



4identity

Guía de integración

www.4identity.eu

Bit4id Ibérica

C/ Marie Curie, 8-14
Forum Nord de Tecnología
08042 – Barcelona
España
Tel. +34 935 35 35 18
info.es@bit4id.com

Oficina Lisboa

Rua Cesário Verde, 32
2790-495 Queijas (Lisboa)
Portugal
Tel. +351 914 58 30 21
info.pt@bit4id.com

Bit4id Italia

Via Coroglio, 57
Città della Scienza
80124 Napoli
Italia
Tel. +39 081 762 56 00
info.it@bit4id.com

Oficina Milán

Corso Vercelli, 11
20144 Milano
Italia
Tel. +39 024 547 42 59
info.it@bit4id.com

Bit4id UK

2 London Wall Buildings
London Wall,
London EC2M 5UU
United Kingdom
Tel. +44 (0) 203 397 3166
info.uk@bit4id.com

Oficina Guatemala


15 avenida, 14-09 zona 10
Oakland - 01010
Guatemala
Guatemala
Tel: +502 22 21 91 63
aor@bit4id.com

Bit4id Perú

Avda. Olavegoya, nº 1835
Distrito Jesus Maria
Lima 11
Perú
Tel: +51 947 744 704
info.pe@bit4id.com




ISO 9001:2008
ISO 14001:2004
ISO 27001:2005

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity


Índice

Revisiones	3
1. Introducción	4
2. Entorno para pruebas de integración	5
3. Funcionalidades	6
3.1. Firma digital	7
3.2. Autenticación con certificado digital	9
3. Integración	11
3.1. Firma digital	11
3.1.1. Integración con PHP	11
3.1.2. Integración con .NET	14
3.1.3. Integración con Java	17
3.2. Autenticación	20
3.2.1. Integración con PHP	20
3.2.2. Integración con .NET	22
3.2.3. Integración con Java	25
3.2.4. Valores de autenticación enviados	27
4. Parámetros (tags) disponibles	30
Filtrar certificados	30
4.1. Parámetros de firma digital	30
Configuración general	30
Firma CAdES	33
Firma PAdES	34
Firma XAdES	37
Sellado de Tiempo	38
4.2. Parámetros de autenticación	38
5. Canal de 4identity	40
5.1. Estado	40
5.2. Monitorización	41
6. Diagramas de flujo	44
6.1. Diagrama de flujo de firma digital	44
6.2. Diagrama de flujo de autenticación	45
7. Errores comunes	46

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto: 4identity	

Revisiones

Revisión	Fechas	Cambio	Autor
1.0	12/06/2013	Primer borrador	CBU
1.1	12/06/2014	Revisión técnica	JGM
1.2	16/06/2014	Revisión técnica	RAV
1.3	27/08/2014	Traducción y revisión técnica	SGM
1.4	29/08/2014	Inserción "diagramas de flujo" y "Errores comunes"	JGM
1.5	09/09/2014	Revisión definitiva	JGM
1.6	30/10/2015	Licenciado, autenticación, mejoras	RAV
1.7	10/11/2015	Nuevos parámetros	RAV
1.8	05/02/2016	Reinserción de enlace as-demo	RAV
1.9	18/05/2016	Localhost	RAV
2.0	09/03/2017	Sesiones de pin y cache	RAV
2.1	06/03/2018	Actualización con nuevos parámetros y cambio en estructura del documento	SGM
2.2	11/04/2018	Se añaden nuevos parámetros	SGM
2.3	17/04/2018	Actualización de firma en lote	SGM
2.4	09/11/2018	Se corrige error de tipografía en parámetro de TS	SGM
2.5	28/11/2018	Corrección de errores	OJP
2.6	12/12/2018	Actualización de parámetro 'bit4id-document'	SGM
2.7	28/06/2019	Actualización entorno pruebas de integración	SGM


	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

1. Introducción

El objetivo de este documento es detallar el proceso de integración de 4identity en un entorno estándar. 4identity posibilita la **firma digital** y la **autenticación con certificado digital** en entornos web.

El entorno está compuesto de:

- **Aplicación Web**, que incluye dos tipos de aplicaciones:
 - ***Aplicación(es) para la firma***, permite seleccionar un documento (en función del ejemplo será firma de un documento local o en almacenamiento remoto) y comenzar con el proceso de firma mediante 4identity. Se recomienda escoger un documento PDF debido a que, una vez finalizado el proceso, resulta más fácil comprobar que la firma se ha efectuado correctamente (firma tipo PAdES, en este caso).
 - ***Aplicación para la autenticación***, permite autenticar a un usuario mediante un certificado. Devuelve el resultado de la validación.
- **4identity Client** es el encargado de las operaciones criptográficas de firma y autenticación que se instala en el terminal del usuario y que se ejecutará para guiar al usuario en el proceso de firma y autenticación.
Este componente es específico para cada cliente y puede traer habilitadas funcionalidades por defecto (en función de lo requerido). P.ej. selección automática de certificado cuando solo se dispone de uno (auto-select), vista previa, imagen de marca gráfica por defecto (en caso de no definir ninguna imagen), verificación de certificados, etc.
- **Escritorio usuario**, donde se encuentra 4identity Client y se inicia el explorador, para acceder a la aplicación web.
- **4identity server** (también conocido como **SMARTENGINE**), dicho componente controla el canal establecido entre 4identity Client y el explorador web del usuario. Además, 4identity server participa activamente en la función de autenticación pues es el responsable de validar el certificado con el que el usuario se autentica.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

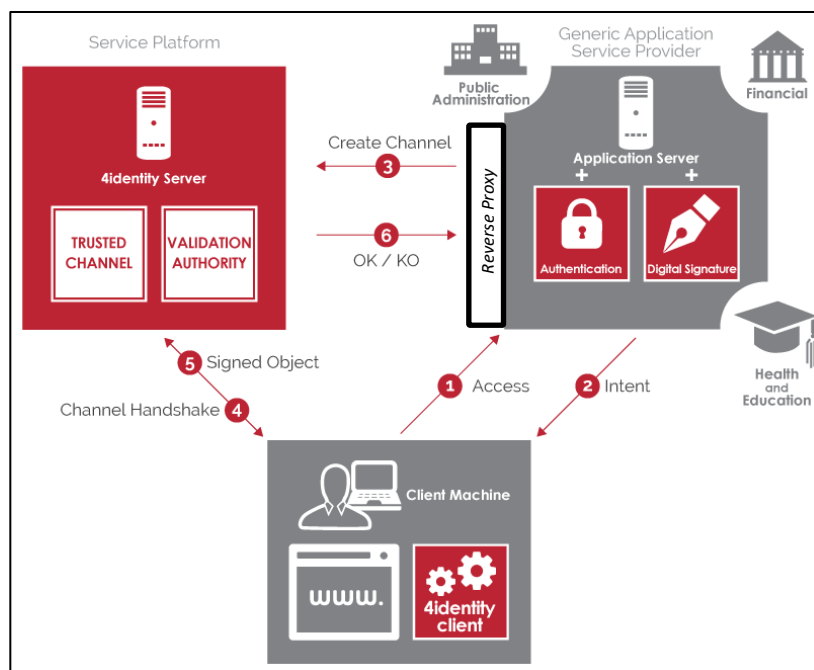


Figura 1 – Ejemplo de arquitectura de 4IDENTITY

2. Entorno para pruebas de integración

Para poder iniciar las pruebas de integración con 4identity será necesario:

A. Utilizar el **4identity Server** DEMO expuesto por BIT4ID para las pruebas con la firma y la autenticación. Para ello, el entorno de prueba debe alojarse en **LOCALHOST (127.0.0.1)** y se deben utilizar los siguientes recursos:


- **FIRMA**

<https://www.4identity.eu/smartengine/bit4id-sign.js>

- **AUTENTICACIÓN**

<https://www.4identity.eu/smartengine/bit4id-auth.js>

NOTA: El script de pruebas ofrecido en 4identity.eu solo funciona si su ambiente de desarrollo hace llamadas desde localhost. Ej.: <http://localhost/>

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

B. Ejemplos de código/páginas (PHP, Java y .NET) para firma y autenticación:

○ FIRMA

- Ejemplos de Código de las páginas **Sign** (form HTML con tags de 4identity), **Sign-OK** (recepción del archivo firmado) y **Sign-end-OK y Error** (landing pages).
- Todas estas páginas deben encontrarse en localhost.

○ AUTENTICACIÓN


- Ejemplos de Código de las páginas **Auth** (form HTML con tags de 4identity), **Auth-OK** (recepción del resultado de autenticación) y **Auth-end-OK y Error** (landing pages).
- **IMPORTANTE:** Dado que el flujo de la autenticación es diferente al de la firma, únicamente la página de **Auth** deberá encontrarse en localhost. El resto de las páginas (especialmente Auth-OK) deberán ser accesibles desde Internet, para poder recibir el resultado de la autenticación por parte del 4identity Server expuesto por BIT4ID.
- En el parámetro '**action**' de **Auth** deberá incluirse la ruta accesible desde Internet hasta **Auth-OK**.

NOTA: Los **ejemplos de código** para firma y autenticación de 4identity (PHP, JAVA y .NET) deben solicitarse al **DEP. TÉCNICO DE BIT4ID**.

3. Funcionalidades

El código que se muestra en los siguientes apartados, tiene el objetivo de guiar al desarrollador a través de las principales funcionalidades de firma y autenticación de 4identity. Este código muestra únicamente las características básicas de la herramienta, pudiéndolas implementar en escenarios más complejos.

La integración de la tecnología de 4identity con la aplicación web del cliente solo requiere de la inserción de un **simple formulario HTML** con unos **tags específicos** que 4identity es capaz de interpretar además del uso de una API HTTP Restful dentro de los flujos soportados. La integración de 4identity concluye con la inserción de un **script alojado en 4identity server** para que este pueda gestionar el canal 4identity

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

Client – Navegador Web. Una vez firmado el documento o autenticado el usuario, la aplicación web recibirá el documento firmado o la respuesta de la validación vía API HTTP Restful (método POST) y tras ello 4identity habrá finalizado su función por lo que se le devolverá el control a la aplicación web.

A continuación, se mostrará el flujo funcional juntamente con el código requerido, para su integración de los procesos de firma y autenticación.

3.1. Firma digital

Los principales componentes involucrados en el proceso de firma son la **aplicación web**, que contiene el código que gestiona el archivo a firmar y **4identity Client** que firma el archivo. La funcionalidad más destacable de 4identity server en este proceso consiste en mantener el ciclo de vida del canal 4identity Client-Explorador web, mediante el uso de cookies de sesión, que expiran luego de cada operación de firma o autenticación.


A modo práctico, basta con tener claro que la aplicación web es capaz de seleccionar el archivo a firmar y de recibirlo firmado mediante un mensaje POST, que se genera en 4identity Client. Seguidamente, la aplicación almacena el archivo en un servidor.

Para que la aplicación web reciba la respuesta de 4identity Client es imprescindible que esta se exponga públicamente (accesible desde fuera de la red local).



Figura 2 – Proceso de firma

En el proceso de firma, 4identity Client muestra al usuario los certificados disponibles para firmar:

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

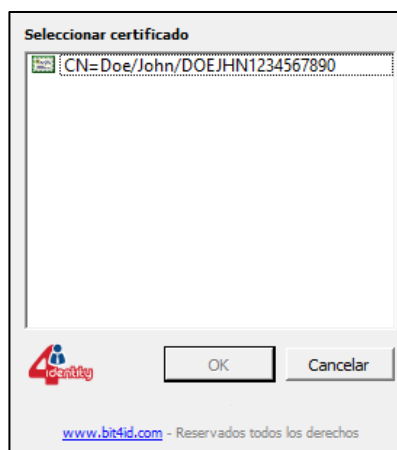


Figura 3 – Selección del certificado con el que llevar a cabo la firma (1)

Seguidamente muestra una vista previa del documento a firmar,

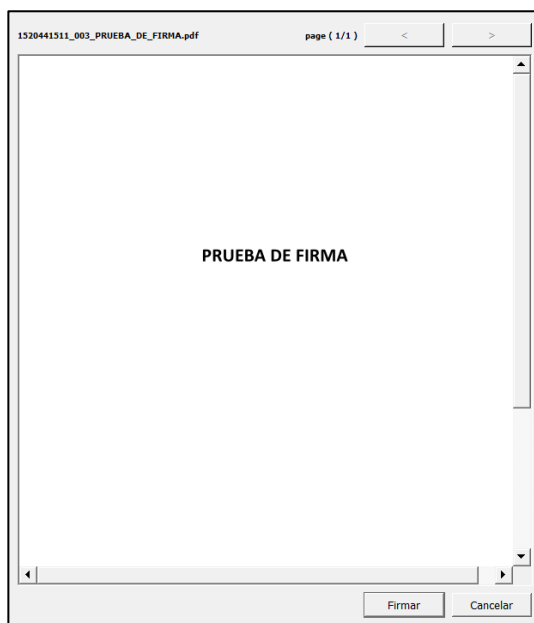



Figura 4 – Vista previa del documento y aprobación (2a)

y comienza el proceso de firma. En el caso en que el documento no fuese de tipo PDF, aparecería el siguiente mensaje en el cual se verifica que el usuario ha revisado el documento. Después de la verificación, se habilita el botón que permite llevar a cabo la firma.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

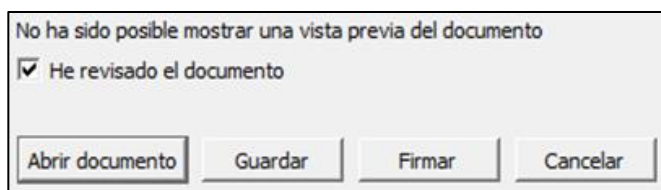


Figura 5 – Aprobación (2b)

Para acceder a las claves asociadas al certificado, 4identity solicita la usuario el PIN de protección.

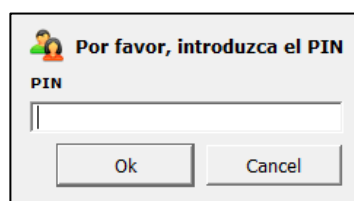


Figura 6. Inserción del PIN por parte del usuario (3)

3.2. Autenticación con certificado digital


Los componentes involucrados en el proceso de autenticación son la **aplicación web** que define el reto (o challenge, cadena de caracteres a firmar) con 4identity Client. **4identity Client** que, usando el certificado seleccionado por el usuario, firma el reto. **4identity Server**, que valida el certificado a partir del reto firmado. Éste devolverá a la aplicación web información acerca del resultado de la autenticación.

Para que la aplicación web reciba la respuesta de 4identity Client es imprescindible que esta se exponga públicamente (accesible desde fuera de la red local).



Figura 7. Proceso de autenticación

De forma análoga a la firma, cuando se inicia el proceso el usuario selecciona el certificado a utilizar para la autenticación.

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

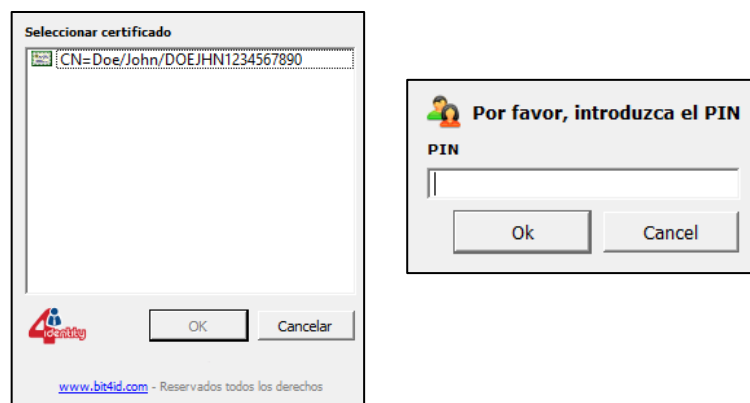


Figura 8. Selección del certificado (1) e inserción de PIN

4identity Client, una vez firmado el reto, lo envía al 4identity Server para validar el certificado. El reto firmado es necesario para asegurar la posesión del par de claves por parte del usuario que se está autenticando.

El resultado final del proceso de autenticación se compone de diferentes variables POST que muestran y dan información del resultado. La variable POST más significativa e importante tiene el nombre de **"result"**, que puede adquirir los valores **ok** o **ko** seguidos de información del certificado validado.

La siguiente cadena de caracteres es un ejemplo del valor que la variable "result" (*atributo "result" del POST*) puede devolver:


result = **ok: Doe/John/DOEJHN1234567890**

Donde se puede ver que, el texto inmediatamente después de **ok** es el **CN del certificado**. Esto significaría una autenticación correcta.

Consultar el apartado **4.2** para ver la descripción de los parámetros de autenticación enviados.

Aunque el código explicado en los siguientes apartados solo verifique el valor del parámetro **"result"**, se debe seguir el siguiente procedimiento para garantizar el máximo grado de seguridad:

1. El reto debe ser cambiado en cada sesión y verificado por el servidor (aplicación web) en el proceso de autenticación.
2. El parámetro de clave secreta (secret key o sk) debe ser verificado por el servidor en el proceso de autenticación.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

3. Integración

En ambas funcionalidades la aplicación web consta básicamente de los mismos elementos/páginas:

1. Una **página principal** que contiene un FORM (formulario) HTML que permite introducir los **parámetros de la firma/autenticación**. Este formulario tendrá asociado un botón "Firma" o "Autenticación" (input del tipo "submit"). En el caso de la autenticación esta página puede conformar la página de *login*.
2. Un **fichero web** que será el encargado de **recibir el archivo firmado** (atributo **"attach"** del POST) o la **respuesta del proceso de autenticación** (atributo **"result"** del POST). En este fichero web, se espera recibir el documento/resultados de la autenticación vía POST. Tras esto se espera una redirección para finalizar el proceso de firma/autenticación y enviar al usuario de nuevo a la aplicación web (con esto concluye la función de 4identity se devuelve el control a la aplicación web).
3. Una **página final** llamada **"landing page"** en los ejemplos del SDK, que muestra al usuario el resultado del proceso llevado a cabo. En el caso de la firma ésta puede mostrar un enlace para la descarga del fichero firmado. En la autenticación podría comportar el inicio de sesión en la plataforma (en caso de una autenticación correcta).

A continuación, se detallan los ejemplos de código que componen el SDK de 4identity divididos por lenguaje de programación. Cabe destacar que 4identity puede ser integrado con cualquier tecnología web. Se han seleccionado aquí tres de las tecnologías más comunes que son **JAVA**, **.NET** y **PHP**.


3.1. Firma digital

3.1.1. Integración con PHP

Para entornos PHP la solución se compondrá de:

- a) Un servidor web (comúnmente Apache) p.ej. **as.example.com:8080** que contiene la aplicación personalizada **/4identity** ;

El nombre definitivo dependerá de la configuración del servidor de aplicaciones. Por esta razón suponemos que **/4identity** será la carpeta donde se alojará la Aplicación Web de ejemplo.

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

b) /4identity se compone de:

- **index.php**: que contiene el formulario y la llamada al script ;
- Una página llamada **sign-ok.php** que lee el POST (*atributo "attach" del POST*) y redirige la sesión a la *landing page* **sign-end-ok.php**.
- **sign-end-ok.php** que muestra al usuario el archivo firmado;
- Una página llamada **sign-end-error.php**. La sesión se redirigirá hacia esta *landing page* si el proceso de firma ha experimentado algún tipo de error.

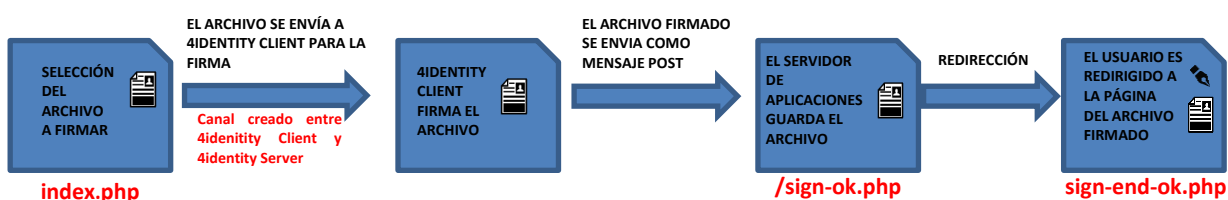


Figura 9. Proceso de firma con PHP

Index.php

A continuación aparece el código de la página principal **index.php** (ejemplo de firma de archivo local):

```


<html>
  <head>
    <meta charset="utf-8" />
    <title>Firma Digital de Múltiples documentos en local</title>
  </head>
  <body>
    <?php
      // $folders=Lista de carpetas necesarias para llegar al fichero sign-ok.php. Tiene que ser el
      // path absoluto de la url.
      // $documentName=Nombre del documento a firmar.
      // $documentID=Identificador del documento. Sirve para relacionar el proceso dado que se
      // envía por método POST al destino.
      // $tipo=Tipo de firma a realizar. Puede ser PAdES, XAdES y CAdES.
      // $url_4identity_server_sign=Url del script con las funciones del 4identity Server.

      $folders="demos_4identity/firmar_mult_docs_local/";
      $documentName="multiples_documentos_local";
      $documentID="_mults_local";
      $tipo="PAdES";
      $url_4identity_server_sign="https://www.4identity.eu/smartengine/bit4id-sign.min.js";

      ?>

      <form class="bit4id-sign" method="post" action="<?php echo $folders; ?>sign-ok.php">
      <div class="bit4id-signReq">
      <div style="visibility: visible" class="bit4id-documentName"><?php echo $documentName; ?></div>
      <div style="visibility: visible" class="bit4id-documentID"><?php echo $documentID; ?></div>
      <div class="bit4id-localFile">YES</div>
    </form>
  </body>
</html>

```

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

```

<div class="bit4id-message">Seleccionar los ficheros a firmar</div>
<div class="bit4id-caption">Seleccione los ficheros a firmar</div>
<div class="bit4id-signatureType"><?php echo $tipo; ?></div>
</div>

<div id="bit4id-status">loading</div>
<input type="submit" disabled="disabled" value="Firmar" />

</form>
<script src="<?php echo $url_4identity_server_sign; ?>"></script>
</body>
</html>

```

En este ejemplo se firma un archivo seleccionado de forma local antes de iniciar el proceso de firma.

Esta página contiene el FORM con la clase personalizada **bit4id-sign**, un "action" (con method="post") que redirige a la página **sign-ok.php** y el botón de tipo submit:

```

<form class="bit4id-sign" method="post" action="<?php echo $folders; ?>sign-ok.php">

[... ]

<input type="submit" disabled="disabled" value="Firmar" />

</form>

```

Gracias al resto de etiquetas del formulario, determinamos la solicitud de firma con la clase **bit4id-signReq**:

- El nombre del archivo, que se pasará por variable ('documentName'):

```
<div style="visibility: visible" class="bit4id-documentName"><?php echo $documentName; ?></div>
```

- Identificador del documento. Sirve para relacionar el proceso dado que se envía por método POST al destino. Especificará si se trata de un archivo online o local, y si la selección es simple o múltiple. También se obtiene mediante variable ('documentID'):

```
<div style="visibility: visible" class="bit4id-documentID"><?php echo $documentID; ?></div>
```

- Etiqueta que determina que el documento a firmar se obtendrá mediante un explorador de archivos local:


```
<div class="bit4id-localFile">YES</div>
```

Con esta etiqueta definida en YES, se incluyen también dos posibles parámetros:

```
<div class="bit4id-message">Seleccionar los ficheros a firmar</div>
```

Muestra el mensaje que aparece en la parte superior del explorador.

```
<div class="bit4id-caption">Seleccione los ficheros a firmar</div>
```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

Que muestra el mensaje que aparecerá en el botón para abrir el explorador de archivos.

- El estándar de firma (PAdES). En este caso se obtiene mediante la variable '*tipo*', definida previamente:

```
<div class="bit4id-signatureType"><?php echo $tipo; ?></div>
```

- El botón de entrega se encuentra **disabled**, ya que se habilita automáticamente una vez que se establece la conexión con 4identity Server:

```
<input type="submit" disabled="disabled" value="Firmar" />
```

- El script en 4identity Server, encargado de establecer la comunicación Explorador-4identity Client. La URL del script se obtiene mediante la variable ('*url_4identity_server_sign*') definida previamente:

```
<script src="<?php echo $url_4identity_server_sign; ?>"></script>
```

<https://www.4identity.eu/smartengine/bit4id-sign.js>, que es el valor de la variable '*url_4identity_server-sign*', es el 4identity Server expuesto por Bit4id para las pruebas de integración.

Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.


3.1.2. Integración con .NET

Para entornos .NET, la solución se compondrá de:

- a) un servidor web IIS **as.example.com:9090** que contiene la aplicación .NET en **/4identity** que es el ejemplo;

Se asume que /4identity será la carpeta donde se alojará la Aplicación Web de ejemplo.

- a) /4identity se compone de:
 - **Sign.aspx**: contiene el formulario y la llamada al script;

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

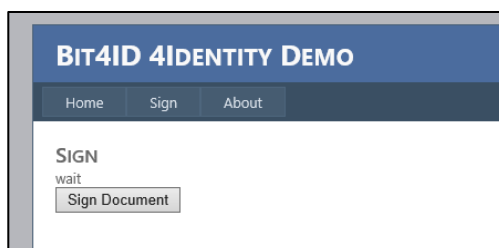


Figura 10. Página principal Sign.aspx

- Un archivo .aspx (Active Server Pages) llamado **SignOk.aspx**, que recibe el POST (atributo **"attach"** del POST) y redirige la sesión a la landing page llamada **landing.aspx**;
- **landing.aspx** muestra al usuario el resultado final y la posibilidad de descargar el archivo firmado.

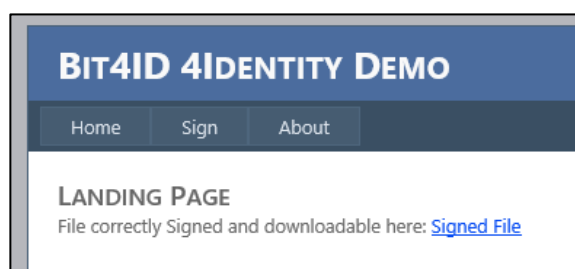


Figura 11. Página final en .NET

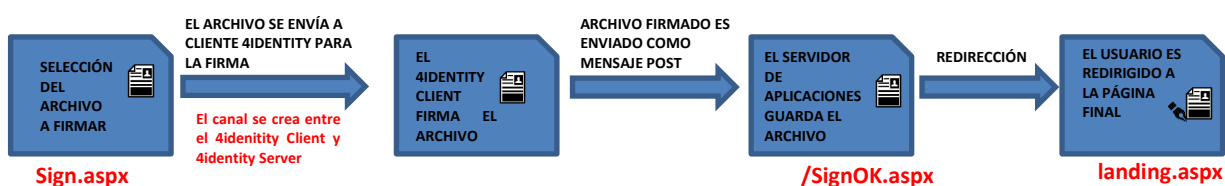



Figura 12. Proceso de firma en .NET

Sign.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Sign.aspx.cs"
Inherits="SmartEngineDemo.Sign" %>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>4Identity Authentication</title>
</head>
<body>
```

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

```

<form class="bit4id-sign" method="post" action="/identity/SignOk.aspx">
  <div class="bit4id-signReq" style="display: none;">
    <div class="bit4id-localFile">NO</div>
    <div class="bit4id-document">http://192.168.187.41:8080/identity/Documents/test.pdf</div>
    <div class="bit4id-documentName">test.pdf</div>
    <div class="bit4id-documentID">01</div>
    <div class="bit4id-signatureType">PAdES</div>
  </div>
  <div id="bit4id-status">wait</div>
  <input disabled="disabled" type="submit" value="Sign Document" />
</form>
<script src="http://dominio.licencia.com /smartengine/bit4id-sign.min.js"></script>

</body>
</html>

```

Esta página contiene el FORM (formulario) con la clase personalizada **bit4id-sign**, un "action" (con method="post") que redirige a nuestro a la página personalizada **SignOk.aspx** y el botón de tipo submit:

```

<form class="bit4id-sign" method="post" action="/4identity/SignOk.aspx">

  [...]

  <input disabled="disabled" type="submit" value="Sign Document" />

</form>

```

Gracias al resto de etiquetas del formulario, definiremos la solicitud de firma con la clase **bit4id-signReq**:

- La dirección del documento, en este caso un archivo (test.pdf) almacenado en el servidor **as.example.com** :

```
<div class="bit4id-document">http://as.example.com:9090/4identity/Documents/test.pdf</div>
```

- El nombre del archivo:

```
<div class="bit4id-documentName">test.pdf</div>
```

- El estandard de firma(PAdES):


```
<div class="bit4id-signatureType">PAdES</div>
```

- El algoritmo de firma utilizado (RSASHA256) :

```
<div class="bit4id-signingAlgorithm">RSASHA256</div>
```

- El atributo del certificado (CN) mostrado en 4identity Client cuando se listen los certificados:

```
<div class="bit4id-certInfo">CN</div>
```


	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

- La información del estado del canal. Utilizado normalmente con propósitos de depuración. Si esta etiqueta se añade en el formulario, index.html mostrará el estado de 4identity desde el inicio de la comunicación hasta finalizar el proceso de firma:

```
<div id="bit4id-status"></div>
```

- El botón de acción firma/autenticación para comenzar el proceso, luego de hacer clic o ejecutar automáticamente se establece la conexión con 4identity Server:

```
<input disabled="disabled" type="submit" value="Sign Document" />
```

- El script en 4identity Server, encargado de establecer la comunicación Explorador-4identity Client:

```
<script src="http://dominio.licencia.com/smartengine/bit4id-sign.min.js"></script>
```

○

```
<script src="http://as-demo.bit4id.org/smartengine/bit4id-sign.min.js"></script>
```

○

```
<script src="https://www.4identity.eu/smartengine/bit4id-sign.js"></script>
```


NOTA: El script de 4identity.eu solo funciona si su ambiente de desarrollo hace llamadas desde localhost ej: 'http://localhost/'

Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.

3.1.3. Integración con Java

Para entornos con Java EE la solución se compondrá de:

- Un servidor de aplicaciones **as.example.com:8080** que contiene la aplicación web **/4identity**;
- /4identity, que es el ejemplo de código JAVA del SDK, se compone de:
 - index.html**: que contiene el formulario y la llamada al script ;

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

- un servlet llamado **Signing** que lee el POST (*atributo "attach" del POST*) de los datos y redirige la sesión a la *landing page* llamada **success.jsp**;
- una página JSP llamada **success.jsp** que muestra al usuario el archivo firmado;

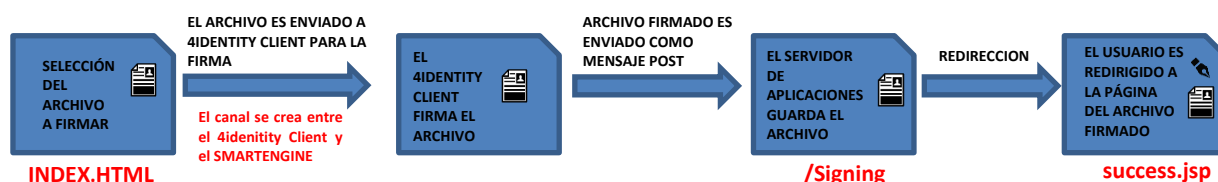


Figura 13. Proceso de firma en entorno Java

INDEX.HTML

A continuación se muestra el código de la página **index.html**.

```
<div>
  <form class="bit4id-sign" action="4identity/Signing" method="post">

    <div class="bit4id-signReq" style="display: none;">
      <div class="bit4id-document">http://as.example.com:8080/4identity/TestData.pdf</div>
      <div class="bit4id-documentName">TEST PDF DOCUMENT</div>
      <div class="bit4id-signatureType">PAdES</div>
      <div class="bit4id-signingAlgorithm">RSASHA256</div>
      <div class="bit4id-certInfo">CN</div>
    </div>


    <div>
      <fieldset>
        <div><h3>Document Signature</h3></div>
        <div><p><strong>Proceeding the document TestData.pdf will be signed, are you
sure?</strong></p></div>
        <div id="bit4id-status"></div>
        <div><input type="submit" value="Sign Document" name="cmd" disabled></div>
      </fieldset>
    </div>
  </form>
  <script src="http://fe.example.com:8082/smartengine/bit4id-sign.min.js"></script>
</div>
```

Esta página contiene el FORM con la clase personalizada **bit4id-sign**, un "action" (con method="post") que redirige a nuestro servlet **Signing** y el botón de tipo submit:

```
<form class="bit4id-sign" action="4identity/Signing" method="post">

[...]
```

```
<div><input type="submit" value="Sign Document" name="cmd" disabled></div>
```

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

```
</form>
```

Gracias al resto de etiquetas del formulario, definiremos la solicitud de firma con la clase **bit4id-signReq**:

- La dirección del documento, en este caso un archivo (TestData.pdf) almacenado en el servidor **as.example.com** :

```
<div class="bit4id-document">http://as.example.com:8080/4identity/TestData.pdf</div>
```

- El nombre del archivo:

```
<div class="bit4id-documentName">TestData.pdf</div>
```

- El estándar de firma (PAdES):

```
<div class="bit4id-signatureType">PAdES</div>
```

- El algoritmo de firma utilizado (RSASHA256) :

```
<div class="bit4id-signingAlgorithm">RSASHA256</div>
```

- El atributo del certificado (CN) que 4identity Client mostrará cuando se listen los certificados:

```
<div class="bit4id-certInfo">CN</div>
```

- La información del estado del canal. Utilizado normalmente con propósitos de depuración. Si esta etiqueta se añade en el formulario, index.html mostrará el estado de 4identity desde el inicio de la comunicación hasta finalizar el proceso de firma.

```
<div id="bit4id-status"></div>
```

- El botón que permite enviar el POST. El nombre ha de tener el valor **cmd** y tiene que permanecer habilitado el parámetro **disabled**:

```
<div><input type="submit" value="Sign Document" name="cmd" disabled></div>
```


- El script en 4identity Server, encargado de establecer la comunicación Explorador-4identity Client:

```
<script src="http://dominio.licencia.com/smartengine/bit4id-sign.min.js"></script>
```

○

```
<script src="http://as-demo.bit4id.org/smartengine/bit4id-sign.min.js"></script>
```

○

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

`<script src="https://www.4identity.eu/smartengine/bit4id-sign.js"></script>`

NOTA: El script 4identity.eu solo funciona si su ambiente de desarrollo hace llamadas desde localhost ej: <http://localhost/>

Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.

3.2. Autenticación

3.2.1. Integración con PHP

Para un entorno con PHP, utilizaremos:

- Un servidor de aplicaciones **as.example.com:8080** que contiene y pone a disposición de los usuarios el componente **/4identity** ;

El nombre definitivo dependerá de la configuración del servidor de aplicaciones. Por esta razón suponemos que **/4identity** será la carpeta donde se alojará la Aplicación Web de ejemplo.

- /4identity** se compone de:

- **index.php**: que contiene el formulario y la llamada al script.
- Una página llamada **auth-ok.php** que lee el POST (*atributo "result" del POST*) de datos y toma la decisión basándose en éstos. Esta página redirige la sesión hacia una *landing page* llamada **auth-end-ok.php** si la autenticación ha sido correcta.
- Una página llamada **auth-end-ok.php** que muestra los resultados del proceso de autenticación, dejando ver los datos recibidos:

Challenge: 4IDENTITYCH

Result: ok

SK: 5787237843

Issuer: CN=Bit4ID - TEST CA, C=IT, S=NA, L=Napoli, O=Bit4ID


Subject: 2.5.4.46=619, 2.5.4.4=ROSSI, 2.5.4.42=MARIO, CN=MARIO ROSSI, O=Bit4id, C=IT

Serial: 026b

Not_Before: Thu, 08 Nov 2012 09:34:58GMT

Not_After: Sun, 06 Nov 2022 09:34:58GMT

Certificate: MIIDiTCCA vKgAwIBAgI CAmSwDQYJKoZIhvcNAQ....

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

- Una página llamada **auth-end-error.php** que nos muestra un mensaje informándonos que ha habido un error en la autenticación.

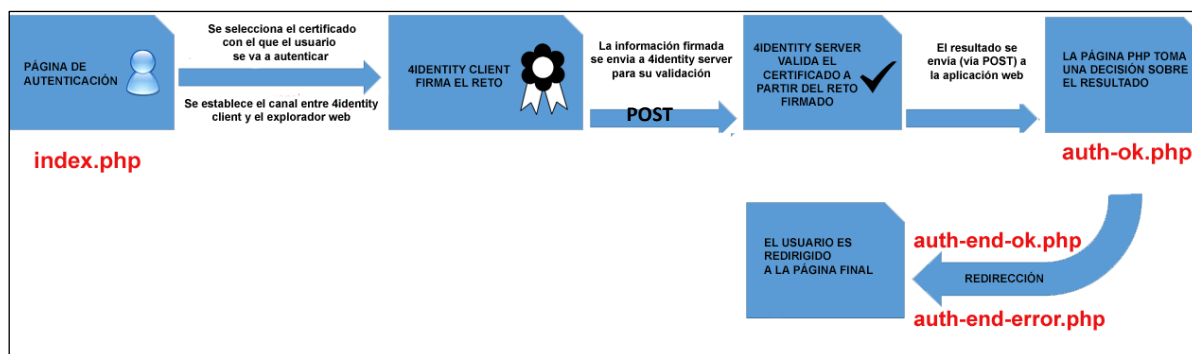


Figura 14. Proceso de autenticación en PHP

index.php

A continuación, se muestra el código de la página **index.php**.

```

<html>
<body>
<div>

    <?php
        $folders="demos_4identity/autenticacion/";
        $url_4identity_server_auth="http://dominio.licencia.com/smartengine/bit4id-auth.min.js";
        $secure_code="4sCCvGMx8PaxuYQ==";

    ?>

    <form class="bit4id-auth" action="<?php echo $folders; ?>auth-ok.php" method="post">
        <div class="bit4id-authReq" style="display: none;">
        <div class="bit4id-challenge"><?php echo $secure_code; ?></div>
        <div class="bit4id-certType">ANY</div>
        <div class="bit4id-issuerFilter"></div>

        <div id="bit4id-status">Espere hasta la conexi&oacute;n</div>
        <input type="submit" disabled="disabled" value="Autenticar" name="cmd">

    </form>
    <script src="<?php echo $url_4identity_server_auth; ?>"></script>

</div>
</body>
</html>


```

Esta página contiene el FORM con la clase personalizada **bit4id-auth**, un "action" (con method="post") que redirige a la página **auth-ok.php** y el botón de tipo submit:

```

<form class="bit4id-auth" action="<?php echo $folders; ?>auth-ok.php" method="post">

[...]
```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

Seguidamente, necesitamos crear la solicitud de autenticación mediante la clase **bit4id-authReq**. La solicitud incluye la siguiente información:

- El reto intercambiado con 4identity Client. El reto puede ser alfanumérico y no tiene límite de caracteres:

```
<div class="bit4id-challenge">4IDENTITYCH</div>
```

- El tipo de certificado:

```
<div class="bit4id-certType">ANY</div>
```

- El atributo del certificado (CN en este caso) que 4identity Client mostrará cuando se listen los certificados:

```
<div class="bit4id-certInfo">CN</div>
```

- El botón que permite el envío de los datos del formulario:

```
<input type="submit" value="Authenticate" />
```

- El script en 4identity Server, encargado de la validación del certificado. La URL del script se obtiene mediante la variable ('url_4identity_server_auth') definida previamente:

```
<script src="<?php echo $url_4identity_server_auth; ?>"></script>
```

Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.

3.2.2. Integración con .NET


Para un entorno con .NET, utilizaremos:

- a) Un servidor web (comúnmente Apache) **as.example.com:8080** que contiene la aplicación personalizada **/4identity** ;

El nombre definitivo dependerá de la configuración del servidor de aplicaciones. Por esta razón suponemos que **/4identity** será la carpeta donde se alojará la Aplicación Web de ejemplo.

- b) El componente 4identity Server está compuesta por:

- **login.aspx**: que contiene el formulario y la llamada al script.
- Una página llamada **AuthOK.aspx** que lee el POST de datos y toma la decisión basándose en éstos. Esta página redirige la sesión hacia una landing page llamada **landingauth.aspx**.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

- Una página llamada **landingauth.aspx** que muestra los resultados del proceso de autenticación.

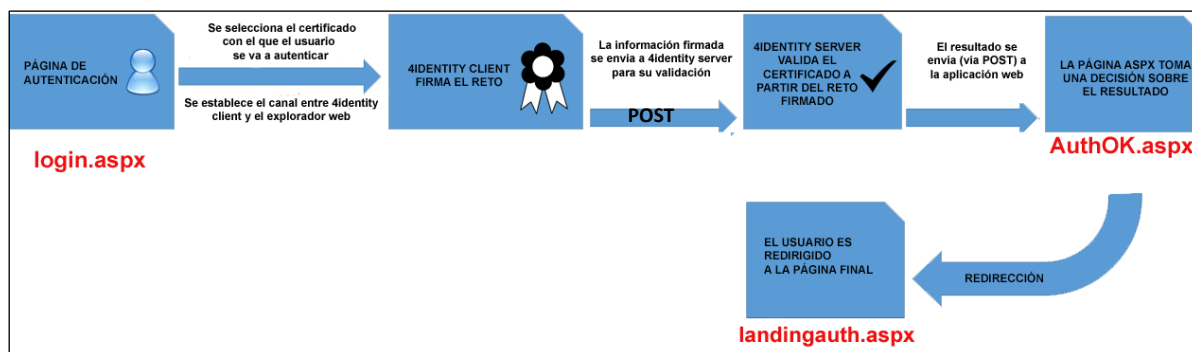


Figura 15. Proceso de autenticación en .NET

login.aspx

A continuación, se muestra el código de la página principal **login.aspx**.

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="login.aspx.cs"
Inherits="SmartEngineDemo.login" %>

<!DOCTYPE html>
<html>

<head>
<meta charset="ISO-8859-1">
<title>4Identity Authentication</title>
</head>

<body>
<form class="bit4id-auth" method="post" action="/4identity/AuthOk.aspx">
  <div class="bit4id-authReq" style="display: none;">
    <div class="bit4id-challenge">4IDENTITYCH</div>
    <div class="bit4id-certType">ANY</div>
    <div class="bit4id-certInfo">CN</div>
  </div>
  <input type="submit" value="Authenticate" />
</form>

<script src="http://fe.example.com:8082/smartengine/bit4id-auth.min.js"></script>

</body>
</html>

```


Esta página contiene el FORM con la clase personalizada **auth-sign**, un "action" (con method="post") que redirige a la página **authok.php** y el botón de tipo submit:

```

<form class="bit4id-auth" action="4identity/AuthOK.aspx" method="post">

  [...]

```

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

```
</form>
```

Seguidamente, necesitamos crear la solicitud de autenticación mediante la clase **bit4id-authReq**. La solicitud incluye la siguiente información:

- El reto intercambiado con 4identity Client. El reto puede ser cualquier texto o cifra:

```
<div class="bit4id-challenge">4IDENTITYCH</div>
```

- El tipo de certificado:

```
<div class="bit4id-certType">ANY</div>
```

- El atributo del certificado (CN en este caso) que 4identity Client mostrará cuando se listen los certificados:

```
<div class="bit4id-certInfo">CN</div>
```

- El botón que permite el envío de los datos del formulario:

```
<input type="submit" value="Authenticate" />
```

- El script en 4identity Server, encargado de la validación del certificado:


```
<script src="http://dominio.licencia.com/smartengine/bit4id-auth.min.js"></script>
```

Como ya se ha comentado, la página **AuthOK.aspx** recibe y gestiona el mensaje POST (resultados recibidos) y toma una decisión basándose en el valor de la variable obtenida **"result"**.

Como ya hemos visto, el código principal está formado por una extracción del valor de la variable **"result"** que pueden ser **"ok"** o **"ko"**.

Los valores recibidos **sk** y **challenge** también deben ser gestionados. El valor de **sk** se especifica en un archivo de configuración, mientras que el reto (**challenge**) debe ser cambiado de forma aleatoria en cada solicitud y comparado/verificado por el servidor.

Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

3.2.3. Integración con Java

Para un entorno de desarrollo en J2EE:

- Un servidor de aplicaciones **as.example.com:8080** que contiene el componente **/4identity** ;

El nombre definitivo dependerá de la configuración del servidor de aplicaciones. Por esta razón suponemos que **/4identity** será la carpeta donde se alojará la Aplicación Web de ejemplo.

- la aplicación 4identity Server consiste en:

- **login.html**: que contiene el formulario y la llamada al script;
- un servlet llamado **Auth** que lee el POST de datos y toma la decisión basándose en éstos. Esta página redirige la sesión hacia una landing page llamada **success.jsp** .
- una página llamada **success.jsp** que muestra los resultados del proceso de autenticación.

Output: **ok**

User: **Doe/John/DOEJHN1234567890**

Secret Server Code: **5787237843**

Challenge: **4IDENTITYCH**

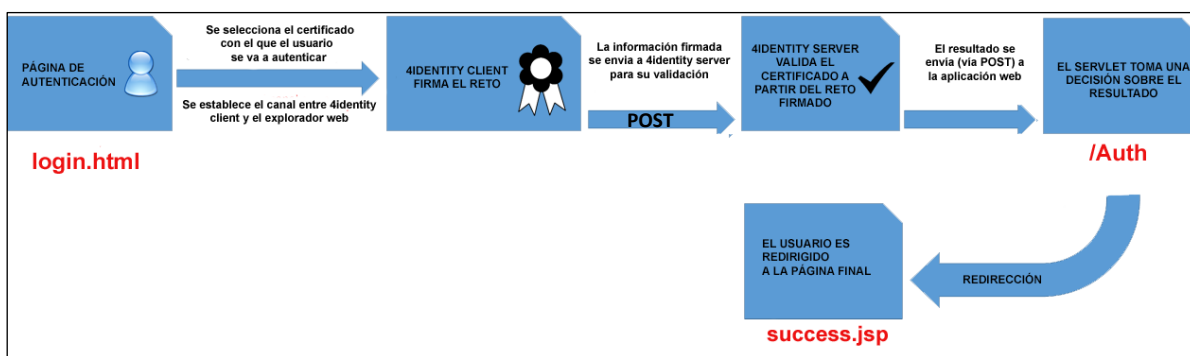



Figura 16. Proceso de autenticación en JAVA

Tal y como se detalla anteriormente, en el caso de los ejemplos del SDK de 4identity la página final es siempre (sea cual sea el certificado) **success.jsp** y en esta se mostrará el resultado de la validación por parte del 4identity Server.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

LOGIN.HTML

A continuación se muestra el código para la página **login.html**.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>4Identity Authentication</title>
</head>
<body>
<form class="bit4id-auth form-stacked" method="post" action="4identity/Auth">
<div class="bit4id-authReq" style="display: none;">
  <div class="bit4id-challenge">4IDENTITYCH</div>
  <div class="bit4id-certType">ANY</div>
  <div class="bit4id-certInfo">CN</div>
</div>
<input type="submit" value="Authenticate" />
</form>
<script src="http://dominio.licencia.com/smartengine/bit4id-sign.min.js"></script>
</body>
</html>
```

Esta página contiene el FORM (formulario) con la clase personalizada **bit4id-auth**, la acción configurada hacia la página **Auth** y el método con el valor **POST**:

```
<form class="bit4id-auth" action="4identity/Auth" method="post">

  [...]

</form>
```

Después necesitamos llevar a cabo la solicitud de autenticación mediante la clase **bit4id-authReq**. La solicitud incluye la siguiente información:

- El reto intercambiado con 4identity Client. El reto puede ser alfanumérico y no tiene límite de caracteres:

```
<div class="bit4id-challenge">4IDENTITYCH</div>
```

- El tipo de certificado que mostrará 4identity Client (filtro de certificados):


```
<div class="bit4id-certType">ANY</div>
```

- El atributo del certificado (CN en este caso) que 4identity Client mostrará cuando se listen los certificados:

```
<div class="bit4id-certInfo">CN</div>
```

- El botón que permite el envío de los datos:

```
<input type="submit" value="Authenticate" />
```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

- El script en 4identity Server, encargado de la validación del certificado:

```
<script src="http://dominio.licencia.com/smartengine/bit4id-auth.min.js"></script>
O
<script src="http://as-demo.bit4id.org/smartengine/bit4id-sign.min.js"></script>
```

Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.

El código anterior muestra un servlet que controla un mensaje POST y toma la decisión basándose en el el valor de la variable recibida "**result**". Los valores de esta variable pueden ser "**ok**" o "**ko**". Éstos gestionarán el proceso de autenticación. Esta parte del código debe ser creada dependiendo de los requerimientos de la aplicación a integrar, por ejemplo se puede utilizar en una aplicación que permita al usuario la identificación y el acceso. Por todo ello, dependiendo del valor de esta variable, podremos dar la autenticación como válida o como errónea.


Para obtener más información acerca de los parámetros que pueden ir incluidos en el FORM consultar sección '**4. Parámetros (tags) disponibles**'.

3.2.4. Valores de autenticación enviados

A continuación, aparecen todos los parámetros enviados desde **4identity Server** hacia la aplicación web. El formato de la petición POST:

EJEMPLO DE AUTENTICACIÓN CORRECTA

```
result (ok: Doe/John/DOEJHN1234567890)
sk (5787237843)
not_before (Mon, 03 Sep 2012 15:30:28GMT)
certificate (MIID1jCCAz
gAwIBAgICAKQwDQYJKoZIhvcNAQELBQAwVzEPMA0GA1UEChMGQmI0NEIEMQ8wDQYDVQQHEwZ
OYXBvbGkxZCZAJBgNVBAGTAk5BMQswCQYDVQQGEwJJVDEZMBcGA1UEAxMQMmI0NEIEMQ8wDQY
CBDBQTAeFw0xMjA5MDMxNTMwMjhaFw0yMjA5MDExNTMwMjhaMHwxZCZAJBgNVBAYTAklUMQ8wDQY
YDVQQKEwZCaXQ0aWwQxIjAgBgNVBAMTGURvZS9Kb2huL0RPRUpITjEyMzQ1Njc4OTAxDTALBgNVBCoT
BEpvaG4xDDAKBgNVBAQTA0RvZTEbMBkGCsQGSIB3DQEJARYMamRvZUB0ZXN0LmI0MmI0BjANBgkqhki
G9w0BAQEFAAOCAQ8AMIBCGKCAQEAvE6LA1SAdKNUTEd3z4hTXkAz8hikcWGEyXKpKqC4eegSPYXx
hBDRfGES8xa/TG0UCQkt2j0Bzh595aUwMr4EavtqsEy03NOI0yY5ROTI4Oxcv5HwV
QAmD34z9mcIVLO1dgichEyj1hKchdZ1UguLTnUcnlrwJl5sKTVWufUpyxlyEQQs22
AdEjU40bU6jQFFADf4ks30ch3DAFF4tKPVBvha1OObb43hxM6a3p5AmWJAyJ/LOJ8v2WYyr9lyUemRW
yuANQUaWwXVgj4rPiUqN6fx79G5rRvOoSvhwleSXEsvRvdsVeccqXZ17HRwPbQup1CjIzQmjVHFPNfwID
AQAB04IBBjCCAQlwKQYJKwYBBAAGCNxQCBBwGgBTAG0AYQByAHQAYwBhAHIAZABVAHMAZQBy
MA4GA1UdDwEB/wQEAwIFoDAPBgNVHSEUelAgBggrBgEFBQcDAGYIKwYBBQUHAWwQGCisGAQQBgjc
UAglwCQYDVROSBAlwADAfBgNVHSMEGDAWgBTmZwDoKVcTR9QJAoNI82/yq4LrsA2BgIghkgBhvC
AQ0EKRYnQmI0NGIKIC0gQXV0aGVudGJlYXRpb24gVGZvdCBDcnRpZmljYXRIMB0GA1UdDgQWBWBBQX
EK5XIGYE9/SopVLv1hqScP/i9zAXBgNVHREEDAOgQxqZG9lQHRlc3QuaXQwDQYJKoZIhvcNAQELBQ
ADgYEAtUgvrMyv5YeAlevt3r51XNsPtXX5JK6A7YV9sjGovECox
82VfBsfedg2qoijNtGbJlBJ1Jxil5fi3Ppcc3DOLEW73jtRvibqSjWqvU3IAA1SvLayFGTgtEAEuu7zRSs7pK1rOq
```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

```

poadHLNC1 djU7C6HjN9FTT4DW1 T1C/z0xA=)
challenge (4IDENTITYCH)
issuer (CN=Bit4ID - TEST CA, C=IT, S=NA, L=Napoli, O=Bit4ID)
subject (E=jdoe@test.it, 2.5.4.4=Doe, 2.5.4.42=John, CN=Doe/John/DOEJHN1234567890, O=Bit4id, C=IT)
signed_challenge (c29 uJLyViof Zg8O6QEpnLou28mkCt0Hut5rOGEUAHdl3HkKF0ABsVKgYDX 4g0Q9rt n
DzNWzmNER2i1W7QeQjTHuUd
yg7I59uJp6hiqRHRqrXBzgo9sp6nQlgFyDjwoYmNpx2PZKIZSewvIF5dkS21Toy
B3OoY4obK2Aq7QQ2NIGsKOhnmTM4gUJIBYiHrJx1Tz39ATT3yAAQ9brKICTVwkEtrvA/GpQndzEisikQtYu
m/pFe9entZ0FB6kNwW3mjfOxcuc5CjQ8YD3MA13T4Msslw47NqnWuvKxN35dcyDH/tNix
Dg6lcFE3TXkfn/SUSylh5GLkaOrPQ==)
serial (0244)
not_after (Thu, 01 Sep 2022 15:30:28GMT)


```

A continuación se detallan estos parámetros:

Parámetro	Valor	Ejemplo
result	Este parámetro contiene el resultado final del reto de verificación. Verificación correcta → ok : {CERTIFICATE CN} Verificación errónea → ko : {ERROR DESCRIPTION}	ok: Doe/John/DOEJHN1234567890
sk	El valor de una clave secreta definida en 4identity Server en el archivo de configuración config.ini , el cual se encuentra en la siguiente dirección: /<<directorio_instalación>>/smartengine/etc/connector	5787237843
not_before	Fecha inicial válida del certificado.	Mon, 03 Sep 2012 15:30:28GMT
certificate	Certificado del usuario codificado en base 64.	...
challenge	Reto utilizado en la fase de autenticación.	4IDENTITYCH
issuer	CN del emisor del certificado.	CN=Bit4ID - TEST CA, C=IT, S=NA, L=Napoli, O=Bit4ID
subject	El valor de los atributos del sujeto dentro del certificado.	E=jdoe@test.it, 2.5.4.4=Doe, 2.5.4.42=John, CN=Doe/John/DOEJHN1234567890, O=Bit4id, C=IT
signed_challenge	Valor del reto firmado	...
serial	Serial del certificado.	0244
not_after	Fecha de expiración del certificado.	Thu, 01 Sep 2022 15:30:28GMT

POSIBLES VALORES DEL ATRIBUTO **RESULT** DADA UNA **AUTENTICACIÓN INCORRECTA**


NOMBRE	DESCRIPCIÓN
ko:UnknownIssuer	Este error se muestra cuando el certificado utilizado para la autenticación es emitido por una autoridad de certificación que no está dentro de la lista de certificados de confianza configurados en el servidor de 4identity
ko:UnknownValidity	Se muestra cuando el certificado no aporta un parámetro correcto de validez o el formato de fecha no es el correcto.
ko:NotYetValid	Este error se muestra cuando el tiempo de validez del certificado no ha comenzado.
ko:Expired	Este error se muestra cuando el certificado utilizado está caducado.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

ko:AlgorithmMismatch	Este error se muestra cuando el algoritmo de firma usado por la CA para firmar el certificado no coincide con el de la CA, los algoritmos de firma soportados son MD5, SHA1, SHA256, SHA512.
----------------------	--

AVISO: Los certificados de confianza (incluidos en la cadena de certificación) utilizados para la autenticación necesitan ser almacenados en la carpeta que se muestra a continuación y en formato CER:

<<DIRECTORIO_INSTALACIÓN_4IDENTITY_SERVER>>/smartengine/etc/smartengine/certificates/

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

4. Parámetros (tags) disponibles

Filtrar certificados

4identity Client permite, a través de distintos parámetros incluidos en el formulario HTML, aplicar filtros a los certificados a mostrar.

Por ejemplo: mostrar solo los certificados emitidos por la Autoridad de Certificación CA Reconocida 1.

Parámetro	Valores	Por defecto
bit4id-certType	"ANY", "SIG", "AUT"	"ANY"
bit4id-issuerFilter	"CN=..., OU=..., T=..., ecc."	-
bit4id-subjectFilter	"CN=..., serialNumber=... OU=..., T=..., ecc."	-

bit4id-certType: Se filtra en función del tipo de certificado:

- Certificado de firma: se define este valor como 'SIG'.
- Certificado de autenticación: se define este valor como 'AUT'.

bit4id-issuerFilter: Posibilita el filtrado en función del emisor del certificado. Para esto, se debe incluir en este parámetro el valor que se requiera del atributo 'X509 Subject' de la CA. Por ejemplo: "**C** = ES, **O** = www.bit4id.com, **CN** = bit4id.uim.test.CA". Se puede también incluir múltiples CAs utilizando el carácter "|".

bit4id-subjectFilter: Define el filtro del certificado de la firma, como un sub-string del nombre completo 'X509 Subject' del propietario del certificado. Por ejemplo: "**C** = ES, **O** = www.bit4id.com".


4.1. Parámetros de firma digital

Seguidamente se detallan los parámetros para la funcionalidad de firma. **Los parámetros obligatorios aparecen acompañados de un asterisco (*)**.

Configuración general

Esta sección describe los parámetros de la firma que son globales independientemente del tipo de firma a realizar. Estos definen, entre otros aspectos, el tipo de firma, el archivo a firmar, la gestión de éste, etc.

Parámetro	Valores/Descripción	Por defecto [ejemplo]
bit4id-signatureType (*)	"CAAdES", "PAdES", "XAdES", "AUTO"	-
bit4id-localFile	"YES", "NO"	"NO"

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

bit4id-filepath (depende de bit4id-localFile)	Path local hasta el archivo.	Si no se define, se abrirá un explorador de archivos.
bit4id-document	URL del archivo o Archivo a firmar (string o stringb64)	(*) Obligatorio si bit4id-localFile no está definido. [http://site.es/doc.pdf]
bit4id-documentName	Nombre del documento	(*) Obligatorio si bit4id-localFile no está definido. [test.pdf]
bit4id-documentID	ID del documento	- [-signed21012018]
bit4id-filter	<x>.<y>	*.pdf [*.*]
bit4id-message	String de texto	-
bit4id-bundle	"YES", "NO"	"NO"
bit4id-documentSuffix	cades:<SUFF-CADES>, padcs: <SUFF-PADES>, xades:<SUFF- XADES>	-
bit4id-preview	"YES", "NO"	"YES"
bit4id-certInfo	"CN, OU, T, ecc." (Relative Distinguished Names X.509)	"CN"
bit4id-pinsession	UUID	-
bit4id-pinsessionTimeout	Entero en segundos	"60"
bit4id-minVersion	"<x>.<y>.<z>"	- [1.9.6]
bit4id-updateMessage	String de texto	- ["Descarga el Nuevo client en http://4identity.com/new_client"]


bit4id-signatureType: Define el tipo de firma que se llevará a cabo sobre el archivo. Cuando se define con el valor "AUTO", 4identity aplicará el formato de firma más adecuado en función de la extensión del archivo a firmar, ya sea en la firma de un único documento o en firma múltiple (archivos a firmar contenidos en un archivo ZIP).

bit4id-localFile: Define que el documento a firmar se encuentra de manera local.

bit4id-filepath: Si "bit4id-localFile" se encuentra como "YES" en este parámetro se especifica el path local hasta el archivo.

bit4id-document: (siempre que bit4id-localFile='NO'). Este parámetro permite realizar UNA de las siguientes opciones:

- Se define la URL donde se encuentra el documento a firmar.
- Incluir directamente el documento a firmar, en formato codificado BASE64 (añadiendo el prefijo 'stringb64:XXX'. Ej: 'stringb64:ZHNkc2RzZHM...') o como String (añadiendo el prefijo 'string:XXX').

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

bit4id-documentName: Nombre del archivo a firmar. Es importante definir también la extensión de éste para la correcta identificación del tipo de archivo.

bit4id-documentID: Se define un ID que se asociará con el documento a firmar. Este valor se envía al servidor de aplicaciones en la petición POST (como "**documentID**") justo con el archivo firmado (y otros parámetros). De esta manera la aplicación web puede luego gestionar el renombrado del archivo y su posterior almacenamiento. Ejemplo: Archivo firmado: test.pdf, documentID: signed080318, Archivo final (renombrado): test.pdf-signed080318.

bit4id-filter: Si "bit4id-localFile" se encuentra en "YES" y no viene definido el path del archivo en "bit4id-filepath", 4identity muestra al usuario la selección del archivo a firmar desde el filesystem local. Este filtro define un regex para limitar los archivos mostrados al usuario.

bit4id-message: Si "bit4id-localFile" se encuentra como "YES" y no viene definido el path del archivo en "bit4id-filepath", 4identity muestra al usuario la selección del archivo a firmar desde el filesystem local. Este tag define el mensaje que aparece en la parte superior del explorador del filesystem.


bit4id-bundle (firma de múltiples archivos): Si este atributo tiene como valor YES, y se desea la firma local, se ofrecen dos opciones:

- Definir el path local hasta un archivo .ZIP
- No definir path y escoger desde el navegador de archivos locales aquellos archivos que se quieran firmar.

Si los archivos se encuentran de forma remota, es necesario indicar la URL hasta un archivo ZIP. Los ficheros a firmar dentro del ZIP serán mostrados en la vista previa, como de costumbre. El fichero de salida que se enviará a la aplicación web será un ZIP que contendrá los ficheros ya firmados.

La firma de los archivos dentro del ZIP se lleva a cabo de forma secuencial. Si se produjera algún tipo de error en el momento de la firma de alguno de los archivos contenidos en el ZIP, el proceso de firma NO se detiene (continúa firmando el resto de archivos) y se añade de forma automática el sufijo '**-signaturefailed**' al nombre del archivo fallido.

bit4id-documentSuffix: En el caso de la firma múltiple (bit4id-bundle = YES) es posible añadir un sufijo al nombre de cada archivo firmado. Este sufijo se añade al archivo antes de ser incluido al archivo .zip final (junto con el resto de archivos firmados). Para cada formato de firma (CADES, PAdES, XAdES) se puede configurar un sufijo diferente haciendo uso de la siguiente sintaxis: *caDES:<SUFF-CADES>* | *pAdES:<SUFF-PADES>* | *xAdES:<SUFF-XADES>*, haciendo uso del carácter "|" para incluir diversos sufijos.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

Ejemplo: *pades: -signedPAdES | xades: -signedXAdES*. En este ejemplo se añadirían los sufijos a aquellos archivos firmados con PAdES o XAdES y no se incluiría ninguno con aquellos firmados con CAdES.

Valores **NO** soportados en el sufijo: \ , / , : , * , " , ? , < , > , | .

bit4id-preview: Permite mostrar o no la vista previa (siempre que la extensión del archivo lo permita) antes de firmar el archivo.

bit4id-certInfo: Define el/los atributo/s mostrado/s de los certificados disponibles en 4identity Client.

bit4id-pinsession: Este parámetro dota a la solución de la capacidad de guardar el PIN en caché para evitar la inserción de este en firmas posteriores. Aquí se debe definir un UUID (con el objetivo de que sea lo suficientemente robusto) que conformará un identificador único asociado al PIN guardado en la caché de 4identity Client. Se recomienda que, en caso de utilizar este valor, sea diferente para cada usuario y para cada sesión.

bit4id-pinsessionTimeout: Este tag define el *timeout* de la sesión especificada en el tag, *bit4id-pinsession*. Por defecto el valor es 60 segundos, si en 60 segundos no se ha enviado de nuevo la misma variable de sesión *bit4id-pinsession*, el Client requerirá el pin de nuevo.

Cada vez que se envía la misma variable de session, con un timeout de `<div class="bit4id-pinsessionTimeout">120</div>` sucede lo siguiente:

Tiempo 0s: 4identity pide el pin y comienza la cuenta regresiva de la session de 120 segundos

Tiempo 37s: 4identity usa el pin en chache y refresca el contador 37+120=157

Tiempo 158s: pin descartado


Tiempo 160s: 4identity pide el pin de nuevo para la misma session.

bit4id-minVersion: Indica el número de versión mínimo de 4identity Client para poder llevar a cabo la firma/autenticación. Si la versión es menor que la requerida por la aplicación web, 4identity Client muestra el mensaje definido en *bit4id-updateMessage*.

bit4id-updateMessage: Define el mensaje mostrado en caso de que la versión de 4identity Client sea inferior a la definida en *bit4id-minVersion*.

Firma CAdES

Esta sección describe los parámetros de la firma que se incluyen junto a la clase *bit4idsignReq* para realizar una solicitud de firma **CAdES**.

	Título del documento: Guía de integración	28/06/2019
	Producto: 4identity	Versión 2.7

Parámetro	Valores	Por defecto
bit4id-signatureType (*)	"CADES"	NO DEFAULT
bit4id-level (*)	"BES", "T"	"BES"
bit4id-signingAlgorithm	"RSASHA256", "RSASHA1", "RSAMD5", "RSASHA512",	"RSASHA256"
bit4id-encoding	"1", "0"	"0"
bit4id-parallel	"CN=..., OU=..., T=..., ecc."	"NO"

bit4id-level: Define el nivel de firma CAdES que se utilizará.

bit4id-signingAlgorithm: Define el algoritmo que se usará para llevar a cabo la firma.

bit4id-encoding: Define la codificación del resultado de la firma.


- 1: Base64
- 0: Binario

bit4id-parallel: Permite habilitar la firma múltiple en paralelo sobre un archivo. Si se define como 'NO', la firma sobre un archivo firmado ya con CAdES (.p7m) se lleva a cabo sobre un nivel superior (formato matrisca/anidada). Si se define a 'YES' la firma se efectúa en el mismo nivel o de forma paralela.

Firma PAdES

La siguiente tabla describe los parámetros que pueden acompañar a la clase: bit4idsignReq para llevar a cabo la solicitud de firma tipo **PAdES**.

Parámetro	Valores	Por defecto
bit4id-signatureType (*)	"PAdES"	-
bit4id-level (*)	"BES", "T", "LTV"	"BES"
bit4id-signingAlgorithm	"RSASHA256", "RSASHA1", "RSAMD5", "RSASHA512"	"RSASHA256"
bit4id-invisible	"YES", "NO"	"NO"
bit4id-page	"0", "1", "n", "ALL"	"1"
bit4id-image	URL: "http://imageurl", B64: "data:image/png;base64,R0lGODdhMA..."	-
bit4id-position	"[x1, y1, x2, y2]"	"[10,10,350,100]" Abajo a la izquierda del documento (S-O).
bit4id-editimage	"YES", "NO"	"NO"
bit4id-editpagepos	"YES", "NO"	"NO"
bit4id-reason	String de texto	-
bit4id-location	String de texto	-
bit4id-paragraphFormat	Formato en JSON de la firma gráfica.	Configurable para cada Client.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

bit4id-signatureType: Define el tipo de firma que se llevará a cabo sobre el archivo.

bit4id-level: Define el nivel de firma PAdES que se utilizará.

bit4id-signingAlgorithm: Define el algoritmo que se empleará para la firma.

bit4id-invisible: Si no se desea incluir ninguna firma gráfica en el documento (se incluye la marca por defecto) se debe definir este valor a 'YES'.

bit4id-page: Define la página del documento pdf donde se añadirá la firma gráfica. La primera página del documento se define en '0'. Si se quiere incluir la firma gráfica en cada página se ha de especificar como 'ALL'.

bit4id-image: Define la url o la data-url en la cual de la imagen a incluir en la firma gráfica.

bit4id-position: Define la posición, en puntos [x1,y1,x2,y2], en la cual se mostrará la firma gráfica dentro del documento PDF.

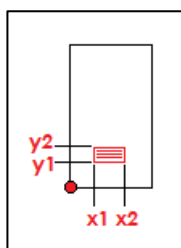


Figura 17. Diagrama de localización de los puntos en el documento


bit4id-editimage: Si se define como 'YES', 4identity Client permitirá al usuario escoger (durante la vista previa del documento) desde el navegador de archivos locales la imagen a incluir en la firma gráfica.

bit4id-editpagepos: Si se define como 'YES', 4identity Client permitirá al usuario escoger de forma interactiva y dinámica (durante la vista previa del documento) la posición de la marca gráfica dentro del documento.

bit4id-location: Define la localización que se incluirá en la firma PAdES.

bit4id-paragraphFormat: Este parámetro permite definir el estilo y contenido de la marca gráfica, siguiendo el formato JSON siguiente:

```
{[{"font": [<FONTFAMILY>,<FONTSIZE>],"align" : <ALIGNVALUE >, "format " : [<LINEA1 >,<LINEA2 >,...,<LINEAN > ]}]}
```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

Con **"font"** (opcional) se define el formato que tendrá el texto incluido en la marca gráfica. Se indica la familia `<FONTFAMILY>` de fuente deseada así como el tamaño de ésta. Los tipos de fuente soportados son los siguientes:

- "Universal" (default) : fuente perteneciente a la categoría Sans-serif (normal).
- "Universal-Bold" : fuente Universal en negrita.
- "Universal-Italic" : fuente Universal en cursiva.
- "Universal-BoldItalic" : fuente Universal en negrita y cursiva.
- "Times" : fuente perteneciente a la categoría Serif.
- "Times-Bold" : fuente Times en negrita.
- "Times-Italic" : fuente Times en cursiva.
- "Times-BoldItalic" : fuente Times en negrita y cursiva.

En `<FONTSIZE>` se indica el tamaño de los caracteres, definidos como puntos. El software realiza en cada caso un ajuste automático del tamaño de la fuente en base a la dimensión de la marca gráfica.


Con **"align"** (opcional) se indica el alineamiento del texto respecto a una imagen (si es que la hay). Puede asumir los siguientes valores:

- right: texto a la derecha de la imagen (default)
- left: texto a la izquierda de la imagen.
- middle: texto superpuesto a la imagen.

Con **"format"** se definen las líneas de texto que se incrustarán en la marca gráfica. Éstas pueden crearse a partir de información obtenida desde el certificado de firma:

- \$(CN)s : CommonName
- \$(L)s : Locality
- \$(S)s : stateOrProvinceName
- \$(OU)s : organizationalUnitName
- \$(E)s : email
- \$(Email)s : email incluido en el Subject Alternative Name
- \$(Issuer)s : Common Name de la CA emisora del certificado.
- \$(date)s : fecha y hora de la firma. Siguiendo el formato internacional ISO 860.
- \$(reason)s : valor del parámetro "bit4-reason"
- \$(location)s : valor del parámetro "bit4-location"

A continuación se muestra un ejemplo de formato de este parámetro, así como su resultado final:

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

```
[{ "font" : [" Universal ",50], "align ":" right", "format ":" [" Digitally Signed by  
$(CN)s", "O=$(O)s", "C=$(C)s", "S=$(S)s", "Date: $(date)s", "CustomField1: CustomValue1 ",  
"CustomField2: CustomValue1 ", "CustomField3: CustomValue3 "]}]
```

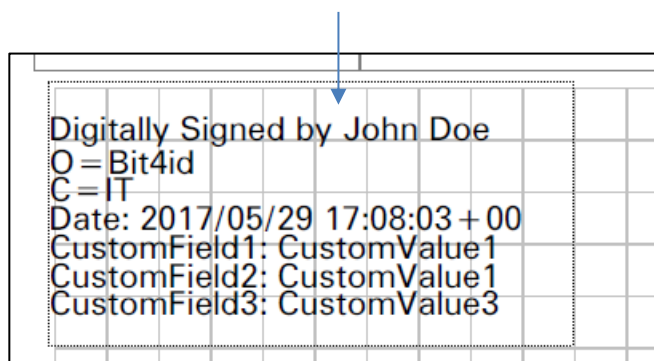


Figura 18. Ejemplo simple de firma gráfica configurada a través de paragraphFormat

Firma XAdES

La siguiente tabla describe los parámetros que pueden acompañar a la clase: `bit4idsignReq` para llevar a cabo la solicitud de la firma tipo **XAdES**.

Parámetro	Valores	Por defecto
<code>bit4id-signatureType</code>	"XAdES"	NO DEFAULT
<code>bit4id-level</code>	"BES", "T", "XL"	"BES"
<code>bit4id-hashAlgorithm</code>	"SHA256", "SHA1", "MD5", "SHA512"	"SHA256"
<code>bit4id-signatureMode</code>	"Enveloping", "Enveloped", "InternalDetached", "Binary"	"Enveloped"
<code>bit4id-level</code>	"BES", "T", "C", "X", "XL"	"BES"
<code>bit4id-binary</code>	"1", "0"	"0"
<code>bit4id-xpath</code>	Xpath del nodo del documento XML a firmar	"/*"


bit4id-hashAlgorithm: Define el tipo de algoritmo hash que se usará para la firma.

bit4id-signatureMode: Define el tipo de firma XAdES que se empleará.

bit4id-level: Define el nivel de firma XAdES que se utilizará.

bit4id-binary: Se debe fijar en 1 si se quiere firmar un archivo binario, o en 0 para firmar un archivo xml.

bit4id-xpath: XPATH del nodo del documento a firmar.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

bit4id-issuerFilter: Define el filtro de certificado de firma, como un sub-string del nombre completo del emisor del certificado.

Sellado de Tiempo

Esta sección describe los parámetros del sellado de tiempo, aplicables a cualquier tipo de firma digital, ya sea CAdES, XAdES o PAdES. Estos parámetros son aplicables a cualquier estándar de firma descrita anteriormente. Dichos parámetros deben ser facilitados por la autoridad de sellado de tiempo.

Parámetro	Valores	Por defecto
bit4id-tsUrl	"Dirección de la autoridad de sellado de tiempo"	""
bit4id-tsUsername	"Nombre de usuario"	""
bit4id-tsPassword	"Contraseña"	""
bit4id-reqPolicy	"OID de la política de sellado de tiempo"	""
bit4id-tsDigestAlgorithm	'sha256','sha512'	

Para llevar a cabo pruebas contra la plataforma de Bit4id se debe introducir como valor del parámetro bit4id-tsUrl la siguiente dirección:

`'http://as-demo.bit4id.org/smartengine/tsa'`


Esta plataforma es de acceso público (no requiere credenciales de acceso) y no requiere ningún OID de política de sellado.

4.2. Parámetros de autenticación

Seguidamente se detallan los parámetros para la funcionalidad de autenticación. Estos parámetros se incluyen juntamente con la clase: **bit4id-authReq**.

Parámetro	Valores	Por defecto
bit4id-challenge	"challenge to sign"	NO DEFAULT
bit4id-signingAlgorithm	"RSASHA256", "RSASHA1", "RSAMD5", "RSASHA512"	"RSASHA256"
bit4id-certInfo	"CN, OU, T, ecc." (Relative Distinguished Names X.509)	"CN"
bit4id-pinsession	UUID	-
bit4id-pinsessionTimeout	Entero en segundos	"60"
bit4id-minVersion	"x.y.z"	- [1.9.6]
bit4id-updateMessage	String de texto	- ["Descarga el Nuevo client en http://4identity.com/new_client "]

bit4id-challenge: Ha de definir un string alfanumérico que será utilizado como reto para llevar a cabo el proceso de autenticación.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

bit4id-signingAlgorithm: Define el algoritmo que se usará para la firma del challenge.

bit4id-certInfo: Define el/los atributo/s mostrado/s de los certificados disponibles en 4identity Client.

bit4id-pinsession: Este parámetro dota a la solución de la capacidad de guardar el PIN en caché para evitar la inserción de este en firmas posteriores. Aquí se debe definir un UUID (con el objetivo de que sea lo suficientemente robusto) que conformará un identificador único asociado al PIN guardado en la caché de 4identity Client. Se recomienda que, en caso de utilizar este valor, sea diferente para cada usuario y para cada sesión.

bit4id-pinsessionTimeout: Este tag define el *timeout* de la sesión especificada en el tag, bit4id-pinsession. Por defecto el valor es 60 segundos, si en 60 segundos no se ha enviado de nuevo la misma variable de sesión bit4id-pinsession, el Client requerirá el pin de nuevo.


Cada vez que se envía la misma variable de session, con un timeout de `<div class="bit4id-pinsessionTimeout">120</div>` sucede lo siguiente:

Tiempo 0s: 4identity pide el PIN y comienza la cuenta regresiva de la session de 120 segundos

Tiempo 37s: 4identity usa el PIN en chache y refresca el contador $37+120=157$

Tiempo 158s: PIN descartado

Tiempo 160s: 4identity pide el PIN de nuevo para la misma session.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

5. Canal de 4identity

Esta sección se dirige al desarrollador que desea llevar a cabo pruebas de integración y manejo de errores, de 4identity. Describe la manera de detectar el cliente, y el manejo de errores de en la firma/autenticación, permitiendo modificar el flujo de la aplicación web para mejor orientar al usuario.


Una característica muy importante que el desarrollador debe realizar para integrar la solución 4identity es la verificación y distribución del 4identity Client en los ordenadores del usuario. Para lograr esto solo es necesario crear un script en JavaScript que pueda monitorear los cambios en el estado de la comunicación entre el navegador y 4identity Client y que reaccione acordemente.

5.1. Estado

El estado de la comunicación entre el navegador y el cliente se muestra en la etiqueta HTML `<div id = "bit4id-status"> </ div>`. El monitoreo se realiza observando el cambio del contenido de esta etiqueta es posible realizar una serie de funciones que pueden reaccionar a cambios en el estado. Un síntoma claro de que el cliente no es detectado o no se ha autorizado la apertura del cliente desde el navegador, es que no hay un cambio de estado en el valor de la etiqueta `<div id = "bit4id-status"> </ div>`. Un simple contador de entre 20 y 30 segundos que monitoree el cambio de estado de esta etiqueta guiaría al usuario a que verifique la instalación del cliente, o si el usuario ha realizado todas las acciones necesarias para el correcto funcionamiento de la aplicación.

POSIBLES VALORES DEL CANAL

NOMBRE	DESCRIPCIÓN
Connected	Se muestra cuando el cliente es detectado correctamente y es invocado para la firma o autenticación.
StopIteration:error	Se muestra cuando la operación ha sido cancelada por el usuario final usando el cliente de 4identity.
Disconnected	Excepción relacionada con el error.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

Disconnected:channel does not exist	Excepción relacionada con el error, y es cuando ha sido terminada la conexión con el cliente.
Disconnected:cancelled	Excepción que es mostrada cuando el cliente 4identity se ha cerrado inesperadamente.
KeyError:'event'	<p>Este error puede presentarse debido a:</p> <ol style="list-style-type: none"> 1. Cuando el reverse proxy entre 4identity server y el servidor de aplicaciones no ha sido correctamente configurado. 2. Cuando el dominio de la licencia no coincide con el dominio en el encabezado HTTP que realiza la petición de firma o autenticación 3. Cuando la licencia para dicho dominio ha caducado


5.2. Monitorización

PETICIÓN DE AUTENTICACIÓN (podría ser de firma)

```
<html>
<body>
  <form class="bit4id-auth" method="post"
    action="https://www.4identity.eu/smartengine/uploadauth" >
    <div class="bit4id-authReq" style="display: none">
      <div class="bit4id-challenge">mUXuTaowfPylJzzRMsfmpg==</div>
      <div class="bit4id-certType">ANY</div>
      <div class="bit4id-certInfo">CN</div>
    </div>
    <div id="bit4id-status"></div>
    <input disabled type="submit" value="auth"/>
  </form>
  <script src="https://www.4identity.eu/smartengine/bit4id-auth.js"></script>
  <script src="https://www.4identity.eu/dev/check4identityclient.js"></script>
</body>
</html>
```

DETECCIÓN EN JAVASCRIPT

```
(function() {
  var componentLoaded = false,
  setinfo = function(msg) {
    bit4id_info.innerHTML = msg;
  },
  addclass = function(el, className) {
    if (el.classList) el.classList.add(className);
    else el.className += ' ' + className;
  },
  removeclass = function(el, className) {
    if (el.classList) el.classList.remove(className);
```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	4identity

```

else el.className = el.className.replace(new RegExp('(' + className.split(' ').join(' ') + '\\b|$', 'gi'), '');
},

bit4id_status = document.querySelector('#bit4id -status '),
bit4id_info = document.querySelector('#bit4id -info'),

// observar cambios en bit4id -status
config_bit4id_status = {
  childList: true,
  characterData: true,
  subtree: true
},
bit4id_status_observer = new MutationObserver(function(mutations) {
  var self = this;
  mutations.forEach(function(mutation) {
    var bit4_event;
    if (mutation.target.id === 'bit4id -status ')
      bit4_event = mutation.target.innerHTML;
    else if (mutation.target.nodeName === '#text')
      bit4_event = mutation.target.data;
    else return;

    if (bit4_event !== 'not_installed ') {
      componentLoaded = true;
    }
    if (bit4_event in bit4id_callbacks) {
      bit4id_callbacks[bit4_event]();
    } else {
      bit4id_callbacks[__default ](bit4_event);
    }
  });
});

bit4id_callbacks = {
  'connected ': function() {
    setinfo("4Identity Client is ready");
  },
  'not_installed ': function() {


    // No es posible detectar el cliente, crear enlace de descarga
    var link = document.createElement("a");
    link.download = "4identity_client_win.exe";
    link.href = "/dev/4identity_demo_win.exe";
    link.innerHTML = "Download link";
    link.onclick = function() {
      removeclass(document.querySelector('.bit4 -link'), "disabled -link");
    }
    setinfo("4Identity Client is not installed.");
    bit4id_info.appendChild(link);

  },
  __default ': function(msg) {
    console.log(msg);
    setinfo("reload page to try another time")
  },
}

handleClick = function(event) {

  bit4id_link = document.querySelector('.bit4 -link');
  addclass(bit4id_link, "disabled -link");
  setinfo("Connecting to 4Identity Client ...")
  setTimeout(function() {

```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto: 4identity	


```

// Si se activa el timeout de conexion mostif connection timeout fire set bit4id -status to
'No_instalado' 7
if (!componentLoaded) {
    bit4id_status.innerHTML = ('not_installed ');
};
}, 20000); // timeout 20 s
return false;
},

bit4id_status_observer.observe(bit4id_status, config_bit4id_status);

})();

```

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

6. Diagramas de flujo

6.1. Diagrama de flujo de firma digital

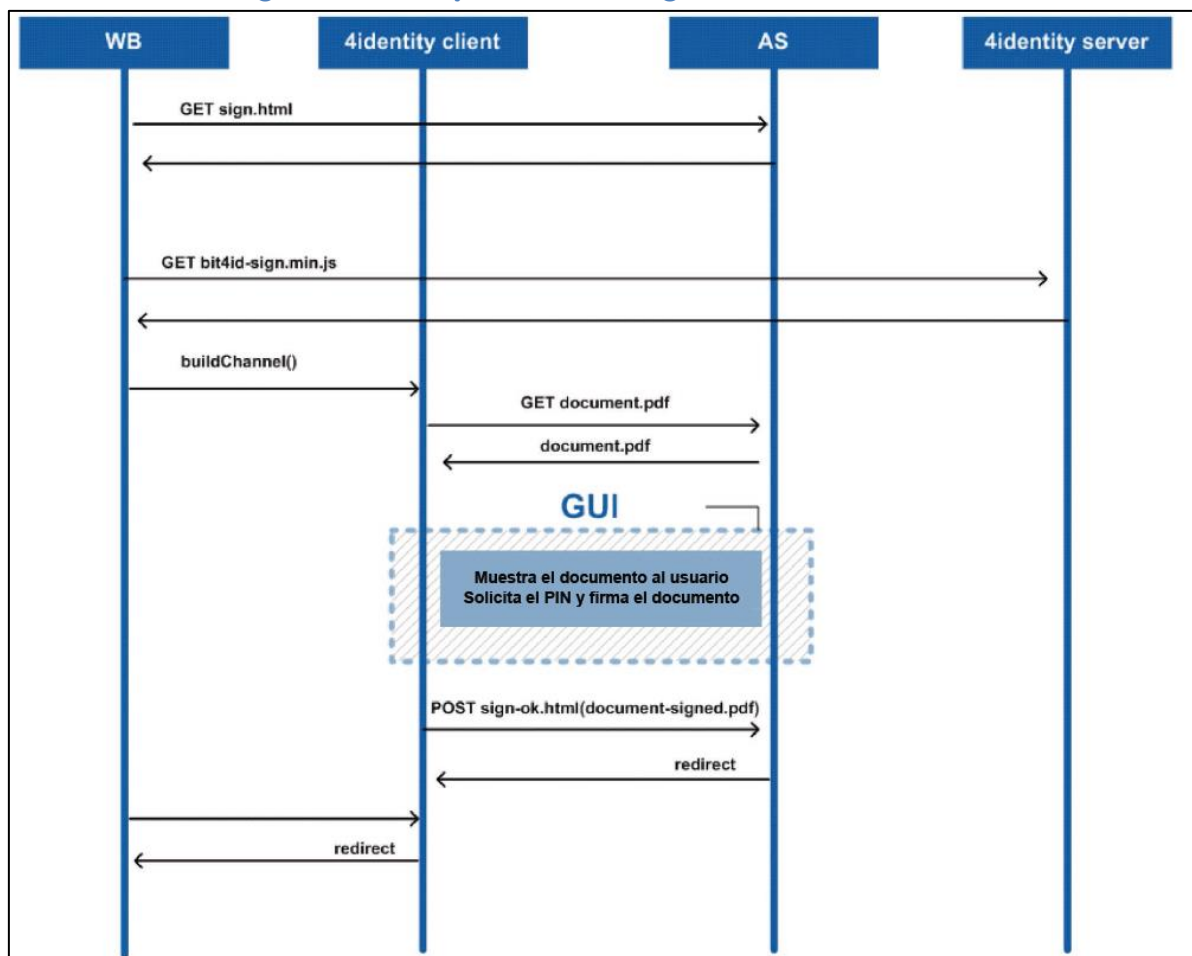



Figura 19. Diagrama de firma digital

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto: 4identity	

6.2. Diagrama de flujo de autenticación

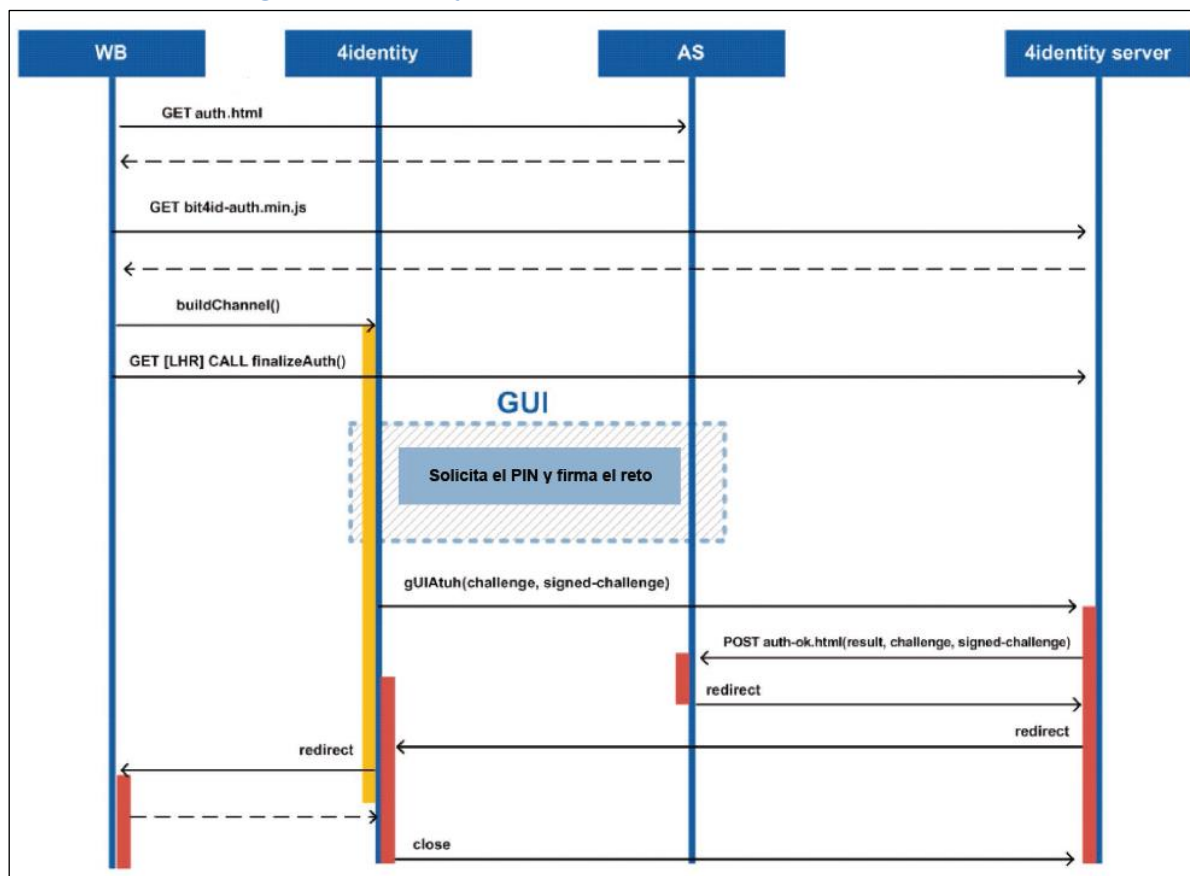



Figura 20. Diagrama de autenticación

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

7. Errores comunes

En esta sección se especifican algunos de los errores más comunes durante la integración de 4identity.

Para el correcto funcionamiento de la solución, se debe asegurar la conectividad (y accesibilidad) entre los siguientes elementos:

- Entre **4identity Client** y el recurso del **servidor de aplicaciones** definido en '**action**' (formulario HTML).
- Entre **4identity Server** y el **servidor de aplicaciones**.

1. Canal explorador-4identity no se ha establecido

Para detectar si el canal se ha establecido se debe ejecutar el proceso de firma o autenticación y ejecutar el índice o la página principal del ejemplo de autenticación para que el script intente establecer el canal.

Si el canal se ha establecido debería mostrarse en la barra de tareas el siguiente logo:



Si no es así, las causas pueden ser:


- 1.1. 4identity Client no está instalado en la máquina

Se debe comprobar si 4identity Cliente está correctamente instalado

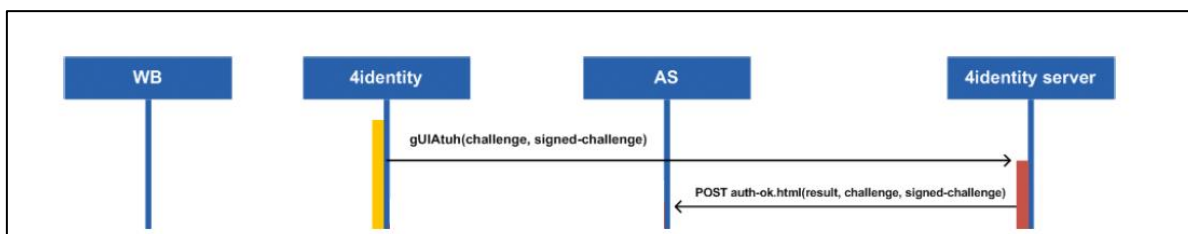
- 1.2. El proxy de la red bloquea la conexión con 4identity Server
- 1.3. Existe algún tipo de conflicto entre la licencia y el dominio de uso.

De esta forma no se puede realizar la llamada al script y por tanto no se establecerá nunca el canal.

2. Autenticación falla y se recibe un error en el explorador

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

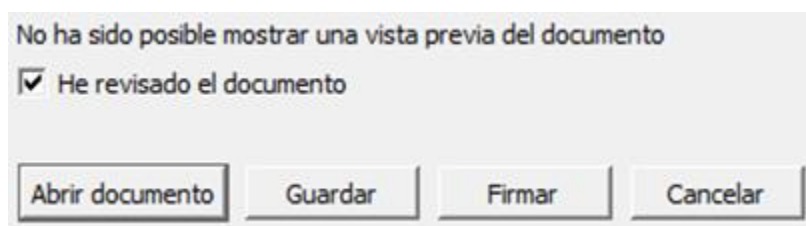
Para que la aplicación web reciba la respuesta del 4identity Server (método POST) este debe tener una dirección IP pública donde dirigir la respuesta. Si la aplicación web está en local 4identity Server tendrá como parámetro de envío la IP privada del servidor web por lo que nunca llegará a su destino.



El apartado “**6.2. Diagrama de flujo de autenticación**” muestra el proceso completo de autenticación.

3. La ruta del documento a firmar no es válida

Si la ruta no es válida y el documento es PDF el proceso de firma mostrará el siguiente mensaje:




Tanto si es o no PDF, tras confirmar la firma, el proceso fallará.

Seleccione la ruta completa del documento a firmar almacenada en la etiqueta “<div class=“bit4id-document”>” y cópiela en el explorador para comprobar si la ruta es correcta. El documento PDF se debería mostrar en el explorador. Si no se visualiza modifique la ruta del documento y compruebe de nuevo la dirección hasta que se consiga cargar el documento.

4. 4identity Client no puede enviar el archivo firmado/datos de autenticación a la aplicación web.

La URL definida en el atributo ‘**action**’ FORM HTML determinará donde se realizará el POST. Si esta dirección no es accesible de forma pública (desde 4identity Client), la aplicación web nunca podrá recibir ningún tipo de información.

	Título del documento:	28/06/2019
	Guía de integración	Versión 2.7
	Producto:	
	4identity	

En entornos locales se debe asegurar que la ruta hasta el archivo que recibe el POST se encuentra correctamente indicada.

ASISTENCIA TÉCNICA: Si no se soluciona el problema a través de las vías mencionadas anteriormente y a través del análisis del código en cuestión se debe habilitar la carpeta de registros de 4identity. Para ello se debe crear una carpeta llamada *log*. En función de la versión de 4identity Client que se tenga instalada, el log puede encontrarse en las siguientes rutas:

- A. C:\Users\"Nombre_Usuario"\AppData\Roaming\Bit4id\keychain\browser_intent\var\log
- B. C:\Users\"Nombre_Usuario\".bit4id\browser_intent\var\log

Estos logs se enviarán al departamento de Soporte Técnico de Bit4id para su análisis.