中文版

HUB Web Service API

IPPC ePhyto HUB

v1.11

Public - FAO/IPPC

Table of Contents 目录

文档简介 **4**

修订历史 **4**

发布 **4**

文档线路图 **5**

1. 介绍 **5**

1.1 目的 5
1.2 目标受众和阅读建议 5
1.3 参考资料 5

2. HUB 加载 **5**

3. 技术支持 **6**

4. 网络服务系统 **6**

4.1 测试环境(UAT) 6
用于测试的 URL 6
网络服务客户端身份验证的证书 6
4.2 生成环境 8
4.3 身份验证 8

5. HUB XML 模式 **8**

5.1 Schema 模式 8
信封内容 10
信封标头数组 10
信封数组 10

6. 操作 **10**

6.1 连接 hub 10
6.2 信封投递 12
6.3 PULLImportEnvelope, AcknowledgeEnvelopeReceipt, AdvancedAcknowledgeEnvelopeReceipt 15
6.4 GetUnderDeliveryEnvelope 17
6.5 GetImportEnvelopeHeaders & PULLSingleImportEnvelope 18
6.6 GetEnvelopeTrackingInfo 20
6.7 GetActiveNppos 22
6.8 ValidatePhytoXML 22
6.9 DeliverPhytoEnvelope 23
6.10 Receiving a PUSH delivery 23

7. 序列图 **29**

8. 使用 SOAP UI 进行测试 **30**

文档简介

Author:	UNICC
Owner:	UNICC
Client:	FAO/IPPC
Document Number:	1.10

修订历史

Version:	Who:	What:	When:
1.0	UNICC	Primary Document	12/12/2016
1.1	UNICC	Revision after PTC meeting in Geneva	22/03/2017
1.2	UNICC	Iteration 2 Review	31/07/2017
1.3	UNICC	Iteration 3 review	11/09/2017
1.4	UNICC	Revision after PTC meeting in Valencia	03/10/2017
1.5	UNICC	Java client sample code added	24/10/2017
1.6	UNICC	Receiving through PUSH Sample implementation added	16/11/2017
1.7	UNICC	Reviewed HUB Admin console urls	04/01/2018
1.8	UNICC	Updates of the March 2018 Release	22/03/2018
1.9	UNICC	Updates of the April 2018 Release	07-May-2018
1.10	UNICC	Review of external links	01-Jun-2018
1.11	UNICC	July 18 release	16-July-2018
1.12	UNICC	Oct 18 release	9-Nov-2018

发布

本文档发布如下，从版本 1.10 开始，该项目作为 HUB 发布通信的一部分进行发布，并在 HUB 管理控制台下托管。

Name	Title	Date of Issue	Version
IPPC	HUB Web Service API	28/03/2017	v1.1_Early Release
IPPC	HUB Web Service API	01/08/2017	V1.2_early Release
IPPC	HUB Web Service API	20/09/2017	1.3
IPPC	Hub Web Service API	12/10/2017	1.4

HUB Users	Hub Web Service API	20/10/2017	1.4
IPPC	Hub Web Service API	24/10/2017	1.5
IPPC	Hub Web Service API	16/11/2017	1.6
IPPC	Hub Web Service API	04/01/2018	1.7
HUB Users	Hub Web Service API	22/03/2018	1.8
IPPC	Hub Web Service API	07-May-2018	1.9

文档线路图

以下，是文档计划要增强的内容。

Feature
Include Sequence Diagram to describe the PUSH receving type flow

1. 介绍

1.1 目的

本文档描述了 IPPC HUB 网络服务的有关内容，它可以看做是解决客户连接 HUB 过程中软件方面问题的指南。

注意: 该文档将随 HUB 项目保持更新，最新版本请查阅
https://www.ephytoexchange.org/doc/HUB_Web_Service_API.pdf

1.2 目标受众和阅读建议

本文档的目标受众是 NPPOs 的开发人员和系统设计者，他们将对连接 HUB 的几个要素进行评估并发布。本文档可用于完善 IPPC HUB 系统和 IPPC GeNS 系统界面，也可作为运行 HUB 系统的标准文档。本文档应当同 ePhyto HUB Software Requirements Specification 对比阅读，它是有关 HUB 系统网络服务操作和使用的文档。

1.3 参考资料

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

2. HUB 加载

- 该过程的第一步是通过 https://www.ephytoexchange.org/onboard 向 NPPO 发送注册请求。
- 通过 IPPC 指定国家官方联络点的验核和确认后，使用者的账户将自动生成，并发送回执。
- 在获得对管理控制台的访问之后，HUB 管理员将会联系中心点，并发送最新版本的文档，给 NPPO 在 UAT 网站进行初始设置提供支持，在最终发布到生产环境之前，在 UAT 网站可以进行测试和验证。

3. 技术支持

在测试阶段遇到问题，或者与 HUB 测试相关的其他技术问题可以在门户网站 https://www.ephytoexchange.org/support 上提出（仅限已注册使用者）。

我们同样鼓励使用协作工具快速获取答案以及分享经验（仅限已注册使用者）。

对于 HUB 系统的一般性问题请点击 https://www.ephytoexchange.org/support 进行查询。

如果你在测试与 HUB 连接的几个要素时遇到问题，你可以在成功登录后，通过以下链接的相应菜单向管理控制台提出技术支持请求(https://www.ephytoexchange.org/AdminConsole)。

系统会发送邮件给技术小组帮助解答问题。

我们建议先查看管理控制台中合作区域的相关资源信息。

4. 网络服务系统

4.1 测试环境(UAT)

测试环境(UAT)是一个实时系统，具有系统的最新版本，可以测试客户端应用程序连接到 HUB 的实施。

为了便于操作，可以提供自签名证书和特殊证书。

用于测试的 URL

HUB UAT/测试环境可以通过以下 URLs 访问：

https://uat-hub.ephytoexchange.org/hub/DeliveryService?wsdl (web service WSDL)
（网络服务端点只接受证书验证）

https://uat-hub.ephytoexchange.org/AdminConsole (Admin interface)

在 NPPO 加载过程中，将提供登录到控制台的信息。

网络服务客户端身份验证的证书

在测试阶段，客户端身份验证将使用自签名的证书。NPPOs 可以使用 JDK 中的“密钥工具”签发证书，或者可以请求 UNICC 提供 NPPO 能够使用的证书样本。

2018 年 3 月公布的 NPPO 管理员可以访问 HUB 管理控制台并更新公共证书，当他们使用服务的时候，这些证书将用于验证 NPPO 客户端应用程序。

4.1.1.1 申请测试证书

NPPO 需要为证书创建 *X.500* 甄别姓名提供以下信息：

- 常用名
- 组织单元
- 组织名称

- 所在地点
- 所在国家
- 两个字母国家代码

UNICC 会发送带有 PKCS12 格式的密钥存储文档的证书，并向 HUB 管理控制台 NPPO 文档中上传公共密钥。

4.1.1.2 生成自签名测试证书

JDK 提供的“密钥”命令可以生成证书，一旦 NPPO 生成证书，就必须将公共钥匙发送到 UNICC，以便于将其添加到服务器的信任存储区。

位于英国伦敦的 NPPO 实体的证书生成示例，有效期为 10 年：

```
C:\certificates>keytool -genkey -alias nppo1 -keyalg RSA -keysize 1024 -keystore nppo.keystore -validity 3650 -keypass nppo1pass -storepass nppoStore1pass
What is your first and last name?
  [Unknown]:  www.nppo.mycountry
What is the name of your organizational unit?
  [Unknown]:  NPPO
What is the name of your organization?
  [Unknown]:  NPPO-MyCountry
What is the name of your City or Locality?
  [Unknown]:  Capital
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:  MC
Is CN=www.nppo.mycountry, OU=NPPO, O=NPPO-MyCountry, L=Capital, ST=Unknown, C=MC correct?
  [no]:  yes
```

从密钥库导出公共密钥的示例：

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

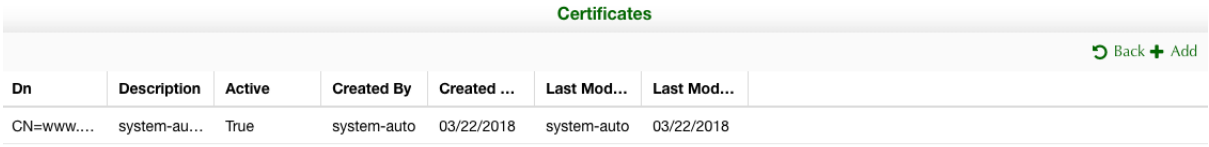
一旦生成，NPPO 可以通过以下步骤在管理控制台上传并配置证书：

1) 进入管理控制台，导航到 NPPO 文档 

2) 使用链接中 NPPO 文档按键，预览证书。 

3) 客户端证书在预览中形成列表（如下图所示），它们可以被设置为非实时的，通过使用“add”按键，

可以上传新的证书。



4.2 生成环境

HUB 生成环境可以通过以下 URLs 进行访问：

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

（网络服务端点只接受证书验证）

https://www.ephytoexchange.org/AdminConsole (Admin interface)

4.3 身份验证

网络服务的身份验证支持与访问 HUB 的每个国家相关联的 TLS 1.1 和 TLS 1.2 客户端证书。X509 证书是客户端凭据。每个连接的应用程序都由一个定义的证书，该证书由被认可的证书颁发机构签发，证书颁发机构将会根据 HTTPS 协议验证 HUB 的客户端应用程序。安全实施的细节超出本文档的范围，但是包含在引用 HUB 需求文档规范中。

HUB 仅接受与 NPPO 连接的“From”字段与 TLS 证书相匹配的“envelopes”

5. HUB XML 模式

5.1Schema 模式

HUB 网络服务由大量实体组成，其中一些实体是 ePhyto 定义的一部分，我们将在每一个网络服务操作中对细节进行更多描述，主要元素详见以下清单：

1) 信封标头
2) 信封内容
3) ePhyto 信封

本文档（第 6 节）中对 WSDL 的定义具有多个操作，主要由以下实体支持：

a. 信封标头
b. 信封=标头+内容
c. ePhyto 信封=标头+SPS 证书
d. 信封标头数组
e. 信封数组
f. HUB 追踪信息
g. NPPO
h. 验证结果

信封标头

信封标头元素用于在没有对实际证书内容进行预览或加工的情况下交换 ePhyto 证书信息。

当这些属性不符合标准时，HUB 将被检测，以验证这些代码是否使用正确并引发通信错误。

在与 HUB 交互作用期间，不强制设置标头的所有元素。但是，一些识别元素是最低限度的要求。

信封标头包括以下元素：

- 来自：按照 ISO 3166-1 标准，出口国家的 2 位字母国家代码
- 去往：按照 ISO 3166-1 标准，进口国家的 2 位字母国家代码
- 证书类型：这是证书类型的 UNECE 代码。在 IPPC 执行过程中，HUB 将仅对以下两类数字类型代码进行核对。
  - 851 for Phyto 代表植物检疫证书的代码 851
  - 657 for Re-Export Phyto 代表转口植物检疫证书的代码 657
- 证书状态：这是证书状态的 UNECE 代码。在 IPPC 执行过程中，HUB 将对以下数字状态代码进行核对。
  - 70:Issued 发布
  - 39:Approved 批准
  - 40:Withdrawn 撤回
  - 41:Rejected 拒绝
- NPPO 证书编号：为了便于参考，NPPO 出口方会将 ePhyto 证书编号录入信封。这样以来，将可以允许 NPPO 国家系统针对系统中的 HubTrackingNumber 与证书匹配。另外，HUB 使用者端口也将检测该数字代码的传递状态。这一元素是多语言同时进行，允许出口国 NPPO 选择不同语言且需要限定在 1000 个字符以内。

- HUB 追踪号码：这是唯一的标识符，当第一次接收到信封时，HUB 将为每一个信封分配标识符。NPPO 系统随后可以针对标识符查询 HUB，以获取通过 HUB 追踪号码标志的任何特定证书的交付信息。这个元素的大小可以扩展到 50 个字符。
- HUB 追踪信息：这个元素包含下面四个状态代码中的一个，显示信封在 HUB 中的交付状态：
  - **尚未投递**：表示信封仍然保存在 HUB 中没有投递，队列有效期也没有超期，因此，HUB 仍然保留着信封。
  - **已投递**：信封已被 HUB 成功投递，并在投递后删除。
  - **投递失败**：HUB 还没有投递，NPPO 出口方设定的队列有效期已到，因此，信封从 HUB 队列里删除。
  - **信封不存在**：对于给定的追踪号码，HUB 没有任何信息。
  - **投递警告**：在 2018 年 3 月发布的版本中，将用来标记那些进口国发出的模式不符的警告文本信封，这些文本可以被出口国读取和使用，并生成全球标准化的 XML 格式。

- HUB 错误消息：这个元素包括与 HUB 交互作用时可能产生的不同错误信息。大部分错误消息都与队列有效期相关。在 2018 年 3 月发布的版本中，进口国家可以设置关于 AdvancedAcknowledge 的警告消息，指出他们收到的 ePhyto XML 中需要改进的元素。

信封内容

信封类型继承信封标头，并使用“Content” 元素对其进行扩展，该元素可以是任何类型的字符串/xml。

电子植物检疫证书可以由 NPPO 出口方客户端应用程序生成，序列化到 XML 格式，使用信封“Content” 属性发送到 HUB。

HUB 不会对证书内容进行验证，必须采取 ISPM 12 模式。NPPO 进口方客户端应用程序可以打开证书内容并确认其符合标准。NPPO 进口方客户端应用程序在接收到信封后必须确认接收消息，无论证书的验核情况如何，这一步骤将与独立的业务流程一并执行。

以下参考文档包含 XML 映射必要条件和资源的所有细节。

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

信封标头数组

这个元素用来同时交换大量信封标头。‘GetUnderDeliveryEnvelope’ 和 ‘GetImportEnvelopeHeaders’的操作详见以下信息。

信封数组

这个元素包含信封清单，每个信封包含-一个标头和一个电子植物检疫证书。这个实体在 ‘PULLImportEnvelope’ 中使用，具体细节详见以下信息。

6. 操作

6.1 连接 hub

连接到 HUB 不是网络服务公开的操作，而是调用远程网络服务操作之前的内部调用。

在本节中，我们将展示用于打开连接 HUB 客户的通用代码，通用代码是使用 C# 和 the .Net 框架 4.6.1 以及 Java 1.8 a 和 the Apache Axis 1 框架，由 WSDL 定义生成客户端代码。

代码将创建新的客户，添加证书和 URL（取决于环境），以便在网络服务的所有后续调用中使用。

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 信封投递

NPPO 出口方将进行把信封发送到 HUB 的操作。信封标头必须最低限度填写一下属性：

- 来自
- 去往
- 证书类型
- 证书状态
- NPPO 证书编号（不是强制的，但是我们建议填写，以便在出口商系统中查询证书原件的传输）
- 'Content' 属性中填入了实际证书，以生成 ePhyto XML 序列化版本完成信封。

HUB 通过信封标头进行反馈，信封标头包含所有 NPPO 出口方客户端应用填入属性，其中包括通过 HUB 应用程序添加的 HUBTrackingNumber 和 the HUBTrackingInfo 属性。

在'Transit'的状态下，证书必须被分发到传递国家，客户端应用程序应当将信封作为单独的消息发送给所有国家，并对每一次传输进行单独跟踪。

Client sample implementation in C# generated as .Net 2.0 standard web service client:

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);
    }
  }
```

如果有错误发生，HUB 追踪信息会设置到“FailedDelivery”，具体细节在信封标头返回的 hubDeliveryErrorMessage 元素中。可能发生的错误包括：

- NPPO 发送的信封不是“From” 字段中的国家
- 在系统“To” 字段中没有 NPPO 国家
- 证书类型无效

- 证书状态无效

例如网络运行中断或者系统无法使用这样的连接问题，当问题无法通过远程应用产生时，将会作为标准 HTTP 协议错误上报。

6.3 PULLImportEnvelope, AcknowledgeEnvelopeReceipt, AdvancedAcknowledgeEnvelopeReceipt

NPPO 进口国配置 PULL 操作将按照系统预设条件使用本操作来检索所有的信封。经过身份验证的客户端代表进口国，将接收 HUB 队列中与进口国一起位于“To ”字段的所有信封。对于每一个信封，进口国应在操作 AckknowledgeEnvelopeReceipt 时，使用 HUBTrackingNumber 将成功接收到的每一个信封的信息传回。确认信息将从队列中移出，下一个 PULL 操作将获取剩余的信息，直到结果为空。

NPPO 配置将允许成为减少每一次 PULL 的消息批量处理，以便优化与网络连接不畅的办公室进行通信。

在 2018 年 3 月发布的版本中，支持通过与确认操作相关的文本消息进行交流，设置跟踪信息到 DeliveredWithWarnings，并提供报错消息，当接收和打开 XML 时，可以看到报错问题的具体细节。

请注意：警告消息被限制在 200 个字符以内，返回消息可能表明这样的警告，为防止超过字符限制，数据将可能被截断。

请参阅下面的示例，可以从模式验证操作来提取此类消息，利用统一的 XML 将其报告给出口方。

客户端实现示例

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);
    }
  }
```

如果在 PullImportEnvelope, AckowledgeEnvelopeReceipt 或者 AdvancedAckowledgeEnvelopeReceipt 的处理过程中发生错误，系统将发送一个标准 SOAP Fault 描述错误的元素。在这些服务中，检测到的错误包括：

- 提出请求的 NPPO 不在系统中
- 接收请求方 NPPO 的确认，并非来自确认标头 “To” 字段中的国家。
- 当发送的号码在 HUB 中找不到，如上所述与确认请求有关，信封无法找到。

6.4GetUnderDeliveryEnvelope

该操作允许出口国 NPPO 获得在传递过程中所有信封标头的列表，经过身份验证的客户端代表出口国。HUB 将返回所有等待传递的信封列表。

客户端应用程序可一次从返回的信封标头使用 HUBTrackingNumber，并更新系统。

客户端实现示例

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);
    }
  }
```

如果处理 GetUnderDeliveryEnvelope 时发生错误，将发送一个标准 SOAP Fault 描述错误的元素。本服务检测到的错误包括：

- 发出请求的 NPPO 不在系统中

6.5GetImportEnvelopeHeaders & PULLSingleImportEnvelope

与上一个操作类似，这个操作允许进口国 NPPO 获得传递过程中所有信封标头的列表。经过身份验证的客户端代表进口国，HUB 将返回所有等待传递的信封列表。

客户端应用程序可以从返回的信封标头使用 HUBTrackingNumber，并逐个提取他们。这将允许进口国处理要传递的消息的整个子集，而不必将它们按批提取。

客户端实现示例

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);
    }
  }
```

如果在处理 GetImportEnvelopeHeader, AcknowledgeEnvelopeReceipt 和 AdvanvcedAcknowledgeEnvelopeReceipt 时发生错误，将发送描述错误的元素，本服务检测到的错误包括：

- 提出请求的 NPPO 不在系统中

6.6GetEnvelopeTrackingInfo

此操作为给定的 HUBTrackingNumber 提供 HUBTrackingInfo。如果客户端应用程序已经发送了信封，而信封标头没有在等待交付的列表中，那么系统应当询问 HUB，以了解它是否传递成功，和/或处于哪个阶段。追踪信息的引用在上面的操作中已完成，下面的代码传递中对此有所注释。

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);
    }
  }
```

如果有错误产生，在 SOAP 接收到时将退回错误信息。检测到的错误可能包括：

- 提出请求的 NPPO 不在系统中
- 提出请求的 NPPO 并非来自 “From” 字段的国家

6.7GetActiveNppos

本操作是一个简单由 HUB 中的 NPPO 返回的咨询行为，仅含有国家代码、发送和接收标记。这些标记可能被客户端应用程序用来自动发送或接收来自相关国家在 HUB 中的状态。

6.8ValidatePhytoXML

此操作公开了管理控制台的功能，用于根据最新的 ePhyto 模式验证 XML。

这可以用来收集和警告有关传入消息和传出消息的一些问题。

当使用 SoapUI 时，建议将要验证的 XML 包装成<![CDATA[….]]>元素来复制粘贴源文本。

这里没有提供代码示例，因为它们与上面描述的操作有所不同，操作的结果将以一个验证结果数组的形式返回，如下所示：

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

该字段表明这一问题的源文件，msg 表明问题的描述性信息。

可能的领域如下：

-强制元素（必须作为文档结构一部分存在的元素）

-映射（与 ePhyto 到模式的映射相关的问题）

-模式（与不符合 XML 模式相关的问题）

可能的级别如下：

-严重：导致读取和查阅证书时出现问题

-警告：没有导致问题，需要对如何生成 XML 进行一些修改

-提醒：需要申请优化等级

6.9DeliverPhytoEnvelope

该操作将在 HUB 中公开一个新的实体，ePhytoEnvelope 是从 EnvelopeHeader 继承而来的，而不是信封中定义的字符串元素。如果在信封中定义，ePhytoEnvelope 具有使用验证工具模式定义的 SPS 证书。

该操作类似于传递信封，将需要上述定义的所有信息和定义有效的 SPS 证书实体。在对传递内容进行排队确认之前的操作，将执行验证，如发现任何 SEVERE 级别的问题，将停止传递。

使用该项操作与定义 DeliverEnvelope 的工作流程相同，可以利用客户端应用程序编译 XML，从出口方应用程序定义的现存实体中填写必要信息，

示例代码，它只是一个如上定义的特殊化的 DeliveryEnvelope，而不是根据可用的信息设定 XML 填入整个 SPS 证书对象。

6.10Receiving a PUSH delivery

为了能够接收 PUSH 传递，NPPO 进口国必须有一个终端可用于同 HUB 连接。

为了生成所需资源，NPPO 必须使用 HUB WSDL 文档，并完善所需功能，以接收 DeliveryEnvelope 操作的信封，并使用上述带有 PULL 的 HUB 操作来确认接收。

如果有错误产生，PUSH 端点应用程序希望再次接收信封，端点的结果应当将带有追踪信息的信封标头设置为 FailedDelivery（在此情况下，HUB 将继续尝试并发送信封，直到得到确认或者得到来自端点的回应）。如果 FailedDelivery 没有得到 HUB 的回应，系统会锁定信封等待确认。

HUB 系统的 NPPO 管理员需要（使用上述支持请求）交流托管 PUSH 网络服务端点的 IP 范围。托管 PUSH 网络服务端点。HUB 的端口打开以后，HUB 系统的 NPPO 管理员将接收到一封来自 HUB 身份认证的带有 SSL 客户证书的确认邮件。

此外，NPPO 文件可设置为通过 PUSH 接收追踪信息更新（Receive Tracking Info Update）。这将意味着 HUB 将在更改跟踪信息时将信封标头发送到操作指令：SetTrackingInfoUpdate，从待传递状态到成功传递或故障状态。

请参阅以下与 PUSH 接收模式相关的 NPPO 设置。

接收模式*： PUSH

注意：为了符合安全政策，HUB 必须允许集线器与指定的端点进行通信。

请提出支持请求，说明国家托管服务中使用的 IP 范围。 HUB 支持团队将负责打开端口 443 以允许 HUB 和国家 Web 服务之间的 HTTPS PUSH 流量。

还要确保 Web 服务信任以下用于标识 HUB 的公共证书

选择以下标志以接收必须在国家推送端点中定义的 Web 服务操作（SetTrackingInfoUpdate）的跟踪信息更新。

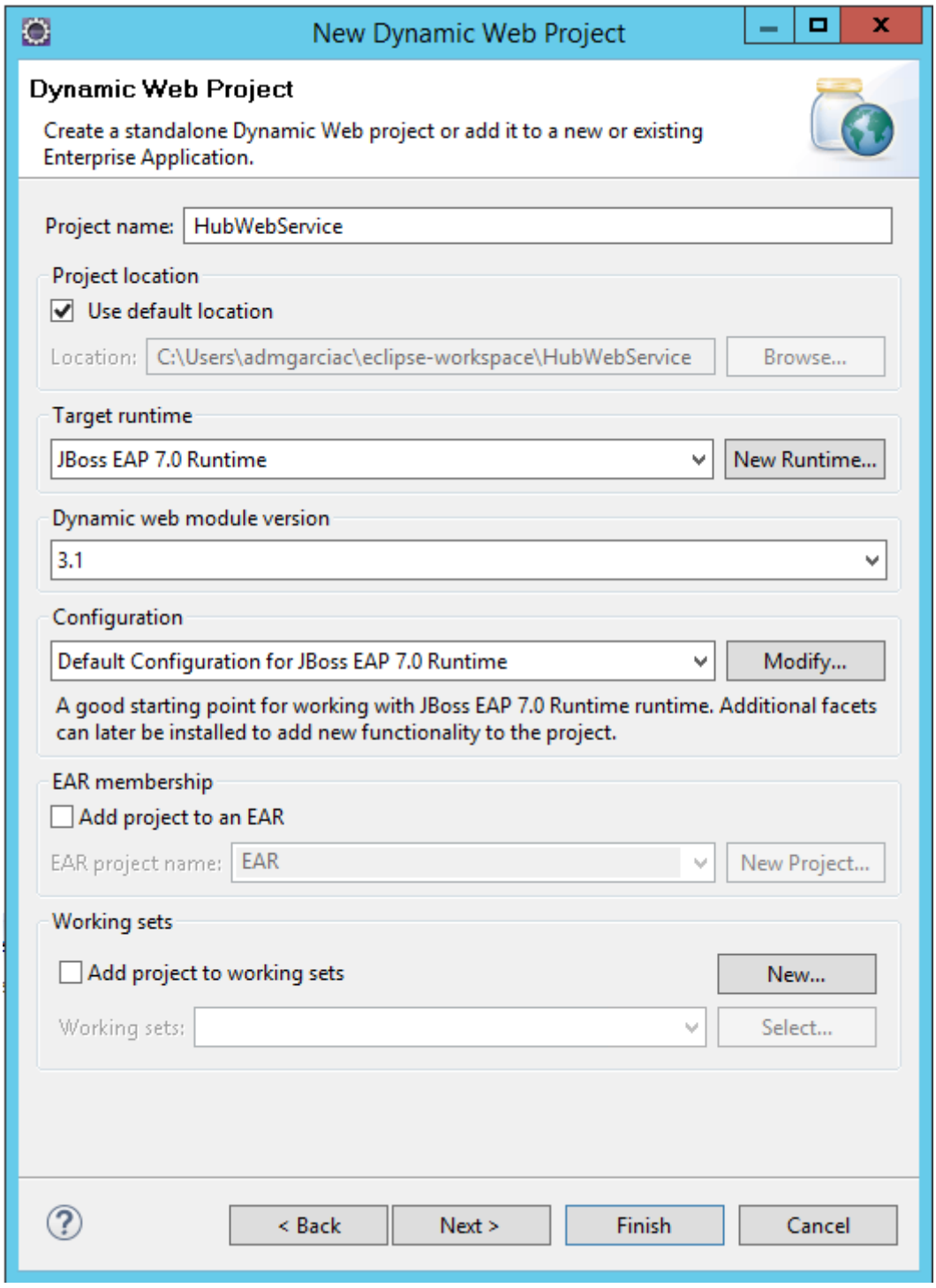
Push URL* https://localhost:8443/hub/deliveryService

Delivery Retries*: 20

Delivery Minutes*: 5

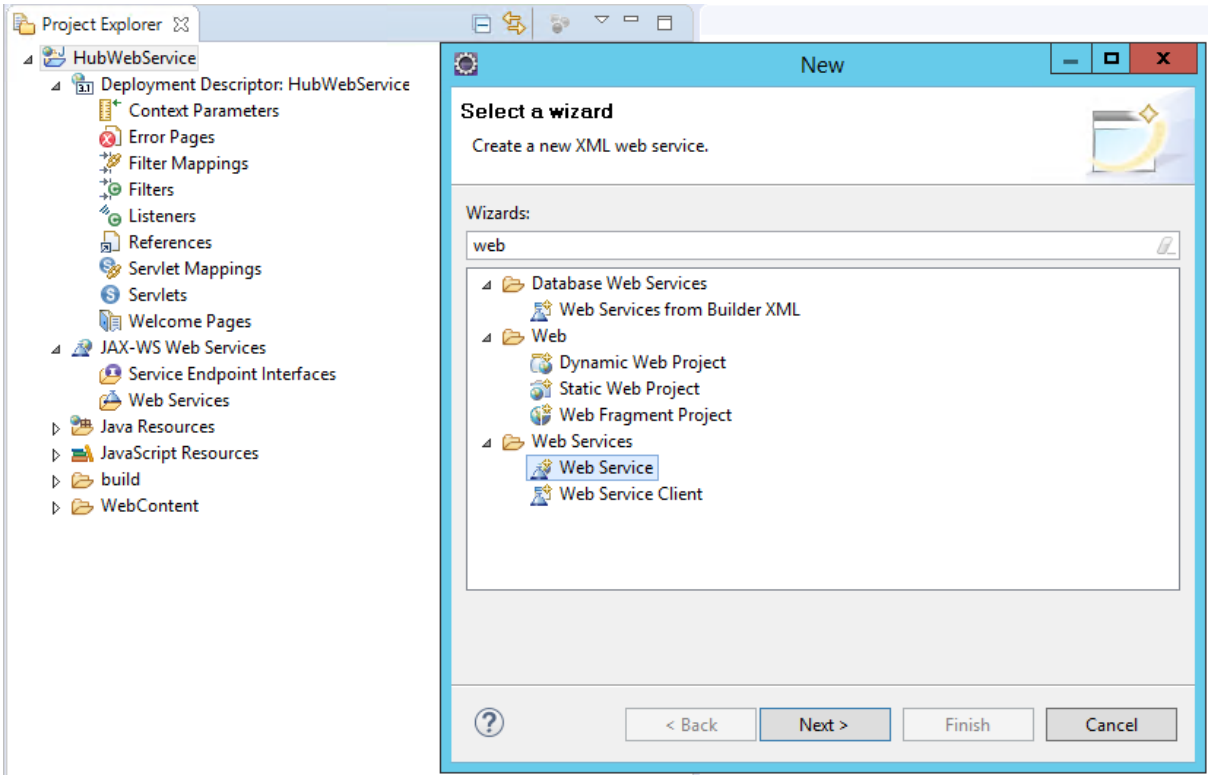
以下是一个示例，关于如何使用 Eclipse 和 Apache Axis 创建基础端点。

首先，在 Eclipse 中创建一个新的动态网络项目（在这里我们使用 JBoss 作为目标运行时间，请选择使用最适合你所需的运行时间）。



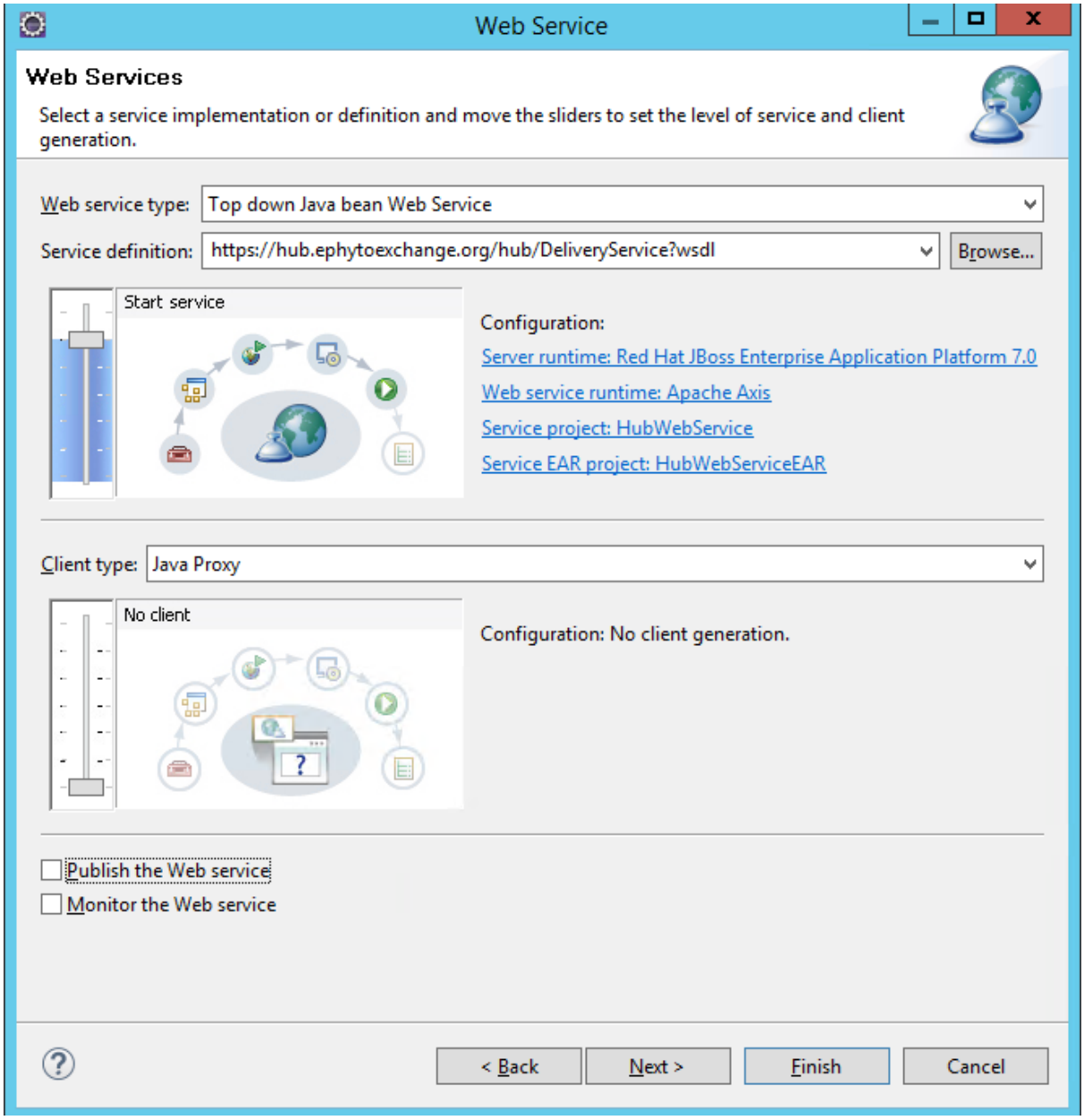
你可以在这个对话框中点击“Finish”键。

当项目建成，你需要创建网络服务端口的执行类别。对此，你需要在项目中添加一个新的网络服务，通过右键点击项目名称，然后点击“new”和“Other…”。



Select “Web Service” and click “Next”

选择“网络服务”，点击“下一步”。



我们已经创建了 WSDL 文档，选择“Top down Java bean Web Service”，然后使用下方的链接进入 Hub WSDL URL：https://hub.ephytoexchange.org/hub/DeliveryService?wsdl

我们使用“Apache Axis” 和 Jboss 作为服务器时，如果你有不同的应用服务器，通过点击“Server runtime”链接进行选择。确认应用服务器已开始运行后，点击 “Finish”。

当整个流程已完成，Eclipse 会打开文档：DeliveryServiceSoapBindingImpl.java，在这个文档中有完整的代码。在此情况下，我们只需要执行“deliverEnvelope”，这是 HUB 的 PUSH 服务将要调用的方法。

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);
    }
```

保存信封，然后确认从 HUB 接受信封，这样就被标记为已传递。

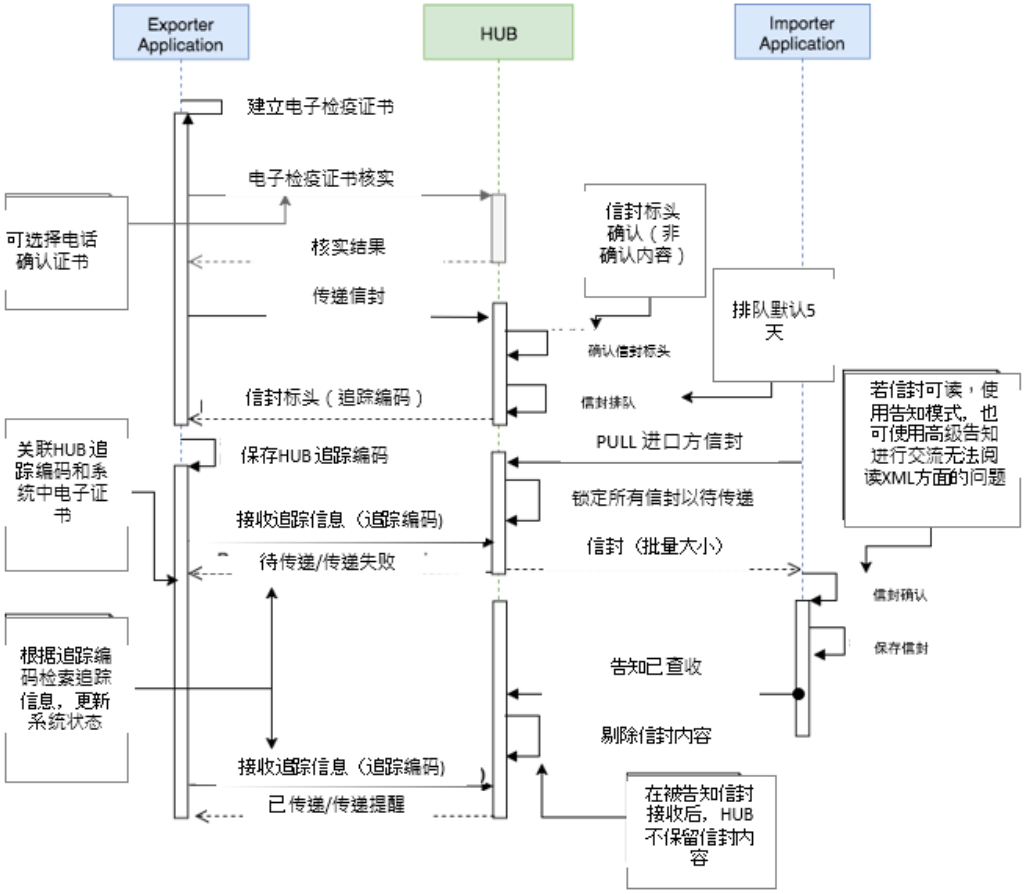
请注意，HUB 客户是指与 HUB 网络服务实现连接的对象。

在上面的示例中，我们没有提供关于如何设置客户端证书身份认证的指南，因为根据底层平台和基础设施的不同，客户端证书身份认证可能会有很大差异。要实现推送端点，NPPO 应用程序应当接受提供客户端身份认证的 HUB 证书（客户端证书将被提供给托管 IP 范围的国家，通过开放所需端口作出回应）。

7.序列图

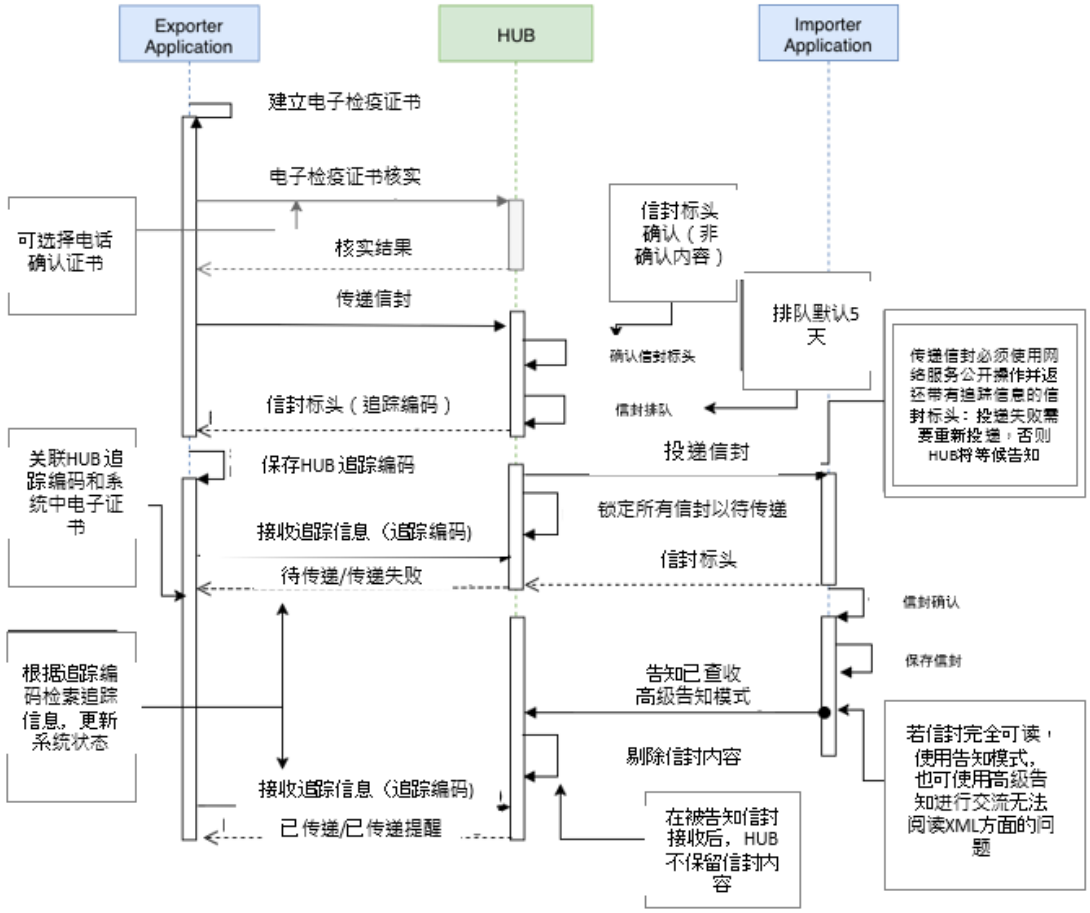
7.1 PULL 传送模式

下面的序列图展示了最佳的信封传递过程，这个过程是在 NPPO 客户端应用和 HUB 系统使用 PULL 接收类型之间交互进行的。



7.2 PUSH 传送模式

下面的序列图展示了在 NPPO 客户端应用和使用 PUSH 接收类型的 HUB 系统之间交互进行的信封传递过程



8.使用 Soap UI 进行测试

注意：每一次 SOAP UI 启动，第 9-15 步都要重复执行。

请按照下面的步骤使用 SOAPUI 进行测试：

1、下载并安装 SOAP UI，我们使用的是 5.3 版本。

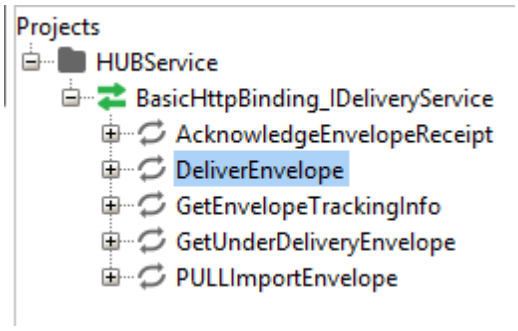
2、 转到安装文件夹 bin 目录，打开 SOAPUI-5.3.0.vmoptions 文档。在 windows 电脑上，文档所在位置为“C:\Program Files\SmartBear\SoapUI-5.3.0\bin”，在 Mac 系统中，文档所在位置在 /Applications/SoapUI-5.3.0.app/Contents/vmoptions.txt. 项下，你可以使用管理员权限对文档进行编辑。

3、 文档的最后，包括以下这行：

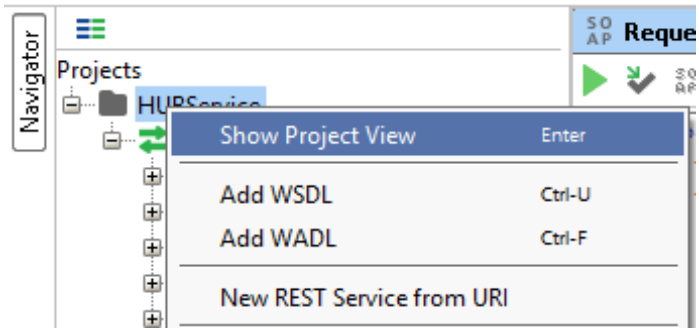
-Dsoapui.https.protocols=TLSv1.1,TLSv1.2

4、保存文档，打开或者再次打开 SOAP UI

5、转到 New SOAP Project 文档
6、在 “Project Name” 字段，选择一个具有描述性的项目名称
7、在“Initial WSDL” 字段，如果你接收文档，或者想要把 wsdl 文档保存到电脑里，请选择为端点提供的 URL ，或者选择 wsdl 文档。
8、点击“OK”以后，SOAP UI 将会生成一些带有操作请求的模板。
这里你可以看到生成请求模板的示例：



9、右键点击项目名称，在我们的例子中是 “HUBService”，然后选择 “Show Project View” 。



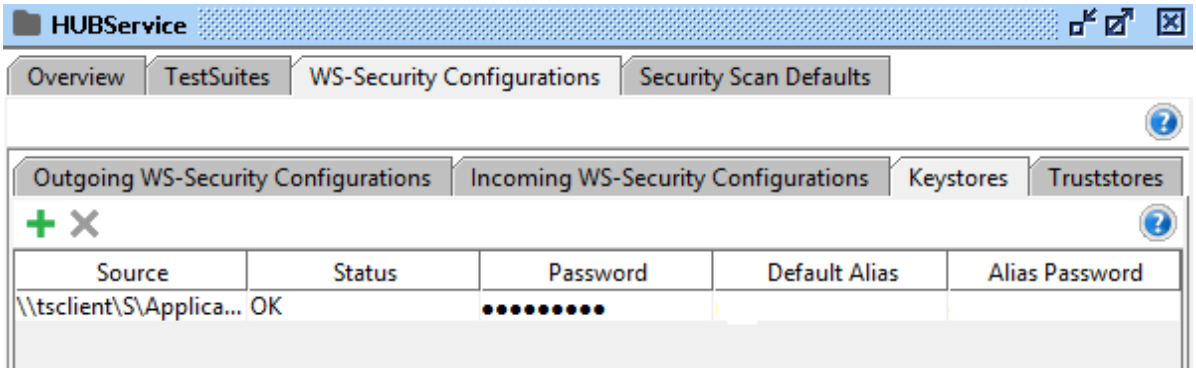
10、在新的对话框中，转到“WS-Security Configurations” 选项卡，在这个选项卡中，转到 “Keystores” 选项卡。

11、点击绿色加号按钮，用以添加你的客户端证书。这个绿色加号按钮位于窗口左上角。

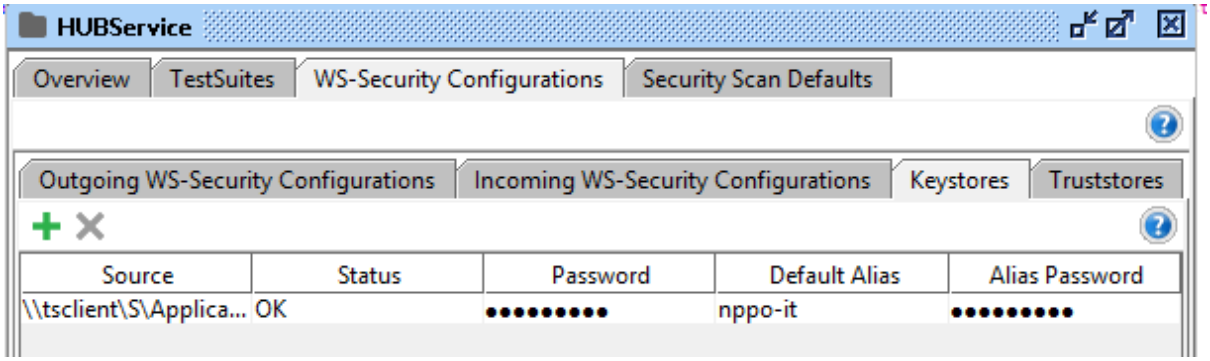
12、在浏览器窗口，选择你的证书密钥库，扩展名和格式为 P12。

13、在提示窗口输入密钥库的密码。

14、窗口将出现一个新的密钥库存储行，显示状态 OK。

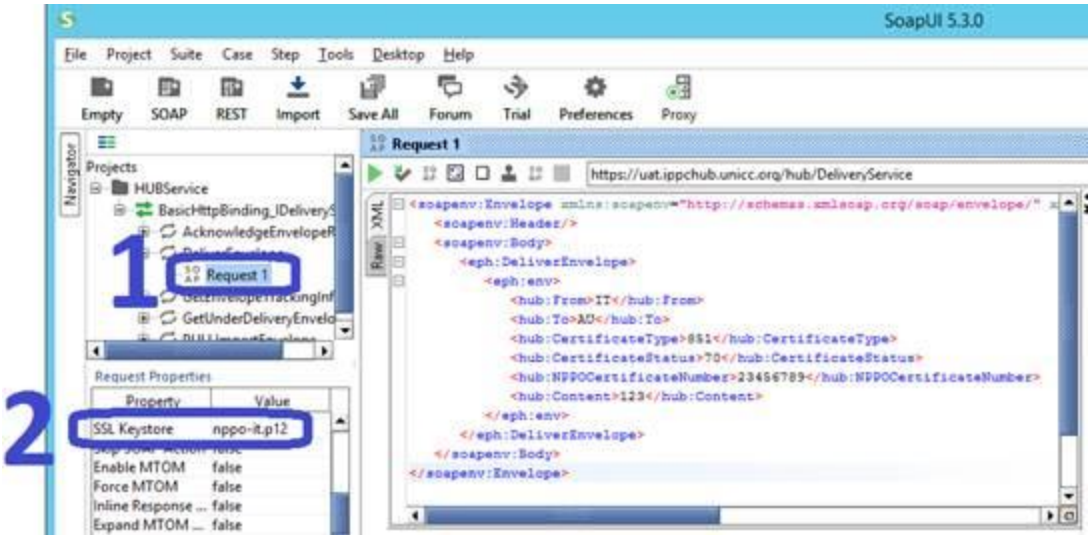


15、Add you certificate alias and password, in our case the certificate alias is “nppo-it”
添加你的证书别名和密码，在我们的示例中，证书别名是“nppo-it”。

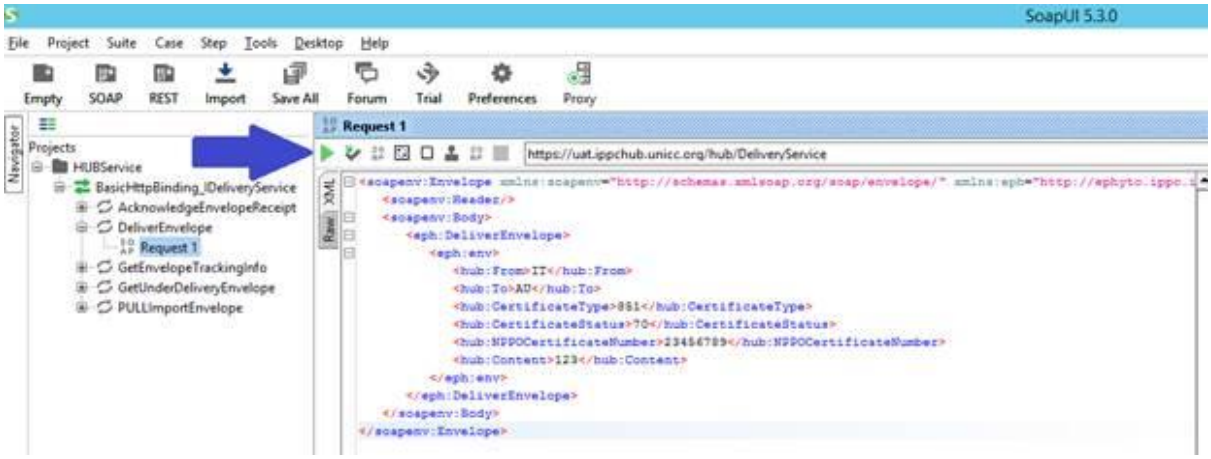


16、关闭窗口，点击 DeliveryEnvelope 请求，会出现一个请求模板：

17、把你的证书在 SSL 密钥库字段增加到请求中。



18、像示例这样，使用有效数据填写模板，然后点击“RUN”。



19、你会收到回复。

20、如果您保存项目并关闭 SOAP UI，当你再次打开 SOAP UI 时，需要重复步骤 9-15.