Peter Wegbrands en Ed van Akkeren zijn beiden werkzaam bij Getronics PinkRoccade

# Wachten op antwoorden

## BIZTALK SERVER EN CORRELATIE

Dit artikel gaat over elektronische correlatie en hoe we dat met behulp van BizTalk kunnen bewerkstelligen. Na een kort voorbeeld bekijken we hoe correlatie in de praktijk wordt gebruikt bij convoy-processing, waarbij de verschillende vormen hiervan kort aan bod komen. Hierna werken we een voorbeeld in algemene zin uit, zodanig dat het als een blauwdruk kan dienen voor eigen toepassingen op dit gebied.

In de film *Sommersby* komt de man van Jodie Foster na jaren terug uit de oorlog. Maar is Richard Gere wel de man die zij ooit uitzwaaide? In de dagelijkse administratie van ons bedrijf staan we vaak voor dezelfde vragen. Is de betaling die we ontvangen wel die van die verzonden factuur? Is die partij goederen die we ontvangen wel wat we via die inkooporder besteld hebben? Gelukkig hebben wij iets wat Jodie Foster moest ontberen: een unieke identificator. Natuurlijk is die betaling voor de juiste factuur: het factuurnummer staat immers erbij vermeld. Natuurlijk is die partij goederen de bedoelde: het inkoopordernummer stond toch op de pakbon? We hadden immers iets waarop we kunnen correleren.

### Een praktijkgeval

Bedrijf A stuurt regelmatig orders naar bedrijf B. Die orders worden verwerkt en bedrijf B stuurt een factuur terug. Deze moeten door A weer worden gekoppeld aan de originele order. In BizTalk lossen we dit op door middel van correlation in een orchestration. Via een instance van een orchestration zal een order worden verstuurd naar bedrijf B. De orchestration zal vervolgens wachten op een door bedrijf B gestuurd antwoord in de vorm van een factuur. Het kan hierbij zo zijn dat meer orders zijn verstuurd en dat er daarmee meer instances van die orchestration zijn die op een factuur wachten. Hoe weet BizTalk nu welke instance moet worden gekoppeld aan de binnenkomende factuur? Antwoord: door gebruik te maken van correlation. Hierbij zijn twee zaken van belang, te weten correlation types (definitie van hoe we een bericht kunnen identificeren) en correlation sets (een variabele van het correlatie type die wordt gevuld met een waarde waarop kan worden gecorreleerd). In het volgende scenario wordt het eerder genoemde beschreven praktijkgeval uitgewerkt. In de werkelijke wereld zal een en ander waarschijnlijk met behulp van SOAP (of HTTP) worden uitgevoerd, maar in ons voorbeeld zullen we het voor de duidelijkheid met het transporttype FILE doen. Ook zullen de schema's in de werkelijke wereld natuurlijk iets uitgebreider (en complexer) zijn, maar het gaat hier om de illustratie.

Open een BizTalk-project en definieer de schema's *Order* en *Factuur* met de volgende structuren zoals aangegeven in afbeelding 1.


Afbeelding 1. Definieer Order en Factuur

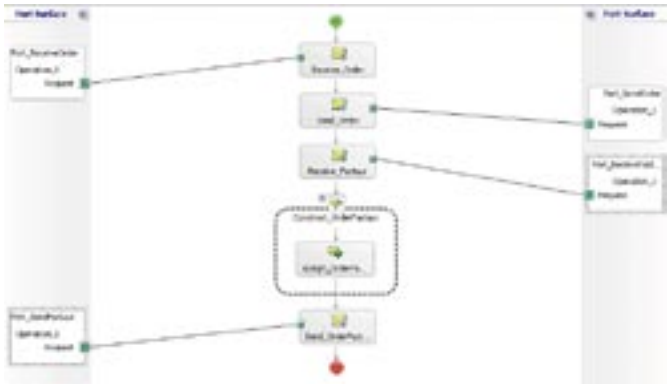
Hierbij dienen de velden *OrderNr* en *FactuurNr* te worden gepromoot, waarmee ze naar een property-schema worden geschreven. Dit laatste is nodig om later te kunnen correleren en een correlation set aan te maken. Genereer nu van elk schema een drietal instanties. In de instanties van *Order*, geef *OrderNr* de waarden 1, 2 en 3 en in de instanties van *Factuur*, geef *FactuurNr* de waarden 1001, 1002 en 1003 en *OrderNr* de waarden 1, 2 en 3. We gaan straks correleren op *OrderNr*. Maak nu een orchestration aan met achter elkaar een receive-shape *Receive_Order*, een send-shape *Send_Order*, een receive-shape *Receive_Factuur*, een message assignment-shape *Assign_OrderFactuur* (met construct-shape *Construct_OrderFactuur*) en een send-shape Send_Order-Factuur. In de Orchestration View, definieer de messages *msgOrder* (op basis van het schema *Order*), *msgOrderFactuur* (ook op basis van het schema *Order*) en *msgFactuur* (op basis van het schema Factuur).

Let op. Als twee messages op basis van eenzelfde schema overbodig lijkt, bedenk dan dat een message die wordt ingelezen niet gewijzigd kan worden en het is nu juist de bedoeling dat we in Order het factuurnummer gaan wegschrijven.

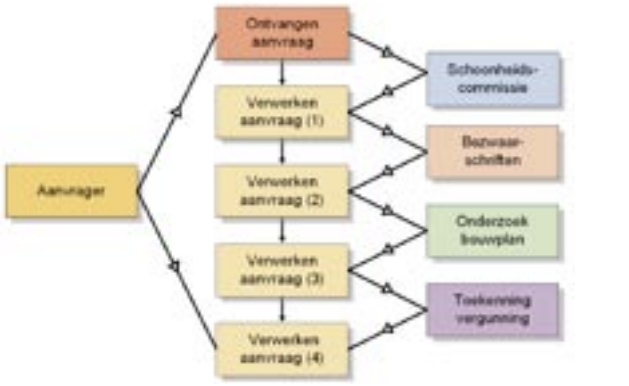
Koppel *msgOrder* aan *Receive_Order* en *Send_Order*, *msgFactuur* aan *Receive_Factuur* en *msgOrderFactuur* aan *Construct_Order-Factuur* en *Send_OrderFactuur*. Definieer in de Orchestration View een correlation type *ctpOrderFactuur* op basis van de promoted property *OrderFactuur.PropertySchema.OrderNr* en definieer daarna een correlation set *corOrderFactuur* op basis van dit correlation-type. Koppel nu *corOrderFactuur* als 'initializing correlation set' aan *Send_Order* en als 'following correlation set' aan *Receive_Factuur*; zie afbeelding 2.

De 'initializing correlation set' houdt het ordernummer bij van het bericht dat werd verstuurd. De 'following correlation set' valideert het binnengekomen ordernummer en koppelt aan de hand hiervan het binnengekomen bericht met het oorspronkelijk verzonden bericht en de daarbij behorende orchestration. De correlation is nu geregeld. We moeten nu nog alleen zorgen dat het factuurnr in

Afbeelding 2. Koppeling tussen order en factuur.



Afbeelding 3. Port-shapes toevoegen




Afbeelding 4. Sequential Convoy

*msgOrderFactuur* wordt geschreven door in de *Assign_OrderFactuur* de expression in codevoorbeeld 1 in te geven.

Voeg nu Port-shapes toe aan de receive- en send-shapes, zodat een orchestration ontstaat als in afbeelding 3. Zorg ervoor dat alle ports naar een andere plek op de schijf wijzen, bijvoorbeeld: *OrderIn*, *OrderOut*, *FactuurIn* en *FactuurOut*.

De orchestration en correlation van dit eenvoudige praktijkvoorbeeld kunnen nu worden getest. Build en deploy het project en plaats de drie instances van *Order* in de *OrderIn*-slot. BizTalk pakt ze daar op en plaatst ze in het *OrderOut*-slot. Kijk nu eerst in de HAT Orchestration debugger: alle drie de messages staan daar keurig te wachten. Plaats nu de tweede instance van *Factuur* in het *FactuurIn*-slot. BizTalk pakt het daar op en in het *FactuurOut*-slot verschijnt een message waarin *Order* 2 wordt gekoppeld aan *Factuur* 1002. Plaats nu de andere instances van *Factuur* in het *FactuurIn*-slot. Ook deze zullen worden gekoppeld aan de juiste *Order*. In de HAT Orchestration Debugger zullen nu de drie messages zijn verdwenen. Eigenlijk is dat alles. We hoeven er dus niet met een of andere service zelf voor te zorgen dat de ontvangen factuur gekoppeld wordt aan de juiste order: dat doet BizTalk voor ons. We hoeven BizTalk door middel van de correlation-set alleen maar te vertellen waar het op moet letten.

## Correlation toegepast: convoys

Correlatie wordt dus gebruikt om berichten aan elkaar te kunnen koppelen. In veel gevallen komt het erop neer dat een binnenkomend antwoord aan een verstuurd bericht wordt gekoppeld en daarmee aan een specifieke orchestration. Dit is precies de toepassing die we hebben gezien in het hierboven beschreven voorbeeld. Er kan echter ook sprake zijn van een veelvoud aan ontvangen antwoorden op een enkel verstuurd bericht, of een aantal verzonden berichten dat leidt tot een enkel antwoord. In dat soort gevallen bieden zogenoemde convoys een oplossing.
Een convoy is een zogenoemde 'messaging pattern'. Het is dus niet BizTalk-specifiek, maar een patroon dat kan worden gebruikt bij bepaalde soorten scenario's. BizTalk kan worden gebruikt om met behulp van correlation een convoy te implementeren. Eigenlijk zou het bovenstaande voorbeeld al kunnen worden gezien als een bepaald soort convoy: enkelvoudig sequentieel. Er wordt gebruik gemaakt van een enkele 'initialising correlation set' voor het verzonden bericht, gevolgd door een enkele 'following correlation set' voor het ontvangen bericht.
In een convoy-patroon zullen we doorgaans meer verzonden of meer ontvangen berichten herkennen. Dit houdt in dat een convoy meer 'initialising'- of 'following'-componenten gebruikt.

```
msgOrderFactuur = msgOrder;
msgOrderFactuur(OrderFactuur.PropertySchema.FactuurNr) =
     msgFactuur(OrderFactuur.PropertySchema.FactuurNr);
```

Codevoorbeeld 1.

Grofweg kunnen we twee soorten convoys onderscheiden:

- Sequential Convoy – hierin worden meer messages sequentieel gekoppeld en wordt aldus een vaste volgorde voor de ontvangst van messages afgedwongen. Hierin is er sprake van een enkele initialising en meer following correlation-sets.
- Parallel Convoy – hierin worden meer messages parallel gekoppeld, wat inhoudt dat er geen vaste volgorde wordt afgedwongen. Hierbij is van tevoren dus niet bekend welk bericht de initialisering correlation-set zal definiëren en welke berichten dus 'following' worden.

## Sequential convoy

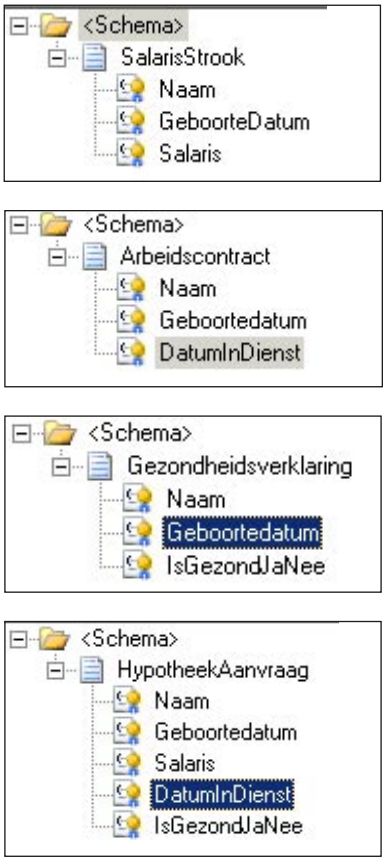
Van een sequential convoy is sprake als twee of meer stappen na elkaar worden afgehandeld, zoals bij de administratieve afhandeling van een aanvraag voor een bouwvergunning bij een gemeente. Elke stap kan pas worden genomen nadat de vorige is afgerond. De vergunning wordt in ontvangst genomen en gaat daarna over een aantal opeenvolgende schijven tot een oordeel wordt geveld; zie afbeelding 4.

In dit voorbeeld wordt een correlation-set aangemaakt op basis van een unieke aanvraag-identificator. Bij het ontvangen van de aanvraag wordt deze geïnitialiseerd ('initializing correlation set'). Bij elk van de daarop volgende verwerkingsslagen wordt de aanvraag opgevolgd ('following correlation set'), totdat een resultaat naar de aanvrager kan worden teruggestuurd. Een dergelijke sequentie kan worden gezien als een kleurplaat: bij elke receive wordt een kleur ingevuld, totdat aan het einde de complete tekening kan worden verstuurd. Functioneel zijn er twee soorten sequential convoys te onderscheiden, te weten de homogene variant, waarin alle inkomende berichten structureel gelijk zijn en de heterogene variant, waarin alle inkomende berichten een andere structuur hebben. Voor de BizTalk-workflow heeft dit geen gevolgen. In het voorgaande voorbeeld is duidelijk sprake van een heterogene sequential convoy.

## Parallel convoy

Van een parallel convoy is sprake wanneer binnen een bepaald proces meer documenten moeten worden ontvangen voordat verder kan worden gegaan met het proces. Hierbij zal gelden dat de berichten in willekeurige volgorde kunnen worden ontvangen. Als voorbeeld kan worden gekeken naar de aanvraag van een hypotheek via een tussenpersoon. De aanvrager zal hierbij de volgende


Afbeelding 5. Parallel convoy




**Afbeelding 6. Message-structuren van salarisstrook, arbeidscontract, gezondheidsverklaring en hypotheekaanvraag**

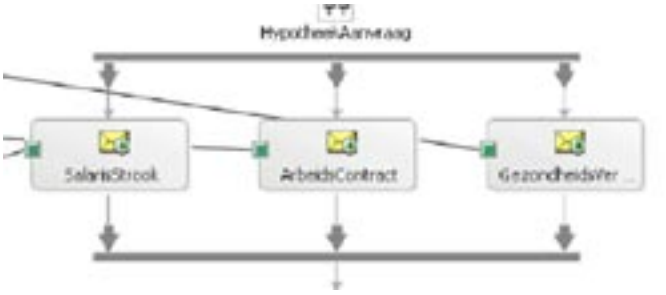
documenten moeten aanleveren voordat de hypotheekaanvraag daadwerkelijk in behandeling kan worden genomen: salarisstrook, arbeidscontract en gezondheidsverklaring. Hierbij geldt dat deze documenten in willekeurige volgorde aangeleverd kunnen worden. Pas als alle documenten binnen zijn kan de aanvraag van de hypotheek verder worden afgehandeld; zie afbeelding 5.

Ook hier kan functioneel weer een onderscheid worden gemaakt tussen een variant waarbij alle messages moeten zijn ontvangen, zoals in het voorgaande voorbeeld, en een variant waarin slechts een enkele message hoeft te zijn ontvangen. Een voorbeeld van dit laatste geval is een inkooporder die verstuurd wordt naar een aantal leveranciers. Als de eerste leverancier heeft gereageerd, dan kunnen de overige worden veronachtzaamd. In tegenstelling tot de varianten van de sequentiële convoy hebben deze varianten wel degelijk gevolgen voor de structuur in BizTalk. We beginnen met een voorbeeld van het eerste geval en komen daarna nog even terug op de andere variant.

## Parallelle convoys in BizTalk

In Biztalk Server kunnen parallelle convoys worden gerealiseerd met behulp van een parallel receive convoy. Voor ons voorbeeld hebben we vier message-structuren nodig, te weten: Salarisstrook, Arbeidscontract, Gezondheidsverklaring en Hypotheekaanvraag, vereenvoudigd als volgt gestructureerd; zie afbeelding 6.

Hierna kunnen we een orchestration aanmaken met daarin messages voor alle structuren (en een inkomende en uitgaande voor *HypotheekAanvraag*) en een correlation-type *ctpHyptheek* bestaande uit *Naam* en *Geboortedatum*, die als basis dient voor de correlation-set *corHypotheek*. De drie binnenkomende messages kunnen in elke gewenste volgorde binnenkomen. Om dit aan te geven hebben we een parallel action-shape nodig, waaronder we drie receive-shapes (voor elke binnenkomende message één) plaatsen; zie afbeelding 7.




**Afbeelding 7. Drie parallelle convoys**

In dit specifieke voorbeeld zal de Parallel Action-shape de eerste actie van de orchestration zijn (dit is echter niet verplicht), zodat de bijbehorende Receive-shape de *activate*-property op 'true' zal moeten hebben staan. Aangezien ieder van de gedefinieerde berichten als eerste bericht binnen kan komen, zal iedere Receive-shape de orchestration moeten kunnen starten en dus zal voor ieder van de receive-shapes de *activate*-property op 'true' moeten worden gezet. Ten slotte laten we iedere receive-shape de aangemaakte correlation-set initialiseren. Wanneer er nu een bericht binnenkomt (dit kan elk van de bovenbeschreven drie zijn) dan zal een nieuwe instance van de orchestration worden opgestart. Er is hier geen behoefte aan een 'following correlation set'. Als er eenmaal een correlation-set is geïnitialiseerd, dan gedragen de andere zich automatisch als volgers. Elk bericht dat hierna binnenkomt voor dezelfde persoon (bijvoorbeeld het arbeidscontract op de gezondheidsverklaring) zal door middel van de correlation automatisch worden gekoppeld aan de al draaiende instance van de orchestration. Pas als alle drie berichten zijn binnengekomen, zal de orchestration verder gaan met de eerste shape (bijvoorbeeld een construct-shape) na de parallel action-shape.
Bij ons voorbeeld met de inkooporders en de leveranciers (de exclusieve parallelle convoy) zit de orchestration iets anders in elkaar. In dit geval is sprake van een enkelvoudige initialisering van de correlation-set als de inkooporder is verstuurd. Voor elk van de leveranciers is er een receive-shape met een 'following correlation set'. Echter, als de eerste leverancier heeft gereageerd en de inkooporder verder is afgewerkt, dan zullen de overige ontvangen messages niet meer gecorreleerd kunnen worden. De message is immers al 'door' en intern volgt een 'subscription not found'-foutmelding, waarna de te laat ontvangen messages verder worden veronachtzaamd. Functioneel is dit natuurlijk wel juist, maar het is wel zo netjes om deze fout als een exception af te vangen en vervolgens in de exception-handler een keurige afmelding terug te sturen naar de leveranciers die net even te laat waren.

## Twee objecten en een paar properties

In dit artikel hebben we in beknopte vorm kunnen zien dat correlation in BizTalk heel eenvoudig is toe te passen. Natuurlijk kan er nog veel meer over gezegd worden, maar het gaat hier om de essentie. Waar bij andere systemen een heleboel complexe code nodig is, volstaat BizTalk met het aanmaken van twee objecten (correlation-type en correlation-set) en het invullen van een paar properties (initialising en following correlation-sets). Ach, was het ICT-leven altijd maar zo simpel.

**Peter Wegbrands** is IT-Designer en **Ed van Akkeren** is product consultant bij Sector Public van Getronics PinkRoccade (www.getronicspinkroccade.nl). Voor vragen en opmerkingen kun je ze bereiken op peter.wegbrands@getronics.com en Ed.vanAkkeren@getronics.com

**Nuttige internetadressen:**

Technische introductie: www.microsoft.com/biztalk/evaluation/introduction.mspx
Development center: msdn.microsoft.com/biztalk/
Virtual lab: msdn.demoservers.com/login.aspx?group=biztalk