



Junta de Andalucía

Plataforma @firma

Cliente Autenticación - Manual de integración

Versión: v01r10

Fecha: 11/04/2022

Queda prohibido cualquier tipo de explotación y, en particular, la reproducción, distribución, comunicación pública y/o transformación, total o parcial, por cualquier medio, de este documento sin el previo consentimiento expreso y por escrito de la Junta de Andalucía.

	Agencia Digital de Andalucía Dirección General de Estrategia Digital	Plataforma @firma Cliente Autenticación - Manual de integración
---	---	--

HOJA DE CONTROL

Título	Cliente Autenticación - Manual de integración		
Entregable	Documentación técnica Plataforma @firma		
Nombre del Fichero	20220411-Manual Integrador cliente de autenticación_v01r10		
Autor	DGED		
Versión/Edición	v01r09	Fecha Versión	11/04/2022
Aprobado por		Fecha Aprobación	
		Nº Total Páginas	19

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Área	Fecha del Cambio
v01r00	Creación del documento	UTE	UTE	15/04/2013
v01r01	Se indica la URL del repositorio de artefactos Maven de CHAP en el apartado 3.3	UTE	UTE	12/05/2014
v01r02	Configuración avanzada	UTE	UTE	21/07/2014
v01r03	Configuración avanzada (corrección)	UTE	UTE	28/11/2014
v01r04	(afirma-authentication-client-1.0.7) Se permite el uso del almacén de confianza por defecto de la JRE (apartado 3.4)	UTE	UTE	26/01/2016
v01r05	Se modifica el Servlet de retorno (ReturnAuthenticationServlet) para que al invocar al servicio del nucleo	UTE	UTE	24/04/2018

Versión	Causa del Cambio	Responsable del Cambio	Área	Fecha del Cambio
	ObtenerInfoValidacionTicket, tenga en cuenta que el componente de autenticación del núcleo de @firma 6 ahora también incluye en sesión el certificado que se ha validado en base 64			
v01r06	Adaptación al nuevo formato de estilos de la Junta	UTE	UTE	02/02/2021
v01r07	Se modifica el cliente para que permita definir también las propiedades de configuración mediante propiedades del sistema	UTE	UTE	01/07/2021
v01r08	Se añade un nuevo método que sirve para separar una cadena con los dos apellidos Para aumentar la seguridad en las comunicaciones, el cliente deja de utilizar como parámetro el identificador de sesión del navegador.	UTE	UTE	19/08/2021
v01r09	Se corrige un problema con el generador interno de cadenas aleatorias	UTE	UTE	25/10/2021
v01r10	Se modifica la dependencia con la librería commons-	UTE	UTE	11/04/2022

	Agencia Digital de Andalucía Dirección General de Estrategia Digital	Plataforma @firma Cliente Autenticación - Manual de integración
---	---	--

Versión	Causa del Cambio	Responsable del Cambio	Área	Fecha del Cambio
	logging. Ahora depende de la versión 1.2 ya que la v1.1 tenía dependencia con log4j.1.2.12			

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Área	Nº Copias
Manuel Escobar Montes	Jefe de Servicio	Servicio de Coordinación de Admón. Electrónica	1
José Ignacio Cortés Santos	Gabinete admón electrónica	Servicio de Coordinación de Admón. Electrónica	1
Miguel Angel Ager Vázquez	Soporte @firma	Soporte de Administración Electrónica	1
Alberto-Manuel Benalli Dominguez	Soporte @firma	Soporte de Administración Electrónica	1



ÍNDICE

1. Introducción.....	6
2. Servicios Autenticación de @Firma.....	7
2.1. Arquitectura Lógica.....	7
2.2. Documentación de Referencia.....	8
3. Integración con afirma-authentication-client.....	9
3.1. Dependencias software.....	9
3.2. Dependencias de servicios.....	10
3.3. Establecer dependencias con Maven.....	10
3.4. Fichero de propiedades.....	11
3.5. Fichero Web.xml.....	13
3.6. Interfaz Componente afirma-authentication-client.....	14
3.7. Procesamiento de Respuesta.....	14
3.7.1. Método separarApellidos.....	15
3.8. Ejemplo de integración.....	15
4. Configuración avanzada.....	17
4.1. Propiedades Dinámicas.....	17
4.2. Propiedades del Sistema.....	18



1. INTRODUCCIÓN

El componente **afirma-authentication-client** tiene como objetivo fundamental facilitar a los integradores la interacción con los servicios de autenticación disponibles en la plataforma @firma a través de la fachada de autenticación por tickets. Se trata de un componente software que implementa la lógica necesaria relacionada con la mensajería XML, las comunicaciones(SOAP) y la redirección a la fachada de autenticación, ofreciendo a los integradores una integración sencilla basada en servlets.

Afirma-authentication-client está desarrollado en java y esta construido con Maven siguiendo las recomendaciones de MADEJA.

Funcionalmente, afirma-authentication-client se divide en tres partes funcionales:

- Por un lado, realiza la composición de las peticiones y el procesamiento de la respuestas implementando la lógica de mensajes basada en XML. Los servicios de autenticación de los que hace uso afirma-authentication-client son **GenerarTicket** y **ObtenerInfoValidacionTicket**.
- Por otro lado, el cliente incluye la implementación del envío y recepción del mensaje SOAP que contiene la petición y respuesta XML respectivamente. por lo que este proceso también es transparente para el integrador. El cliente hace uso de librerías de código abierto como Axis y Web Service Security para realizar el intercambio de información con el servicio web de @firma.
- Por último, un modulo compuesto por dos servlets se encarga de gestionar las peticiones a los servicios de @firma y la redirección a la fachada de autenticación por tickets. Este componente devuelve en sesión los datos necesarios para que la aplicación cliente pueda autorizar o no el acceso del usuario.

Este manual presenta la última funcionalidad y describe con detalle cómo integrar las aplicaciones cliente con los servlets de llamada y retorno proporcionados por el componente de autenticación.



2. SERVICIOS AUTENTICACIÓN DE @FIRMA

@firma ofrece los mecanismos necesarios para autenticar a un usuario en las aplicaciones cliente. El cliente de autenticación implementa dicho proceso, el cual requiere la interacción con distintos componentes de la plataforma @firma. Los servicios de los que hace uso el componente afirma-authentication-client son los siguientes:

- Fachada SSL de autenticación por ticket
- Servicios web de autenticación por ticket.

2.1. ARQUITECTURA LÓGICA

La distribución lógica de los distintos módulos software que participan en el proceso de autenticación se indica en el siguiente gráfico:

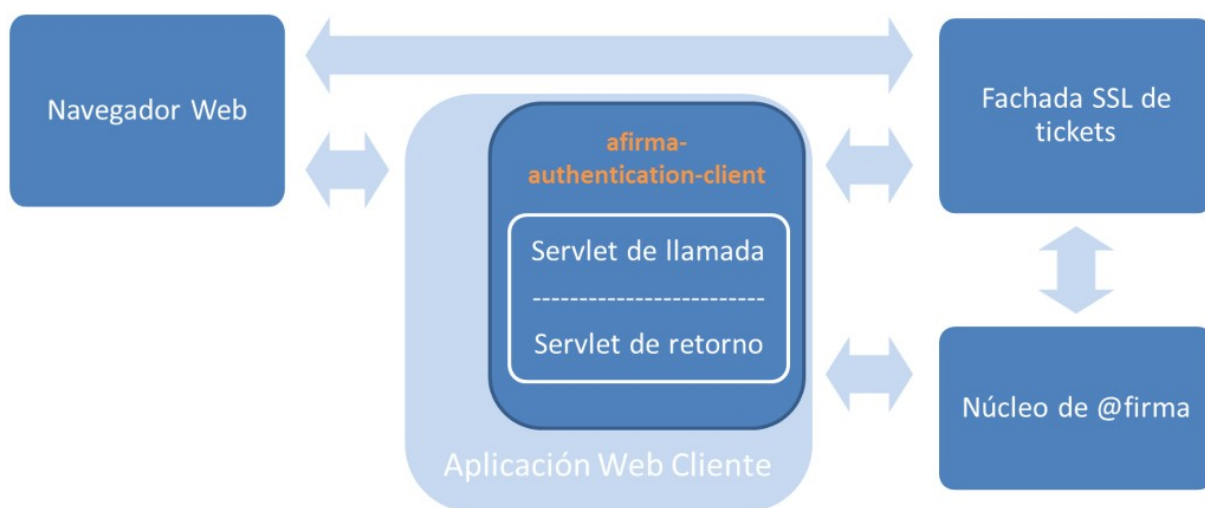


Ilustración 1: Diagrama de bloques del proceso de autenticación

La interfaz de afirma-authentication-client se compone de dos servlets, que se presentan como los puntos de entrada y salida del aplicativo. A continuación se describen cada uno de ellos:



- Servlet de llamada (**CallAuthenticationServlet**): Este objeto se encarga de recopilar la información necesaria del navegador web (identificador de sesión), solicitar el ticket invocando el servicio GenerarTicket y redirigir el flujo de la aplicación a la fachada SSL de autenticación por ticket. Los parámetros que se facilitan a la fachada SSL son los siguientes: identificador del ticket, identificador de la aplicación en @firma, identificador de sesión, URL del servlet de retorno.
- Servlet de retorno (**ReturnAuthenticationServlet**): El objeto de retorno recibe la llamada de la fachada SSL de autenticación que provee los siguientes datos:
 - Identificador del ticket
 - Identificador de la aplicación
 - Identificador de sesión
 - Código de error, en su caso.

El servlet verifica la corrección de los datos facilitados por la fachada e invoca al servicio ObtenerInfoValidacionTicket, el cual provee los datos de validación del ticket y del certificado.

Una vez finalizado el proceso, el componente de autenticación incluye los siguientes datos en sesión como resultado del proceso de validación del ticket:

- Resultado del proceso (éxito o error).
- Descripción del resultado del proceso.
- Datos del certificado validado.
- Certificado en base 64.

La información incluida en sesión debe ser recuperada por la aplicación web cliente para finalizar el proceso de autorización de acceso a la misma.

2.2. DOCUMENTACIÓN DE REFERENCIA

El componente afirma-authentication-client se ha construido basándose en los siguientes manuales disponibles en la web de soporte de administración electrónica:

- Configuraciones SSL para aplicaciones integradas con plataformas de la AE.pdf
- Seminario_Fachada_Tickets.pdf
- TI-@Firma-SSLServer-Tickets-AdaptacionAplicaciones-MAN.pdf
- TI-@Firma-SSLServer-Tickets-Prgs-MAN.pdf



3. INTEGRACIÓN CON AFIRMA-AUTHENTICATION-CLIENT

3.1. DEPENDENCIAS SOFTWARE

El componente `afirma-authentication-client` está implementado en java y compilado con jdk 5. Son necesarias una serie de dependencias (ficheros jar) que deben estar en el *classpath* de ejecución de la librería principal para su correcto funcionamiento. La librería que implementa el cliente se denomina ***afirma-authentication-client-1.x.x.jar*** y las dependencias mencionadas son las siguientes:

- `es.juntadeandalucia.afirma:afirma.authentication-client:jar:1.x.x`
- `junit:junit:jar:3.8.1`
- `commons-logging:commons-logging:jar:1.2`
- `logkit:logkit:jar:1.0.1`
- `avalon-framework:avalon-framework:jar:4.1.3`
- `javax.servlet:servlet-api:jar:2.3`
- `commons-lang:commons-lang:jar:2.1`
- `org.apache.ws.commons.util:ws-commons-util:jar:1.0.2`
- `xml-apis:xml-apis:jar:1.0.b2`
- `xerces:xercesImpl:jar:2.4.0`
- `axis:axis:jar:1.4`
- `org.apache.axis:axis-jaxrpc:jar:1.4`
- `org.apache.axis:axis-saaj:jar:1.4`
- `axis:axis-wsdl4j:jar:1.5.1`
- `commons-discovery:commons-discovery:jar:0.2`
- `org.apache.ws.security:wss4j:jar:1.6.7`
- `org.apache.santuario:xmlsec:jar:1.5.2`
- `org.opensaml:opensaml:jar:2.5.1-1`
- `org.opensaml:openws:jar:1.4.2-1`
- `org.opensaml:xmltooling:jar:1.3.2-1`
- `org.slf4j:slf4j-api:jar:1.6.1`
- `joda-time:joda-time:jar:1.6.2`
- `javax.servlet:servlet-api:jar:2.5`



3.2. DEPENDENCIAS DE SERVICIOS

Para el correcto funcionamiento de afirma-authentication-client es necesario que los servicios web de @firma 5 sean visibles desde el servidor donde se pretende ejecutar el componente (host y puerto). Del mismo modo, es necesaria la visibilidad de la fachada SSL de autenticación por tickets.

3.3. ESTABLECER DEPENDENCIAS CON MAVEN

El componente afirma-authentication-client puede ser incluido en aplicaciones desarrolladas en maven añadiendo la siguiente dependencia al pom.xml de la aplicación:

```
<dependency>
    <groupId>es.juntadeandalucia.afirma</groupId>
    <artifactId>afirma-dss-client</artifactId>
    <version>1.x.x</version>
</dependency>
```

Este componente, y todos los desarrollados por la Consejería de Hacienda y Administración Pública pueden encontrarse en el Repositorio Maven del Servicio de Coordinación de Administración Electrónica (<https://ws024.juntadeandalucia.es/ae/adminelec/repositoriomaven>).

Para que Maven obtenga el artefacto que contiene el componente y resuelva automáticamente sus dependencias, es necesario incluir el siguiente fragmento XML en el fichero pom.xml correspondiente:

```
<repositories>
    <repository>
        <id>SCAERepo</id>
        <url>https://ws024.juntadeandalucia.es/maven/repository/internal</url>
    </repository>
</repositories>
```

NOTA:El Repositorio Maven del Servicio de Coordinación de Administración Electrónica de la Consejería está accesible a través de HTTPS utilizando el certificado **cn=*.juntadeandalucia.es, o=Junta de Andalucía, l=Sevilla, s=sevilla c=ES** de la Junta de Andalucía. Para que Maven confíe en el dominio



ws024.juntadeandalucia.es a través de HTTPS y logre descargar el componente y sus dependencias, es necesario que incluya el certificado indicado en el almacén de certificados por defecto (cacerts) de la Máquina Virtual Java sobre la que se ejecute el script **mvn**.

3.4. FICHERO DE PROPIEDADES

Con el objetivo de facilitar la integración se establece un fichero de propiedades que contiene los argumentos estáticos de las peticiones a realizar. A continuación se muestra un ejemplo del fichero **afirma.properties** que debe incluirse en el *classpath* donde se ejecute el cliente:

```
#####
# Configuración del componente afirma-authentication-client
#####
#####
# Credenciales de la aplicación (obligatorio)
#
# Si se establecen las propiedades de autenticación mediante BinarySecurityToken
# (certificado) este tipo de autenticación prevalece respecto a la autenticación
# mediante UsernameToken (usuario / contraseña).
#####
afirma.idapp = APPID
#####
# BinarySecurityToken
#afirma.authorization.ks.path =
# Tipo del almacen anterior: JKS, PKCS12
#afirma.authorization.ks.type = PKCS12
#afirma.authorization.ks.password = usr
#afirma.authorization.ks.cert.alias = pss
#####
# UsernameToken
afirma.user = user
afirma.password = pass
#####

#####
# Datos del servidor (obligatorio)
```



```
#  
# El componente siempre realizará las peticiones al siguiente endpoint:  
# https://<afirma\_host>/afirmaws/services/<nombre\_del\_servicio>  
#####  
afirma.host = ws028.juntadeandalucia.es  
#####  
#####  
# Almacén de certificados de confianza (opcional)  
#  
# El componente presupone que el almacén de certificados de confianza está  
# incluido en el CLASSPATH de la aplicación, por lo que en 'afirma.truststore'  
# debe indicarse el nombre del fichero JKS (o bien, la ruta relativa al mismo  
# dentro del CLASSPATH). Si se establece el valor "default" se hace uso del  
# almacén de confianza de la JRE que ejecuta la aplicación (cacerts)  
#  
# Por ejemplo, una aplicación web (empaquetada en un WAR) cuyo almacén está  
# situado en el directorio WEB-INF/classes/almacenconfianza.jks, deberá  
# establecer la propiedad afirma.truststore de la siguiente manera:  
#  
# afirma.truststore = almacenconfianza.jks  
#  
# También se admite referenciar al almacén de confianza mediante su ruta absoluta.  
#  
#####  
afirma.truststore = default  
afirma.truststorePassword =  
#####  
#####  
# Autenticación Fachada de Tickets  
#  
# Es necesario incluir los siguientes datos si se va a hacer uso de la  
# autenticación mediante la fachada de tickets: servidor de autenticación y urls  
# de retorno  
#####  
# Host del servidor de la fachada SSL de autenticación por tickets  
afirma.tickets.auth.host = ws116.juntadeandalucia.es  
# URL del servlet de retorno definido en el fichero web.xml  
afirma.tickets.url.servlet = https://\[HOST\_APP\]/\[APP\_CONTEXT\]/ReturnAuthenticationServlet
```



```
# URL de vuelta de la aplicación cliente tras finalizar el proceso de autenticación  
afirma.tickets.url.app = https://\[HOST\_APP\]/\[APP\_CONTEXT\]/\[AUTH\_PATH\]
```

Las propiedades propias de la fachada de tickets son las siguientes:

- *afirma.tickets.auth.host*: host del servidor de la fachada SSL de autenticación por tickets.
- *afirma.tickets.url.servlet*: URL del servlet de retorno, para indicar a la fachada el punto de vuelta a la aplicación.
- *afirma.tickets.url.app*: URL de la aplicación donde vuelve el flujo de la aplicación web una vez completada la ejecución del servlet de retorno.

3.5. FICHERO WEB.XML

En el apartado se describen los servlets de llamada y retorno que coordinan el proceso de autenticación y que forman parte del componente `afirma-authentication-client`. Es requisito indispensable para la llevar a cabo la integración que dichos servlets sean definidos, como cualquier otro integrante de la aplicación web cliente, en el fichero `web.xml`. Se muestra a continuación un ejemplo del fichero `web.xml` con la definición de los servlets mencionados:

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://  
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee  
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">  
  
    <display-name>AuthenticationApp</display-name>  
  
    <welcome-file-list>  
        <welcome-file>index.jsp</welcome-file>  
    </welcome-file-list>  
  
    <servlet>  
        <servlet-name>CallAuthenticationServlet</servlet-name>  
        <servlet-class>  
            es.juntadeandalucia.afirma.servlet.CallAuthenticationServlet  
        </servlet-class>  
    </servlet>  
  
    <servlet-mapping>  
        <servlet-name>CallAuthenticationServlet</servlet-name>  
        <url-pattern>/CallAuthenticationServlet</url-pattern>  
    </servlet-mapping>
```



```
<servlet>
    <servlet-name>ReturnAuthenticationServlet</servlet-name>
    <servlet-class>
        es.juntadeandalucia.afirma.servlet.ReturnAuthenticationServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ReturnAuthenticationServlet</servlet-name>
    <url-pattern>/ReturnAuthenticationServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Las clases que implementan los servlets son los siguientes:

- Servlet de llamada: *es.juntadeandalucia.afirma.servlet.CallAuthenticationServlet*
- Servlet de retorno: *es.juntadeandalucia.afirma.servlet.ReturnAuthenticationServlet*

3.6. INTERFAZ COMPONENTE AFIRMA-AUTHENTICATION-CLIENT

Las aplicaciones web deben seguir los siguientes pasos para realizar la autenticación mediante el componente afirma-authentication-client:

1. Incluir las dependencias java en el classpath de la aplicación.
2. Establecer la configuración adecuada en el fichero *afirma.properties* e incluirlo en el classpath de la aplicación.
3. Definir los servlets de llamada y retorno en el fichero *web.xml*.
4. Invocar el servlet de llamada (*CallAuthenticationServlet*) desde algún punto de la aplicación.
5. Recuperar los datos devueltos por el servlet de retorno (*ReturnAuthenticationServlet*) que se incluyen en el atributo de sesión “*afirma_authentication_client_response*” .

En la documentación java-doc del componente pueden consultarse la descripción de las distintas clases que lo conforman.

3.7. PROCESAMIENTO DE RESPUESTA



El servlet de retorno (ReturnAuthenticationServlet) introduce en sesión la información necesaria para que la aplicación cliente decida autorizar el acceso del usuario. Concretamente, la información se registra en el atributo “afirma_authentication_client_response”, el cual debe asignarse a un objeto de tipo **ResultAuthenticationBean** para obtener los siguientes datos:

- Resultado del proceso de autenticación (éxito o error): Campo lógico que tiene el valor true si el proceso de autenticación ha sido correcto o false si el proceso ha fallado por algún motivo.
- Descripción del resultado del proceso: Incluye un texto legible y descriptivo del resultado del proceso de validación.
- Datos del certificado validado: Lista de objetos de tipo *CertificateInfo*. Dicho objeto contiene dos campos que, a modo de par clave valor, almacenan los distintos datos obtenidos del certificado del usuario autenticado. Los campos de la clase *CertificateInfo* son los siguientes:
 - FieldIdentity: Incluye el texto con la clave de un atributo del certificado.
 - FieldValue: Contiene el texto con la información del atributo indicado en su correspondiente *fieldIdentity*.
 - Certificado: Campo de tipo String que contiene el certificado del usuario autenticado codificado en Base 64.

En el siguiente apartado se muestra un ejemplo de cómo recuperar los datos de sesión y tratarlos mediante la clase **ResultAuthenticationBean**.

3.7.1. MÉTODO SEPARAR APELLIDOS

Se incluye un método en la clase **ResultAuthenticationBean** que sirve para separar una cadena formada por dos apellidos (ambos apellidos pueden ser compuestos). Como resultado devuelve cada apellido en una cadena.

- Cabeceras del método:

```
public String separarApellidos(String apellidos) throws AfirmaException;
```
- Parámetros de entrada:
 - apellidos: Cadena formada por dos apellidos
- Respuesta:
 - Objeto de tipo `String[]` con los dos apellidos por separado

3.8. EJEMPLO DE INTEGRACIÓN



La aplicación cliente requiere la inclusión de las dependencias descritas y la configuración de los ficheros afirma.properties y web.xml. Una vez establecida dicha configuración se debe ejecutar el servlet de llamada y recopilar la información del proceso que se incluye en la sesión que provee el servidor de aplicaciones.

A continuación se muestra un ejemplo de invocación al servlet de llamada desde una página JSP:

```
<a href="CallAuthenticationServlet" target="_self">Autenticación  
mediante certificados digitales basado en Tickets</a>
```

Al pulsar sobre el enlace se lanza el proceso de autenticación y la fachada SSL solicita la elección de un certificado a través del navegador web. Una vez seleccionado el certificado, el sistema continua con el proceso de autenticación y almacena el resultado en el atributo "afirma_authentication_client_response" .

A continuación se muestra un ejemplo de asignación del atributo "afirma_authentication_client_response" a un objeto del tipo ResultAuthenticationBean en una página JSP:

```
<%  
ResultAuthenticationBean result= new ResultAuthenticationBean();  
result= (ResultAuthenticationBean) session.getAttribute( "afirma_authentication_client_response");  
%>
```

Se muestran en la página JSP anterior el resultado de la autenticación y los datos del certificado del usuario mediante un sencillo bucle:

```
<%  
if(result!= null){  
    out.println( "\nTicket válido: "+ result.isValidTicket( ));  
    out.println( "\nDescripción resultado autenticación: "+result.getResultDescription( ));  
    if(result.getCertificateData( ) != null){  
        out.println( "\nDATOS DEL CERTIFICADO:");  
        Iterator<CertificateInfo>iterador =result.getCertificateData( ).iterator( );  
        while(iterador.hasNext( )){  
            CertificateInfo cer = (CertificateInfo) iterador.next( );  
            out.print( cer.getFieldIdentity( ) + "- ");  
            out.print( cer.getFieldValue( ) );  
            out.println( "\n");  
        }  
    }  
}
```



```
    }  
  
    }  
    if(result.getB64Certificate() !=null)  
    {  
        out.print( "\nCERTIFICADO BASE 64: "+result.getB64Certificate());  
    }  
}  
%>
```



4. CONFIGURACIÓN AVANZADA

4.1. PROPIEDADES DINÁMICAS

Con el objetivo de flexibilizar la configuración del componente afirma-authentication-client, éste permite ser configurado opcionalmente mediante un objeto de la clase **java.util.Properties**. Dicho objeto debe contener las propiedades de forma análoga a como se definen en el fichero `afirma.properties` (). Para establecer la configuración dinámica se debe invocar al método estático **"TicketClientBuilder.setConfiguration(afirmaProperties)"** cada vez que se desee modificar la configuración del componente.

A modo de ejemplo, se indica a continuación como debe invocarse el componente con esta configuración, previamente a la llamada del servlet, desde una página JSP:

```
<%  
  
    /*  
     * Definición del fichero de propiedades  
     */  
  
    Properties afirmaProperties = new Properties();  
    afirmaProperties.put( "afirma.idapp", "IDAPP");  
    afirmaProperties.put( "afirma.user", "user");  
    afirmaProperties.put( "afirma.password", "pass");  
    afirmaProperties.put( "afirma.host", "ws028.juntadeandalucia.es");  
    afirmaProperties.put( "afirma.truststore", "trustStore.jks");  
    afirmaProperties.put( "afirma.truststorePassword", "passTrustStore");  
    afirmaProperties.put( "afirma.tickets.auth.host", "ws116.juntadeandalucia.es");  
  
    afirmaProperties.put( "afirma.tickets.url.servlet",  
        "https://[HOST_APP]/[APP_CONTEXT]/ReturnAuthenticationServlet");  
    afirmaProperties.put( "afirma.tickets.url.app", "https://[HOST_APP]/[APP_CONTEXT]/[AUTH_PATH]");  
  
    // Se establece la configuración personalizada del componente  
    TicketClientBuilder.setConfiguration( afirmaProperties );  
  
    <a href="CALLAuthenticationServlet" target="_self">Autenticaci&ocute;n  
        mediante certificados digitales basado en Tickets</a>  
%>
```



4.2. PROPIEDADES DEL SISTEMA

Con el objetivo de flexibilizar la configuración del componente afirma-authentication-client, éste también puede ser configurado opcionalmente mediante propiedades del sistema, ya sea de manera programática o bien a través de parámetros en la máquina virtual.

A modo de ejemplo, se indica a continuación como debe invocarse el componente con esta configuración, previamente a la llamada del servlet, desde una página JSP:

```
<%  
  
    /*  
     * Definición de las propiedades  
     */  
  
    System.setProperty( "afirma.idapp", "IDAPP");  
    System.setProperty( "afirma.user", "user");  
    System.setProperty( "afirma.password", "pass");  
    System.setProperty( "afirma.host", "ws028.juntadeandalucia.es");  
    System.setProperty( "afirma.truststore", "trustStore.jks");  
    System.setProperty( "afirma.truststorePassword", "passTrustStore");  
    System.setProperty( "afirma.tickets.auth.host", "ws116.juntadeandalucia.es");  
  
    System.setProperty( "afirma.tickets.url.servlet",  
        "https://[HOST_APP]/[APP_CONTEXT]/ReturnAuthenticationServlet");  
  
    System.setProperty( "afirma.tickets.url.app", "https://[HOST_APP]/[APP_CONTEXT]/[AUTH_PATH]");  
  
    // Se establece la configuración personalizada del componente  
  
%>  
  
<a href="CallAuthenticationServlet" target="_self">Autenticación  
    mediante certificados digitales basado en Tickets</a>
```