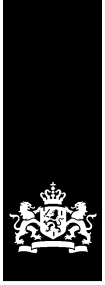


Logius
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Creating a WUS connection with test stubs using VB.NET or C#

Version	1.0
Date	January, 13th 2014
Status	Final

Colophon

Project name	Creating a WUS connection with test stubs available on the Aansluit Suite Digipoort using VB.NET or C#
Version	1.0
Organization	Logius Postbus 96810 2509 JE Den Haag Nederland servicecentrum@logius.nl
Annexes	None

Disclaimer

The code published in this document are limited examples and not completely tested. Therefore we cannot give any guarantee regarding the correctness and completeness of the code.

Logius cannot be held responsible for any damage arising from the use of this publication.

Without any warning Logius can change the content of this publication.

No rights can be derived from this publication.

Content

Summary

This document explains with non-normative code how a connection can be established with the test stubs available at the Aansluit Suite Digipoort. Aansluit Suite Digipoort provides test stubs for the so-called 'AanleverService' (for the delivery of XBRL filings) and 'StatusinformatieService' (for requesting status information regarding the XBRL filings). Both services should be implemented for a proper operation of submitting XBRL filings.

The non-normative code used is constructed with VB.NET and C# (supported by Microsoft Visual Studio). The programming languages are based on the .NET Framework 4.5.

Please notice: the code published in this document is not complete. Not all possible scenarios are covered with the code. For example, dealing with a SOAP Fault is not being described in this document.

Prerequisites

In order to develop a connection with the available test stubs at the Aansluit Suite Digipoort preparatory activities must be done. In short:

1. Knowledge of programming
2. Knowledge of web services
3. Get access to the Aansluit Suite Digipoort (located at http://aansluiten.procesinfrastructuur.nl/)
4. Obtain a test certificate
5. Read the necessary documentation available at the Aansluit Suite Digipoort
6. Obtain the correct certificate of Aansluit Suite Digipoort and WSDL's for the different services

Information regarding creating XBRL filings can be found here: http://www.sbr-nl.nl/fileadmin/SBR/documenten/Technical_Starters_Guide_English_march_2013.pdf

Development steps

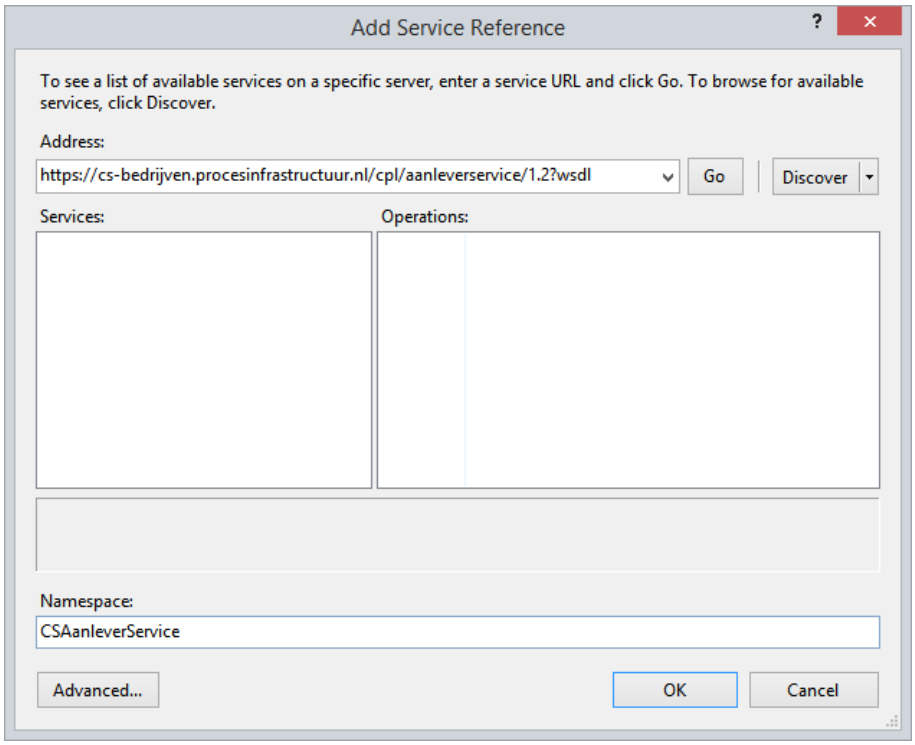
Step 1 Creating a service reference

A service reference can be automated generated by a development environment through importing a WSDL. A WSDL is a XML-based file which describes the functionality of a web service. A WSDL can be found at the endpoint of a specific web service by adding *?wsdl* at the end of the endpoint. Please notice: for retrieving WSDL's via Internet of Digipoort and the Aansluit Suite Digipoort, a correct (test)certificate[1] must be installed in the web browser.

The test stub endpoints for AanleverService (1) and StatusinformatieService (2) are:

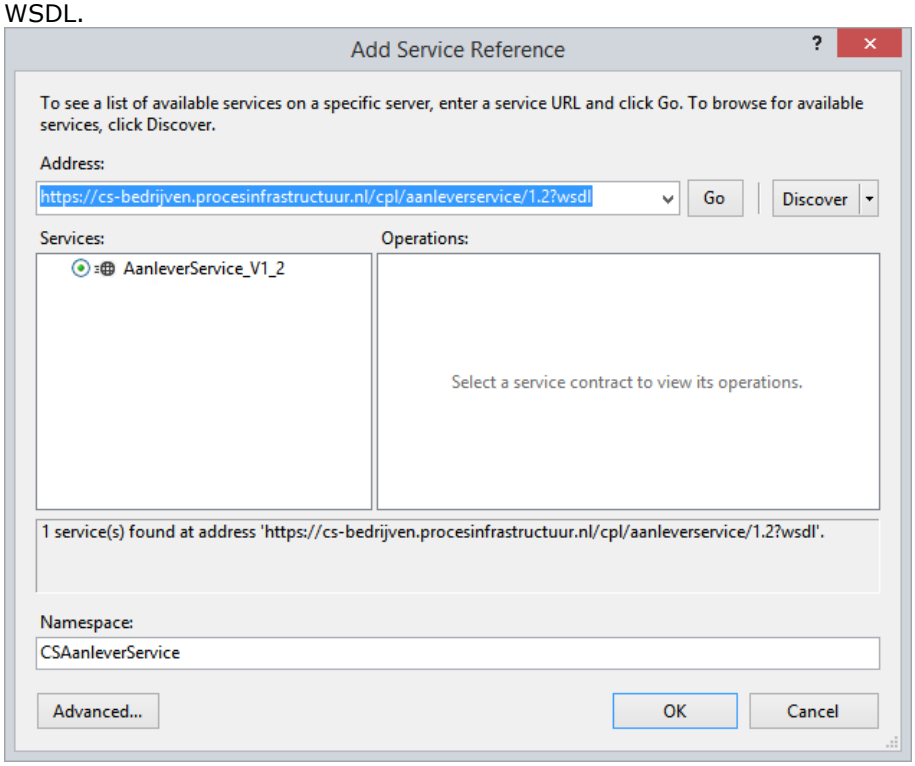
1. https://cs-bedrijven.procesinfrastructuur.nl/cpl/aanleverservice/1.2
2. https://cs-bedrijven.procesinfrastructuur.nl/cpl/statusinformatieservice/1.2

Within Microsoft Visual Studio a service reference can be added to a development project by selecting 'Project – Add Service Reference' from the menu bar.
The endpoint with addition of the suffix ?wsdl must be used as the address for specifying the location of the WSDL.



1 A test certificate can be obtained by completing the registration on the Aansluit Suite Digipoort.

After clicking on "Go" the available services are being retrieved from the WSDL.



After clicking on "OK" a namespace (in this case named as CSAanleverService) is added to the project.

Following the generating of the CSAanleverService namespace an modification must be made to the Public Interface AanleverService_V1_2. The exact location for the modification is in the service contract interface[2] This can be found by searching for the System.ServiceModel.ServiceContractAttribute attribute. An additional property and value (ProtectionLevel:=System.Net.Security.ProtectionLevel.Sign)must be added to the attribute.

```
<System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel",
"4.0.0.0"),
System.ServiceModel.ServiceContractAttribute([Namespace]:=
"http://logius.nl/digipoort/wus/2.0/aanleverservice/1.2/",
ConfigurationName:="CSAanleverService.AanleverService_V1_2",
ProtectionLevel:=System.Net.Security.ProtectionLevel.Sign)>
Public Interface AanleverService_V1_2
```

Table 1 VB.NET - After modification

```
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel",
"4.0.0.0")]
[System.ServiceModel.ServiceContractAttribute(Namespace=
"http://logius.nl/digipoort/wus/2.0/aanleverservice/1.2/",
ConfigurationName=" CSAanleverService.AanleverService_V1_2",
ProtectionLevel=System.Net.Security.ProtectionLevel.Sign)]
public interface AanleverService_V1_2
```

Table 2 C# - After modification

2 Additional information can found at http://msdn.microsoft.com/en-us/library/ms733881(v=vs.110).aspx

This modification ensures signing of specific elements defined by the WSDL.

Finally the namespace must be added to the General Declarations part of the project, with Imports (VB.NET) or using (C#)

Step 2 Creation of the delivery request

The delivery request, which is based on the WSDL, defines the content for the SOAP-body. It contains the data for the correct delivery of the XBRL filings. The delivery request is the parameter for the main function in the public interface.

```
Function aanleveren(ByVal request As
CSAanleverService.aanleverenRequest) As
CSAanleverService.aanleverenResponse
```

Table 3 VB.NET - Main function in the service reference

```
CSAanleverService.aanleverenResponse
aanleveren(CSAanleverService.aanleverenRequest request);
```

Table 4 C# - Main function in the service reference

The content of the request depends primarily on the purpose of the communication. In this document we focus on the XBRL filings. An XBRL filing in the delivery request must be base64 encoded. The actual base64 encoding is being done by the generated service reference. A condition for doing this, is that the XBRL filing must be converted to an array of bites. A delivery request (aanleverenRequest) consists of the following elements:

1. `aanleverkenmerk`, a unique message identifier of the sender (for the test stubs a value of a predefined list must be used to specify the type of test)
2. `berichtsoort`, the message type (for example *Omzetbelasting*)
3. `identiteitBelanghebbende.nummer`, a specific identifier of the accountable entity (equal to the first xbrli:identifier value in the dispatched XBRL instance)
4. `identiteitBelanghebbende.type`, identifying type of identifier (for example *BTW*)
5. `rolBelanghebbende`, explains the role of the sender (for example *Intermediair*)
6. `berichtInhoud.bestandsnaam`, the file name of the XBRL instance
7. `berichtInhoud.inhoud`, the content of the XBRL instance converted to an array of bites
8. `berichtInhoud.mimeType`, an indication of the type of data that the XBRL filing contains (has a mandatory value of `"application/xml"`)
9. `autorisatieAdres`, address of external authorisation provider. For XBRL filings an external authorisation provider is not allowed, for this the value `"http://geenausp.nl"` must be used. This value explains that no external authorisation provider is being used.

Please note: the sender and the accountable entity can be two different entities. An intermediary (sender) files a XBRL filing on behalf of his client (accountable entity).

In the examples below for creating a delivery request and request for status information services, we assume the following test scenario: a successful filing of a VAT message by an intermediary. For this the aanleverkenmerk of the aanleverenRequest has a value of "Happyflow". Example XBRL instance files can be found at http://www.sbr-nl.nl/werken-met-sbr/software-leveranciers/nederlandse-taxonomie/2014/.

```
Shared request As aanleverenRequest

Private Shared Function createRequest() As aanleverenRequest

        ' The input String.
        Dim value As String =
My.Computer.FileSystem.ReadAllText(location + filename
of the XBRL filing)

        ' Convert String to Byte array.
        Dim array() As Byte =
System.Text.Encoding.ASCII.GetBytes(value)

        ' Create request
        Dim request As New aanleverenRequest() With { _
          .aanleverRequest = New aanleverRequest() With { _
              .aanleverkenmerk = "Happyflow", _
              .berichtsoort = "Omzetbelasting", _
              .identiteitBelanghebbende = New
Aanleverservice.identiteitType() With { _
              .nummer = "001000044B37", _
              .type = "BTW" _
             }, _
              .rolBelanghebbende = "Intermediair", _
              .berichtInhoud = New berichtInhoudType() With { _
              .bestandsnaam = (filename of the XBRL filing), _
              .inhoud = array, _
              .mimeType = "application/xml" _
             }, _
              .autorisatieAdres = "http://geenausp.nl"
         } _
     }
        Return request

End Function
```

Table 5 VB.NET – Sample code for delivery request

```
private static aanleverenRequest createRequest()
        {

            // The input String.
            string value =(location + filename
of the XBRL filing);

            // Convert String to Byte array.
            byte[] array =
System.Text.Encoding.ASCII.GetBytes(value);

            // Create request
            aanleverenRequest request = new aanleverenRequest
            {
                aanleverRequest = new aanleverRequest
                {
                    aanleverkenmerk = "Happyflow",
                    berichtsoort = "Omzetbelasting",
                    identiteitBelanghebbende = new
Aanleverservice.identiteitType
                    {
                        nummer = "001000044B37",
                        type = "BTW"
                    },
                    rolBelanghebbende = "Intermediair",
                    berichtInhoud = new berichtInhoudType
                    {
                        bestandsnaam = (filename of the XBRL
filing),
                        inhoud = array,
                        mimeType = "application/xml"
                    },
                    autorisatieAdres = "http://geenausp.nl"
                }
            };
            return request;

        }
```

Table 6 C# – Sample code for delivery request

Step 3 Submitting delivery request to Aansluit Suite Digipoort (via test stub)

To create a channel for sending the aanleverenrequest to the endpoint of the Aansluit Suite Digipoort primarily bindings for message security, encoding and transport needs to be created. Certificates for client and server (endpoint) are added to the channel. Based on the public interface defined by the service reference a connection with the endpoint is being made and the aanleverenRequest is being submitted.

```
Private Sub subSubmit()
        ‘Create binding element for security
        Dim secBE = DirectCast(SecurityBindingElement.
CreateMutualCertificateBindingElement(MessageSecurityVersion.
WSSecurity10WSTrust13WSSecureConversation13WSSecurityPolicy12BasicSecurityProfile10),
AsymmetricSecurityBindingElement)
        secBE.MessageSecurityVersion = MessageSecurityVersion.
WSSecurity10WSTrust13WSSecureConversation13WSSecurityPolicy12BasicSecurityProfile10
        secBE.EnableUnsecuredResponse = True
        secBE.MessageProtectionOrder = MessageProtectionOrder.EncryptBeforeSign
        secBE.IncludeTimestamp = True
        secBE.DefaultAlgorithmSuite = SecurityAlgorithmSuite.TripleDesRsa15
        'Explicit accept secured answers from endpoint
        secBE.AllowSerializedSigningTokenOnReply = True

        ‘Create binding element for encoding
        Dim textBE = New
TextMessageEncodingBindingElement(MessageVersion.Soap11WSAddressing10, Encoding.UTF8)

        ‘Create binding element for transport
        Dim httpsBE = New HttpsTransportBindingElement()
        httpsBE.RequireClientCertificate = True
        httpsBE.AuthenticationScheme = System.Net.AuthenticationSchemes.Anonymous

        Dim cBinding As New CustomBinding()

        cBinding.Elements.Add(secBE)
        cBinding.Elements.Add(textBE)
        cBinding.Elements.Add(httpsBE)

        Dim address As New EndpointAddress("https://cs-
bedrijven.procesinfrastructuur.nl/cpl/aanleverservice/1.2")

        Dim factory As New ChannelFactory(Of AanleverService_V1_2)(cBinding, address)

        Dim certClient As New X509Certificate2(location + filename client
certificate, password)

        Dim certService As New X509Certificate2(location + filename server
certificate)

        'Explicit prevent check on chain of trust.
factory.Credentials.ServiceCertificate.Authentication.CertificateValidationMode =
X509CertificateValidationMode.None

        factory.Credentials.ClientCertificate.Certificate = certClient
        factory.Credentials.ServiceCertificate.DefaultCertificate = certService

        Dim client As AanleverService_V1_2 = factory.CreateChannel()

        Dim result As aanleverenResponse = client.aanleveren(createRequest())

    End Sub
```

Table 7 VB.NET - Submitting aanleverenrequest

```
private void subSubmit
        {
            //Create binding element for security
            dynamic secBE = (AsymmetricSecurityBindingElement)SecurityBindingElement.
CreateMutualCertificateBindingElement(MessageSecurityVersion.
WSSecurity10WSTrust13WSSecureConversation13WSSecurityPolicy12BasicSecurityProfile10);
            secBE.MessageSecurityVersion = MessageSecurityVersion.
WSSecurity10WSTrust13WSSecureConversation13WSSecurityPolicy12BasicSecurityProfile10;
            secBE.EnableUnsecuredResponse = true;
            secBE.MessageProtectionOrder = MessageProtectionOrder.EncryptBeforeSign;
            secBE.IncludeTimestamp = true;
            secBE.DefaultAlgorithmSuite = SecurityAlgorithmSuite.TripleDesRsa15;

            //Explicit accept secured answers from endpoint
            secBE.AllowSerializedSigningTokenOnReply = true;

            //Create binding element for encoding
            dynamic textBE = new
TextMessageEncodingBindingElement(MessageVersion.Soap11WSAddressing10,
Encoding.UTF8);

            //Create binding element for transport
            dynamic httpsBE = new HttpsTransportBindingElement();
            httpsBE.RequireClientCertificate = true;
            httpsBE.AuthenticationScheme =
System.Net.AuthenticationSchemes.Anonymous;

            CustomBinding cbinding = new CustomBinding();

            cbinding.Elements.Add(secBE);
            cbinding.Elements.Add(textBE);
            cbinding.Elements.Add(httpsBE);

            EndpointAddress address = new EndpointAddress
("https://cs-bedrijven.procesinfrastructuur.nl/cpl/aanleverservice/1.2");

            ChannelFactory<AanleverService_V1_2> factory = new
ChannelFactory<AanleverService_V1_2>(cbinding, address);

            X509Certificate2 certClient = new X509Certificate2
(location + filename client certificate, password);

            X509Certificate2 certService = new X509Certificate2
(location + filename server certificate)

            //Explicit prevent check on chain of trust
            factory.Credentials.ServiceCertificate.Authentication.
CertificateValidationMode = X509CertificateValidationMode.None;

            factory.Credentials.ClientCertificate.Certificate = certClient;
            factory.Credentials.ServiceCertificate.DefaultCertificate = certService;

            AanleverService_V1_2 client = factory.CreateChannel();

            aanleverenResponse result = client.aanleveren(createRequest());
}
```

Table 8 C# - Submitting aanleverenrequest

Step 4 Processing of the delivery response

If a filing is successful the result of the aanleverenRequest is an aanleverenResponse (see table 3 for VB.NET and table 4 for C#). The aanleverenResponse consists of two properties, the `kenmerk` and `tijdstempelAangeleverd`.

The kenmerk is an unique identifier returned by the test stub of the Aansluit Suite Digipoort (normally the production environment of Digipoort) and is being represented by a globally unique identifier (GUID).

The `tijdstempelAangeleverd` is the timestamp of acceptance by the Aansluit Suite Digipoort. The format of `tijdstempelAangeleverd` is a standard XML Schema Definition dateTime representation.

Step 5 Requesting for status information regarding the XBRL filings

In the complete process of submitting filings it is important to determine if the submission is successful. A submission is successful if the processing gateway (Digipoort) and the authority responsible for receiving and processing the XBRL filing, confirms that the filing complies to the rules of the gateway and the authority. If this is the case then the filing will have positive intermediate states from the gateway and an end status 500 (XBRL filing has been accepted).

If the filing does not comply, it will get an negative intermediate state of the gateway or a final end status 510[3] (in both cases the XBRL filing has not been accepted).

Status information can be retrieved with different processes belonging to the StatusinformatieService. For the StatusinformatieService a separate WDSL is available. The service reference is created the same way as the AanleverService. In this case the modification in the service contract interface must be made to the Public Interface StatusinformatieService_V1_2.

One of the requests belonging to the StatusinformatieService is the getStatussenProcesRequest1 With the kenmerk from the aanleverenResponse the getStatussenProces collects all the intermediate states and end state for a specific filing.

A `getStatussenProcesRequest1` consists of the following elements:

1. `kenmerk`, this is the kenmerk (GUID) retrieved from the aanleverenResponse
2. `autorisatieAdres`, address of external authorisation provider. For XBRL filings an external authorisation provider is not allowed, for this the value `"http://geenausp.nl"` must be used. This value explains that no external authorisation provider is being used.

3 The Centraal Bureau van de Statistiek (CBS) does not return a final end status 510

```
Shared getStatus As getStatussenProcesRequest1

Private Shared Function creategetNieuweStatus() As
getStatussenProcesRequest1

        Dim getStatus As New getStatussenProcesRequest1() With {
_
        .getStatussenProcesRequest = New
getStatussenProcesRequest() With { _
          .kenmerk = result.aanleverResponse.kenmerk,
          .autorisatieAdres = "http://geenausp.nl"}}

        Return getStatus

End Function
```

Table 9 VB.NET - Sample code for status information request

Step 6 Submitting status request to Aansluit Suite Digipoort (via test stub)

Submitting a status request (for example getStatussenProcesRequest1) is done the same way as an aanleverenrequest. The only difference is the address of the endpoint, the connection (based on the public interface of service reference of the StatusinformatieService_V1_2 and the request being submitted.

```
Dim address As New
EndpointAddress("https://preprod.procesinfrastructuur.nl/wus/2.0/
statusinformatieservice/1.2")
```

Table 10 VB.NET – Endpoint for StatusinformatieService

```
Dim client As StatusinformatieService_V1_2 =
factory.CreateChannel()
```

Table 11 VB.NET - Public interface for StatusinformatieService

```
Function getStatussenProces(ByVal request As
CSStatusInformatieService.getStatussenProcesRequest1) As
CSStatusInformatieService.getStatussenProcesResponse1
```

Table 12 VB.NET – Main function from service reference StatusinformatieService_V1_2

Step 7 Processing of the delivery response

If a request for status information is successful a getStatussenProcesResponse1 is being returned. A getStatussenProcesResponse1 is an array of getStatussenProcesReturn's. Each getStatussenProcesReturn consist of the following properties (defined by the service reference):

1. `identiteitBelanghebbende.nummer`, , a specific identifier of the accountable entity (equal to the first xbrli:identifier value in the dispatched XBRL instance)
2. `identiteitBelanghebbende.type`, identifying type of identifier (for example BTW)

3. kenmerk, the GUID (originating from the aanleverenResponse and used by the getStatussenProcesRequest1
4. statuscode; contains a specific code for each intermediate and final state. The final statuscode of a 'Happyflow' test scenario is always 500. This means that the receiving authority accepted the XBRL filing.
5. statusdetails; if delivery is successful this contains a XML-file from the receiving authority with further details regarding the acknowledgement.
6. statusFoutcode, contains a description of a raised error.
7. statusomschrijving, a description of the statuscode. In case of status 500:" Validatie bij uitvragende partij gelukt.".
8. tijdstempelStatus, the timestamp of each status, based on the standard XML Schema Definition dateTime representation.

Appendix A: Needed imports in the General Declarations section

Imports / using WUS.Aanleverservice (=defined by service reference)
Imports / using WUS.Statusinformatieservices (=defined by service reference)
Imports / using System.Security.Cryptography.X509Certificates
Imports / using System.ServiceModel
Imports / using System.ServiceModel.Channels
Imports / using System.ServiceModel.Security
Imports / using System.Text