

Het Information Bridge Framework 1.0

Titel: Het Information Bridge Framework 1.0

Subtitel: De brug van MS Office 2003 naar Line-Of-Business (LOB) toepassingen

Auteurs: Wim Uyttersprot, Patrick Tisseghem, Gert Servranckx (U2U)

Traditioneel is er een diepe kloof tussen de Line-Of-Business (LOB) systemen waarin operationele gegevens worden beheerd en de desktopomgeving met Microsoft Office die gebruikt wordt door de informatiewerkers. Beide omgevingen hebben een gezamenlijk doel, met name het ondersteunen van de bedrijfsprocessen, maar hun invalshoek is verschillend. Het Microsoft Information Bridge Framework (IBF) laat ons toe om beide werelden te verbinden. IBF is een nieuw framework speciaal ontwikkeld voor de informatiewerker die gebruik maakt van een of meerdere Line-Of-Business-toepassingen maar gelijktijdig met Microsoft Office 2003 wil werken. IBF is vooral inzetbaar bij organisaties waar er een nood is om informatiewerkers meer productief te maken, door bedrijfstoepassingen en -processen toegankelijk te maken vanuit Microsoft Office smart-clients.

Situering van IBF

Hoe worden vandaag de bedrijfsgegevens die binnenkomen via emails op een efficiënte manier verwerkt? Velen zullen antwoorden: door manueel de data uit de mails te kopiëren naar de bedrijfstoepassingen. Hoe worden vandaag operationele gegevens toegevoegd aan Excel en Word-documenten? Al te dikwijls luidt het antwoord: manueel.

Het Microsoft Information Bridge Framework (IBF) legt een spreekwoordelijke brug tussen MS Office 2003 en de verschillende bedrijfstoepassingen die gebruikt worden door de informatiewerker. Microsoft heeft IBF ontworpen volgens een uitbreidbare, bedrijfs-schaalbare architectuur waarop een nieuw soort toepassing kan ontwikkeld worden dat LOB-systemen of een efficiënte wijze verbindt met de Office ervaring.

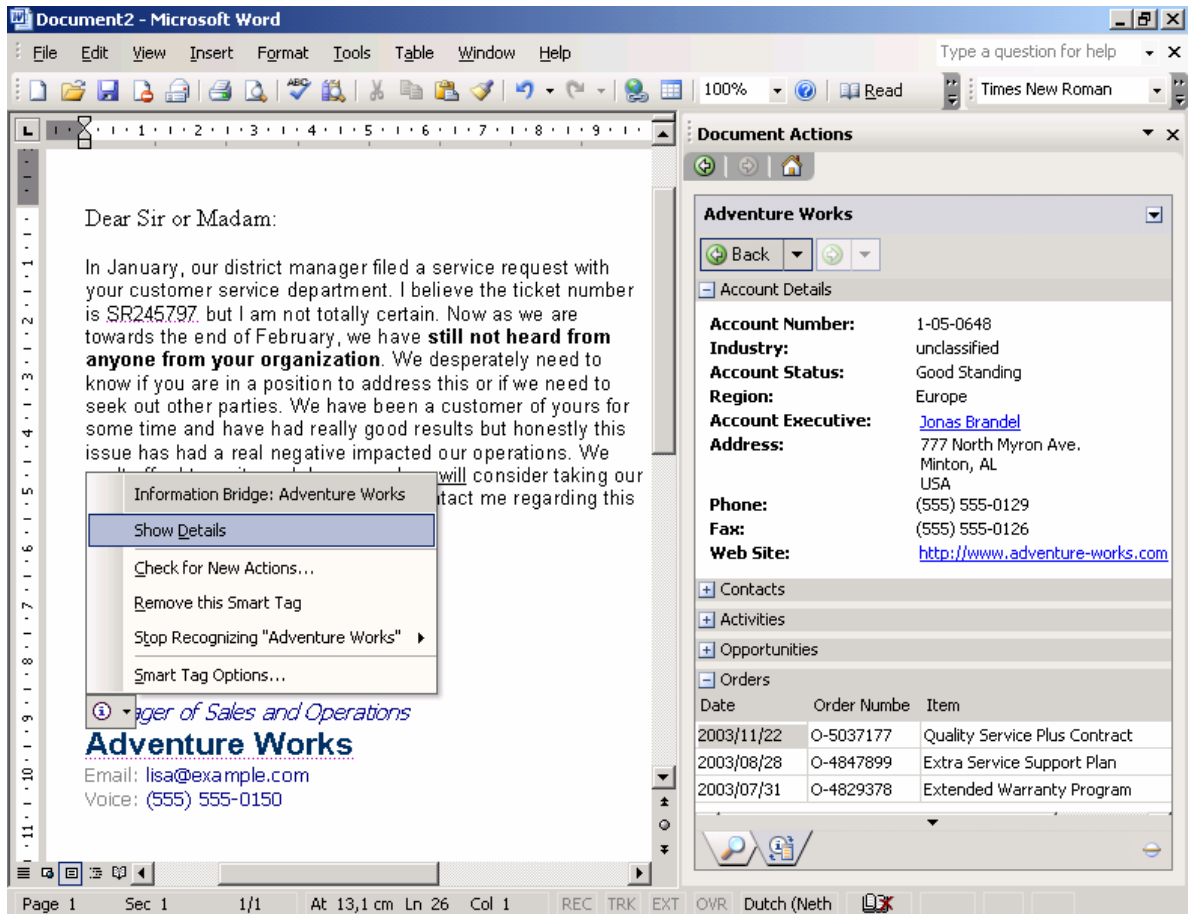
IBF laat toe om bedrijfsgegevens uit diverse LOB-systemen te consulteren, te analyseren en aan te passen vanuit Microsoft Office smart-clients. M.a.w. IBF zorgt ervoor dat business-informatie en de bijhorende methodes kunnen gedefinieerd blijven in een enterprise omgeving, en door webservices aangeboden worden binnen de context van een Office-document. Informatiewerkers zijn op deze manier in staat hun bedrijfsgegevens efficiënt en centraal te beheren volgens procedures die vastgelegd zijn binnen bedrijfstoepassingen.

IBF versie 1.0 werkt met Word 2003, Excel 2003 en Outlook 2003. In versie 1.1 zullen bijkomende metadata-designers toegevoegd worden, en zal er support zijn voor InfoPath 2003. In latere versies wordt support verwacht voor Microsoft Windows SharePoint Services en uiteindelijk zal IBF integraal deel uitmaken van het nieuwe desktop besturingssysteem, code naam 'Longhorn.'

IBF biedt een gestandaardiseerde aanpak aan die gestoeld is op het definiëren van metadata. Developers van LOB-toepassingen ontwikkelen met name business-objecten en bieden de diensten geleverd door deze objecten aan door middel van webservices. De solution-developers definiëren dan welke webservices door de smart-client worden geconsumeerd. Ze combineren de business-objecten en associëren deze met de gebruikersinterface van de Microsoft Office omgeving. Informatiewerkers gebruiken tenslotte de business-objecten vanuit de context van hun Office documenten en doen hierbij beroep op smart-tags, documenten die verrijkt zijn met XSD schema's en een nieuwe takenpaneel (task pane).

Een typisch business-scenario

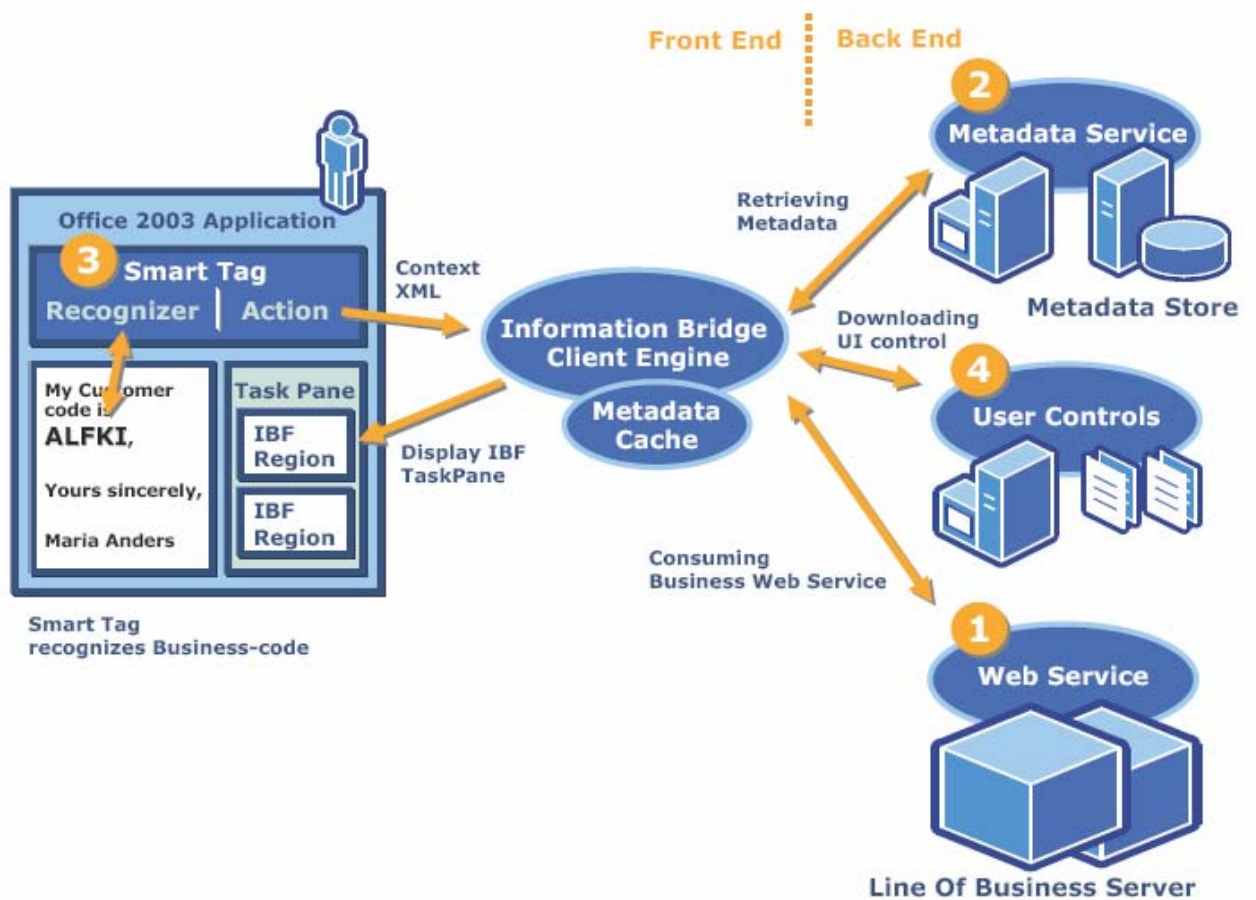
De kracht van IBF komt goed tevoorschijn in een typisch business-scenario: een informatiewerker wenst de nodige business-gegevens te consulteren betreffende een klant waarvan de firma naam voorkomt in een Word 2003-document. In figuur 1 wordt de firma naam "Adventure Works" herkend door middel van een smart-tag waardoor de informatiewerker m.b.v. IBF de business-gegevens en services kan opvragen in de taskpane.



Figuur 1: Typische business-scenario voor het Information Bridge Framework: via een smart-tag in een Word 2003-document legt IBF een link naar de Line-Of-Business applicatie die de corresponderende business-gegevens en -services aanbiedt in de taskpane.

Een IBF-toepassing ontwikkelen

Een typische IBF-toepassing ziet er vanuit het standpunt van de informatiewerker niet anders uit dan elk ander Word-, Excel- of Outlook-document dat interageert met de taskpane. Maar inwendig bestaat de IBF-toepassing uit een aantal onderdelen op de desktop, een metadata-service aangeboden door de IBF-server, en de webservices als facades voor de een of meerdere Line-Of-Business-applicaties (zie figuur 2).



Figuur 2: De onderdelen van een IBF-toepassing zijn de LOB-webservice, de Metadata-service, de smart-tag en de UI-control.

Op de desktop vereist IBF als systeem een Microsoft Office 2003 Professional, Microsoft .NET Framework 1.1 en Windows 2000 Professional of hoger. De front-end onderdelen draaien binnen de Microsoft Office 2003-applicaties en bieden er de data aan die ze uitwisselen met de webservices op de server. De onderdelen zijn: de smart tag componenten, de gebruikersinterface-componenten voor o.a. de taskpane en de Information Bridge client-engine. Deze client-engine haalt metadata op van de IBF-server, houdt deze bij in de cache op de client, en biedt geïnterpreteerde metadata aan in de Office-omgeving onder de vorm van context-afhankelijke navigatie, menus, business-objecten en andere informatie-elementen.

Langs de serverzijde, draait de IBF metadata-service op een Windows Server 2003. De metadata op de server is opgeslagen in een zogeheten metadata-store beheerd door MS SQL Server 2000. De metadata beschrijft op een gestandaardiseerde manier de views, acties, relaties en business-entiteiten van de IBF-applicatie.

Een IBF-toepassing ontwikkelen is in versie 1.0 nog niet zo eenvoudig: De IBF-ontwikkelomgeving is wel beschikbaar als een add-in in Visual Studio.NET maar nog niet onmiddellijk als een rapid-application-development (RAD) tool. Het aantal designers en wizards is beperkt, wat betekent dat er vandaag nog veel manueel programmeer- en configuratiewerk bij komt kijken. De eerste IBF toepassing die u zal ontwikkelen zal m.a.w. behoorlijk wat scholing en planning van u vergen.

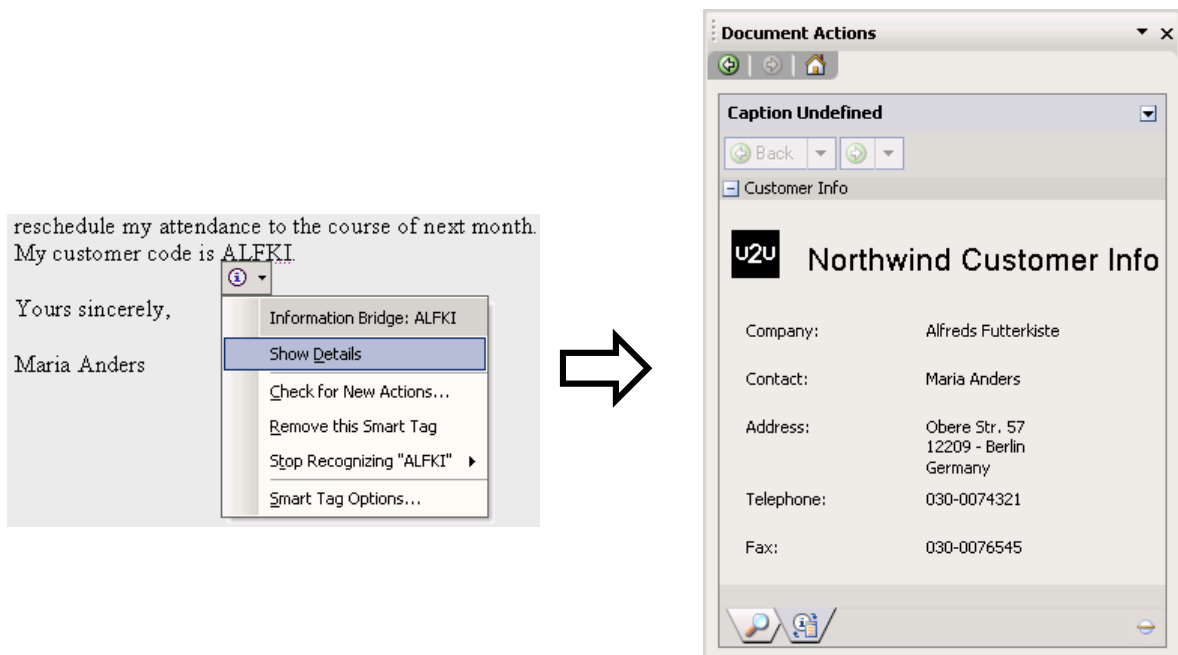
Gebruikmakend van versie 1.0 kunnen we 5 hoofdstappen onderscheiden in het programmeerwerk nodig om een IBF-toepassing aan te maken:

1. Maak een webservice aan conform de Information Bridge Framework-regels
2. Creëer de IBF-metadata
3. Ontwikkel de user-interface die zal getoond worden in de IBF-taskpane
4. Definieer de smart-tags hetzij een attached schema document als ingangspunt voor IBF-operaties
5. Installeer en test de IBF-toepassing

Voor de details van dit programmeerwerk verwijzen we naar het document "IBF in 5 steps, U2U tutor on the Information Bridge Framework" (<http://www.u2u.net/ibf>). Om het concreet en vooral onmiddellijk toepasbaar te houden, wordt de IBF-technologie in dit voorbeeld toegepast op de bekende Northwind databank (Microsoft SQL Server 2000).

De onderdelen van een IBF-toepassing

Laat ons de eenvoudige business-case beschouwen waarbij een informatiewerker zijn cursor plaatst op een klantcode in een Word-document en de corresponderende klantinformatie wenst op te vragen in de taskpane. De informatiewerker wijst de klantcode ALFKI aan in een Word-document, waardoor er een smart-tag verschijnt. Selecteert hij of zij het menu "Show Details" dan heeft dit tot gevolg dat in de taskpane de bedrijfsinformatie van Alfreds Futterkiste verschijnt (zie Figuur 3).



Figuur 3: De business-case die in dit artikel wordt besproken: de klantcode ALFKI wordt herkend door de smart-tag waardoor IBF de bedrijfsinformatie in de taskpane toont.

Wat gebeurt er nu achter de schermen? Welke informatie wordt hier op welke manier uitgewisseld? We overlopen de verschillende onderdelen van de IBF-toepassing en beschrijven de weg gevolgd vanaf het herkennen van de klantcode als een smart-tag in Word tot het weergeven van de klantinformatie in de taskpane.

Onderdeel 1: De webservice conform IBF

IBF klanten halen hun business-data op via webservices. Deze webservices moeten conform zijn met de richtlijnen van het Information Bridge Framework, wat o.a. betekent dat hun definities moeten overeenkomen met schema's in de IBF metadata-store.

In onze business-case maken we gebruik van de Customers webservice met een GetCustomerInfo-webmethode: als input geven we een klantcode op om als output de klantinformatie te bekomen. De richtlijnen van IBF toegepast op ons voorbeeld verplichten ons om de klantcode te encapsuleren in een reference-type – in ons geval de CustomerReference-class. Daarnaast moet de klantinformatie geëncapsuleerd zijn in een view-type – de CustomerView class (Figuur 5).

```
[WebService(Namespace="urn:schemas-u2u-net/Customers")]
public class Customers : System.Web.Services.WebService
{
    [WebMethod(Description="Retrieve info for a specific customer")]
    public CustomerView GetCustomerInfo(CustomerReference customerRef)
    {
        ...
    }
}
```

Figuur 4: Webservices moeten conform zijn aan de richtlijnen van IBF

```
[XmlRoot("CustomerReference", Namespace="urn:schemas-u2u-net/Customers")]
public class CustomerReference
{
    [XmlAttribute]
    public string CustomerID;
}
```

Figuur 5: Het reference-type (input) conform de richtlijnen van IBF

```
[XmlRoot("CustomerView", Namespace="urn:schemas-u2u-net/Customers")]
public class CustomerView
{
    [XmlAttribute]
    public string CustomerID;
    [XmlElement]
    public string CompanyName;
    [XmlElement]
    public string Address;
    ...
}
```

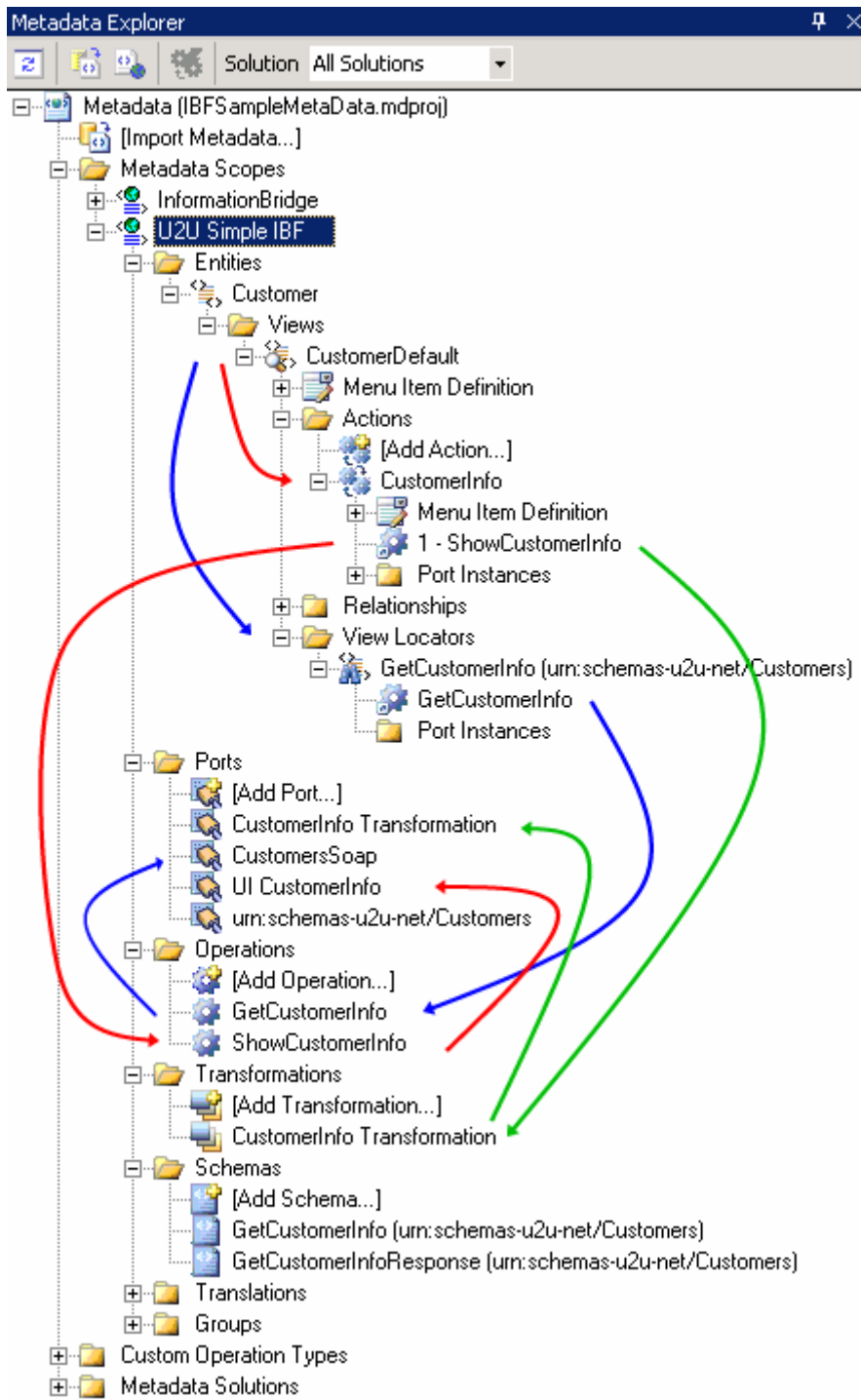
Figuur 6: Het view-type (output) conform de richtlijnen van IBF

Onderdeel 2 – De metadata-store op de IBF Server

De metadata-store in een IBF-toepassing bevat alle informatie in verband met de business-data die nodig is om een vraag te beantwoorden komende van de IBF-client, bijvoorbeeld een smart-tag. Zo

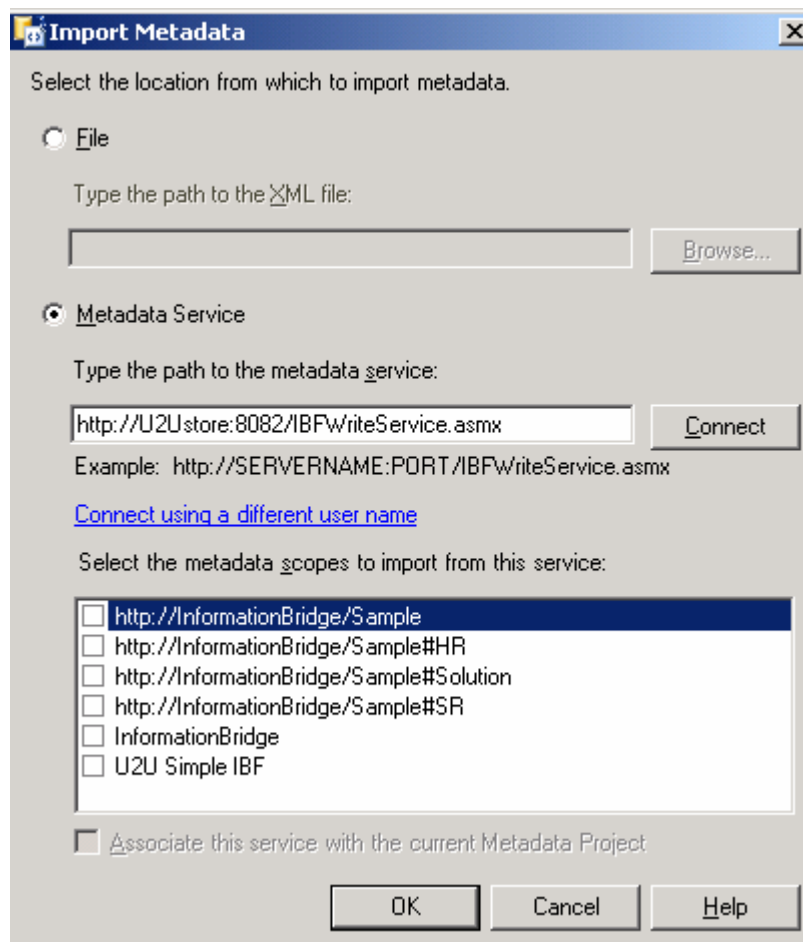
beschrijft de metadata o.a. de structuur van de business-data (schemas), waar de business-data zich bevindt (viewlocators), wat er met de business-data kan gebeuren (actions), hoe de business-data moet getoond worden (transformaties).

De metadata-store wordt automatisch geïnstalleerd als een Microsoft SQL Server database tijdens de setup van IBF. Het beheer van de metadata-store gebeurt in MS Visual Studio.NET via een speciale IBF project-template en een add-in, met name de Metadata Explorer (Figuur 7). Zowel de IBF template als de Metadata Explorer worden automatisch geïnstalleerd tijdens de setup van IBF.



Figuur 7: De Metadata-Explorer

Metadata kan uitgewisseld worden tussen de metadata-store en het metadata-project via de metadata-webservice (bv. <http://U2Ustore:8082/IBFWriteService.asmx> zoals te zien in Figuur 8).



Figuur 8: Importeren van metadata gebeurt van de metadata-store naar het metadata-project; publiceren van metadata gebeurt richting store.

Metadata toevoegen aan het project kan op verschillende manieren, manueel door de achterliggende xml-file te editeren bij Entities, Views, en Viewlocators), via wizards (Actions, Ports, Operations, Schemas...), via import uit xml-files.

De Metadata-explorer laat toe de belangrijke metadata-elementen te beheren en te editeren. Belangrijke elementen zijn:

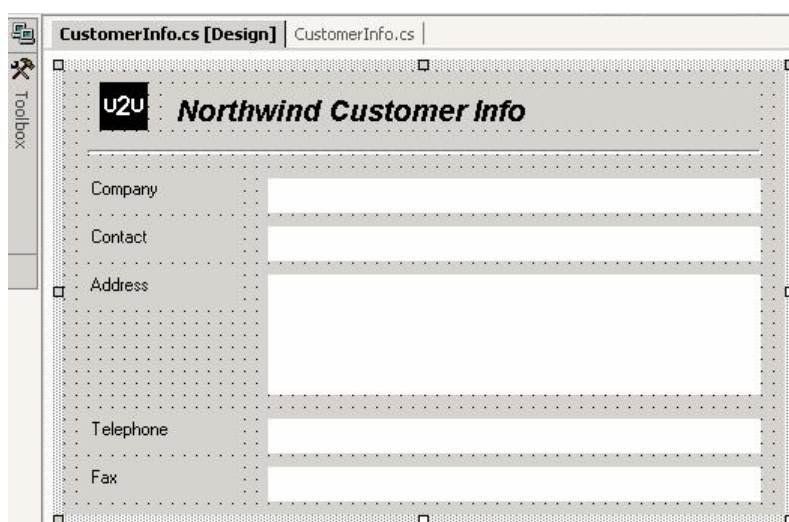
- **Ports** - de poorten naar de web service die de SOAP-details abstraheren, en de poorten naar de XSD schema-files voor de input en output types
- **Schemas** - de schema's van de IBF web service in apart request en response message-formaat
- **Operations** - de methoden die door IBF aanroepen kunnen worden
- **Entities** - de data-entiteiten en hun views die door IBF gebruikt moeten worden.

Onderdeel 3: De smart-tag component of het “attached schema-document”.

Als informatiewerkers met business-data omgaan, dan gebeurt dit vanuit de tekst in een Office document. Een smart-tag is een object dat zich at runtime creëert en nestelt in het Office document. Door toedoen van de smart-tag technologie, wordt het Office-document mogelijk onderdeel van de IBF-toepassing. Of informatiewerkers nu een bestaand document consulteren, of een nieuw document aanmaken, de smart-tag laat ons toe om de link van het Office-document naar de IBF-toepassing te maken. Een smart-tag wordt gedefinieerd door een smart-tag-component, welke in .NET typisch wordt aangemaakt als een class library project.

In plaats van gebruik te maken van smart-tags laat IBF ook toe dat we werken met schemas die een vaste structuur opleggen aan een Word of een Excel document om dan vanuit de gemapte XML velden de context informatie door te geven naar de IBF client-engine.

Onderdeel 4: De IBF user-interface



Figuur 9: De IBF user-interface wordt gedefinieerd als een Windows control in .NET

Een wezenlijk onderdeel van een IBF-toepassing is de gebruikers-interface in de office omgeving. In Word en Excel gebruikt de informatiewerker hiervoor de taskpane, in Outlook wordt deze zelfde taskpane als een top-level window getoond. De gebruikers-interface wordt in ons voorbeeld aangemaakt in .NET als een Windows user control, maar IBF ondersteunt ook gewone HTML, ingebouwde IBF controls (bv. de listcontrol) en de transformatietaal XSL voor de definitie van de interface binnen de taskpane.

IBF in werking

We volgen nu de weg die wordt afgelegd doorheen de IBF-functionaliteit vanaf het moment dat een term herkend wordt als smart-tag in Word tot het tonen van de business data in de taskpane.

Stap 1: De business-code wordt herkend

We beginnen bij de informatiewerker die een tekst typt in MS Word 2003 waarin de term ALFKI voorkomt. ALFKI wordt door de smart-tag engine van Office herkend als klantcode. Achter de

schermen werd de tekst namelijk gescant door de SmartTagRecognizer (zie Figuur 11) van onze smart-tag. De smart-tag (zoals hoger beschreven als onderdeel 3) is operationeel in Office en vergelijkt elk woord uit de tekst in ons geval met de codes uit een XML-bestand genaamd SmartTagTerms.xml. Onze smart-tag component herkent ALFKI als de code van een klant omdat deze code is opgenomen in dit XML-bestand(zie figuur x).

```
<Terms>
  <Customer>ALFKI</Customer>
  <Customer>Anton</Customer>
</Terms>
```

Figuur 10: SmartTagTerms.xml bevat de Customer-code ALFKI

De taak van de smart-tag bestaat erin om een XML-string op te bouwen die conform is aan een schema gedefinieerd in de IBF metadata-store en waarin zich de contextinformatie bevindt die vervolgens door de IBF client-engine zal worden verwerkt. De recognizer bereidt dit voor nadat de klantcode werd herkend (Figuur 11). De smart-tag onderlijnt de term ALFKI in de tekst en de informatiewerker kan met de cursor de smart-tag in werking stellen (Figuur 3).

```
public class SmartTagRecognizer : ...
{
  void Recognizer2(...)
  {
    ...
    switch (node.Name)
    {
      ...
      case "Customer":
        contextString = CustomerContextXml;
        break;
    }
    ...
  }
}
```

Figuur 11: de SmartTagRecognizer

Step 2: De request wordt verwerkt door de IBF client-engine

Klikt de informatiewerker op "Show Details" in het context-menu van de smart-tag, dan stuurt de Execute-methode van het Action object in de smart-tag een XML-message (zie Figuur 12) door naar de IBF client-engine. Deze engine pakt de boodschap dan weer uit om ze verder te verwerken op basis van de metadata die werd opgehaald uit de metadata repository.

```
<?xml version="1.0"?>
<ContextInformation xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  MetadataScopeName="U2U Simple IBF"
  EntityName="Customer" ViewName="CustomerDefault"
  xmlns="http://schemas.microsoft.com/
    InformationBridge/2004/ContextInformation">
  <Reference>
    <GetCustomerInfo xmlns="urn:schemas-u2u-net/Customers">
      <CustomerReference CustomerID="ALFKI"
        iwb:MetadataScopeName="U2U Simple IBF"
```

```

xmlns:iwb="http://schemas.microsoft.com/InformationBridge/2004"
iwb:EntityName="Customer" iwb:ViewName="CustomerDefault" />
</GetCustomerInfo>
</Reference>
</ContextInformation>

```

Figuur 12: de XML-message die door de IBF-client-engine naar de IBF-server wordt gestuurd

Het formaat van deze XML-message wordt bepaald door de metadata op de IBF-server, met name het schema GetCustomerInfo, welke verwijst naar de poort urn:schemas-u2u-net/Customers.

De XML-message bevat een request GetCustomerInfo met als argument een CustomerReference die de CustomerID bevat; de request verwijst naar de entiteit Customer en de view CustomerDefault. In wezen vraagt de XML-message dus om de Customer-gegevens van ALFKI te tonen volgens een CustomerDefault-view.

Step 3: Het ophalen van de benodigde metadata door de client-engine

De XML-contextmessage wordt beantwoord door de IBF client-engine waar logisch beschouwd de CustomerDefault-view in de metadata vanaf nu het coördinerende werk overneemt, welliswaar binnen de omkadering van het Information Bridge Framework. Het antwoord aangeleverd door de IBF client-engine wordt geformuleerd op basis van de metadata die wordt opgehaald en gecached via het aanspreken van de metadata-webservices aangeboden door de IBF-servercomponent. De metadata bestaat in ons voorbeeld uit drie onderdelen: de url van de business-webservice, de url van de .NET assembly die de user control bevat en de naam van de usercontrol-class. Met behulp van de metadata-explorer (Figuur 7) vinden we de verschillende onderdelen:

Onderdeel 1: de url van de business-webservice:

De CustomerDefault-View gaat naar de ViewLocators (blauwe pijlen in figuur 7) en vindt aldaar de operatie GetCustomerInfo die moet worden uitgevoerd om de Customer-data op te halen. De operatie GetCustomerInfo verwijst via een property naar de poort CustomerSoap, welke het eerste onderdeel van het antwoord bevat. Dit is in ons voorbeeld de locatie van de business-webservice: <http://www.u2u.net/CustomerServices/Customer.asmx>

onderdeel 2: url van de .NET assembly

De CustomerDefault-View gaat naar de Actions (rode pijlen in figuur 7) en vindt er de actie CustomerInfo. Deze actie verwijst naar de operatie ShowCustomerInfo. In de Operations verwijst ShowCustomerInfo via een property naar de poort UI CustomerInfo. Bij de Ports vermeldt de poort UI CustomerInfo via een property naar de url van de Windows Control-DLL, in ons voorbeeld: [IBFNorthwindUserControls.dll](#)

Onderdeel 3: de naam van de usercontrol-class

Eén van de properties van de ShowCustomerInfo-operatie in de Actions, met name de TransformationInstances-property verwijst (groene pijlen in figuur 7) naar de transformatie "CustomerInfo Transformation". Bij de Transformations verwijst de "CustomerInfo Transformation" via een property naar de poort CustomerInfo Transformation. Bij Ports vinden we deze poort waarvan de Data-property de XSL weergegeven in Figuur 13:

```

<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <msibf:Region
      xmlns:msibf="http://schemas.microsoft.com/InformationBridge/2004"
      Enabled="true">
      <msibf:RegionProperties RegionName="RegionCustomerInfo"
        Caption="Customer Info" Description="Customer Info"

```

```

        TypeName="IBFNorthwindUserControls.CustomerInfo"
        ShowAs="ExpandedRegion">
        <xsl:copy-of select="/" />
        </msibf:RegionProperties>
        </msibf:Region>
      </xsl:template>
</xsl:stylesheet>

```

Figuur 13: de XSL-property die verwijst naar de User Interface

Deze XSL bevat een link naar de usercontrol-class met naam "IBFNorthwindUserControls.CustomerInfo", welke de class is die zal gebruikt en getoond worden in een region dat deel uitmaakt van het IBF gebruikersinterface (zie Figuur 9).

De uiteindelijke response welke gegeven wordt door de IBF-metadata-service bestaat dus uit:

- "http://www.u2u.net/CustomerServices/Customer.asmx" als url van de business-webservice
- "IBFNorthwindUserControls.dll" als url van de .NET assembly
- "IBFNorthwindUserControls.CustomerInfo" als naam van de user control klasse

Stap 4: Het antwoord wordt verwerkt door de IBF client-engine

De IBF client-engine zorgt nu voor het daadwerkelijk aanroepen van de business-webservice, en het tonen van de hieruit verkregen data in de IBF taskpane. De usercontrol ontvangt de business-data via de IRegion.Data property; deze data is geformateerd als XmlNode conform het GetCustomerInfoResponse-schema. De Data property van de user control zorgt voor het daadwerkelijk invullen van de klantgegevens in de IBF region (Figuur 14).

```

public class CustomerInfo : UserControl, IRegion
{
    ...
    public XmlNode Data
    {
        set
        {
            labelCompany.Text = value.FirstChild.ChildNodes[0].InnerText;
            labelAddress.Text = value.FirstChild.ChildNodes[2].InnerText +
            ...
        }
    }
}

```

Figuur 14: De Data-property van de usercontrol zorgt voor het daadwerkelijk invullen van de klantgegevens in de UI controls

Voor de business-case die we in dit artikel bespraken is het gewenste resultaat bekomen, maar in business-cases waar data moet worden onderhouden of waar business-activiteiten gestart moeten worden, is er nog wat bijkomend, welliswaar analoog, werk te doen.

Referenties

MSDN over de Information Bridge Framework:

<http://msdn.microsoft.com/office/understanding/ibframework/default.aspx>

“IBF in 5 steps, U2U tutor on the Information Bridge Framework”

<http://www.u2u.net/ibf>

Over de auteurs

Wim Uyttersprot (wim@u2u.be) en Patrick Tisseghem (patrick@u2u.be) zijn managing partners van U2U, Gert Servranckx (gert@u2u.be) werkt bij U2U als .NET trainer. U2U is als gecertificeerd .NET trainingscenter te Brussel gespecialiseerd in .NET programmatie. U2U verzorgt tal van workshops voor Microsoft EMEA (www.u2u.net).