 

**Manual para uso y configuración de conectores en FIRe**

Versión: 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | |
| Título | | Manual de para uso y configuración de conectores en FIRe | |
| Autor | | Secretaría General de Administración Digital  Ministerio de Asuntos Económicos y Transformación Digital | |
| Fecha versión 1.0 | | 17 de agosto de 2021 | |
|  | | | |
| Versión | Fecha | Responsable | Cambios introducidos |
| 1.0 | 17-08-2021 | DTIC | Creación del documento |

ÍNDICE

[1. Objeto del documento 4](#_Toc80091579)

[2. Introducción 5](#_Toc80091580)

[3. procesos soportados 6](#_Toc80091581)

[4. Funciones de los conectores 7](#_Toc80091582)

[5. Propiedades de los conectores 11](#_Toc80091583)

[5.1. Ejemplo de configuración de archivo de propiedades. 12](#_Toc80091584)

# Objeto del documento

El presente manual detalla cómo usar, configurar e integrar los distintos conectores de los que dispone FIRe.

# Introducción

FIRe es un sistema de firma basado en el uso de certificados en la nube y diseñado para su integración en los flujos de firma de aplicativos web.

FIRe hace uso de conectores Java con los cuales se integran con los distintos proveedores de servicios de custodia compatibles. En un despliegue de FIRe se configurará uno o más de estos conectores para permitir a los usuarios seleccionar el proveedor que desea utilizar y hacer uso de sus certificados de firma custodiados por este proveedor.

El diseño de estos conectores se ha realizado en base al modelo de uso definido por el API utilizado para la integración con Cl@ve Firma, el servicio de custodia de la Gerencia Informática de la Seguridad Social (GISS).

El uso del API proporcionado por cualquier proveedor puede estar restringido por cualquier mecanismo de autenticación determinado por el propio proveedor. Así, por ejemplo, podría ser necesario que el proveedor registrase primeramente a las entidades que quisieran hacer uso de su API y que después el API usase certificados SSL cliente conocidos por el proveedor para autenticarse en las llamadas a sus servicios.

# procesos soportados

FIRe define dos procesos que hacen uso de los conectores con los proveedores de certificados en la nube:

* Firma de datos: Este proceso comprende la selección de un certificado de firma, la carga de los datos a firmar, la autorización de la operación y la recuperación del resultado. Este proceso es utilizado de igual manera por las funciones de firma simple y firma masiva de FIRe.

Para el proceso de firma de datos se seguirá la siguiente secuencia de operaciones:

* 1. Inicialización del conector.
  2. Listado de certificados de firma del usuario.
  3. Carga de los datos a firmar.
  4. Autorización de la operación de firma por parte del usuario.
  5. Recuperación de la firma generada.
  6. Liberación de recursos.
* Generación de certificados: Este proceso es opcional y permite la emisión de certificados para un usuario que previamente se ha dado de alta en el sistema de custodia y del que este ha confirmado su identidad.

Para el proceso de generación de certificados se seguirá la siguiente secuencia de operaciones:

1. Inicialización del conector.
2. Solicitud de emisión de un certificado de firma para el usuario.
3. Autorización de la operación de emisión por parte del usuario.
4. Recuperación del certificado generado.

# Funciones de los conectores

Las secuencias de operación presentadas se componen mediante llamadas a las siguientes funciones de los conectores:

* void **init**(Properties config);
  + Método encargado de inicializar una operación, ya sea de firma o de emisión de certificados.
  + Este método realizará el inicio de la operación mediante el API del proveedor proporcionándole, al menos:
    - URL a la que redirigir al usuario en caso de producirse un error durante la operación de firma, emisión de certificado o autorización/autenticación del usuario.
    - URL a la que redirigir al usuario en caso de que la operación finalice correctamente.
  + Este método puede recibir además todas las propiedades que el API del proveedor considere necesarias para su inicialización.
* void **initOperation**(Properties config);
  + Método encargado de inicializar una transacción con el conector.
  + Este método realizará el inicio de la operación mediante el API del proveedor proporcionándole, al menos:
    - URL a la que redirigir al usuario en caso de producirse un error durante la inicialización de la transacción.
    - URL a la que redirigir al usuario en caso de que la operación finalice correctamente.
  + Este método puede recibir además todas las propiedades que el API del proveedor considere necesarias para la inicialización de la transacción.
* X509Certificate[] **getCertificates**(  
   String subjectId)  
   throws  
   FIReCertificateException,  
   FIReConnectorUnknownUserException,  
   FIReConnectorNetworkException,  
   CertificateBlockedException,  
   WeakRegistryException;
  + Método para la recuperación de los certificados de firma.
  + Este método realizará las llamadas oportunas al API del proveedor proporcionándole el DNI de un usuario para poder recuperar los certificados de firma que puede utilizar.
  + El comportamiento del API tras la llamada debe permitir identificar los siguientes casos de error.
    - El usuario no está dado de alta en el sistema de custodia.
    - El usuario no tiene certificados de firma.
    - El usuario no tiene certificados y no se le pueden emitir certificados (ya sea porque el proveedor no permita generar certificados en ningún caso o porque ese usuario no tiene permisos por haber realizado una autenticación no fehaciente o similar).
    - El usuario o sus certificados de firma están bloqueados y no pueden utilizarse.
    - No se ha podido conectar con el sistema de custodia.
    - Ocurrió un error al recuperar los certificados.
* LoadResult **loadDataToSign**(  
   String subjectId,  
   String algorithm,  
   TriphaseData td,  
   Certificate signCert)  
   throws  
   FIReCertificateException,  
   FIReSignatureException,  
   IOException,  
   FIReConnectorUnknownUserException,   
   FIReConnectorNetworkException;
  + Método para la carga de datos.
  + Este método realiza las llamadas oportunas al API del proveedor para indicarle qué datos se desean firmar y con qué certificado y algoritmo.
  + Los parámetros que se le pueden proporcionar al API son el DNI de la persona que desea realizar la firma, el certificado de firma que desea utilizar (obtenido mediante una llamada previa al método de recuperación de certificados), los datos que se desean firmar (concretamente el hash de todos los datos que se desean firmar, ya sólo para realizar una firma PKCS#1), un identificador para cada dato/hash y el algoritmo de firma o algoritmo de huella digital con el que se calcularon todos los hashes.
  + El API del proveedor deberá proporcionar un identificador de transacción y una URL a la que redirigir al usuario para que autorice la operación de firma. Al usuario se le mostrará la página resultante de cargar esa URL y, una vez introduzca los datos de identificación necesarios, la propia página le deberá redirigir a la URL de éxito y error según los datos introducidos y si fue posible completar la operación de firma. Opcionalmente, el API puede recibir el nombre y/o título de los documentos que se firman (a los cuales corresponden los hashes) para que se le muestren al usuario en la página de autorización de la operación.
  + El comportamiento del API tras la llamada debe permitir identificar los siguientes casos de error:
    - El usuario no está dado de alta en el sistema de custodia.
    - El certificado indicado está bloqueado o no es válido.
    - No se ha podido conectar con el sistema de custodia.
    - Ocurrió un error al cargar los datos.
* Map<String, byte[]> **sign**(String transactionId)  
   throws  
   FIReSignatureException,  
   FIReConnectorUnknownUserException;
  + Método para la recuperación de las firmas electrónicas generadas.
  + Este método realiza las llamadas oportunas al API para recuperar la firma generada una vez realizada la autorización.
  + La información que se le pueden proporcionar al API del proveedor es el identificador de transacción proporcionado como respuesta al método de carga de datos.
  + La respuesta del API debe incluir, al menos, todas las firmas PKCS#1 generadas y los identificadores de los documentos a los que pertenecen cada una de ellas.
  + El comportamiento del API tras la llamada debe permitir identificar los siguientes casos de error:
    - El usuario no está dado de alta en el sistema de custodia.
    - No se ha podido conectar con el sistema de custodia.
    - Ocurrió un error al recuperar las firmas generadas.
* void **endSign**(final String transactionId);
  + Informa del fin de una transacción de firma.
  + Este método puede llamar, opcionalmente, a una función del API dell proveedor al que se le pase el identificador de transacción proporcionado por el método de carga de datos para que dé finalizada la transacción y así pueda liberar los recursos que hubiese bloqueado.
* boolean **allowRequestNewCerts**();
  + Indica si se permite generar nuevos certificados de firma a los usuarios.
  + Devolverá ‘true’ en caso de que a los usuarios se les permita generar un certificado cuando no tengan ya uno. Devolverá ‘false’ en caso contrario.
* GenerateCertificateResult **generateCertificate**(  
   String subjectId)  
   throws  
   FIReCertificateAvailableException,  
   FIReCertificateException,  
   FIReConnectorNetworkException,  
   FIReConnectorUnknownUserException,  
   WeakRegistryException;
  + Método para la solicitar la emisión inmediata de un certificado de firma.
  + Si el proveedor lo soporta, este método realiza las llamadas oportunas a su API para solicitar la emisión de un certificado para un usuario.
  + La información que se le proporciona es el DNI del usuario que solicita el certificado.
  + La respuesta del API debe contener, al menos, el identificador de la transacción de generación de certificado y la URL de una página a la que redirigir al usuario para que autorice la operación.
  + El comportamiento del API debe permitir identificar los siguientes casos de error:
    - El usuario no está dado de alta en el sistema de custodia.
    - El usuario ha alcanzado el máximo número de certificados que se le permite obtener.
    - El usuario no realizó un registro fehaciente y, por tanto, no se pueden emitir certificados a su nombre.
    - No se ha podido conectar con el sistema de custodia.
    - Ocurrió un error al solicitar la generación del certificado.
* byte[] **recoverCertificate**(String transactionId)  
   throws  
   FIReCertificateException,  
   FIReConnectorNetworkException;
  + Método para la recuperación de un certificado recién generado.
  + Si el proveedor lo soporta, este método realizará las llamadas oportunas al API del proveedor para poder recuperar un certificado de firma recién generado. En caso de que la aplicación sólo permitiese (ahora y en el futuro) generar un certificado por usuario sería posible cubrir esta funcionalidad con una llamada a los métodos de recuperación de todos los certificados de firma.
  + Este método puede proporcionar al API el identificador de transacción obtenido como respuesta al método de generación de certificados.
  + Como resultado, el API debe poder recuperar al menos el que sin duda es el certificado recién generado.
  + Los casos de error que debe ser capaz de identificar el API del proveedor son:
    - El usuario no está dado de alta en el sistema de custodia.
    - No se ha podido conectar con el sistema de custodia.
    - Ocurrió un error al recuperar el certificado generado.
* String **userAutentication**(String subjectId, String okRedirectUrl, String errorRedirectUrl);
  + Método que permite implementar un mecanismo de autenticación para la obtención de certificados de la nube.
  + Este método necesitará que se le proporcione al menos:
    - ID del usuario que se autenticará.
    - URL a la que redirigir al usuario en caso de que la autenticación finalice correctamente.
    - URL a la que redirigir al usuario en caso de que la operación falle.
  + El método se encargará de devolver un ‘String’ con la URL que se utilizará para redirigir al servicio de autenticación.

# Propiedades de los conectores

Las propiedades que se deben establecer en el fichero de configuración de estilo son las siguientes:

* logo
  + Logo del proveedor.
  + Debe indicarse la ruta a un recurso interno con la imagen PNG o una URL a la propia imagen. La URL puede ser de tipo "http:", "https:" o "data:". Con este último tipo se podría empotrar la imagen en Base 64 dentro de la propiedad.
  + El tamaño recomendado de imagen es 319 x 109px.
* title
  + Título del proveedor. Se mostrará también como texto alternativo de la imagen.
* header
  + Cabecera a mostrar en la opción seleccionable del proveedor.
* description
  + Descripción a mostrar en la opción seleccionable del proveedor.
* noregistered
  + Texto a mostrar cuando se detecte que el usuario no está registrado en el proveedor.
* needjavascript
  + Indica si el proveedor necesita JavaScript para funcionar (true o false).
  + Si se necesita y se detecta que el navegador no lo tiene habilitado, se bloqueará el acceso a este proveedor.
* certselectioninprovider
  + En caso de que este activada permite al usuario seleccionar un certificado en la web del proveedor, para consecuentemente no tener que volver a seleccionarlo dentro de FIRe.
* allowexternalproviderinfo
  + Permite usar a los conectores archivos de tipo ‘provider\_info’ externos o no.

## Ejemplo de configuración de archivo de propiedades.

A continuación, se muestra cómo está configurado el archivo de configuración para el conector ‘clavefirmatest2’, que utiliza el archivo ‘external\_provider\_info\_clavefirmatest2.properties’.

# =============================================================================

# ESTE ES UN FICHERO DE PRUEBA QUE SE USA UNICAMENTE PARA MOSTRAR EL USO DE UN

# PROVIDER\_INFO EXTERNO. NO DEBERIA USARSE EN UN DESPLIEGUE DE PRODUCCION.

# =============================================================================

# Logo del proveedor. Debe indicarse la ruta interna a una imagen PNG

# con el logo o una URL a la propia imagen. La URL puede ser de tipo

# "http:", "https:" o "data:". El tamano recomendado de imagen es

# 309 x 109px.

logo=es/gob/fire/server/connector/test/logo-clavefirma-test.png

# Titulo del proveedor

title=Simulador de Cl@ve Firma con autenticación

# Cabecera a mostrar en la opcion seleccionable del proveedor

header=Firma con Cl@ve Firma con autenticación

# Descripcion a mostrar en la opcion seleccionable del proveedor

description=Simula la firma un certificado de Cl@ve Firma. Solicita previamenente autenticaci&oacute;n del usuario.

# Texto a mostrar cuando se detecte que el usuario no esta registrado en el proveedor

noregistered=El usuario no est&aacute; dado de alta en Cl@ve, por lo que no tiene disponible esta opci&oacute;n.

# Indica si el proveedor necesita JavaScript para funcionar

needjavascript=false

# Autenticacion del usuario para la obtencion de certificados desde la nube

requireduserautentication=true

# Propiedad que en caso de que este activada permite al usuario seleccionar un certificado en la web del proveedor,

# para consecuentemente no tener que volver a seleccionarlo dentro de FIRe

certselectioninprovider=false

allowexternalproviderinfo=true