Invocación por protocolo de aplicaciones nativas desde páginas Web

# ¿Qué es la invocación por protocolo?

Es un funcionamiento universal que los sistemas operativos mantengan una serie de asociaciones entre tipos de fichero y las aplicaciones que son capaces de tratarlos. Así, si en un sistema operativo Windows se indica que se abra un documento de texto, este consultará en el Registro de Windows cual es la aplicación por defecto asociada para su tratamiento (usualmente el Bloc de Notas), y procederá a abrir esta aplicación pasando como parámetro la ruta completa del fichero en el esquema de argumentos definido en el propio Registro de Windows como parte de la asociación.

Esta asociación se hace de distintas formas según el sistema operativo, en Windows es por su extensión (“.txt” en nuestro ejemplo), pero por ejemplo en Linux es por su MIME-Type (text/plain en el ejemplo).

Este mismo esquema se define igualmente en la mayoría de los sistemas operativos también para los esquemas comunes de protocolos basados en URN/URI/URL. Así, si por ejemplo en un sistema operativo Windows indicamos que queremos abrir <http://www.atos.net> (por ejemplo, desde la línea de comandos con la sentencia “start <http://www.atos.net>”) se iniciará el navegador Web por defecto, que es la aplicación asociada para tratar el protocolo *http,* procediendo a abrir esa página Web.

Este modo de abrir aplicaciones se conoce como invocación por protocolo, y de forma análoga a la invocación de aplicaciones indicando abrir un fichero, donde antes se recibía la ruta completa del fichero a abrir, ahora se recibe la URL/URI/URN completa que se indicó abrir.

# Invocación por protocolo desde navegador Web

Este mecanismo de invocación por protocolo de los sistemas operativos es usualmente accesible desde los navegadores Web. Esto quiere decir que si en la barra de direcciones del navegador Web indicamos una URI, el navegador Web trasladará el control al sistema operativo para que este localice la aplicación apropiada para tratar el protocolo asociado a la URI, y la abra pasándole dicha URI.

Un ejemplo de este mecanismo en Apple iOS podría ser el soporte del protocolo tel en forma de URN con el formato tel: //1-408-555-5555, donde 1-408-555-5555 es un número de teléfono. Así una llamada desde una página Web a esta URN con una sentencia HTML como la siguiente, <a href="tel://1-408-555-5555">1-408-555-5555</a>,provoca que se active el teléfono (en un iPhone) y realice una llamada a ese número, ya que la aplicación nativa de teléfono de iOS tiene registrado ese esquema de protocolo.

En este caso tenemos una salvedad evidente, y es que el navegador Web obviará esta transferencia de control al sistema operativo cuando el propio navegador sepa tratar el protocolo, por ejemplo, con *http*, *https*, *ftp*, etc.

## Advertencias de apertura

Como la invocación por protocolo no deja de ser una transferencia de datos desde una página Web (que no tiene porqué ser de confianza) a una aplicación nativa, los navegadores Web acostumbran a advertir de este cambio al usuario:

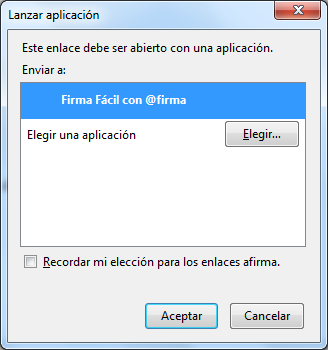


Ilustración 1: Advertencia en Firefox sobre Windows 7

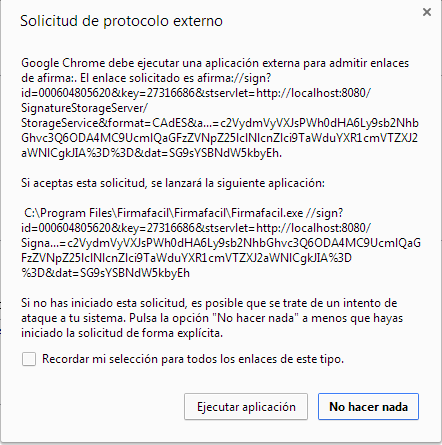


Ilustración 2: Advertencia en Google Chrome sobre Windows 7

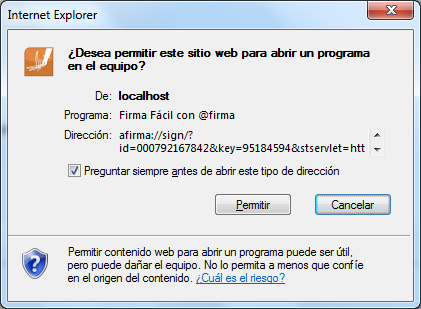


Ilustración 3: Advertencia en Internet Explorer 9 sobre Windows 7

En general, todos los navegadores Web muestran algún tipo de advertencia, excepto Apple Safari en Windows, OS X e iOS y WebKit (Android). En el caso de Internet Explorer se comprueba además la firma electrónica del ejecutable del programa nativo invocado.

## Soporte de la invocación por protocolo en los distintos navegadores

En general, únicamente Internet Explorer sobre Windows Phone 7.x presenta incompatibilidades con la invocación por protocolo (que no funciona cuando se pide abrir un protocolo desde JavaScript, pero sí cuando se hace una redirección mediante un META HTML). La invocación por protocolo funciona adecuadamente en Windows Phone 8.x.

Las pruebas de compatibilidad positiva se han realizado en Windows (XP, Vista, 7, 8, 8.1), Windows RT (8, 8.1), Windows Phone (8), Android (4.x), iOS (5, 6, 7), OS X (10.7, 10.8, 10.9) y Linux (varias distribuciones y versiones).

# La invocación por protocolo como sustituto de los Applets de Java

La invocación por protocolo puede, en ciertos casos, plantearse como un sustituto de los Applets de Java, si bien es necesario tener en cuenta siempre:

* En ciertos sistemas operativos, la invocación de una aplicación desde el navegador Web provoca un cambio de contexto gráfico molesto para el usuario (desde el navegador a la aplicación), y además al cerrarse la aplicación no siempre se vuelve de forma automática al navegador.
  + Sistemas operativos con cambios de contexto:
    - Microsoft Windows 8 en modo UI Moderno.
    - Apple iOS
  + Sistemas operativos sin cambio de contexto:
    - Apple OS X
    - Microsoft Windows
    - Google Android
    - Linux
  + Sistemas operativos con un “cambio parcial” de contexto (la experiencia de usuario es aceptable)
    - Microsoft Windows 8.1
* No es posible tener un UI integrado en la página Web, como ocurre con los Applets de Java, Adobe Flash o los controles Active X.
* Una vez se invoca la aplicación desde el navegador, no puede existir una comunicación bidireccional directa entre ambos (aplicación nativa y JavaScript de la página Web), como sí ocurre en los Applets de Java.
  + La aplicación nativa, no obstante, puede recibir información en la propia URL de invocación, aunque hay que tener en cuenta que la longitud de esta es limitada.

## Comunicación entre aplicación nativa y aplicación JavaScript en el navegador que la invocó

Como hemos comentado, una aplicación JavaScript ejecutándose en un navegador Web puede invocar una aplicación nativa siempre que esta esté registrada como la aplicación por defecto para tratar un protocolo que no trate el propio navegador (por ejemplo, el Cliente @firma usa el protocolo *afirma* en una URI del estilo *afirma://*), proporcionando ciertos datos como parte de la propia URI de invocación, pero… ¿Cómo puede la aplicación nativa devolver datos a la aplicación JavaScript?

El modo más directo y sencillo es usar un servidor (accesible por ambas partes) como intermediario en ese diálogo, en una secuencia acorde al siguiente esquema:



1. El navegador Web invoca a una *App* nativa mediante una URI especial, indicando una serie de información (datos a firmar, formato, opciones, etc.).
2. La *App* recibe los datos y realiza la firma electrónica usando las funciones nativas de gestión de claves y certificados.
3. La *App* nativa deposita el resultado de la firma en un servidor intermediario mediante una llamada a un servicio Web simple.
4. El navegador Web recoge el resultado de la operación de firma del servidor intermediario y continúa la ejecución de la lógica de negocio.

Como resultado tenemos una comunicación cuasi-bidireccional entre navegador Web (JavaScript) y *App* nativa, supliendo completamente a los complementos tradicionales.

Este modelo no obstante, requiere de ciertas precauciones para resultar eficaz y seguro:

* El servidor intermediario y la aplicación Web deben estar en el mismo servidor (para evitar advertencias de *cross site scripting*). Si se usa SSL cliente, este debe requerirse únicamente en el servidor que aloja la aplicación Web, y no en el servidor intermediario (para evitar una solicitud de autenticación a la aplicación nativa), pero siempre estando ambos con HTTPS en el mismo nombre de servidor.
* El servidor intermediario debe implementar mecanismos para asegurarse de que los datos depositados por una aplicación nativa sean recogidos únicamente por la aplicación JavaScript que la invocó:
  + Para ello deben implementarse al menos todos estos mecanismos:
    - Los datos deben cifrarse mediante una clave aleatoria de un solo uso generada al vuelo desde el programa JavaScript, que se pasa a la aplicación nativa mediante la URI de invocación.
    - Los datos deben tener un identificador aleatorio de un solo uso generado al vuelo desde el programa JavaScript, que se pasa a la aplicación nativa mediante la URI de invocación.
    - El servidor debe borrar cualquier dato que se deposite en él y que no sea requerido en un tiempo determinado (unos pocos minutos, que es lo máximo que puede durar la operación en la aplicación nativa).
  + No deben implementarse mecanismos derivados de la dirección IP, ya que las conexiones 3G pueden variar de IP en una misma sesión.

# El Cliente @firma y la invocación por protocolo

El Cliente @firma está actualmente adoptando la estrategia de la invocación por protocolo como sustitución de los Applets de Java (por los constante problemas de seguridad y compatibilidad) mediante:

* Windows, Linux, OS X.
  + Distribución de la aplicación “Firma Fácil con @firma”, una aplicación Java de escritorio cuyo instalador registra como aplicación por defecto para el protocolo “afirma”.
* Windows RT
  + Aplicación nativa específica.
    - Se usa el protocolo “afirmametro” para no interferir en los casos de Windows 8 y 8.1 que puedan tener Java y “Firma Fácil con @firma” instalado.
* iOS
  + Aplicación nativa específica.
* Android
  + Aplicación nativa específica

Se plantean para el futuro aplicaciones nativas para otros entornos que no soporten Java, como Windows Phone o BlackBerry 10.