Hoja de ruta del MiniApplet Cliente @firma

Índice de contenidos

[1 Evolución del MiniApplet @firma 3](#_Toc336788746)

[1.1 Versión 2.0 3](#_Toc336788747)

[1.1.1 Adopción de un modelo asíncrono en el API JavaScript 3](#_Toc336788748)

[1.1.2 Soporte de Google Android 4 3](#_Toc336788749)

[1.1.3 Soporte de Apple iOS 4](#_Toc336788750)

[1.2 Versión 2.1 4](#_Toc336788751)

[1.2.1 Soporte de Windows 8 y Windows 8 RT en modo nativo 4](#_Toc336788752)

[1.2.2 Soporte de PAdES en Google Android y Windows 8 4](#_Toc336788753)

[1.2.3 Soporte de multifirmas CAdES en Google Android y Windows 8 5](#_Toc336788754)

[1.3 Versión 2.2 5](#_Toc336788755)

[1.3.1 Soporte de PAdES en Apple iOS 5](#_Toc336788756)

[1.3.2 Soporte de multifirmas CAdES en Apple iOS 5](#_Toc336788757)

[1.4 Versión 2.3 5](#_Toc336788758)

[1.4.1 Soporte de Mac OS X en modo nativo 5](#_Toc336788759)

[1.4.2 Soporte de claves y certificados en tarjeta SIM 6](#_Toc336788760)

# Evolución del MiniApplet @firma

## Versión 2.0

### Adopción de un modelo asíncrono en el API JavaScript

En la actualidad, el MiniApplet Cliente @firma usa un modelo síncrono en la comunicación entre el Applet de Java y el código JavaScript. Este modelo presenta ciertos problemas de importancia:

* El código bloqueante (síncrono) ofrece una mala experiencia de usuario, y en ciertos navegadores (Firefox, etc.) automáticamente se le invita al usuario a detener la ejecución del MiniApplet por considerarlo falto de respuesta.
* El código bloqueante es incompatible con los modelos de ejecución necesarios para implementar la operativa de firma prescindiendo de Java, como por ejemplo, con *Apps* en entornos móviles.

La tarea propuesta comprenderá una transición inicial a un modelo asíncrono en el API JavaScript mediante el uso de punteros a funciones (*callbacks*).

### Soporte de Google Android 4

Ya con el modelo de API JavaScript asíncrono implementado, se propone el dar soporte a Google Android (usando una App nativa en sustitución del Applet de Java) de al menos las siguientes características del MiniApplet:

* Firmas CAdES simples
* Funciones auxiliares de conversión a y desde Base64 (la gestión de juegos de caracteres queda limitada al soporte que ofrece JavaScript, ya que no se cuenta con Java).
* Funciones auxiliares adicionales

Quedaría directamente excluida la funcionalidad de filtro de certificados, ya que en los dispositivos móviles Android la selección de certificado es un proceso opaco sobre el cual no es posible incluir (es una medida de seguridad del sistema operativo).

### Soporte de Apple iOS

Se propone la implementación para Apple iOS de las características anteriormente descritas para Google Android 4, con un soporte equivalente de funcionalidades:

* Uso del API JavaScript asíncrono.
* Uso de una *App* nativa para la gestión de claves y certificados y la realización de la propia firma.
* Soporte de CAdES simple.

## Versión 2.1

### Soporte de Windows 8 y Windows 8 RT en modo nativo

Internet Explorer 10 en Windows 8 (modo Metro), Windows 8 RT y, en gran medida, en Windows Phone 8 no soportan Applets de Java en las páginas Web, por lo que es completamente incompatible con los actuales Applet y MiniApplet Cliente @firma.

No obstante, Microsoft abre la puerta a una más segura integración de aplicaciones externas con aplicaciones Web mediante la incorporación en Windows 8 de los llamados “Contratos de Compartición” (*Share Contracts*). Esta técnica de construcción de aplicaciones permite el traspaso de información entre aplicaciones Web basadas en JavaScript y aplicaciones nativas instaladas en Windows 8.

Siguiendo este nuevo modelo, se propone la construcción de un módulo nativo (C#, .NET) Windows capaz de suplir las funcionalidades del MiniApplet Java y permitiendo una operativa equivalente al MiniApplet en plataformas Apple iOS o Google Android, de forma que sea posible la ejecución del MiniApplet en Windows 8 (modo Metro) y Windows 8 RT, y sirviendo este desarrollo de base al soporte de Windows Phone 8.

### Soporte de PAdES en Google Android y Windows 8

Dado que el soporte propuesto para Windows 8 y Google Android se limitaba a firmas CAdES simples, se propone una ampliación en estas plataformas para el soporte de PAdES, para lo cual se desarrollará:

* Motor PAdES nativo para Android (Dalvik)
* Motor PAdES nativo para Windows 8 (C# .NET)

El soporte de PAdES se hará de una forma uniforme respecto al API actual para uso de PAdES con el MiniApplet (basado en Applet de Java).

### Soporte de multifirmas CAdES en Google Android y Windows 8

Dado que el soporte propuesto para Windows 8 y Google Android se limitaba a firmas CAdES simples, se propone una ampliación de los motores CAdES para Android y Windows 8 para el adecuado soporte de multifirmas CAdES (cofirmas y contrafirmas).

## Versión 2.2

### Soporte de PAdES en Apple iOS

Dado que el soporte propuesto para Apple iOS se limitaba a firmas CAdES simples, se propone una ampliación en estas plataformas para el soporte de PAdES, para lo cual se desarrollará:

* Motor PAdES nativo para Apple iOS (Objective C / Xcode)

El soporte de PAdES se hará de una forma uniforme respecto al API actual para uso de PAdES en el resto de plataformas.

### Soporte de multifirmas CAdES en Apple iOS

Dado que el soporte propuesto para Apple iOS se limitaba a firmas CAdES simples, se propone una ampliación del motor CAdES para Apple iOS para el adecuado soporte de multifirmas CAdES (cofirmas y contrafirmas).

## Versión 2.3

### Soporte de Mac OS X en modo nativo

El soporte de Java como entorno de ejecución para aplicación Web es una característica a extinguir en los sistemas operativos modernos, ya que es una fuente común de entrada de software malintencionado y supone un serio problema de seguridad.

En Mac OS X, la dirección en la que se avanza consiste en dificultar la ejecución de Applets de Java, más que en impedir completamente su ejecución (como ocurre en Windows 8):

* Apple ya no mantiene un entorno de ejecución de Java (Java 7 en Mac OS X ya no es un producto Apple, como ocurría con Java 5 y 6.
* Las actualizaciones del sistema operativo (Lion, Mountain Lion) desinstalan directamente el entorno de ejecución de Java (JRE) sin consultar al usuario.
* Tras instalar Java, la ejecución de Applets viene desactivada, y es necesario activarla manualmente.
  + Tras activarla, se desactiva automáticamente y sin intervención del usuario tras un tiempo sin usar Java, obligando al usuario a volver a activarla.
* Aun con la ejecución de Applets activada, no es posible ejecutar un Applet descargado directamente de la Web de un organismo público español (descargado de cualquier sitio que no sea un servicio de distribución de aplicaciones de la propia Apple), y para permitirlo el usuario debe relajar las restricciones de seguridad del sistema operativo (lo cual aumenta los riesgos de usos no autorizados, virus, software malintencionado, etc.).

Con esas restricciones, la experiencia de usuario del MiniApplet (o Applet) Cliente @firma en Mac OS X es francamente deficiente, por lo que es conveniente plantear alternativas.

Se propone el desarrollo de un complemento para navegadores Web en Apple Mac OS X para la realización de firmas electrónicas sustituyendo de forma transparente al Applet Java del MiniApplet Cliente @firma, usando preferentemente NSAPI para proporcionar compatibilidad tanto a Apple Safari como a Mozilla Firefox o Google Chrome.

Se propone un soporte inicial de CAdES simple.

### Soporte de claves y certificados en tarjeta SIM

Uno de los problemas de la mayoría de los dispositivos móviles a la hora de hacer firmas electrónicas es la imposibilidad de usar SSCD (dispositivos seguros de creación de firmas), ya que es difícil, incómodo o simplemente imposible el conectar un lector de tarjetas inteligentes, un SSCD en USB u otro SSCD al terminal.

No obstante, una característica común de la mayoría de los terminales móviles es que disponen de una tarjeta SIM/USIM para el acceso a la red, y estas tarjetas pueden ser además un SSCD.

La especificación GSM (GSM 11.11 y GSM 11.14) especifican medios para el acceso a la tarjeta SIM (SIM Application Toolkit) desde o aplicaciones externas accediendo al API JavaCard.

Se propone una tarea de investigación sobre las posibilidades de soportar certificados y claves de firma en tarjeta SIM desde las aplicaciones móviles del proyecto @firma.