Chapter 1

Introduction to BizTalk Server
BizTalk Server is part of Microsoft’s stated goal to provide tools, products, and services to allow organizations to implement business-to-business (B2B) electronic commerce (e-commerce) over the Internet. This goal is called the BizTalk Initiative, and it consists of three main activities: a proposed messaging standard called the BizTalk Framework, a not-for-profit web site (http://www.biztalk.org) containing resources for organizations, and the BizTalk Server product itself.

This book will tell you everything you need to know (and almost everything there is to know) about Microsoft BizTalk Server the product. In a nutshell, BizTalk Server provides a rich set of tools and services to enable not just B2B integration, but also Enterprise Application Integration (EAI), workflow management, and business process automation. However, before you can properly understand the product, it’s important to appreciate its background and the larger picture surrounding its conception. Let’s begin by taking a look at how the BizTalk Initiative came about.

Why BizTalk?

The whole idea behind BizTalk Server grew out of an earlier product—Microsoft Site Server 3.0 Commerce Edition. Just as Microsoft was making its initial foray into the realm of business-to-consumer (B2C) e-commerce with Site Server, many other Internet heavyweights were taking a serious look at another aspect of this burgeoning industry. This aspect—business-to-business (B2B) e-commerce—was only lightly addressed by the Microsoft product, however, because companies were only starting to appreciate its potential. The reason why it would eventually become more lucrative than B2C e-commerce can be summed up in one statement: Businesses have more money to spend than consumers do. Nobody could accuse Microsoft of being blissfully ignorant of the potential of B2B e-commerce, however. Site Server Commerce Edition already had groundbreaking support for an emerging data interchange format known as the Extensible Markup Language (XML), and it attempted to address the issues of B2B exchanges with its Commerce Interchange Pipeline (CIP) feature.

The CIP allowed businesses to package order information using this new XML format and to transport it to a trading partner. This trading partner might be responsible for sourcing parts for the customer order, providing warehousing facilities, or collecting and delivering the order itself. Normally, in this kind of relationship, there is a flow of information that begins with the customer’s order. For example, if a customer places an order for a new PC on an e-commerce web site, the online store might have to contact its supplier to see whether the item is in stock. If it’s not in stock, the supplier may need to contact the manufacturer. The manufacturer may not have all the parts in stock to build the product, such as memory chips or CD-ROM units. Therefore, it might have to contact a wholesaler. The wholesaler will also have its suppliers, and so on. This kind of flow is called a supply chain, or value chain, and it is this kind of process that Microsoft wished to address with the BizTalk Initiative.
The BizTalk Initiative

The BizTalk Initiative is made up of several parts. Foremost, Microsoft hoped to position BizTalk as a messaging framework for the various supply-chain documents that would be buzzing around the Internet. This framework would attempt to do over the Internet what Electronic Data Interchange (EDI) had already accomplished for business partners across proprietary value-added networks (VANs) during the previous 20 years. Moreover, the framework would take advantage of the increasingly popular XML data format, as opposed to the complicated flat-file formats used by EDI standards.

Although Microsoft has been the driving force behind this initiative, as with similar ventures, they have been supported or assisted by other leading players in the industry. In fact, the BizTalk Initiative was originally under the auspices of a BizTalk Steering Committee consisting of such diverse companies as Ariba, Boeing, CommerceOne, Compaq, Dell, J.D. Edwards, Reuters, RosettaNet, SAP, Siebel, UPS, and WebMethods. Many of these companies still partner with Microsoft today on BizTalk- and XML-related technologies.

The Framework

The BizTalk Framework was announced in March 1999 as a new cross-platform e-commerce framework for establishing a communication standard for companies and making it easy for businesses to integrate applications and conduct business over the Internet with their trading partners and customers. The BizTalk Framework sets the tone for how documents should be exchanged and is based on standard data structures called XML Schemas and an XML-based protocol for integrating disparate applications across distributed networks—the Simple Object Access Protocol (SOAP). These standards enable integration across industries and between business systems, regardless of platform, operating system, or underlying technology.

A schema is an XML-based definition of the structure of an XML document. Schemas are discussed in more detail in Chapter 2.

The Web Site

After creating the BizTalk Framework, Microsoft (and the BizTalk Steering Committee) obviously wanted to publicize it in the hope that it would be adopted by the industry. To do this, they created a web site (BizTalk.org) that would serve as an online resource for anybody interested in the BizTalk Initiative. However, Microsoft often encounters skepticism when they try to promote an emerging technology as a standard. For this reason, they deliberately chose not to brand BizTalk.org as a Microsoft site, hoping instead that it would serve as a neutral playing field for concerned parties. On the site, they also created an online repository where organizations could upload XML schemas for the benefit of others, because it makes sense for people to use the same data structures whenever possible. Unfortunately, the site never really lived up to its potential, and
although it still exists, Microsoft rebranded the site early in 2001 with a more traditional style and color scheme, as shown in Figure 1-1. However, neither this web site nor the BizTalk Framework is the jewel in the BizTalk Initiative crown.

The Product
Microsoft BizTalk Server is the product that directly addresses the primary goal of the BizTalk Initiative in a tangible way. BizTalk Server provides not only for business-to-business integration, but also for enterprise application integration. In fact, “integration” is the key point here. To successfully implement BizTalk Server in an enterprise, it should not be necessary to throw away systems; BizTalk Server complements legacy systems, allowing communication where none was possible before.

The product has undergone many dramatic changes since its conception, reflecting the evolution of the industry and the requirements of those who shape it. In September 1999,
Microsoft released the BizTalk Jumpstart Kit—a collection of tools to help developers design BizTalk Framework 1.0–compliant schemas and to integrate XML into their applications. This was released simultaneously with the launch of the BizTalk.org website. Version 2.0 of the toolkit followed in February 2000, and this release included a Software Development Kit (SDK) that would assist developers in the creation of adapters to allow XML-based data flow between applications. These would serve as the precursors to Application Integration Components (AICs), which still exist in the current version of the product to provide the various outbound transports.

In April 2000, the first version of BizTalk Server proper made its debut as the BizTalk Server Technology Preview. The final product was now starting to take shape. Graphical tools such as the BizTalk Editor and BizTalk Mapper made the creation of document specifications and the transformation of those specifications simple. The BizTalk Management Desk (now called BizTalk Messaging Manager) allowed the creation of partner agreements (channels) and pipelines (messaging ports) to effect the exchange of business documents with external trading partners. Out-of-the-box parsers provided support for common data formats such as ANSI X12 and EDIFACT, and application adapters (AICs) allowed the product to communicate with enterprise resource planning (ERP) systems such as SAP.

Over the next few months, excitement grew for the potential of the final product. One more milestone in the development cycle seemed to secure its success—the inclusion of BizTalk Orchestration in the first public beta of the product, released in July 2000. This new feature introduced the idea of business process automation, with a graphical interface based on the newly acquired Microsoft Visio. This interface, called the Application Designer (now the Orchestration Designer) allowed a business process to be graphically described, with each action implemented by a developer through technologies such as Component Object Model (COM) components and Message Queuing. This business process drawing could then be compiled into an executable program called an XLANG schedule, where XLANG is an XML-based language that defines how the process should run.

Finally, BizTalk Server 2000 was released to manufacturing in December 2000, boasting a complete infrastructure for business-to-business integration, enterprise application integration, workflow management, and business process automation. Before we get into detail on the services and tools provided by the product, however, let’s take a step back to examine another aspect of the BizTalk Initiative. If you think about it, the only reason the product is called BizTalk Server in the first place is because it is fully compliant with the BizTalk Framework 2.0.

**The BizTalk Framework**

The Microsoft BizTalk Framework 2.0 is a framework based on the Extensible Markup Language (XML) + Namespaces 1.0 and Simple Object Access Protocol (SOAP) 1.1 specifications.
for application integration and e-commerce. It includes a design framework for implementing XML schemas and a set of predefined XML tags—called BizTags—that can be used to wrap documents sent between applications. These tags serve to provide descriptive and routing information not present in the document itself.

Schemas

Because the BizTalk Framework uses the recognized standard XML for describing documents, it makes it easier for other vendors and organizations to adopt the Framework and build on it. When the Framework was originally developed, Microsoft had also created a way of describing XML documents called schemas. Schemas describe the structure of an XML document using XML, if you can believe that. (No, this does not create a circular reference—you’ll see how this works in Chapter 2.) However, while Microsoft was promoting its schema technology, called XML Data Reduced (XDR), the World Wide Web Consortium (W3C), a nonprofit organization devoted to the standardization of technologies for the Internet, was also working on a similar specification. The long wait for W3C schemas to appear is often cited as one reason why many companies held back from adopting the BizTalk Framework.

Either way, the intent was that interested software companies or industry standards bodies could use the BizTalk Framework to produce XML schemas that defined not only their business documents, but also how they should be processed. This processing, or routing, information is contained in the extra BizTags that envelop the document. The resulting envelope must comply with the SOAP 1.1 specification. SOAP is an XML-based syntax that defines an envelope for calls to methods exposed by an application and the data returned by those method calls. The SOAP 1.1 envelope forms the basis for the BizTalk Framework message structure shown in Figure 1-2.

The BizTalk Framework itself is not yet a standard, but the language used to define it—XML—is. One of the goals of the BizTalk Initiative is to accelerate the adoption of XML and XML schemas as the underlying data format and specification language for business document exchange. BizTalk Framework schemas, which are business documents and messages expressed using an extension of XML schemas, can be registered and stored on the BizTalk.org web site. Anyone can download the BizTalk Framework from this web site and use it to define their own XML schemas. As long as the schemas pass a validation test, they may be referred to as BizTalk Framework schemas. The BizTalk.org web site also provides an automated submission and validation process. Individuals or organizations can freely use XML schemas from the BizTalk.org web site within their applications for as long as the schema is published for public use. Businesses also have the option of publishing their schemas on the BizTalk.org web site in a secure area for private use between trading partners.

**Note**

The full BizTalk Framework 2.0 specification is reproduced in Appendix A.
The end result is that the BizTalk Framework describes business documents as BizTalk Messages. BizTalk messages are an extension of SOAP documents, with additional BizTalk-specific tags referred to as BizTags. A server that implements the BizTalk Framework should utilize these BizTags as specified in the Framework. Microsoft BizTalk Server is, so far, the only publicly available BizTalk Framework–compliant software, although other companies are developing similar BizTalk server products.

Benefits

I must stress that you do not have to implement the BizTalk Framework to use Microsoft BizTalk Server. However, if your business and your trading partners are already using XML-based document structures, there are plenty of reasons why you should. Some of the benefits of using the BizTalk Framework are as follows:

- **Roadmap for consistent XML implementations**  Many companies report a strong interest in XML. XML, however, is so flexible that this is like expressing
a strong interest in ASCII characters. XML enables advancements, but they are hard to achieve without a consistent framework for XML implementations. The BizTalk Framework implements a set of rules that makes it possible for a broad audience to adopt a common approach to using XML. As companies move beyond data modeling using XML and start automating business processes, BizTalk Framework messages define the necessary routing information for processing systems.

- **Easier mapping across schemas**  
  By formalizing the process of expressing business process interchanges in a consistent and extensible format, the BizTalk Framework makes it easier for **Independent Software Vendors (ISVs)** and developers in a wide variety of industries to map from one business document structure to another. This promotes the adoption of electronic interchange using open standards such as XML.

- **Design target for software vendors**  
  By establishing a critical mass of schemas implemented in a consistent format, the BizTalk Framework provides a clear design target for tools and infrastructure ISVs building the next generation of e-commerce and application integration products.

- **Framework for standards bodies**  
  The BizTalk Framework provides a means for migrating an existing set of industry interchange standards to XML. This is especially useful for the Electronic Data Interchange (EDI) community.

- **Repository for BizTalk schemas**  
  The BizTalk.org web site is an interactive place where industry groups and developers can publish their schemas. The web site allows public and private publication based on the decision of the publishing organization. Once a BizTalk Framework schema is accepted and published, the repository will provide versioning and specialization support for adoption and alteration. The repository will support dynamic detection of schemas, processes, and visualization maps connected to any given version of a BizTalk Framework schema.

- **Showcase for the best practices in developing XML interchanges**  
  Many organizations involved in the standardization of business interchanges are more skilled in business process modeling than in systems programming and XML. These groups can use the BizTalk.org web site to discover best practices for implementing their own schemas or to discover other XML schemas they can use in their applications.

Microsoft has vowed to support the BizTalk Framework in its product line and will publish XML schemas to the BizTalk.org web site for public use. It has also promised to implement complete support for the W3C schema recommendation after it has been finalized. This recommendation was published in June 2001, and it is clear that future versions of the Framework, and BizTalk Server, will be compatible with these schema structures. In fact, Microsoft has provided the facility to upload W3C-compliant schemas to the BizTalk.org site since early in 2001. Other software vendors supporting the BizTalk Framework have also made this commitment.
BizTalk Framework Architecture Principles

The BizTalk Framework is designed to foster application integration and e-commerce through data interchange standards based on XML. It assumes that applications are distinct entities and that application integration takes place using a loosely coupled, asynchronous approach to pass messages. There is no need for a common object model, programming language, network protocol, database, or operating system for two applications to exchange XML messages formatted using the BizTalk Framework. The two applications simply need to be able to receive, parse, process, and transmit a standardized XML message.

Messages underlie the most significant contributions of the BizTalk Framework. A message flow between two or more applications is a means to integrate applications at the business-process level by defining a loosely coupled, request-based communication process. Since many business processes involve one party performing a service at the request of another party, the mapping of messages to requests is natural. Application solutions making tighter integration demands, such as those based on special programming languages or shared distributed computing “platforms,” are highly appropriate to tightly connected applications on single machines or in controlled environments. They do not, however, adequately support distributed, loosely coupled, extensible business process integration. An XML-based messaging system with open, extensible wire formats captures the essentials of a business communication while allowing flexible implementations.

Microsoft anticipates that the vast majority of interchanges (messages containing one or more documents wrapped in an envelope) exchanged using the BizTalk Framework will use a simple HTTP (HyperText Transfer Protocol) post transport. But businesses can also use other transports including FTP (File Transfer Protocol) and messaging technologies, such as IBM’s MQSeries, Microsoft Message Queuing 2.0, and SMTP (Simple Mail Transfer Protocol). As we will see in the next section, most of these transport methods enjoy out-of-the-box support in Microsoft BizTalk Server.

One other consideration is that since a limited (but growing) number of software applications provide support for XML, many businesses will need to implement specialized adapters to enable their existing applications to participate in the first generation of BizTalk Framework exchanges. For many applications, these adapters take an existing function call, translate it into an XML document, and route the document to a target destination, whether it is a trading partner or another application within a corporate intranet. As discussed earlier, these adapters may be implemented in Microsoft BizTalk Server as Application Integration Components (AICs).

If you would like to read the complete BizTalk Framework 2.0 specification, it is included as Appendix A, because it is a bit more in-depth than necessary for an introductory chapter such as this. Next up, we will examine in more detail how Microsoft BizTalk Server implements the BizTalk Framework and how it provides tools and services to simplify the integration of dissimilar systems.
BizTalk Server

As you’re probably aware if you’ve read this far, BizTalk Server is a product from Microsoft that addresses the requirements of business-to-business integration, enterprise application integration, workflow management, and business process orchestration. To accomplish these feats, the product is built on core services and supporting tools to assist in the rapid development and deployment of BizTalk solutions. Before I discuss these tools and services, however, let’s make sure we know what we mean by “enterprise application integration” and “business-to-business integration.”

Enterprise Application Integration

Enterprise application integration may be defined as the cooperation of disparate systems and components to implement business rules in a distributed environment. Or, in less fancy words: different types of computer systems within an organization talking to each other. Although we can express the concept simply, its implementation can be far from simple. Getting different types of systems to talk to each other typically involves the following:

- The transportation of information between one or more applications, perhaps involving the transformation of data into a particular format
- Timing and sequencing rules that govern when and how the transportation and transformation take place
- Integrity constraints that determine the success or failure of the communication

For example, if my enterprise includes an enterprise resource planning (ERP) system, a customer relationship management (CRM) system, a web site, and a database, as shown in Figure 1-3, the business rules that govern my organization may require that orders arriving at my web site are first logged to my database. Some component, or middleware, will be necessary to effect this by ensuring that the order information is transformed into the correct hierarchy of records and fields that the database expects. The order may also need to be logged to the CRM system to be available for after-sales support queries, and again, the data will need to be in the correct format as expected by that system. The CRM system may also create an entry in the ERP system that provides information internally to accounting software and analysis tools. Once more, these systems will need to be able to communicate with each other for the smooth operation of the process. Typically, these pieces of software will have been developed separately in-house, or by a number of third parties, so it is extremely unlikely that they are designed to share information among themselves with no intervention.

Enterprise application integration may be achieved using one of three integration methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application programming</td>
<td>Programmatic interfaces provided by the software for synchronous communication</td>
</tr>
<tr>
<td>interfaces (APIs)</td>
<td></td>
</tr>
</tbody>
</table>
Method Description

Messaging The asynchronous sending and receiving of messages between applications through an intermediary

File transfer The importing and exporting of data to text or binary files in various formats

In each case, it may still be necessary to transform the data from one format to another as required by the interacting systems. Common data formats used by applications are as follows:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>Data encoded using the Extensible Markup Language (XML), which uses a hierarchical structure of elements, with each item of information described by tags.</td>
</tr>
<tr>
<td>EDI</td>
<td>Electronic Data Interchange (EDI) formats developed in the 1970s to define common business documents for transfer between trading partners. Two main standards are currently in use—ANSI X12, typically used in the United States, and UN/EDIFACT, used elsewhere.</td>
</tr>
</tbody>
</table>
Format | Description
--- | ---
Flat-file | Plain text in a simple hierarchical format, two or possibly three levels deep. There are two types—*positional*, where each item of data has a fixed length, and *delimited*, where special characters indicate the beginning or end of a data item.

*Note* | These data formats will be discussed in more detail in Chapter 2.

Furthermore, because these application systems may expect information to arrive by various means, enterprise application integration will also require the implementation of different transport mechanisms including, but not limited to, the following:

| Transport | Description |
--- | ---
DCOM | The Component Object Model (COM) provides a way for applications written using Visual Basic or Visual C++ to define public interfaces with methods that can be invoked programmatically for synchronous communication. Distributed COM (DCOM) describes such communication between remote networked applications. |
HTTP | The HyperText Transfer Protocol (HTTP) defines a standard synchronous communication system between clients that make requests, and file or application servers that recognize, accept, and respond to those requests in a uniform way. |
SMTP | The Simple Mail Transfer Protocol (SMTP) defines a standard asynchronous communication where specially formatted messages can be relayed across a network of servers. |
Message Queuing | Message Queuing provides a way for disparate application systems to asynchronously communicate through the sending of messages that are stored for later retrieval. This can circumvent problems faced by distributed applications, such as availability and network connectivity. |
File | Documents may be written to or read from a standard file folder on a local machine, or at a shared location on the network for asynchronous storage and retrieval. |

**Business-to-Business Integration**

If enterprise application integration can greatly improve communication and increase efficiency for a single organization, extending the idea to encompass customers and trading partners can provide benefits for all. Because the same difficulties will arise in
terms of disparate data formats, application architectures, and transport mechanisms, business-to-business integration is effectively nothing more than enterprise application integration conducted between multiple organizations, as opposed to within an organization.

In this context, business-to-business integration (and its sibling, business-to-consumer integration) should promote a seamless, consistent customer and supplier experience through real-time messaging workflow and automatic business process execution. I’m sure it seems obvious, but this is beneficial for a number of reasons. Traditional ways of doing business can be inefficient, even if both businesses claim to be electronically enabled, or e-enabled. For example, the lack of integration between systems can mean that an arriving purchase order must be manually converted into a corresponding invoice, and this reentry of information can sometimes lead to data errors. This involvement of human beings in the process also results in time delays between the arrival of a document and the transmission of an acknowledgement. By developing electronic business-to-business relationships where the processes are automated, companies can cut costs and overhead by taking the manual labor out of the loop.

Business-to-business integration can significantly improve your company’s bottom line by promoting many of the key principles of business success. These include the following:

- Being faster to market with your products and services
- Providing a more efficient customer service
- Operating a smoother sales process
- Reducing operational, production, and inventory costs

In Chapter 2, we will see how businesses have attempted to address these problems in the past by implementing Electronic Data Interchange (EDI). EDI describes the process by which trading partners can exchange business documents and also describes the format of those documents. This sounds like a suitable solution, and to many organizations it is, but EDI systems can be costly to develop and maintain. One of the primary reasons for this is that EDI exchanges typically take place over value-added networks (VANs), which are expensive leased lines operated between business partners. Since the advent of the Internet, various consortia have tried to migrate EDI techniques to use public networks with varying degrees of success. Some of the industry groups involved in this work include RosettaNet (http://www.rosettanet.org), the Organization for the Advancement of Structured Information Standards (OASIS) (http://www.oasis-open.org), and the BizTalk Steering Committee mentioned earlier.

Today, a wide range of technologies, data formats, and transport mechanisms is in use for both enterprise application integration and business-to-business integration. It has become obvious that no one technology, data format, or transport will replace everything currently available, so integration is the key. That is why BizTalk Server provides tools and services to promote integration between disparate systems, rather than trying to take their place.
BizTalk Server Services

BizTalk Server provides two core services to address the issues discussed previously. These are BizTalk Messaging and BizTalk Orchestration. Parts II and III of this book will go into complete detail on these services, also covering the supporting tools that enable the implementation of these services.

BizTalk Messaging Service

In my discussions of both business-to-business integration and enterprise application integration, I described how various data formats, structures, and transport methods are involved when disparate systems attempt to communicate. BizTalk Messaging provides a means for the differences exposed by dissimilar systems to be reconciled.

Let's look at a typical business-to-business example. Figure 1-4 shows a purchase order in an X12 format arriving from a remote trading partner via HTTP. Our business rules determine that an acknowledgement should be sent back using the same format as an e-mail, and that an invoice should be generated for the ordered goods and sent back to the trading partner’s web site using HTTP. The purchase order should also be logged with our ERP system and a copy stored in a database. Obviously, the data required by our ERP system may not be in the same format or structure as the original message; likewise, some transformation will be necessary to insert the details into our database.

BizTalk Messaging can be configured to accept the purchase order submitted from the Active Server Page (ASP) on the web server. It will then parse the document from

![Figure 1-4. A typical B2B scenario](P:\010Comp\CompRef8\498-4\ch01.vp)

Friday, October 19, 2001 12:21:29 PM
its original format into XML, transform the document as necessary using a special map file created with the Extensible Stylesheet Language for Transformation (XSLT), and serialize the XML data into the structure and format required by the ERP system. A custom Application Integration Component (AIC) can then be created to transport the data to the ERP system. Similarly, another AIC can be used to insert the information into the database. Finally, BizTalk Messaging can automatically construct both an acknowledgement and an invoice for return to the trading partner and can send them using SMTP and HTTP, respectively.

Although this example involves the programmatic submission of data from an ASP, this could also be achieved through programmatic submission from an Exchange Server script if the original purchase order had been sent using SMTP. Also, BizTalk Server provides File and Message Queuing receive functions to support those inbound protocols. Equally, although the example uses custom AICs to transport the transformed data to the ERP and database systems, SMTP to return the acknowledgement, and HTTP to return the invoice, BizTalk Messaging also provides out-of-the-box support for HTTPS, File, and Message Queuing transmission methods.

These workflows can be created using BizTalk Messaging Manager, a graphical utility shown in Figure 1-5. Using this application will be discussed in detail in Chapter 6. BizTalk Messaging Manager allows the creation and configuration of programmatic objects that represent the organizations and applications involved in a workflow, as well as the types of documents that are being processed, the transformations that might be needed for those documents, and the routing information necessary to deliver them to their final destinations.

**BizTalk Orchestration**

Many business processes are not just simple straight-through affairs such as those handled by BizTalk Messaging. Sometimes, a business process must be dynamic—capable of detecting different scenarios in real time and acting accordingly. A modern automated business process must be able to implement decision factors, perform tasks multiple times based on business rules, take account of transactions, and handle exceptions. For example, if a purchase request is generated by an intranet purchasing application, there may be expressly defined criteria that establish whether the request should be approved by a manager before it becomes an actual purchase order to be sent to a supplier. An automated business process should recognize these criteria and determine whether to seek approval, without the need for manual intervention.

Similarly, an automated process must be able to detect errors that occur within the process and react accordingly, perhaps undoing previously completed tasks.

Transactions are business processes that must either succeed or fail as a whole. To use a common example, if a customer is transferring money from one bank account to another, there are two parts to the process—a withdrawal from the first account, and a deposit into the second account. If something goes wrong in the middle of this process, we must be careful not to leave the system in an inconsistent state. Bank customers tend to get upset if they try to transfer funds and the money just disappears, so
developers have to ensure either that the entire process completes successfully or that none of it takes place. Likewise, exceptions are errors that can occur in a business process, perhaps even within a transaction. For the process to be fully automated, these exceptions must be handled to ensure the system remains in a consistent state.

BizTalk Orchestration allows automated business processes defined using an XML-based language called XLANG to execute on a computer running BizTalk Server. These processes are defined in a graphical application called BizTalk Orchestration Designer, which is based on Microsoft Visio. As Figure 1-6 shows, by drawing the business process using flowchart shapes and by using implementation shapes representing various technologies to actually perform each action, a developer can describe the entire sequence of events and how each step will be carried out. BizTalk Orchestration Designer also allows developers to implement transactions in their business processes and to take remedial action in the event of an exception or a failed transaction.

**BizTalk Server Tools**

To implement the workflows and automated business processes supported by BizTalk Messaging and BizTalk Orchestration, BizTalk Server includes a number of tools that can be used to create the components used by these services. Both Microsoft and some third parties have also created other tools and utilities to simplify the creation of other types of processes. Chapter 19 will discuss some of the accelerators for BizTalk Server currently available.
Because BizTalk Server primarily deals with business documents, it is important that we can represent different types of document structure. This will be necessary for BizTalk Server to parse incoming documents from their original format into XML and to serialize outgoing documents from XML into the format they will be required to have when they reach their destination. BizTalk Editor is a graphical utility for defining the structure and format of business documents. These structures will be stored as XML schemas, which will be discussed in Chapter 2. The creation of document specifications using BizTalk Editor will be covered in Chapter 4.

BizTalk Mapper
We have seen that different organizations represent their business data in different formats and different structures. Although the parsers and serializers included with BizTalk Server enable the transformation of documents from one format to another, we must also be able to restructure our business documents to reflect this diversity. BizTalk Mapper is a graphical utility for creating the map files that will convert one document
structure to another. It does this by defining an XSLT stylesheet that effects the necessary transformation on the XML-encoded documents used internally by BizTalk Server. The creation of these map files will be covered in Chapter 5.

**BizTalk Messaging Manager**

As discussed, BizTalk Messaging allows the execution of workflows that accept, transform, and route documents between applications and organizations. Within the service, this is performed by programmatic objects that represent each aspect of the process. BizTalk Messaging Manager is a graphical HTML-based interface that simplifies the creation of these objects and the configuration of the workflows that involve them. Using BizTalk Messaging Manager to define these objects is dealt with in Chapter 6.

**BizTalk Orchestration Designer**

In Chapter 7, we will see how to use the Microsoft Visio–based application called BizTalk Orchestration Designer to create automated business processes called XLANG schedules. We will also see how these processes can take advantage of decision-making, looping, transactions, and exceptions. In Chapter 8, the integration of BizTalk Messaging and BizTalk Orchestration to create more sophisticated processes will be discussed.

**BizTalk Administration**

Although we have only discussed the deployment of a single BizTalk Server for various purposes in this introductory chapter, it is important to realize that multiple BizTalk Server computers can be clustered into server groups. The creation and management of these server groups, as well as the configuration of each individual server, is performed through the BizTalk Administration interface, which is a Microsoft Management Console. This console also allows the creation of the receive functions previously mentioned that provide a means for documents to be submitted to BizTalk Messaging. The use of this tool will be discussed in Chapter 9.

**BizTalk Document Tracking**

Because BizTalk Server can process hundreds or even thousands of documents a second, you might think that once a business document passes through BizTalk Messaging or through BizTalk Orchestration, all data pertaining to the interchange is destroyed. This would not make good business sense, however, as it is often necessary to call up previous documents to provide customer service or to settle legal disputes. By default, all documents processed by BizTalk Messaging are stored in a database for later retrieval. There is also a web-based application called BizTalk Document Tracking to allow you to search for and view these documents. The Document Tracking application and the management of the database that stores each interchange will be covered in Chapter 10.

We’re almost ready to begin looking at each aspect of BizTalk Server in detail. Before we do, we should take a moment to see how BizTalk Server fits into the greater Microsoft plan. Like the BizTalk Initiative before it, Microsoft is now promoting another set of tools, technologies, and products that promise to change the way applications and businesses use the Internet to communicate. This initiative goes by the simple name of .NET.
The .NET Initiative

The .NET Initiative comprises a number of standards, technologies, and products. All serve to promote loosely coupled communication between distributed applications across the Internet. It is not the intention of this book to delve too deeply into the .NET Initiative or its technologies, but it is important to understand how it relates to BizTalk Server.

.NET is an evolving concept, and as such is perhaps too much of a moving target to address in this book. However, there are a couple of aspects that have been announced or are already available in one form or other. First, there are the .NET Enterprise Servers, of which BizTalk Server is one. Second, there is the .NET Framework, which describes a collection of services and classes that applications can use regardless of the language in which they are written or the operating system on which they run.

.NET Enterprise Servers

The .NET Enterprise Servers are the logical evolution of Microsoft BackOffice Servers for the Internet age. They are designed for interoperability and take advantage of open web standards such as XML. It is an important indication of Microsoft’s commitment to the .NET Initiative that they have rebranded almost every product as a member of the .NET family, even if the brand fits some products better than others. Again, it is not the goal of this book to provide an in-depth analysis of each of these products, but because many of them will find a place in your enterprise alongside BizTalk Server, I will list them here with a brief description of their intended use.

Application Center 2000

Application Center 2000 is a deployment and management tool for high-availability distributed web applications built on the Windows 2000 or Windows.NET platform. It provides features such as component load-balancing and a health monitor for performance optimization. It also builds on the Content Replication Services originally provided by Site Server. Although it does not integrate directly with BizTalk Server, it could prove important in large solutions with multiple web servers that receive documents for BizTalk Server. In Chapter 12 we will see how Application Center can be used to provide load balancing for core products used by BizTalk Server 2000.

Commerce Server 2000

The successor to Site Server 3.0 Commerce Edition, Commerce Server 2000 provides key services and features to build an online business. It includes an extensible Business Desk application for the management of e-commerce web sites, ready-made solution sites to simplify the development process, and a full suite of analysis, prediction, profiling, and marketing tools. It integrates well with BizTalk Server to provide a complete value chain experience. In Chapter 14 we will explore how this integration can be achieved.
Content Management Server 2001
This is another recent acquisition by Microsoft that builds on the Content Management feature of Site Server 3.0 to provide document storage and retrieval facilities that again take advantage of XML. It integrates with SharePoint Portal Server to create complete collaboration solutions, and also with Commerce Server to simplify the publication of personalized content.

Exchange Server 2000
Exchange Server 2000 is an enterprise-level messaging and collaboration server, and it was the first of the .NET servers (though the term wasn’t used at the time), coming hot on the heels of Windows 2000 and making full use of the Active Directory technology built into the platform. BizTalk Server can use its messaging features and its active scripting support to send and retrieve documents.

Host Integration Server 2000
Probably the most unlikely server product to bear the .NET name, Host Integration Server 2000 provides connectivity with legacy systems and mainframes. It is the successor to Systems Network Architecture (SNA) Server 4.0. BizTalk Server could use an AIC to communicate with such systems.

Internet Security and Acceleration Server 2000
A complete revamp of Proxy Server 2.0, Internet Security and Acceleration Server is a full-featured firewall, caching, and proxy server for enterprise security. This will often provide the required network security for a BizTalk Server deployment. The use of firewalls and proxy servers to secure BizTalk Server will be discussed in Chapter 13.

Mobile Information 2001 Server
One of the later additions to the fold, as its name suggests, Mobile Information Server 2001 provides a secure, extensible platform for the delivery of enterprise application data such as e-mail and intranet content to mobile employees using hand-held devices. BizTalk Server could potentially integrate with it to transform application content for delivery to portable devices in a variety of formats.

SharePoint Portal Server 2000
Another late addition, this was originally planned as Site Server version 4.0 and retains many of that product’s features, such as Knowledge Management and Internet Crawl and Search. It provides a collaborative environment for companies, integrating heavily with Exchange Server 2000.

SQL Server 2000
Microsoft’s enterprise-level database product, this provides important back-end support to BizTalk Server and Commerce Server among others. It includes groundbreaking XML support, as well as data warehousing and analysis services.
Windows.NET Server
On the desktop, the next version of Windows after Windows 2000 (code-named “Whistler” during development) will be Windows XP Home Edition (home use) or Windows XP Professional Edition (office use). In the enterprise, however, it will be Windows.NET Server or Advanced Server that provides the platform for your applications, with full support for the .NET Framework. Windows.NET Server undoubtedly will be the platform of choice for later versions of the BizTalk Server product, such as BizTalk Server 2002 Enterprise Edition, which is discussed at the end of this book, and the next version of BizTalk Server currently in development, code-named “Mozart”.

.NET Framework
The .NET Framework is the overall term used to describe services such as the Common Language Runtime (CLR), the Base Class Library, and the Common Type System that underlie applications written using the next generation of Microsoft’s integrated development suite—Visual Studio.NET. One of the biggest problems in programming today is that it can be hard to be really productive because there are so many programming languages and technologies to learn. If you were, say, a Visual C++ developer, you might find it difficult to make an easy transition into Active Server Pages (ASP) development (or you may consider it beneath you). You would have to learn a whole new language (VBScript or JScript) and a whole new set of objects (the ASP intrinsic objects) to be able to create an ASP web application. One of the benefits of the .NET Framework will be to simplify the development process such that you will have a consistent set of objects and APIs available to you no matter what programming language or programming model you use.

One important goal of .NET is to simplify the development of distributed applications. By using open standards such as SOAP and HTTP, you can easily build distributed, disconnected, or stateless applications that are not tied into a specific platform or programming language. For example, you could build an application that resides on a Windows 2000 server and that communicates with an application on a Linux server. This is possible because .NET prefers standard protocols such as HTTP and SOAP instead of proprietary protocols like IIOP, CORBA, or DCOM.

To gain wide acceptance for the .NET Framework, Microsoft has submitted a subset of the .NET Framework called the common language infrastructure, and a new programming language called C# to ECMA (previously known as the European Computer Manufacturers Association, but now just an acronym!), which could lead to eventual multiplatform support. Indeed, there is already an open-source version of the .NET Framework in development for use with UNIX-based systems.

The core services in the .NET Framework are summarized here to give you an idea of the paradigm-shift that developers will face in becoming acquainted with these new technologies.
**Common Language Runtime (CLR)**
The CLR is one of the more compelling aspects of the .NET Framework. For example, through the CLR, developers will be able to create an interface containing object classes in C#, extend one of these classes using Visual Basic.NET (which now has true inheritance), and then make calls to it in an ASP application written in COBOL! This is because the CLR provides full language interoperability to any language that supports the .NET Framework. This is available not just at run time, as happens already today with COM components, but at design time as well. As mentioned, COBOL is but one of the supporting languages along with Perl, Python, Java, Pascal, Smalltalk, and a host of others. The CLR also provides important features useful to developers, such as exception handling, debugging, security, and garbage collection.

**Base Class Library**
This is a hierarchical system of classes and members similar to the Java class libraries included in the Java SDK that can be utilized by any .NET-compliant language. These classes are grouped into sets called *namespaces* (not related to XML namespaces) with System as its root. For example, a Visual Basic.NET developer could use the System.Windows.Forms namespace to create a button in a user interface or use the System.Data namespace to create a dataset containing information retrieved from a database.

**Common Type System**
This is a rationalization of the existing data type system in place across the different languages available in Visual Studio. Also, it is one of the fundamental aspects of the .NET Framework that must be supported by other compliant languages. The types include interfaces, classes, delegates, and value types, and conformance to the Common Type System provides much of the interoperability required by the CLR. One of the more interesting aspects is that for value types, the base is *object*, which means that even primitive data types such as string and integer can have properties, methods, and events.

**BizTalk Server and .NET**
So why would any of this be important to BizTalk Server? Although BizTalk Server is already fairly full-featured in its support for business-to-business integration and enterprise application integration, and although it already uses accepted standards such as XML and SOAP, and Internet transports such as HTTP and SMTP, there is always room for improvement. .NET promotes the use of special applications called *web services*. These are traditional applications, written using any type of architecture or programming language, that expose methods that can be called using SOAP across HTTP. In this way, an ASP application written in VBScript could include data provided from a Java application on a web server on the far side of the world. It doesn’t matter that the applications can’t communicate directly because SOAP acts as the go-between.
Obviously, BizTalk Server could make great use of this technology. As we will see in full detail in Chapter 7, one of the core features of BizTalk Server is BizTalk Orchestration, where automated business processes can call COM components on a local system or retrieve documents from a remote message queue. Taking this a step further, BizTalk Orchestration could benefit greatly if an XLANG schedule could call a web service to provide external processing of a document or supply real-time data. Although there is currently no web service implementation shape in the BizTalk Orchestration Designer, in Chapter 17, we will see how similar functionality might be implemented. Correspondingly, in the future, it will be possible to expose BizTalk Server itself as a web service, although again we can imitate this behavior now using a little ingenuity.

Before we get ahead of ourselves talking about future developments for BizTalk Server and Microsoft.NET, throughout this chapter you’ve probably noticed that three letters come up again and again. These three letters represent humanity’s best hope for eventually agreeing on a standard data format for the transfer of information. These three letters are the cornerstone of both Microsoft’s .NET Enterprise Servers and the whole .NET Initiative. In particular, these three letters are the way in which BizTalk Server represents data internally, regardless of its original or eventual formats as required by applications or trading partners. To be fully armed for a headlong odyssey through all that is BizTalk Server, you must become intimately familiar with those three letters—X, M, and L.