Microsoft BizTalk Server 2006 Part-1

Table of Contents

Getting Started

BizTalk Server interacts with your back-end and legacy systems to automate transactions between you and your business partners, and to automate your day to day business workflow.

Imagine being the parts manager for an automobile manufacturing company. Your database system informs you that you are low on door handles, so you need to act quickly. You fax a purchase order (PO) to a supplier, who sends you a PO acknowledgment, and ships you the order along with an invoice. You add the shipment to your inventory, pay the invoice, and update your database. And you repeat this process for thousands of other parts.

With BizTalk® Server installed, you arrive at your office in the morning to find that a shipment of door handles that you did not even know you needed is already on your receiving dock ready to be stocked. Not only that, the shipment has already been paid for and your database has been automatically updated.

If you do not already know how you want to use Microsoft® BizTalk® Server at your company, it is recommended that you start by exploring BizTalk Server features and functionality:

- Read the Beginner's Guide to BizTalk Server 2006.
- Find out what's new in BizTalk Server 2006.
- Perform the steps in the BizTalk Server 2006 Tutorials.

This section provides an overview of developing, managing, and using BizTalk Server 2006 features.

In This Section

- Prerequisite Skills and Knowledge
- Getting Started for Administrators
- Getting Started for Information Workers
- Getting Started for Solution Developers
- Getting Started for Component Developers
- Understanding BizTalk Server 2006
- What's New in BizTalk Server 2006
- Quick Start Guide to Installing and Configuring BizTalk Server 2006
- BizTalk Server 2006 Tutorials
- Getting BizTalk Server 2006 Assistance

- Community Resources
- Accessibility for People with Disabilities

Introducing BizTalk Server 2006

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

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

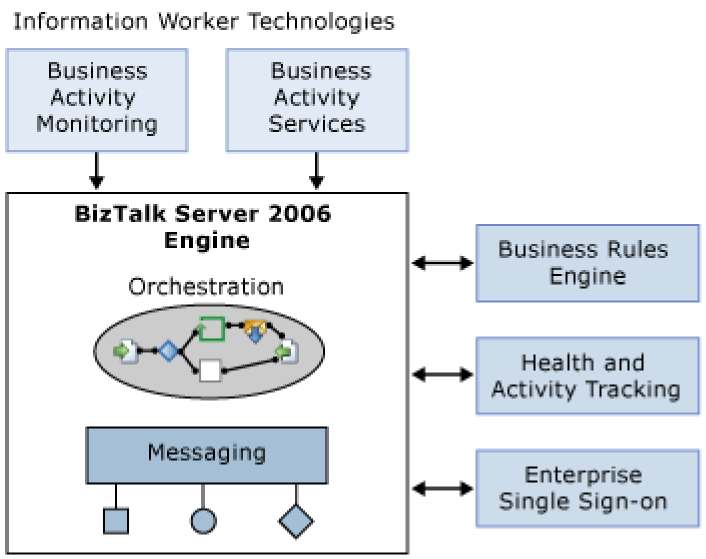
The most important new additions in BizTalk Server 2006 are:

- Better support for deploying, monitoring, and managing applications
- Significantly simpler installation
- Improved capabilities for Business Activity Monitoring (BAM).

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

What is BizTalk Server?

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


Information Worker Technologies
Business Activity Monitoring
Business Activity Services
BizTalk Server 2006 Engine
Orchestration
Messaging
Business Rules Engine
Health and Activity Tracking
Enterprise Single Sign-on

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

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

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

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

- A Business Rule Engine that evaluates complex sets of rules.

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

- An Enterprise Single Sign-On (SSO) facility that provides the ability to map authentication information between Windows and non-Windows systems.

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

- Business Activity Monitoring, which information workers use to monitor a running business process. The information is displayed in business rather than technical terms, and business users determine what information is displayed.

- Business Activity Services, which information workers use to set up and manage interactions with trading partners.

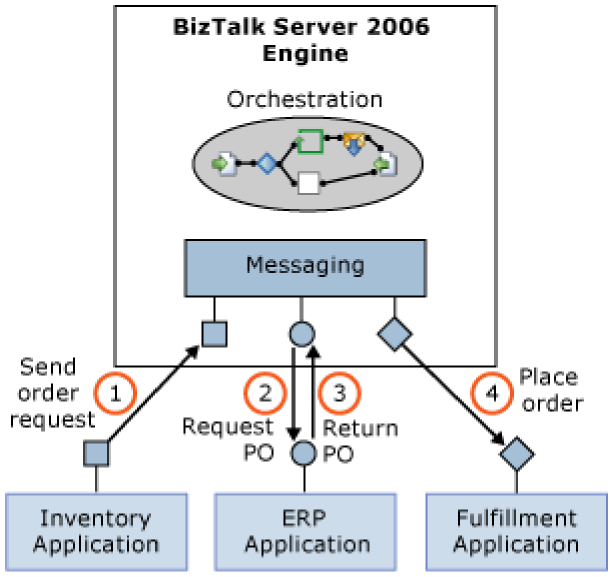
These technologies are focused on solving the problems inherent in using a diverse set of software to support automated business processes. The next section examines how these solutions might look.

BizTalk Server and the Challenge of Connecting Diverse Systems

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

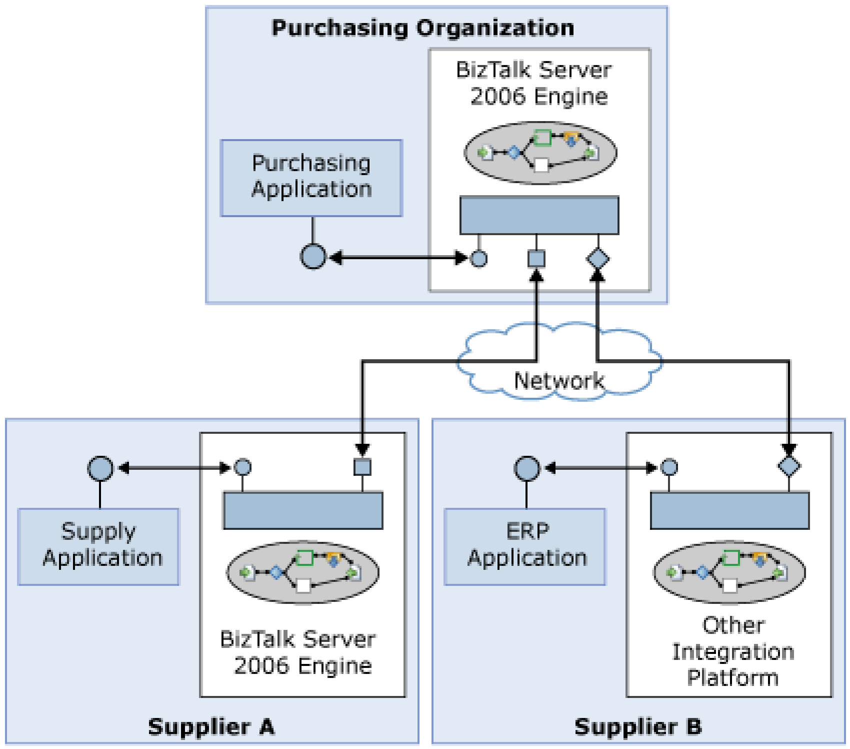
Addressing this challenge goes by various names: business process automation (BPA), business process management (BPM), and others. Regardless of the name, two scenarios are most important for application integration. One is connecting applications within a single organization, commonly referred to as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

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



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

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



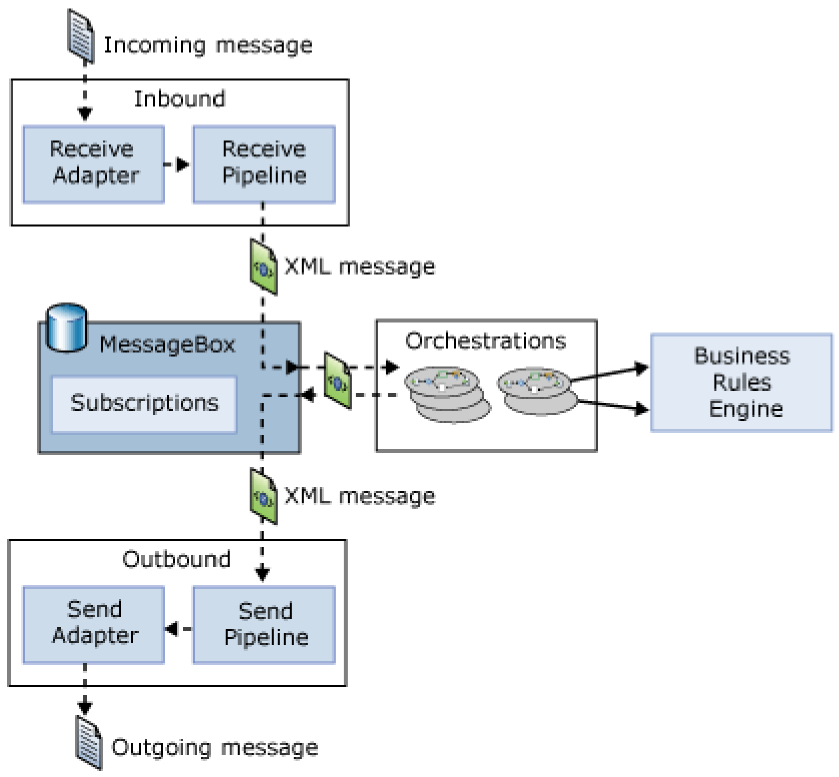
BizTalk Server 2006 Engine

The BizTalk Server 2006 engine enables users to create business processes that spans multiple applications by providing two primary things:

A way to specify and implement the logic driving that business process

A mechanism for communicating across the applications that the business process uses.

The figure below illustrates the main components of the engine that address these two problems.



As the diagram shows, a message is received 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. The message is then processed through a **receive pipeline**. This pipeline can contain various components that do things such as converting the message from its native format into an XML document, validating a message's digital signature, and more. The message is then delivered into a database called the **MessageBox**, which is implemented using Microsoft SQL Server.

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

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

A complete **solution** built on the BizTalk Server 2006 engine can contain various parts (sometimes referred to as artifacts): orchestrations, pipelines, message schemas, and more. These parts, or artifacts, can be worked with as a single unit, referred to as a **BizTalk application**. A BizTalk application wraps all of the pieces required for a solution into a single logical unit, making it the fundamental abstraction for management and deployment.

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

In This Section

- Connecting Systems
- Defining Business Processes
- Management and Monitoring
- Enterprise Single Sign-On [BTS]

Connecting Systems

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

Sending and Receiving Messages: Adapters

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

Microsoft BizTalk Server 2006 includes the following adapters:

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

- File adapter: supports 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.

- FTP adapter: supports sending and receiving information between a File Transport Protocol (FTP) Server and BizTalk Server.

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

- MSMQ adapter: supports sending and receiving messages using Microsoft Message Queuing (MSMQ).

- MSMQT adapter: supports sending and receiving messages using BizTalk Message Queuing (MSMQT). MSMQT is an implementation of the MSMQ protocol that can receive and send MSMQ messages into the MessageBox. Although BizTalk Server 2006 still includes this adapter, developers should use the MSMQ adapter instead.

- WebSphere MQ adapter: supports sending and receiving messages using IBM's WebSphere MQ (formerly known as MQSeries).

- SMTP adapter: supports sending messages using SMTP. Standard e-mail addresses are used to identify the parties.

- POP3 adapter: supports receiving e-mail messages and their attachments using version three of the Post Office Protocol (POP3).

- Windows SharePoint Services (WSS) adapter: supports accessing and publishing documents stored in Microsoft Windows SharePoint document libraries.

- SQL adapter: supports reading and writing information to a SQL Server database.

- Base EDI adapter: supports sending and receiving messages in the electronic data interchange (EDI) format.

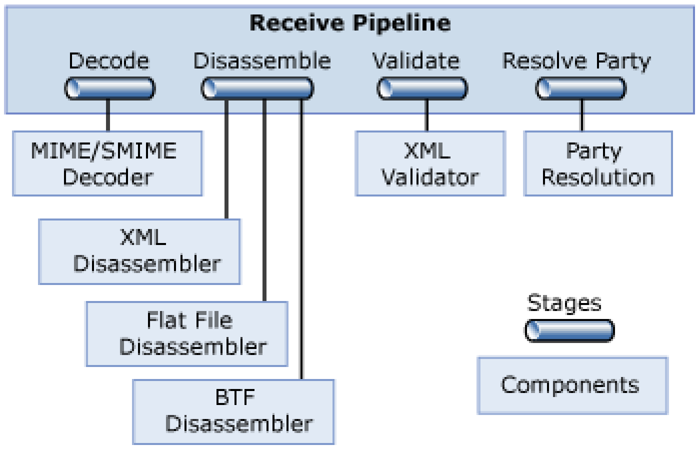
Adapters for commonly-used business software are also available from Microsoft, including adapters for Siebel, PeopleSoft, Oracle applications and Oracle databases, JD Edwards OneWorld and EnterpriseOne, TIBCO Rendezvous and Enterprise Messaging Service, and Amdocs Clarify.

Whichever adapter is used to receive data, the messages it receives must commonly be processed before they can be accessed by an orchestration. Similarly, outgoing messages produced by an orchestration often need to be processed before they are sent by an adapter.

Processing Messages: Pipelines

The applications underlying a business process communicate by exchanging various kinds of documents: for examples, purchase orders and invoices. For a BizTalk Server 2006 application to execute a business process, it must be able to correctly deal with the messages that contain these documents. This processing can involve multiple steps, and hence 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 applications are being designed to understand XML documents, many—probably the majority today—cannot. Because the BizTalk Server 2006 engine works only with XML documents internally, it must provide a way to convert other formats to and from XML. Other services may also be required, such as authenticating the sender of a message. To handle these and other tasks in a modular, yet customizable way, a pipeline is constructed from some number of stages, each of which contains one or more .NET-enabled or Component Object Model (COM) components. Each component handles a particular part of message processing. The BizTalk Server 2006 engine provides several standard components that address the most common cases. If these aren't sufficient, developers can also create custom components for both receive and send pipelines.



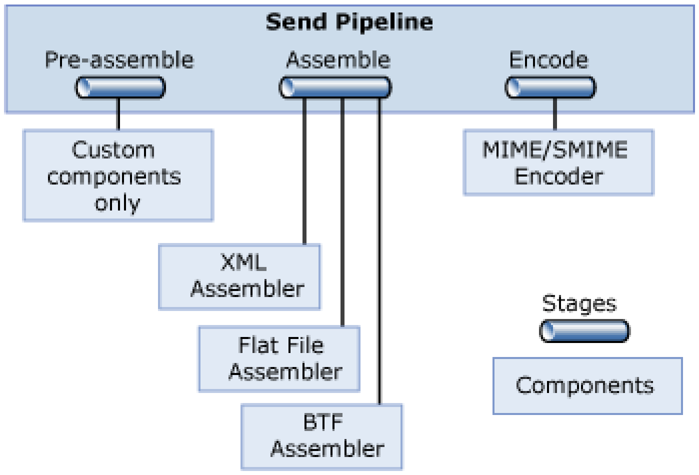
The figure above illustrates the stages in a receive pipeline, along with the standard components provided for each one. Those stages and their associated components are:

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

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

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

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



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

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

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

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

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

Choosing Messages: Subscriptions

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

When a message is processed by a receive pipeline, a message context is created that contains various properties of the message. An orchestration or a send pipeline can subscribe to messages based on the values of these properties. For example, an orchestration might create a subscription that matches all messages of the type "Invoice", or all messages of the type "Invoice" received from the QwickBank corporation, or all messages of the type "Invoice" received from the QwickBank corporation that are for more than $10,000. However it 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 pipeline has established.

- In BizTalk Server 2006, it is also possible to subscribe to specific error conditions. An error message can be processed in a particular way or routed to a specific destination, such as a Windows SharePoint Services folder.

Defining Business Processes

The exchange of messages between different systems is a necessary part of solving the problems that BizTalk Server 2006 addresses. The real goal, however, is to define and execute business processes based on the applications. The BizTalk Server 2006 engine uses orchestrations to define the logic of these business processes.. To create and evaluate groups of business rules, it uses the Business Rule Engine. This section describes both orchestrations and the Business Rule Engine.

Using Orchestrations

The logic of an automated business process can be implemented directly in a language such as Microsoft C# or Visual Basic. Yet creating, maintaining, and managing complex business processes in conventional programming languages can be challenging. Unlike its predecessors, BizTalk Server 2006 takes a different approach. It enables business managers or developers to define a business process graphically. Doing this can be faster than building the process directly in a programming language, and it also makes 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.

Successfully creating an automated business process usually requires collaboration between software developers and business people. Accordingly, BizTalk Server 2006 provides appropriate tools for each. The developer tools run inside Visual Studio 2005, an environment in which software professionals feel at home. For business people, BizTalk Server 2006 also provides a subset of the developer tool functionality in an add-in for Microsoft Visio. After it has been defined, this process, named as an orchestration, is automatically transformed into standard assemblies that run on .NET Framework.

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

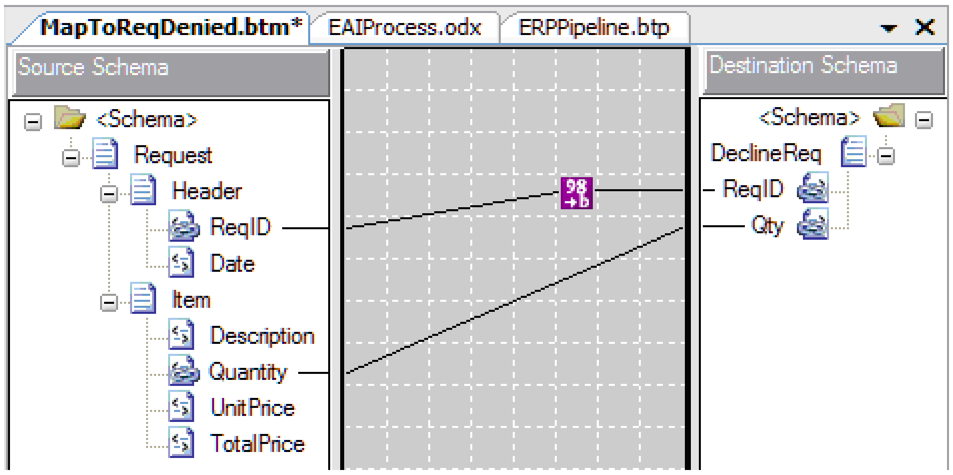
Creating Schemas: The BizTalk Editor

Orchestrations work with XML documents, each of which conforms to some XML schema. The BizTalk Editor. enables developers to define these schemas, which essentially define the structure and types of a document's information, using the XML Schema Definition language (XSD).

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

Mapping Between Schemas: The BizTalk Mapper

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



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

The transformation defined in a map can be simple, such as copying values unchanged from one document to another. Direct data copies like this are expressed using a link, which is shown in the BizTalk Mapper as a line connecting the appropriate elements in the source schema with their counterparts in the destination schema. More complex transformations are also possible using functoids. A functoid is a chunk of executable code that can define arbitrarily complex mappings between XML schemas, and as shown above, the BizTalk Mapper represents it as a box on the line connecting the elements being transformed. Because some of those transformations are fairly common, BizTalk Server 2006 includes a number of built-in functoids. These built-in functoids are grouped into categories, which include the following:

- Mathematical functoids that perform operations such as adding, multiplying, and dividing the values of fields in the source document and storing the result in a field in the target document.
- Conversion functoids that convert a numeric value to its ASCII equivalent and vice-versa.
- Logical functoids that can be used to determine whether an element or attribute should be created in the target document based on a logical comparison between specified values in the source document. Those values can be compared for equality, greater than/less than, and in other ways.
- Cumulative functoids that compute averages, sums, or other values from various fields in the source document, then store the result in a single field in the target document.
- Database functoids that can access information stored in a database.

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

It is essential that you have a way to define a document's XML schema, as well as a mechanism for mapping information across documents with different schemas. The BizTalk Editor and BizTalk Mapper address these two problems. Yet defining schemas and maps is only part of the process. You must also specify the business logic that will use the schemas and invoke the maps.

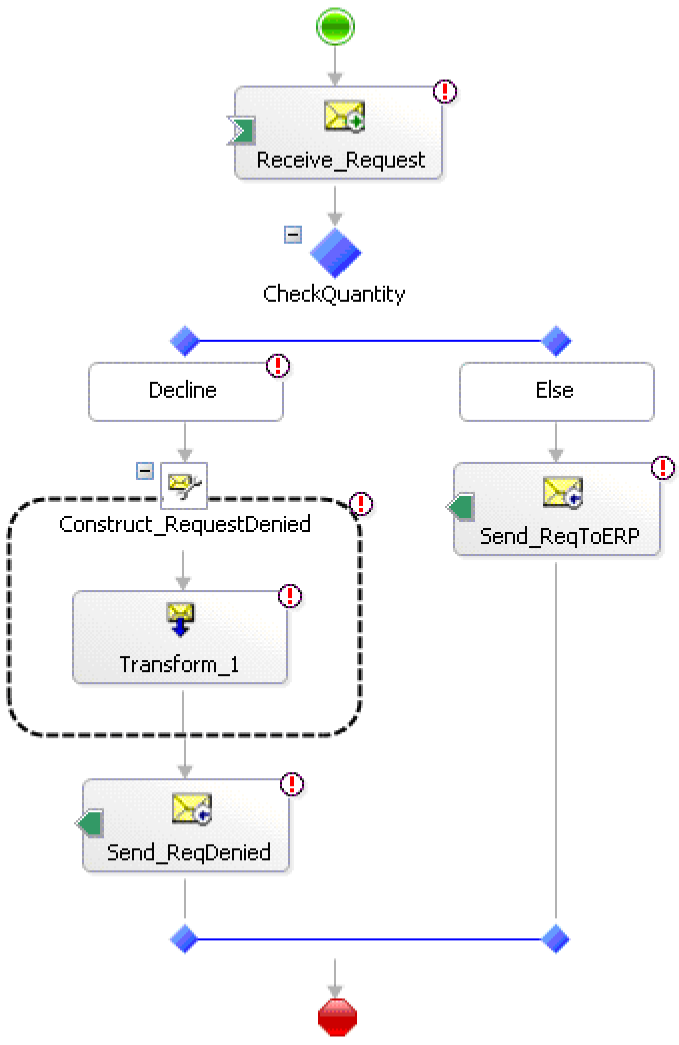
Defining Business Logic: The Orchestration Designer

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

- The Receive shape, which allows the orchestration to receive messages. A Receive shape can have a filter that defines the types of messages that will be received, and it can also be configured to start a new instance of an orchestration when a new message arrives.
- The Send shape, which allows the orchestration to send messages.
- The Port shape, which defines how messages are transmitted. Each instance of a Port shape is connected to either a Send or Receive shape. Each port also has a type, which defines things such as the kinds of messages the port can receive; a direction, such as send or receive; and a

binding, which determines how a message is sent or received by, for example, specifying a particular URL and other information.

- The Decide shape, which represents an if-then-else statement that allows an orchestration to perform different tasks based on Boolean conditions. An Expression Editor, part of the Orchestration Designer, can be used to specify this conditional statement.
- The Loop shape, which controls the repeated performing of an action while some condition is true.
- The Construct Message shape, which allows building a message.
- The Transform shape, which allows the transfer of information from one document to another, and transforming it by invoking maps defined with the BizTalk Mapper.
- The Parallel Actions shape, which allows a developer to specify that multiple operations should be performed in parallel rather than in sequence. The shape that follows this one is not executed until all of the parallel actions have completed.
- The Scope shape, which allows the grouping of operations into transactions and defining exception handlers for error handling. Both traditional atomic transactions and long-running transactions are supported. Unlike atomic transactions, long-running transactions rely on compensating logic rather than rollback to handle unexpected events.
- The Message Assignment shape, which allows assigning values to orchestration variables. These variables can be used to store state information used by the orchestration, such as a message being created or a character string.
- The figure below shows an orchestration created in the Orchestration Designer using a few of these shapes. In this simple example, a message is received, a decision is made based on the content of that message, and one of two paths is executed as a result of that decision. Orchestrations that solve real problems can be significantly more complex than this; to make these more complex diagrams easier to work with, the Orchestration Designer includes the ability to zoom in and out. Develo[ers can view only those parts of an orchestration that they are currently interested in.



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

The Role of Web Services

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

- The rise of Web services also has an impact on how business processes are defined. For example, think about the case where two organizations interact using Web services. To interoperate effectively, it might be necessary for each party to the interaction to know something about the business process the other is using. If both organizations use BizTalk Server 2006, this isn't a big problem; tools such as the Trading Partner Management technology can be used to distribute this knowledge. But what if they use different products? For cases like this, it is useful to have a way to describe aspects of business processes in a non-vendor specific way.

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

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

- Orchestrations are the fundamental mechanism for creating business processes in BizTalk Server. Some aspects of an orchestration tend to change more often than others, however. In particular, the decisions embedded in a business process—the business rules—are commonly its most volatile aspect. A manager's spending limit was $100,000 last week, but her promotion bumps this up to $500,000, or a slow-paying customer's maximum allowed order decreases from 100 units to only 10. You can specify and update these rules using the Business Rule Engine.

Using the Business Rule Engine

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

One situation in which the BRE is useful is when a complex set of business rules must be evaluated. Deciding whether to grant a loan, for example, might entail working through a large set of rules based on the customer's credit history, income, and other factors. Similarly, determining whether to sell life insurance to an applicant depends on a number of things, including the applicant's age, gender, and health. Expressing all of these rules as conditional statements using, say, an orchestration's Decide shape might be possible, but would be fairly complex to implement. For rule-intensive processes like these, the BRE can make a developer's life simpler.

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

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

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

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

Both vocabularies and business rules can be much more complicated—and much more powerful—than the simple examples described here. But the core idea of defining a vocabulary, then defining sets of rules that use that vocabulary is the heart of the Business Rule Engine.

Management and Monitoring

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

Installing BizTalk Server 2006

Microsoft BizTalk Server 2006 includes a number of components, and it depends on multiple aspects of the Windows environment. Making sure that the correct version of everything the product needs is available, then installing all of the product components correctly could potentially be a challenging process.

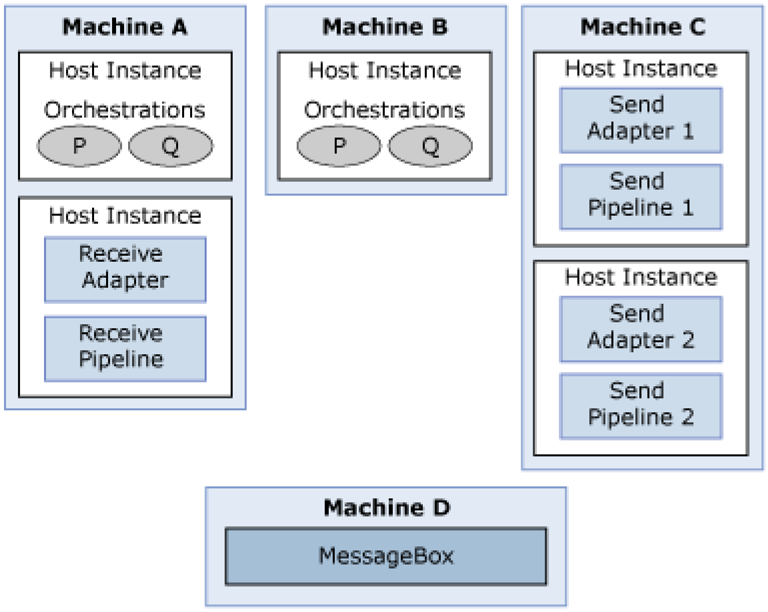
Installation is, however, straightforward. Upgrading from BizTalk Server 2006 is automatic, and all items built for this earlier version—orchestrations, maps, and so on—will work unchanged in the 2006 version. To ensure that the right environment exists, an administrator doing a new installation of BizTalk Server 2006 can download a standard .CAB file or reference an already available .CAB file that was downloaded earlier. In either case, this file contains the redistributable components that the product requires for installation. These include things such as the correct versions of Microsoft Data Access Components (MDAC), Microsoft XML Parser (MSXML), the latest security hot fixes, and other necessary software.

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

Creating Scalable Configurations

If the BizTalk application is not too large, the entire BizTalk Server engine can be installed on a single machine. In some situations, however, this would not be the right solution. Perhaps the number of messages the engine must handle is too great for one machine, or maybe redundancy is required to make the system more reliable. To meet requirements like these, the BizTalk Server 2006 engine can be deployed in a number of ways.

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



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

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

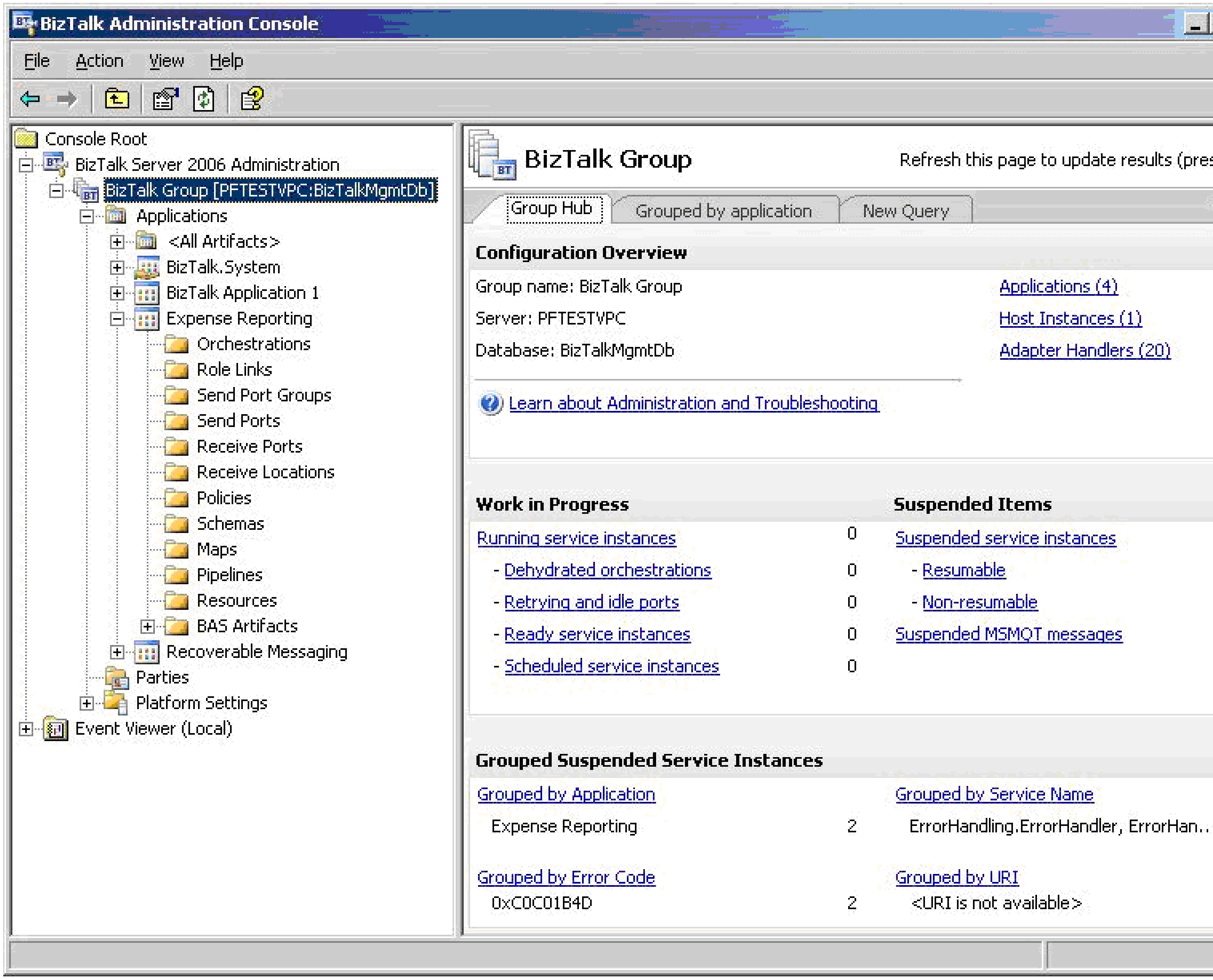
Managing Applications

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

- Deploy BizTalk applications. BizTalk Server enables administrators to work with a complete BizTalk application as a unit. Using the BizTalk Administration console, an administrator can create a BizTalk application, and deploy it to one or more servers.

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

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



Configuration Overview

Group name: BizTalk Group
Server: PFTESTVPC
Database: BizTalkMgmtDb

Applications (4)
Host Instances (1)
Adapter Handlers (20)

Work in Progress		Suspended Items
Running service instances	0	Suspended service instances
- Dehydrated orchestrations	0	- Resumable
- Retrying and idle ports	0	- Non-resumable
- Ready service instances	0	Suspended MSMQT messages
- Scheduled service instances	0	

Grouped Suspended Service Instances

Grouped by Application		Grouped by Service Name
Expense Reporting	2	ErrorHandling.ErrorHandler, ErrorHan..
Grouped by Error Code		Grouped by URI
0xC0C01B4D	2	<URI is not available>

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

Reporting On and Debugging Applications: Health and Activity Tracking

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

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

Enterprise Single Sign-On [BTS]

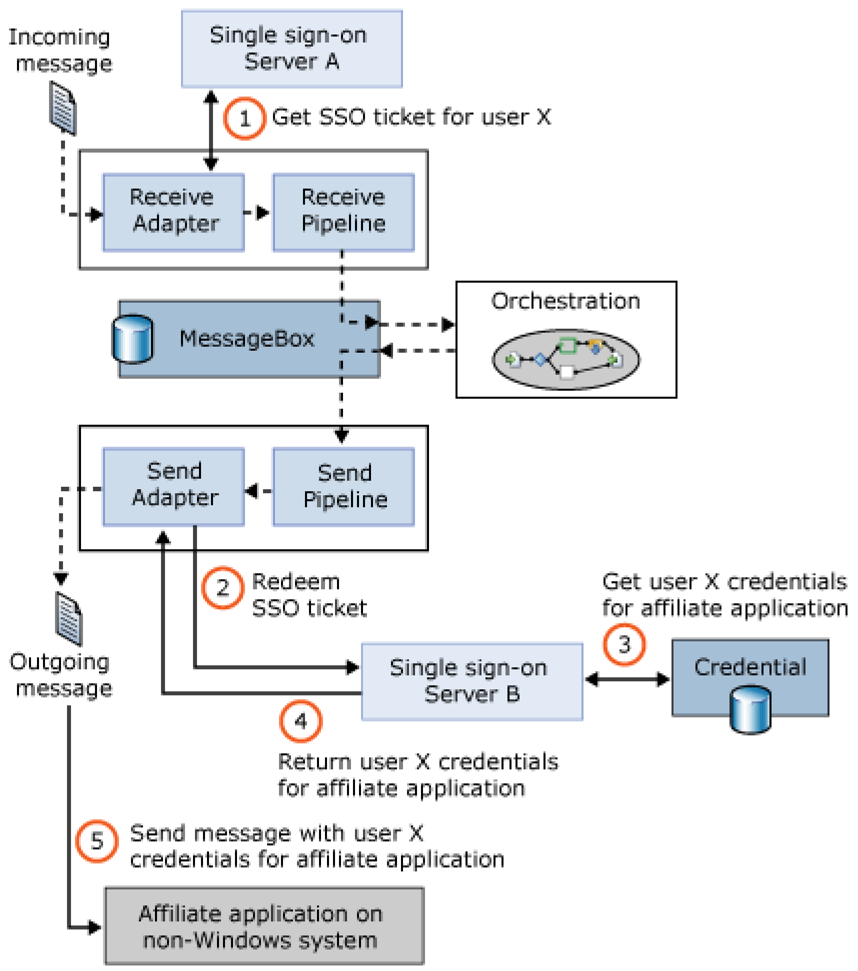
A business process that relies on several different applications may have to cross several different security domains. Accessing an application on a Microsoft Windows system may require one set of security credentials, while accessing an application on an IBM mainframe may require different credentials,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

Don’t be confused—this isn’t a mechanism that lets people have one login for all applications. Instead, Enterprise Single Sign-On provides a way to map a Windows user ID to non-Windows user credentials. It does 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.

Creating Affiliate Application for Non-Windows Systems

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

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


Incoming message
Single sign-on Server A
1 Get SSO ticket for user X
Receive Adapter
Receive Pipeline
MessageBox
Orchestration
Send Adapter
Send Pipeline
Outgoing message
2 Redeem SSO ticket
Get user X credentials for affiliate application
3
Single sign-on Server B
Credential
4 Return user X credentials for affiliate application
5 Send message with user X credentials for affiliate application
Affiliate application on non-Windows system

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

Message Processing With an SSO Ticket

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

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

Administering SSO

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

The example described above shows a typical use of Enterprise Single Sign-On, but SSO can be configured in other ways. A smaller BizTalk Server installation may have only a single SSO server, for example, and it's possible to use Enterprise Single Sign-On independently from the BizTalk Server engine. (The technology also ships with Microsoft Host Integration Server.)

Information Worker Technologies

The Microsoft BizTalk Server 2006 engine provides support for automated business processes that span multiple applications. But once those processes have been created, the information workers that use them—business people, not developers—have other requirements. They might need to monitor various business-related aspects of the process, for example, or work with trading partners in various ways.

BizTalk Server 2006 provides several components that address these problems. Those components are grouped into two categories: Business Activity Monitoring and Business Activity Services. This section describes both.

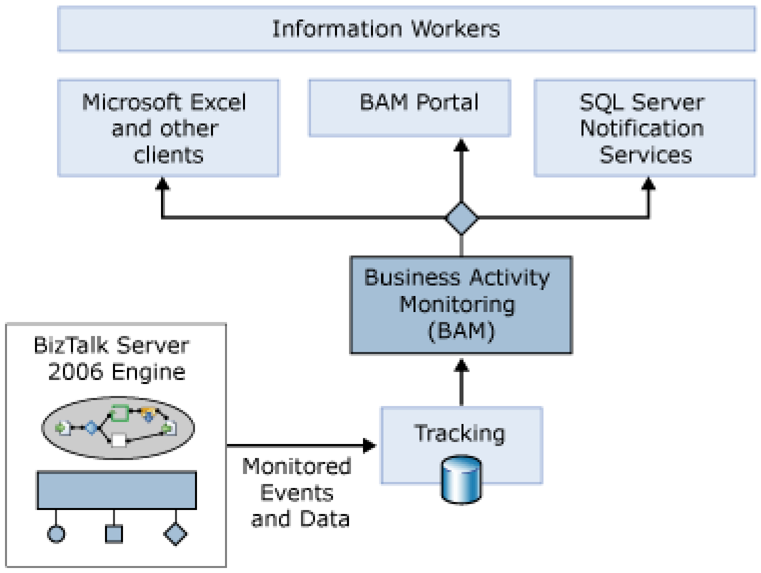
In This Section

- Business Activity Monitoring

- Business Activity Services [BTS]

Business Activity Monitoring

Information workers need flexibility in looking at and evaluating business processes. A purchasing manager might need to see how many POs are approved and denied each day, for example, while a sales manager might want an hourly update on what products are being ordered. Meeting these diverse needs requires a general framework for tracking what's going on with a particular business process. This is exactly what the Business Activity Monitoring (BAM) component in Microsoft BizTalk Server 2006 provides.


Information Workers
Microsoft Excel and other clients
BAM Portal
SQL Server Notification Services
Business Activity Monitoring (BAM)
BizTalk Server 2006 Engine
Tracking
Monitored Events and Data

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

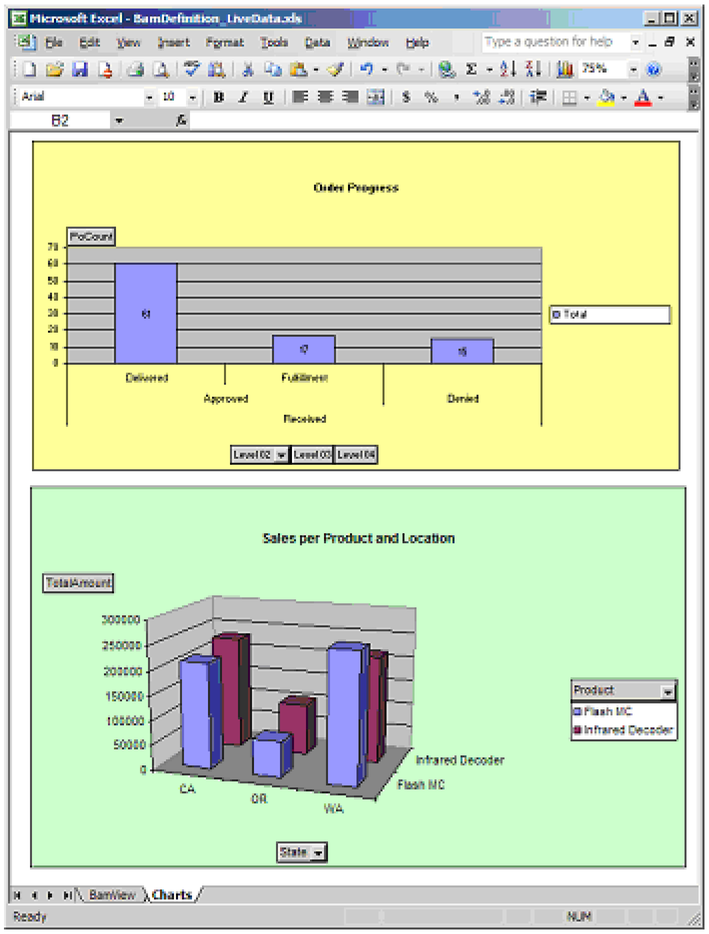
- Through Microsoft Excel or other desktop clients, such as a custom dashboard application.

- Using a BAM portal, a new component in BizTalk Server that enables business users to examine and configure BAM information. Using the BAM portal, an information worker can select a particular instance of some business process, and then choose a specific BAM view into the process. Each of these views provides a different perspective, such as graphical depictions of per-product sales trends or current inventory levels, or other key performance indicators. The information in these views might be updated every day, every hour, or more frequently. Using the BAM portal, an information worker can also define aggregations of data, such as the number of orders filled, canceled, or in progress over the last hour. Implemented as a set of ASP.NET pages, the BAM portal can also be hosted as a Web part inside Windows SharePoint Services.

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

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

For information workers accessing BAM through Excel, BAM activities and BAM views can be created using an Excel add-in. The BAM Activity Wizard for this add-in allows the defining activities, while the BAM View Wizard allows the defining of views based on those activities. In fact, the BAM View Wizard really just helps an information worker build a standard Excel PivotTable using the information in one or more BAM activities. The information that this view provides can then be shown directly in Excel, as the figure below illustrates.



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

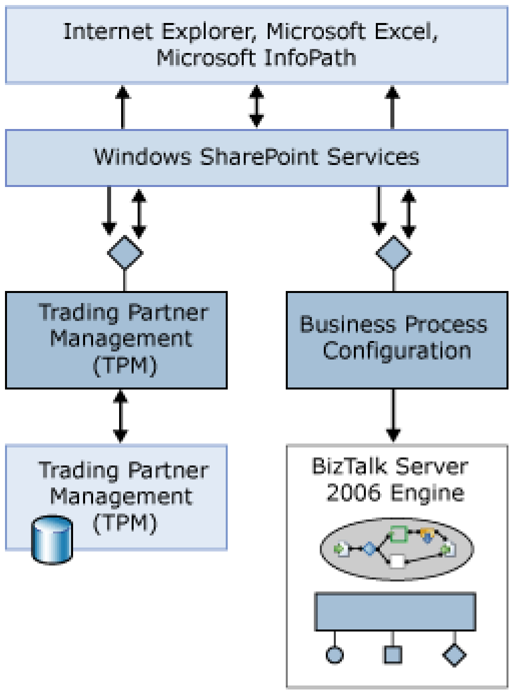
While information workers can create BAM views and BAM activities on their own, these views and activities depend on information provided by the orchestrations they monitor. Accordingly, developers still have a role to play. Using a tool called the Tracking Profile Editor (TPE), a developer must configure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

In BizTalk Server 2006, the TPE can also be used to specify how pipelines generate events. More important, BAM can now accept and display events generated by any user code, whether or not it is built as an orchestration. Any application built using the .NET Framework can potentially be monitored using the BAM component of BizTalk Server 2006.

Business Activity Services [BTS]

As part of running a business process, a business analyst may beed to create a relationship with a new trading partner, for example, that defines the partner's role, the business agreement between the two firms, and other aspects of this new association. Maybe a purchasing manager needs tools that can wrap together and distribute the artifacts required to let a partner quickly implement and begin participating in a business process. In BizTalk Server 2006, these functions are provided by Business Activity Services (BAS).

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


Internet Explorer, Microsoft Excel, Microsoft InfoPath
Windows SharePoint Services
Trading Partner Management (TPM)
Business Process Configuration
Trading Partner Management (TPM)
BizTalk Server 2006 Engine

Trading Partner Management

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

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

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

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

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

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

Business Process Configuration

Although a business analyst may not be able to create the orchestration that implements a business process,it’s not unreasonable to expect an information worker to set parameters for that orchestration. An example of where this would be useful is when an orchestration will be used with multiple partners, but each partner requires slightly different behavior. Suppose, for instance, that the maximum dollar value of a purchase order is different with different trading partners, or perhaps the maximum quantity that can be requested varies depending on the customer’s credit rating. For this type of change, business analysts could make these small configuration changes themselves.

To enable information workers configure an orchestration, the developer creating it can define parameters for that orchestration. Information workers can then set these parameters as needed, perhaps assigning different values for different business partners or different parts of their own organizations. An information worker sets those parameters using the TPM service, described in the previous section, by specifying their values in the addendum to this partner’s agreement. For agreement that reference multiple orchestrations, an addendum can be created for each orchestration.

What's New in BizTalk Server 2006?

This section contains information about the enhancements and new features in BizTalk Server 2006. For detailed pricing and licensing information see http://go.microsoft.com/fwlink/?LinkId=59795.

In This Section

- New Features in BizTalk Server 2006
- Deprecated Features, Tools, and APIs
- Changed Features and Tools
- What's New in Help

Quick Start Guide to Installing and Configuring BizTalk Server 2006

This Quick Start Guide provides scaled down instructions for installing and configuring a complete installation of Microsoft BizTalk Server 2006 (BizTalk Server 2006) on a single computer running Microsoft Windows Server 2003. The information in this Quick Start Guide assumes certain prerequisite knowledge on behalf of the reader. For detailed step-by-step instructions for installing and configuring a custom installation of BizTalk Server 2006 on other platforms see http://go.microsoft.com/fwlink/?LinkId=46922.

Install Prerequisites

The following table lists l the prerequisite software needed for a complete installation of BizTalk Server 2006.

Step/Task	Notes
Step 1: Install Windows Server 2003 with Service Pack 1	• After installing Windows, you must be logged on as an administrator, or add a Windows user account to the Administrators group in order to install additional software.
Step 2: Install Critical Updates for Windows	• Install any high priority updates listed at the Windows Update Web site
Step 3: Install Internet Information Services 6.0	• Enable ASP.NET
Step 4: Install Office Excel 2003 and InfoPath 2003 with Service Pack 2	• Install using default settings

Step 5: Install Visual Studio 2005	• Install Microsoft C# and the Framework SDK • Microsoft SQL Server 2005 Express Edition should not be installed
Step 6: Install SQL Server 2005	• Install using default settings
Step 7: Install and configure Windows SharePoint Services 2.0 with Service Pack 2	• Installation type must be *Server Farm*
Step 8: Extend Default Web Site as a virtual server	• Select NTLM for the security type. Kerberos authentication is not supported with Business Activity Services.
Step 9: Create a network share for Business Activity Monitoring (BAM) alerts	• Create a share named *Alerts* • Give Everyone *Change* permissions to this share

Install and Configure BizTalk Server 2006

In this section you install and configure a complete installation of BizTalk Server 2006 using basic configuration.

The CAB file contains the following software:

- Office Web Components (OWC) 11
- Microsoft Data Access Components (MDAC) 2.8 SP1
- Microsoft XML Core Services (MSXML) 3.0 SP7
- MSXML 6.0
- SQLXML 3.0 SP3
- Microsoft .NET Framework 2.0
- Microsoft SQL Server ADOMD.NET 8.0
- ADOMD.Net-KB893091-v8.00.0991-x86.EXE
- ADOMD.NET 9.0

Step/Task	**Notes**
Step 1: Install BizTalk Server 2006	• The account you are logged on as must be part of the local administrators group and have DBA rights on SQL Server. After

	configuration is complete, you can lower the privileges to DBO • Install all components
Step 2: Configure BizTalk Server 2006	• Install using Basic configuration

Post-Installation Notes

After you install BizTalk Server 2006, you must follow these post-installation steps in the order listed:

- Add the BizTalk Service account to the SharePoint Enabled Hosts Windows group on the server.

BizTalk Server 2006 Tutorials

The BizTalk Server 2006 tutorials contain detailed information about how you can use Microsoft BizTalk Server 2006 to facilitate Enterprise Application Integration (EAI) within your company and automate business-to-business processes among your business partners.

The BizTalk Server tutorials contain simple scenarios to give new users an experience of using a variety of BizTalk tools while creating compiled, testable solutions. For more advanced users, or users who are designing BizTalk solutions, see Business Solutions Scenarios .

For a list of the BizTalk Server tools featured in the tutorials, see Step 5: Using the BizTalk Server Tools.

Use the following tutorials to learn how to use BizTalk Server 2006:

- Tutorial 1: Enterprise Application Integration. Provides step-by-step instructions for implementing a BizTalk solution that receives inventory replacement request messages from a warehouse and evaluates the request messages. If the solution denies a request, it sends a denied message to the warehouse. If the solution approves a request, it forwards the message to an Enterprise Resource Planning (ERP) system.

- Tutorial 2: Purchase Order Process. Provides step-by-step instructions for implementing a BizTalk solution for the ERP system that generates purchase orders from the request messages and sends them to an external partner.

- Tutorial 3: Invoice and Payment Process. Provides step-by-step instructions for extending the solution you create in Tutorial 2 to incorporate a payment process that includes verifying that the partner received the purchase order, receiving the invoice from the partner, constructing a payment voucher for the partner using rules to determine the payment policy, sending the payment and receiving confirmation that the partner received the payment, and forwarding the payment acknowledgment internally.

- Tutorial 4: Trading Partner Management. Provides step-by-step instructions for updating an existing BizTalk solution to incorporate trading partner management. You use the solution to implement a trading partner relationship.

- Tutorial 5: Business Activity Monitoring. Provides step-by-step instructions for defining and viewing data of interest.

The tutorials provide you with an end-to-end learning experience spanning the features of BizTalk Server 2006. Each tutorial contains several lessons. The lessons in each tutorial are designed to allow you to take breaks between them. Each lesson will take you no more than two hours to complete.

Before you begin the BizTalk Server 2006 tutorials, you must complete the steps in Before You Begin the Tutorials. Additionally, an understanding of the fundamental concepts surrounding BizTalk Server 2006, and the tools and processes that are required to start building solutions with BizTalk Server 2006 will help you understand the tasks that you complete in the tutorials.

To learn more about BizTalk Server concepts and architecture, we recommend that you read the following sections of BizTalk Server 2006 Help:

- Getting Started

- Planning and Architecture

- For a complete list of topics about using the BizTalk Server tools featured in the tutorials, see Step 5: Using the BizTalk Server Tools.

In This Section

- Before You Begin the Tutorials

- Tutorial 1: Enterprise Application Integration

- Tutorial 2: Purchase Order Process

- Tutorial 3: Invoice and Payment Process

- Tutorial 4: Trading Partner Management

Before You Begin the Tutorials

The Microsoft BizTalk Server 2006 tutorials are designed to work on a full BizTalk Server 2006 installation on a single computer. A full BizTalk Server 2006 installation includes installing Business Activity Services and Business Activity Monitoring.

This section provides additional information and instructions for configuring your computer for the tutorials. You must complete this section before you begin the BizTalk Server 2006 tutorials.

In This Section

- Step 1: Install BizTalk Server 2006 for the Tutorials
- Step 2: Configure the BizTalk Hosts as Trusted
- Step 3: Install the Tutorial Files
- Step 4: Create a Strong Name Key File
- Step 5: Using the BizTalk Server Tools

Tutorial 1: Enterprise Application Integration

Microsoft BizTalk Server 2006 provides a development and run-time environment for business process management (BPM) and automation. This tutorial presents an end-to-end exercise in setting up and deploying enterprise application integration (EAI) using BizTalk Server 2006.

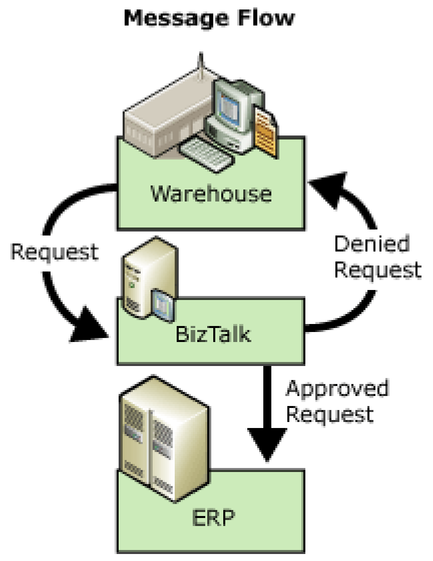
In this solution, you are automating a warehouse order process for an enterprise. When inventory in the warehouse reaches an unacceptably low level, the warehouse sends a message to a BizTalk orchestration requesting inventory replacement. The enterprise uses different computer systems for the warehouse and the orchestration. These systems must exchange data to process the inventory replacement requests. You use BizTalk Server 2006 to automate the business process that begins when the inventory is depleted and ends when the inventory is replaced.

Within the inventory replacement process, several things happen:

- The orchestration evaluates whether a request is approved, using the company's business policies for approval criteria.
- When the orchestration approves a request, the orchestration submits the request to an Enterprise Resource Planning (ERP) system that generates the purchase order.
- When the orchestration does not approve a request, the orchestration returns the request to the warehouse inventory system.
- When the ERP system receives the approved inventory replacement request, the ERP system generates a purchase order (PO).

The warehouse inventory system is a stand-alone system that operates separately from the ERP system. BizTalk Server 2006 ensures that the inventory system and the ERP system interact smoothly and efficiently. This includes applying business rules to the approval process and routing the replenishment request appropriately.

The message flow for this EAI scenario looks similar to the following illustration.



In this tutorial, you use BizTalk Server developer tools to design and deploy the business process. This tutorial provides detailed instructions so that you can complete the lessons even if you have no programming skills.

There are four lessons in this tutorial. Each lesson should take approximately one to two hours to complete. Instructions for saving your work after each step enable you to stop between lessons.

In This Section

- Lesson 1: Create the EAI Solution
- Lesson 2: Define the Business Process
- Lesson 3: Deploy the Solution

Lesson 1: Create the EAI Solution

Time to complete: The lesson will take you approximately two hours to complete.

Objective: In this lesson, you create and build the first project in the enterprise application integration (EAI) solution. The project contains two message schemas, a pipeline, and a map.

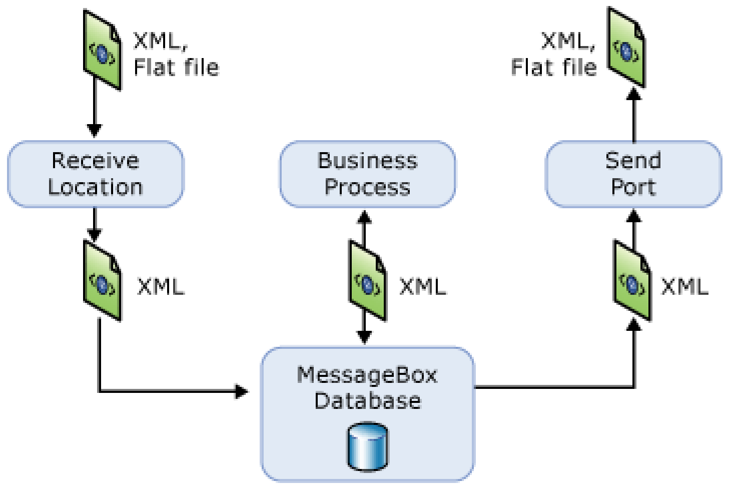
Purpose: In the EAI solution, a warehouse system sends a request for inventory replacement message to BizTalk Server for processing. In this lesson, you create the following items:

- The EAI solution, to hold the project.
- The project, to hold the schemas, pipeline, and map.
- The schema for the message the warehouse sends to BizTalk Server to request inventory replacement.

- The schema for the message BizTalk Server sends to the warehouse if the request for inventory replacement is denied.
- A pipeline, which transforms both messages so BizTalk Server can use the FILE adapter to send and receive the messages.
- A map, which connects the schemas of the two messages. By mapping the schemas, data from the inventory replacement request message can be included in the request denied message.

Finally you build the project before starting Lesson 2. In Lesson 2, you create the business process that routes the messages and evaluates the contents of the inventory replacement request message against approval criteria.

The following illustration shows the flow of data through BizTalk Server in the Tutorial 1 scenario.



The following table provides an overview of the steps you complete in this lesson.

Step	Artifact
Step 1: Open a Blank Solution in Visual Studio 2005	In this step, you create the Empty **EAISolution** solution.
Step 2: Add a New Project to the Solution	In this step, you add the **EAISchemas** project to the solution.
Step 3: Create the Inventory Request Schema	In this step, you add the **Request.xsd** schema to the **EAISchemas** project.
Step 4: Modify the Request Schema Properties	In this step, you modify the properties of the items in the **Request.xsd** schema.
Step 5: Create the	In this step, you add the **RequestDenied.xsd** schema to the

Request Denied Schema	**EAISchemas** project.
Step 6: Create the Distinguished Fields	In this step, you promote the **Quantity** field in the **Request.xsd** schema and the Qty field in the **RequestDenied.xsd** schema by making them distinguished fields.
Step 7: Create a Custom Send Pipeline	In this step, you add the **ERPPipeline.btp** pipeline to the **EAISchemas** project.
Step 8: Create the Map	In this step, you add the **MapToReqDenied.btm** map to the **EAISchemas** project.
Step 9: Build the EAISchemas Project	In this step, you build the **EAISchemas** project.

In This Section

- Step 1: Open a Blank Solution in Visual Studio 2005
- Step 2: Add a New Project to the Solution
- Step 3: Create the Inventory Request Schema
- Step 4: Modify the Request Schema Properties
- Step 5: Create the Request Denied Schema
- Step 6: Create the Distinguished Fields
- Step 7: Create a Custom Send Pipeline
- Step 8: Create the Map
- Step 9: Build the EAISchemas Project

Step 1: Open a Blank Solution in Visual Studio 2005

Purpose: Opening a new BizTalk solution is the first step in building the enterprise application integration (EAI) solution. The BizTalk solution you open is a framework to hold BizTalk artifacts. In subsequent steps, you will add BizTalk artifacts that perform the tasks that automate the EAI process.

Prerequisites

Note the following requirements before you begin this step:

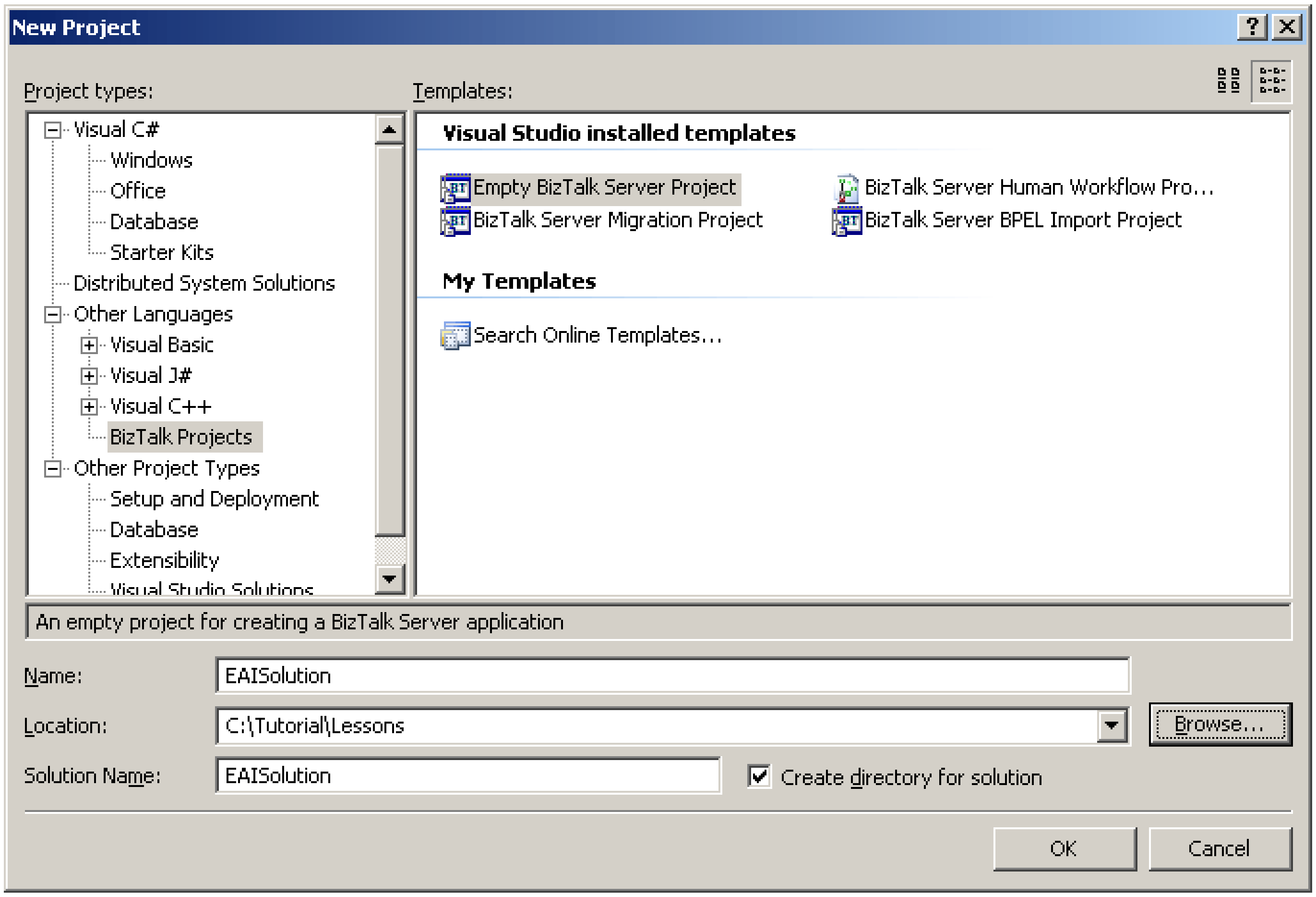
- You must complete the steps in Before You Begin the Tutorials before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To open a blank solution in Visual Studio 2005

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **New**, and then click **Project**.
3. In the **New Project** dialog box, do the following:

Use this	**To do this**
Project types	Click **BizTalk Projects**.
Templates	Click **Empty BizTalk Server Project**.
Name	Type **EAISolution**.
Location	Type **C:\Tutorial\Lessons**.
Solution Name	Visual Studio uses the name you typed as the solution name.
Create directory for solution	Select this check box to create a directory for the solution files.

The following figure shows the **New Project** dialog box.



An empty project for creating a BizTalk Server application

Name: EAISolution

Location: C:\Tutorial\Lessons

Solution Name: EAISolution

Create directory for solution [✓]

4. Click **OK**.

5. In Solution Explorer, right-click the **EAISolution** project and select **Remove**.

6. In the **'EAISolution' will be removed** message box, click **OK**.

7. On the **File** menu, click **Save All**.

Step 2: Add a New Project to the Solution

Purpose: You add BizTalk projects to BizTalk solutions to organize complicated solutions more efficiently. The new project holds the BizTalk artifacts—including schemas, a pipeline, and a map—that you add to the solution in Lesson 1. For example, if you were integrating many diverse processes, you would create a project or set of projects for each process.

Prerequisites

Note the following requirements before you begin this step:

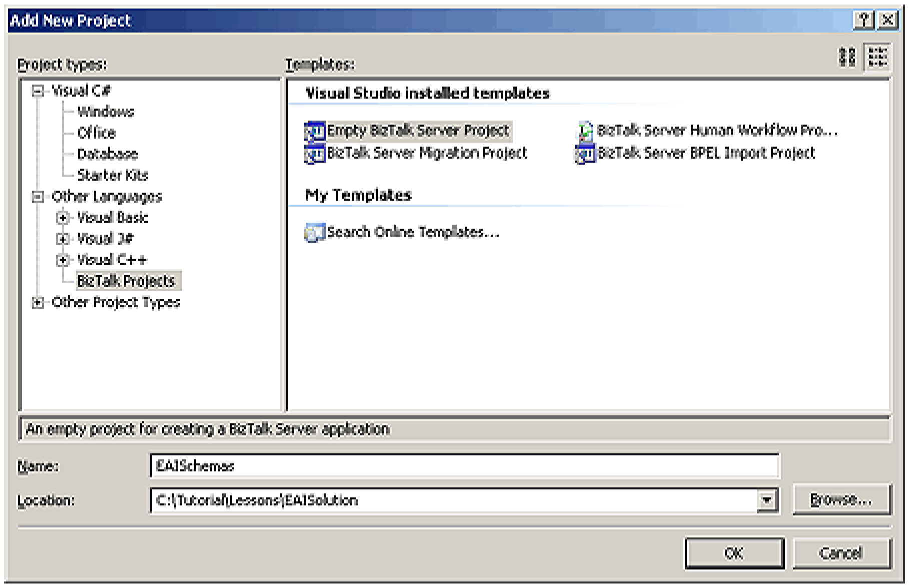
- You must complete Step 1: Open a Blank Solution in Visual Studio 2005 before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a new project to your solution

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click **Solution 'EAISolution'**, point to **Add**, and then click **New Project**.
5. In the **Add New Project** dialog box, do the following:

Use this	To do this
Project types	Click **BizTalk Projects**.
Templates	Click **Empty BizTalk Server Project**.
Name	Type **EAISchemas**.
Location	Verify that the location is **C:\Tutorial\Lessons\EAISolution**.

The following figure shows the **Add New Project** dialog box.



An empty project for creating a BizTalk Server application

Name: EAISchemas

Location: C:\Tutorial\Lessons\EAISolution

6. Click **OK**.

7. On the **File** menu, click **Save All**.

Step 3: Create the Inventory Request Schema

Purpose: The schema defines the data and the structure of the inventory replacement request message. BizTalk Server uses the schema to identify and interact with the data in the message.

Prerequisites

Note the following requirements before you begin this step:

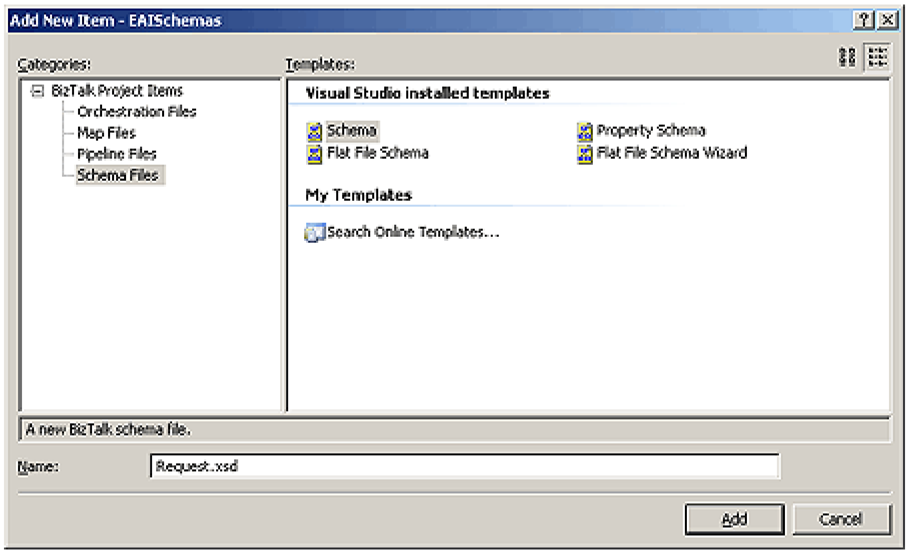
- You must complete Step 2: Add a New Project to the Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a new schema to the project

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click the **EAISchemas** project, point to **Add**, and then click **New Item**.
5. In the **Add New Item - EAISchemas** dialog box, do the following:

Use this	To do this
Categories	Select **Schema Files**.
Templates	Select **Schema**.
Name	Type **Request.xsd**.

The following figure shows the **Add New Item** dialog box.

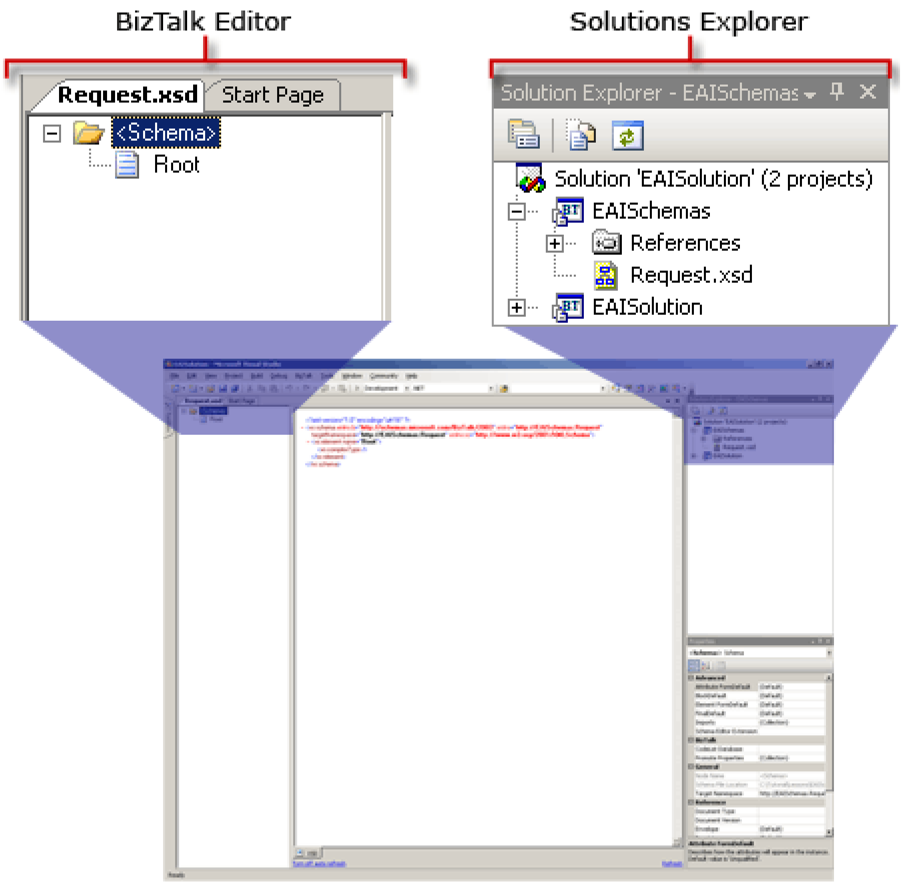


A new BizTalk schema file.

Name: Request.xsd

6. click add

The following figure shows Solution Explorer and the Request.xsd schema.



7. On the **File** menu, click **Save All**.

To add items to the inventory replacement schema

1. In BizTalk Editor, right-click the **Root** node, and then click **Properties**.

2. In the Properties pane, change the value of the **Node Name** property to **Request**, and then press ENTER.

3. In BizTalk Editor, right-click the **Request** node, point to **Insert Schema Node**, and then click **Child Record**.

4. Type **Header** as the new name for the child record, and press ENTER.

5. Create a second child record for the Request node, and name it **Item**.

6. Right-click the **Header** node, point to **Insert Schema Node**, and then click **Child Field Element**.

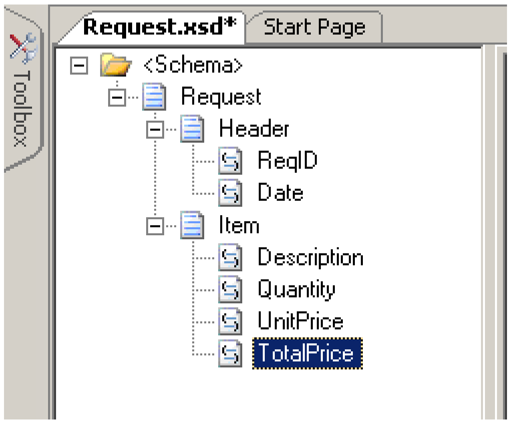
7. Type **ReqID** as the new name for the element, and press ENTER.

8. Create a second child field element for the **Header** node, and name it **Date**.

9. Right-click the **Item** node, and add the following child field elements:

- **Description**
- **Quantity**
- **UnitPrice**
- **TotalPrice**

The completed Request.xsd should look similar to the following figure.



10. On the **File** menu, click **Save All**.

Step 4: Modify the Request Schema Properties

Purpose: You change the Quantity field element to an integer so you can create an expression that evaluates whether the value is greater than or less than 500. You will create the expression in Step 8: Write the XLANG/s Expression for the Decision. You associate the Request schema with flat file extensions because flat file extensions add annotations that maintain data in the schema. For more information about flat file extensions, see Flat File Schemas .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Create the Inventory Request Schema before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To modify the Request schema properties

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

4. In Solution Explorer, double-click **Request.xsd**.

5. In BizTalk Editor, right-click **Quantity**, and then click **Properties**.

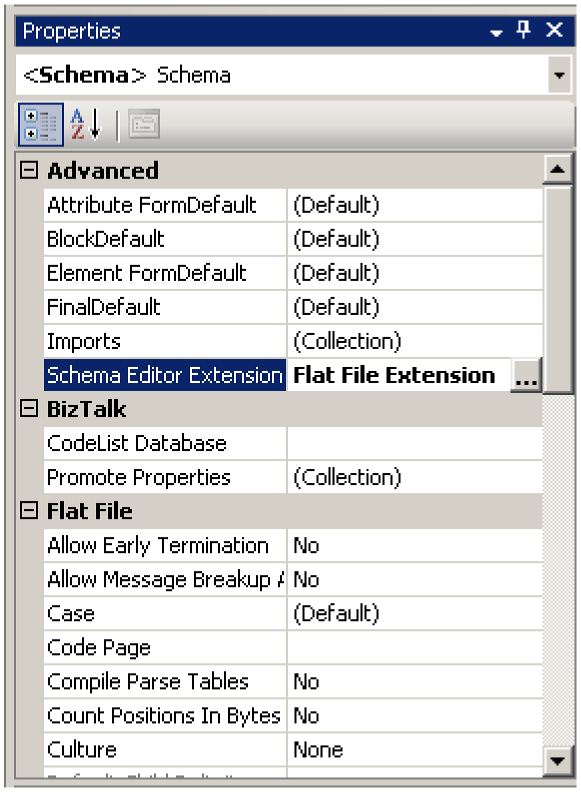
6. In the Properties pane, from the **Data Type** drop-down list, select **xs:unsignedInt**.

7. In BizTalk Editor, click **<Schema>** to highlight the node.

8. In the Properties pane, click the **Schema Editor Extensions** ellipsis [**...**].

9. In the **Schema Editor Extensions** dialog box, select the **Flat File Extensions** check box, and then click **OK**.

The following figure shows the Properties pane for the **<Schema>** node.



Advanced	
Attribute FormDefault	(Default)
BlockDefault	(Default)
Element FormDefault	(Default)
FinalDefault	(Default)
Imports	(Collection)
Schema Editor Extension	**Flat File Extension**
BizTalk	
CodeList Database	
Promote Properties	(Collection)
Flat File	
Allow Early Termination	No
Allow Message Breakup A	No
Case	(Default)
Code Page	
Compile Parse Tables	No
Count Positions In Bytes	No
Culture	None

10. In BizTalk Editor, click **Request** to highlight the node.

11. In the Properties pane, from the **Structure** drop-down list, select **Positional**.

12. In the **Confirm Change Record Structure** dialog box, click **Yes**.

13. For each of the child field elements in the following table, in the BizTalk Editor schema tree, select the child field element. Then, in the Properties pane for each child field element, change the values of the **Positional Length** and **Positional Offset** properties.

Field element	**ReqID**	**Date**	**Description**	**Quantity**	**UnitPrice**	**TotalPrice**
Positional length	10	10	14	3	6	12
Positional offset	0	1	1	1	1	1

14. On the **File** menu, click **Save All**.

Step 5: Create the Request Denied Schema

Purpose: The schema defines the data and the structure of the request denied message. BizTalk Server uses the schema to identify and interact with the data in the message.

Prerequisites

Note the following requirements before you begin this step:

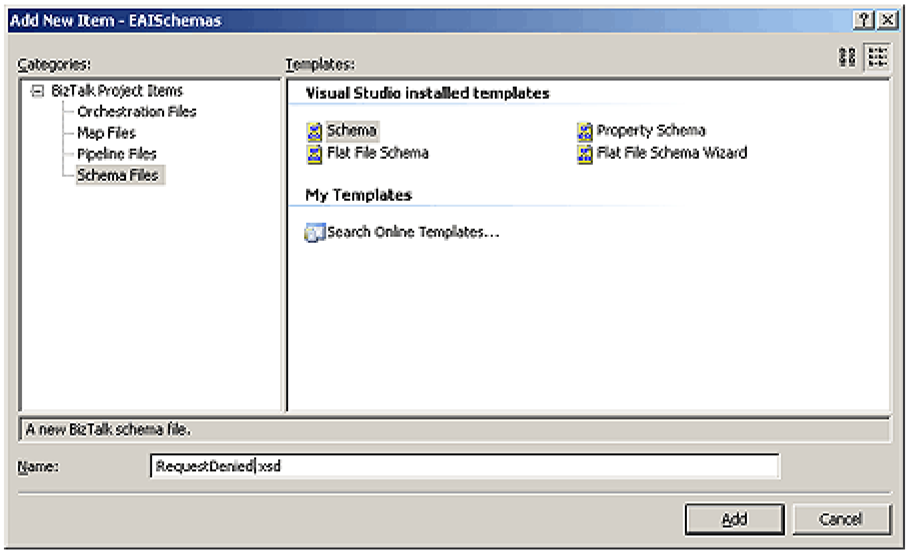
- You must complete Step 4: Modify the Request Schema Properties before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the Request Denied schema

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click the **EAISchemas** project, point to **Add**, and then click **New Item**.
5. In the **Add New Item - EAISchemas** dialog box, do the following:

Use this	To do this
Categories	Select **Schema Files**.
Templates	Select **Schema**.
Name	Type **RequestDenied.xsd**.

The following figure shows the **Add New Item** dialog box.



A new BizTalk schema file.

Name: RequestDenied.xsd

6. click add

7. In BizTalk Editor, right-click the **Root** node, and then click **Properties**.

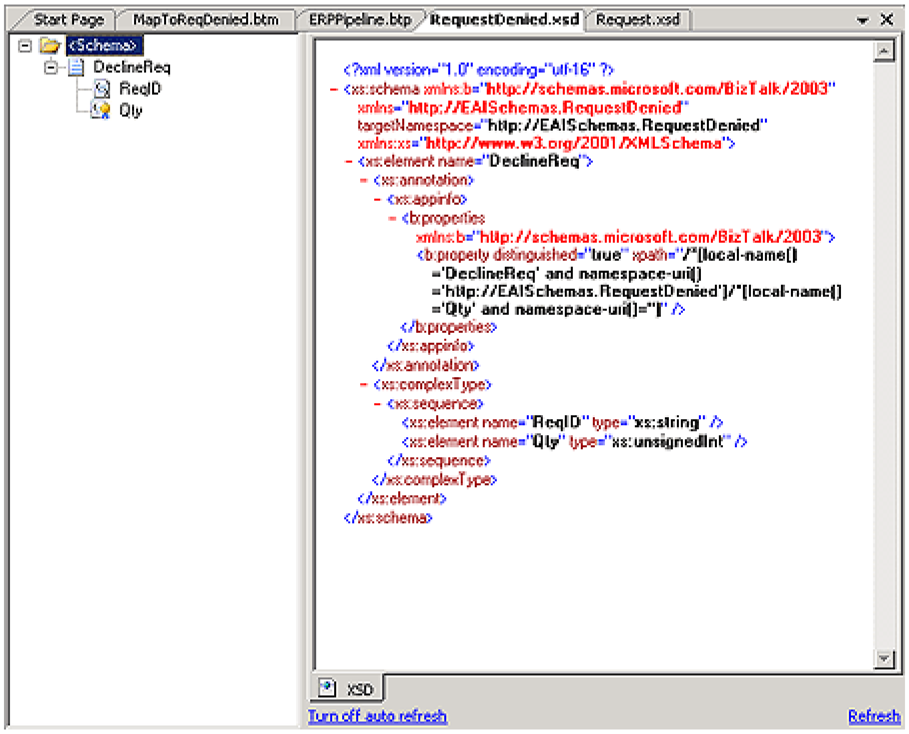
8. In the Properties pane, change the value of the **Node Name** property to **DeclineReq**, and then press ENTER.

9. In BizTalk Editor, right-click the **DeclineReq** node, point to **Insert Schema Node**, and then click **Child Field Element**.

10. Type **ReqID** as the new name for the element, and press ENTER.

11. Add a second child field element named **Qty**.

The following figure shows the RequestDenied schema.



```
  <?xml version="1.0" encoding="utf-16" ?>
- <xs:schema xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
    xmlns="http://EAISchemas.RequestDenied"
    targetNamespace="http://EAISchemas.RequestDenied"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <xs:element name="DeclineReq">
    - <xs:annotation>
      - <xs:appinfo>
        - <b:properties
            xmlns:b="http://schemas.microsoft.com/BizTalk/2003">
            <b:property distinguished="true" xpath="/*[local-name()
              ='DeclineReq' and namespace-uri()
              ='http://EAISchemas.RequestDenied']/*[local-name()
              ='Qty' and namespace-uri()='']" />
          </b:properties>
        </xs:appinfo>
      </xs:annotation>
    - <xs:complexType>
      - <xs:sequence>
          <xs:element name="ReqID" type="xs:string" />
          <xs:element name="Qty" type="xs:unsignedInt" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

12. In BizTalk Editor, right-click **Qty**, and then click **Properties**.

13. In the Properties pane, from the **Data Type** drop-down list, select **xs:unsignedInt**.

14. On the **File** menu, click **Save All**.

Step 6: Create the Distinguished Fields

Purpose: Distinguished fields are message data of special interest that you use primarily to make decisions or to manipulate data in your orchestration. As a distinguished field, the Quantity field is accessible for each instance of a request. The process can evaluate the value of the Quantity field in each request instance because the field is accessible. You promote the Qty field of the request denied instance schema, so that the quantity of the items requested, in addition to the request ID number, are included in the request denied message returned to the warehouse inventory system.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Create the Request Denied Schema before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create a distinguished field

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

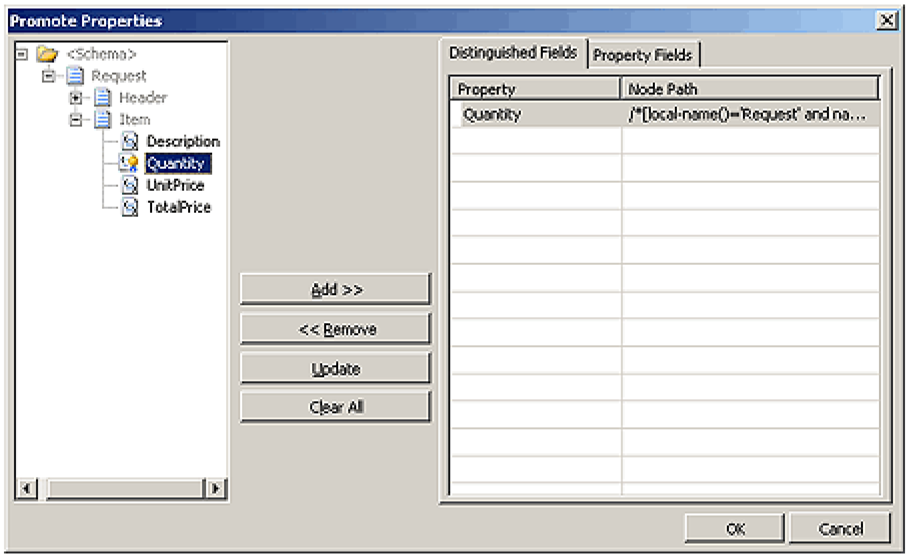
4. In Solution Explorer, double-click the **Request.xsd** schema to open it.

5. In BizTalk Editor, right-click **<Schema>**, and then click **Properties**.

6. In the Properties pane, click the **Promote Properties** ellipsis [**…**].

7. In the **Promote Properties** dialog box, on the **Distinguished Fields** tab, on the left side, expand **Request**, expand **Item**, select **Quantity**, click **Add**, and then click **OK**.

   The following figure shows the **Promote Properties** dialog box for the Request schema.



Property	Node Path
Quantity	/*[local-name()='Request' and na…

8. Open the **RequestDenied.xsd** schema, and add the **Qty** field as a distinguished field.

9. On the **File** menu, click **Save All** to save your work.

Step 7: Create a Custom Send Pipeline

Purpose: You create a custom send pipeline so that you can connect messages with the file adapter. The file adapter transfers files in and out of BizTalk Server.

Prerequisites

Note the following requirements before you begin this step:

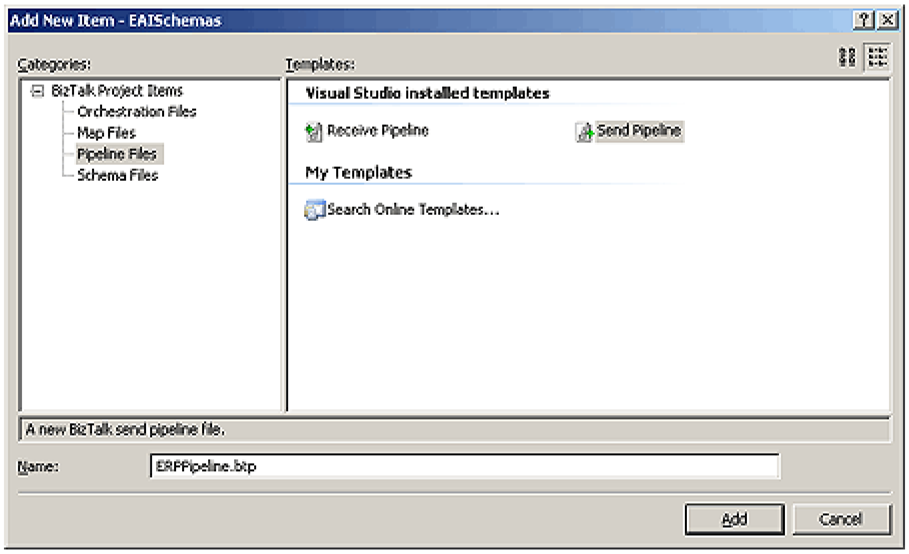
- You must complete Step 6: Create the Distinguished Fields before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a pipeline to your project

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click **EAISchemas**, click **Add**, and then click **New Item**.
5. In the **Add New Item - EAISchemas** dialog box, do the following:

Use this	To do this
Categories	Select **Pipeline Files**.
Templates	Select **Send Pipeline**.
Name	Type **ERPPipeline.btp**.

The following figure shows the **Add New Item** dialog box.



A new BizTalk send pipeline file.

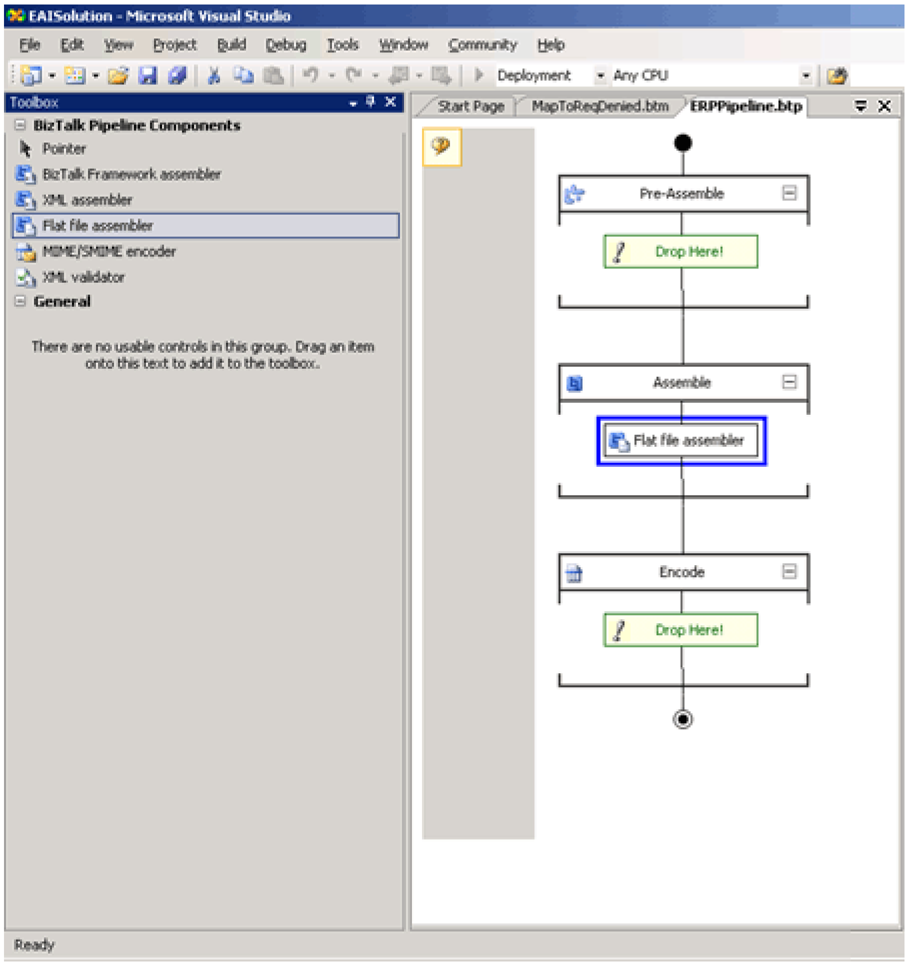
Name: ERPPipeline.btp

6. Click **Add**.

Pipeline Designer opens and the BizTalk Pipeline Components appear in the Toolbox.

7. From the **BizTalk Pipeline Components** Toolbox, drag the **Flat file assembler** from the toolbar to the **Assemble Drop Here!** target.

The following figure shows the ERPPipeline.


EAISolution - Microsoft Visual Studio
File Edit View Project Build Debug Tools Window Community Help
Deployment
Any CPU
Toolbox
BizTalk Pipeline Components
Pointer
BizTalk Framework assembler
XML assembler
Flat file assembler
MIME/SMIME encoder
XML validator
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Start Page
MapToReqDenied.btm
ERPPipeline.btp
Pre-Assemble
Drop Here!
Assemble
Flat file assembler
Encode
Drop Here!
Ready

8. In the **Properties** pane for the Flat file assembler pipeline component, from the **Document Schema** drop-down list, select **EAISchemas.Request**.

9. On the **File** menu, click **Save All** to save your work.

Step 8: Create the Map

Purpose: The map ensures that the request ID number and the quantity of the items requested are included in the request denied message returned to the warehouse inventory system. You use BizTalk Mapper to link fields in an incoming message to fields defined for the outgoing message. This is necessary because these two messages do not have the same schema structure.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 7: Create a Custom Send Pipeline before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

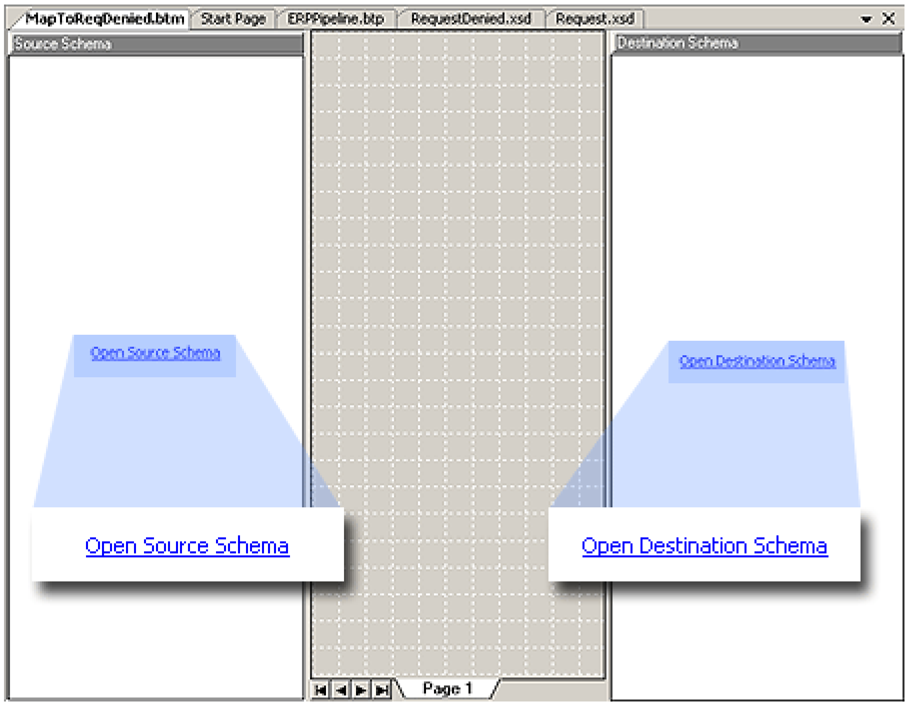
To create the map

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click the **EAISchemas** project, point to **Add**, and then click **New Item**.
5. In the **Add New Item** dialog box, do the following:

Use this	To do this
Categories	Click **Map Files**.
Templates	Click **Map**.
Name	Type **MapToReqDenied.btm**.

6. Click **Add**.

The following figure shows the Source Schema, Destination Schema, and Mapper Grid.



7. In the **Source Schema** pane, click **Open Source Schema**.

8. In the **BizTalk Type Picker** dialog box, expand **EAISchemas**, expand **Schemas**, click **EAISchemas.Request**, and then click **OK**.

9. In the **Source Schema** pane, right-click **<Schema>**, and then click **Expand Tree Node**.

10. In the **Destination Schema** pane, click **Open Destination Schema**.

11. In the **BizTalk Type Picker** dialog box, expand **EAISchemas**, expand **Schemas**, click **EAISchemas.RequestDenied**, and then click **OK**.

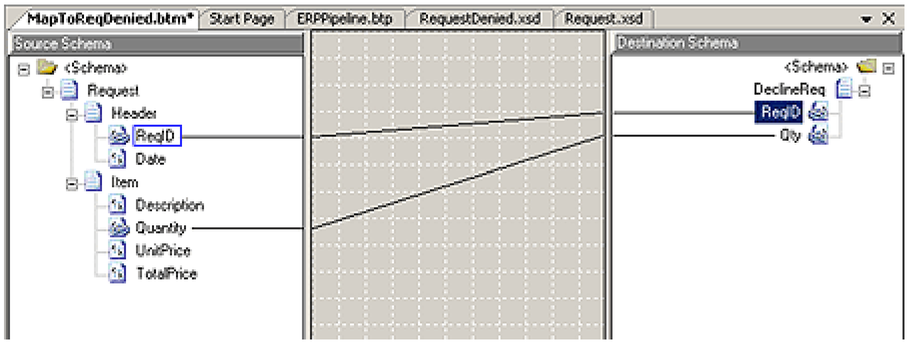
12. In the **Destination Schema** pane, right-click **<Schema>**, and then click **Expand Tree Node**.

13. In the **Source Schema** pane, drag the **Quantity** field to the **Qty** field in the **Destination Schema** pane to map the data from one schema to the other.

14. In the **Source Schema** pane, drag the **ReqID** field to the **ReqID** in the **Destination Schema** pane.

    A line appears connecting the two elements.

    The following figure shows the mapped fields.



15. On the **File** menu, click **Save All** to save your work.

Step 9: Build the EAISchemas Project

Purpose: After you create the EAISchema project and add the necessary items to it, you must build the project to generate an assembly. An assembly is a collection of resources in the project, such as schemas and maps, which are stored in a DLL file.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 8: Create the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To build the EAISchema project

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click the **EAISchemas** project and click **Properties**.
5. In the **EAISchemas Property Pages** dialog box, in the tab list, expand **Common Properties**.
6. On the **Assembly** tab, click the **Assembly Key File** ellipsis [**...**].
7. In the **Assembly Key File** dialog box, do the following:

Use this	To do this
Look in	Navigate to **C:\Tutorial**.
File name	Type **Tutorial.snk**.

8. Click **Open**.

9. In the **EAISchemas Property Pages** dialog box, in the tab list, expand **Configuration Properties**.

10. On the **Deployment** tab, do the following:

Use this	To do this
Application Name	Type **EAIApplication**.
Redeploy	From the drop-down list, select **True**.

11. Click **OK**.

12. On the **File** menu, click **Save All** to save your work.

13. In Solution Explorer, right-click **EAISchemas**, and then click **Build**.

The Output pane at the bottom of the screen should display: **Build: 1 succeeded, 0 failed, 0 skipped.**

Lesson 2: Define the Business Process

Purpose: In the EAI solution, a warehouse system sends a request for inventory replacement message to BizTalk Server for processing. BizTalk Server uses the orchestration you create to automate the business process. The orchestration provides the workflow, actions, and expressions to move messages through the system and process their contents.

Finally you build the project before starting Lesson 3. In Lesson 3, you deploy the solution.

The following table provides an overview of the steps you complete in this lesson.

Step	Description
Step 1: Add a Second Project to your Solution	In this step, you create the EAIOrchestrations project to hold the orchestration and add the EAIProcess orchestration
Step 2: Create the EAI Business	In this step, you add shapes to the EAIProcess orchestration

Process	that define the workflow for the business process.
Step 3: Add a Reference to the EAISchemas Project	In this step, you add a reference from the EAIOrchestrations project to the EAISchemas project.
Step 4: Create Message Instance Variables	In this step, you create the message instance variables.
Step 5: Add Ports to the Orchestration	In this step, you create and configure the ports.
Step 6: Specify Action Shape Messages and Connect the Ports	In this step, you specify action shapes and connect the ports.
Step 7: Configure the Transform Shape to Use the Map	In this step, you configure the transform shape to use the map.
Step 8: Write the XLANG/s Expression for the Decision	In this step, you write the expression that evaluates the inventory replacement message.
Step 9: Build the EAIOrchestrations Project	In this step, you build the EAIOrchestrations project.

In This Section

- Step 1: Add a Second Project to your Solution
- Step 2: Create the EAI Business Process
- Step 3: Add a Reference to the EAISchemas Project
- Step 4: Create Message Instance Variables
- Step 5: Add Ports to the Orchestration
- Step 6: Specify Action Shape Messages and Connect the Ports
- Step 7: Configure the Transform Shape to Use the Map
- Step 8: Write the XLANG/s Expression for the Decision
- Step 9: Build the EAIOrchestrations Project

Step 1: Add a Second Project to your Solution

Purpose: You create a separate project for the orchestration. This is helpful when you have several different people working on one solution. You use the new orchestration to automate the business process in this lesson.

Prerequisites

Note the following requirements before you begin this step:

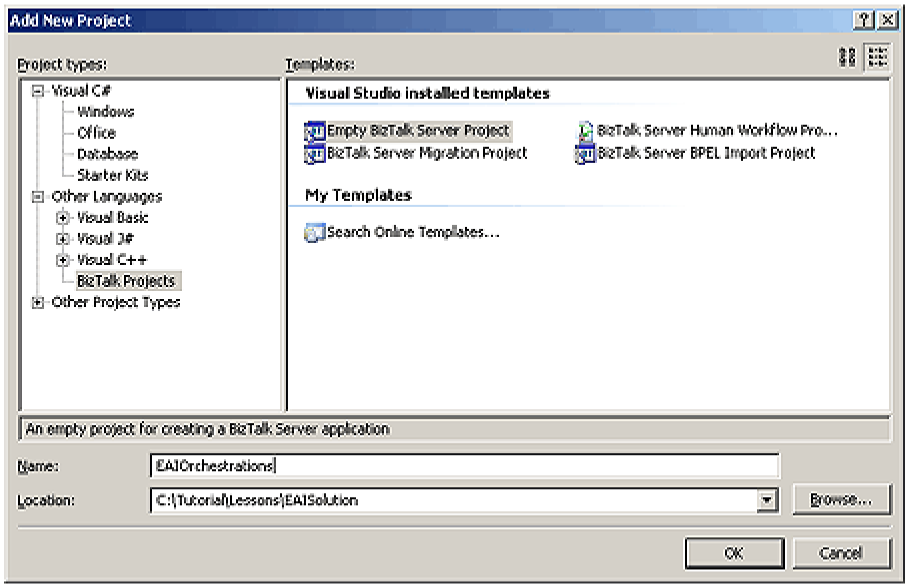
- You must complete Lesson 1: Create the EAI Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add another project to your solution

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click **Solution 'EAISolution'**, point to **Add**, and then click **New Project**.
5. In the **Add New Project** dialog box, do the following:

Use this	To do this
Project types	Click **BizTalk Projects**.
Templates	Click **Empty BizTalk Server Project**.
Name	Type **EAIOrchestrations**.

6. The following figure shows the **Add New Project** dialog box.


Add New Project

Project types:
- Visual C#
  - Windows
  - Office
  - Database
  - Starter Kits
- Other Languages
  - Visual Basic
  - Visual J#
  - Visual C++
  - BizTalk Projects
- Other Project Types

Templates:

Visual Studio installed templates
- Empty BizTalk Server Project
- BizTalk Server Human Workflow Pro...
- BizTalk Server Migration Project
- BizTalk Server BPEL Import Project

My Templates
- Search Online Templates...

An empty project for creating a BizTalk Server application

Name: EAIOrchestrations

Location: C:\Tutorial\Lessons\EAISolution

7. Click **OK**.

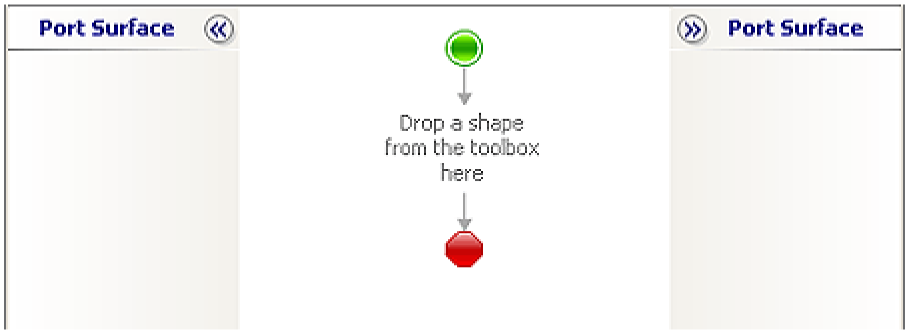
8. In Solution Explorer, right-click **EAIOrchestrations**, point to **Add**, and then click **New Item**.

9. In the **Add New Item** dialog box, do the following:

Use this	To do this
Categories	Click **Orchestration Files**.
Templates	Click **BizTalk Orchestration**.
Name	Type **EAIProcess.odx**.

10. Click **Add**.

Orchestration Designer opens. The following figure shows Orchestration Designer with the EAIProcess orchestration.



11. On the **File** menu, click **Save All** to save your work.

Step 2: Create the EAI Business Process

Purpose: The workflow of the EAIProcess orchestration represents and automates your company's business process for approving inventory replacement requests.

In Orchestration Designer, you define your business process by adding action shapes to the workflow. Some shapes require input. For example, in this lesson, some of the shapes require an expression. When you add a shape that requires input to your orchestration, the shape appears marked with a Smart Tag. You will add the missing expressions in later steps in this lesson.

The following figure shows the Smart Tag.

Prerequisites

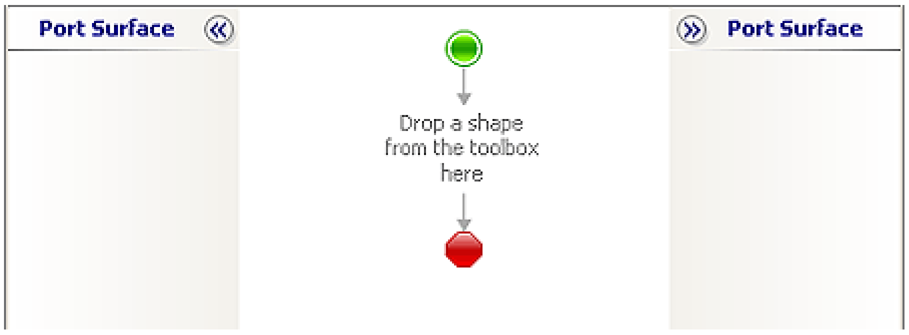
Note the following requirements before you begin this step:

- You must complete Step 1: Add a Second Project to your Solution before you begin this step.
- You must login as a member of the BizTalk Server Administrators group.

To create the EAI business process workflow

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Orchestration Designer, from the orchestration Toolbox, drag the **Receive** shape to the space indicated between the **Begin** (green circle) and **End** (red octagon) shapes.

   The following figure shows the **Begin** (green circle) and **End** (red octagon) shapes.

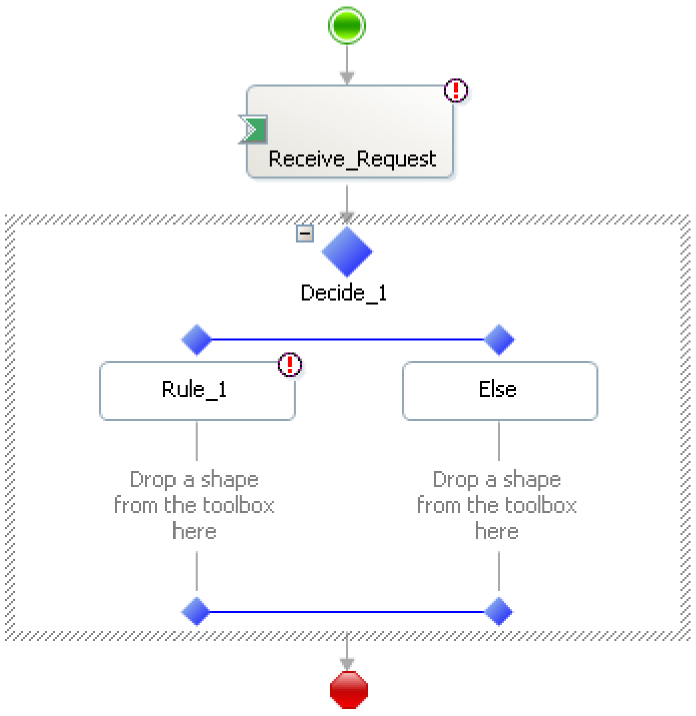


5. In the Properties pane for the **Receive** shape, do the following:

Use this	To do this
Name	Type **Receive_Request**.
Activate	From the drop-down list, select **True**.

6. From the Toolbox, drag the **Decide** shape onto the connecting line directly below the **Receive_Request** shape.

The **Decide** shape expands to show a branch for the If statement (**Rule_1**) and a branch for the **Else** statement. The following figure shows the **Decide** shape.

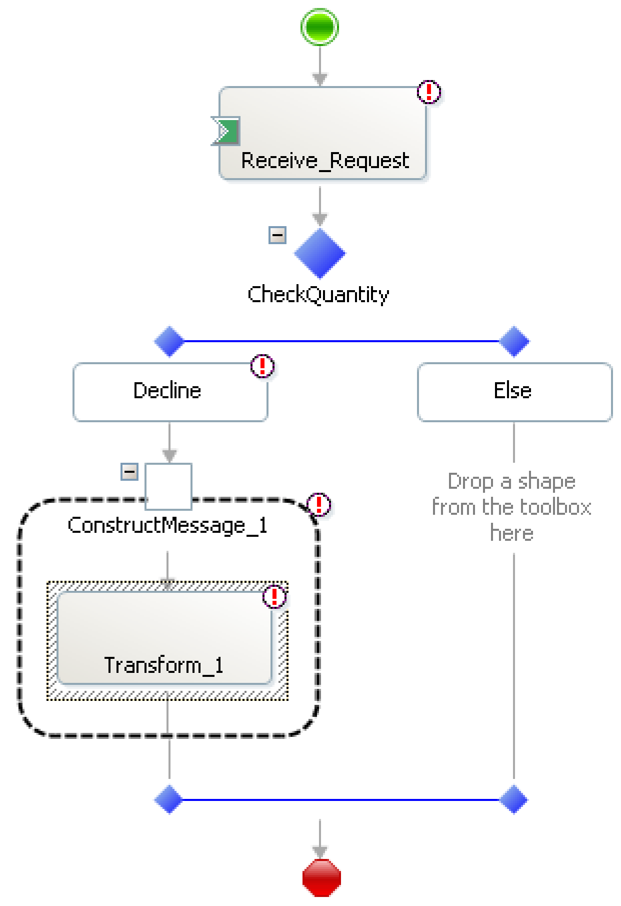

Receive_Request
Decide_1
Rule_1
Else
Drop a shape from the toolbox here
Drop a shape from the toolbox here

7. In the Properties pane for the **Decide** shape, in the **Name** property, type **CheckQuantity**.

8. In the Properties pane for **Rule_1** (inside of the **Decide** shape), in the **Name** property, type **Decline**.

9. From the Toolbox, drag the **Transform** shape to the space indicated, directly below the **Decline** shape.

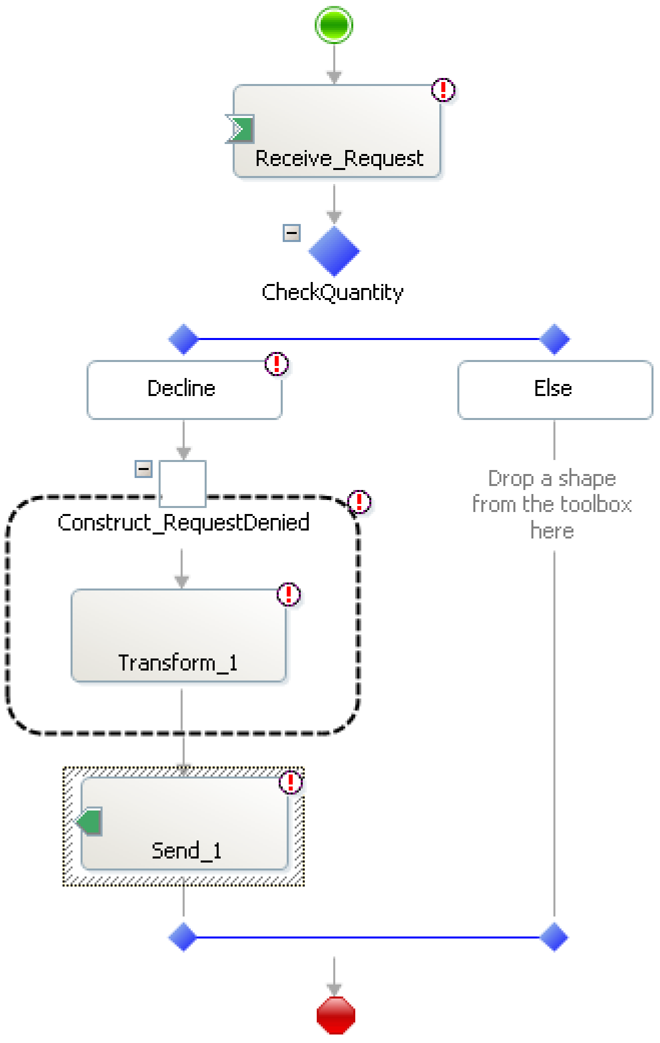
The following figure shows the **Transform** shape.



10. In the Properties pane for the **ConstructMessage_1** shape, in the **Name** property, type **Construct_RequestDenied**.

11. From the Toolbox, drag the **Send** shape to the connecting line directly below the **Construct_RequestDenied** shape, inside of the **CheckQuantity** shape.

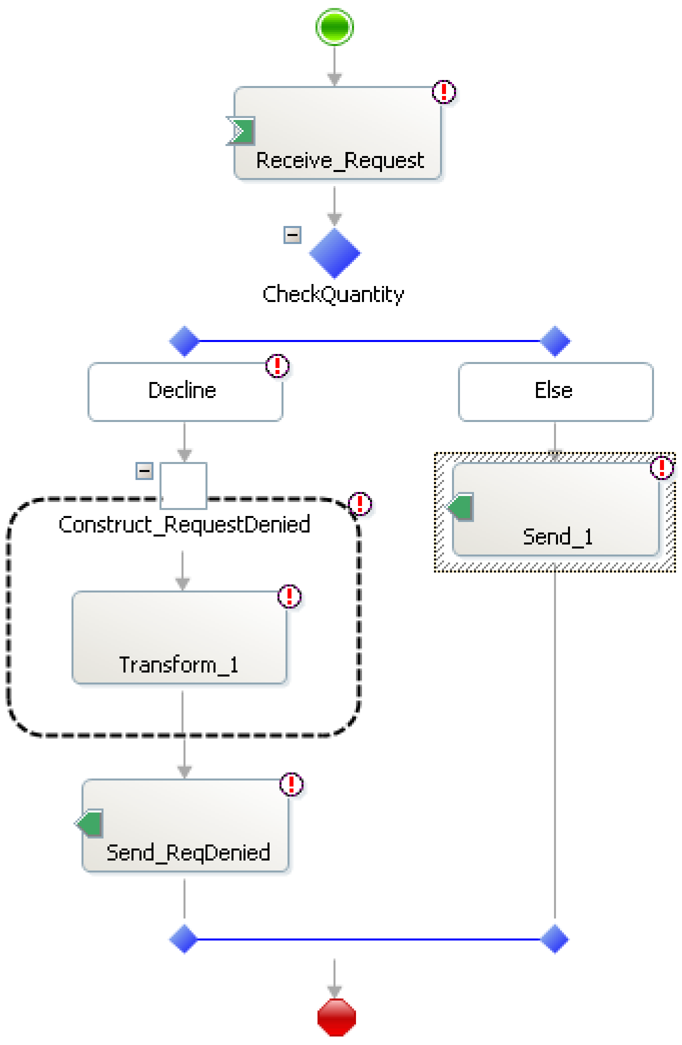
The following figure shows the **Send** shape.



12. In the Properties pane for the **Send** shape, in the **Name** property, type **Send_ReqDenied**.

13. From the Toolbox, drag a second **Send** shape to the space indicated, directly below the **Else** shape, inside of the **CheckQuantity** shape.

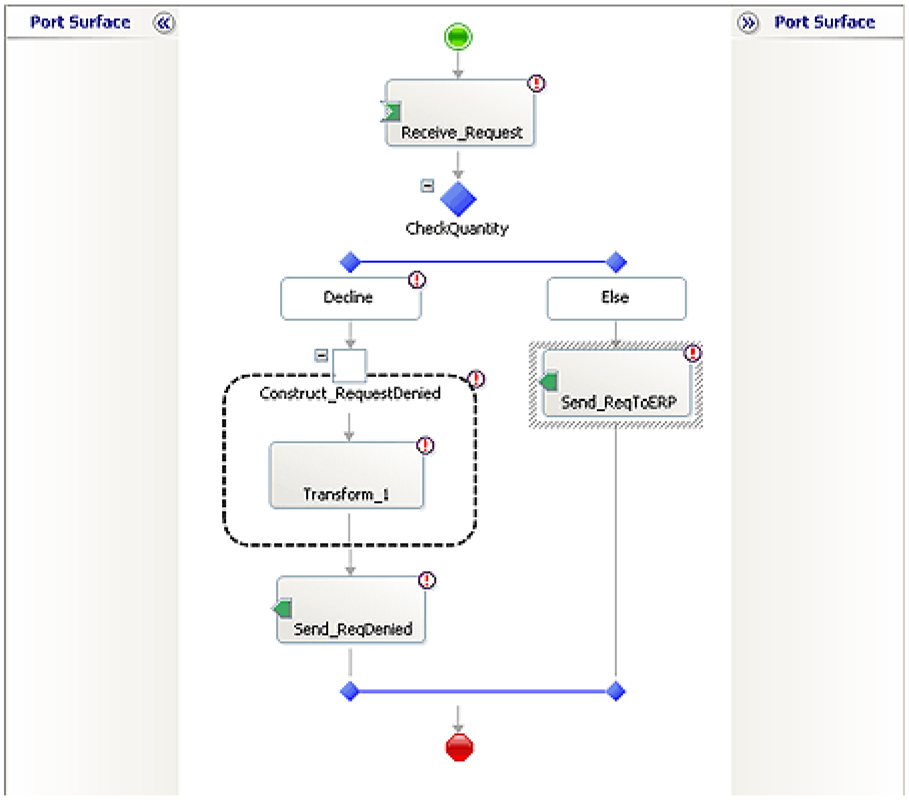
The following figure shows the second **Send** shape.



14. In the Properties pane for the second **Send** shape, in the **Name** property, type **Send_ReqToERP**.

15. On the **File** menu, click **Save All**.

Your Orchestration Design surface should appear similar to the following illustration.



Step 3: Add a Reference to the EAISchemas Project

Purpose: In Orchestration Designer, you define the messages that are being sent and received. The messages use the schemas that you created earlier in the EAISchemas project and must be available to your orchestration. To make the schemas available to the orchestration, you need to add a reference to the EAISchemas assembly.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Create the EAI Business Process before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

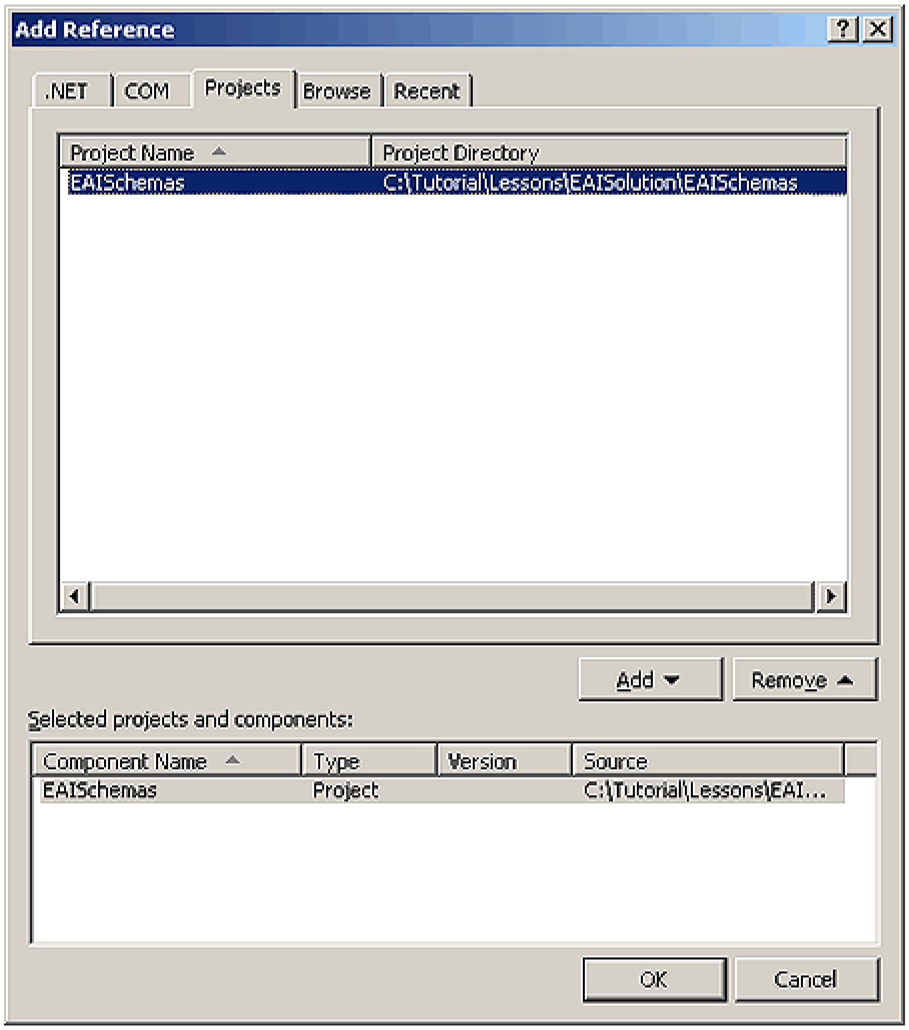
⊟To add a reference to the EAISchemas project

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

4. In Solution Explorer, expand the **EAIOrchestrations** project, right-click the **References** node, and then click **Add Reference**.

5. In the **Add Reference** dialog box, on the **Projects** tab, double-click the **EAISchemas** project to add it as a selected component.

   The **EAISchemas** project appears in the **Selected projects and components** pane.



Project Name	Project Directory
EAISchemas	C:\Tutorial\Lessons\EAISolution\EAISchemas

Selected projects and components:

Component Name	Type	Version	Source
EAISchemas	Project		C:\Tutorial\Lessons\EAI...

6. Click **OK**.

7. Select the **EAISchemas** reference, and in the Properties pane, from the **Copy Local** drop-down list, select **False**.

8. On the **File** menu, click **Save All**.

Step 4: Create Message Instance Variables

Purpose: In Orchestration Designer, a message variable identifies an instance of a message that uses a specific schema structure, rather than dealing directly with the schemas. In the following procedure you will create message variables for each message instance, and associate each variable with a schema.

Prerequisites

Note the following requirements before you begin this step:

You must complete Step 3: Add a Reference to the EAISchemas Project before you begin this step.

You must log on as a member of the BizTalk Server Administrators group.

To create message instance variables

Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

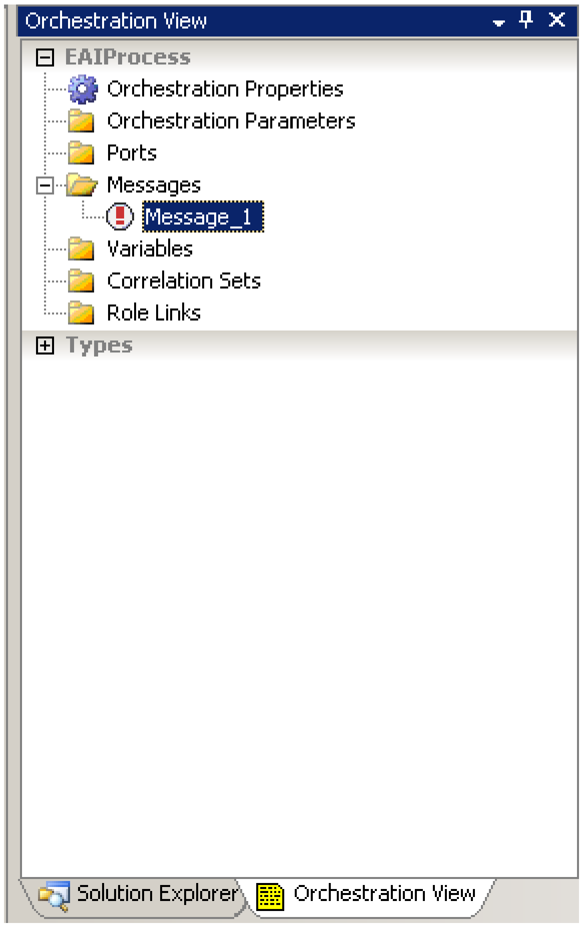
In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

In Solution Explorer, double-click **EAIProcess.odx**.

In **Orchestration View**, right-click **Messages**, and then click **New Message**.

The following figure shows the Orchestration View with the new message.



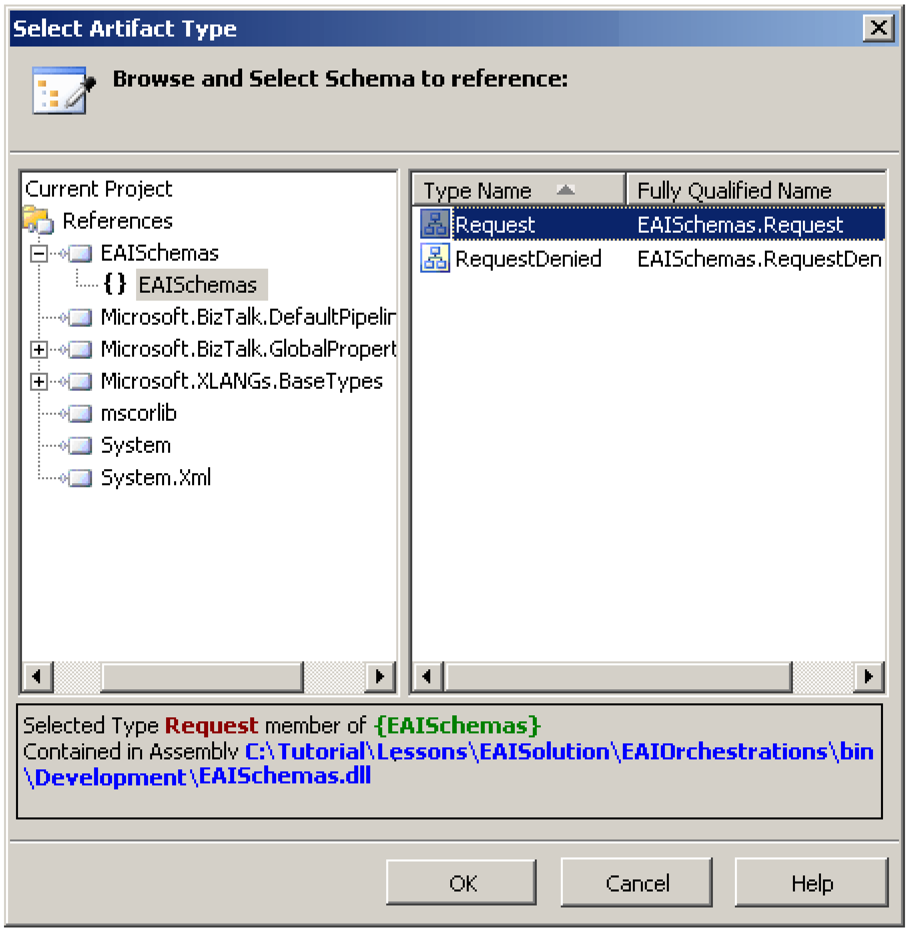
In the Properties pane for **Message_1**, do the following:

Use this	To do this
Identifier	Type **RequestInstance**, and then press ENTER.
Message Type	From the drop-down list, expand **Schemas**, and then select **Select from referenced assembly**.

In the **Select Artifact Type** dialog box, do the following:

Use this	To do this
Current Project	Expand **References**, and then select **EAISchemas**.
Type Name	Select **Request**.

The following figure shows the Select Artifact Type dialog box.


Select Artifact Type
Browse and Select Schema to reference:
Current Project
References
EAISchemas
{ } EAISchemas
Microsoft.BizTalk.DefaultPipelir
Microsoft.BizTalk.GlobalPropert
Microsoft.XLANGs.BaseTypes
mscorlib
System
System.Xml
OK
Cancel
Help

Type Name	Fully Qualified Name
Request	EAISchemas.Request
RequestDenied	EAISchemas.RequestDen

Selected Type **Request** member of **{EAISchemas}**
Contained in Assembly **C:\Tutorial\Lessons\EAISolution\EAIOrchestrations\bin\Development\EAISchemas.dll**

Click **OK**.

Add a second message named **RequestDeniedInstance** with a reference to the **RequestDenied** item in **EAISchemas**.

On the **File** menu, click **Save All**.

Step 5: Add Ports to the Orchestration

Purpose: The three ports you create and configure in this step fulfill the following roles:

- The first port, **ReceiveReqPort**, receives inventory replacement request messages from the warehouse.
- The second port, **SendToERP**, sends request accepted messages to a purchase order process.
- The third port, **SendDeclinePort**, sends request denied messages back to the warehouse.

In this procedure, for each port, you use the Port Configuration Wizard to:

1. Create and configure the port type. A port type is a property that defines the communication pattern (one-way or request-response) and the access restrictions (private, internal, or public) for the port.

2. Establish the direction of communication for the port (sending or receiving).

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Create Message Instance Variables before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To create and configure the ports

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

4. In Solution Explorer, double-click **EAIProcess.odx**.

5. In Orchestration Designer, from the orchestration Toolbox, drag the **Port** shape to the left-side **Port Surface**, parallel to the **Receive_Request** shape.

6. On the **Welcome to the Port Configuration Wizard** page, click **Next**.

7. On the **Port Properties** page, in the **Name** box, type **ReceiveReqPort**, and then click **Next**.

8. On the **Select a Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name:	Type **ReceiveReqType**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Internal - limited to this project** option.

9. Click **Next**.

10. On the **Port Binding** page, do the following:

Use this	To do this
Port direction of communication	From the drop-down list, select **I'll always be receiving messages on this port**.
Port binding	From the drop-down list, select **Specify later**.

11. Click **Next**.

The following table shows the summary information for ReceiveReqPort displayed on the **Completing the Port Wizard** page:

Property	Value
Port Name	ReceiveReqPort
Communication	This port will be used to receive messages only.The binding for this port will be specified through BizTalk Explorer or by script.

12. On the **Completing the Port Wizard** page, click **Finish**.

13. From the orchestration Toolbox, drag the **Port** shape to the right-side **Port Surface,** parallel to the **Send_ReqToERP** shape.

14. Use the information in the following table to complete the Port Configuration Wizard for the **SendToERP** send port.

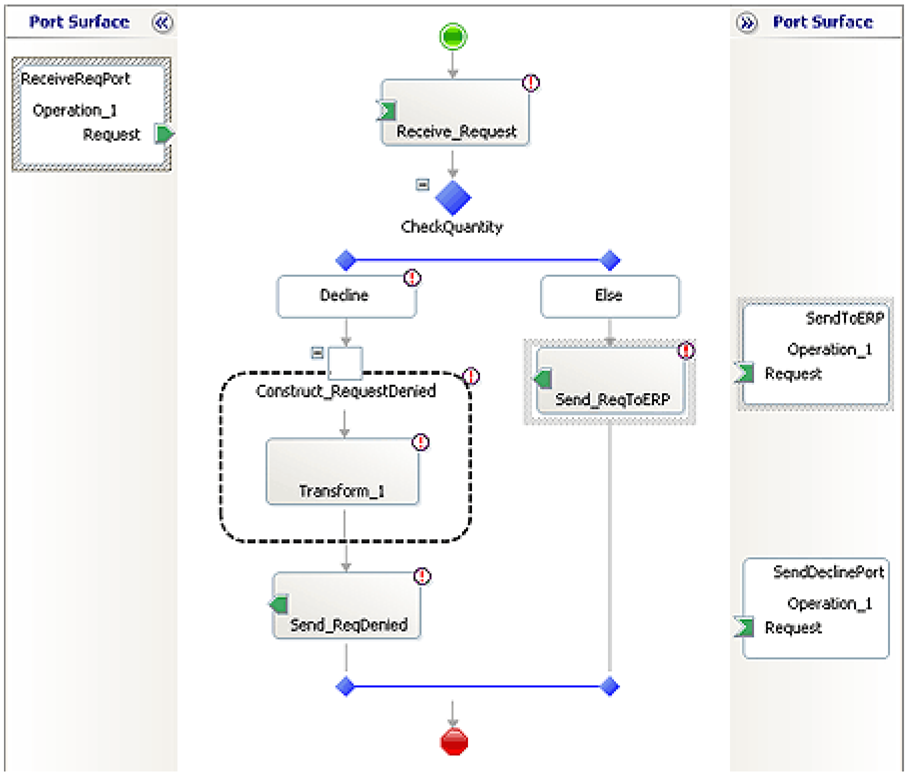
15. From the orchestration Toolbox, drag the **Port** shape to the right-side **Port Surface,** parallel to the **Send_ReqDenied** shape.

16. Use the information in the following table to create the **SendDeclinePort** send port.

Property	Value
Name	Type **SendDeclinePort**.
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **SendDeclineType**.
Communication Pattern	Select the **One Way** option.
Access Restrictions	Select the **Internal - limited to this project** option.

Port direction of communication	From the drop-down list, select **I'll always be sending messages on this port**.
Port bindings	From the drop-down list, select **Specify later**.

The following figure shows the EAIProcess orchestration with the ports.



17. On the **File** menu, click **Save All**.

Step 6: Specify Action Shape Messages and Connect the Ports

Purpose: Send actions send messages. Receive actions receive messages. By specifying the messages that send actions send, and receive actions receive, and then connecting the send and receive actions to the appropriate ports, you configure the processing of the messages into and out of BizTalk Server.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Add Ports to the Orchestration before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To specify the message sent or received by the action shapes

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

4. In Solution Explorer, double-click **EAIProcess.odx**.

5. In Orchestration Designer, on the design surface, select the **Receive_Request** action shape.

6. In the Properties pane for the **Receive_Request** action shape, from the **Message** drop-down list, select **RequestInstance**.

7. Use the information in the following table to specify the message for the **Send_ReqDenied** shape and the **Send_ReqToERP** shape.

Shape	Message
Send_ReqDenied	RequestDeniedInstance
Send_ReqToERP	RequestInstance

8. On the **File** menu, click **Save All**.

Connect the action shapes to the ports

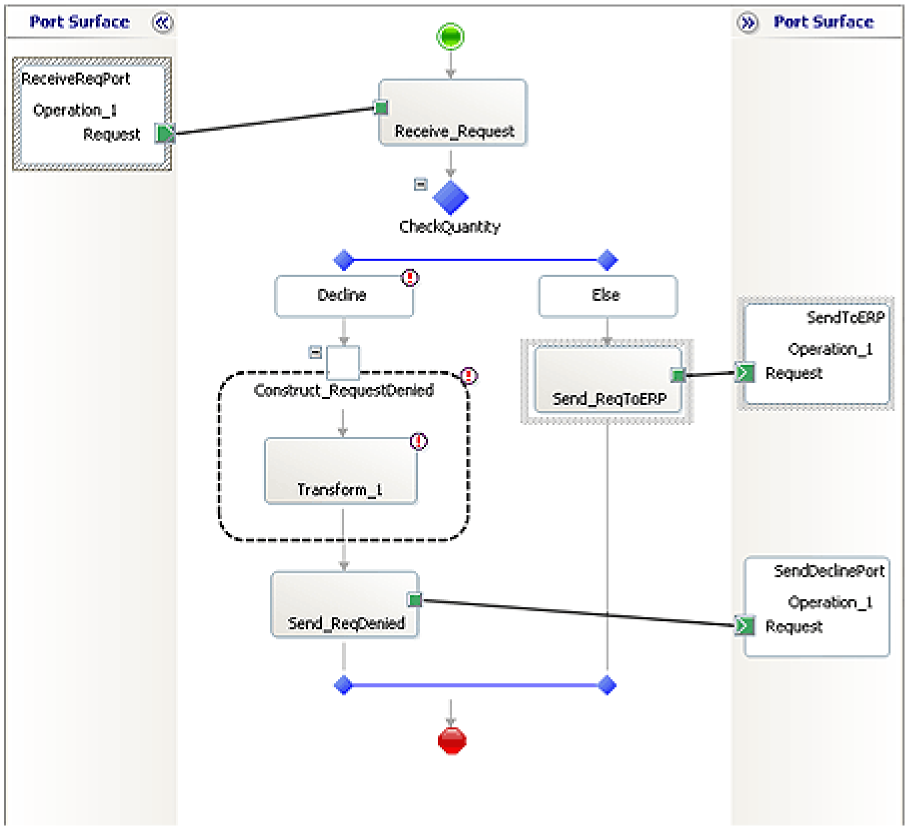
To connect the ports to the action shapes

1. In Orchestration Designer, on the design surface, drag the green arrow shaped handle for each port to the corresponding green handle of the action shape.

   Use the information in the following table to connect action shape messages to the appropriate send shapes.

Connect this port	To this action shape
ReceiveReqPort	Receive_Request
SendDeclinePort	Send_ReqDenied
SendToERP	Send_ReqToERP

   The following figure shows the EAIProcess orchestration with all of the ports connected.



2. On the **File** menu, click **Save All**.

Step 7: Configure the Transform Shape to Use the Map

Purpose: In Lesson 1, you defined the schema of the inventory replacement request message that the warehouse sends to BizTalk Server. You defined the schema of the request denied message that BizTalk Server sends to the warehouse. You defined a map that connects parts of the two schemas so that data from the request message can be included in the request denied message. When you connect the map to the transform action, the transform action uses the map to construct the request denied message with data from the request message.

In this step, you use the **Transform Configuration** dialog box to specify a map, and the input and output messages.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 6: Specify Action Shape Messages and Connect the Ports before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure the Transform action to use the map

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

4. In Solution Explorer, double-click **EAIProcess.odx**.

5. In Orchestration Designer, on the design surface, select the **Construct_RequestDenied** shape.

6. In the Properties pane, from the **Messages Constructed** drop-down list, select the **RequestDeniedInstance** check box.

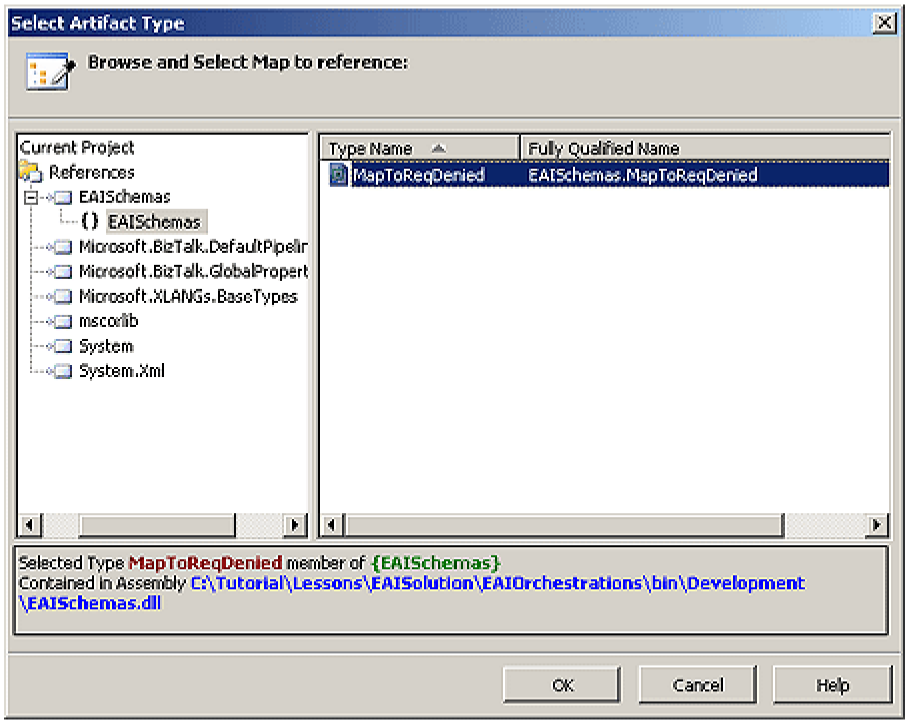
7. Double-click the **Transform_1** shape.

8. In the **Transform Configuration** dialog box, do the following:

Use this	**To do this**
Enter the configuration information	Select the **Existing Map** option.
Fully Qualified Map Name	From the drop-down list, select **Select from Referenced Assembly**.

9. In the **Select Artifact Type** dialog box, on the **EAISchemas** tab, from the **Type Name** list, select **MapToReqDenied**, and then click **OK**.

The following figure shows the Select Artifact Type dialog box.



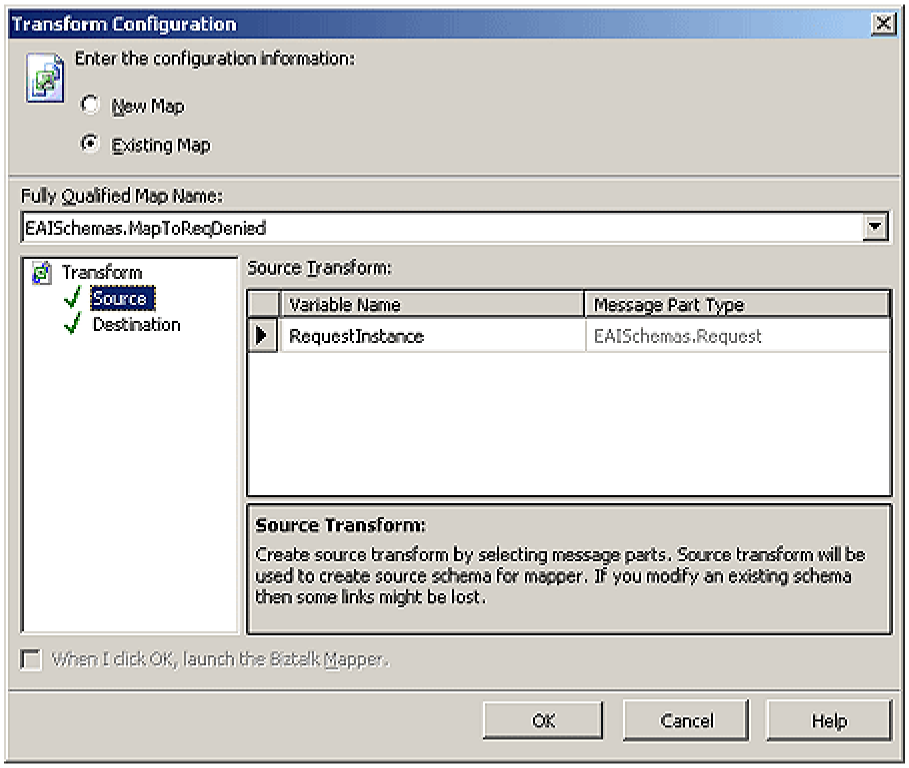
Type Name	Fully Qualified Name
MapToReqDenied	EAISchemas.MapToReqDenied

Selected Type MapToReqDenied member of {EAISchemas}
Contained in Assembly C:\Tutorial\Lessons\EAISolution\EAIOrchestrations\bin\Development\EAISchemas.dll

10. In the **Transform Configuration** dialog box, on the **Source** tab, from the **Variable Name** drop-down list, select **RequestInstance**.

11. In the **Transform Configuration** dialog box, on the **Destination** tab, from the **Variable Name** drop-down list, select **RequestDeniedInstance**, and then click **OK**.

The following figure shows the **Transform Configuration** dialog box.



Transform Configuration

Enter the configuration information:

New Map [✓ Existing Map]

Fully Qualified Map Name: EAISchemas.MapToReqDenied

Transform
- Source
- Destination

Source Transform:

Variable Name	Message Part Type
RequestInstance	EAISchemas.Request

Source Transform:
Create source transform by selecting message parts. Source transform will be used to create source schema for mapper. If you modify an existing schema then some links might be lost.

When I click OK, launch the Biztalk Mapper.

12. On the **File** menu, click **Save All**.

Step 8: Write the XLANG/s Expression for the Decision

Purpose: The expression you write in this step evaluates the value of the quantity field in the inventory replacement request message. If the quantity in a request instance exceeds 500, the orchestration declines the request. In Lesson 1, you defined the quantity field when you created the schema for the message. Because the expression evaluates which messages BizTalk Server rejects, you add the expression to the decline side of the decision shape.

You use the Expression Editor to create expressions to expand the capabilities of various orchestration shapes. In this procedure, you use it to construct a Boolean expression in the **Decision** shape. The Expression Editor includes an IntelliSense feature that helps guide you in creating the expression.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 7: Configure the Transform Shape to Use the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To write the XLANGs expression for the decision rule

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.

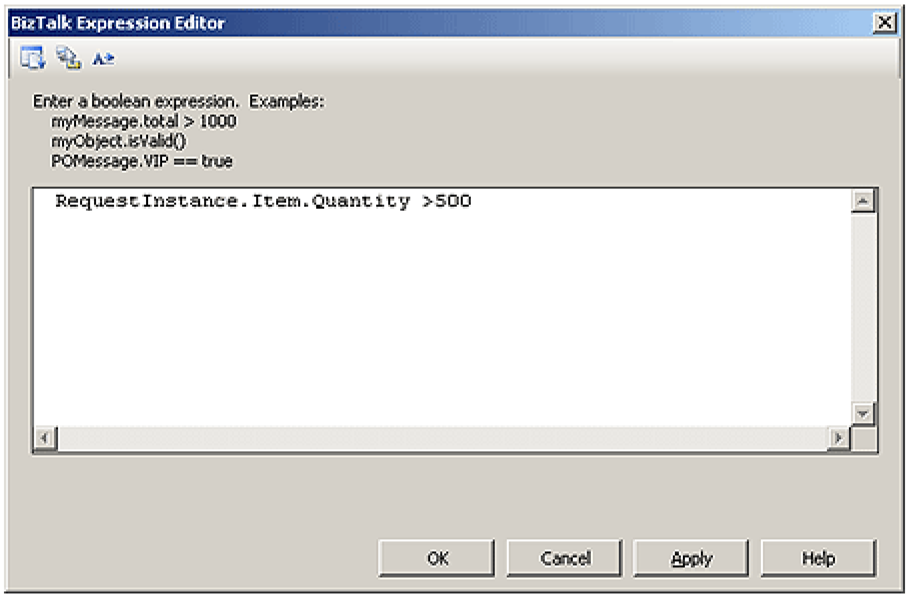
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.

4. In Solution Explorer, double-click **EAIProcess.odx**.

5. In Orchestration Designer, on the design surface, double-click the **Decline** shape.

6. In the BizTalk Expression Editor, type the following expression:

The following figure shows the Expression Editor with the expression.



Enter a boolean expression. Examples:
myMessage.total > 1000
myObject.isValid()
POMessage.VIP == true

```
RequestInstance.Item.Quantity >500
```

7. Click **OK** to close the BizTalk Expression Editor.

8. On the **File** menu, click **Save All**.

Step 9: Build the EAIOrchestrations Project

Purpose: After you create the EAIOrchestrations project and add the necessary items to it, you build the project to generate an assembly. An assembly is a collection of resources in the project that are stored in a DLL file.

All resources the EAIOrchestrations project requires must be available to it before you build the EAIOrchestrations project. You make the resources available by building the project that contains those resources. You did this when you built the EAISchemas project at the end of Lesson 1. In this case, when you built the EAISchemas project, you made the schemas available to the EAIProcess orchestration.

The output tab located at the bottom of your screen displays the results of building the project.

Prerequisites

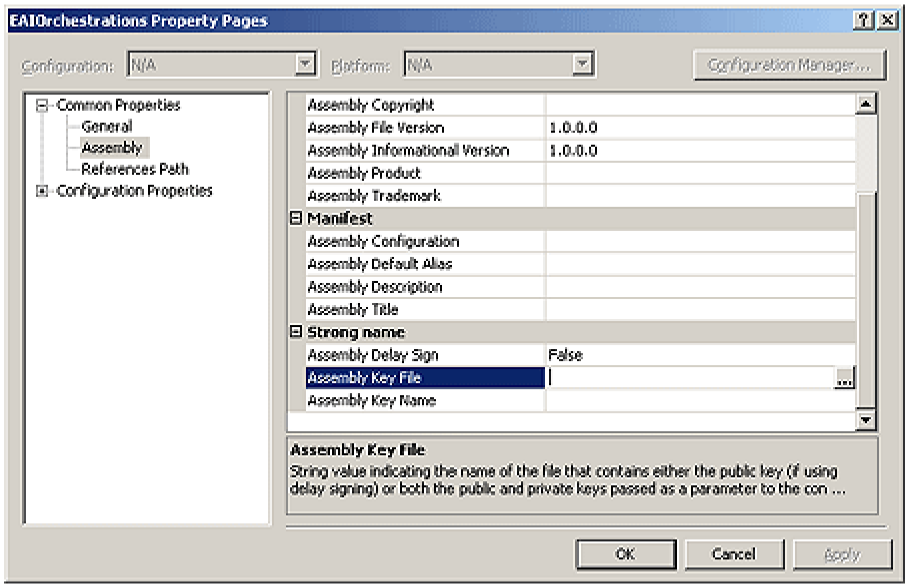
Note the following requirements before you begin this step:

- You must complete Step 8: Write the XLANG/s Expression for the Decision before you begin this step.
- You must login as a member of the BizTalk Server Administrators group.

To build the EAIOrchestrations project

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio 2005, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, click **EAISolution**, and then click **Open**.
4. In Solution Explorer, right-click the **EAIOrchestrations** project and click **Properties**.
5. In the **EAIOrchestrations Property Pages** dialog box, in the tree pane, expand **Common Properties**,click **Assembly**, and then click the **Assembly Key File** ellipsis [**...**].

   The following figure shows the **EAIOrchestrations Property Pages** dialog box.



Configuration: N/A   Platform: N/A

Assembly Copyright	
Assembly File Version	1.0.0.0
Assembly Informational Version	1.0.0.0
Assembly Product	
Assembly Trademark	
Manifest	
Assembly Configuration	
Assembly Default Alias	
Assembly Description	
Assembly Title	
Strong name	
Assembly Delay Sign	False
Assembly Key File	
Assembly Key Name	

Assembly Key File
String value indicating the name of the file that contains either the public key (if using delay signing) or both the public and private keys passed as a parameter to the con ...

6. In the **Assembly Key File** dialog box, do the following:

Use this	**To do this**
Look in	Navigate to **C:\Tutorial**.
File name	Type **Tutorial.snk**.

7. Click **Open**.

8. In the **EAIOrchestrations Property Pages** dialog box, in the tree pane, expand **Configuration Properties**, click **Deployment**, and then do the following:

Use this	**To do this**
Application Name	Type **EAIApplication**.
Redeploy	From the drop-down list, select **True**.

9. Click **OK**.

10. On the **File** menu, click **Save All**.

11. In Solution Explorer, right-click **EAIOrchestrations**, and then click **Build**.

    The Output pane at the bottom of the screen should read: **Build: 2 succeeded or up-to-date, 0 failed, 0 skipped.**

    You are now ready to deploy your solution to a testing environment.

Lesson 3: Deploy the Solution

The first step in deploying your BizTalk solution is to add it to the BizTalk Management database and the global assembly cache.

The second step is to configure EAIApplication and start it. In Lesson 2, you specified the EAIApplication application for both projects in the EAISolution solution. When you deploy the solution, BizTalk adds the projects and their artifacts to the EAIApplication application. You use the BizTalk Administration console to configure EAIApplication.

Configuring an application involves specifying a BizTalk host for the application, and binding the ports in the application. In Lesson 2, you added logical ports to the EAIProcess orchestration. In this lesson, you define the physical ports which include transport type and location for the ports.

The last step in the deployment process is to start the EAIApplication application. Starting EAIApplication starts the orchestration and the ports, so the solution can receive and process messages.

The lesson will take you approximately thirty minutes to complete.

To begin deploying the projects, go to Step 1: Deploy the Projects.

In this lesson, you learn about administering BizTalk Server 2006 artifacts by using the following tools:

- **BizTalk Administration console**. For information about using the BizTalk Administration console, see Using the BizTalk Server Administration Console.

In This Section

- Step 1: Deploy the Projects
- Step 2: Configure and Start the EAI Application

Step 1: Deploy the Projects

Purpose: You deploy a BizTalk Server project to place a copy of the assembly in the BizTalk Management database (also known as the Configuration database) and install it in the global assembly cache. You created assemblies when you built the EAISchemas and EAIOrchestrations projects in the preceding lesson.

In this step, you deploy both projects (also known as assemblies) in your solution: EAISchemas.dll and EAIOrchestrations.dll.

You can run the deploy process on the solution. Deploying the solution deploys both projects in the proper order. If you deploy the projects individually, you must deploy them in reverse order of dependency, starting from the independent assemblies and then moving on to the assemblies that are dependent upon those developed earlier.

In the EAISolution solution, the EAIOrchestrations project references schemas in the EAISchemas project. If you deploy the projects individually, you would deploy EAISchemas first. If you build or deploy projects in the wrong order, the process fails.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 9: Build the EAIOrchestrations Project before you begin this step.
- You must login as a member of the BizTalk Server Administrators group.

To deploy the projects

1. In Solution Explorer, right-click **Solution 'EAISolution'**, and then click Properties.
2. In the **Solution 'EAISolution' Properties Pages** dialog box, in the navigation pane, expand **Configuration Properties**, and then click **Configuration**.

3. On the **Configuration** page, in the **Deploy** column, select the check boxes for both the **EAISchemas** and **EAIOrchestrations** projects, and then click **OK**.

4. Right-click **Solution 'EAISolution'**, and then click **Deploy Solution**.

5. The Output pane at the bottom of the screen should read: **Deploy succeeded**.

Step 2: Configure and Start the EAI Application

Purpose: When you configure EAIApplication, you associate the logical artifacts you created in Visual Studio with their physical counterparts. In this step, you bind the EAIProcess orchestration to a BizTalk host. A BizTalk Server host is a BizTalk Server run-time process. In addition, you bind the logical ports you created in Step 5: Add Ports to the Orchestration to physical ports you define in this step.

You use the BizTalk Administration console to configure EAIApplication. In the following procedure, you select a host for the application, and define a receive location and physical receive port, and two physical send ports.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Deploy the Projects before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To configure EAIApplication

1. Click **Start**, point to **Program Files**, point to **Microsoft BizTalk Server 2006**, and then click **BizTalk Server Administration**.

2. In the console tree on the left side of the BizTalk Administration console, expand **BizTalk Server 2006 Administration**, right-click **BizTalk Group [<computer name>:BizTalkMgmtDb]**, and then click **Refresh**.

3. Expand **BizTalk Group [<computer name>:BizTalkMgmtDb]**, expand **Applications**, right-click **EAIApplication**, and then click **Configure**.

4. In the **Configure Application** dialog box, on the **EAIProcess** tab, do the following:

Use this	To do this
Host	From the drop-down list, select **BizTalkServerApplication**.
ReceiveReqPort	From the drop-down list under **Receive Ports**, select **New receive port**.

5. In the **ReceivePort1 - Receive Port Properties** dialog box, in the **Name** box, type **ReceivePort_ReceiveReq**.

6. On the **Receive Locations** tab, click **New**.

7. In the **ReceiveLocation1 - Receive Location Properties** dialog box, on the **General** tab, do the following:

Use this	**To do this**
Name	Type **ReceiveLocation_ReceiveReq**.
Type	From the drop-down list, select **FILE**, and then click **Configure**.

8. In **the FILE Transport Properties** dialog box, on the **General** tab, click **Browse**.

9. In the **Browse For Folder** dialog box, expand **My Computer**, navigate to **C:\Tutorial\Filedrop\ReceiveRequest**, and then click **OK**.

10. In the **FILE Transport Properties** dialog box, click **OK**.

11. In the **ReceiveLocation_ReceiveReq - Receive Location Properties** dialog box, from the **Receive pipeline** drop-down list, select **XMLReceive**, and then click **OK**.

12. In **the ReceivePort_ReceiveReq - Receive Port Properties** dialog box, click **OK**.

13. In the **Configure Application** dialog box, on the **EAIProcess** tab, from the drop-down list for **SendDeclinePort**, under **Send Ports**, select **New send port**.

14. In the **SendPort1 - Send Port Properties** dialog box, on the **General** tab, do the following:

Use this	**To do this**
Name	Type **SendPort_SendDeclinePort**.
Type	From the drop-down list, select FILE, and then click **Configure**.

15. In **the FILE Transport Properties** dialog box, on the **General** tab, click **Browse**.

16. In the **Browse For Folder** dialog box, expand **My computer**, navigate to **C:\Tutorial\FileDrop\RequestDenied**, and then click **OK**.

17. In the **FILE Transport Properties** dialog box, on the **General** tab, in **the File name** box, type **Denied%MessageID%.xml**, and then click **OK**.

18. In the **SendPort_SendDeclinePort - Send Port Properties** dialog box, click **OK**.

19. In the **Configure Applications** dialog box, on the **EAIProcess** tab, from the drop-down list for **SendToERP**, under **Send Ports**, select **New send port**.

20. In the **SendPort1 - Send Port Properties** dialog box, do the following:

Use this	To do this
Name	Type **SendPort_SendToERP**.
Type	From the drop-down list, select **FILE**, and then click **Configure**.

21. In the **FILE Transport Properties** dialog box, on the **General** tab, click **Browse**.

22. In the **Browse For Folder** dialog box, expand **My Computer**, navigate to **C:\Tutorial\FileDrop\ERPSys**, and then click **OK**.

23. In the **FILE Transport Properties** dialog box, click **OK**.

24. In the **SendPort_SendToERP - Send Port Properties** dialog box, click **OK**.

25. In the **Configure Application** dialog box, click **OK**.

26. In the console tree, right-click **EAIApplication**, and then click **Start**.

27. In the **Start 'EAIApplication' Application** dialog box, click **Options**, ensure that all of the check boxes are selected, and then click **Start**.

Step 3: Test the EAI Solution

Purpose: In this step, you check that the EAIProcess orchestration processes messages correctly. You do this by dropping sample messages into the receive location specified for the EAI application. If the EAI solution is working properly, if the EAIProcess orchestration receives a message from the warehouse requesting more than 500 items, the orchestration generates a decline request message. If the EAIProcess orchestration receives a message from the warehouse requesting fewer than 500 items, the orchestration passes the message on to the ERP system.

Prerequisites

You must complete Step 2: Configure and Start the EAI Application before you begin this step.

To test the EAI solution

1. In Windows Explorer, navigate to **C:\Tutorials**.

2. Copy **RepRequestAccepted.xml** and paste it into **C:\tutorial\Filedrop\ReceiveRequest**.

3. When the file disappears, check **C:\tutorial\Filedrop\ERPSys**.

The file name of the approved message is now a guid. The RepRequestAccepted.xml message meets the condition to be approved. You set the condition for approving messages in the orchestration.

4. In Windows Explorer, navigate to **C:\Tutorials**.

5. Copy **RepRequestDenied.xml** and paste it into **C:\tutorial\Filedrop\ReceiveRequest**.

6. When the file disappears, check **C:\tutorial\Filedrop\RequestDenied**.

   The file name of the denied message is now "Denied" followed by a guid. The RepRequestDenied.xml message does not meet the condition to be approved. You set the condition for approving messages in the orchestration. You added "Denied" to the file name when you set the transport properties for the **SendPort_SendDeclinePort** send port.

Tutorial 2: Purchase Order Process

In this tutorial, you automate generating and sending purchase orders to a supplier.

This tutorial expands the solution that begins in **Tutorial 1: Enterprise Application Integration**by adding an inventory procurement solution that automates the exchange of data between the ERP solution and an outside supplier that fulfills purchase orders. In the tutorial, a Web service acts as the supplier.

In Tutorial 1: Enterprise Application Integration, you create an inventory replenishment business solution. The solution automates the exchange of data between a warehouse and an Enterprise Resource Planning (ERP) system.

When the inventory at the warehouse reaches a certain level, the warehouse sends a replenishment request to BizTalk. If the request is approved, BizTalk sends it to the ERP system. When the ERP system receives the approved request, the ERP system generates a purchase order (PO).

The scenario in this tutorial begins when the ERP process generates the purchase order. Within the purchase order solution, several things happen:

1. The ERP system generates a purchase order.

2. BizTalk Server maps the purchase order to the CommonPO schema.

3. BizTalk Server routes the CommonPO to the supplier.

4. The ERP system sends a purchase order confirmation message to the warehouse.

5. The ERP system receives an advanced shipping notice from the supplier that indicates what will be shipped and when.

BizTalk Server 2006 ensures that the ERP system and the supplier interact smoothly and efficiently. This includes applying business rules to the payment approval process and routing the payment appropriately.

The scenario is continued in Tutorial 3: Invoice and Payment Process.

In This Section

- Lesson 1: Set Up the Supplier Side
- Lesson 2: Create the B2B Solution
- Lesson 3: Add Functoids to the Map
- Lesson 4: Design the Purchase Order Process
- Lesson 5: Build and Deploy the Projects

Lesson 1: Set Up the Supplier Side

In this lesson, you set up the supplier side of the scenario.

In This Section

- Step 1: Set Up the Accounts for the Tutorial
- Step 2: Enable Web Service Publishing
- Step 3: Exclude the Root Directory from Windows SharePoint Services Managed Paths
- Step 4: Set Up the Supplier Web Service

Step 1: Set Up the Accounts for the Tutorial

Purpose: You must add the account you used to install and configure BizTalk Server 2006 to the following Windows user groups:

- **BizTalk Isolated Host Users**. The BizTalk Isolated Host Users group gives transports that run outside of the BizTalk Server process, such as HTTP and SOAP transports, access to BizTalk Server databases. The BizTalk Isolated Host Users group provides Windows Integrated Security access to the Microsoft SQL Server computer that contains the BizTalk Management database.
- **IIS_WPG**. The Information Services Worker Process user group provides the minimum set of privileges that are required by IIS, and it provides a convenient way to use a specific user for the identity account without having to manually assign the privileges to that identity. In cases where the account is not in the IIS_WPG group and does not have the appropriate permissions, the worker process fails to start.

Prerequisites

Note the following requirements before you begin this step:

- You must complete the steps in Before You Begin the Tutorials before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To add an account to the BizTalk Isolated Host Users group

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Computer Management**.
2. In **Computer Management**, expand **Local Users and Groups**, and then click **Groups**.
3. In the **Groups** results pane, right-click **BizTalk Isolated Host Users**, and then click **Properties**.
4. In the **BizTalk Isolated Host Users Properties** dialog box, in the **Members** list, check for the account you are using to set up and run this tutorial.

   If the account is not in the **Members** list, do the following:

   a. Click **Add**.

   b. In the **Select Users, Computers, or Groups** dialog box, in the **Enter the object name to select** box, type the account name in the form *<domain>\account name*, and then click **OK**.
5. Click **OK**.

To add an account to the IIS_WPG group

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Computer Management**.
2. In **Computer Management**, expand **Local Users and Groups**, and then click **Groups**.
3. In the **Groups** results pane, right-click **IIS_WPG**, and then click **Properties**.
4. In the **IIS_WPG Properties** dialog box, in the **Members** list, check for the account you are using to set up and run this tutorial.

   If the account is not in the **Members** list, do the following:

   a. Click **Add**.

   b. In the **Select Users, Computers, or Groups** dialog box, in the **Enter the object name to select** box, type the account name in the form *<domain>\account name*, and then click **OK**.
5. Click **OK**.

Step 2: Enable Web Service Publishing

Purpose: Publishing Web services enables you to create a Web service that can submit messages to BizTalk Server for use by orchestrations and other send adapters. You use the BizTalk Web Services Publishing Wizard to create published Web services.

For more information about publishing Web services in BizTalk Server 2006, see Publishing Web Services .

The instructions for enabling Web service publishing are different for computers running Windows Server 2003 than they are for computers running Windows 2000 or Windows XP. Use the instructions for your operating system:

- Windows Server 2003: Enable Web Service Publishing
- Windows XP and Windows 2000: Enable Web Service Publishing

In This Section

- Windows Server 2003: Enable Web Service Publishing
- Windows XP and Windows 2000: Enable Web Service Publishing

Windows Server 2003: Enable Web Service Publishing

You must perform the following tasks to enable BizTalk Server Web service publishing in Windows Server 2003:

1. Create a new application pool for Web service publishing.
2. Configure the new application pool to run under the user account you added to the BizTalk Isolated Host Users group.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Set Up the Accounts for the Tutorial before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create a new application pool for Web service publishing

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Internet Information Services Manager**.
2. In **Internet Information Services Manager**, expand **<computer name> (local computer)**, right-click **Application Pools**, click **New**, and then click **Application Pool**.

3. In the **Add New Application Pool** dialog box, do the following:

Use this	To do this
Application pool ID	Type **TutorialAppPool**.
Application pool settings	Select the **Use default settings for new application pool** option.

4. Click **OK**.

To configure the new application pool to run under the user account you added to the BizTalk Isolated Host Users group

1. In **Internet Information Services (IIS) Manager**, expand **<computer name> (local computer)**, expand **Application Pools**, right-click **TutorialAppPool**, and then click **Properties**.

2. In the **TutorialAppPool Properties** dialog box, on the **Identity** tab, do the following:

Use this	To do this
Application pool identity	Select the **Configurable** option.
User name	Type the user name of the account you added to the IIS_WPG group in Step 1: Set Up the Accounts for the Tutorial.
Password	Type the password for the account. Type the password a second time to confirm it.

3. Click **OK**.

Windows XP and Windows 2000: Enable Web Service Publishing

On Windows XP and Windows 2000, the BizTalk Server Web services run in the ASP.NET worker process (aspnet_wp.exe). This process runs under the ASPNET user context by default. To provide access to the BizTalk Management database, you must add the local ASP .NET user account to the BizTalk Isolated Host Users group.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Set Up the Accounts for the Tutorial before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To add ASPNET to the BizTalk Isolated Host Users group

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Computer Management**.

2. In **Computer Management**, expand **Local Users and Groups**, and then click **Groups**.

3. In the **Groups** results pane, right-click **BizTalk Isolated Host Users**, and then click **Properties**.

4. In the **BizTalk Isolated Host Users Properties** dialog box, in the **Members** list, click **Add**.

5. In the **Select Users, Computers, or Groups** dialog box, in the **Enter the object name to select** box, type **ASPNET**, and then click **OK**.

6. Click **OK** to close the **BizTalk Isolated Host Users Properties** dialog box.

Step 3: Exclude the Root Directory from Windows SharePoint Services Managed Paths.

Purpose: Because Windows SharePoint Services is configured in the default Web site, it can interfere with Web services. By excluding the root directory from Windows SharePoint Services managed paths, you enable Web services to co-exist with Windows SharePoint Services.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Enable Web Service Publishing before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To exclude the root directory from Windows SharePoint Services managed paths

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **SharePoint Central Administration**.

2. On the **Windows SharePoint Services Central Administration** page, in the **Virtual Server Configuration** section, click **Configure virtual server settings**.

3. On the **Virtual Server List** page, click **Default Web Site**.

4. On the **Virtual Server Settings** page, in the **Virtual Server Management** section, click **Define managed paths**.

5. On the **Define Managed Paths** page, in the **Included Paths** section, select the **(root)** check box, and then click **Remove selected paths**.

6. Close the browser window.

Step 4: Set Up the Supplier Web Service

Purpose: Because the BizTalk Server 2006 tutorials are designed to work on one computer, you use a configuration script to set up the provided supplier Web service. The script creates the virtual directories in Internet Information Services (IIS) and builds the supplier Web service.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Exclude the Root Directory from Windows SharePoint Services Managed Paths before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To run the supplier Web service configuration scripts

1. Use Windows Explorer to run the scripts in the following table in the order presented. Make sure that you wait until each script is finished before starting the next script.

Script location	Script file
C:\Tutorial\Solutions\SupplierWebServices\	Setup.bat
C:\Tutorial\Solutions\	SetInstallPathEnv.vbs
C:\Tutorial\Solutions\B2BSolution\	SetupWebService.bat

Lesson 2: Create the B2B Solution

In this lesson, you set up the business-to-business (B2B) solution and projects, add a reference to the supplier's Web service, and add existing items to one of the projects.

In This Section

- Step 1: Open a Blank Solution
- Step 2: Add the Projects to the Solution
- Step 3: Add a Reference to the Supplier's Web Service
- Step 4: Add Existing Schemas and Maps

Step 1: Open a Blank Solution

Purpose: You begin building the B2B solution by opening a blank solution in the Visual Studio environment. In this procedure, you create a solution that contains a project file for the schemas and a separate project file for the orchestrations.

Prerequisites

Note the following requirements before you begin this step:

- You must complete the steps in Before You Begin the Tutorials before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To open a blank solution in Visual Studio

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. On the **File** menu, point to **New**, and then click **Project**.
3. In the **New Project** dialog box, do the following:

Use this	To do this
Project types	In the tab list, click **BizTalk Projects**.
Templates	Click **Empty BizTalk Server Project**.
Name	Type **B2BSolution**.
Location	Type **C:\Tutorial\Lessons**.
Create directory for solution	Select this check box to automatically create the C:\Tutorial\Lessons\B2BSolution directory.

4. Click **OK**.

   Visual Studio 2005 adds both the B2BSolution solution and a B2BSolution project. We do not use the B2BSolution project.

Remove the B2BSolution project

1. In Solution Explorer, right-click the **B2BSolution** project, and then click **Remove**.
2. In the message box, click **OK**.

3. On the **File** menu, click **Save All**.

Step 2: Add the Projects to the Solution

Purpose: The core element of your solution is a BizTalk project — a collection of items, such as schemas, orchestrations, message types, and references that you build and generate into an assembly before deploying it. In this procedure, you create a project file for the schemas and a separate project file for the orchestration that will be used in the business-to-business (B2B) process.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Open a Blank Solution before you begin this step.
- You must login as a member of the BizTalk Server Administrators group.

To add a new project to your solution

1. In Solution Explorer, right-click **Solution 'B2BSolution'**, point to **Add**, and then click **New Project**.
2. In the **Add New Project** dialog box, do the following:

Use this	**To do this**
Project types	In the tab list, click **BizTalk Projects**.
Templates	Click **Empty BizTalk Server Project**.
Name	Type **B2BSchemas**.
Location	Verify that the location is **C:\Tutorial\Lessons\B2BSolution**.

3. Click **OK**.
4. Repeat steps 1 through 3 to create a second project in the solution named **B2BOrchestrations**.
5. On the **File** menu, click **Save All**.

Step 3: Add a Reference to the Supplier's Web Service

Purpose: When you add a Web reference, the items within the Web service become available to your solution. The supplier Web service that you use for this tutorial has two services, one for the purchase order and one for the payment. You add both in the following procedure.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Add the Projects to the Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add references to the supplier's Web service

1. In Solution Explorer, in the **B2BSchemas** project, right-click **References**, and click **Add Web Reference**.
2. In the **Add Web Reference** dialog box, in the **URL** combo box, type **http://localhost/B2BSupplierProcessPO/Process.asmx**, and then click **Go**.
3. When the service is found, click **Add Reference**.
4. Repeat steps 1 through 3 for **http://localhost/B2BSupplierProcessPayment/Payment_Service.asmx**
5. On the **File** menu, click **Save All**.

Step 4: Add Existing Schemas and Maps

Purpose: We have provided the schemas and maps you need to complete the tutorial. You must add the schemas and maps to the B2BSchemas project.

For information about creating schemas and maps from scratch, see Tutorial 1: Enterprise Application Integration.

Prerequisites

Note the following requirements before you begin this step:

- You must complete the steps in Step 3: Add a Reference to the Supplier's Web Service before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add existing schemas and maps to your project

1. In Solution Explorer, right-click **B2BSchemas**, point to **Add**, and then click **Existing Item**.

2. In the **Add Existing Item** dialog box, navigate to **C:\Tutorial\Solutions\Schemas**, and then do the following:

   Press CTRL while you click the following schemas and maps:

   - **AdvancedShipNotice.xsd**
   - **CommonBaseTypes.xsd**
   - **CommonInvoice.xsd**
   - **InvoiceToPayment.btm**
   - **MapToCommonPO.btm**
   - **PO.xsd**

3. Click **Add**.

4. On the **File** menu, click **Save All**.

Lesson 3: Add Functoids to the Map

In this lesson, you add functoids to the MapToCommonPO map that you added to the B2BSchemas project in the previous lesson. You add functoids to a map to transform data that is mapped from the source schema to the destination schema.

In This Section

- Step 1: Add a Date/Time Functoid to the Map
- Step 2: Add a Multiplication Functoid to the Map
- Step 3: Add a Looping Functoid to the Map
- Step 4: Add a Record Count Functoid to the Map
- Step 5: Add a Cumulative Sum Functoid to the Map
- Step 6: Assign a Constant Value in the Map
- Step 7: Build the B2BSchema Project and Test the Map

Step 1: Add a Date/Time Functoid to the Map

Purpose: You add functoids to a map to transform data that is mapped from the source schema to the destination schema. The **Date** functoid you add in this step generates the current date stamp for the **CreationDate** field in CommonPO. For more information, see Date and Time Functoids .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 2: Create the B2B Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add the Date functoid

1. In Solution Explorer, expand **B2BSchemas**, and then double-click **MapToCommonPO.btm**.
2. In **MapToCommonPO.btm**, expand the **Source Schema** and the **Destination Schema**.
3. On the **View** menu, click **Toolbox**, and then in the Toolbox, click the pushpin to dock it.
4. In the **Toolbox**, expand **Date/Time Functoids**, drag the **Date** functoid to the mapping grid, and drop it parallel to **CommonPO\POHeader\CreationDate** in the destination schema.
5. In the **Destination Schema** schema tree, select and drag the **CreationDate** field to the **Date** functoid in the mapping grid.
6. On the **File** menu, click **Save All**.

Step 2: Add a Multiplication Functoid to the Map

Purpose: You add functoids to a map to transform data that is mapped from the source schema to the destination schema. You use the **Multiplication** functoid to multiply the **Quantity** field by the **Price** field in the source schema, and place this resulting value in the **ExtendedPrice** field in the destination schema. For more information, see **Mathematical Functoids Reference** .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Add a Date/Time Functoid to the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add the Multiplication functoid

1. In Solution Explorer, expand **B2BSchemas**, and then double-click **MapToCommonPO.btm**.

2. In **MapToCommonPO.btm**, expand the **Source Schema** and the **Destination Schema**.

3. On the **View** menu, click **Toolbox**, and then in the **Toolbox**, click the pushpin to dock it.

4. In the **Toolbox**, expand **Mathematical Functoids**, and then drag the **Multiplication** functoid to the mapping grid near the **Item** node in the source schema.

5. In the **Source Schema** tree, drag the **Item\Quantity** field to the functoid.

6. In the **Source Schema** tree, drag the **Item\Price** field to the functoid.

7. In the **Destination Schema**, expand **Item\ItemHeader\<Equivalent>\<ItemHeaderType>**, and then drag the **Multiplication** functoid to the **ExtendedPrice** field in the **Destination Schema** tree.

8. On the **File** menu, click **Save All**.

Step 3: Add a Looping Functoid to the Map

Purpose: You add functoids to a map to transform data that is mapped from the source schema to the destination schema. The **Looping** functoid combines multiple items into a single repeating structure. You use a **Looping** functoid because there can be several items on a PO. For more information, see **Looping Functoid** .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Add a Multiplication Functoid to the Map before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To add a Looping functoid

1. In Solution Explorer, expand **B2BSchemas**, and then double-click **MapToCommonPO.btm**.

2. In **MapToCommonPO.btm**, expand the **Source Schema** and the **Destination Schema**.

3. On the **View** menu, click **Toolbox**, and then in the **Toolbox**, click the pushpin to dock it.

4. In the **Toolbox**, expand **Advanced Functoids**, and then drag the **Looping** functoid to the mapping grid near the **Source Schema Item** node.

5. In the **Source Schema** tree, drag the **Item** node to the **Looping** functoid.

6. In the **Destination Schema** tree, drag the **Item** node to the **Looping** functoid.

7. On the **File** menu, click **Save All**.

Step 4: Add a Record Count Functoid to the Map

Purpose: You add functoids to a map to transform data that is mapped from the source schema to the destination schema. You use a **Record Count** functoid to count the number of items listed in the PO. For more information, see **Record Count Functoid** .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Add a Looping Functoid to the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a Record Count functoid

1. In Solution Explorer, expand **B2BSchemas**, and then double-click **MapToCommonPO.btm**.

2. In **MapToCommonPO.btm**, expand the **Source Schema** and the **Destination Schema**.

3. On the **View** menu, click **Toolbox**, and then in the **Toolbox**, click the pushpin to dock it.

4. In the **Toolbox**, expand **Advanced Functoids**, and then drag the **Record Count** functoid to the mapping grid near the **Source Schema Item** node.

5. In the **Source Schema** tree, drag the **Item** node to the **Record Count** functoid.

6. In the **Destination Schema** tree, drag **Total\LineItemTotal** to the **Record Count** functoid.

7. On the **File** menu, click **Save All**.

Step 5: Add a Cumulative Sum Functoid to the Map

Purpose: You add functoids to a map to transform data that is mapped from the source schema to the destination schema. You use a **Cumulative Sum** functoid to calculate the sum of the quantity of all items ordered. For more information, see **Cumulative Functoids Reference** .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Add a Record Count Functoid to the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a Cumulative Sum functoid

1. In Solution Explorer, expand **B2BSchemas**, and then double-click **MapToCommonPO.btm**.
2. In **MapToCommonPO.btm**, expand the **Source Schema** and the **Destination Schema**.
3. On the **View** menu, click **Toolbox**, and then in the **Toolbox**, click the pushpin to dock it.
4. In the **Toolbox**, expand **Cumulative Functoids**, and then drag the **Cumulative Sum** functoid to the mapping grid near the **Item** node.
5. In the **Source Schema** tree, drag the **Item\Quantity** field to the functoid.
6. In the **Destination Schema** tree, drag the **Total\QuantityTotal** field to the functoid.
7. In the **Toolbox**, drag a second **Cumulative Sum** functoid to the mapping grid and place it to the right of the **Multiplication** functoid.
8. On the mapping grid, select the **Multiplication** functoid and drag it to the **Cumulative Sum** functoid.
9. In the **Destination Schema** tree, drag the **Total\POTotal** field to the **Cumulative Sum** functoid.
10. On the **File** menu, click **Save All**.

Step 6: Assign a Constant Value in the Map

Purpose: You assign a constant value to a field in the destination schema for the name of your company (My Company Name). This will be the value of the **BillTo/Address/Name** field for every message based on the destination schema.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Add a Cumulative Sum Functoid to the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To assign a constant value to the BillTo record

1. In Solution Explorer, expand **B2BSchemas**, and then double-click **MapToCommonPO.btm**.
2. In **MapToCommonPO.btm**, expand the **Destination Schema**.
3. In the **Destination Schema** tree, click the **BillTo/Address/Name** field.
4. On the **View** menu, click **Properties Window**.
5. In the Properties pane, in the **Value** combo box, type **My Company Name**, and then press ENTER.
6. On the **File** menu, click **Save All**.

Step 7: Build the B2BSchema Project and Test the Map

Purpose: After you create the B2BSchemas project and add the necessary items to it, you must build the project to generate an assembly. An assembly is a collection of resources in the project, such as schemas and maps, that are stored in a DLL file.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 6: Assign a Constant Value in the Map before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To build the B2BSchemas project

1. In Solution Explorer, right-click **B2BSchemas**, and then click **Build**.

To test the map against an instance of the PO

1. In Solution Explorer, right-click **MapToCommonPO.btm**, and then click **Properties**.
2. In the **MapToCommonPO.btm Property Pages** dialog box, click the **TestMap Input Instance** ellipsis [**...**].
3. In the **Select Input File** dialog box, browse to **C:\Tutorial**, select **POInstance.xml**, and then click **Open**.
4. In the **MapToCommonPO.btm Property Pages** dialog box, in the **TestMapInput** drop-down list, select **XML**, and then click **OK**.
5. In Solution Explorer, right-click **MapToCommonPO.btm**, and then click **Test Map**.

6. On the **Output** tab located at the bottom of the screen, verify that the test succeeded and open the output file specified on the **Output** tab.

   The output file shows you what the PO XML document looks like after it is transformed by the map.

7. On the **File** menu, click **Save All**.

Lesson 4: Design the Purchase Order Process

In this lesson, you create the orchestration for the B2BSolution. The orchestration receives purchase orders (PO) and then performs a parallel action: on one side, a confirmation message is sent back to the originator of the PO. On the other side, the PO is transformed into a CommonPO message and sent out to the supplier.

In This Section

- Step 1: Add an Orchestration to the Solution
- Step 2: Add a Reference to the B2BSchemas Project
- Step 3: Set Up the Deployment Environment
- Step 4: Create the Purchase Order Process
- Step 5: Create the Message Instance Variables
- Step 6: Configure the Message Assignment Shape
- Step 7: Configure the Transform Shape
- Step 8: Create and Configure the Internal Ports
- Step 9: Create and Configure the External Port
- Step 10: Connect the Ports

Step 1: Add an Orchestration to the Solution

Purpose: An orchestration is a representation of your business process. In later steps, you will add action shapes that automate the business process.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 3: Add Functoids to the Map before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To add an orchestration to the project

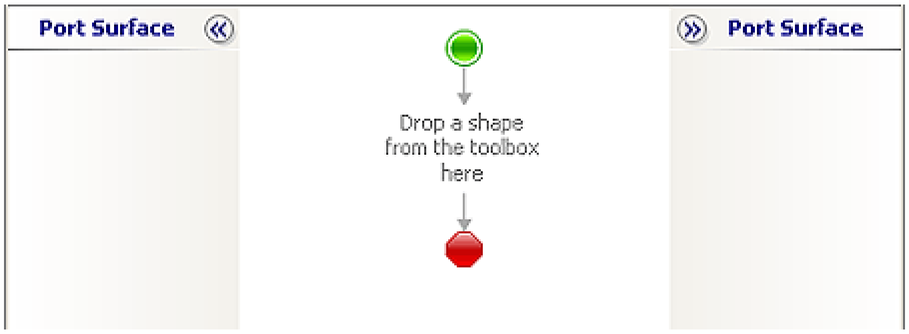
1. In Solution Explorer, right-click **B2BOrchestrations**, point to **Add**, and then click **New Item**.

2. In the **Add New Item** dialog box, do the following:

Use this	**To do this**
Categories	Click **Orchestration Files**.
Templates	Click **BizTalk Orchestration**.
Name	Type **B2BProcess.odx**.

3. Click **Add**.

The following figure shows the orchestration design surface for the B2BProcess orchestration.



4. On the **File** menu, click **Save All** to save your work.

Step 2: Add a Reference to the B2BSchemas Project

Purpose: The B2BProcess orchestration processes messages defined with the schemas in the B2BSchemas project. You add a reference from the B2BOrchestrations project to the B2BSchemas project to make the schemas in the B2BSchemas project available to the orchestration in the B2BOrchestrations project. The new reference in the B2BOrchestrations project references the B2BSchemas assembly file (DLL) that you created when you built the B2BSchemas project in Lesson 3.

Prerequisites

Note the following requirements before you begin this step:

- You must build the B2BSchemas project before you begin this step. You built the B2BSchemas project when you completed Step 7: Build the B2BSchema Project and Test the Map in the previous lesson.
- You must complete Step 1: Add an Orchestration to the Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a reference to the B2BSchemas project

1. In Solution Explorer, expand the **B2BOrchestrations** project, right-click the **References** node, and then click **Add Reference**.
2. In the **Add Reference** dialog box, click the **Projects** tab, and then double-click the **B2BSchemas** project to add it as a selected component.
3. Verify that **B2BSchemas** is listed as a selected component, and then click **OK**.
4. Select the **B2BSchemas** reference, and in the Properties pane, from the **Copy Local** drop-down list, select **False**.
5. On the **File** menu, click **Save All** to save your work.

Step 3: Set Up the Deployment Environment

Purpose: You must assign the assembly key file for the project to be able to build and deploy your project successfully. Setting the **Redeploy** property to **True** enables you to redeploy your project as many times as necessary during development without changing the version number. This is useful when you are making numerous iterative changes during development. However, because the **Redeploy** option bypasses version control, use it only during development.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Create the Purchase Order Process before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To set the development environment

1. In Solution Explorer, right-click **B2BOrchestrations**, and then click **Properties**.
2. In the **B2BOrchestrations Property Pages** dialog box, in the left pane, expand **Common Properties**, and then click **Assembly**.
3. In the right pane, click the **Assembly Key File** ellipsis [**...**].

4. In the **Assembly Key File** dialog box, browse to **C:\Tutorial**, click **Tutorial.snk**, and then click **Open**.

5. In the **B2BOrchestrations Property Pages** dialog box, in the left pane, expand **Configuration Properties**, and then click **Deployment**.

6. On the right side, from the **Redeploy** drop-down list, click **True**.

7. Click **OK** to close the **B2BOrchestrations Property Pages** dialog box.

8. Repeat steps 1 through 7 for the **B2BSchemas** project to select **Tutorial.snk** as the assembly key file and set the **Redeploy** property to **True**.

9. On the **File** menu, click **Save All** to save your work.

Step 4: Create the Purchase Order Process

Purpose: In Orchestration Designer, you define the first part of your business process by using shapes such as **Receive**, **Parallel Actions**, **Message Assignment**, and **Transform** instead of more complex items, such as documents, schemas, components, and messages.

The **Receive** shape receives the purchase order (PO) and starts the orchestration. If a **Receive** shape is the first action in an orchestration, it must have the **Activate** property set to **True** to run, unless the orchestration is called by another orchestration.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Set Up the Deployment Environment before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To add the Receive, Parallel Actions, and Construct shapes

1. In Orchestration Designer, from the Toolbox, drag the **Receive** shape to the orchestration design surface and drop the shape between the **Begin** (green circle) and **End** (red octagon) shapes.

2. In the Properties pane, change the **Name** property to **Receive_PO** and set **Activate** to **True**.

3. From the Toolbox, drag the **Parallel Actions** shape to the orchestration design surface and drop the shape on the connecting line directly below the **Receive_PO** shape.

   The **Parallel Actions** shape expands to show two branches for different actions that take place concurrently but independently.

4. In the Properties pane, change the **Name** property to **Route_PO**.

5. From the Toolbox, drag the **Message Assignment** shape to the orchestration design surface, and drop the shape on the left side in the **Route_PO** shape.

6. Select the **ConstructMessage_1** shape, and change the **Name** property to **Construct_POConfirmed**.

7. From the Toolbox, drag the **Send** shape to the orchestration design surface, drop the shape on the connecting line directly below the **Construct_POConfirmed** shape, and change the **Name** property to **Send_POConfirmed**.

8. From the Toolbox, drag the **Transform** shape to the orchestration design surface, and drop the shape on the right side of the **Route_PO** shape.

9. Select the **ConstructMessage** shape, and change the **Name** property to **Construct_CommonPO**.

10. From the Toolbox, drag the **Send** shape to the orchestration design surface, drop the shape on the connecting line directly below the **Construct_CommonPO** shape, and change the **Name** property to **Send_CommonPO**.

11. On the **File** menu, click **Save All** to save your work.

Step 5: Create the Message Instance Variables

Purpose: Orchestration Designer uses message instances to manage data. You use a message instance variable to indicate the schema for the message instance.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Create the Purchase Order Process before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the message instance variables

1. In Visual Studio, open the B2BProcess orchestration.

2. In **Orchestration View**, right-click **Messages**, and then click **New Message**.

3. Select **Message_1**, in the Properties pane, in the **Identifier** property, type **PO_Instance**, and then press ENTER.

4. In the drop-down list to the right of **Message Type**, expand **Schemas**, and then select **Select from referenced assembly**.

5. In the **Select Artifact Type** dialog box, select **B2BSchemas**, from the **Type Name | Fully Qualified Name** list, select **B2BSchemas.PO**, and then click **OK**.

6. Repeat steps 1 through 4 to create two additional new messages with the following parameters:

Identifier	**Message Type**	**Referenced assembly**
CommonPO_Instance	Web Message Types	B2BSchemas.localhost.Process_.ReceivePO_oneway
POConfirmed_Instance	Schemas	B2BSchemas.PO

7. On the **File** menu, click **Save All** to save your work.

Step 6: Configure the Message Assignment Shape

Purpose: The expression you enter in this step creates the PO_Instance message from the POConfirmed_Instance message.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Create the Message Instance Variables before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure the message assignment shape

1. On the Orchestration Design surface, select the **Construct_POConfirmed** shape.
2. In the Properties pane, from the **Messages Constructed** drop-down list, select the **POConfirmed_Instance** check box, and then press ENTER.
3. In the **Construct_POConfirmed** shape, double-click the **MessageAssignment_1** shape to open BizTalk Expression Editor.
4. In BizTalk Expression Editor, type the following expression:
5. Click **OK** to close BizTalk Expression Editor.
6. On the **File** menu, click **Save All** to save your work.

Step 7: Configure the Transform Shape

Purpose: Because the MapToCommonPO map is specified in the **Transform** shape, the **Transform** shape uses the map to construct the CommonPO message.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 6: Configure the Message Assignment Shape before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure the Transform shape

1. On the Orchestration Design surface, select the **Construct_CommonPO** shape.
2. In the Properties pane, from the **Messages Constructed** drop-down list, select the **CommonPO_Instance** check box, and then press ENTER.
3. Double-click the **Transform_1** shape,
4. In the **Transform Configuration** dialog box, on the **Source** tab, select the **Existing Map** option, and then from the **Fully Qualified Map Name** drop-down list, select **Select from referenced assembly**.
5. In the **Select Artifact Type** dialog box, expand **References**, and then click **B2BSchemas**.
6. In **the Type Name | Fully Qualified Map Name** list, select **MaptoCommonPO**, and then click **OK**.
7. In the **Transform Configuration** dialog box, on the **Source** tab, from the **Source Transform** drop-down list, select **PO_Instance**.
8. In the **Transform Configuration** dialog box, on the **Destination** tab, from the **Destination Transform** drop-down list, select **CommonPO_Instance.InputPO.xml**.
9. Click **OK**.
10. On the **File** menu, click **Save All** to save your work.

Step 8: Create and Configure the Internal Ports

Purpose: The B2BProcess orchestration uses one of the ports you create in this step to receive internal purchase orders, and the other port to send out a purchase order confirmation message.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 7: Configure the Transform Shape before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create and configure two Internal ports

1. In Orchestration Designer, from the Toolbox, drag a **Port** shape to the left side Port Surface of the Orchestration Design surface parallel to the **Receive_PO** shape.
2. On the **Welcome to the Port Configuration Wizard** page, click **Next**.
3. On the **Port Properties** page, in the **Name** field, type **ReceivePO_Port**, and then click **Next**.
4. On the **Select a Port Type** page, do the following:

Use this	**To do this**
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **ReceivePO_Type**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Internal - limited to this project** option.

5. Click **Next**.
6. On the **Port Binding** page, do the following:

Use this	**To do this**
Port direction of communication	From the drop-down list, select **I'll always be receiving messages on this port**.

Port binding	From the drop-down list, select **Specify now**.
URI	Type **C:\Tutorial\Filedrop\ReceivePO\*.xml**.
Transport	From the drop-down list, select **FILE**.
Receive pipeline	Leave the default receive pipeline.

7. Click **Next**, and then click **Finish.**

8. Repeat steps 1 through 7 to configure the port that sends the PO confirmation. Use the following values:

Property	**Value**
Port Name	**SendPOConfirmed_Port**
Port Type Name	**SendPOConfirmed_Type**
Communication Pattern	**One-Way**
Access Restrictions	**Internal - Limited to this project**
Direction of communication	**I'll always be sending messages on this port**
Port Binding	**Specify now**
URI	**C:\Tutorial\Filedrop\FileArchive\Confirmed%MessageID%.xml**
Transport	**FILE**

9. On the **File** menu, click **Save All** to save your work.

Step 9: Create and Configure the External Port

Purpose: The port you add to the orchestration in this step archives the CommonPO. You will reconfigure this port later to send the CommonPO to your supplier's Web service. Because this port will eventually be calling a Web service, access restrictions on this port are set to public, meaning that access is not restricted.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 8: Create and Configure the Internal Ports before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To create and configure a temporary port to archive the PO

1. In Orchestration Designer, from the Toolbox, drag a **Port** shape to the right-side Port Surface, and drop the **Port** shape parallel to the **Send_CommonPO** shape.

2. On the **Welcome to the Port Configuration Wizard** page, click **Next**.

3. On the **Port Properties** page, in the **Name** field, type **SendCommonPO_Port**, and then click **Next**.

4. On the **Select a Port Type** page, do the following:

Use this	**To do this**
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **SendCommonPO_Type**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public - no limit** option.

5. Click **Next**.

6. On the **Port Binding** page, do the following:

Use this	**To do this**
Port direction of communication	From the drop-down list, select **I'll always be sending messages on this port**.
Port binding	From the drop-down list, select **Specify now**.
URI	Type **C:\Tutorial\Filedrop\FileArchive\PO%MessageID%.xml**.
Transport	From the drop-down list, select **FILE**.
Send pipeline	Leave the default send pipeline.

7. Click **Next**, and then click **Finish**.

8. On the **File** menu, click **Save All** to save your work.

Step 10: Connect the Ports

Purpose: You connect messages and action shapes to ports to specify how the process sends and receives the messages.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 9: Create and Configure the External Port before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To select the messages and connect the ports

1. In Orchestration Designer, on the Orchestration Design surface, select the **Receive_PO** action shape.

2. In the Properties pane, from the **Message** drop-down list, select **PO_Instance**.

3. Connect the **ReceivePO_Port** to the **Receive_PO** action shape by selecting the green arrow-shaped handle in the **ReceivePO_Port** and dragging it to the green handle on the **Receive_PO** action shape.

4. Repeat steps 1 through 3 to select messages in a shape, and connect the send ports to the send shapes. Use the following values:

Shape	Message property	Port
Send_POConfirmed	**POConfirmed_Instance**	**SendPOConfirmed_Port**
Send_CommonPO	**CommonPO_Instance**	**SendCommonPO_Port**

5. On the **File** menu, click **Save All** to save your work.

Lesson 5: Build and Deploy the Projects

In this lesson, you build and deploy the B2BSchemas and B2BOrchestrations projects. You bind, enlist, and start the B2BProcess orchestration, and you test the B2B solution.

In This Section

- Step 1: Build and Deploy the Projects
- Step 2: Bind, Enlist, and Start the Orchestration
- Step 3: Test the Solution

Step 1: Build and Deploy the Projects

Purpose: You build an assembly (project), and then deploy it to place a copy of the assembly in the BizTalk Management database (also known as the Configuration database) and install it in the global assembly cache.

In this step, you deploy both projects (also known as assemblies) in your solution: B2BSchemas.dll and B2BOrchestrations.dll.

You can run the deploy process on the solution. Deploying the solution deploys both projects in the solution, in the proper order. If you deploy the projects individually, you must deploy them in reverse order of dependency, starting from the independent assemblies and then moving on to the assemblies that are dependent upon those developed earlier.

In **Step 2: Add a Reference to the B2BSchemas Project** This is because in the B2BSolution solution, the B2BOrchestrations project references schemas in the B2BSchemas project.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 4: Design the Purchase Order Process before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To build and deploy the projects

1. In Solution Explorer, right-click **B2BSolution**, and then click **Build Solution**.
2. In Solution Explorer, right-click **B2BSolution**, and then click **Deploy Solution**.

Step 2: Bind, Enlist, and Start the Orchestration

Purpose: You enlist and start the send ports so that they are available to the orchestration. You bind the orchestration by selecting a host in which the orchestration process will run. You enlist the orchestration so that the necessary subscriptions are created in the MessageBox database and the configuration of the associated receive location is verified. You start the orchestration so that messages can be processed.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Build and Deploy the Projects before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To enlist and start the send ports

1. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **BizTalk Server Administration**.
2. In the BizTalk Administration console, expand **BizTalk Server 2006 Administration**, expand **BizTalk Group**, expand **Applications**, expand **BizTalk Application 1**, and then click **Send Ports**.
3. In the **Send Ports** results pane, right-click the **B2BOrchestrations_1.0.0.0_B2BOrchestrations.B2BProcess_SendPOConfirmed_Port_**<PublicKeyToken> send port, and then click **Enlist**.
4. Right-click the **B2BOrchestrations_1.0.0.0_B2BOrchestrations.B2BProcess_SendPOConfirmed_Port_**<PublicKeyToken> send port, and then click **Start**.
5. Right-click the **B2BOrchestrations_1.0.0.0_B2BOrchestrations.B2BProcess_SendCommonPO_Port_**<PublicKeyToken> send port, and then click **Enlist**.
6. Right-click the **B2BOrchestrations_1.0.0.0_B2BOrchestrations.B2BProcess_SendCommonPO_Port_**<PublicKeyToken> send port, and then click **Start**.

To bind, enlist, and start the orchestration

1. In the BizTalk Administration console, expand **BizTalk Server 2006 Administration**, expand **BizTalk Group**, expand **Applications**, expand **BizTalk Application 1**, and then click **Orchestrations.**
2. In the **Orchestrations** results pane, right-click **B2BOrchestrations.B2Bprocess**, and then click **Properties**.
3. In the **Orchestration Properties** dialog box, on the **Bindings** tab, from the **Host** drop-down list, select **BizTalkServerApplication**, and then click **OK**.
4. In the **Orchestrations** results pane, right-click **B2BOrchestrations.B2Bprocess**, and then click **Enlist**.

5. In the **Orchestrations** results pane, right-click **B2BOrchestrations.B2Bprocess**, and then click **Start**.

6. Close the BizTalk Administration console.

Step 3: Test the Solution

Purpose: You test the B2B solution by simulating the process of receiving a purchase order (PO) from the Enterprise Resource Planning (ERP) system. You simulate the process by dropping an instance of a PO into a file location. The B2BProcess orchestration constructs a POConfirmed message and routes it to the warehouse. At the same time, the orchestration constructs and archives a CommonPO message.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Bind, Enlist, and Start the Orchestration before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To verify that this portion of the scenario runs

1. In Windows Explorer, browse to **C:\Tutorial**, and copy **POInstance.xml** to **C:\Tutorial\Filedrop\ReceivePO**.

2. The purchase orders should be routed to C:\Filedrop\FileArchive.

3. Browse to **C:\Tutorial\Filedrop\FileArchive** to verify that the purchase order was routed correctly.

   You should have a PO{GUID}.xml message and a Confirmed{GUID}.xml message in the C:\Filedrop\FileArchive directory.

4. Open the .xml files to verify the data contained in each.

Tutorial 3: Invoice and Payment Process

In Tutorial 2: Purchase Order Process, you created a business-to-business (B2B) solution by importing schemas and maps, and by designing an orchestration that processes purchase orders.

In this tutorial, you extend the orchestration to process invoices and payments. After you extend the orchestration, you incorporate correlation, connect the solution to the supplier's Web service, and add business rules and incorporate them with the orchestration.

The scenario in this tutorial continues from the end of Tutorial 2 as follows:

1. BizTalk Server applies business rules to ensure that the invoice matches the products received.

2. BizTalk Server transforms the invoice to a payment voucher.

3. BizTalk Server sends the payment voucher to the supplier.

4. The supplier sends a payment acknowledgment.

5. BizTalk Server sends the payment acknowledgment to the Enterprise Resource Planning (ERP) system.

BizTalk Server 2006 ensures that the ERP system and the supplier interact smoothly and efficiently. This includes applying business rules to the payment approval process and routing the payment appropriately.

In This Section

- Lesson 1: Design the Invoice and Payment Process
- Lesson 2: Incorporate Correlation
- Lesson 3: Connect the B2B Solution to the Supplier Web Service
- Lesson 4: Create a Payment Policy
- Lesson 5: Integrate the Payment Policy with the Orchestration

Lesson 1: Design the Invoice and Payment Process

In this lesson, you extend the B2BProcess orchestration to include an invoice and payment process.

In This Section

- Step 1: Add the Invoice and Payment Shapes to the Orchestration
- Step 2: Create the Invoice and Payment Message Instance Variables
- Step 3: Construct the Payment Voucher Message
- Step 4: Create, Configure, and Connect the Ports

Step 1: Add the Invoice and Payment Shapes to the Orchestration

Purpose: In this step, you extend the B2BProcess orchestration to include an invoice and payment process by adding additional shapes to route the shipping notice, invoice, and payment messages that are necessary to complete the procurement process.

Prerequisites

Note the following requirements before you begin this step:

- You must complete the steps in Before You Begin the Tutorials before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add the action shapes for sending and receiving invoices and payments

1. In Solution Explorer, expand **B2BOrchestrations**, and then double-click **B2BProcess.odx** to open it in Orchestration Designer (if it is not already open).
2. From the Toolbox, drag the **Receive** shape to the orchestration design surface, drop the shape on the connecting line directly below the horizontal line that denotes the bottom of the **Route_PO** Parallel shape, and then change the **Name** property to **Receive_ASN**.
3. From the Toolbox, drag another **Receive** shape to the orchestration design surface, drop the shape on the connecting line directly below the **Receive_ASN** shape, and then change the **Name** property to **Receive_Invoice**.
4. From the Toolbox, drag the **Transform** shape to the orchestration design surface, and then drop the shape on the connecting line directly below the **Receive_Invoice** shape.
5. Select the **Construct Message** shape, and change the **Name** property to **Construct_PaymentVoucher**.
6. From the Toolbox, drag the **Send** shape to the orchestration design surface, drop the shape on the connecting line directly below the **Construct_PaymentVoucher** shape, and then change the **Name** property to **Send_Payment**.
7. From the Toolbox, drag the **Receive** shape to the orchestration design surface, drop the shape on the connecting line directly below the **Send_Payment** shape, and then change the **Name** property to **Receive_PaymentAck**.
8. From the Toolbox, drag the **Send** shape to the orchestration design surface, drop the shape on the connecting line directly below the **Receive_PaymentAck** shape, and then change the **Name** property to **Send_PaymentAck**.
9. On the **File** menu, click **Save All**.

Step 2: Create the Invoice and Payment Message Instance Variables

Purpose: Because Orchestration Designer deals with message instances for managing data, rather than directly with the schemas, in this procedure you create variables for the message instances. Note that some of the message types point to schemas in your supplier's Web service, while others point to schemas in the B2BSchemas project.

Prerequisites

Note the following requirements before you begin this step:

- You must complete the steps in Step 1: Add the Invoice and Payment Shapes to the Orchestration before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the message instance variables

1. On the **Orchestration View** tab, right-click **Messages**, and then click **New Message**.

   By default, the new message is named Message_1.

2. In the Properties pane, do the following:

Use this	To do this
Identifier	Type **ASN_Instance**, and then press ENTER.
Message Type	From the drop-down list, expand **Schemas**, and then select **Select from referenced assembly**.

3. In the **Select Artifact Type** dialog box, expand **References**, and then select **B2BSchemas**.
4. From the **Type Name | Fully Qualified Name** list, select **AdvancedShipNotice**, and then click **OK**.
5. Repeat steps 1 through 4 to create the following new messages.

Message identifier	Message type	Referenced assembly
CommonInvoice_Instance	Schemas	B2BSchemas.CommonInvoice
PaymentVoucher_Instance	Web Message	B2BSchemas.ProcessPayment_request
PaymentAck_Instance	Web Message	B2BSchemas.ProcessPayment_response

6. On the **File** menu, click **Save All**.

Step 3: Construct the Payment Voucher Message

Purpose: You associate a map with a **Transform** shape to transform the invoice message received from the supplier into the PaymentVoucher message you send to the supplier.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Create the Invoice and Payment Message Instance Variables before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To construct the PaymentVoucher message using a map

1. On the orchestration design surface, select the **Construct_PaymentVoucher** shape.
2. In the Properties pane, from the **Messages Constructed** drop-down list, select the **PaymentVoucher_Instance** check box, and then press ENTER.
3. On the orchestration design surface, double-click the **Transform_2** shape.
4. In the **Transform Configuration** dialog box, on the **Source** tab, do the following:

Use this	**To do this**
Enter the configuration information	Select the **Existing Map** option.
Fully Qualified Map Name	From the drop-down list, select **Select from Referenced Assembly**.

5. In the **Select Artifact Type** dialog box, on the **References\B2BSchemas** tab, in the **Type Name** list, click **InvoiceToPayment**, and then click **OK**.
6. In the **Transform Configuration** dialog box, on the **Source** tab, from the **Variable Name** drop-down list, select **CommonInvoice_Instance**.
7. In the **Transform Configuration** dialog box, on the **Destination** tab, from the **Variable Name** drop-down list, select **PaymentVoucher_Instance.Pay_xml**.
8. Click **OK**.
9. On the **File** menu, click **Save All**.

Step 4: Create, Configure, and Connect the Ports

Purpose: The following table describes the ports you create in this step.

Port	Purpose
ReceiveASN_Port	This port receives PO-received confirmation messages from the supplier.
ReceiveInvoice_Port	This port receives invoice messages from the supplier.
RR_Payment_Port	This is a request-response port that sends a payment voucher to the supplier and receives a confirmation response from the supplier.
SendPaymentAck_Port	This is an internal port that forwards (sends) payment acknowledgment within your company.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Construct the Payment Voucher Message before you begin this step.
- You must login as a member of the BizTalk Server Administrators group.

To create and configure the ports

1. From the Toolbox, drag the **Port** shape to the right-side port surface, and drop the shape parallel to the **Receive_ASN** shape.
2. On the **Welcome to the Port Configuration Wizard** page, click **Next**.
3. On the **Port Properties** page, in the **Name** box, type **ReceiveASN_Port**, and then click **Next**.
4. On the **Select a Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Create a new Port** Type option.
Port Type Name	Type **ReceiveASN_Type**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public - no limit** option.

5. Click **Next**.
6. On the **Port Binding** page, do the following:

Use this	To do this
Port direction of communication	From the drop-down list, select **I'll always be receiving messages on this port**.
Port binding	From the drop-down list, select **Specify later**.

7. Click **Next**, and then click **Finish.**

8. Use the information in the following table to add and configure a receive port called ReceiveInvoice_Port.

Property	**Value**
Port Name	ReceiveInvoice_Port
Port Type Name	ReceiveInvoice_Type
Communication Pattern	One-Way
Access Restrictions	Public - no limit
Port direction of communication	I'll always be receiving messages on this port
Port binding	Specify later

10. Use the information in the following table to add and configure a send port called RR_Payment_Port _Port.

Property	**Value**
Port Name	RR_Payment_Port
Port Type Name	RR_Payment_Type
Communication Pattern	Request-Response
Access Restrictions	Public - no limit
Port direction of communication	I'll be sending a request and receiving a response.
Port binding	Specify later

12. Use the information in the following table to add and configure a send port called SendPaymentAck_Port.

Property	Value
Port Name	SendPaymentAck_Port
Port Type Name	SendPaymentAck_Type
Communication Pattern	One-Way
Access Restrictions	Internal - limited to this project
Port direction of communication	I'll always be sending messages on this port
Port binding	Specify later

14. On the **File** menu, click **Save All**.

    Next, you must connect the ports.

To select the messages in the Action shapes and connect the ports

1. In Orchestration Designer, on the orchestration design surface, select the **Receive_ASN** action shape.

2. In the Properties pane, from the **Message** drop-down list, select **ASN_Instance**.

3. Connect the **ReceiveASN_Port** to the **Receive_ASN** action shape by selecting the green arrow-shaped handle in the **ReceiveASN_Port** and dragging it to the green handle on the **Receive_ASN** action shape.

4. Repeat steps 1 and 2 to select messages for the action shapes.

Shape	Message
Receive_Invoice	**CommonInvoice_Instance**
Send_Payment	**PaymentVoucher_Instance**
Receive_PaymentAck	**PaymentAck_Instance**
SendPaymentAck	**PaymentAck_Instance**

5. Repeat step 3 to connect the send ports to the **Send** shapes.

Send ports	**Send shapes**
ReceiveInvoice_Port	**Receive_Invoice**
RR_Payment_Port **->Request**	**Send_Payment**
RR_Payment_Port **->Response**	**Receive_PaymentAck**
SendPaymentAck_Port	**SendPaymentAck**

6. On the **File** menu, click **Save All**.

Lesson 2: Incorporate Correlation

In this lesson, you incorporate correlation into the B2BProcess orchestration. Correlation is the mechanism by which messages are associated with particular running instances of an orchestration, so that your business process gets the appropriate information when many instances are running and many messages are being sent back and forth.

In This Section

- Step 1: Create the Correlation Property Schema
- Step 2: Promote the Properties for the Correlation
- Step 3: Create the Correlation Types
- Step 4: Create the Correlation Set
- Step 5: Initialize the Correlation Set
- Step 6: Set the Correlation Properties

Step 1: Create the Correlation Property Schema

Purpose: the property schema contains only one element to check. The property schema identifies the PO number field element as the field that must be accessible in the business process for correlation. Using the PO number field for correlation enables the process to route messages containing the PO number appropriately.

Property schemas contain only a flat list of fields (with no hierarchy) under the <Schema> node. For this reason, there is no root node. The XSD schema compiler gives an error if it finds a violation of this condition.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 1: Design the Invoice and Payment Process before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the correlation property schema

1. In Solution Explorer, right-click **B2BSchemas**, point to **Add**, and then click **New Item**.
2. In the **Add New Item** dialog box, do the following:
3.

Use this	To do this
Categories	Click **Schema Files**.
Templates	Click **Property Schema**.
Name	Type **CorProperties.xsd**.

4. Click **Add**.
5. In the schema tree, click **Property 1**.
6. In the Properties pane, change the value of the **Node Name** property to **PO_Num**, and then press ENTER.
7. In the schema tree, click **<Schema>**, and then in the Properties pane, scroll down and make sure the **Schema Type** property is set to **Property**.
8. On the **File** menu, click **Save All**.

Step 2: Promote the Properties for the Correlation

Purpose: You use the CorProperties schema inside the document schema to promote the PO_Num property. Notice that you set correlation properties on a schema in the supplier's Web service as well as on schemas from the B2BSchemas project.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Create the Correlation Property Schema before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To promote the correlation properties in the schemas

1. In Solution Explorer, expand **B2BSchemas**, expand **Web References**, expand **localhost**, expand **Reference.map**, and then double-click **Reference.xsd** to open the schema.
2. In the schema tree, click <**Schema**>, and then in the Properties pane, click the **Promote Properties** ellipsis [**...**].
3. In the **Promote Properties** dialog box, on the **Property Fields** tab, click the **Folder** icon.
4. In the **BizTalk Type Picker** dialog box, expand **B2BSchemas**, expand **Schemas**, click **B2BSchemas.CorProperties**, and then click **OK**.
5. In the **Promote Properties** dialog box, do the following:

Use this	**To do this**
In the schema tree	Expand **CommonPO**, expand **POHeader**, click **Number**, and then click **Add**.
On the **Property Fields** tab	From the **Property fields list** drop-down list (in the grid under the **Property** heading), select **ns0:PO_Num**.

6. Click **OK**.
7. In Solution Explorer, expand **B2BSchemas**, and then double-click CommonInvoice.xsd to open the schema.
8. In the schema tree, click <**Schema**>, and then in the Properties pane, click the **Promote Properties** ellipsis [**...**].
9. In the **Promote Properties** dialog box, on the **Property Fields** tab, click the **Folder** icon.
10. In the **BizTalk Type Picker** dialog box, expand **B2BSchemas**, expand **Schemas**, click **B2BSchemas.CorProperties**, and then click **OK**.
11. In the **Promote Properties** dialog box, do the following:

Use this	**To do this**
In the schema tree	Expand InvoiceHeader expand PONumber, click PO_Num, and then click **Add**.
On the **Property**	From the **Property fields list** drop-down list (in the grid under the **Property**

Fields tab	heading), select **ns0:PO_Num**.

12. Click **OK**.

13. In Solution Explorer, expand **B2BSchemas**, and then double-click AdvancedShipNotice.xsd to open the schema.

14. In the schema tree, click <**Schema**>, and then in the Properties pane, click the **Promote Properties** ellipsis [**...**].

15. In the **Promote Properties** dialog box, on the **Property Fields** tab, click the **Folder** icon.

16. In the **BizTalk Type Picker** dialog box, expand **B2BSchemas**, expand **Schemas**, click **B2BSchemas.CorProperties**, and then click **OK**.

17. In the **Promote Properties** dialog box, do the following:

Use this	**To do this**
In the schema tree	Expand Shipment expand Order, expand PONumber, click PO_Num, and then click **Add**.
On the **Property Fields** tab	From the **Property fields list** drop-down list (in the grid under the **Property** heading), select **ns0:PO_Num**.

18. Click **OK**.

19. In Solution Explorer, right-click **B2BSchemas**, and then click **Rebuild**.

20. On the **File** menu, click **Save All**.

Step 3: Create the Correlation Types

Purpose: A correlation type is a list of message properties. After creating the correlation types, you create a correlation set, which you use to associate values with those properties.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Promote the Properties for the Correlation before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To create the correlation types

1. In Solution Explorer, expand **B2BOrchestrations**, and then double-click **B2BProcess.odx** to open it in Orchestration Designer.
2. In the Orchestration View window, expand **Types**, right-click **Correlation Types**, and then click **New Correlation Type.**
3. In the **Correlation Properties** dialog box, expand **B2BSchemas**, click **PO_Num**, click **Add**, and then click **OK**.
4. In the Properties pane, change the value of the **Identifier** property to **Correl_PO**.
5. On the **File** menu, click **Save All**.

Step 4: Create the Correlation Set

Purpose: Correlation sets are lists of message properties and associated values that you apply when sending and receiving messages in your orchestration, to guarantee that incoming messages will be associated with the correct running instance of your orchestration.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Create the Correlation Types before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the correlation set

1. In the Orchestration View window, right-click **Correlation Sets**, and then click **New Correlation Set**.
2. In the Properties pane, do the following:

Use this	To do this
Identifier	Type **PO_CorrelSet**.
Correlation Type	From the drop-down list, select **B2BOrchestrations.Correl_PO**.

3. On the **File** menu, click **Save All**.

Step 5: Initialize the Correlation Set

Purpose: You initialize the correlation set values in the **Send** shape so that subsequent incoming messages will correlate to the PO number. If you expect to receive either a direct or indirect response to a message that you have previously sent, you must correlate the message with the currently running instance of the orchestration. For more information, see Correlating Messages with Orchestration Instances .

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Create the Correlation Set before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To initialize the correlation set

1. In Orchestration Designer, on the orchestration design surface, select the **Send_CommonPO** shape.
2. In the Properties pane, from the **Initializing Correlation Sets** drop-down list, select the **PO_CorrelSet** check box, and then press ENTER.
3. On the **File** menu, click **Save All**.

Step 6: Set the Correlation Properties

Purpose: Content to be provided.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Initialize the Correlation Set before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To set the correlation properties

1. In Orchestration Designer, on the orchestration design surface, select the **Receive_ASN** shape.
2. In the Properties pane, from the **Following Correlation Sets** drop-down list, select the **PO_CorrelSet** check box.
3. In Orchestration Designer, on the orchestration design surface, select the **Receive_Invoice** shape.

4. In the Properties pane, from the **Following Correlation Sets** drop-down list, select the **PO_CorrelSet** check box.

5. On the **File** menu, click **Save All**.

6. In Solution Explorer, right-click **B2BOrchestrations**, and then click **Rebuild**.

Lesson 3: Connect the B2B Solution to the Supplier Web Service

In this lesson, you connect the B2B solution to the supplier's Web service.

In This Section

- Step 1: Configure the Ports that Call the Web Service
- Step 2: Undeploy the Solution
- Step 3: Redeploy the Solution
- Step 4: Publish the Orchestration as a Web Service
- Step 5: Update the Supplier Web Service References
- Step 6: Configure the Receive Ports
- Step 7: Configure the Send Ports
- Step 8: Bind the Orchestration Ports
- Step 9: Enlist and Start the Orchestration
- Step 10: Test the Solution

Step 1: Configure the Ports that Call the Web Service

Purpose: To consume a Web service in an orchestration, you must add a Web reference, a port, and message variables, and construct Web message instances.

When you add a new port to the orchestration, the Web reference is listed as an existing Web port type. In this step, you use the Port Configuration Wizard to associate the Web reference with the ports and messages.

To export an orchestration as a Web service, you set the access modifier on the port type of the port to be exposed to Public. When you create a port type, the default access modifier is set to Internal. You must make this change in the Port Type property pages.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 2: Incorporate Correlation before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure a port that sends the PO to the supplier's Web service

1. In Solution Explorer, double-click **B2BProcess.odx**.
2. On the orchestration design surface, right-click **SendCommonPO_Port**, and then click **Configure Port**.
3. On the **Welcome to the Port Configuration Wizard** page, click **Next**.
4. On the **Port Properties** page, click **Next**.
5. On the **Select a Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Use an existing Port Type** option.
Available Port Types	In the list, e **C:\Tutorial\Lessons\B2BSolution\B2BOrchestrations\Bin\Development\B2BSchemas** expand **Web Port Types**, and then click **B2BSchemas.localhost.Process_.Process**.

6. Click **Next**.
7. On the **Port Binding** page, click **Next**.
8. Click **Finish** to exit the Port Configuration Wizard.
9. Connect the **SendCommonPO_Port** port to the **Send_CommonPO** action shape.

To configure the Payment port to interact with the supplier's Web service

1. On the orchestration design surface, right-click **RR_Payment_Port**, and then click **Configure Port**.

2. On the **Welcome to the Port Configuration Wizard** page, click **Next**.

3. On the **Port Properties** page, click **Next**.

4. On the **Select a Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Use an existing Port Type** option.
Available Port Types	In the list, expand **C:\Tutorial\Lessons\B2BSolution\B2BOrchestrations\Bin\Development\B2BSchemas** expand **Web Port Types**, and then click **B2BSchemas.localhost1.Payment_Service_.Payment_Service**.

5. Click **Next**.

6. On the **Port Binding** page, from the **Port Binding** drop-down list, select **Specify now**.

7. Click **Next**, and then click **Finish**.

8. Reconnect **RR_Payment_Port (Request)** to the **Send_Payment** shape, and then reconnect **RR_Payment_Port (Response)** to the **Receive_PaymentAck** shape.

9. On the **File** menu, click **Save All**.

Step 2: Undeploy the Solution

Purpose: You stop and unenlist the B2BProcess orchestration to remove the orchestration binding. You undeploy both projects so that you can redeploy them.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Configure the Ports that Call the Web Service before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To undeploy the solution

1. In Visual Studio, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the <**ServerName**>.**BizTalkMgmtDb.dbo**, expand **Orchestrations**, right-click **B2BOrchestrations.B2BProcess**, and then click **Stop**.
3. Right-click **B2BOrchestrations.B2BProcess** again, and then click **Unenlist**.
4. In the **BizTalk Explorer - Express Unenlist** dialog box, select the **Terminate existing Orchestration Instances** check box, and then click **OK**.
5. In BizTalk Explorer, expand **Assemblies**, right-click **B2BOrchestrations**, and then click **Undeploy**.
6. In the confirmation message box, click **Yes**.
7. In BizTalk Explorer, expand **Assemblies**, right-click **B2BSchemas**, and then click **Undeploy**.
8. In the confirmation message box, click **Yes**.

Step 3: Redeploy the Solution

Purpose: You deploy the assemblies (projects) to place a copy of each of the assemblies in the BizTalk Management database (also known as the Configuration database) and install each assembly in the global assembly cache.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Undeploy the Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To redeploy the updated solution

1. In BizTalk Explorer, right-click **<ServerName>.BizTalkMgmtDb.dbo**, and click **Refresh**.
2. Expand **Assemblies** and verify that B2BOrchestrations and B2BSchemas are no longer listed as deployed assemblies.
3. In Solution Explorer, right-click **B2BSchemas**, and then click **Deploy**.
4. In Solution Explorer, right-click **B2BOrchestrations**, and then click **Deploy**.

Step 4: Publish the Orchestration as a Web Service

Purpose: You publish (expose) your orchestration as a Web service to separate the Web service logic from the business process logic.

To publish an orchestration as a Web service, you first have to design and build the orchestration and generate an assembly.

Then, you use the BizTalk Web Service Publishing Wizard to create a Web service based on an orchestration in a BizTalk Server assembly. Using the wizard, you can select orchestrations and receive ports to expose as Web services. The wizard enables you to define target namespaces, SOAP header requirements, and the locations for the wizard's generated Web service project.

The wizard creates one Web service (.asmx file) for each receive port. Each operation in the receive port becomes a Web method. The operation message types define the Web method signatures. Each message type part is a parameter in the Web method.

For more information about how ASP.NET Web services work, see http://go.microsoft.com/fwlink/?LinkId=25200.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Redeploy the Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To publish the B2BOrchestration as a Web service

1. In Visual Studio, on the **Tools** menu, click **BizTalk Web Services Publishing Wizard.** The BizTalk Web Services Publishing Wizard appears.
2. On the **Welcome to the BizTalk Web Services Publishing Wizard** page, click **Next**.
3. On the **Create Web Service** page, verify that the **Publish BizTalk orchestrations as Web services** option is selected, and then click **Next**.
4. On the **BizTalk Assembly** page, browse to **C:\Tutorial\Lessons\B2BSolution\B2BOrchestration\bin\Development**, select **B2BOrchestrations.dll**, click **Open**, and then click **Next**.
5. On the **Orchestrations and Ports** page, verify that **ReceiveASN_Port** and **ReceiveInvoice_Port** are selected, and click **Next**.
6. On the **Web Service Properties** page, in the **Target namespace of Web services** box, type **http://B2BOrchestrations_WebService.org**, and then click **Next**.

7. On the **Web Service Project** page, do the following:

Use this	To do this
Location (http://host(:port)/path)	Type **B2BOrchestrations_Proxy**.
Allow anonymous access to this Web service	Select this check box to enable anonymous access.

8. Click **Next**.

9. On the **Web Service Project Summary** page, click **Create**.

10. On the **Completing the BizTalk Web Services Publishing Wizard** page, click **Finish**.

11. On the **File** menu, click **Save All**.

Step 5: Update the Supplier Web Service References

Purpose: You update the supplier Web service references, to make sure that the published orchestration and the supplier Web service have the correct references.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Publish the Orchestration as a Web Service before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To update the supplier Web service Web references

1. In Windows Explorer, browse to **C:\Tutorial\Solutions\SupplierWebServices\**.
2. Double-click **B2BSupplierWebService.sln** to open it in a new instance of Visual Studio 2005.
3. In Solution Explorer, expand **B2BSupplierProcessPO**, and then expand **App_Web References**.
4. Right-click **localhost**, and then click **Update Web References**.
5. Right-click **localhost1**, and then click **Update Web References**.
6. Right-click **Solution 'B2BSupplierWebService'**, and then click **Rebuild Solution**.
7. When the build succeeds, close the solution and the instance of Visual Studio.

8. Close Windows Explorer.

Step 6: Configure the Receive Ports

Purpose: Use BizTalk Explorer to create the receive ports and receive locations.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Update the Supplier Web Service References before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure the receive ports

1. In Visual Studio, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand **BizTalkMgmtDb.dbo**, right-click **Receive Ports**, and then select **Add Receive Port**.
3. Specify the type of port as a **One-Way Port**, and then click **OK**.
4. In the **Receive Port General Configuration** dialog box, on the **General** tab, in the **Name** field, type **ASN_ReceivePort**, and then click **OK**.
5. In BizTalk Explorer, right-click **Receive Locations**, and then click **Add Receive Location**.
6. In the **Receive Location Properties** dialog box, on the **General** tab, do the following:

Use this	To do this
Name	type **ASN Receive Location**
Transport Type	From the drop-down list, select **SOAP**.
Address URI	Click the ellipses (**...**).

7. In the **SOAP Transport Properties** dialog box, in the **Virtual directory plus Web Service asmx file** box type **/B2BOrchestrations_WebService/B2BOrchestrations_B2BProcess_ReceiveASN_Port.asmx**, and then click **OK**.
8. In the **Receive Location Properties** dialog box, on the **General** tab, do the following:

Use this	To do this
Receive Handler	From the drop-down list, select **BizTalkServerIsolatedHost**.
Receive Pipeline	From the drop-down list, select **Microsoft.BizTalk.DefaultPipelines.XMLReceive (Microsoft.BizTalk.DefaultPipelines.XMLReceive, Microsoft.BizTalk.DefaultPipelines, Version=3.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35)**.

9. Click **OK**.

10. Repeat steps 2 through 9 to create a second receive port and receive location. Use the values in the following table.

Property	Value
Receive Port Type	One-Way
Receive Port Name	Invoice_ReceivePort
Receive Location Name	Invoice Receive Location
Transport Type	SOAP
URI	/B2BOrchestrations_WebService/B2BOrchestrations_B2BProcess_ReceiveInvoice_Port.asmx
Receive Handler	BizTalkServerIsolatedHost
Receive Pipeline	Microsoft.BizTalk.DefaultPipelines.XMLReceive (Microsoft.BizTalk.DefaultPipelines.XMLReceive, Microsoft.BizTalk.DefaultPipelines, Version=3.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35)

11. On the **File** menu, click **Save All**.

Step 7: Configure the Send Ports

Purpose: Define a static one-way send port, the destination address, the Uniform Resource Identifier (URI), and the pipelines that are used to send payment acknowledgments to the ERP system.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 6: Configure the Receive Ports before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure the send ports

1. In BizTalk Explorer, right-click **Send Ports,** and then click **Add Send Port**.
2. Specify the type of send port as a **Static One-Way Port**, and then click **OK**.
3. In the **Static One-Way Send Port Properties** dialog box, on the **Transport\Primary** tab, do the following:

Use this	To do this
Name	Type **SendPort_PaymentAck**.
Transport Type	From the drop-down list, select **File**.
Address URI	Click the ellipsis [**...**]

4. In the **FILE Transport Properties** dialog box, do the following:

Use this	To do this
Destination folder	Type **C:\Tutorial\FileDrop\ERPSys**.
File Name	Type **PayAck%MessageID%.xml**.

5. Click **OK**.
6. In the **Static One-Way Send Port Properties** dialog box, on the **Send** tab, from the **Send Pipeline** drop-down list, select **Microsoft.BizTalk.DefaultPipelines.XMLTransmit (Microsoft.BizTalk.DefaultPipelines.XMLTransmit, Microsoft.BizTalk.DefaultPipelines, Version=3.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35).**
7. Click **OK**.

8. On the **File** menu, click **Save All**.

Step 8: Bind the Orchestration Ports

Purpose: Binding the orchestration associates the logical ports on the orchestration design surface with physical ports created in BizTalk Explorer. Associating the logical ports with physical ports makes the URI and adapter configurations available to the business process during run time.

Use BizTalk Explorer to bind an orchestration to a receive port or send location.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 7: Configure the Send Ports before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To bind the orchestration ports

1. In BizTalk Explorer, right-click **BizTalk Configuration Databases**, and then click **Refresh**.
2. Expand **Orchestrations**, right-click **B2BOrchestrations.B2BProcess**, and then click **Bind**.
3. In the **Port Binding Properties** dialog box, on the **Binding** tab, do the following:

Use this	To do this
ReceiveASN_Port	From the drop-down list, select **ASN_ReceivePort**.
ReceiveInvoice_Port	From the drop-down list, select **Invoice_ReceivePort**.
SendPaymentAck_Port	In the drop-down list, expand **Send Ports**, and then select **SendPort_PaymentAck**.

4. In the **Port Binding Properties** dialog box, on the **Host** tab, from the **Host** drop-down list, select **BizTalkServerApplication**.
5. Click **OK**.
6. On the **File** menu, click **Save All**.

Step 9: Enlist and Start the Orchestration

Purpose: You use BizTalk Explorer to enlist and start the orchestration. When you enlist the orchestration, you are associating the business process that you designed in Orchestration Designer with the physical environment in which it will run, and creating the core subscriptions in the MessageBox database. When you start the service, it will begin processing.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 8: Bind the Orchestration Ports before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To enlist, enable, and start the services

1. In BizTalk Explorer, expand **Orchestrations**, right-click **B2BOrchestrations.B2BProcess**, and then click **Enlist**.
2. In BizTalk Explorer, right-click **B2BOrchestrations.B2BProcess**, and then click **Start**.
3. Verify that all the **Dependency Options** are selected, and then click **OK**.
4. On the **File** menu, click **Save All**.

Step 10: Test the Solution

Purpose:

Simulate the process of receiving a purchase order (PO) from the Enterprise Resource Planning (ERP) system by dropping an instance of a PO into a file location. The PO will be transformed and routed. A CommonPO will be sent to the supplier's Web service. An advanced shipping notice (ASN) and invoice will be returned from the supplier and a payment will be sent back. When the transaction is complete, you should have a POConfirmed message at the warehouse and a payment acknowledgment sent to the ERP system.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 9: Enlist and Start the Orchestration before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To test the solution

1. In Windows Explorer, browse to **C:\Tutorial**, and copy **POInstance.xml** to **C:\Tutorial\Filedrop\ReceivePO**.

2. Browse to **C:\Tutorial\Filedrop\FileArchive** to verify that POConfirmed was archived.

3. Browse to **C:\Tutorial\Filedrop\ERPSys** to verify that a payment acknowledgment was sent to the ERP system and that the B2B solution completed.

4. Close Windows Explorer.

Lesson 4: Create a Payment Policy

In this lesson, you add business rules to the orchestration.

In This Section

- Step 1: Add a Vocabulary
- Step 2: Define Get Elements for the Vocabulary
- Step 3: Define Set Elements for the Vocabulary
- Step 4: Publish the Vocabulary
- Step 5: Add a Business Policy
- Step 6: Add Rules to the Policy

Step 1: Add a Vocabulary

Purpose: Vocabularies are used to bind data sources to more user-friendly language expressions. You do not have to use vocabularies when you create business rules, because you can use data elements directly in rule expressions. However, vocabularies enable you to create rules that are easier to understand.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 3: Connect the B2B Solution to the Supplier Web Service before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add the new vocabulary

1. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **Business Rule Composer**.

2. In the Business Rule Composer, in the Facts Explorer, on the **Vocabularies** tab, right-click the **Vocabularies** node, and then click **Add New Vocabulary.**

3. Change the **Name** to **B2BVocabulary**, and then press ENTER.

Step 2: Define Get Elements for the Vocabulary

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Add a Vocabulary before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To define an XML document Get element

1. In Facts Explorer, expand the B2BVocabulary folder, right-click Version 1.0(not saved), and then select Add New Definition.

2. On the Welcome to the Vocabulary Definition Wizard page, select the XML Document Element or Attribute option, and then click Next.

3. On the XML Document Element or Attribute page, in the Definition Name box, type PO Quantity, and then click Browse.

4. In the Schema Files dialog box, navigate to C:\Tutorial\Lessons\B2BSolution\B2BSchemas\ and open PO.xsd.

5. In the Select Binding dialog box, expand PO, expand Item, select Quantity, and then click OK.

6. In the Vocabulary Definition Wizard, do the following:

Use this	**To do this**
Document Type	Type B2BSchemas.PO.
Select operation	Select the Perform "Get" operation option.
Display Name	Type PO Quantity.

7. Click Finish.

8. Repeat steps 1 through 7 to define additional XML Document Get Elements with the following values.

Property	Value
Definition Name	Quantity Shipped
Schema File	AdvancedShipNotice.xsd
Select Binding	CommonAdvancedShipNotice/HashTotal/ItemCount
Document Type	B2BSchemas.AdvancedShipNotice
Select operation	Perform "Get" operation
Display Name	QuantityShipped

Property	Value
Definition Name	Quantity Invoiced
Schema File	CommonInvoice.xsd
Select Binding	CommonInvoice/InvoiceSummary/ShipDetail/<AttrGroup:VolumeAttGroupType>/Volume
Document Type	B2BSchemas.CommonInvoice
Select operation	Perform "Get" operation
Display Name	Quantity Invoiced

Property	Value
Definition Name	Quantity Paid for
Schema File	Browse to: /Web Reference/localhost1/Reference.xsd
Select Binding	Payment/Item/Quantity
Document Type	B2BSchemas.localhost1.Reference.Payment

Select operation	Perform "Get" operation

Property	Value
Definition Name	Supplier Name
Schema File	In the Schema Files dialog box, navigate to C:\Tutorial\Lessons\B2BSolution\B2BSchemas\ and then open CommonInvoice.xsd.
Select Binding	Seller/Address/Name
Document Type	B2BSchemas.CommonInvoice
Select operation	Perform "Get" operation
Display Name	Supplier Name

Step 3: Define Set Elements for the Vocabulary.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Define Get Elements for the Vocabulary before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To define an XML document Set element

1. In Facts Explorer, expand the **B2BVocabulary** folder, right-click **Version 1.0(not saved)**, and then select **Add New Definition**.
2. On the **Welcome to the Vocabulary Definition Wizard** page, select the **XML Document Element or Attribute** option, and then click **Next**.
3. On the **XML Document Element or Attribute** page, in the **Definition Name** box, type **Set Payment Terms to**, and then click **Browse**.

4. In the **Schema Files** dialog box, navigate to **C:\Tutorial\Lessons\B2BSolution\B2BSchemas\**, double-click **Web Reference**, double-click **localhost1**, and then double-click **Reference.xsd**.

5. In the **Select Binding** dialog box, expand **Payment**, expand **PaymentSummary**, select **Terms**, and then click **OK**.

6. On the **XML Document Element or Attribute** page, do the following:

Use this	**To do this**
Document type	Type **B2BSchemas.localhost1.Reference**.
Select operation	Select the **Perform "Set" operation** option.

7. Click **Next**.

8. On the **Specify the Display Name** page, accept the defaults, and then click **Finish**.

9. Repeat steps 1 through 8 to create a Set element with the following parameters:

Property	**Value**
Definition Name	Set Quantity Paid for to
Schema File	C:\Tutorial\Lessons\B2BSolution\B2BSchemas\Web Reference\localhost1\Reference.xsd
Select Binding	Payment/Item/Quantity
Document Type	B2BSchemas.localhost1.Reference
Select operation	Perform "Set" operation

Step 4: Publish the Vocabulary

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Define Set Elements for the Vocabulary before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To save and publish the vocabulary

1. In Facts Explorer, expand the **B2BVocabulary** folder, right-click **Version 1.0**, and then click **Save**.

2. In Facts Explorer, under the **B2BVocabulary** folder, right-click **Version 1.0**, and then click **Publish**

Step 5: Add a Business Policy

Purpose: A policy represents a set of rules that can be implemented from the orchestration. When a policy is applied, the conditions for each rule in the policy are evaluated. Rule actions are completed for those rules whose conditions evaluate to True, similar to an IF-THEN semantic. Conditions consist of predicates, facts, and vocabulary definitions. Actions often consist of setting data values or calling functions.

In this procedure,

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Publish the Vocabulary before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a new business policy

1. In the Policy Explorer pane, right-click **Policies**, and then select **Add New Policy**.

2. Change the name of **Policy1** to **Payment Policy**, and then press ENTER.

Step 6: Add Rules to the Policy

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Add a Business Policy before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add the policy rule and conditions for the 3 Way Match rule

1. In the Policy Explorer pane, expand the **Payment Policy** folder, right-click **Version 1.0(not saved)**, and then select **Add New Rule**.

2. Change the name **Rule1** to **3 Way Match**, and then press ENTER.

3. In the IF pane, right-click **Conditions**, and then click **Add logical AND**.

4. In the Facts Explorer pane, expand **Vocabularies**, expand **Predicates**, expand **Version 1.0**, and then drag **Equal** to the IF pane composer surface and drop it on the **AND** placeholder.

5. Repeat step 4 to add a second **Equal** statement to the conditions.

6. In the Facts Explorer pane, expand **Vocabularies**, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and drag **PO Quantity** to the IF pane composer surface and drop it onto the first **argument1** placeholder.

7. Drag **Quantity Shipped** and drop it onto the first **argument2** placeholder.

8. In the Facts Explorer pane, expand **Vocabularies**, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and drag **PO Quantity** to the IF pane composer surface and drop it onto the second **argument1** placeholder.

9. Drag **Quantity Invoiced** and drop it onto the second **argument2** placeholder.

To add the policy rule and conditions for the Negation of 3 Way Match rule

1. In the Policy Explorer pane, under the **Payment Policy** folder, right-click **Version 1.0(not saved)**, and then select **Add New Rule**.

2. Change the name **Rule1** to **Negation of 3 Way Match**, and then press ENTER.

3. In the IF pane, right-click **Conditions**, and then click **Add logical NOT**.

4. In the IF pane, right-click **NOT**, and then click **Add logical AND**.

5. In the Facts Explorer pane, under **Vocabularies**, expand **Predicates**, expand **Version 1.0**, and drag **Equal** to the IF pane composer surface and drop it onto the **AND** placeholder.

6. Repeat step 5 to add a second **Equal** statement to the conditions.

7. In the Facts Explorer pane, under **Vocabularies**, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and drag **PO Quantity** and drop it onto the first **argument1** placeholder

8. Drag **Quantity Shipped** to the IF pane and drop it onto the first **argument2** placeholder.

9. Drag **PO Quantity** to the IF pane and drop it onto the second **argument1** placeholder.

10. Drag **Quantity Invoiced** to the IF pane and drop it onto the second **argument2** placeholder.

To define the Actions for the rules

1. In the Policy Explorer pane, select **3 Way Match**.

2. In the Facts Explorer pane, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and then drag **Set Quantity Paid for to** to the THEN pane and drop it onto the **Actions** placeholder.

3. In the Facts Explorer pane, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and then drag **Quantity Invoiced** and drop it onto the **0** in the **Action** statement.

   The **Action** statement should read: **Set Quantity Paid for to Quantity Invoiced**.

4. In the Policy Explorer pane, select **Negation of 3 Way Match**.

5. In the Facts Explorer pane, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and then drag **Set Quantity Paid for to** to the THEN pane and drop it on the **Actions** placeholder.

6. In the Facts Explorer pane, expand **B2BVocabulary**, expand **Version 1.0 - Published**, and then drag **Quantity Shipped** to the THEN pane and drop it onto the **0** in the **Action** statement.

   The **Action** statement should read: **Set Quantity Paid for to Quantity Shipped**.

To add the policy rule for the payment terms

1. In the Policy Explorer pane, expand **Payment Policy**, right-click **Version 1.0**, and then click **Add New Rule**.

2. Change the name of the new rule to **Payment Terms for Supplier A**, and then press ENTER.

3. In the Facts Explorer pane, expand **Vocabularies**, and then drag the **Equal** predicate to the IF pane and drop it onto the **Conditions** placeholder.

4. In the Facts Explorer pane, expand **Vocabularies**, and then drag **Supplier Name** and drop it onto the **argument1** placeholder.

5. In the IF pane, click **argument2**, type **Supplier A**, and then press ENTER.

6. In the Facts Explorer pane, expand **Vocabularies**, and then drag **Set Payment Terms to** to the THEN pane, and drop it onto the **Actions** placeholder.

7. Click **Empty string**, type **30**, and then press ENTER.

   The **Action** statement should read: **Set Payment Terms to 30**.

8. Repeat steps 1 through 7 to create additional rules with the following values:

Use this	To do this
Rule Name	Payment Terms for Supplier B
Condition	Supplier Name **is equal to** Supplier B
Action	Set Payment Terms to 15

Use this	To do this
Rule Name	Payment Terms for Other Suppliers
Condition	AND Supplier Name (is) **not equal** (to)Supplier A Supplier Name (is) **not equal** (to) Supplier B
Action	Set Payment Terms to 45

10. You must save, publish, and deploy the policy before you leave the Business Rule Composer.

To save, publish, and deploy the policy

1. In the Policy Explorer pane, expand **Payment Policy**, right-click **Version 1.0**, and then click **Save**.
2. In the Policy Explorer pane, expand **Payment Policy**, right-click **Version 1.0**, and then click **Publish**.
3. In the Policy Explorer pane, expand **Payment Policy**, right-click **Version 1.0**, and then click **Deploy**.

Lesson 5: Integrate the Payment Policy with the Orchestration

In this lesson, you integrate the rules you created in Lesson 4 with the B2BProcess orchestration.

In This Section

- Step 1: Add a Reference to the Rule Engine
- Step 2: Define an Atomic Rules Scope
- Step 3: Add a Call Rules Shape to the Orchestration
- Step 4: Undeploy and Redeploy the Solution
- Step 5: Bind the Orchestration Ports

- Step 6: Test the Solution with the Rules

Step 1: Add a Reference to the Rule Engine

Purpose: You also define an atomic scope within the orchestration. An atomic scope is specified to enable use of nonserializable objects.

When the policy is implemented, the Rules Engine populates data in the payment voucher.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 4: Create a Payment Policy before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a reference to the rule engine DLL

1. In Visual Studio, in Solution Explorer, expand **B2BOrchestrations**, right-click **References**, and then select **Add Reference**.
2. In the **Add Reference** dialog box, on the **Browse** tab, browse to **<installation drive>:\Program Files\Microsoft BizTalk Server 2006**, double-click **Microsoft.RuleEngine.dll** to add the file as a referenced component, and then click **OK**.
3. On the **File** menu, click **Save All**.

Step 2: Define an Atomic Rules Scope

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Add a Reference to the Rule Engine before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To define an atomic rules scope in the B2BProcess

1. In Visual Studio, in Solution Explorer, double-click **B2BProcess.odx**.
2. On the orchestration design surface, right-click the solid green circle (begin shape) at the top of the design, and then select **Properties Window**.
3. In the Properties pane, change the value of the **Transaction Type** property to **Long Running**.

4. From the orchestration Toolbox, drag the **Scope** shape to the orchestration design surface, and drop the shape on the connecting line directly below the **Construct_PaymentVoucher** shape.

5. In the Properties pane, change the value of the **Name** property to **Rules Scope**.

6. In the Properties pane, from the **Transaction type** drop-down list, select **Atomic**.

Step 3: Add a Call Rules Shape to the Orchestration

.Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Define an Atomic Rules Scope before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To add a Call Rules shape to the B2BProcess

1. In Visual Studio, in Orchestration Designer, from the Toolbox, drag the **Call Rules** shape and drop it inside the **Rules Scope** shape.

2. In the Properties pane, change the value of the **Name** property to **Call_PaymentPolicy**.

3. Double-click the **Call_PaymentPolicy** shape.

4. In the **CallRules policy configuration** dialog box, from the **Select the business policy you wish to call** drop-down list, select **Payment Policy**.

5. Click **OK** to exit the **CallRules policy configuration** dialog box.

6. In Solution Explorer, right-click **B2Borchestrations**, and then click **Rebuild**.

7. On the **File** menu, click **Save All**.

Step 4: Undeploy and Redeploy the Solution

Purpose: You stop, unenlist, and undeploy the B2BProcess to ensure that the redeployment succeeds. Then you redeploy the B2BOrchestration to place a copy of the updated assembly in the BizTalk Management database (also known as the Configuration database) and install it in the global assembly cache. Because the B2BSchemas project is unchanged, it is not redeployed.

The orchestration binding also needs to be re-established. You will do this in Step 5: Bind the Orchestration Ports.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Add a Call Rules Shape to the Orchestration before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To undeploy and redeploy the solution

1. In Visual Studio, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand <**ServerName**>**.BizTalkMgmtDb.dbo**, expand **Orchestrations**, right-click **B2BOrchestrations.B2BProcess**, and then click **Stop**.
3. Right-click **B2BOrchestrations.B2BProcess** again, and then click **Unenlist**.
4. In the **BizTalk Explorer - Express Unenlist** dialog box, select all of the **Dependency Options** check boxes, select the **Terminate existing Orchestration Instances** check box, and then click **OK**.
5. In BizTalk Explorer, expand **Assemblies**, right-click **B2BOrchestrations**, and then click **Undeploy**.
6. In Solution Explorer, right-click **B2BOrchestrations**, and then click **Deploy**.

Step 5: Bind the Orchestration Ports

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Undeploy and Redeploy the Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To bind the orchestration ports

1. In BizTalk Explorer, right-click **<ComputerName>.BizTalkMgmtDb.dbo** and then click **Refresh**.
2. In BizTalk Explorer, expand **Orchestrations**, right-click **B2BOrchestrations.B2BProcess**, and then click **Bind**.
3. In the **Port Binding Properties** dialog box, on the **Binding** tab, do the following:

Use this	To do this
ReceiveASN_Port	From the drop-down list, select **ASN_ReceivePort**.
ReceiveInvoice_Port	From the drop-down list, select **Invoice_ReceivePort**.
SendPaymentAck_Port	In the drop-down list, expand **Send Ports**, and then select **SendPort_PaymentAck**.

4. In the **Port Binding Properties** dialog box, on the **Host** tab, from the **Host** drop-down list, select **BizTalkServerApplication**.

5. Click **OK**.

6. On the **File** menu, click **Save All**.

Step 6: Test the Solution with the Rules

Purpose: BizTalk Explorer is used to enlist and start an orchestration. When you enlist an orchestration you are associating the business process that you designed in Orchestration Designer with the physical environment in which it will run, and creating the core subscriptions in the MessageBox database. When you start the service, it will begin processing.

Prerequisites

Note the following requirements before you begin this step:

- You must complete the steps in Before You Begin the Tutorials before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To enlist, enable, and start the services

1. In BizTalk Explorer, expand **Orchestrations**, right-click **B2BOrchestrations.B2BProcess**, and then click **Enlist**.

2. In BizTalk Explorer, right-click **B2BOrchestrations.B2Bprocess**, and then click **Start**.

3. Verify that all the **Dependency** options are selected, and then click **OK**.

To test the solution

1. In Windows Explorer, browse to **C:\Tutorial**, and copy **POInstance.xml** to **C:\Tutorial\Filedrop\ReceivePO**.

2. Browse to **C:\Tutorial\Filedrop\FileArchive** to verify that POConfirmed was archived.

3. Browse to **C:\Tutorial\Filedrop\ERPSys** to verify that a payment acknowledgment was sent to the ERP system and that the B2B solution completed.

4. To verify that the payment voucher was updated before it was sent out, use BizTalk Health and Activity Tracking (HAT), or add a port to archive the payment voucher before it is sent out. Then compare the data in the archived payment voucher to the data in the payment acknowledgment.

Tutorial 4: Trading Partner Management

In this tutorial, you walk through the process in which business users use Business Activity Services (BAS) to set up business partners and business agreements. You also walk through the processes that the developer uses to implement this business relationship and to route messages into and out of the BAS Web site. You modify an existing orchestration by adding new elements, such as role links, so that it can communicate with the BAS Web site.

BAS is a Web site hosted in Microsoft Windows SharePoint Services. Business users use the BAS Web interface to set up company profiles and business agreements, and to manage business relationships. A developer, working in Microsoft Visual Studio 2005, implements the interaction of BAS profiles and relationships with the automated business process (orchestration) that is running on BizTalk Server.

If you worked through Tutorial 1: Enterprise Application Integration, you practiced implementing business solutions from the buyer's perspective. Tutorial 1 covered enterprise application integration (EAI) between two disparate computer systems, one in the warehouse and one used for enterprise resource planning (ERP).

This tutorial covers the supplier's side of the order fulfillment process. In this tutorial, you enable business users to manage trading partners.

Trading partners include buyers that send purchase orders to the supplier. Each partner has predefined business contracts with the supplier. The contracts define the minimum dollar amount that is required for a purchase order to be processed. If the amount of the incoming purchase order meets or exceeds the minimum dollar amount, the supplier sends a confirmation message to the buyer; otherwise, the supplier sends a negative acknowledgment to the buyer.

Before you begin this tutorial, you must set up your computer. For information about setting up your computer, see Before You Begin the Tutorials.

In This Section

- Lesson 1: Set Up BAS
- Lesson 2: Define the Business Relationship
- Lesson 3: Create the Messages and Expressions
- Lesson 4: Create the Trading Partner Relationship

Lesson 1: Set Up BAS

In this lesson, you complete several tasks to set up the Business Activity Services (BAS) site.

In This Section

- Step 1: Set Up the Host Instance
- Step 2: Configure IIS with the HTTP Adapter Receive Location
- Step 3: Register BizTalk Server with the Business Activity Services Portal
- Step 4: Create a Trusted Site in Internet Explorer

Step 1: Set Up the Host Instance

Purpose: When you add the account that the BizTalkServerApplication host instance runs as to the SharePoint Enabled Hosts group, you make the BizTalk host instance a Contributor on the SharePoint site so that it can send and receive messages from that site. Contributor is a SharePoint Services role. For more information about the Contributor role, see Windows SharePoint Services Help at http://go.microsoft.com/fwlink/?LinkId=18110.

Prerequisites

You must complete all of the steps in Before You Begin the Tutorials before you begin this step.

To add the user account to the SharePoint Enabled Hosts group

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Computer Management**.
2. In **Computer Management**, expand **Local Users and Groups**, and then click **Groups**.
3. In the Groups results pane, right-click **SharePoint Enabled Hosts**, and then click **Properties**.
4. In the **SharePoint Enabled Hosts Properties** dialog box, click **Add**.
5. In the **Select Users, Computers, or Groups** dialog box, in the **Enter the object name to select** box, type the account name you use for the BizTalkServerApplication host instance in the form <domain>\account name, and then click **OK**.
6. Click OK.

To restart the BizTalkServerApplication host instance

1. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **BizTalk Server Administration**.

2. In the BizTalk Administration console, expand **BizTalk Server 2006 Administration**, expand **BizTalk Group**, expand **Platform Settings**, and then click **Host Instances**.

3. In the Host Instances details pane, right-click **BizTalkServerApplication**, and then click **Restart**.

Step 2: Configure IIS with the HTTP Adapter Receive Location

Purpose: This tutorial uses party resolution to match an incoming request with a trading partner. In BizTalk Server, a trading partner is considered a party. Party resolution is the process of determining who sent a particular message. In this tutorial, you use an HTTP adapter receive location for party resolution.

You use an HTTP adapter to exchange information between BizTalk Server and an application. In this case, the HTTP adapter receive location accepts purchase order documents from the buyer.

The HTTP adapter is an IIS Internet Server Application Programming Interface (ISAPI) extension. The IIS process hosts and controls the receive locations that use the HTTP adapter.

The script configures IIS to work with the HTTP adapter receive location by creating an application pool and a virtual directory, allowing the BTSHttpReceive.dll to be served by IIS, and excluding the virtual directory from Windows SharePoint Services control.

For more information about configuring IIS, see How to Configure IIS for an HTTP Receive Location .

Prerequisites

Note the following requirement before you begin this step:

- You must complete all of the steps in Before You Begin the Tutorials before you begin this step.

To configure IIS with the HTTP adapter receive location

1. Click **Start**, and then click **Run**.

2. In the **Run** dialog box, type **cmd**, and then press ENTER.

3. At the command prompt, type **cd C:\Tutorial\Lessons\BAS\**, and then press ENTER.

4. At the command prompt, type the following command:

5. **SetupHTTPReceive.vbs /Username:***<Domain or ServerName>******<UserName>* **/Password:***<Password>*

6. Press ENTER.

7. When the script is finished, click **OK.**

8. At the command prompt, type **exit**, and then press ENTER.

Step 3: Register BizTalk Server with the Business Activity Services Portal

Purpose: You register a server running BizTalk Server with Business Activity Services (BAS) to associate the BizTalk Management database with the BAS Web site and create a receive location for the BAS outbox.

This process creates a receive port named STS.Outbox that receives messages sent from the BAS outbox.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Configure IIS with the HTTP Adapter Receive Location before you begin this step.
- You must log on as a member of the BAS Administrators group.

To register a BizTalk server

1. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **Business Activity Services Site**.
2. On the **Business Activity Services** site home page, click **BizTalk Servers**.
3. On the **BizTalk Servers** page, click **Register BizTalk Server**.
4. On the **BizTalk Servers: New Item** page, do the following:

Use this	To do this
Registration Name	Type **BizTalk Integration Server**.
BizTalk Management Database Server Name	Type <*ServerName*>.
BizTalk Management Database	Type **BizTalkMgmtDB**.

5. Click **Refresh Hosts Lists**.

   BAS updates the **Outbox Receive Location Host** box and the **Parameter Service Host** box with the default application.

6. Click **Save and Close**.

Step 4: Create a Trusted Site in Internet Explorer

Purpose: This tutorial uses script inside an InfoPath template to save a confirmation message to the BAS outbox. The code first saves the file to your temporary directory. As a result, you may need to add your server to your Trusted Sites, and you may have to accept a security warning in InfoPath.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Register BizTalk Server with the Business Activity Services Portal before you begin this step.

To add your server to the list of Trusted Sites in Internet Explorer

1. Open Internet Explorer.
2. On the **Tools** menu, click **Internet Options**.
3. In the **Internet Options** dialog box, on the **Security** tab, click **Trusted Sites**, and then click **Sites**.
4. In the **Trusted sites** dialog box, do the following:

Use this	To do this
Require server verification (https:) for all sites in this zone	Clear the check box.
Add this Web site to the zone	Type **http://**<*ServerName*>.

Click **Add**, and then click **Close**.

5. In the **Internet Options** dialog box, click **OK**.

Lesson 2: Define the Business Relationship

In this lesson, you modify an existing orchestration by creating role links, which define the business relationship between your organization and your trading partner's organization.

You can think of role links as paired ports. One port in a role link is the seller, and the other port is the buyer. For each role link, you must define a port type, a role link type, and the role link itself.

A *role link type* is a property that characterizes the relationship between two services or orchestrations by defining the part played by each of the services in the relationship and specifying the port types provided by each role.

Role link types represent service links in the Business Process Execution Language for Web Services (BPEL4WS) specification. Service links define and represent a business relationship such as an interaction between a buyer and seller. When you specify a business relationship in an agreement addendum in the BAS portal, you select a role link type.

A **role** is a collection of port types that either uses a service or implements a service. A role defines how parties interact with orchestrations. For example, an orchestration might use the role of Shipper. The Shipper has one or two parties associated with it. When the orchestration decides which shipping company to use to ship an item, it compares the prices of the parties in the Shipper role.

A **port type** is a property that defines the set of message interaction patterns, called operations, permitted at that endpoint. An operation can be one-way, in which case one message is sent or received, or it can be request-response, in which case a message is sent (or received) and a response is received (or sent).

An **operation** is a request or request-response pairing on a port, associated with either a send or receive action.

A **role link** is the relationship between roles defined by the message and port types used in the interactions in both directions.

In This Section

- Step 1: Open the OrderProcess Solution
- Step 2: Create the Business Relationship Role Link
- Step 3: Create the Business Policy Role Link
- Step 4: Create the SendDocumentsToSharePoint Role Link
- Step 5: Create the ReceiveDocumentsFromOutbox

- Role Link
- Step 6: Create the Structured Parameter Schema for the Business Policy

Step 1: Open the OrderProcess Solution

Purpose: The orchestration in the OrderProcess solution represents and automates a business process. In this lesson, you will modify the orchestration to work with BAS.

Prerequisites

Note the following requirement before you begin this step:

- You must complete Step 4: Create a Trusted Site in Internet Explorer before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To open the OrderProcess solution

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

Step 2: Create the Business Relationship Role Link

Purpose: The Purchasing role link connects the Buyer port and the Seller port. You must complete four tasks to create the Purchasing role link:

1. Create the role link type and the associated roles.
2. Create and configure the port type for the Seller role.
3. Create and configure the port type for the Buyer role.
4. Create the Purchasing role link that connects the ports of the roles.

You create the port types for the roles you created in the previous task. Each port type contains request operations that have message types. The Seller port type has a request operation called PO. The PO operation has a message type defined by the PurchaseOrder schema.

The Buyer port type has a request operation called Confirmation. The Confirmation operation has a message type defined by the Confirmation schema. The Buyer port type also has a request operation called NegativeAck. The NegativeAck operation has a message type defined by the NAK schema.

The Purchasing role link connects two ports: the Buyer port and the Seller port.

Prerequisites

Note the following requirement before you begin this step:

- You must complete Step 1: Open the OrderProcess Solution before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the PurchasingType role link type

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.
4. In Solution Explorer, double-click **OrderService.odx**.
5. In the Orchestration View window, expand **Types**, right-click **Role Link Types**, and then click **New Role Link Type**.
6. Right-click **RoleLinkType_1** and then click **Properties Window**.
7. In the Properties pane, change the value of the **Identifier** property to **PurchasingType** and press ENTER.

To create the PurchasingType roles

1. Expand **PurchasingType**, right-click **Role_1**, and then click **Properties Window**.
2. In the Properties pane, change the value of the **Identifier** property to **Seller**, and then press ENTER.
3. Right-click **PurchasingType**, and then click **New Role**.
4. Right-click **Role_1**, and then click **Properties Window**.
5. In the Properties pane, change the value of the **Identifier** property to **Buyer**, and then press ENTER.

To create the Seller port type

1. In the Orchestration View window, right-click **Seller**, and then click **Add Port Type**.

2. On the **Welcome to the Port Type Wizard** page, click **Next**.

3. On the **Select a Port Type or create a new Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **SellerPortType**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public-no limit** option.

4. Click **Next**.

5. On the **Completing the Port Wizard** page, click **Finish**.

To configure the Seller operation

1. In the Orchestration View window, expand **Port Types**, expand **SellerPortType**, right-click **Operation _1**, and then click **Properties Window**.

2. In the Properties pane, change the value of the **Identifier** property to **PO**, and then press ENTER.

3. Expand **PO**, right-click **Request**, and then click **Properties Window**.

4. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.PurchaseOrder**.

To create the Buyer port type

1. In the Orchestration View window, expand **Role Link Types**, expand **PurchasingType**, right-click **Buyer**, and then click **Add Port Type**.

2. On the **Welcome to the Port Type Wizard** page, click **Next**.

3. On the **Select a Port Type or create a new Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **BuyerPortType**.

Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public-no limit** option.

4. Click **Next**.

5. On the **Completing the Port Wizard** page, click **Finish**.

To configure the Buyer operations

1. In the Orchestration View window, expand **Port Types**, expand **BuyerPortType**, right-click **Operation _1**, and then click **Properties Window**.

2. In the Properties pane, change the value of the **Identifier** property to **Confirmation**, and then press ENTER.

3. Expand **Confirmation**, right-click **Request**, and then click **Properties Window**.

4. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.Confirmation.**

5. In the Orchestration View window, expand **Port Types**, right-click **BuyerPortType**, and then click **New Operation.**

6. Right-click **Operation_1**, and then click **Properties Window**.

7. In the Properties pane, change the value of the **Identifier** property to **NegativeAck**, and then press ENTER.

8. Expand **NegativeAck**, right-click **Request**, and then click **Properties Window**.

9. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.NAK**.

To create the Purchasing role link

1. In Orchestration Designer, right-click the left **Port Surface** and then click **New Role Link.**

2. On the **Welcome to the Role Link Wizard** page, click **Next.**

3. On the **Role Link Name** page, in the **Name** box, type **Purchasing**, and then click **Next**.

4. On the **Role Link Type** page, do the following:

Use this	To do this
Would you like to create a new type for this Role Link, or reuse an existing Role Link type that has already been defined?	Select the **Use an existing Role Link Type** option to use an existing role link type.
Role Link Type Name	From the drop-down list, select **OrderProcess.PurchasingType**.

5. Click **Next**.

6. On the **Role Identification** page, from the **Which role will this orchestration implement to receive and process messages from partners?** drop-down list, select **Seller**, and then click **Next**.

7. On the **Role Link Usage** page, select the **I will be receiving the first message through my implemented role** option, and then click **Finish**.

8. On the **File** menu, click **Save All**.

Step 3: Create the Business Policy Role Link

Purpose: Business Activity Services-enabled orchestrations must use predefined names for the role link types, roles, and port types that interact with the Windows SharePoint Services collaboration site and the trading partner publishing Web service (TPPubWS). For more information about the predefined names, see Business Activity Services Messaging Infrastructure .

Prerequisites

Note the following requirement before you begin this step:

- You must complete Step 2: Create the Business Relationship Role Link before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

Add a reference to the BizTalk Server RoleLinkTypes assembly

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, double-click **OrderService.odx**.

5. In Visual Studio, in Solution Explorer, expand the **OrderProcess** project, right-click **References**, and then click **Add Reference**.

6. In the **Add Reference** dialog box, on the **Browse** tab, navigate to **C:\program files\Microsoft BizTalk Server 2006**, select **Microsoft.BizTalk.KwTpm.RoleLinkTypes.dll**, click **Add**, and then click **OK**.

Create the ParameterWebService Role Link

1. In Visual Studio, in the orchestration design surface for the OrderProcess.odx orchestration, right-click the left **Port Surface**, and then click **New Role Link**.

2. In the Role Link Wizard, on the **Welcome to the Role Link Wizard** page, click **Next**.

3. On the **Role Link Name** page, in the **Name** box, type **ParameterWebService**, and then click **Next**.

4. On the **Role Link Type** page, do the following:

Use this	To do this
Would you like to create a new type for this Role Link, or reuse an existing Role Link type that has already been defined?	Select the **Use an existing Role Link Type** option.
Role Link Type Name	From the drop-down list, select **Microsoft.BizTalk.KwTpm.BasParameterLT**.

5. Click **Next**.

6. On the **Role Link Usage** page, select the **Consumer Role: I will be sending the first message** option, and then click **Finish**.

Step 4: Create the SendDocumentsToSharePoint Role Link

Purpose: The SendDocumentsToSharePoint role link sends documents to the BAS site.

Prerequisites

Note the following requirement before you begin this step:

- You must complete Step 3: Create the Business Policy Role Link before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the sendToInboxPT role link type

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, double-click **OrderService.odx**.

5. In the Orchestration View window, expand **Types**, right-click **Role Link Type**, and then click **New Role Link Type**.

6. Right-click **RoleLinkType_1**, and then click **Properties Window**.

7. In the Properties pane, change the value of the **Identifier** property to **sendBusinessDocumentsLT**, and then press ENTER.

8. In the Orchestration View window, expand **sendBusinessDocumentsLT**, right-click **Role_1**, and then click **Properties Window**.

9. In the Properties pane, change the value of the **Identifier** property to **sender**, and then press ENTER.

10. Right-click **sender**, and then click **Add Port Type**.

11. On the **Welcome to the Port Type Wizard** page, click **Next**.

12. On the **Select a Port Type or create a new Port Type** page, do the following:

Use this	**To do this**
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **sendToInboxPT**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public - no limit** option.

13. Click **Next**.

14. On the **Completing the Port Wizard** page, click **Finish**.

To add a second port type to the sender role

1. In the Orchestration View window, right-click **sender**, and then click **Add Port Type**.

2. On the **Welcome to the Port Type Wizard** page, click **Next**.

3. On the **Select a Port Type or create a new Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **sendToSentItemsPT**.
Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public - no limit** option.

4. Click **Next**.

5. On the **Completing the Port Wizard** page, click **Finish**.

To configure the port types for sending documents into BAS

1. In the Orchestration View window, expand **PortTypes**, expand **sendToInboxPT**, right-click **Operation _1**, and then click **Properties Window**.

2. In the Properties pane, change the value of the **Identifier** property to **PO**, and then press ENTER.

3. Expand **PO**, right-click **Request**, and then click **Properties Window**.

4. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.PurchaseOrder**.

5. In the Orchestration View window, expand **PortTypes**, expand **sendToSentItemsPT**, right-click **Operation _1**, and then click **Properties Window**.

6. In the Properties pane, change the value of the **Identifier** property to **Confirmation**, and then press ENTER.

7. Expand **Confirmation**, right-click **Request**, and then click **Properties Window**.

8. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.Confirmation**.

9. In the Orchestration View window, right-click **sendToSentItemsPT**, and then click **New Operation**.

10. Right-click **Operation_1**, and then click **Properties Window**.

11. In the Properties pane, change the value of the **Identifier** property to **NegativeAck**, and then press ENTER.

12. Expand **NegativeAck**, right-click **Request**, and then click **Properties Window**.

13. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.NAK**.

To create the SendDocumentsToSharePoint role link

1. In Orchestration Designer, right-click the right **Port Surface**, and then click **New Role Link**.

2. On the **Welcome to the Role Link Wizard** page, click **Next**.

3. On the **Role Link Name** page, in the **Name** box, type **SendDocumentsToSharePoint**, and then click **Next**.

4. On the **Role Link Type** page, do the following:

Use this	To do this
Would you like to create a new type for this Role Link, or reuse an existing Role Link type that has already been defined?	Select the **Use an existing Role Link Type** option.
Role Link Type Name	From the drop-down list, select **OrderProcess.sendBusinessDocumentsLT**.

5. Click **Next**.

6. On the **Role Link Usage** page, select the **Consumer Role: I will be sending the first message** option, and then click **Finish**.

7. On the **File** menu, click **Save All**.

Step 5: Create the ReceiveDocumentsFromOutbox Role Link

Purpose: Integrating the orchestration with the BAS outbox lets the orchestration move messages to the BAS outbox.

Prerequisites

Note the following requirement before you begin this step:

- You must complete Step 4: Create the SendDocumentsToSharePoint Role Link before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the receiveBusinessDocumentsLT role link type

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.
4. In Solution Explorer, double-click **OrderService.odx**.
5. In the Orchestration View window, expand **Types**, right-click **Role Link Types**, and then click **New Role Link Type**.
6. Right-click **RoleLinkType_1**, and then click **Properties Window.**
7. In the Properties pane, change the value of the **Identifier** property to **receiveBusinessDocumentsLT**, and then press ENTER.
8. In the Orchestration View window, expand **receiveBusinessDocumentsLT**, right-click **Role_1**, and then click **Properties Window.**
9. In the Properties pane, change the value of the **Identifier** property to **receiver**, and then press ENTER.
10. In the Orchestration View window, right-click **receiver**, and then click **Add Port Type**.
11. On the **Welcome to the Port Type Wizard** page, click **Next**.
12. On the **Select a Port Type or create a new Port Type** page, do the following:

Use this	To do this
Select the port type to be used for this port	Select the **Create a new Port Type** option.
Port Type Name	Type **receiveFromOutboxPT**.

Communication Pattern	Select the **One-Way** option.
Access Restrictions	Select the **Public - no limit** option.

13. Click **Next**.

14. On the **Completing the Port Wizard** page, click **Finish**.

To configure a port type for receiving documents from BAS

1. In the Orchestration View window, expand **PortTypes**, expand **receiveFromOutboxPT**, right-click **Operation _1**, and then click **Properties Window**.

2. In the Properties pane, change the value of the **Identifier** property to **Confirmation**, and then press ENTER.

3. Expand **Confirmation**, right-click **Request**, and then click **Properties Window**.

4. In the Properties pane, in the **Message Type** drop-down list, expand **Schemas**, and then select **OrderProcess.Confirmation**.

To create the ReceiveDocumentsFromOutbox role link

1. In Orchestration Designer, right-click the right **Port Surface** and then click **New Role Link**.

2. On the **Welcome to the Role Link Wizard** page, click **Next**.

3. On the **Role Link Name** page, in the **Name** box, type **ReceiveDocumentsFromOutbox**, and then click **Next**.

4. On the **Role Link Type** page, do the following:

Use this	**To do this**
Would you like to create a new type for this Role Link, or reuse an existing Role Link type than has already been defined?	Select the **Use an existing Role Link Type** option.
Role Link Type Name	From the drop-down list, select **OrderProcess.receiveBusinessDocumentsLT**.

5. Click **Next**.

6. On the **Role Link Usage** page, select the **Provider Role: I will be receiving the first message** option, and then click **Finish**.

7. On the **File** menu, click **Save All**.

Step 6: Create the Structured Parameter Schema for the Business Policy

Purpose: In Step 3: Create the Business Policy Role Link, you created the Business Policy role link. In this step, you create the schema for the business policy, and promote the schema fields so that you can use them in messages.

Prerequisites

Note the following requirement before you begin this step:

- You must complete Step Step 5: Create the ReceiveDocumentsFromOutbox Role Link before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the schema file

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.
4. In Solution Explorer, right-click **OrderProcess**, point to **Add**, and then click **New Item**.
5. In the **Add New Item – OrderProcess** dialog box, from **Categories**, select **Schema Files**, from **Templates**, select **Schema**, in the **Name** box, type **PurchasingParameters.xsd**, and then click **Add**.

To define the schema

1. In Solution Explorer, double-click **PurchasingParameters.xsd**.
2. In BizTalk Editor, in the **PurchasingParameters** schema tree control on the left, click the **<Schema>** node.
3. In the Properties pane, click the **Schema Editor Extensions** ellipsis [**...**].
4. In the **Schema Editor Extension** dialog box, select the **BAS Parameter Schema Extension** check box, and then click **OK**.
5. In the tree control on the left, click the **Root** node.

6. In the Properties pane, change the value of the **Node Name** property to **PurchaseOrder**, and then press ENTER.

7. In the tree control, right-click **PurchaseOrder**, point to **Insert Schema Node**, and then click **Child Field Element**.

8. Type **MinPOAmount** as the field name, and then press ENTER.

9. In the **MinPOAmount** Properties pane, from the **Data Type** drop-down list, select **xs:float**.

10. Right-click **PurchaseOrder**, point to **Insert Schema Node**, and then click **Child Field Element**.

11. Type **DiscountRate** as the field name, and then press ENTER.

12. In the **DiscountRate** Properties pane, from the **Data Type** drop-down list, select **xs:integer**.

13. In the tree control, click the **PurchaseOrder** node.

14. In the **PurchaseOrder** Properties pane, from the **Role Link Types** drop-down list, select the check box next to **OrderProcess.PurchasingType**, and then press ENTER.

To promote the schema fields

1. In BizTalk Editor, in the **PurchasingParameters** schema tree control on the left, click the **<Schema>** node.

2. In the Properties pane, click the **Promote Properties** ellipsis [**...**].

3. In the **Promote Properties** dialog box, on the **Distinguished Fields** tab, in the schema tree, expand **PurchaseOrder**, click **MinPOAmount**, and then click **Add**.

4. In the schema tree, click **DiscountRate**, click **Add**, and then click **OK**.

5. On the **File** menu, click **Save All Files**.

Lesson 3: Create the Messages and Expressions

In this lesson, you add messages and expressions to the orchestration and connect the role links to the action shapes.

In This Section

- Step 1: Create the Messages for the Structured Parameter
- Step 2: Connect the Role Link Ports

- Step 3: Create the Construct PurchaseParameterMsg Message
- Step 4: Create the ParamRequestMsg Message
- Step 5: Create the Calculate PO Info Expression
- Step 6: Set the File Name for the Inbox Message
- Step 7: Build and Deploy the Solution

Step 1: Create the Messages for the Structured Parameter

Purpose: You use the message instance variables to associate the schema the orchestration associates with each instance of a message.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 6: Create the Structured Parameter Schema for the Business Policy before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the PurchaseParameterMsg message instance variable

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.
4. In Solution Explorer, double-click **OrderService.odx**.
5. In the Orchestration View window, right-click **Messages**, and then click **New Message.**
6. Click **Message_1**, and then in the **Message_1** Properties pane, do the following:

Use this	To do this
Identifier	Type **PurchaseParameterMsg**, and then press ENTER.
Message Type	In the drop-down list, expand **Schemas**, and then click **OrderProcess.PurchasingParameters**.

To create the request message instance variable

1. In the Orchestration View window, right-click **Messages**, and then click **New Message.**
2. Click **Message_1**, and then in the **Message_1** Properties pane, do the following:

Use this	**To do this**
Identifier	Type **ParamRequestMsg**, and then press ENTER.
Message Type	In the drop-down list, expand **Web Message Types**, and then click **Select from referenced assembly**.

3. In the **Select Artifact Type** dialog box, expand the **Microsoft.BizTalk.KwTpm.RoleLinkTypes** assembly, and then click the **Microsoft.BizTalk.KwTpm.TPPubWS.TPPubWS_** node.
4. In the right pane, select **GetStructuredParameterValue_request**, and then click **OK.**

To create the response message instance variable

1. In the Orchestration View window, right-click **Messages**, and then click **New Message**.
2. Click **Message_1**, and then in the **Message_1** Properties pane, do the following:

Use this	**To do this**
Identifier	Type **ParamResponseMsg**, and then press ENTER.
Message Type	In the drop-down list, expand **Web Message Types**, and then click **Select from referenced assembly**.

3. In the **Select Artifact Type** dialog box, expand the **Microsoft.BizTalk.KwTpm.RoleLinkTypes** assembly, and then click the **Microsoft.BizTalk.KwTpm.TPPubWS.TPPubWS_** node.
4. In the right pane, select **GetStructuredParameterValue_response**, and then click **OK**.

To associate the Construct request to retrieve structured parameter value shape with the ParamRequestMsg message instance variable

1. In Orchestration Designer, on the orchestration design surface, select the **Construct request to retrieve structured parameter value** shape
2. In the Properties pane, in the **Messages Constructed** drop-down list, select the **ParamRequestMsg** check box, and then press ENTER.

To associate the Construct PurchaseParameterMsg shape with the PurchaseParameterMsg message instance variable

1. In Orchestration Designer, on the orchestration design surface, select the **Construct PurchaseParameterMsg** shape

2. In the Properties pane, in the **Messages Constructed** drop-down list, select the **PurchaseParameterMsg** check box, and then press ENTER.

To associate the Request Value shape with the ParamRequestMsg message instance variable

1. In Orchestration Designer, on the orchestration design surface, select the **Request Value** shape

2. In the Properties pane, in the **Message** drop-down list, select **ParamRequestMsg**, and then press ENTER.

To associate the Receive Value shape with the ParamResponseMsg message instance variable

1. In Orchestration Designer, on the orchestration design surface, select the **Receive Value** shape

2. In the Properties pane, in the **Message** drop-down list, select **ParamResponseMsg**, and then press ENTER.

Step 2: Connect the Role Link Ports

Purpose: Connecting ports to the send and receive shapes tells the orchestration where it receives messages from and where it sends messages to. You defined logical (also known as early binding) ports for the orchestration. In Step 5: Configure and Start the Application, you define the physical ports the orchestration uses.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Create the Messages for the Structured Parameter before you begin this step.

- You must log on as a member of the BizTalk Server Administrators group.

To connect the role link ports to the Send and Receive shapes

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, double-click **OrderService.odx**.

5. In Orchestration Designer, select the green handle of a role link port and drag it to the green handle of a **Receive** shape, or to the green handle of a **Send** shape.

   The following table shows you which ports to connect to which shapes.

Connect this role link port	**To this shape**
SellerPortType - PO - Request	**Receive PO**
BuyerPortType - Confirmation - Request	**Send Confirmation to Partner**
BuyerPortType - NegativeAck - Request	**Send NAK to Partner**
TpPubWS - GetStructured ParameterValue -Request	**Request Value**
TpPubWS - GetStructured ParameterValue - Response	**Receive Value**
sendToInboxPT – PO - Request	**Send PO to Inbox**
sendToSentItemsPT – Confirmation - Request	**Send Confirmation to Sent Items**
sendToSentItemsPT – NegativeAck - Request	**Send NAK to Sent Items**
receiveFromOutboxPT – Confirmation - Request	**Receive Confirmation from Outbox**

6. On the **File** menu, click **Save All**.

Step 3: Create the Construct PurchaseParameterMsg Message

Purpose: The expression associated the values returned by the the structured parameter in the ParameterResponse message with the orchestration.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Connect the Role Link Ports before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the message for the Construct PurchaseParameterMsg shape

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, double-click **OrderService.odx**.

5. In the **Construct PurchaseParameterMsg** shape, double-click the interior **Construct PurchaseParameterMsg** shape, and then in the Expression Editor, type the following:

6. Click **OK** to close the Expression Editor.

Step 4: Create the ParamRequestMsg Message

Purpose: At run time, your orchestration makes a call to the BAS TpPubWS Web service to retrieve the values defined for the structured parameters associated with the OrderProcess.PurchaseingType role link type. You can assign the retrieved values to a System.Xml.XmlDocument object and cast them back to a message of the original schema type: Purchasing Parameters.

In the message that is sent to the Web service, you need to set six parameter values:

- **orchestrationName**. This is the fully qualified name of the current orchestration.
- **orchestrationAssemblyStrongName**. This is the assembly strong name of the current assembly.
- **parameterSchemaFullName**. This is the fully qualified parameter schema name.
- **partnerID**. This is the ID of the partner for which parameter values are retrieved.

  In this example, we extract the partner ID from the purchase order message.
- **roleLinkTypeAssemblyStrongName**: This is the strong name of the assembly in which you defined the role link type.
- **roleLinkTypeFullName**. This is the name of the role link type that you associated with the parameter schema.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Create the Construct PurchaseParameterMsg Message before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the ParamRequestMsg Message

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.
4. In Solution Explorer, double-click **OrderService.odx**.
5. In Orchestration Designer, in the **Construct request to retrieve structured parameter value** shape, double-click **Initialize ParamRequestMsg** to open the expression editor.
6. In BizTalk Expression Editor, type the following expression:
7. Click **OK** to close the expression editor.
8. On the **File** menu, click **Save All**.

Step 5: Create the Calculate PO Info Expression

Purpose: The Calculate PO expression calculates the value of a purchase order, including the discount rate and minimum amount defined in the structured parameter schema. The discount rate and minimum amount are collected on the BAS site in the **Agreement** template, on the **Addendums** tab.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Create the ParamRequestMsg Message before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To create the Calculate PO Info expression

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, double-click **OrderService.odx**.

5. In Orchestration Designer, double-click the **Calculate PO Info** shape, and then in the Expression Editor, type the following:

6. Click **OK**

Step 6: Set the File Name for the Inbox Message

Purpose: To guarantee that a particular message appears in the Inbox only one time, you set the file name property for the message before the BizTalk Host sends it to the Inbox. If the BizTalk Host sends this message more than once to the Inbox — for example, in an error condition in which the host retries the send — the new message replaces the existing message with the same file name in the Inbox. If you did not create this file name and instead relied on a unique message ID created by BizTalk Server, the same message could appear more than once in the Inbox (but with different file names).

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Create the Calculate PO Info Expression before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To set the file name for the Inbox message

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, expand **OrderProcess**, right-click **References**, and then click **Add Reference**.

5. In the **Add Reference** dialog box, on the **Browse** tab, navigate to **<***BizTalk Server installation directory***>\BAS Runtime Components**, select **Microsoft.BizTalk.KwTpm.StsDefaultPipelines.dll**, click **Add**, and then click **OK**.

6. In the message box, click **OK**.

7. In Orchestration Designer, in the **Construct Inbox Message** shape, double-click **Assign Inbox Filename** to open the expression editor.

8. In BizTalk Expression Editor, type the following expression:

9. Click **OK**.

10. On the **File** menu, click **Save All**.

Step 7: Build and Deploy the Solution

Purpose: Build the project to generate an assembly, and then deploy the assembly so that it is added to the global assembly cache and the BizTalk Management database.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 6: Set the File Name for the Inbox Message before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To verify that the key file is set for the assembly

1. Click **Start**, point to **Programs**, point to **Microsoft Visual Studio 2005**, and then click **Microsoft Visual Studio 2005**.

2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

3. In the **Open Project** dialog box, browse to **C:\Tutorial\Lessons\BAS\OrderProcess\**, click **OrderProcess.sln**, and then click **Open**.

4. In Solution Explorer, right-click **OrderProcess**, and then click **Properties**.

5. In the **OrderProcess Property Pages** dialog box, on the **Assembly** tab, scroll down to **Assembly Key File** and verify that it is set to **C:\Tutorial\Tutorial.snk**.

   If the **Assembly Key File** is not set to **C:\Tutorial\Tutorial.snk**, do the following:

   - Select **Assembly Key File**, and then click the ellipsis button (**...**).
   - In the **Assembly Key File** dialog box, browse to **C:\Tutorial**, click **Tutorial.snk,** and then click **Open**.

6. Click **OK**.

7. On the **File** menu, click **Save All**.

To build and deploy the project

1. In Solution Explorer, right-click **Solution 'OrderProcess'**, and then click **Build Solution**.

2. In Solution Explorer, right-click **OrderProcess**, and then click **Deploy**.

To verify the roles have been deployed to the BizTalk Management database

1. On the **View** menu, click **BizTalk Explorer**.

2. In **BizTalk Explorer**, expand **<***servername***>.BizTalkMgmtDb.dbo**, right-click **<***servername***>.BizTalkMgmtDb.dbo**, and then click **Refresh**.

3. Expand **Roles** to verify that the five roles were added to the BizTalk Management database when you deployed the assembly.

   The roles you created are:

   - Microsoft.BizTalk.KwTpm.BasParameterLT.ParameterConsumer
   - OrderProcess.PurchasingType.Buyer
   - OrderProcess.PurchasingType.Seller
   - OrderProcess.receiveBusinessDocumentsLT.receiver
   - OrderProcess.sendBusinessDocumentsLT.sender

After you deploy the assembly, you can discover the assembly strong name.

To discover the assembly strong name of the current assembly

1. Click **Start**, point to **Program Files**, point to **BizTalk Server 2006**, and then click **BizTalk Server Administration**.

2. In the console tree on the left side of the BizTalk Administration console, expand **BizTalk Server 2006 Administration**, expand **BizTalk Group**, expand **Applications**, expand **BizTalk Application 1**, and then click **Resources**.

3. In the **Resources** results pane, expand the name column so that you can see the entire contents.

4. In the **OrderProcess** row, find the value for the **PublicKeyToken** property. In the following example:

   OrderProcess, Version=1.0.0.0, Culture=neutral, PublicKeyToken=51bf5941723a688d

   The value of the public key token is 51bf5941723a688d.

To update the ParamRequestMsg Message

1. In Orchestration Designer, in the **Construct request to retrieve structured parameter value** shape, double-click **Initialize ParamRequestMsg** to open the expression editor.
2. In BizTalk Expression Editor, replace each instance of xxxxxxxxxxxxxxxx with the value of the public key token that you obtained in the previous procedure.
3. Click **OK** to close the expression editor.
4. On the File menu, click Save All.
5. Build and deploy the projects and solution again.

Lesson 4: Create the Trading Partner Relationship

You use the BAS portal site to create the trading partner relationship. You start by creating a self profile for your organization and a partner profile for your partner. The profiles contain information about both business entities. Your self profile contains information about your organization that you want to share with a partner. A partner profile contains information about the partner organization, and information about how you will communicate with the partner.

When you create and activate an agreement between your self profile and a partner profile, you begin the trading partner relationship.

In This Section

- Step 1: Create the Self Profile
- Step 2: Create the Partner Profile
- Step 3: Associate Messages with a Template
- Step 4: Create and Activate the Partner Agreement
- Step 5: Configure and Start the Application
- Step 6: Test the Scenario

Step 1: Create the Self Profile

Purpose: You use the self profile to store information about your organization that you want to share with a partner. The self profile allows you to identify internally your business or different departments within your business.

In Step 2: Create the Partner Profile, you define a partner profile, and in Step 4: Create and Activate the Partner Agreement, you create an agreement between the self profile you create in this step and the partner profile you create in step 2.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Lesson 3: Create the Messages and Expressions before you begin this step.
- You must log on as a member of one of the following groups:
  - BAS Business Users
  - BAS Business Managers
  - BAS Administrators

To create a self profile

1. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **Business Activity Services Site**.
2. On the **Business Activity Services** site home page, click **My Profiles**.
3. On the **My Profiles** page, click **New Profile**.
4. In the **My Company Profile** template, on the **General Information** tab, do the following:

Use this	To do this
Partner Name	Type **My profile**.
Description	Type **This is a profile for my company**.

5. Click **Submit Profile** to upload the information to the Trading Partner Management database, and to store the profile in the BAS site.
6. In the profile submitted successfully message box, click **OK**.

The **My Profiles** page refreshes automatically to show the new self profile.

Step 2: Create the Partner Profile

Purpose: You use the partner profile to store information about the partner. This information allows BAS to automatically generate the necessary physical ports for the partner, and establish and maintain the relationship with the partner.

The partner profile contains preferences for your partner's organization, including the global ports that the partner uses to receive information from you. In Step 4: Create and Activate the Partner

Agreement, you create an agreement between the self profile you created in step 1 and the partner profile you create in this step.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 1: Create the Self Profile before you begin this step.
- You must log on as a member of one of the following groups:
  - BAS Business Users
  - BAS Business Managers
  - BAS Administrators

To create a partner profile

1. On the **Business Activity Services** site home page, click **Partner Profiles**.
2. On the **Partner Profiles** page, click **New Partner**.
3. In the **Partner Profile** template, on the **General Information** tab, in the **Partner Name** box, type **Partner1**, and then click **Insert Custom Property**.
4. In the **Custom properties** section, click the **Property qualifier** ellipsis [**...**].
5. In the **Select Property** dialog box, from the **Select property** list, select **Windows User**, and then click **OK**.
6. In the **Custom properties** section, do the following:

Use this	To do this
Property value	type <*ServerName or DomainName*>\<*UserName*>
Run-time process property	Select the check box to make the property available to BizTalk Server.

1.

To configure a port for the partner profile

1. In the **Partner Profile** template, on the **Advanced** tab, click **Insert global port**, and then do the following:

Use this	To do this
Global port name	Type **Partner1DestinationPort**.
Choose BizTalk send port	Right-click **Choose BizTalk send port**, and then click **Specify port configuration**.

3. In the **Configure port** section, click the **Send Pipeline** ellipsis [**…**].

4. In the **Select Send Pipeline** dialog box, select **Microsoft.BizTalk.DefaultPipeLines.XmlTransmit**, and then click **OK**.

5. In the **Configure port** section, do the following:

Use this	To do this
Retry Count	Type **3**.
Retry Interval	Type **5**.
Transport Type	From the drop-down list, select **FILE**.
Destination Location	Type **C:\Tutorial\Filedrop\Partner1**.
File Name	Type **Confirmation%MessageID%.xml**.

6. Click **Submit Profile** to upload the information to the Trading Partner Management database, and to store the profile in the BAS site.

7. In the profile submitted successfully message box, click **OK**.

You must deploy the partner profile before you can associate it with an agreement.

To deploy the partner profile

1. On the **Partner Profiles** page, point to **Partner1**, and from the drop-down list, select **Deploy Partner**.

2. On the **Deploy Partner** page, select the **BizTalk Integration Server** option, and then click **Deploy**.

Step 3: Associate Messages with a Template

Purpose: You can upload InfoPath document templates to partner templates. This allows all outgoing and incoming messages to be automatically associated with InfoPath templates.

In this procedure, you upload the following templates:

- Purchase order
- Confirmation

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 2: Create the Partner Profile before you begin this step.
- You must log on as a member of one of the following groups:
  - BAS Business Users
  - BAS Business Managers
  - BAS Administrators

To upload the document templates

1. On the **Partner Profiles** page, in the **Partner1** row, click **Templates**.
2. On the **Partner1** page, click **Upload Document**.
3. On the **Partner1: Upload Document** page, do the following:

Use this	To do this
Name	Type **C:\Tutorial\Lessons\BAS\PO Template\PurchOrd.xsn**.
Namespace	Type **http://OrderProcess.PurchaseOrder**.

4. Click **Save and Close**.
5. On the **Partner1** page, click **Upload Document**.
6. On the **Partner1: Upload Document** page, do the following:

Use this	To do this
Name	Type **C:\Tutorial\Lessons\BAS\PO Template\Confirmation.xsn**.
Namespace	Type **http://OrderProcess.Confirmation**.

7. Click **Save and Close**.

Step 4: Create and Activate the Partner Agreement

Purpose: A partner agreement links a partner profile to a self profile. You must define at least one addendum, which is specific to an orchestration and defines the business relationship between the partners. Each business process requires a separate addendum. A single agreement can have multiple addendums.

You create an addendum within the agreement by using the Purchasing relationship (role link type) you created in Lesson 2. The agreement template shows both the buyer and seller roles in the relationship. You must map the agreement to the global port you defined in the partner profile before you can activate the agreement. When you activate the agreement, BizTalk Server writes the agreement details, including profile and port information, to the BizTalk Management database.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 3: Associate Messages with a Template before you begin this step.
- You must log on as a member of one of the following groups:
  - BAS Business Managers
  - BAS Administrators

To create the agreement

1. On the **Business Activity Services** site home page, click **Partner Profiles**.
2. On the **Partner Profiles** page, point to **Partner1**, and from the drop-down list, select **New Agreement**.
3. In the **Agreement** template, on the **General Information** tab, in the **Agreement name** box, type **Partner1 Purchasing Agreement**.
4. In the **Select self profile** section, click the **Profile name** ellipsis [**...**].
5. In the **Select My Profile** dialog box, from the **Select my profile list**, select **My Profile**, and then click **OK**.
6. In the **Agreement** template, on the **Addendums** tab, do the following:

Use this	To do this
Addendum name	Type **Partner1 Purchasing Addendum**.
Business	Click the ellipsis [**...**]. In the **Select Relationship** dialog box, select

relationship name	**OrderProcess.PurchasingType**, and then click **OK**.
MinPOAmount	Type **100**.
DiscountRate	Type **10**.

7. Click **Submit Agreement**.

8. In the agreement submitted successfully message box, click **OK**.

To activate the partner agreement

1. On the **Business Activity Services** site home page, click **Agreements**.

2. On the **Agreements** page, point to **Partner1 Purchasing Agreement**, and from the drop-down list, select **Activate Agreement**.

3. In the **Activate Agreement** template, on the **General Information** tab, do the following:

Use this	**To do this**
NegativeAck Operation port	Click the ellipsis [**...**]. In the **Select Port** dialog box, in the **Select port** list, select **Partner1DestinationPort**, and then click **OK**.
Confirmation Operation port	Click the ellipsis [**...**].In the **Select Port** dialog box, in the **Select port** list, select **Partner1DestinationPort**, and then click **OK**.

4. Click **Submit Mapping**.

5. In the agreement mapping submitted successfully message box, click **OK**.

Submitting the mapping for the agreement activates the agreement. Activating the agreement writes the agreement and profile data, including port information, to the BizTalk Management database. After you submit the agreement mapping, you can access the agreement and profile data in both the BizTalk Administration console and BizTalk Explorer.

Step 5: Configure and Start the Application

Purpose: You bind the order process orchestration to the HTTP receive port and the STS.Outbox receive port, and you start the order process orchestration.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 4: Create and Activate the Partner Agreement before you begin this step.
- You must log on as a member of the BizTalk Server Administrators group.

To configure BizTalk Application 1

1. Click **Start**, point to **Program Files**, point to **Microsoft BizTalk Server 2006**, and then click **BizTalk Server Administration**.
2. In the console tree on the left side of the BizTalk Administration console, expand **BizTalk Server 2006 Administration**, right-click **BizTalk Group [<computer name>:BizTalkMgmtDb]**, and then click **Refresh**.
3. Expand **BizTalk Group [<computer name>:BizTalkMgmtDb]**, expand **Applications**, right-click **BizTalk Application 1**, and then click **Configure**.
4. In the **Configure Application** dialog box, on the **OrderService** tab, do the following:

Use this	To do this
Host	From the drop-down list, select **BizTalkServerApplication**.
_Purchasing_SellerPortType row	From the **ReceivePorts** drop-down list, select **New receive port**.

5. In the **ReceivePort1 - Receive Port Properties** dialog box, on the **General** tab, do the following:

Use this	To do this
Name	Type **OrderProcessReceivePort**.
Authentication	Select the **Keep messages if authentication fails** option.

6. In the **ReceivePort1 - Receive Port Properties** dialog box, on the **Receive Locations** tab, click **New**.
7. In the **ReceiveLocation1 - Receive Location Properties** dialog box, on the **General** tab, do the following:

Use this	To do this
Name	Type **orderprocessHTTPLocation**.
Type	From the drop-down list, select **HTTP**, and then click **Configure**.

8. In the **HTTP Transport Properties** dialog box, do the following:

Use this	To do this
Virtual directory plus ISAPI extension	Type **/HTTPReceive/BTSHTTPReceive.dll?Order**.
Public address	Type **HTTP://<ServerName>/HTTPReceive/BTSHTTPReceive.dll?Order**.

9. Click **OK**.

10. In the **orderprocessHTTPLocation - Receive Location Properties** dialog box, on the **General** tab, do the following:

Use this	To do this
Receive Handler	From the drop-down list, select **BizTalkServerIsolatedHost**.
Receive Pipeline	From the drop-down list, select **XMLReceive [Microsoft.BizTalk.DefaultPipelines.XMLReceive, Microsoft.BizTalk.DefaultPipelines, Version=3.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]**.

11. Click **OK**.

12. In the **OrderProcessReceivePort - Receive Port Properties** dialog box, click **OK.**

13. In the **Configure Application** dialog box, on the **OrderService** tab, on the **_ReceiveDocumentsFromOutbox_receiveFromOutboxPT** row, from the **ReceivePorts** drop-down list, select **Sts.Outbox**, and then click **OK.**

14. In the console tree, **expand BizTalk Application 1**, and click **Orchestrations**.

15. In the orchestrations results pane, right-click **OrderProcess.OrderService**, and then click **Enlist**.

16. In the orchestrations results pane, right-click **OrderProcess.OrderService**, and then click **Start**.

17. In the console tree, expand **BizTalk Application 1**, and click **Receive Locations**.

18. In the receive locations results pane, right-click **orderprocessHTTPLocation**, and then click **Enable**.

Step 6: Test the Scenario

Purpose: The batch file simulates sending messages to the Outbox. While the messages remain in the Outbox, the message's status property updates. The messages remain in the Outbox until the orchestration sends a "receipt" to Sent Items. After the orchestration sends the receipt, the messages move from the Outbox to the partner's Sent Items folder.

Prerequisites

Note the following requirements before you begin this step:

- You must complete Step 5: Configure and Start the Application before you begin this step. \
- You must log on as a member of the BizTalk Server Administrators group.

To run the scenario

1. In Windows Explorer, browse to **C:\Tutorial\Lessons\BAS\HttpPost**, and then double-click **HttpPost.sln** to open it in Visual Studio 2005.
2. In Visual Studio, on the **Build** menu, click **Configuration Manager**.
3. In the **Configuration Manager** dialog box, from the **Active Solution Configuration** drop-down list, select **Release**, and then click **Close**.
4. On the **Build** menu, click **Build Solution**.
5. Close Visual Studio 2005.
6. In Windows Explorer, browse to **C:\Tutorial\Lessons\BAS**, and then double-click **SubmitOrder.bat** to run the batch file, which posts a purchase order document to the orchestration.

To check the BAS Inbox

1. Click **Start** menu, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **Business Activity Services Site**.
2. On the **Business Activity Services** site home page, click **Partner Profiles**.
3. On the **Partner Profiles** page, in the **Partner1** row, click **Inbox.**

   The **Partner1 Inbox** page displays all documents received by the seller from Partner1. If the **Partner1 Inbox** page does not show the incoming purchase order document, you may need to refresh the browser.
4. On the **Partner1 Inbox** page, click the *<purchase order document>* to open the form.

To confirm the purchase order

1. In the **PurchaseOrder** template, in the right pane, click **Confirm Purchase Order**.

2. In the **PurchaseOrder Confirmation**, scroll down to the **Itemized List** section, in the **Confirm Quantity** box (the field has a red asterisk (*)), type **8888**, and then press TAB.

   The purchase order totals are calculated.

3. Click **Submit**.

To verify sent items

1. In Internet Explorer, type **http://<ServerName>/sites/BASSite**.

2. On the **Business Activity Services** site home page, click **Partner Profiles**.

3. On the **Partner Profiles** page, on the **Partner1** row, click **Sent Items**.

4. In Windows Explorer, browse to the **C:\Tutorial\Filedrop\Partner1** folder to verify that a confirmation message is present.

   If no message arrives in this folder, verify that the account running your BizTalk Host has sufficient permissions to this directory.

Tutorial 5: Business Activity Monitoring

In this tutorial, you set up BizTalk Server 2006 Business Activity Monitoring (BAM) to track the performance of the Order Process business solution created in Lesson 2. This tutorial walks you through the steps that a business analyst completes to identify the key Order Process business activities that need to be tracked and aggregated. You also walk through the steps that a business analyst uses to define views for different business users.

Before you begin this tutorial, you should understand fundamental BizTalk Server concepts and introductory information about BAM. For information about BAM, see Monitoring Business Activities with BAM .

In this tutorial, you create an activity and set up a view specifically for sales managers. You define the activities and views in a Microsoft Office Excel spreadsheet with the BAM Add-In that ships with BizTalk Server.

You perform the tasks of the following three roles:

- A business analyst defines the key performance indicators to monitor, and the views of the data to make available to sales managers.

- A system administrator deploys the BAM definition workbook. Deploying the workbook dynamically generates the BAM infrastructure.

- A solutions developer creates and deploys a tracking profile that maps the data in the workbook to the automated business process (orchestration) running on BizTalk Server. The tracking profile enables business users to view real-time tracking and aggregated data.

After you deploy the BAM definition and the tracking profile, you test the scenario by simulating a few purchase orders and verifying that the data you defined is available in the live data spreadsheet and in the BAM portal.

This tutorial takes a novice user approximately two hours to complete.

To begin setting up Business Activity Monitoring, go to Lesson 1: Define the Purchase Order Activity.

In This Section

- Lesson 1: Define the Purchase Order Activity
- Lesson 2: Define the Sales Manager View
- Lesson 3: Organize the Data in a PivotTable Report
- Lesson 4: Connect the BAM Definition File to BizTalk Server Data
- Lesson 5: Run the Scenario and View the Tracking Results

Lesson 1: Define the Purchase Order Activity

A Business Activity Monitoring (BAM) activity is a collection of milestones (defining date and time) and data of interest that relate to a particular business process or processing pattern. Data of interest, also known as tracking data, is data you want to monitor, such as customers, orders, and amounts. Milestones are transitions between activities in a process measured in time.

For example, an activity for tracking purchase orders might include milestones such as PO received and PO approved, and data items such as order number and product.

A business analyst who is knowledgeable about the business process and the visibility requirements for managing the process typically defines activities.

In This Section

- Step 1: Add the BAM Add-In to Microsoft Office Excel
- Step 2: Create the Purchase Order Activity
- Step 3: Define the Milestones
- Step 4: Define the Data of Interest

Step 1: Add the BAM Add-In to Microsoft Office Excel

Prerequisites

You must complete Tutorial 4: Trading Partner Management before you begin this step.

To add the BAM add-in

1. In Microsoft Office Excel, on the **Tools** menu, click **Add-Ins**.

2. In the **Add-Ins** dialog box, select the **Business Activity Monitoring** check box, and then click **OK**.

3. On the **File** menu, click **Save As**. Save the workbook as **C:\Tutorial\Lessons\BAM\SalesManagerView.xls**.

You can now create the activity. Go to Step 2: Create the Purchase Order Activity.

Step 2: Create the Purchase Order Activity

Prerequisites

You must complete Step 1: Add the BAM Add-In to Microsoft Office Excel before you begin this step.

To create the purchase order activity

1. In Microsoft Office Excel, on the **BAM** menu, click **BAM Activity**.

2. In the **Business Activity Monitoring Activity Definition** dialog box, click **New Activity**.

3. In the **New Activity** dialog box, in the **Activity name** box, type **PurchaseOrder**.

Step 3: Define the Milestones

Purpose: You define the following milestones in the purchase order activity:

- Approved
- Delivered
- Denied
- Received

Prerequisites

You must complete Step 2: Create the Purchase Order Activity before you begin this step.

To define the milestones for the purchase order activity

1. In the **New Activity** dialog box, click **New Item** to open the **New Activity Item** dialog box.
2. In the **New Activity Item** dialog box, do the following:

Use this	To do this
Item name	Type **Approved**.
Item Type	From the drop-down list, select **Business Milestones**.

3. Click **OK**.
4. Use the information in the following table to create three more milestones.

Item name	Item type
Delivered	Business Milestones
Denied	Business Milestones
Received	Business Milestones

Step 4: Define the Data of Interest

Purpose: You define the following data items in the purchase order activity:

- Customer Name
- Product
- City
- State
- Amount

Prerequisites

You must complete Step 3: Define the Milestones before you begin this step.

To define the data of interest for the purchase order activity

1. In the **New Activity** dialog box, click **New Item**.
2. In the **New Activity Item** dialog box, do the following:

Use this	To do this
Item name	Type **Customer Name**.
Item Type	From the drop-down list, select **Business Data – Text**.
Maximum Length	Leave the default value.

3. Click **OK**.
4. Use the information in the following table to create four more activity items.

Item name	Item type
Product	**Business Data – Text**
City	**Business Data – Text**
State	**Business Data – Text**
Amount	**Business Data – Decimal**

5. Click **OK**.

Lesson 2: Define the Sales Manager View

A Business Activity Monitoring (BAM) view is information from an activity intended for a particular audience. Views can contain both data and aggregations of data. For example, a view could contain all of the customers from whom you receive a purchase order, or a view could contain the average amount of all of the purchase orders you receive. When you create the sales manager view, you define what data and aggregations the sales manager will see.

In this lesson, you create a view that sales managers will use to examine information based on the data of interest and milestones you defined in the BAM activity. This view filters out information that does not need to be available to sales managers.

You create dimensions and measures that reflect the key performance indicators (KPIs) relevant to the purchase order that end users want to monitor.

A business analyst who is knowledgeable about the business process and the visibility requirements for managing the process typically defines views.

In This Section

Step 1: Create the Sales Manager View

Purpose: A BAM view defines the information from a BAM activity you want to make available to sales managers.

Prerequisites

You must complete Step 4: Define the Data of Interest before you begin this step.

To create a view

1. In the **Business Activity Monitoring Activity Definition** dialog box, click **OK** to start the Business Activity Monitoring View Wizard.
2. On the **Business Activity Monitoring View Wizard Welcome** page, click **Next**.
3. On the **BAM View** page, select the **Create a new view** option, and then click **Next.**
4. On the **New Bam View: Name and Activities** page, do the following:

Use this	To do this
View name	Type **SalesManagerView**.
PurchaseOrder	Select the **PurchaseOrder** check box. The new view references the data of interest and milestones defined in the purchase order activity.

5. Click **Next**.

6. On the **New BAM View: View Items** page, select the **Select All Items** check box, and then click **Next**.

Step 2: Create the OrderDecision Group

Purpose: The OrderDecision group contains the Approved and Denied milestones. Because purchase order approval and denial occur at the same stage in the purchase order process, grouping these milestones lets you treat them as one.

Prerequisites

You must complete Step 1: Create the Sales Manager View before you begin this step.

To create the OrderDecision group

1. On the **New BAM View: View Items** page, click **New Group**.

2. In the **New Group** dialog box, do the following:

Use this	To do this
Business milestone alias	Type **OrderDecision**.
Approved	Select this check box to include the Approved milestone in the group.
Denied	Select this check box to include the Denied milestone in the group.

3. Click **OK**.

Step 3: Create the Durations

Purpose: A duration represents the interval between milestones (milestones are points in time). The following table shows the durations that you create in this step.

Duration	Start milestone	End milestone	Description
OrderCycleTimeDur	Received	Delivered	The amount of time it takes to deliver a purchase order after it is received.
OrderFulfillmentDur	Approved	Delivered	The amount of time it takes to deliver a purchase order after it is approved.
OrderVerificationDur	Received	OrderDecision	The amount of time it takes to decide if a purchase order is approved or denied after it is received.

Prerequisites

You must complete Step 2: Create the OrderDecision Group before you begin this step.

To create the durations

1. On the **New BAM View: View Items** page, click **New Duration**.
2. In the **New Duration** dialog box, do the following:

Use this	To do this
Duration name	Type **OrderCycleTimeDur**.
Start business milestone	From the drop-down list, select **Received**.
End business milestone	From the drop-down list, select **Delivered**.
Time Resolution	From the drop-down list, select **Day**.

3. Click **OK**.
4. Use the information in the following table to create two more durations.

Duration name	Start business milestone	End business milestone	Time resolution
OrderFulfillmentDur	Approved	Delivered	Day
OrderVerificationDur	Received	OrderDecision	Day

5. Click **Next** to continue.

Step 4: Create the Data Dimensions

Purpose: Dimensions aggregate data into logical groupings. In this step, you create data dimensions that aggregate purchase order data by customer, by product, and by location, which is a combination of city and state.

Prerequisites

You must complete Step 3: Create the Durations before you begin this step.⊟To create the data dimensions

1. On the **New BAM View: Aggregation Dimensions and Measures** page, click **New Dimension**.

2. In the **New Dimension** dialog box, do the following:

Use this	To do this
Dimension name	Type **Customer**.
Dimension type	From the drop-down list, select **Data Dimension**.
Available data items	Select **Customer Name** from the list, and then click **Add**.

3. Click **OK**.

4. Use the information in the following table to create two more data dimensions.

Dimension name	Data items
Location	**PurchaseOrder.State** **PurchaseOrder.City**
Product	**PurchaseOrder.Product**

Step 5: Create the Amount Size Numeric Range Dimension

Purpose: You use the numeric range dimension to monitor high, medium, and low purchase order amounts.

Prerequisites

You must complete Step 4: Create the Data Dimensions before you begin this step.

To create a numeric range dimension

1. On the **New BAM View: Aggregation Dimensions and Measures** page, click **New Dimension**.

2. In the **New Dimension** dialog box, do the following:

Use this	To do this
Dimension name	Type **AmountSize**.
Dimension type	From the drop-down list, select **Numeric Range Dimension**.

Base data items	From the drop-down list, select **Amount**.

3. Click **New Range**.

4. In the **New Range** dialog box, do the following:

Use this	To do this
Range name	Type **Low**.
From (inclusive)	Type **0**.
To (exclusive)	Type **50**.

5. Click **OK**.

6. Use the information in the following table to create two more ranges.

Range name	From (inclusive)	To (exclusive)
Medium	**50**	**500**
High	**500**	**1000000**

7. The following figure shows the **New Dimension** dialog box with the numeric range dimension.

8. Click **OK** to save the dimension.

Step 6: Create the OrderReceivedTime Time Dimension

Purpose: You use the time dimension to determine if the purchase order is for today or for another time, based on the Approved milestone.

Prerequisites

You must complete Step 5: Create the Amount Size Numeric Range Dimension before you begin this step.

To create a time dimension

1. On the **New BAM View: Aggregation Dimensions and Measures** page, click **New Dimension**.

2. In the **New Dimension** dialog box, do the following:

Use this	To do this
Dimension name	Type **OrderReceivedTime**.
Dimension type	From the drop-down list, select **Time Dimension**.
Base business milestone	From the drop-down list, select **Received**.
Display settings	Select the **Year**, **Week**, **Day**, **Hour**, **Minute** option.

3. Click **OK** to save the dimension.

Step 7: Create the OrderProgress Progress Dimension

Purpose: You use a progress dimension to define a hierarchy of time as it relates to a process. The progress dimension aggregates data about activities still in process.

For example, you use a progress dimension to find out, at a given time, how many purchase orders the purchase order process has received, but has not yet fulfilled. In this case, you would create a stage for the process between the Received milestone and the Fulfilled milestone.

Within the progress dimension you organize the milestones and stages in relation to the first milestone you define. Progress dimension milestones are sequential: the first step is completed before the next step is started. Progress dimension stages can be completed in tandem.

The following table describes the milestones and stages in the OrderProgress progress dimension.

Milestone	Description
Received	The process begins as soon as a purchase order arrives.
Verifying	Verifying is a transient stage. Each purchase order appears in Verifying immediately after it is received.
Approved	Approved is a peer of Verifying and Denied. At any given time, a purchase order is in exactly one of the three stages.
Fulfilling	Fulfilling is a transient stage. When a purchase order is approved, it immediately appears in the fulfilling stage.
Delivered	Fulfilling and Delivered are peers because all approved purchase orders are either in the Fulfilling stage or already delivered.
Denied	Verifying, Approved, and Denied are peers because a purchase order is always in exactly one of those stages. Note that you must create the transient Verifying stage first.

When you have finished creating the milestones and stages for the OrderProcess progress dimension, the **New Dimension** dialog box will look similar to the sample in the following figure.

Prerequisites

You must complete Step 6: Create the OrderReceivedTime Time Dimension before you begin this step.

To create the progress dimension

1. On the **New BAM View: Aggregation Dimensions and Measures** page, click **New Dimension**.

2. In the **New Dimension** dialog box, do the following:

Use this	**To do this**
Dimension name	Type **OrderProgress**
Dimension type	From the drop-down list, select **Progress Dimension**

3. Click **New Milestone**.

4. In the **New Progress Milestone** dialog box, do the following:

Use this	**To do this**
Progress milestone	Type **Received**.
Business milestone	From the drop-down list, select **Received**.

5. Click **OK**.

6. Use the following information to create the remaining milestones and stages:

Select	**Click**	**Progress milestone**	**Business milestone**
Received	New Stage	Verifying	Received
Verifying	New Milestone	Approved	Approved
Approved	New Stage	Fulfilling	Approved
Fulfilling	New Milestone	Delivered	Delivered

Approved	New Milestone	Denied	Denied

7. Click **OK** to save the new progress dimension and return to the **New BAM View: Aggregation Dimensions and Measures** page.

Step 8: Create the Measures

Purpose: You use measures to define mathematical relationships within data. The following table shows the four measures you create in this step.

Measure name	**Type of measure**	**Description**
AvgCycleTimeDur	Average	The average time it takes to complete the purchase order process.
AvgFulfillmentDur	Average	The average time it takes to fulfill a purchase order.
AvgVerificationDur	Average	The average time it takes to verify a purchase order.
Count	Count	The number of purchase orders processed.

Prerequisites

You must complete Step 7: Create the OrderProgress Progress Dimension before you begin this step.

To create the measures

1. On the **New BAM View: Aggregation Dimensions and Measures** page, click **New Measure**.

2. In the **New Measure** dialog box, do the following:

Use this	**To do this**
Measure name	Type **AvgCycleTimeDur**.
Base data item	From the drop-down list, select **OrderCycleTimeDur**.
Aggregation type	Select the **Average** option.

3. Click **OK**.

4. Use the information in the following table to create three more measures.

Measure name	**Base data item**	**Aggregation type**
AvgFulfillmentDur	**OrderFulfillmentDur**	**Average**
AvgVerificationDur	**OrderVerificationDur**	**Average**
Count	**PurchaseOrder**	**Count**

5. Click **Next**.

6. On the **New BAM View: Summary** page, click **Next**.

7. On the **Completing the Business Activity Monitoring View Creation** page, click **Finish**.

8. Save the workbook as **C:\Tutorial\Lessons\BAM\SalesManagerView.xls**.

Lesson 3: Organize the Data in a PivotTable Report

A Microsoft Office Excel PivotTable report enables you to create new views of worksheet data. PivotTable reports organize and summarize data, offer comparisons, reveal patterns and relationships, and analyze trends.

In this lesson, you walk through the steps a business analyst would complete to define the PivotTable that contains the data that end users will analyze.

You use the dimensions and measures you created in Lesson 2 to create the key performance indicators (KPIs) in the PivotTable.

In This Section

Step 1: Create the Key Performance Indicators PivotTable Report

Step 2: Create a Sales Trends PivotTable Report

Step 1: Create the Key Performance Indicators PivotTable Report

Purpose: The PivotTable shows order progress based on the stage an order is in. The three stages that you defined are Verifying, Approved, and Denied.

Prerequisites

You must complete Step 8: Create the Measures before you begin this step.

To create the KPI PivotTable report

1. In Microsoft Office Excel, from the **PivotTable Field List**, drag the **OrderProgress** field onto the **Drop Row Fields Here** box in the PivotTable report.

   The following figure shows the PivotTable Field List and the PivotTable in its original form.

2. From the **PivotTable Field List**, drag **Count** to the **Drop Data Items Here** box in the PivotTable report.

   The following figure shows the PivotTable after you add the **Count** measure.

3. In the PivotTable report, double-click **Received** to expand it.

   The following figure shows the PivotTable after you expand **Received**.

4. From the **PivotTable Field List**, drag **AmountSize** onto **Approved** in the PivotTable report.

   Your PivotTable should look like the following figure.

5. On the PivotTable toolbar, click the RTA icon (the last icon on the right) to mark the selected PivotTable report as an RTA table. . Unlike the other BizTalk Server 2006 tutorials, we expect you to complete all of the steps in Tutorial 5: Business Activity Monitoring without stopping.

Step 2: Create a Sales Trends PivotTable Report

Purpose: The PivotTable shows hourly sales trends based on the (content to be provided).

Prerequisites

You must complete Step 1: Create the Key Performance Indicators PivotTable Report before you begin this step.

To create the sales trends PivotTable report

1. In the PivotTable report, right-click the **Count** cell, point to **Select**, and then click **Entire Table**.

2. On the **Edit** menu, click **Copy** to copy the PivotTable report.

3. Place your cursor five cells below the first PivotTable report, and on the **Edit** menu, click **Paste** to create a second PivotTable report.

4. In the second PivotTable report, drag the **Level 02** and **AmountSize** columns off of the PivotTable report. Only the **Count** and **Total** columns remain.

   The following figure shows the sales trends PivotTable.

5. From the **PivotTable Field List**, drag **OrderReceivedTime** onto the **Total** row located directly under **Count**.

   The following figure shows the sales trends PivotTable with **OrderReceivedTime**.

6. From the **PivotTable Field List**, drag **Product** onto the **Total** column.

   The following figure shows the sales trends PivotTable with **Product**.

7. Double-click the year **2004** cell to show weeks on the PivotTable report.

   The following figure shows the sales trends PivotTable with weeks.

8. From the **PivotTable Field List**, drag **OrderProgress** onto the **Drop Page Fields Here** box above the second table.

   The following figure shows the sales trends PivotTable with **OrderProgress**.

9. In the second PivotTable report, in the **All OrderProgress** drop-down list, expand **All OrderProgress**, expand **Received**, select **Approved**, and then click **OK**.

10. In the second PivotTable report, do the following:

Use this	**To do this**
Week	Double-click the first value in the **Week** column to expose the **Day** cells.
Day	Double-click the first value in the **Day** column to expose the **Hour** cells.
Week	Right-click the **Week** column heading and select **Hide Levels** to hide all values except for **Hour**.

11. Right-click the PivotTable report and click **PivotChart** to open a chart.

12. Right-click the new chart, and then select **Chart Type**.

13. In the **Chart Type** dialog box, from the **Chart type** list, select **Line**, and then click **OK**.

14. Right-click the **Chart1** tab at the bottom of the screen and click **Rename** to rename the chart to **Hourly Sales Trend**.

15. Save the workbook as **C:\Tutorial\Lessons\BAM\SalesManagerView.xls**.

Lesson 4: Connect the BAM Definition File to BizTalk Server Data

In this lesson, you perform the tasks to connect the BAM definition file to data in BizTalk Server.

You use the BizTalk Server Tracking Profile Editor (TPE) to map the BAM definition file to a BizTalk orchestration. When you use the TPE to map the BAM definition file to an orchestration, you map the business milestones in the activity to the action shapes in the orchestration, and the business data in the activity to the specific elements and attributes in the orchestration schemas.

A developer uses TPE to map events by dropping items from the message schemas and orchestration shapes onto the appropriate business milestone (event) and data item folders. When the mapping of the order activity to the orchestration schedule is complete, the developer can deploy the tracking profile to the BizTalk Management database as well as save a BizTalk Server tracking (.btt) file if desired.

In This Section

Step 1: Deploy the Workbook

Step 2: Open the Assembly in TPE

Step 3: Import the Activity Definition

Step 4: Map the Events

Step 5: Deploy the Tracking Profile

Step 1: Deploy the Workbook

Purpose: The SalesManagerView.xls workbook is the BAM definition file because it contains a BAM activity, view, and PivotTable reports. The BAM Definition file defines the data to track and aggregate, as well as the business user's view of the tracking data.

When you deploy the BAM definition file, the BAM deployment process uses the BAM definition file and the BAM configuration .xml file to dynamically generate the BAM database infrastructure based on the activity defined in the workbook.

In addition to dynamically creating the BAM data infrastructure, the deployment process generates a live data version of the SalesManagerView.xls workbook, named SalesManagerView_LiveData.xls. The live data workbook appears in the same folder as SalesManagerView.xls.

Prerequisites

You must complete Step 2: Create a Sales Trends PivotTable Report before you begin this step.

To deploy the workbook

1. Copy **SalesManagerView.xls** to **C:\Program Files\Microsoft BizTalk Server 2006\Tracking**.

2. Open a Command Prompt window and type:

   CD C:\Program Files\Microsoft BizTalk Server 2006\Tracking

3. Press ENTER, and then type:

   BM deploy-all -DefinitionFile:SalesManagerView.xls

4. Press ENTER. When the deployment process is complete, you should see the following at the bottom of the Command Prompt window:

5. Type **exit** and then press ENTER to close the Command Prompt window.

Step 2: Open the Assembly in TPE

Purpose: Tracking profiles map a specific view of internal business activities and associated data to an orchestration.

Prerequisites

You must complete Step 1: Deploy the Workbook before you begin this step.

To open the OrderProcess assembly in Tracking Profile Editor

1. Click **Start**, point to **All Programs**, point to **Microsoft BizTalk Server 2006**, and then click **Tracking Profile Editor**.

2. On the **File** menu, click **New**.

3. Click **Click here to select an event source**.

4. In the **Select Event Source Parent Assembly** dialog box, from the **Assembly Name** list, select **OrderProcess**, and then click **Next**.

5. In the **Select Orchestration** dialog box, from the **Orchestration Name** list, select **OrderProcess.OrderService**, and then click **OK**.

Step 3: Import the Activity Definition

Purpose: The BAM activity definition defines the business milestones and data to collect.

Prerequisites

You must complete Step 2: Open the Assembly in TPE before you begin this step.

To import the BAM definition

1. In Tracking Profile Editor, click **Click here to import a BAM Activity Definition**.

2. In the **Import BAM Activity Definition** dialog box, from the **BAM Activity Definition Name** list, select **PurchaseOrder**, and then click **OK.**

Step 4: Map the Events

Prerequisites

You must complete Step 3: Import the Activity Definition before you begin this step.

To map the events

1. In Tracking Profile Editor, in the Assembly view, do the following:

Use this	To do this
Receive PO	Drag onto the **Received** node in the PurchaseOrder schema.
Send NAK	Drag onto the **Denied** node in the PurchaseOrder schema.
Send PO to Inbox	Drag onto the **Approved** node in the PurchaseOrder schema.
Send Confirmation to Partner	Drag onto the **Delivered** node in the PurchaseOrder schema.
Receive PO	Right-click and select **Message Payload Schema**.

2. In the **OrderProcess.PurchaseOrder** schema, expand **<Schema>**, expand **PurchaseOrder**, expand **Header**, and then drag **ID** onto the **ActivityID** node in the PurchaseOrder schema.

3. In the **OrderProcess.PurchaseOrder** schema, expand **<Schema>**, expand **PurchaseOrder**, expand **orderBody**, expand **order**, expand **items**, expand **item**, and then drag **description** onto the **Product** node in the PurchaseOrder schema.

4. In the **OrderProcess.PurchaseOrder** schema, expand **<Schema>**, expand **PurchaseOrder**, expand **orderBody**, expand **order**, expand **actualTotal**, and then drag **total** onto the **Amount** node in the PurchaseOrder schema.

5. In the **OrderProcess.PurchaseOrder** schema, expand **<Schema>**, expand **PurchaseOrder**, expand **orderBody**, expand **submittedBy**, expand **<Equivalent>**, expand **<contactWithCompanyType>**, expand the first **<Sequence>**, expand **address**, and then drag **stateProvince** onto the **State** node in the PurchaseOrder schema.

6. In the **OrderProcess.PurchaseOrder** schema, expand **<Schema>**, expand **PurchaseOrder**, expand **orderBody**, expand **submittedBy**, expand **<Equivalent>**, expand **<contactWithCompanyType>**, expand the first **<Sequence>**, expand **name**, and then drag **singleName** onto the **CustomerName** node in the PurchaseOrder schema.

7. In the **OrderProcess.PurchaseOrder** schema, expand **<Schema>**, expand **PurchaseOrder**, expand **orderBody**, expand **submittedBy**, expand **<Equivalent>**, expand **<contactWithCompanyType>**, expand the first **<Sequence>**, expand **address**, and then drag **city** onto the **City** node in the PurchaseOrder schema.

8. On the **File** menu, click **Save As** and name the file **C:\Tutorial\Lessons\BAM\OrderProcess.btt**.

9. Close Tracking Profile Editor.

Step 5: Deploy the Tracking Profile

Purpose: After the tracking profile is created, saved, deployed, and tested, a business analyst can retrieve the data collected to analyze the business activities of interest.

Prerequisites

You must complete Step 4: Map the Events before you begin this step.

To deploy the tracking profile from the command-line utility

1. Copy **OrderProcess.btt** to **C:\Program Files\Microsoft BizTalk Server 2006\Tracking**.

2. Open a Command Prompt window and type:

   CD C:\Program Files\Microsoft BizTalk Server 2006\Tracking

3. Press ENTER, and then type:

   bttdeploy OrderProcess.btt

4. Press ENTER.

5. At the command prompt, type **exit**, and then press ENTER.

Lesson 5: Run the Scenario and View the Tracking Results

Run ten purchase orders through the OrderProcess solution created in the previous lesson. Examine the real-time aggregated data using the Business Activity Monitoring (BAM) Excel live data workbook that you created during deployment. Then drill into the different individual cases and look at the related business documents like the PO or confirmation using the business activity search in the Business Activity Services (BAS) site.

When you are looking at the live data sheet the real-time aggregated information is changing in real time as the ten orders flow through the business process.

In This Section

Step 1: Run the Scenario

Step 2: View Aggregated Data in Excel

Step 3: Grant Permission to the View

Step 4: View the Data in the Business Activity Monitoring Portal

Step 1: Run the Scenario

Purpose: In this step, you confirm the orders by dropping confirmation files to the outbox.

Prerequisites

You must complete Step 5: Deploy the Tracking Profile before you begin this step.

To run the scenario

1. In Windows Explorer, navigate to **C:\Tutorial\Lessons\BAS** and double-click **SubmitOrderBatch.bat**.

2. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **Business Activity Services Site**.

3. On the **Business Activity Services** site home page, click **BizTalk Servers**.

4. On the **BizTalk Servers** page, click **Outbox**.

5. On the **BizTalk Integration Server_Outbox** page, click **Upload Document**.

6. On the **BizTalk Integration Server_Outbox: Upload Document** page, click **Upload Multiple Files**.

7. Browse to **C:\Tutorial\Lessons\BAS\Messages**, select all ten Confirmation messages, and then click **Save and Close**.

8. Click **Yes** to confirm the file upload.

   This simulates the processing of ten purchase orders. You must run the DTS package to process the purchase order data.

To run the DTS package to process the OLAP cube

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.

2. Expand **Microsoft SQL Servers**, expand **SQL Server Group**, expand the local server, expand **Data Transformation Services**, and then click **Local Packages**.

3. In the display pane, right-click **BAM_AN_SalesManagerView**, and then click **Execute Package**.

4. When the package has finished running, click **OK**, and then click **Done**.

Step 2: View Aggregated Data in Excel

Prerequisites

You must complete Step 1: Run the Scenario before you begin this step.

To view the aggregated data in a live data Excel template

1. In Windows Explorer, browse to **C:\Program Files\Microsoft BizTalk Server 2006\tracking** and double-click **SalesManagerView_LiveData.xls**

2. As the orders are flowing through the business process, refresh the PivotTable report to see the real-time aggregations change.

3. The first PivotTable report shows the real-time aggregation data while the second PivotTable report shows data aggregated during the execution of the DTS task.

Step 3: Grant Permission to the View

Purpose: Before you can see the SalesManagerView data in the BAM portal, you must have permission to access the view.

Prerequisites

You must complete Step 2: View Aggregated Data in Excel before you begin this step.

To grant permission to the BAM view

1. Open a Command Prompt window and type:

   CD C:\Program Files\Microsoft BizTalk Server 2006\Tracking

2. Press ENTER, and then type:

   BM add-account -AccountName:<account name> -View:SalesManagerView

3. Press ENTER. When the add process is complete, you should see the following at the bottom of the Command Prompt window:

   '<account name>' was successfully added to the view role for 'SalesManagerView'.

Step 4: View the Data in the Business Activity Monitoring Portal

Purpose: You use the BAM portal to access BAM views.

Prerequisites

You must complete Step 3: Grant Permission to the View before you begin this step.

To view the data in the BAM portal

1. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, and then click **BAM Portal Web Site**.

2. On the BAM portal **Home** page, expand **Activity Search**, and then click **PurchaseOrder**.

3. On the **Activity Search: PurchaseOrder** page, do the following:

Use this	To do this
Business Data	From the drop-down list, select **Customer Name**.
Operator	From the drop-down list, select **Is not empty**.
Available Data and Milestones	From the list, move all of the items to the **Items to show list**.

4. Click **Execute Query**.

   The query results appear at the bottom of the page in the Results section. Use the data to create more queries.

Getting BizTalk Server 2006 Assistance

There are three principal sources of information from Microsoft about BizTalk Server 2006:

- The documentation, tutorials, and samples installed with BizTalk Server.
- The BizTalk Server sites on the Microsoft Developer Network (MSDN) and TechNet.
- BizTalk Server information on additional Microsoft Web sites.

You can also get help from others either through the BizTalk Server community or directly from Microsoft support.

BizTalk Server Documentation, Tutorials, and Samples

When you install BizTalk Server, you can install the BizTalk Server documentation, sample applications, and tutorials. The core documentation, samples, and tutorials cover the concepts and procedures required to use BizTalk Server effectively. BizTalk Server help also includes reference material for the languages and programming interfaces for building applications using BizTalk Server.

If you install the documentation and tutorials using the BizTalk Server 2006 Setup program, you can access them in two ways:

- Through the **Microsoft BizTalk Server 2006** program group on the start menu.
- Through F1 Help and Dynamic Help in the BizTalk Server tools and utilities.

BizTalk Server 2006 documentation will also be made available on the Web, both as Web pages as well as in downloadable form, giving you access to the documentation from a computer on which BizTalk Server is not installed.

Information on MSDN and TechNet

The Microsoft Developer Network (MSDN) provides online and offline services that help developers write applications using Microsoft products and technologies. The Microsoft TechNet Program delivers comprehensive technical resources that help IT professionals efficiently evaluate, deploy, and support Microsoft solutions.

Several sites on MSDN and TechNet provide a specific audience with links to the BizTalk Server information most relevant to their jobs.

The following table links to these resources.

Resource	**Location**
BizTalk Server Developer Center	http://go.microsoft.com/fwlink/?LinkId=49339
The BizTalk Server TechNet Resource Site	http://go.microsoft.com/fwlink/?LinkId=49957

Additional Online Information

Additional BizTalk Server 2006 information is available from these Microsoft Web sites.

Resource	Description
Microsoft BizTalk Server Homepage	Provides pre-sales information about evaluating BizTalk Server as a data storage and business intelligence tool.
Microsoft Knowledge Base	Provides a searchable repository of BizTalk Server articles written by Microsoft Support.
Microsoft Learning	Describes Microsoft training and certifications for BizTalk Server.
Microsoft Press	Lists Microsoft Press books about BizTalk Server 2006.

Getting Assistance From Others

If you have not found the information you are looking for in the product documentation or on the Web, you can either ask a question in the BizTalk Server community or request help from Microsoft support.

The following table links to these resources.

Resource	Location
BizTalk Server Community Portal	http://go.microsoft.com/fwlink/?LinkId=49340
Microsoft BizTalk Bloggers	http://go.microsoft.com/fwlink/?LinkId=49361
BizTalk Server Newsgroups	http://go.microsoft.com/fwlink/?LinkId=49362
BizTalk Server Webcasts	http://go.microsoft.com/fwlink/?LinkId=49363
BizTalk Server on GotDotNet	http://go.microsoft.com/fwlink/?LinkId=49364
BizTalk Server Support Center	http://go.microsoft.com/fwlink/?LinkId=49366
BizTalk Server Partners	http://go.microsoft.com/fwlink/?LinkId=49338

Planning and Architecture

This section provides information about BizTalk Server to help you plan your environment.

In This Section

- BizTalk Server Architecture
- Planning for Sustained Performance
- Performance Differences from BizTalk Server 2006
- Performance Tips and Tricks
- Planning for High Availability
- Planning for Disaster Recovery
- Optimizing Resource Usage Through Host Throttling
- Business Solutions Scenarios
- How BizTalk Server Processes Large Messages

BizTalk Server Architecture

This section provides detailed information on the architecture and internal workings of BizTalk Server 2006.

In This Section

- Messaging Overview
- Runtime Architecture
- Management and Tracking Architecture
- Business Activity Services Architecture

Messaging Overview

BizTalk Server 2006 is, at its core, a message-handling engine. To understand the details of BizTalk Server 2006, it is important to understand messages and how they are represented, stored, and processed by BizTalk Server. The details of how BizTalk Server works with messages become easier to understand after you have gained an understanding of what a message is.

In This Section

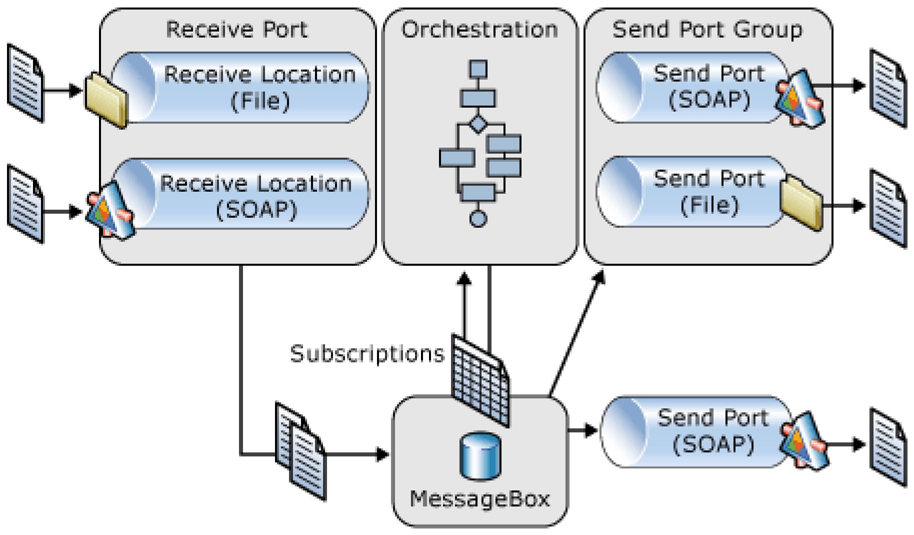
The BizTalk Server Message

Each message in BizTalk Server is considered a multi-part message and is made up of zero or more parts. Each part consists of a binary chunk of data which can represent an XML document, flat file, a serialized .NET class, or other binary stream of data. Each message with one or more parts has one of these parts identified as the body part. You use the body part of the message to identify the type of the message that can be used for routing.

A very important concept to understand is that all messages are immutable in BizTalk Server 2006. This means that after a message is constructed, it cannot be changed. Any changes to the message require a new message to be created and used from that point forward. This is especially clear in the Orchestration Designer, where compilation rules force you to follow strict guidelines about constructing a message before using it, and not allowing the message to be altered outside of its construction block. If you need to change the message, you must create a new construction block that creates a message of the same type, copy the original message to the new message, and then make any changes to the new message before leaving the construction block.

Lifecycle of a Message

The following figure provides a high-level overview of the BizTalk Server architecture from a messaging perspective.



In this simplified view, a message is received through a receive location defined in a given receive port. This message is processed by the receive location and then published to the MessageBox database, the main persistence and routing mechanism for BizTalk Server. The MessageBox evaluates active subscriptions and routes the message to those orchestrations and send ports with matching subscriptions.

Components Involved

The following are key components involved in BizTalk Server message processing.

Receive Ports and Receive Locations

A receive port is a collection of one or more receive locations that define specific entry points into BizTalk Server. A receive location is a combination of a receive adapter and a receive pipeline. The adapter is responsible for the transport and communications part of receiving a message. Examples include the File adapter and SOAP adapter, each of which receives messages from different types of sources. The receive pipeline is responsible for preparing the message for publishing into the MessageBox. A pipeline is a series of components that are executed in sequence, each providing specific processing to a message such as decryption/encryption, parsing, or validation. For more information on pipelines, receive ports, and receive locations, see Artifacts.

Send Ports and Send Port Groups

A send port group is a collection of send ports and works much like an e-mail distribution list. A message sent to a send port group will be sent to all send ports in that group. A send port is the combination of a send pipeline and a send adapter. The send pipeline is responsible for preparing a message coming from BizTalk Server for transmission to another service. The send adapter is responsible for actually sending the message using a specific protocol such as SOAP, or FTP. For more information on send ports and send port groups, see Artifacts.

MessageBox Database

The heart of the publish/subscribe engine in BizTalk Server is the MessageBox database. The MessageBox is made up of two components: one or more Microsoft® SQL Server™ databases and the Messaging Agent. The SQL Server database provides the persistence store for many things including messages, message parts, message properties, subscriptions, orchestration state, tracking data, host queues for routing, and others.

In addition, using stored procedures, the database provides some of the business logic related to routing messages and fulfilling subscriptions. The Message Agent, however, is the component that encapsulates and abstracts the database component and is the interface used by all parts of BizTalk Server to interact with the messaging subsystem. The Message Agent is a Component Object Model (COM) component that provides interfaces for publishing messages, subscribing to messages, retrieving messages, and so on. This interface is the only mechanism used by other BizTalk Server components, including the adapter framework and orchestrations, to interact with the MessageBox. For more information on the MessageBox database, see The MessageBox Database.

Endpoint Manager

The Endpoint Manager, not shown in the figure, is the service responsible for managing send and receive ports and acts as the intermediary between the ports and the MessageBox.

Hosts and Host Instances

A host is a logical representation of an application that will host services that provide specific functionality in BizTalk Server. A host can be either an in-process host, which means it is owned and managed by BizTalk Server, or an isolated host, which means that the BizTalk Server code is running in a process that is not controlled by BizTalk Server. A good example of an isolated host is Internet Information Services (IIS), which hosts the receive functionality of the HTTP and SOAP adapters. Hosts are defined for an entire BizTalk Server group; a collection of BizTalk Servers that share configuration, MessageBoxes, ports, and so on. For more information on hosts and host instances, see Topology.

Message Properties

In addition to the parts that make up a message, each message in the system has a set of properties that go along with it in what is known as the message context. These properties can be values extracted from the message itself or values related to the message or the processing of the message. For example, adapters put properties into the context related to the receiving of messages such as the location at which the message was received, and what type of adapter was used to receive the message. Properties can either be written to the context, or promoted into the context. The difference between these two options is that promoted properties can be used as criteria in message routing while written properties cannot.

This concept of writing or promoting values to the context is related to, but not the same as, promoting properties in the BizTalk Editor. In the BizTalk Editor, an element or attribute in a schema can be flagged as a promoted property or a distinguished field. Items that are marked as promoted properties are meant to be promoted into the context, while those marked as distinguished fields are meant to be written into the context.

The design for promoted properties started with the design of message correlation: the ability to relate a message being received to an already running orchestration instance. For correlation, there is a need to define a property or set of properties that provide the link between messages in the orchestration. For example, in a purchasing process, there exists a need to correlate messages based on the PurchaseOrderID. However, in many business cases, the name of the particular field or attribute in the messages may not match. A purchase order schema might have an element named POId, while the companion invoice schema may have an element named OrderID. To correlate messages on named property such as PurchaseOrderID in this situation, the developer must be able to abstract the name of the property to be correlated on from the source of the value. Property schemas allow this abstraction.

A property schema enables you to define promoted properties in a common location and have them referenced by other schemas. Like other schemas, a property schema has a namespace, but unlike other schemas, it can only have defined elements (that is, not records or attributes). Each element that is defined in the property schema has a name and type. Because the property schema may need to be referenced by more than one schema, and because the information in the property schema must be available to components at runtime, the property schema must get deployed to BizTalk

Server like all other schemas. In addition to the normal schema deployment steps, information about the promoted properties is extracted and stored in the bts_documentSpec table in the Management database.

After a property schema has been created, elements and attributes with the same type (for example. integer) can be promoted as one of the named properties in the property schema.

To complete the example case from above, a developer would perform the following steps to define the shared property needed for correlation.

1. Create a property schema and define an element of type xs:int named PurchaseOrderId.

2. Create a PurchaseOrder schema and add an element or attribute of type xs:int named POId.

3. Using the show promotions command in the BizTalk editor, the developer adds the POId field to the list of promoted properties and indicates that is should be promoted as the PurchaseOrderId property defined in the property schema by selecting the PurchaseOrderId named property from the list.

4. Create an Invoice schema and add an element or attribute of type xs:int named OrderId.

5. Using the show promotions command in the BizTalk Editor, the developer adds the OrderId field to the list of promoted properties and indicates that is should be promoted as the PurchaseOrderId property defined in the property schema by selecting the PurchaseOrderId named property from the list.

Now that this definition exists in the document schemas, pipeline components can properly promote OrderId and POId as the named property PurchaseOrderID so that it can be used for routing and correlation. For more details on this promotion process, see the topic "Message Processing" in this document.

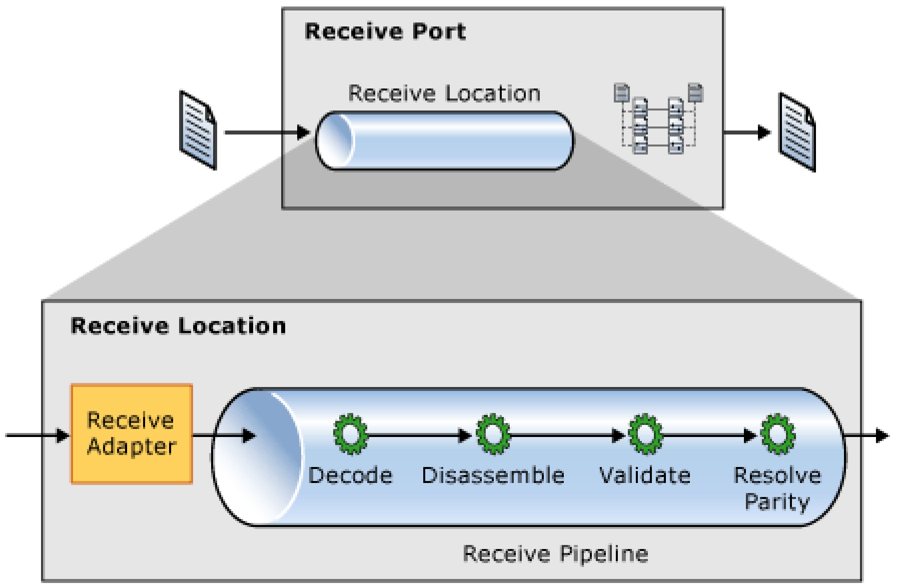
One of the benefits of promoted properties is that the value of the element that is promoted is available in the context of the message. This means that retrieving that value is inexpensive, as it does not require loading the message into memory to execute an XPath statement on the message. Instead, a simple property bag can be used along with a key to get the value. This type of behavior is desirable in situations other than message routing and is the reason for creating distinguished fields. While promoted properties are promoted into the message context, distinguished fields are written into the message context. Unlike promoted properties however, there is no property schema for distinguished fields. This is why distinguished fields cannot be used for routing and are therefore not available as filter criteria in a send port or orchestration receive shape. Distinguished fields can, however, be used in orchestrations to read or write values from the message context instead of having to load the message into memory and extract the value.

In addition to promoting or writing properties into the message context, message predicates can also be added to the context. Message predicates are used as a security measure in BizTalk Server and provide contextual information about the message, which must match values specified for any host that the message is to be routed to. This security measure allows you to configure your BizTalk Server environment in such a way as to allow specific hosts to be the only hosts that can receive and process particular messages. As an example, in the BizTalk Server Administration console, a host can be configured with the thumbprint of a certificate to use for message decoding and decryption. Configuring this property creates an application property for that host in the MessageBox. Secure messages that are received and decrypted using this thumbprint are then only routed to the hosts with the thumbprint configured.

Processing the Message

All of the components described so far play a part in the processing of messages as they flow through BizTalk Server. This section provides more detail about how these components interact to make the messaging infrastructure work, beginning with receiving a message. The figure below shows the make-up of a receive port and the flow of a message through the receive process.

A receive port consists of one or more receive locations and zero or more maps. Maps are XSLT-style sheets used to transform messages from one structure or format to another and are often used in the receive process to normalize messages into an internal format. A message is received into a receive port by a single receive location. The receive location consists of the receive adapter and a receive pipeline.



The receive adapter initiates the process of receiving messages by reading a stream of data and creating a message. For example, the file adapter sees that a file has been placed in its configured location and reads that file in a stream. The adapter creates a message (an implementation of the Microsoft.BizTalk.Message.Interop.IBaseMessage interface), adds a part to it (an implementation of the Microsoft.BizTalk.Message.Interop.IBasePart interface), and provides the stream of data as the part content.

In addition, the adapter writes and promotes properties into the message context related to the location, adapter type, and other adapter specific properties. After the message and its context have been created, the adapter passes the message to the Transport Proxy, which is also managed by the Endpoint Manager. The message is then processed through the receive pipeline, which has been configured for the receive location. After the message has been processed by the pipeline, a map may be used to transform the message into the format desired before the Endpoint Manager passes the message to the Message Agent for publishing.

While it is the responsibility of the adapter to create the initial message, most of the processing that occurs on a received message happens in the receive pipeline. Pipeline processing deals with message content as well as message context. Message content is generally handled in decoding, disassembling and validating stages, while message context can be handled in all stages. A pipeline, however, does not have to impact either the content or the context; for example, the default pass-through pipeline has no components configured and performs no processing on the message content or context. For simplicity, this document focuses on the disassembling components as they generally have the greatest impact on message routing.

The job of the disassembler is to process an incoming message from an adapter and to disassemble it into many messages, and parse the message data. When an incoming message has many smaller messages, this is known as an interchange. Both the flat file disassembler and the XML disassembler handle interchanges by enabling a developer to configure information about the wrapping content (a header and trailing schema for the flat file disassembler and an envelope schema for the XML disassembler) and the, potentially repeating, body content. In addition, both of these disassemblers parse the original message into XML content. A custom disassembler need not necessarily parse the content into XML if further XML processing in BizTalk Server is not required. An example scenario might include a simple routing situation in which messages entering the system at a particular receive location are sent to a specific send port with no mapping or other XML-based processing.

One of the most common message properties used in routing is the message type. When a developer creates a schema to define the structure of messages, this schema defines the message type for that message. The type is determined by the root node and namespace in the schema definition. For example, an XML document that looks like the following would have a message type of http://tempuri.org/samples/MessageType#Message

Notice that the message type consists of the namespace, followed by the pound (#) symbol, and the name of the root node. To use message type in routing, it must be promoted into the context. Disassemblers are generally responsible for promoting this value into the message context as well as the pipeline components with the most specific knowledge of message structure. The XML and Flat File disassemblers promote the message type as they are processing messages, and any custom disassembler should also promote this property to ensure proper routing.

It is important to note that a message is not required to have a type. As mentioned previously, the parts of a message can be any binary data and need not have a schema that defines their structure. This type of message part is generally passed through BizTalk Server without much, if any, processing done on it by BizTalk Server itself, though custom pipeline components, adapters, or code called from orchestrations may interact with these parts.

Pipeline components, like adapters, also write and promote properties into the message context. In fact, pipeline components are the most common mechanism most developers use to get properties into the message context. As discussed in the topic "Messages and Message Properties," developers create schemas and can promote properties in the schema or mark them as distinguished fields. This information is stored in the schema as annotations which can then be used by pipeline components. All of the built-in disassembler and assembler components - FlatFile, XML, and BizTalk Framework - use the document schema to retrieve information about the properties that are to be promoted. Using the XPath statement from the annotations, the disassembler knows the location in the document of elements to be promoted. During the process of streaming through the document, the disassembler finds those elements that match one of the XPath statements and promotes or writes the value into the context as appropriate.

Custom pipeline components can also be written to handle getting properties into the context for arbitrary data in a received or sent message. In order to promote a property into the context and have it be useful for routing, which is presumably why the value is being promoted, a property schema with a definition for the property should be created and deployed to BizTalk Server. Before defining a property schema to be used by custom components, you should understand the different types of promoted properties. Promoted properties defined in a property schema can have one of two base types: MessageContextPropertyBase or MessageDataPropertyBase.

A property with a base type of MessageDataPropertyBase indicates that the value for this property comes from the content of the message. This is the default value for properties defined in a property schema and is the most common usage. MessageContextPropertyBase indicates a property that is intended to be part of the message context but does not necessarily come from the message data directly. Properties with MessageContextPropertyBase as their base type are often promoted by adapters and disassemblers and include common properties such as message type and adapter type.

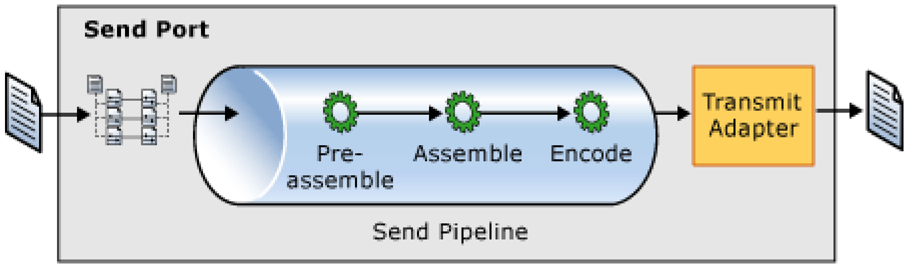
It is important to understand the different types and use them appropriately when defining properties. One of the most significant implications occurs when acessing context properties for a message in an orchestration. If a property is identified as a MessageDataPropertyBase, the orchestration designer examines the schema of the message being received and ensures that it defines a matching promoted property. If no property is found in the schema tied to the promoted property being accessed, then the designer does not allow you to access it. On the other hand, if the property is defined as being a MessageContextPropertyBase, the message type does not matter and the property can be accessed.

In custom pipelines, the mechanism for promoting or writing properties to the context is very similar. For writing properties, you use a call to the IBaseMessageContext.Write method to place the value in the context. For promoted properties, you simply use the IBaseMessageContext.Promote method instead. Each of these methods takes a property name, namespace and value. For the promoted properties, the name and namespace are those of the property defined in the property schema and are most easily accessed by referencing the property schema assembly and using the properties on the class created for the property. Distinguished fields use a common namespace, http://schemas.microsoft.com/BizTalk/2003/btsDistinguishedFields, and the XPath expression used to retrieve the value is usually used as the name.

The code below shows an example of both promoting and writing properties into the context. Note that in this example, a distinguished field is being written into the context. This is only useful for orchestrations in which the message schema identifies the distinguished field so the orchestration designer knows about the field. It may be useful to write properties into the context for use by other pipeline components on the receiving or sending side.

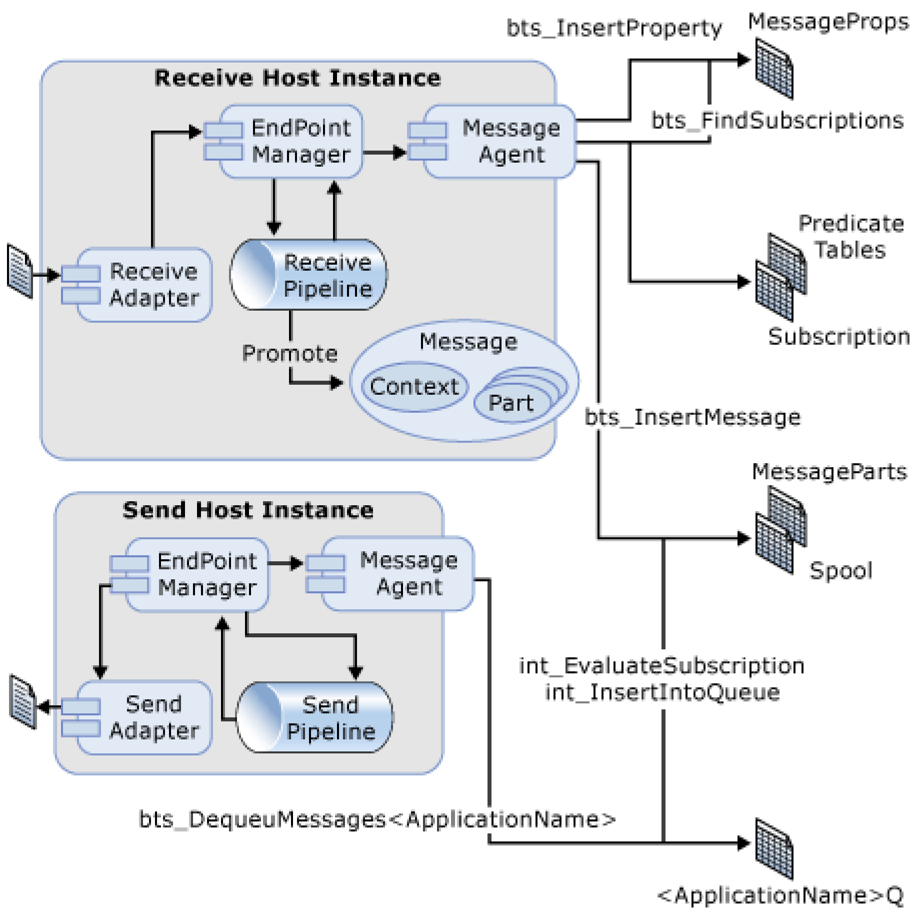
There are several facts to keep in mind when writing or promoting values into the context. First, writing a value into the context with the same name and namespace that were used previously to promote the property causes that property to no longer be promoted. The write essentially overwrites the promotion. Next, writing a value of null into the context deletes the value, because null-valued properties are not permitted. Finally, promoted properties are limited to 256 characters in length while written properties have no length limitation. Promoted properties are used in message routing and are limited in size for reasons of efficiency in comparison and storage. While written properties have no hard limits on size, using excessively large values in the context will have an impact on performance, because those values must still be processed and passed with the message.

When a message is ready to be sent from BizTalk Server, it undergoes a complementary process in the send port. Maps are applied to messages before the send pipeline is executed, allowing a message to be transformed to a customer or application-specific format before being processed by the pipeline and sent through the adapter. In the send pipeline, instead of promoting properties into the message context, properties are demoted from the context into the message. This only occurs for MessageDataPropertyBase properties because they are the only type of promoted properties that affect message data.

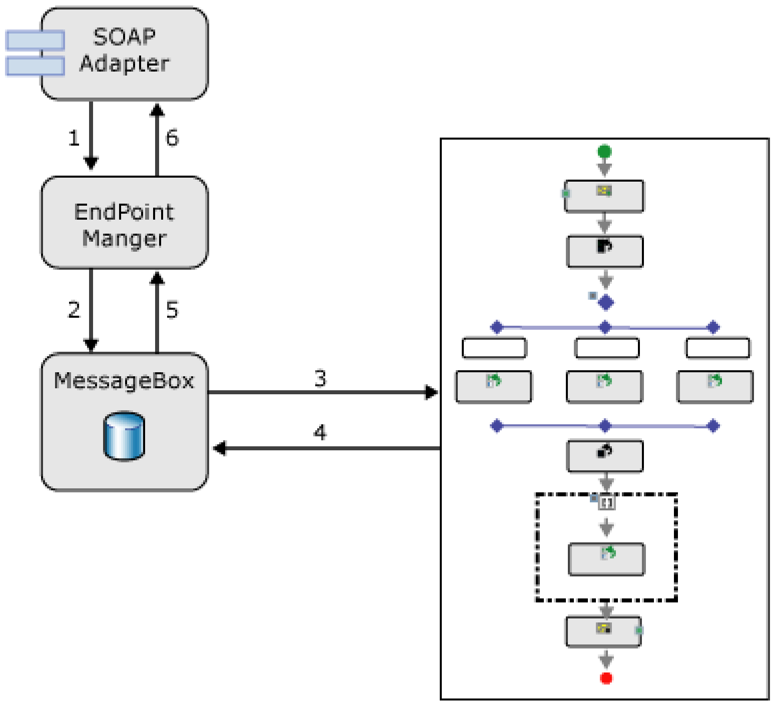


Request / Response Messaging

In a request/response messaging pattern, one party sends a request message and the receiving party returns a response message. Two typical examples of request/response processing are the interaction that a browser has with a web server using the HTTP protocol, and web service processing using the SOAP protocol. In BizTalk Server 2006, both the request and the response messages are handled in a typical publish/subscribe fashion. This is an important consideration to understand for performance tuning a BizTalk application as a system requiring high throughput may be configured differently than one requiring low latency for individual messages.



When a message is received by a request/response style receive adapter, BizTalk first publishes the request message to the MessageBox. Next this message is received by the appropriate subscriber, which is likely an orchestration bound to a receive port. This subscriber formulates a response message and publishes it to the MessageBox along with properties that cause it to be sent back to the receive port from which the request came. Finally, the response message is picked up by the publisher of the request, the receive adapter which submitted the request, and is returned to the calling application. The diagram below provides a detailed graphical representation of these steps.



Flow of request/response message received by SOAP adapter

1. The SOAP adapter submits messages to the Endpoint Manager.

2. The Endpoint Manager uses the Message Agent API to insert the message into the MessageBox.

3. The orchestration, which is bound to the receive port and therefore has a subscription for the message, receives the message and processes it.

4. The orchestration sends a response message that is published to the MessageBox.

5. The Endpoint Manager receives the response message.

6. The Endpoint Manager returns the response to the SOAP adapter

The implications of this type of behavior on performance can be overlooked if the internal implementation is not understood. BizTalk Server is initially tuned for high throughput scenarios but may be configured for an environment with lower throughput and a need for lower latency, especially in request/response scenarios. Several components need to be considered for tuning in this scenario. First, subscribers find out about published messages through a polling mechanism. If the polling interval is set too high, this could cause request/response style interactions to have a higher latency than desired.

Note that in this scenario, there are two subscriptions to be filled: the subscription for the initial message, as well as the one for the response message, and this increases the impact of this polling interval. Second, receive adapters are configured to insert messages into the MessageBox in batches of varying sizes. Most adapters enable you to configure the batch size through the typical adapter configuration UI or through parameters in BizTalk Server or the registry. If the batch size is set too high, the latency for individual messages may be increased. For more information about performance characteristics of BizTalk Server 2006 and how to tune for low latency scenarios, see BizTalk Server 2006 Performance Characteristics.

Runtime Architecture

Before reviewing more detailed information about the various components in BizTalk Server, it is important to have an understanding of how the components fit into the overall architecture of the product. The BizTalk Server runtime is built on a publish/subscribe architecture in which a message is published into the system, and then received by one or more active subscribers. Different flavors of this architecture exist, but the model implemented in BizTalk Server is often called content-based publish/subscribe.

In a content-based publish/subscribe model, subscribers specify the messages they want to receive using a set of criteria about the message. The message is evaluated at the time it is published, and all of the active subscribers with matching subscriptions (indicated by filter expressions), receive the message. In the case of BizTalk Server, content-based is a bit of a misnomer, however, because the criteria used to build subscriptions do not have to come from the message content and may include contextual information about the message as well. For details of the subscription mechanism, see Publish and Subscribe.

The sections that follow describe the various components of the BizTalk Server runtime architecture.

In This Section

- Engine
- Topology
- Artifacts
- Deployment and Binding
- Enterprise Single Sign On (SSO)
- Business Rules Engine
- BizTalk Assemblies

Engine

This topic discusses the components and architecture that comprise the BizTalk Server engine.

In This Section

- Publish and Subscribe

- Database Structure and Jobs
- The MessageBox Database
- Batching

Publish and Subscribe

In a publish/subscribe design, you have three components:

- Publishers
- Subscribers
- Events
- Events in BizTalk Server simply refer to your messages. Publishers include receive ports, orchestrations, and send ports. Publishers include receive ports that publish messages that arrive in their receive locations, orchestrations that publish messages when sending messages or starting another orchestration asynchronously, and solicit-response send ports that publish messages when they receive a response from the target application or transport.

In BizTalk Server, there are two main types of subscriptions: activation and instance. An activation subscription is one specifying that a message that fulfills the subscription should activate, or create, a new instance of the subscriber when it is received. Examples of things that create activation subscriptions include send ports with filters or send ports that are bound to orchestrations, and orchestration receive shapes that have their "Activate" property set to true. An instance subscription indicates that messages that fulfill the subscription should be routed to an already-running instance of the subscriber. Examples of things that create instance subscriptions are orchestrations with correlated receives and request/response-style receive ports waiting for a response from BizTalk Server.

The difference between the two types of subscription at the information level is that an instance subscription includes the unique instance ID, stored in the subscription table in the master MessageBox. When an orchestration instance or receive port completes processing, instance subscriptions are removed from the MessageBox while activation subscriptions remain active as long as the orchestration or send port is enlisted. For more information about how orchestrations interact with the Messaging system, see the topic "Orchestrations and Messaging" in this document.

Creating Subscriptions

Subscriptions are created by service classes in BizTalk Server, which are listed in the adm_ServiceClass table in the BizTalk Management database. These services were identified earlier in the document in the section "Hosts and Host Instances" and include the caching service; in-process and isolated messaging, hosted by the Endpoint Manager; orchestrations/XLANG hosted by the XLANG subservice; and MSMQT. Each of these service classes can create subscriptions and receive published messages.

A subscription is a collection of comparison statements, known as predicates, involving message context properties and the values specific to the subscription. For example, Message Type is a context property of messages and many subscriptions specify the message type in their subscription. General information about the subscription is inserted into the subscriptions table by the Message Agent while the specific predicates go into one of the following predicate tables, depending on the type of operation specified for the subscription:

- BitwiseANDPredicates
- EqualsPredicates
- EqualsPredicates2ndPass
- ExistsPredicates
- FirstPassPredicates
- GreaterThanOrEqualsPredicates
- GreaterThanPredicates
- LessThenOrEqualsPredicates
- LessThenPredicates
- NotEqualsPredicates

All of this is accomplished by calling the Bts_CreateSubscription_<application name> and Bts_InsertPredicate_<application name> stored procedures in the MessageBox where <application name> is the name of the BizTalk host that is creating the subscription.

When a send port is enlisted, the port creates, at a minimum, a subscription for any message with that send port's transport ID in the context. This allows the send port to always receive messages intended specifically for it. When an orchestration port is bound to a particular send port, the information about that binding is stored in the BizTalk Configuration database. When messages are sent from the orchestration through the port bound to the physical send port, the Transport ID is included in the context so that the message gets routed to that send port. However, it is important to note that this send port is not the only send port that can receive messages sent from the orchestration. When an orchestration sends a message, that message is published to the MessageBox with all pf the relevant promoted properties. The bound send port is guaranteed to receive a copy of the message because the transport ID is in the context, but any other send port, or orchestration, can have a subscription that also matches the message properties. It is very important to understand that any time a message is published directly to the MessageBox, all subscribers with matching subscriptions will receive a copy of the message.

Publishing and Routing

After a subscription is created and enabled, a message must be published before any processing takes place. Messages are published when they are received into BizTalk Server from one of the services mentioned previously. For this discussion of routing, we will focus on messages received into BizTalk Server via an adapter.

When messages go through the receive pipeline processing, properties are promoted into the message context. After a message is ready to be published into the MessageBox after being processed by the receive adapter and pipeline, the first thing that happens is the Message Agent inserts the property values for the promoted properties and predicate values from the message context into the master MessageBox SQL Server database. Having these values in the database enables the Message Agent to make routing decisions.

The next step is for the Message Agent to ask the master MessageBox database to find subscriptions for the current batch of messages being published. Keep in mind that the messages have not yet been written to the database, only the properties from the context. The bts_FindSubscriptions stored procedure in the MessageBox queries the subscription and predicate tables identified above, linking them to the message properties stored for the current batch of messages.

With this information, the Message Agent inserts the message once into each MessageBox database that has a subscription by calling the bts_InsertMessage stored procedure. The bts_InsertMessage stored procedure is first called with a subscription ID. On this first pass, the stored procedure calls the int_EvaluateSubscriptions stored procedure which is responsible for looking up the subscription detail information, validating that the message meets security requirements for the application by checking that message predicates match application properties for the host, and inserting a reference to the message in the application specific queue or application specific suspended queue depending on the state. The message ID, subscription ID, service ID, and other subscription information are inserted into the application specific queue table for each subscription that was found for this application. After the messages are inserted, the message properties and message predicates tables are cleared of the batch related values.

The bts_InsertMessage stored procedure is called subsequently for each part in the message. On the first call, the message context is passed and is then inserted into the SPOOL table along with metadata about the message such as the number of parts, the body part name and ID. In addition the message body part is inserted into the PARTS table using the int_InsertPart stored procedure. The bts_InsertMessage stored procedure is then called for each of the remaining message parts where they are simply passed to the int_InsertPart stored procedure to be persisted in the PARTS table.

When messages are routed, references are added for each service that receives the specific instance of the message and its parts by inserting records into the following tables:

- MessageRefCountLog1
- MessageRefCountLog2
- PartRefCountLog1
- PartRefCountLog2

These references keep the messages and parts from being deleted by cleanup jobs that run periodically to keep the MessageBox from getting full with message data for messages that are no longer in the system. Two tables are used to reduce contention and locking issues.

Now that the message has been routed to the right queue, stored in the Spool and Parts tables, and referenced in the application specific queues, the messages must be pulled off the queues by the application instances. Each host instance has a number of dequeuing threads that continuously poll the database on an interval that is configured in the adm_ServiceClass table in the BizTalk Management database. This same table has a column to indicate the number of dequeuing threads to be used. For information on tuning these parameters, see BizTalk Server 2006 Performance Characteristics. Each thread calls into the MessageBox database and calls the bts_DequeueMessages_<application name> stored procedure appropriate for the host application it is running in. This stored procedure uses locking semantics to make sure that only one instance and one dequeuing thread are able to operate on a message in the queue at a given time. The host instance that gets the lock gets the message, and is then responsible for handing the message to the subservice for which it is intended.

If the service receiving the message is the Endpoint Manager, then the send port is invoked (or the response portion of a request/response receive port) and if it is the XLANG/s subservice, an orchestration is either created, or located to service the subscription depending on whether there is an instance ID in the subscription. The service then releases the reference to the message and its part so that if no other services have references, the message data can be deleted

Database Structure and Jobs

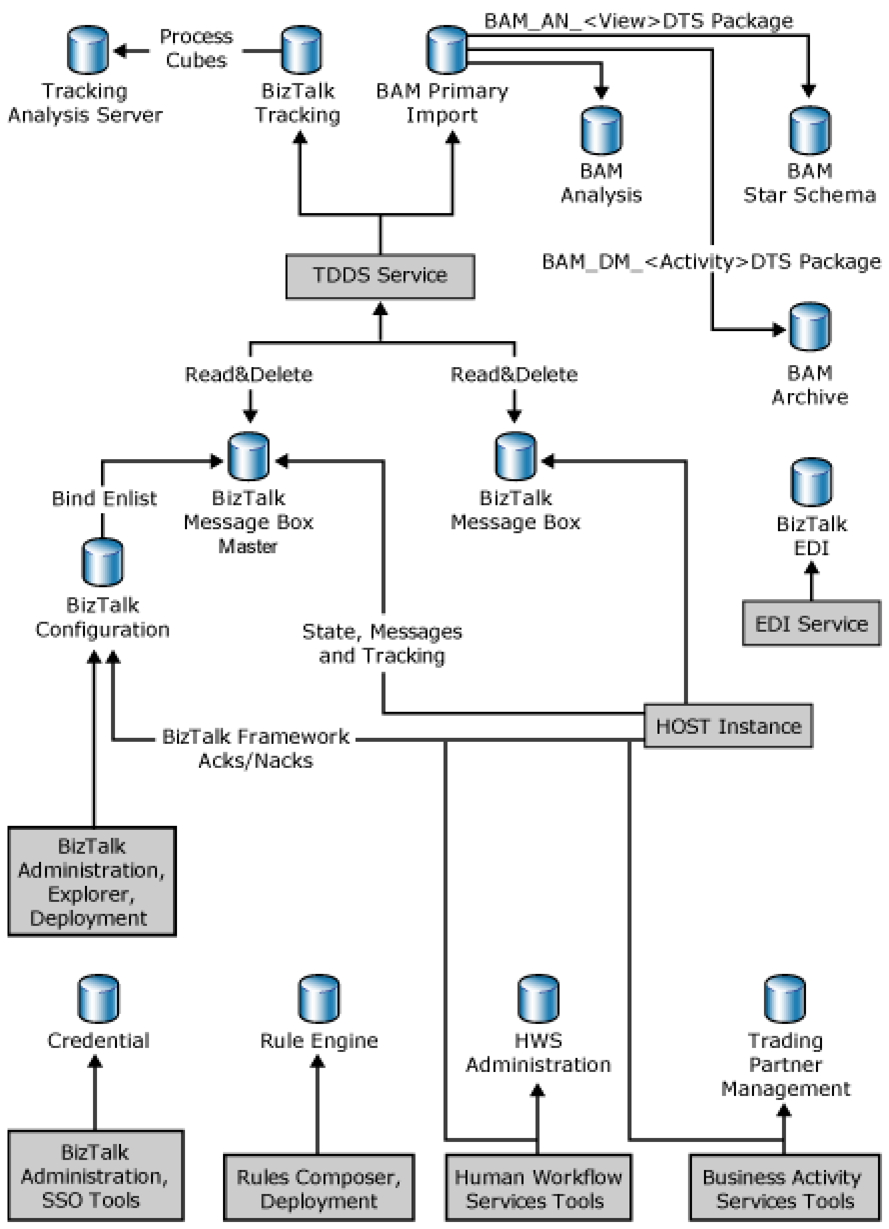
BizTalk Server Database Structure

Content to be provided

Database Write Diagram

The following figure shows the processes and entities that write to the BizTalk Server databases.

Database write diagram showing the processes and entities that write to the BizTalk Server databases


Tracking Analysis Server
Process Cubes
BizTalk Tracking
BAM Primary Import
BAM_AN_<View>DTS Package
BAM Analysis
BAM Star Schema
TDDS Service
BAM_DM_<Activity>DTS Package
BAM Archive
Read&Delete
Read&Delete
Bind Enlist
BizTalk Message Box Master
BizTalk Message Box
BizTalk EDI
BizTalk Configuration
State, Messages and Tracking
EDI Service
HOST Instance
BizTalk Framework Acks/Nacks
BizTalk Administration, Explorer, Deployment
Credential
Rule Engine
HWS Administration
Trading Partner Management
BizTalk Administration, SSO Tools
Rules Composer, Deployment
Human Workflow Services Tools
Business Activity Services Tools

BizTalk Server Database Jobs

BizTalk Server 2006 includes the following SQL Server Agent jobs to assist you in managing the BizTalk Server databases:

Job	Description
Backup BizTalk Server (BizTalkMgmtDb)	This job performs full database and log backups of the BizTalk Server databases. For more information about configuring and running this job, see Backing Up and Restoring

	BizTalk Server Databases.
CleanupBTFExpiredEntriesJob_BizTalkMgmtDb	This job cleans up expired BizTalk Framework (BTF) entries in the BizTalk Management (BizTalkMgmtDb) database.
DTA Purge and Archive (BizTalkDTADb)	This job automatically archives data in the BizTalk Tracking (BizTalkDTADb) database and purges obsolete data. For more information about configuring and running this job, see Archiving and Purging the BizTalk Tracking Database.
MessageBox_DeadProcesses_Cleanup_BizTalkMsgBoxDb	This job detects when a BizTalk Server host instance (NT service) has stopped and releases all work that was being done by that host instance so that it can be worked on by another host instance.
MessageBox_Message_Cleanup_BizTalkMsgBoxDb	This job removes all messages that are no longer being referenced by any subscribers in the BizTalk MessageBox (BizTalkMsgBoxDb) database tables.
MessageBox_Message_ManageRefCountLog_BizTalkMsgBoxDb	This job manages the reference count logs for messages and determines when a message is no longer referenced by any subscriber.
MessageBox_Parts_Cleanup_BizTalkMsgBoxDb	This job removes all message parts that are no longer being referenced by any messages in the BizTalk MessageBox (BizTalkMsgBoxDb) database tables. All messages are made up of one or more message parts, which contain the actual message data.
MessageBox_UpdateStats_BizTalkMsgBoxDb	This job manually updates the statistics for the BizTalk MessageBox (BizTalkMsgBoxDb) database.
Operations_OperateOnInstances_OnMaster_BizTalkMsgBoxDb	This job is needed for multiple MessageBox deployments. It asynchronously performs operational actions such as bulk terminate on the

	master MessageBox after those changes have been applied to the subordinate MessageBox.
PurgeSubscriptionsJob_BizTalkMsgBoxDb	This job purges unused subscription predicates from the BizTalk MessageBox (BizTalkMsgBoxDb) database.
Rules_Database_Cleanup_BizTalkRuleEngineDb	This job automatically purges old audit data from the Rule Engine (BizTalkRuleEngineDb) database every 90 days. This job also purges old history data (deploy/undeploy notifications) from the Rule Engine (BizTalkRuleEngineDb) database every 3 days.
TrackedMessages_Copy_BizTalkMsgBoxDb	This job copies the messages bodies of tracked messages from the BizTalk MessageBox (BizTalkMsgBoxDb) database to the BizTalk Tracking (BizTalkDTADb) database. For more information about running this job, see How to Copy Tracked Messages into the BizTalk Tracking Database.

The MessageBox Database

Administrators use the BizTalk Administration console or Windows Management Instrumentation (WMI) to manage MessageBox databases.

For information about using WMI to manage MessageBox databases, see **MSBTS_MsgBoxSetting (WMI)**.

The BizTalk Server group may have one or more MessageBox databases into which it publishes messages and from which subscribers to those messages extract messages.

A MessageBox database subscription is a set of established information and service information. The established (or predicate) information is the criteria that a message must meet. The service information is what to do with the message that meets the criteria. All of this information is stored in a set of tables that call the messaging and orchestration engine.

When BizTalk Server receives a message, it processes the message in a pipeline, and places the message in the MessageBox database. The incoming message has a context. The message context is a set of properties associated with the message. The three types of message context properties are:

- Simple written properties

- Promoted properties
- Predicate properties

The promoted and predicate message properties indicate the business process that subscribes to this message, and whether the business process has the permissions necessary to receive the message.

If a business process subscribes to the message, the MessageBox database sends the message to the business process. When the business process receives the message, it processes the message on an available host instance. After processing the message, if the business process subscribes to a pipeline or send port, the business process sends a return message to the MessageBox database.

For each BizTalk Host, the associated Message box has one work queue and one suspended queue. Additionally, each MessageBox database contains a set of tables for static states, dynamic states, and instance state. For information about BizTalk Hosts, see Managing BizTalk Hosts and Host Instances.

You create the first MessageBox database when you run the Configuration Wizard. This MessageBox database is the master MessageBox database. The master MessageBox database evaluates and routes subscriptions to all other MessageBox databases in the BizTalk Server environment. For information about improving performance for the master MessageBox database, see How to Disable New Message Publication.

Suspended messages in the MessageBox database

BizTalk stores messages associated with suspended pipelines in the MessageBox database. If a failure occurs in the pipeline, BizTalk Server suspends the instance of a message. There are two types of suspended service instances:

- Suspended instances that you can resume.
- Suspended instances that you cannot resume. For example, if an instance is corrupt.

Depending on the cause of the suspension, you may be able to resume services that BizTalk Server suspends. For example, if an orchestration hits a Suspend shape, or if a transport was unable to deliver a message,

BizTalk Server does not automatically remove suspended instances that you cannot resume from the MessageBox database. You can choose to save a service instance to disk before removing it from the suspended queue. You must use Health and Activity Tracking (HAT) to find these service instances and terminate them in order for BizTalk Server to remove them from the MessageBox database.

For more information about suspended instances, see Health and Activity Tracking.

For information about backing up MessageBox databases, see Backing Up and Restoring BizTalk Server Databases.

BizTalk Groups

The BizTalk group is a unit of organization that usually represents an enterprise, department, hub, or other business unit that requires a contained BizTalk Server implementation. The BizTalk group has a one-to-one relationship with a BizTalk Management database (also known as the Configuration database).

The BizTalk Management database stores configuration information for the BizTalk group and the servers in that group. This configuration information specifies part of the message-processing logic for the servers and where this logic will physically run.

You must specify the same BizTalk Management database for each server installation in the group so that you can administer each server from the administration console.

The BizTalk group properties page contains three tabs: **General**, **Database Connections**, and **Large Message**.

The following properties are on the **General** tab of the BizTalk group properties page:

- **Enterprise Single Sign-On server name.** BizTalk Server Administration needs to connect to the Credential database by means of an Enterprise Single Sign-On (SSO) server anytime that it needs access to the adapters' configuration information. This is the name of the SSO server used to connect to the Credential database.

- **Cache Refresh (seconds).** When you start BizTalk Server, all items in BizTalk Server Administration, such as MessageBox databases, server properties, adapters, and connections to the Tracking database, are stored in the administration cache. By default, all items in the cache are refreshed every 60 seconds, except for the server database connections and server properties. This means that if you change the general properties for a BizTalk group, such as the SMTP host, the changes are picked up within 60 seconds.

- For information about modifying the cache refresh interval, see Modifying BizTalk Group Properties.

- **Signing Certificate.** BizTalk Server uses one signing certificate per BizTalk group to sign all outbound messages. You must enter the thumbprint of the certificate with which outbound messages will be signed.

The **Database Connections** tab of the BizTalk group properties page lists the names of all databases and the servers they reside on specified during configuration.

When storing and retrieving large messages from the MessageBox databases, BizTalk Server supports fragmenting the messages. Fragmenting the messages enables BizTalk Server to handle large messages (for example, messages that are gigabytes in size), while significantly enhancing the runtime performance. The BizTalk properties on the **Large Message** tab that determine when and how to fragment messages are:

- **Threshold (in bytes).** When the size of a message is equal to or greater than this threshold size, the message will be fragmented.

- **Fragment Size (in bytes).** When the message is fragmented, this is the size of the fragments into which BizTalk Server will divide a large message when its size equals or exceeds threshold size. The default is 102,400 bytes.

Hosts

Administrators use the BizTalk Administration console or Windows Management Instrumentation (WMI) to create, modify, and delete hosts.

For information about using WMI to manage hosts, see **MSBTS_Host (WMI)** .

The BizTalk **Host** object represents a logical set of zero or more runtime processes in which you can deploy services, pipelines, and other artifacts. The **Host** object also represents a collection of runtime instances (zero or more) where the deployed items physically run.

After you create a host (a logical container), you can add physical BizTalk servers (host instances) to the host. You cannot add a Biztalk server to the same host more than once. A single host instance can be added to multiple hosts.

Items—such as adapter handlers, receive locations (including pipelines), and orchestrations—contained in BizTalk hosts can perform the following functions:

- **Receiving.** These items do the initial processing of messages after they are picked up in a receive location. When a host contains a receiving item, such as a receive location or pipeline, it acts as a security boundary, and the message decoding and decrypting occurs in a pipeline within the host.

- **Sending.** These items do the final processing of messages before they are sent out to the send port. When a host contains a sending item, such as a send port or pipeline, the host acts as a security boundary, and the message signing and encryption occurs in a pipeline within the host.

- **Processing.** These items process messages based on the instructions in an orchestration.

One BizTalk Host can contain items that receive, send, and process messages. It is recommended that you create different hosts for each function to create security boundaries and facilitate management. In particular, it is recommended that you use different hosts for processing and for receive/send, and that you separate trusted and non-trusted items.

Based on the physical configuration and type of adapter hosted, there are two types of hosts: in-process hosts and isolated hosts.

In-process hosts

In-process hosts represent service instances that an administrator creates, deletes, and fully controls with WMI and the BizTalk Administration console.

In-process hosts have the following characteristics:

- You can enlist any orchestration into an in-process host.
- An in-process host can host any send handler.
- An in-process host can host only the receive handlers associated with File and MSMQT adapters (the in-process-type adapters).
- The first in-process host you create in a BizTalk Server 2006 deployment is the **default host** and you cannot delete it. The BizTalk Message Queuing adapter uses the default host for static handler configuration. Adding an adapter automatically creates receive and send ports for the default host.

Isolated hosts

Isolated hosts represent service instances that a solutions developer programmatically creates, deletes, and controls. An administrator uses WMI and the BizTalk Administration console to configure these hosts (for example, to configure the host service account and authentication trust).

Isolated hosts primarily host adapters that must run outside of the normal BizTalk Server runtime process. For example, you use isolated hosts to host adapters for external processes such as ISAPI extensions and ASP.NET.

Isolated hosts have the following characteristics:

- You cannot enlist orchestrations into an isolated host.
- An isolated host cannot host send handlers.
- An isolated host can host only the receive handlers associated with HTTP/S and SOAP adapters (the isolated-type adapters).
- An isolated host cannot host tracking.
- An isolated host cannot be the default host.
- The status of an isolated host is always **Status Unavailable**. BizTalk Server does not access the status information for external processes.

Host Groups

You can associate a host with a Windows group. You can only associate a host with one Windows group (called a host group). The host group must have SQL Server login and privileges in all relevant BizTalk databases. When you associate a host with the host group, you grant the host the privileges of the host group.

If you use the Configuration Wizard to create hosts, and you specify a local Windows group to use for hosts, the Configuration Wizard automatically creates two Windows groups. The default names of these groups are the BizTalk Application Users group and the BizTalk Isolated Host Users group.

When you create a host instance of a host associated with the host group, the host instance inherits the database privileges of the host group.

The host group is a property of the Host WMI object. You can change the Windows group a host belongs to if there are no host instances of the host.

You specify the host group a host belongs to when you create the host. The BizTalk WMI provider grants the privileges that the host group has (for example, the BizTalk privileges required for runtime functionality, including SQL Server logins and database user access) to the host. You must specify the correct service account (host) when you create a host instance, so that the BizTalk WMI provider grants the privileges of the host group to the host instance.

The host group is the Windows group (named the BizTalk Application Users group by default) that you use for accounts with access to the in-process BizTalk hosts (host processes in BizTalk Server). It is recommended that you use one host group for each in-process host in your environment.

The host group requires the following privileges:

- It must be a member of the BTS_HOST_USERS SQL Server role in the following databases:
    - BizTalk Management (also known as the Configuration database)
    - MessageBox
    - Rule Engine
    - Tracking
    - BAM Primary Import
- It must be a member of the BTS_<in-process host name>_USERS SQL Server role for the MessageBox database
- It must be a member of the BAM_EVENT_WRITER SQL Server role in the BAM Primary Import database.

Host Instances

Administrators use the BizTalk Administration console or Windows Management Instrumentation (WMI) to create, modify, and delete host instances.

A host instance is the physical installation of a host in a BizTalk Server. An administrator uses Windows Management Instrumentation (WMI) or the BizTalk Administration console to install host instances. A BizTalk Server can support multiple host instances.

A host instance displays the following statuses in the **Status** column in the results pane:

- **Start pending.** This status is displayed when you have started the host instance. However, there is some delay between starting the host instance and the change taking effect. You might have to click the refresh button to see the change in status.

- **Running.** The host instance is currently running.

- **Stopped.** The host instance is currently stopped.

- **Status Unavailable.** The host instance displays this status when the host instance install or uninstall fails, or when BizTalk Server cannot retrieve the status of the host (for example, if the host is an Isolated host and the processes it contains are external to BizTalk Server).

A host instance displays the following statuses in the **Installation Status** column in the results pane:

- **Installation failed.** BizTalk Server could not install the host instance (for example, network problems interrupted the installation process). When you install a host instance, you must have the privileges to create a Windows service, and to grant SQL Server logon and access rights for the BizTalk databases to the Host Windows user group.

  If you do not have sufficient Windows and SQL Server privileges, host instance installation fails.

- **Installed.** You installed the host instance successfully.

- **Un-installation failed**. Host instance deletion failed.

- **Not installed.** There are no installed host instances.

Receive Locations

Creating a receive location involves specifying an address at which inbound messages arrive and the messaging pipeline that processes the message received at that address. Both solutions developers and administrators can create and enable receive locations. The following list describes how solutions developers and administrators work with receive locations:

- A solutions developer uses BizTalk® Explorer in Microsoft® Visual Studio® .2005 to create receive locations, bind receive locations to orchestrations, and enable receive locations.

- An administrator uses Microsoft Windows® Management Instrumentation (WMI) to create receive locations, bind receive locations to an orchestration, enable receive locations, disable receive locations, and set properties for receive locations.

- An administrator uses the BizTalk Administration console to enable receive locations, disable receive locations, and set global properties for receive locations. Administrators cannot use the BizTalk Administration console to create receive locations, delete receive locations, or bind receive locations.

The following steps describe the lifecycle of a receive location:

1. A solutions developer uses BizTalk Explorer in Visual Studio 2005, or an administrator uses WMI to create a receive location.

2. A solutions developer uses BizTalk Explorer in Visual Studio 2005, or an administrator uses WMI, or an administrator uses the BizTalk Administration console to associate a receive location with a host.

3. A solutions developer uses BizTalk Explorer in Visual Studio 2005, or an administrator uses WMI to bind a receive location to an orchestration.

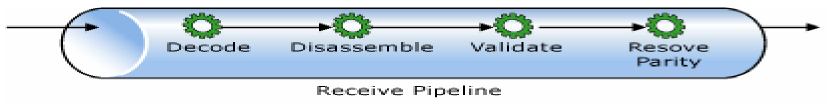
4. A solutions developer uses BizTalk Explorer in Visual Studio 2005, or an administrator uses WMI, or an administrator uses the BizTalk Administration console to enable a receive location.

5. The receive location receives messages.

Pipelines

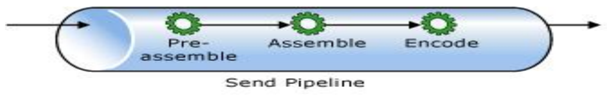
Pipelines are a component of BizTalk Server that provides an implementation of the Pipes and Filters integration pattern. During the receiving and sending of messages, there are business reasons to perform transformations on messages to prepare them to enter or leave BizTalk Server. A common example is that you may need to transform a comma-delimited flat file into an XML file in order to take advantage of certain features in BizTalk Server such as maps; the flat file disassembler component does just that. It is common in integration scenarios to have a need to perform several types of transformations to a message before receiving or sending it; that is where pipelines come into play. Pipelines enable the developer to define a series of transformations that will be performed on a message as it is being received or sent.
There are two types of pipelines, send and receive, and these match the ports in which they execute. Send pipelines are executed in send ports and in the response portion of a request/response receive port while receive pipelines are executed in receive locations, and in the response portion of a solicit/response send port. Essentially, receive pipelines are intended to be used to transform messages that are being published to the MessageBox while send pipelines are intended to be used on messages which have been subscribed to and are being sent out of BizTalk Server.
Each pipeline has a set of stages that are executed in order when the pipeline is executed. Each stage can contain zero or more components. The maximum number of components depends on the stage.

Receive Pipeline



Stage	Purpose
Decode	Decrypts or decodes the message data
Disassemble	Disassembles an interchange into smaller messages and parses message contents
Validate	Validates the message data, generally against a schema
Resolve Party	Identifies the BizTalk Server party associated with some security token in the message or message context

Send Pipeline



Stage	Purpose
Pre-assemble	Performs any message processing necessary before assembling the message
Assemble	Assembles the message and prepares it to be transmitted by taking steps such as adding envelopes, converting XML to flat files, or other tasks complementary to the disassemble stage in a receive pipeline
Encode	Encodes or encrypts the message before delivery

A stage in a pipeline has an execution mode of either All or First Match, which controls the components that get executed if more than one component is added to a stage. For stages with a mode of All, each component is called to process the message in the order inn which they are configured in the stage. When the mode is First Match, each component is polled to indicate that it is the right component until a match is found, at which point the component that matches is executed, while the remaining components do not get executed.

As an example of execution modes, the Disassemble stage of a receive pipeline is a first match stage, thus each component in the stage is called to see if it recognizes the message and can process it. If the component responds in the affirmative, then no other components in that stage will be queried to see if they can also handle the message. However, the Decode stage of a receive pipeline has an execution mode of All, meaning that each component in this stage is called to process the message in the order in which they were configured. The first decoder might be to decrypt the message, while the second might be to decompress the message from a zipped format.

One common consequence of execution mode in pipeline processing occurs when a developer wants to use multiple disassemblers in a single receive pipeline. Often the disassembling components differ only slightly, for example two flat file disassemblers with similar but different schemas configured. In this case, while the message might actually match the schema defined in the second disassembler configured, the first disassembler might determine through its probing that it can process the message. It is only after processing the message that the error is discovered and the message suspended. In these cases, the choice is to either create a new disassembler which has more specific probing logic in it, or to create two different pipelines and receive the different messages in different receive locations.

Orchestrations

Orchestrations can subscribe to (receive) and publish (send) messages through the MessageBox. In addition, orchestrations can construct new messages. Receiving messages occurs via the subscription and routing infrastructure already discussed. When subscriptions are filled for orchestrations, a new instance is activated and the message is delivered, or in the case of instance subscriptions, the instance is rehydrated if necessary and the message is then delivered. When messages are sent from an orchestration, they are published to the MessageBox in the same manner as a message arriving on a receive location with the appropriate properties getting inserted into the database for use in routing.

Messages that are constructed in an orchestration must be persisted in the MessageBox and referenced by the orchestration instance, but they should not be published because they have not yet been sent. The XLANG/s subservice makes calls to the Message Agent API to insert messages directly. This allows the engine to insert the message body into the MessageBox and have it directly associated with the running orchestration instance. The persistence of the constructed message in the MessageBox is coordinated with persistence points in the orchestration as an additional optimization of database operations.

The concept in orchestrations that makes the publish and subscribe seem to act differently is binding. Orchestration ports are logical ports that describe an interaction. These logical ports need to be bound to some physical port in order for messages to actually get delivered, but this binding process is nothing more than configuring subscriptions for message routing. There are four basic options for binding these ports:

- Specify Now (specifying them directly in the orchestration)
- Specify Later (specifying them at deployment time)
- Using a dynamic send port where the address is set in the orchestration code
- Creating a direct binding from the orchestration to either the MessageBox or another orchestration

When a binding is specified at design time, a physical port that matches the parameters configured in the orchestration gets created when the orchestration is deployed. When the binding is configured at deployment time, any port that matches the requirements of the logical port can be bound to the orchestration port. For dynamic binding, a physical port is created just as with the Specify Now option, but the port is a dynamic send port that has no address information configured.

A confusing concept for many developers is that while a send port in an orchestration is bound to a physical send port, this does not preclude that message from getting delivered to other subscribers. That is, if another send port happens to have a subscription, via its filters, for the message being sent to the bound port, both send ports will receive the message. Binding simply creates the subscription such that the message sent from the orchestration always matches the criteria for the bound send port. Likewise, the orchestration port bound to a receive port creates the appropriate subscription based on message type and receive port ID. The subscriptions guarantee that the messages going in and out of the orchestration get delivered to the bound ports, but the messages still go through the same publish and subscribe mechanism described earlier.

Probably the most misunderstood, and misused or underused binding option is the Direct binding option. Direct binding allows an orchestration to publish messages to the MessageBox with varying routing properties much like messages are published by receive locations. In simple direct messaging, the message is published to the MessageBox with its promoted properties to be routed like any other published message received into BizTalk. This allows any subscriber(s) to receive this message, but requires that at least one subscriber exist or the orchestration will get an exception thrown back to it because a routing failure has occurred. For more information about routing failures, see the topic "Routing Failures" in this document.

Another option for direct binding is to use self-correlating ports. Self-correlating ports are ports that create a unique correlation token and use that token alone in correlating messages between instances. The most common use of a self-correlating port is to call or start an orchestration passing in a port parameter. In the called orchestration, the port can be used to send a message, while in the calling orchestration the same port can be used to receive a message. Because the port has a unique correlation token, the message is routed back to the calling orchestration. Self-correlation ports act as private communication channels between orchestration instances.

The final option is to use a partner orchestration in which, in both the calling orchestration and the called orchestration, the port is configured using the same shared port type and in the port configuration, the same port is selected. For example, in both Orch1 and Orch2, Orch2.MyDirectPort is selected. This type of binding sets up a subscription for the receiving orchestration based on the sending orchestration type, the port name, and the operation name. This again ensures that the messages get routed to the correct instance.

One thing to keep in mind with these direct messaging options, as well as using the Start orchestration shape to asynchronously start an orchestration, is that they all use the underlying publish and subscribe model. The difference between these options is in the properties that are used for creating subscriptions and routing, and in the use cases they help solve.

One common problem encountered when using direct bound ports in orchestrations is that an orchestration may publish a message that it is also subscribed to. For example, an orchestration is configured to be activated by a PurchaseOrder message. This orchestration uses a direct port to publish the PurchaseOrder message to the MessageBox. However, in addition to receiving the message as expected, another instance of an orchestration gets started because it too had a subscription for PurchaseOrder messages. The processing gets into an endless loop and it may take some time for a developer to figure out what has happened.

Correlation

Correlation in orchestrations is the mechanism for being able to receive related messages into the same running orchestration instance. In Orchestration Designer, in order to use correlation, a developer follows these general steps:

- Defines a correlation type that includes the promoted properties that are used to relate messages.
- Defines a correlation set that is an instance of the correlation type just defined.
- Specifies on send and receive ports whether they initiate or follow a given correlation set.

Instance subscriptions come into play when a correlation set is initiated, as this is when subscriptions are created for all of those ports that follow this correlation set to receive messages. Because the correlation type defines the properties to be used for correlation, the orchestration engine can extract these properties from the message being sent or received by the initiating action. These values are then used to define subscriptions for all of the remaining actions which follow this correlation set.

Due to how correlation works, it is important that messages received into BizTalk Server and intended for use in a correlation have their promoted properties correctly defined and promoted to the message context. In order for most properties to get promoted when a message is initially received, a disassembler component is used to extract the values. For this reason, it is not possible to use the PassThrough receive pipeline to receive messages that must be correlated to a running instance of an orchestration. This is a common issue that comes up when using the SOAP receive adapter to receive correlated messages because the default value of the receive pipeline when using the Web Services Publishing Wizard is the PassThrough pipeline.

BizTalk Assemblies

The most important aspect of BizTalk and .NET is that all BizTalk Server artifacts; maps, schemas, orchestrations, and pipelines, get compiled into .NET assemblies. The two most important implications of this design are that these assemblies must be strong named, and because of that, they also follow .NET versioning rules. The main implication of this is that a BizTalk project, once built against a particular version of another .NET project/assembly (including BizTalk projects), continues to use that version until it has been rebuilt against a newer version.

A common problem that occurs during development related to .NET versioning is when the version numbers on a BizTalk project are not changed and the assembly is redeployed without stopping and starting the BizTalk host instance that the types are loaded into. When the process is run again, the changes do not take effect. This is due to the way in which .NET assemblies are loaded into memory. Because the host already has an in-memory copy of the assembly, it does not reload the assembly when a new copy is put into the Global Assembly Cache. For example, if version 1.0.0.0 of an assembly with an orchestration is deployed and running, and changes are made to the orchestration but the version number is not changed, then the changes do not take effect. After the host instance is stopped, the in-memory copy of the assembly is released and when the host instance starts again it reloads the new copy of the assembly and gets the changes. If a new version was deployed, say version 2.0.0.0, and it was loaded, then the changes would have taken effect.

Deploying assemblies to BizTalk Server is a two part process. The first part is traditional .NET assembly deployment in which the strong named assembly is deployed to the Global Assembly Cache (GAC) on each server where the assembly will be used. The second step is to deploy metadata about the assemblies and their types to the BizTalk Server Management database. When BizTalk assemblies are loaded by BizTalk Server, they are most often loaded using their strong name, found in the Management database.

The BizTalk Server artifacts that a developer creates get compiled into classes which derive from built in BizTalk Server types. For example, an orchestration becomes a class which derives from the Microsoft.BizTalkXLANGs.BTXEngine.BTXService class. It is because these base classes are deployed in assemblies to the Global Assembly Cache, and these assemblies have dependencies on other assemblies in the GAC, that a developer's assemblies must also get deployed to the GAC.

Another important implication of BizTalk Server artifacts being deployed to the Global Assembly Cache and therefore being strong named, is that strong named assemblies cannot call other assemblies that are not also strong named. This means that any assemblies a developer creates that are used by these BizTalk assemblies must also be strong named. Likewise, assemblies deployed to the GAC that load other assemblies without using a specific path, must load those assemblies from the GAC.

Unlike many of the other components that developers create, Pipeline components do not get deployed to the GAC. Pipeline components work entirely with BizTalk Interfaces and therefore do not have base classes deployed in the GAC. Instead, Pipeline components are found and loaded using another common method for deploying .NET assemblies, which is to deploy them to a known location. Pipeline components are deployed to a folder named "pipeline components" in the BizTalk Server installation directory.

Pipeline components are added to a developer's toolbox in Visual Studio .NET to make them available to be dragged onto the pipeline designer. When a BizTalk pipeline is compiled into a .NET assembly, the information about all of the components in the various stages of the pipeline get compiled into the assembly. When this pipeline is deployed to BizTalk Server, the information about the components, including their file name, is inserted into the BizTalk Management database. When the pipeline is executed, these components are loaded from their known location, and the interfaces they implement called as appropriate.

It is important to note two things about pipeline components here. First, any assemblies that the pipeline component depends upon must be deployed to the GAC in order to be found at runtime. This is because only the pipeline component itself has metadata in BizTalk allowing it to be loaded from a known location. Second, due to new features in BizTalk Server 2006, this will be changing, and pipeline components will be deployed into the GAC.

Health and Activity Tracking (HAT)

In This Section

- Planning for Health and Activity Tracking
- Live Messages
- Archived Messages
- Record Size in Tracking Databases

Planning for Health and Activity Tracking

You should decide during the planning stages which information you need to track, so that after you deploy the project you can set the tracking options and limit the amount of tracked data to give you only the information you need.

We recommend that you do not track all messages, because each time a message is touched, Microsoft® BizTalk® Server Health and Activity Tracking (HAT) makes another copy. For example, you can narrow the scope by tracking only a specific port. This helps to maximize the performance of your system and keep the databases uncluttered.

Record Size in Tracking Databases

To help you plan your hardware requirements for BizTalk, the following table shows the expected record size for various event records in the Tracking database.

Action Type	Size	Notes
Deploying a Service	1864 + Symbolic Information Size	Symbolic Information depends on the size of the orchestration. For example, an orchestration that receives one message and sends one message out with 2 shapes in it takes approximately 4000 bytes.
Started and Successfully Completed Service Instance	Normally 252 bytes. Extreme cases, can reach 735 bytes.	
Started and Failed/Exception Met Service Instance	Normally 252 bytes + error information. Extreme cases can reach 735 bytes + error information.	Error information is the text data returned by a BizTalk or user component. Ranges from 100 bytes to 2 KB.
Start/End of a Shape In an Orchestration	120 bytes each.	
Message In/Out Events	Minimum 162 bytes. First time message is seen it is 202 bytes + Message Property (if tracked) Extreme cases can reach 2930 bytes.	You can safely assume that the average is 182 bytes if there is no message property tracking.
Message Property Size	40 to 288 bytes if property the DTA database recognizes it. Add up to 268 bytes for tracking the property the first time.	

Capacity Planning for the BizTalk Tracking Database

This section describes sizing considerations for the Data Tracking (DTA) Database in BizTalk Server 2006. It explains how to use equations and message variables to determine how large the DTA database will become over a given period of time, and provides specific examples of how to apply the equations.

"Using Message Variables to Determine DTA Size" provides an overview of the entire process. "Track Message Bodies" explains how to account for the size of message bodies in calculations. The remaining topics provide specific instructions for calculating DTA size in different scenarios.

In This Section

- Use Message Variables to Determine DTA Size
- Track Message Bodies
- Calculate the DTA Size for Simple BizTalk Messages
- Calculate the DTA Size for Messages in Orchestrations
- Calculate the DTA Size for Messages in Orchestrations Sent Out to Distribution Lists
- Calculate the DTA Size for Simple Messages, Messages in Orchestrations, and Messages in Orchestrations Sent Out to a Distribution List

Use Message Variables to Determine DTA Size

In Microsoft® BizTalk® Server 2006, you can use a number of variables to determine how large the BizTalk Server Tracking (DTA) database will become over a given period of time. These variables are:

- Number of pipelines used
- Number of orchestrations involved
- Number of events generated
- Number of message properties tracked
- Number of additional messages created
- Estimated number of messages received in the specified timeframe

While the equation you use to estimate the size of the DTA database is straightforward, you must apply it to each incoming and outgoing message process that uses the BizTalk Server implementation. In other words, you will need to apply this equation for every distinct message scenario and then add up the results to obtain the final estimated database size. In this document we will look at two scenarios. The scenarios are:

1. Receiving a message, transforming the message, and then sending the resulting message
2. Receiving a message, running a business process using the message, and then sending the resulting message.

Both of these scenarios may be present in a BizTalk Server installation, and each scenario generates a different amount of tracking data. The total tracking data generated for the BizTalk Server installation is the sum of all the scenarios.

The following are some variables used in the equation:

Variable	**Description**
Nserv	Number of services (number of pipelines + number of orchestrations)
Events	Number of generated message events
Properties	Number of message properties tracked
PropSize	Size (in bytes) of the promoted property (field)
CMsgs	Number of additional messages created per incoming message
Msgs	Number of estimated incoming messages in a given time period
MsgSize	Message size
MsgNum	Number of messages tracked for each incoming message

The equation is as follows:

This equation calculates only the tracking data generated by the messages and does not include the tracking data generated for the Orchestration Debugger. You must apply this formula to each message process to estimate the size of the DTA database.

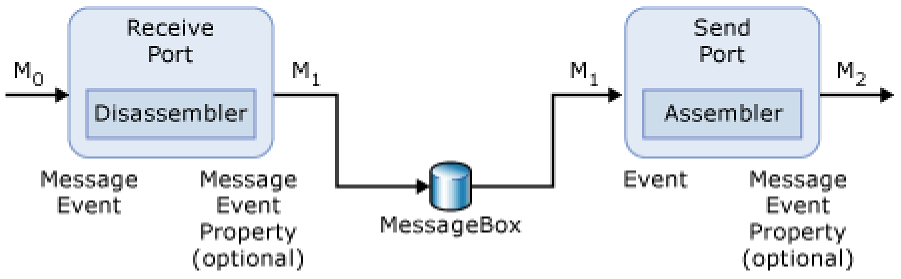
Track Message Bodies

If you plan to track the message bodies in the DTA database, then you will also need to account for the size of these bodies in your calculation. Use the following equation:

The MsgNum variable is determined by turning on the tracking features at the port level or at the orchestration level using Microsoft® BizTalk® Server Health and Activity Tracking (HAT) or BizTalk Explorer. You must apply this formula to each message process as well.

Calculate the DTA Size for Simple BizTalk Messages

In the following figure, a simple BizTalk Server message passes in and out of BizTalk Server without undergoing any message transformation.



A simple BizTalk Server message - no transformation

Before you apply the formula in the previous section, you will need to gather some facts about this scenario. In this example, we use the following:

- The message size is 5K.
- No properties will be promoted.
- The number of messages you receive in a year is 3.5 million.
- Tracking is turned on for all events. There are four events in this scenario. The events are:
    - Receipt of message M0
    - Output of message M1 from the receive port
    - Receipt of message M1 by the transmit pipeline
    - Output of message M2 by the send pipeline
- Two additional messages are created in this scenario. Message M0 is the incoming message and is therefore not created by BizTalk Server. Message M1 is the output message from the receive port and M2 is the output from the transmit port. M1 and M2 are created by BizTalk Server.

Applying this information to the formula gives the following:

Messages with a single promoted property

Let's take another look at this example and add one additional fact to the scenario. You want to promote a single field, approximately 10 bytes in size, from the original message. The maximum size of a promoted field is 256 characters, or approximately 256 bytes (512 bytes if the characters are Unicode).

With this additional fact, the equation now appears as follows:

If you wanted to promote an additional field that is 10 bytes in size, the equation would be:

As you can see, if you promote a single 10-byte property in this scenario, it will add an additional 333.79 MB ~ 0.33 GB per year to the size of the DTA database.

Promoting two 10-byte properties will add an additional 667.58 MB ~ 0.65 GB of additional space per year to the size of the DTA database.

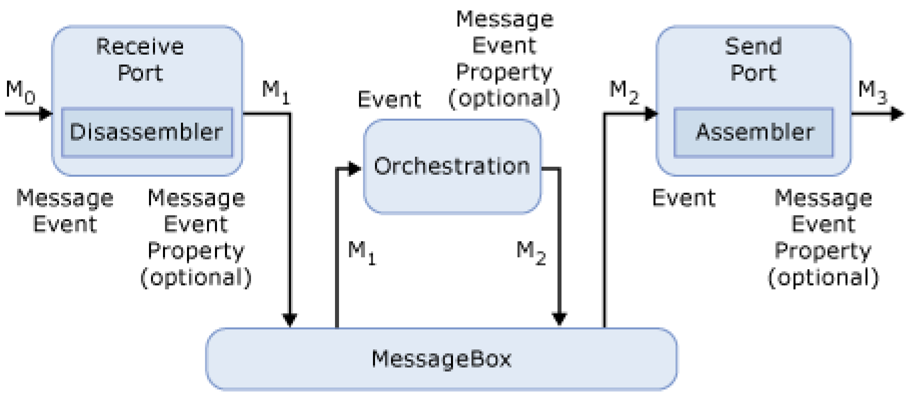
Messages with message body tracking activated

In this example, let us also assume that we are planning on activating message body tracking. We will need to add the second equation for message tracking, shown in the previous section. The equation will look like the following for this example:

By adding the results of the two equations, we can estimate that the DTA database will grow approximately 54.44 GB to 54.77 GB per year.

Calculate the DTA Size for Messages in Orchestrations

Let's look at an example that includes an orchestration. The following figure displays the entire business process. In this scenario, a message comes into BizTalk Server, is sent through an orchestration, is changed within the orchestration, and is then sent out through a send port.



The BizTalk Server message process

Here are some of the facts concerning this scenario:

- The message size is 5K.
- We are not promoting any properties.
- The number of messages we receive in a year is 3.5 million.
- Tracking is turned on for all events. There are six events in this scenario:
    - Receipt of message M0
    - Output of message M1 from the receive port
    - Receipt of message M1 by the orchestration
    - Output of message M2 from the orchestration
    - Receipt of message M2 by the send port
    - Output of message M3 by the send pipeline
- Three additional messages are created in this scenario. Message M0 is the incoming message and is therefore not created by BizTalk Server. Message M1 is the output message from the receive port, M2 is the output message from the orchestration, and M3 is the output message from the transmit port.

Applying this information to the formula gives the following result:

Messages in orchestrations with a single promoted property

Now let's promote a single field in this scenario, as in the earlier example. The promoted property is approximately 10 bytes in size. The equation now looks like this:

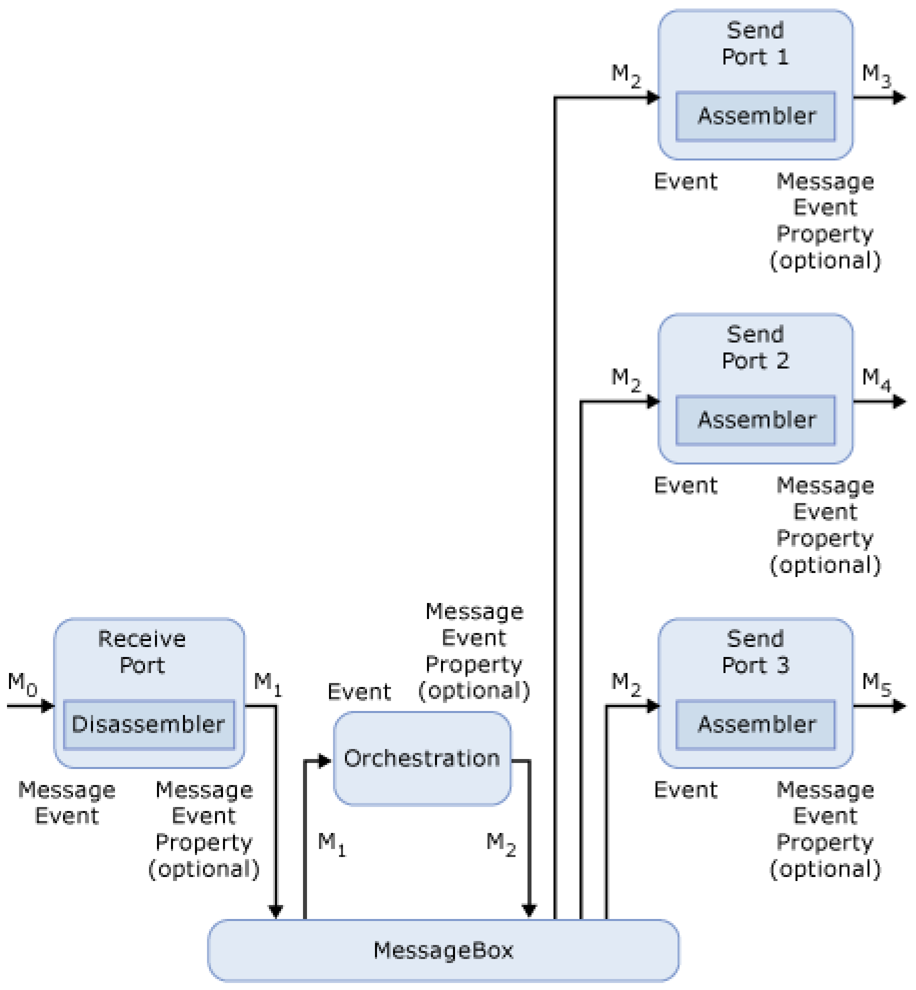
If you need to promote an additional property that is 20 bytes in size, the formula now looks like this:

Messages in orchestrations with message body tracking activated

If you want to accommodate message tracking, the result from calculating the additional space needed is identical to the result in the earlier scenario, or 50.1 GB per year.

Calculate the DTA Size for Messages in Orchestrations Sent Out to Distribution Lists

the following figure, you have a message that proceeds through an orchestration, is changed within the orchestration, and is then sent out to several different send ports through a distribution list.



BizTalk Server message that proceeds through an orchestration and is sent out to several different ports

Here are some of the facts concerning this scenario:

- The message size is 5K.
- You are not promoting any properties.
- The number of messages you receive in a year is 3.5 million.
- Tracking is turned on for all events. There are five events in this scenario:
  - Receipt of message M0
  - Output of message M1 from the receive port
  - Output of message M3 by the send port
  - Output of message M4 by the send port

- Output of message M5 by the send port

Applying this information to the equation gives the following:

Messages in an orchestration that are sent out to a distribution list with a single promoted property

In this example, let's promote a single property, approximately 10 bytes in size, as we did in an earlier scenario. The equation now looks like this:

If we promote an additional property that is 20 bytes in size the equation now looks like this:

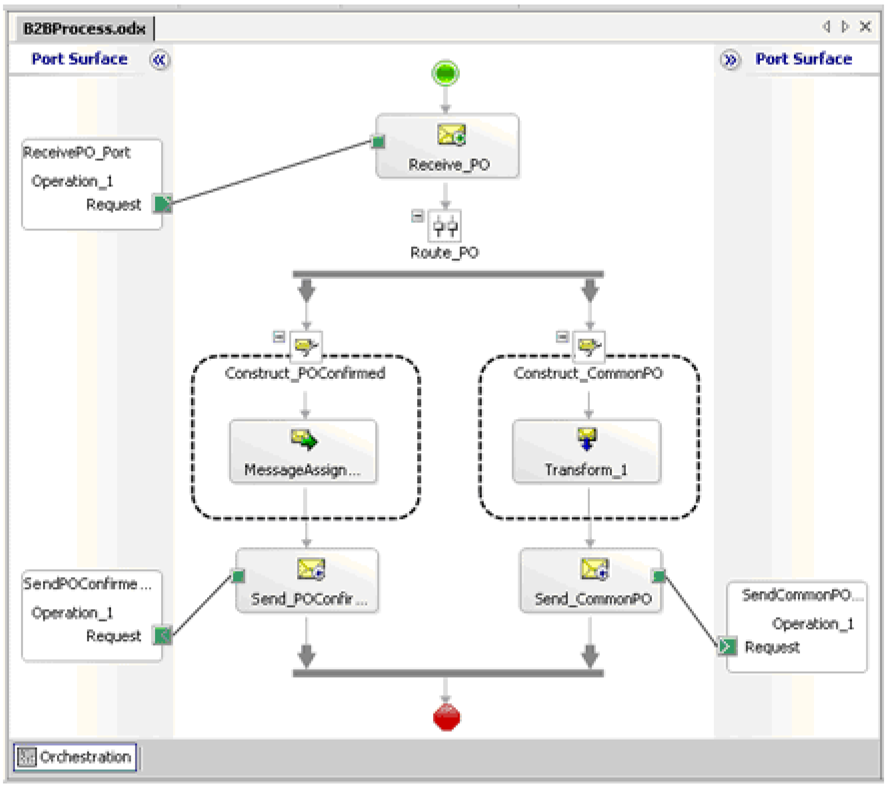
Messages in an orchestration that are sent out to a distribution list with message body tracking activated

If you want to accommodate message tracking, the equation will look like the following for this example:

Calculate the DTA Size for Simple Messages, Messages in Orchestrations, and Messages in Orchestrations Sent Out to a Distribution List

The formula to determine the size needed to track shape status is:

For example, in the following figure, you would use the following formula to determine the DTA size:



If we assume that this orchestration processes 3.5 million messages, the additional spaceneeded to track this orchestration would be:

You will need to account for each orchestration that has the orchestration debugger set to "on" to get an approximate size for the DTA database.

Planning for Sustained Performance

Insert introduction here.

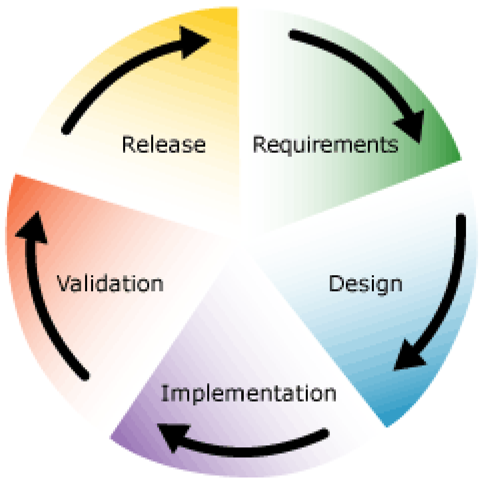
In This Section

- Sustainable Performance Defined
- Persistence and Durability
- Project Planning Recommendations by Phase
- Engine Performance and Capacity
- Tracking Performance and Capacity

Project Planning Recommendations by Phase

In This Section

Project Planning Cycle



Measuring Maximum Sustainable Engine Throughput

One of the primary considerations when planning a BizTalk Server system should be to determine the maximum sustainable throughput of the system. The maximum sustainable throughput of a BizTalk Server system is measured as the highest load of message traffic that the BizTalk system can handle indefinitely in production. This is important because as load exceeds the maximum sustainable message traffic, messages are backlogged in the messagebox database and delivery of potentially time sensitive documents is delayed. Additionally, the overhead associated with maintaining the message backlog can further reduce performance which may contribute to additional backlogging. If you calculate the maximum sustainable throughput of your BizTalk Server system as part of normal testing then you can scale your system to avoid extended overload scenarios in production.

In This Section

- Factors that Affect Maximum Sustainable Performance
- SQL Agent Jobs that Maintain the Messagebox Database
- Test Scenarios for Measuring Maximum Sustainable Performance
- Recommendations When Testing Engine Performance

Factors that Affect Maximum Sustainable Performance

Maximum sustainable performance is most directly impacted by factors such as available server resources, message size, and overall message load. There are other factors to be considered though that may not be immediately obvious. The following factors should also be considered when estimating maximum sustainable performance:

Load Patterns

Messages do not typically flow into a production BizTalk Server environment at a predictable, consistent rate. More typically, business needs dictate that BizTalk Server process messages in large batches that flow into BizTalk at certain intervals. When this occurs, the BizTalk Server processing requirements may quickly jump from negligible to an overdrive condition. When an overdrive condition occurs, published messages are backlogged in the messagebox database until BizTalk delivers the backlogged messages to the appropriate subscribers. As long as BizTalk Server is able to process the backlog of messages accumulated between batches then this is not problematic. If the backlog of messages continues to grow over time then the message publishing rate is exceeding the message delivery rate. In this case, the message publishing rate has exceeded the maximum sustainable output of the BizTalk Server environment and the BizTalk Server environment should be scaled accordingly.

Because of the typically unpredictable pattern of message flow into a BizTalk Server environment, testing scenarios should be run for an extended period of time to ensure that BizTalk can recover from overload conditions.

Monitoring and Maintenance activities

The following activities directly or indirectly impact BizTalk performance and so should be factored into any testing scenarios:

- **BizTalk Administration Console and HAT queries** These queries consume resources and affect overall throughput depending on the type and frequency of the query.

- **Backup, archiving, and purging activities** These activities also consume resources and should be factored into any testing scenarios.

SQL Agent Jobs that Maintain the Messagebox Database

Every message that is received by or created within BizTalk Server 2006 is immutable. That is, once it has been received or created, its content cannot be changed. Received messages may have multiple subscribers. Each subscriber of a particular message references the same single copy of that message. While this approach minimizes storage, a reference count must be kept for each message and garbage collection must be performed periodically to remove those messages that have a reference count of 0. The following SQL Agent jobs in BizTalk Server 2006 perform garbage collection for messages:

MessageBox_Message_Cleanup_BizTalkMsgBoxDb Removes all messages that are no longer being referenced by any subscribers. This job searches the messagebox database for messages that have a reference count of 0 and removes them.

MessageBox_Parts_Cleanup_BizTalkMsgBoxDb Removes all message parts that are no longer being referenced by any messages. All messages are made up of one or more message parts, which contain the actual message data. This job searches the messagebox database for message parts that have a reference count of 0 and removes them.

PurgeSubscriptionsJob_BizTalkMsgBoxDb Removes unused subscription predicates left over from things like correlation subscriptions.

MessageBox_DeadProcesses_Cleanup_BizTalkMsgBoxDb Called when BizTalk detects that a BizTalk Server in the group has failed and releases the work that the server was working on so another BizTalk Server can pick that work up.

TrackedMessages_Copy_BizTalkMsgBoxDb Copies tracked message bodies from the engine spool tables into the tracking spool tables in the messagebox database

Test Scenarios for Measuring Maximum Sustainable Performance

This topic describes a test scenario that was implemented to measure the effect of driving a BizTalk system at three different levels of load, sustainable, overdrive, and floodgate. For purposes of these tests, sustainable load is defined as a consistent load that is well within the capabilities of the BizTalk system to handle. Overdrive load is defined as a consistent load that exceeds the capabilities of the BizTalk system to handle. Floodgate load is defined as low load with intermittent peaks of high load. This topic describes the test hardware configuration, the test scenarios, and discusses the findings of each of these tests.

Test System Configuration

The following hardware was used for this test scenario:

- **Six BizTalk Servers** Each equipped with dual 3GHz processors and 2GB of RAM. Each server is using local disks. Two of the BizTalk Servers are configured with an instance of the receiving host, two are configured with an instance of the sending host, and two are configured with an instance of the host used for processing orchestrations.

- **One SQL Server for the master MessageBox database and the non-MessageBox databases** Equipped with eight 2GHz processors and 4 GB of RAM. This server is connected to a fast SAN disk subsystem via fiber. Message publication is disabled.

- **Three SQL Servers for the Messagebox databases that are being published to** Equipped with four 2GHz processors and 4GB of RAM. These servers are each connected to a fast SAN disk subsystem via fiber. The data and transaction log files for the MessageBox database are on separate storage area network (SAN) logical unit numbers (LUNs).

- **Load driver client machine** Equipped with dual 3GHz processors and 2GB of RAM. This server was used to generate the load for testing the BizTalk system.

The topology used for the load tests is illustrated below.

Topology used for load tests

The Test Scenario

The test scenario is very simple. The load generator tool Loadgen was installed on the load driver server and was used to send copies of a file to shares monitored by the file adapter. The load generation tool distributes copies of the input file instance evenly across the file shares. The BizTalk File adapter is configured to monitor the file shares and publish the messages into the MessageBox. A simple orchestration that contains only a receive shape and a send shape subscribes to the published message. Messages that are published back into the MessageBox by the orchestration are picked up by a file send port and sent to a common share, defined on the SAN. Files arriving on the output SAN share are immediately deleted in order to avoid file buildup on that share during long test runs.

Four hosts were defined for the scenario one each for the receive location, the orchestrations, the send port, and for tracking. For the purposes of observing engine backlog behavior, tracking is completely turned off during the test runs. Tracking is only enabled for pipelines and orchestrations by default so tracking was explicitly turned off in the BizTalk Administrator for the orchestration and pipeline that was used.

No instances of the tracking host were created since tracking was turned off to isolate core MessageBox behavior for these tests.

A simple schema was used and the instance files used for the test were all 10KB in size. No mapping or pipeline components were used inbound or outbound in order to keep the test scenario simple and focus observations on the behavior of the MessageBox.

The parameters measured in this test are as follows:

- The growth rate of the MessageBox backlog as measured by the **Spool Size** counter that is available with the **BizTalk:MessageBox:General Counters** performance object. Messagebox backlog is a key indicator of sustainability. Clearly, the MessageBox cannot continue to grow indefinitely without eventually running into problems. So, the depth of the MessageBox database backlog, monitored over time, is used to evaluate sustainability. The MessageBox table named spool contains a record for each message in the system (active or waiting to be garbage collected). Monitoring the number of rows in this table, and the number of messages received per second, while increasing system load provides an easy way to find the maximum sustainable throughput. This measurement is also referred to as the **spool table depth** or **spool depth**.

- The number of documents received per second as measured by the **Documents received/Sec** counter that is available with the **BizTalk:Messaging** performance object.

- The physical disk idle time for the MessageBox data file disk as measured by the **% Idle Time** counter available with the **LogicalDisk** performance object.

- The CPU utilization (%) for the MessageBox server as measured by the **%Processor Time** counter available with the **Processor** performance object.

- The lock timeouts per second on the MessageBox database as measured by the **Lock Timeouts/sec** counter available with the **SQLServer:Locks** performance object.

- The time in seconds for the most recent run of the SQL agent job that cleans up message box tables associated with removed messages. This is measured by the **MsgBox Msg Cleanup(Purge Jobs)** counter available with the **BizTalk:MessageBox:General Counters** performance object.

- The time in seconds for the most recent run of the SQL agent job which cleans up message box tables associated with removed message parts. This is measured by the **MsgBox Parts Cleanup(Purge Jobs)** counter available with the **BizTalk:MessageBox:General Counters** performance object.

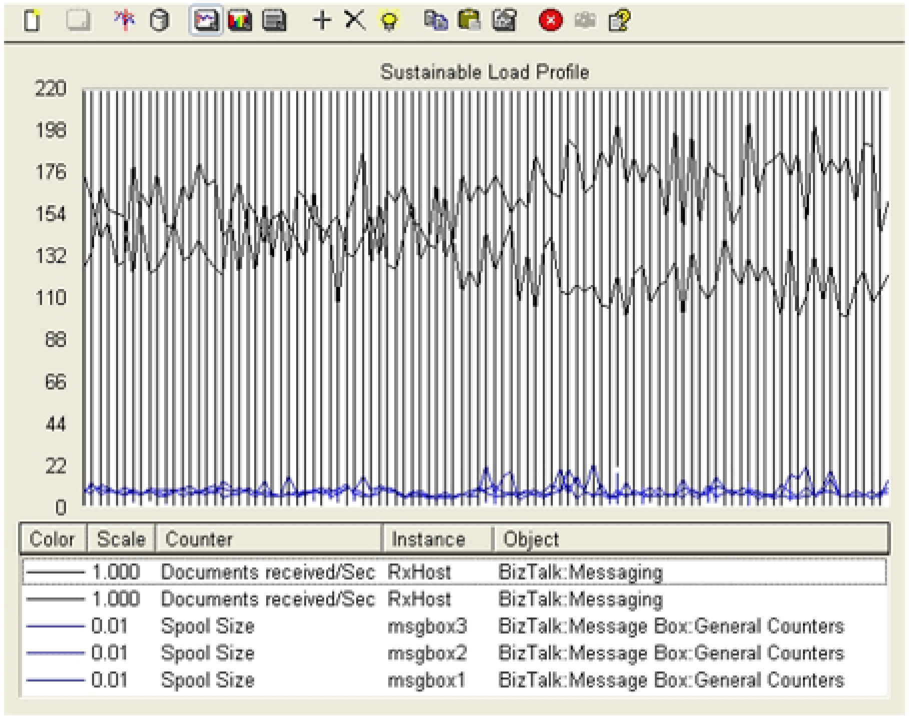
When testing to determine the maximum sustainable throughput, input load was increased until either the spool table started to grow indefinitely or the number of messages received per second hit a plateau. The maximum sustainable throughput of the system is typically estimated as the load required to reach either of these thresholds.

Sustainable Load Test

For the first test, the system was driven at a level that closely approaches its maximum sustainable throughput so that observations of a healthy system can be made.

The following graph shows key indicators after using this approach to find the maximum sustainable throughput of the test system.

Load profile of sustainable load test



Color	Scale	Counter	Instance	Object
——	1.000	Documents received/Sec	RxHost	BizTalk:Messaging
——	1.000	Documents received/Sec	RxHost	BizTalk:Messaging
——	0.01	Spool Size	msgbox3	BizTalk:Message Box:General Counters
——	0.01	Spool Size	msgbox2	BizTalk:Message Box:General Counters
——	0.01	Spool Size	msgbox1	BizTalk:Message Box:General Counters

The black lines at the top of the graph shows the total messages received per second by the system (i.e., for both of the receiving servers), the lines at the bottom of the graph indicate the MessageBox spool depth on each of the SQL Servers. What this graph shows is that, for the hour of the test, the spool depth was stable and not growing and making the sustainable throughput equal to the number of messages received per second, in this case ~ 290 messages per second.

Part of any analysis of a BizTalk deployment performance should include checking some key indicators to understand resource bottlenecks. The key measures and their values used for this deployment running under maximum sustainable throughput were as follows:

CPU Utilization

Server	Average CPU Utilization
BizTalk Servers	55%
SQL Server (Master MessageBox Server)	76%
SQL Server (Other MessageBox Servers)	83%

Physical Disk Idle Time

Server	Average Disk Idle Time
Average for all SQL Servers	69%

SQL Locks on SQL Server

Parameter	Value
Average Total Lock Timeouts per second (per SQL Server)	1980
Average Total Lock Wait Time (ms)	495

Cleanup Jobs

SQL Agent Job	Run time in seconds
MessageBox_Message_Cleanup_BizTalkMsgBoxDb	113
MessageBox_Parts_Cleanup_BizTalkMsgBoxDb	43

During this test no errors were generated in the BizTalk or SQL Server application log.

From this data, we can draw the following conclusions:

- There are no obvious resource bottlenecks in our system.
- All of these indicators are well within healthy limits.

- CPU and Disk Idle Times show that there is plenty of headroom and they are not even close to being pegged.

- The SQL lock indicators look good, **Lock Timeouts/sec** doesn't start to become an issue until around 5000 or so (depending on your SQL Server) and Lock Wait times under 1 second are also healthy.

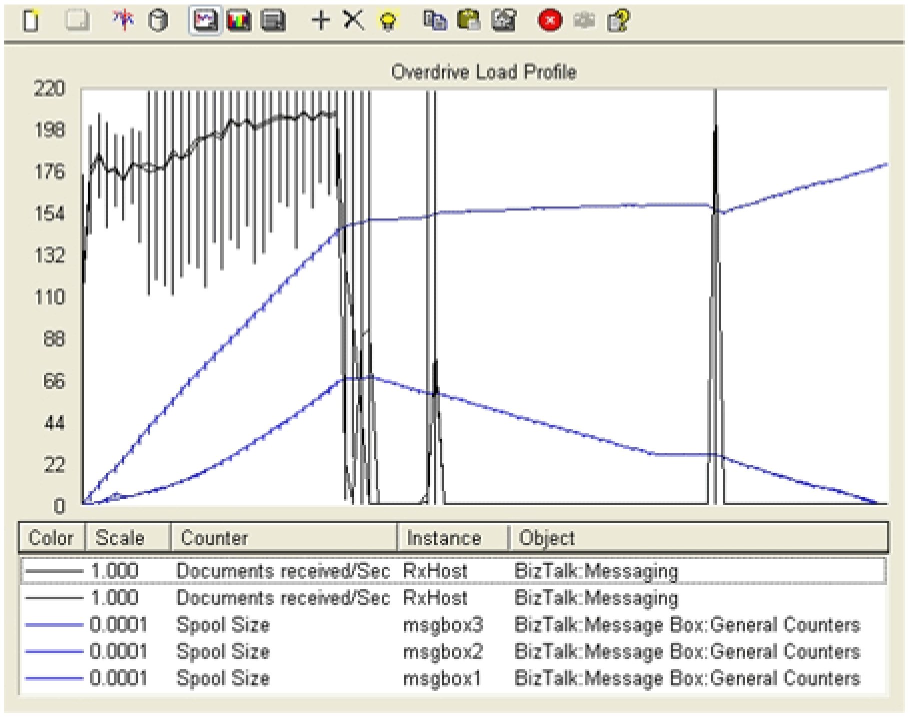
- Finally, the cleanup jobs are completing successfully every time and are taking 2 minutes or less to run. If the cleanup jobs start failing often, or start taking over two minute, this is an indication that the system is being over-driven and will typically be accompanied by an increasing spool depth.

Now that we have shown how to find the maximum sustainable throughput and seen what the key indicators look like for a sustainable, healthy system, let's explore some behavior associated with a system that is receiving faster than it is processing and collecting garbage.

Overdrive Load Test

To simulate a continuously overdriven system, the load generation tool was configured to send in about 410 msgs/sec, 120 msgs/sec more than the measured maximum sustainable throughput. The test was designed not only to overdrive the system, but also to get an idea of how long it would take to recover from a spool backlog depth of around 2 million records. To accomplish this, the system was driven at the increased rate until the spool depth was around 2 million records. Once the spool depth reached the desired level then no further load was generated. To ensure that the BizTalk throttling mechanism did not throttle the receive host, which would prevent the accumulation of a spool table backlog, the **Message count in database throttling threshold** for the receive host was changed from the default value of 50000 to 2000000. For information about changing the default host throttling settings see How to Modify the Default Host Throttling Settings. The following graph shows the same indicators as in the graph above.

Load profile of overdrive load test



Color	Scale	Counter	Instance	Object
——	1.000	Documents received/Sec	RxHost	BizTalk:Messaging
——	1.000	Documents received/Sec	RxHost	BizTalk:Messaging
——	0.0001	Spool Size	msgbox3	BizTalk:Message Box:General Counters
——	0.0001	Spool Size	msgbox2	BizTalk:Message Box:General Counters
——	0.0001	Spool Size	msgbox1	BizTalk:Message Box:General Counters

As can be seen from the graph, the spool depth started building up immediately, peaking at just above 2 million records. At this rate, it took just under 3 hours to get to the targeted 2 million record backlog. After the load was stopped, it took around 6 hours for the cleanup jobs to recover from the backlog.

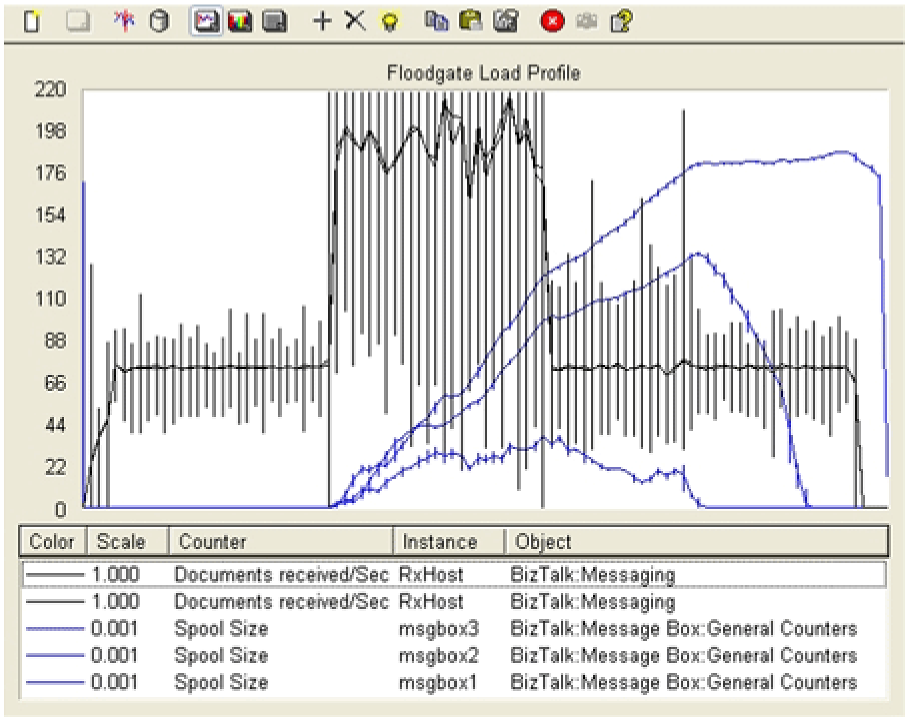
Floodgate Load Test

There are a number of scenarios where just a few large peaks (also known as **floodgate events**) of messages arrive at the system each day. Between these peaks, the throughput can be very low. Examples of these types of scenarios include equity trading (for example at market open and market close) and banking systems (for example during end of day transaction reconciliation).

Other types of events can cause backlog behavior similar to floodgate events. For example, if a partner send address goes off line so that messages destined for that address must be re-tried and/or suspended, this can result in backlog building up. When the partner comes back on line, there may be a large number of suspended messages that need to be resumed, resulting in another type of floodgate event. The following test of the system illustrates this behavior.

For this test the system was initially driven at around half the maximum sustainable throughput which of course was very stable. Then, to simulate a floodgate event, the load generation tool was configured to send in about 410 msgs/sec (the same as for the overdrive test). The resultant load profile measuring messages received per second and spool depth is displayed below.

Load profile of floodgate load test



Color	Scale	Counter	Instance	Object
—	1.000	Documents received/Sec	RxHost	BizTalk:Messaging
—	1.000	Documents received/Sec	RxHost	BizTalk:Messaging
—	0.001	Spool Size	msgbox3	BizTalk:Message Box:General Counters
—	0.001	Spool Size	msgbox2	BizTalk:Message Box:General Counters
—	0.001	Spool Size	msgbox1	BizTalk:Message Box:General Counters

Note from the graph that the spool tables quickly built up a backlog during the floodgate event. However, because the event was relatively short lived and the subsequent receive rate after the event was below the maximum sustainable rate, the cleanup jobs were able to run and recover from the event without requiring a system receive outage.

Of course, every system is different, so “your mileage will vary”. The best way to verify that you can recover is to test with a representative load before going into production.

Recommendations When Testing Engine Performance

The following guidelines should be followed when testing BizTalk engine performance:

Know your load behavior profile As the three load tests have illustrated, it is critical to know the profile of your load in terms of messages processed over time. The better this is understood, the more accurately you can test and adjust your system capacity. If all you know is your peak throughput requirement, then the most conservative approach would be to size your system so that your maximum sustainable throughput is the same or higher than your peak load. However, if you know that you have predictable peaks and valleys in your load, you can better optimize your system to recover between peaks, resulting in a smaller, less expensive overall deployment.

Test performance early Avoid falling into the trap of investing significant effort in designing and testing the functionality of your solution, but waiting until the last minute to test performance on production hardware. Run performance tests on your system that simulate the expected load profile as early as you possibly can in your development cycle. If you have to change anything in your design or architecture to achieve your goals knowing this early will give you time to adjust and test again.

Emulate your expected load profile when testing performance There are two primary components to this:

1. Emulate the load profile over time.

2. Run the test long enough to evaluate if it is sustainable.

If, as is commonly the case, your cycles are daily in nature you should plan to run performance tests for at least one day to validate sustainability. The longer that you run the tests, the better.

Emulate the production configuration For example, the number and type of ports, the host and host instance configuration, database configuration, and adapter setup. Don't assume that configuration changes will not significantly impact performance.

Use real messages Message sizes and message complexity affect performance, so be sure and test with the same message schemas and instances that you plan to see in production.

Emulate your normal operations during performance tests Though the load tests did not include them, standard operations activities such as periodic database queries, backups, and purging will affect your sustainable throughput, so make sure you are performing these tasks during your performance and capacity test runs. This includes both DTA and BAM tracking, if you plan to use them in production.

The speed of the I/O subsystem for the MessageBox is a key success factor The load tests that were performed made use of a fast SAN for the MessageBox database files that is dedicated to this build-out. Even in this case, the cleanup jobs were able to drive the idle time to near zero for the SQL data file. The I/O subsystem is frequently a bottleneck in production systems. A common strategy to optimize SQL I/O is to place the database data file(s) and log file(s) on separate physical drives, if possible.

Make sure the SQL Agent is running on all MessageBox servers Clearly, the cleanup jobs will never run if the SQL Agent is not running, so make sure these are running.

Spool depth and cleanup job run time are key indicators Regardless of other indicators, these two measures will give you a quick and easy way to assess if your system is being overdriven or not.

Planning for High Availability

Many companies use Microsoft BizTalk Server to process data that their business depends on. For these companies, the consequences of a prolonged downtime because of a hardware outage can mean diminished productivity and profitability. This section contains information to help you understand, plan, and implement a highly available BizTalk Server 2006 environment.

In This Section

Planning Your Platform for Fault Tolerance

Microsoft® BizTalk® Server 2006 is built on the Microsoft Windows® and Microsoft SQL Server™ platforms. The ability of BizTalk Server to survive or recover from a disaster depends on the ability of the underlying platform to survive or recover.

For your Business Activity Monitoring (BAM) databases and your MessageBox database, it is recommended that you do the following:

- Set up fail-over clustering, which is available in SQL Server Enterprise Edition. Fail-over clustering enables SQL Server to automatically switch the processing for an instance of SQL Server from a failed server to a working server.

  The BAM Primary Import database collects event data. In the event of a disaster, data that was written to the BAM Primary Import database since the last backup will be lost. Because there is no way to regenerate lost events, it is especially important that you enable fail over clustering on your BAM Primary Import database.

- Use SQL Server RAID (redundant array of independent disks), especially for the MessageBox database and the BAM Primary Import database.

Use the following resources to design your Windows and SQL Server deployments for fault tolerance. Take time to learn about hardware and server redundancy technologies, such as clustering and disk mirroring, to prevent service outages and data loss.

- Chapter 15 - High Availability Options, SQL Server Resource Kit.

  The Microsoft SQL Server Resource Kit covers a wide range of administrative and deployment planning areas. Chapter 15 covers planning for fault tolerance and recovery.

- Microsoft Windows Server 2003 Deployment Kit.

  The Microsoft Windows Server™ 2003 Deployment Kit provides guidelines and recommended processes for designing and deploying Windows Server 2003 technologies.

- Windows Server 2003 Deployment Kit: Planning Server Deployments.

  This book provides information about planning server storage, and information about designing and deploying file servers, print servers, and terminal servers in medium and large organizations.

  You can also use the guidelines in this book to maximize the availability and scalability of your servers by planning for remote server management, designing and deploying server clusters, and designing and deploying Network Load Balancing clusters.

Backing up your platform

After you have configured your system, prepare full backups of your servers so you can quickly restore an identical server in the event of data loss.

To back up your platform, perform the documented backup procedures for each of the following technologies:

- Microsoft Windows® 2000 Server or Microsoft Windows Server 2003 Standard, Enterprise, or Datacenter Edition
- Internet Information Services (IIS)
- Microsoft SQL Server
- Windows SharePoint™ Services, which is used by Business Activity Services (BAS)

Thoroughly test your backup and restore procedures, and put them in a safe, remote location.

Creating a Highly Available BizTalk Server Environment

This section describes how to provide high availability for the data and communications in Microsoft® BizTalk® Server 2006 when integrating disparate systems and applications. BizTalk Server 2006 separates the data from the hosts that process the data, enabling you to resolve availability issues by scaling the databases and hosts independently.

Demonstrating High Availability

High availability for BizTalk Server 2006 focuses on recovering functional components that might disrupt availability in a BizTalk Server deployment.

To demonstrate high availability in BizTalk Server 2006, you have to apply failure and measure the product's effectiveness in recovery. A highly available BizTalk Server deployment makes errors and failures transparent to external applications and systems, and makes sure that all services continue functioning correctly with minimal disruption.

Designing for High Availability

Designing a BizTalk Server 2006 deployment that provides high availability involves implementing redundancy for each functional component involved in an application integration or business process integration scenario. BizTalk Server 2006 simplifies the implementation of these scenarios by conceptually separating the data from the hosts that process the data. So providing high availability for BizTalk Server 2006 involves running multiple host instances and clustering the BizTalk Server databases, as follows:

- **Architecture for BizTalk Hosts.** BizTalk Server 2006 lets you separate hosts and run multiple host instances to provide high availability for key functions such as receiving messages, processing orchestrations, and sending messages. These hosts do not require any additional clustering or load-balancing mechanism because BizTalk Server 2006 automatically distributes workload across multiple computers through host instances. However, hosts running the receive handler for the HTTP, SOAP, and BizTalk Message Queuing (MSMQT) adapters require a load-balancing mechanism such as Network Load Balancing (NLB) to provide high availability.

- **Architecture for BizTalk Server databases.** High availability for the BizTalk Server 2006 databases typically consists of two or more database computers configured in an active/passive server cluster configuration. These computers share a common disk resource (such as a RAID5 SCSI disk array or storage area network) and use Windows Clustering to provide backup redundancy and fault tolerance.

This document provides information about how to address high availability in each of these categories. Because BizTalk Server 2006 is built on Microsoft Windows Server™ 2003 (or Windows® 2000 Server) and Microsoft SQL Server™ 2000 or 2005, make sure that you deploy these products with high availability before configuring hosts for BizTalk Server 2006. The following links include information about providing high availability for these underlying products:

- **Chapter 15 - High Availability Options, SQL Server Resource Kit**, available at http://go.microsoft.com/fwlink/?LinkId=24431.

  The Microsoft SQL Server Resource Kit covers many administrative and deployment planning areas. Chapter 15 covers planning for fault tolerance and recovery.

- **Microsoft Windows Server 2003 Deployment Kit**, available at http://go.microsoft.com/fwlink/?LinkId=24432.

  The Microsoft Windows Server 2003 Deployment Kit provides guidelines and recommended processes for designing and deploying Windows Server 2003 technologies.

- **Windows Server 2003 Deployment Kit: Planning Server Deployments**, available at http://go.microsoft.com/fwlink/?LinkId=24433.

  This book provides information about planning server storage, and information about designing and deploying file servers, print servers, and terminal servers in medium and large organizations.

Sample BizTalk Server High-Availability Scenarios

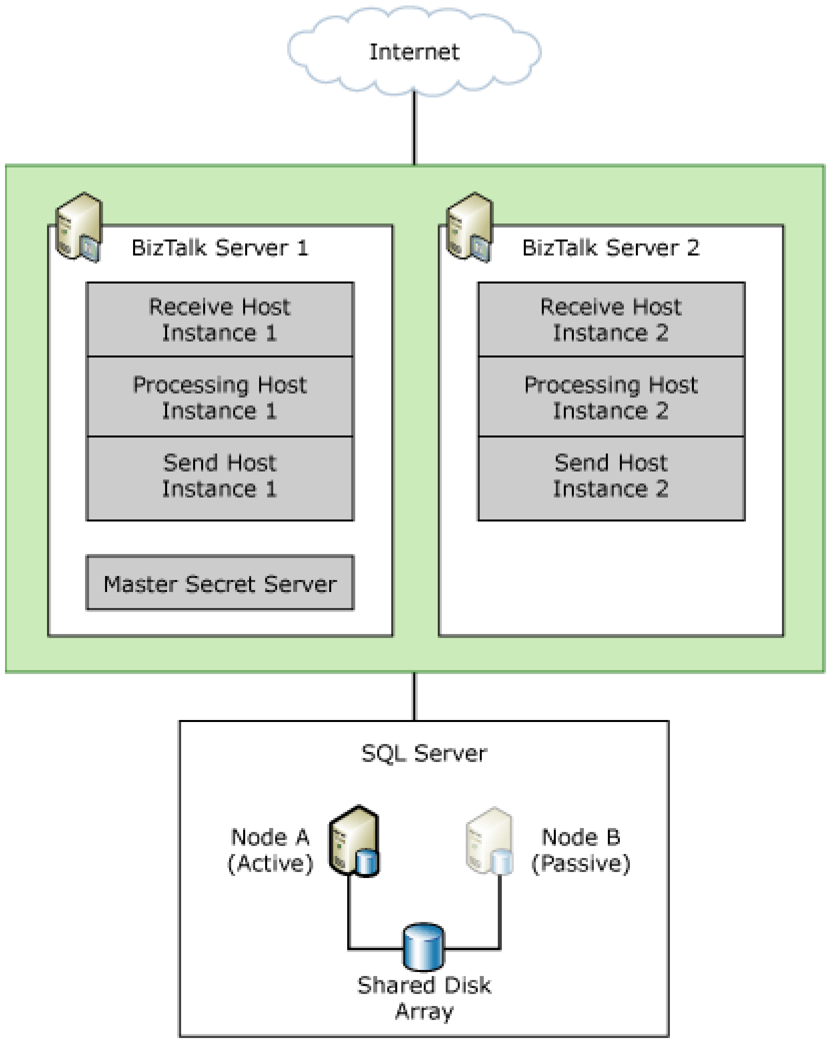
This section describes the scenarios in Microsoft® BizTalk® Server 2006 that provide high availability through scaled-out tiers of hosts. By separating areas of functionality into different hosts and tiers in BizTalk Server 2006, administrators can provide redundancy for each host and scale them independently of other hosts. To provide high availability for each functional area, you create separate hosts for each primary function—receiving, processing, sending, and tracking—and cluster the BizTalk Server 2006 databases. For a BizTalk Server 2006 solution that provides complete high availability, you must also cluster the master secret server.

Small BizTalk Server 2006 Deployments

The smallest available BizTalk Server 2006 deployment is made up of two computers that have an active/active cluster configuration for SQL Server 2000. Both computers contain instances of all the BizTalk Hosts in the environment. If one computer fails or encounters errors, the other computer maintains service availability for both SQL Server and BizTalk Server. This configuration is not highly available because you cannot cluster the master secret server. For more information about clustering the master secret server, see High Availability for Enterprise Single Sign-On.

For small BizTalk Server 2006 deployments that contain fewer than five computers, we recommend that the SQL Server cluster that contains the BizTalk Server databases runs on separate computers from the BizTalk Server computers. The BizTalk Server computers can run all the BizTalk Hosts (receive, process, and send). To make this deployment highly available, cluster the SQL Server, and have two BizTalk Servers, each running an instance of each host in your environment.

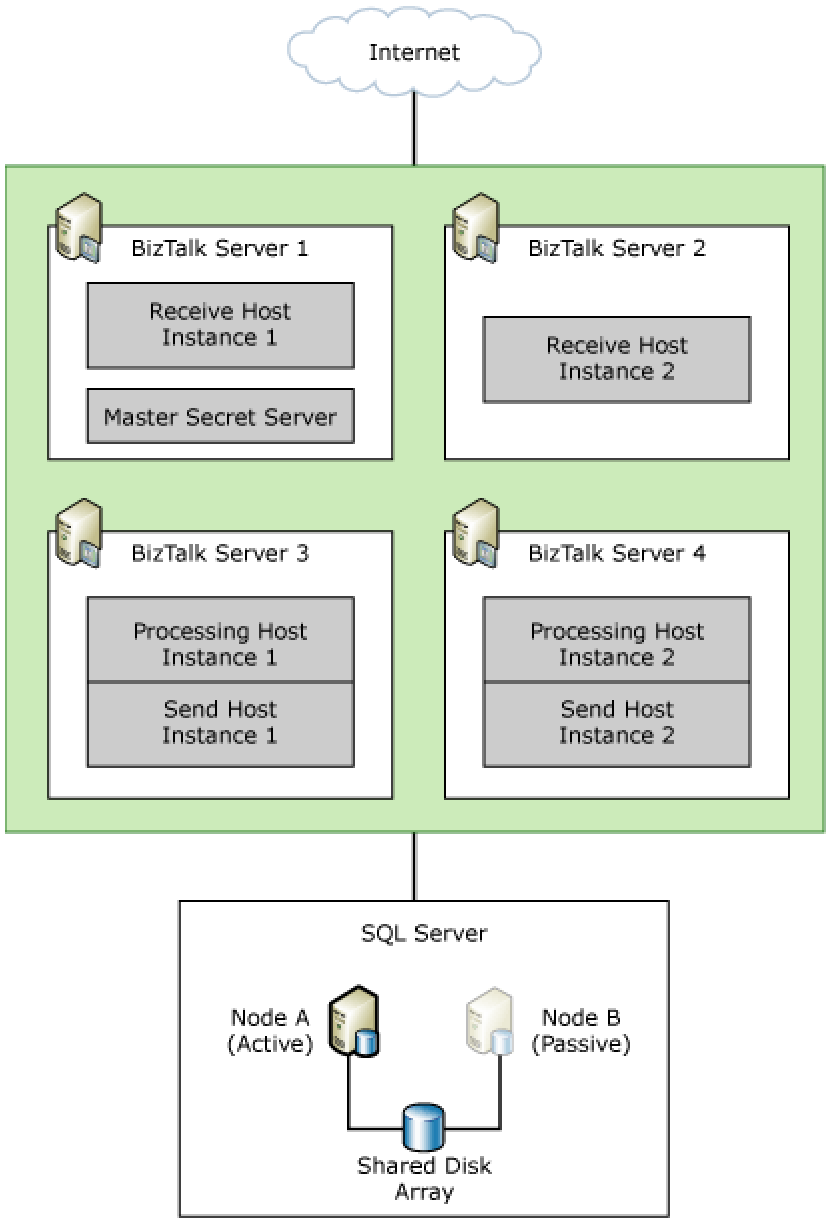
The following figure shows a small BizTalk Server deployment that is highly available.



Medium-Sized BizTalk Server 2006 Deployments

For medium-sized deployments that contain five to ten computers, we recommend that you cluster the SQL Server that contains the BizTalk Server databases. If your scenario is receive-intensive, you may want to dedicate two BizTalk Servers to run the receiving host instances to provide a highly available solution. You can then have two more computers that are running both the processing and send host instances. To make this a highly available deployment, create host instances of both the processing and send hosts on two BizTalk Servers. Similarly, if you have a processing-intensive scenario, you may want to dedicate two BizTalk Servers to run the processing host instances, and have the remaining two BizTalk Servers running instances of both the receive and send hosts.

The following figure shows a highly available medium-sized BizTalk Server deployment with two BizTalk Servers dedicated to receiving operations.



For more information about high availability for Enterprise Single Sign-On, see High Availability for Enterprise Single Sign-On.

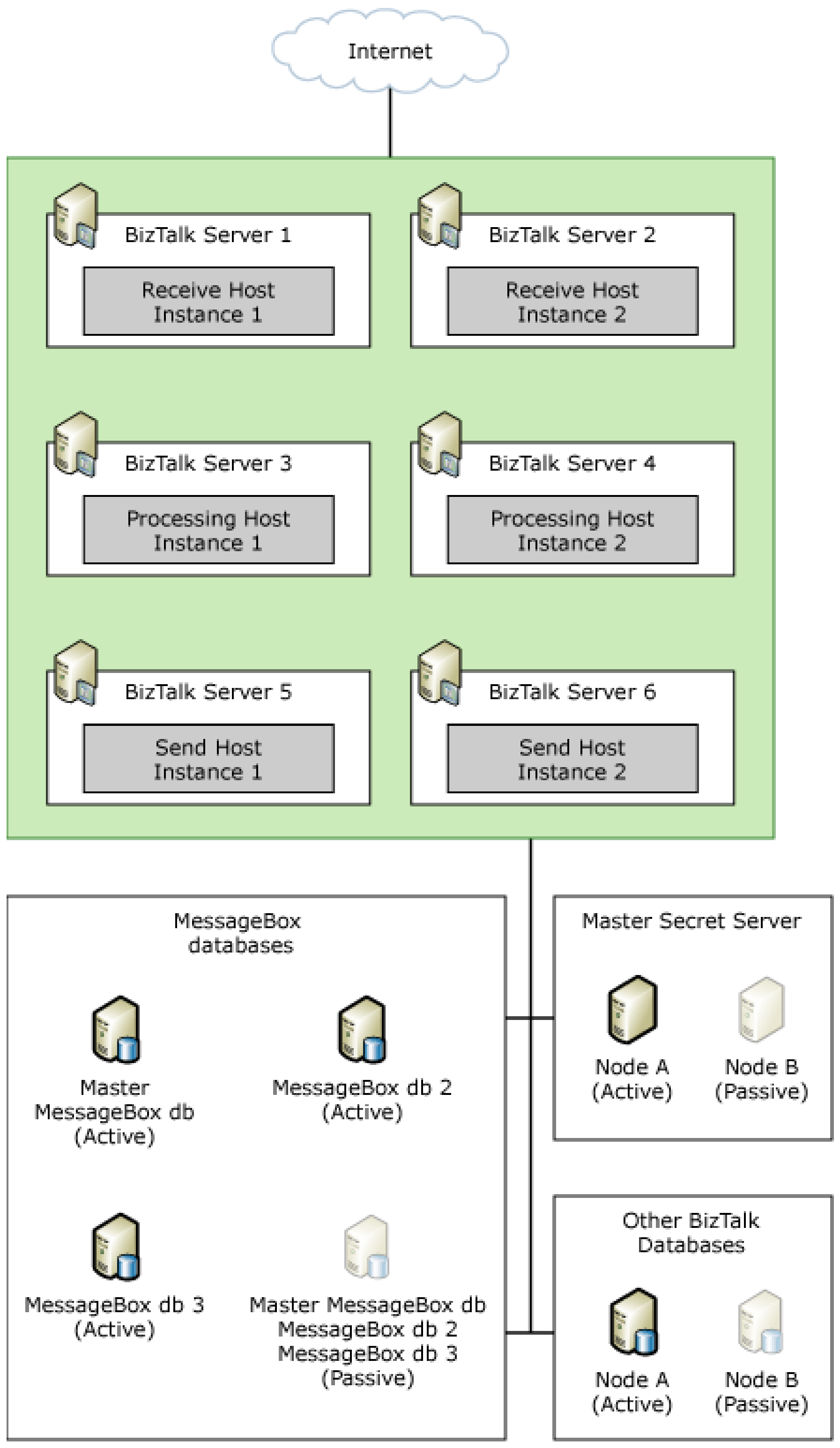
Large-Scale BizTalk Server 2006 Deployments

For large-scale deployments that contain 10 or more computers, dedicate separate BizTalk Server computers for the receiving, processing, and sending functions. Also, if you have many BizTalk Server computers in a group, you can include additional MessageBox database computers to increase performance. In this case, cluster the MessageBox databases to provide high availability.

Such a distributed configuration demonstrates the flexibility of BizTalk Server 2006 because it lets you to evaluate and identify specific points of failure in your deployment, and then strategically allocate resources to reduce those points in the system. Today's dynamic business environment demands such flexibility because workload fluctuations and business requirements can change daily.

Instead of spending additional money to upgrade or acquire new hardware, you can use existing resources to achieve high availability by moving resources from computers that consume few resources to computers that are resource-intensive.

The following figure shows a large-scale BizTalk Server 2006 deployment.



For more information about high availability for Enterprise Single Sign-On, see High Availability for Enterprise Single Sign-On.

High Availability for the BizTalk Base EDI Adapter

High Availability for the BizTalk Base EDI adapter is provided by configuring two key components of the BizTalk Base EDI adapter to be highly available, the EDI Subsystem and the EDI adapter handlers. Implementation of high availability for these components requires that you install multiple BizTalk servers into a BizTalk group and that you create and use a highly available file share.

The EDI Subsystem on BizTalk Server 2006 is responsible for validating, parsing, and/or serializing EDI documents.

The EDI adapter handlers are responsible for receiving documents into and sending documents out of the EDI Subsystem.

Providing high availability for the EDI Subsystem through the use of a heartbeat mechanism

High availability can be provided for the EDI Subsystem in a BizTalk Server environment if multiple BizTalk Servers in a BizTalk Server group are configured with the BizTalk EDI Adapter.

If a BizTalk group contains multiple BizTalk Servers that are configured with the EDI Adapter, one and only one of the servers in the group assumes the role of the EDI translation server and a heartbeat mechanism is utilized to detect the availability of this server. If for any reason the EDI translation server becomes unavailable then another BizTalk Server configured with the EDI adapter will assume the role of EDI translation server and resume the processing of EDI documents. The heartbeat mechanism is used as follows:

1. If more than one BizTalk Server in a group is configured with the EDI Adapter, then one of the BizTalk Servers in the group is designated as the EDI translation server. The EDI translation server is the server that actually validates, parses, and/or serializes EDI documents. For more information about the role of the EDI Subsystem and the EDI Adapter handlers, see What Is the Base EDI Adapter?.

2. The EDI translation server is designated by the **servername** field in the **heartbeat** table of the BizTalkEDIDb database. The EDI translation server is selected from the list of BizTalk servers with a running instance of the EDI Subsystem service. This list of servers is maintained in the **ediservers** table of the BizTalkEDIDb database.

3. The EDI translation server periodically writes a new timestamp to the **heartbeat** field of the **heartbeat** table and each non-translation server in the group periodically reads the timestamp and compares it to the current time as returned by the database server. The interval in seconds that the EDI translation server updates the heartbeat field and that the non-translation servers check the heartbeat field is stored in the **Heartbeat** parameter listed in the esp.ini file. The **Heartbeat** parameter is listed in the [Tuning] section of the esp.ini file.

4. If a failover period as determined by multiplying the **Heartbeat** parameter by the **NoHeartBeatTreshold** parameter is exceeded, then it is assumed that the EDI translation server is down and a new EDI translation server is chosen. The **servername** field in the **heartbeat** table of the BizTalkEDIDb database is then updated with this information and the new EDI translation server resumes processing of EDI documents.

The **Heartbeat** and **NoHeartbeatThreshold** parameters are stored in the **[Tuning]** section of the esp.ini file. The esp.ini file is stored in the [documentshome] directory which is specified in the documentshome field of the parame table of the BizTalkEDIDb database. These parameters can be left at their default values or changed if you would like to tune this functionality.

Parameter	**Value**
Heartbeat	Set to the number of seconds used as the Heartbeat interval. The Heartbeat interval is the number of seconds difference there can be between the timestamp in the heartbeat table (updated by the EDI translation server) and the current time (as measured by the database server). The heartbeat table is in the BizTalkEDIDb database. Minimum value: 1 Maximum value: 60 Default value: 10
NoHeartbeatThreshold	Set the number of heartbeat intervals after which if an EDI translation server is not responding, a new EDI translation server is selected. For example, when Heartbeat=10 and NoHeartBeatThreshold=4, the non-translation servers will wait, at most, between 40 and 49 seconds before assuming that the translation server has failed. Minimum value: 1 Maximum value: 60 Default value: 4

Ensuring that the EDI Subsystem on all BizTalk servers are using a common, highly available [documentshome] directory

The EDI [documentshome] directory is used as the working directory by the EDI Subsystem that is running on each BizTalk Server in a BizTalk Server group. Since the EDI Subsystem on each BizTalk Server in the group needs access to this directory it should be configured as a file share and should be made highly available. Complete the following steps to configure the [documentshome] directory as a file share and make it highly available:

If you have not yet configured the Base EDI adapter with the BizTalk Configuration program, follow these steps:

1. Create a highly available file share for use as the EDI Subsystem working directory. Grant the <Domain>\EDI Subsystem Users group "Full Control" access to this file share at the file system and share level. The recommended method for creating a highly available file share is to create a clustered file share resource on a Windows Server Cluster. For more information about configuring a Windows Server Cluster and creating a clustered file share resource, please see the Windows Server online help.

2. Launch the BizTalk Configuration program and specify the location of the file share that you created as the **Documents Home** parameter for the EDI feature.

3. Complete the configuration of your BizTalk Server. The file share will be used as the working directory for the EDI Subsystem service on all BizTalk servers in the BizTalk group.

If you have already configured the Base EDI adapter with the BizTalk Configuration program you have two options. The first option is to unconfigure the EDI feature and then re-configure the EDI feature using the steps above. The second option is to follow these steps:

1. Stop the EDI Subsystem service on each BizTalk Server in the group. The EDI Subsystem service is listed as the **BizTalk Base EDI service** in the Services control panel applet.

2. Create a highly available file share for use as the [documentshome] directory as described earlier.

3. Copy all of the file and subdirectories under the current [documentshome] directory to the highly available file share. The current [documentshome] directory is specified in the documentshome field of the parame table of the BizTalkEDIDb database.

4. Modify the documentshome field of the parame table of the BizTalkEDIDb database to point to the highly available file share.

5. Start the EDI Subsystem service on each BizTalk Server in the group.

Providing high availability for the EDI adapter handlers

High availability can be provided for the EDI adapter handlers in a BizTalk Server environment if multiple BizTalk Servers in a BizTalk Server group are configured with the BizTalk Base EDI Adapter. To provide high availability for the EDI adapter handlers you create instances of the host for the EDI adapter handlers on multiple servers in a BizTalk server group.

Each BizTalk server that houses an instance of the host for the EDI adapter handlers must also be running the EDI Subsystem in order to process EDI documents. Ensure that the EDI Subsystem is running on each BizTalk server in the group that is configured to run an instance of the host that contains the EDI adapter handlers.

Incomplete document processing when the EDI Subsystem or EDI adapter handler instance fails

For any given BizTalk server in a group that is running the EDI Adapter, if either the EDI Subsystem or the host instance for the EDI adapter handlers becomes unavailable then EDI messages will not be processed completely. If this occurs, you should either start the failed component (EDI Subsystem or EDI adapter handler host instance) or stop the running component on the BizTalk server. The following table describes the behavior of the Base EDI adapter if one of the components (EDI Subsystem or EDI adapter handler host instance) is down while the other is running on a given BizTalk Server:

Failed component	Adapter behavior for Inbound Scenario (EDI -> XML)	Adapter behavior for Outbound scenario (XML -> EDI)
EDI Subsystem	EDI files will not be picked up; no event log messages will be generated.	The EDI adapter handler issues a warning message to the event log. The adapter handler will retry with a 1-minute interval until the EDI Subsystem is restarted. The messages will then be processed and no further event log messages will be generated.
EDI adapter handler host instance	EDI files will be picked up and translated to XML but will have a status of "In internal format". As soon as the adapter handler host instance is started the files will be processed. No error or warning messages will be generated	Assuming the XML file is picked up by a different host, the message will remain in a "Ready to run" status until the EDI adapter handler host instance is started. No event log messages will be generated.

Configuring the EDI receive handler to monitor a highly available file share

To provide high availability for the location that the EDI receive handler picks up documents from, you should configure the EDI receive handler to pick up documents from a clustered file share. To modify the location that the EDI receive handler picks up documents from, follow these steps:

1. In the **BizTalk Administration** console, click to expand **BizTalk Server 2006 Administration**, click to expand **BizTalk Group [<servername>:<management database>]**, click to expand **Platform Settings**, and click to expand **Adapters**. The list of adapters appears under the folder.

2. Click the EDI adapter, right-click the EDI receive handler in the right-hand pane and select **Properties** from the short cut menu to display the **EDI - Adapter Handler Properties** dialog box.

3. Click the **Properties** button to display the **EDI Transport Properties** dialog box.

4. Click to expand **Connector Properties**, click to expand **File System**, click to expand **Account**.

5. Enter the location for the file share into the **Folder (full path or UNC)** text box.

6. Click **OK** and click **OK** again.

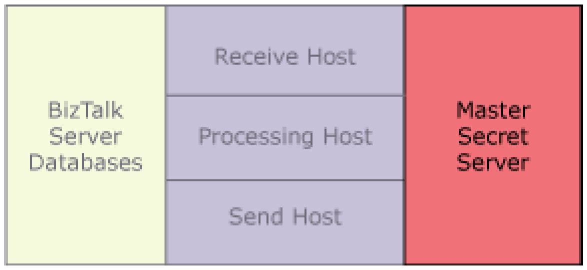
High Availability for Enterprise Single Sign-On

Even if you do not use the Enterprise Single-Sign-On (SSO) functionality for mapping credentials and single sign-on, SSO is a critical part of the overall Microsoft® BizTalk® Server 2006 infrastructure, because BizTalk Server uses SSO to help secure information for the receive locations.

You must configure the first computer where you install the SSO service as the master secret server. The master secret server is the SSO server that stores the master secret (encryption key). The master secret is the encryption key that the SSO system uses to encrypt and decrypt the data it stores in the Credential database.

If an SSO server fails, and if you have other BizTalk Server computers (and therefore SSO servers) running the same host instance, the other SSO servers continue doing their work. This means that the master secret server still functions correctly, and therefore the BizTalk Server processing continues.

If the master secret server fails, all run-time operations that were running before it failed, including decryption of credentials, continue successfully. However, you cannot encrypt new credentials. Therefore, the BizTalk Server environment has a dependency on the availability of the master secret server, as shown in the following figure.



Making the Master Secret Server Available

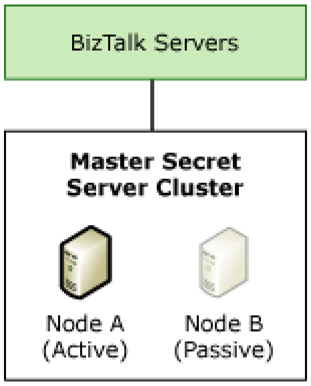
For availability of the SSO system, and therefore of the BizTalk Server environment, it is critical that you back up the master secret as soon as it is generated. If you lose it, you lose the data that the SSO system encrypted by using that master secret. For more information about backing up the master secret, see "Backing Up the Master Secret" in BizTalk Server 2006 Help.

You can make the master secret server available in two ways:

- **Available, but not highly available.** All the SSO servers have the master secret cached in memory, and run-time operations will continue even if the master secret server fails. However, you will not be able to change the configuration of ports or the SSO configuration. The BizTalk Server runtime will continue working without problems, but you cannot make any design changes. You can create a Microsoft Operations Manager (MOM) event to notify you when the master secret server becomes unavailable, and you can then manually promote an SSO server to master secret server and restore the master secret on this server.

  Even if this configuration is not highly available, it can be satisfactory for most scenarios and it is consistent with scaling out the receiving, sending, and processing hosts.

- **Highly available.** To provide redundancy for the master secret server, use Windows Clustering on a separate master secret server cluster, or configure the master secret server on an existing database cluster. The services provided by the master secret server do not consume many resources, and typically do not affect database functionality or performance when installed on a database cluster. The following figure shows how you can make the master secret server highly available.



While this configuration is highly available, it requires additional hardware resources. For more information about clustering the master secret server, see "Clustering the Master Secret Server" in BizTalk Server 2006 Help.

High Availability and the Microsoft Operations Framework

Applying the Microsoft Operations Framework (MOF) process model to the planning and implementation of a highly available Microsoft® BizTalk® Server 2006 solution can help you make sure that you have appropriate processes at the different stages of the release life cycle. By looking ahead at all the life cycle stages where high availability surfaces, you can make the installation, maintenance, and troubleshooting of availability issues in your environment easier.

This section contains information about the MOF processes where you have to consider high-availability tasks.

Microsoft Operations Framework Process Model

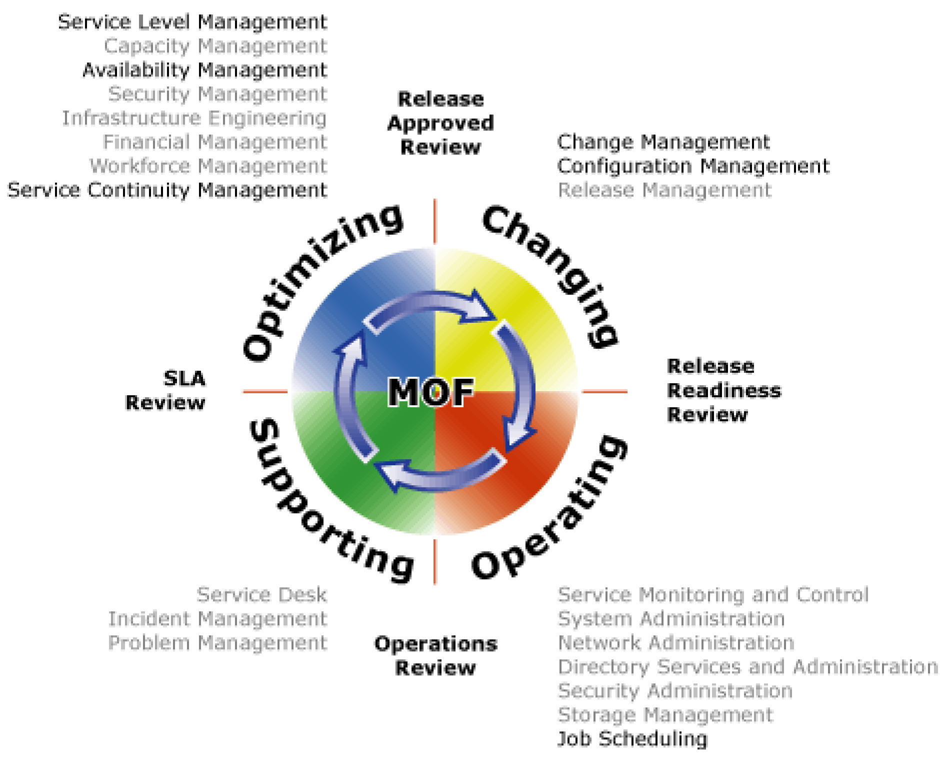
The Microsoft Operations Framework (MOF) provides guidance that enables organizations to achieve mission-critical system reliability, availability, supportability, and manageability of Microsoft products and technologies. MOF provides operational guidance in the form of white papers, operations guides, assessment tools, best practices, case studies, templates, support tools, and services. This guidance addresses the people, process, technology, and management issues pertaining to complex, distributed, and heterogeneous IT environments. For more information about the Microsoft Operations Framework, see http://go.microsoft.com/fwlink/?LinkId=31988.

The MOF process model enables companies to:

- Facilitate consistent IT service management across service solutions.
- Establish a structure for IT functions, processes, and procedures.
- Represent a life-cycle approach.

Central to the MOF process model is its division into four quadrants of operational processes and procedures, named service management functions (SMFs). The SMFs are the foundation-level best practices and prescriptive guidance for operating and maintaining an IT environment.

The following figure shows the MOF processes where you have to consider high availability.



Changing Quadrant

The changing quadrant includes the service management functions (SMFs) required to identify, review, approve, and incorporate change into a managed IT environment. This includes changes in software, hardware, documentation, roles and responsibilities, and also specific process and procedural changes.

Change Management

Change management is responsible for changes in technology, systems, applications, hardware, tools, documentation, and processes, and also changes in roles and responsibilities.

During the change management process, as part of designing your BizTalk Server implementation, you can do the following:

- Determine whether the service level agreement with your partners or customers requires a certain level of availability, uptime, and load-processing capabilities.
- If you are upgrading from BizTalk Server 2000 or BizTalk Server 2002 to BizTalk Server 2006, you must determine whether your existing hardware will satisfy the minimum hardware requirements for BizTalk Server 2006 and the requirements from the service level agreement.
- Determine the best cluster configuration for the BizTalk Server databases for your business needs. The run-time processes write to the BizTalk Management database, MessageBox databases, Tracking Analysis Services database, BAM Analysis database, BAM Star Schema database, BAM Primary Import database, and BAM Archive database. Therefore, these databases are especially important if a disaster occurs, and must have greater priority when determining

what databases to cluster. Only users or tools write to the other databases. For the MessageBox databases, you can consider an active/active/active/passive four-server cluster to minimize the hardware needed.

- Determine whether to cluster the master secret server, or if manually restoring the master secret on another Enterprise Single Sign-On server is satisfactory for your scenario. This solution is available, but not highly available.
- Determine the number of hosts and host instances that you will need to process your expected message load and to provide high availability.
- Create a list of the people that will be involved in the change-management process. This list will include, but is not limited to, the domain administrator, database administrator, infrastructure administrator, BizTalk Server administrator, and IT operations staff.

Configuration Management

Configuration management is responsible for identifying, controlling, and tracking all versions of software, hardware, documentation, processes, procedures, and all other components of the IT environment under the control of change management.

During the configuration management process, you must create a detailed plan for how you are going to implement your highly available solution for BizTalk Server 2006. You must also document the steps that you took to create your solution. At a high level, the steps are:

- The domain controller creates the domain groups and accounts that you will use in your BizTalk Server environment.
- The infrastructure administrator creates the Windows® cluster for the BizTalk Server databases and the Windows cluster for the master secret server.
- The database administrator installs and configures Microsoft SQL Server™ 2000 on the Windows cluster for the BizTalk Server databases.
- The BizTalk Server administrator configures the master secret server cluster.
- The BizTalk Server administrator installs and configures BizTalk Server 2006 on the processing, receiving, and sending servers.
- The BizTalk Server administrator creates the hosts and installs the host instances on the appropriate servers to provide high availability or to increase capacity, or both.

Operating Quadrant

The operating quadrant includes the SMFs required to monitor, control, manage, and administer service solutions daily to achieve and maintain service levels within predetermined parameters.

Job Scheduling

Job scheduling involves the continuous organization of jobs and processes in the most efficient sequence, maximizing system throughput and use to meet service level agreement requirements.

Make sure that you schedule planned downtime, such as scheduled updates, at times when the message load is low (for example, late at night) to minimize the potential effect on your business.

Supporting Quadrant

The supporting quadrant includes the SMFs required to identify, assign, diagnose, track, and resolve incidents, problems, and requests within the approved requirements that are contained in the service level agreements.

Optimizing Quadrant

The optimizing quadrant includes the SMFs that contribute to maintaining business and IT alignment by focusing on decreasing IT costs while maintaining or improving service levels. This includes review of outages and incidents, examination of cost structures, staff assessments, availability and performance analysis, and capacity forecasting.

Service Level Management

The goal of service level management is to maintain and continuously improve the quality of IT service, through a constant cycle of negotiating and monitoring service level requirements. The successful service level management function causes an improvement in quality of service, greater levels of customer productivity, and ideally, a reduction in the overall cost of services provided.

During the service level management process, you can do the following:

- Evaluate how the current environment satisfies your service level agreement requirements.
- Recommend the addition of new servers for processing, receiving, or sending messages to meet the requirements.
- If necessary, recommend creating highly available solutions for points of failure that were not originally mitigated to meet the availability requirements in the service level agreement.

Availability Management

The single goal of availability management is to make sure that your customers can use a particular IT service whenever they want.

For the availability management process, you can establish mechanisms for notifying IT personnel when a hardware failure occurs so that they can fix or replace the hardware as quickly as possible, and mechanisms for notifying IT personnel when the server load is larger than a particular threshold.

Service Continuity Management

The objective of the service continuity management function is to make sure that a specified IT service provides value to the customer if regular-availability solutions fail.

During the service continuity function you must examine what high-availability configuration to implement to make sure that you continue providing your customers with the services they expect even when a planned or unplanned downtime occurs. Examples of unplanned downtime are hardware failures or acts of nature.

Optimizing Resource Usage Through Host Throttling

BizTalk Server 2006 makes use of several different Microsoft technologies each of which can consume a significant portion of the memory, disk, and CPU resources available on the BizTalk Server and the SQL Server that houses the BizTalk Server databases. BizTalk Server 2006 employs a throttling mechanism to help manage the use of available resources to minimize resource use contention. This topic discusses the design of this mechanism, and how to tweak the throttling algorithm for specific scenarios.

In This Section

- What Is Host Throttling?
- How Does BizTalk Implement Host Throttling?
- How to Modify the Default Host Throttling Settings
- Host Throttling Performance Counters
- Throttling Design Recommendations

What Is Host Throttling?

Most of the processing that takes place on a BizTalk Server occurs within a logical entity known as a BizTalk Server host instance, which is a process running as a Windows service or an isolated host process on the BizTalk Server. To manage the use of resources by a host instance process, BizTalk Server utilizes an adjustable throttling mechanism that governs the flow and processing of messages through a host instance. The throttling mechanism moderates the work load of the host instance to ensure that the work load does not exceed the capacity of the host instance or any downstream host instances. The throttling mechanism also prevents a condition known as resource contention that can lower the overall performance of the host instance process or other system processes. Resource contention occurs when one or more processes consume a limited resource to the detriment of themselves and/or another process. For example, the consumption of excessive memory or threads can lead to memory allocation failure or high thread context-switches, which can impact the performance of the process. Resource contention like this can be detrimental to the overall performance of BizTalk Server.

The host throttling mechanism also detects when available resources are being under utilized. If available resources are under utilized then the throttling mechanism allows additional messages to be processed by a host instance. The host throttling mechanism continually monitors if available resources are being over or under utilized and adjusts message flow through the host instance accordingly.

The BizTalk Server host throttling mechanism helps to ensure that the system operates at an optimal and sustainable level.

Host throttling configuration parameters are set on a per host basis in the BizTalk Server Administration console.

Inbound host throttling

Inbound host throttling is applied to host instances that contain receive adapters or orchestrations that publish messages to the MessageBox database. An inbound host throttling condition can be triggered under the following conditions:

- The amount of memory, the number of threads, or the number of database connections used by the host instance exceed the throttling thresholds defined on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. These values are measurable with performance monitor counters available under the **BizTalk:Message Agent** performance object category.
- Downstream hosts are unable to process the messages that are published. This increases the value of the **Message count in database** parameter. The threshold at which the **Message count in database** value triggers a throttling condition is configurable on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. The message count in database can be measured with the **Database Size** counter under the **BizTalk:Message Agent** performance object category.
- The **Message Publishing Incoming Rate** for the host instance exceeds the **Message Publishing Outgoing Rate *** the specified **Rate overdrive factor (percent)** value. The **Rate overdrive factor (percent)** value is defined on the **Message Publishing Throttling Settings** dialog box available from the **Advanced** page of the **Host Properties** dialog box. The message publishing incoming and outgoing rates can be measured with the corresponding performance monitor counters under the **BizTalk:Message Agent** performance object category.
- The default throttling behavior has been modified by setting a registry value or values to control the throttling behavior of a host process. The registry keys for overriding the default throttling behavior are described in the **Registry settings that affect default throttling behavior** section of the topic How to Modify the Default Host Throttling Settings.

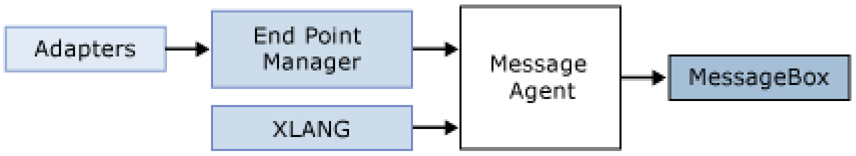
Depending upon the severity of the throttling condition, the following actions are taken:

- A progressive delay in the processing logic of the host instance is implemented. The delay may be implemented when an End Point Manager (EPM) thread receives a batch of messages from the transport adapter, and/or when the EPM submits a batch of messages to be published into the BizTalk MessageBox database. Both the duration of the processing delay and the rate at which the duration is incremented scale with the severity of the throttling condition.

- The number of threads that are available to the End Point Manager (EPM) is restricted. The EPM receives batches of messages from adapters and publishes the messages to the BizTalk MessageBox database. By default, the EPM is configured to use 20 threads per CPU. If the host throttling mechanism detects a stress condition for inbound processing then it can temporarily reduce the number of threads available to the EPM until the stress condition is eliminated. The EPM cannot process messages from transport adapters or deliver message batches to the MessageBox database unless an EPM thread is available to service the inbound message batch.

- The use of memory and other resources is reduced as applicable. BizTalk can send instructions to other service classes to limit memory use by dehydrating running schedules, shrinking memory cache size, and by limiting the usage of memory intensive threads.

Inbound message flow from Adapter to MessageBox



Outbound host throttling

Outbound host throttling is applied to host instances that contain orchestrations or send adapters that receive and deliver or process messages that have been published to the MessageBox. An outbound host throttling condition can be triggered under the following conditions:

- The amount of memory, the number of threads, or the number of database connections used by the host instance exceed the throttling thresholds defined on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. These values are measurable with performance monitor counters available under the **BizTalk:Message Agent** performance object category.

- The **Message Delivery Incoming Rate** for the host instance exceeds the **Message Delivery Outgoing Rate** * the specified **Rate overdrive factor (percent)** value. The **Rate overdrive factor (percent)** value is defined on the **Message Processing Throttling Settings** dialog box available from the **Advanced** page of the **Host Properties** dialog box. The message delivery incoming and outgoing rates can be measured with the corresponding performance monitor counters under the **BizTalk:Message Agent** performance object category.

- The number of messages being processed concurrently by the host instance exceeds the **In-process messages per CPU** * the number of CPUs available on the box. The **In-process messages per CPU** threshold is defined on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. The number of messages being processed concurrently by the host instance can be measured with the **In-Process Message Count** performance counter under the **BizTalk:Message Agent** performance object category.

- The default throttling behavior has been modified by setting a registry value or values to control the throttling behavior of a host process. The registry keys for overriding the default throttling behavior are described in the **Registry settings that affect default throttling behavior** section of the topic How to Modify the Default Host Throttling Settings.

Depending upon the severity of the throttling condition, the following actions are taken:

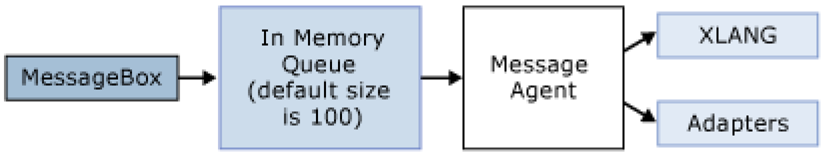
- A progressive delay in the processing logic of the host instance is implemented before delivering the messages to the outbound transport adapter or the orchestration engine for processing the messages. Both the duration of the processing logic delay and the rate at which the duration is incremented scale with the severity of the throttling condition.

- The number of messages that can be held by the in-memory queue is limited. The in-memory queue serves as a temporary placeholder for delivering messages from the MessageBox to the Message Agent which in turn delivers messages to XLANG and send adapters. By default, the in-memory queue is set to hold 100 messages per CPU. When the queue is full, no more messages are de-queued from the MessageBox until the in-memory queue is freed up.

- The size of the Message Agent thread pool is limited. By limiting the Message Agent thread pool size, the host throttling mechanism effectively reduces the amount of messages that are delivered to XLANG and adapters.

- The use of memory and other resources is reduced as applicable. BizTalk can send instructions to other service classes to limit memory use by dehydrating running schedules, shrinking memory cache size, and by limiting the usage of memory intensive threads.

Outbound message flow from MessageBox to Adapters and XLANG



How Does BizTalk Implement Host Throttling?

The BizTalk Server 2006 host throttling mechanism continually monitors for a throttling condition, calculates the severity of the throttling condition, and applies host throttling progressively depending upon the calculated severity. The throttling mechanism is self tuning and the default configuration options are suitable for the majority of BizTalk Server 2006 processing scenarios. BizTalk Server 2006 host throttling exposes several configurable options that can be used to tune throttling for specific scenarios. See How to Modify the Default Host Throttling Settings for information on changing these configuration options.

Components of the host throttling algorithm

BizTalk Server 2006 uses the following algorithm when applying host throttling:

1. Continuously monitor the following parameters to see if they exceed certain thresholds. If the values for the parameter exceed the threshold for the parameter then a throttling condition exists.

    - Amount of memory in use (both system wide and host process memory).

- Number of in-process messages being delivered or processed (threshold for outbound throttling).
- Number of threads in use.
- Database size, measured by the number of items in the queue tables for all hosts and the number of items in the spool and tracking tables.
- Number of concurrent database connections.
- Rate of message publishing (inbound) and delivery or processing(outbound).

2. Determine the severity of the throttling condition. Throttling conditions are ranked in severity as follows (from most severe to least severe).

- Host process memory in use exceeds threshold.
- Number of in-process messages exceeds threshold.
- Count of threads in use exceeds threshold.
- Database size exceeds threshold.
- All other throttling conditions.

3. Progressively apply throttling based upon the severity of the throttling condition(s). Throttling is applied more aggressively as the severity level increases. Progressive throttling is achieved as follows:

- One or more throttling conditions are detected and assigned a severity.
- An instruction to implement throttling is issued based upon the condition with the highest severity. Depending on the throttling condition, if the condition persists, the size of various thread pools may be reduced and memory may be freed by dehydrating running orchestrations.
- A delay is applied in the publishing or processing of the message, depending on whether the message is inbound or outbound. The delay period is proportional to the severity of the throttling condition, so higher severity throttling conditions will initiate a longer throttling period than lower severity throttling conditions. This delay period is adjusted up and down within certain ranges by the throttling mechanism as conditions change. The current delay period is exposed through the **Message Delivery Delay (ms)** and the **Message Publishing Delay (ms)** performance counters associated with the **BizTalk:Message Agent** performance object category. These performance object counters are documented in the topic Host Throttling Performance Counters.
- The throttling mechanism continues to check if a throttling condition is present. If the throttling conditions are mitigated then the throttled messages are unblocked and the thread pool and other resources are allowed to operate in an unrestricted mode. If the system continues to operate without any throttling conditions, the delay period is greatly

reduced. If the throttling condition persists, the delay amount is incremented proportional to the severity of the condition and subsequent messages are subject to the higher delay.

- Throttling is no longer implemented when the delay period has elapsed.

Type of throttling conditions

There are three primary types of throttling conditions, rate based, resource based, and user controlled.

1. **Rate based throttling** is divided into two categories; inbound (published) and outbound (delivered):

    - For inbound (published) messages, BizTalk throttles publishing of messages if The **Message Publishing Incoming Rate** for the host instance exceeds the **Message Publishing Outgoing Rate *** the specified **Rate overdrive factor (percent)** value. The **Rate overdrive factor (percent)** parameter is configurable on the **Message Publishing Throttling Settings** dialog box. Rate based throttling for inbound messages is accomplished primarily by inducing a delay before publishing the batch of messages into the MessageBox database. No other action is taken to accomplish rate based throttling for inbound messages.

    - For outbound (delivered) messages, BizTalk throttles delivery of messages if The **Message Delivery Incoming Rate** for the host instance exceeds the **Message Delivery Outgoing Rate** * the specified **Rate overdrive factor (percent)** value. The **Rate overdrive factor (percent)** parameter is configurable on the **Message Processing Throttling Settings** dialog box. Rate based throttling for outbound messages is accomplished primarily by inducing a delay before removing the messages from the in-memory queue and delivering the messages to the End Point Manager (EPM) or orchestration engine for processing. No other action is taken to accomplish rate based throttling for outbound messages.

    For more information about the rate overdrive factor see How to Modify the Default Host Throttling Settings.

2. **Resource based throttling** monitors system resources such as threads, memory, and database size and can be applied to any service class.

3. **User controlled throttling** occurs when the default throttling behavior has been modified by setting a registry value or values to control the throttling behavior of a host process. The registry keys for overriding the default throttling behavior are described in the **Registry settings that affect default throttling behavior** section of the topic How to Modify the Default Host Throttling Settings.

Throttling condition triggers, actions, and mitigation strategies

This section describes the triggers for various throttling conditions, the resultant actions by the throttling mechanism, and techniques that may be employed to mitigate the throttling condition.

Inbound Throttling

The BizTalk throttling mechanism applies inbound throttling to the orchestration engine (XLANG) and to inbound adapters.

Use the **Message publishing throttling state** and **Message publishing throttling state duration** counters associated with **BizTalk:MessageAgent** performance object category to measure the current throttling state and duration of throttling. For more information about the available host throttling performance counters see Host Throttling Performance Counters.

Inbound throttling can cause inbound messages to backlog at the source. If inbound throttling is applied to a receive adapter then the receive adapter may stop receiving messages until the throttling condition is mitigated.

Message Publishing throttling state	Trigger for throttling condition	Throttling actions taken	Mitigation Strategy	Performance Object	Performance Counte
2	**Message publishing incoming rate** for the host instance exceeds the **Message publishing outgoing rate *** the specified **Rate overdrive factor (percent)** value. The database cannot keep up with the publishing rate.	Block the publishing thread for a dynamically computed time period until the **Message Publishing Incoming Rate** is at par with the **Message Publishing Outgoing Rate** * the specified **Rate overdrive factor (percent)** value.	Use performance counters to determine the message publishing incoming and message publishing outgoing rate. Consider an appropriate over drive factor for your environment. Verify that the values supplied for the **Sampling window duration (milliseconds)** and **Minimum number of samples** parameters are appropriate for your scenario. For more information about these parameters see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent	Message publish incoming rate Message publish outgoing rate
4	Process memory exceeds the specified threshold. This can occur if the batch to be published has steep memory	Reduce the size of the thread pool used by EPM. Block the EPM threads to	Consider reducing the amount of load by reducing the EPM thread pool, and/or the size of adapter batches. If the process is not consuming excessive memory, consider increasing the **Process memory usage** threshold for the host. For	BizTalk:MessageAgent	High process memory Process memory usa (MB) Process memory usa threshold (MB)

	requirements or too many threads are processing messages	prevent processing of new messages batches. If there is a steep memory requirement to persist the messages in a batch to the database, the publishing thread is also subject to a progressive delay before the messages are persisted to the database. Whether the publishing batch will be blocked or not due to process memory pressure depends on several factors including the number of messages in the batch or if there are dehydration or message deletion	more information on changing the **Process memory usage threshold** see How to Modify the Default Host Throttling Settings.		

		commands in the batch.			
6	Host message queue size, the spool table size or the tracking table size exceed the specified threshold. Possible reasons for this condition include: • The SQL Agent jobs used by BizTalk to maintain the BizTalk databases not running or are running slowly. • Down-stream components are not processing messages from the in-memory queue in a timely manner. • Number of suspended messages is high. • Maximum sustainable load for the system has been	Reduce the size of the thread pool used by EPM. Block the EPM threads to prevent processing of new messages batches. The publishing thread is also subject to a progressive delay before the messages are persisted to the database.	Ensure that the SQL Agent jobs used by BizTalk to maintain the BizTalk databases are running and are not failing. For more information about these SQL Agent jobs see SQL Agent Jobs that Maintain the Messagebox Database. Terminate and resume suspended instances as needed. Increase the default value for the **Message count in database** threshold taking into consideration the space requirements of SQL Server that houses the BizTalk databases. If your database is sized appropriately to handle additional message backlog, consider increasing the **ThrottlingSpoolMultiplier** and and **ThrottlingTrackingDataMultiplier** registry values to allow additional backlog in the Spool and Tracking tables. For more information on changing the values see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent BizTalk:Message Box:General Counters BizTalk:Message Box:Host Counters	MessageAgent /Databa size Message Box:Gene Counters /Spool size Message Box:Gene Counters /Tracking d size Message Box:H Counters/Host queue length Message Box:H Counters/Host queue suspended msgs – leng

	reached.				
8	Database sessions being used by the host instance exceed the specified threshold.	Reduce the size of the thread pool used by EPM. Block the EPM threads to prevent processing of new messages batches. The publishing thread is also subject to a progressive delay before the messages are persisted to the database.	Consider increasing the **Database connections per CPU** threshold for the host. For more information on changing this value see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent	Database session
9	Process thread count exceeds the specified threshold.	Reduce the size of the thread pool used by EPM. Block the EPM threads to prevent processing of new messages batches. The publishing thread is also subject to a	Consider adjusting the different thread pool sizes to ensure that the system does not create a large number of threads. For more information about modifying the thread pool sizes see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent	Thread count Thread count threshold

		progressive delay before the messages are persisted to the database.			
5	System memory exceeds the specified threshold.	Reduce the size of the thread pool used by EPM. Block the EPM threads to prevent processing of new messages batches. If there is a steep memory requirement to persist the messages in a batch to the database, the publishing thread is also subject to a progressive delay before the messages are persisted to the database. Whether the publishing	Consider reducing load by reducing the default size of the EPM thread pool, and/or the size of adapter batches. If the process is not consuming excessive memory, consider increasing the **Physical memory usage** threshold for the host. For more information on changing the **Physical memory usage** threshold see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent	Physical memory usa (MB) Physical memory usa threshold (MB)

		batch will be blocked or not due to process memory pressure depends on several factors including the number of messages in the batch or if there are dehydration or message deletion commands in the batch.			

Outbound Throttling

The BizTalk throttling mechanism applies outbound throttling to the orchestration engine (XLANG) and to outbound adapters.

Use the **Message delivery throttling state** and **Message delivery throttling state duration** counters associated with **BizTalk:MessageAgent** performance object category to measure the current throttling state and duration of throttling. For more information about the available host throttling performance counters see Host Throttling Performance Counters.

Outbound throttling can cause delayed message delivery and messages may build up in the in-memory queue and cause de-queue threads to be blocked until the throttling condition is mitigated. When de-queue threads are blocked no additional messages are pulled from the MessageBox into the in-memory queue for outbound delivery.

Message Delivery throttling state	**Trigger for throttling condition**	**Throttling actions taken**	**Mitigation Strategy**	**Performance Object**	**Performance Counter**
1	**Message delivery incoming rate** for the host instance exceeds the **Message delivery outgoing rate ***	Block the delivery thread for a dynamically computed time period until the Message delivery	Use performance counters to determine the message delivery incoming and message delivery outgoing rate. Consider an	BizTalk:MessageAgent	Message delivery incoming rate Message publishing

	the specified **Rate overdrive factor (percent)** value This can be caused by high processing complexity, slow outbound adapters, or a momentary shortage of system resources.	incoming rate is at par with the **Message delivery outgoing rate *** the specified **Rate overdrive factor (percent)** value.	appropriate over drive factor for your environment. Verify that the values supplied for the **Sampling window duration (milliseconds)** and **Minimum number of samples** parameters are appropriate for your scenario. For more information about these parameters see How to Modify the Default Host Throttling Settings.		outgoing rate
4	Process memory exceeds the specified threshold. This can occur in memory intensive processing scenarios, when processing large messages, or when send adapters are attempting to process a high number of messages simultaneously.	Slow down message delivery to adapters or XLANG. Reduce process memory consumption by dehydrating service instances and shrinking cache when applicable. Reduce the size of the thread pools used by the EPM and/or the Message Agent. Periodically force a .NET garbage collection	If the system is not becoming idle due to process memory based throttling, no action may be required. If the **In-process message count** value is high and the CPU utilization is not excessive even when there is process memory based throttling, no additional action may be required. If the system appears to be over-throttling, consider increasing the value associated with the **Process memory usage** threshold for the host and verify that the host	BizTalk:MessageAgent	High process memory Process memory usage(MB) Process memory usage threshold (MB) In-process message count Active instance count

		(GC).	instance does not generate an "out of memory" error. As long as an "out of memory" error is not raised then the increased threshold should work fine. If an "out of memory" error is raised by increasing the **Process memory usage** threshold, then consider reducing the values for the **Internal message queue size** and **In-process messages per CPU** thresholds. This strategy is particularly relevant in large message processing scenarios.		
3	Number of in-process messages delivered to a service class exceeds the specified threshold. This can be caused by high processing complexity, slow outbound adapters, or a momentary shortage of system resources.	Slow down message delivery to adapters or XLANG. Reduce the size of the thread pool used by the Message Agent.	If over-throttling occurs, consider increasing the value associated with the **In-process messages per CPU** threshold.	BizTalk:MessageAgent	In-process message count In-process message count threshold

9	Process thread count exceeds the specified threshold.	Reduce the size of the thread pools used by the EPM and/or the Message Agent	Consider adjusting the different thread pool sizes to ensure that the system does not create a large number of threads. For more information about modifying the thread pool sizes see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent	Thread count Thread count threshold
5	System memory exceeds a threshold.	Slow down message delivery to adapters or XLANG. Reduce process memory consumption by dehydrating service instances and shrinking cache when applicable. Reduce the size of the thread pools used by the EPM and/or the Message Agent.	Consider reducing load by reducing the default size of the EPM thread pool, and/or the size of adapter batches. If the process is not consuming excessive memory, consider increasing the **Physical memory usage** threshold for the host. For more information on changing the **Physical memory usage** threshold see How to Modify the Default Host Throttling Settings.	BizTalk:MessageAgent	Physical memory usage

How to Modify the Default Host Throttling Settings

You can adjust the default throttling settings by modifying the configuration options that are exposed on the **Advanced** page of the **Host Properties** dialog box. The default configuration options are suitable for most BizTalk Server processing scenarios. This section describes the available configuration options and when you may want to change the default values.

Display the host throttling configuration options

To access the host throttling configuration properties for a host instance, follow these steps:

1. Launch the BizTalk Server Administration Console. Click **Start**, point to **Programs**, point to **Microsoft BizTalk Server 2006**, click **BizTalk Server Administration**.

2. In the BizTalk Server Administration console, click to expand **BizTalk Server 2006 Administration**, click to expand **BizTalk Group [<servername>:<management database>]**, click to expand **Platform Settings**, and click **Hosts**. The list of configured hosts is displayed in the right pane of the **BizTalk Server 2006 Administration Console**.

3. Right click the host for which you want to modify the default host throttling settings and click the **Properties** menu option to display the **Host Properties** dialog box.

4. Click the **Advanced** option in the left pane of the **Host Properties** dialog box to display the **Advanced** page of the **Host Properties** dialog box.

Host Properties - Advanced page

Setting	Description
Maximum number of messaging engine threads per CPU	This option specifies the maximum number of threads that can be used by the End Point Manager (EPM). The EPM starts with the number of threads equivalent to 10% of this value and adds threads up to the specified value as load increases. The number of threads allocated is reduced as load is reduced or as necessary for throttling. This setting largely affects only inbound (published) messages as the EPM is responsible for receiving messages from adapters and publishing the messages to the MessageBox database. If the EPM thread pool is set to a small number, then the likelihood of a throttling condition is reduced but doing this may also cause resources to be under utilized. Ideally, you want to set a thread pool that is large enough to fully utilize available resources during normal load and let the throttling mechanism dynamically reduce available threads if a throttling condition occurs. This value does not include the threads created by the adapters configured to run in the host. The default value is 20.

Throttling Thresholds dialog box

This dialog box is displayed by clicking the **Settings** button in the **Throttling Thresholds** section of the **Advanced** page of the **Host Properties** dialog box. This dialog box exposes the following thresholds above which a throttling condition may be triggered:

Threshold	Description
Internal message queue size	This is the size of the in-memory queue that the host instance maintains as a temporary placeholder for delivering messages. The host retrieves messages from the MessageBox database and places them in this interim queue before they are delivered to the respective components for processing. Setting a large value for this parameter may improve low-latency scenarios to some extent since more messages will be proactively retrieved from the MessageBox database for processing. Since the messages in this queue consume memory, setting this parameter to a smaller value may be desirable for scenarios involving large messages in order to avoid memory based throttling of the process. For example, if the memory consumption of a host instance for an outbound transport remains high despite a low value for **In-process Message Count**, the memory consumption could be attributed to the size of the messages residing in the in-memory queue. In this scenario, the queue size should be greatly reduced to avoid memory-based throttling. This setting affects only outbound (delivered) messages. The default value is 100.
Database connections per CPU	Maximum number of concurrent database sessions (per CPU) allowed before throttling begins. The idle database sessions in the common per-host session pool do not add to this count, and this check is made strictly on the number of sessions actually being used by the host instance. This option is disabled by default; typically this setting should only be enabled if the database server is a bottleneck in the BizTalk Server system. You can monitor the number of active Database connections by using the **Database Session** performance counter under the **BizTalk:Message Agent** performance object category. This parameter only affects outbound message throttling. Enter a value of 0 to disable throttling that is based on the number of database sessions. The default value is 0.
Threads per CPU	Total number of threads in the host process including threads used by adapters. If this threshold is exceeded, BizTalk will try to reduce the size of EPM thread pool and message agent thread pool. Thread based throttling should be enabled in scenarios where high load can lead to the creation of a large number of threads.

	This parameter affects both inbound and outbound throttling. Enter a value of 0 to disable thread based throttling. The default value is 0.
In-process messages per CPU	Maximum number of messages delivered to the End Point Manager (EPM) or XLANG that have not been processed. This does not include the messages retrieved from database but still waiting for delivery in the in-memory queue. You can monitor the number of In-Process Messages by using the **In-Process Message Count** performance counter under the **BizTalk:Message Agent** performance object category. This parameter provides a hint to the throttling mechanism for consideration of throttling conditions. The actual threshold is subject to self-tuning. You can verify the actual threshold by monitoring the **In-Process Message Count** performance counter. This parameter can be set to a smaller value for large message scenarios, where either the average message size is high, or the processing of messages may require a large amount of messages. Such a case would be evident if a scenario experiences memory-based throttling too often and if the memory threshold gets auto-adjusted to a substantially low value. Such a behavior would indicate that the outbound transport should process fewer messages concurrently to avoid excessive memory usage. Also, for scenarios where the adapter is more efficient when processing a few messages at a time (for example when sending to a server that limits concurrent connections), this parameter may be tuned to a lower value than the default. This parameter only affects outbound message throttling. Enter a value of 0 to disable throttling based on the message count in database parameter. The default value is 1000.
Message count in database	Total number of messages published by this host instance to the work, state, and suspended queues of the subscribing hosts. This is calculated as a weighted average as follows: • Determine the historical percentage of the total messages published by the host instance to each queue. • Multiply the number of items published to each queue by the historical percentage to derive the weighted number of messages published to each queue. • Add the weighted number of messages published to all queues. For example, if a host instance has historically published 60 percent of its messages to queue X and 40 percent of its messages to queue Y then the weighted average is calculated as follows:

	((#msgs published to X * .60) + (#msgs published to Y * .40)) Publishing batches may be exempt from throttling if the batch also contains message deletion commands and the number of messages to be deleted exceeds the number of messages to be published. Example of such batches may include solicit-response messaging scenarios or for orchestration persistence points following a message receive. The **Message count in database** setting also indirectly defines the threshold for a throttling condition based upon the number of messages in the spool table or tracking table. If the number of messages in the spool table or tracking table exceeds 10 times this value then a throttling condition will be triggered. By default the **Message Count in Database** value is set to 50,000 which will cause a throttling condition if the spool table or the tracking table exceeds 500,000 messages. The name of the tracking table in the MessageBox database is **TrackingData**. The name of the spool table in the MessageBox database is **Spool**. This parameter only affects inbound message throttling. Enter a value of 0 to disable throttling based on the message count in database parameter. The default value is 50000.
Physical memory usage	Specified as one of: • A percentage of memory consumption compared to the total amount of available physical memory if a value from 1 to 100 is entered. • Total amount of available physical memory in MB if a value greater than 100 is entered. Since this measures total system memory, a throttling condition may be triggered if non BizTalk processes are consuming an extensive amount of system memory. Enter a value of 0 to disable throttling based on physical memory usage. The default value is 0.
Process memory usage	Specified as one of: • A percentage of memory used compared to the sum of the working set size and total available virtual memory for the process if a value from 1 to 100 is entered. When a percentage value is specified the process memory threshold is recalculated at regular intervals. • Total amount of available memory in MB computed as the sum of the working

	set size and total available virtual memory for the process if a value greater than 100 is entered. When process memory exceeds this threshold, a throttling condition is triggered. Inbound (published) batches are usually exempt from throttling because memory is released when messages are published if the batch contains a dehydration or message completion command and these actions reduce memory consumption. Since the managed runtime performs a lazy release of memory through offline .NET garbage collection (GC), a .NET GC may be triggered by the throttling mechanism if the memory used approaches or is above the specified threshold value. Since a .NET GC is an expensive operation, the frequency of the GC trigger is moderated. If the memory released by a GC is minimal, the GC trigger may occur only every 120 seconds. Conversely if a GC is calculated to release more than 25 percent of used memory, then another GC may be triggered within 15 seconds in high memory usage scenarios. When the in-process message count drops to zero and memory pressure is still high, a GC is forced. The minimum intervals between subsequent GCs are hard-coded and can not be changed externally. Enter a value of 0 to disable throttling based on process memory usage. This parameter provides a hint to the throttling mechanism for consideration of throttling conditions. The actual threshold is subject to self-tuning. You can verify the actual threshold by monitoring the **Process memory usage (MB)** performance counter. The default value is 25.

Message Publishing Throttling Settings dialog box

This dialog box is displayed by clicking the **Settings** button in the **Message Publishing Throttling** section of the **Advanced** page of the **Host Properties** dialog box.

These settings affect rate based throttling for inbound (published) messages.

Threshold	**Description**
Minimum number of samples	Minimum number of messages BizTalk will sample for the **Sampling window duration** before considering rate-based throttling. If the actual number of samples in a sampling window fall below this value then the samples are discarded and throttling is not applied. This value should be consistent with a rate at which messages can be published under a medium load. For example, if your system is expected to handle 1000 documents per second under a medium load, then this parameter should be set to 1000 * Sample window duration in seconds (or more precisely, 1 * **Sample window duration (milliseconds)**). If the value is set too low, then the system may experience a throttling condition under low load. If the value is set too high, then there may not be enough samples for this technique to be effective.

	Enter a value of zero to disable rate based inbound throttling. The default value is 100.
Sampling window duration (milliseconds)	The time-window measured in milliseconds, which is used to calculate the publishing rate based on the samples collected. The duration should be increased if the latency required for publishing a single message is high. Enter a value of zero to disable rate based inbound throttling. The default value is 15000.
Rate overdrive factor (percent)	This controls how much higher you allow the request rate to be than the completion rate before a throttling condition occurs. For example, if messages are being published at a rate of 100 per second and this parameter is set to 125, then the system will allow the publishing of up to 125 messages per second before applying throttling. Specifying too small of value for this parameter will cause the system to throttle more aggressively and could lead to over throttling. Specifying too large of a value for this parameter will cause under throttling and prevent the throttling mechanism from recognizing a legitimate throttling condition. The default value is 125.
Maximum throttling delay	This is the maximum delay BizTalk Server will impose on a message instance due to throttling. The actual delay depends on the severity of the throttling condition. Enter a value of zero to disable inbound throttling. The default value is 300000.

Message Processing Throttling Settings dialog box

This dialog box is displayed by clicking the **Settings** button in the **Message Processing Throttling** section of the **Advanced** page of the **Host Properties** dialog box.

These settings affect rate based throttling for outbound (delivered) messages.

Setting	**Description**
Minimum number of samples	Minimum number of messages BizTalk will sample for the **Sampling window duration** before considering rate-based throttling. If the actual number of samples in a sampling window fall below this value then the samples are discarded and throttling is not applied. This value should be consistent with a rate at which messages can be delivered under a medium load. For example, if your system is expected to handle 1000 documents per second under a medium load, then this parameter should be set to 1000 * Sample window duration in seconds (or more precisely, 1 * **Sample window duration (milliseconds)**). If

	the value is set too low, then the system may experience a throttling condition under low load. If the value is set too high, then there may not be enough samples for this technique to be effective. Enter a value of zero to disable rate based outbound throttling. The default value is 100.
Sampling window duration (milliseconds)	The time-window measured in milliseconds, which is used to calculate the processing rate based on the samples collected. The duration should be increased if the latency required for processing a single message is high. Enter a value of zero to disable rate based outbound throttling. The default value is 15000.
Rate overdrive factor (percent)	This controls how much higher you allow the delivery rate to the Orchestration or Messaging engine to be than the completion rate before a throttling condition occurs. For example, if messages are being delivered at a rate of 100 per second and this parameter is set to 125, then the system will allow the delivery of up to 125 messages per second before applying throttling. Specifying too small of value for this parameter will cause the system to throttle more aggressively and could lead to over throttling. Specifying too large of a value for this parameter will cause under throttling and prevent the throttling mechanism from recognizing a legitimate throttling condition. The default value is 125.
Maximum throttling delay	This is the maximum delay BizTalk Server will impose on a message instance due to throttling. The actual delay depends on the severity of the throttling condition. Enter a value of zero to disable outbound throttling. The default value is 300000.

Registry settings that affect default throttling behavior

You can modify the default throttling behavior by creating values under the following key in the BizTalk Server registry:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\BTSSvc$<HostName>

BizTalk Server checks the values stored in this location of the registry every cache refresh interval, which is set to 60 seconds by default. For information on changing the cache refresh interval see How to Modify Group Properties.

Create the following registry values to modify the default throttling behavior:

Name	Type	Description	Comments
ThrottlingPublishOverride	DWORD	Valid Values are: 0: Do not override 1: Initiate throttling condition 2: Do not throttle	The default value is 0 (Do no override)
ThrottlingDeliveryOverride	DWORD	Valid Values are: 0: Do not override 1: Initiate throttling condition 2: Do not throttle	The default value is 0 (Do no override)
ThrottlingPublishOverrideSeverity	DWORD	Valid values are from 1 to 1000	A higher value increases the severity of an inbound throttling condition initiated when **ThrottlingPublishOverride** is se to 1. The default value is 100
ThrottlingDeliveryOverrideSeverity	DWORD	Valid values are from 1 to 1000	A higher value increases the severity of an outbound throttling condition initiated when **ThrottlingDeliveryOverride** is se to 1. The default value is 100
AgentThreadPoolSize	DWORD	Valid values are from 1 to 50	The maximum number of threads (per CPU) that the message agen can allocate to deliver de-queued messages to the XLANG/messaging engine. The message agent has a dynamic thread-pool that shrinks o grows depending on the load. The thread pool always maintains a least 1/10th of the maximum value For example, if the maximum threa

			pool size is 20, there will be at leas 2 threads allocated even when there is no load. The default value is 20.
AgentThreadsPerCpu	DWORD	Valid values are from 1 to 20.	The maximum number of threads per CPU that can compete with each other for CPU time. Since BizTalk is I/O intensive, many threads may be waiting. Nevertheless, this value should be kept low to avoid excessive context switching. The default value is 2.
AgentThreadPoolQueueLen	DWORD	Valid values are from 20 to 1000.	The place holder for work items waiting to be scheduled for a thread. Items waiting in the thread pool queue consume memory and so this value should be kept low fo large message scenarios. The default value is 50.
ThrottlingSpoolMultiplier	DWORD	Valid values are from 0 to 1000.	Specified the factor by which the **Message Count in Database** threshold will be multiplied and then compared against the current record count in the spool table to determine whether the system should throttle on spool table size. I this is set to 0, spool table size is not used as a consideration fo determining a throttling condition. The default value is 10.
ThrottlingTrackingDataMultiplier	DWORD	Valid values are from 0 to 1000.	Specified the factor by which the **Message Count in Database** threshold will be multiplied and then compared against the current record count in the tracking table to determine whether the system should throttle on tracking table size. If this is set to 0, tracking table size is not used as a consideration for determining a throttling condition.

			The default value is 10.
ThrottlingSeverityProcessMemory	DWORD	Valid values are from 1 to 1000.	Controls the severity of a process memory triggered throttling condition. This is specified in percentage value and this parameter sets the severity of a throttling condition caused when the **Process Memory Usage** threshold is exceeded. The default value is 500.
ThrottlingSeverityDatabaseSize	DWORD	Valid values are from 1 to 1000.	Controls the severity of a database sized triggered throttling condition. This is specified in percentage value and this parameter sets the severity of a throttling condition caused when the **Message Count in Database** threshold is exceeded. The default value is 1.
ThrottlingSeverityInflightMessage	DWORD	Valid values are from 1 to 1000.	Controls the reaction time of throttling when the In-process size exceeds the threshold. This is specified in percentage value and this parameter sets the severity of a throttling condition caused when the **In-process messages per CPU** threshold is exceeded. The default value is 75.
ThrottlingBatchMemoryThresholdPercent	DWORD	Valid values are from 0 to 100.	Controls the memory threshold beyond which to throttle the publishing of a batch of messages. The batch memory threshold is computed by multiplying this percentage factor by the **Process Memory Usage** threshold. If the memory estimated to execute a publishing batch exceeds the batch memory threshold, the batch will be subject to process memory based throttling. Otherwise the batch will be exempt from process memory based throttling even when the total process memory exceeds the **Process Memory Usage** threshold

			A value of zero indicates that al publishing batches may be subjec to process memory based throttling even if the memory estimated to execute the batch is minimal. The default value is 1.
ThrottlingLimitToTriggerGC	DWORD	Valid values are from 50 to 100.	Controls when a .NET garbage collection (GC) will be triggered as process memory consumption increases and approaches the threshold. When the memory consumption exceeds this percentage value of the memory threshold, a GC is triggered. The default value is 80.
MsgAgentPerfCounterServiceClassID (String Value)	DWORD	Valid values are: {226FC6B9-0416-47A4-A8E8-4721F1DB1A1B} (XLANG) {59F295B0-3123-416E-966B-A2C6D65FF8E6} (Messaging) {3D7A3F58-4BFB-4593-B99E-C2A5DC35A3B2} (MSMQT) {683AEDF1-027D-4006-AE9A-443D1A5746FC} (Messaging Isolated)	Controls the Service Class for which message agent performance counters are displayed in Performance Monitor. If this registry value is not present, the message agent performance counters for the most active service class of a hos are displayed in performance monitor (since a host may actually include several service classes) Setting this value will change the default behavior and display the values generated from the counters used by the specified service class.

It is possible to dynamically control the BizTalk host throttling mechanism by following these steps:

1. Create a custom resource monitoring component to measure system load.

2. Programmatically change these registry values based upon the data returned from the monitoring component.

Host Throttling Performance Counters

This section describes the performance monitor counters that measure system parameters that impact host throttling. The following performance counters are accessible for each host instance under the **BizTalk:Message Agent** performance object category:

Counter	Description
Active instance count	Number of service instances active in memory. For the orchestration engine, a service instance refers to each running instance of an orchestration schedule. For the End Point Manager, a service instance may either correspond to a single stateless message, or to a collection of stateful messages. **Note** Stateful instances are those that maintain certain state information about the messages associated with the instance. Messages belonging to a stateful instance are co-related in some form or the other. For example an MSMQT message stream that maintains information about the sender and the message sequence, or an ordered send port that maintains information about the ordering are considered stateful instances. Most messaging scenarios involve stateless instances where messages are processed completely independent of each other. Each such stateless instance corresponds to a single message within the EPM.
Database session	Number of concurrent MessageBox database connections being used.
Database session threshold	The current threshold for concurrent database sessions. This is initially set to the value specified for **Database connections per CPU** on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. This value is auto-tuned based on the database session usage of the process. If the number of concurrent database sessions exceeds this threshold at any time, host throttling is implemented.
Database size	Number of messages in the database queues that this process has published. This value is measured by the number of items in the queue tables for all hosts and the number of items in the spool and tracking tables. If a process is publishing to multiple queues, this counter reflects the weighted average of all the queues.
High database session	• 0: Normal • 1: Database session count exceeds threshold
High database size	• 0: Normal • 1: Database size has grown beyond threshold
High in-process message count	• 0: Normal • 1: In-process message count exceeds limit

High message delivery rate	• 0: Normal • 1: Message delivery rate exceeds the message processing Rate
High message publishing rate	• 0: Normal • 1: Publishing request rate exceeds completion rate
High process memory	• 0: Normal • 1: Process memory exceeds threshold
High system memory	• 0: Normal • 1: System memory exceeds threshold
High thread count	• 0: Normal • 1: Thread count exceeds threshold
In-process message count	Number of in-memory messages delivered to the XLANG engine or the outbound messaging engine that are not yet processed.
In-process message count threshold	The current threshold for in-process message count.
Message delivery delay (ms)	The current delay in ms imposed on each message delivery batch (applicable if the message delivery is being throttled).
Message delivery incoming rate	Number of messages per second that are being delivered to the Orchestration engine or the Messaging engine in the given sample interval.
Message delivery outgoing rate	Number of messages per second that are being processed by the Orchestration engine or the Messaging engine in the given sample interval.
Message delivery throttling state	A flag indicating whether the system is throttling message delivery (affecting XLANG message processing and outbound transports). • 0: Not throttling • 1: Throttling due to imbalanced message delivery rate (input rate exceeds output rate) • 3: Throttling due to high in-process message count • 4: Throttling due to process memory pressure

	• 5: Throttling due to system memory pressure • 9: Throttling due to high thread count • 10: Throttling due to user override on delivery
Message delivery throttling state duration	Seconds since the system entered this state. If the host is throttling, how long it has been throttling; if it is not throttling, how long since throttling was applied.
Message delivery throttling user override	This counter reflects the user override that is monitored by the engine and interpreted as follows: • 0: No override • 1: Always throttle message delivery • 2: Do not throttle message delivery
Message publishing delay (ms)	The current delay in ms imposed on each message publishing batch (applicable if the message publishing is being throttled and if the batch is not exempted from throttling).
Message publishing incoming rate	Number of messages per second that are being sent to the database for publishing in the given sample interval.
Message publishing outgoing rate	Number of messages per second that are actually published in the database in the given sample interval.
Message publishing throttling state	A flag indicating whether the system is throttling message publishing (affecting XLANG message processing and inbound transports). • 0: Not throttling • 2: Throttling due to imbalanced message publishing rate (input rate exceeds output rate) • 4: Throttling due to process memory pressure • 5: Throttling due to system memory pressure • 6: Throttling due to database growth • 8: Throttling due to high session count

	• 9: Throttling due to high thread count • 11: Throttling due to user override on publishing
Message publishing throttling state duration	Seconds since the system entered this state. If the host is throttling, how long it has been throttling; if it is not throttling, how long since throttling was applied.
Message publishing throttling user override	This counter reflects the user override that is monitored by the engine and interpreted as follows: • 0: No override • 1: Always throttle message publishing • 2: Do not throttle message publishing
Physical memory usage (MB)	The amount of physical memory in MB being used on the machine by all processes.
Process memory usage (MB)	The process memory consumption in MB. This is the maximum of the process's working set size and the total space allocated for the page file for the process.
Process memory usage threshold (MB)	The current threshold for process memory consumption in MB. This is initially set to the value specified for **Process Memory Usage** on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. If a percentage value is specified, it is computed based on the available memory to commit
Service class ID	The decimal value of the initial part of the service class GUID that this performance counter instance corresponds to. A process may host more than one service class and the message agent performance counters show the data for the most active service class.
Thread count	Number of threads being used within the process.
Thread count threshold	The current threshold for the number of threads in the process. This is initially set to the value specified for **Threads per CPU** on the **Throttling Thresholds** dialog available from the **Advanced** page of the **Host Properties** dialog box. This value is auto-tuned depending on the thread requirements of the current process. If the number of threads in the process exceeds this threshold value at any point in time, host throttling is implemented.
Total batches committed	Number of database batches that the service class has committed.

Total messages delivered	Number of outbound messages delivered to the Orchestration engine or the End Point Manager (EPM).
Total messages published	Number of messages published.

To access performance counters

1. On the Desktop, click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Performance**.

2. In the **Performance** dialog box, click **Add**.

3. In the **Add Counters** dialog box, from the **Performance** object drop-down list, select from the available **BizTalk:Message Agent** performance counters, and then click **Add**.

4. In the **Add Counters** dialog box, do one of the following:

Use this	**To do this**
All counters	Select this option to select all counters from the provided list.
Select counters from list	Select this option to select specific counters from the provided list.

5. After selecting the counters, click **Add** and then click **Close**.

The selected performance counters appear on the Performance screen.