PayEx Connect

Implementation manual – Common

PayEx Sverige AB
Postadress:
621 88 Visby
Besöksadress:
S:t Hansplan 1
Tel: +46(0)498-20 20 00
Fax: +46(0)498-20 20 01
info@payex.com
Org.nr: 556735-5671
Säte: Stockholm
www.payex.com

1 Versions

Version	Date	Description	Created by	Approved by
1.0	2006-03-20	Document created	Reine Olofsson	
1.0.1	2006-05-03	Java JWSDP sample added	Helge Dahl	Reine Olofsson
1.0.2	2006-05-12	Name change to FaktabConnect	Reine Olofsson	
1.1	2006-08-22	Name change to PayEx Connect	Reine Olofsson	
1.2	2006-10-04	Updated exception handling with detailed faultCodes	Reine Olofsson	
1.3	2006-12-20	Updated data format information	Reine Olofsson	
1.4	2007-08-30	Added information about session support and optimistic versioning concurrency control in Crm services	Reine Olofsson	
1.5	2009-01-19	Changed URL	Johan Steen	
1.6	2009-03-10	Added RoleBasedSecurityHeader	Fredrik Gradelius	
1.7	2010-11-29	Added SOAP request example. Added .NET WCF and PHP example code.	Thomas Gustafsson	

2 References

Document	Description
WS Security UsernameToken Profile 1.0	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
SOAP 1.1	http://www.w3.org/TR/2000/NOTE-SOAP-20000508
SOAP 1.2	http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

PayEx
EXPERTS IN PAYMENTS

Index

3 Architecture

3.1 Overview

PayEx Connect is a Web Service system that enables customers to access legacy services in a controlled and secure manner. PayEx Connect use standard 128 bit SSL encryption to secure the communication channel. Client authentication is managed through UsernameTokens and IP filtering. The only protocol supported is Http Soap. Http Get and Http post is not allowed.

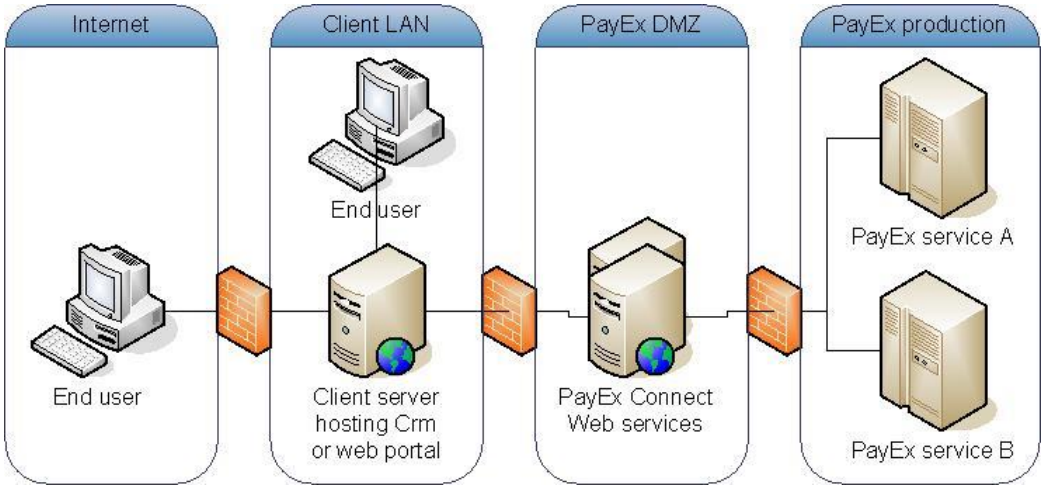


Image 1 - Architectural overview

3.2 Security

PayEx Connect authenticates the client using a UsernameToken security token. The SOAP message is not encrypted or digitally signed, the transportation is secured and encrypted through SSL. Microsoft Asp.Net clients can use the UsernameOverTransportAssertion defined in WSE 3.0 to create and manage UsenameTokens.

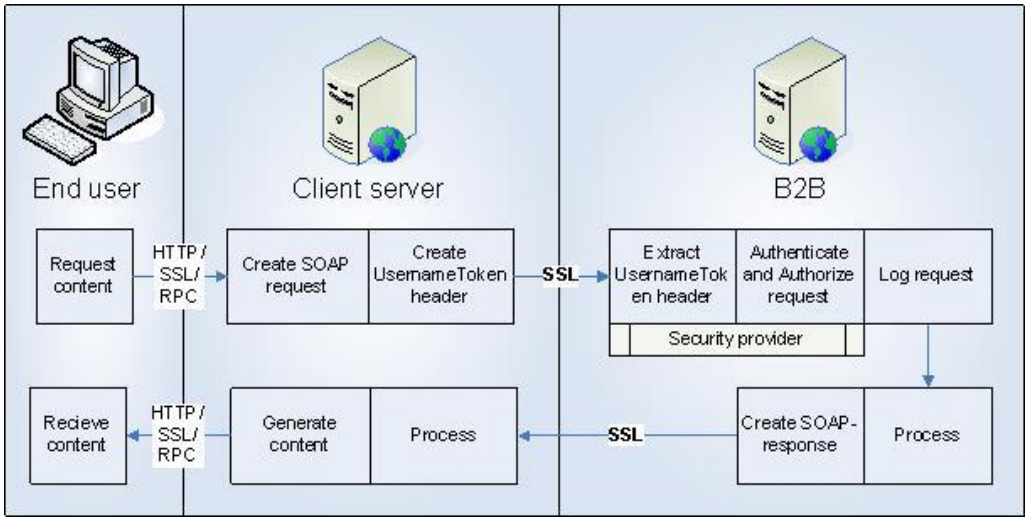


Image 2 - Security

4 Implementing PayEx Connect

4.1 Prerequisites

This chapter describes the prerequisites needed before you start to integrate a system with PayEx Connect.

4.1.1 Web Service support

Implementing PayEx Connect requires basic knowledge about Web Services including Wsdl, Soap and WS-Security.

PayEx Connect is designed to simplify integration for clients using Microsoft .Net, Java Axis or other frameworks with built in support for WS-Security and other W3C standards.

4.1.2 Service specific documentation

Detailed documentation is available on a per service level. Detailed documentation for the Sample service is always supplied by default.

4.2 Security

This chapter describes the security concepts used in PayEx Connect.

4.2.1 Service authentication

Username and password is used for authentication. The company number at PayEx is used as username. The password is client unique and provided by PayEx. Replay attacks is prevented by using message timestamps and nonces.

The security token consists of the following elements:

Username	Company number provided by PayEx.
Password	Authentication key provided by PayEx.
Nonce	Nonce is a random value that the sender creates to include in each UsernameToken that it sends. Protects against replay attacks.
Created	Created is a timestamp value that the sender creates to include in each UsernameToken that it sends. Protects against replay attacks.

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="...">
 <S11:Header>
  ...
  <wsse:Security>
   <wsse:UsernameToken>
    <wsse:Username>1234</wsse:Username>
    <wsse:Password>MyKey1</wsse:Password>
    <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
    <wsu:Created>2006-01-01T01:24:32Z</wsu:Created>
   </wsse:UsernameToken>
  </wsse:Security>
  ...
 </S11:Header>
 ...
```

Image 3 - Sample token

A failed authentication will generate error code 3011 (see 4.3). Possible causes for a failed authentication may be invalid client IP, invalid company number or invalid password.

4.2.2 RoleBasedSecurityHeader

This header is used to restrict the usage of services to predefined levels and enable trace in logging. It contains EndUserName and EndUserRole. EndUserRole is used if the referrer ip is associated with a ServiceConsumer. EndUserName is only used for logging purpose. Contact PayEx for the proper solution for your company.

The RoleBasedSecurityHeader consists of the following elements:

EndUsername	The identifier for loggtracing.
EndUserRole	Predefined role that grants access to requested service.

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="...">
 <S11:Header>
  ...
  <wsse:Security>

  ...
  </wsse:Security>
  ...
  <ns1:RoleBasedSecurityHeader
   EndUserName="name" EndUserRole="role"
   xmlns:ns1="http://faktab.se/B2B/RoleBasedSecurity"/>
  ...
 </S11:Header>
 ...
</S11:Envelope>
```

Image 4 - Sample header

4.2.3 Service authorization

PayEx Connect restrict the company numbers allowed for each client. Different company numbers may have a different subsets of services available through PayEx Connect. Requests to an unauthorized service will generate error code 3011 (see 4.3).

The functionality provided by some services may differ based on company level rules. This may result in some response data being empty or unused input variables .

4.3 SOAP Request

This is a sample SOAP request.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
  <soap:Header>
    <wsa:Action>http://faktab.se/B2B/[SERVICE]/[ACTION]</wsa:Action>
    <wsa:MessageID>urn:uuid:[MESSAGEID]</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>
         http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To> https://connect.payex.com/[SERVICE].asmx</wsa:To>
    <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="Timestamp-[ID]">
          <wsu:Created>[CREATED]</wsu:Created>
          <wsu:Expires>[EXPIRES]</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="SecurityToken-[ID]">
        <wsse:Username>[USERNAME]</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">[PASSWORD]</wsse:Password>
        <wsse:Nonce>[NONCE]</wsse:Nonce>
        <wsu:Created>[CREATED]</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <[ACTION] xmlns="http://faktab.se/B2B/[SERVICE]">
      <req />
    </[ACTION>
  </soap:Body>
</soap:Envelope>
```

[*SERVICE*] = The name of the service you are using.

[*ACTION*] = Name of the action in the service you are calling.

[*MESSAGEID*] = Unique id in the following format: aaaabbbb-cccc-dddd-eeee-ffffffffffff

[*ID*] = Random unique id, there are two if them in the document, they can't be the same!
Also they must be unique for each request made.

[*USERNAME*] = Your PayEx username.

[*PASSWORD*] = Your PayEx password.

[*NONCE*] = Random unique id for each request.

[*CREATED*] = Timestamp when the request was created in the following format: yyyy-MM-ddTHH:mm:ssZ (2010-12-31T18:35:50Z)

[*EXPIRES*] = Should be as [*CREATED*] + five minutes. (2010-12-31T18:40:50Z)

4.4 Error handling

All errors is formatted according to the current standard for SOAP Faults. Detailed information such as error code, description and id can be found in the detail element.

Code - The faultcode element is intended for use by software to provide an algorithmic mechanism for identifying the fault. Possible values are VersionMismatch, MustUnderstand, Client and Server.
In PayEx Connect the faultcode element is extended to also provide detailed error codes. See 4.4.1 for a list of custom error codes.

Actor - The faultactor element is intended to provide information about who caused the fault to happen within the message path.

Detail - The detail element is intended for carrying application specific error information. Error code, description and Guid.

```
<detail>
 <Error
   ErrorCode="3011"
   Description="Access denied for client 1.2.1.2"
   Guid="37ab9f0a-f261-4a8b-8bca-a4dcaa70de8a"
   xmlns="http://faktab.se" />
</detail>
```

Image 2 - Sample detail element

4.4.1 Custom error codes

Detail.ErrorCode	faultCode	Descrption
3001	Server.3001	Service unavailable due to maintenance etc.
3002	Server.3002	Service timed out while processing request.
3010	Client.3010	Request data validation failed. Possible causes might be missing required value, invalid date format etc.
3011	Client.3011	Required service not allowed for specified client or company number. Possible causes may be unauthorized client IP, invalid company number or invalid password.
3021	Client.3021	Requested item not found.
3022	Client.3022	The requested item is to large for processing.
4001	Client.4001	Concurrency control failure. Update not allowed for items not loaded into session state.
4002	Client.4002	Concurrency control failure. Failed to update item due to version mismatch between session state item and database item.
9998	Client.9998	Undefined client error.
9999	Server.9999	Undefined server error.

4.5 Input/Output formats

Date, time and amounts are all formatted to a common format before returned to the client.

Data type	Description
Date	Formatted as a sort able string according to the ISO 8601 specification. Sample: 2006-01-01T00:00:00
DateTime	Formatted as a sort able string according to the ISO 8601 specification. Sample: 2006-01-01T23:01:30
Amount	Formatted as a four decimal string with '.' as decimal place holder. Sample: 12345.1234

4.6 Session and versioning support

Some Crm web methods utilizes optimistic versioning concurrency control to ensure object consistency. This requires per-user session management by the service consumer. Sessions state is supported through standard Http cookies.

The session sliding timeout limit is 30 minutes.

Multiple users on the consumer system can't share the same session against PayEx Connect. If the service consumer handles multiple simultaneous users, one Http session cookie must be managed for each end user.

The SampleService web service contains sample web methods supporting per-user sessions. The Sample service can be used for testing and developing per-user sessions support in the consuming application. See PayExConnect_ManualSampleService.doc for details about the sample methods.

Error codes:

Detail.ErrorCode	faultCode	Descrption
4001	Client.4001	Concurrency control failure. Update not allowed for items not loaded into session state.
4002	Client.4002	Concurrency control failure. Failed to update item due to version mismatch between session state item and database item.

5 Usage samples

5.1 Microsoft DotNet 2.0 and WSE 3.0

5.1.1 Enabling WSE support

Before adding any Web references you need to enable WSE. This is done by right-clicking the project in the Solution explorer and selecting "WSE Settings 3.0..". This brings up the "WSE settings 2.0 tool" that helps you to configure all the settings for WSE.

Enable WSE by selecting "Enable this project for Web Services Enhancements". This option adds the required settings to the applications configuration file and ensures that all Web references added later will support WSE.

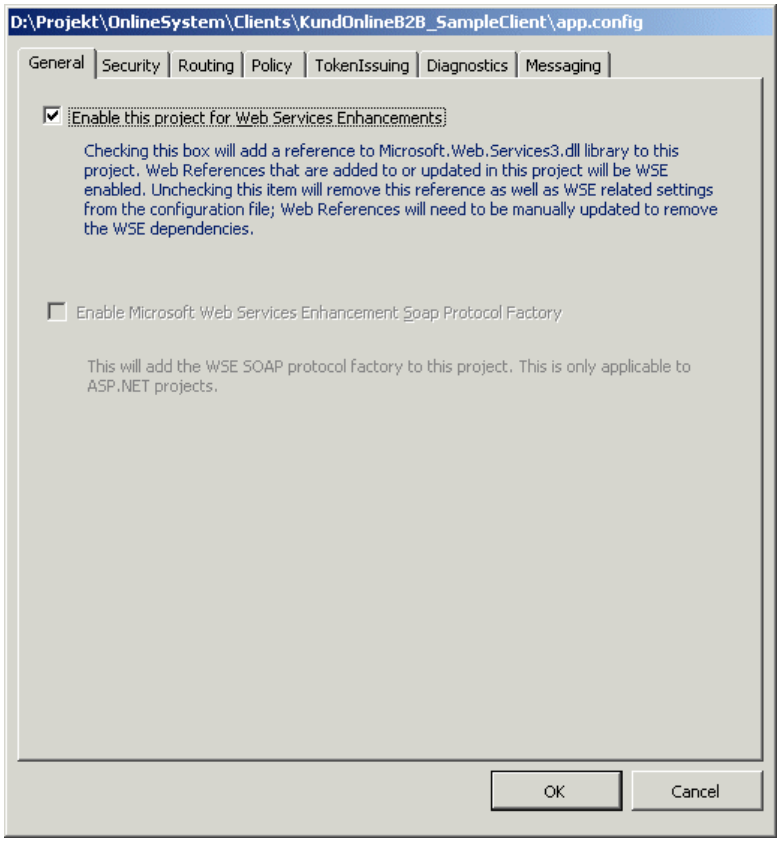


Image 3 - WSE settings 2.0 tool"

5.1.2 Defining the security policy

Next define the Policy to be used in all web method requests. Click the Policy tab and select "Enable Policy". Press the "Add" button and provide a name for the policy, i.e. "PayExConnectPolicy", and click OK. This brings up the "WSE Security Settings Wizard".

Select "Secure a client application". Select "Username" to support the UsernameToken model used by PayEx Connect, as shown in Image 4.

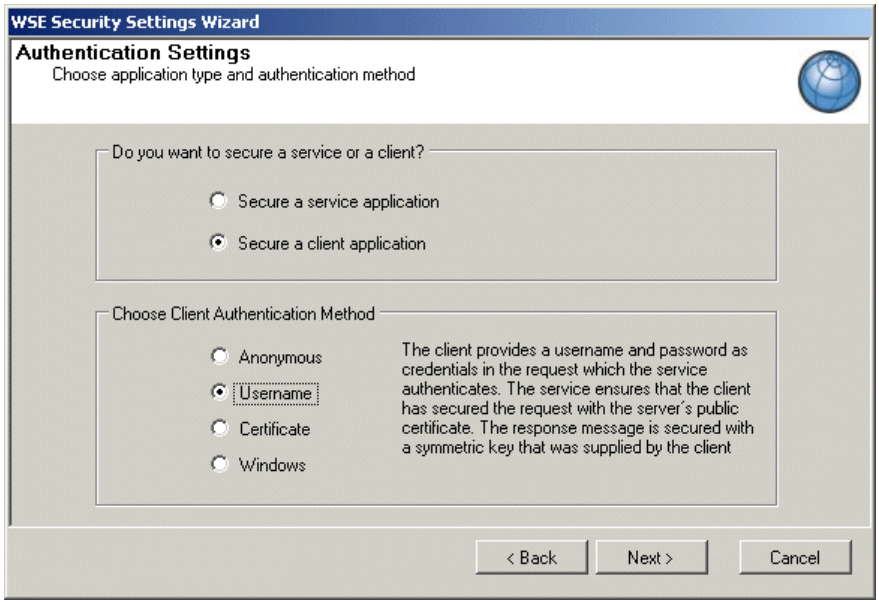


Image 4 - Security wizard, Authentication

Specify if the username and password should be managed from code our through the policy file. In this example we manage these properties through code, as shown in Image 5. Click Next.

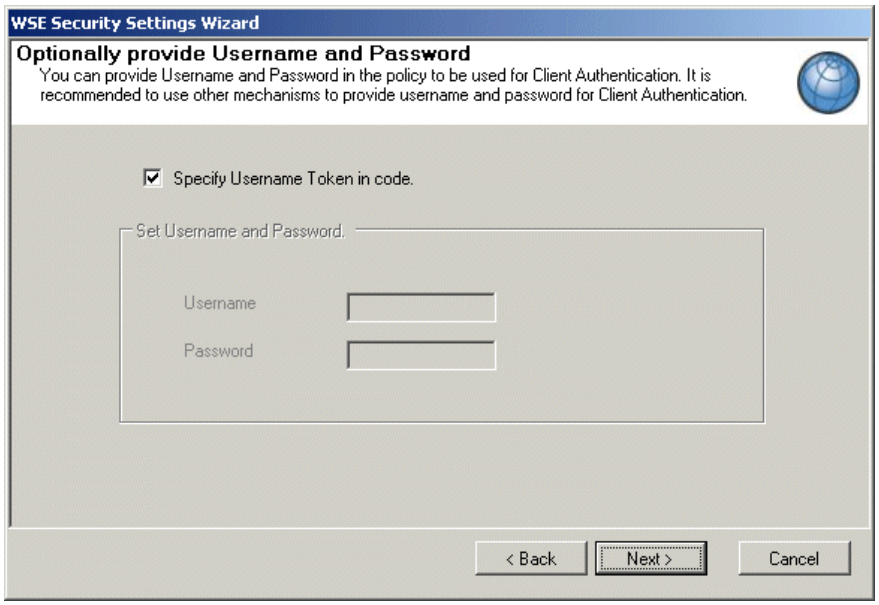


Image 5 - Security wizard, Username and password

Validate that WS-Security Extensions are enabled and select the "None (rely on transport protection" option, as shown in Image 6. Transport protection is managed through the SSL protocol. Thus there is no need for additional signing and encryption of SOAP messages between the client and PayEx Connect.

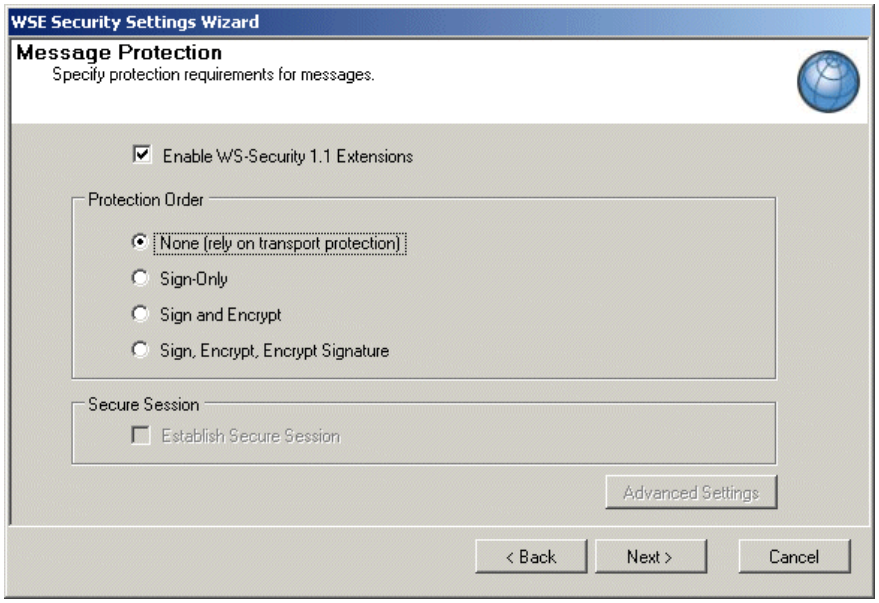


Image 6 - Security wizard, Message protection

After the wizard has completed, the policy will be listed in the "WSE settings 2.0 tool" as shown in Image 7. The wizard also adds the file wse3policyCache.config to your project in Visual Studio.

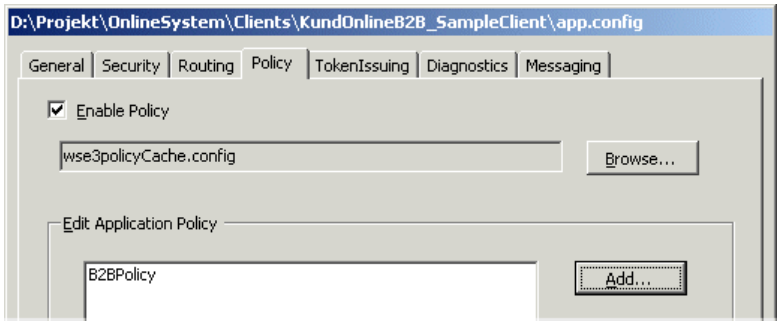


Image 7 - Policy listing

5.1.3 Adding the Web Reference

The Web Reference must be added after WSE support is enabled. Any Web References added before enabling WSE, will be missing the proxy classes that supports WSE.

Start the "Add Web Reference" wizard by right-clicking the project in Solution explorer and selecting "Add Web Reference.." from the popup menu.

Enter the URL to the sample Web service in PayEx Connect, https://connect.payex.com/SampleService.asmx. Specify a name that identifies the service, i.e. "SampleService", and click Add Reference.

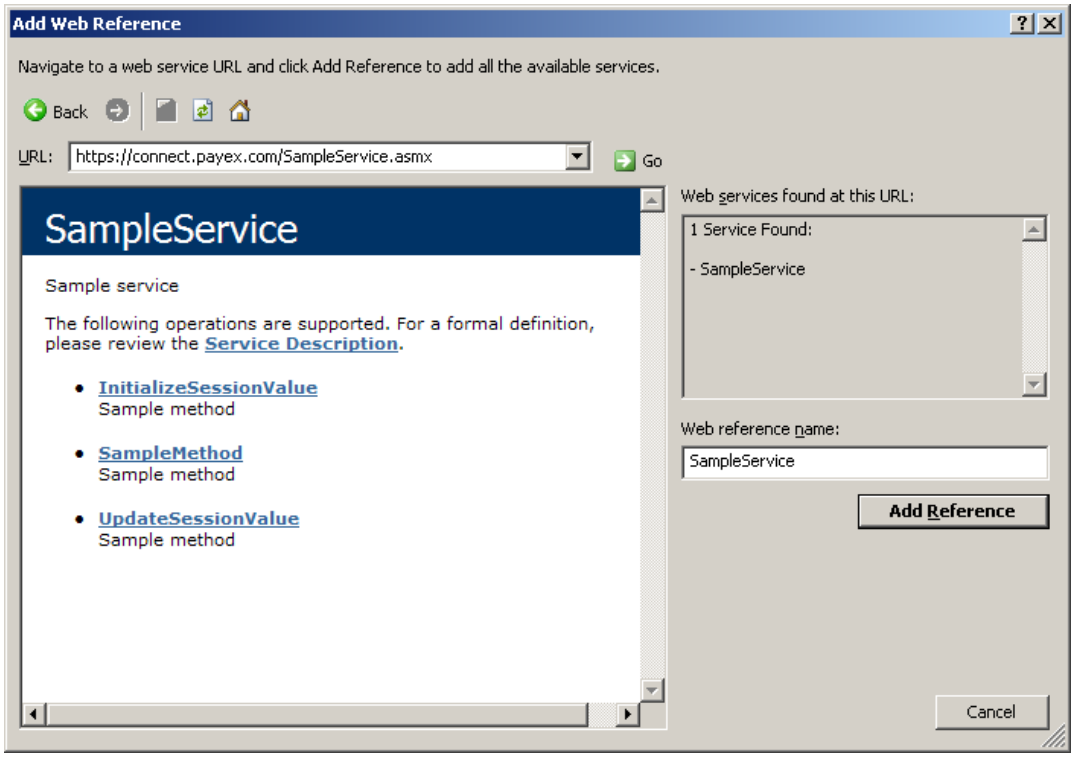


Image 8 - Add Web Reference

5.1.4 Calling the service

The final part of this sample is the code that prepares the request and invokes the web method. This code consists of four different parts: setting credentials, preparing and submitting the request, processing the response and error handling.

```
Imports Microsoft.Web.Services3.Design
Imports Microsoft.Web.Services3.Security.Tokens
Imports System.Web.Services.Protocols

Module MyModule

    Sub Main()

        Dim sampleProxy As New SampleService.SampleServiceWse
        Dim sampleRequest As New SampleService.SampleMethodRequestType
        Dim sampleResponse As New SampleService.SampleMethodResponseType
        Dim tokenProvider As New UsernameTokenProvider

        Try

            'Set client credentials
            tokenProvider.Username = "" 'Company number
            tokenProvider.Password = "" 'Password key
            sampleProxy.SetClientCredential(Of UsernameToken)(tokenProvider.GetToken())
            sampleProxy.SetPolicy("B2BPolicy")
            'Prepare and execute request
            sampleRequest.SampleText = "My text"
            sampleResponse = sampleProxy.SampleMethod(sampleRequest)
            'Process response
            Console.WriteLine("{0}", sampleResponse.SampleText)
            Console.WriteLine("Sample Amount: {0}", sampleResponse.SampleAmount)
            Console.WriteLine("Sample Date: {0}", sampleResponse.SampleDate)
            Console.WriteLine("Sample DateTime: {0}", sampleResponse.SampleDateTime)
            Console.Read()

        Catch ex As SoapException
            'Soap faults
            Console.WriteLine("Error code: {0}", ex.Detail.FirstChild.Attributes("ErrorCode").Value)
            Console.WriteLine("Description: {0}", ex.Detail.FirstChild.Attributes("Description").Value)
            Console.WriteLine("Guid: {0}", ex.Detail.FirstChild.Attributes("Guid").Value)
            Console.WriteLine("Message: {0}", ex.Message)
            Console.Read()
        Catch ex As Exception
            'Other errors
            Console.WriteLine("Error: {0}", ex.Message)
        Finally
            'Clean up
            sampleProxy.Dispose()
        End Try

    End Sub

End Module
```

Setting credentials

Create a new UsernameTokenProvider and set the username and password properties (The username represents the company number supplied by PayEx). Add the newly created token as the client credentials through the SetClientCredentials method. Refer to the policy name specified in 5.1.2 through the SetPolicy method.

Preparing and submitting the request

Create a new instance of the SampleMethodRequestType class and set the properties. Execute the web method and store the response in a SampleMethodResponseType object.

Processing the response

The response object encapsulates all the information supplied by the service request. The information can be accessed through properties and sub-objects of the response object.

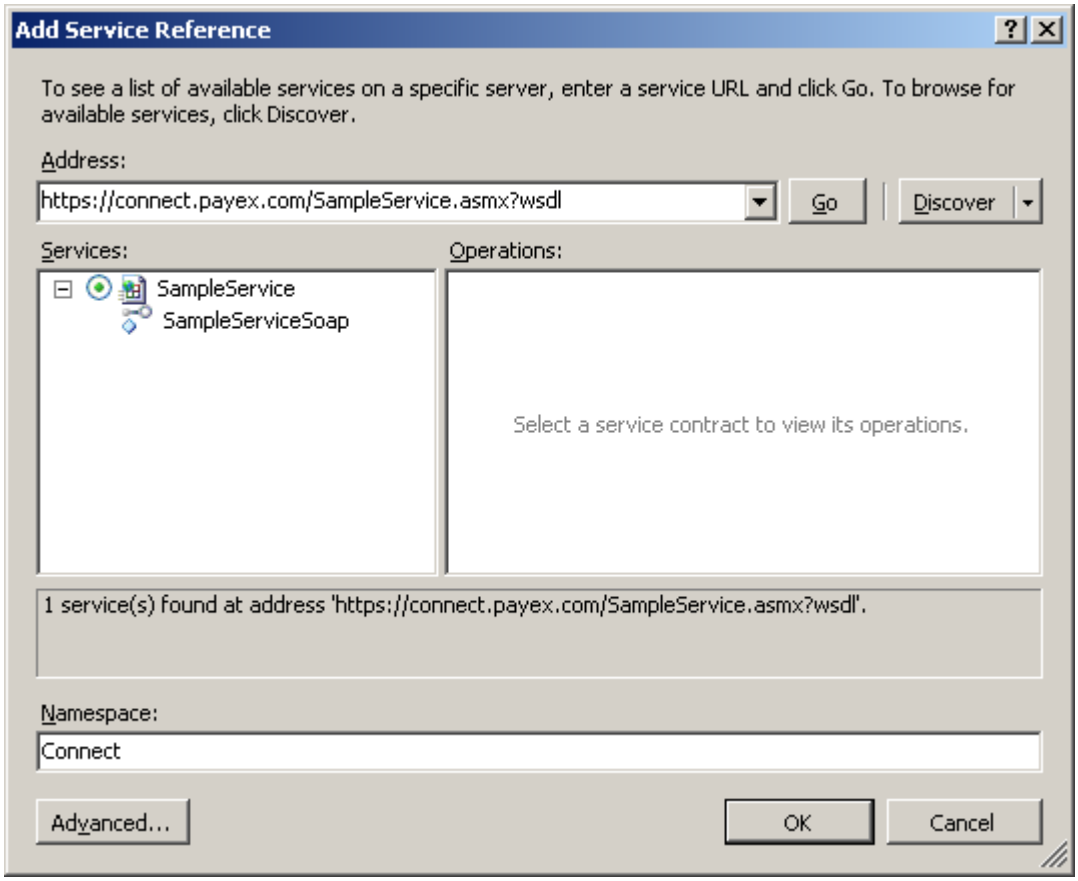
Error handling

Separate SoapExceptions from other error types when catching errors. A SoapException triggered by PayEx Connect contains additional information in the detail-element such as error code, description and unique id.

5.2 Microsoft .NET WCF and WSE

5.2.1 Adding the service reference

Right click on the project and select "Add Service Reference…"



Open app.config/web.config and change Secruity mode to TransportWithMessageCredential.

```
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="SampleServiceSoap" closeTimeout="00:01:00"
openTimeout="00:01:00"
            receiveTimeout="00:10:00" sendTimeout="00:01:00"
allowCookies="false"
            bypassProxyOnLocal="false"
hostNameComparisonMode="StrongWildcard"
            maxBufferSize="65536" maxBufferPoolSize="524288"
maxReceivedMessageSize="65536"
            messageEncoding="Text" textEncoding="utf-8"
transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
maxArrayLength="16384" maxBytesPerRead="4096"
maxNameTableCharCount="16384" />
          <security mode="TransportWithMessageCredential">
            <transport clientCredentialType="None"
proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName"
algorithmSuite="Default" />
```

```
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="https://connect.payex.com/SampleService.asmx"
          binding="basicHttpBinding"
bindingConfiguration="SampleServiceSoap"
          contract="Connect.SampleServiceSoap" name="SampleServiceSoap" />
    </client>
  </system.serviceModel>
```

5.2.2 Example application

```
using System;

namespace PayExConnectSample {
    class Program {
        static void Main(string[] args) {

            string username = "USERNAME";
            string password = "PASSWORD";

            Connect.SampleServiceSoapClient sampleProxy = new
Connect.SampleServiceSoapClient();
            sampleProxy.ClientCredentials.UserName.UserName = username;
            sampleProxy.ClientCredentials.UserName.Password = password;

            Connect.SampleMethodRequestType request = new
Connect.SampleMethodRequestType();
            Connect.SampleMethodResponseType response = null;

            request.SampleText = "Hello Connect!";

            try {
                response = sampleProxy.SampleMethod(request);
                Console.WriteLine(response.SampleText);
            } catch (Exception ex) {
                Console.Write(ex);
            }
        }
    }
}
```

5.3 Java 5 JWSDP 2.0

This sample uses Java Web Services Developer Pack 2.0 from Sun Microsystems.

5.3.1 Installing JWSDP

Sun Java Web Services Developer Pack enables web services for the Java5 platform. Installation packages can be obtained here:

http://java.sun.com/webservices/jwsdp/index.jsp

Download and install using default install options.

5.3.2 Project setup

Create a new project in your favorite IDE and add the following files to your classpath:

<JWSDP-HOME>/jaxb/lib/*.jar

<JWSDP-HOME>/jaxp/lib/*.jar

<JWSDP-HOME>/jwsdp-shared/lib/*.jar

<JWSDP-HOME>/sjsxp/lib/*.jar

<JWSDP-HOME>/saaj/lib/*.jar

<JWSDP-HOME>/xws-security/lib/*.jar

5.3.3 Security configuration

Create a new file named client-wss-config.xml and add the following content:

```
<xwss:JAXRPCSecurity xmlns:xwss="http://java.sun.com/xml/ns/xwss/config">

    <xwss:Service>
            <!--
              Dump all messages to stdout.
            -->
        <xwss:SecurityConfiguration dumpMessages="true">
            <!--
              Default: Digested password will not be sent.
            -->
            <xwss:UsernameToken name="1000" digestPassword="false" password="159X"/>
        </xwss:SecurityConfiguration>
    </xwss:Service>

    <xwss:SecurityEnvironmentHandler>
        se.faktab.b2b.SecurityEnvironmentHandler
    </xwss:SecurityEnvironmentHandler>

</xwss:JAXRPCSecurity>
```

A sample SecurityEnvironmentHandler can be found here:

<JWSDP-HOME>/xws-security/samples/jaxws2.0/simple/src/simple/client

5.3.4 Generate client proxies

JWSDP ships a utility called wsimport that generates client proxies from a wsdl-url.
Use the following command to generate proxies:

wsimport.bat –keep https://connect.payex.com/SampleService.asmx?wsdl

Proxies will be generated in the current directory. Make sure the generated proxies are added to project classpath.

5.3.5 Sample application

The following code snippet resembles a sample application calling sampleMethod:

```
//specify the security configuration file..
FileInputStream configFile = new FileInputStream( "/foo/bar/client-wss-config.xml" );
XWSSecurityConfiguration config =
        SecurityConfigurationFactory.newXWSSecurityConfiguration( configFile );

//create sample service port..
SampleServiceSoap port = new SampleService().getSampleServiceSoap();
//set security configuration..
((BindingProvider)port).getRequestContext().put(
        XWSSecurityConfiguration.MESSAGE_SECURITY_CONFIGURATION, config );

//call sampleMethod..
SampleMethodRequestType reqType = new ObjectFactory().createSampleMethodRequestType();
reqType.setSampleText( "Sample text" );
SampleMethodResponseType response = port.sampleMethod( reqType );
System.out.println( response.getSampleText() );
System.out.println( response.getSampleAmount() );
System.out.println( response.getSampleDate() );
System.out.println( response.getSampleDateTime() );
```

Notice that the sample code expects to find the security configuration file in /foo/bar/. Adjust to your own environment.

5.4 PHP

NuSOAP library can be found at http://sourceforge.net/projects/nusoap/

The following code will call the SampleMethod in the SampleService with the argument "Hello Connect!".

```
<?php
/* NuSOAP Library */
require 'lib/nusoap.php';

function GUID() {
    if (function_exists('com_create_guid') === true) {
        return trim(com_create_guid(), '{}');
    }
    return sprintf('%04X%04X-%04X-%04X-%04X-%04X%04X%04X', mt_rand(0,
65535), mt_rand(0, 65535), mt_rand(0, 65535), mt_rand(16384, 20479),
mt_rand(32768, 49151), mt_rand(0, 65535), mt_rand(0, 65535), mt_rand(0,
65535));
}

$username = 'USERNAME';
$password = 'PASSWORD';

$currentTime = time();
$secondsToExpire = 300;
$created = gmdate('Y-m-d\TH:i:s\Z', $currentTime);
$expires = gmdate('Y-m-d\TH:i:s\Z', $currentTime + $secondsToExpire);

$header = '
<wsa:Action>http://faktab.se/B2B/SampleService/SampleMethod</wsa:Action>
<wsa:MessageID>urn:uuid:' . GUID() . '</wsa:MessageID>
<wsa:ReplyTo>

<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymo
us</wsa:Address>
</wsa:ReplyTo>
<wsa:To>https://connect.payex.com/SampleService.asmx</wsa:To>
<wsse:Security SOAP-ENV:mustUnderstand="1">
  <wsu:Timestamp wsu:Id="Timestamp-' . GUID() . '">
      <wsu:Created>' . $created . '</wsu:Created>
      <wsu:Expires>' . $expires . '</wsu:Expires>
  </wsu:Timestamp>
  <wsse:UsernameToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="SecurityToken-' . GUID() . '">
    <wsse:Username>' . $username . '</wsse:Username>
    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">' . $password .
'</wsse:Password>
    <wsse:Nonce>' . base64_encode(GUID()) . '</wsse:Nonce>
    <wsu:Created>' . $created . '</wsu:Created>
  </wsse:UsernameToken>
</wsse:Security>
';

$body = '
<SampleMethod xmlns="http://faktab.se/B2B/SampleService">
  <req>
    <SampleText>Hello Connect!</SampleText>
  </req>
</SampleMethod>
';

/* Connect to the SOAP Server */
```

```
$client = new
nusoap_client('https://connect.payex.com/SampleService.asmx?wsdl',
'WSDL');

/* Set encoding */
$client->soap_defencoding = 'utf-8';

/* namespaces */
$client->namespaces['wsa'] =
'http://schemas.xmlsoap.org/ws/2004/08/addressing';
$client->namespaces['wsse'] = 'http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd';
$client->namespaces['wsu'] = 'http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd';

/* Check for error */
$error = $client->getError();
if($error) { die('<h2>Constructor Error</h2><pre>' . $error . '</pre>'); }

/* Set header */
$client->setHeaders($header);

/* Make the call */
$result = $client->call('SampleMethod', $body);

/* Check for fault/error */
if($client->fault) { print_r('<h2>Fault</h2><pre>' . $client->fault .
'</pre>'); }
if($client->getError()) { print_r('<h2>Error</h2><pre>' . $client-
>getError() . print_r($result) . '</pre>'); }

/* Print result */
print_r($result);

echo '<h1>Request:</h1>';
echo '<div><pre>' . htmlspecialchars($client->request, ENT_QUOTES) .
'</pre></div>';
echo '<h1>Response:</h1>';
echo '<div><pre>' . htmlspecialchars($client->response, ENT_QUOTES) .
'</pre></div>';
echo '<h1>Debug:</h1>';
echo '<div><pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) .
'</pre></div>';
?>
```