016	DTIC	Actualización de las sentencias de bases de datos
1.0 Rev 4	18-03-2016	DTIC	Cambio de contraseña del certificado de prueba
1.0 Rev 5	18-04-2016	DTIC	Configuración del directorio de configuración y correcciones varias
1.0 Rev 6	28-04-2016	DTIC	Se renombra el conector al backend de custodia de claves
1.0 Rev 7	25-05-2016	DTIC	Se agrega la propiedad de configuración del ProviderName del servicio.
1.0 Rev 8	30-05-2016	DTIC	Actualización de los componentes distribuidos .Net y PHP.
1.0 Rev 9	13-06-2016	DTIC	Actualización del componente distribuido PHP.
1.0 Rev 10	19-01-2017	SGAD	Aclaración sobre la carga de la configuración.
1.1	07-02-2017	SGAD	Aclaración sobre el componente distribuido PHP.
1.1 Rev 1	23-03-2017	SGAD	Se actualizan las claves de los certificados de prueba y se agregan los problemas comunes.
2.0	12-05-2017	SGAD	Se actualiza la información a la de FIRe.
2.0 Rev 1	10-07-2017	SGAD	Aclaraciones en el apartado de despliegue.
2.1	10-10-2017	SGAD	Se documentan los requisitos para el uso de un gestor de documentos de FIRe y la configuración de la gestión compartida de sesiones entre varios servidores balanceados.
2.1.1	28-11-2017	SGAD	Se documenta la propiedad para la configuración del contexto público del componente central.
2.2	16-05-2018	SGAD	Documentación de los cambios de la nueva versión y revisión general.

2.3	11-10-2018	SGAD	Cifrado de contraseñas, información de migración y consideraciones adicionales.

ÍNDICE

1 OBJETO DEL DOCUMENTO

El presente manual detalla la arquitectura de FIRe, los requisitos software de su kit integración y los pasos a seguir para su despliegue y configuración.

2 INTRODUCCIÓN

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

FIRe se compone principalmente de un API cliente (Componente distribuido) y unos servicios de firma en servidor (Componente Central). Por medio de este API cliente es posible invocar a funciones para la firma de uno o varios documentos (firma de lote) y posteriormente recuperar el resultado de estas operaciones. La principal ventaja de FIRe es que permite a los usuarios firmar tanto con sus certificados locales como con sus certificados en la nube sin que la aplicación que lo utiliza tenga que definir flujos de trabajo distintos para cada una de estas opciones.

Para que el componente central admita las peticiones realizadas por una aplicación, esta deberá haberse registrado en el sistema. Este registro puede hacerlo un administrador a través del módulo de administración de FIRe, mediante el cual dará de alta la nueva aplicación, establecerá el certificado con el que deberá autenticarse y obtendrá como resultado el código alfanumérico que deberá utilizar el componente distribuido como identificador de aplicación (`AppId`).

Opcionalmente, si sólo habrá una única aplicación dada de alta en el sistema, es posible omitir el despliegue y uso de una base de datos y el componente de administración. En este caso, el administrador del componente central dará de alta el certificado cliente que debe usar la aplicación y un identificador de aplicación cualquiera directamente en el fichero `config.properties`.

Los certificados en la nube utilizados por FIRe son los certificados de firma de Cl@ve Firma. Estos certificados están custodiados por el Cuerpo Nacional de Policía (CNP) y se hace uso de los mismos por medio de la pasarela de la Gerencia Informática de la Seguridad Social (GISS).

FIRe, por defecto, permitirá a los usuarios seleccionar el origen de los certificados de firma (local o cualquiera de los proveedores de firma en la nube), pero una aplicación puede configurar que se utilice directamente un proveedor si desea que el usuario firme con unos certificados concretos.

En caso de que el usuario seleccione un proveedor y no tenga certificados emitidos por él, si el proveedor lo soporta, FIRe permitirá emitir nuevos certificados al usuario y realizar con ellos la operación de firma. Esto se realizará de forma totalmente transparente para la aplicación.

3 ARQUITECTURA SOFTWARE

El sistema de firma FIRe cuenta con un componente central encargado de atender las distintas solicitudes de firma. Este componente central puede dar servicio a múltiples aplicaciones cliente, que harán uso del API o componente distribuido de FIRe.

Cada una de estas aplicaciones cliente deberá autenticarse ante el componente central mediante certificado e identificador de aplicación. El componente central almacena la información de autenticación de cada una de las aplicaciones cliente en base de datos. Para asistir al administrador del sistema en esta tarea, existe un componente de administración cuya finalidad es exclusivamente gestionar qué aplicaciones cliente pueden conectar con el componente central.

En el caso excepcional en el que sólo se fuese a dar servicio a una única aplicación cliente, sería posible omitir el uso de la base de datos, haciendo innecesario el uso de la herramienta de administración. Para esto se incluiría la información de autenticación en uno de los ficheros de configuración del componente central en lugar de en la base datos.

Las peticiones al componente central se realizan por medio de un componente cliente o distribuido, que el integrador podrá utilizar a modo de API desde su aplicación. La comunicación entre el componente distribuido y el componente central siempre se debe realizar sobre una conexión SSL con autenticación cliente. El certificado SSL cliente que utilice la aplicación es el que usará el componente central para autenticarla. Para permitir este modelo de autenticación, el servidor de aplicaciones no debe restringir las peticiones según el certificado cliente utilizado. Será el propio componente central el que lo haga.

Existen implementaciones del componente distribuido en lenguaje Java, .Net y PHP, de tal forma que un integrador podrá elegir el que mejor se ajuste a su aplicación.

FIRe permite la firma electrónica de datos mediante certificados en los almacenes locales del usuario y mediante los certificados en la nube de diversos proveedores.

Las firmas mediante certificados locales se realizan mediante el Cliente @firma. Este permitirá al usuario utilizar los certificados del almacén interno de su navegador web y de dispositivos criptográficos externos (como el DNIe). En el caso de contar con PIN o contraseña el almacén de claves seleccionado, será el propio almacén (o el Cliente @firma cuando se delegue esta tarea) el que se encargue de pedírselo al usuario.

En el caso de las firmas con certificados en la nube, FiRe incorpora los conectores de varios proveedores, aunque el administrador del sistema es libre de incorporar otros nuevos.

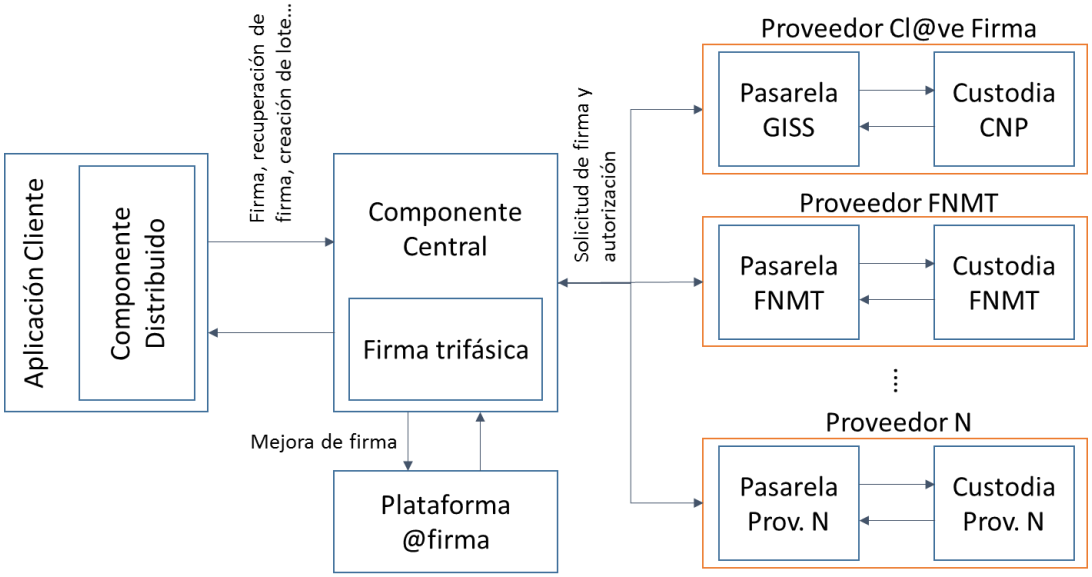
Entre los conectores por defecto de FIRe, destaca el de Cl@ve Firma. Cl@ve Firma utiliza certificados de firma custodiados por el Cuerpo Nacional de Policía (CNP) a los que accede a través de la pasarela proporcionada por la Gerencia Informática de la Seguridad Social (GISS). En este caso el usuario deberá autorizar las operaciones de firma mediante la inserción de su PIN y una clave OTP desde una web de la GISS a la que se le redirigirá.

FIRe también incorpora un conector para el uso de certificados de funcionario de la FNMT y un conector a un servicio de pruebas que emula el comportamiento de Cl@ve Firma. Con este último, las distintas aplicaciones pueden probar su integración con FIRe en entornos distintos a los de producción.

Todas las firmas realizadas por FIRe se realizan en 3 fases, donde la primera fase (composición de los datos a firmar) y la tercera fase (composición de la firma electrónica) se realizarán en el componente central, mientras que la segunda fase (firma digital), que es la única que involucra a la clave privada del certificado, se realiza en el equipo del usuario, en el caso de la firma local; o en los servidores proveedor, en el caso de la firma con certificado remoto. Todo este proceso es transparente para las aplicaciones que integran FIRe.

Si así lo indica la aplicación cliente, es posible que tras realizar una firma electrónica deba actualizarse ésta a un formato longevo. Esto es, incrustarle información adicional como, por ejemplo, un sello de tiempo o información de revocación. Estas actualizaciones de firma se realizan por medio de la Plataforma @firma, por lo que el componente central deberá tener acceso a una instancia de la Plataforma @firma si desea permitir que las aplicaciones cliente realicen mejoras de firma.

En el siguiente diagrama se ilustra la arquitectura básica del sistema:



Los componentes software que se incluyen en el kit de integración y su correspondencia con los bloques del diagrama son los siguientes:

- Componente central:
  - `fire-signature.war`

- Componente distribuido (librería de integración con las aplicaciones):
  - Java: `fire-client-2.2.jar`
  - .Net: `fire_client.dll`
  - PHP: `fire_api.php`

- Aplicación cliente de demostración (incluye componente distribuido java):

- `fire-test-jsp.war`

- Simulador de firma centralizada:
  - `clavefirma-test-services.war`

- Aplicación de Administración del Componente central:
  - `fire-admin-jsp.war`

4 COMPONENTE CENTRAL

El objetivo del componente central es concentrar en un solo elemento todas las funciones de firma que tengan las aplicaciones de un organismo. De esta forma, las conexiones con las distintas plataformas y proveedores (Plataforma @firma, GISS, CNP…) sólo deben configurarse desde este componente.

A continuación, se detallan los diferentes aspectos que deben tenerse en cuenta para desplegar y configurar el componente central.

4.1 DEPENDENCIAS EXTERNAS

El componente central de FIRe debe tener acceso a los siguientes sistemas externos:

- **Pasarela GISS de custodia de certificados:**
  - Es el componente desplegado en los entornos de la Gerencia Informática de la Seguridad Social que gestiona el acceso a los almacenes de clave del Cuerpo Nacional de Policía.
  - Se deberá dar acceso sólo si se configura el proveedor de Cl@ve Firma.
  - La URL de acceso será proporcionada por la GISS una vez apruebe la conexión del organismo.
  - Adicionalmente, los ciudadanos pueden verse redirigidos al dominio "*clave-dninbpcert.policia.gob.es*" del CNP para realizar determinadas operaciones (generación de certificados, cambio de contraseñas…). Si los usuarios de su aplicación deben acceder a través de la red del organismo, asegúrese de que tienen acceso a este dominio.
- **Plataforma @firma:**
  - Se utiliza para la mejora de las firmas generadas (creación de firmas longevas).
  - El acceso a la Plataforma @firma es opcional. Sólo se accederá a ella si se solicita la creación de firmas longevas desde la aplicación.
  - La URL y el resto de datos de acceso serán proporcionados por el organismo propietario de la instancia de la Plataforma @firma a la que se desea acceder.
- **Google Analytics:**
  - Es el sistema de Google para la recopilación de estadísticas. Mediante él podemos obtener mediciones acerca de cuantas llamadas se realizan a los distintos servicios de firma centralizada.
  - El acceso a este sistema es opcional y puede ser deshabilitado, renunciando a la recopilación de estadísticas. Para deshabilitar el uso de Google Analytics basta con no configurar el identificador de aplicación de Google Analitycs.
  - El acceso a este sistema se realiza a través del dominio:
    - *www.google-analytics.com*
- **Otros entornos:**
  - FIRe permite dar de alta otros proveedores de firma en la nube. Cada proveedor tendrá requisitos de acceso a distintos servicios. Consulte la documentación de los proveedores configurados para conocer sus requisitos de acceso.

- FIRe permite definir a través del fichero de configuración del componente central un conjunto de clases gestoras de documentos que recuperen los datos solicitados por aplicaciones cliente y guarden las firmas generadas. Estas clases se implementarán conforme a las necesidades de las aplicaciones cliente y pueden requerir el acceso a distintos entornos. El proveedor de estas clases gestoras deberá especificar a qué entornos debe acceder para que se le habiliten los permisos necesarios.
  - Las clases gestoras incluidas por defecto en FIRe no requieren que se habilite ningún acceso adicional.

Los administradores del sistema en el que se despliegue el componente central deben habilitar el acceso de red a estos sistemas.

4.2 DESPLIEGUE

El componente central de FIRe expone una serie de servicios que sólo deben estar disponibles en el entorno en el que se encuentren las aplicaciones web que se conecten a él por medio de los componentes distribuidos. Sin embargo, es necesario que otros servicios y recursos del componente central estén disponibles de cara a los usuarios finales y, por tanto, que se despliegue en un entorno público o DMZ.

Los recursos y servicios que pueden solicitarse desde los entornos cliente son aquellos que se encuentran en la ruta "`public`" del módulo "`fire-signature`". Esto significa que se debe configurar el que, por defecto, sería el contexto "fire-signature/public" para que sea visible desde el exterior.

Por otra parte, es necesario configurar el servidor de aplicaciones para que los accesos al componente central sólo puedan realizarse sobre SSL con certificado cliente. Con esto se consigue que sólo puedan hacer uso de sus servicios las aplicaciones que se hayan dado de alta en el sistema y se autentiquen mediante el certificado cliente con el que se registrasen.

Sin embargo, las mismas páginas web, servicios y recursos del componente central que se publican de cara al usuario (ruta "`public`" del módulo "`fire-signature`") deberán ser accesibles sin necesidad de autenticación, por lo que se deberá configurar una conexión SSL sin autenticación cliente para el acceso a ellos. Si se requiriese certificado cliente para acceder a estos recursos, el navegador web o Java se lo pediría directamente al usuario, dando lugar a un diálogo de selección que puede confundirle. Esto significa que se debe configurar el que, por defecto, sería el contexto "fire-signature/public" para que no solicite certificados cliente. Disponer esta configuración podría requerir del uso de un servidor web.

En el contexto “fire-signature/public” también se despliegan algunos servicios que requiere el Cliente @firma para su funcionamiento. Si en nuestro despliegue de FIRe habilitamos el proveedor “`local`” para permitir que las aplicaciones habiliten los procesos de firma con certificado local, es importante tener en cuenta que el Cliente @firma comprueba la validez de los certificados SSL. Para evitar incidencias al respecto, debemos utilizar en nuestro despliegue un certificado SSL válido, correspondiente al dominio en el que se realiza el despliegue y reconocido por defecto por Java. En caso contrario, es posible que AutoFirma no pueda conectarse con los servicios del componente central de FIRe. Como alternativa, sobre todo para los entornos de desarrollo, el usuario podría dar de alta la URL del contexto “/public” en el que se encuentre la aplicación en el listado de sitios seguros de la configuración de Java de su equipo. De esta forma, el Cliente @firma no tendrá problemas para acceder a los servicios del componente central.

Por ejemplo, si tuviésemos un servidor de aplicaciones Apache Tomcat y, como frontal, el servidor web Apache HTTPD con *virtualhosts* configurados, podríamos establecer esta configuración de la siguiente manera:

```
…
  SSLEngine on
  SSLVerifyClient none

  # Habilitamos la autenticacion SSL cliente para acceso al componente central
  <Location "/fire-signature">
    SSLRenegBufferSize 20982000
    LimitRequestBody 20982000
    SecRequestBodyAccess Off
    SSLVerifyClient optional_no_ca
    SSLVerifyDepth 2
  </Location>

  # Deshabilitamos la autenticacion SSL cliente para el subcontexto publico
  <Location "/fire-signature/public">
    SSLVerifyClient none
  </Location>

  # Redirigir los certificados SSL cliente a Tomcat
  SSLOptions +ExportCertData
…
```

4.3 REQUISITOS SOFTWARE

- Oracle Java 7 o superior.
- Servidor de aplicaciones. Puede optarse por una de las siguientes opciones:
  - Servidor de aplicaciones JEE versión 6 o superior (JBoss, WebSphere, GlassFish, etc.)
  - Servidor de *servlets* de Java versión 2.4 o superior (Apache Tomcat, etc.)
- Sistema de Gestión de Base de Datos compatible JDBC (MySQL, Oracle, etc.). Esto es necesario sólo si deseamos que el componente central dé servicio a más de una aplicación.

Si el servidor de aplicaciones utilizado no nos permitiese dejar sin autenticación SSL cliente el subcontexto "/public", mientras que sí está activado para el resto, se debería contar con un servidor web como frontal que nos permita configurar los accesos de esta manera. Esto ocurre, por ejemplo, con Apache Tomcat, que para establecer esta configuración debería estar antecedido por un servidor web Apache, con el que se podría comunicar por AJP. Consulte el apartado Despliegue para más información sobre esto.

En caso de contar con un servidor de *servlets*, se desplegará la distribución de los servicios compuesta de archivos WAR.

En caso de contar con un servidor completo de aplicaciones JEE, se podrá desplegar la distribución compuesta de archivos WAR o el fichero EAR del proyecto.

Para saber cómo realizar el despliegue de aplicaciones en su servidor, consulte la documentación de su software.

Los controladores JDBC del sistema de gestión de base de datos elegido deberán estar instalados en el *classpath* del servidor de aplicaciones.

4.3.1 REQUISITOS DEL USUARIO

El entorno de ejecución del usuario de FIRe también debe cumplir con ciertos requisitos cuando se permita u obligue al uso de certificados locales. Estos requisitos, varían según la herramienta de firma que se configure. Los requisitos recomendados son:

- Navegador web:
  - Microsoft Internet Explorer
  - Microsoft Edge
  - Mozilla Firefox
  - Google Chrome
  - Apple Safari
- Java:
  - Java 8 o superior

En caso de que un usuario utilice Internet Explorer, la firma se intentará realizar mediante el MiniApplet @firma, en cuyo caso sólo es necesario disponer de Java 7 o superior. En caso de no disponer de él o que el usuario rechazase los permisos solicitados por el MiniApplet, se intentará utilizar AutoFirma.

Si en el componente central se ha configurado el uso de AutoFirma nativo, el usuario deberá tener esta aplicación instalada en el sistema. Si, en cambio, se ha configurado el uso de AutoFirma WebStart, el usuario deberá tener instalado Java 8 o superior.

IMPORTANTE: El usuario debe disponer de todo el software necesario antes de iniciar el trámite. Es responsabilidad de la aplicación que conecta con FIRe notificar al usuario esta necesidad, ya que es posible que una vez se acceda a las páginas de FIRe ya no se puedan instalar sin interrumpir el trámite.

En la página de firma con certificado local, se muestran al usuario los requisitos que necesita basándose en el entorno detectado.

Si desea exponer estos requisitos a sus usuarios antes de iniciar el trámite web, puede permitir mostrarles la página web localizada en "`public/static/miniapplet_compatibility.html`" dentro del componente central de FIRe:

4.4 Configuración – Ficheros de Propiedades

Los ficheros de propiedades pueden editarse con cualquier editor de textos, como el Bloc de Notas de Windows o el "vi" de UNIX.

Los siguientes ficheros de configuración se buscarán en el directorio configurado mediante la variable de entorno "`fire.config.path`" proporcionada a la JVM. Esto puede hacerse comúnmente a través de la variable JAVA_OPTS que carga el servidor de aplicaciones, a la que se le proporciona el valor `-Dfire.config.path="/directorio"`. Si no se configura, los ficheros se buscarán en el CLASSPATH del servidor de aplicaciones.

- `config.properties`
  - Configura las conexiones con la base de datos, ficheros temporales, comportamiento del Cliente @firma, el sistema de estadísticas y los proveedores que se pueden utilizar para firmar.
- `platform.properties`
  - Configura el acceso a la Plataforma @firma en caso de desear mejorar las firmas.

  - Este fichero sólo se utilizará cuando las aplicaciones soliciten la actualización a formatos longevos de las firmas generadas. No es necesario incorporarlo si nuestras aplicaciones no necesitan esto.
- *PROVEEDOR*_config.properties
  - Por cada proveedor de firma en la nube configurado en el fichero "config.properties", se puede crear un fichero "*PROVEEDOR*_config.property" en donde "*PROVEEDOR*" es el nombre asignado al proveedor en cuestión.
  - Cada uno de estos ficheros será utilizado para configurar su conector correspondiente.
  - Estos ficheros se podrán guardar junto al resto de ficheros de configuración de FIRe.

4.4.1 FICHERO CONFIG.PROPERTIES

El fichero config.properties contiene la configuración con la base de datos, ficheros temporales, el identificador de Google Analytics, el comportamiento del Cliente @firma y la firma de lotes, el sistema de estadísticas, la clase para el descifrado de propiedades y el conector de *backend* que se debe utilizar. El listado completo de claves que se puede configurar son:

- bbdd.driver (opcional)
  - Clase del driver JDBC que se desea utilizar. Si no se especifica esta propiedad, no se cargará el controlador para acceder a la base de datos. Revise la propiedad bbdd.conn, para más detalles de cómo evitar el uso de base de datos.
- bbdd.conn (opcional)
  - Cadena de conexión a la base de datos del sistema.

  Si esta propiedad no se encuentra en el fichero o su valor es none, se descartará el uso de base de datos, lo cual sólo permitiría el acceso a una única aplicación cliente. Si no se configura esta propiedad, **será obligatorio** incluir las siguientes dos propiedades de seguridad:

  - default.appId
    - Identificador de la única aplicación que puede solicitar firmas. Este identificador se comprobará cada vez que se reciba una petición.
  - default.certificate
    - Certificado codificado en base 64 con el que se autentica la aplicación.

- cipher.class (opcional)
  - Nombre de la clase encargada de descifrar propiedades de este o el resto de ficheros de configuración del componente central.
  - La clase configurada debe implementar la interfaz "es.gob.fire.server.decipher.PropertyDecipher" localizada en el módulo fire-signature-decipher.
  - Los valores o fragmentos cifrados del fichero de configuración deberán expresarse de la forma: {@ciphered: DATO_CIFRADO_BASE64 }
  - Consulte el apartado Cifrado de propiedades para más información.

- google.trackingId (opcional)

  - Identificador de estadísticas de Google Analitycs.
  - No es necesario configurar esta propiedad si no se desean recopilar estadísticas de la aplicación.
- `afirma.appId (opcional)`
  - Clave con la que se identifica el sistema frente a la Plataforma @firma para realizar la actualización de las firmas generadas.
  - No es necesario configurar esta propiedad si no se desea conectar con la Plataforma @firma para la actualización de firmas.
- `signature.alternativeXmldsig`
  - Parámetro para notificar que se incluyen bibliotecas Xerces entre las extensiones del servidor de aplicaciones.
  - En caso de detectar errores al generar firmas XAdES, podemos establecer a `true` esta propiedad para intentar corregirlos.
  - Este caso es común al realizar el despliegue en servidores JBoss.
- `batch.maxDocuments`
  - Número máximo de documentos que se pueden agregar a un lote de firma. Si se intentan agregar más documentos, la operación de agregar documento devolverá un error. Si se establece el valor 0, se considerará que no hay límite de tamaño de lote.
  - Por defecto, 10.
- `temp.dir`
  - Ruta del directorio para el almacenamiento temporal de documentos. Si no se indica, se utilizará el directorio de temporales del sistema.
  - Se recomienda que se configure esta propiedad ya que sobre este directorio se aplicara la política de borrado de ficheros caducados.
- `temp.fire.timeout`
  - Número de segundos que pueden transcurrir antes de considerar caducado un fichero temporal de FIRe. Pasado ese tiempo, la sesión se considerará caducada y el fichero podría borrarse.
  - Por defecto, 10 minutos.
- `temp.afirma.timeout`
  - Número de segundos que pueden transcurrir antes de considerar caducado un fichero temporal de intercambio del Cliente @firma. Pasado ese tiempo el fichero podría borrarse.
  - Por defecto, 10 minutos.
- `sessions.dao`
  - Gestor para la gestión conjunta de sesiones entre nodos balanceados.
  - Esto solo debe usarse cuando se despliegue el componente central en varios nodos balanceados y no se compartan los objetos en memoria entre ellos.
  - Por defecto, ninguno.
  - Valores disponibles:
    - `filesystem`

    - Gestión de sesiones en disco.
- `http.cert.attr`
    - Nombre del atributo en el que buscar los certificados SSL cliente cuando no se encuentren como atributos de la operación.
    - Esto puede ser necesario cuando se conecta un Apache y el servidor de aplicación con un proxy-pass en lugar de mediante AJP.
    - Comúnmente no será necesario configurar este parámetro.
    - Por defecto, "`X-Client-Cert`".
- `docmanager.default`
    - Gestor de documentos por defecto. Este es el gestor que se aplicará cuando no se indique ningún otro.
    - Este valor sólo debería cambiarse cuando el comportamiento de todas las aplicaciones que hagan uso del componente central obtienen y almacenan la configuración mediante un mismo gestor, distinto al por defecto, y se desea simplificarles la integración para que omitan el parámetro.
    - Por defecto, `es.gob.fire.server.services.document.DefaultFIReDocumentManager`
    - Consulte el apartado Configuración de clases gestoras de documentos para saber más de las clases gestoras de documentos.
- `pages.title`
    - Título que aparecerá en las páginas web del componente central.
    - Se permite el uso de entidades HTML para insertar caracteres que puedan producir problemas de codificación ("á", "ñ", "&"...)
    - Por defecto, `FIRma Electrónica - FIRe`
- `pages.logo`
    - URL externa del logotipo que mostrar en las páginas web del componente central.
    - Por defecto, no se proporciona un valor y se muestra el logotipo de Gobierno de España .
- `pages.public.url`
    - URL base desde la que acceder al contexto público del componente central (páginas y servicios que deben estar accesibles para los usuarios finales).
    - Esta propiedad sólo se debería configurar cuando en el servidor web se establezcan URL diferenciadas para el acceso al contexto público y el general del componente central. A la URL que se configure siempre se le agregará el sufijo "`/public`". Esto puede ser necesario cuando no se pueda o no se desee exponer los servicios privados de FIRe de cara a Internet.
    - Si se realizan dos despliegues separados del componente central (uno para atender las llamadas de los usuarios y otro para atender las llamadas del servidor) será necesario configurar la propiedad "sessions.dao" para permitir compartir las sesiones entre ellos.
    - Por defecto, no se proporciona un valor y se accederá a las páginas utilizando el contexto de despliegue del WAR seguido de "`/public`" .
- `clienteafirma.forceAutoFirma`

  - Establece si se debe forzar el uso de AutoFirma en lugar de intentar cargar previamente el MiniApplet @firma.
  - Por defecto, `false`
- `clienteafirma.forceNative`
  - Establece si, en caso de usarse AutoFirma, debe forzarse el uso de una versión nativa previamente instalada en el equipo del usuario, en lugar de cargar AutoFirma mediante el despliegue JNLP
  - Por defecto, `false`.
- `providers`
  - Listado de proveedores habilitados para su uso por parte de las aplicaciones.
  - Los valores se ponen consecutivos, separados por comas (',').
  - Al usuario se le mostraran todos los proveedores configurados en el orden que se indique en esta propiedad, salvo que la aplicación cliente defina una selección de proveedores. En ese caso, se mostrarán sólo los proveedores solicitados y en el orden indicado por la aplicación.
  - Si sólo se define un proveedor, FIRe lo seleccionará automáticamente sin necesidad de que lo haga el usuario.
  - Si el nombre de algún proveedor se antecede del carácter arroba ('@'), se considerará que es imprescindible que aparezca y se mostrará al usuario incluso si no estaba entre la selección de proveedores de la aplicación. Esta opción debe manejarse con cuidado pues puede conllevar que en una aplicación se firme con certificados distintos a los esperados.
  - El nombre de proveedor "`local`", permite el uso de certificados locales.
  - Todos los proveedores distintos de "`local`" deben declarar en este fichero su clase conectora mediante una propiedad llamada "`provider.`*`NOMBRE_CONECTOR`*".
  - Por cada proveedor distinto de "local" se debería agregar un fichero "*`NOMBRE_CONECTOR`*`_config.properties`" con la configuración de su conector.
  - Por defecto, `clavefirmatest,local`
- `provider.`*`NOMBRE_CONECTOR`*
  - Propiedad que deberá establecerse por cada proveedor definido en la propiedad "`provider`".
  - Deberá asignarse a esta propiedad el nombre completo de la clase conectora del proveedor.
  - FIRe integra, por defecto las siguientes clases conectoras de proveedores en la nube:
    - `es.gob.fire.server.connector.clavefirma.ClaveFirmaConnector`
      - Clase conectora de Cl@ve Firma
    - `es.gob.fire.server.connector.test.TestConnector`
      - Clase conectora del simulador de Cl@ve Firma
    - `es.fnmt.fire.signature.connector.TrustedXConnector`
      - Clase conectora del proveedor de la FNMT.
      - Para que este conector sea funcional es necesario el despliegue del servicio "`fnmt-fire-service.war`".

- Consulte el apartado Componentes adicionales para saber cómo agregar nuevos proveedores en FIRe.

Un ejemplo de fichero de configuración sería:

```
bbdd.driver=com.mysql.jdbc.Driver
bbdd.conn=jdbc:mysql://172.24.31.110:3306/fire_db?user=clave&password=123
cipher.class=
google.trackingId=UA-12345678-1
afirma.appId=minhap.seap.dtic.clavefirma
signature.alternativeXmldsig=false
batch.maxDocuments=30
temp.dir=C:/pruebas/temp_clavefirma
temp.fire.timeout=600
temp.afirma.timeout=600
docmanager.default=es.gob.fire.server.services.document.DefaultFIReDocumentManager
pages.title=FIRe
clienteafirma.forceAutoFirma=false
clienteafirma.forceNative=true
providers=clavefirma,local
provider.clavefirma=es.gob.test.server.connector.clavefirma.ClaveFirmaConnector
```

En este fichero se indica que la cadena de conexión JDBC con la base de datos de administración es "jdbc:mysql://172.24.31.110:3306/fire_db?user=clave&password=1234" (consulte con la documentación de su controlador JDBC para determinar el formato apropiado para su cadena de conexión), no se indica clase de descifrado de propiedades (ya que no se indican cifradas) y el identificador de estadísticas de Google Analytics es "UA-12345678-1". El identificador de la aplicación frente a la Plataforma @firma será "minhap.seap.dtic.clavefirma" (para la propia conexión con la plataforma será necesario configurar el fichero `platform.properties`). Se ha definido que el número máximo de documentos que se puede adjuntar a un lote será 30 y cuál es el directorio para el guardado de temporales. Se han dejado diversas propiedades a sus valores por defecto (tiempos de espera, configuración de firma XML, comportamiento de AutoFirma, etc). Se han definido dos proveedores: "clavefirma", que tiene la clase conectora "es.gob.test.server.connector.clavefirma.ClaveFirmaConnector" y permite usar los certificados de Cl@ve Firma, y "local", que permite utilizar los certificados disponibles en el equipo del usuario.

4.4.2 FICHERO PLATFORM.PROPERTIES

Este fichero configura la conexión con la Plataforma @firma para permitir la actualización de las firmas generadas a formatos longevos. Si no se fuesen a realizar mejoras de firma a formatos longevos, no será necesario configurar este fichero ni acceder a la Plataforma @firma.
Para poder hacer uso de una instancia de la Plataforma @firma, el administrador de la misma deberá dar de alta su aplicación y habilitarle las credenciales de acceso correspondientes. Consulte con el administrador de la Plataforma @firma a la que desea acceder para más detalles.

Este fichero de propiedades sigue el formato definido en las bibliotecas Integr@, y necesita tener definidas al menos las siguientes claves:

- `webservices.timeout`
  - Tiempo de espera a las conexiones, en milisegundos
- `webservices.authorization.method`
  - Tipo de autenticación contra la Plataforma @firma. Debe tener siempre el valor BinarySecurityToken.
- `webservices.endpoint`
  - URL del servicio general de la Plataforma @firma, debe tener la barra "/" al final.
  - La URL del servicio la proporciona el administrador de la instancia de la Plataforma @firma.
- `webservices.service.signupgrade`
  - Nombre del servicio Web de mejora de firmas.
  - El nombre del servicio lo proporciona el administrador de la instancia de la Plataforma @firma.
- `webservices.service.certverify`
  - Nombre del servicio Web de verificación de certificados.
  - El nombre del servicio lo proporciona el administrador de la instancia de la Plataforma @firma.
- `com.trustedstore.path`
  - Ruta del almacén con los certificados servidor de confianza para las conexiones SSL seguras.
  - Debe estar configurado de tal forma que los servidores de la instancia de la Plataforma @firma a la que se conecta sea den de confianza.
- `com.trustedstore.password`
  - Contraseña del almacén con los certificados servidor de confianza para las conexiones SSL seguras.
- `com.trustedstore.type`
  - Tipo del almacén con los certificados servidor de confianza para las conexiones SSL seguras. Puede tener los valores:
    - JKS
      - Almacén de claves de Java (Recomendado).
    - PKCS12
      - Almacén de claves PKCS#12.
- `webservices.authorization.ks.path`

  - Almacén de claves que contiene los certificados y claves privadas para las conexiones seguras SSL con certificado cliente de autenticación contra la Plataforma @firma.
  - El administrador de la instancia de la Plataforma @firma en cuestión deberá habilitar el acceso por medio de la clave pública del certificado que se desee utilizar.
- `webservices.authorization.ks.type`
  - Tipo del almacén de claves que contiene los certificados y claves privadas para las conexiones seguras SSL con certificado cliente de autenticación contra la Plataforma @firma. Puede tener estos valores:
    - JKS
      - Almacén de claves de Java (Recomendado)
    - PKCS12
      - Almacén de claves PKCS#12.
- `webservices.authorization.ks.password`
  - Contraseña del almacén de claves que contiene los certificados y claves privadas para las conexiones seguras SSL con certificado cliente de autenticación contra la Plataforma @firma.
- `webservices.authorization.ks.cert.alias`
  - Alias del certificado a usar para las conexiones seguras SSL con certificado cliente de autenticación contra la Plataforma @firma dentro del almacén indicado. Se recomienda que se evite el uso de alias con caracteres no ASCII.
- `webservices.authorization.ks.cert.password`
  - Contraseña del certificado a usar para las conexiones seguras SSL con certificado cliente de autenticación contra la Plataforma @firma dentro del almacén indicado.

Un ejemplo de fichero de propiedades para acceso a la Plataforma @firma para mejora de firmas podría ser el siguiente, a expensas de rellenar los datos proporcionados por el administrador de su proveedor de servicios de @firma.

```
# Timeout de conexion (-1 sin timeout)
webservices.timeout=50000

# Metodo de autenticacion
webservices.authorization.method = BinarySecurityToken

# URL del servicio Afirma. Debe tener la barra ("/") final
webservices.endpoint=

webservices.service.signupgrade=
webservices.service.certverify=

# Almacen de confianza para conexiones seguras
com.trustedstore.path      =
com.trustedstore.password =
com.trustedstore.type      = JKS

# Propiedades para el método BinarySecurityToken
webservices.authorization.ks.path =
```

```
webservices.authorization.ks.type = PKCS12
webservices.authorization.ks.password =
webservices.authorization.ks.cert.alias =
webservices.authorization.ks.cert.password =
```

4.4.3 FICHERO PROVIDER_CLAVEFIRMA.PROPERTIES

Es el fichero de configuración del conector de Cl@ve Firma. Este fichero sólo será necesario cuando se configure la clase conectora "`es.gob.test.server.connector.clavefirma.ClaveFirmaConnector`". En caso de que el administrador cambiase el nombre de proveedor que configura esta clase conectora, será necesario cambiar el nombre de este fichero a como corresponda.

Las propiedades de este fichero deberían ser proporcionadas por la GISS.

Las propiedades a configurar en este fichero son:

- `providerName`
  - Configura el nombre de proveedor con el que se identifica el servicio contra la GISS.
  - Este parámetro es obligatorio, debe ir en MAYÚSCULAS y se forma de la concatenación del NIF del organismo y del código DIR3 unidos por un guion bajo (NIF_DIR3), tal como se ha especificado en al ACL (formulario) de alta en el sistema de la GISS.
- `allowRequestNewCert`
  - Configura si el proveedor debe permitir a los usuarios el generar certificados cuando no tengan ninguno expedido (`true`) o que no se permita generarlos (`false`).
  - Por defecto, `true`.
- URL_GATEWAY
  - URL hacia la pasarela de firma centralizada.
- AUTH_STORE
  - Ruta hacia el almacén PKCS#12 con el certificado cliente autorizado para acceder al servidor remoto de firma centralizada.
- AUTH_STORE_PASS
  - Contraseña del almacén PKCS#12 con el certificado cliente autorizado para acceder al servidor remoto de firma centralizada indicado anteriormente.

Un ejemplo de este fichero podría ser:

```
providerName=S1234567E_E12345678
allowRequestNewCert=true
```

```
URL_GATEWAY=https://xxxx/ClaveFirmaService
AUTH_STORE=//users/apache/clavefirma/auth.p12
AUTH_STORE_PASS=12345678
```

4.4.4 FICHERO PROVIDER_CLAVEFIRMATEST.PROPERTIES

Es el fichero de configuración del conector del simulador de pruebas de Cl@ve Firma. Este fichero sólo será necesario cuando se configure la clase conectora "`es.gob.test.server.connector.test.Test-Connector`". En caso de que el administrador cambiase el nombre de proveedor que configura esta clase conectora, será necesario cambiar el nombre de este fichero a como corresponda.

Las propiedades a configurar en este fichero son:

- `endpoint`
  - Permite configurar la ruta base de los servicios de prueba.
  - Si no se establece esta propiedad o se establece vacía, se interpretará que la URL base de los servicios de prueba es: https://127.0.0.1:8443/clavefirma-test-services/
- `allowRequestNewCert`
  - Configura si el proveedor debe permitir a los usuarios el generar certificados cuando no tengan ninguno expedido (`true`) o que no se permita generarlos (`false`).
  - Esta opción permite emular a la propiedad homónima del conector de Cl@ve Firma.
  - Por defecto, `true`.
- `ssl.keystore`
  - Ruta absoluta del almacén de claves de autenticación en caso de que el servicio de pruebas se encuentre en un servidor con SSL con autenticación cliente.
- `ssl.keystorePass`
  - Contraseña del almacén de claves de autenticación en caso de que el servicio de pruebas se encuentre en un servidor con SSL con autenticación cliente.
- `ssl.keystoreType`
  - Tipo de almacén de claves de autenticación en caso de que el servicio de pruebas se encuentre en un servidor con SSL con autenticación cliente. Los valores posibles son "PKCS12" y "JKS" (valor por defecto y recomendado debido a su mejor soporte por los proveedores de Java).
- `ssl.truststore`

- Ruta absoluta del almacén de certificados de confianza en caso de querer configurar un almacén particular en lugar del por defecto de Java. Si no se desean realizar autenticaciones sobre el certificado SSL del servidor, se puede configurar el valor "all".

- `ssl.truststorePass (Opcional)`
  - Contraseña del almacén de certificado de confianza en caso de querer configurar un almacén particular en lugar del por defecto de Java.
  - No es necesario configurarla si en la propiedad "`ssl.truststore`" se ha configurado el valor "all".

- `ssl.truststoreType (Opcional)`
  - Tipo de almacén de certificados de confianza en caso de querer configurar un almacén particular en lugar del por defecto de Java. Los valores posibles son "PKCS12" y "JKS" (valor por defecto y recomendado debido a su mejor soporte por los proveedores de Java).
  - No es necesario configurarla si en la propiedad "`ssl.truststore`" se ha configurado el valor "all".

Un ejemplo de este fichero podría ser:

```
endpoint=https://clavefirmagiss:8443/clavefirma-test-services

ssl.keystore=C:/Users/ClaveFirma/certificados/client_ssl.jks
ssl.keystorePass=12341234
ssl.keystoreType=JKS

ssl.truststore=all
#ssl.truststorePass=
#ssl.truststoreType=
```

4.4.5 FICHERO PROVIDER_FNMT.PROPERTIES

Este es el fichero de configuración del conector con el servicio de firma en la nube de la FNMT y sólo será necesario cuando se configure la clase conectora "`es.fnmt.fire.signature.connector.TrustedXConnector`". En caso de que el administrador cambiase el nombre de proveedor que configura esta clase conectora, será necesario cambiar el nombre de este fichero a como corresponda.

Para el funcionamiento del conector de la FNMT, es necesario desplegar el servicio "`fnmt-fire-service.war`". Consulte el apartado Servicio auxiliar del conector de la FNMT para más información.

El conector de la FNMT está inicialmente preparado para la firma con certificados de empleado público y las opciones de configuración incluidas sólo admiten actualmente valores orientados a tal fin.

Las propiedades de este fichero deberían ser proporcionadas por la FNMT en base al acuerdo alcanzado con ella. En este apartado se muestran los valores comunes para la configuración del uso de certificados de empleado público.

Las propiedades a configurar en este fichero son:

- `dtsCacher`
    - Indica la clase Java que se encargará del almacén temporal de firmas en el sistema.
    - Por defecto, debe utilizarse el valor "`es.fnmt.fire.signature.connector.DataToSignCacherFileSystem`".
- `trustedXUrl`
    - URL del servicio TrustedX eIDAS.
- `trustedXUser`
    - Nombre de usuario del servicio TrustedX eIDAS.
- `trustedXPassword`
    - Contraseña del usuario del servicio TrustedX eIDAS.
- `trustedXClientId`
    - Identificador de la aplicación cliente de TrustedX eIDAS.
- `trustedXApiKey`
    - Código del API de TrustedX eIDAS para la aplicación cliente.
- `authServiceId`
    - Identificador del servicio de autenticación.
    - Por defecto, debe utilizarse el valor "`as-employee`".
- `defaultDomain`
    - Dominio por defecto del servicio.
    - Por defecto, debe utilizarse el valor "`employee`".
- `definedLabels`
    - Etiquetas que debe tener una identidad de firma para ser utilizada en el sistema (separadas por comas).
    - Por defecto, debe utilizarse el valor "`grupo2,x509:keyUsage:contentCommitment,cloudid,fnmt`".
- `codeService`

- URL del servicio de gestión de redirecciones *Oauth*. Este es un servicio auxiliar que acompaña a FIRe y que deberá desplegarse única y exclusivamente cuando se configure este conector.
- Por ejemplo, "`http://demo.com:8080/fnmt-fire/OauthHelper`".

- `scopeAttrsManageAndUserList`
  - *Scopes* de gestión y listado de atributos y usuarios (separados por un espacio).
  - Por defecto, debe utilizarse el valor "`urn:safelayer:eidas:account:user:attributes:manage urn:safelayer:eidas:account:user:list`".

- `scopeProfile`
  - *Scope* de perfil.
  - Por defecto, debe utilizarse el valor "`urn:safelayer:eidas:sign:identity:profile`".

- `scopeProfileAndServer`
  - *Scopes* de perfil y servidor (separados por un espacio).
  - Por defecto, debe utilizarse el valor "`urn:safelayer:eidas:sign:identity:profile urn:safelayer:eidas:sign:identity:use:server`".

4.5 CONFIGURACIÓN – BASE DE DATOS

La base de datos del sistema de firma centralizada se utiliza para la configuración del sistema y la autenticación de las peticiones de firma. Queda a elección del integrador crear una nueva base de datos para el sistema o utilizar una ya existente.

El Sistema de Gestión de Base de Datos queda a elección del integrador, pero se debe tener encuentra que el componente central deberá poder acceder a la misma a través de un controlador JDBC. Tanto el driver JDBC a utilizar como la cadena de conexión deben quedar reflejadas en el fichero de configuración del componente central.

4.5.1 CREACIÓN – BASE DE DATOS

Para la creación de las tablas de la base de datos dispondremos de scripts, para los SGBD MySQL y ORACLE. No obstante, la creación de las tablas necesarias deberá ser realizada o supervisada por el Administrador del SGBD en cuestión.

Se distribuye junto a los servicios de firma centralizada las sentencias DDL (fichero "`01_clave-firma_ddl.sql`") para la creación de las tablas de base de datos.

A continuación, se indican definiciones de las distintas tablas en el SGBD:

Tabla de aplicaciones
Nombre de la tabla: tb_aplicaciones Campos: Nombre=id, Tipo=cadena 48 caracteres, NO NULO, Clave Primaria. Nombre=nombre, Tipo=cadena 45 caracteres, NO NULO. Nombre=responsable, Tipo=cadena 45 caracteres, NO NULO. Nombre=resp_correo, Tipo=cadena 45 caracteres, NULO. Nombre=resp_telefono, Tipo=cadena 30 caracteres, NULO. Nombre=fecha_alta, Tipo=fecha y hora, NO NULO. Nombre= fk_certificado, Tipo=numérico entero 11 cifras, NO NULO, Clave Ajena.

Tabla de certificados
Nombre de la tabla: tb_certificados Campos: Nombre= id_certificado, Tipo=numérico entero 11 cifras auto-incremental, NO NULO, Clave Primaria. Nombre= nombre_cert, Tipo=cadena 45 caracteres, NO NULO. Nombre= fec_alta, Tipo=fecha y hora, Valor por defecto= fecha Actual, NO NULO. Nombre= cert_principal, Tipo=cadena 5000 caracteres, NULO. Nombre= cert_backup, Tipo=cadena 5000 caracteres, NULO. Nombre= huella_principal, Tipo=cadena 45 caracteres, NULO. Nombre= huella_backup, Tipo=cadena 45 caracteres, NULO.

Tabla de usuarios
Nombre de la tabla: tb_usuarios Campos: Nombre= id_usuario, Tipo=numérico entero 11 cifras auto-incremental, NO NULO, Clave Primaria. Nombre= nombre_usuario Tipo=cadena 30, NO NULO. Clave única Nombre= clave, Tipo=cadena 45 caracteres, NO NULO. Nombre= nombre, Tipo=cadena 45 caracteres, NO NULO. Nombre= apellidos, Tipo= cadena 120 caracteres, NO NULO. Nombre= correo_elec, Tipo= cadena 45 caracteres, Valor por defecto= NULO Nombre= fec_alta, Tipo= fecha y hora, Valor por defecto= fecha Actual, NO NULO. Nombre= telf_contacto, Tipo= cadena 45 caracteres, Valor por defecto= NULO Nombre= rol, Tipo= cadena 45 caracteres, NO NULO, Valor por defecto= 'admin' Nombre= usu_defecto, Tipo=numérico entero 4 cifras, NO NULO, Valor por defecto= 0

Restricciones tabla aplicaciones
Campo =fk_certificado, como clave foránea referencia con tabla tb_certificados campo id_certificado. Restricción al borrar, se actualiza en cascada.

Estas tablas estarán inicialmente vacías a excepción de una contraseña inicial para el acceso al componente de administración, que se realizará mediante una sentencia de inserción en la tabla `tb_usuarios` anteriormente creada:

Definición de la contraseña de administrador
Tabla=tb_usuarios registro: nombre_usuario = 'admin' clave='D/4avRoIIVNTwjPW4AlhPpXuxCU4Mqdhryj/N6xaFQw=', codificado como SHA-256 clave=1111. nombre='default name', apellidos='default surnames' usu_defecto=1

La población de la tabla de administración de aplicaciones se realizará a través del componente de administración. Para poder utilizar la herramienta de administración será necesario utilizar la contraseña configurada para el administrador en la tabla de opciones de configuración. Si desea cambiar la contraseña de administrador, siga los pasos indicados en el apartado "**Error! Reference source not found.**".

4.6 CONFIGURACIÓN – REGISTRO (LOG)

El componente central utiliza el sistema de registro de Java (*Java Logging API:* https://docs.oracle.com/javase/6/docs/technotes/guides/logging/), utilizando dos nombres de registro a lo largo del aplicativo:

- `es.gob.fire`
  - Registro para los componentes específicos de FIRe.
- `es.gob.afirma`
  - Registro para los componentes software compartidos con el Cliente @firma (módulos de firma trifásica y módulos de utilidad relacionados con las firmas electrónicas).

El sistema de registro de Java puede configurarse para que este aparezca en consola, se escriba en un fichero, se envíe por red, etc., así como para definir qué mensajes deben o no aparecer (errores, advertencias, información, etc.).

Consulte la documentación de su entorno de ejecución de Java (JRE) y de su servidor de aplicaciones JEE para configurar apropiadamente el registro:

- https://docs.oracle.com/javase/7/docs/technotes/guides/logging/
- https://docs.oracle.com/javase/7/docs/api/java/util/logging/LogManager.html
- https://docs.oracle.com/javase/7/docs/technotes/guides/logging/overview.html

El sistema no implementa ningún otro sistema de registro (Apache Log4J, etc.).

4.7 CONFIGURACIÓN DE CLASES GESTORAS DE DOCUMENTOS

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

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

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

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

Si un integrador desease conectarse con FIRe y utilizar una clase gestora de documentos, debería, primeramente, solicitar el permiso del administrador de FIRe, ya que este es el que debe habilitar el uso de esa clase gestora.

Antes de dar su visto bueno, el administrador deberá valorar:

- El beneficio que obtendrá la aplicación o aplicaciones cliente frente a la carga adicional que recibirá el componente central por llevar a cabo esta tarea.

- La confianza que se tiene en el integrador y el desarrollo que haga de la clase gestora. Hay que tener en cuenta que este desarrollo se ejecutará en el servidor del componente central, con los mismos permisos que tenga el usuario con el que se ejecute el servidor de aplicaciones, y tendrá acceso a sus recursos.
- Que el desarrollo realizado no incluya dependencias incompatibles con las de FIRe, el servidor de aplicaciones o alguna otra clase gestora, ya que estas deberán desplegarse también como parte del componente central.
- La posibilidad y conveniencia de dar acceso a los recursos de red que necesite la clase gestora para la carga de los datos y el tratamiento y guardado de la firma.

En caso de dar el visto bueno al uso de la clase gestora, el integrador deberá proporcionar al administrador esta clase (empaquetada en forma de JAR), el nombre completo de la clase, las bibliotecas de las que dependa, el listado exacto de recursos y entornos a los que se deberá tener acceso (para habilitar los permisos del sistema y permisos de red necesarios) y, si procede, el fichero de configuración de la clase y las instrucciones para configurarlo.

Para dar de alta la clase gestora, el administrador deberá:

1. Introducir el JAR de la clase gestora y sus dependencias dentro del directorio de bibliotecas del WAR del componente central (`fire-signature.war\WEB-INF\lib`).
2. Agregar al fichero de configuración `config.properties` una entrada que relacione el nuevo gestor de documentos con la clase que lo implementa. La forma de esta entrada deberá ser:
   - `docmanager.NOMBRE=NOMBRE_CLASE`.

   Donde:
   - `NOMBRE`: Es el nombre que queramos darle a la clase gestora para identificarla claramente y diferenciarla de cualquier otra que demos de alta.
   - `NOMBRE_CLASE`: Nombre completo de la clase gestora de documentos (incluyendo el paquete) proporcionada por el integrador.

   Por ejemplo:
   - `docmanager.bbddMinisterio=es.minhap.ddbb.MinisterioDocManager`
3. Si se ha proporcionado un fichero de configuración, se deberá copiar al directorio en el que se encuentra el fichero de configuración del componente central (`config.properties`) y renombrarlo según el patrón:
   - `docmanager.NOMBRE.properties`

   Donde:
   - `NOMBRE`: Es el nombre que le hemos dado a la clase gestora en el paso anterior para identificarla claramente y diferenciarla de cualquier otra que demos de alta.

   Por ejemplo:
   - `docmanager.bbddMinisterio.properties`

4. Si el integrador ha proporcionado instrucciones para la configuración del fichero de configuración, seguir estás instrucciones.
5. Habilitar el acceso a los recursos y entornos de red necesarios para el funcionamiento de la clase lectora indicados por el integrador.
6. Proporcionar al integrador el nombre que hemos designado para referenciar a la clase gestora de documentos.

   Por ejemplo:

   - `bbddMinisterio`

4.8 CIFRADO DE PROPIEDADES

El componente central permite que los valores de las propiedades configuradas en los distintos ficheros de configuración se indiquen cifrados y codificados en base64. El mecanismo de cifrado es ajeno a FIRe. Es el administrador del sistema el que debe cifrar los datos y proporcionar a FIRe una forma de descifrarlos en el momento en el que vaya a necesitarlos.

Para poder utilizar valores cifrados en los distintos ficheros de configuración del componente central se deberá:

1. Establecer los valores o fragmentos cifrados y codificados en base 64 en las propiedades que se deseen de los distintos ficheros de configuración. Esto se hará mediante la cadena:

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

   Por ejemplo:

   ```
   bbdd.conn=jdbc:mysql://127.0.0.1:3306/fire_db?user=miusuario&pass-
   word={@ciphered: aDbb+4nmBHk7ifT= }
   ```

2. Implementar la interfaz Java "`es.gob.fire.server.decipher.PropertyDecipher`" y su método "`decipher`". Este método recibe el binario resultado de decodificar el Base64 configurado en la propiedad y se le debe implementar la lógica para el descifrado de ese dato. El mecanismo de descifrado puede ser cualquiera y utilizar cualquier número de claves, certificados o recursos externos.

3. Configurar la propiedad "`cipher.class`" del fichero "`config.properties`" con el nombre de la clase que implementa "`PropertyDecipher`".

   Por ejemplo:

   ```
   cipher.class = es.gob.miapp.fire.MiDecipher
   ```

4. Agregar al CLASSPATH del servidor de aplicaciones la clase de descifrado. Por ejemplo, se podría empaquetar esta clase en un JAR y agregar este al WAR "`fire-signature.war`" antes de desplegarlo.

5 SIMULADOR DE CL@VE FIRMA

Junto con el componente central se distribuye un servicio de prueba que simula la comunicación con Cl@ve Firma, permitiendo así probar el comportamiento de nuestras aplicaciones sin necesidad de tener conexión con ningún proveedor real de firma en la nube.

5.1 DESPLIEGUE

Para poder conectar el componente central con el servicio simulador de Cl@ve Firma hay que realizar los siguientes pasos:

1. Desplegar el servicio de pruebas:
   - `clavefirma-test-services.war`
2. Agregar al CLASSPATH o al directorio determinado por la variable de entorno "`fire.config.path`" el fichero de configuración del simulador:
   - `test-backend.properties`
3. Configurar las distintas propiedades del fichero configuración:
   - `tmp_dir`: Directorio en el que se almacenarán los ficheros temporales necesarios durante el proceso de firma. El servicio debe tener permisos para crear, modificar y eliminar ficheros en este directorio. Si no se establece un valor, se usará el directorio de temporales del sistema.
   - `url_service`: URL base del servicio de pruebas. Esta URL debe ser accesible desde el equipo del usuario. Es decir, debe indicarse el nombre de dominio o IP necesario para que el equipo del usuario pueda acceder directamente a las páginas y servicios del servicio de pruebas. En caso de no indicarse, se interpretará que el despliegue de cliente y servidor se realizan en el mismo equipo y se buscaré en la dirección:
     - `http://localhost:8080/clavefirma-test-services`
4. Modificar el fichero "`config.properties`" del componente central para permitir la conexión con el servicio de pruebas:
   - Modificar el valor de la propiedad `providers` para que utilice el conector con el servicio de prueba:
     - `providers=clavefirmatest,local`
5. Configurar el fichero "`provider_clavefirmatest.properties`" del componente central para permitir la conexión con el servicio de pruebas:
   - Modificar el valor de la propiedad `test.endpoint` para establecer la URL base de despliegue del servicio. Por ejemplo:
     - `test.endpoint=https://XX.XX.XX.XX:8443/clavefirma-test-services`

- Establecer a través de las distintas propiedades que empiezan por "`test.ssl.`" la configuración para la conexión con el servicio de pruebas.
  - `test.ssl.keyStore` (Opcional)
    - Ruta del almacén de claves para la autenticación mediante certificado con el servicio simulador.
  - `test.ssl.keyStorePassword` (Opcional)
    - Contraseña del almacén de claves de autenticación SSL.
  - `test.ssl.keyStoreType` (Opcional)
    - Tipo del almacén de claves del certificado de autenticación SSL: "`JKS`" (almacén de Java) o "`PKCS12`" (almacén PKCS12/PFX). Por defecto, "`JKS`".
  - `test.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`".
  - `test.ssl.trustStorePassword` (Opcional)
    - Contraseña del almacén de confianza.
  - `test.ssl.trustStoreType` (Opcional)
    - Tipo del almacén de confianza: "`JKS`" (almacén de Java) o "`PKCS12`" (almacén PKCS12/PFX). Por defecto, "`JKS`".

5.2 USUARIOS DE PRUEBA

Los servicios de prueba incluyen por defecto varios usuarios de prueba:

- Usuario con certificado para la prueba de las operaciones de firma:
  - Usuario: **00001**
  - Contraseña: **1111**
- Usuario sin certificado para la prueba de las operaciones de generación de certificado. Durante la prueba de este servicio se dará al usuario la posibilidad de modificar la contraseña. Dado que este servicio tiene como único objetivo ayudar a la integración del servicio, los datos introducidos por el usuario se ignorarán y la contraseña siempre será la aquí indicada:
  - Usuario: **00002**
  - Contraseña: **1111**

- Usuario con certificado bloqueado:
    - Usuario: **00003**
- Usuario con registro débil/no fehaciente:
    - Usuario: **00004**

5.3 AGREGAR NUEVOS USUARIOS DE PRUEBA

Si se desea añadir nuevos usuarios de prueba al servicio, hay que seguir los siguientes pasos:

- Agregar un fichero de propiedades en un directorio "`testservice`" dentro del CLASSPATH del servidor de aplicaciones. El fichero debe tener la extensión ".properties" y el nombre debe ser el identificador que se desee para el usuario. Por ejemplo, crear el fichero "mi_usuario.properties" daría de alta al usuario "mi_usuario".

  El fichero debe contener las siguientes propiedades:

    - `state`: Estado del usuario. En esta propiedad se pueden usar los siguientes valores:
        - OK: Es un usuario correcto con un certificado válido. Este es el valor por defecto y requiere que se proporcione un almacén y se establezca su contraseña en la propiedad "password".
        - NOCERT: Es un usuario correcto pero sin certificados. En este caso no es necesario indicar almacén de certificados, pero se permitirá al usuario "generar" uno nuevo que siempre será el localizado en el directorio "testservice/new/new.p12". Se deberá establecer también la propiedad "password" proporcionándole la contraseña de ese almacén.
        - BLOCKED: Es un usuario con su certificado bloqueado. En este caso no es necesario proporcionar un almacén de certificados ni indicar una contraseña.
    - `password`: Contraseña del almacén.

  Un ejemplo de fichero sería:

```
state=OK
password=1111
```

- Agregar, si procede, el certificado del nuevo usuario en el mismo directorio y con el mismo nombre pero extensión ".p12".

Puede hacer uso del servicio de prueba desde su propia aplicación o desde la aplicación de ejemplo que se distribuye con FIRe. Recuerde que esta aplicación de ejemplo no está pensada para ser subida a producción. Para obtener más información consultar el capítulo 9 del manual del integrador ("MAN - Manual de integrador.docx").

6 COMPONENTE DISTRIBUIDO

El componente distribuido consiste en una biblioteca que permite acceder a las funcionalidades proporcionados por el componente central del sistema.

Se distribuyen tres implementaciones diferentes del componente distribuido para facilitar su integración en las aplicaciones cliente: Java, .NET y PHP.

6.1 JAVA

Se requiere un entorno de ejecución de Java en versión 6 o superior, en 32 o 64 bits.

La versión de Java del componente distribuido es un JAR de Java firmado digitalmente, que se acompaña de 2 bibliotecas externas con las que mantiene dependencias.

Las bibliotecas son:

- Componente distribuido Java:
  - `fire-client-2.1.jar`
- Dependencias:
  - `javax.json-api-1.0.jar`
  - `javax.json-1.0.4.jar`
  -

El primer paso para el uso del componente distribuido es agregar estas bibliotecas al CLASSPATH de su aplicación o del JRE que queramos usar para la integración. Para esto último, consulte la documentación de su JRE. Típicamente, lo normal es añadir directamente estas bibliotecas dentro de la variable de entorno CLASSPATH (https://docs.oracle.com/javase/tutorial/essential/environment/paths.html) o hacer uso de algún gestor de dependencias como Apache Maven.

Adicionalmente, si no se le indica al objeto cliente la configuración para la conexión con el componente central, esta se cargará del fichero de configuración:

- `client_config.properties`

Este debe ser un fichero de propiedades y contener aquellas necesarias para la conexión con el componente central.

Las propiedades que se permiten configurar son:

- `fireUrl`
  - URL del servicio del componente central.

- `javax.net.ssl.keyStore` (Opcional)
  - Ruta del almacén de claves para la autenticación mediante certificado con el componente central. Este certificado debe estar dado de alta en la base de datos del componente central, asignado al identificador de la aplicación cliente en la que se esté integrando el componente distribuido.
  - Si se omite este parámetro se usará la configuración establecida a nivel global en la JRE.
- `javax.net.ssl.keyStorePassword` (Opcional)
  - Contraseña del almacén de claves de autenticación SSL.
- `javax.net.ssl.keyStoreType` (Opcional)
  - Tipo del almacén de claves del certificado de autenticación SSL: "JKS" (almacén de Java) o "PKCS12" (almacén PKCS12/PFX).
  - Por defecto, se considera que el almacén es de tipo JKS.
- `javax.net.ssl.trustStore` (Opcional)
  - Ruta del almacén de certificados de confianza SSL. Esto se usa cuando el certificado con el que se ha montado el SSL del componente central no está en el almacén de confianza de Java y se desea establecer un almacén de confianza alternativo.
  - En caso de querer desactivar la comprobación del certificado SSL del servidor, se puede configurar el valor "all".
  - Si se omite este parámetro se usará la configuración establecida a nivel global en la JRE. Por defecto, se confiará en los certificados dados de alta en el almacén "cacerts".
- `javax.net.ssl.trustStorePassword` (Opcional)
  - Contraseña del almacén de confianza.
- `javax.net.ssl.trustStoreType` (Opcional)
  - Tipo del almacén de confianza: "JKS" (almacén de Java) o "PKCS12" (almacén PKCS12/PFX).
  - Por defecto, se considera que el almacén es de tipo JKS.

Ejemplo de objeto de configuración o contenido de `client_config.properties`:

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

En caso de utilizarse este fichero de configuración, debe estar en el directorio que se haya proporcionado a la aplicación mediante la propiedad "`fire.config.path`" de Java o, si no se ha configurado, en el CLASSPATH del servidor de aplicaciones.

Consulte la documentación de su servidor de aplicaciones o contenedor de aplicaciones Web Java para conocer como proporcionarle propiedades o como introducir ficheros de propiedades dentro del CLASSPATH de las aplicaciones.

Consulte el manual de integración de FIRe para más información sobre cómo hacer uso de FIRe desde sus aplicaciones.

6.1.1 Trazas de registro del Componente Distribuido Java

El componente distribuido Java utiliza SLF4J como biblioteca para la generación de trazas de registro. Esta biblioteca permite que el integrador enlace los logs del componente distribuido de FIRe con los del resto de su aplicación por medio de la "biblioteca puente" correspondiente a su sistema de logs (Log4J, Log4J 2, Apache Logging API,…). Para más información, consulte el manual del integrador de FIRe.

Todos los logs del componente distribuido Java de FIRe se imprimen con el nombre de la clase que los genera. Así, pueden recogerse todos utilizando el nombre "`es.gob.fire.client`".

6.2 .NET

Se requiere un entorno de ejecución de Microsoft .NET en versión 4 o superior.

La versión .NET del componente distribuido se encuentra en formato de biblioteca de enlace dinámico (DLL) de .NET (construida con C#). Esta biblioteca se llama `fire.dll`.

La instalación de la biblioteca necesita que esta se encuentre en uno de los directorios de carga de bibliotecas de la aplicación que la utilice, que típicamente son:

1. El mismo directorio que la aplicación (EXE, DLL, etc.) que use el componente distribuido.
2. Un directorio dentro del PATH del sistema.
3. El directorio de bibliotecas del sistema, normalmente `%SystemRoot%/System32`

Para usarla en tiempo de desarrollo, consulte la documentación de su entorno de desarrollo.

En Microsoft Visual Studio necesitará importar la referencia:

- https://msdn.microsoft.com/en-us/library/7314433t%28v=vs.90%29.aspx

Los servicios del componente central a los que accederá esta biblioteca y la configuración de acceso deben establecerse en el registro de Windows dentro de la clave "`HKEY_CURRENT_USER\Software\FIRe`" con los nombres de valor:

- `fire_service`: URL del servicio de FIRe.
- `ssl_client_pkcs12`: Ruta absoluta del almacén PKCS#12/PFX con las clave y certificado para la autenticación cliente SSL contra el componente central.
- `ssl_client_pass`: Contraseña del almacén PKCS#12/PFX para la autenticación cliente SSL contra el componente central.
- `admit_all_certs`: Opcional. Indica si debe obviarse la autenticación del certificado SSL ("true") o si debe comprobarse la validez y confianza del certificado como el algoritmo, la fecha de caducidad, que se encuentre en el almacén de confianza de Windows, etc. ("false"). Por defecto, realizarán las comprobaciones de seguridad.

Estas claves deben ser de tipo cadena de texto.

Para agregarse estas claves al registro de Windows puede utilizarse un fichero REG como el que se indica a continuación de ejemplo:

```
[HKEY_CURRENT_USER\Software\FIRe]

"fire_service"="https://servidor.com/fire-signature/fireService"

"ssl_client_pkcs12"="C:/users/usuario/Documents/client_ssl.pfx"

"ssl_client_pass"="12345678"

"admit_all_certs"="true"
```

En este ejemplo, se configura la URL de acceso del servicio localizado en el dominio de ejemplo "servidor.com". Además, las peticiones al servicio se autenticarán con el certificado cliente del almacén indicado y no se comprobará la validez y confianza del certificado SSL del servidor.

Dado que la configuración del registro se almacena en la rama de usuario del registro, la importación debe hacerse con el mismo usuario que se utilice para ejecutar la aplicación que haga uso de la DLL del componente distribuido.

6.3 PHP

Se requiere un entorno de ejecución de PHP, en versión 5 o superior, y la extensión "php_curl" habilitada. Si se va a utilizar la función auxiliar `parse_certificate()`, también será necesario activar la extensión "php_openssl".

El componente distribuido PHP se presenta como un único fichero PHP llamado "`fire.php`". Su integración requiere únicamente su referencia desde el fichero PHP que haga uso de él. En este fichero se incluyen las variables de configuración con las URL de los servicios del componente central y la propia lógica comunicación.

Las variables para la configuración de las URL de los servicios del componente central son:

- `FIREURL`: URL del servicio del componente central.

La definición de variables se realizará mediante sentencias `define`.

Un ejemplo de configuración de estas variables es:

```
<?php

   // Definimos las URL con las que invocar cada servicio

   define ("FIREURL","https://servidor.com/fire-signature/fireService ");

      …
```

En este ejemplo, se configura las URL de acceso del servicio localizado en el dominio de ejemplo "servidor.com".

También será necesario configurar la conexión SSL cliente contra el componente centralizado a través de cURL. Para esto, estableceremos en el array de propiedades "`$client_ssl_curl_options`" que se encuentra al principio de "`fire.php`" con las propiedades necesarias.

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

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

Se puede consular la información completa sobre estos y otros parámetros de configuración de cURL en la web: https://curl.haxx.se/libcurl/c/easy_setopt_options.html

Puede importar el componente de distribuido PHP desde su aplicación cliente PHP, mediante el comando `include`.

Por ejemplo:

```
<html>
 <head>
  <title>Ejemplo de importacion</title>
 </head>
 <body>
 <?php
      include 'fire.php';
 …
```

7 COMPONENTE DE ADMINISTRACIÓN

La aplicación de administración es independiente al componente central y permite gestionar qué aplicaciones cliente tienen permiso para acceder a él. El componente de administración parte con un único usuario administrador; el usuario "**admin**", con contraseña "**1111"**. Un administrador de FIRe debería acceder al módulo de administración, cambiar esta contraseña por defecto y, opcionalmente, dar de alta otros usuarios para el resto de administradores si los hubiera.

Para saber más de las opciones de administración de FIRe, consulte el manual del módulo de administración.

7.1 CONEXIÓN CON LA BASE DE DATOS

Para que el componente de administración pueda acceder a la base de datos de la aplicación será necesario configurar el fichero "`admin_config.properties`" con la cadena de conexión y el controlador JDBC a utilizar. Este fichero debe estar en el directorio configurado mediante la variable de entorno "`fire.config.path`" o, si no se ha configurado, en el CLASSPATH del servidor de aplicaciones.

Las propiedades que deben configurarse son:

- `bbdd.driver`
  - Clase del driver JDBC que se desea utilizar.
- `bbdd.conn`
  - Cadena de conexión a la base de datos del sistema.
- `cipher.class`
  - Nombre de la clase encargada de descifrar valores del resto de propiedades. Esta clase debe implementar la interfaz `es.gob.fire.server.decipher.PropertyDecipher`. El uso de esta funcionalidad es igual a la descrita en el apartado "Cifrado de propiedades" para el cifrado de propiedades del componente central.

Ejemplos de fichero de configuración son los siguientes:

```
bbdd.driver=com.mysql.jdbc.Driver
bbdd.conn=jdbc:mysql://172.24.31.110:3306/fire_db?user=clave&password=123
#cipher.class=
```

```
bbdd.driver=com.mysql.jdbc.Driver
bbdd.conn=jdbc:mysql://172.24.31.110:3306/fire_db?user=clave&password={@ci-
phered: Gh76/bas6FtC4ep= }
cipher.class=es.gob.miapp.fire.MiPasswordDecipher
```

8 COMPONENTES ADICIONALES

Los componentes relevantes de FIRe se han descrito en los anteriores apartados, pero pueden existir otros componentes que den soporte a alguna de las funcionalidades agregadas a FIRe, como conectores para el uso de proveedores en la nube o gestores de datos para la recuperación y guardado de los datos procesados por FIRe.

En este apartado se describen los componentes adicionales que se distribuyen junto a FIRe. Considere que estos componentes sirven a un fin concreto y, según el despliegue o la funcionalidad configurada en FIRe, es posible que no sea necesario su despliegue.

8.1 SERVICIO AUXILIAR DEL CONECTOR DE LA FNMT

Este servicio permite al conector del proveedor de la FNMT procesar las peticiones *Oauth* para la autorización del usuario. Este servicio sólo necesita desplegarse cuando se configure el conector de la FNMT en su componente central.

El servicio se distribuye en el fichero "`fnmt-fire-service.war`" y debe desplegarse en un contenedor de *servlets* compatible. Para el funcionamiento de este servicio es necesario disponer de Java 7 o superior. Este servicio puede desplegarse en los mismos nodos en los que se despliegue el componente central.

Este servicio no requiere ningún tipo de configuración.

9 CONFIGURACIÓN DE NUEVOS PROVEEDORES

FIRe permite configurar múltiples proveedores de firma en la nube. En la distribución básica de FIRe se distribuyen los conectores de varios proveedores, pero se pueden agregar otros nuevos, que se distribuyan de forma independiente o desarrollados por su organismo, y que se deseen incorporar en un despliegue de FIRe.

Es muy importante comprobar que los conectores que se desean incorporar a FIRe proceden de entidades de confianza. Estos conectores se ejecutarán dentro del servidor de aplicaciones de su organismo, con lo cual pueden llegar a afectar a FIRe u otras aplicaciones desplegadas, así como acceder a información de los usuarios de FIRe y los documentos que se procesan. No despliegue en FIRe conectores de dudosa procedencia o que crea que pueden suponer un riesgo para la seguridad de su información o de sus usuarios.

Tenga en cuenta que el despliegue de un nuevo conector implica la modificación del despliegue actual y debería realizarse con el servidor de aplicaciones detenido para garantizar el correcto funcionamiento. Prepare todo de antemano antes de realizar el despliegue e intente realizarlo en un momento de baja carga de trabajo del servidor. En un despliegue sobre múltiples nodos puede ir abordando esta tarea de forma secuencial para no interrumpir el servicio.

En este apartado se describen los pasos genéricos para el despliegue de nuevos conectores en el componente central de FIRe. Es probable que a estos pasos se agreguen otros específicos para cada conector particular. Consulte la documentación de los conectores que desee integrar para conocer las necesidades concretas de estos y las opciones de configuración que soportan.

La sucesión de pasos a seguir es:

1. Agregar al componente central de FIRe las bibliotecas del nuevo conector.
    - El componente central de FIRe consiste en un archivo WAR que debe desplegarse en un servidor de aplicaciones. Este WAR contiene todos los JAR que son dependencias del componente central y de los conectores que incorpora por defecto.
    - Para desplegar el nuevo conector, introduzca en el WAR "`afirma-signature.war`" los JAR del nuevo conector y sus dependencias. Esto puede hacerlo mediante una herramienta de compresión de ficheros, ya que un fichero WAR es un fichero ZIP con otra extensión. Estos ficheros JAR deberán copiarse en el subdirectorio "`WEB-INF\lib`".
    - Compruebe que las dependencias del nuevo conector no incluyen bibliotecas que puedan generar incompatibilidades con alguna dependencia de algún otro conector desplegado. Por ejemplo, que haya varias bibliotecas iguales con distinto nombre (si las hay, elimine todas salvo una de ellas), varias versiones de una misma biblioteca (elimine todas salvo la versión más nueva), o bibliotecas que se sepan incompatibles. En el caso de que no se puedan solventar estas incompatibilidades, consulte la documentación del conector por si existiese un modo alternativo de despliegue.
2. Desplegar los servicios auxiliares y recursos externos.
    - En caso de que su conector cuente con servicios auxiliares externos para su funcionamiento, despliéguelos normalmente en el servidor de aplicaciones del componente central o en algún otro servidor accesible desde este.

- Si se requiere del uso de un recurso externo, se requiere el uso de un directorio temporal o similar, dispóngalos siguiendo las pautas dadas en el manual del conector.

3. Preparar el fichero de configuración del conector
    - Cree en el directorio de ficheros de configuración de FIRe un nuevo fichero llamada *PROVEEDOR*`_config.properties`, en donde *PROVEEDOR* será el nombre que se desea asignar al conector. Este fichero será un `Properties` en el que se deberán configurar todas las propiedades determinadas en la documentación del conector.
4. Habilitar el conector
    - Edite el fichero `config.properties` de FIRe para agregar la propiedad "`provider.`*PROVEEDOR*", en donde *PROVEEDOR* será el nombre de proveedor que ya usamos en el paso anterior. Como valor de esta propiedad se establecerá el nombre de la clase conectora. El nombre completo de esta clase se le debe proporcionar en la documentación del conector.
    - Edite la propiedad "`providers`" del fichero `config.properties` para agregar al listado el nombre del nuevo proveedor.

10 DESPLIEGUE EN ENTORNOS BALANCEADOS

Los componentes distribuidos y de administración de FIRe no almacenan datos temporales o de sesión en el servidor, por lo que pueden desplegarse en un entorno balanceado sin necesidad de realizar ninguna actuación adicional para garantizar su funcionamiento.

El componente centralizado y el servicio de prueba, en cambio, almacenan datos temporales en disco. El componente centralizado, además, trabaja con sesiones y dispone tanto de servicios utilizados por la aplicación cliente como de páginas web a las que acceden los usuarios finales, por lo que es común que en la ejecución de un mismo trámite participen distintos nodos que deberían compartir unidad de disco y objetos en memoria.

Para simplificar el despliegue en entornos balanceados, el componente central permite el guardado de sesiones para su compartición entre los distintos nodos en los que se ha desplegado el componente central. Para el uso de esta facilidad, se deberán seguir los siguientes pasos:

1. Disponer de almacenamiento compartido entre todos los nodos que desplieguen el componente central. Es imprescindible que todos estos nodos tengan acceso a este almacenamiento compartido.
2. Configurar que los objetos temporales del componente central se almacenen en el almacenamiento compartido. Esto se realizará a través de la propiedad "`temp.dir`" del fichero "`config.properties`", mediante la cual se configura la ruta del directorio de ficheros temporales de la aplicación.
3. Mantener la hora de todos los nodos sincronizados en la medida de lo posible. Cada uno de los nodos puede ejecutar revisiones periódicas de los ficheros temporales para eliminar aquellos que se consideren caducados. En caso de desincronización, un nodo podría escribir un fichero y otro, con una hora más, eliminarlo de inmediato por considerarlo caducado según su reloj.
4. Configurar un gestor para la compartición de sesiones a través del fichero de configuración del componente central.
    - Se puede establecer el gestor de sesiones a través de la propiedad "`sessions.dao`" del fichero "`config.properties`".
    - Los valores aceptados son:
        - `filesystem`
            - Gestor que comparte las sesiones a través de disco. Para ello utiliza el directorio configurado en la propiedad "`temp.dir`".
    - No se debería configurar un gestor de sesiones si no se está realizando un despliegue en un entorno balanceado, ya que su uso implica la realización de tareas innecesarias cuando el despliegue se realiza en un único nodo.
5. Configurar que los objetos temporales del servicio simulador se almacenen en el almacenamiento compartido. Esto se realizará a través de la propiedad "`tmp_dir`" del fichero "`test-backend.properties`", mediante la cual se configura la ruta del directorio de ficheros temporales de la aplicación.

ANEXO I. MIGRACIÓN A FIRE 2.3

I.1. MIGRACIÓN DESDE FIRE 2.1 / 2.1.1

I.1.1. MIGRACIÓN DE LA BASE DE DATOS

FIRe 2.2 introdujo la posibilidad de gestionar una mayor cantidad de elementos desde su módulo de administración, lo que implicó cambios en su base de datos y en cómo el componente central accede a la información de las aplicaciones.

Se distribuye junto a FIRe un script SQL para actualizar las bases de datos Oracle y MySQL desde versiones anteriores de FIRe ("`Migracion_fire_2_-_2_2.sql`"). Este script creará las nuevas tablas de base de datos necesarias y migrará los datos de su base de datos al nuevo modelo para que no tenga que hacer nada de forma manual.

Algunos datos de las nuevas tablas, se rellenarán con datos por defecto por no haber existido hasta el momento. Consulte el manual de administración e FIRe para conocer las ventajas del nuevo módulo de administración y como, opcionalmente, completar estos datos.

La información de las aplicaciones de la antigua base de datos se conservará en la tabla "`tb_aplicaciones_old`".

Si tiene problemas al migrar a la nueva versión de FIRe, pruebe a eliminar el resto de tablas de la base de datos, ejecutar los scripts de su anterior versión de FIRe y copiar los datos de la tabla "`tb_aplicaciones_old`" a "`tb_aplicaciones`".

Si completa satisfactoriamente la migración a la última versión de FIRe, ya puede eliminar la tabla "`tb_aplicaciones_old`".

Si se utiliza un SGBD distinto a Oracle y MySQL, su administrador (DBA), deberá generar las sentencias SQL necesarias para generar el nuevo modelo de dato y migrar los datos de uno a otro.

Los pasos para realizar esta tarea de forma manual son:

1. Traspasar los datos de usuario por defecto creado en la tabla `tb_configuracion`, a `tb_usuarios` donde el campo `parametro` se corresponderá con el campo `nombre_usuario` y el campo `valor` se corresponde con el campo `clave`. Los campos `nombre` y `apellidos` al ser obligatorios deberán rellenarse con algún dato por defecto al insertarlos.

- Traspasar los datos de los certificados de la tabla `tb_aplicaciones` a la nueva tabla `tb_certificados`. Los campos `cer` y `huella` a los campos `cert_principal` y `huella_principal` respectivamente. Se debe tener en cuenta que, al crear los registros en la nueva tabla, el campo `id_certificado` que es auto-numérico se corresponderá con la nueva tabla `tb_aplicaciones` con el campo `fk_certificado` y deberá tener un nombre para el certificado, en los ejemplos de los scripts de traspaso para MySQL y ORACLE el nombre del certificado se compone de "CERT_" + campo `id` de la tabla `tb_aplicaciones`.
- Traspasar los datos de la anterior tabla de `tb_aplicacione` a la nueva tabla, sin los datos anteriormente mencionados de certificados (`cer` y `huella`). En dicho traspaso se deberán también indicar el dato numérico en el campo `fk_certificado` correspondiente al campo `id_certificado` creado anteriormente en la tabla de certificados.

I.1.2. MIGRACIÓN DE LA CONFIGURACIÓN DEL COMPONENTE CENTRAL

FIRe 2.2 introduce nuevas opciones de configuración, así como el nuevo sistema multiproveedor que permite agregar nuevos proveedores y obliga a que estos se configuren de forma independiente.

Para reutilizar la configuración de su FIRe 2.1/2.1.1 en el nuevo FIRe 2.2 deberá aplicar los siguientes cambios:

- Fichero `config.properties`
    - La propiedad "`temp.clavefirma.timeout`" ha cambiado de nombre. Renómbrela como "`temp.fire.timeout`".
    - La propiedad "`backendClassName`" desaparece, ya que ahora es posible configurar varios proveedores simultáneos. Para migrar su configuración actual, realice las siguientes acciones según el conector que tuviese configurado:
        - Si tenía configurado el conector del simulador de pruebas:
            - Renombre la propiedad "`backendClassName`" como "`provider.clavefirmatest`".
            - Agregue a su fichero de configuración la propiedad "`providers`" con el valor "`clavefirmatest,local`".

```
providers=clavefirmatest,local
provider.clavefirmatest=es.gob.fire.server.connector.test.TestConnector
```

        - Si tenía configurado el conector del simulador de pruebas:
            - Renombre la propiedad "`backendClassName`" como "`provider.clavefirma`".

- Agregue a su fichero de configuración la propiedad "`providers`" con el valor "`clavefirma,local`".

```
providers=clavefirma,local
provider.clavefirma=es.gob.fire.server.connector.clavefirma.ClaveFirmaConnector
```

  - La propiedad de configuración correspondiente al conector de Cl@ve Firma se deberá trasladar al fichero "`provider_clavefirma.properties`". Esta es:
    - `clavefirma.providerName`

    Consulte el punto correspondiente al fichero "`provider_clavefirma.properties`" para más información.

  - Las propiedades de configuración correspondientes al conector del simulador de Cl@ve Firma se deberán trasladar al fichero "`provider_clavefirmatest.properties`".
    - `test.endpoint`
    - `test.ssl.keystore`
    - `test.ssl.keystorePass`
    - `test.ssl.keystoreType`
    - `test.ssl.truststore`
    - `test.ssl.truststorePass`
    - `test.ssl.truststoreType`

    Consulte el punto correspondiente al fichero "`provider_clavefirmatest.properties`" para más información.

- Fichero `gatewayapi.properties`
  - Este fichero desaparece y sus propiedades deben trasladarse al fichero "`provider_clavefirma.properties`".
  - Consulte el punto correspondiente al fichero "`provider_clavefirma.properties`" para más información.
- Fichero `provider_clavefirma.properties`
  - Este fichero configura el comportamiento del conector de Cl@ve Firma y debe contener la propiedad "`clavefirma.providerName`" extraída del fichero "`config.properties`" y todas las propiedades del fichero "`gatewayapi.properties`".
  - Este fichero también permite configurar nuevas propiedades. Consulte el apartado Fichero provider_clavefirma.properties para conocerlas.
  - Por ejemplo:

```
clavefirma.providerName=MI_PROVIDER_NAME
URL_GATEWAY=https://direccionclavefirma.gob.es:452/servicio
AUTH_STORE=RUTA_ABSOLUTA_ALMACEN_PKCS12
AUTH_STORE_PASS=CONTRASEÑA_ALMACEN_PKCS12
```

- Fichero `provider_clavefirmatest.properties`
    - Este fichero configura el comportamiento del conector del simulador de Cl@ve Firma para pruebas y debe contener las propiedades que comenzaban por "`test.`" del fichero "`config.properties`".
    - Por ejemplo:

```
test.endpoint=https://127.0.0.1:8443/clavefirma-test-services
test.ssl.keystore=C:/Users/usuario/SSL/client_ssl.jks
test.ssl.keystorePass=12345678
test.ssl.keystoreType=JKS
test.ssl.truststore=all
#test.ssl.truststorePass=
#test.ssl.truststoreType=
```

Consulte el apartado Fichero config.properties para identificar si la nueva versión de FIRe incluye alguna nueva propiedad de configuración que pueda ser de interés para su despliegue.

I.1.3. MIGRACIÓN DE LAS APLICACIONES

Las aplicaciones que utilicen FIRe no deberán realizar ningún cambio en su código. Sin embargo, es necesario que actualicen el componente distribuido que estén utilizando (Java, .NET o PHP) por el correspondiente de la versión de FIRe que se haya desplegado.

En el caso de utilizar el componente distribuido Java, también será necesario importar en la aplicación la biblioteca de SLF4J (versión 1.7.25) y la biblioteca correspondiente al sistema de log que se desee utilizar. Consulte el manual del integrador de FIRe para más información.

I.2. MIGRACIÓN DESDE FIRE 2.2

I.2.1. MIGRACIÓN DE LA BASE DE DATOS

No es necesario realizar ningún cambio en la base de datos para migrar de FIRe 2.2 a FIRe 2.3.

I.2.2. MIGRACIÓN DE LA CONFIGURACIÓN DEL COMPONENTE CENTRAL

No es necesario realizar ningún cambio en la configuración del componente central para migrar de FIRe 2.2 a FIRe 2.3.

I.2.3. MIGRACIÓN DE LAS APLICACIONES

Las aplicaciones que utilicen el componente distribuido de FIRe 2.2 no deberán realizar ningún cambio en su código. Los componentes distribuidos de FIRe 2.2 son compatibles con FIRe 2.3, por lo que tampoco es obligatoria su actualización, aunque sí aconsejable para aprovechar las nuevas características integradas en esta versión.

Para actualizar al nuevo componente distribuido sólo es necesario sustituir el antiguo componente por el nuevo (JAR, DLL o fichero PHP).

En el caso de utilizar el componente distribuido Java, también será necesario importar en su aplicación la biblioteca de SLF4J (versión 1.7.25) y la biblioteca correspondiente al sistema de log que se desee utilizar. Consulte el manual del integrador de FIRe para más información.

ANEXO II. DESPLIEGUE DE DEMOSTRACIÓN SOBRE APACHE TOMCAT

En el kit de integración se ofrece un despliegue de demostración realizado sobre Apache Tomcat para que sirva de referencia y de ayuda en el uso de los diferentes ficheros de configuración.

Este Tomcat:

- Tiene desplegado los WAR del componente central (`fire-signature.war`), la aplicación de ejemplo (`fire-test-jsp.war`), el servicio de prueba (`clavefirma-test-services.war`) y el componente de administración (`fire-admin-jsp.war`).

- Tiene configurado en el fichero "`catalina.properties`" una nueva ruta al CLASSPATH para que los ficheros de configuración se puedan cargar desde "*webapps/fire_config*".

- Tiene configurado en el fichero "`server.xml`" el uso de SSL con autenticación cliente y un TrustManager a medida para que deje pasar peticiones independientemente del certificado utilizado. El certificado SSL servidor se incluye en el propio Tomcat.

- En el directorio de bibliotecas de Tomcat se ha incluido el JAR "`fire-trustManagerClassName-1.0.jar`", para permitir que el servidor deje pasar cualquier certificado cliente de autenticación y sea la propia aplicación la que decida si lo acepta o rechaza.

Para hacer funcionar la aplicación de prueba de FIRe sobre este despliegue es necesario:

- Modificar el fichero "`client_config.properties`" para configurar el certificado de autenticación cliente (que es el que debe darse de alta en base de datos para la aplicación).

- Configurar en el fichero "`config.properties`" la cadena de conexión a la base de datos y los datos del certificado cliente SSL del servicio de pruebas.

- Configurar en el fichero "`admin_config.properties`" los elementos de la cadena de conexión a la base de datos.

- Dar de alta en base de datos la aplicación a través de la herramienta de administración.

- Configurar en el fichero "`test-app.properties`" el identificador que la herramienta de administración ha asignado a la aplicación.

- Configurar en el fichero "`test-backend.properties`" el que será el directorio temporal y asegurarse de que tiene permisos para que la aplicación desplegada cree y elimine ficheros en él.

- **ADVERTENCIA**: El componente de firma local AutoFirma incluye como medida de seguridad no firmar cuando se ejecuta desde una web desplegada en local (127.0.0.1 o localhost). Para poder utilizar las funciones de firma con certificado local, se deberá optar por una de las siguientes opciones:

    - Realizar el despliegue en un equipo remoto y modificar todos los ficheros de configuración con la IP o nombre de dominio correspondiente.
    - Configurar en todos los ficheros la IP asignada al equipo (según la configuración de red esta IP podría variar a lo largo del tiempo).
    - Agregar uno o más alias a la dirección 127.0.0.1 en el fichero "hosts" del sistema y utilizar este alias como nombre de dominio en todos los ficheros de configuración.

Algunos aspectos a tener en cuenta:

- Puede cambiarse la ruta en la que están desplegados los servicios *backend* de pruebas a través del fichero de configuración del componente central. Si no se establece, se buscará por defecto en "https://127.0.0.1:8443/clavefirma-test-services/".

- Si en el fichero "`config.properties`" se configura el conector "`es.gob.fire.signature.connector.clavefirma.ClaveFirmaConnector`", se dejará de usar el fichero "`test-backend.properties`" y se usará "`gatewayapi.properties`" para conectar contra el servicio real de custodia de claves.

ANEXO III. PROBLEMAS CONOCIDOS

III.1. ERROR EN LAS FIRMAS XADES EN DESPLIEGUES SOBRE JBOSS

El servidor de aplicaciones JBoss, en algunas de sus versiones, incluye las bibliotecas de Xerces como extensiones accesibles por las aplicaciones. Estas bibliotecas sustituyen a la versión de las mismas incluidas en Java, lo que provoca que FIRe se encuentre con una versión distinta a la esperada. Este problema deriva en que las firmas XAdES fallen a la hora de construir el XML de la firma.

En algunas versiones de JBoss, como con JBoss 7, este problema se puede solventar mediante un mecanismo incluido en el propio FIRe. Por medio de la propiedad `signature.alternativeXmldsig,` incluida en el fichero de configuración `config.properties,` se puede notificar al componente para que intente utilizar correctamente la nueva versión de la biblioteca.

Sin embargo, esto no resuelve todos los casos de error detectados. En algunos casos, como con JBoss FUSE, será necesario indicar expresamente al servidor de aplicaciones que no utilice estas bibliotecas. En ese caso, podemos dejar la propiedad `signature.alternativeXmldsig` con el valor por defecto (`false`) y seguir los siguientes pasos:

1. En el fichero "`etc/config.properties`" de JBoss, se deben eliminar los paquetes "`org.apache.xerces.*`" de la propiedad "`org.osgi.framework.bootdelegation`" para así evitar la carga de Xerces.
2. En el fichero "`etc/jre.properties`" se deben agregar las siguientes referencias para permitir su carga desde las bibliotecas de la JRE:
    - `com.sun.org.apache.xerces.internal.dom`
    - `org.jcp.xml.dsig.internal.dom`