# Manual del Programador del módulo de Firma de Ficheros de @Firma 5 - Extensión

| | |
|---|---|
| Documento nº: | TI-20-1074-@firmaExtension-PFF-001 |
| Revisión: | 001 |
| Fecha: | 09/02/2007 |
| Período de retención: | Permanente durante su período de vigencia + 3 años después de su anulación |

## CONTROL DE COMPROBACIÓN Y APROBACIÓN

Documento nº: TI-20-1074-@firmaExtension-PFF-001

Revisión: 001

Fecha: 09/02/2007

## Realizado

09/02/2007

| | |
|---|---|
| Pedro Luis | José Angel |
| Alvarez – Ossorio | Román |
| Torres | López |

## Comprobado

09/02/2007

José Antonio

Márquez

Contreras

Director @firma

## APROBADO

09/02/2007

# Control de Modificaciones

| | |
|---|---|
| Documento nº: | TI-20-1074-@firmaExtension-PFF-001 |
| Revisión: | 001 |
| Fecha: | 09/02/2007 |

| | |
|---|---|
| Rev. | 001 |
| Fecha | 09/02/2007 |
| Autor/es | PAOT,JARL |
| Descripción | Documentación inicial |

## Control de Distribución

Documento nº: TI-20-1074-@firmaExtension-PFF-001

Revisión: 001

Fecha: 09/02/2007

### Copias Electrónicas:

La distribución de este documento ha sido controlada a través del sistema de información.

### Copias en Papel:

La vigencia de las copias impresas en papel está condicionada a la coincidencia de su estado de revisión con el que aparece en el sistema electrónico de distribución de documentos.

El control de distribución de copias en papel para su uso en proyectos u otras aplicaciones es responsabilidad de los usuarios del sistema electrónico de información.

Fecha de impresión 05/03/2007 6:04

### Distribución en Papel:

| Nombre o Cargo y (Organización) | Nº de Ejemplares | Referencia de la carta de transmisión y fecha |
|---|---|---|
| | | |

# Índice

# 1 Objeto

El objeto de este documento es describir la utilización del módulo de Firma Ficheros de la Plataforma @Firma 5 - Extensión.

El módulo de Firma Ficheros contiene las interfaces necesarias para la realización de los siguientes procesos:

- Firma/MultiFirma de Ficheros por Usuario.
- Firma/MultiFirma de Ficheros por Servidor.
- Firma/MultiFirma de Ficheros en Bloque por Usuario.
- Consulta de Transacciones y Verificación de Firmas.

Los objetivos globales de este proceso son:

- Describir los pasos necesarios para desarrollar una aplicación que utilice las interfaces RMI-IIOP y WebServices disponibles en la extensión de la plataforma de firma.
- Describir los métodos disponibles en las interfaces anteriores y la lógica de utilización de los mismos.
- Describir detalladamente el ejemplo de utilización de dichas interfaces que se adjunta en la extensión de la plataforma para facilitar el trabajo al desarrollador de aplicaciones, tanto RMI-IIOP como WebServices.
- Especificar los diferentes tipos de errores que se pueden dar al integrar una aplicación con la plataforma, y su resolución.

# 2 Alcance

El presente documento recoge la utilización del módulo de Firma de Ficheros de la Plataforma de @Firma 5 - Extensión.

El módulo de Firma Ficheros contiene las interfaces necesarias para la realización de los siguientes procesos:

- Firma/MultiFirma de Ficheros por Usuario.
- Firma/MultiFirma de Ficheros por Servidor.
- Firma/MultiFirma de Ficheros en Bloque por Usuario.
- Consulta de Transacciónes y Verificación de Firmas.

# 3 Siglas

| | |
|---|---|
| AC | Autoridad de Certificación |
| EJB | Enterprise Java Bean |
| HTTPS | HyperText Transfer Protocol |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| JSP | JavaServer Pages |
| Keystore | Almacén de certificados digitales |
| RSA | Rivest Shamir Adleman |
| SSL | Secure Socket Layer |
| UO | Unidad Organizativa |
| URL | Uniform Resource Locator |

# 4 Documentos de Referencia

- Documento TI-20-1178-@Firma-Administracion-MAN, Manual de Administración de @firma 5.

- Documento TI-20-1074-@FirmaExtension-Administracion-MAN, Manual de Administración de @firma 5 - Extensión.

- Documento TI-20-1074-@FirmaExtension-MICFF, Manual del Integrador del Cliente de Firma de Fichero @firma 5 - Extensión.

- Documento TI-20-1178-@Firma-Global-MICF, Manual del Integrador del Cliente de Firma @firma 5

# 5 Introducción

El módulo de Firma Ficheros es una herramienta de Firma Digital basada en Tecnología Java que permite realizar la Firma Digital de Documentos. Puede ser integrada en aplicaciones ya existentes o que se vayan a desarrollar. Utiliza Certificados Digitales (X.509) para firmar los datos digitalmente.

El módulo de Firma Ficheros contiene las interfaces necesarias para la realización de los siguientes procesos:

- Firma/MultiFirma de Ficheros por Usuario.
- Firma/MultiFirma de Ficheros por Servidor.
- Firma/MultiFirma de Ficheros en Bloque por Usuario.
- Consulta de Transacciones y Verificación de Firmas.

Las interfaces se encuentran disponibles mediante tecnología RMI-IIOP y WEBSERVICES, ambas securizadas mediante SSL y JAAS, lo cual proporciona un doble nivel de seguridad.

## 5.1 Sobre módulo Firma Ficheros de @Firma 5 - Extensión

En el caso de Firma/MultiFirma de Ficheros por Usuario, en una primera fase se registran los documentos en el sistema de Custodia y posteriormente se firman digitalmente por un usuario desde su propia máquina. El documento, la firma y los datos asociados a la transacción de firma quedan almacenados en el sistema de Custodia de la plataforma.

En el caso de Firma/MultiFirma de Ficheros por el Servidor de Firma, se registran los documentos en el sistema de Custodia firmándose digitalmente por el Servidor de Firma en el momento del registro utilizando un certificado digital configurado para ello. El documento, la firma y los datos asociados a la transacción de firma son almacenados en el sistema de Custodia de la plataforma.

En el caso de Firma/MultiFirma de Ficheros en Bloque por Usuario, en una primera fase se registran los documentos en el sistema de Custodia quedando firmados con un Certificado Servidor configurado para ello. Posteriormente un usuario firma digitalmente un conjunto de documentos registrados (internamente se firma un resumen construido a partir de los documentos registros y firmados por Servidor). El documento, la firma y los datos asociados a la transacción de firma son almacenados en el sistema de Custodia de la plataforma.

La plataforma proporciona las interfaces RMI-IIOP y WEBSERVICES necesarios para Consultar toda la información disponible sobre cualquier transacción de firma realizada y proceder a la Verificación de cualquier firma.

Con la extensión de la plataforma de Firma se distribuyen ejemplos que muestran la utilización todas las interfaces mencionadas de la manera más eficiente y correcta. A lo largo de este

documento se describen todas las interfaces disponibles y los ejemplos de utilización de las mismas.

Adicionalmente, para facilitar la tarea del desarrollador / integrador de aplicaciones se distribuye el **"javadoc"** correspondiente a todas las interfaces, en el cual se detallan los métodos, parámetros, excepciones, etc... Esto permitirá que en el presente manual nos centremos principalmente en el funcionamiento omitiendo detalles específicos.

## 5.2 Interfaces de la plataforma de Firma para el módulo Firma de Ficheros

A continuación se muestran los nombres de las interfaces disponibles en la extensión de la plataforma para el módulo de Firma de Ficheros y se describen el ámbito de cada una de ellas.

- Interfaz com.telventi.firmaweb.FirmaWebMCA

  Permite realizar los procesos Firma de Ficheros por Usuario y Firma de Ficheros por Servidor. Adicionalmente contiene algunos métodos de consulta generales frecuentes. Se agrupan en esta interfaz por cuestiones de eficiencia en el desarrollo de aplicaciones.

- Interfaz com.telventi.firmaenbloquemca.FirmaEnBloqueMCAFacade

  Permite realizar el proceso Firma de Ficheros en Bloque por Usuario. Adicionalmente contiene los métodos de consulta específicos de bloques. Se agrupan en esta interfaz por cuestiones de eficiencia en el desarrollo de aplicaciones.

- Interfaz com.telventi.custodia.CustodiaDocumentosFacade

  Permite realizar consultas para obtener información sobre cualquier transacción de firma realizada.

- Interfaz com.telventi.verificacionfirmas.VerificarFirmas

  Permite realizar la verificación de cualquier firma realizada por la extensión de la plataforma.

## 5.3 Creación y configuración de una aplicación en la extensión de la plataforma

Para poder desarrollar una **aplicación** que utilice las distintas interfaces publicadas para el módulo de firma de ficheros es necesario dar de alta una aplicación en el módulo de Administración de la extensión de la plataforma de Firma.

A continuación se describen los pasos necesarios a llevar a cabo en dicho módulo, para información más detallada sobre el módulo de Administración de la extensión de @firma 5.0 véase el manual de Administración de la extensión:

Una vez se encuentre en el módulo de administración se accede al gestor de aplicaciones. Esto nos mostrará en la parte izquierda el árbol de unidades organizativas y aplicaciones registradas en la extensión de la plataforma:

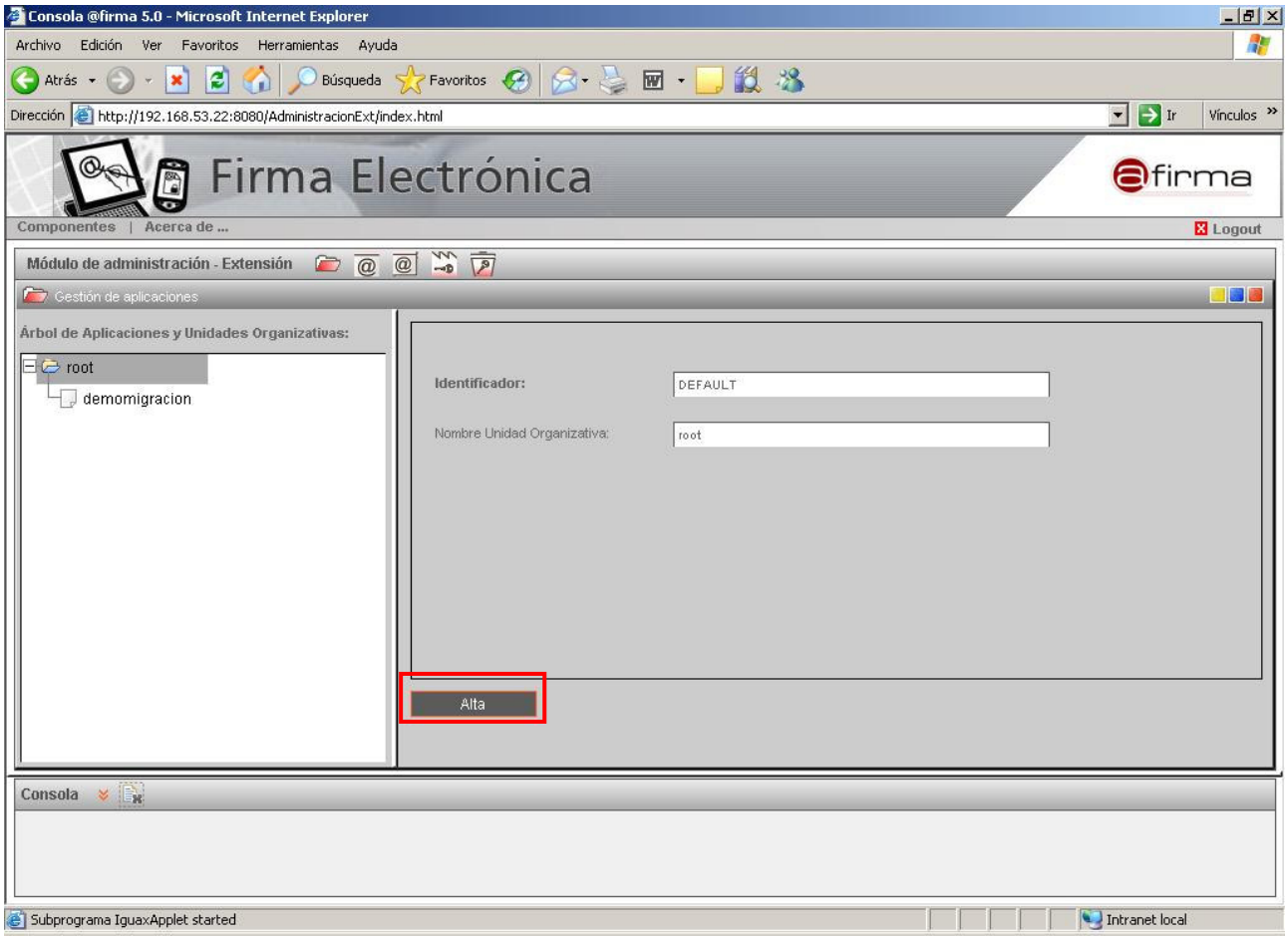


Figura 1 – Interfaz de Gestión de UO's y aplicaciones

Tras esto, se selecciona la aplicación @Firma 5.0 que se desea registrarse en la extensión de la plataforma. En este momento se mostrarán los parámetros que son configurables para una aplicación en la extensión y que deberán ser cumplimentados.

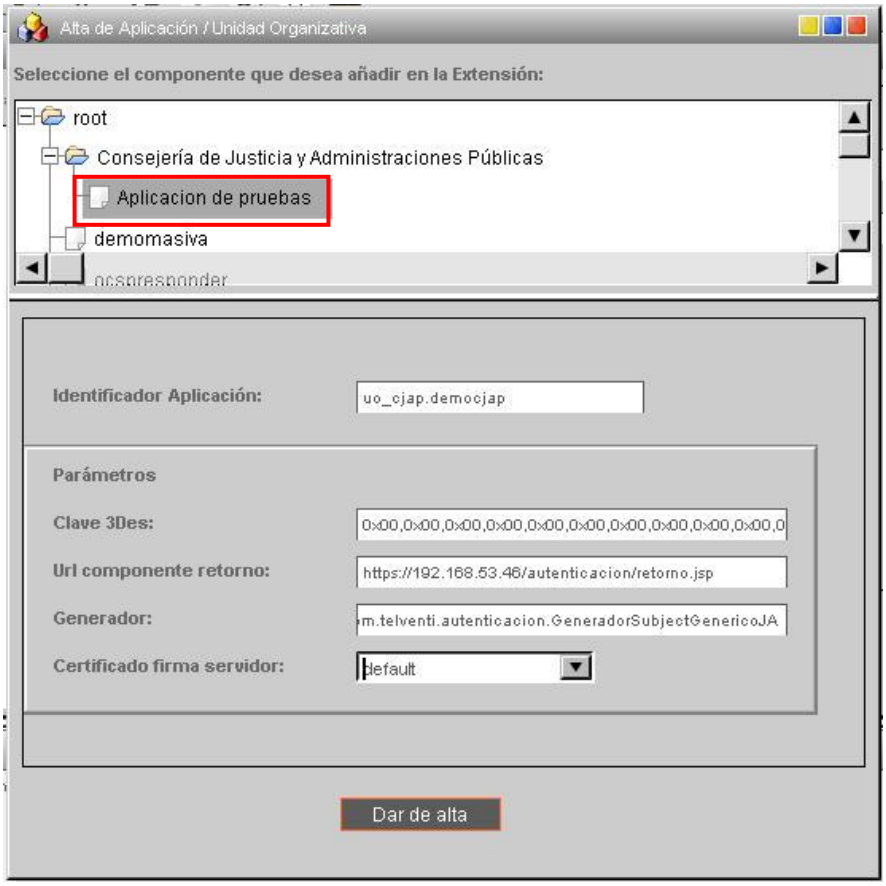


Figura 2 – Interfaz de Gestión de UO's y aplicaciones. Ventana de registro de Aplicación

Los parámetros configurables para una aplicación son los siguientes:

- Certificado de firma de servidor: Indica el certificado de servidor utilizado para realizar las firmas de servidor en procesos de firma en bloque y firmas de servidor. El conjunto de certificados mostrado se configura en el módulo de administración de @Firma 5.0, en el componente de gestión de UO's y aplicaciones mediante el parámetro "Certificados para Firma".

- Clave 3Des: Es la clave DES necesaria para encriptar los datos de vuelta en las aplicaciones de autenticación web. Es importante anotar esta clave para distribuirla posteriormente a los desarrolladores que utilicen esta aplicación. La clave puede y ha de ser cambiada con "sensata" frecuencia poniendo especial cuidado es distribuirla a todas aquellas personas que trabajen con la aplicación configurada.

- URL componente retorno. Indica la URL del componente que trata los datos de vuelta en un proceso de autenticación mediante certificados, y que está ubicado en el servidor de aplicaciones donde se necesite dicha validación.

- Generador. Es el componente de lógica de negocio desarrollado en Java que ofrece la funcionalidad requerida para una aplicación en concreto, y que ha sido desarrollado

según las especificaciones del manual del programador del Módulo de Autenticación. Por defecto, se utiliza el componente"com.telventi.autenticacion.GeneradorSubjectGenerico". Aunque se recomienda utilizar "com.telventi.autenticacion.GeneradorSubjectGenericoJA". Este parámetro no es obligatorio para aplicaciones que sólo hagan uso de los servicios de firma de ficheros de la extensión de la plataforma.

Tras pulsar el botón "Dar de Alta" la nueva aplicación quedará registrada en la plataforma con los valores que han sido introducidos.

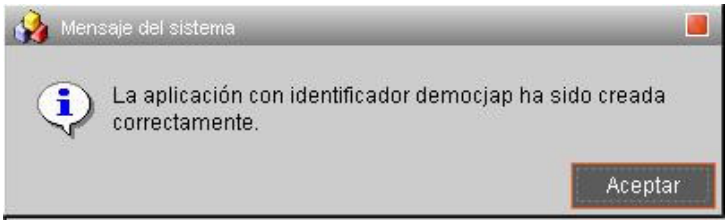


Figura 3 – Interfaz de Gestión de UO's y aplicaciones. Aplicación registrada correctamente

Si se desea acceder a la configuración de la aplicación creada, tan solo hay que posicionarse sobre la aplicación en el árbol de UO's y aplicaciones.

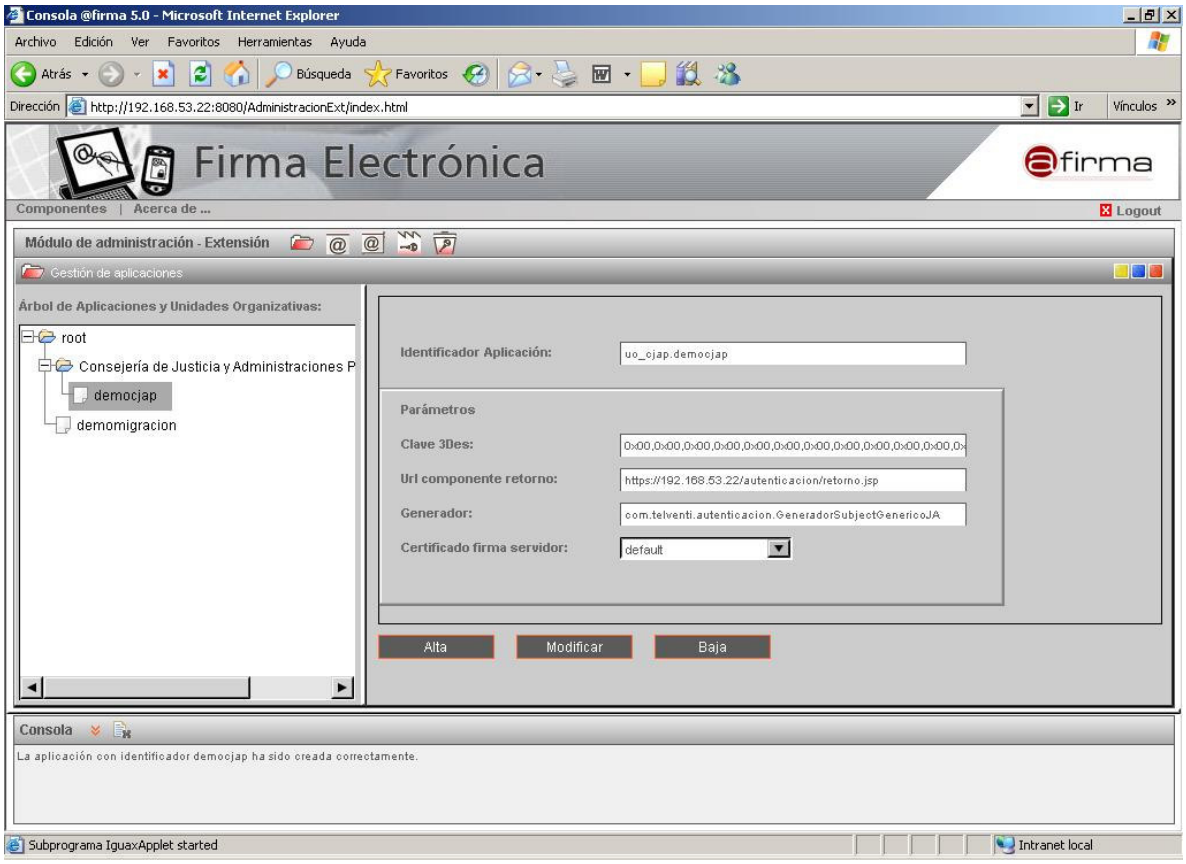

Figura 4 – Interfaz de Gestión de UO's y aplicaciones. Configuración aplicación

# 6 Proceso Firma/MultiFirma de Ficheros por Usuario

El proceso de Firma/MultiFirma de Ficheros por Usuario es implementado por la interfaz com.telventi.firmaweb.FirmaWebMCA. Esta interfaz contiene métodos para realizar un proceso de firma/multifirma de usuario y algunos métodos frecuentes para realizar consultas.

En el proceso de firma/multifirma por usuario de ficheros intervienen tres agentes: **usuario** (cliente), **aplicación** que utiliza la interfaz y la **plataforma de Firma** (interfaz de firma).

El proceso de firma se puede describir como un procedimiento en 3 pasos:

1) La **aplicación** registra el documento a firmar en la **plataforma de Firma**. (método *registrarDocumento*).

2) La **aplicación** solicita a la **plataforma de Firma** la generación de los datos a firmar, que serán enviados a la máquina del **usuario** para ser firmados. (método *generarDatosAFirmar*).

3) El **usuario** firma los datos y la **aplicación** envía el resultado a la **plataforma de Firma** para terminar el proceso. (método *generarFirma*).

La siguiente figura muestra el proceso de firma en 3 fases:

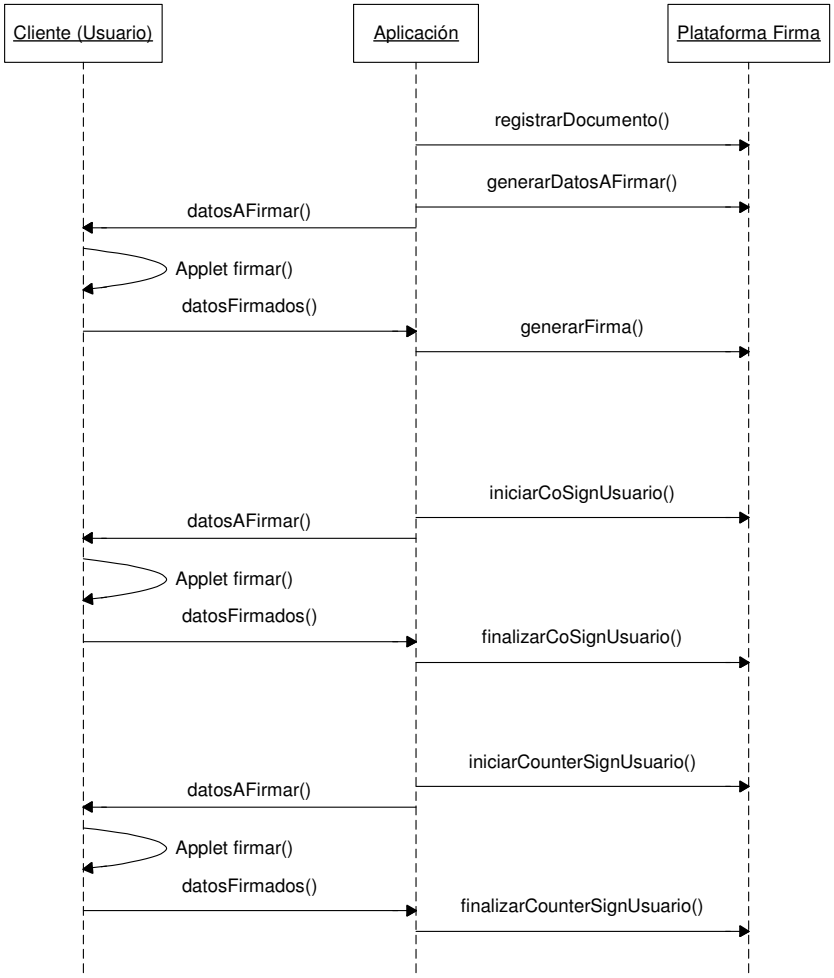


**Figura 5 – Firma / Multifirma Ficheros por usuario. Diagrama de interacción**

En un proceso de multifirma de ficheros por usuario, el documento ya fue registrado, así pues solamente se necesitan los pasos 2 y 3, pero en este caso los métodos de la interfaz son *iniciarCoSignUsuario()* y *finalizarCoSignUsuario()* para firma en paralelo y *iniciarCounterSignUsuario()* y *finalizarCounterSignUsuario()* para firma en cascada.

Los métodos de la interfaz que permiten realizar algunas consultas frecuentes y las llamadas entre la Aplicación y la plataforma de Firma se muestran en la siguiente figura.

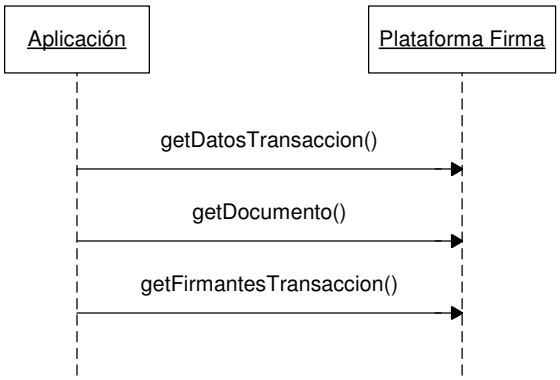


Figura 6 – Interfaz de Custodia. Diagrama interacción

Para obtener información detallada sobre los métodos de la interfaz consultar el "javadoc" proporcionado en el directorio "documentacion/javadoc/firmaficheros" del CD Desarrollo.

## 6.1 Acceso mediante RMI-IIOP

Para utilizar la interfaz RMI-IIOP FirmaWebMCA de la extensión de la plataforma se requieren clientes con máquina virtual de JAVA JDK 1.4 o superior.

En el "CD Desarrollo" se proporcionan las librerías necesarias para acceder a la interfaz. Se encuentran ubicadas en el directorio "ejemplos/firma ficheros/apiRMI-IIOPCliente". El contenido de dicho directorio es:

- lib/ApiFirmaCliente.jar : Clases e Interfaces de acceso a PKI.
- lib/jbossall-client.jar : Librerías cliente acceso JBOSS.
- conf/auth.conf : Fichero de para configuración de acceso a interfaz mediante JAAS. Este fichero debe copiarse en el directorio desde el cual se ejecuta la aplicación.
- conf/jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno CLASSPATH.

Las clases necesarias en este caso son las siguientes:

- com.telventi.utilidades.ConexionFirma
- com.telventi.firmaweb.DTOFirmante

- com.telventi.firmaweb.DTOIniciarMultifirma
- com.telventi.firmaweb.FirmaException
- com.telventi.firmaweb.FirmaWebMCA

A continuación se muestra un pequeño extracto de código JAVA que obtiene una referencia a la interfaz para poder utilizar sus métodos.

```
// IMPORTAMOS LAS CLASES NECESARIAS EN LA CABECERA
import com.telventi.firmaweb.*;
import com.telventi.utilidades.ConexionFirma;
        ……
try {
    // ESTABLECEMOS PROPIEDADES DE OBJETO ConexionFirma (SERVIDOR, USUARIO, PASSWORD)
    String host = "192.168.53.18";
    String usuario = "user01";
    String password = "12345";
    ConexionFirma.setParametrosConexion(host,usuario,password);

    // OBTENEMOS UNA REFERENCIA A LA INTERFAZ A PARTIR DE LA CONEXION
    FirmaWebMCA ficheros =  ConexionFirma.getInstance().getFirmaFicheros();
    // AHORA PODEMOS UTILIZAR CUALQUIERA DE SUS METODOS
    double id = ficheros.registrarDocumento (……..)
}catch(java.lang.Exception ex)  { }
```

El fichero auth.conf debe copiarse en el directorio desde el cual se ejecuta la aplicación, o bien, si se desea ubicar en un sitio diferente utilizar el método "setFicheroAuth()" de la clase ConexionFirma para indicarle la nueva ubicación.

```
ConexionFirma.setFicheroAuth("C:/auth.conf");
```

## 6.2 Acceso mediante WEBSERVICES

Para realizar el acceso mediante Web Services es necesario crea un cliente de conexión que sea capaz de realizar peticiones al núcleo de la plataforma.

Estos clientes pueden ser generados de forma automática mediante los Ficheros de Descripción de los Servicios Web (WSDL) publicados en el núcleo de la plataforma para la extensión, en la siguiente URL:

https://<servidor_firma>:<puerto>/axis/servlet/AxisServlet

(Será necesario introducir el usuario/password para Web Services. Consultar con algún administrador de la plataforma.)

Existen una serie de herramientas que facilitan este trabajo, entre ellas se citan las siguientes:

- Paquete AXIS JAVA: La clase "org.apache.axis.wsdl.WSDL2Java", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología JAVA.
- Paquete AXIS C/C++: La clase "org.apache.axis.wsdl.wsdl2ws.WSDL2Ws", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología C/C++.
- GSoap. Permite generar clases cliente C/C++ a partir de un fichero descriptor WSDL.
- Etc.

A los clientes generados por las herramientas anteriores posiblemente será necesario añadir el código necesario para realizar la comunicación SSL y la autenticación JAAS con la plataforma de firma. En los ejemplos proporcionados con la plataforma (JAVA) se muestran claramente los mecanismos adicionales incorporados.

Como ayuda adicional puede consultarse el "Javadoc" proporcionado en el directorio "documentacion/javadoc/firmaficheros" del CD Desarrollo y los ejemplos desarrollados en JAVA (directorio "ejemplos/firma ficheros/Ejemplos WebServices" del CD Desarrollo)

Concretamente, para obtener el fichero descriptor WSDL correspondiente a la interfaz FirmaWebMCA conectarse a la siguiente URL:

https://<servidor_firma>:<puerto>/axis/services/FirmaFicheros?wsdl

En el menú del navegador *Archivo*, seleccionar *Guardar como...* indicando el nombre firmaficheros.wsdl.

## 6.3 Utilización del Componente Cliente de Firma (Applet Cliente)

En un proceso de firma de ficheros por usuario también es necesario incorporar un componente que firme los datos en la máquina del usuario. Este componente es un Applet de Firma que debe

ser cargado en la página de la aplicación a desarrollar. Para ello se suministran los siguientes ficheros en el directorio "ejemplos / firma ficheros / clienteDeFirma" del "CD Desarrollo":

- instaladorClienteFirmaAFirma5.jar:
- clienteFirmaAFirma.jar
- clienteFirmaAFirma5.zip
- linuxLibraries.zip
- win32libraries.zip
- xades-plugin.zip
- version.properties
- scriptfirma.js

El procedimiento a seguir consiste en incorporar el fichero javascript "scriptfirma.js" en la página que utilizará el usuario para firmar.

```
<script language="javascript" src ="scriptfirma.js"></script>
```

Una vez copiado el fichero "scripfirma.js", se debe establecer los parámetros de configuración existentes en él:

- baseDownloadURL: Ruta a los instalables, si no se establece supone que esta en el mismo directorio que la pagina Web.
- base: Ruta al instalador, si no se establece supone que esta en el mismo directorio que la pagina Web.
- signatureAlgorithm: No aplica para @Firma 5.0 - Extensión.
- signatureFormat: No aplica para @Firma 5.0 - Extensión.
- showErrors: No aplica para @Firma 5.0 - Extensión.

Este fichero javascript se encarga de cargar los Applets necesarios para el proceso de firma en cliente, este proceso de carga podemos dividirlo en dos pasos:

1. En primer lugar se carga el Applet Instalador en la página Web, este componente es el encargado de instalar en el cliente los componentes necesarios para el funcionamiento del Applet Cliente.

    a. En el caso de no haber sido instalado nunca pide la confirmación del usuario.


b. En el caso de tener una versión distinta a la del usuario, se pide confirmación para actualizarla.


c. Sobrescribe los ficheros en el directorio del Cliente con los ficheros actualizado

d. Solicita permiso para sobrescribir las librerías (no es necesario). En Linux, puede ser necesario ejecutar el instalador como *root* (si no se tienen permisos de escritura sobre directorios en el LD_LIBRARY_PATH).


e. Se finaliza la instalación. Al finalizar la instalación puede ser necesario reiniciar el navegador.


2. Tras la carga del instalador se procede a la carga del Applet Cliente de firma en el document denominado "SignApplet", de esta forma pone a disposición un método denominado "Firma()" que permite firmar datos.

Se debe tener en cuenta que la carga de los applets es asíncrona y hay que esperar a que finalice para utilizar la funcionalidad de firma.

A continuación debemos desarrollar un código javascript que realice una llamada al método "Firma()" de dicho Applet cuando se quiera firmar. Este método devuelve una cadena con los datos firmados, cadena vacía cuando el usuario cancela la firma o "null" cuando ocurrió un error inesperado. Aquí se muestra un ejemplo:

```
<script language=JavaScript>

function clickboton ()

{

        // LLAMADA A METODO FIRMA DEL APPLET

        var a = document.SignApplet.Firma("datosafirmar");

        if(a == null)          // ERROR FIRMANDO DATOS

                alert("No se ha podido firmar");

        else if(a == "")       // FIRMA CANCELADA POR USUARIO

                alert("Firma cancelada");

        else                   // FIRMA CORRECTA. PASAMOS A SIGUIENTE ACCION …….

        {                      // EJEMPLO: SUBMIT DEL FORMULARIO

                document.formulario.firmagenerada.value = a;

                document.formulario.submit();

        }

}

</script>
```

El valor que debe recibir el método Firma del objeto SignApplet es simplemente el resultado de llamar al método generarDatosAFirma() de la interfaz FirmaWebMCA.

Los datos generados por el applet de firma serán el parámetro de entrada del método generarFirma() de la interfaz FirmaWebMCA.

Para ver un ejemplo de una aplicación de firma de ficheros por usuario véase el ejemplo del directorio "ejemplos/firma ficheros/Ejemplos * / demoMultifirma" del CD Desarrollo. Aquí se demuestra la utilización del Applet Cliente de Firma.

# 7 Proceso Firma/MultiFirma de Ficheros por Servidor

El proceso de Firma/MultiFirma de Ficheros por Servidor es implementado por la interfaz com.telventi.firmaweb.FirmaWebMCA. Esta interfaz contiene métodos para realizar un proceso de firma/multifirma de servidor y algunos métodos frecuentes para realizar consultas.

El proceso de Firma de Ficheros por Servidor se realiza en un solo paso. Mediante la interfaz se registra un documento en la plataforma firmándose con un Certificado Digital ubicado en el Servidor de Firma y configurado para ello mediante el módulo de Administración de la extensión de la plataforma (método *generarFirmaServidor*). La Multifirma del Fichero por Servidor se realiza sobre una Firma de Servidor ya realizada, esta vez no hay registro de documento (métodos *coSignServidor* y *counterSignServidor*).

La siguiente figura muestra un proceso de firma/multifirma de servidor:

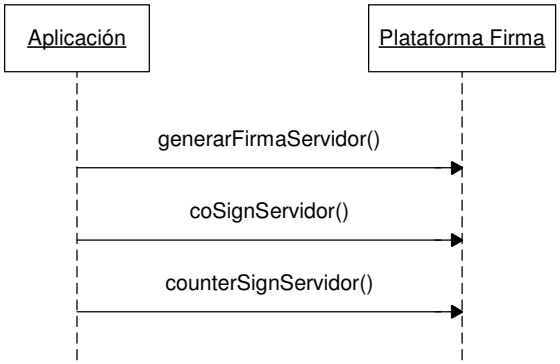


Figura 7 –Firma / Multifirma de Ficheros por servidor. Diagrama interacción

Los métodos de la interfaz que permiten realizar algunas consultas frecuentes y las llamadas entre la Aplicación y extensión de la plataforma de Firma se muestran en la siguiente figura.

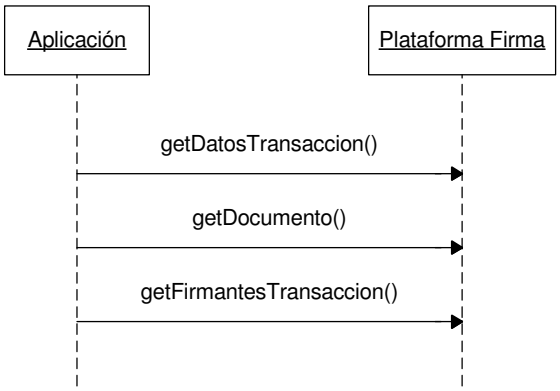


Figura 8 – Interfaz de Custodia. Diagrama interacción

Para obtener información detallada sobre los métodos de la interfaz consultar el "javadoc" proporcionado en el directorio "documentacion/javadoc/firmaficheros" del CD Desarrollo.

## 7.1 Acceso mediante RMI-IIOP

Para utilizar la interfaz RMI-IIOP FirmaWebMCA de la extensión de la plataforma se requieren clientes con máquina virtual de JAVA JDK 1.4 o superior.

En el "CD Desarrollo" se proporcionan las librerías necesarias para acceder a la interfaz. Se encuentran ubicadas en el directorio "ejemplos/firma ficheros/apiRMI-IIOPCliente". El contenido de dicho directorio es:

- lib/ApiFirmaCliente.jar : Clases e Interfaces de acceso a PKI.
- lib/jbossall-client.jar : Librerías cliente acceso JBOSS.
- conf/auth.conf : Fichero de para configuración de acceso a interfaz mediante JAAS. Este fichero debe copiarse en el directorio desde el cual se ejecuta la aplicación.
- conf/jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno CLASSPATH.

Las clases necesarias en este caso son las siguientes:

- com.telventi.utilidades.ConexionFirma
- com.telventi.firmaweb.DTOFirmante

- com.telventi.firmaweb.FirmaException
- com.telventi.firmaweb.FirmaWebMCA

A continuación se muestra un pequeño extracto de código JAVA que obtiene una referencia a la interfaz para poder utilizar sus métodos.

```
// IMPORTAMOS LAS CLASES NECESARIAS EN LA CABECERA
import com.telventi.firmaweb.*;
import com.telventi.utilidades.ConexionFirma;
        ……
try {
    // ESTABLECEMOS PROPIEDADES DE OBJETO ConexionFirma (SERVIDOR, USUARIO, PASSWORD)
    String host = "192.168.53.18";
    String usuario = "user01";
    String password = "12345";
    ConexionFirma.setParametrosConexion(host,usuario,password);

    // OBTENEMOS UNA REFERENCIA A LA INTERFAZ A PARTIR DE LA CONEXION
    FirmaWebMCA servidor =  ConexionFirma.getInstance().getFirmaFicheros();
    // AHORA PODEMOS UTILIZAR CUALQUIERA DE SUS METODOS
    double id = servidor. generarFirmaServidor (……..)
}catch(java.lang.Exception ex)  {}
```

El fichero auth.conf debe copiarse en el directorio desde el cual se ejecuta la aplicación, o bien, si se desea ubicar en un sitio diferente utilizar el método "setFicheroAuth()" de la clase ConexionFirma para indicarle la nueva ubicación.

```
ConexionFirma.setFicheroAuth("C:/auth.conf");
```

## 7.2 Acceso mediante WEBSERVICES

Para realizar el acceso mediante Web Services es necesario crea un cliente de conexión que sea capaz de realizar peticiones al núcleo de la plataforma.

Estos clientes pueden ser generados de forma automática mediante los Ficheros de Descripción de los Servicios Web (WSDL) publicados en el núcleo de la plataforma para la extensión, en la siguiente URL:

https://<servidor_firma>:<puerto>/axis/servlet/AxisServlet

(Será necesario introducir el usuario/password para Web Services. Consultar con algún administrador de la plataforma.)

Existen una serie de herramientas que facilitan este trabajo, entre ellas se citan las siguientes:

- Paquete AXIS JAVA: La clase "org.apache.axis.wsdl.WSDL2Java", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología JAVA.
- Paquete AXIS C/C++: La clase "org.apache.axis.wsdl.wsdl2ws.WSDL2Ws", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología C/C++.
- GSoap. Permite generar clases cliente C/C++ a partir de un fichero descriptor WSDL.
- Etc.

A los clientes generados por las herramientas anteriores posiblemente será necesario añadir el código necesario para realizar la comunicación SSL y la autenticación JAAS con la plataforma de firma. En los ejemplos proporcionados con la plataforma (JAVA) se muestran claramente los mecanismos adicionales incorporados.

Como ayuda adicional puede consultarse el "Javadoc" proporcionado en el directorio "documentacion/javadoc/firmaficheros" del CD Desarrollo y los ejemplos desarrollados en JAVA (directorio "ejemplos/firma ficheros/Ejemplos WebServices" del CD Desarrollo)

Concretamente, para obtener el fichero descriptor WSDL correspondiente a la interfaz FirmaWebMCA conectarse a la siguiente URL:

https://<servidor_firma>:<puerto>/axis/services/FirmaFicheros?wsdl

En el menú del navegador *Archivo*, seleccionar *Guardar como...* indicando el nombre firmaficheros.wsdl.

# 8 Proceso Firma/MultiFirma de Ficheros en Bloque por Usuario

En un proceso de firma tradicional a cada documento firmado le corresponde su propia firma. Esta firma, cuando es llevada a cabo por un usuario, requiere por parte de éste, la selección de un certificado digital para ser empleado en el proceso de firma, además de la introducción del password de desbloqueo del mismo. Cuando se procede a realizar una firma masiva de documentos el usuario tendría que realizar los pasos descritos anteriormente para cada uno de los documentos a firmar. Este hecho hace a priori inviable la firma en procesos que requieran de la firma de un gran número de documentos por parte de un usuario.

Para solventar este problema se propone el siguiente mecanismo para realizar firmas masivas de documentos, sin perder ninguna de las características del proceso de firma individual de documentos, es decir, la verificación de la firma efectuada sobre un documento en concreto que fue firmado en grupo.

En una primera aproximación parece lógico pensar que la firma de un grupo de documentos de una sola vez debería contener cada uno de los documentos originales y firmar todo el conjunto resultante. Este proceso realizado así es poco eficiente ya que el tamaño del documento resultante (la suma de cada documento que se firma en grupo) podría hacer inviable la transmisión del documento de firma. En este caso el proceso de verificación de la firma de un documento firmado dentro de un bloque de documentos requeriría por un lado la verificación de la firma realizada por el usuario de todo el bloque de datos y por otro la comprobación de que realmente el documento que se quiere verificar está incluido dentro de los datos que fueron firmados dentro del bloque. Este sistema llevado a cabo de esta forma es inviable debido a la gran cantidad de información que interviene y presenta numerosos problemas de eficiencia y sobre todo de confidencialidad, ya que existe un documento de firma que incluye un gran número de documentos que son enviados a un usuario como parte de la firma.

Para mejorar la eficiencia del sistema y eliminar los problemas de confidencialidad se debe de cambiar la información referente a cada archivo que es incluida en el bloque de firma. Esa información debe cumplir una serie de requisitos, como que no contengan datos confidenciales del documento original y que garanticen que los datos hacen referencia unívoca al documento del cual se han generado. Esa información a incluir puede ser el propio hash del documento o la firma del mismo por una entidad de confianza (plataforma de Firma con Certificado Servidor). Tanto la firma como el hash son soluciones válidas para el problema en el que nos encontramos. Emplear el hash del documento es la opción que menos recursos de espacio consume, y emplear la firma proporciona un elemento más de confianza, la propia firma del documento por la extensión de la plataforma de Firma.

El empleo de la firma simplifica enormemente el diseño del sistema. Por ello se opta por seguir la solución de la firma de los documentos, ya que esta solución aunque conlleva un coste mayor en espacio simplifica enormemente los procesos de firma, custodia, verificación y visualización.

Bajo la solución elegida, la firma en bloque requerirá una preparación previa de los documentos que son firmados dentro del bloque. Esta preparación consistiría en la firma de estos de forma individual y automática por parte de la plataforma de Firma. El bloque de documentos a firmar por el usuario estaría formado, no por los documentos, sino por las firmas de éstos identificadas por un identificador único para facilitar su localización, en este caso, el identificador sería el identificador de la sesión de firma llevada a cabo por la plataforma de Firma sobre cada documento individual. El conjunto de las firmas de los documentos del bloque serían los datos

que firmaría el usuario. Por comodidad el fichero resultante de la firma incluiría estos datos y su formato sería el de un documento en formato estándar ASN1.

El proceso de verificación de un documento firmado en bloque consistiría en primer lugar en la verificación del fichero de firma generado en el proceso de firma en bloques. Con ello se sabe que el usuario ha firmado ese bloque de firmas. Para saber si el usuario firmó el documento individual se deberá recuperar dentro de los datos del fichero de firma la firma individual del fichero, haciendo uso para ello del identificador comentado anteriormente. Una vez obtenido la firma individual (firma de la plataforma de Firma) se comprobaría que esta firma se verifica con el documento individual que se desea verificar. Si la firma se satisface se puede determinar que el documento fue firmado por el usuario que firmó el bloque, al estar su firma individual incluida dentro del bloque de datos firmados. Además con la firma del documento realizada por la plataforma de Firma el usuario tiene la garantía de que los datos que firmó realmente pasaron por la plataforma de Firma y que no fueron modificados durante su proceso de firma.

El proceso de generación de firma se puede resumir en el siguiente diagrama:

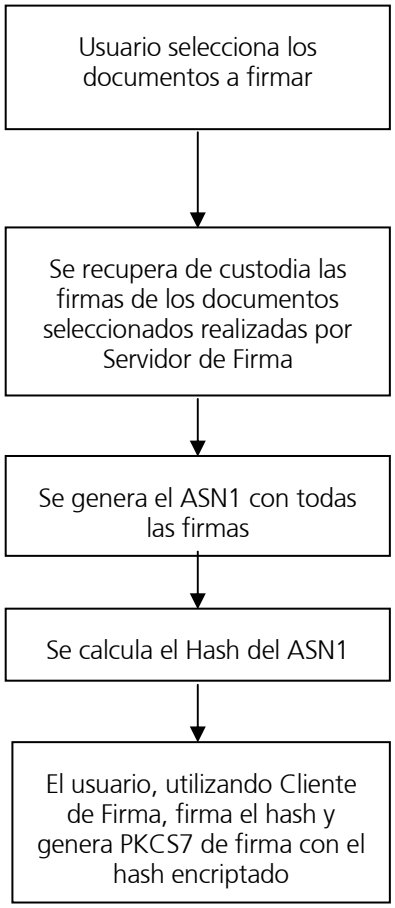


Figura 9 – Proceso de generación de firma en bloque

De forma similar al caso anterior el proceso de verificación de firma en bloques se puede resumir en el siguiente diagrama:

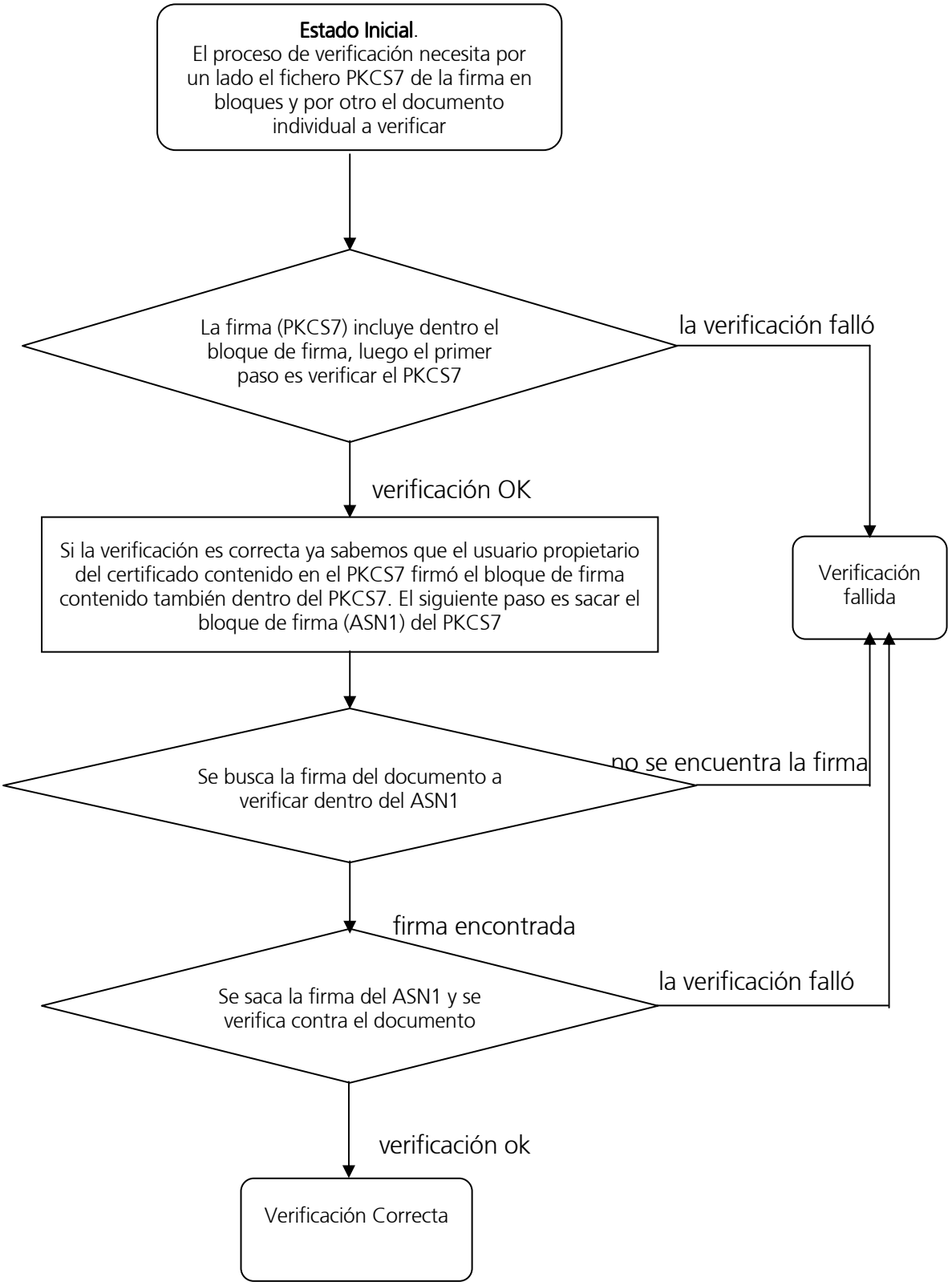


**Figura 10 – Proceso de verificación de firma en bloque**

El proceso de Firma/MultiFirma de Ficheros en Bloque es implementado por la interfaz com.telventi.firmaenbloquemaca.FirmaEnBloqueMCAFacade. Esta interfaz contiene métodos para realizar un proceso de firma en bloque y métodos para realizar consultas específicas de bloques.

En el proceso de firma en bloque intervienen tres agentes: **usuario** (cliente), **aplicación** que utiliza la interfaz y la **plataforma de Firma** (interfaz de Firma).

1) La **aplicación** registra los documentos a firmar en la **plataforma de Firma** (método *registrarDocumento*). Cada documento es firmado por el Servidor de Firma con un Certificado Digital de Servidor configurado para ello mediante la herramienta de Administración. Como resultado se obtiene un identificador para cada documento registrado.

2) La **aplicación** solicita a la **plataforma de Firma** la preparación del Bloque a firmar, que será enviado a la máquina del **usuario** para ser firmado (método *iniciarFirma*).

3) El **usuario** firma los datos y la **aplicación** envía el resultado a la **plataforma de Firma** para terminar el proceso. (método *finalizarFirma*)

La siguiente figura muestra el proceso descrito:

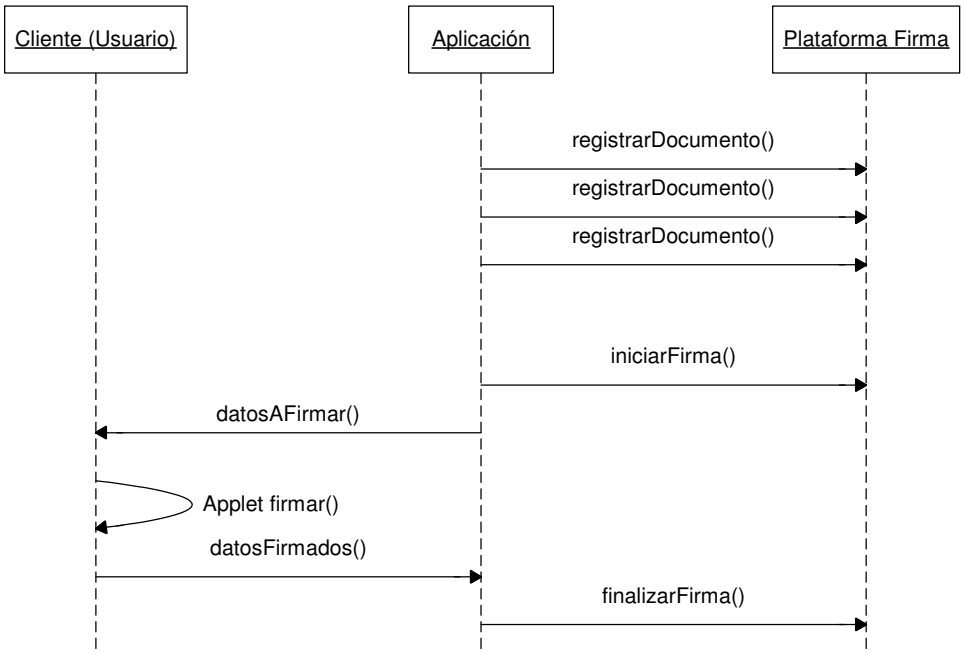


**Figura 11 – Firma en bloque. Diagrama interacción**

La interfaz también proporciona unos métodos adicionales para realizar consultas específicas sobre firmas en bloque. La siguiente figura muestra las llamadas entre la Aplicación y la plataforma de firma.

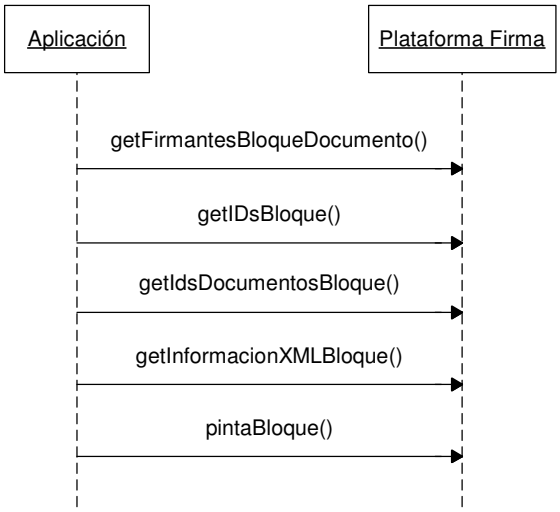


Figura 12 – Firma en bloque, información sobre transacciones. Diagrama interacción

Para obtener información detallada sobre los métodos de la interfaz consultar el "javadoc" proporcionado en el directorio "documentacion/javadoc/firmaficheros" del CD Desarrollo.

## 8.1 Acceso mediante RMI-IIOP

Para utilizar la interfaz RMI-IIOP FirmaEnBloqueMCAFacade de la extensión de la plataforma se requieren clientes con máquina virtual de JAVA JDK 1.4 o superior.

En el "CD Desarrollo" se proporcionan las librerías necesarias para acceder a la interfaz. Se encuentran ubicadas en el directorio "ejemplos/firma ficheros/apiRMI-IIOPCliente". El contenido de dicho directorio es:

- lib/ApiFirmaCliente.jar : Clases e Interfaces de acceso a PKI.
- lib/jbossall-client.jar : Librerías cliente acceso JBOSS.
- conf/auth.conf : Fichero de para configuración de acceso a interfaz mediante JAAS. Este fichero debe copiarse en el directorio desde el cual se ejecuta la aplicación.

- conf/jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno CLASSPATH.

Las clases necesarias en este caso son las siguientes:

- com.telventi.utilidades.ConexionFirma
- com.telventi.firmaenbloquemca.DTOFirmante
- com.telventi.firmaenbloquemca.DTOIDs
- com.telventi.firmaenbloquemca.TDOIDs
- com.telventi.firmaenbloquemca.TDOIniciarFirma
- com.telventi.firmaenbloquemca.FirmaEnBloqueException
- com.telventi.firmaenbloquemca.FirmaEnBloqueMCAFacade

A continuación se muestra un pequeño extracto de código JAVA que obtiene una referencia a la interfaz para poder utilizar sus métodos.

```
// IMPORTAMOS LAS CLASES NECESARIAS EN LA CABECERA

import com.telventi.firmaenbloquemca.*;

import com.telventi.utilidades.ConexionFirma;

        ……

try {

    // ESTABLECEMOS PROPIEDADES DE OBJETO ConexionFirma (SERVIDOR, USUARIO, PASSWORD)

    String host = "192.168.53.18";

    String usuario = "user01";

    String password = "12345";

    ConexionFirma.setParametrosConexion(host,usuario,password);


    // OBTENEMOS UNA REFERENCIA A LA INTERFAZ A PARTIR DE LA CONEXION

    FirmaEnBloqueMCAFacade bloques =  ConexionFirma.getInstance().getFirmaEnBloque();

    // AHORA PODEMOS UTILIZAR CUALQUIERA DE SUS METODOS

    double id = bloques. registrarDocumento (……..)
```

```
}catch(java.lang.Exception ex) {}
```

El fichero auth.conf debe copiarse en el directorio desde el cual se ejecuta la aplicación, o bien, si se desea ubicar en un sitio diferente utilizar el método "setFicheroAuth()" de la clase ConexionFirma para indicarle la nueva ubicación.

```
ConexionFirma.setFicheroAuth("C:/auth.conf");
```

## 8.2 Acceso mediante WEBSERVICES

Para realizar el acceso mediante Web Services es necesario crea un cliente de conexión que sea capaz de realizar peticiones al núcleo de la plataforma.

Estos clientes pueden ser generados de forma automática mediante los Ficheros de Descripción de los Servicios Web (WSDL) publicados en el núcleo de la plataforma para la extensión, en la siguiente URL:

https://<servidor_firma>:<puerto>/axis/servlet/AxisServlet

(Será necesario introducir el usuario/password para Web Services. Consultar con algún administrador de la plataforma)

Existen una serie de herramientas que facilitan este trabajo, entre ellas se citan las siguientes:

- Paquete AXIS JAVA: La clase "org.apache.axis.wsdl.WSDL2Java", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología JAVA.
- Paquete AXIS C/C++: La clase "org.apache.axis.wsdl.wsdl2ws.WSDL2Ws", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología C/C++.
- GSoap. Permite generar clases cliente C/C++ a partir de un fichero descriptor WSDL.
- Etc.

A los clientes generados por las herramientas anteriores posiblemente será necesario añadir el código necesario para realizar la comunicación SSL y la autenticación JAAS con la plataforma de firma. En los ejemplos proporcionados con la plataforma (JAVA) se muestran claramente los mecanismos adicionales incorporados.

Como ayuda adicional puede consultarse el "Javadoc" proporcionado en el directorio "documentacion/javadoc/firmaficher" del CD Desarrollo y los ejemplos desarrollados en JAVA (directorio "ejemplos/firma ficheros/Ejemplos WebServices" del CD Desarrollo).

Concretamente, para obtener el fichero descriptor WSDL correspondiente a la interfaz FirmaEnBloqueMCAFacade conectarse a la siguiente URL:

https://<servidor_firma>:<puerto>/axis/services/FirmaBloques?wsdl

En el menú del navegador *Archivo*, seleccionar *Guardar como...* indicando el nombre firmabloques.wsdl.

## 8.3 Utilización del Componente Cliente de Firma (Applet Firma)

En un proceso de firma en bloque de ficheros por usuario también es necesario incorporar un componente que firme los datos en la máquina del usuario. Este componente es el mismo que en el caso de la firma de ficheros, por lo que el proceso de integración y configuración es similar al descrito en el apartado 6.3.

Para utilizar la funcionalidad de firma debemos desarrollar un código javascript que realice una llamada al método "Firma()" de dicho Applet cuando se quiera firmar. Este método devuelve una cadena con los datos firmados, cadena vacía cuando el usuario cancela la firma o "null" cuando ocurrió un error inesperado. Aquí se muestra un ejemplo:

```
<script language=JavaScript>

function clickboton ()

{
        // LLAMADA A METODO FIRMA DEL APPLET
        var a = document.SignApplet.Firma("datosafirmar");
        if(a == null)          // ERROR FIRMANDO DATOS
                alert("No se ha podido firmar");
        else if(a == " ")      // FIRMA CANCELADA POR USUARIO
                alert("Firma cancelada");
        else                   // FIRMA CORRECTA. PASAMOS A SIGUIENTE ACCION …….
        {                      // EJEMPLO: SUBMIT DEL FORMULARIO
                document.formulario.firmagenerada.value = a;
                document.formulario.submit();
        }

}

</script>
```

El valor que debe recibir el método Firma del objeto SignApplet es simplemente el resultado de llamar al método iniciarFirma() de la interfaz FirmaEnBloqueMCAFacade. En concreto, este método devuelve un objeto TDOIniciarFirma, y el valor a pasar al método del applet es el atributo hash de este objeto.

Los datos generados por el applet de firma serán el parámetro de entrada del método finalizarFirma de la interfaz FirmaEnBloqueMCAFacade.

Para ver un ejemplo de una aplicación de firma de ficheros en bloque véase el ejemplo del directorio "ejemplos / firma ficheros/ Ejemplos * / firmaenbloquedemo" del CD Desarrollo. Aquí se demuestra la utilización del Applet Cliente de Firma.

# 9 Consulta de Transacciones

La extensión de la plataforma de firma proporciona una interfaz para recuperar información sobre transacciones de firma de ficheros realizadas. Esta interfaz se denomina com.telventi.custodia.CustodiaDocumentosFacade.

Las transacciones corresponderán a los procesos descritos en los tres apartados anteriores (firma usuario, firma servidor y firma usuario en bloque).

La siguiente figura muestra los métodos de la interfaz y la comunicación entre la Aplicación y la plataforma de firma.

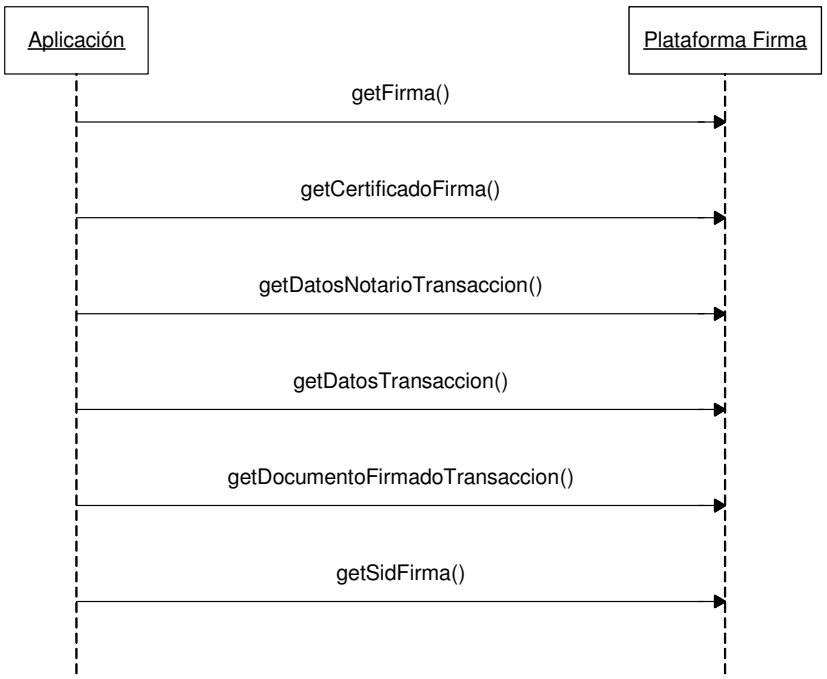


Figura 13 – Interfaz de Custodia. Diagrama interacción

Para obtener información detallada sobre los métodos de la interfaz consultar el "javadoc" proporcionado en el directorio "/documentacion/javadoc/firmaficheros" del CD Desarrollo.

## 9.1 Acceso mediante RMI-IIOP

Para utilizar la interfaz RMI-IIOP CustodiaDocumentosFacade de la extensión de la plataforma se requieren clientes con máquina virtual de JAVA JDK 1.4 o superior.

En el "CD Desarrollo" se proporcionan las librerías necesarias para acceder a la interfaz. Se encuentran ubicadas en el directorio "/ejemplos/firma ficheros/apiRMI-IIOPCliente". El contenido de dicho directorio es:

- lib/ApiFirmaCliente.jar : Clases e Interfaces de acceso a PKI.
- lib/jbossall-client.jar : Librerías cliente acceso JBOSS.
- conf/auth.conf : Fichero de para configuración de acceso a interfaz mediante JAAS. Este fichero debe copiarse en el directorio desde el cual se ejecuta la aplicación.
- conf/jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno CLASSPATH.

Las clases necesarias en este caso son las siguientes:

- com.telventi.utilidades.ConexionFirma
- com.telventi.custodia.CustodiaDocumentosFacade
- com.telventi.custodia.CustodiaException
- com.telventi.utilidades.notario.DTONotario
- com.telventi.custodia.DTODocumentoFirmado

A continuación se muestra un pequeño extracto de código JAVA que obtiene una referencia a la interfaz para poder utilizar sus métodos.

```
// IMPORTAMOS LAS CLASES NECESARIAS EN LA CABECERA

import com.telventi.custodia.*;

import com.telventi.utilidades.ConexionFirma;

import com.telventi.utilidades.notario.*;

        ……

try {

    // ESTABLECEMOS PROPIEDADES DE OBJETO ConexionFirma (SERVIDOR, USUARIO, PASSWORD)

    String host = "192.168.53.18";

    String usuario = "user01";

    String password = "12345";

    ConexionFirma.setParametrosConexion(host,usuario,password);
```

```
// OBTENEMOS UNA REFERENCIA A LA INTERFAZ A PARTIR DE LA CONEXION

CustodiaDocumentosFacade custodia =  ConexionFirma.getInstance().getCustodiaDocumentos();

// AHORA PODEMOS UTILIZAR CUALQUIERA DE SUS METODOS

DTODocumentoFirmado dto = custodia. getDatosTransaccion (……..)

}catch(java.lang.Exception ex)  { }
```

El fichero auth.conf debe copiarse en el directorio desde el cual se ejecuta la aplicación, o bien, si se desea ubicar en un sitio diferente utilizar el método "setFicheroAuth()" de la clase ConexionFirma para indicarle la nueva ubicación.

```
ConexionFirma.setFicheroAuth("C:/auth.conf");
```

## 9.2 Acceso mediante WEBSERVICES

Para realizar el acceso mediante Web Services es necesario crea un cliente de conexión que sea capaz de realizar peticiones al núcleo de la plataforma.

Estos clientes pueden ser generados de forma automática mediante los Ficheros de Descripción de los Servicios Web (WSDL) publicados en el núcleo de la plataforma para la extensión, en la siguiente URL:

https://<servidor_firma>:<puerto>/axis/servlet/AxisServlet

(Será necesario introducir el usuario/password para Web Services. Consultar con algún administrador de la plataforma)

Existen una serie de herramientas que facilitan este trabajo, entre ellas se citan las siguientes:

- Paquete AXIS JAVA: La clase "org.apache.axis.wsdl.WSDL2Java", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología JAVA.
- Paquete AXIS C/C++: La clase "org.apache.axis.wsdl.wsdl2ws.WSDL2Ws", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología C/C++.
- GSoap. Permite generar clases cliente C/C++ a partir de un fichero descriptor WSDL.
- Etc.

A los clientes generados por las herramientas anteriores posiblemente será necesario añadir el código necesario para realizar la comunicación SSL y la autenticación JAAS con la plataforma de

firma. En los ejemplos proporcionados con la plataforma (JAVA) se muestran claramente los mecanismos adicionales incorporados.

Como ayuda adicional puede consultarse el "Javadoc" proporcionado en el directorio "Documentación "/documentacion/javadoc/firmaficheros" del CD Desarrollo y los ejemplos desarrollados en JAVA (directorio "ejemplos/firma ficheros / Ejemplos WebServices" del CD Desarrollo)

Concretamente, para obtener el fichero descriptor WSDL correspondiente a la interfaz CustodiaDocumentosFacade conectarse a la siguiente URL:

https://<servidor_firma>:<puerto>/axis/services/Custodia?wsdl

En el menú del navegador *Archivo*, seleccionar *Guardar como...* indicando el nombre custodia.wsdl.

# 10 Verificación de Firmas

La extensión de la plataforma de firma proporciona una interfaz para realizar la verificación de las firmas realizadas en el módulo de Firma Ficheros. Esta interfaz se denomina com.telventi.verificacionfirmas.VerificarFirmas.

El método *verificarFirma()* permite realizar la verificación de los procesos de Firma/MultiFirma de Usuario y Firma/MultiFirma de Servidor.

Por otro lado, el método *verificarFirmaBloquePKCS7()* permite realizar la verificación del proceso de Firma/MultiFirma en Bloque por Usuario.

La siguiente figura muestra los métodos de la interfaz y la comunicación entre la Aplicación y la plataforma de firma.

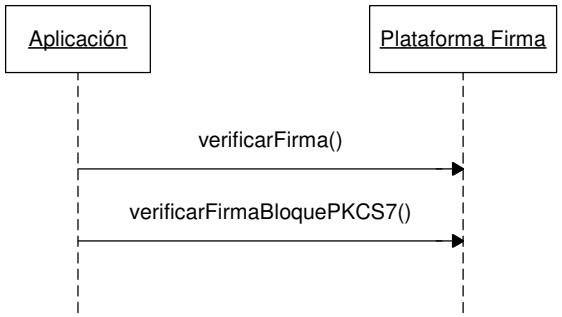


Figura 14 – Verificación de firmas. Diagrama interacción

Para obtener información detallada sobre los métodos de la interfaz consultar el "javadoc" proporcionado en el directorio "/documentacion/javadoc/firmaficheros" del CD Desarrollo.

## 10.1 Acceso mediante RMI-IIOP

Para utilizar la interfaz RMI-IIOP VerificarFirmas se requieren clientes con máquina virtual de JAVA JDK 1.4 o superior.

En el "CD Desarrollo" se proporcionan las librerías necesarias para acceder a la interfaz. Se encuentran ubicadas en el directorio "/ejemplos/firma ficheros/apiRMI-IIOPCliente". El contenido de dicho directorio es:

- lib/ApiFirmaCliente.jar : Clases e Interfaces de acceso a PKI.
- lib/jbossall-client.jar : Librerías cliente acceso JBOSS.
- conf/auth.conf : Fichero de para configuración de acceso a interfaz mediante JAAS. Este fichero debe copiarse en el directorio desde el cual se ejecuta la aplicación.
- conf/jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno CLASSPATH.

Las clases necesarias en este caso son las siguientes:

- com.telventi.utilidades.ConexionFirma
- com.telventi.verificacionfirmas.VerificarFirmas
- com.telventi.verificacionfirmas.DTOVerificacionFirma
- com.telventi.verificacionfirmas.DTOVerificacionFirmante

A continuación se muestra un pequeño extracto de código JAVA que obtiene una referencia a la interfaz para poder utilizar sus métodos.

```
// IMPORTAMOS LAS CLASES NECESARIAS EN LA CABECERA

import com.telventi.verificacionfirmas.*;

import com.telventi.utilidades.ConexionFirma;      ……

                ……

try {

        // ESTABLECEMOS PROPIEDADES DE OBJETO ConexionFirma (SERVIDOR, USUARIO, PASSWORD)

        String host = "192.168.53.18";

        String usuario = "user01";

        String password = "12345";

        ConexionFirma.setParametrosConexion(host,usuario,password);


        // OBTENEMOS UNA REFERENCIA A LA INTERFAZ A PARTIR DE LA CONEXION

        VerificarFirmas verif =  ConexionFirma.getInstance().getVerificarFirmas();

        // AHORA PODEMOS UTILIZAR CUALQUIERA DE SUS METODOS
```

```
        DTOVerificacionFirma dto = verif.verificarFirma(……..)

}catch(java.lang.Exception ex) {}
```

El fichero auth.conf debe copiarse en el directorio desde el cual se ejecuta la aplicación, o bien, si se desea ubicar en un sitio diferente utilizar el método "setFicheroAuth()" de la clase ConexionFirma para indicarle la nueva ubicación.

```
ConexionFirma.setFicheroAuth("C:/auth.conf");
```

## 10.2 Acceso mediante WEBSERVICES

Para realizar el acceso mediante Web Services es necesario crea un cliente de conexión que sea capaz de realizar peticiones al núcleo de la plataforma.

Estos clientes pueden ser generados de forma automática mediante los Ficheros de Descripción de los Servicios Web (WSDL) publicados en el núcleo de la plataforma para la extensión, en la siguiente URL:

https://<servidor_firma>:<puerto>/axis/servlet/AxisServlet

(Será necesario introducir el usuario/password para Web Services. Consultar con algún administrador de la plataforma)

Existen una serie de herramientas que facilitan este trabajo, entre ellas se citan las siguientes:

- Paquete AXIS JAVA: La clase "org.apache.axis.wsdl.WSDL2Java", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología JAVA.
- Paquete AXIS C/C++: La clase "org.apache.axis.wsdl.wsdl2ws.WSDL2Ws", dado un fichero descriptor WSDL permite generar las clases cliente en tecnología C/C++.
- GSoap. Permite generar clases cliente C/C++ a partir de un fichero descriptor WSDL.
- Etc.

A los clientes generados por las herramientas anteriores posiblemente será necesario añadir el código necesario para realizar la comunicación SSL y la autenticación JAAS con la plataforma de firma. En los ejemplos proporcionados con la plataforma (JAVA) se muestran claramente los mecanismos adicionales incorporados.

Como ayuda adicional puede consultarse el "Javadoc" proporcionado en el directorio "/documentacion/javadoc/firmaficheros" del CD Desarrollo y los ejemplos desarrollados en JAVA (directorio "ejemplos/firma ficheros / Ejemplos WebServices" del CD Desarrollo)

Concretamente, para obtener el fichero descriptor WSDL correspondiente a la interfaz VerificarFimas conectarse a la siguiente URL:

https://<servidor_firma>:<puerto>/axis/services/VerificarFirmas?wsdl

En el menú del navegador *Archivo*, seleccionar *Guardar como...* indicando el nombre verificarFirmas.wsdl.

# 11 Aplicaciones de Ejemplo Firma Ficheros de la plataforma

## 11.1 Aplicación "demoMultifirma"

La aplicación "demoMultifirma" muestra ejemplos de los procesos de Firma/Multifirma de Ficheros por Usuario y Servidor.

Se hace uso de las siguientes interfaces de la extensión de la plataforma:

- com.telventi.firmaweb.FirmaWebMCA: para realizar la firma/multifirma de ficheros usuario y servidor.
- com.telventi.verificacionfirmas.VerificarFirmas: para realizar la verificación de las firmas realizadas.

La aplicación está desarrollada en JAVA – JSP y se encuentra disponible utilizando las dos tecnologías de la plataforma: RMI-IIOP y WebServices.

El código de la aplicación se encuentra disponible en los siguientes directorios del CD Desarrollo, respectivamente:

- "ejemplos / firma ficheros / Ejemplos RMI-IIOP / Codigo / demoMultifirma"
- "ejemplos / firma ficheros / Ejemplos WebServices / Codigo / demoMultifirmaWS"

### 11.1.1 Poner en marcha la aplicación RMI-IIOP

Es una aplicación WEB, así pues, necesitamos un contenedor de servlets (Tomcat), servidor Web con un contenedor de servlets (Apache + Tomcat) o un servidor de aplicaciones, por ejemplo TOMCAT.

En el directorio "ejemplos / firma ficheros / Ejemplos RMI-IIOP / demoMultifirma" del CD Desarrollo se encuentran los ficheros necesarios para poner en marcha la aplicación:

- demoMultifirma: aplicación que debe ser desplegada en el directorio de deploy del servidor de aplicaciones. (ej: $CATALINA_HOME/webapps)
- demoMultifirma.properties: parámetros para poder ejecutar la aplicación. Este fichero debe colocarse en el directorio de ejecución del servidor (ej: $CATALINA_HOME/bin). Los parámetros a configurar son los siguientes:
    - servidorfirma=<nombre o ip del servidor de firma de la plataforma de firma>
    - usuario=<usuario para acceso JAAS RMI-IIOP a servidor de firma>
    - password=<password para acceso JAAS RMI-IIOP a servidor de firma>
    - idaplicacion=<identificador de aplicación registrada en el módulo de administración de la extensión de la plataforma>Ej: DEMOMULTIFIRMA

- ficherofirma=<ruta completa del fichero a firmar en la demo>Ej: c:/demo.pdf

- auth.conf: Fichero de configuración para autenticación JAAS. Debe ser colocado en el directorio de ejecución del servidor de aplicaciones (ej: $CATALINA_HOME/bin).

- jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno **CLASSPATH** (ej: $CATALINA_HOME/conf).

Las librerías necesarias para que funcione la demo se encuentran en el directorio "ejemplos / firma ficheros /apiRMI-IIOPCliente /lib", que deben ser colocadas en el directorio de librerías del servidor de aplicaciones (ej: $CATALINA_HOME/server/lib).

Una vez finalizados los pasos anteriores, se accederá mediane la url:

https://<servidor_aplicaciones>:<puerto>/multifirma/multifirmas.jsp

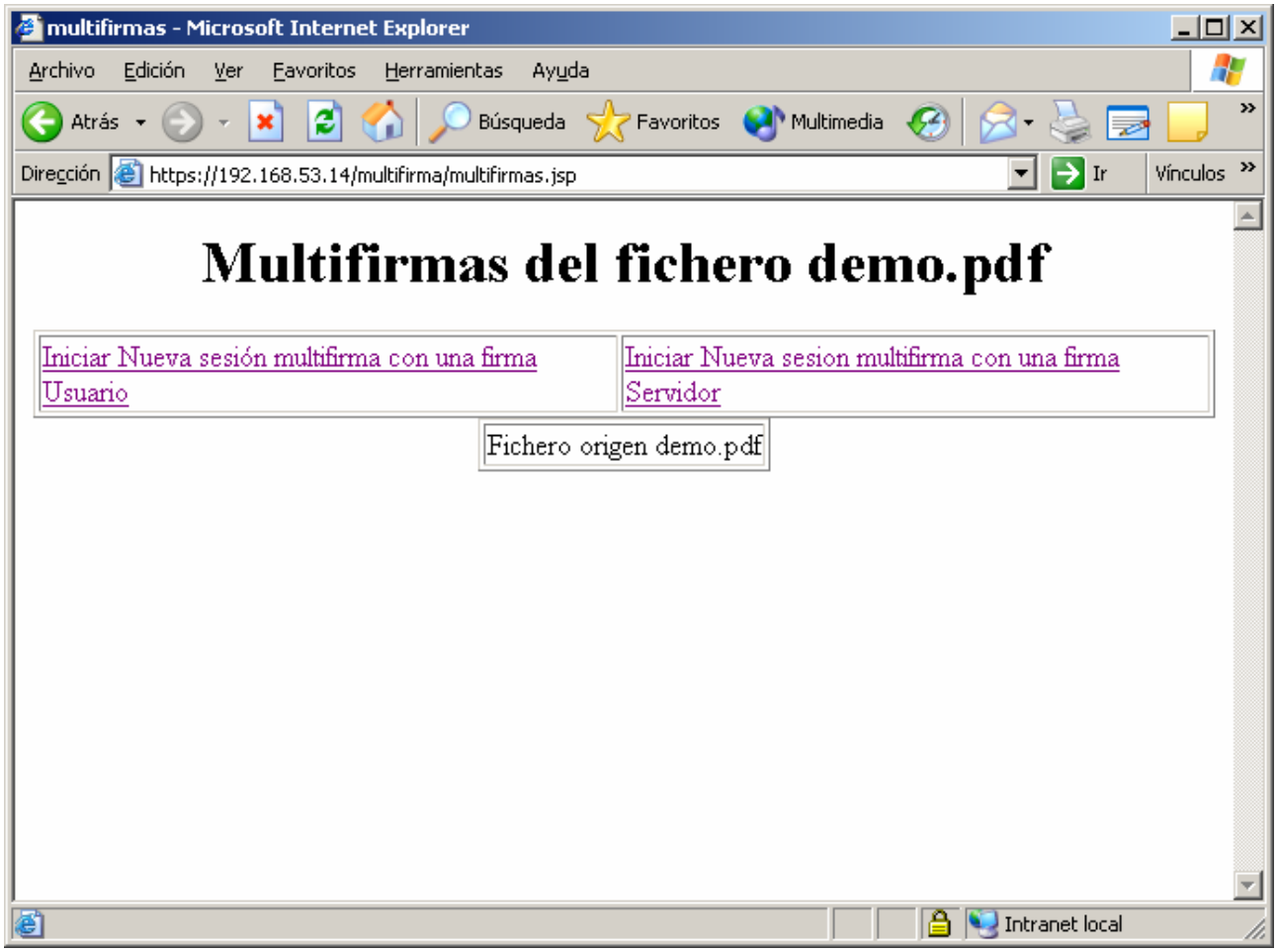


Figura 15 – Aplicación de ejemplo de firma /multifirma de ficheros. Acceso RMI-IIOP

### 11.1.2 Poner en marcha la aplicación WebServices

Es una aplicación WEB, así pues, necesitamos un contenedor de servlets (Tomcat), servidor Web con un contenedor de servlets (Apache + Tomcat) o un servidor de aplicaciones, por ejemplo JBOSS.

En el directorio "/ejemplos /firma ficheros /Ejemplos WebServices/demoMultifirmaWS" del CD Desarrollo se encuentran los ficheros necesarios para poner en marcha la aplicación:

- demoMultifirmaWS: aplicación que debe ser desplegada en el directorio de deploy del servidor de aplicaciones. (ej: $CATALINA_HOME/webapps)

- demoWS.properties: parámetros para poder ejecutar la aplicación. Este fichero debe colocarse en el directorio de ejecución del servidor (ej: $CATALINA_HOME/bin). Los parámetros a configurar son los siguientes:

    - servidorfirma==<nombre o ip del servidor de firma de la plataforma de firma>.

    - usuario=<usuario para acceso JAAS WebServices a servidor de firma>.

    - password=<password para acceso JAAS WebServices a servidor de firma>.

    - trustedstore=<url del fichero trustkeystore que contiene la clave pública del certificado para comunicaciones SSL usado por el servidor de firma de la plataforma de firma>.Ej:c:\trustkeystore

    - trustedstorepassword=<password el keystore anterior>.

    - idaplicacion=<identificador de aplicación registrada en el módulo de administración de la extensión de la plataforma>Ej: DEMOMULTIFIRMAWS

    - ficherofirma=<ruta completa del fichero a firmar en la demo>Ej: c:/demo.pdf

- trustKeystore: keystore que contiene la clave pública del certificado digital del servidor de firma de la plataforma de firma. Se utiliza para el acceso mediante SSL al servidor de firma.

Una vez finalizados los pasos anteriores, se accederá mediane la url:

https://<servidor_aplicaciones>:<puerto>/multifirmaWS/multifirmasWS.jsp

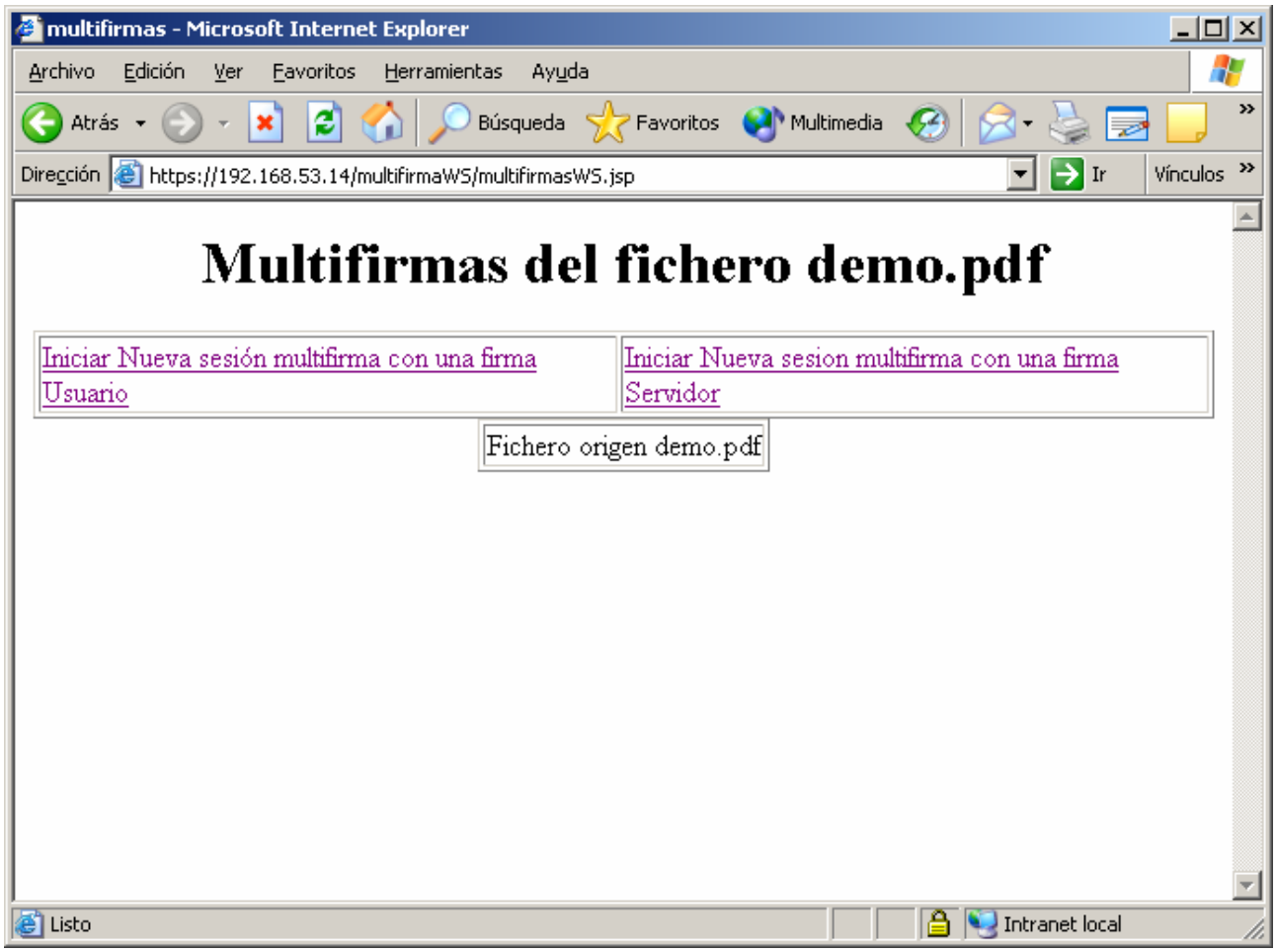


Figura 16 – Aplicación de ejemplo de firma /multifirma de ficheros. Acceso Web Services

## 11.2 Aplicación "firmaenbloquedemo"

La aplicación "firmaenbloquedemo" muestra ejemplos del proceso de Firma/Multifirma de Ficheros en Bloque por Usuario.

Se hace uso de las siguientes interfaces de la extensión de la plataforma:

- com.telventi.firmaenbloquemca.FirmaEnBloqueMCAFacade: para realizar la firma/multifirma de ficheros en bloque por usuario.
- com.telventi.verificacionfirmas.VerificarFirmas: para realizar la verificación de las firmas realizadas.
- com.telventi.custodia.CustodiaDocumentosFacade: para recuperar información sobre una transacción realizada.

La aplicación está desarrollada en JAVA – JSP y se encuentra disponible utilizando las dos tecnologías de la plataforma: RMI-IIOP y WebServices.

El código de la aplicación se encuentra disponible en los siguientes directorios del CD Desarrollo, respectivamente:

- "ejemplos/firma ficheros /Ejemplos RMI-IIOP/Codigo/firmaenbloquedemo"
- "ejemplos/firma ficheros //Ejemplos WebServices/Codigo/firmaenbloquedemoWS"

### 11.2.1 Poner en marcha la aplicación RMI-IIOP

Es una aplicación WEB, así pues, necesitamos un contenedor de servlets (Tomcat), servidor Web con un contenedor de servlets (Apache + Tomcat) o un servidor de aplicaciones, por ejemplo TOMCAT.

En el directorio "ejemplos/firma ficheros / Ejemplos RMI-IIOP / firmaenbloquedemo" del CD Desarrollo se encuentran los ficheros necesarios para poner en marcha la aplicación:

- firmaenbloquedemo: aplicación que debe ser desplegada en el directorio de deploy del servidor de aplicaciones. (ej: CATALINA_HOME/webapps)
- demobloques.properties: parámetros para poder ejecutar la aplicación. Este fichero debe colocarse en el directorio de ejecución del servidor (e: CATALINA_HOME/bin). Los parámetros a configurar son los siguientes:
    - servidorfirma=<nombre o ip del servidor de firma de la plataforma de firma>
    - usuario=<usuario para acceso JAAS RMI-IIOP a servidor de firma>
    - password=<password para acceso JAAS RMI-IIOP a servidor de firma>
    - idaplicacion=<identificador de aplicación registrada en módulo de administración de la extensión de la plataforma>Ej: DEMOBLOQUES
- auth.conf: Fichero de configuración para autenticación JAAS. Debe ser colocado en el directorio de ejecución del servidor de aplicaciones (ej:JBOSS_HOME/bin).
- jndiMigration.properties: Fichero de propiedades que contiene los parámetros del contexto JNDI de conexión con el servidor @Firma. Este fichero, o el directorio que lo aloja, debe estar incluido en la variable de entorno CLASSPATH (ej: $CATALINA_HOME/conf).

Las librerías necesarias para que funcione la demo se encuentran en el directorio "ejemplos/firma ficheros/apiRMI-IIOPCliente/lib", que deben ser colocadas en el directorio de librerías del servidor de aplicaciones (ej: $CATALINA_HOME/server/lib).

Una vez finalizados los pasos anteriores, se accederá mediane la url:

https://<servidor_aplicaciones>:<puerto>/firmaenbloque/index.jsp

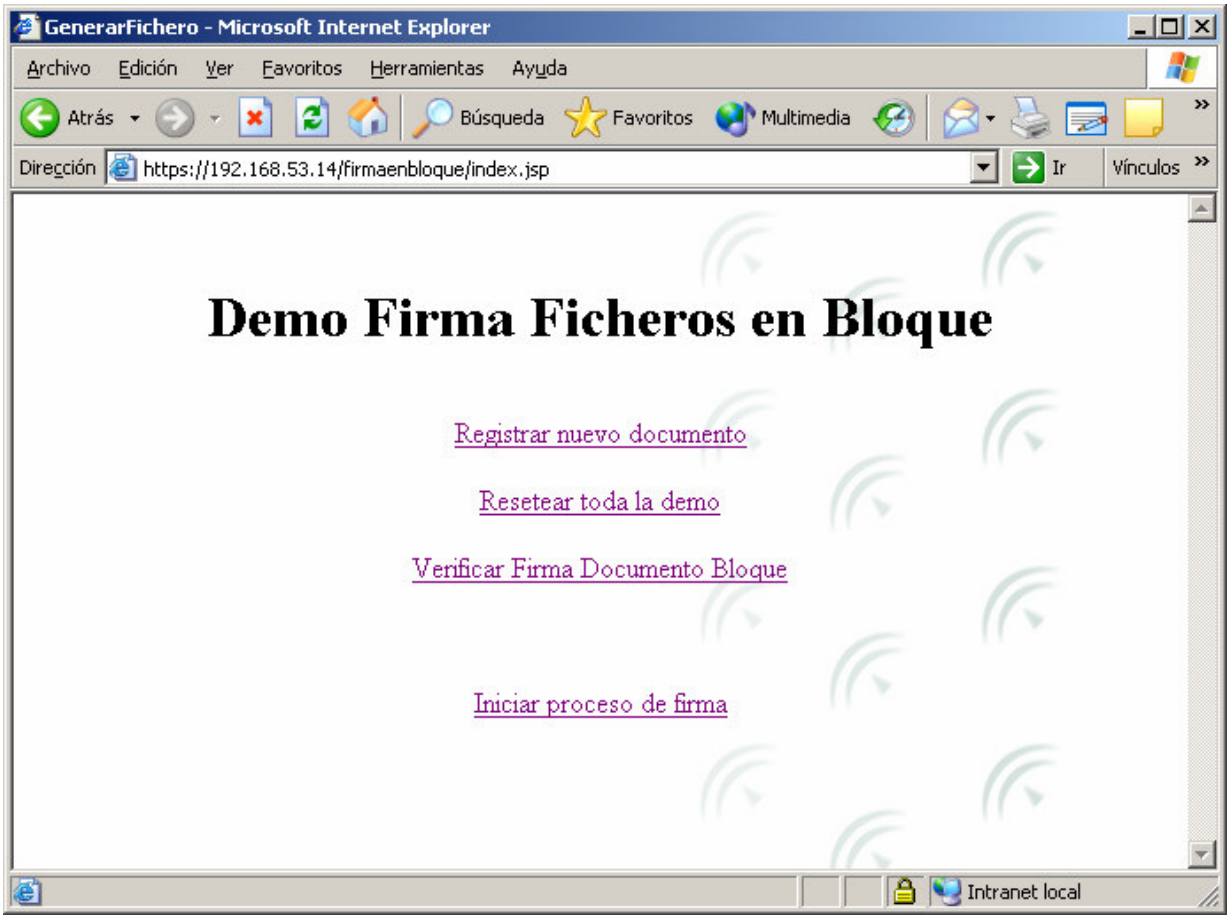


Figura 17 – Aplicación de ejemplo de firma /multifirma de bloques. Acceso RMI-IIOP

### 11.2.2 Poner en marcha la aplicación WebServices

Es una aplicación WEB, así pues necesitamos un servidor de aplicaciones, por ejemplo TOMCAT.

En el directorio "ejemplos/firma ficheros/ Ejemplos WebServices / firmaenbloquedemoWS" del CD Desarrollo se encuentran los ficheros necesarios para poner en marcha la aplicación:

- firmaenbloquedemoWS: aplicación que debe ser desplegada en el directorio de deploy del servidor de aplicaciones. (ej: CATALINA_HOME/webapps)
- demobloquesWS.properties: parámetros para poder ejecutar la aplicación. Este fichero debe colocarse en el directorio de ejecución del servidor (e: CATALINA_HOME/bin). Los parámetros a configurar son los siguientes:
  - servidorfirma==<nombre o ip del servidor de firma de la plataforma de firma>
  - usuario=<usuario para acceso JAAS WebServices a servidor de firma>

- password=<password para acceso JAAS WebServices a servidor de firma>
- trustedstore=<url del fichero trustkeystore que contiene el certificado digital SSL del servidor de firma de la plataforma de firma>Ej:c:\trustkeystore
- trustedstorepassword=<password el keystore anterior>
- idaplicacion=<identificador de aplicación registrada en la herramienta de administracion>Ej: DEMOBLOQUESWS

- trustKeystore: keystore que contiene la clave pública del certificado digital del servidor de firma de la plataforma de firma. Se utiliza para el acceso mediante SSL al servidor de firma.

Una vez finalizados los pasos anteriores, se accederá mediane la url:

https://<servidor_aplicaciones>:<puerto>/firmaenbloqueWS/index.jsp


Figura 18 – Aplicación de ejemplo de firma /multifirma de bloques. Acceso Web Sevices