API del servicio web del Hub

Hub ePhyto de la CIPF

V1.12

FAO/CIPF - Público

Índice

Perfil del documento

Autor:	CICE-ONU
Propietario:	CICE-ONU
Cliente:	FAO/CIPF
Número de documento:	1.10

Historial de revisiones

Fecha de la próxima revisión: N/A

Versión:	Quién:	Qué es:	Cuándo:
1.0	CICE-ONU	Documento principal	(12/12/2016).
1.1	CICE-ONU	Actualizaciones después de la reunión de Ginebra del Comité técnico del proyecto ePhyto	(22/03/2017).
1.2	CICE-ONU	Examen de la iteración 2	(31/07/2017).
1.3	CICE-ONU	Examen de la iteración 3	(11/09/2017).
1.4	CICE-ONU	Revisión después de la reunión de Valencia del Comité técnico del proyecto ePhyto	(03/10/2017).
1.5	CICE-ONU	Se añade la muestra del código del cliente Java	(24/10/2017).
1.6	CICE-ONU	Recibir a través de la aplicación del ejemplo de PUSH añadido	(16/11/2017).
1.7	CICE-ONU	Url revisadas de la consola de administración del Hub	(04/01/2018).
1.8	CICE-ONU	Actualizaciones de la versión de marzo de 2018	(22/03/2018).
1.9	CICE-ONU	Actualizaciones de la versión de abril de 2018	07 de mayo, 2017
1.10	CICE-ONU	Revisión de varios enlaces externos	1º de junio de 2018
1.11	CICE-ONU	Versión del 18 de julio	16 de julio de 2018
1.12	CICE-ONU	Versión del 18 de octubre	9 de noviembre de 2018

Distribución

Este documento se ha distribuido como sigue, a partir de la versión 1.10 se publica y distribuye como parte de la comunicación del Hub, y reside en la consola de administración del mismo.

Nombre	Título	Fecha de publicación	Versión
CIPF	HUB Web Service API	(28/03/2017).	V1.1_pronta publicación
CIPF	HUB Web Service API	(01/08/2017).	V1.2_pronta publicación
CIPF	API del servicio web del Hub	(20/09/2017).	1.3
CIPF	API del servicio Web del Hub	(12/10/2017).	1.4
Usuarios del Hub	API del servicio Web del Hub	(20/10/2017).	1.4
CIPF	API del servicio Web del Hub	(24/10/2017).	1.5
CIPF	API del servicio Web del Hub	(16/11/2017).	1.6
CIPF	API del servicio Web del Hub	(04/01/2018).	1.7
Usuarios del Hub	API del servicio Web del Hub	(22/03/2018).	1.8
CIPF	API del servicio Web del Hub	07 de mayo, 2017	1.9

Mapa del documento

A continuación figuran las mejoras previstas de este documento

Característica
Incluir el diagrama de la secuencia para describir el tipo de flujo PUSH de recepción

1. Introducción

1.1 Propósito

Este documento describe el servicio web del Hub de la CIPF. Se deberá utilizar como guía para utilizar los componentes necesarios del software del cliente para conectarse con el Hub.

Nota: Este documento es una versión inicial y podrá actualizarse durante la fase piloto del proyecto del Hub. Las actualizaciones se presentarán siguiendo el proceso estándar de administración de cambios. La última versión estará disponible en el repositorio de documentos en https://www.ephytoexchange.org/doc/HUB_Web_Service_API.pdf

1.2 Destinatarios y lecturas recomendadas

Los destinatarios de este documento son los desarrolladores y arquitectos del sistema de las ONPF, que evaluarán y harán disponibles los componentes para conectarse con el Hub. También se utilizará para la creación de la interfaz entre el Hub de la CIPF y el Sistema nacional genérico (GeNS), y servirá como prototipo de la documentación de la utilización del Hub. Este documento deberá leerse junto con la Especificación de los requisitos de software del Hub de ePhyto y está totalmente relacionado con la utilización de los servicios web del Hub.

1.3 Bibliografía

- ePhyto HUB Software Requirements Specification
- https://www.ippc.int/en/ephyto/
- https://www.ippc.int/en/ephyto/ephyto-technical-information/

2. Conexión al Hub

- El primer paso en este proceso es enviar una solicitud de registro a la ONPF mediante https://www.ephytoexchange.org/onboard
- Tras la validación y confirmación del contacto oficial de la CIPF del país indicado, se crea automáticamente la cuenta del usuario (mediante el correo electrónico del contacto indicado) y se envía la acreditación.
- Después de obtener acceso a la consola de administración del Hub, el administrador se pondrá en contacto con el punto focal al que enviará la última versión de este documento y apoyará a la ONPF para la configuración inicial de la UAT (prueba de aceptación por el usuario) del sitio, en la que se puede probar la aplicación y validarla antes de la versión final para producción.

3. Soporte Técnico

Si se presentan problemas durante la fase de pruebas o hay otras consultas técnicas relacionadas con las pruebas del Hub, es posible plantearlos a través del portal del Hub: https://www.ephytoexchange.org/support *(solo usuarios registrados)*

También alentamos la utilización de la herramienta de colaboración para obtener respuestas rápidas y compartir experiencias. *(Solo usuarios registrados)*

Para consultas generales sobre el Hub, ir a https://www.ephytoexchange.org/support

Si surgen problemas durante la prueba de la ejecución de los componentes necesarios para conectarse al Hub, se puede formular una solicitud de apoyo técnico en la consola de administración (https://www.ephytoexchange.org/AdminConsole) siguiendo el enlace disponible en el menú, una vez iniciada la sesión.

El sistema enviará un correo electrónico al equipo técnico que responderá a la consulta.

Se recomienda mirar primero en el área de colaboración de la consola de administración como posible fuente de información.

4. Sistemas de servicio web del Hub

4.1 Entorno de pruebas (UAT)

El entorno de pruebas (UAT) es un sistema activo con la última versión del sistema que está constantemente disponible para probar la ejecución de la conexión de la aplicación del cliente al Hub.

Es posible proporcionar certificados autofirmados y una acreditación especial a fin de facilitar las actividades.

4.1.1 URL para hacer las pruebas

Es posible acceder a la UAT del Hub/entorno de pruebas desde la siguiente URL:

(https://uat-hub.ephytoexchange.org/hub/DeliveryService?wsdl WSDL del servicio web)

(el endpoint del servicio web sólo aceptará la autentificación a través de certificados)

https://uat-hub.ephytoexchange.org/AdminConsole(interfaz de administración)

La acreditación para el inicio de sesión en la consola se proporciona durante el proceso de conexión de la ONPF.

4.1.2 Certificados para la autentificación del cliente del servicio Web

Durante la fase de prueba, la autentificación del cliente del servicios web utilizará certificados autofirmados. Las ONPF pueden emitir sus certificados con el comando "keytool" que figura en el Java Development Kit (JDK) o pueden solicitar a la CICE-ONU que proporcione un certificado de muestra para la ONPF que esta pueda utilizar.

Con la versión de marzo de 2018 el administrador de la ONPF puede tener acceso a la consola de administración del Hub y poner al día los certificados públicos que se utilizarán para autentificar la aplicación cliente de la ONPF al utilizar los servicios

4.1.2.1 Solicitar un certificado de prueba al CICE-ONU

La ONPF necesita proporcionar la información necesaria para crear la *denominación distinguida X.500* para el certificado:

- Nombre común
- Unidad de organización
- Nombre de la organización
- Localidad (ciudad)
- Estado
- Código de dos letras del país

El CICE-ONU enviará un archivo en formato PKCS12 con el certificado, y subirá la clave pública desde la consola de administración del Hub al perfil de la ONPF.

4.1.2.2 Generar un certificado autofirmado de prueba

Los certificados se pueden generar con el comando "keytool" proporcionado por el JDK. Una vez que la ONPF ha generado el certificado, tiene que enviar la clave pública al CICE-ONU para que pueda añadirse al *truststore* del servidor .

Ejemplo de generación de certificados para una entidad ONPF ubicada en Londres, Reino Unido, con una validez de diez años:

```
C:\certificates>keytool -genkey -alias nppo1 -keyalg RSA -keysize 1024 -keystore nppo.keystore -validity 3650 -keypass nppo1pass -storepass nppoStore1pass
¿Cuál es su nombre y apellido?
  [Desconocido]:  www.nppo.mycountry
```

```
¿Cuál es el nombre de su unidad organizativa?
  [Desconocido]:  ONPF
¿Cuál es el nombre de su organización?
  [Desconocido]:  ONPF-MiPaís
¿Cuál es el nombre de su ciudad o localidad?
  [Desconocido]:  Capital
¿Cuál es el nombre de su estado o provincia?
  [Desconocido]:
¿Cuál es el código de dos letras del país para esta unidad?
  [Desconocido]:  MC
Is CN=www.nppo.mycountry, OU=NPPO, O=NPPO-MyCountry, L=Capital, ST=Unknown, C=MC correct?
  [no]:  sí
```

Ejemplo de exportación de clave pública desde el almacén de claves:

```
C:\certificates>keytool -export -keystore nppo.keystore -alias nppo1 -file nppo1.cer -keypass nppo1pass -storepass nppoStore1pass
```

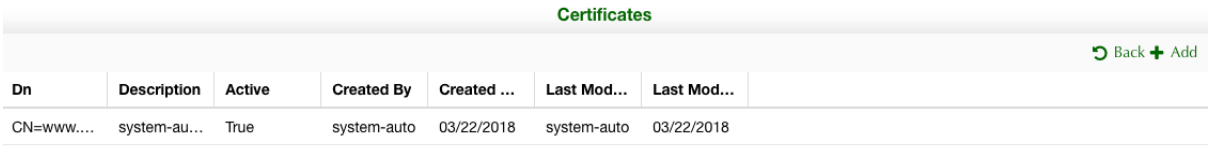
Una vez generada, la ONPF puede cargar y configurar el certificado en la consola de administración del Hub, con los siguientes pasos:

1) Entre en la consola de administración y vaya al Perfil de la ONPF 

2) Abra la vista de certificados con el enlace que figura en la parte inferior del Perfil de la ONPF 

3) Los certificados de cliente se enumeran en la vista (ver captura de pantalla más abajo), se pueden definir como no activo, mediante el botón Agregar 

se puede cargar el nuevo certificado (la consola del Hub sólo necesitará la clave pública en forma de un archivo .cer)



4.2 Entorno de producción

Se puede tener acceso al entorno de producción del Hub desde las siguientes URL:

(https://hub.ephytoexchange.org/hub/DeliveryService?wsdl WSDL del servicio web)

(el endpoint del servicio web sólo aceptará la autentificación de certificados)

https://www.ephytoexchange.org/AdminConsole (interfaz de administración)

4.3 Autentificación

La autentificación en el servicio web admite certificados de cliente TLS 1.1 y TLS 1.2 que estén asociados a cada país que tenga acceso al Hub. Los certificados X509 son la acreditación del cliente. Cada aplicación conectada tendrá un certificado definido, expedido por una autoridad de certificación reconocida que autentificará la aplicación cliente al Hub en el protocolo HTTPS. Los detalles de la implementación de seguridad están fuera del alcance de este documento, pero figuran en el documento de referencia especificado de requisitos del Hub.

El Hub sólo aceptará "sobres" en los que el campo "De" (descrito a continuación) coincida con el certificado TLS de conexión de la ONPF.

5. Esquemas XML del Hub

5.1 Esquema

El esquema del servicio web del Hub está compuesto por un gran número de entidades, algunas de ellas forman parte de la definición de ePhyto y se describirán con myor detalle en cada operación del servicio web. Consulte a continuación la lista de los principales elementos:

1) Encabezado de los sobres
2) Contenido de los sobres
3) Sobre de ePhyto

El WSDL definido en este documento (sección 6) tiene varias operaciones; apoyadas principalmente por las siguientes entidades:

a. Encabezado de los sobres
b. Sobre= encabezado + contenido
c. Sobre de ePhyto = encabezado = Certificado SPS
d. Conjunto de encabezados de sobres
e. Conjunto de sobres
f. HUBTrackingInfo
g. ONPF
h. Resultado de la validación

5.1.1 Encabezado de los sobres

El elemento del encabezado del sobre se utiliza para intercambiar información sobre los certificados ePhyto sin ver o procesar el contenido del certificado.

El Hub se preparará para verificar la correcta utilización de esos códigos y señalar los errores de comunicación cuando estos atributos no cumplan las normas. Esta será una característica del software del Hub.

Durante la interacción con el Hub no es obligatorio establecer todos los elementos en el encabezado. Sin embargo, se requiere un mínimo de elementos específicos.

El encabezado del sobre tiene los siguientes elementos:

- **De:** ISO 3166-1 Alfa código del país de dos letras, del país exportador
- **Para:** ISO 3166-1 Alfa código del país de dos letras, del país importador

- **Tipo de certificado (CertificateType):** Este es el código de la CEPE para los tipos de certificados. Para la aplicación de la CIPF, el Hub deberá verificar que el código de tipo corresponda a los dos siguientes números únicamente.
  - 851 para Phyto
  - 657 para Re-Export Phyto
- **Estado del certificado (CertificateStatus):** Este es el código de la CEPE para el estado del certificado. Para la aplicación de la CIPF, el Hub verificará que el código del estado corresponda a los dos siguientes números únicamente.
  - 70: Emitido
  - 39: Aprobado
  - 40: Retirado
  - 41: Rechazado
- **Número de certificado de la ONPF (NPPOCertificateNumber):** Para su propia referencia, la ONPF **exportadora** puede insertar en este campo el número de certificado ePhyto que contiene el sobre. Esto permitirá que el sistema nacional de la ONPF ajuste un certificado con el número de seguimiento del Hub (HubTrackingNumber) en su propio sistema nacional. Además, la interfaz de usuario del Hub mostrará también este número junto con el estado de la entrega. Este elemento es multilingüístico, permitir a la ONPF exportadora utilizar el idioma de su elección. Esto se limita a 1000 caracteres.
- **Número de seguimiento del Hub (HUBTrackingNumber):** Es un identificador único que el Hub asignará a cada sobre al recibirlo por primera vez. El sistema de la ONPF posteriormente puede consultar al Hub respecto a este identificador; para obtener información sobre la entrega de cualquier certificado específico identificado por el número de seguimiento del Hub. El tamaño de este elemento puede tener 50 caracteres.
- **Información de seguimiento del Hub (HUBTrackingInfo):** Este elemento tiene uno de los siguientes cuatro códigos de estado; indica el estado de la entrega del sobre en el Hub:
  - **Entrega pendiente** (**PendingDelivery**): significa que el sobre sigue en el Hub y todavía no se entrega. Además, el período de caducidad de la fila no ha terminado, el Hub todavía tiene el sobre.
  - **Entregado**: El Hub entregó debidamente el sobre y después de la entrega se eliminó
  - **Entrega fallida** (**FailedDelivery**): El Hub no pudo entregar el sobre y se alcanzó el período de expiración de la fila establecido por la ONPF exportadora. Entonces, el sobre se eliminó de la fila del Hub.
  - **No existe el sobre** (**EnvelopeNotExists**): El Hub no tiene información sobre un determinado número de seguimiento.
  - **Entregado con advertencias** (**DeliveredWithWarnings**): introducido con la versión de marzo de 2018, se utilizará para marcar los sobres que reconoce el país importador con algunas advertencias de incumplimiento del esquema que el país exportador puede leer y utilizar para afinar la producción de XML mundialmente normalizados

- **Mensaje de error del Hub (HUBErrorMessage):** Este elemento tendrá mensajes para los diferentes errores que pueden ocurrir durante la interacción con el Hub. La mayoría de los mensajes de error se relacionan con la expiración del tiempo de retención en la fila. A partir de la versión de marzo de 2018 el país importador puede configurar los mensajes de advertencia relacionados con las operaciones de AdvancedAcknowledge (ver las operaciones más adelante) indicando los elementos que deben mejorarse en los certificados ePhyto en XML que hayan recibido.

5.1.2 Contenido de los sobres

El tipo de sobre *hereda* el encabezado del sobre y lo amplía con el elemento "Contenido", que puede ser cualquier tipo de cadena/xml.

La aplicación cliente de la ONPF exportadora crea el certificado fitosanitario electrónico, se serializa en XML y se envía al Hub utilizando el atributo de contenido del sobre.

El Hub no valida el contenido del certificado y su cumplimiento del esquema de la NIMF 12. La aplicación cliente de la ONPF importadora será la encargada de abrir el contenido del certificado y asegurar que cumpla con la norma correspondiente. En el momento de la recepción del sobre, la aplicación cliente de la ONPF importadora tiene que reconocer la recepción correcta del mensaje, independientemente de la validación del certificado que se realizará mediante un proceso de trabajo independiente.

El siguiente documento de referencia contiene todos los detalles acerca de los requisitos y recursos de asignación para XML.

https://www.ephytoexchange.org/doc/mapping/Mapping_ISPM_12_to_ePhyto_standard_Export_certificate_V.2.pdf

5.1.3 Conjunto de encabezados para los sobres (Array of EnvelopeHeader)

Este elemento se utiliza para intercambiar una serie de encabezados para los sobres, agrupados. Las operaciones "GetUnderDeliveryEnvelope' y 'GetImportEnvelopeHeaders' se describen a continuación.

5.1.4 Conjunto de sobres (Array of Envelopes)

Este elemento contiene una lista de sobres. Cada sobre contiene un encabezado y un certificado ePhyto. Esta entidad se utiliza en la operación "PULLImportEnvelope", descrita en detalle a continuación.

6. Operaciones

6.1 Acceso al Hub

Conectarse con el Hub no es una operación expuesta por el servicio web, sino la llamada interna necesaria para invocar alguna de las operaciones del servicio web remoto.

En esta sección mostraremos el código genérico necesario para abrir una conexión de cliente con el Hub utilizando C# y .Net Framework 4.6.1, Java 1.8 y Apache Axis 1 para generar el código de cliente de WSDL.

El código va a crear el nuevo cliente, agregar el certificado y la URL (de acuerdo al entorno) para utilizarlo en todas las posteriores llamadas al servicio web.

C#

```
    private static DeliveryService getClientConnection()
    {
        // the following code is use to prevent security protocol
exceptions
        // raised by using self-signed certificates (test environment)
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls11;
        System.Net.ServicePointManager.ServerCertificateValidationCallback
= delegate (
        Object obj, X509Certificate certificate, X509Chain chain,
        SslPolicyErrors errors)
        {
            return (true);
        };

        //This is the actual implementation of the generated proxy
        //from the given or downloaded WSDL
        DeliveryService client = new DeliveryService();

        //setting the test environment URL
        client.Url = "https://uat.ippchub.unicc.org/hub/DeliveryService";

        //adding the certificate
        X509Certificate2 cert = new
X509Certificate2("/Users/luca/repos/IPPCHubDev/certificates/nppo-it.p12",
"nppoITp12");
        client.ClientCertificates.Add(cert);

        //returning the client object
        return client;
    }
```

Java

```
  private static final String KEYSTORE_TRUSTED =
"G:\\certificates\\trustedStore";
  private static final String KEYSTORE_TRUSTED_PASSWORD = "changeit";

  private static final String KEYSTORE_SERVER =
"G:\\certificates\\privateStore";
  private static final String KEYSTORE_SERVER_PASSWORD = "changeit";

  private static IDeliveryServiceProxy getClientConnection() {
    // Configure the stores with certificates
    System.setProperty("sun.security.ssl.allowUnsafeRenegotiation",
"true");  // true for self-signed certificates, false in production

    // Trusted certificates, IPPC HUB certificate should be here
    System.setProperty("javax.net.ssl.trustStore", KEYSTORE_TRUSTED);
    System.setProperty("javax.net.ssl.trustStorePassword",
KEYSTORE_TRUSTED_PASSWORD);

    // Private Key store, with NPPO certificate
    System.setProperty("javax.net.ssl.keyStore", KEYSTORE_SERVER);
    System.setProperty("javax.net.ssl.keyStorePassword",
KEYSTORE_SERVER_PASSWORD);

    // Uncomment next line to have handshake debug information
```

```
    // System.setProperty("javax.net.debug", "ssl");

    // Getting the proxy to the appropriate URL
    IDeliveryServiceProxy proxy = new IDeliveryServiceProxy("https://uat-
hub.ephytoexchange.org/hub/DeliveryService");

    return proxy;
  }
```

6.2 Entrega de sobre (DeliverEnvelope)

La ONPF exportadora utilizará esta operación para enviar los sobres al Hub. El encabezado deberá llenarse con los siguientes atributos mínimos:

- De
- Para
- Tipo de certificado
- Estado del certificado
- Número de certificado de la ONPF (no es obligatorio, pero recomendamos utilizar este campo para poder referenciar fácilmente cada transmisión con el certificado original en el sistema de exportación)
- y el atributo del "Contenido" se llena con el certificado; para completar el sobre con la versión serializada denXML del certificado ePhyto generado.

El Hub responde con el encabezado del sobre (EnvelopeHeader), que contiene todos los atributos asignados por la aplicación cliente de la ONPF exportadora, así como los atributos de número e información de seguimiento (HUBTrackingNumber y HUBTrackingInfo) del Hub que este asigna.

Respecto al "tránsito", cuando el certificado se tiene que distribuir a los países de tránsito también, la aplicación cliente deberá enviar los sobres a todos los países en mensaje separados y se dará seguimiento independiente a cada transmisión.

Ejemplo de aplicación cliente en C# generado como cliente estándar del servicio web .Net 2.0:

https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/transport-security-with-certificate-authentication

C#

```
            // initialize the client
             DeliveryService client = getClientConnection();

             // simulating an Issue certificate from Italy to United States
             Envelope env = new Envelope()
             {
                 From = "IT",
                 To = "US",
                 CertificateType = 851,
                 CertificateStatus = 70,
                 NPPOCertificateNumber = "Internal NPPO Certificate Number"

             };

             //load the actual electronic certificate XML
             var ePhyto = new System.Xml.XmlDocument();
```

```
            ePhyto.LoadXml("<?xml version=\"1.0\" encoding=\"UTF-
8\"?><ephyto><contents/></ephyto>");

            //set the XML to the content element of the message
            env.Content = ePhyto.InnerXml;

            try
            {
                // send the message to the hub and get back the header
                EnvelopeHeader header = client.DeliverEnvelope(env);

                //handle internal issues
                if (header.HUBTrackingInfo == "FailedDelivery")
                {
                    //manage the exception and provide errors to the client
                    //in this case the error is due to one of the following
                    //Header Validation error (certificate, destination
country not boarded...)
                    //Internal error of the system

                    //get the error message
                    string error = header.hubDeliveryErrorMessage;
                    System.Console.WriteLine("Message failed delivery,
"+error);
                }
                else
                {
                    //get the hub tracking number...
                    string hubTrackingNumber = header.hubDeliveryNumber;
                    System.Console.Write("header delivered with tracking
number : " + hubTrackingNumber);

                    //persist the header details to record that the message
is under delivery
                }

            }catch(Exception ex){
                //manage the exception and provide errors to the client
                //in this case the error is due to one of the following
                //Header Validation error (certificate, destination country
not boarded...)
                //network
                //unavailability of the remote system
                Console.WriteLine("Failed to deiver the message to the HUB"
+ ex.Message);
            }
```

Java

```
  private static EnvelopeHeader DeliverEnvelope() throws HubClientException
{
    IDeliveryServiceProxy proxy = getClientConnection();
```

```
    DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();

    // Envelope creation, from Italy to United States
    Envelope envelope = new Envelope();
    envelope.setFrom("IT");
    envelope.setTo("US");
    envelope.setCertificateType(851);
    envelope.setCertificateStatus(70);
    envelope.setNPPOCertificateNumber("EPHYTO-IT-2017-0010277");

    try {
      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
      Document doc = dBuilder.parse("<?xml version=\"1.0\" encoding=\"UTF-
8\"?><ephyto><contents/></ephyto>");
      DOMSource domSource = new DOMSource(doc);
      StringWriter writer = new StringWriter();
      StreamResult result = new StreamResult(writer);
      TransformerFactory tf = TransformerFactory.newInstance();
      Transformer transformer = tf.newTransformer();
      transformer.transform(domSource, result);
      envelope.setContent(writer.toString());
    } catch (SAXException | IOException | ParserConfigurationException |
TransformerException e1) {
      //manage the exception and provide errors to the client
      //in this case the error is due to one of the following
      //The XML string could not be parsed
      System.out.println("Failed to load certificateinto XML document.");
      throw new HubClientException(e1); // Without certificate we cannot
continue
    }

    try {
      // send the message to the hub and get back the header
      EnvelopeHeader header = proxy.deliverEnvelope(envelope);

      // Handle internal issues
      if (header.getHUBTrackingInfo().equals("FailedDelivery")) {
        //manage the exception and provide errors to the client
        //in this case the error is due to one of the following
        //Header validation error
        String error = header.getHubDeliveryErrorMessage();
        System.out.println(String.format("Message failed delivery. %s",
error));
      } else {
        //get the hub tracking number...
        String hubTrackingNumber = header.getHubDeliveryNumber();
        System.out.println(String.format("Header delivered with tracking
number: %s", hubTrackingNumber));
      }
      return header;
    } catch (RemoteException e) {
      //manage the exception and provide errors to the client
      //in this case the error is due to one of the following
      // network
      // unavailability of the remote system
      System.out.println(String.format("Failed to deliver the message to
the HUB. ", e.getMessage()));
      throw new HubClientException(e);
    }
  }
```

Si se produce algún error, el HUBTrackingInfo está configurado para hacer un "FailedDelivery" y los detalles se encuentran en el elemento hubDeliveryErrorMessage del encabezado del sobre devuelto. Los posibles errores que se detecten son:

- La ONPF que envía el sobre no es del país indicado en el campo "De"
- El sistema no tiene una ONPF para el país indicado en "Para"
- Tipo de certificado no válido
- Estado de certificado no válido

Los problemas de conectividad, tales como interrupciones de red o falta de disponibilidad del sistema se señalarán como errores estándar del protocolo HTTP, ya que no los genera la aplicación remota.

6.3 PULLImportEnvelope, AcknowledgeEnvelopeReceipt, AdvancedAcknowledgeEnvelopeReceipt

La ONPF importadora configurada para la operación PULL utilizará esta operación para recuperar todos los sobres que estén dirigidos a ella. El cliente autentificado representa el país importador y recibirá todos los sobres (*conjunto de sobres*) que están en la cola del Hub con el país importador en el campo Para. Para cada uno de estos sobres, el país importador deberá responder con un mensaje sobre el funcionamiento de AckknowledgeEnvelopeReceipt, la correcta recepción de cada uno de los sobres; con HUBTrackingNumber. Los avisos confirmados se eliminarán de la cola y la próxima operación de extracción recuperará los mensajes restantes hasta que el resultado quede vacío.

La configuración de la ONPF permitirá para reducir el conjunto de mensajes de cada obtención, para afinar la comunicación con una oficina que utilice una conexión insuficiente.

A partir de la versión de marzo de 2018 el sistema de apoyo comunicará un mensaje de texto relacionado con la operación de reconocimiento que establecerá la información de rastreo para DeliveredWithWarnings y proporcionará en el mensaje de error los detalles de los problemas encontrados durante la recepción y apertura del archivo XML.

Téngase en cuenta que el mensaje de advertencia se limita a 200 caracteres, el mensaje de respuesta puede indicar esas advertencias, los datos se truncarán en caso de que superen dicho límite.

En el siguiente ejemplo esos mensajes se pueden extraer de una acción de validación del esquema y notificarse al exportado para aprovechar la compatibilización del XML.

Ejemplo de aplicación cliente:

C#

```
// initialize the client
DeliveryService client = getClientConnection();

//get all the envelopes pending delivery
Envelope[] envelopesToImport = client.PULLImportEnvelope();

foreach(Envelope env in envelopesToImport)
```

```
    {
        System.Console.WriteLine("Processing hub delivery number : "
+env.hubDeliveryNumber);

        try
        {
            //get the content containing the certificate XML
            String xmlContent = env.Content;

            //verifications in xml
            var ePhyto = new System.Xml.XmlDocument();
            ePhyto.LoadXml(xmlContent);


            //save the ePhyto to the client application
            //acknowledge the receipt back to the server (this could be
done as separate action based on user validation ??)
            client.AcknowledgeEnvelopeReceipt(env.hubDeliveryNumber);


            //perform schema/xml checks
           client.AdvancedAcknowledgeEnvelopeReceipt(env.hubDeliveryNumber,
"please indicate the date elements without milliseconds");
        }
        catch(Exception ex)
        {
            //handle the content parsing error
            System.Console.WriteLine(String.Format("error when parsing
content of {0} {1}", env.hubDeliveryNumber,ex.Message));
        }
    }
```

Java

```
  private static void pullAcknowledge() throws HubClientException {
    IDeliveryServiceProxy proxy = getClientConnection();

    try {
      // get all the envelopes pending delivery
      Envelope[] envelopesToImport = proxy.PULLImportEnvelope();

      for (Envelope envelope : envelopesToImport) {
        System.out.println(String.format("Processing hub delivery number:
%s", envelope.getHubDeliveryNumber()));

        // get the content containing the certificate XML
        String xmlContent = envelope.getContent();

        // verifications in XML
        DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder;
        try {
          dBuilder = dbFactory.newDocumentBuilder();
```

```
            InputStream content = new
ByteArrayInputStream(envelope.getContent().getBytes(StandardCharsets.UTF_8.
name()));
            Document doc = dBuilder.parse(content);
          } catch (ParserConfigurationException | SAXException | IOException
e) {
            // The content of the envelope is not a proper XML file
            System.out.println(String.format("Error parsing content of %1$s
%2$s", envelope.getHubDeliveryNumber(), e.getMessage()));

            // This envelope won't be acknowledged

proxy.advancedAcknowledgeEnvelopeReceipt(envelope.getHubDeliveryNumber(),
"error while parsing the XML");
            continue;
          }

          //acknowledge the receipt back to the server (this could be done as
a separate action based on user validation)
          proxy.acknowledgeEnvelopeReceipt(envelope.getHubDeliveryNumber());
        }
      } catch (RemoteException e) {
        //manage the exception and provide errors to the client
        //in this case the error is due to one of the following
        // network
        // unavailability of the remote system
        System.out.println(String.format("Failed to deliver the message to
the HUB. ", e.getMessage()));
        throw new HubClientException(e);
      }
    }
```

Si se produce un error en el procesamiento del PullImportEnvelope, AckowledgeEnvelopeReceipt o AdvancedAckowledgeEnvelopeReceipt, se enviará un elemento estándar de fallo SOAP con la descripción del error. Los errores detectados en estos servicios incluyen:

- La ONPF que hace la solicitud no está en el sistema
- La ONPF que hace el acuse de recibo no es del país que figura en el campo "Para" del encabezado reconocido
- No se ha encontrado la sobre, como anteriormente relacionado con la petición de reconocimiento. Cuando el número enviado no se encuentra en el Hub.

6.4 Obtener sobre en proceso de entrega (GetUnderDeliveryEnvelope)

Esta operación permite a la ONPF exportadora obtener una lista de todos los encabezados de sobres que están en el proceso de entrega (es decir, con HUBDeliveryStatus como PendingDelivery). El cliente autentificado representa al país exportador. El Hub devolverá la lista de todos los sobres pendientes de entrega (conjunto de EnvelopeHeader).

La aplicación cliente puede utilizar HUBTrackingNumber de los encabezados de los sobres devueltos y actualizaciones del sistema.

Ejemplo de aplicación cliente:

C#

```
      DeliveryService client = getClientConnection();
      try
      {
            //get the envelopes under delivery (received by the HUB and
queued to be delivered to the destination)
            EnvelopeHeader[] headers = client.GetUnderDeliveryEnvelope();

            //cicles the records to update the client system
            foreach (var head in headers)
            {
                //updates the client records
System.Console.WriteLine("Env:"+head.hubDeliveryNumber+",Tracking
Info:"+head.HUBTrackingInfo);
            }
      }
      catch (Exception ex)
      {
            System.Console.WriteLine(ex.Message);
      }
```

Java

```
  private static void getUnderDeliveryEnvelope() throws HubClientException
{
    IDeliveryServiceProxy proxy = getClientConnection();

    try {

      // get the envelopes under delivery
      EnvelopeHeader[] headers = proxy.getUnderDeliveryEnvelope();

      //clicles the records to update the client system
      for(EnvelopeHeader header : headers) {
        // updates client records
        System.out.println(String.format("Envelope: %1$s - Tracking info:
%2$s", header.getHubDeliveryNumber(), header.getHUBTrackingInfo()));
      }
    } catch (RemoteException e) {
      //manage the exception and provide errors to the client
      //in this case the error is due to one of the following
      // network
      // unavailability of the remote system
      System.out.println(String.format("Failed to deliver the message to
the HUB. ", e.getMessage()));
      throw new HubClientException(e);
    }
  }
```

Si se produce un error en el procesamiento de GetUnderDeliveryEnvelope se enviará un elemento estándar de fallo SOAP con la descripción del error. Los errores detectados en estos servicios incluyen:

- La ONPF que hace la solicitud no está en el sistema

6.5 Obtener los encabezados de los sobres de importación (GetImportEnvelopeHeaders) y PULL de un sobre único de importación (PULLSingleImportEnvelope)

Esta operación permite a la ONPF exportadora obtener una lista de todos los encabezados de sobres que están en el proceso de entrega (es decir, con HUBDeliveryStatus como PendingDelivery). El cliente autentificado representa al país importador. El Hub devolverá la lista de todos los sobres pendientes de entrega (conjunto de EnvelopeHeader).

La aplicación cliente puede utilizar HUBTrackingNumber de los encabezados de los sobres devueltos y recuperarlos uno por uno. Esto permitirá que el país importador trabajar sobre con todo el subconjunto de mensajes pendientes de entrega, en lugar de tener que sacarlos por grupos

Ejemplo de aplicación cliente:

C#

```
      DeliveryService client = getClientConnection();
      try
      {
            //get the envelopes under delivery (received by the HUB and
queued to be delivered to the destination)
            EnvelopeHeader[] headers = client.GetImportEnvelopeHeaders();

            //cicles the records to update the client system
            foreach (var head in headers)
            {
                Envelope env =
client.PULLSingleImportEnvelope(head.hubDeliveryNumber);
                //get the content containing the certificate XML
                String xmlContent = env.Content;

                //verifications in xml
                var ePhyto = new System.Xml.XmlDocument();
                ePhyto.LoadXml(xmlContent);


                //save the ePhyto to the client application
                //acknowledge the receipt back to the server (this could be
done as separate action based on user validation ??)
                client.AcknowledgeEnvelopeReceipt(env.hubDeliveryNumber);


               //perform schema/xml checks

client.AdvancedAcknowledgeEnvelopeReceipt(env.hubDeliveryNumber, "please
indicate the date elements without milliseconds");



            }
      }
      catch (Exception ex)
      {
```

```
            System.Console.WriteLine(ex.Message);
        }
```

Java

```
  private static void getImportEnvelopeHeaders() throws HubClientException
{
    IDeliveryServiceProxy proxy = getClientConnection();

    try {

      // get the envelopes under delivery
      EnvelopeHeader[] headers = proxy.getUnderDeliveryEnvelope();

      //clicles the records to update the client system
      for(EnvelopeHeader header : headers) {

         // get the envelope
         Envelope env = proxy.PULLSingleImportEnvelope(header.
getHubDeliveryNumber());

        // get the content containing the certificate XML
        String xmlContent = env.getContent();

        // verifications in XML
        DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder;
        try {
          dBuilder = dbFactory.newDocumentBuilder();
          InputStream content = new
ByteArrayInputStream(env.getContent().getBytes(StandardCharsets.UTF_8.name(
)));
          Document doc = dBuilder.parse(content);
        } catch (ParserConfigurationException | SAXException | IOException
e) {
          // The content of the envelope is not a proper XML file
          System.out.println(String.format("Error parsing content of %1$s
%2$s", env.getHubDeliveryNumber(), e.getMessage()));

          // This envelope won't be acknowledged

proxy.advancedAcknowledgeEnvelopeReceipt(env.getHubDeliveryNumber(), "error
while parsing the XML");
          continue;
        }

        //acknowledge the receipt back to the server (this could be done as
a separate action based on user validation)
        proxy.acknowledgeEnvelopeReceipt(env.getHubDeliveryNumber());
      }
    } catch (RemoteException e) {
      //manage the exception and provide errors to the client
      //in this case the error is due to one of the following
      // network
      // unavailability of the remote system
      System.out.println(String.format("Failed to pull envelopes from the
HUB. ", e.getMessage()));
      throw new HubClientException(e);
    }
```

```
}
```

Si se produce un error en el procesamiento de GetImportEnvelopeHeader, AcknowledgeEnvelopeReceipt y AdvanvcedAcknowledgeEnvelopeReceipt, se enviará un elemento estándar de fallo SOAP con la descripción del error. Los errores detectados en estos servicios incluyen:

- La ONPF que hace la solicitud no está en el sistema

6.6 Información de seguimiento del sobre (GetEnvelopeTrackingInfo)

Esta operación proporciona el HUBTrackingInfo de un determinado HUBTrackingNumber;un sobre cada vez. La idea es que si la aplicación cliente ha enviado el sobre y el encabezado del sobre no está en la lista de pendientes de entrega, entonces el sistema debería consultar al Hub para saber si se ha entregado correctamente y en qué etapa está. Arriba se hace referencia ala información de rastreo y se comenta en el ejemplo de código siguiente.

C#

```
      DeliveryService client = getClientConnection();
            try
            {
                  EnvelopeHeader head= client.GetEnvelopeTrackingInfo(num);

                System.Console.WriteLine(string.Format("The envelope {0}
tracking info is {1}",head.hubDeliveryNumber,head.HUBTrackingInfo ));

                switch(head.HUBTrackingInfo){
                    case "Delivered":
                        //perform client updates to mark the envelope
delivered
                        break;
                    case "DeliveredWithWarnings":
                        //perform client updates to mark the envelope
delivered, capture the text and send the information to technical people
                        break;
                    case "FailedDelivery":
                        string error = head.hubDeliveryErrorMessage;
                        //update the client state with the informational
error message
                        break;
                    case "EnvelopeNotExists":
                        //the message was received by the hub but not yet
added to the queue or the number is not correct
                        //resending of the original can be applied
                        break;
                    case "PendingDelivery":
                        //still in the queue on the hub, waiting to be
pulled or pushed
                        break;
```

```
            }
        }
        catch (Exception ex)
        {
            System.Console.WriteLine(ex.Message);
        }
```

Java

```
  private static void getEnvelopeTrackingInfo(String hubTrackingNumber)
throws HubClientException {
    IDeliveryServiceProxy proxy = getClientConnection();

    try {
      EnvelopeHeader header =
proxy.getEnvelopeTrackingInfo(hubTrackingNumber);
      System.out.println(String.format("The envelope %1$s tracking info is
%2$s", header.getHubDeliveryNumber(), header.getHUBTrackingInfo()));

      switch (header.getHUBTrackingInfo()) {
        case "Delivered":
          // perform client updates to mark the envelope as delivered
          break;
        case "DeliveredWithWarnings":
          // perform client updates to mark the envelope as delivered,
capture the error message and send it to the technical people
          break;
        case "FailedDelivery":
          String errorMessage = header.getHubDeliveryErrorMessage();
          // update the client state with the informational error message
          break;
        case "EnvelopeNotExists":
          //the message was received by the hub but not yet added to the
queue or the number is not correct
          //resending of the original can be applied
          break;
        case "PendingDelivery":
          //still in the queue on the hub, waiting to be pulled or pushed
          break;
      }
    } catch (RemoteException e) {
      //manage the exception and provide errors to the client
      //in this case the error is due to one of the following
      // network
      // unavailability of the remote system
      System.out.println(String.format("Failed to deliver the message to
the HUB. ", e.getMessage()));
      throw new HubClientException(e);
    }
  }
```

Si se produce un error, éste será devuelto como excepción de SOAP. Los posibles errores que se detecten son:

- La ONPF que hace la solicitud no figura en el sistema
- La ONPF solicitante no es del país que aparece en el campo "De"

6.7 ONPF activas (GetActiveNppos)

Esta operación es una simple acción de consulta que proporciona todas las ONPF activas del Hub, sólo con el código de país, y los indicadores de envío y recepción. La aplicación cliente puede utilizar esos indicadores para automatizar el envío o la recepción del país pertinente, de acuerdo con su estado en el Hub.

6.8 Validación del XML (ValidatePhytoXML)

Esta operación expone la funcionalidad de AdminConsole para validar el archivo XML respecto al último esquema de ePhyto (basado en la versión UNCEFACT 17A).

Esto puede ser utilizado para recoger y advertir sobre problemas en mensajes entrantes y salientes.

Cuando se usa desde SoapUI, se recomienda para envolver el XML para validar en un elemento <![CDATA[….]]> a fin de poder copiar y pegar el texto de origen tal como es.

No se proporcionan aquí ejemplos de código, ya que son sólo una operación de petición diferente de las descritas anteriormente. El resultado de la operación devolverá un conjunto de resultados de validación como el siguiente

```
<ns3:ValidatePhytoXMLResult>
        <area>MandatoryElements</area>
        <field>SPSExchangedDocument.IssueDateTime.DateTimeString</field>
        <level>SEVERE</level>
        <msg>Issue date is mandatory field</msg>
</ns3:ValidatePhytoXMLResult>
```

El campo indica el elemento origen de la cuestión, el msg indica un mensaje descriptivo de la cuestión.

Las áreas posibles son las siguientes:

- Elementos obligatorios (elementos que deben estar presentes como parte de la estructura del documento)
- Mapeo (asuntos relacionados con el mapeodel ePhyto al esquema)
- Esquema (cuestiones relativas a incumplimientos del esquema XML)

Los niveles posibles son los siguientes:

- GRAVES : llevar a problemas en la lectura y visualización del certificado
- ADVERTENCIA: no conduce a problemas, puede necesitar alguna revisión sobre cómo se produce el XML
- INFORMACIÓN: pueden aplicarse cambios en el nivel de optimización

6.9 Entrega del sobre del ePhyto (DeliverPhytoEnvelope)

Esta operación expone una nueva entidad en el Hub, el ePhytoEnvelope que es heredada del EnvelopeHeader y en lugar de tener un elemento de contenido de cadena de texto, como se define en el sobre, tiene el SPSCertificate definido con el esquema utilizado por la herramienta de validación (basado en el UNCEFACT).

Esta operación, de forma similar a DeliverEnvelope, requerirá toda la información definida anteriormente y una entidad válida de SPSCertificate definida. La operación antes de poner el contenido en la cola para la entrega realizará una validación y si se encuentra cualquier nivel GRAVE de problemas, detendrá la entrega.

El uso de esta operación sigue exactamente el mismo flujo de trabajo definido para DeliverEnvelope y puede aprovechar la aplicación cliente de compilar el código XML, al proporcinar la información necesaria de las entidades definidas en la solicitud del exportador.

Código de muestra ya que es una especialización de DeliveryEnvelope definida anteriormente, en lugar de establecer un XML todo el objeto SPSCertificate deberá llenarse, dependiendo de la información disponible (esta situación puede variar de acuerdo a cómo se implemente el sistema de llenado para completar dinámicamente los diferentes casos)

6.10 Recibir una entrega PUSH

A fin de recibir una entrega PUSH, la ONPF importadora deberá tener un punto final listo para que se conecte el Hub.

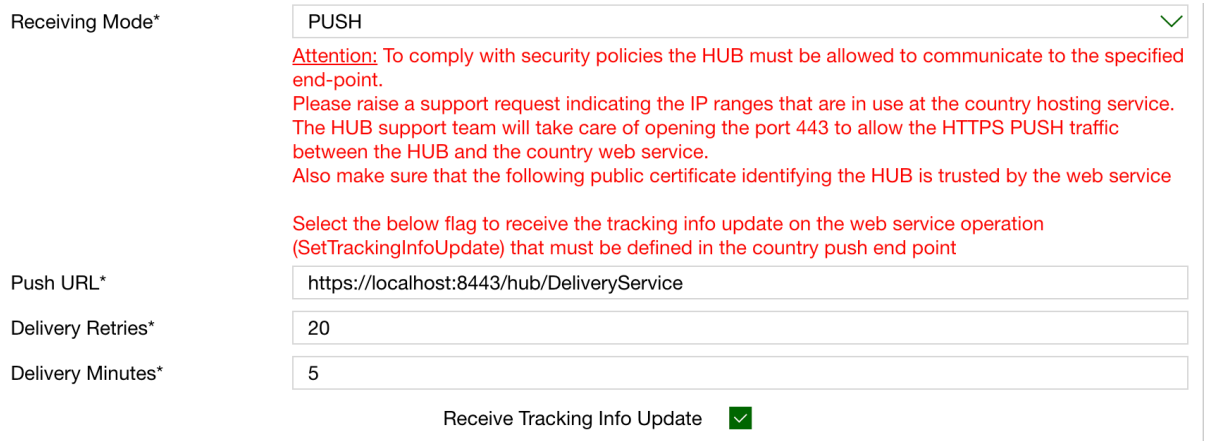
La ONPF deberá utilizar el archivo WSDL del Hub para generar las fuentes necesarias, y desarrollar las funcionalidades necesarias para recibir en la operación DeliveryEnvelope los sobres y confirmar la recepción utilizando las operaciones de reconocimiento del Hub descritas anteriormente con el PULL.

Si hay un error y el punto final de la aplicación del PUSH quiere recibir de nuevo el sobre, el resultado de punto final debería tener un encabezado de los sobres con la información de rastreo como FailedDelivery (en este caso el Hub seguirá tratando de mandar el sobre hasta que se reconozca o la respuesta desde el punto final sea correcta). Si no se responde con FailedDelivery al Hub, el sistema cerará el sobre esperando el reconocimiento.

El administrador de la ONPF para los asuntos del Hub tendrá que comunicar (mediante la solicitud de ayuda descrita anteriormente) los rangos de IP que están acogiendo el servicio web del PUSH. Después de la apertura de los puertos en el Hub, el administrador de la ONPF para los asuntos del Hub recibirá un mensaje de correo electrónico de confirmación con el certificado de cliente SSL utilizado por el Hub para autentificar.

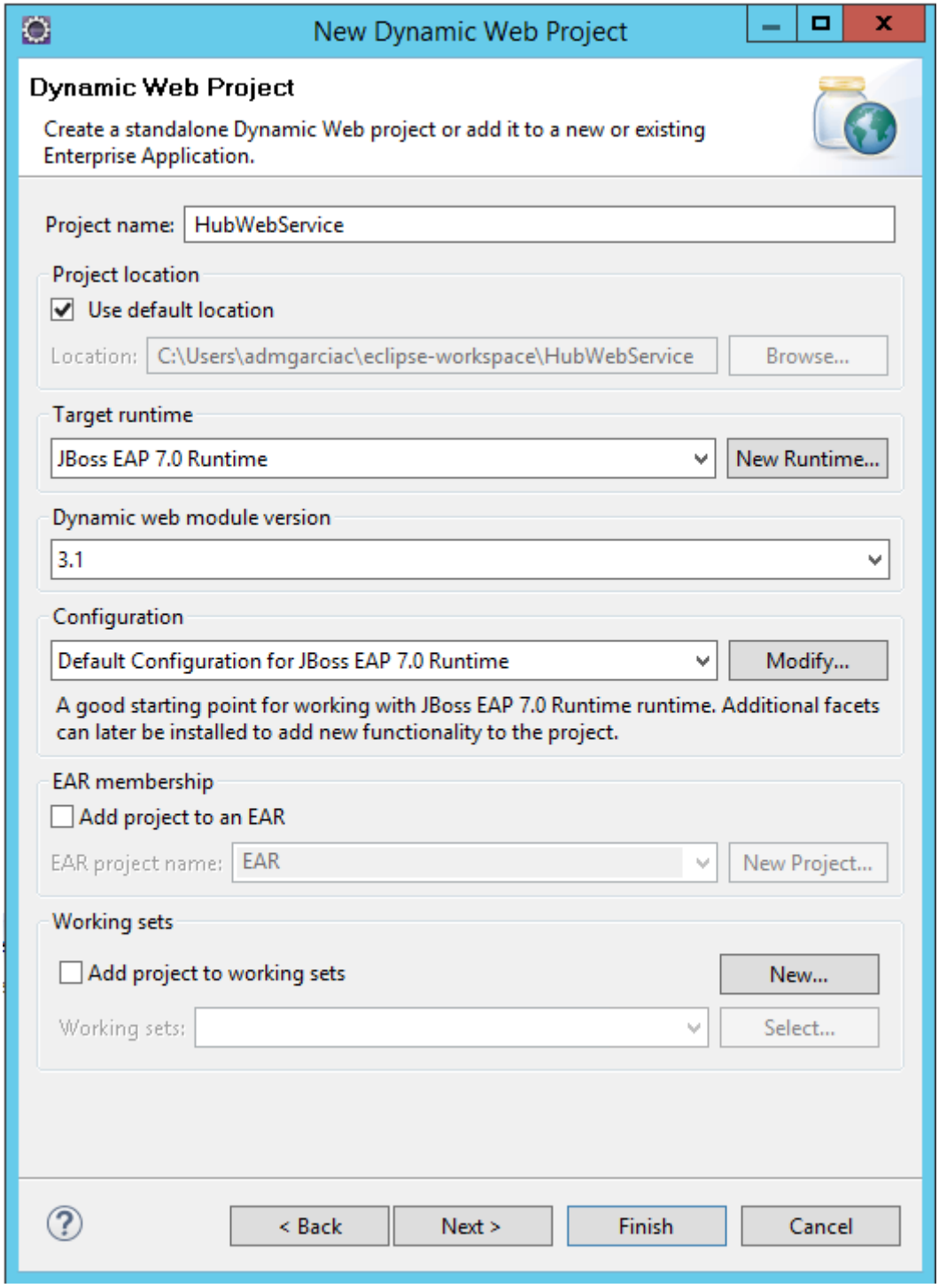
Además, el perfil de la ONPF se puede configurar para Receive Tracking Info Update a través de PUSH. Esto significa que el HUB enviará el encabezado del sobre a la operación **SetTrackingInfoUpdate** sobre el cambio de la información de seguimiento, de pendiente de entrega a entrega final o error.

Abajo se muestra la configuración de la ONPF respecto al modo de recepción de PUSH.



A continuación un ejemplo de cómo crear un endpoint básico con Eclipse y Apache Axis.

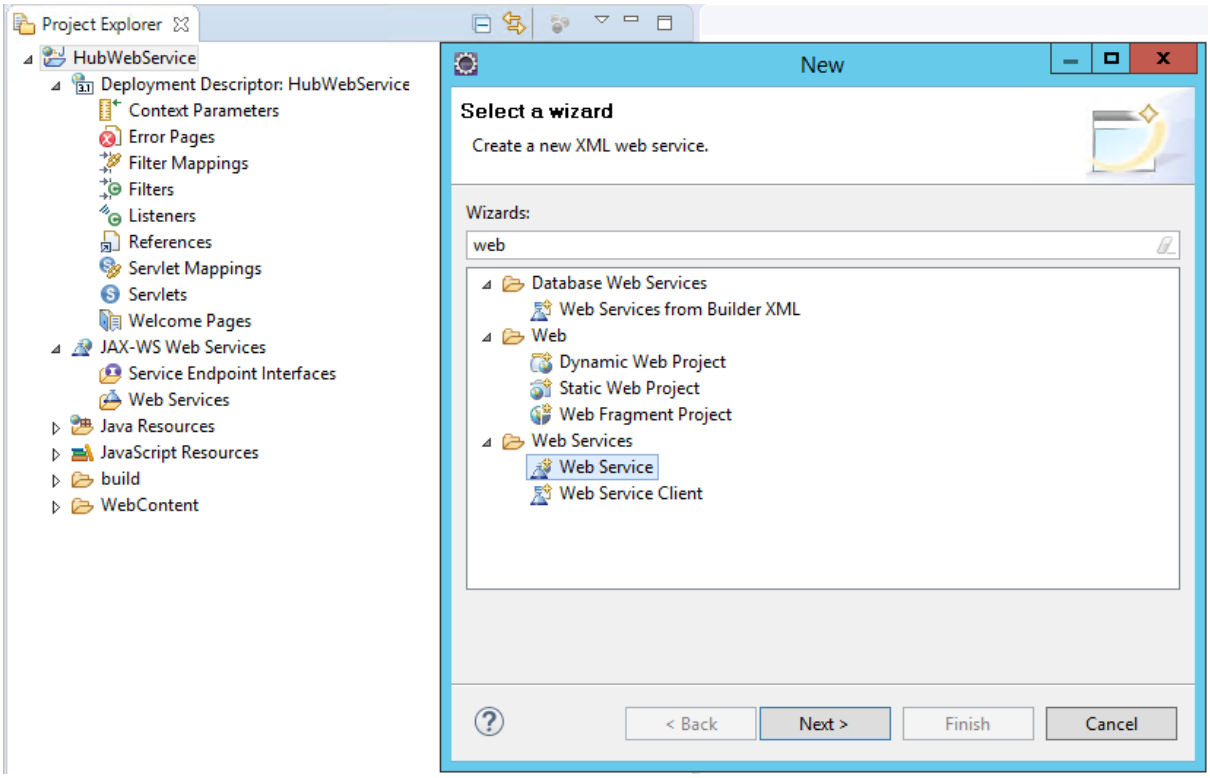
En primer lugar, crear un nuevo proyecto Web dinámico en Eclipse (aquí utilizamos JBoss como destino del tiempo de ejecución, utilice el motor de tiempo de ejecución que mejor se adapte a sus necesidades)



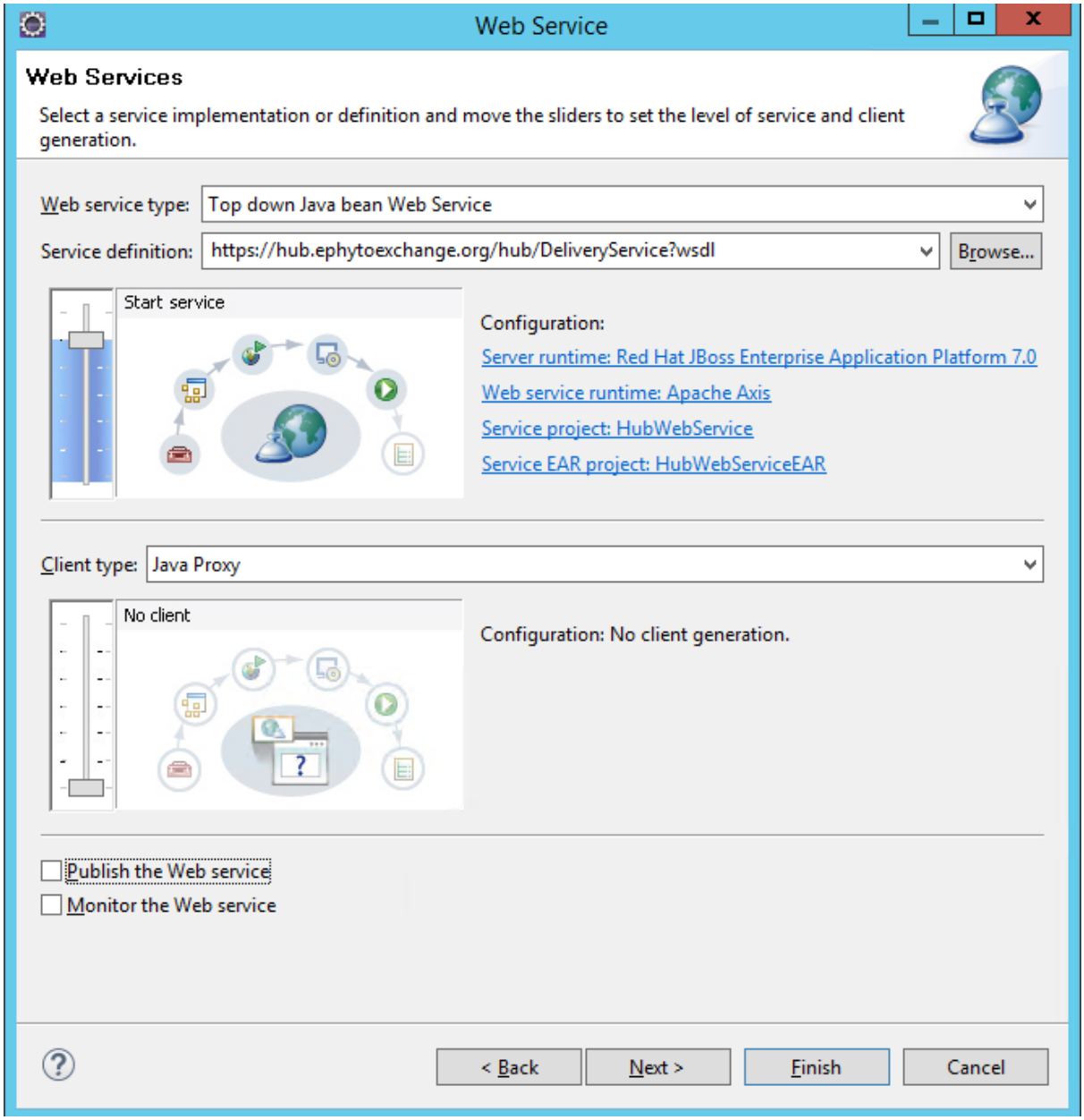
Puede hacer clic en "Finalizar" en este diálogo.

Una vez que el proyecto esté allí (omitir la creación del proyecto si desea implementar el servicio en un proyecto existente), será necesario crear las clases que implementan la interfaz del servicio web. Para

ello, vamos a agregar un nuevo servicio Web al proyecto haciendo clic con el botón derecho en el nombre del proyecto y luego en "Nuevo" y "Otros…".



Seleccionar "Web Service" y hacer clic en "Siguiente".



Como ya tenemos el archivo WSDL, seleccionar "top down de servicios Web Java Bean".

Después de eso introducir la dirección URL WSDL del Hub en la definición del servicio:

https://hub.ephytoexchange.org/hub/DeliveryService?wsdl

Utilizaremos "Apache Axis" y JBoss como nuestro servidor para el despliegue. Si tiene un servidor de aplicaciones diferente, simplemente selecciónelo haciendo clic en el enlace "Servidor" en tiempo de ejecución. Hay que asegurarse de que el servidor de aplicaciones se esté ejecutando y hacer clic en "Finalizar".

Cuando el proceso finalice, Eclipse abrirá el archivo: DeliveryServiceSoapBindingImpl.java es donde el código se tiene que completar. En este caso, sólo hay que aplicar el método "deliverEnvelope", que es el que será llamado por el servicio PUSH en el Hub.

Java

```
    public _int.ippc.ephyto.HUB_Entities.EnvelopeHeader
deliverEnvelope(_int.ippc.ephyto.HUB_Entities.Envelope env) throws
java.rmi.RemoteException, _int.ippc.ephyto.HubWebException {
        saveEnvelope(env);
        return env;
    }

    private void saveEnvelope(_int.ippc.ephyto.HUB_Entities.Envelope env) {
      // do checks and store the envelope in the suitable place

      // acknowledge the reception
      HubClient.acknowledge(env);
    }
```

Guarde el sobre y, a continuación, confirme la recepción del sobre al Hub de forma que se pueda marcar como entregado.

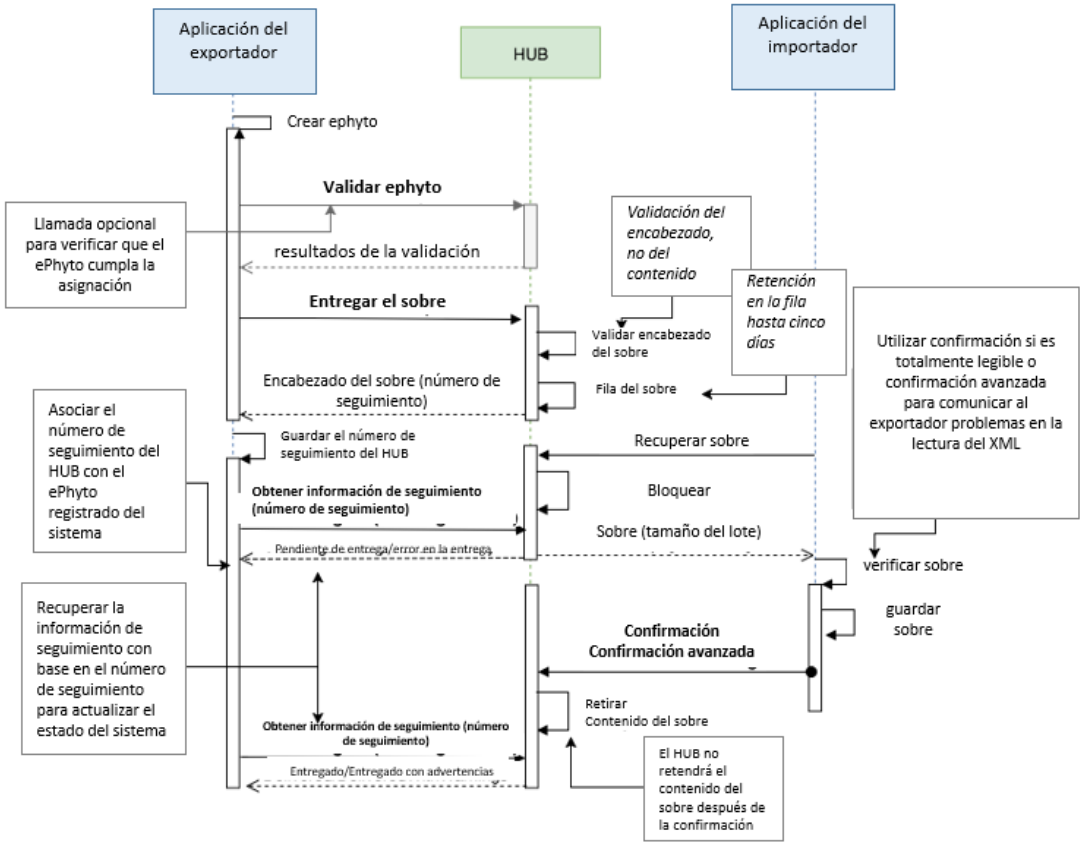
Nótese que HubClient se refiere al objeto que implementa la conexión al los servicios web del Hub.

En el ejemplo anterior no se proporcionan directrices sobre cómo configurar la autentificación de certificados de cliente, ya que puede variar considerablemente de acuerdo a la plataforma y la infraestructura de base. Para implementar el punto final del PUSH la aplicación de la ONPF deberá aceptar el certificado del Hub para la autentificación del cliente (el certificado del cliente se proporcionará con la respuesta de la apertura de los puertos necesarios al país huésped de los rangos de IP).

7. Diagrama de secuencia

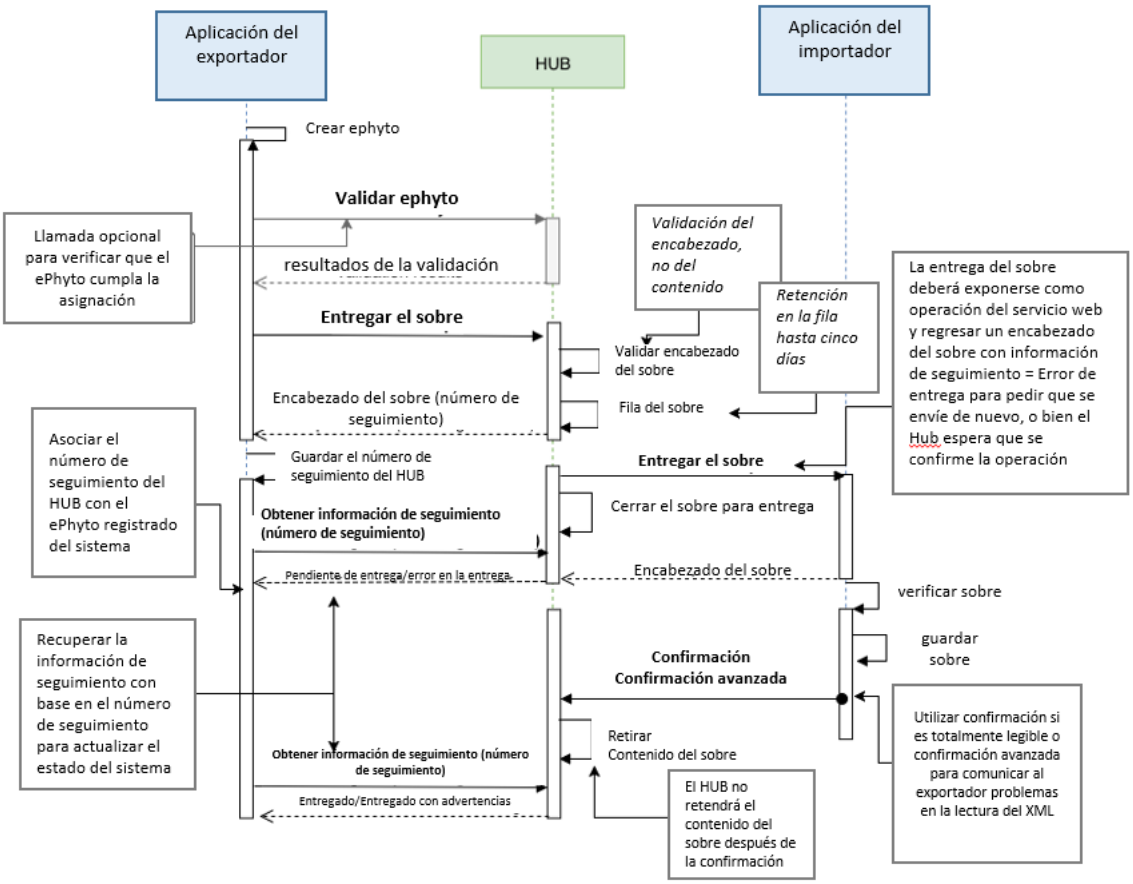
7.1 Entrega con PULL

Seguir un diagrama de secuencia que defina el la interacción óptima del proceso de entrega de sobres entre las aplicaciones cliente de la ONPF y el Hub utilizando el tipo de receptor PULL.



7.1 Entrega con PUSH

Seguir un diagrama de secuencia que defina el la interacción óptima del proceso de entrega de sobres entre las aplicaciones cliente de la ONPF y el Hub utilizando el tipo de receptor PUSH.



8. Pruebas con Soap UI

Nota: Cada vez que se inicia SOAP UI, es necesario seguir de nuevo los pasos 9 a 15.

Sírvase sguir los siguientes pasos para probar con SOAP UI:

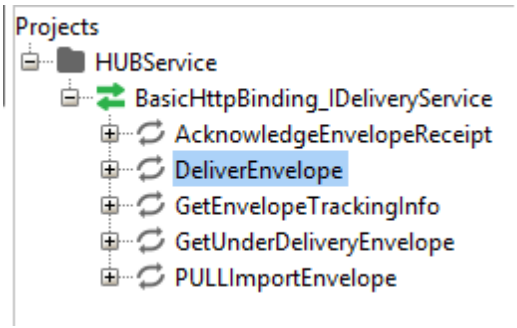
1. Descargar e instalar SOAP UI. Estamos usando la versión 5.3
2. Ir al directorio bin de la carpeta de instalación y abrir el archivo SOAPUI-5.3.0.vmoptions. En equipo con Windows, el archivo se encuentra en "C:\Program

Files\SmartBear\SoapUI-5.3.0\bin", en Mac está bajo el directorio /Applications/SoapUI-5.3.0.app/Contents/vmoptions.txt. Es necesario editar este archivo con derechos de administrador.

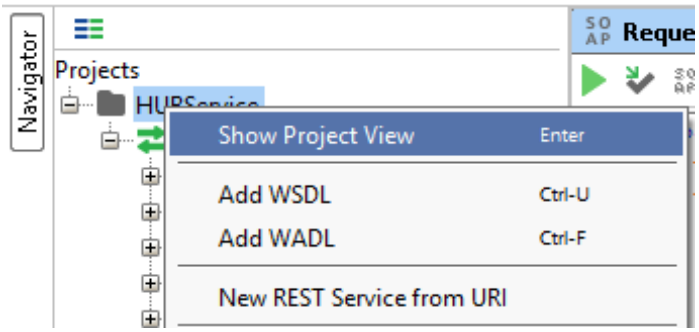
3. Incluir esta línea al final del archivo (el servicio web del Hub sólo acepta TLSv1.1 y TLSv1.2):

   -Dsoapui.https.protocols=TLSv1.1,TLSv1.2

4. Guardar el archivo y abrir o reabrir SOAP UI.
5. Ir a Archivo New SOAP Project.
6. En el campo "Nombre del proyecto", elija un nombre de proyecto descriptivo.
7. En "Initial WSDL", elija la URL proporcionada para el punto final (esta URL debería terminar en "?wsdl"), o seleccione el archivo wsdl, si recibió el archivo o si guardó el archivo wsdl en el ordenador.
8. Después de hacer clic en Aceptar, SOAP UI generará algunas plantillas con peticiones de la operación. Aquí puede ver un ejemplo de las solicitudes generadas de esta plantilla:

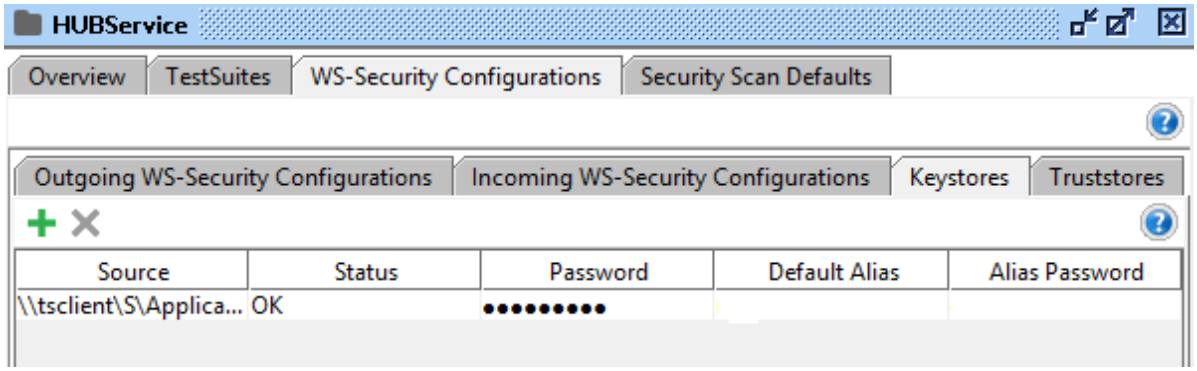


9. Haga clic con el botón derecho en el nombre del proyecto, en nuestro caso "HUBService". Y elija la opción "Mostrar vista del proyecto".

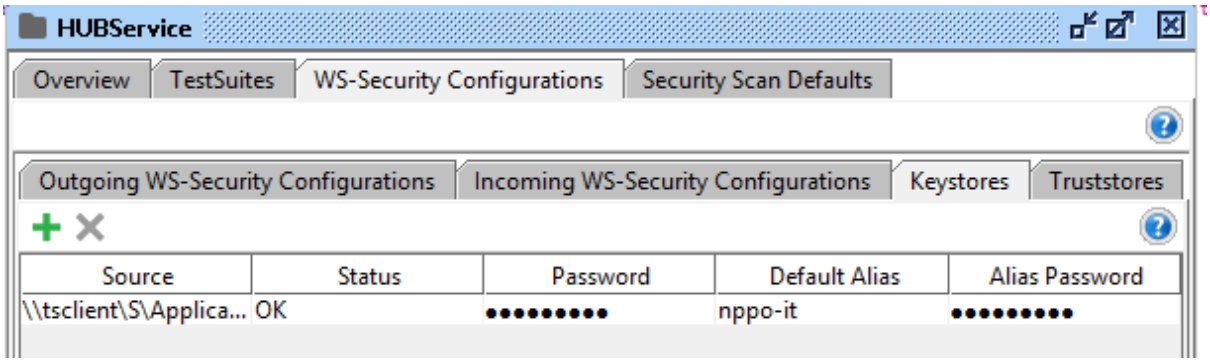


10. En el nuevo cuadro de diálogo ir a "Configuraciones de WS-Security" y dentro de esta pestaña, ir a la pestaña "Depósito de claves".
11. Haga clic en el botón de signo más verde para agregar su certificado de cliente. Este símbolo de botón verde está en la esquina superior izquierda de la ventana.
12. En la ventana del buscador, seleccione el depósito de claves del certificado con extensión P12 y formato.

13. Escriba la contraseña del almacén de claves en la línea de comandos de Windows.
14. Una nueva fila de depósito de claves aparece en la ventana con el estado OK.

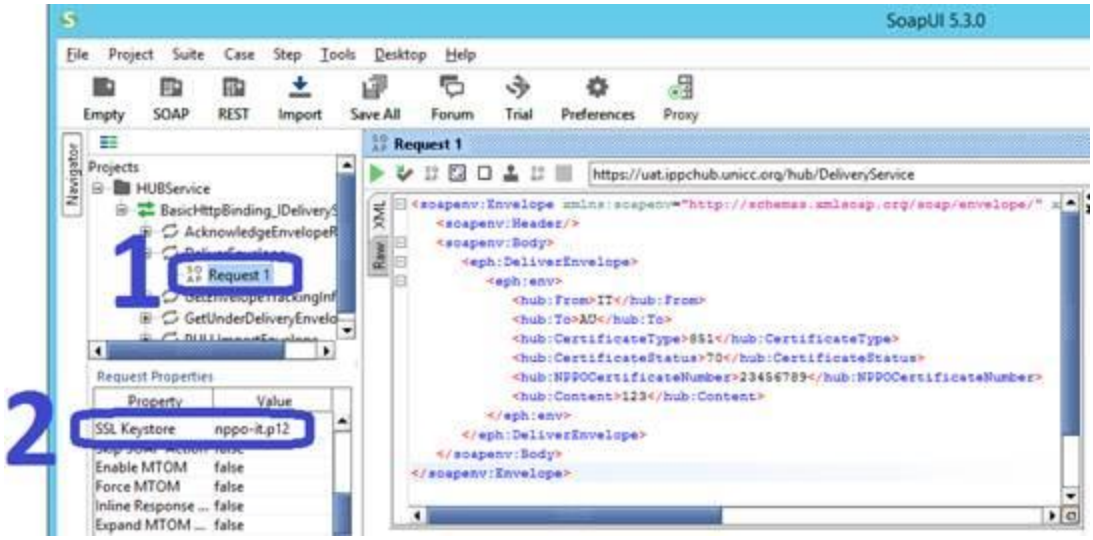


15. Puede agregar el sobrenombre de su certificado y la contraseña. En nuestro caso el sobrenombre del certificado es "onpf-"

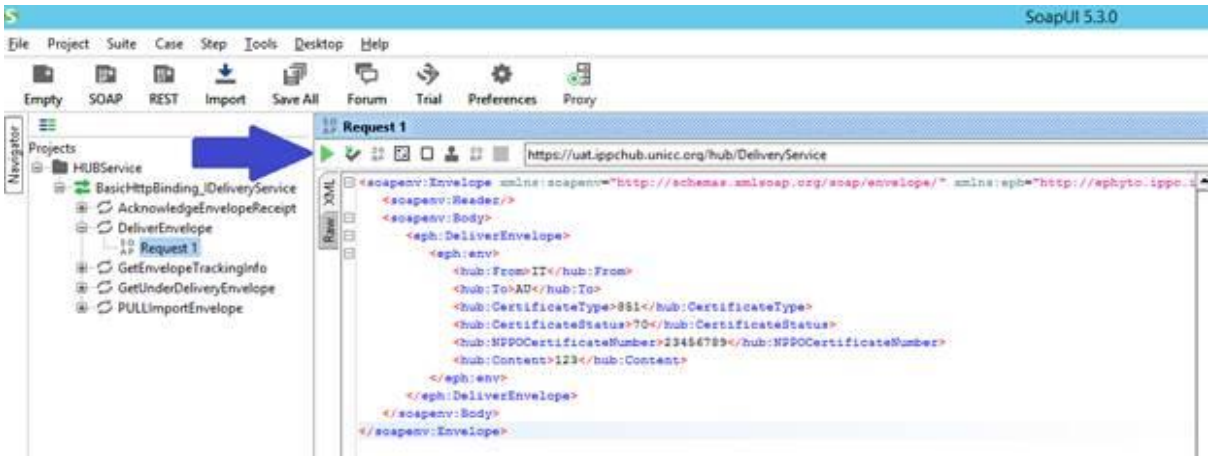


16. Cierre esta ventana y haga clic en la solicitud de DeliveryEnvelope (el siguiente procedimiento es válido para todas las solicitudes). Aparece una plantilla de solicitud:

17. Incluya su certificado en la solicitud, presentada en el almacén de claves SSL (hacer lo mismo para el resto de las peticiones que desee probar):



18. Rellenar la plantilla con datos válidos como en el ejemplo, y haga clic en RUN (recuerde: "hub:From" en el código de su país, y debe ser el mismo que el código de país de su certificado):



19. Usted recibirá la respuesta:

20. Si guarda el proyecto y cierra SOAP IU, es necesario repetir los pasos 9 a 15 cuando abra de nuevo SOAP IU.