Descripción de la sintaxis para la invocación por protocolo del Cliente @firma

Índice de contenidos

[1 Objetivo 3](#_Toc53138705)

[2 Introducción 3](#_Toc53138706)

[3 Estructura de la URL de invocación 5](#_Toc53138707)

[4 Comunicación por WebSockets 6](#_Toc53138708)

[5 Comunicación por *sockets* 7](#_Toc53138709)

[6 Comunicación por servidor intermedio 7](#_Toc53138710)

[6.1 URL con parámetros de operación 9](#_Toc53138711)

[6.2 URL sin parámetros de operación 10](#_Toc53138712)

[7 Operaciones del Cliente @firma 12](#_Toc53138713)

[7.1 Operación de firma (*sign*) 12](#_Toc53138714)

[7.2 Operación de cofirma o firma en paralelo (*cosign*) 14](#_Toc53138715)

[7.3 Operación de contrafirma o firma en cascada (*contersign*) 16](#_Toc53138716)

[7.4 Firma en lote (*batch*) 18](#_Toc53138717)

[7.5 Selección de certificado (*selectcert*) 20](#_Toc53138718)

[7.6 Operación de guardado (*save*) 21](#_Toc53138719)

[7.7 Carga de datos (*load*) 22](#_Toc53138720)

[7.8 Firma y guardado (*signandsave*) 23](#_Toc53138721)

[7.9 Recuperación del log de la aplicación (*getLog*) 24](#_Toc53138722)

[ANEXO I. Servidor intermedio 25](#_Toc53138723)

[ANEXO I.1 Guardado de datos 25](#_Toc53138724)

[ANEXO I.2 Recuperación de datos 26](#_Toc53138725)

# Objetivo

El presente documento presenta la sintaxis de invocación de las aplicaciones del Cliente @firma para permitir el uso de sus capacidades de firma desde una aplicación externa. El documento expone los distintos mecanismos de comunicación existentes entre la aplicación cliente y la aplicación de firma, las operaciones admitidas y los parámetros que se deben proporcionar para cada una de estas operaciones.

# Introducción

El Cliente @firma es un conjunto de aplicaciones cliente que permiten al usuario ejecutar operaciones de firma con sus certificados electrónicos locales. Las aplicaciones englobadas en la denominación Cliente @firma son:

* AutoFirma:
  + Aplicación de escritorio para sistemas Windows, Linux y macOS. Puede ser invocada desde una aplicación externa para la generación de firmas electrónicas o generar firmas directamente desde su interfaz gráfica. Permite el uso de distintos almacenes de certificados del sistema, almacenes PKCS#12/JKS y tarjetas inteligentes.
* Cliente móvil Android:
  + Aplicación de firma para sistemas Android. Puede ser invocada desde una aplicación externa para la generación de firmas electrónicas o generar firmas directamente desde su interfaz gráfica. Utiliza los certificados del almacén del sistema.
* Cliente móvil iOS:
  + Aplicación de firma para sistemas iOS. Puede ser invocada desde una aplicación externa para la generación de firmas electrónicas. Utiliza certificados previamente importados en la aplicación a través del sistema de ficheros o iTunes.

El modo común de operación es la invocación de estas aplicaciones desde un trámite Web por medio del JavaScript de despliegue (autoscript.js) suministrado con el kit de integración del Cliente @firma. Este modo de operación permite a las aplicaciones web enviar datos al Cliente @firma para firmarlos y obtener las firmas resultantes como parte del flujo de un trámite online.

El mecanismo de invocación del Cliente @firma sólo permite enviar a la aplicación de firma una serie de parámetros, pero no obtener una respuesta. Esto obliga a establecer un mecanismo de comunicación bidireccional que será definido por la aplicación que realiza la llamada al Cliente @firma. A lo largo de la vida del Cliente @firma, este mecanismo de comunicación ha ido evolucionando, pero se ha mantenido la compatibilidad hacia atrás para permitir la compatibilidad entre versiones del Cliente @firma y dadas las limitaciones del entorno de algunas de las aplicaciones.

Los mecanismos de comunicación que existen actualmente son:

* Comunicación a través de servidor intermedio:
  + El paso de datos entre la aplicación cliente y la aplicación de firma se realizará directamente en la llamada de invocación y la respuesta se subirá a un servidor intermedio del que la descargará la aplicación cliente. En caso de que los parámetros de la llamada sean demasiado grandes, estos se enviarán al servidor intermedio para que los descargue la aplicación de firma.
  + El servidor intermedio lo proporcionará el organismo propietario de la aplicación y los servicios para el guardado y la descarga de la información se distribuyen en el kit de despliegue del Cliente @firma.
  + Las URL de acceso a los servicios de carga y descarga de datos del servidor los proporciona la aplicación cliente en la llamada a la aplicación de firma.
  + Este modo de operación obliga a que la aplicación de firma se tenga que abrir por cada petición de la aplicación cliente.
  + Este mecanismo de comunicación está soportado por todas las aplicaciones cliente.
* Comunicación por Sockets:
  + El paso de datos entre la aplicación cliente y la aplicación de firma se hace a través de un socket abierto por la aplicación de firma. Tras la invocación por protocolo, la aplicación de firma abrirá el socket. La aplicación cliente solicitará todas las peticiones a través de ese socket y obtendrá la respuesta del mismo.
  + La aplicación cliente envía tres números de puerto aleatorios a la aplicación de firma en la llamada por protocolo y la aplicación tratará de abrir el socket en uno de ellos. Si fallase la apertura del socket, lo intentaría con el siguiente puerto hasta conseguir abrir uno o fallar si no se pudo en ninguno de los puertos proporcionados. La aplicación cliente tratará de comunicarse con la aplicación de firma por medio de los 3 puertos hasta que identifique cual será finalmente el puerto por el que se establecerá la comunicación.
  + Este modo de operación mantiene abierta la aplicación cliente hasta que pasa un minuto de inactividad tras la última llamada.
  + Este mecanismo de comunicación sólo está soportado por AutoFirma.
* Comunicación por WebSockets:
  + El paso de datos entre la aplicación cliente y la aplicación de firma se hace a través de un WebSocket abierto por la aplicación de firma en el puerto 63117. Tras la invocación por protocolo, la aplicación de firma abrirá el WebSocket. La aplicación cliente solicitará todas las peticiones a través de ese socket y obtendrá la respuesta del mismo.
  + Este modo de operación mantiene abierta la aplicación cliente hasta que la aplicación cliente cierra la conexión.
  + Este mecanismo de comunicación sólo está soportado por AutoFirma 1.7 y superiores.

Todas las operaciones con el Cliente @firma deben hacerse de forma secuencial. No se debe realizar una nueva petición a la aplicación hasta que no se ha recibido la respuesta de la petición anterior.

# Estructura de la URL de invocación

Durante la instalación de las aplicaciones cliente de firma, estas registran el esquema “afirma” en el sistema. De esa manera, todas las llamadas que se realicen con ese protocolo abrirán la aplicación de firma instalada.

Podemos distinguir las siguientes partes en la estructura de una URL:

[scheme://][authority][path][?query][#fragment]

* [scheme://] Esquema de la URL:
  + En las llamadas al Cliente @firma siempre será ‘afirma’.
* [authority] Autoridad del protocolo:
  + Comúnmente puede contener una sección *host*, un puerto e información del usuario.
  + En las llamadas al Cliente @firma este parámetro expresará la operación a realizar. Los valores que se pueden establecer en las llamadas al Cliente @firma dependen de la operación y del modo de comunicación que se desee activar entre la aplicación de firma y el cliente que lo solicita.
* [path] Ruta del protocolo:
  + En la llamada al Cliente @firma, no se admite que figuren rutas.
* [?query] Consulta del protocolo:
  + Son los parámetros que deseamos transmitir a la aplicación.
  + En las llamadas al Cliente @firma, contendrá los parámetros que configuran la operación indicada en el apartado de [authority].
  + Los parámetros se proporcionarán con codificación URL Encoded. En el caso de valores en Base 64, también se proporcionarán con una codificación URL SAFE (caracteres ‘+’ sustituidos por ‘-’ y caracteres ‘/’ sustituidos por ‘\_’).
* [#fragment] Fragmento del protocolo:
  + En la llamada al Cliente @firma, no se admite que figure el apartado de fragmento.

Algunos ejemplos de llamadas al Cliente @firma son:

* afirma://websocket/?v=3
  + URL de invocación para iniciar la comunicación por WebSockets.
* afirma://sign/?ver=3&op=sign&id=yy1oMJsxoHSoMzLdJywW&key=71313199&stservlet=http%3A%2F%2Fappprueba%3A8080%2Fafirma-signature-storage%2FStorageService&format=CAdES&algorithm=SHA512withRSA&properties=c2VydmVyVXJsPWh0dHA6Ly9hcHBwcnVlYmE6ODA4MC9hZmlybWEtc2VydmVyLXRyaXBoYXNlLXNpZ25lci9TaWduYXR1cmVTZXJ2aWNl&aw=true
  + URL de invocación para solicitar la firma de un fichero seleccionado por el usuario con la configuración proporcionada.

En los siguientes apartados se describen las URL de invocación que se pueden utilizar para la comunicación con el Cliente @firma según el tipo de comunicación escogido.

# Comunicación por WebSockets

La comunicación por WebSockets requiere de una única llamada por protocolo al Cliente @firma. Una vez se le invoca, el resto de la comunicación se realizará a través del *websocket* abierto.

La composición de la URL para el inicio de la comunicación por WebSockets es la siguiente:

* Esquema:
  + afirma
* Host:
  + websocket
* Parámetros:
  + v
    - Número de versión del protocolo.
    - La versión actual de protocolo que se soporta es: 3

La URL de llamada al Cliente @firma para el inicio de una comunicación por WebSocket es la siguiente:

|  |
| --- |
| afirma://websocket?v=3 |

Para ordenar a través de WebSocket la ejecución las distintas operaciones del Cliente @firma se enviarán al WebSocket nuevas URL con los parámetros específicos para la ejecución de esa operación. Se pueden consultar las distintas operaciones en el apartado 7 Peticiones de operación.

Adicionalmente a las peticiones de operación, se puede enviar a través del socket la cadena “echo=” para comprobar si el WebSocket está activo. La respuesta a esta petición deber ser la cadena “OK”.

La comunicación por WebSocket sólo está soportada por AutoFirma 1.7 y superiores.

# Comunicación por *sockets*

La comunicación por *sockets* requiere de una única llamada por protocolo al Cliente @firma. Una vez se le invoca, el resto de la comunicación se realizará a través del *socket* abierto.

La composición de la URL para el inicio de la comunicación por *sockets* es la siguiente:

* Esquema:
  + afirma
* Host:
  + service
* Parámetros:
  + v
    - Número de versión del protocolo.
    - La versión actual de AutoFirma soporta la versión: 1
  + ports
    - Listado de puertos (separados por comas) por los que se va a intentar establecer la comunicación.
  + idsession
    - Cadena base 64 a modo de identificador de sesión.
    - Todas las peticiones enviadas a través del socket una vez abierto deberán notificar este identificador de sesión.

Una URL de ejemplo de llamada al Cliente @firma para el inicio de una comunicación por sockets es la siguiente:

|  |
| --- |
| afirma://service?ports=60572,64733,56552&v=1&idsession=CT8Dubp4IDxjSnhxh15A |

Para ordenar a través de *socket* la ejecución las distintas operaciones del Cliente @firma se enviarán al *socket* nuevas URL con los parámetros específicos para la ejecución de esa operación. Se pueden consultar las distintas operaciones en el apartado 7 Peticiones de operación.

La comunicación por Sockets sólo está soportada por AutoFirma.

# Comunicación por servidor intermedio

La comunicación por servidor intermedio no permite abrir el cliente de firma y mantener una comunicación con varios mensajes. Este modo requiere que se habrá la aplicación de firma por cada petición, motivo por el cual no permite el uso de algunas de las funciones auxiliares, ya que no está justificado el que se abra la aplicación sólo por ellas. Tampoco es adecuada para casos de uso en los que se realizan múltiples aplicaciones simultaneas.

La comunicación por servidor intermedio tiene las siguientes ventajas:

* Es compatible con los clientes móviles Android e iOS.
* No requieren que en la instalación de la aplicación se instale un certificado SSL en el almacén de confianza de los navegadores.

Por el contrario, presenta las siguientes desventajas frente a las opciones anteriores:

* Requiere desplegar los servicios de intercambio de datos en un servidor accesible por la aplicación cliente y la aplicación de firma.
* Más lento.
* Puede tener algún recargo para el usuario si el uso de la red para la conexión con el servidor tiene un gasto asociado (como en el caso de la comunicación por datos móviles).
* No es apto para ser usado recurrentemente. Algunas aplicaciones, incluso, pueden haber establecido restricciones de seguridad que bloqueen las llamadas, como Google Chrome que establece un tiempo máximo que puede transcurrir entre una interacción del usuario y la llamada por protocolo.

La principal particularidad de las llamadas para el uso del servidor intermedio es que requieren que se a la aplicación de firma las URL de los servicios para el guardado del resultado de la operación y, si aplica, la descarga de los datos de la petición.

Adicionalmente, pueden darse dos casos de uso:

* Todos los parámetros de la operación pueden enviarse en la URL.
  + En este caso, se llama a la aplicación de firma utilizando la URL correspondiente a la operación que deseemos proporcionándole todos los parámetros. Este puede ser el caso de uso normal en muchos contextos, pero no en otros en los que se establece un tamaño máximo de la URL de entrada. Esto ocurre, por ejemplo, con los navegadores web.
* Los parámetros de la operación no pueden enviarse en la URL.
  + En este caso, los parámetros a enviar conforman una URL demasiado grande para ser transmitida directamente. Para poder transferir los datos a la aplicación de firma, será necesario que se suban estos datos al servidor intermedio y luego se llame a la aplicación de firma para que se los descargue de ahí y pueda procesarlos a continuación.

La comunicación con el servidor intermedio es asíncrona, dado que la aplicación cliente debe consultar activamente el servidor intermedio a través del servicio de recuperación de datos para saber si la operación se ha procesado o no. En este caso, además de la respuesta con el resultado de la operación, la aplicación podría recuperar también alguna de las siguientes respuestas:

* Cadena que empieza por “err-06”:
  + Señala que el servidor intermedio no ha recibido el resultado de la operación. La aplicación cliente debería esperar un poco más antes de volver a reintentar recuperar el resultado de la operación.
* Cadena que empieza por “#wait”:
  + Señala que la aplicación de firma todavía no tiene el resultado de la aplicación, pero ha indicado que la petición se está en procesó y que la aplicación cliente debería seguir esperando el resultado.
  + Este valor se usará cuando se haya invocado a AutoFirma con el parámetro “aw” para que el Cliente @firma (solo soportado por AutoFirma) vaya notificando periódicamente si la aplicación cliente debería seguir esperando el resultado.

## URL con parámetros de operación

En el caso de llamar al Cliente @firma con los parámetros de la operación, pasaremos todos los parámetros necesarios para la propia operación más los parámetros necesarios para guardar el resultado en el servidor intermedio.

La composición de la URL para este tipo de petición sería la siguiente:

* Esquema:
  + afirma
* Host:
  + Código de la operación en cuestión
* Parámetros:
  + ver
    - Número de versión del protocolo.
    - La versión actual de protocolo que se soporta es: 3
  + id
    - Identificador de la transacción. Es el identificador con el que debe guardarse la respuesta en el servidor intermedio.
    - Para más información sobre el guardado y la recuperación de datos del servidor intermedio, consulte el apartado ANEXO I Servidor intermedio.
  + stservlet
    - URL del servicio para el guardado del resultado en el servidor intermedio.
    - Para más información sobre el guardado y la recuperación de datos del servidor intermedio, consulte el apartado ANEXO I Servidor intermedio.
  + key
    - Clave DES para el cifrado del fichero de configuración.
    - Los datos que se envíen al servidor intermedio se deben cifrar con esta clave DES. La aplicación cliente después deberá usar esta misma clave para descifrar el resultado.
  + aw
    - Booleano que indica si la aplicación de firma debe notificar periódicamente a través del servidor intermedio si aún está operando. Esto es útil para evitar esperar indefinidamente el resultado de la operación, cuando es posible que esta haya fallado por algún motivo y no se haya podio notificar a través del servidor intermedio. Por ejemplo, si la aplicación de firma no recibió la petición o si no pudo enviar el resultado al servidor intermedio.
    - Por defecto, false.
  + Parámetros de operación
    - Parámetros propios del tipo de operación que deseemos realizar.

Un ejemplo de URL para la invocación de una operación de firma sería:

|  |
| --- |
| afirma://sign/?ver=1&op=sign&id=llUQSzruWssVQxIeFNtV&key=72993139&stservlet=https%3A%2F%2Fvalide.redsara.es%2FfirmaMovil%2Fafirma-signature-storage%2FStorageService&format=CAdES&algorithm=SHA256withRSA&properties=bW9kZT1leHBsaWNpdApzZXJ2ZXJVcmw9aHR0cHM6Ly92YWxpZGUucmVkc2FyYS5lcy9maXJtYU1vdmlsL2FmaXJtYS1zZXJ2ZXItdHJpcGhhc2Utc2lnbmVyL1NpZ25hdHVyZVNlcnZpY2U%3D&aw=true |

Las operaciones disponibles y sus parámetros están indicadas en el apartado 7 Peticiones de operación.

## URL sin parámetros de operación

En el caso de llamar al Cliente @firma sin los parámetros de la operación, deberemos haber enviado estos parámetros previamente al servidor intermedio, y pasar al Cliente @firma los parámetros necesarios para que se los descargue.

La composición de la URL para este tipo de petición sería la siguiente:

* Esquema:
  + afirma
* Host:
  + Código de la operación en cuestión
* Parámetros:
  + fileId
    - Identificador asignado al fichero de configuración en el servidor remoto. Este es el identificador que se le asigna a la configuración cuando se envía a almacenar en el servidor intermedio.
    - Para más información sobre el guardado y la recuperación de datos del servidor intermedio, consulte el apartado ANEXO I Servidor intermedio.
  + rtservlet
    - URL del servicio para la recuperación del fichero de configuración.
    - Para más información sobre el guardado y la recuperación de datos del servidor intermedio, consulte el apartado ANEXO I Servidor intermedio.
  + key
    - Clave DES para el descifrado del fichero de configuración.
    - Los datos que se envían al servidor intermedio deberían cifrarse con un cifrado DES. En caso de hacerlo, en este parámetro se debería indicar la clave empleada (que debería ser una cadena de texto).

Un ejemplo de URL de invocación sería:

afirma://sign/?fileid=1mlL13MJMKMU4KMUI4Co&rtservlet=https%3A%2F%2Fvalide.redsara.es%2FfirmaMovil%2Fafirma-signature-retriever%2FRetrieveService&key=56542304

Los parámetros que se deberán enviar al servidor intermedio son los mismos que usaríamos si todos los parámetros se enviasen en la URL de petición. Consulte el apartado anterior para comprobar los parámetros necesario y el apartado 7 Peticiones de operación para ver el listado de operaciones disponibles y los parámetros que estas necesitan.

Para el envío de los datos al servidor, los estructuraremos en forma de fichero XML con el formato:

<COD\_OPERACION>

<e k=”CLAVE1” v=”VALOR1” />

<e k=”CLAVE2” v=”VALOR2” />

…

<e k=”CLAVEn” v=”VALORn” />

</COD\_OPERACION>

En donde:

* *COD\_OPERACION*: Código de la operación que se configura.
* *CLAVEx*: Nombre del parámetro del listado de parámetros de configuración de la operación.
* *VALORx*: Valor que deseemos asignar a ese parámetro.

Un ejemplo de XML es:

|  |
| --- |
| <sign>  <e k="id" v="000987508929"/>  <e k="key" v="39772569"/>  <e k="stservlet" v="http://192.168.177.173:8080/afirma-signature-storage/StorageService"/>  <e k="format" v="CAdES"/>  <e k="algorithm" v="SHA1withRSA"/>  <e k="properties" v="c2VydmVyVXJsPWh0dHA6Ly8xOTIuMTY4LjE3Ny4xNzM6ODA4MC9hZmlybWEtc2VydmVyLXRyaXBoYXNlLXNpZ25lci9TaWduYXR1cmVTZXJ2aWNlCgkJIA%3D%3D"/>  <e k="dat" v="ZnVsYW5pdG9AYXRvcy5uZXQ%3D"/>  </sign> |

Este XML se enviará, codificado en base 64 al mismo servicio de almacenamiento cuya ruta se establece a través del parámetro “stservlet”. Esta cadena a su vez puede estar cifrada con una clave DES (la proporcionada en el parámetro “key” de la llamada) y codificada en base 64.

Es importante recordar que no todas las operaciones del Cliente @firma pueden ejecutarse cuando la comunicación se realiza a través de servidor intermedio.

# Operaciones del Cliente @firma

Aquí se listan todas las operaciones soportadas por el Cliente @firma cuando se le invoca por protocolo y los parámetros necesarios para configurar dichas operaciones. Algunas de ellas, sin embargo, no están disponibles en todas las aplicaciones.

El listado completo de operaciones es el siguiente:

* Firma.
* Cofirma.
* Contrafirma.
* Firma de lote.
  + Sólo disponible en AutoFirma.
* Selección de certificado.
* Guardado.
  + Sólo disponible en AutoFirma y la app de firma Android.
* Firma y guardado.
  + Sólo disponible en AutoFirma.
* Carga de datos.
  + Sólo disponible en AutoFirma.
* Recuperación del log.
  + Sólo disponible en AutoFirma.

Los parámetros de las aplicaciones siempre se proporcionarán codificados para URL (“*URL Encoded*”). Si el valor de un parámetro se expresase en forma de cadena Base 64, ésta siempre se codificará en forma *URL SAFE* (el símbolo ‘+’ se sustituye por ‘-’ y el símbolo ‘/’ se sustituye por ‘\_’).

## Operación de firma (*sign*)

Operación para la ejecución de una firma electrónica.

Código de operación:

* sign

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: sign
* dat
  + Datos a firmar codificados en Base 64 URL SAFE.
  + Si no se indican y la aplicación es AutoFirma, se permitirá al usuario seleccionar un fichero.
* format
  + Formato de firma.
* algorithm
  + Algoritmo de firma.
* properties
  + Propiedades de configuración específicas del formato de firma y configuración de la operación.
  + Se proporcionarán en forma cadena compuesta por duplas propiedad=valor separadas por ‘\n’. La cadena resultante se proporcionará en Base 64 URL SAFE.
* ksb64 (opcional)
  + Configuración del almacén de claves a utilizar codificada en base 64.
  + Sólo se procesa en AutoFirma.
* sticky (opcional)
  + Valor booleano que indica si debe recordarse el certificado de firma que se seleccione o, si ya se recuerda uno anterior, si debe usarse ese automáticamente (true); o si se debe permitir seleccionar un nuevo certificado y olvidar cualquiera que se recuerde (false).
  + Sólo se procesa en AutoFirma.
* resetsticky (opcional)
  + Valor booleano que indica si debe olvidarse cualquier certificado previamente seleccionado independientemente del valor del parámetro sticky (true), o si no (false, valor por defecto).
  + Este valor se puede utilizar junto con sticky si ya teníamos un certificado seleccionado y queremos olvidar ese y recordar el nuevo que se seleccione.
  + Sólo se procesa en AutoFirma.

Para ver los formatos, algoritmos, parámetros de configuración del formato de firma y los almacenes de claves a configurar, consulte el manual del integrador del Cliente @firma.

En caso de finalizar correctamente, esta operación obtiene como resultado una cadena con el formato:

CERT\_B64|FIRMA\_B64|FILENAME\_B64

En donde:

* CERT\_B64
  + Certificado de firma utilizado codificado en Base 64 URL SAFE.
* FIRMA\_B64
  + Firma generada codificada en Base 64 URL SAFE.
* FILENAME\_B64
  + Si no se indicaron los datos a firmar y el usuario tuvo que seleccionar un fichero, este será el nombre del fichero seleccionado codificado en Base 64 URL SAFE.
  + Este valor solo se devolverá en AutoFirma 1.7 y superiores.
  + En caso de no aparecer, tampoco lo hará el carácter separador ‘|’.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de firma es:

|  |
| --- |
| afirma://sign/?ver=1&op=sign&id=llUQSzruWssVQxIeFNtV&format=CAdES&algorithm=SHA256withRSA&properties=bW9kZT1leHBsaWNpdApzZXJ2ZXJVcmw9aHR0cHM6Ly92YWxpZGUucmVkc2FyYS5lcy9maXJtYU1vdmlsL2FmaXJtYS1zZXJ2ZXItdHJpcGhhc2Utc2lnbmVyL1NpZ25hdHVyZVNlcnZpY2U%3D |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Operación de cofirma o firma en paralelo (*cosign*)

Operación para la ejecución de una firma electrónica.

Código de operación:

* cosign

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: cosign
* dat
  + Firma a cofirmar codificada Base64 URL SAFE.
  + Si no se indican y la aplicación es AutoFirma, se permitirá al usuario seleccionar un fichero.
* format
  + Formato de firma.
* algorithm
  + Algoritmo de firma.
* properties
  + Propiedades de configuración específicas del formato de firma y configuración de la operación.
  + Se proporcionarán en forma cadena compuesta por duplas propiedad=valor separadas por ‘\n’. La cadena resultante se proporcionará en Base 64 URL SAFE.
* ksb64 (opcional)
  + Configuración del almacén de claves a utilizar codificada en base 64.
  + Sólo se procesa en AutoFirma.
* sticky (opcional)
  + Valor booleano que indica si debe recordarse el certificado de firma que se seleccione o, si ya se recuerda uno anterior, si debe usarse ese automáticamente (true); o si se debe permitir seleccionar un nuevo certificado y olvidar cualquiera que se recuerde (false).
  + Sólo se procesa en AutoFirma.
* resetsticky (opcional)
  + Valor booleano que indica si debe olvidarse cualquier certificado previamente seleccionado independientemente del valor del parámetro sticky (true), o si no (false, valor por defecto).
  + Este valor se puede utilizar junto con sticky si ya teníamos un certificado seleccionado y queremos olvidar ese y recordar el nuevo que se seleccione.
  + Sólo se procesa en AutoFirma.

Para ver los formatos, algoritmos, parámetros de configuración del formato de firma y los almacenes de claves a configurar, consulte el manual del integrador del Cliente @firma.

En caso de finalizar correctamente, esta operación obtiene como resultado una cadena con el formato:

CERT\_B64|COFIRMA\_B64|FILENAME\_B64

En donde:

* CERT\_B64
  + Certificado de firma utilizado codificado en Base 64 URL SAFE.
* COFIRMA\_B64
  + Cofirma generada codificada en Base 64 URL SAFE.
* FILENAME\_B64
  + Si no se indicó una firma a cofirmar y el usuario tuvo que seleccionar un fichero, este será el nombre del fichero seleccionado codificado en Base 64 URL SAFE.
  + Este valor solo se devolverá en AutoFirma 1.7 y superiores.
  + En caso de no aparecer, tampoco lo hará el carácter separador ‘|’.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de cofirma (en la que se omite parte del parámetro de datos por legibilidad) es:

|  |
| --- |
| afirma://cosign?op=cosign&algorithm=SHA512withRSA&format=XAdES&properties=c2VydmVyVXJsPWh0dHA6Ly9hcHBwcnVlYmE6ODA4MC9hZmlybWEtc2VydmVyLXRyaXBoYXNlLXNpZ25lci9TaWduYXR1cmVTZXJ2aWNl&sticky=false&dat=PD94bWwgdmVyc2lv…B-PC9wcm9qZWN0Pg== |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Operación de contrafirma o firma en cascada (*contersign*)

Operación para la ejecución de una firma electrónica.

Código de operación:

* countersign

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: countersign
* dat
  + Firma a contrafirmar codificada en Base64 URL SAFE.
  + Si no se indican y la aplicación es AutoFirma, se permitirá al usuario seleccionar un fichero.
* format
  + Formato de firma.
* algorithm
  + Algoritmo de firma.
* properties
  + Propiedades de configuración específicas del formato de firma y configuración de la operación.
  + Se proporcionarán en forma cadena compuesta por duplas propiedad=valor separadas por ‘\n’. La cadena resultante se proporcionará en Base 64 URL SAFE.
  + Para la operación de contrafirma, se puede indicar la propiedad “target” con el valor “tree” para indicar que se desea contrafirmar todo el árbol de firmas o con cualquier otro valor para indicar que sólo se deben contrafirmas los nodos hoja.
* ksb64 (opcional)
  + Configuración del almacén de claves a utilizar codificada en base 64.
  + Sólo se procesa en AutoFirma.
* sticky (opcional)
  + Valor booleano que indica si debe recordarse el certificado de firma que se seleccione o, si ya se recuerda uno anterior, si debe usarse ese automáticamente (true); o si se debe permitir seleccionar un nuevo certificado y olvidar cualquiera que se recuerde (false).
  + Sólo se procesa en AutoFirma.
* resetsticky (opcional)
  + Valor booleano que indica si debe olvidarse cualquier certificado previamente seleccionado independientemente del valor del parámetro sticky (true), o si no (false, valor por defecto).
  + Este valor se puede utilizar junto con sticky si ya teníamos un certificado seleccionado y queremos olvidar ese y recordar el nuevo que se seleccione.
  + Sólo se procesa en AutoFirma.

Para ver los formatos, algoritmos, parámetros de configuración del formato de firma y los almacenes de claves a configurar, consulte el manual del integrador del Cliente @firma.

En caso de finalizar correctamente, esta operación obtiene como resultado una cadena con el formato:

CERT\_B64|CONTRAFIRMA\_B64|FILENAME\_B64

En donde:

* CERT\_B64
  + Certificado de firma utilizado codificado en Base 64 URL SAFE.
* CONTRAFIRMA\_B64
  + Contrafirma generada codificada en Base 64 URL SAFE.
* FILENAME\_B64
  + Si no se indicó una firma a contrafirmar y el usuario tuvo que seleccionar un fichero, este será el nombre del fichero seleccionado codificado en Base 64 URL SAFE.
  + Este valor solo se devolverá en AutoFirma 1.7 y superiores.
  + En caso de no aparecer, tampoco lo hará el carácter separador ‘|’.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de contrafirma (en la que se omite parte del parámetro de datos por legibilidad) es:

|  |
| --- |
| afirma://countersign?op=countersign&algorithm=SHA512withRSA&format=XAdES&properties=c2VydmVyVXJsPWh0dHA6Ly9hcHBwcnVlYmE6ODA4MC9hZmlybWEtc2VydmVyLXRyaXBoYXNlLXNpZ25lci9TaWduYXR1cmVTZXJ2aWNl&sticky=false&dat=C9hZmlybWEtc2Vy…YXR1cmU-PC9wcm9qZWN0Pg== |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Firma en lote (*batch*)

Operación para la firma, cofirma o contrafirma de múltiples documentos en una única operación.

Esta operación sólo es compatible con AutoFirma.

Código de operación:

* batch

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: batch
* dat
  + XML codificado en Base64 URL SAFE con la configuración del lote de firma.
  + Se consultar el formato del XML en el manual del integrador del Cliente @firma.
* batchpresignerurl
  + URL del servicio de prefirma de lotes (disponible junto al servicio de firma trifásica).
* batchpostsignerurl
  + URL del servicio de postfirma de lotes (disponible junto al servicio de firma trifásica).
* needcert
  + Booleano para indicar si se debe devolver también en el resultado el certificado utilizado en la operación (true) o si no (false, valor por defecto).
* ksb64 (opcional)
  + Configuración del almacén de claves a utilizar codificada en base 64.
  + Sólo se procesa en AutoFirma.
* sticky (opcional)
  + Valor booleano que indica si debe recordarse el certificado de firma que se seleccione o, si ya se recuerda uno anterior, si debe usarse ese automáticamente (true); o si se debe permitir seleccionar un nuevo certificado y olvidar cualquiera que se recuerde (false).
  + Sólo se procesa en AutoFirma.
* resetsticky (opcional)
  + Valor booleano que indica si debe olvidarse cualquier certificado previamente seleccionado independientemente del valor del parámetro sticky (true), o si no (false, valor por defecto).
  + Este valor se puede utilizar junto con sticky si ya teníamos un certificado seleccionado y queremos olvidar ese y recordar el nuevo que se seleccione.
  + Sólo se procesa en AutoFirma.

Para ver los formatos, algoritmos, parámetros de configuración del formato de firma y los almacenes de claves a configurar, consulte el manual del integrador del Cliente @firma.

En caso de finalizar correctamente, esta operación obtiene como resultado una cadena con el formato:

XML\_B64|CERT\_B64

En donde:

* XML\_B64
  + XML codificado en Base 64 URL SAFE con el resultado de la firma del lote.
  + Se consultar el formato del XML en el manual del integrador del Cliente @firma.
* CERT\_B64
  + Certificado de firma utilizado codificado en Base 64 URL SAFE.
  + Este valor solo se devolverá en AutoFirma 1.7 y superiores.
  + En caso de no aparecer, tampoco lo hará el carácter separador ‘|’.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de firma de lotes (en la que por legibilidad se omite parte del parámetro de datos) es:

|  |
| --- |
| afirma://batch?op=batch&batchpresignerurl=http%3A%2F%2Fappprueba%3A8080%2Fafirma-server-triphase-signer%2FBatchPresigner&batchpostsignerurl=http%3A%2F%2Fappprueba%3A8080%2Fafirma-server-triphase-signer%2FBatchPostsigner&properties=c2VydmVyVXJsPWh0dHA6Ly9hcHBwcnVlYmE6ODA4MC9hZmlybWEtc2VydmVyLXRyaXBoYXNlLXNpZ25lci9TaWduYXR1cmVTZXJ2aWNl&sticky=false&needcert=true&dat=PD94bWwgdmVyc2lvbj0iMS4wIi…WduPgo8L3NpZ25iYXRjaD4= |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Selección de certificado (*selectcert*)

Operación para la obtención de un certificado de firma.

Código de operación:

* selectcert

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: selectcert
* ksb64 (opcional)
  + Configuración del almacén de claves a utilizar codificada en base 64.
  + Sólo se procesa en AutoFirma.
* properties (opcional)
  + Propiedades con configuración de filtros de certificados.
* sticky (opcional)
  + Valor booleano que indica si debe recordarse el certificado de firma que se seleccione o, si ya se recuerda uno anterior, si debe usarse ese automáticamente (true); o si se debe permitir seleccionar un nuevo certificado y olvidar cualquiera que se recuerde (false).
  + Sólo se procesa en AutoFirma.
* resetsticky (opcional)
  + Valor booleano que indica si debe olvidarse cualquier certificado previamente seleccionado independientemente del valor del parámetro sticky (true), o si no (false, valor por defecto).
  + Este valor se puede utilizar junto con sticky si ya teníamos un certificado seleccionado y queremos olvidar ese y recordar el nuevo que se seleccione.
  + Sólo se procesa en AutoFirma.

Para ver los formatos, algoritmos, parámetros de configuración del formato de firma y los almacenes de claves a configurar, consulte el manual del integrador del Cliente @firma.

En caso de finalizar correctamente, esta operación obtiene como resultado el certificado seleccionado codificado en Base 64 URL SAFE.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de selección de certificado es:

|  |
| --- |
| afirma://selectcert?op=selectcert&properties=ZmlsdGVycz1ub25leHBpcmVkOmZhbHNl&sticky=true |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Operación de guardado (*save*)

Operación para el guardado de datos en disco.

Esta operación sólo está soportada por AutoFirma y el Cliente de firma Android.

Código de operación:

* save

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: save
* dat
  + Datos a guardar codificados en Base64 URL SAFE.
* title (opcional)
  + Título del diálogo de guardado.
* filename (opcional)
  + Nombre propuesto para el fichero que contendrá los datos a guardar.
* exts (opcional)
  + Lista de extensiones, separadas por coma, propuestas para el fichero.
* desc (opcional)
  + Descripción del tipo de fichero.

En caso de finalizar correctamente, esta operación obtiene como resultado el valor “OK” o “OK\r\n”.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de guardado es:

|  |
| --- |
| afirma://save?op=save&title=Guardar%20firma%20electr%C3%B3nica&dat=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiID8-CjxzaWducz4KIDxzaWducmVzdWx0IGlkPSI3NzAwMzAyIiByZXN1bHQ9IkRPTkVfQU5EX1NBVkVEIiBkZXNjcmlwdGlvbj0iIi8-CiA8c2lnbnJlc3VsdCBpZD0iNzAzOTg2OSIgcmVzdWx0PSJET05FX0FORF9TQVZFRCIgZGVzY3JpcHRpb249IiIvPgogPHNpZ25yZXN1bHQgaWQ9Ijg5ODMxMjIiIHJlc3VsdD0iRE9ORV9BTkRfU0FWRUQiIGRlc2NyaXB0aW9uPSIiLz4KIDxzaWducmVzdWx0IGlkPSI3OTk1MDciIHJlc3VsdD0iRE9ORV9BTkRfU0FWRUQiIGRlc2NyaXB0aW9uPSIiLz4KIDxzaWducmVzdWx0IGlkPSIzNDc1MjYzIiByZXN1bHQ9IkRPTkVfQU5EX1NBVkVEIiBkZXNjcmlwdGlvbj0iIi8-CiA8c2lnbnJlc3VsdCBpZD0iMTk3MjI5NCIgcmVzdWx0PSJET05FX0FORF9TQVZFRCIgZGVzY3JpcHRpb249IiIvPgogPHNpZ25yZXN1bHQgaWQ9IjQxNTk2MzEiIHJlc3VsdD0iRE9ORV9BTkRfU0FWRUQiIGRlc2NyaXB0aW9uPSIiLz4KIDxzaWducmVzdWx0IGlkPSIxMzA0NTYxIiByZXN1bHQ9IkRPTkVfQU5EX1NBVkVEIiBkZXNjcmlwdGlvbj0iIi8-Cjwvc2lnbnM- |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Carga de datos (*load*)

Operación para la carga de datos.

Esta operación sólo es compatible con AutoFirma.

Código de operación:

* load

Parámetros admitidos:

* op
  + Código de operación. En este caso, será: load
* title (opcional)
  + Título del diálogo de guardado.
* filePath (opcional)
  + Ruta del directorio por defecto.
* exts (opcional)
  + Lista de extensiones, separadas por coma, propuestas para el fichero.
* desc (opcional)
  + Descripción del tipo de fichero.
* multiload (opcional)
  + Booleano que indica si se debe permitir cargar más de un fichero (true) o sólo uno (false, valor por defecto).

En caso de finalizar correctamente, esta operación obtiene como resultado los datos cargados codificados en Base 64 URL SAFE. En caso de ser una carga de múltiples ficheros, se recibirá el contenido de cada uno de ellos codificado en Base64 URL SAFE y separados por el caracter ‘|’.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de carga de datos es:

|  |
| --- |
| afirma://load?op=load&title=&exts=csig%2Cxsig%2Csig%2Cpdf%2Cxml&desc=Fichero%20de%20firma%20electr%C3%B3nica&multiload=false |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Firma y guardado (*signandsave*)

Operación para la ejecución de una firma, cofirma o contrafirma electrónica seguido del guardado del fichero resultante.

Esta operación sólo es compatible con AutoFirma.

Código de operación:

* signandsave

Parámetros admitidos:

* op
  + Código de operación. Debe ser: signandsave
* cop
  + Operación de firma a ejecutar. Puede ser sign, cosign o countersign, según se desee realizar una firma, cofirma o contrafirma, respectivamente.
* dat
  + Datos a firmar/multifirmar codificados en Base 64 URL SAFE.
  + Si no se indican, se permitirá al usuario seleccionar un fichero.
* format
  + Formato de firma.
* algorithm
  + Algoritmo de firma.
* properties
  + Propiedades de configuración específicas del formato de firma.
  + Se proporcionarán en forma cadena compuesta por duplas propiedad=valor separadas por ‘\n’. La cadena resultante se proporcionará en Base 64 URL SAFE.
* filename (opcional)
  + Nombre propuesto para el fichero de salida.
* ksb64 (opcional)
  + Configuración del almacén de claves a utilizar codificada en base 64.
  + Sólo se procesa en AutoFirma.
* sticky (opcional)
  + Valor booleano que indica si debe recordarse el certificado de firma que se seleccione o, si ya se recuerda uno anterior, si debe usarse ese automáticamente (true); o si se debe permitir seleccionar un nuevo certificado y olvidar cualquiera que se recuerde (false).
  + Sólo se procesa en AutoFirma.
* resetsticky (opcional)
  + Valor booleano que indica si debe olvidarse cualquier certificado previamente seleccionado independientemente del valor del parámetro sticky (true), o si no (false, valor por defecto).
  + Este valor se puede utilizar junto con sticky si ya teníamos un certificado seleccionado y queremos olvidar ese y recordar el nuevo que se seleccione.
  + Sólo se procesa en AutoFirma.

En caso de éxito, esta operación obtiene como resultado una cadena con el formato:

CERT\_B64|FIRMA\_B64|FILENAME\_B64

En donde:

* CERT\_B64
  + Certificado de firma utilizado codificado en Base 64 URL SAFE.
* FIRMA\_B64
  + Firma generada codificada en Base 64 URL SAFE.
* FILENAME\_B64
  + Si no se indicaron los datos a firmar y el usuario tuvo que seleccionar un fichero, este será el nombre del fichero seleccionado codificado en Base 64 URL SAFE.
  + Este valor solo se devolverá en AutoFirma 1.7 y superiores.

En caso de no aparecer, tampoco lo hará el carácter separador ‘|’.

En caso de cancelarse la operación, se obtiene como resultado la cadena “CANCEL” o “CANCEL\r\n”.

En caso de error, se obtiene una cadena que empieza por “err-”.

Un ejemplo de llamada a la operación de firma y guardado es:

|  |
| --- |
| afirma://signandsave?op=signandsave&cop=cosign&algorithm=SHA512withRSA&format=XAdES&properties=c2VydmVyVXJsPWh0dHA6Ly9hcHBwcnVlYmE6ODA4MC9hZmlybWEtc2VydmVyLXRyaXBoYXNlLXNpZ25lci9TaWduYXR1cmVTZXJ2aWNl&sticky=false&filename=cofirma.xsig |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

## Recuperación del log de la aplicación (*getLog*)

Operación para la recuperación de las trazas de ejecución de la aplicación de firma.

Esta operación sólo es compatible con AutoFirma.

Código de operación:

* getlog

Parámetros admitidos:

* op
  + Código de operación. Debe ser: getlog

En caso de finalizar correctamente, se devuelve la cadena de texto con las trazas de ejecución.

En caso de error, se obtiene una cadena que empieza por “err-”.

La llamada a la operación se realiza con la petición:

|  |
| --- |
| afirma://getLog?op=getLog |

Recuerde que, esta llamada deberá incluir los parámetros necesarios para el uso del servidor intermedio si se utiliza este mecanismo de comunicación.

1. Servidor intermedio

El servidor intermedio será un servidor proporcionado por el organismo que hace integra el uso del Cliente @firma en su aplicación y en el que se guardarán temporalmente los datos que se transfieren entre la aplicación cliente y la aplicación de firma (Cliente @firma) cuando se utilice el modo de comunicación a través de servidor intermedio.

Para el uso de este servidor se distribuye en el kit de integración del Cliente @firma dos archivos WAR para la ejecución de las operaciones de guardado y carga de datos del servidor:

* afirma-signature-storage.war
  + Archivo con el *servlet* para la subida de datos al servidor intermedio.
* afirma-signature-retriever.war
  + Archivo con el *servlet* para la descarga de datos del servidor intermedio.
  1. Guardado de datos

La subida de datos al servidor intermedio se realiza mediante la llamada a la URL del servicio de guardado junto con varios parámetros de configuración.

El nombre por defecto del servicio de guardado es: StorageService

Los parámetros necesarios son:

* op
  + Operación que se desea realizar. Se establecerá el valor: put
* v
  + Versión del servicio. Actualmente se debe configurar el valor: 1\_0
* id
  + Identificador aleatorio asignado a los datos que se almacenan.
* dat
  + Datos que se desean almacenar en base 64 URL SAFE y URL Encoded.

Por ejemplo:

http://miapp.com/afirma/afirma-signature-storage**/StorageService**?**op**=put&**v**=1\_0&**id**=000987508929&**dat**=ZnVsYW5pdG9AYXRvcy5uZXQ%3D

* 1. Recuperación de datos

La descarga de datos del servidor intermedio se realiza mediante la llamada a la URL del servicio de recuperación junto con varios parámetros de configuración.

El nombre por defecto del servicio de guardado es: RetrieveService

Los parámetros necesarios son:

* op
  + Operación que se desea realizar. Se establecerá el valor: get
* v
  + Versión del servicio. Actualmente se debe configurar el valor: 1\_0
* id
  + Identificador que se utilizó para el almacenamiento de los datos.

Por ejemplo:

http://miapp.com/ afirma/ afirma-signature-retriever**/RetrieveService**?**op**=get&**v**=1\_0&**id**=000987508929

[Creative Commons License](#Licencia_Creative_Commons)

Esta obra está bajo una licencia [Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 Unported](#Licencia_Creative_Commons).