Manual de Instalación, configuración e Integración de STORK para Proveedores de Servicio para Java

Resumen: Esta es un manual para la instalación, configuración e integración a la plataforma STORK para proveedores de servicio. También integradores de sistemas se pueden beneficiar de este manual. Este manual debe ser leído por administradores de sistemas e integradores de aplicaciones.

Historial del documento

Versión	*Fecha*	*Causa modificación*
0.1	15/09/2011	Versión inicial
0.2	17/10/2011	Actualizaciones
1.0	01/02/2012	Actualizaciones

Índice

HISTORIAL DEL DOCUMENTO **2**
ÍNDICE **3**
LISTA DE ABREVIATURAS **4**
RESUMEN EJECUTIVO **5**
1 INTRODUCCIÓN **6**
1.1 ESTRUCTURA DEL DOCUMENTO 6
2 ANTES DE EMPEZAR **7**
2.1 LA COMUNICACIÓN EN LA RED DE CONFIANZA STORK 7
2.2 KEYSTORE 7
2.3 CONFIGURACIÓN DEL SERVIDOR 8
2.3.1 TOMCAT 5 8
2.3.2 TOMCAT 6 8
2.3.3 JBOSS 5 8
2.3.4 GLASSFISH V3 8
2.3.5 WEBLOGIC 9
3 DESPLIEGUE RÁPIDO **10**
3.1 CONFIGURACIONES 10
3.2 COMPILACIÓN Y DESPLIEGUE 10
4 CONFIGURAR STORKDEMOSP **11**
4.1 FICHEROS DE CONFIGURACIÓN 11
4.1.1 SP.PROPERTIES 11
4.1.2 SAMLENGINESIGNCONFIG.XML 14
4.2 INICIAR LA APLICACIÓN STORKDEMOSP 15
5 STORK API **20**
5.1 JARS 20
5.2 EJEMPLOS DE USO 20
5.2.1 GENERANDO PETICIONES DE AUTENTICACIÓN EN STORK 20
5.2.2 LEER Y VALIDAR RESPUESTAS DE AUTENTICACIÓN DE STORK 20
6 CONJUNTO DE CERTIFICADOS DE PRUEBAS **22**
7 PREGUNTAS FRECUENTES **23**
8 ANEXO: PETICIÓN DE ALTA COMO PROVEEDOR DE SERVICIOS CON ACCESO A LA PLATAFORMA STORK **24**

Lista de abreviaturas

<Abreviatura>	<Explicación>
STORK	Secure idenTity acrOss boRders linKed
PEPS	Pan European Proxy Server
SP	Service Provider, Proveedor de servicio

Resumen ejecutivo

Este documento facilita la instalación, configuración e integración rápidas de STORK en aplicaciones y provee información detallada para este fin. Con este manual el administrador e integrador deben ser capaces de construir y desplegar su aplicación habilitada para STORK con facilidad.

Este manual supone que en su sitio Web ya hay un servidor de aplicaciones funcionando, para el cual se incluyen configuraciones básicas. Posteriormente este manual describe lo que necesita saber de diferentes configuraciones de varios proyectos, para poder construir la aplicación con éxito.

1 Introducción

1.1 Estructura del documento

Este documento se divide en varias partes que se describe a continuación.

En primer lugar se presenta un conjunto de pasos para configurar su servidor de aplicaciones (Tomcat, JBoss, Glassfish o WebLogic) y dar servicio a la aplicación demo.

Luego se describe en una guía rápida como desplegar y ejecutar la aplicación de ejemplo en unos minutos.

Posteriormente se explican todas las configuraciones en detalle, para facilitarle cambiar aspectos específicos.

Finalmente se presenta la sección de preguntas frecuentes.

2 Antes de empezar

Para la confección de este documento y la explicación de los diferentes pasos se elige el servidor de aplicaciones Tomcat. Se dan también ciertas pinceladas sobre su adaptación a JBoss, Glassfish y WebLogic, pero asegúrese de adaptar los pasos a su específico entorno.

Antes de empezar, asegurase tener de Tomcat 5.5 o superior instalado en su servidor, y que incluye la librería de **OpenSAML**. Los ficheros para construir (build) el proyecto tienen formato de **Maven2**, que también debe estar instalado. Y finalmente requiere Java SDK 1.6 o superior.

En este documento se utilizarán algunas variables de entorno que se explican a continuación.

- `$TOMCAT_HOME` – Directorio base del servidor Tomcat. (por ejemplo `/home/user/apps/apache-tomcat-5.5.28`)
- `$JBOSS_HOME` – Directorio base del servidor JBoss. (por ejemplo `/home/user/apps/jboss-5.1.0.GA`)
- `$SERVER_CONFIG` – Nombre de configuración del servidor JBoss (por ejemplo `default`)
  Si quiere usar la configuración del servidor "default", el *path* completo sera: `/home/user/apps/jboss-5.1.0.GA/server/default`
- `$GLASSFISH_HOME` – Directorio base del servidor Glassfish. (por ejemplo `/home/user/apps/glasshfishv3);`

2.1 La comunicación en la red de confianza STORK

Toda comunicación con un componente de la Red STORK se realiza a través del intercambio de tokens que pasan previamente por el navegador del ciudadano. De esta manera cada Servidor en la Plataforma STORK sólo responde al ciudadano desde el que ha recibido una petición STORK.

Para la confianza en los diferentes tokens, todo receptor de una petición o una respuesta debe confiar en el firmante de dicho token, así como todo emisor de una petición o una respuesta debe asegurarse de que el receptor conoce quién es él (tiene su certificado público).

2.2 Keystore

El paquete de instalación incluye dentro del proveedor demo (a partir de ahora lo llamaremos StorkDemoSP) un almacén de certificados PKI, en el cual hay 3 certificados:

- un certificado de ejemplo con el que el StorkDemoSP va a firmar sus peticiones antes de enviárselas al Nodo STORK de pruebas.
- el certificado público del Nodo STORK español de pruebas (necesario para confiar en las respuestas de dicho nodo)
- el certificado público del Nodo STORK español de producción (necesario para confiar en un fichero XML firmado del que hablaremos más adelante).

Este almacén está en el fichero "`storkKeyStore.jks`", y tiene que ser copiado al directorio que elija, pero asegúrese que esta ubicación es la indicada en la *property* "keystorePath" en el fichero
`STORKDemoSP\src\main\resources\SamlEngineSignConfig.xml`.

Para más información consulte la sección 0

Ficheros de configuración .

2.3 Configuración del servidor

2.3.1 Tomcat 5

Extrae el contenido del fichero OpenSAML zip y copia las siguientes librerías a `$TOMCAT_HOME/common/endorsed`

```
endorsed\xml-apis-2.9.1.jar
endorsed\resolver-2.9.1.jar
endorsed\serializer-2.9.1.jar
endorsed\xalan-2.7.1.jar
endorsed\xercesImpl-2.9.1.jar
```

2.3.2 Tomcat 6

Crea el directorio `endorsed` en `$TOMCAT_HOME`.
Crea un directorio `shared` en `$TOMCAT_HOME` y un subdirectorio `lib` in `$TOMCAT_HOME\shared`

Modifica el fichero `$TOMCAT_HOME\conf\catalina.properties y cambia la` property `shared.loader para que tenga el valor:`
`shared.loader=${catalina.home}/shared/lib/*.jar`

Abre `OpenSAML zip` y copia las siguientes librerías a `$TOMCAT_HOME/endorsed`

```
endorsed\xml-apis-2.9.1.jar
endorsed\resolver-2.9.1.jar
endorsed\serializer-2.9.1.jar
endorsed\xalan-2.7.1.jar
endorsed\xercesImpl-2.9.1.jar
```

2.3.3 JBoss 5

Abre `OpenSAML zip` y copia las siguientes librerías a `$JBOSS_HOME/lib/endorsed`

```
endorsed\xml-apis-2.9.1.jar
endorsed\resolver-2.9.1.jar
endorsed\serializer-2.9.1.jar
endorsed\xalan-2.7.1.jar
endorsed\xercesImpl-2.9.1.jar
```

2.3.4 Glassfish V3

Abre `OpenSAML.zip` y copia las siguientes librerías a `$GLASSFISH_HOME/glassfish/lib/endorsed`

```
endorsed\xml-apis-2.9.1.jar
endorsed\resolver-2.9.1.jar
endorsed\serializer-2.9.1.jar
endorsed\xalan-2.7.1.jar
```

```
endorsed\xercesImpl-2.9.1.jar
```

2.3.5 Weblogic

Open the OpenSAML zip and copy the following libs to `$GLASSFISH_HOME/glassfish/lib/endorsed`

3 Despliegue rápido

Este despliegue rápido le permite configurar, compilar y ejecutar el proyecto en unos minutos, con solo una instancia del servidor de aplicaciones. Acuérdese que su servidor tiene que estar configurado y preparado, tal y como describe el capítulo anterior. Encontrará más información acerca de ficheros de configuración, atributos etc. en el capítulo siguiente.

3.1 Configuraciones

Edite el fichero STORKDemoSP\src\main\resources\sp.properties y cambie la siguientes *property*:

- **sp.return**=https:// **inserte.su.ip.aqui** /StorkDemoSP/ReturnPage

Así ha configurado la aplicación para funcionar en su servidor en Internet.

3.2 Compilación y despliegue

Tiene que compilar, instalar y desplegar los proyectos en la siguiente secuencia:

1. En la carpeta STORKDemoSP teclee:
   - mvn clean Packaged
   - despliegue el proyecto resultante (target/STORKDemoSP.war) en su propio Servidor de Aplicaciones e arránquelo.

Finalmente entre con su navegador en la página siguiente:
https://**inserte.su.ip.aqui**/StorkDemoSP

4 Configurar StorkDemoSP

4.1 Ficheros de configuración

La aplicación Demo incluye ficheros de configuración que pueden ser modificados. En esta sección se explica que ficheros son, y el significado de cada *property*.

4.1.1 sp.properties

El fichero sp.properties permite la configuración principal de la aplicación.

`sp.name`	Nombre del proveedor de servicio
`sp.sector`	Sector del proveedor de servicio, por ejemplo Government
`sp.application`	Nombre de la aplicación
`sp.country`	País del proveedor de servicio; "ES" en nuestro caso
`sp.environment`	Entorno al que se conecta dentro de STORK. En el caso español hay sólo dos opciones: TEST (pruebas) y PROD (producción).
`sp.qaalevel`	La calidad del identificador del ciudadano que su aplicación requiere
`sp.return`	URL donde el PEPS español debe enviar la respuesta a la petición
`sp.euromap`	Si esta opción está a "true" la aplicación presentará un mapa con los países en los que es posible la autenticación STORK, en caso de "false" presentará una lista con esos mismos países.

Atributos del ciudadano que su aplicación quiere obtener en su petición de autenticación:

`attribute.number`	Número de atributos a pedir
`attributeX.name`	Nombre del atributo
`attributeX.mandatory`	Indica si el atributo es obligatorio u opcional
`attributeX.value`	Parámetro opcional para pedir un valor del atributo, como es el caso con isAgeOver

Países que desea filtrar para que no sean presentados:

`discountry.number`	Número de países a ser filtrados
`discountryX`	Código del país número X a filtrar

Ahora se debe elegir entre dos opciones a la hora de obtener los países a presentar en su aplicación para que los ciudadanos los elijan:

1.- Control automático de países: Se utiliza un fichero XML que el nodo español de Producción facilita y ha de ser descargado diariamente (pues la información puede cambiar).

2.- Control manual de países: Se indican en este fichero de propiedades uno a uno los países

`sp.versioncontrol`	Si es “true” estaremos en el caso del control automático de países. Y si es “false”, en el caso del control manual

OPCIÓN 1)

En el caso del "sp.versioncontrol=true", necesitaremos dar valor a otra property:

`versioninfofile`	Dirección local donde se ha descargado el fichero de control de países. Este fichero contiene tanto información de países como información acerca de qué atributos son capaces de proporcionar cada país.

OPCIÓN 2)

En el caso del "sp.versioncontrol=false", necesitaremos dar valor a estas otras properties:

`sp.pepsurl`	Es la URL del nodo español al que conecta la aplicación.

Lista de países disponibles para este StorkDemoSP:

`country.number`	Número de países a los que se puede conectar
`countryX.name`	Código del país X

4.1.2 SamlEngineSignConfig.xml

Este fichero es utilizado para configurar los módulos de firma y validación de los *tokens* SAML, al enviar peticiones y recibir respuestas.

`keystorePath`	*Path* donde se encuentra el *keystore*
`keyStorePassword`	Contraseña del keystore
`keyPassword`	Contraseña del certificado con clave privada
`issuer`	Emisor del certificado con clave privada
`serialNumber`	Número de serie del certificado a usar para firmar
`keystoreType`	Tipo del keystore (JKS)

En el *keystore* debe haber (por lo menos) un certificado con clave privada para firmar las peticiones SAML. También tiene que incluir el certificado que el nodo STORK español utiliza para firmar las respuestas.

4.2 Iniciar la aplicación StorkDemoSP

Una vez que se han satisfecho los requerimientos técnicos listados en el apartado 2 y que se haya configurado el servidor y los ficheros de configuración, el proveedor de servicios puede comenzar a trabajar con la aplicación.

Lanza un web browser y navega hasta el siguiente URL: https://**inserte.su.ip.aqui**/StorkDemoSP. Si el servidor está listado en otro puerto, se puede cambiar el URL.

Una vez que acceda, se encontraría con una página similar a esta:

Cuando un ciudadano accede al programa STORK, el proveedor de servicio hará una petición de datos del usuario, sobretodo cuando se trata de la primera vez que accede. Estos datos se extraen de sus credenciales o de bases de datos verificadas y mantenidas por las autoridades competentes, de tal manera que el proveedor de servicios pueda fiarse totalmente de los datos recibidos. Además, la calidad de estos datos está ligada al nivel de garantía de calidad de las credenciales requeridas por el proveedor de servicios; algunos de ellos pueden solicitar unos datos de calidad alta, mientras que otros se conformarán con un nivel medio o más bajo.

El proveedor de servicios se basa en los resultados obtenidos de la autenticación online para establecer la identidad de un subscriptor/usuario para realizar la transacción. El proveedor de servicios y el verificador pueden ser la misma entidad o pueden ser entidades diferentes. Si son entidades diferentes, el proveedor de servicios recibe una confirmación por parte del verificador.

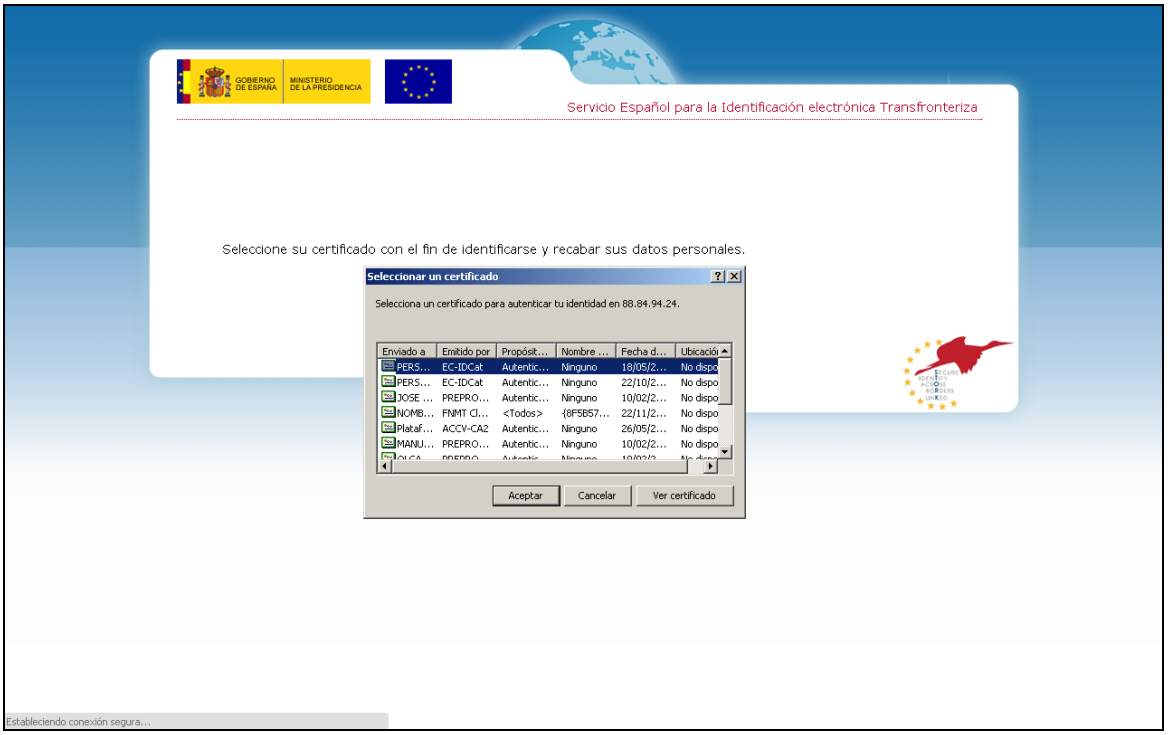
El Proveedor de Servicios es responsable de validar que la confirmación vino de un verificador de confianza. Cuando estas confirmaciones indican la fecha de su creación o atributos asociados al demandante, el Proveedor de Servicios es también responsable de verificar esta información.

El Proveedor de Servicios determina que credenciales son requeridas para proporcionar al demandante o subscriptor acceso. Es por lo tanto el Proveedor de Servicios el que determina el nivel de autenticación para acceder a los datos.

Examinemos el siguiente ejemplo que detalla el funcionamiento del flujo de una petición:

Si un usuario desea acceder a un Proveedor de Servicios de un Estado Miembro B, el usuario comienza el proceso de autenticación seleccionando “autenticarse” con el proveedor de servicios. El proveedor de servicios envía entonces una petición de autenticación y la información relativa al nivel de la QAA (Quality Authentication Assurance) requerida por el Estado Miembro B para que proceda con la verificación del demandante. El Estado Miembro B pregunta entonces al usuario qué Estado Miembro le expidió la Identidad electrónica que va a utilizar para autenticarse y le proporciona un listado de Estados Miembros. El usuario selecciona uno de este listado; de acuerdo con esta selección, tanto la demanda de autenticación como el nivel de servicio de QQA requerido son enviados del Estado Miembro B al Estado Miembro seleccionado por el usuario (en este caso lo llamaremos Estado Miembro A).

El Estado Miembro A proporciona al usuario un listado de Identidades electrónicas que cumplen con la autenticación demandada y con los requisitos del nivel de servicio de QAA. El usuario escoge en ese momento la Identidad electrónica con la que quiere autenticarse.



La validación de la Identidad electrónica seleccionada se lleva a cabo basándose en la interacción existente entre el Estado Miembro A y el usuario; el Estado Miembro A demanda al Estado Miembro IDP (Proveedor de Identificación) la validación de la Identidad electrónica. Se pueden dar varios casos:

- si no se puede realizar la validación, se informa al usuario y el proceso termina. El fín del proceso consiste en el envío por parte del usuario de un token SAML informado del tipo de error producido.

- si se consigue validar su Identidad electrónica, el Estado Miembro A crea una confirmación que es presentada al usuario para que de su consentimiento, paso necesario ya que en algunos Estados Miembros esta confirmación es considerada como datos personales.

- si el usuario niega su consentimiento, el proceso termina (con el envío del un token SAML al SP informando de la negación del consentimiento).

- si lo da, se envía al Estado Miembro B que a su vez se pone en contacto con el proveedor de servicios que responde afirmativamente a la demanda del usuario. En este caso el SP recibe un token SAML con la información solicitada sobre el ciudadano.

5 STORK API

5.1 JARs

El API de STORK para proveedores de servicio está formado por dos JARes:

- saml-engine-xxx.jar: crea y valida mensajes SAML
- stork-commons-xxx.jar: gestiona la extracción de datos del mensaje SAML.

5.2 Ejemplos de uso

5.2.1 Generando peticiones de autenticación en STORK

```
//Creando petición de autenticación para STORK --------------------------------
STORKAuthnRequest authnRequest = new STORKAuthnRequest();

//Llenando los campos de la petición ------------------------------
authnRequest.setIssuer(..);
authnRequest.setDestination(..);
authnRequest.setProviderName(..);
authnRequest.setQaa(Integer.parseInt(..));
authnRequest.setAssertionConsumerServiceURL(..);
authnRequest.setSpSector(..);
authnRequest.setSpInstitution(..);
authnRequest.setSpApplication(..);
authnRequest.setSpCountry(..);
authnRequest.setSPID(..);
authnRequest.setCitizenCountryCode(..);        //País pedido por el ciudadano
//Loading Stork attributes to request
IPersonalAttributeList attributelist = new PersonalAttributeList();
       PersonalAttribute attribute1 = new PersonalAttribute();
       attribute1.setName(..);                 //Nombre del atributo
       attribute1.setIsRequired(true);         //Atributo obligatorio
       attributelist.add(attribute1);
       // The same for more attributes
authnRequest.setPersonalAttributeList(attributelist);
//Obtener la referencia del motor SAML ------------------------------------
STORKSAMLEngine engine = STORKSAMLEngine.getInstance("SP");
authnRequest = engine.generateSTORKAuthnRequest(authnRequest);
byte[] token = authnRequest.getTokenSaml();
SAMLRequest = PEPSUtil.encodeSAMLToken(token);
```

Then, this SAMLRequest must be placed into a HTML form:

```
<form name="redirectForm" method="post" action="stork.country.server.URL">
       <input hidden name="SAMLRequest" value="above-SAMLRequest"/>
       <input name="country" value="selectedCitizenCountry"/>
</form>
```

5.2.2 Leer y validar respuestas de autenticación de STORK

En primer lugar recibimos el formulario con el campo "SAMLResponse".

Luego:

```
//Descodifocar la respuesta SAML recibida
byte[] decSamlToken = PEPSUtil.decodeSAMLToken(SAMLResponse);

// Obtener la referencia del motor SAML
STORKSAMLEngine engine = STORKSAMLEngine.getInstance("SP");


//Validar respuesta de autenticación SAML
STORKAuthnResponse authnResponse = null;
authnResponse = engine.validateSTORKAuthnResponse(decSamlToken,
(String)request.getRemoteHost());

//Leer la respuesta
if(!authnResponse.isFail()){

  //Obtener atributos
  IPersonalAttributeList attributeList=authnResponse.getPersonalAttributeList();
  setAttrList(new ArrayList<PersonalAttribute>( attributeList.values()));
  Iterator<String> list = attrList.iterator();
  while ( attrList.hasNext() ){
       PersonalAttribute attribute=attrList.next();
       System.out.println( "Attribute: "+ attribute.name);
       System.out.println( "Value: "+ attribute.value.get(0));
       System.out.println( "Status: "+ attribute.status);
    }
}else{
  System.out.println ("Saml Response is fail:" + authnResponse.getMessage());
}
```

6 Conjunto de certificados de pruebas

La aplicación Demo es distribuida con un conjunto de certificados, para facilitarle la realización de pruebas transfronterizas.[1]

Nombre del fichero del certificado	País	Contraseña
Ana Vzorec 02.pfx	Eslovenia	Ana Vzorec 02
Alice Auth*.p12	Bélgica	BelgaCom.
Stork_Active.p12	Portugal	#ptcert!
sanmiguel.p12	España	1234

[1] Estos certificados no son válidos en entorno de producción

7 Preguntas frecuentes

Nota que hay más preguntas frecuentes en el documento "Introducción a STORK para proveedores de Servicios"

¿Para qué sirve el control de versiones?

Mediante el control de versiones la autoridad española de STORK puede conocer la versión del software que Vd tiene instalado y por lo tanto puede saber si una modificación que está planteando implantar es compatible o no con sus instalaciones. De esta forma se evitan problemas de incompatibilidades.

Así mismo, el selector de países de su instalación se modificará automáticamente, si un nuevo país se conecta a la plataforma STORK, si cumple con sus requisitos de atributos necesarios y calidad de la autenticación.

¿Y si no quiero aceptar credenciales de algún país?

Si un país sí cumple con los requisitos de calidad y de atributos disponibles, pero por cualquier otra razón no quiere aceptar sus credenciales, puede excluirlas mediante "`discountryX`" (disallow country), como indicado en 4.1.1. Una razón para no aceptar credenciales españolas a través de STORK podría ser que Vd ya tiene implantada la aceptación de credenciales españolas en modo nativo. O se puede excluir las credenciales de DE y GR pues su número de identificación no es persistente.

¿Dónde puedo acudir a soporte?

mailto:stork@indra.es

8 ANEXO: Petición de Alta como Proveedor de Servicios con acceso a la Plataforma STORK

Fecha_Petición	*Fecha actual*
Nombre	*Nombre del Proveedor de Servicio*
Descripción	
Entorno	*PRE o PRO*
Persona_Contacto	*Persona de Contacto del Proveedor de Servicio (teléfono y email)*
Cerificado_SP	*Parte pública del certificado usado por el Proveedor de Servicio en la firma SAML*
-----BEGIN CERTIFICATE----- -----END CERTIFICATE-----	
Fecha_ prevista	*Fecha de Inicio prevista para pruebas en el entorno indicado*

(SP: Proveedor de Servicios)

Nota que nombre del proveedor de servicio y el certificado de firma tienen que coincidir con el "`sp.name`" configurado en 4.1.1 y el certificado en el keystore indicado por "`keystorePath`" configurado en 4.1.2, y almacenado con alias sp.name. El certificado en el keystore tiene que incluir su clave privada, en este formulario sólo hace falta la parte pública del mismo.