

# **Understanding BizTalk Server 2004**

**BizTalk Server 2004 Technical Article**

**Applies to:** Microsoft® BizTalk® Server 2004

**Published:** February 2004



---

## Copyright

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2003 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, BizTalk, InfoPath, JScript, Outlook, SharePoint, Visio, Visual Basic, Visual Studio, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

<b>Understanding BizTalk Server 2004.....</b>	<b>1</b>
Abstract.....	5
Introducing BizTalk Server 2004.....	5
BizTalk Server 2004 Engine .....	8
<b>Connecting Applications .....</b>	<b>9</b>
<b>Sending and Receiving Messages: Adapters.....</b>	<b>10</b>
<b>Processing Messages: Pipelines.....</b>	<b>11</b>
<b>Choosing Messages: Subscriptions .....</b>	<b>14</b>
<b>Defining Business Processes .....</b>	<b>14</b>
<b>Orchestration .....</b>	<b>14</b>
<i>Creating Schemas: BizTalk Editor .....</i>	<i>15</i>
<i>Mapping Between Schemas: BizTalk Mapper.....</i>	<i>15</i>
<i>Defining Business Processes: Orchestration Designer.....</i>	<i>16</i>
<i>Additional Orchestration Support .....</i>	<i>17</i>
<b>Business Rule Engine .....</b>	<b>18</b>
<b>Management and Monitoring.....</b>	<b>20</b>
<b>Creating Scalable Configurations.....</b>	<b>20</b>
<b>Managing Applications .....</b>	<b>21</b>
<b>Monitoring Applications: Health and Activity Tracking .....</b>	<b>22</b>
<b>Enterprise Single Sign-On .....</b>	<b>22</b>
<b>Looking Ahead: The Engine and "Indigo" .....</b>	<b>24</b>
Information Worker Technologies .....	25
<b>Business Activity Services .....</b>	<b>26</b>
<b>Trading Partner Management .....</b>	<b>27</b>
<b>Business Process Configuration.....</b>	<b>27</b>
<b>Business Process Provisioning.....</b>	<b>28</b>
<b>Business Activity Monitoring Framework .....</b>	<b>28</b>
<b>Human Workflow Services .....</b>	<b>29</b>
Conclusions.....	31
About the Author .....	32

## Abstract

Microsoft® BizTalk® Server 2004 is an integration server product that enables you to develop, deploy, and manage integrated business processes and XML-based Web services. It supports the goal of creating business processes that unite separate applications into a coherent whole.

This latest version of BizTalk Server provides deep integration between messaging and orchestration, as well as enhanced security and support for industry standards. BizTalk Server 2004 also provides components—Business Activity Services, Human Workflow Services, and the Business Activity Monitoring Framework—that enable information workers to interact with business processes.

## Introducing BizTalk Server 2004

No application is an island. Even though many are still created with an internal focus, the reality is that tying applications together has become the norm. Yet connecting software is about more than just exchanging bytes. As organizations move toward a service-oriented architecture (SOA), the real goal—creating business processes that unite separate applications into a coherent whole—comes within reach.

Microsoft® BizTalk® Server 2004 supports this goal. This new release enables you to connect diverse applications, and then to graphically create and modify business processes that use the services that those applications provide. This release also brings more to the core BizTalk Server 2004 engine, including a mechanism for specifying business rules, better ways to manage and monitor the applications built on it, and support for single sign-on for those applications.

BizTalk Server 2004 also adds new services for information workers. These include a group of Business Activity Services (BAS), a Business Activity Monitoring (BAM) Framework for analyzing running business processes, support for business process provisioning and configuration, and services that enable information workers to set up and manage interactions with trading partners. BizTalk Server 2004 also adds technology for creating Human Workflow Services (HWS), making possible business processes that people can interact with from Microsoft Outlook® and other familiar clients. Given the amount of new technology it contains, it is fair to say that BizTalk Server 2004 is a significant update from its predecessor.

BizTalk Server has changed because the world in which customers use it has changed. The biggest change, one that confronts every organization, is the rapid spread of Web services. Because they allow universal connectivity among applications, Web services will lead most organizations to some form of SOA. Other important changes flow from this, such as the ability to define Web services-based business processes by using the Business Process Execution Language for Web Services (BPEL4WS, commonly called just BPEL). The changes in the Microsoft world itself are also substantial, with the most important flowing from the advent of the Microsoft .NET Framework. Because it provides a new foundation for building applications, the .NET Framework transforms how Microsoft Windows®-oriented developers create software.

All of these changes have strongly impacted Microsoft's approach to building this new BizTalk Server release. Unlike its COM-based predecessors, BizTalk Server 2004 is built completely around the .NET Framework and Microsoft Visual Studio® .NET. It also has

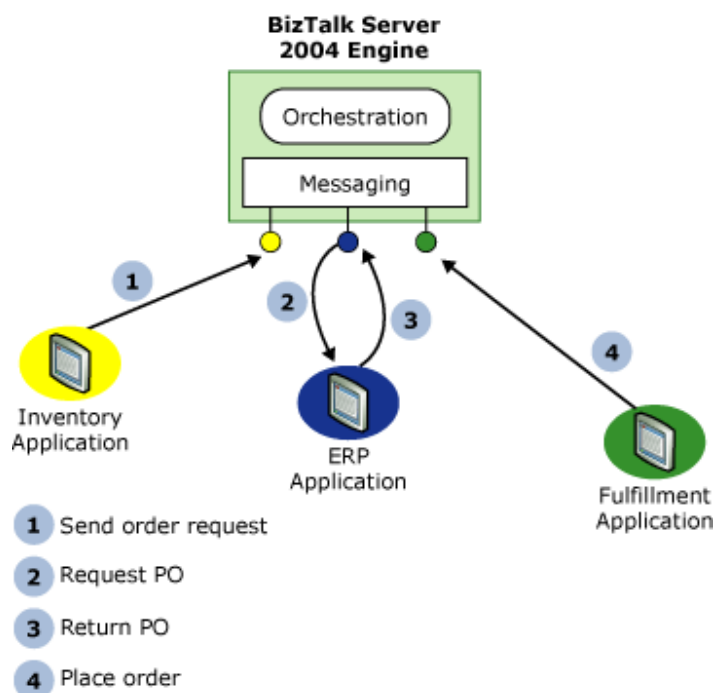
native support for communicating through Web services, along with the ability to import and export business processes described in BPEL. Designed to work well both with the emerging world of standard Web services and with the large number of applications already in place, BizTalk Server 2004 is a significant step forward for the BizTalk Server product line.

You can use BizTalk Server 2004 in a variety of ways. Traditionally, BizTalk Server has been used for application integration, where the following two scenarios are most important:

Connecting applications within a single organization, commonly referred to as enterprise application integration (EAI)

Connecting applications in different organizations, often called business-to-business (B2B) integration

The following figure shows a simple example of the core BizTalk Server 2004 engine applied to an EAI problem.



### BizTalk Server used for EAI

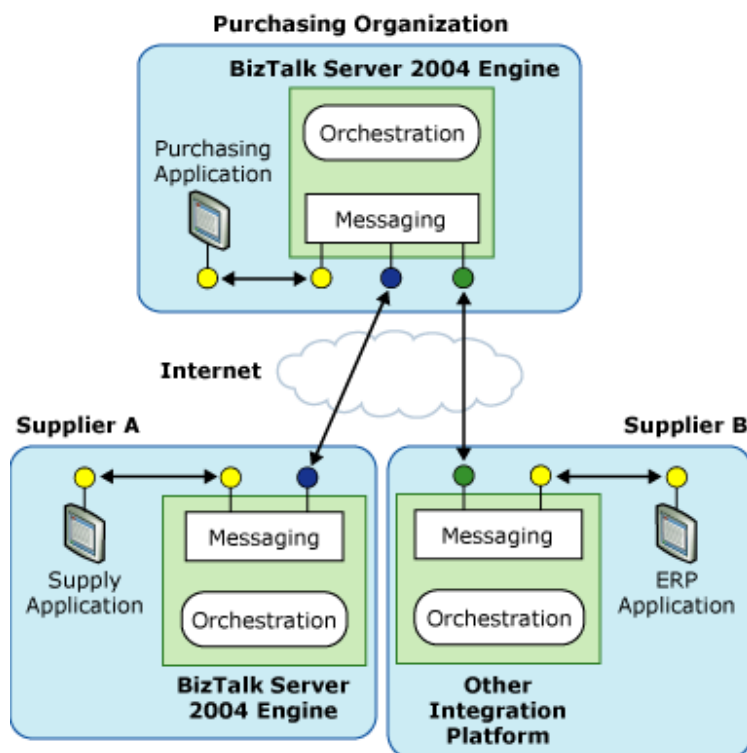
In this scenario, an inventory application, perhaps running on an IBM mainframe, notices that the stock of an item is low and issues a request to order more of that item. The following steps occur:

1. The request is sent to a BizTalk Server 2004 application.
2. The BizTalk Server 2004 application requests a purchase order (PO) from the organization's Enterprise Resource Planning (ERP) application.
3. The ERP application, which might be running on a Unix system, sends back the requested PO.

4. The BizTalk Server 2004 application informs a fulfillment application, perhaps built on Microsoft Windows by using the .NET Framework, that the item should be ordered.

In this example, each application communicates by using a different protocol. Accordingly, the messaging infrastructure of the BizTalk Server 2004 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. That business process, along with the intelligence required to coordinate all of the parts, is implemented in the BizTalk Server 2004 application.

Connecting applications within an organization is important, but connecting applications that span organizations can sometimes have even more business value. The following figure shows a simple example of a B2B integration scenario.



### BizTalk Server used for B2B

The applications are connected as follows:

The purchasing organization at the top of the figure runs a BizTalk Server 2004 application that interacts with two supplier organizations.

Supplier A also uses BizTalk Server 2004, providing indirect access to its Supply application.

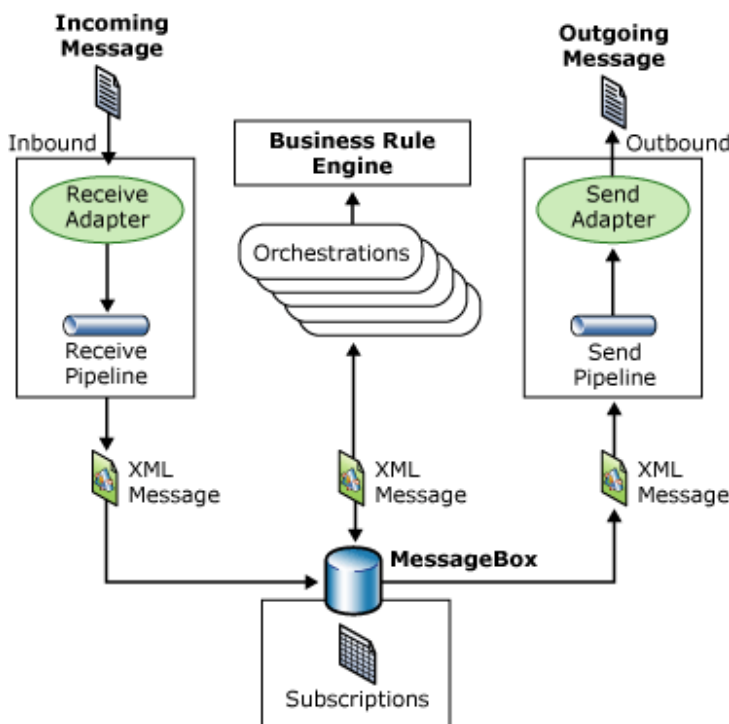
Supplier B uses an integration platform from another vendor, connecting to the purchasing organization's BizTalk Server 2004 application by using, say, Web services. Supplier B is executing the same business process as the others, and so it may have been sent a BPEL definition of that process by the purchasing organization, which exported this definition from BizTalk Server 2004.

The integration of existing applications—whether they are used in a single company or spread across different organizations—into a single business process is a fundamental goal of BizTalk Server 2004. But it is not the whole story. Other useful services are built on this foundation. For example, the people who use a business process need to interact with it in various ways. Accordingly, BizTalk Server 2004 provides services that give information workers—and not just technical workers—visibility into running business processes, and let them interact with the processes when necessary.

When learning to use BizTalk Server 2004, it is useful to divide the product conceptually into two parts: the core engine and the services for information workers that are built on top of that engine. This document describes both, starting with the BizTalk Server 2004 engine.

## BizTalk Server 2004 Engine

To enable users to create a business process that spans multiple applications, the BizTalk Server 2004 engine must provide two capabilities: a way to specify the business process, and some mechanism for communicating between the applications the business process uses. The following figure shows the main components of the BizTalk Server 2004 engine that provide these capabilities.



### BizTalk Server engine

A business process is implemented as one or more orchestrations, each of which consists of executable code. These orchestrations are not created by writing code in a language such as C#, however. Instead, a business analyst uses the Orchestration Designer for Business Analysts (a Microsoft Visio® snap-in) to graphically organize a defined group of shapes to express the conditions, loops, and other behavior of the business process. Business processes can also use the Business Rule Engine, which

provides a simpler and more easily modified way to express the rules in a business process. Each orchestration creates subscriptions to indicate the kinds of messages it wants to receive.

The message processing shown in the preceding figure is as follows:

1. A message is received through 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.
2. The message is processed through a receive pipeline. This pipeline can contain components that perform actions such as converting the message from its native format into an XML document or validating the message's digital signature.
3. The message is delivered into a database called the MessageBox database, which is implemented by using Microsoft SQL Server™.
4. The message is dispatched to its target orchestration, which takes whatever action the business process requires.
5. The result of this processing is typically another message, produced by the business process and saved in the MessageBox database.
6. This message, in turn, is processed by a send pipeline, which may convert it from the internal XML format used by BizTalk Server 2004 to the format required by its destination, add a digital signature, and more.
7. The message is sent out through a send adapter, which uses an appropriate mechanism to communicate with the application for which this message is destined.

Three roles are necessary to create and maintain BizTalk Server 2004 applications. These roles, and the functions they perform with the BizTalk Server 2004 engine, are as follows:

**Business analyst.** Defines the rules and behaviors that make up a business process, and determines the flow of the business process by defining what information gets sent to each application and how one business document is mapped into another.

**Developer.** Implements the business process defined by the business analyst. Implementation includes tasks 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 business process.

**Administrator.** Performs tasks such as setting up communication among the parts and deploying the application in an appropriately scalable way.

## Connecting Applications

Effectively exchanging messages with applications is an absolute requirement for integration. Given the diversity of communication styles that exist, the BizTalk Server 2004 engine must support a variety of protocols and message formats. As described next, a significant portion of the BizTalk Server 2004 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 is always converted to an XML document after it is received. Similarly, if the recipient of a

document cannot accept that document as XML, the engine converts it into the format expected by the target application.

## Sending and Receiving Messages: Adapters

Because the BizTalk Server 2004 engine must talk to a wide range of other software, it relies on a range of adapters to make this possible. An adapter is an implementation of a communication mechanism, such as a particular protocol. BizTalk Server 2004 provides built-in adapters, and adapters have been created for popular applications such as SAP. A developer can determine which adapters to use in a given situation, or can create custom adapters for specific needs. All adapters are built on a standard base called the Adapter Framework. New with BizTalk Server 2004, this framework provides a common way to create and run adapters. It also enables you to use the same tools to manage both standard and custom adapters. While adapters written for earlier versions of BizTalk Server will not work with BizTalk Server 2004, the Adapter Framework makes it easier to create new adapters for this new release.

BizTalk Server 2004 provides the following adapters:

**SOAP adapter.** Enables sending and receiving messages by using SOAP over HTTP. Because SOAP is the core protocol for Web services, this adapter gives BizTalk Server 2004 the ability to interact in a Web services world. As usual with Web services, URLs are used to identify the sending and receiving systems.

**BizTalk Message Queuing adapter.** Enables sending and receiving messages by using BizTalk Message Queuing (MSMQ). BizTalk Message Queuing is an implementation of the Microsoft Message Queuing (MSMQ) protocol that can receive and send MSMQ messages from and to the MessageBox database. It is not a replacement for MSMQ, but rather an efficient way to use the MSMQ transport with BizTalk Server.

**File adapter.** Enables 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.** Enables sending and receiving information by using HTTP. The BizTalk Server 2004 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.

**SMTP adapter.** Enables sending messages by using Simple Mail Transfer Protocol (SMTP). Standard e-mail addresses are used to identify the parties.

**SQL adapter.** Enables reading and writing information from and to a SQL Server database.

**Base EDI adapter.** Enables sending and receiving messages by using the American National Standards Institute (ANSI) X-12 and Electronic Data Interchange for Administration, Commerce, and Trade (EDIFACT) standards.

**FTP adapter.** Enables exchange of files between BizTalk and FTP servers.

The messages that an adapter receives usually need to be processed before an orchestration can access them. Similarly, outgoing messages produced by an

orchestration often need to be processed before an adapter sends them. How this processing is done is described next.

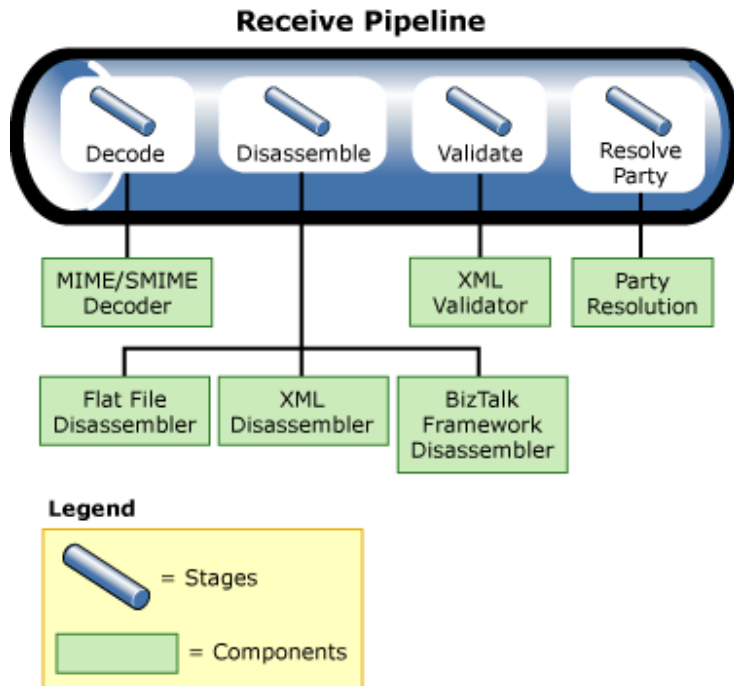
## Processing Messages: Pipelines

The applications underlying a business process communicate by exchanging various kinds of documents, such as purchase orders and invoices. For a BizTalk Server 2004 application to execute a business process, it must be able to correctly process the messages that contain these documents. This processing can involve multiple steps, and it is 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 even the majority today—do not. Because the BizTalk Server 2004 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 way, a pipeline is constructed from some number of stages, each of which contains one or more .NET or COM components. Each component handles a particular part of the message processing. The BizTalk Server 2004 engine provides several standard components that address the most common cases. If these are not sufficient, developers can also create custom components for both receive and send pipelines.

BizTalk Server 2004 defines some default pipelines, including a simple receive/send pair that you can use for handling messages that are already expressed in XML. A developer can also create custom pipelines by using the Pipeline Designer tool. This tool, which runs inside Visual Studio .NET, provides a graphical interface that enables dragging and dropping components to create receive and send pipelines with the required behavior.

The following figure illustrates the stages in a receive pipeline, along with the standard components provided for each one.



### Receive pipeline

The stages and their associated components are:

**Decode.** BizTalk Server 2004 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.** BizTalk Server 2004 provides three standard disassembler components. These are:

**Flat file disassembler.** 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.

**XML disassembler.** Parses incoming messages that are already described by using XML.

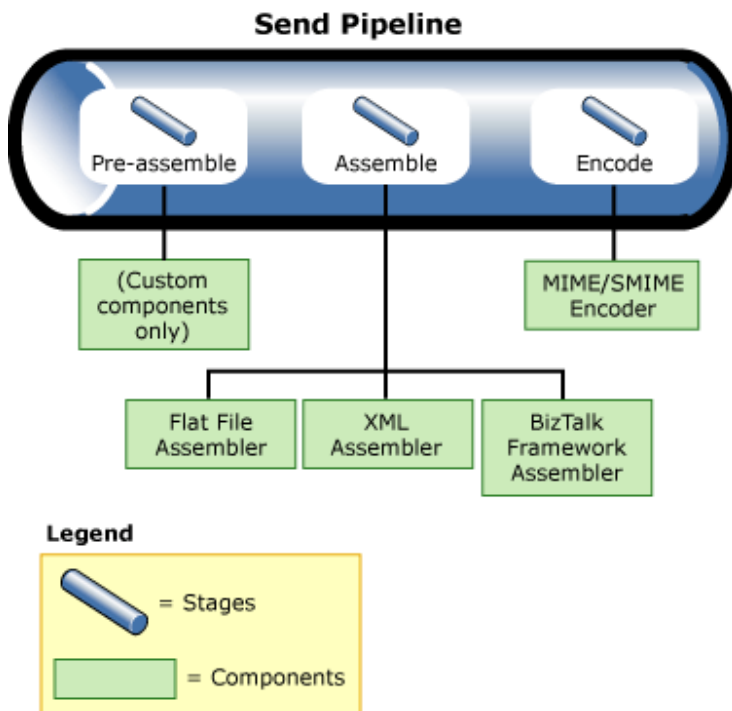
**BizTalk Framework disassembler.** Accepts messages sent by using the reliable messaging mechanism defined by the BizTalk Framework. The BizTalk Framework was implemented in BizTalk Server 2000 and will eventually become less relevant when Web services reliable messaging specifications are widely adopted.

**Validate.** BizTalk Server 2004 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 does not conform to one of those schemas.

**Resolve Party.** The only standard component for this stage, party resolution, attempts to determine an identity for the message's sender. If the message was

digitally signed, the signature is used to look up a Windows identity in the BizTalk Server 2004 Configuration database. 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 a send pipeline. The following figure shows the stages and standard components for a send pipeline.



### Send pipeline

The stages and their associated components 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:

**Flat file assembler.** Converts an XML message into a positional or delimited flat file.

**XML assembler.** Enables adding an envelope and making other changes to an outgoing XML message.

**BizTalk Framework assembler.** Packages messages for reliable transmission by using the BizTalk Framework messaging technology.

**Encode.** BizTalk Server 2004 defines 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.

## Choosing Messages: Subscriptions

After a message has passed through an adapter and a receive pipeline, the business process must determine where it should go. A message is most often targeted to an orchestration, but it is also possible for a message to go directly to a send pipeline, using the BizTalk Server 2004 engine purely as a messaging system. In either case, messages are matched with their destinations through subscriptions.

When a receive pipeline processes a message, it creates a message context 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 Woodgrove Bank, or all messages of the type "Invoice" received from Woodgrove Bank that are for more than \$10,000. However it is specified, a subscription returns to its subscriber only those messages that match the criteria that the 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 the pipeline has established.

## Defining Business Processes

Sending messages between different applications is a necessary part of solving the problems that BizTalk Server 2004 addresses. Yet in most cases, it is only a means to an end. The real goal is to define and execute business processes based on these applications. The BizTalk Server 2004 engine provides two technologies for doing this: orchestration and the Business Rule Engine. This section describes both.

## Orchestration

You can implement a business process directly in a language such as C# or Microsoft Visual Basic® .NET. However, creating, maintaining, and managing complex business processes in conventional programming languages can be challenging. Just as important, a process created in this way is likely to act as a set of disparate applications rather than as a coherent business process. Like its predecessors, then, BizTalk Server 2004 does not take this approach. Instead, it enables you to create a business process graphically. Doing this can be faster than building the process directly in a programming language, and it can make the process easier to understand, explain, and change. Business processes built in this fashion can also be monitored more easily, a fact that is exploited by the Business Activity Monitoring (BAM) technology described later in this document.

Successfully creating an automated business process usually requires collaboration between technology-oriented developers and less-technical business people. Accordingly, BizTalk Server 2004 provides appropriate tools for each. The developer tools run inside Visual Studio .NET, an environment in which software professionals feel at home. Most business people do not find Visual Studio especially inviting, however, so BizTalk Server 2004 also provides a subset of the developer tool functionality in the Orchestration Designer for Business Analysts (ODBA), an add-in for Microsoft Visio. Information created in the ODBA tool can be imported into the Visual Studio-based

tools and vice-versa, which helps developers and business analysts work together when creating a business process. After it exists, this process, known as an orchestration, is automatically transformed into standard assemblies that run on the .NET Framework.

**Note** Orchestrations were called schedules in BizTalk Server 2000 and BizTalk Server 2002.

A developer relies on three primary tools for creating an orchestration: BizTalk Editor for creating XML schemas, BizTalk Mapper for defining translations between those schemas, and Orchestration Designer for specifying the flow of business processes. Unlike earlier versions of the product, all of the developer tools in BizTalk Server 2004 are hosted inside Visual Studio .NET, providing a consistent environment. This section describes what each of these tools does and how they work together.

## *Creating Schemas: BizTalk Editor*

Orchestration works with XML documents, each of which conforms to some XML schema. Accordingly, there must be a way to define these schemas. To do this, the BizTalk Server 2004 engine provides BizTalk Editor. This tool enables you to use the XML Schema definition language (XSD) to create schemas, which are essentially definitions of the types and structure of a document's information. While BizTalk Editor is not new with this release, this complete reliance on standard XSD is. Because the final Worldwide Web Consortium (W3C) standard for XSD was not complete when they were developed, previous versions of BizTalk Server used XML Data Reduced (XDR), a Microsoft-specific language for defining schemas. Now that XSD has been finalized, BizTalk Server 2004 relies on this standard approach for describing the structure of XML documents.

Creating raw XSD schemas without some tool support is not simple. To make this necessary step more approachable, BizTalk Editor enables 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 are acquired, schemas are used as the basis for BizTalk maps, which are described next.

## *Mapping Between Schemas: BizTalk Mapper*

An orchestration implementing a business process typically receives some documents and sends others. It is 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, and then send back an acknowledgment indicating that the order was sent. It is possible that information from the order, such as the name and address of the purchaser, might be copied from fields in the received order into fields in the order acknowledgment. You can use BizTalk Mapper to define a transformation—a map—from one document to the other.

To the developer creating it, 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;

therefore, as in previous versions of BizTalk Server, maps in BizTalk Server 2004 are implemented as XSLT transformations.

The transformation defined in a map can be simple, such as copying a name and address from one document to another. Direct data copies like this are expressed by using a link, which is shown in 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 by using functoids. A functoid is a chunk of executable code that can define arbitrarily complex mappings between XML schemas, and BizTalk Mapper represents it as a box on the line connecting the elements being transformed. Because some of those transformations are fairly common, BizTalk Server 2004 includes a number of built-in functoids. These built-in functoids are grouped into categories that include the following:

**Mathematical functoids.** 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.** Convert a numeric value to its ASCII equivalent and vice-versa.

**Logical functoids.** 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.** Compute averages, sums, or other values from various fields in the source document, and then store the result in a single field in the target document.


**Database functoids.** Access information stored in a database.


You can create custom functoids directly in XSLT or by using .NET languages like C# and Visual Basic .NET. (Earlier versions of BizTalk Server used Visual Basic Scripting Edition (VBScript) or Microsoft JScript® to create custom functoids, and you may need to modify maps that use these to work with BizTalk Server 2004.) 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. BizTalk Editor and BizTalk Mapper address these two functions. Yet defining schemas and maps is not enough. You must also specify the business process that will use the schemas and invoke the maps. How this is done is described next.


## *Defining Business Processes: Orchestration Designer*


A business process is a set of actions that together meet some useful business need. With the BizTalk Server 2004 engine, you can use Orchestration Designer to define these actions graphically. This tool enables you to create an orchestration by connecting a series of shapes in a logical way, rather than expressing the steps in a programming language. Some of the most commonly used shapes are:


 **Receive.** Enables 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.


 **Send.** Enables the orchestration to send messages.


 **Port.** 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.


 **Decide.** Represents an if-then-else statement that allows an orchestration to perform different tasks based on Boolean conditions. You can use the Expression Editor, part of Orchestration Designer, to specify this conditional statement.


 **Loop.** Enables performing an action repeatedly while some condition is true.

 **Construct Message.** Enables building a message.

 **Transform.** Enables transferring information from one document to another, transforming it on the way by invoking maps defined with BizTalk Mapper.

 **Parallel Actions.** Enables specifying that multiple operations should be performed in parallel rather than in sequence. The shape that follows this one will not be executed until all of the parallel actions have completed.

 **Scope.** Enables 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.

 **Message Assignment.** Enables 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.

After you have defined a business process in this way, the group of shapes and relations between them is converted into the Microsoft intermediate language (MSIL) that is used by the .NET Framework common language runtime (CLR). Ultimately, the group of shapes that you define in BizTalk Server 2004 becomes just a standard .NET assembly. And of course you can still add explicit code to an orchestration when necessary by calling a COM or .NET object from inside a shape.

## *Additional Orchestration Support*

The rise of Web services is having an impact on how business processes are defined. For example, think about the case where two organizations interact by 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 2004, this is not a big problem; tools such as the Trading Partner Management technology described later in this document can be used to distribute this knowledge. But what if the organizations use different software? What if one organization uses BizTalk Server 2004 and the other uses software from another vendor? For cases like this, it is useful to have a way to describe the business process interactions in a cross-vendor way.

To support this type of interaction, Microsoft, IBM, and others have created Business Process Execution Language (BPEL). A business process defined by using Orchestration Designer can be exported to BPEL, and BizTalk Server 2004 can also import processes defined in BPEL. While the language is useful for describing and sharing the external-facing parts of a business process between trading partners, it is important to realize that BPEL is not focused on cross-platform execution of business processes. It is also important to understand that BPEL is built entirely on Web services, while BizTalk Server 2004 and other products that support this language provide a broader set of services. For example, BizTalk Server 2004 provides support for mapping between different XML schemas, calling methods in local objects, executing transactions, and other features that are not available in BPEL.

Web services enable applications to exchange XML documents through SOAP, and they have had a big impact on integration platforms. Unlike previous versions of the product, BizTalk Server 2004 has built-in support for Web services. To access an external Web service, an orchestration's creator can use the Add Web Reference option in Visual Studio .NET along with the SOAP adapter to directly invoke operations, just as with any other .NET assembly. Similarly, BizTalk Server 2004 provides the 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 enable you 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.

Like the other developer tools provided by BizTalk Server 2004, Orchestration Designer runs inside Visual Studio .NET. In some cases, however, a business analyst rather than a developer may wish to graphically define business processes. Because business analysts may not be comfortable using Visual Studio .NET, BizTalk Server 2004 also includes the Orchestration Designer for the Business Analyst (ODBA) tool, an add-in for Microsoft Visio that enables you to define a business process and then import that process into Orchestration Designer.

Orchestrations are the fundamental mechanism for creating business processes in BizTalk Server 2004. 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 a 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 the Business Rule Engine does, as described next.

## Business Rule Engine

Orchestration Designer, together with BizTalk Editor and BizTalk Mapper, is an effective way to define a business process and the rules it uses. It is sometimes useful, though, to have an easier way to define and change business rules. To allow this, BizTalk Server 2004 provides the Business Rule Engine to enable more business-oriented users to directly create and modify sets of business rules. These rules are created by using a tool called the Business Rule Composer, and then executed directly by the engine. This technology is new in BizTalk Server 2004, and it is one of the most interesting features in the product.

To see why this approach is useful, think about what is required to change a business rule that is implemented within an orchestration. A developer must first open the orchestration in Visual Studio .NET, modify the appropriate shapes (and perhaps the .NET or COM objects they invoke), and then build and deploy the modified assembly. If instead this business rule is implemented by using the Business Rule Engine, it can be modified without recompiling or restarting anything. All that is needed is to use the Business Rule Composer to change the desired rule, and then redeploy the new set of rules. The change takes effect immediately. And while orchestrations are typically created and maintained by developers, business rules are readable enough to be modified by business analysts without the need to involve more technical people.

When you create a set of business rules, you 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 an SQL query against some database, or even to a value in a .NET object.

**Note** You can also use XML element and attribute names directly in definitions of business rules, so creating a vocabulary is not strictly required. It just makes business rules easier to create and understand, especially if business analysts are doing this rather than developers.

After you have defined a vocabulary, you can use the Business Rule Composer to create business policies that use the 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, or Is Equal To 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 is sent, or how those received values should affect what value is written in a database, or other things.

Imagine, for example, 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 can contain a Call Rules shape that creates an instance of the Business Rule Engine, identifies which policy to execute, and then passes in the information this rule needs, such as a received XML document. The Business Rule Engine can also be invoked programmatically through a .NET-based object model, which allows it to be called from applications that do not use the BizTalk Server 2004 engine. This means that Windows Forms applications, software exposing Web services, and anything else built on the .NET Framework can use the Business Rule Engine whenever it helps to solve the problem at hand.

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 and then defining sets of rules that use that vocabulary is the heart of the

Business Rule Engine. The goal is to provide a straightforward way for BizTalk Server 2004 users of all kinds to create and work with the rules that define business processes.

Orchestration abstracts the flow of a business process, expressing it graphically rather than in code. Similarly, the Business Rule Engine enables you to express the rules of a business process in a higher-level way. Together, these two technologies provide an effective approach to creating business logic.

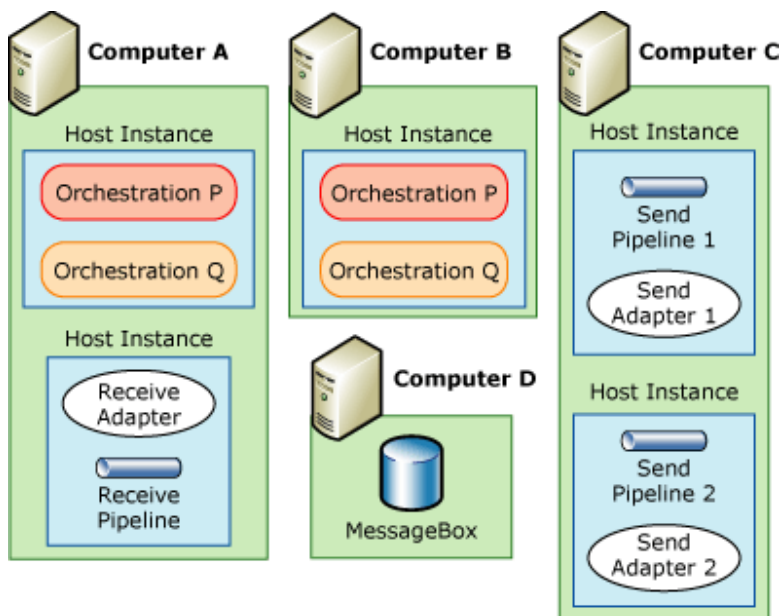
## Management and Monitoring

Every application built on the BizTalk Server 2004 engine requires management. What configurations are possible? How are new applications installed? What is happening inside the system right now? This section looks at the tools provided to answer these questions.

## Creating Scalable Configurations

If the application using it is not too large, the entire BizTalk Server 2004 engine can be installed on a single computer. In many situations, however, this is not the right solution. The number of messages the engine must handle might be too great for one computer, or redundancy might be required to make the system more reliable. To meet requirements like these, the BizTalk Server 2004 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; to use them, a BizTalk Server 2004 administrator must cause actual host instances to be created. Each host instance is a Windows process, and as the following figure shows, it can contain various elements.



Host computers

In this figure:

Computer A runs two host instances. One contains a receive adapter and receive pipeline, while the other contains the orchestrations P and Q.

Computer B runs just one host instance, also containing the orchestrations P and Q.

Computer C, like computer A, runs two host instances, but neither of them contains an orchestration. Instead, each of these instances contains a send pipeline and send adapter.

Computer D houses the MessageBox database that is used by all of the host instances in this configuration.

This example illustrates several ways in which hosts might be used. For example, because both computers A and B run host instances that contain the orchestrations P and Q, BizTalk Server 2004 can automatically assign requests to these orchestrations based on the availability and current load on each computer. This allows a business process to scale up as needed for high-volume applications. Notice also that computer C contains two different ways to handle outgoing messages. One way might rely on a standard BizTalk Server 2004 adapter, such as the HTTP adapter, while the other might use a custom adapter to communicate with a particular application. Grouping all output processing on a single computer like this can make good sense in some situations. And because each host instance is isolated from every other host instance—they are different processes—it is safer to run code that is not completely trusted, such as a new custom adapter, in a separate instance. It is also worth pointing out that even though this example contains only a single instance of the MessageBox database, you can also replicate or cluster the databases to avoid creating a single point of failure.

## Managing Applications

The BizTalk Server 2004 engine provides a range of services, and several different tools are used to manage this environment. The primary tool is the BizTalk Administration console, a Microsoft Management Console (MMC) snap-in. This tool enables you to create hosts, assign hosts to computers, start and stop orchestrations, and perform many other administration tasks. You can even dynamically add computers and specify what hosts should be assigned to them while an application is running—there is no need to shut the application down to make these changes. You can also access the Administration console's functions programmatically through Windows Management Instrumentation (WMI), which enables you to create scripts that automate management functions.

Other management tools are also available for more specific purposes. For example, you can use the BizTalk Deployment Wizard to deploy assemblies to computers, and you can use the Subscription Viewer to examine subscriptions in the BizTalk Server 2004 engine. All of these tools, along with the BizTalk Administration console, rely on a common Configuration database to store the information that they work with.

One more important tool for working with BizTalk Server 2004 applications is BizTalk Explorer. It is new with BizTalk Server 2004, and to understand what it does, think about how a developer builds an application. Initially, a developer creating an application works mainly in logical terms, without defining details. For example, a developer might specify that the BizTalk Server 2004 engine will communicate with a

particular application through the HTTP adapter without defining the URL that will be used, or specify that the send pipeline should add a digital signature to outgoing messages without defining the key that will be used to create this signature.

Yet to make the application work, these details must be specified, which is where BizTalk Explorer comes in. Relying on the same Configuration database that is used by the other BizTalk Server 2004 management tools, this tool enables mapping between the logical view of an orchestration and the concrete physical view required to actually use that orchestration. BizTalk Explorer can also be helpful in working with partners, because it enables you to create a single application configuration, and then deploy this configuration differently for each partner. And because it exposes its services through a .NET-based object model as well as a graphical interface, you can create scripts for configuring applications programmatically.

## Monitoring Applications: Health and Activity Tracking

BizTalk Server 2004 applications do many things: send and receive messages, process messages within orchestrations, communicate with various applications using different protocols, and more. It is essential to keep track of what is going on, especially when failures occur. Doing this requires a way to look into running applications and to monitor what is happening with the system, which exactly describes the services provided by the Health and Activity Tracking (HAT) component of BizTalk Server 2004.

The HAT tool provides graphical access to detailed information, including when an orchestration starts and ends, when each shape within it is executed, when each of its messages is sent and received, what is in those messages, and much more. You can even set breakpoints, enabling you to stop the orchestration and examine it at predetermined places. You can also use the HAT tool 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

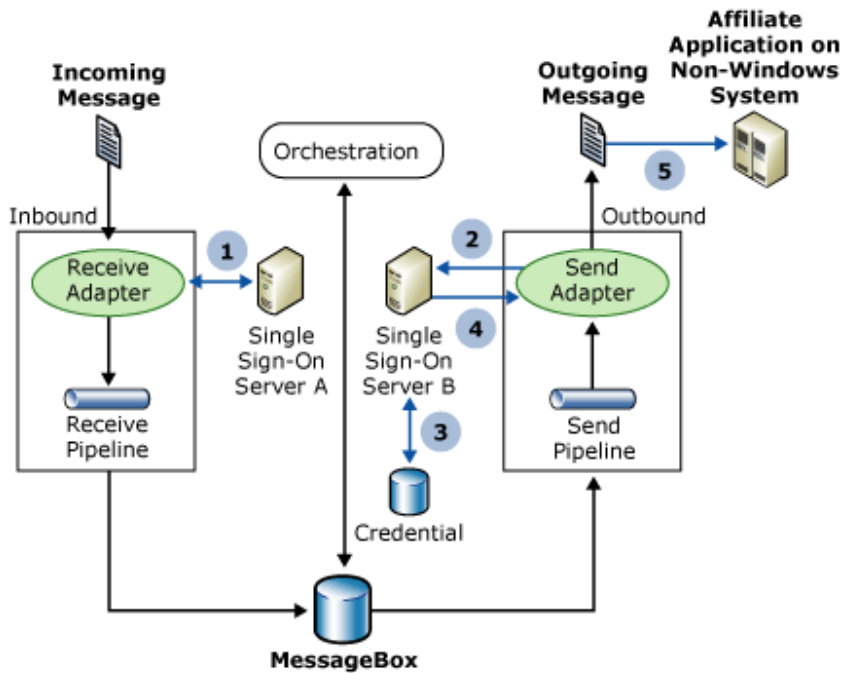
A business process that relies on several different applications is likely to face the challenge of dealing with several different security domains. Accessing an application on a Windows system may require one set of security credentials, while accessing an application on an IBM mainframe may require different credentials, such as an RACF user name 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 2004 includes Enterprise Single Sign-On (SSO).

Enterprise Single Sign-On provides a way to map a Windows user ID to non-Windows user credentials. It will not 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 SSO, you define 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.

SSO stores an encrypted mapping between a user's Windows user ID and that user's credentials for one or more affiliate applications in the Credential database. When a user needs to access an affiliate application, an SSO server can look up the user's credentials for that application in the Credential database. The following figure shows how this works.



- 1 =Get SSO ticket for user X
- 2 =Redeem SSO ticket
- 3 =Get user X's credentials for affiliate application
- 4 =Return user X's credentials for affiliate application
- 5 =Send message with user X's credentials for affiliate application

### Enterprise Single Sign-On

In this example, a message sent by some application to BizTalk Server 2004 is processed by an orchestration and 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 (for example, the right user name and password) are sent with the message when it is passed to the affiliate application.

The figure illustrates a five-step process:

1. When a receive adapter gets a message, the adapter can request an SSO ticket from SSO server A. This encrypted ticket contains the Windows identity of the user that made the request and a time-out period. (Do not confuse this with a Kerberos ticket—it is not the same thing.) After it is acquired, the SSO ticket is added as a

property to the incoming message. The message then takes its normal path through the BizTalk Server 2004 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.

2. 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, supplying the message (which contains the SSO ticket) that it just received and the name of the affiliate application for which it wants to retrieve the credentials.
3. This operation, called redemption, causes SSO server B to verify the SSO ticket, and then look up this user's credentials for that application.
4. SSO server B returns those credentials to the send adapter.
5. The send adapter uses the credentials to send an appropriately authenticated message to the affiliate application.

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 enable you to monitor these operations and set various audit levels. Other tools enable you to disable a particular affiliate application, turn on and off an individual mapping for a user, and perform other functions. There is also a client utility that enables end users to configure their own credentials and mappings. And like other parts of BizTalk Server 2004, 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 preceding example shows what is likely to be a typical use of Enterprise Single Sign-On, but it is not the only option. A smaller BizTalk Server 2004 installation might have only a single SSO server, for example, and it is even possible to use Enterprise Single Sign-On independently from the BizTalk Server 2004 engine. Because business processes implemented by using BizTalk Server 2004 need to interact with diverse applications, however, making this service part of the product makes good sense.

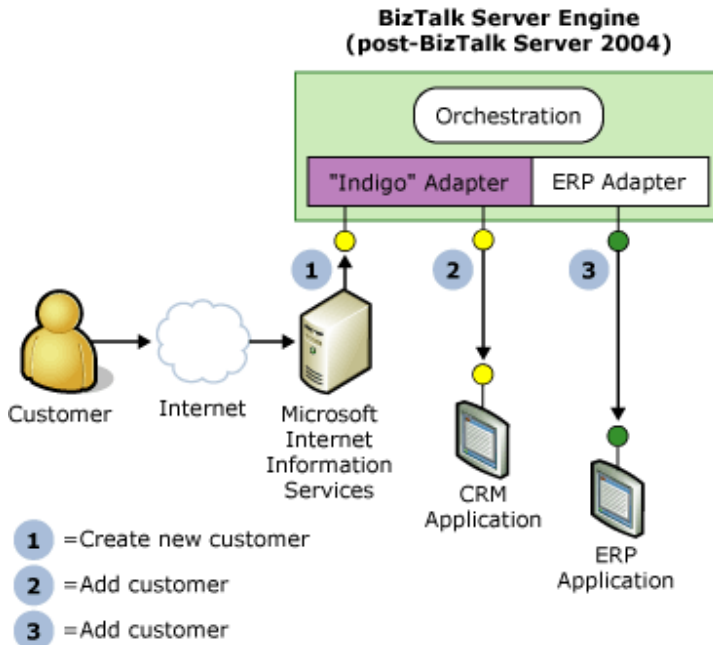
## Looking Ahead: The Engine and "Indigo"

"Indigo" is the code name for a forthcoming Microsoft technology to support applications that expose services through secure, reliable, and transactional messaging. Implemented with the .NET Framework, it is the successor to several existing technologies, including .NET Remoting, ASP.NET Web services, and Enterprise Services (COM+). Designed for a world of Web services, "Indigo" also supports industry-standard specifications such as WS-Security, WS-ReliableMessaging, and WS-AtomicTransaction. Because of this, software built on "Indigo" can communicate by using secure, reliable, and transactional messaging with other software running on any system that implements these specifications.

BizTalk Server 2004 supports Web services through the SOAP adapter and ASP.NET. However, after "Indigo" ships—which will be with the next version of Windows, codenamed "Longhorn," in 2005/6—it would make sense for the BizTalk Server engine to rely instead on this more full-featured platform. Accordingly, the BizTalk Server

release following BizTalk Server 2004 will include an adapter that allows sending and receiving messages through "Indigo."

The following figure shows how this adapter might be used.



### Using the "Indigo" adapter

This example shows three steps:

1. A new customer is created through the Web.
2. An orchestration adds the customer's information to a Customer Relationship Management (CRM) application. Both of these first two interactions rely on the "Indigo" adapter, and so both can potentially use the services provided by WS-Security, WS-ReliableMessaging, and other Web services standards. The day when every application supports Web services is far in the future, however, and so the other adapters provided by BizTalk Server will continue to be important.
3. Information about the new customer is also added to an ERP system such as SAP. This communication relies on a product-specific adapter rather than on "Indigo," an approach that will be common for some time to come.

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 "Indigo" will allow BizTalk Server to better meet this requirement.

## Information Worker Technologies

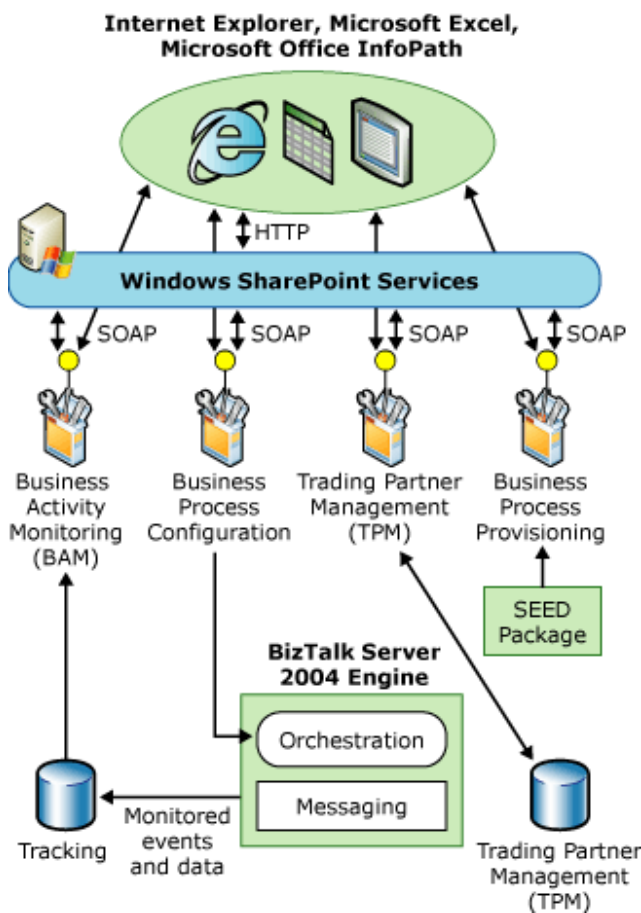
The BizTalk Server 2004 engine is an effective way to create business processes that span multiple applications. But after those processes have been created, the information workers that use them—business people, not developers—have other requirements. They might need to monitor various business-related aspects of the

activity, or perhaps create processes with human intervention, enabling people to step in where necessary. Unlike previous versions of BizTalk Server, BizTalk Server 2004 provides several components that address these needs. Those components are grouped into two categories, called Business Activity Services (BAS) and Human Workflow Services (HWS). This section describes both.

## Business Activity Services

An information worker might want to do many things with a running business process. A sales manager might want to look at sales trends or examine other aspects of a business process in real time. A business analyst might need to create a relationship with a new trading partner, defining the partner’s role, the business agreement between the two firms, and other aspects of this new association. A purchasing manager might need tools that can wrap together and distribute what is needed to let a partner quickly implement and begin participating in a business process. In BizTalk Server 2004, all of these capabilities are provided by Business Activity Services. By making it easier to track what is happening with business processes, these services enable less-technical people, whether they are in big organizations or smaller companies, to monitor their environments effectively.

As shown in the following figure, a common user interface to all of these services is provided through Microsoft Windows SharePoint™ Services, Internet Explorer, Microsoft Excel, and Microsoft Office InfoPath™.



## Business Activity Services

Because Business Activity Services are meant to be used by business people, and not by developers, it makes sense to expose them through these familiar tools. Behind this common interface are four different software components, all of which can communicate with Internet Explorer, Windows SharePoint Services, Excel, and InfoPath through SOAP. This section describes each of these components.

## Trading Partner Management

Connecting 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 be complex, especially when many organizations are involved or when the players change frequently.

To enable information workers to perform these common tasks, the BizTalk Server 2004 Business Activity Services include a Trading Partner Management (TPM) component. This component relies on a TPM database, shown in the preceding figure, 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 2004. Each agreement describes the relationship between two parties, and contains information including:

- 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 2004 adapter) should be used to communicate with that partner.

- The business process itself, implemented as one or more orchestrations, along with the role that 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. The next section describes how these parameters are used.

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

## Business Process Configuration

It is too much to expect an information worker to create the orchestration that implements a business process. It is not unreasonable, though, to expect an information worker to be able to set parameters of that orchestration. One example of where this would be useful is when the same orchestration is used with multiple partners, but each partner requires slightly different behavior. Suppose, for example, that the maximum dollar value of a purchase order is different for different trading partners, or perhaps the maximum quantity that can be requested varies depending on

the customer's credit rating. Making these small configuration changes should not require a developer to modify the orchestration. The Business Process Configuration service in BizTalk Server 2004 enables information workers to make the modifications

To enable information workers to configure an orchestration, the developer creating it can define parameters for that orchestration. An information worker can then set these parameters as needed, perhaps assigning different values for different business partners or different parts of the organization. An information worker sets the parameters for a partner through the TPM service, described in the previous section, by specifying their values in the addendum to the partner's agreement. If an agreement references multiple orchestrations, multiple addendums can be created, one for each orchestration.

In BizTalk Server 2004, the Business Process Configuration service is tied to TPM, because parameters can be set only through addendums. In future releases, however, the intent is to make this service more generally available, which is why it is called out separately in this version of the product.

## Business Process Provisioning

When several organizations are participating in a business process built around BizTalk Server 2004, getting them all correctly configured can be time-consuming. For example, a common B2B topology is a hub-and-spoke, where one organization, often a big company, establishes connections with many different suppliers. The business process is typically defined by the hub organization, and then passed on to the spokes to implement their parts. Something similar can also happen inside a single organization, where one group wishes to involve others in a common internal process. In cases like these, it is useful to have a simple way to configure the systems involved, a process that is sometimes called provisioning.

The fourth component in the BizTalk Server 2004 Business Activity Services, Business Process Provisioning, addresses this challenge. Using this component, an information worker can package the information needed by other participants in a business process into a SEED package. The SEED package can then be deployed to other BizTalk Server 2004 systems to quickly provide them with what they need to play their parts.

A SEED package created by a hub system can contain a broad range of things, including .NET assemblies for the orchestrations that implement a business process, BAM views and BAM activities, a profile for the system generating the package, and much more. After a spoke system receives a SEED package, its administrator can deploy it, and then return a SEED package of its own that contains its profile and the other information required to successfully interact with the hub. Yet while administrators are required to deploy SEED packages, the Business Process Provisioning component of Business Activity Services lets information workers create these packages, allowing one more task to be performed by business people rather than by technicians.

## Business Activity Monitoring Framework

Information workers need to look at business processes in different ways. For example, a purchasing manager might need to see how many purchase orders are approved and denied each day, or 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 is going on with a particular business process. This is exactly what the Business Activity Monitoring (BAM) Framework provides.

To use BAM, an information worker accesses the standard Windows SharePoint Services-based interface with Internet Explorer. The information worker can select a particular business process, and then choose a specific BAM view into the process. Each view can give a different perspective on this business process. For example, a BAM view might provide graphical depictions of per-product sales trends, current inventory levels, or other key performance indicators. The information in these views might be updated every day, every hour, or more frequently.

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.

Both BAM activities and BAM views are created by an information worker, and not by a developer. The BAM Activity Wizard enables defining activities, while the BAM View Wizard enables defining views based on those activities. BAM views are implemented in Excel, so the BAM View Wizard really just helps an information worker to build a standard Excel pivot table using the information in one or more BAM activities. A BAM view can be quite complex, and its creator can also control which users are allowed to see the data it exposes. For example, a purchasing manager might be able to access certain things in a view into the purchase order process that are hidden from purchasing clerks. Information workers can also modify BAM views after they have been created.

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, 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 enables 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 2004 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 Business Activity Monitoring component. As the figure shows, Excel can directly access the BAM component through SOAP to get up-to-the-minute information for BAM views.

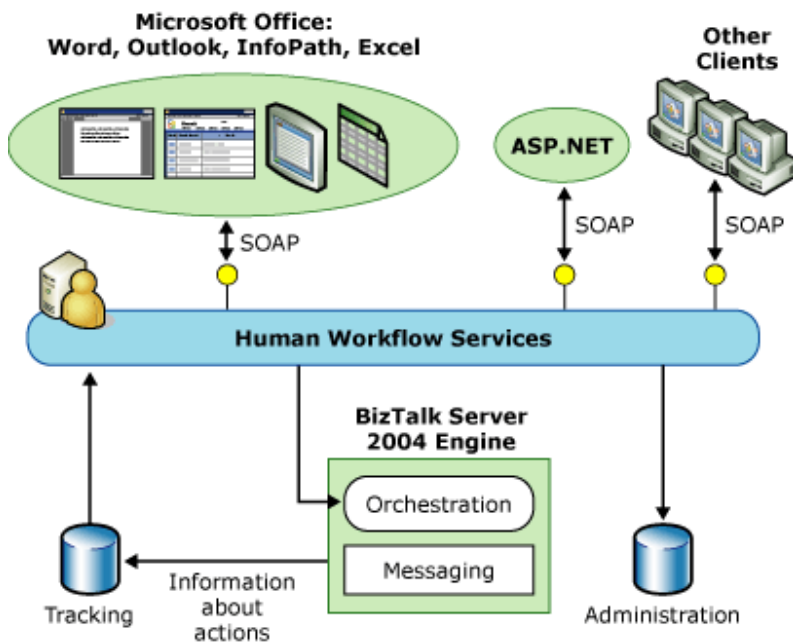
## Human Workflow Services

The BizTalk Server 2004 engine enables you to connect applications to carry out a business process. But what if that process requires human intervention? Think about handling a purchase order (PO), for example. An organization might require human approval at various stages in this process, and the people involved might change depending on the PO's size. Or imagine a request for proposal (RFP) arriving in a sales office. Creating a response might require a shifting group of people working together over days or weeks. Human-oriented business processes like these, commonly called "workflows," can certainly benefit from automation, but the BizTalk Server 2004 engine

by itself is not sufficient. More is required to enable people to interact with and control the process.

These extras are provided by Human Workflow Services (HWS), a standard part of BizTalk Server 2004. This service provides a workflow infrastructure built on the BizTalk Server 2004 engine. The HWS infrastructure is accessed through Web services, and so it can be used by any client application. Among the most important clients, however, are the applications in Microsoft Office. Word, Outlook, Excel, and InfoPath are fundamental to how many information workers work with information, and so it makes sense to allow these common tools to be the environment from which people participate in workflows.

The following figure shows the basic structure of Human Workflow Services. As just described, the technology relies on the BizTalk Server 2004 engine, as well as an Administration database and the Tracking database (also used by Business Activity Services) that holds information about the various actions that make up this workflow. Outlook and Word are likely to be the most common clients, along with custom forms built by using InfoPath. ASP.NET applications and other applications that can access standard Web services can also use this part of BizTalk Server 2004. For clients built on the .NET Framework, Human Workflow Services provides a library that exposes all of its Web services as .NET-based objects. And although it is not shown in the diagram, applications are administered by using the HWS Administration console, an MMC snap-in.



### Human Workflow Services

To get a sense of how all of this might be used, imagine that an application based on Human Workflow Services has been created to help an organization automate its response to an RFP. The process of handling the RFP might be started by the original recipient of the request invoking a Web service that initiates the workflow. In Microsoft Office 2003, Web services can be invoked from Outlook, from an InfoPath form, from a new feature in Word called SmartDocs that allows a document to contain embedded

actions, and in other ways. The RFP and any associated documents, such as a draft response, might be sent to multiple people involved in the process, each of whom takes some action. Several people might need to approve the response, for example, and so each one can directly interact with the workflow application from an Outlook message requesting this approval.

Many other activities are also possible, depending on how the workflow is defined. An Office 2003 SmartDocs document can contain a list of people to whom a task can be delegated, for example, and a user can click the name of the person to whom this task should be assigned. Doing this invokes a Web service that causes the Human Workflow Services application to delegate the task to that person. A user can also access an ASP.NET page that graphically depicts the status of a particular workflow, indicating who has and has not taken various actions, who has delegated tasks (and to whom they were delegated), and more.

To support this quite general approach to workflow applications, Human Workflow Services defines a few core abstractions, all of which are built on the BizTalk Server 2004 engine. Every workflow is built from some number of actions, each of which is actually implemented as an orchestration. For example, a workflow for responding to an RFP might have actions such as Review, Approve, Delegate, and Escalate. Like all orchestrations, those used with Human Workflow Services are created by using Orchestration Designer, but each one has a number of standard behaviors along with the customized behavior created by the developer who builds this application.

The actions in a workflow occur in a defined order, called an activity flow. Each activity flow has an activity model that defines the actions within it and how they relate to one another. The people who use a Human Workflow Services application are called actors, and communication between actors and actions happens through tasks, which are actually XML-defined messages. Each action has one or more tasks associated with it, and so when an actor clicks a button in an Office application that does something in a workflow, that actor is really sending a particular task message to some action (that is, to an orchestration).

A Human Workflow Services application can also impose constraints on the people who use it based on their roles. An application might allow only managers to approve POs of more than a million euros, for example, or limit who is allowed to delegate tasks to a vice president. To support this, the creator of a workflow application can define constraints that rely on roles defined in Microsoft Active Directory®, a SQL Server database, or in other ways.

Developers are certainly required to build Human Workflow Services applications. After they are created, though, those applications can be used solely by information workers. Workflows can be created, new actions can be added to the activity flow, existing actions can be interrupted, and more, all by the business people who use this application. While the BizTalk Server 2004 engine provides a way to connect software together, Human Workflow Services adds what is needed to let people participate directly in the business process.

## Conclusions

The roots of BizTalk Server are in integrating diverse applications. As the technology of Business Process Management (BPM) continues to evolve, however, BizTalk Server

2004 is growing into a hub for creating business logic of all kinds. This growth has some clear benefits. Imagine trying to create standard built-in mechanisms for monitoring business processes, for example, with logic built by using the .NET Framework (or for that matter, logic built by using a Java application server). And as organizations increasingly adopt service-oriented architectures based on Web services, the ability of BizTalk Server 2004 to expose diverse applications as a set of Web services with common XML schemas for business documents becomes more and more attractive. The support that BizTalk Server 2004 provides for human interaction with business processes, such as Human Workflow Services, also makes it an appealing platform for new application logic.

Managing business processes in a unified and centralized way, the goal of BPM, can lead to more effective and more adaptable organizations. Yet whatever happens, it is clear that BizTalk Server is growing up. BizTalk Server 2004 is a significant step forward from its predecessor, adding a number of useful new services. Just as important, it is also a broad foundation for what is to come as the Jupiter vision unfolds.

## About the Author

David Chappell is Principal of Chappell & Associates (<http://go.microsoft.com/fwlink/?LinkId=21434>) in San Francisco, California. His most recent book, *Understanding .NET*, was published by Addison-Wesley in 2002.