Ewoud Jansen
is Developer Consultant bij Microsoft. Hij kan bereikt worden op ewoudj@microsoft.com

Bedrijfsprocesintegratie met Microsoft BizTalk Server 2002

Het belang van messaging

Microsoft BizTalk Server 2002 is een geavanceerd platform voor bedrijfsprocesintegratie. In de eerste editie van .NET Magazine is Microsoft BizTalk Server al uitgebreid aan bod gekomen, deze keer gaat de auteur dieper in op een belangrijk aspect van bedrijfsprocesintegratie: Enterprise Application Integration (EAI) op basis van BizTalk messaging.

Het eerste dat de meeste managers voor ogen staat als ze denken aan BizTalk Server is de BizTalk Orchestration Designer; visueel gezien het indrukwekkendste onderdeel van BizTalk Server en – voor de manager – het meest aansprekende. Dit product stelt de bedrijfsanalist in staat een bedrijfsproces visueel te definiëren in een op Microsoft Visio gebaseerde omgeving. De stappen in dit bedrijfsproces kunnen vervolgens, ook geheel op grafische wijze, worden gekoppeld aan technologiecomponenten. Het resultaat van de BizTalk Orchestration Designer, een zogenaamd XLANG-schedule, kan bovendien direct uitgevoerd worden alsof het een executable is. Het is mogelijk langlopende processen die bijvoorbeeld maanden duren te definiëren. Een dergelijk proces kan bijvoorbeeld wachten op de levering van bepaalde goederen. Zodra deze goederen worden geleverd, gaat het proces verder waar het gebleven is. Orchestration heeft een aantal indrukwekkende eigenschappen. Zo kan bijvoorbeeld een draaiend XLANG-schedule system reboots of zelfs een system restore overleven. Bovendien maakt Orchestration het ondernemingen mogelijk bedrijfsprocessen te veranderen zonder dat daarvoor de ondersteunende technische informatiesystemen moeten worden aangepast.

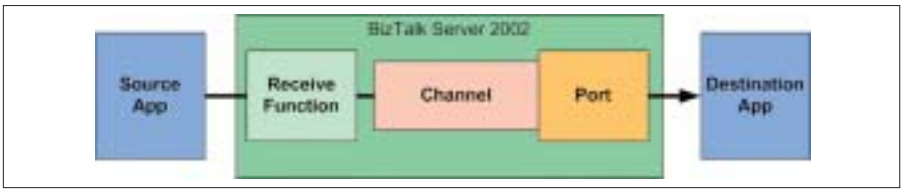
Vanuit de BizTalk Server architectuur gezien, is Orchestration slechts de bovenste van twee lagen. Aan de onderste laag, BizTalk Messaging, zal de ontwikkelaar veel tijd besteden. Niet omdat het moeilijk of tijdrovend is, maar juist omdat veel zaken zich daar eenvoudig en snel laten oplossen. Veel op BizTalk Server gebaseerde oplossingen gebruiken Orchestration zelfs helemaal niet, of pas in een later stadium. Van belang is in te zien dat de meeste geautomatiseerde bedrijfsprocessen langlopend zijn en daarom gedeeltelijk of zelfs geheel asynchroon kunnen worden afgehandeld. Asynchrone communicatie tussen informatiesystemen, ook wel Message Queuing genoemd, geniet bij verre de voorkeur boven synchrone. Tot de belangrijkste voordelen behoren:

- *Hogere beschikbaarheid*: systemen zijn niet meer afhankelijk van elkaars beschikbaarheid, zelfs als een informatiesysteem tijdelijk onbeschikbaar is kan het toch berichten accepteren en in een queue stoppen.
- *Hogere schaalbaarheid*: een informatiesysteem kan tijdens piekuren berichten in een queue stoppen en deze vervolgens op een later tijdstip verwerken. Op deze manier wordt werkdruk verspreid over de tijd.
- *Hogere betrouwbaarheid*: gebruikmakende van het BizTalk Framework 2.0 is het mogelijk gegarandeerd eenmalig berichten af te leveren tussen heterogene omgevingen over het Internet.

Men kan als vuistregel nemen dat wanneer de situatie het toelaat, altijd gekozen dient te worden voor asynchrone communicatie.

BizTalk Messaging

BizTalk Messaging is verantwoordelijk voor het berichtenverkeer in BizTalk Server. Om een goed begrip te krijgen van de werking en mogelijkheden van BizTalk Messaging kijken we kort naar de centrale begrippen. Een bericht dat aankomt bij BizTalk wordt opgevangen door een



Afbeelding 1. De centrale begrippen

Receive Function. De Receive Function geeft het bericht aan BizTalk Messaging die het bericht in een Channel stopt. Een Receive Function kan het bericht in een specifieke Channel stoppen of BizTalk Server kan op basis van de vorm en/of inhoud zelf bepalen in welke Channel een bericht thuis hoort. De Channel voert vervolgens een aantal acties uit op het binnenkomende bericht. Tot de mogelijke acties behoren het controleren van de berichtstructuur, het controleren van een digitale handtekening, decryption en de transformatie van het bericht van het ene formaat in het andere, ook wel mapping genoemd. Als de Channel klaar is, wordt het bericht afgeleverd op een 'port' die het bericht gaat versturen. De port heeft hierbij de keuze uit een aantal protocollen waaronder HTTP, SMTP en Message Queuing. Het bericht kan ook naar een filesysteem worden weggeschreven, worden afgeleverd bij een COM-component of een Orchestration Schedule. Voordat de port het bericht aflevert kan het (onder andere) ook nog worden voorzien van encryption of een digitale handtekening. Het eventuele adres dat gebruikt word door de port in combinatie met het protocol kan van tevoren worden geconfigureerd of kan uit het bericht gelezen worden.

BizTalk Messaging kan berichten op twee manieren ontvangen en verwerken: synchroon en asynchroon. BizTalk Server 2002 biedt standaard drie zogenaamde Receive-functies: een File Receive Function, een Message Queue Receive Function en een HTTP Receive Function. De File en Message Queue Receive Function communiceren asynchroon. De HTTP Receive Function kan, optioneel, ook synchroon communiceren. Deze Receive-functies zijn te configureren met behulp van de BizTalk Server Administration Console. Als de standaard Receive Functions niet de gewenste functionaliteit bieden kan men besluiten er zelf een te schrijven. De BizTalk Server API biedt de IInterchange-interface om berichten BizTalk in te sturen met als belangrijke methoden: Submit en SubmitSync. Deze methoden leveren een bericht af bij BizTalk Messaging. SubmitSync geeft ook een resultaatbericht terug waardoor men een synchrone interface kan maken.

Synchrone communicatie is niet met alle porttypen mogelijk. Van de BizTalk porttypen, ondersteunen alleen Application Integration Component, HTTP(S) en Loopback de synchrone communicatie. Laten we nu een drietal integratiemodellen bekijken dat we kunnen hanteren als we onze messaging-oplossing op basis van BizTalk ontwerpen: Point-To-Point Integration, Hub-Spoke Integration, Publish-Subscribe Integration. Vervolgens zullen we een korte blik werpen op hoe BizTalk Straight Thru Processing-diensten levert en hoe vanuit Microsoft Visual Studio .NET gebruik gemaakt kan worden van de diensten die BizTalk biedt.

Integratie modellen

Point-To-Point is de meest eenvoudige vorm van EAI. Twee systemen worden direct door BizTalk met elkaar verbonden. In een dergelijk model is het ook eenvoudig om synchrone communicatie te implementeren. Men zal zich afvragen wat in dit scenario de toegevoegde waarde is van BizTalk. Het antwoord op deze vraag is echter eenvoudig: BizTalk biedt allerlei diensten zoals mapping, certificate checking, logging en tracking.

In een zogenaamd *Hub-Spoke* integratiemodel is sprake van een centraal systeem: de Hub. In dit geval is dat BizTalk Server, dat al het berichtenverkeer coördineert tussen een aantal perifere systemen, de Spokes. In dit scenario is het zo dat een spoke een bericht aflevert bij de hub die het bericht op zijn beurt weer aflevert bij een andere spoke. Het belangrijke voordeel van dit model is dat de complexiteit beheersbaar blijft bij de integratie van een groter aantal systemen. BizTalk Messaging kan berichten routeren zodat ze op de juiste bestemming aankomen. In het Point-To-Point model is het zo dat er voor iedere verbinding twee mappings nodig zijn in het geval van tweewegcommunicatie. Voor ieder extra systeem dat wordt toegevoegd, zijn dan twee mappings nodig naar ieder reeds bestaand berichtformaat. Het aantal nodige mappings kan worden berekend met de formule: *2*(n^2)*, waar n het aantal systemen is. In het Hub-Spoke model kan men werken met een zogenaamde intermediary berichtformaat, zo heb je voor elk systeem dat wordt toegevoegd twee mappings nodig naar de intermediary taal. Het aantal benodigde mappings voor dit scenario kan worden berekend met de formule: *2*n*, waar n het aantal systemen is. In een situatie waar tien systemen met elkaar moeten kunnen communiceren zijn met de eerste methode 200 mappings nodig, terwijl met de tweede methode 20 mappings voldoende zijn.

In het zogenaamde *Publish-Subscribe* model verstuurt een Publisher berichten naar een centraal systeem, in dit geval

```
Public Function BTSProcessDocument(ByVal strDocument As String) As String

    Dim oInterchange As New BTSInterchangeLib.Interchange
    Dim varResponse

    oInterchange.SubmitSync BIZTALK_OPENNESS_TYPE_NOTOPEN, _
         strDocument, , , , , , , , , , varRespone
    BTSProcessDocument = varResponse

End Function
```

Afbeelding 2. Voorbeeld gebruik van IInterchange::SubmitSync

is dat de BizTalk Server. Het centrale systeem verstuurt het bericht vervolgens door naar de Subscribers: de systemen die aangeven interesse te hebben in alle of sommige van de berichten van het Publisher-systeem. De belangrijkste voordelen van deze methode zijn dat het publicerende systeem geen weet hoeft te hebben van de zich abonnerende systemen, het centrale systeem zorgt voor alle mappings. Bovendien kunnen de Subscribers zich op ieder moment af- en aanmelden voor de berichten waarin zij geïnteresseerd zijn. Nadeel van de oplossing is dat tweewegcommunicatie erg moeilijk is. Publish-Subscribe Integration is niet standaard mogelijk met BizTalk Server. Microsoft biedt echter een gratis toolkit aan die het mogelijk maakt applicaties op deze wijze met elkaar te integreren. De toolkit bevat een aantal componenten, een voorbeeldtoepassing en documentatie die de ontwikkelaar op weg helpt.

Straight Thru Processing

Straight Thru Processing, vaak ook wel kortweg STP genoemd, is momenteel een populair buzzword. In wezen is het niets anders dan synchrone communicatie. Zoals we eerder zagen dient men altijd asynchrone communicatie te verkiezen boven synchrone. Er zijn echter situaties waar een synchrone oplossing vereist is. We kunnen hierbij denken aan toepassingen die een grafische gebruikersinterface ondersteunen of deel uit maken van een kortlopende transactie. Om synchrone communicatie mogelijk te maken moeten zowel aan de ontvangende kant als aan de versturende kant handelingen verricht worden. Dit betekent dat, om een resultaat te kunnen ontvangen, de synchrone versie van de HTTP Receive-functie moet worden gebruikt, of het bericht moet met behulp van de SubmitSync-functie op de IInterchange interface worden verstuurd. Aan de versturende kant moet een port worden geconfigureerd die in staat is antwoord te geven.

Hier zijn nu drie keuzes:

- *Een Loopback port*: in het geval van een synchroon verzoek geeft deze port het bericht dat het ontvangt weer terug aan de aanroeper. Veel mensen vragen zich af wat de toegevoegde waarde is van de Loopback port. Deze is er zeker. Het bericht dat de port uiteindelijk teruggeeft is eerst verwerkt door een Channel. Op deze manier is het dus mogelijk om een Channel met een Loopback port te gebruiken om een bericht te vertalen, valideren, loggen, tracken, decrypten, enzovoort. Zo kan men de functionaliteit van een Channel benutten zonder het bericht te versturen.
- *Een HTTP(S) port*: doet een binaire post op een gespecificeerde URL. Het resultaat van de post, de response van de URL, wordt vervolgens teruggegeven aan de aanroeper.
- *Een Application Integration Component port*: het resultaat van de call naar de AIC-component wordt teruggegeven aan de aanroeper. Het implementeren van een Application Integration Component is eenvoudig. Het behelst niet veel meer dan het maken van een ActiveX DLL; bijvoorbeeld met Visual Basic, die een component bevat dat de IBTSAppIntegration interface implementeert (zie afbeelding 3). Deze interface is trouwens simpel en bestaat slechts uit één functie: ProcessMessage met een parameter voor het binnenkomende bericht en het resultaat kan worden teruggegeven als uitkomst van de functie.

```
Option Explicit

Implements BTSComponentsLib.IBTSAppIntegration

Private Function IBTSAppIntegration_ProcessMessage(ByVal bstrDocu-
ment As String) _
As String

    'Doe hier iets zinnigs met het binnenkomende bericht.
    'Als synchrone communicatie gewenst is geef het resultaat terug
    Dim strResult As String

    strResult = DoSomeThingUseFull(strResult)
    IBTSAppIntegration_ProcessMessage = strResult

End Function
```

Afbeelding 3. Voorbeeld AIC in Visual Basic

BizTalk Server Toolkit

Afsluitend wil ik de lezers graag nog attenderen op de gratis te downloaden BizTalk Server Toolkit for Microsoft .NET. Ondanks het feit dat Microsoft BizTalk Server deel uitmaakt van de .NET Server familie, is de API nog geheel gebaseerd op COM. Om het leven van de programmeur een stuk gemakkelijker te maken heeft Microsoft een speciale toolkit voor .NET-ontwikkelaars gemaakt. De toolkit bevat .NET wrappers voor interfaces in de BizTalk API, sample code en documentatie. Samenvattend kunnen we stellen dat BizTalk ons een bijzonder flexibel 'messaging platform' biedt. Men kan gebruik maken van verschillende integratiemodellen en deze zowel synchroon als asynchroon toepassen. Met behulp van de Microsoft Toolkit voor .NET, de ondersteuning van een breed scala aan protocollen en de beschikbaarheid van meer dan 300 applicaties en protocoladapters kan vrijwel iedere integratieklus met behulp van BizTalk Server worden geklaard.

Nuttige internetadressen

- Microsoft BizTalk Server 2002 Publish-Subscribe Toolkit:
  http://msdn.microsoft.com/downloads/sample.asp?url=/MSDN-FILES/027/001/910/msdncompositedoc.xml
- Microsoft BizTalk Server 2002 Toolkit for Microsoft .NET:
  http://msdn.microsoft.com/downloads/sample.asp?url=/MSDN-FILES/027/001/870/msdncompositedoc.xml
- Meer dan 300 BizTalk Adapters:
  http://www.microsoft.com/biztalk/evaluation/adapters/adapterslist.asp
- BizTalk Server op MSDN:
  http://msdn.microsoft.com/biztalk
- BizTalk Server Product Homepage:
  http://www.microsoft.com/biztalk