Het succes van de Service Oriented Architecture wordt bepaald door de mate waarin we daadwerkelijk herbruikbare services tot stand weten te brengen die meerdere malen losjes gekoppeld gecombineerd kunnen worden in implementaties van bedrijfsprocessen. BPEL – Business Process Execution Language – wordt wel de 'SQL van SOA' genoemd, om aan te geven welke cruciale rol deze standaardtaal speelt bij de beschrijving van bedrijfsprocessen als een georkestreerde collectie van service-aanroepen.

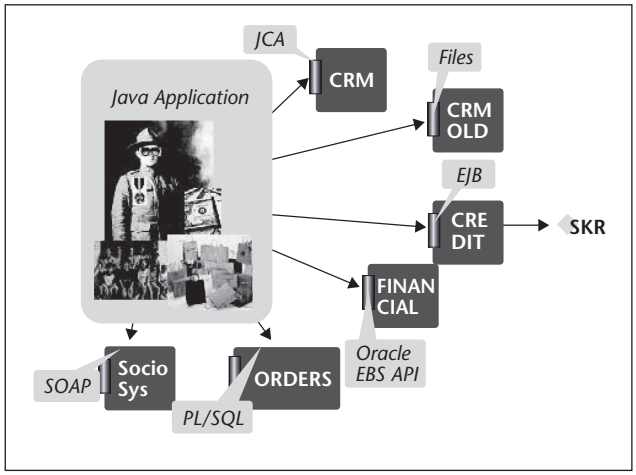
Een veelbelovend huwelijk

Wat Java kan doen voor BPEL

Er is inmiddels een flink aantal BPEL-containers beschikbaar dat in staat is om de gewenste combinatie van technologie-overschrijdende service aanroepen, vaak noodzakelijke menselijke taken ('workflow'), beperkte datamanipulatie en eenvoudige flow-logica (beslispunten, iteraties, parallelle paden) uit te voeren. Hiermee is BPEL en daarmee SOA voor vrijwel iedere organisatie binnen handbereik gekomen. In dit artikel gaan we in op de rol die SOA en BPEL kunnen spelen voor een Java-programmeur, maar vooral ook hoe de Java-ontwikkelaars kunnen bijdragen aan de succesvolle implementatie van BPEL-processen. In enkele voorbeelden in dit artikel wordt gebruik gemaakt van Oracle BPEL Process Manager, maar zijn conceptueel ook van toepassing op de engines van andere leveranciers. Verder kan in enkele gevallen in plaats van een BPEL-engine ook een Enterprise Service Bus worden toegepast.

TOEPASSING Een wel heel voor de hand liggende samenwerking van Java en BPEL is de aanroep van een BPEL-proces vanuit een Java-applicatie. Laten we eens kijken naar een voorbeeld:

Een Java-applicatie heeft behoefte aan een volledig klantbeeld. Dit klantbeeld wordt opgebouwd uit informatie die uit zes verschillende systemen moet worden verzameld. Elk van deze systemen kent een ander soort interface, variërend van JCA en EJB tot PL/SQL en SOAP-webservice. Elk van deze technologieën kan vanuit Java worden benaderd. Het ontwikkelen van de programmacode om alles zes de systemen aan te roepen is echter geen sinecure. Er gaat een hoop tijd en moeite zitten in logica die niet applicatiespecifiek is. Bovendien moet de Java applicatie – evenals iedere andere applicatie die rechtstreeks deze interfaces benaderd – worden aangepast op het moment dat een van de systemen zijn API wijzigt. Tenslotte moeten we als we in de toekomst in een andere applicatie ook het klantbeeld nodig hebben opnieuw code gaan ontwikkelen om die zes systemen te benaderen: het werk dat we daar nu aan doen als we de interactie helemaal zelf in Java gaan ontwikkelen is vrijwel niet te hergebruiken.



FIGUUR 1. Dit klantbeeld wordt opgebouwd uit informatie die uit zes verschillende systemen moet worden verzameld.

Hier kunnen we uitstekend gebruikmaken van BPEL. BPEL (en overigens tot op zekere hoogte ook een Enterprise Service Bus) is zeer geschikt voor het doen van dit soort service-aanroepen, naar allerlei interfaces in diverse technologieën. Bijvoorbeeld Oracle BPEL kent adapters voor alle genoemde interfaces, die bovendien met groot gemak via wizards kunnen worden geconfigureerd. Het ontwikkelen – en te zijner tijd onderhouden - van een BPEL-proces dat het complete klantbeeld oplevert is kortom een simpele activiteit.

Een aanmerkelijk betere performance kan worden bereikt door de BPEL-engine via zijn Java-API te benaderen

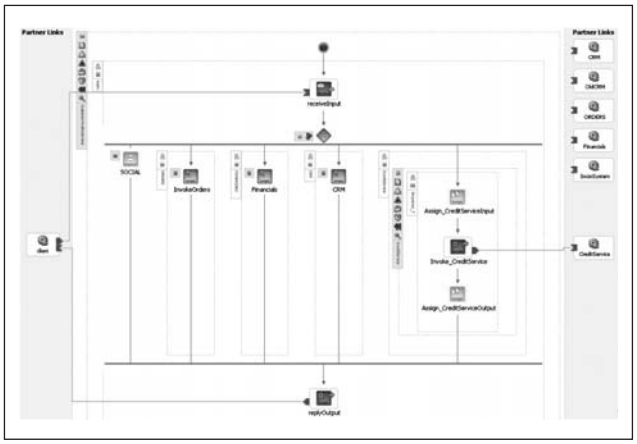
Het aanroepen van dat proces vanuit de Java-applicatie kan op diverse manieren gebeuren. Een voor de hand liggende manier is een aanroep van het BPEL-proces als webservice, via de gebruikelijke SOAP-WS methode, bijvoorbeeld met toepassing van het Apache AXIS framework. Een aanmerkelijk betere performance kan worden bereikt door de BPEL-engine via zijn Java-API te benaderen. De Oracle BPEL PM biedt een EJB Session Bean die door remote clients kan worden benaderd, ondermeer om BPEL Process instanties aan te roepen.

Een klein codefragment met de kern van de zaak:

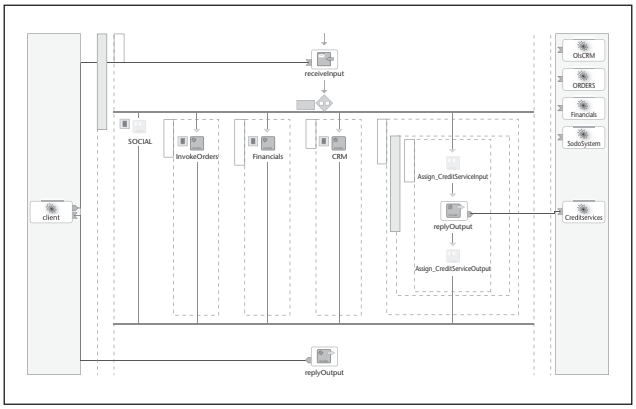
```
         // instantieer een Locator object
met de connectie-details van de RMI Host
        Locator locator = new
Locator("default", "bpel", jndi);
        IDeliveryService deliveryService =
            (IDeliveryService)locator.lookup
Service(IDeliveryService.SERVICE_NAME);

        // construct the normalized message
and send to collaxa server
        NormalizedMessage nm = new
NormalizedMessage();
        String xml =
            "<CustomerProfileServiceProce
ssRequest xmlns=\"http://xmlns.oracle.com/
CustomerProfileService\"><customerId>" +
            customerId +
            "</customerId></CustomerProfileS
erviceProcessRequest> ";
        nm.addPart("payload", xml);

        NormalizedMessage res =
            deliveryService.request("Custome
rProfileService", "process", nm);

        Map payload = res.getPayload();
Element partEl = (Element)payload.
get("payload");
Element contactDetails = (Element)(partEl.get
ElementsByTagName("contactDetails")).item(0);
System.Out.println("Klantnaam: "+contactDe-
tails.getElementsByTagName("name").item(0).
getFirstChild().getNodeValue());
```

BPEL Services smoel geven Java is ook buitengewoon geschikt om de Services die door BPEL-processen worden geïmplementeerd – zoals hier het voorbeeld van de CustomerProfile Service – van een User Interface te voorzien. Een webapplicatie met de juiste invoervelden en een correcte weergave van het resultaat van de service-aanroep is met behulp van moderne IDE's en frameworks in een oogwenk in elkaar gesleuteld. Dit zou al waardevol zijn, al was het alleen maar om services te testen.

Het zou echter een stap verder kunnen gaan: de Portlet-standaarden (JSR-168, WSRP 1.0 en 2.0) bie-



Figuur 2. Een voor de hand liggende manier is een aanroep van het BPEL- proces als webservice.



Figuur 3. De Oracle BPEL PM biedt een EJB Session Bean die door remote clients kan worden benaderd.

FIGUUR 4. Java is ook zeer geschikt om de Services die door BPEL-processen worden geïmplementeerd van een User Interface te voorzien.

den uitzicht op een nieuwe vorm van SOA: 'UI-SOA' waarbij services niet alleen maar business-functionaliteit bieden, maar ook een stukje (inplugbare) user interface. Een UI voor de CustomerProfileService kan geportletized worden en vervolgens eenvoudig worden opgenomen in Portal-frameworks of zelfs 'gewone' webapplicaties, bijvoorbeeld met behulp van de Oracle WebCenter technologie. Omdat portlets binnen een pagina-informatie kunnen uitwisselen, kunnen eenvoudig herbruikbare componenten op basis van webservices zoals BPEL-processen worden ontwikkeld en toegepast.

WAT JAVA KAN DOEN VOOR BPEL Zoals we hebben gezien kan vanuit Java-programmatuur goed gebruik worden gemaakt van BPEL-processen. Omgekeerd kan je vanuit BPEL-processen en vanuit de BPEL-engine vaak nuttig gebruikmaken van in Java ontwikkelde functionaliteit. Hierbij kan je een onderscheid maken tussen de volgende categorieën:

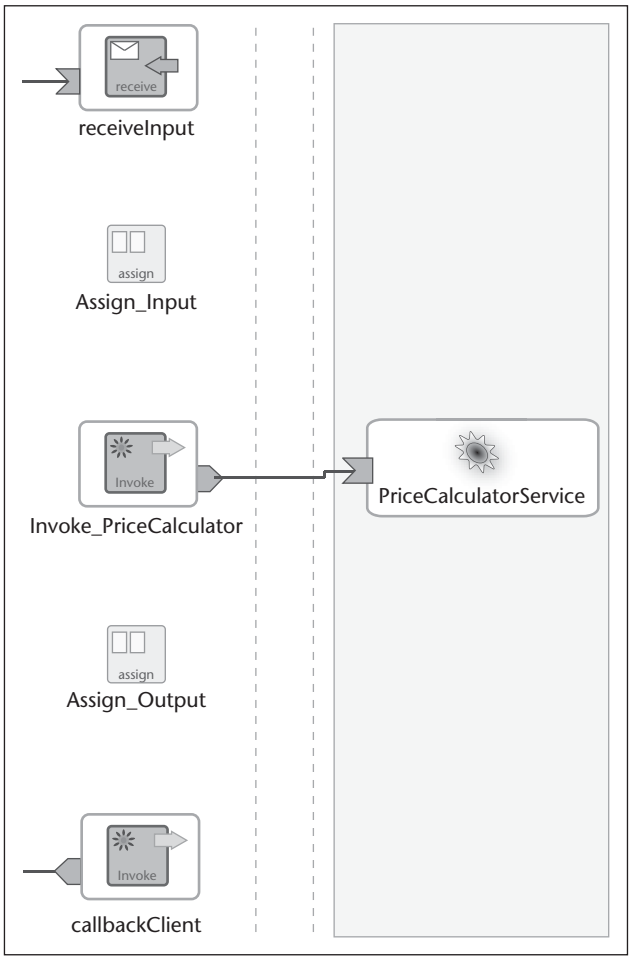
- Aanroepen van in Java ontwikkelde services vanuit een BPEL-proces
- Implementeren van User Interfaces voor de handmatige taken (workflow) in de BPEL-processen
- Verrijken van de BPEL-engine infrastructuur

AANROEPEN VAN JAVA-GEBASEERDE SERVICE Het aanroepen van Services vanuit een BPEL-proces kan via diverse wegen plaatsvinden. Naast de voor de hand liggende aanroep van een Java-gebaseerde service via de SOAP Webservice aanpak – waarbij de Java-service als zodanig moet zijn gedeployed – kan een BPEL ook een EJB (remote of local) aanroepen of een via WSIF-gepubliceerde Java-class. Een eenvoudig voorbeeld van een in Java ontwikkelde service is de volgende class:

```
package nl.amis.sales;

public class PriceCalculator
{
  public PriceCalculator()
  {
  }

  public static float calculateOrderTotal
          ( int itemCount, int warrantyLevel,
String product)
  {
    float warrantyFactor = (float)((Math.
max(warrantyLevel -1,0) ) * 0.02);
    return (float)((getProductPrice(product)*
itemCount)*(1+ warrantyFactor));
  }

  private static float getProductPrice(String
product)
  {
    float price = 99.0f;
    if ("Teddy".equalsIgnoreCase(product))
price=67;
    if ("Boat".equalsIgnoreCase(product))
price=184543;
    if ("Book".equalsIgnoreCase(product))
price=50;
    return price;
  }

}
```

Deze 'service' kan beschreven worden in een WSDL, die via java:bindings ook de vertaling maakt van het WSDL-domein naar de Java-wereld:

```
<definitions name="CalculatePrice"
targetNamespace="urn:CalculatePrice"
             xmlns="http://schemas.xmlsoap.
org/wsdl/"
             xmlns:xsd="http://www.
w3.org/2001/XMLSchema"
             xmlns:tns="urn:CalculatePrice"
             xmlns:plnk="http://schemas.
xmlsoap.org/ws/2003/05/partner-link/"
             xmlns:format="http://schemas.
xmlsoap.org/wsdl/formatbinding/"

             xmlns:java="http://schemas.
xmlsoap.org/wsdl/java/">
  <message name="calculateOrderTotal0Request
">
    <part name="itemCount" type="xsd:
integer"/>
    <part name="warrantyLevel" type="xsd:
```



FIGUUR 5. De PriceCalculator-class moet in het Classpath van de BPEL-engine beschikbaar zijn.

```
integer"/>
    <part name="product" type="xsd:string"/>
  </message>
  <message name="calculateOrderTotal0Respons
e">
    <part name="return" type="xsd:float"/>
  </message>
  <portType name="CalculatePricePortType">
    <operation name="calculateOrderTotal">
      <input name="calculateOrderTotal0Reque
st"
             message="tns:calculateOrder
Total0Request"/>
      <output name="calculateOrderTotal0
Response"
              message="tns:calculateOrder
Total0Response"/>
    </operation>
  </portType>
  <binding name="JavaBinding" type="tns:
CalculatePricePortType">
    <java:binding/>
    <format:typeMapping encoding="Java"
style="Java">
      <format:typeMap typeName="xsd:string"
formatType="java.lang.String"/>
      <format:typeMap typeName="xsd:integer"
formatType="int"/>
      <format:typeMap typeName="xsd:float"
formatType="float"/>
    </format:typeMapping>
    <operation name="calculateOrderTotal">
      <java:operation methodName="calculateOr
derTotal"/>
      <input name="calculateOrderTotal0
Request"/>
      <output name="calculateOrderTotal0
Response"/>
    </operation>
  </binding>
  <service name="CalculatePrice">
    <documentation>CalculateOrderPrice</
documentation>
    <port name="JavaPort" binding="tns:
JavaBinding">
      <java:address className="nl.amis.sales.
PriceCalculator"/>
    </port>
  </service>
  <plnk:partnerLinkType
name="CalculatePrice">
    <plnk:role name="CalculatePriceService
Provider">
      <plnk:portType name="tns:CalculatePrice
PortType"/>
    </plnk:role>
  </plnk:partnerLinkType>
</definitions>
```

Op basis van dit WSDL-document kan in het BPEL-proces een Partner Link worden gedefinieerd die vervolgens in het proces met een Invoke kan worden benaderd.

ONTWIKKELEN UI'S VOOR WORKFLOW Veel BPEL-processen omvatten naast aanroepen van geautomatiseerde services ook handmatige stappen of workflow. Mensen zijn noodzakelijk, onder meer voor het nemen van beslissingen of het vaststellen van de waarde van bepaalde parameters, voor overleg met andere (menselijke) betrokkenen of het aanroepen van een service die niet via een automatische interface kan worden benaderd.

De integratie van menselijke acties in BPEL-processen wordt inmiddels dusdanig belangrijk geacht dat de grote industrie partijen (IBM, Oracle, SAP) bezig zijn met een workflow gerichte aanvulling op de BPEL standaard onder de naam BPEL4PEOPLE. Op dit moment is een menselijke workflow vaak een aanroep vanuit het BPEL-proces van een Workflow Service. Aan de service worden details overgedragen vanuit het BPEL-proces met betrekking tot de task-allocatie, het escalatiemechanisme en de inhoud van de taak. Vaak wordt de allocatie van een

taak via een email gemeld aan de nieuwe taakeigenaar.

Vervolgens is er behoefte aan schermen waar de eindgebruikers hun 'to-do lijst' kunnen opvragen met een overzicht van alle aan hen toegewezen taken. Ook zijn er schermen nodig waarop een gebruiker een taak kan afhandelen. Oracle BPEL PM komt met een voorgedefinieerde generieke workflow applicatie die deze schermen aanbiedt. In veel gevallen zullen we echter schermen willen aanbieden die meer op maat zijn gemaakt, die de taken specifieker ondersteunen. Veelal zal hiertoe een webapplicatie worden ontwikkeld, bijvoorbeeld met standaard Java-technologie zoals JSF (Java Server Faces), JSP/Spring MVC of een JSP/Struts-combinatie. Ook Java-GUI of een met .NET of andere proprietary-technologie kan worden ingezet voor de ontwikkeling van deze gebruikersinterface.

De workflow service in Oracle BPEL PM is via een EJB Session Bean benaderbaar. Deze bean biedt ondermeer een overzicht van alle openstaande taken van een gebruiker, details van iedere afzonderlijke taak en operaties om een taak te manipuleren – bijvoorbeeld invullen van de resultaten van de taak, bekrachtigen van een beslissing en escaleren of afsluiten van een taak.

De Model-component van de webapplicatie gaat interactie aan met de Session Bean om voor de momenteel ingelogde gebruiker de taken op te vragen – titel, datum, urgentie, type – die nog open staan. Deze collectie kan door de View worden gepresenteerd als tabel met hyperlinks. Het selecteren van een taak start een taakspecifiek scherm met alle taakdetails inclusief de door het BPEL-proces aan de workflow-service overgedragen data. In het scherm kan de gebruiker taakgegevens aanpassen, een commentaar vastleggen, documenten aan de taak attachen en de taakstatus wijzigen. Deze operaties worden via de Session Bean doorgevoerd. Als de taak een eindtoestand bereikt zal de workflow service een callback uitvoeren naar de BPEL-engine om het proces weer in gang te zetten.

VERRIJKEN VAN INFRASTRUCTUUR De laatste belangrijke rol die Java kan spelen met betrekking tot BPEL is het aanvullen en op maat maken van de BPEL-engine zelf. De meeste BPEL-engines kunnen worden geconfigureerd en bieden via die configuratie ook de mogelijkheid om eigen factory's en classes in te pluggen om specifieke taken uit te voeren. Via deze 'hooks' kan je de BPEL-engine meer naar je hand zetten.

De Oracle BPEL Process Manager geeft ons bijvoorbeeld de mogelijkheid om een Identify Service te configureren. Deze wordt door de BPEL-engine aangeroepen om identificatie en autorisatie van gebruikers uit te voeren en gegevens op te vragen over de gebruikereigenschappen, rollen en organisatiehiërarchie. De BPEL-engine heeft ingebouwde voorzieningen om een LDAP-directory of JAAS-provider te benaderen, en biedt daarnaast de mogelijkheid om een eigen mechanisme in te pluggen. In Java programmeren we daartoe een class die

De workflow service in Oracle BPEL PM is via een EJB Session Bean benaderbaar

de BPMIdentityService interface implementeert. Deze class kan op zijn eigen manier – bijvoorbeeld uit een XML-document of een relationeel database schema – de autorisatie data verkrijgen.

Een ander voorbeeld van uitbreiding van de BPEL runtime omgeving is de implementatie van een Dynamic Task Assignment functie. Bij het toekennen van workflow-taken aan gebruikers en groepen kan de Oracle BPEL PM verschillende algoritmes toepassen, zoals meest productief, momenteel minst belast of om-de-beurt. We kunnen door een Java-interface te implementeren onze eigen algoritmes toevoegen, bijvoorbeeld taak toewijzen aan de medewerker die de meeste kennis van het onderwerp heeft – te herkennen aan een bepaalde procesvariabele – of taak toewijzen aan een medewerker in een tijdzone waar het momenteel tussen 10 en 16 uur is. Een voorbeeld van zo'n eigen task assignment service:

```
public synchronized String
getUserAssignment(List usernames, String
realm,

    String[] parameters) throws DynamicAssign
mentException {
   List<String> candidates;
   candidates = getUsersForTopic(parameters[
1]);
   if (candidates.size() == 0) {
       candidates = getUsersForTopic("others
");
       // als er niemand te vinden is, dan
moet gebruiker manus het maar doen
       if (candidates.size() == 0) {
         candidates.add("manus");
       }
   }
   Random generator = new Random();
   int randomIndex = generator.nextInt(
```

```
candidates.size() );
   return candidates.get(randomIndex);
}
```

De methode getUsersForTopic heeft een topic als input parameter en heeft een collectie van Strings met de namen van geselecteerde medewerkers terug.

CONCLUSIE BPEL wordt gebruikt voor de implementatie van processen die bestaan uit een combinatie van service-aanroepen, menselijke acties en enige flow-logica. BPEL-processen worden zijn voor Java-applicaties als services – via SOAP-WS of EJB – benaderbaar. Dit is een heel eenvoudige route voor Java om deel te nemen in de SOA-wereld.

Vanaf de andere kant geredeneerd kan een BPEL-proces gebruikmaken van services die in Java zijn geïmplementeerd – direct via WSIF-binding of meer indirect via EJB- of webservice-interfaces. Daarnaast is Java bij uitstek geschikt om gebruikersinterfaces te ontwikkelen voor de ondersteuning van workflow-stappen binnen BPEL-processen. De API's van de BPEL-engine ondersteunen het opvragen en manipuleren van taken vanuit Java-code.

Tenslotte bieden BPEL-engines 'hooks' die maatwerkaanpassingen en uitbreidingen van de run-time infrastructuur mogelijk maken. Eenvoudige Java componenten kunnen worden geconfigureerd om specifieke taken – zoals gebruiker-autorisatie of workflow-taak-allocatie - op een specifieke manier uit te voeren. Voor Java-ontwikkelaars is het kortom waardevol kennis te maken met de wereld van SOA en BPEL. Om er gebruik van te maken en om er aan bij te dragen.

Lucas Jellema (Oracle ACE) is Technisch Consultant en Expertise Manager bij AMIS Services in Nieuwegein. Hij is een frequent spreker op conferenties en enthousiast schrijver van artikelen en weblog-postings (zie http://technology.amis.nl/blog). Lucas oriënteert zich met name op services (BPEL, ESB) en Java/J2EE-technologie.

_bâtisse

Toe aan een andere baan of freelancen?

Bâtisse is gespecialiseerd in recruitment en bemiddeling van java professionals. Als specialist voorzien we je graag van integer advies en helpen we je bij het vinden van de juiste baan.

Momenteel zoeken we voor diverse opdrachtgevers, uiteenlopend van multinational tot software-producent of consultancy organisatie, **Websphere Portal developers, Java/J2EE developers, Java architecten en Business analysts**.

Daarnaast helpen we freelancers bij het vinden van interessante opdrachten.

Kijk voor meer informatie op
WWW.BATISSE.EU
of bel ons op **020-4234340**

PATCHES *Patches* PATCHES

Sybase iAnywhere versterkt marktpositie mobiele applicaties

Sybase iAnywhere heeft onlangs nieuwe versies geïntroduceerd van M-Business Anywhere, Advantage Database Server en XTNDConnect PC. Met deze productverbeteringen versterkt het bedrijf de marktleiderspositie als leverancier van oplossingen voor het ontwikkelen en gebruiken van mobiele applicaties. In het IDC-rapport Worldwide Mobile Device Management Enterprise 2006-2010 Forecast and 2005 Vendor Shares (IDC #203353, September 2006) staat Sybase iAnywhere voor het vijfde achtereenvolgende jaar op de eerste plaats in IDC1s top 10. In het genoemde rapport voorspelt IDC tot en met 2010 een jaarlijkse marktgroei van 28,8%.

Met versie 6.0 van XTNDConnect PC van Sybase iAnywhere kunnen zowel ontwikkelaars van mobiele oplossingen als ICT-managers het synchroniseren van gebruikersdata verbeteren. Wereldwijd worden ruim 25 miljoen licenties van XTNDConnect PC gebruikt voor het synchroniseren van email berichten, contact- en kalenderinformatie, taken en notities met veelgebruikte PC-applicaties als Microsoft Outlook, Lotus Notes, Novell GroupWise, Lotus Organizer en ACT. Belangrijke nieuwe functies zijn: de eenvoudiger te gebruiken interface, ondersteuning van foto1s bij Outlook contacten, meer mogelijkheden voor maatwerktoevoegingen en bredere taalondersteuning. Verder ondersteunt versie 6.0 datasynchronisatie met een groot aantal nieuwe SmartPhones en PDA's, waaronder de Palm 700 en Sony Ericsson W810 modellen.