Internet

Tijdens LEAP 2008, het architectenprogramma van Microsoft, was er een sessie over de Internet Service Bus (ISB). De ISB is een implementatie van Enterprise Service Bus (ESB) technologie, waarbij de services niet noodzakelijk lokaal bij de klant aanwezig hoeven te zijn, maar bij de leverancier, een partner of een willekeurige hosting-partij.

Bouwen van servicegeoriënteerde applicaties

Internet Service Bus

De hosting van services wordt hiermee flexibel. Services kunnen verplaatst worden van de lokale omgeving in de organisatie naar een derde partij of juist omgekeerd. De mogelijkheid van dergelijke services valt onder de nieuwe mantra van Microsoft: het zogenaamde Software plus services. Hierbij biedt het bedrijf services zoals Office, Exchange, BizTalk enzovoort aan op het internet, via derde partijen of gewoon lokaal voor eigen gebruik. Er is in feite sprake van een nieuwe set aan services, gebaseerd op Biztalk-technologie om ontwikkelaars te ondersteunen bij het bouwen van dienstgeoriënteerde applicaties.

Software plus services

Met Software plus services, niet te verwarren met Software as a Service waarbij alle services zich op het internet bevinden, heeft Microsoft een visie over de combinatie van locale software en internetservices die met elkaar werken. Software maakt services beter en andersom, zodat de softwareleverancier beide bij elkaar willen brengen en zo de keuze geeft van het beste van beide werelden. Deze visie komt van Chief Software Architect Ray Ozzie, die deze rol van Bill Gates heeft overgenomen. De ISB biedt services aan op het internet (in de wolk), waarbij services bij organisaties zelf (dit concept software genoemd) draaien. Services kunnen ook gewoon systemen zijn, die de mogelijkheden in zich hebben met services te communiceren, al dan niet via een bus. De ISB heeft een verband met of past in deze visie van Software plus services.

Internet Service Bus

Wat is de Internet Service Bus nu precies, en hoe verhoudt het zich tot de Enterprise Service Bus? Een ESB kan men in een organisatie toepassen ter ondersteuning van bijvoorbeeld het beheren van verbindingen tussen systemen die samengestelde applicaties (*composite applications*) kunnen gebruiken. De ESB heeft dan een rol als broker en adresseert problemen die je tegenkomt bij een aanpak met enkelvoudige (zogenaamde point-to-point) verbindingen. Het laatste wordt lastig te beheren naarmate applicaties meer verbindingen naar diverse systemen nodig hebben. Naast de genoemde broker-functie wordt het beheer van de verbindingen eenvoudig en biedt de ESB ook ondersteuning in naamgeving, identiteit, formaten en het werken met berichtentechnologie die nodig zijn voor het verbinden van diverse diensten met applicaties. Dit alles is binnen het bereik van een bedrijf. Wil men connectiviteit tussen bedrijven, en delen of services *outsourcen* naar derden, dan wordt het bereik groter. Een ESB-aanpak zal dan

Steef-Jan Wiggers
is werkzaam bij Inter Access

niet voldoende zijn, omdat men dan het liefst verbindingen tussen services wil leggen over het internet: een bus over het internet.

Voordelen ISB

Een ISB kan uitkomst bieden wanneer bedrijven een infrastructuur nodig hebben om services tussen interne systemen en services bij zakelijke partners te laten integreren. Daarbij kunnen bedrijven snel schalen en kan de ISB besparing op IT-personeel of technologie opleveren, omdat je alleen nog maar van services gebruikmaakt. De ISB is een verzameling van services, die zoals eerder vermeld de ontwikkeling van servicegeoriënteerde applicaties vereenvoudigt. Eigenlijk is het een combinatie van twee soorten services. Door Microsoft via netwerk beschikbaar gestelde aanpasbare services (bouwblokken) en lokale services die toegankelijk zijn via API's en programmeerraamwerkextensies. Diagram 1 geeft een overzicht van het ISB-concept.

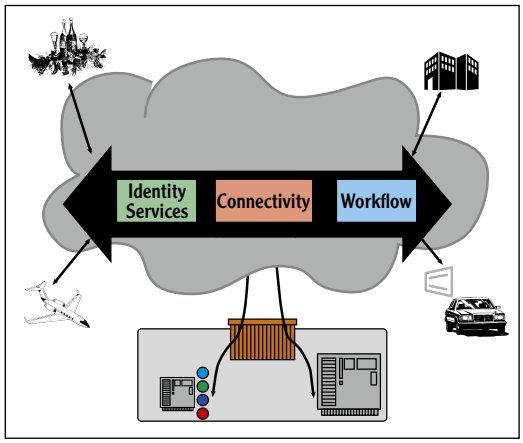
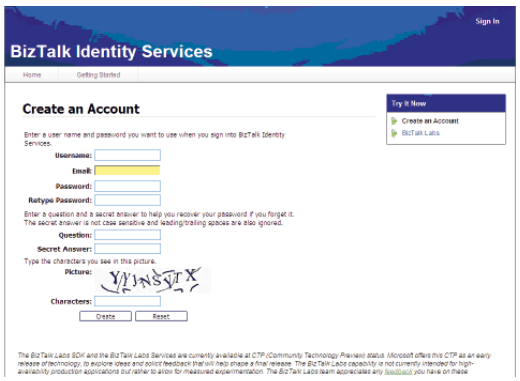


Diagram 1. Overzicht ISB.

De ISB bevat drie bouwblokservices: Identity, Connectivity en Workflow. Deze services kun je beschouwen als een geïntegreerd geheel. Een volledig dienstverleningsproces kan worden ondersteund door meerdere partijen; het boeken van een vlucht met hotel, het huren van een auto of taxi in combinatie met eten in een restaurant. Het meeste van de technologie achter ISB is gebaseerd op Windows Communication Foundation (WCF). WCF is een technologie die al beschikbaar is via .NET Framework 3.0/3.5 en is een programmeer-framework, waarbij alle vorige communicatieprogrammeermodellen zijn verenigd, zoals SOAP-gebaseerde communicatie (webservices), .NET remoting, transactionele communicatie, en asynchrone communicatie (message queues). Met dit raamwerk hoeft een programmeur alleen nog maar één programmeermodel te kennen. Een bijkomstig voordeel van WCF is dat de implementatie gescheiden wordt van de configuratie zoals adressering en communicatie (binding). De ISB bezit de volgende vier eigenschappen in de vorm van de eerdergenoemde bouwblokservices, waarbij sommige al beschikbaar zijn en andere nog in ontwikkeling: Identity Management en Access Control, Connectivity, Workflow, Logging.

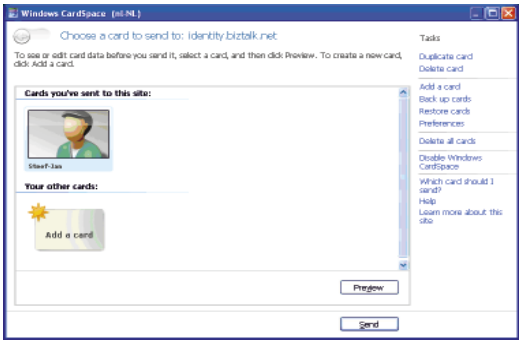
Identity Management en Access Control

BizTalk Services biedt via http://identity.biztalk.net accountmanagement voor gebruikers, nieuwe Active Directory-toegang en federation-diensten, ondersteuning voor eigen zogenaamde cards (Windos Cardspace) en claimgebaseerde toegangscontrole. Hoe werkt dit? Voordat je met services aan de gang kunt gaan, moet een account worden aangemaakt bij http://biztalk.net/, waar men onder 'Getting Started' het startpunt vindt. Daar aanbeland moet een gebruikersnaam, e-mailadres en wachtwoord worden opgegeven, zie figuur 1.



Figuur 1. Aanmaken account.

Wanneer dat is gebeurd, zal een associatie gecreëerd moeten worden met een Windows CardSpace-card. CardSpace is een nieuwe technologie in het .Net Framework 3.0 die een vereenvoudiging en verbetering is voor veilige toegang van bronnen en het delen van persoonlijke informatie op het internet. Het beidt simpele on-line authenticatie, verhoogde website-security en bescherming tegen digitale identiteitdiefstal; vandaar de toepassing van deze technologie bij BizTalk Services. Figuur 2 geeft de user-interface weer bij het selecteren van een card. Deze interface is beschikbaar als .Net Framework 3.0 is geïnstalleerd.



Figuur 2. Windows CardSpace user-interface, selecteren van een card.

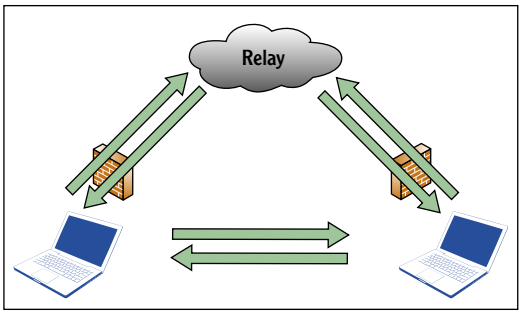
Workflow- en logging-service-bouwblokken zijn nog in ontwikkeling

Connectivity

Connectivity geeft ontwikkelaars de mogelijkheid zogenaamde 'loosely coupled' applicaties te bouwen met de volgende eigenschappen:

- Eenvoudig publish/subcribe-mechanisme vergelijkbaar met die van BizTalk Server
- Voorziening in globaal adresseerbare namen voor services
- Veilig beschikbaar stellen van services via het internet achter een firewall of NAT (zie diagram 1)
- Meerdere zogenaamde 'listeners' en 'senders' op een URI ('*multicast*', simultaan leveren van berichten over een netwerk naar meer ontvangers tegelijk of vice versa)
- Automatisch verzorgen van een directe verbinding tussen endpoints, waar mogelijk.

Een algemeen probleem bij de ontwikkeling van verbonden peer-to-peer applicaties is het opzetten van verbindingen tussen applicaties over netwerkapparaten zoals firewalls of network address translators (NAT). Deze apparaten staan initieel wel uitgaand verkeer toe, maar voorkomen eventuele inkomende netwerkverbindingen. Dit is lastig als communicatie gewenst is tussen services bij verschillende (on)afhankelijke organisaties. De oplossing wordt vaak gezocht in het schrijven van eigen code om te kunnen omgaan met verschillende netwerktopologieën. Dit kan echter een complexe zaak worden. BizTalk Services Connectivity biedt daarom een infrastructuur om eenvoudige verbindingen te kunnen realiseren. Ontwikkelaars kunnen nu services aanbieden achter een firewall of NAT, zonder enige code te hoeven schrijven voor het netwerklogistieke proces; zie figuur 3.



Figuur 3. Relay mogelijk via de connectivity-service in de wolk (ISB).

Workflow en Logging

Workflow- en logging-servicebouwblokken zijn nog in ontwikkeling. BizTalk Services zullen in de toekomst een gehoste instantie leveren van Windows Workflow Foundation (WWF), waardoor workflow-mogelijkheden worden geboden. Workflows die in de ISB draaien, kunnen worden geactiveerd door berichten die worden ontvangen via URI's. Workflows zelf bieden ondersteuning voor de verwerking van berichten in meerdere stappen waaronder transformatie, versturen en ontvangen van berichten van en naar verbonden services, enzovoort. Naast workflow zullen waarschijnlijk logging-services worden geleverd voor de logging van berichten die via een URI binnenkomen.

Experimenteren

Om meer te weten te komen over ISB kun je naar http://labs.biztalk.net/ gaan. Daar kun je volop experimenteren met een beschikbare Community Technology Preview (CTP) en kun je vroegtijdig inzicht krijgen in de technologie. Download de SDK en installeer eventueel vooraf de vereiste software als Internet Explorer 7 en .NET framework 3.0. Op dat moment lijkt het alsof er een omgeving is gecreëerd met alle vereiste componenten om met de ISB-technologie aan de gang te gaan, maar er ontbreekt nog één essentieel onderdeel: Visual Studio. Omdat de SDK alleen voorbeelden levert in C# en de nodige assemblies installeert, is deze ontwikkelomgeving vereist. Installeer dus ook Visual Studio 2005 of 2008.
Met een volledige geconfigureerde en geïnstalleerde ontwikkelomgeving kan het experimenteren met de voorbeelden beginnen. Toegang tot BizTalk Labs is wel nodig en een account zoals eerder beschreven dient te worden aangemaakt. De SDK bevat onder andere voorbeelden voor het gebruik van Identity- en Connectivity-services. Als eerste kun je een simpel echovoorbeeld proberen, waarbij je inzicht krijgt in een vraag/antwoordcontract (request-reply contract) dat beschikbaar wordt gesteld, en dat gebruikmaakt van de relay-binding en de nieuwe claim based access-controlemogelijkheden. Het voorbeeld bestaat uit een serviceproject met een eenvoudig contract (EchoContract).

```
    [ServiceContract(Name = "EchoContract", Namespace
=     "http://samples.microsoft.com/ServiceModel/
Relay/")]
     public interface EchoContract
     {
         [OperationContract]
         string Echo(string text);
     }
```

Listing 1.

De service implementeert dit contract in de EchoService-class. Om de service te laten authenticeren met de relay, wordt een CardSpace tokenprovider aangemaakt. De gebruikersnaam wordt geëxtraheerd vanuit deze tokenprovider en wordt gebruikt om een service-URI aan te maken. Vervolgens is een service-endpoint beschikbaar en wordt de tokenprovider aan de endpoint behaviours-collectie toegevoegd.

```
CardSpaceTokenProvider tokenProvider = new
CardSpaceTokenProvider();
                string userName = tokenProvider.
GetUserName();

       Uri address = new Uri(String.Format("sb://{0}/
services/{1}/EchoService/",      RelayBinding.
DefaultRelayHostName, userName));

            ServiceHost host = new ServiceHost(typ
eof(EchoService), address);
           host.Description.Endpoints[0].Behaviors.
Add(tokenProvider);
            host.Open();
<Bijschrift> Listing 2.
```

Uiteindelijk creëert de service een endpoint die luistert naar de relay-service. De afnemer van de service kan verbinding maken met de service en er berichten naar versturen. Daarvoor wordt de endpoint geopend, zoals in listing 3 die zich in de afnemer (client) bevindt.

```
Console.Write("Enter the name of the user you want
to connect with: ");
                string serviceUserName = Console.
ReadLine();

     Uri serviceUri = new Uri(String.Format("sb://{0}/
services/{1}/EchoService/",      RelayBinding.
DefaultRelayHostName, serviceUserName));

     ChannelFactory<EchoChannel> channelFactory = new
ChannelFactory<EchoChannel>("RelayEndpoint",   new
EndpointAddress(serviceUri));

             EchoChannel channel = channelFactory.
CreateChannel();
            channel.Open();
```

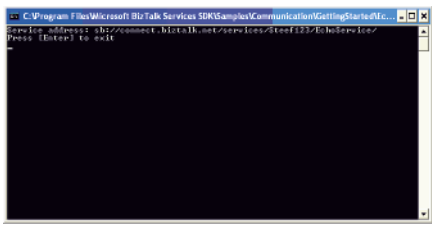
Listing 3.

Nadat de ChannelFactory is aangemaakt, creëert de serviceafnemer een channel naar de service. Communicatie over en weer vindt plaats door de aanroep-channel.Echo(input). Wanneer dit heeft plaatsgevonden, worden de Channel en ChannelFactory gesloten. Het beschreven verloop van het echovoorbeeld gaat niet helemaal vanzelfsprekend, wanneer men een VS 2008 ontwikkelomgeving ter beschikking heeft. Ten eerste moet voorbeeldcode worden geconverteerd, maar dat verloopt nog probleemloos. Ten tweede ontstaat een probleem met de configuratie bij het executeren van de service. De relay-binding in de configuratie wordt niet begrepen, omdat deze binding niet bekend is in de machineconfiguratie van .NET Framework die terug te vinden is in de map: C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG. In de sectie <system.serviceModel> in de machine configfile dienen de nodige toevoegingen te worden gedaan; zie listing 4.

```
 <bindings>
<relayBinding>
         <binding  name="metadataExchangeRelayBindin
g" />
  </relayBinding>
 </bindings>

<extensions>
<behaviorExtensions>
         <add    name="connectionStatusBehavior"
type="System.ServiceBus.Configuration.
ConnectionStatusElement,     System.ServiceBus,
Version=0.11.0.0, Culture=neutral, PublicKeyToken=31
bf3856ad364e35" />
</behaviorExtensions>
<bindingElementExtensions>
         <add  name="relayTransport"  type="System.
ServiceBus.Configuration.RelayBindingElementSection,
System.ServiceBus, Version=0.11.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
</bindingElementExtensions>

<bindingExtensions>
         <add  name="relayBinding"  type="System.
ServiceBus.Configuration.
RelayBindingCollectionElement,  System.ServiceBus,
Version=0.11.0.0, Culture=neutral, PublicKeyToken=31
bf3856ad364e35" />
</bindingExtensions>

</extensions>

<client>
         <endpoint address="" binding="relayBinding"
bindingConfiguration="metadataExchangeRelayBinding"
contract="IMetadataExchange" name="sb" />
<metadata>
<policyImporters>
                                        <extension
type="System.ServiceBus.Description.
RelayBindingElementImporter,   System.ServiceBus,
Version=0.11.0.0, Culture=neutral, PublicKeyToken=31
bf3856ad364e35" />
                  </policyImporters>
<wsdlImporters>
                             <extension
type="System.ServiceBus.Description.
RelayBindingImporter,      System.ServiceBus,
Version=0.11.0.0, Culture=neutral, PublicKeyToken=31
bf3856ad364e35" />
                                        <extension
type="System.ServiceBus.Description.
RelayBindingElementImporter,   System.ServiceBus,
Version=0.11.0.0, Culture=neutral, PublicKeyToken=31
bf3856ad364e35" />
                  </wsdlImporters>
          </metadata>
  </client>
```

Listing 4.

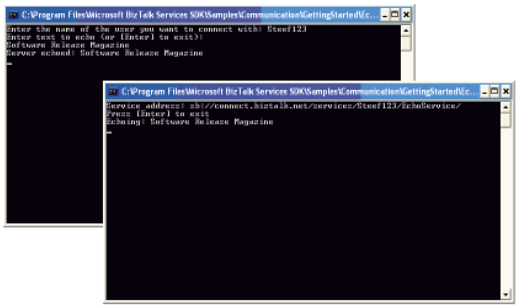
De assembly System.ServiceBus is tijdens de installatie van de SDK geïnstalleerd in Global Assembly Cache (GAC). Bij deze assembly moet de versie worden opgezocht en dit nummer moet met het nummer in het attribuut Version overeenkomen. Dit geldt eveneens voor het attribuut PublicToken. Met het aanpassen van de machineconfig kan nogmaals getracht worden de service



Figuur 4: Start van de Echoservice

te starten. Nu zal de service waarschijnlijk wel starten en verzoeken om het selecteren van een card (zie figuur 2) die verzonden dient te worden. Figuur 4 zal dan getoond worden.

De afnemer van de service kan nu gestart worden, waarbij men wederom gevraagd zal worden een card te sturen. Dan kan gekozen worden met welke gebruiker verbinding wordt gezocht en kan een tekst worden verstuurd. Figuur 5 geeft het resultaat weer.



Figuur 5. Serviceafnemer en de servicecommunicatie

Op het eerste gezicht een vrij eenvoudig voorbeeld, maar wel een waar je als eerste mee zou moeten starten om dan vervolgens met de andere voorbeelden aan de slag te gaan. Dit omdat je rekening dient te houden met de omgeving waar de voorbeelden worden uitgevoerd. Dat wil zeggen: welke Visual Studio-omgeving is geïnstalleerd en is minimaal het .NET 3.0 Framework aanwezig, waar WCF en CardSpace onder andere van afhankelijk zijn. Ten slotte wordt hier uitgegaan van een voorbeeld dat is uitgevoerd met Visual Studio 2008 en de nieuwste versie van de BizTalk Services SDK. De uitvoer van de voorbeelden met een andere SDK of Visual Studio 2005 kan wel eens heel anders verlopen en zijn de beschreven kunstgrepen misschien niet nodig.

Toekomst

In de toekomst zullen meer bouwblokservices beschikbaar komen. De door Biztalk Labs beschikbaar gestelde technologie van de ISB is experimenteel, zoals Microsoft aangeeft. Er zijn nog geen officiële beslissingen genomen over de exacte naamgeving en dergelijke, iets dat je vaker tegenkomt als Microsoft producten of technologieën ontwikkelt die nog niet volledig zijn vrijgegeven en ondersteund. De nodige aanpassingen en veranderingen zullen nog worden doorgevoerd. De ISB (BizTalk Services) wordt ook genoemd in het Oslo-project van Microsoft, waar het deel uit maakt van een aantal technologieën en producten zoals .NET Framework 4.0, volgende versies van BizTalk Server, Visual Studio en System Center. Oslo is een codenaam voor technische investeringen die Microsoft doet in de eerder genoemde producten en technologieën. In Oslo komen applicatiemodellering en SOA samen, waarbij het bedrijf een uniform platform wil creëren waarbij service en modellering worden geïntegreerd. In de plaats dat modellen applicaties beschrijven, zijn de modellen de applicaties zelf. Volgens analisten van onder andere Gartner is Oslo zeer ambitieus en ligt het nu nog ver weg. Buiten het Oslo-project om zou een ISB ook deel kunnen uitmaken van een Service Oriented Architecture vanwege de eenvoudige manier van samenstellen van applicaties en geboden flexibiliteit. Voor het ontwikkelen van samengestelde applicaties, die over meerdere organisaties heen gaan, kan de ISB een uitkomst bieden. Daarbij verleg je de scope van het samenstellen van applicaties en communiceren met diverse systemen binnen een organisatie (ESB) naar buiten. Bijvoorbeeld wanneer je over meerdere communicatieprotocollen heen wilt integreren, koppelingen tussen organisaties wilt managen, en beveiliging wil regelen tussen de organisaties over het internet. Zover is het echter nog lang niet. Naast technologische verbeteringen zullen de nodige kwesties als vertrouwen, service level agreements tussen bedrijven, governance, bestaande architecturen, eigenaarschap van data, wet en regelgeving, complexiteit en productiviteit belangrijke rollen hebben om een ISB-implementatie binnen een SOA te plaatsen. Deze technologie is ook nog niet volwassen genoeg en bevindt zich nog in een experimentele fase.

Tot slot

Veel acroniemen en namen als SOA, SaaS, Software plus services, Oslo-project, Enterprise Service Bus en nu dus de Internet Service bus zijn voorbij gekomen. De ISB wordt in meer of mindere mate in verband gebracht met deze zaken. Vandaar dat deze termen in dit artikel zijn gebruikt en is getracht deze in context met de ISB te verduidelijken. Met de ISB-technologie zelf kan nu worden geëxperimenteerd en je kunt eventueel gaan nadenken over hoe en in welke mate je het in de toekomst zou kunnen toepassen. Aan al de genoemde acroniemen wordt door Microsoft in meer en mindere mate ruchtbaarheid gegeven en daarmee wordt interesse gewekt in de IT-wereld. Vervolgens is het lastig het een en ander in de juiste context te brengen en te begrijpen. ISB is interessant, maar of het slaagt, moeten we afwachten, wat ook voor de andere genoemde zaken als het Oslo-project, Microsoft ESB en Software plus services geldt. «