



---

# Understanding BizTalk Server 2006

David Chappell, Chappell & Associates

August 2005

## CONTENTS

<b>INTRODUCING BIZTALK SERVER 2006.....</b>	<b>3</b>
WHAT BIZTALK SERVER 2006 PROVIDES .....	3
HOW BIZTALK SERVER 2006 IS USED.....	5
<b>THE BIZTALK SERVER 2006 ENGINE .....</b>	<b>6</b>
CONNECTING SYSTEMS.....	8
<i>Sending and Receiving Messages: Adapters</i> .....	8
<i>Processing Messages: Pipelines</i> .....	9
<i>Choosing Messages: Subscriptions</i> .....	11
DEFINING BUSINESS PROCESSES .....	12
<i>Using Orchestrations</i> .....	12
<i>Using the Business Rules Engine</i> .....	17
MANAGEMENT AND MONITORING.....	18
<i>Installing BizTalk Server 2006</i> .....	18
<i>Creating Scalable Configurations</i> .....	19
<i>Managing Applications</i> .....	20
<i>Reporting On and Debugging Applications: Health and Activity Tracking</i> .....	21
ENTERPRISE SINGLE SIGN-ON .....	22
<b>INFORMATION WORKER TECHNOLOGIES.....</b>	<b>23</b>
BUSINESS ACTIVITY MONITORING .....	24
BUSINESS ACTIVITY SERVICES .....	26
<i>Trading Partner Management</i> .....	27
<i>Business Process Configuration</i> .....	28
<b>BIZTALK SERVER 2006 AND OTHER WINDOWS TECHNOLOGIES.....</b>	<b>28</b>
BIZTALK SERVER 2006 AND WINDOWS WORKFLOW FOUNDATION.....	28
BIZTALK SERVER 2006 AND WINDOWS COMMUNICATION FOUNDATION.....	30
<b>CONCLUSIONS .....</b>	<b>30</b>
<b>ABOUT THE AUTHOR.....</b>	<b>31</b>

## Introducing BizTalk Server 2006

No application is an island. Whether we like it or not, tying systems together has become the norm. Yet connecting software is about more than just exchanging bytes. As organizations move toward a service-oriented world, the real goal—creating effective business processes that unite separate systems into a coherent whole—comes within reach.

BizTalk Server 2006 supports this goal. Like its predecessors, this latest release allows connecting diverse software, then graphically creating and modifying process logic that uses that software. The product also lets information workers monitor running processes, interact with trading partners, and perform other business-oriented tasks.

Built on the foundation of its predecessor, BizTalk Server 2004, this new release will look familiar to anyone who's used this earlier version. The most important new additions in BizTalk Server 2006 are:

- Better support for deploying, monitoring, and managing applications.

- Significantly simpler installation.

- Improved capabilities for Business Activity Monitoring (BAM).

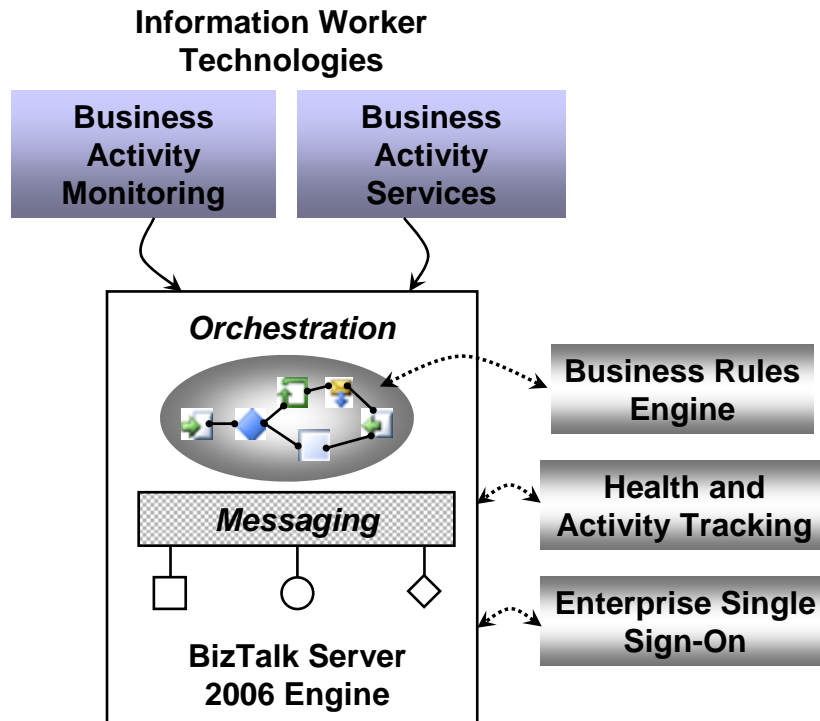
BizTalk Server 2006 also uses the latest releases of other Microsoft technologies. It's built on version 2.0 of the .NET Framework, for example, and its developer tools are hosted in Visual Studio 2005. For storage, the product can use SQL Server 2005, the latest version of Microsoft's flagship database product, or SQL Server 2000, the previous release. BizTalk Server 2006 can also run on 64-bit Windows, taking advantage of the larger memory and other benefits this new generation of hardware offers<sup>1</sup>.

### What BizTalk Server 2006 Provides

Combining different systems into effective business processes is a challenging problem. Accordingly, BizTalk Server 2006 includes a range of technologies. The figure below illustrates the product's major components.

---

<sup>1</sup> This paper is based on the first beta release of BizTalk Server 2006. Some aspects of this technology may change before its final release.



As the figure suggests, the heart of the product is the *BizTalk Server 2006 Engine*. The engine has two main parts:

A *messaging* component that provides the ability to communicate with a range of other software. By relying on pluggable adapters for different kinds of communication, the engine can support a variety of protocols and data formats, including web services and many others.

Support for creating and running graphically-defined processes called *orchestrations*. Built on top of the engine's messaging components, orchestrations implement the logic that drives all or part of a business process.

Several other technologies can also be used in concert with the engine, including:

A *Business Rules Engine* that allows evaluating complex sets of rules.

A *Health and Activity Tracking* tool that lets developers and administrators monitor and manage the engine and the orchestrations it runs.

An *Enterprise Single Sign-on* facility, providing the ability to map authentication information between Windows and non-Windows systems.

On top of this foundation, BizTalk Server 2006 provides a group of technologies that address the more business-oriented needs of information workers. Those technologies are:

*Business Activity Monitoring*, allowing information workers to monitor a running business process. The information is displayed in business rather than technical terms, and what gets displayed can be controlled directly by business people.

*Business Activity Services*, allowing information workers to set up and manage interactions with trading partners.

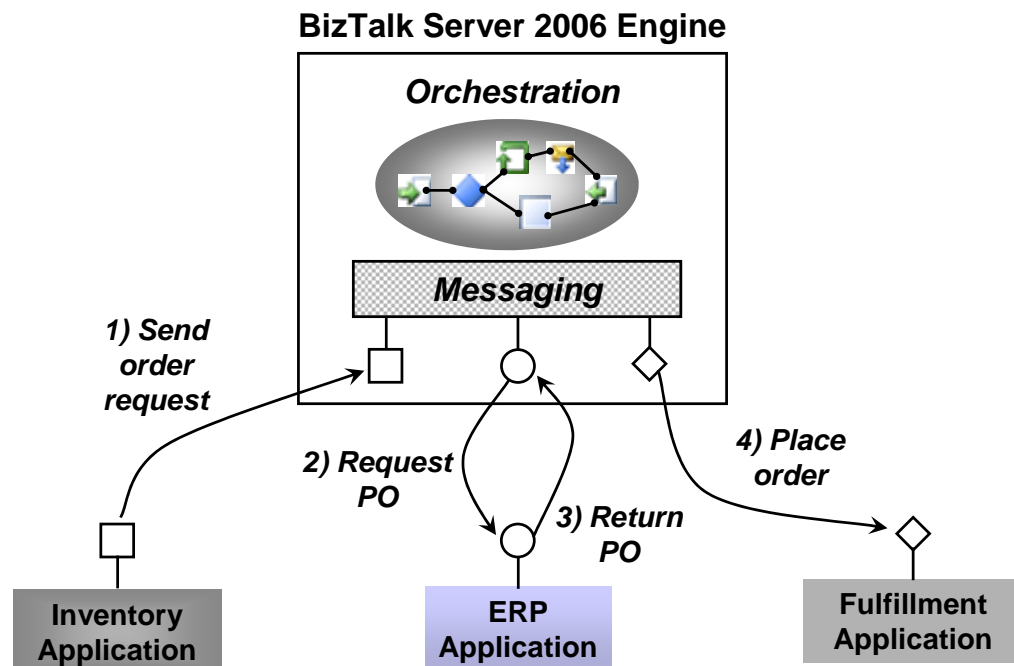
All of these technologies are focused on solving the problems inherent in using a diverse set of software to support automated business processes. The next section examines how these solutions might look.

## How BizTalk Server 2006 Is Used

The great majority of modern business processes depend at least in part on software. While some of these processes are supported by a single application, many others rely on diverse software systems. This software has commonly been created at different times, on different platforms, and using different technologies. Given this, automating those business processes requires connecting diverse systems.

Addressing this challenge goes by various names: business process automation (BPA), business process management (BPM), and others. Whatever it's called, two scenarios are most important for application integration. One is connecting applications within a single organization, commonly referred to as *enterprise application integration (EAI)*. The other, called *business-to-business (B2B) integration*, connects applications in different organizations.

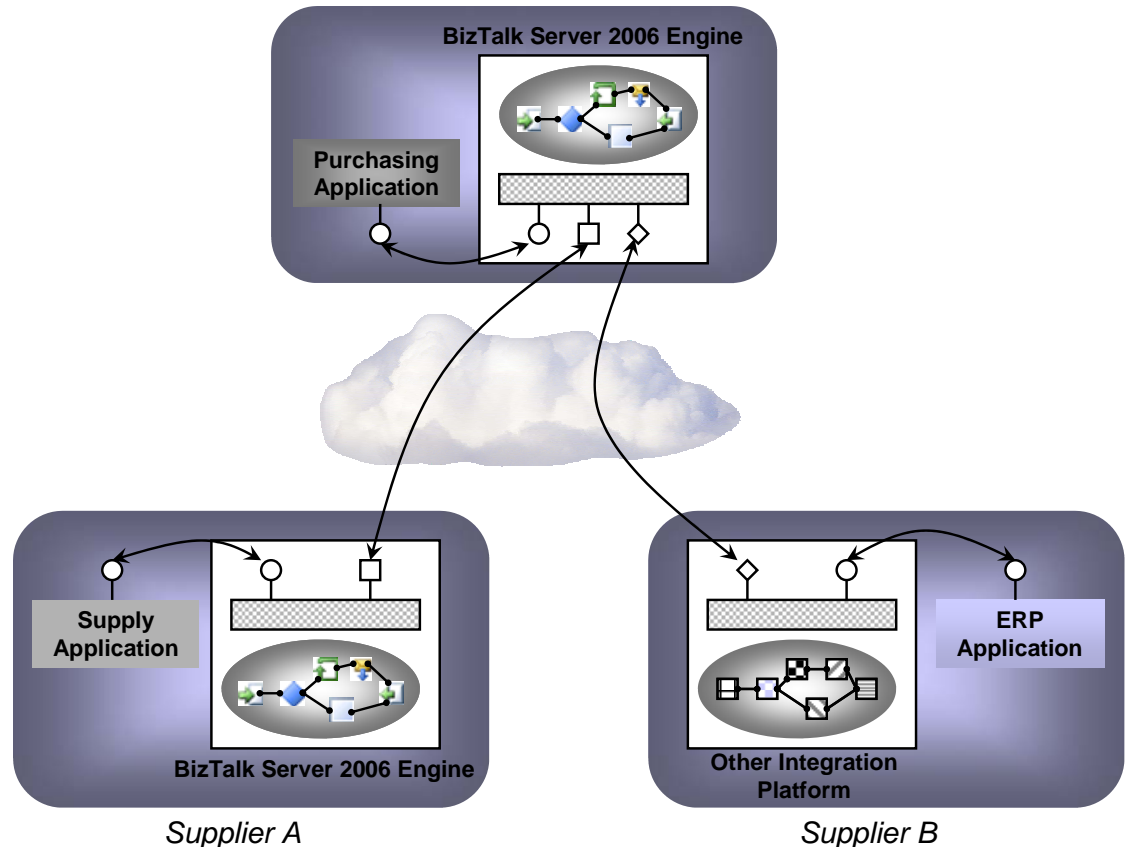
The figure below shows a simple example of the core BizTalk Server 2006 engine applied to an EAI problem. In this scenario, an inventory application, perhaps running on an IBM mainframe, notices that the stock of an item is low and so issues a request to order more of that item. This request is sent to a BizTalk Server 2006 orchestration (step 1), which then issues a request to this organization's ERP application requesting a purchase order (step 2). The ERP application, which might be running on a Unix system, sends back the requested PO (step 3), and the BizTalk Server 2006 orchestration then informs a fulfillment application, perhaps built on Windows using the .NET Framework, that the item should be ordered (step 4).



In this example, each application communicates using a different protocol. Accordingly, the messaging component of the BizTalk Server 2006 engine must be able to talk with each application in its native communication style. Also, notice that no single application is aware of the complete business process.

The intelligence required to coordinate all of the software involved is implemented in the BizTalk Server 2006 orchestration.

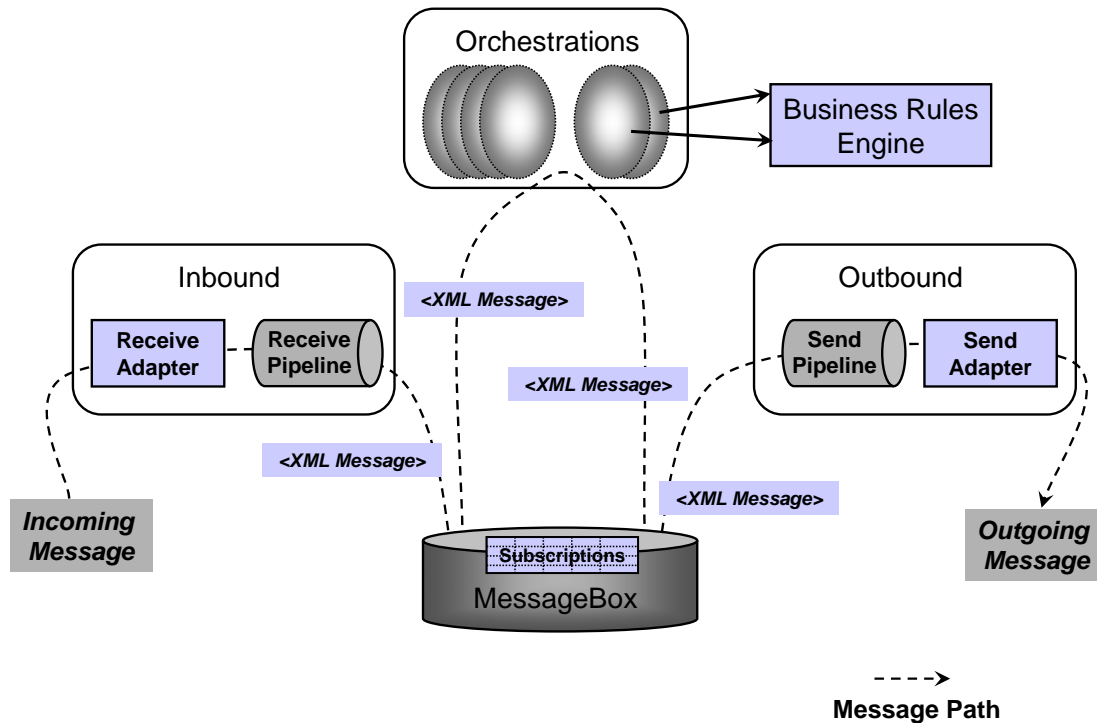
Connecting applications within an organization is important, but connecting applications that span organizations can have at least as much value. The figure below shows a simple example of this kind of B2B integration. In this case, the purchasing organization at the top of the figure runs a BizTalk Server 2006 orchestration that interacts with two supplier organizations. Supplier A also uses BizTalk Server 2006, providing indirect access to its Supply application. Supplier B uses an integration platform from another vendor, connecting to the purchasing organization's BizTalk Server 2006 orchestration using, say, web services.



Integrating existing applications, whether in a single company or across different organizations, into a single automated business process is a fundamental goal of BizTalk Server 2006. Once those automated processes exist, the product also gives business people, not just technicians, visibility into what's happening inside the process. In the complex and diverse world of enterprise software today, this kind of integration is a necessity for many organizations.

## The BizTalk Server 2006 Engine

To allow its users to create a business process that spans multiple applications, the BizTalk Server 2006 engine must provide two primary things: a way to specify and implement the logic driving that business process, and some mechanism for communicating between the applications the business process uses. The figure below illustrates the main components of the engine that address these two problems.



As the diagram shows, a message is received via a *receive adapter*. Different adapters provide different communication mechanisms, so a message might be acquired by accessing a web service, reading from a file, or in some other way. The message is then processed through a *receive pipeline*. This pipeline can contain various components that do things such as converting the message from its native format into an XML document, validating a message's digital signature, and more. The message is then delivered into a database called the *MessageBox*, which is implemented using SQL Server.

The logic that drives a business process is implemented as one or more orchestrations, each of which consists of executable code. These orchestrations aren't created by writing code in a language such as C#, however. Instead, a business analyst or (more likely) a developer uses an appropriate tool to graphically organize a defined group of *shapes* to express conditions, loops, and other behavior. Orchestrations can optionally use the Business Rules Engine, which provides a simpler and more easily modified way to express complex sets of rules in a business process.

Each orchestration creates *subscriptions* to indicate the kinds of messages it wants to receive. When an appropriate message arrives in the *MessageBox*, that message is dispatched to its target orchestration, which takes whatever action the business process requires. The result of this processing is typically another message, produced by the orchestration and saved in the *MessageBox*. This message, in turn, is processed by a *send pipeline*, which may convert it from the internal XML format used by BizTalk Server 2006 to the format required by its destination, add a digital signature, and more. The message is then sent out via a *send adapter*, which uses an appropriate mechanism to communicate with the application for which this message is destined.

A complete solution built on the BizTalk Server 2006 engine can contain various parts (sometimes referred to as *artifacts*): orchestrations, pipelines, message schemas, and more. To allow working with these as a single unit, this release of the product formalizes the notion of a *BizTalk application*. A BizTalk application wraps all of the pieces required for a solution into a single logical unit, making it the fundamental abstraction for management and deployment.

Different kinds of people perform different functions with the BizTalk Server 2006 engine. A business analyst, for example, might define the rules and behaviors that make up a business process. She also determines the flow of the business process, defining what information gets sent to each application and how one business document is mapped into another. Once the business analyst has defined this process, a developer can create a BizTalk application that implements it. This includes things such as defining the XML schemas for the business documents that will be used, specifying the detailed mapping between them, and creating the orchestrations necessary to implement the process. An administrator also plays an important role by setting up communication among the parts, deploying the BizTalk application in an appropriately scalable way, and performing other tasks. All three roles—business analyst, developer, and administrator—are necessary to create and maintain BizTalk Server 2006 solutions.

## Connecting Systems

Effectively exchanging messages across different software on different machines is an absolute requirement for integration. Given the diversity of communication styles that exist, the BizTalk Server 2006 engine must support a variety of protocols and message formats. As described next, a significant portion of the engine is devoted to making this communication work. One important fact to keep in mind, however, is that the engine works only with XML documents internally. Whatever format a message arrives in, it's always converted to an XML document after it's received. Similarly, if the recipient of a document can't accept that document as XML, the engine converts it into the format expected by the target.

### Sending and Receiving Messages: Adapters

Because the BizTalk Server 2006 engine must talk to a variety of other software, it relies on adapters to make this possible. An adapter is an implementation of a communication mechanism, such as a particular protocol. A developer can determine which adapters to use in a given situation. He might choose one of the built-in adapters BizTalk Server 2006 provides, for example, or use an adapter created for a popular product such as SAP, or even create a custom adapter. In all of these cases, the adapter is built on a standard base called the *Adapter Framework*. This framework provides a common way to create and run adapters, and it also allows using the same tools to manage all adapters.

BizTalk Server 2006 includes the following adapters:

**Web Services Adapter:** allows sending and receiving messages using SOAP over HTTP. Since SOAP is the core protocol for web services, this adapter is critical for BizTalk Server 2006's ability to interact in a service-oriented world. As usual with web services, URLs are used to identify the sending and receiving systems.

**File Adapter:** allows reading from and writing to files in the Windows file system. Because the applications involved in a business process can often access the same file system, either locally or across a network, exchanging messages through files can be a convenient option.

**HTTP Adapter:** allows sending and receiving information using HTTP. The BizTalk Server 2006 engine exposes one or more URLs to allow other applications to send data to it, and it can use this adapter to send data to other URLs.

**MSMQ Adapter:** allows sending and receiving messages using Microsoft Message Queuing (MSMQ).

**MSMQT Adapter:** allows sending and receiving messages using *BizTalk Message Queuing (MSMQT)*. MSMQT is an implementation of the MSMQ protocol that can receive and send



MSMQ messages into the MessageBox. While BizTalk Server 2006 still includes this adapter, applications built today should use the MSMQ adapter instead.

WebSphere MQ Adapter: allows sending and receiving messages using IBM's WebSphere MQ (formerly known as MQSeries).

SMTP Adapter: allows sending messages using SMTP. Standard email addresses are used to identify the parties.

POP3 Adapter: allows receiving email messages and their attachments using version three of the Post Office Protocol (POP3).

Windows SharePoint Services (WSS) Adapter: allows accessing and publishing documents stored in SharePoint document libraries.

SQL Adapter: allows reading and writing information to a SQL Server database.

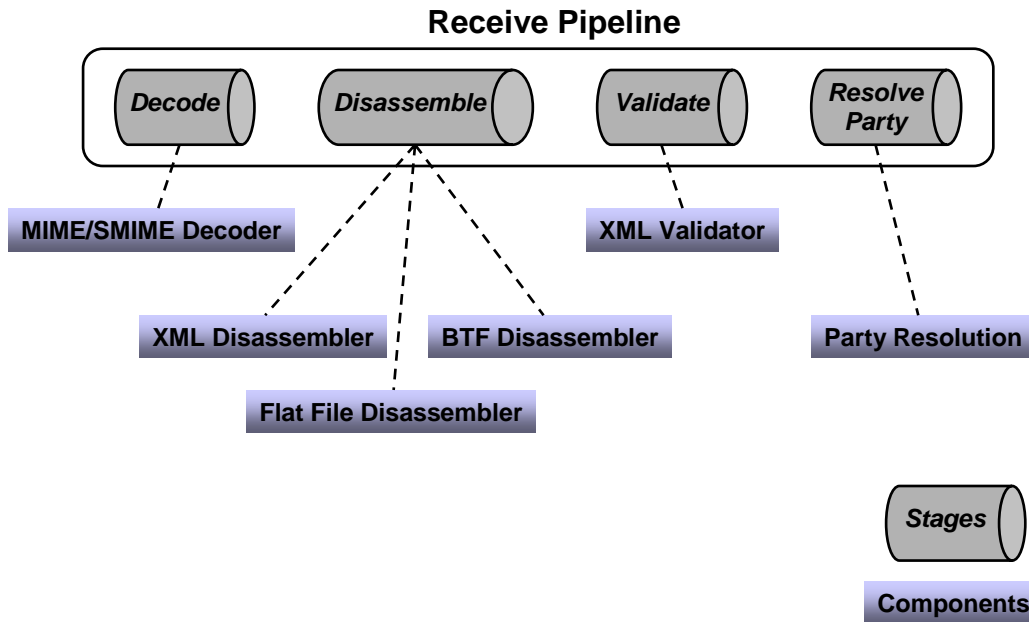
Adapters for commonly-used business software are also available from Microsoft, including adapters for Siebel, PeopleSoft, Oracle applications and Oracle databases, JD Edwards OneWorld and EnterpriseOne, TIBCO Rendezvous and Enterprise Messaging Service, and Amdocs Clarify. Microsoft partners provide still more adapters, including connectors for electronic data interchange (EDI) and others.

Whatever adapter is used to receive data, the messages it gets must commonly be processed before they can be accessed by an orchestration. Similarly, outgoing messages produced by an orchestration often need to be processed before they are sent by an adapter. How both kinds of processing are done is described next.

#### Processing Messages: Pipelines

The applications underlying a business process communicate by exchanging various kinds of documents: purchase orders, invoices, and many more. For a BizTalk Server 2006 application to execute a business process, it must be able to correctly deal with the messages that contain these documents. This processing can involve multiple steps, and so it's performed by a message pipeline. Incoming messages are processed through a receive pipeline, while outgoing messages go through a send pipeline.

For example, even though more and more applications understand XML documents, many—probably the majority today—cannot. Since the BizTalk Server 2006 engine works only with XML documents internally, it must provide a way to convert other formats to and from XML. Other services may also be required, such as authenticating the sender of a message. To handle these and other tasks in a modular, composable way, a pipeline is constructed from some number of *stages*, each of which contains one or more .NET or Component Object Model (COM) components. Each component handles a particular part of message processing. The BizTalk Server 2006 engine provides several standard components that address the most common cases. If these aren't sufficient, developers can also create custom components for both receive and send pipelines.



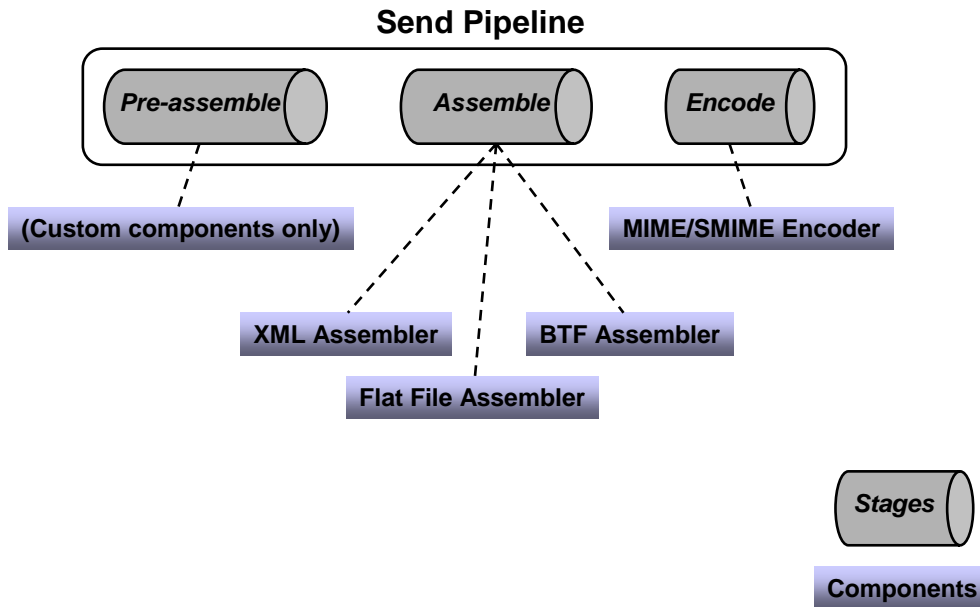
The figure above illustrates the stages in a receive pipeline, along with the standard components provided for each one. Those stages and their associated components are:

**Decode:** BizTalk Server 2006 provides one standard component for this stage, the *MIME/SMIME Decoder*. This component can handle messages and any attachments they contain in either MIME or Secure MIME (S/MIME) format. The component converts both kinds of messages into XML, and it can also decrypt S/MIME messages and verify their digital signatures.

**Disassemble:** Three standard components are provided. The *Flat File Disassembler* component turns flat files into XML documents. Those files can be positional, where each record has the same length and structure, or delimited, with a designated character used to separate records in the file. The second standard component, the *XML Disassembler*, parses incoming messages that are already described using XML. The third standard component, one that's not often used today, is the *BTF Disassembler*. It accepts messages sent using the reliable messaging mechanism defined by the BizTalk Framework (BTF), which was implemented in BizTalk Server 2000.

**Validate:** BizTalk Server 2006 provides an *XML Validator* component for this stage. As its name suggests, this component validates an XML document produced by the Disassemble stage against a specified schema or group of schemas, returning an error if the document doesn't conform to one of those schemas.

**Resolve Party:** the only standard component for this stage, *Party Resolution*, attempts to determine an identity for this message's sender. If the message was digitally signed, the signature is used to look up a Windows identity in BizTalk Server 2006's *Configuration* database. (As described later, this database is also used by BizTalk Server's management tools.) If the message carries the authenticated security identifier (SID) of a Windows user, this identity is used. If neither mechanism succeeds, the message's sender is assigned a default anonymous identity.



Outgoing messages can also go through multiple stages, as defined by the send pipeline in use. The figure above shows the stages and standard components for a send pipeline. They are:

**Pre-assemble:** No standard components are provided. Instead, custom components can be inserted here as needed.

**Assemble:** Paralleling the Disassemble stage in a receive pipeline, this stage also has three standard components. The *Flat File Assembler* converts an XML message into a positional or delimited flat file, while the *XML Assembler* allows adding an envelope and making other changes to an outgoing XML message. The third (seldom used) option, the *BTF Assembler*, packages messages for reliable transmission using the BizTalk Framework messaging technology.

**Encode:** BizTalk Server 2006 defines only one standard component for this stage, the *MIME/SMIME Encoder*. This component packages outgoing messages in either MIME or S/MIME format. If S/MIME is used, the message can also be digitally signed and/or encrypted.

BizTalk Server 2006 defines some default pipelines, including a simple receive/send pair that can be used for handling messages that are already expressed in XML. A developer can also create custom pipelines using the *Pipeline Designer*. This tool, which runs inside Visual Studio 2005, provides a graphical interface that allows dragging and dropping components to create pipelines with whatever behavior is required.

#### Choosing Messages: Subscriptions

Once a message has made its way through an adapter and a receive pipeline, the next problem is to determine where it should go. A message is most often targeted to an orchestration, but it's also possible for a message to go directly to a send pipeline, allowing the BizTalk Server 2006 engine to be used as purely a messaging system. In either case, this matching of messages with their destinations is done via subscriptions.

When a message is processed by a receive pipeline, a *message context* is created that contains various *properties* of the message. An orchestration or a send pipeline can subscribe to messages based on the values of these properties. For example, an orchestration might create a subscription that

matches all messages of the type “Invoice”, or all messages of the type “Invoice” received from the QwickBank corporation, or all messages of the type “Invoice” received from the QwickBank corporation that are for more than \$10,000. However it’s specified, a subscription returns to its subscriber only those messages that match the criteria that subscription defines. A received message might initiate a business process by instantiating some orchestration or it might activate another step in an already running business process. Similarly, when an orchestration sends a message, that message is matched to a send pipeline based on a subscription that pipeline has established.

In BizTalk Server 2006, it’s also possible to subscribe to specific error conditions. Unlike the previous release, in which messages that caused errors were simply suspended, an error message can now be processed in a particular way or routed to a specific destination, such as a WSS folder.

## Defining Business Processes

Sending messages between different systems is a necessary part of solving the problems that BizTalk Server 2006 addresses. Yet most often, it’s only a means to an end. The real goal is to define and execute business processes based on these applications. To define the logic of a business process, the BizTalk Server 2006 engine provides orchestrations. To create and evaluate groups of business rules, it provides the Business Rules Engine. This section describes both.

### Using Orchestrations

The logic of an automated business process can be implemented directly in a language such as C# or Visual Basic. Yet creating, maintaining, and managing complex business processes in conventional programming languages can be challenging. Like its predecessors, then, BizTalk Server 2006 doesn’t take this approach. Instead, it allows creating a business process graphically. Doing this can be faster than building the process directly in a programming language, and it can also make the process easier to understand, explain, and change. Business processes built in this fashion can be monitored more easily, too, a fact that’s exploited by the Business Activity Monitoring technology described later in this paper.

Successfully creating an automated business process usually requires collaboration between software developers and business people. Accordingly, BizTalk Server 2006 provides appropriate tools for each. The developer tools run inside Visual Studio 2005, an environment in which software professionals feel at home. Most business people don’t find Visual Studio especially inviting, however, so BizTalk Server 2006 also provides a subset of the developer tool functionality in an add-in for Visio. Information created in the Visual Studio-based tools can be imported into the Visio-based tools and vice-versa, which helps these two kinds of people work together when creating a business process. Once it exists, this process, defined as an orchestration, is automatically transformed into standard assemblies that run on the .NET Framework.

For a developer, creating an orchestration relies on three primary tools: the *BizTalk Editor* for creating XML schemas, the *BizTalk Mapper* for defining translations between those schemas, and the *Orchestration Designer* for specifying the logic of business processes. All of these tools are hosted inside Visual Studio 2005, providing a consistent environment for developers. This section describes what each of these tools does and how they work together.

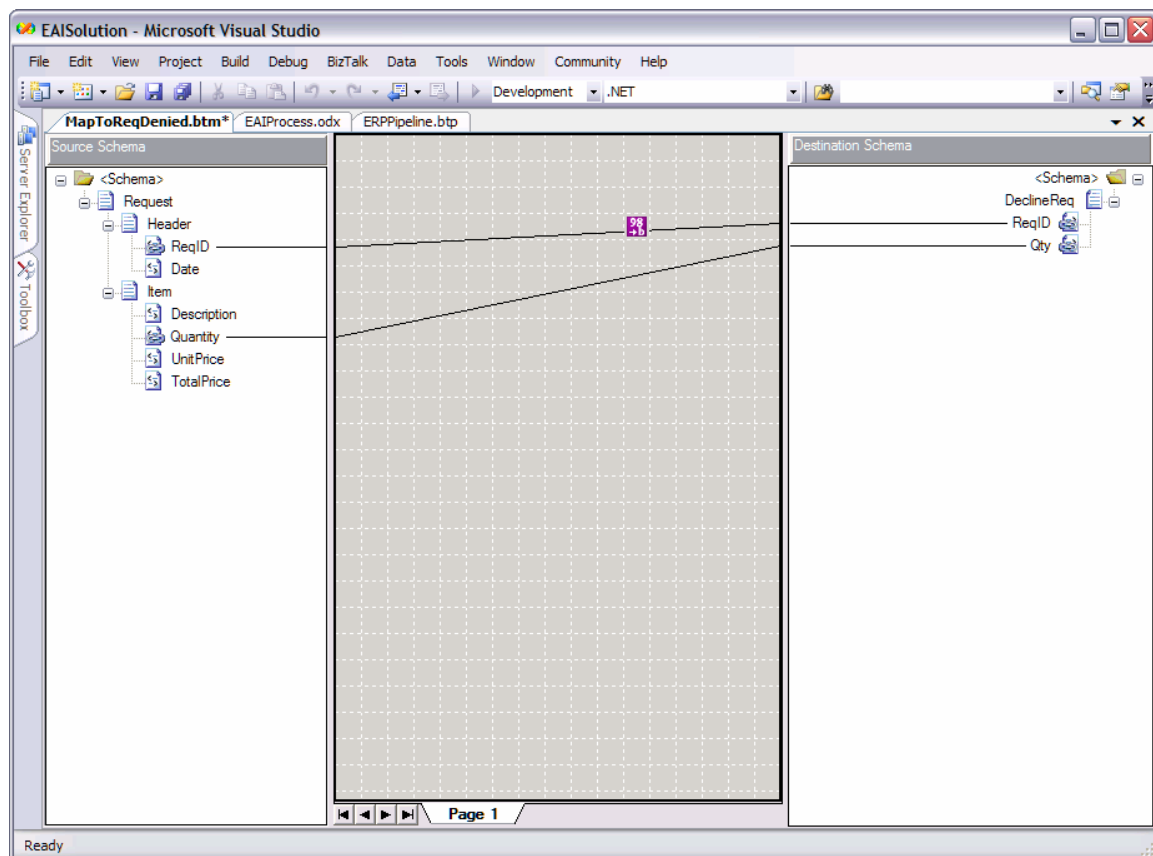
### *Creating Schemas: The BizTalk Editor*

Orchestrations work with XML documents, each of which conforms to some XML schema. Accordingly, there must be a way to define these schemas. To do this, BizTalk Server 2006 provides the BizTalk Editor. This tool allows creating schemas, which are essentially definitions of the structure and types of a document’s information, using the XML Schema Definition language (XSD).

Creating raw XSD schemas without some tool support is not simple. To make this necessary step more approachable, the BizTalk Editor allows its user—probably a developer—to build a schema by defining its elements in a graphical hierarchy. Existing schemas can also be imported from either files or accessible web services. However they're acquired, schemas are used as the basis for BizTalk maps, which are described next.

### *Mapping Between Schemas: The BizTalk Mapper*

An orchestration implementing a business process typically receives some documents and sends others. It's common for part of the information in the received documents to be transferred to the sent documents, perhaps transformed in some way. For example, an order fulfillment process might receive an order for some number of items, then send back a message indicating that the order was declined for some reason. It's possible that information from the order, such as a request identifier and the quantity ordered, might be copied from fields in the received order message into fields in the rejection message. The BizTalk Mapper can be used to define a transformation, called a *map*, from one document to the other.



As the figure above shows, each map is expressed as a graphical correlation between two XML schemas that defines a relationship between elements in those schemas. The W3C has defined the Extensible Stylesheet Language Transformation (XSLT) as a standard way to express these kinds of transformations between XML schemas, and so maps in BizTalk Server 2006 are implemented as XSLT transformations.

The transformation defined in a map can be simple, such as copying values unchanged from one document to another. Direct data copies like this are expressed using a *link*, which is shown in the BizTalk Mapper as a line connecting the appropriate elements in the source schema with their counterparts in the destination schema. More complex transformations are also possible using

*functoids*. A functoid is a chunk of executable code that can define arbitrarily complex mappings between XML schemas, and as shown above, the BizTalk Mapper represents it as a box on the line connecting the elements being transformed. Since some of those transformations are fairly common, BizTalk Server 2006 includes a number of built-in functoids. These built-in functoids are grouped into categories, which include the following:

Mathematical functoids that perform operations such as adding, multiplying, and dividing the values of fields in the source document and storing the result in a field in the target document.

Conversion functoids that convert a numeric value to its ASCII equivalent and vice-versa.

Logical functoids that can be used to determine whether an element or attribute should be created in the target document based on a logical comparison between specified values in the source document. Those values can be compared for equality, greater than/less than, and in other ways.

Cumulative functoids that compute averages, sums, or other values from various fields in the source document, then store the result in a single field in the target document.


Database functoids that can access information stored in a database.


It's also possible to create custom functoids directly in XSLT or using .NET languages like C# and Visual Basic. Functoids can also be combined in sequences, cascading the output of one into the input of another.


Having a way to define a document's XML schema is essential, as is a mechanism for mapping information across documents with different schemas. The BizTalk Editor and BizTalk Mapper address these two problems. Yet defining schemas and maps isn't enough. The business logic that will use the schemas and invoke the maps must also be specified. How this is done is described next.


### *Defining Business Logic: The Orchestration Designer*







A business process is a set of actions that together meet some useful business need. With the BizTalk Server 2006 engine, a developer can use the Orchestration Designer to define these actions graphically. Rather than expressing the steps in some programming language, this tool allows a developer to create an orchestration by connecting together a series of shapes in a logical way. The shapes available to an orchestration's creator include:

 The Receive shape, which allows the orchestration to receive messages. A Receive shape can have a filter that defines exactly what kinds of messages should be received, and it can also be configured to start a new instance of an orchestration when a new message arrives.

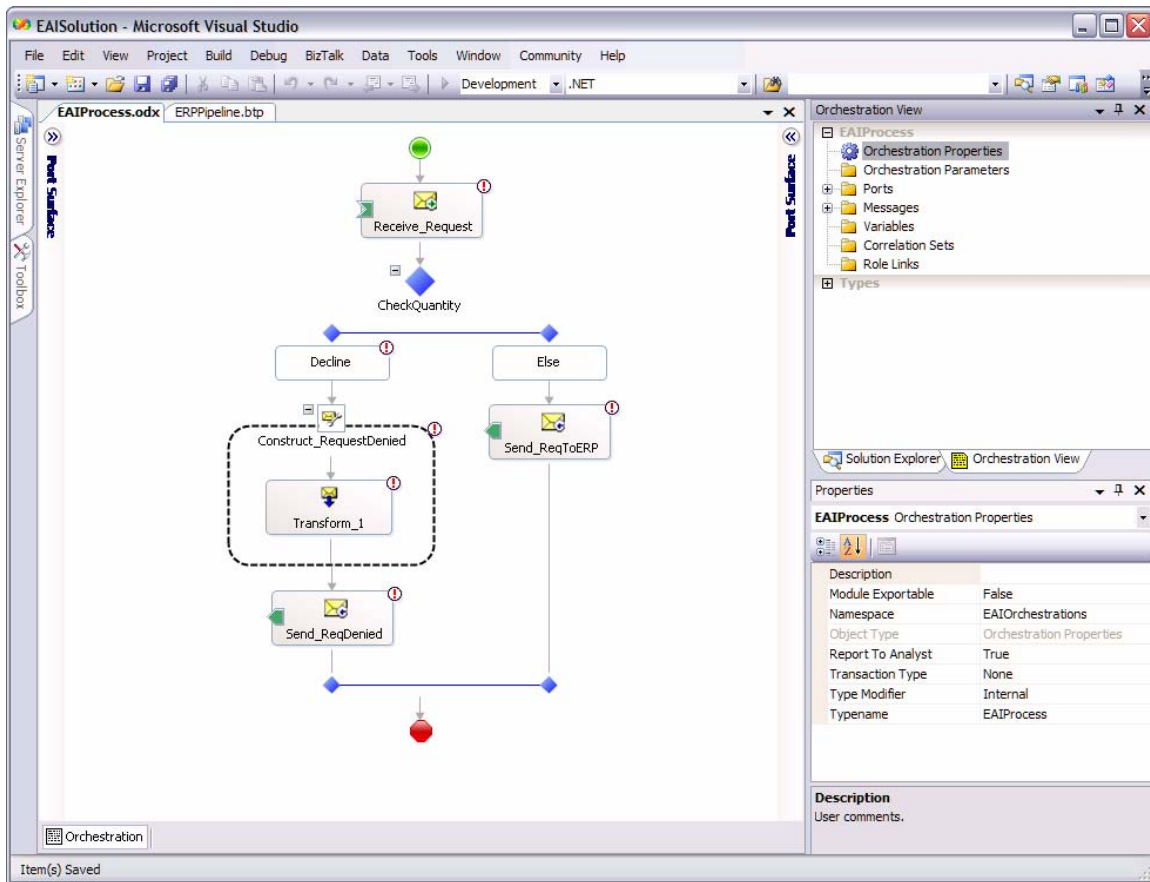
 The Send shape, which allows the orchestration to send messages.

 The Port shape, which defines how messages are transmitted. Each instance of a port shape is connected to either a Send or Receive shape. Each port also has a type, which defines things such as what kinds of messages this port can receive; a direction, such as send or receive; and a binding, which determines how a message is sent or received by, for example, specifying a particular URL and other information.

 The Decide shape, which represents an if-then-else statement that allows an orchestration to perform different tasks based on Boolean conditions. An Expression Editor, part of the Orchestration Designer, can be used to specify this conditional statement.

-  The Loop shape, which allows performing an action repeatedly while some condition is true.
-  The Construct Message shape, which allows building a message.
-  The Transform shape, which allows transferring information from one document to another, transforming it on the way by invoking maps defined with the BizTalk Mapper.
-  The Parallel Actions shape, which allows specifying that multiple operations should be performed in parallel rather than in sequence. The shape that follows this one won't be executed until all of the parallel actions have completed.
-  The Scope shape, which allows grouping operations into transactions and defining exception handlers for error handling. Both traditional atomic transactions and long-running transactions are supported. Unlike atomic transactions, long-running transactions rely on compensating logic rather than rollback to handle unexpected events.
-  The Message Assignment shape, which allows assigning values to orchestration variables. These variables can be used to store state information used by the orchestration, such as a message being created or a character string.

The figure below shows an orchestration created in the Orchestration Designer using a few of these shapes. In this simple example, a message is received, a decision is made based on the content of that message, and one of two paths is executed as a result of that decision. Orchestration diagrams that solve real problems can be significantly more complex than this, of course, and so to help in working with these more complex diagrams, the Orchestration Designer in BizTalk Server 2006 provides the ability to zoom in and out. This lets a developer view only those parts of an orchestration that she's currently interested in.



Once a developer has defined an orchestration, the group of shapes and relations between them is converted into the Microsoft Intermediate Language (MSIL) used by the .NET Framework's Common Language Runtime (CLR). Ultimately, the group of shapes defined by a BizTalk Server 2006 developer becomes just a standard .NET assembly. And of course it's still possible to add explicit code to an orchestration when necessary by calling a COM or .NET object from inside a shape.

Web services allow applications to exchange XML documents via SOAP, and they've had a big impact on integration platforms. To access an external web service, an orchestration's creator can use the Add Web Reference option in Visual Studio 2005 along with the Web Services adapter to directly invoke operations. Similarly, BizTalk Server 2006 provides a *Web Services Publishing* wizard that can generate an ASP.NET Web Service project exposing one or more of an orchestration's operations as SOAP-callable web services. These two options allow developers to both access existing web services from within a business process and expose an orchestration's functionality as a web service to other business processes.

The rise of web services is also having an impact on how business processes are defined. For example, think about the case where two organizations interact using web services. To interoperate effectively, it might be necessary for each side of the interaction to know something about the business process the other is using. If both organizations use BizTalk Server 2006, this isn't a big problem; tools such as the Trading Partner Management technology described later in this paper can be used to distribute this knowledge. But what if they're not? Suppose one organization uses BizTalk Server 2006 and the other uses software from another vendor? For cases like this, it's useful to have a way to describe some aspects of business processes in a cross-vendor way.



To allow this, Microsoft, IBM, and others have created the Business Process Execution Language (BPEL). A business process defined using the Orchestration Designer can be exported to BPEL, and BizTalk Server 2006 can also import processes defined in BPEL. While the language is useful for describing and sharing externally visible parts of a business process, it's important to realize that BPEL is focused more on solving this problem than on cross-platform execution of complete business processes. It's also important to understand that BPEL is built entirely on web services, while BizTalk Server 2006 and other products that support this language provide more. For example, BizTalk Server 2006 supports mapping between different XML schemas, calling methods in local objects, and other features that aren't available in BPEL. For these and other reasons, BPEL isn't a complete language for defining business processes. And given that BPEL is still in the process of being standardized by the Organization for the Advancement of Structured Information Standards (OASIS), it's hard to view it today as a fully mature technology.

Like the other developer tools provided by BizTalk Server 2006, Orchestration Designer runs inside Visual Studio 2005. In some cases, however, a business analyst rather than a developer may wish to graphically define a business process. Since business analysts aren't likely to be comfortable using Visual Studio, BizTalk Server 2006 also includes an add-in for Visio that allows defining a business process, then importing this definition into Orchestration Designer.

Orchestrations are the fundamental mechanism for creating business processes in BizTalk Server 2006. Some aspects of an orchestration tend to change more often than others, however. In particular, the decisions embedded in a business process—the business rules—are commonly its most volatile aspect. A manager's spending limit was \$100,000 last week, but her promotion bumps this up to \$500,000, or a slow-paying customer's maximum allowed order decreases from 100 units to only 10. Why not provide an explicit way to specify and update these rules? This is exactly what's done by the Business Rules Engine, as described next.

#### Using the Business Rules Engine

The Orchestration Designer, together with the BizTalk Editor and the BizTalk Mapper, provide an effective way to define a business process and the rules it uses. It's sometimes useful, though, to have an easier way to define and change business rules. To allow this, BizTalk Server 2006 provides the Business Rules Engine (BRE). Developers will most often use the BRE, but it's also possible for more business-oriented users to create and modify sets of business rules using a tool called the *Business Rule Composer*.

One situation in which the BRE is useful is when a complex set of business rules must be evaluated. Deciding whether to grant a loan, for example, might entail working through a large set of rules based on the customer's credit history, income, and more. Similarly, determining whether to sell life insurance to an applicant depends on a number of things, including the applicant's age, gender, and a myriad of health factors. Expressing all of these rules as conditional statements using, say, an orchestration's Decide shape might be possible, but it's not simple. For rule-intensive processes like these, the BRE can make a developer's life significantly simpler.

The BRE can also make changing rules faster and easier. To see why, think about what's required to change a business rule that's implemented within an orchestration. A developer must first open the orchestration in Visual Studio 2005, modify the appropriate shapes (and perhaps the .NET or COM objects they invoke), then build and deploy the modified assembly. Doing this also requires stopping and re-starting the BizTalk application that includes this orchestration. If instead this business rule is implemented using the BRE, it can be modified without recompiling or restarting anything. All that's needed is to use the Business Rule Composer to change the desired rule, then redeploy the new set of rules. The change will take effect immediately. And while orchestrations are typically created and maintained by developers, business rules are readable enough that in some cases they can be modified by business analysts without the need to involve more technical people.

The creator of a set of business rules will typically begin by using the Business Rule Composer to define a *vocabulary* for use in specifying those rules. Each term in the vocabulary provides a user-friendly name for some information. For example, a vocabulary might define terms such as Number Shipped or Maximum Quantity of Items or Approval Limit. Each of these terms can be set to a constant or be mapped to a particular element or attribute in some XML schema (and thus in an incoming message) or to the result of a SQL query against some database or even to a value in a .NET object.

Once a vocabulary has been defined, the Business Rule Composer can be used to create *business policies* that use this vocabulary. Each policy can contain one or more business rules. A rule uses the terms defined in some vocabulary together with logical operators such as Greater Than, Less Than, Is Equal To, and others to define how a business process operates. A business rule can define how values contained in a received XML document should affect the values created in an XML document that's sent, or how those received values should affect what value is written in a database, or other things.

Imagine, for instance, a simple vocabulary that defines the term Maximum Allowed Order Quantity, whose value is set to 100, and the term Quantity Requested, whose value is derived from a specified element in received XML documents that correspond to the schema used for placing orders. A business analyst might create a rule stating that if the Quantity Requested in an incoming order is greater than the Maximum Allowed Order Quantity, the order should be rejected, perhaps resulting in an appropriate XML document being sent back to the originator of this order.

To execute a business policy, an orchestration uses a CallRules shape. This shape creates an instance of the BRE, specifies which policy to execute, then passes in the information this policy needs, such as a received XML document. The BRE can also be invoked programmatically via a .NET-based object model, which allows it to be called from applications that don't use the BizTalk Server 2006 engine. This means that Windows Forms applications, software exposing web services, and anything else built on the .NET Framework can potentially use the BRE whenever it helps solve the problem at hand<sup>2</sup>.

Both vocabularies and business rules can be much more complicated—and much more powerful—than the simple examples described here. But the core idea of defining a vocabulary, then defining sets of rules that use that vocabulary is the heart of the Business Rules Engine. The goal is to provide a straightforward way for BizTalk Server 2006 users of all kinds to create and work with the rules that define business processes.

## Management and Monitoring

Whatever it does, every application built on the BizTalk Server 2006 engine requires management. How are new applications installed? What configurations are possible? What's happening inside the system right now? This section looks at the tools provided to answer these questions.

### Installing BizTalk Server 2006

BizTalk Server 2006 includes a number of components, and it depends on multiple aspects of the Windows environment. Making sure that the correct version of everything the product needs is available, then installing all of the product's components correctly could potentially be a challenging process. In fact, it was a challenging process with BizTalk Server 2004, the product's previous release.

In BizTalk Server 2006, however, installation is straightforward. Upgrading from BizTalk Server 2004 is automatic, and all items built for this earlier version—orchestrations, maps, and so on—will work unchanged in the 2006 version. To ensure that the right environment exists, an administrator doing a

---

<sup>2</sup> The BRE is licensed as part of BizTalk Server 2006, however.

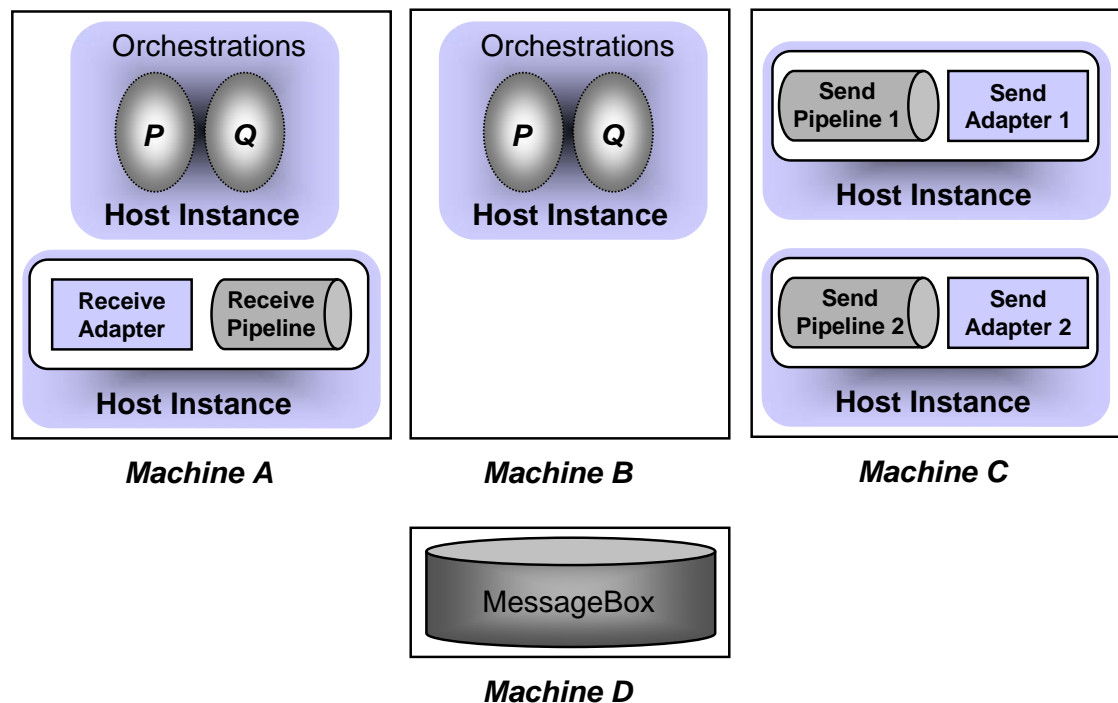
new installation of BizTalk Server 2006 can download a standard .CAB file or reference an already available .CAB file that was downloaded earlier. In either case, this Microsoft-provided file contains the redistributable components the product requires to install and run. These include things such as the correct versions of Microsoft Data Access Components (MDAC), MSXML, the latest security hot fixes, and other necessary software.

Once the contents of this .CAB file have been installed, there are two main options for installing BizTalk Server itself. The default approach, typical of what a developer creating a BizTalk Server 2006 environment for her own use might do, installs all of the product's components under a single account on one machine. Once the process is started, the developer can just watch while these components are installed. An administrator setting up a production BizTalk Server 2006 environment, by contrast, can use the custom configuration option. This choice allows deploying the product to different machines, defining and using different accounts, and other more detailed configurations.

#### Creating Scalable Configurations

If the BizTalk application using it isn't too large, the entire BizTalk Server 2006 engine can be installed on a single machine. Yet it's not hard to imagine situations where this isn't the right solution. Perhaps the number of messages the engine must handle is too great for one machine, or maybe redundancy is required to make the system more reliable. To meet requirements like these, the BizTalk Server 2006 engine can be deployed in a number of ways.

A fundamental concept for deploying the engine is the idea of a *host*. A host can contain various things, including orchestrations, adapters, and pipelines. Hosts are just logical constructs, however. To use them, a BizTalk Server 2006 administrator must cause actual *host instances* to be created. Each host instance is a Windows process, and as the diagram below shows, it can contain various things. In the example shown here, Machine A is home to two host instances. One contains a receive adapter and receive pipeline, while the other contains the orchestrations P and Q. Machine B runs just one host instance, also containing the two orchestrations P and Q. Machine C, like machine A, is home to two host instances, but neither of them contains an orchestration. Instead, each of these instances contains a different send pipeline and send adapter. Finally, machine D houses the MessageBox database that's used by all of the host instances in this configuration.



This example illustrates several ways in which hosts might be used. For instance, since both machines A and B are home to the orchestrations P and Q, BizTalk Server 2006 can automatically load balance requests to these orchestrations based on the availability and current load on each machine. This allows a BizTalk application to scale up as needed for high-volume processes. Notice also that machine C contains two different ways to handle outgoing messages. Perhaps one relies on a standard BizTalk Server 2006 adapter, such as the HTTP adapter, while the other uses a custom adapter to communicate with a particular system. Grouping all output processing on a single machine like this can make good sense in some situations. And because each host instance is isolated from every other host instance—they're different processes—it's safer to run code that's not completely trusted, such as a new custom adapter, in a separate instance. It's also worth pointing out that even though this example contains only a single instance of the MessageBox database, it's also possible to replicate or cluster it to avoid creating a single point of failure.

The abstraction of BizTalk applications introduced in BizTalk Server 2006 isn't intrinsically associated with hosts. For a simple BizTalk application, all of its components might be contained in a single host, with all of them installed on the same machine. In a more complex case, however, the various artifacts that make up the application—orchestrations, adapters, pipelines, and more—might be spread across multiple hosts on multiple machines, as in the figure above. Accordingly, the process of mapping these artifacts to physical machines doesn't depend on the notion of a BizTalk application.

### Managing Applications

The main tool for managing the BizTalk Server 2006 engine is the *BizTalk Administration console*, a Microsoft Management Console (MMC) snap-in that provides a new user interface for BizTalk Server 2006 administrators. While this new tool gives administrators a number of capabilities, the most important are the ability to do three things:

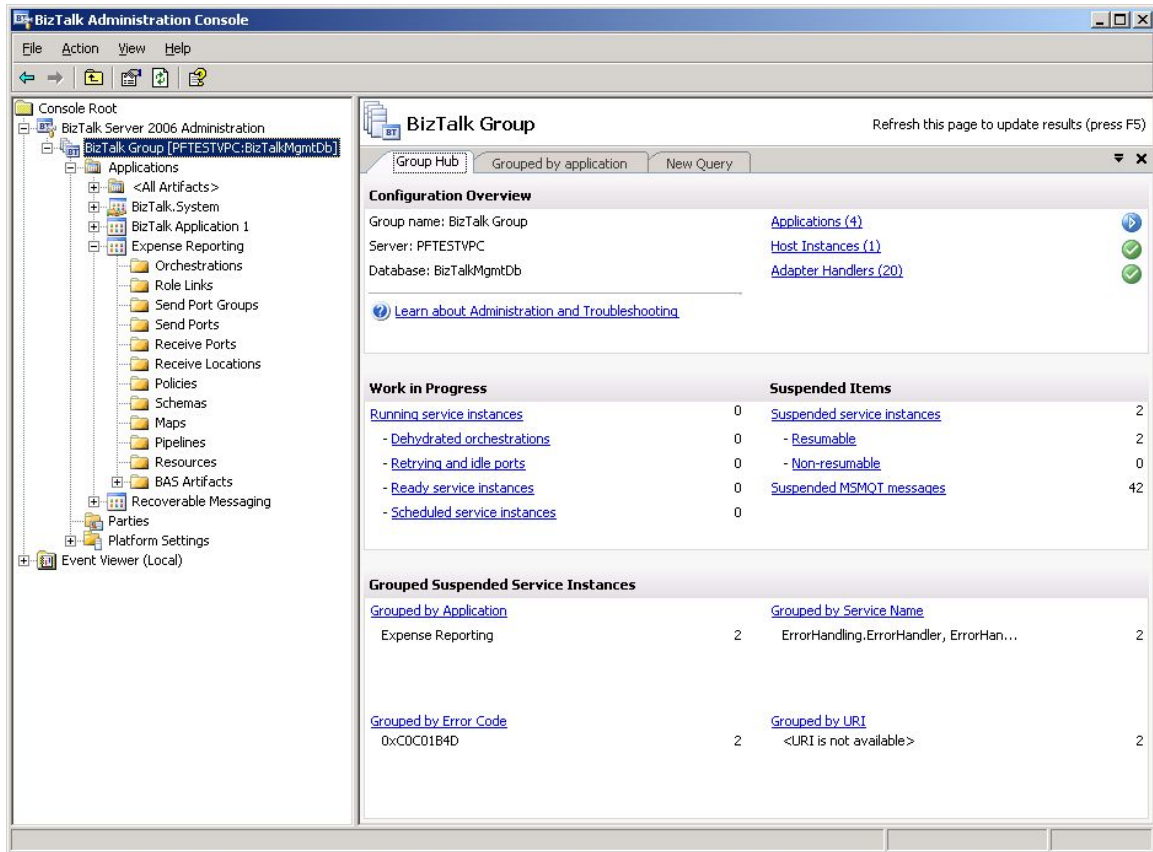
Deploy BizTalk applications. Unlike BizTalk Server 2004, which didn't have a well-defined way to wrap together all of the parts in a solution, BizTalk Server 2006 lets administrators work with a complete BizTalk application as a unit. Using the BizTalk Administration console, an admin can create a BizTalk application, deploy it to one or more servers, and more.

Configure BizTalk applications. When a developer creates an orchestration, she works largely in logical terms. To define how the BizTalk Server 2006 engine will communicate with a particular application, for example, the developer can select an HTTP adapter without worrying about the specific URL that will be used. Similarly, she can specify that the send pipeline should include a component that adds a digital signature to outgoing messages without worrying about exactly what key will be used to create this signature. Yet to make the application work, these details must be specified. The BizTalk Administration console allows an admin to create and modify configurations like these<sup>3</sup>.

Monitor BizTalk applications. Using the BizTalk Administration console's Group Hub page, an admin can monitor the operation of BizTalk applications. As the example below shows, information about the current status of these applications can be examined in various ways. Rather than requiring an administrator to search for problems, for example, the Group Hub page uses color-coded indicators to display those problems. This lets administrators take a more proactive approach to application monitoring.

---

<sup>3</sup> The BizTalk Explorer was used for this kind of configuration in BizTalk Server 2004. While the Explorer is still supported, all of its functions can now be performed using the BizTalk Administration console.



The BizTalk Administration console, which relies on BizTalk Server 2006's Configuration database, also provides other services. An administrator can dynamically add machines and specify what hosts should be assigned to them while an application is running, for example. There's no need to shut down the application to make these changes. The Administration console's functions can also be accessed programmatically through Windows Management Instrumentation (WMI), which allows administrators to create scripts that automate management functions.

### Reporting On and Debugging Applications: Health and Activity Tracking

BizTalk applications do lots of things: send and receive messages, process those messages within orchestrations, communicate with various systems using different protocols, and more. Keeping a record of what's going on, especially when failures occur, is very useful. Similarly, having some way to debug orchestrations and other application components is essential. Both of these things are provided by the *Health and Activity Tracking (HAT)* component of BizTalk Server 2006.

The HAT tool provides graphical access to information about applications running on the engine. This information can include when an orchestration starts and ends, when each shape within it is executed, when each of its messages is sent and received, what's in those messages, and more. A developer or administrator can even set breakpoints, allowing the orchestration to be stopped and examined at pre-determined places. The HAT tool can also be used to examine archived data, looking for patterns and trends in the execution of a business process. This information is useful for debugging, answering business questions (such as verifying that a message really was sent to a customer), and keeping ongoing statistics that can be used to improve performance.

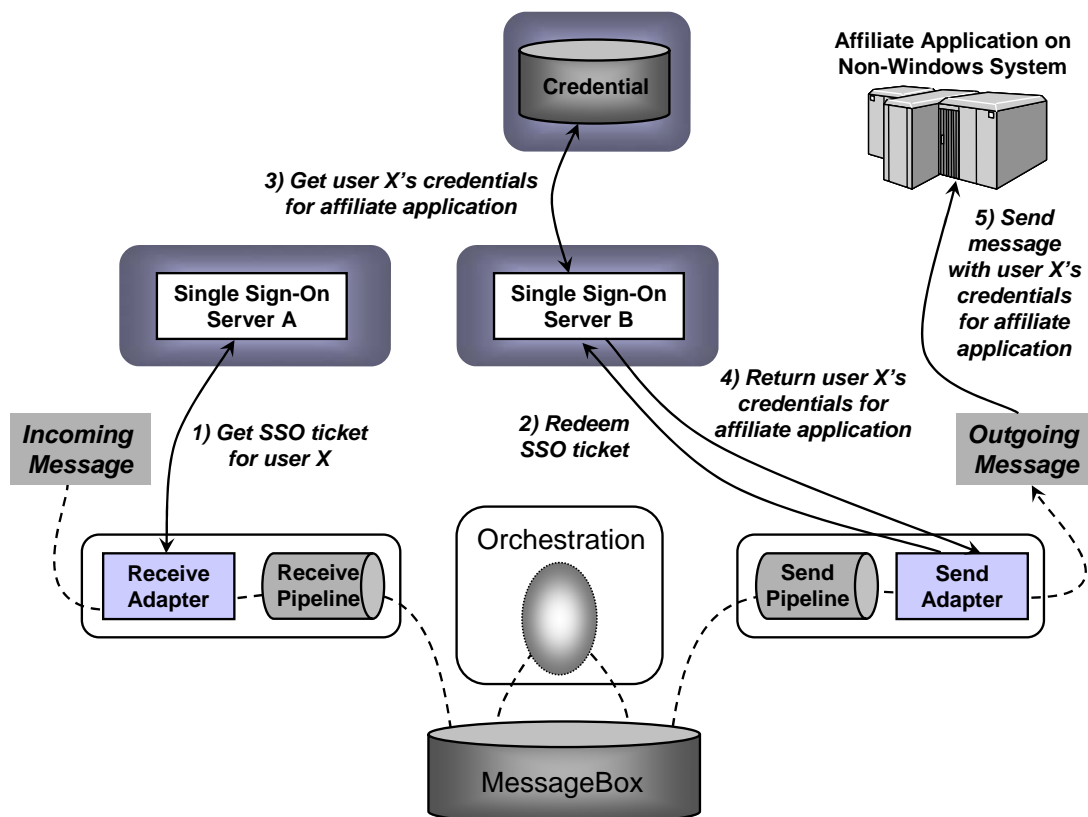
## Enterprise Single Sign-On

A business process that relies on several different applications is likely to face the challenge of dealing with several different security domains. Accessing an application on a Windows system may require one set of security credentials, while accessing an application on an IBM mainframe may require different credentials, such as a RACF username and password. Dealing with this profusion of credentials is hard for users, and it can be even harder for automated processes. To address this problem, BizTalk Server 2006 includes *Enterprise Single Sign-On*.

Don't be confused—this isn't a mechanism that lets people have one login for all applications. Instead, Enterprise Single Sign-On provides a way to map a Windows user ID to non-Windows user credentials. It won't solve all of an organization's enterprise sign-on problems, but this service can make things simpler for business processes that use applications on diverse systems.

To use Enterprise Single Sign-On, an administrator defines *affiliate applications*, each of which represents a non-Windows system or application. For example, an affiliate application might be a CICS application running on an IBM mainframe, an SAP ERP system running on Unix, or any other kind of software. Each of these applications has its own mechanism for authentication, and so each requires its own unique credentials.

Enterprise Single Sign-On stores an encrypted mapping between a user's Windows user ID and his credentials for one or more affiliate applications in a *credential* database. When this user needs to access an affiliate application, his credentials for that application can be looked up in the Credential database by a Single Sign-On (SSO) Server. The diagram below shows how this works.



In this example, a message sent by some application to BizTalk Server 2006 is processed by an orchestration, then sent to an affiliate application running on an IBM mainframe. The job of Enterprise Single Sign-On is to make sure that the correct credentials (e.g., the right username and password) are sent with the message when it is passed to the affiliate application.

As the diagram shows, when a receive adapter gets a message, the adapter can request an SSO *ticket* from SSO server A (step 1). This encrypted ticket contains the Windows identity of the user that made the request and a timeout period. (Don't confuse this with a Kerberos ticket—it's not the same thing.) Once it's acquired, the SSO ticket is added as a property to the incoming message. The message then takes its normal path through the BizTalk Server 2006 engine, which in this example means being handled by an orchestration. When this orchestration generates an outgoing message, that message also contains the SSO ticket acquired earlier.

This new message is destined for the application running on an IBM mainframe, and so it must contain the appropriate credentials for this user to access that application. To get these credentials, the send adapter contacts SSO server B (step 2), supplying the message (which contains the SSO ticket) it just received and the name of the affiliate application it wishes to retrieve the credentials for. This operation, called *redemption*, causes SSO server B to verify the SSO ticket, and then look up this user's credentials for that application (step 3). SSO Server B returns those credentials to the send adapter (step 4), which uses them to send an appropriately-authenticated message to the affiliate application (step 5).

Enterprise Single Sign-On also includes administration tools to perform various operations. All operations performed on the credential database are audited, for example, so tools are provided that allow an administrator to monitor these operations and set various audit levels. Other tools allow an administrator to disable a particular affiliate application, turn on and off an individual mapping for a user, and perform other functions. There's also a client utility that allows end users to configure their own credentials and mappings. And like other parts of BizTalk Server 2006, Enterprise Single Sign-On exposes its services through a programmable API. The creators of third-party BizTalk Server adapters use this API to access the single sign-on services, and administrators can use it to create scripts for automating common tasks.

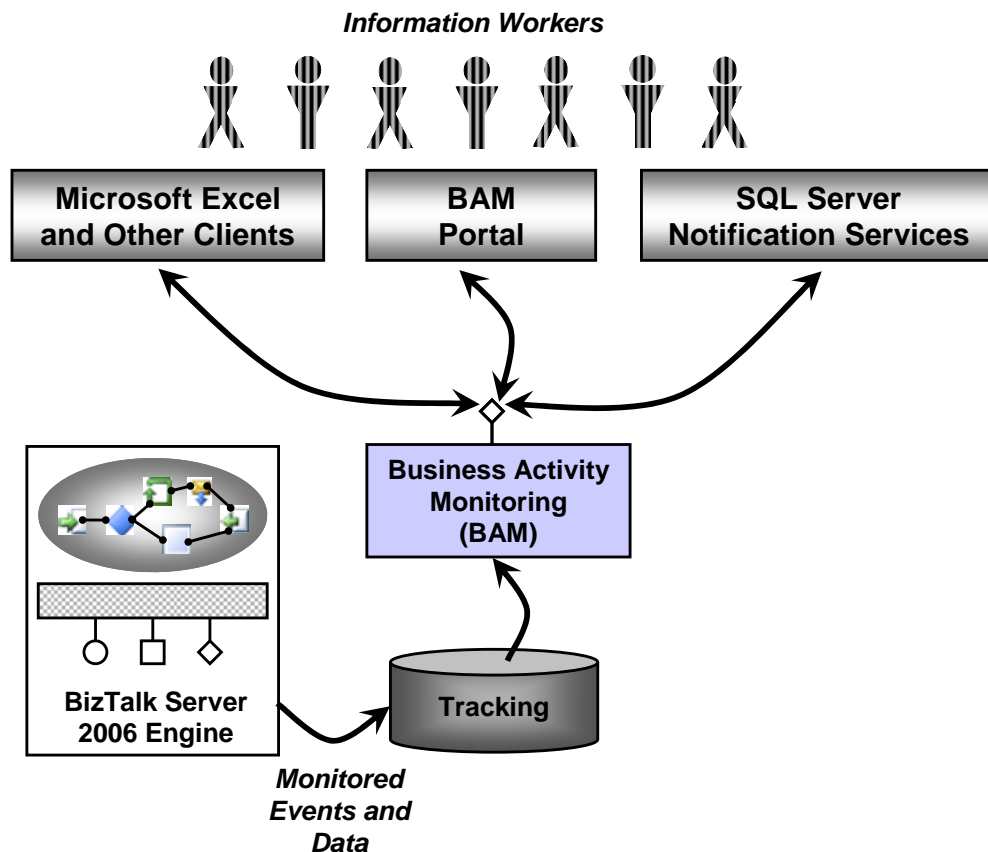
The example described above shows a typical use of Enterprise Single Sign-On, but it's not the only option. A smaller BizTalk Server 2006 installation may have only a single SSO server, for example, and it's even possible to use Enterprise Single Sign-On independently from the BizTalk Server 2006 engine. (In fact, the technology also ships with Microsoft's Host Integration Server product.) Because business processes implemented using BizTalk Server 2006 need to interact with diverse applications, including this component as part of the product makes good sense.

## Information Worker Technologies

The BizTalk Server 2006 engine provides support for automated business processes that span multiple applications. But once those processes have been created, the information workers that use them—business people, not developers—have other requirements. They might need to monitor various business-related aspects of the process, for example, or work with trading partners in various ways. BizTalk Server 2006 provides several components that address these problems. Those components are grouped into two categories: *Business Activity Monitoring* and *Business Activity Services*. This section describes both.

## Business Activity Monitoring

It's not hard to think of different ways that an information worker might want to look at a business process. A purchasing manager might need to see how many POs are approved and denied each day, for instance, while a sales manager might want an hourly update on what products are being ordered. Meeting these diverse needs requires a general framework for tracking what's going on with a particular business process. This is exactly what the *Business Activity Monitoring (BAM)* component in BizTalk Server 2006 provides.



As the figure above illustrates, the BAM component allows monitoring of events and data produced by a BizTalk application. This information is made accessible via SOAP-callable web services, and it can be accessed in several different ways, including:

Through Microsoft Excel or other desktop clients, such as a custom dashboard application.

Via a *BAM portal*, a new component in BizTalk Server 2006 that allows examining and configuring BAM information. Using the BAM portal, an information worker can select a particular instance of some business process, then choose a specific *BAM view* into the process. Each of these views can give a different perspective, such as graphical depictions of per-product sales trends or current inventory levels or other key performance indicators. The information in these views might be updated every day, every hour, or more frequently. Using the BAM portal, an information worker can also define aggregations of data, such as the number of orders filled, canceled, or in progress over the last hour. Implemented as a set of

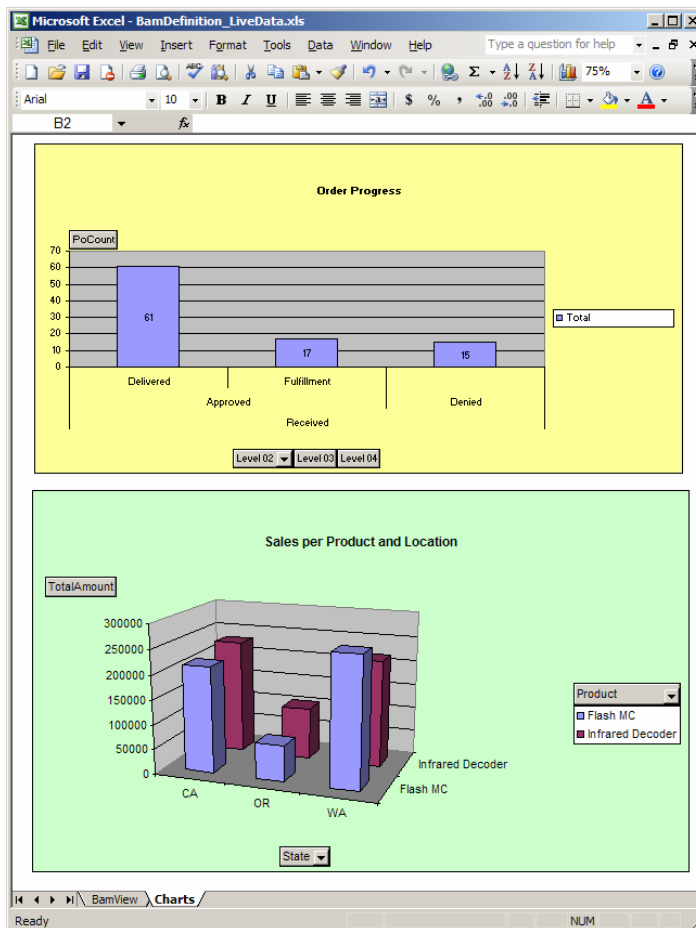


ASP.NET pages, the BAM portal can also be hosted as a web part inside Windows SharePoint Services.

Through SQL Server Notification Services, allowing BAM information to be delivered as notifications. While the first two options allow information workers to examine BAM information, this third option lets them be notified when something interesting happens. Using the BAM portal's alert manager, an information worker can define alerts that should be sent when an interesting event occurs. For example, a BAM user might choose to send an email to a particular manager whenever the number of cancelled orders in a day exceeds ten, or perhaps inform a certain sales associate any time an order arrives from her largest customer.

Under the covers, each BAM view relies on one or more *BAM activities*. A BAM activity represents a specific business process, such as handling purchase orders or shipping a product, and each one has a defined set of *milestones* and *business data*. For example, a purchase order activity might have milestones such as Approved, Denied, and Delivered along with business data like Customer Name and Product.

For information workers accessing BAM through Excel, BAM activities and BAM views can be created using an Excel add-in. This add-in's *BAM Activity* wizard allows defining activities, while its *BAM View* wizard allows defining views based on those activities. In fact, the BAM View wizard really just helps an information worker build a standard Excel pivot table using the information in one or more BAM activities. The information this view provides can then be shown directly via Excel, as the figure below illustrates.



In this simple example, two Excel charts display information about order progress and sales. A BAM view can also be more complex than this, and its creator can control which users are allowed to see the data it exposes. Maybe a purchasing manager can access certain things in a view into the purchase order process, for instance, that are hidden from purchasing clerks.

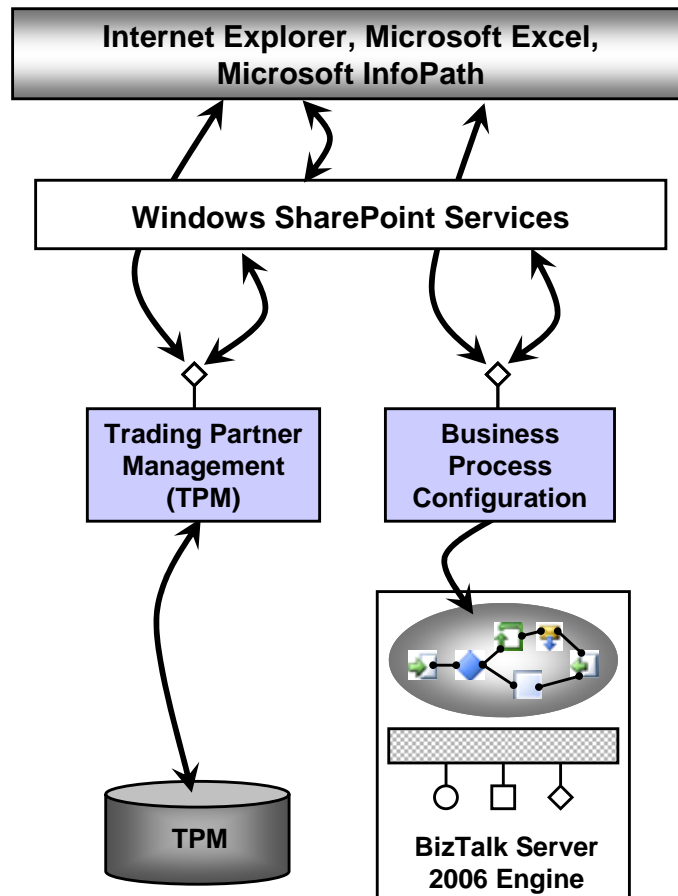
While information workers can create BAM views and BAM activities on their own, these views and activities depend on information provided by the orchestrations they monitor. Accordingly, developers still have a role to play. Using a tool called the *Tracking Profile Editor (TPE)*, a developer must configure an orchestration so that it provides the information required for a particular BAM activity, and thus for the BAM views that depend on this activity. This tool allows a developer to graphically associate the appropriate events and message fields in an orchestration with corresponding milestones and business data in a BAM activity. The BizTalk Server 2006 engine then sends these events and message field values to the *Tracking* database, as shown in the earlier figure, where they can be accessed by the BAM component. Yet while developers must play their part, BAM activities and BAM views aren't their concern. These business-oriented services are created, maintained, and used solely by information workers.

BizTalk Server 2006 brings other important additions to the product's BAM capabilities. In the previous release, BAM could only be used to monitor events happening within BizTalk orchestrations. In BizTalk Server 2006, however, the TPE can also be used to specify how pipelines generate events. More important, BAM can now accept and display events generated by any user code, whether or not it's built as an orchestration. Any application built using the .NET Framework or WinFX, a set of new development technologies that will appear with Windows Vista, can potentially be monitored using the BAM component of BizTalk Server 2006.

## Business Activity Services

There are many things an information worker might wish to do with a running business process. BAM addresses an important aspect of these, but there are others. Maybe a business analyst needs to create a relationship with a new trading partner, for example, defining the partner's role, the business agreement between the two firms, and other aspects of this new association. Maybe a purchasing manager needs tools that can wrap together and distribute what's needed to let a partner quickly implement and begin participating in a business process. In BizTalk Server 2006, all of these things are provided by Business Activity Services.

As shown below, a common user interface to all of these services is provided through Windows SharePoint Services, Internet Explorer, Microsoft Excel, and Microsoft InfoPath. Because Business Activity Services are meant to be used by business people, not developers, it makes sense to expose them through these familiar tools. Behind this common interface are two different software components, both of which expose their services via SOAP. This section describes these two components.



### Trading Partner Management

Creating B2B connections between trading partners is a common use of BizTalk Server today. Establishing these connections requires agreeing on several things, including the communication protocol that will be used, the formats of messages that will be exchanged, and the business process that drives the interaction. Managing these trading partner relationships can get complex, especially when many organizations are involved or when the players change frequently.

To allow information workers to perform these common tasks, BizTalk Server 2006's Business Activity Services include a *Trading Partner Management (TPM)* component. This component relies on a *TPM* database, as shown above, that stores information about trading relationships. Using the common Business Activity Services interface, information workers can create and modify *agreements* with trading partners who use BizTalk Server 2006. Each agreement describes the relationship between two parties, and the things it contains include:

- A *profile* for each of the partners. Each profile contains business information about the organization, such as a contact person and address, along with technical information such as what protocol (and thus which BizTalk Server 2006 adapter) should be used to communicate with them.

- The business process itself, implemented as one or more orchestrations, along with what role each of the partners plays. One organization might act as the seller, for example, while the other acts as the buyer.

An *addendum* with parameters for the business process that control the behavior of the orchestration implementing it. How these parameters are used is described in the next section.

Profiles, agreements, and addendums are all stored in the TPM database. Using the TPM component (and for addendums, the Business Process Configuration component, described next), all of them can be configured directly by an information worker. This allows business people to establish and modify new partner relationships without relying on developers.

#### Business Process Configuration

It's too much to expect an information worker to create the orchestration that implements a business process. It's not unreasonable, though, to expect an information worker to be able to set parameters of that orchestration. One good example of where this would be useful is when the same orchestration will be used with multiple partners, but each partner requires slightly different behavior. Suppose, for instance, that the maximum dollar value of a purchase order is different with different trading partners, or perhaps the maximum quantity that can be requested varies depending on the customer's credit rating. Requiring a developer to modify the orchestration to make these small configuration changes is overkill. Why not let information workers do it themselves?

This is what the *Business Process Configuration* service in BizTalk Server 2006 allows. To let information workers configure an orchestration, the developer creating it can define parameters for that orchestration. An information worker can then set these parameters as he sees fit, perhaps assigning different values for different business partners or different parts of his own organization. An information worker sets those parameters via the TPM service, described in the previous section, by specifying their values in the addendum to this partner's agreement. If an agreement references multiple orchestrations, multiple addendums can be created, one for each orchestration.

## BizTalk Server 2006 and Other Windows Technologies

The need to support automated business processes more effectively increases every day, while the global vendor agreement on web services has laid the foundation for a service-oriented world. To address the first of these changes, Microsoft has created *Windows Workflow Foundation*, while the second change spurred the development of *Windows Communication Foundation*. Both of these new technologies relate to BizTalk Server in important ways, as this section describes.

### BizTalk Server 2006 and Windows Workflow Foundation

The orchestration capabilities in BizTalk Server 2006 are focused on system-to-system communication, supporting business processes that depend on integrating diverse software. Yet the ability to coordinate work done by software and by people, sometimes referred to by the more general term *workflow*, can be useful in more than just integration scenarios. Why not provide a framework that lets any Windows application use this technology? Windows Workflow Foundation (WWF), scheduled to be released in 2006, is designed to meet this goal.

Unlike BizTalk Server, with its focus on integrating independent systems, WWF provides a general framework for creating applications that are themselves built around workflows. Over time, WWF will become the common workflow technology used by Microsoft products, including the Microsoft Office System and others. In fact, the BizTalk release that follows BizTalk Server 2006 will include the ability to create WWF workflows alongside its current orchestration capabilities.

To get a sense of what WWF provides, here are some examples of how it might be used:

An ASP.NET application that displays pages to its users might use a WWF workflow to control the order in which those pages are shown. Doing this can make it easier to change the page flow without changing the pages themselves, as well as cleanly separating the application's user interface from its controlling logic.

The next version of Microsoft Office, code-named Office "12", will let information workers create and modify document-oriented workflows. This ability relies on WWF hosted in Windows SharePoint Services.

A composite application in a service-oriented environment might implement its core logic using a workflow. As more and more applications expose their behavior through web services, WWF can provide a foundation for the process logic that drives these services.

An application built by an independent software vendor targeting a specific problem, such as customer relationship management, or a particular vertical market, such as financial services, might be built around a WWF workflow. This kind of application commonly implements a number of different business processes, and so designing it around workflow technology can make the application faster to build and easier to change.

BizTalk Server and WWF have some obvious similarities. To a developer, for example, BizTalk Server's Orchestration Designer looks much like the Workflow Designer provided by WWF. This shouldn't be surprising, since the same group within Microsoft is responsible for both. But the two technologies address quite distinct problems. Here are some guidelines for deciding when to use each one.

Use BizTalk Server when:

Solving an EAI problem that requires communication with diverse applications on diverse platforms. Because of its focus on cross-platform integration, BizTalk Server provides adapters for communicating with other software, tools for mapping between message formats, and more. WWF is focused solely on workflow, not EAI, and so it doesn't provide these things.

B2B services are required. WWF doesn't address this area, while BizTalk Server provides tools for working with trading partners, accelerators for RosettaNet, SWIFT, and other industry standards, and more.

BPM services such as BAM are needed. WWF provides a basic tracking infrastructure that can be used to create these services, but BizTalk Server includes a much more complete set of tools in this area.

A complete management infrastructure and support for increased scalability are necessary. As described earlier, BizTalk Server includes a full set of tools for administering and scaling a production environment, something that's not provided by WWF.

Use WWF when:

An application will itself host workflows. WWF lets workflow be built into an application, allowing the workflow to be deployed and managed as a native part of the application. Because it's focused on integrating diverse applications rather than

providing a general workflow framework, BizTalk Server always runs orchestrations within the BizTalk Server process.

The business process being implemented requires human workflow. BizTalk Server addresses system workflow, and so it lacks WWF's support for things such as state machine workflows and dynamic update<sup>4</sup>. A scenario that requires both human workflow and more complex system integration services could be addressed by using WWF and BizTalk Server together, however. For example, the Office "12" support for document-centric workflows, based on Windows SharePoint Services, might be used for the human aspects of the problem, while BizTalk Server handles the system integration aspects. The two can interoperate using the BizTalk Server Adapter for SharePoint.

The workflow will execute on a client system. BizTalk Server is a server-focused product, and so it's less well-suited to run on desktop machines.

WWF is part of WinFX, which will be introduced with Windows Vista. For an introduction to WWF, see *Introducing Windows Workflow Foundation*, available at [www.microsoft.com](http://www.microsoft.com).

## BizTalk Server 2006 and Windows Communication Foundation

BizTalk Server 2006 supports web services through the Web Services adapter and ASP.NET. In 2006, Microsoft is scheduled to release Windows Communication Foundation (WCF), a new framework for creating service-oriented applications. Like WWF, WCF is part of the new WinFX interface to Windows. Implemented as an extension to the .NET Framework, WCF is the successor to several existing .NET technologies, including ASP.NET Web Services, .NET Remoting, and Enterprise Services. Designed for a world of web services, WCF also supports industry-standard specifications such as WS-Security, WS-ReliableMessaging, and WS-AtomicTransaction. This gives applications built on WCF secure, reliable, and transactional communication with other applications running on any system that also implements these specifications.

Once WCF is available, an adapter will be provided for BizTalk Server 2006 that allows using WCF rather than ASP.NET Web Services. (In fact, a community-supported version of this adapter is available now; see [www.gotdotnet.com](http://www.gotdotnet.com) for details.) As organizations increasingly move toward service-oriented architectures, having an infrastructure with built-in support for both web services and other kinds of communication will become critical. Incorporating WCF will allow BizTalk Server to meet this requirement more effectively.

## Conclusions

The goal of BizTalk Server 2006 is to help organizations meet the challenges of creating automated business processes that rely on diverse systems. The product's foundation is the BizTalk Server 2006 engine, which provides core messaging and orchestration capabilities. Developers can also use the Business Rules Engine to address complex business scenarios, the Health and Activity Tracking tool to debug and examine BizTalk applications, and Enterprise Single Sign-On to create more secure environments. Information workers can use the product's Business Activity Monitoring support to get business-oriented information about a running process and Business Activity Services to work with trading partners.

---

<sup>4</sup> BizTalk Server 2004, the product's previous release, introduced a component called Human Workflow Services to help include human interactions in BizTalk-based business processes. While this component is still included in BizTalk Server 2006, its use is now deprecated.

From its initial roots in EAI and B2B integration, BizTalk Server 2006 has grown into the foundation for supporting a range of business processes. As the change to a service-oriented world rolls on, BizTalk Server 2006 will continue to play an important part in Windows-based automation of business processes.

### About the Author

David Chappell is Principal of Chappell & Associates ([www.davidchappell.com](http://www.davidchappell.com)) in San Francisco, California. Through his speaking, writing, and consulting, he helps information technology professionals around the world understand, use, market, and make better decisions about enterprise software technologies.