Pro BizTalk 2006

George Dunphy and
Ahmed Metwally

Apress®

Pro BizTalk 2006

Copyright © 2006 by George Dunphy, Ahmed Metwally

ISBN-13 (pbk): 978-1-59059-699-9

ISBN-10 (pbk): 1-59059-699-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jonathan Gennick
Technical Reviewer: Stephen Kaufman, David Stucki
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade
Project Manager: Beth Christmas
Copy Edit Manager: Nicole Flores
Copy Editor: Ami Knox
Assistant Production Director: Kari Brooks-Copony
Senior Production Editor: Laura Cheu
Compositor: Linda Weidemann, Wolf Creek Press
Proofreader: April Eddy
Indexer: John Collin
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code/ Download section. You will need to answer questions pertaining to this book in order to successfully download the code.

CHAPTER 1

BizTalk in the Enterprise

The BizTalk Server 2006 product is a group of application services that facilitate the rapid creation of integration solutions. BizTalk Server is designed specifically to integrate disparate systems in a loosely coupled way. BizTalk Server is a toolkit, and within this toolkit you will find tools to help you build your application. The trick, like the wise Scottish man said, is "using the right tool for the right job."

The art of creating a BizTalk solution is exactly that, using the right tool from the BizTalk toolkit to accomplish the task at hand. If you were to look at two BizTalk projects that address identical problems, designed by two architects, you would, most likely, see two completely different-looking solutions. Both solutions would probably work; however, generally one solution would be more correct because it would properly use BizTalk's tools as they were designed. This book will attempt to address how to properly use each of the tools within BizTalk in the manner in which they were intended. The other thing the book will do is show how each of the tools can be used to solve integration problems in an efficient way. In reality though, most of the features within BizTalk are flexible enough that you can generally solve most problems using only one piece of the BizTalk puzzle.

Since BizTalk allows you to address a problem in dozens of different ways, there is no one answer as to how to implement a BizTalk solution. To help address this issue of how best to implement a given solution, Microsoft will be releasing several enterprise application guidelines for BizTalk 2006 that should alleviate some of this confusion, but this problem will never go away. This book will build upon those patterns as well as provide some advanced concepts, examples, and patterns to allow software architects to properly build complex solutions using BizTalk Server.

What Is in the Toolkit

Inside BizTalk Server, you will find tools, each of which address a specific type of problem. Before learning about those tools, it is important to know what the architecture is for a "typical" BizTalk solution. Figure 1-1 illustrates the typical architecture for a BizTalk-based solution.

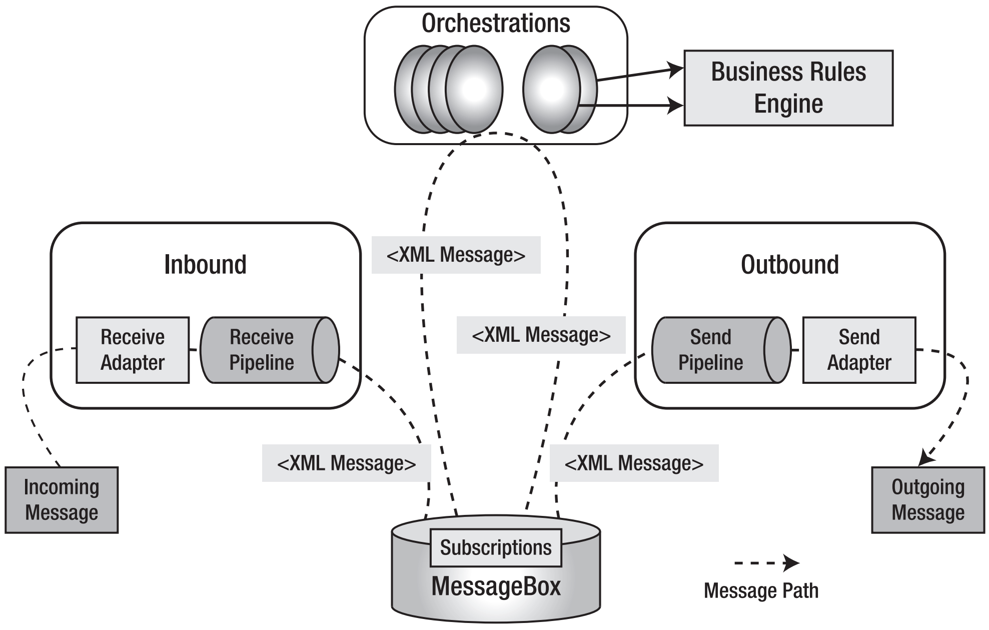


Figure 1-1. *Typical BizTalk scenario*

At its most basic, BizTalk is designed to receive inbound messages, pass them through some form of logical processing, and then deliver the result of that processing to an outbound location or subsystem. The art of architecting a BizTalk project begins with how to solve these three simple yet all-important tasks. Each of the tools in the following list is dissected throughout various sections of the book, but a simple explanation of each is provided here after the list.

- Ports and adapters
- Business Activity Monitoring and Business Activity Services
- Pipelines
- Pipeline components
- Orchestrations
- Transformations
- Messaging Engine
- Business Rule Engine
- EDI Services

Ports and **adapters** provide the logical abstraction for sending and receiving messages to and from BizTalk. They allow you to code your application in a generic fashion and not worry about the implementation details of how these messages will be consumed and delivered. A **port** is a logical construct that can receive and send messages to/from the BizTalk

Messagebox. The port must be married to a specific receive location to accept the message. The receive location then is tied to a specific **adapter**, which provides the details of how the message will be transported. These two constructs along with the BizTalk Messagebox provide the basis for the messaging infrastructure within the product.

BizTalk provides several adapters out of the box including those for FTP, File Access, SOAP, HTTP, SMTP, POP3, MSMQ, and MQSeries, all examples of transport-specific adapters. Transport adapters are usually tied to a specific wire protocol, providing the means by which to send the message. BizTalk also includes application adapters, which are used to integrate with specific third-party products such as Oracle's Database Server and ERP (Enterprise Resource Planning) packages like SAP and PeopleSoft. The complete list of transport and application adapters included within BizTalk can be found at www.microsoft.com/biztalk. Additional adapters can be purchased from third-party vendors or custom developed within Visual Studio 2005 using the BizTalk Adapter Framework.

Business Activity Monitoring (**BAM**) and **Business Activity Services** (**BAS**) provide the infrastructure to perform application instrumentation and metric reporting. BAM provides the ability to instrument your application to provide business-level data that is much more relevant than the default system-level information that is available in the base product. Examples of this would be

- How many purchase orders were processed?
- How many transactions failed last week? Last month?
- What was the total volume of messages received from a supplier?

BAS, on the other hand, provides a simple yet powerful way to display metric data from BAM and other system-level subservices using Microsoft SharePoint Portal Services. Most organizations will integrate the BAS portal into an existing SharePoint infrastructure rather than build an entire SharePoint site around BAS itself.

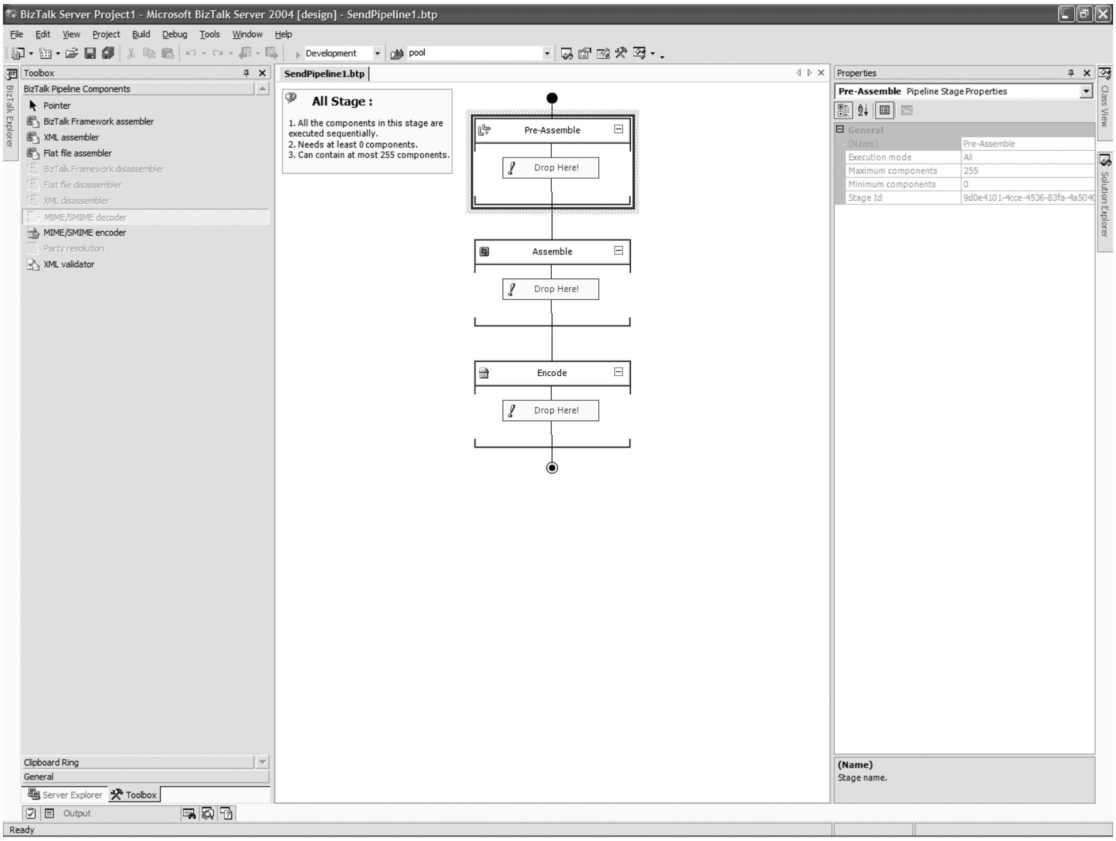
Pipelines provide a way to examine, verify, and potentially modify messages as they are received from and sent to BizTalk. They allow you to deconstruct messages that contain multiple documents and/or header information into a format that is more logical to the application or business user. Pipelines are applied to ports and are either a **send pipeline** or a **receive pipeline** depending on the directional flow of the message. Pipelines provide the necessary infrastructure for pipeline components (which we'll discuss in more detail a little later) or components that are executed within the various stages of a pipeline. Pipelines and pipeline components are unique constructs that only exist within the context of a BizTalk solution, and as such are generally unique to solutions that use BizTalk. Pipelines and pipeline components are created within the Visual Studio development environment using C# or VB .NET.

Pipelines function according to the concept of **assembly and disassembly**. Pipelines, which can assemble or disassemble, contain pipeline components, which are responsible for preparing the message to be sent. These components convert the internal BizTalk XML message to the appropriate XML or non-XML outbound format of the message, based on the type of assembler and properties set in the schema. For example, the assembling component of the pipeline may dictate that the message is to be sent in a flat-file text format and not XML format. BizTalk ships with default assemblers that allow you to assemble XML or flat-file messages by default with the option of writing custom assembling components. In addition, pipelines that contain assembling components assemble and wrap the message in an envelope or add a header or trailer (or both) to the message. During assembly, some properties are moved from the message context to the body of the document or to the envelope. The message context is the internal BizTalk representation of the metadata about the message such as the inbound transport type, the message type, and any special properties to be included that describe the message.

The opposite of assembling components are disassembling components, which execute on the receiving side of a message flow. Disassembling components prepare a message to be broken down into separate message documents according to the envelope and document schemas defined within BizTalk. Like assembling components, disassembling components may convert non-XML messages into their XML representation, to be processed by BizTalk. The message is then disassembled into individual messages that can be consumed by separate orchestrations or send ports. The message is disassembled by stripping the envelope information, breaking the message up into individual documents, and then copying envelope or message body property information to the individual message **contexts**. The message context is metadata about the message that is tied to the message data when it is processed by BizTalk. Exactly which properties are copied to the context is determined by the schema of the document and what properties it has defined for **promotion**, a term used to describe the way BizTalk copies properties from the message body to the message context.

Pipelines have various **stages** in which components can be executed. Stages are much like events in that they have a set order in which they execute and can be used to ensure that pipeline component logic executes in the correct order. Each stage can potentially execute more than one pipeline component depending on where it is located. This is explored in detail in Chapter 4. The pipeline is coded within the Visual Studio 2005 environment as shown in Figure 1-2. The stages for the two types of pipeline are as follows:

- Send pipeline stages:
  - Pre-Assemble
  - Assemble
  - Encode
- Receive pipeline stages:
  - Decode
  - Disassemble
  - Validate
  - Resolve Party



All Stage :

1. All the components in this stage are executed sequentially.
2. Needs at least 0 components.
3. Can contain at most 255 components.

Pre-Assemble Pipeline Stage Properties

General	
(Name)	Pre-Assemble
Execution mode	All
Maximum components	255
Minimum components	0
Stage Id	9d0e4101-4cce-4536-83fa-4a504

(Name)
Stage name.

Figure 1-2. *BizTalk Send Pipeline Designer*

Pipeline components are classes that are executed within the various stages of a BizTalk server pipeline. Custom pipeline components implement a specific set of application interfaces required by the BizTalk framework. Several default pipeline components are shipped with the product that provide standard functionality that most pipelines require, examples of which are as follows:

- **Disassemblers and assemblers**: Components that allow a pipeline to examine an inbound document and separate it into logical parts, or likewise take several separate documents and assemble them into one document. In most projects, the inbound or outbound document is a container or an **envelope** document that may contain several other distinct but related document types, each with their own schema.

- **Validators**: These allow for the pipeline to validate the document according to a default specification. The default validators allow the pipeline to verify that the document is valid XML. Custom validators can be written to perform solution-specific validation.

- **Encoders and decoders**: As the name suggests, these allow the pipeline to either decode an inbound message or encode an outbound message. The default BizTalk components allow you to encode or decode S/MIME messages. Most custom encoder/decoder components perform either custom security routines or specialized cryptographic operations. (See Chapter 5 for a pipeline component implementation that performs cryptographic operations.)

Orchestrations are used to graphically model workflow and provide the primary mechanism to implement business process automation within the product. Orchestrations are by far the most powerful tool within the BizTalk Server toolbox as they allow for the rapid development and deployment of complex processes that in many circumstances can be implemented with little to no coding. Orchestrations are created within Visual Studio and are compiled into .NET assemblies that are installed into the BizTalk Management Database. Assemblies deployed to this database must also be installed into the Global Assembly Cache and have a strong name. The Orchestration Designer is a primarily visual tool. It allows you to graphically see the workflow you are creating, as Figure 1-3 demonstrates.

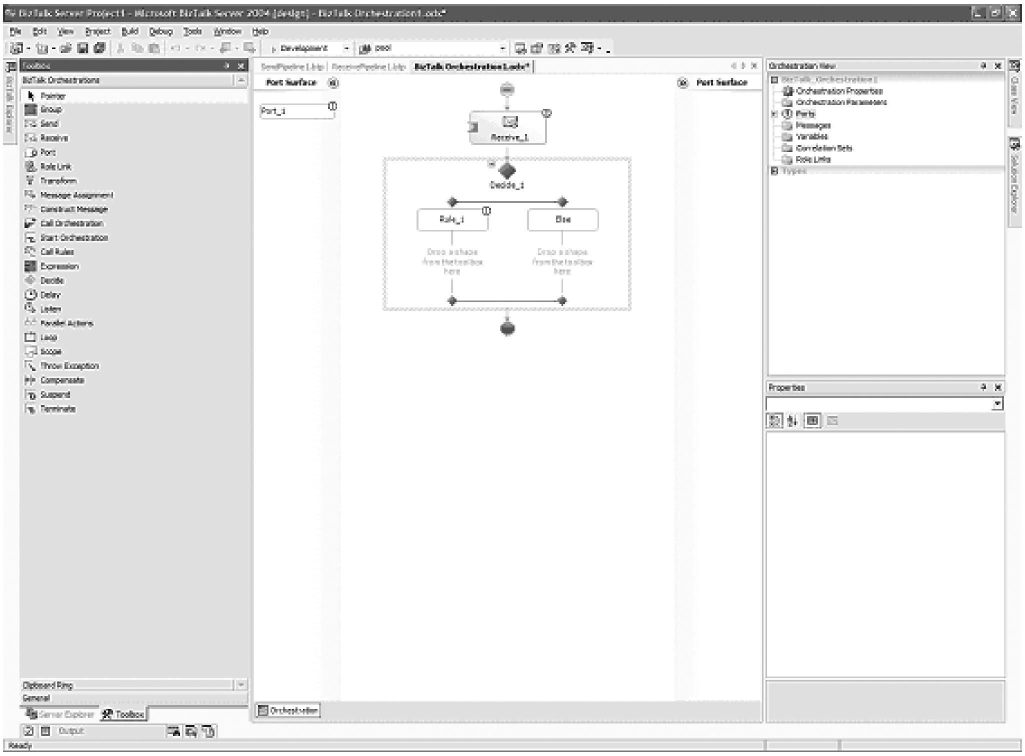


Figure 1-3. *BizTalk Orchestration Designer*

Transformations allow the application to map one message specification to another and transform data as it is processed. BizTalk messages are XML documents within the system, and as such, transformations are created from XSL (eXtensible Stylesheet Language) stylesheets. Transformations in BizTalk 2004 used the Microsoft XML Document Object Model (XML DOM) as their primary transformation engine. Starting in BizTalk 2006, the transformation engine is a custom solution developed by the BizTalk Server team. This new transformation engine is designed to increase the performance of transforming complex and especially large messages while preserving fault tolerance. Transforming large messages (greater than 10MB) proved to be problematic in BizTalk 2004 due to issues with out-of-memory conditions that occurred within the transformation engine, which was the main reason for this features' redesign.

Transformations can be applied in two places within a BizTalk solution—on a port when a message is sent or received or from within an orchestration. Each design pattern has its own pros and cons, which are discussed later in Chapter 6.

The **Messaging Engine** is the heart of BizTalk. The engine is responsible for ensuring that messages are received and routed to the proper location, that transaction isolation and consistency occurs, and that errors are reported. In reality, the Messaging Engine is the "Server" component of BizTalk Server. The Messaging Engine uses several BizTalk databases to store message information and metadata as well as system-level parameters. The Messaging Engine uses Microsoft SQL Server as its data storage facility. The central database for messages within BizTalk is called the **Messagebox**. SQL Server is not required to be physically installed on the same machine as BizTalk Server, and we recommend it be installed in its own environment for high-transaction and fault-tolerant solutions. A SQL Server license is required for BizTalk and is not included in the price of the product.

Typically, organizations will invest in separate and robust SQL Server development, testing, and production environments. Each BizTalk solution within the organization would use the appropriate SQL Server environment depending on the requirements of the application and its phase of implementation.

The **Business Rule Engine** (**BRE**) is a facility where business rules can be modeled using a simple GUI and called from within the BizTalk Server environment. The BRE is designed to allow for versioning and modification of implemented business rules without having to change the processes within BizTalk that use them. Most solutions use the BRE to implement things that require frequent updates such as a discount policy percentage or calculations that are updated frequently as a result of legal or government regulations.

Electronic Data Interchange (**EDI) Services** is a documented and accepted set of standards for exchanging electronic documents between organizations. EDI is traditionally used within legacy applications that have been implemented prior to the XML standard from the W3C. EDI documents are text based in nature and have a variety of document specifications depending on the data to be exchanged and the industry that uses them. BizTalk Server provides mechanisms for parsing these EDI documents into XML as well as connectivity to legacy data sources such as Value Added Networks and proprietary dial-up services.

Common Enterprise Usage

Most enterprises are looking for BizTalk to solve a particular problem. In many scenarios, this problem is related to having unrelated and disconnected systems exchange data in a standard, consistent, and reliable way. Often enterprises need to automate and streamline manual or inefficient processes in order to achieve a competitive advantage. Whether you are implementing an integration solution or a workflow automation solution, the tools that BizTalk provides allow an architect to design reliable and robust solutions faster than is often achievable by custom coding a solution from scratch using standard development languages and tools. With this in mind, it is important to understand what BizTalk achieves well and what is does not in order to fully realize any efficiency that a BizTalk solution can bring.

In other scenarios, organizations are using BizTalk orchestrations to connect and route messages to services within SOA environments. Often, the web services within the organization may be managed within different groups or even external parties. In these cases, orchestrations provide an excellent way to expose higher-level business functionality by calling, transforming, and routing messages between decoupled web services.

New BizTalk Solution Checklist

One of the key decisions that most new BizTalk application architects have to make is whether to use BizTalk or build their solution completely from scratch. The choice can only come from experience and an understanding of what BizTalk does well. The following sections explore some questions that need answering before starting any major BizTalk initiative.

What Are the Teams' Current Core Skill Sets? What Skill Sets Will the Team Need to Attain?

BizTalk projects require many skills to varying degrees depending on the size and complexity of the solution being implemented. These skills along with knowledge of the Microsoft Windows Server operating system represent a base of knowledge to start planning a BizTalk development and support team.

.NET Development

BizTalk is designed to be extended in several ways: the most common being through the custom development of .NET code, the second being through the implementation of custom business logic within orchestrations. If your project will require a significant amount of customization and business logic code, it is imperative that your team's core skill set be in .NET development.

Most teams underestimate the amount of custom code that they will need to write. It is extremely rare that a solution will have its entire business logic exclusively written within orchestrations or BizTalk transformations. If custom pipeline or adapter coding is required, a solid understanding of application programming interfaces (APIs) and object inheritance is a necessity.

The native BizTalk APIs are exposed exclusively to Windows applications usually running managed code; however, it is possible to access the BizTalk assemblies using COM but not as efficiently. All samples that are shipped with the product are written in Managed Code.

XML

People often overlook the fact that BizTalk is built entirely using XML as the data representation mechanism. Everything that passes through BizTalk is represented as an XML document at one point or another. If the solution requires the use of BizTalk schemas, knowledge of XML Schema Definition (XSD) language and XML document manipulation is a must-have. One of the great features of the product is that it exposes nearly all of its internal data structures and properties and allows you to examine, search, and modify most of the metadata that is stored for a BizTalk artifact, whether it is a message in the system or a system object such as a port. To take advantage of this, however, you must understand how XSD schemas are coded, and how to examine and manipulate them using the Microsoft XML Document Object Model.

Windows Management Instrumentation

Windows Management Instrumentation, or WMI, is one of the most underused and potentially least understood features of the Windows operating system. WMI is a management technology that allows scripts to monitor and control managed resources throughout the network. Resources include hard drives, file systems, operating system settings, processes, services, shares, registry settings, networking components, event logs, users, and groups. WMI is built into clients with Windows 2000 or above, and can be installed on any other 32-bit Windows client.

Thorough understanding of WMI is not required for a small to medium-sized BizTalk implementation, but it certainly helps. BizTalk has numerous WMI events that can be subscribed to and monitored to help give detailed information as to the overall health of a BizTalk solution. Additionally, custom WMI events can be coded and inserted into your BizTalk application to allow for custom instrumentation code that will be available to most enterprise server monitoring tools such as Microsoft Operations Manager. This is something that is often overlooked. Teams will generally implement instrumentation in the forms of performance logging, text files, debug output, etc., but this data is rarely available to system administrators who can most benefit from it. Implementing custom application instrumentation using WMI can help facilitate transitions for an application into a production environment scenario.

Microsoft Operations Manager (MOM) is Microsoft's enterprise system management tool and is able to listen to WMI events. MOM can be used to notify administrators in the case of a system failure and provide information as to the overall health of a system. BizTalk 2006 has a management pack for MOM that is used for this purpose. Additionally, custom management packs can be created to provide additional counters and information to MOM.

SQL Server

BizTalk uses Microsoft SQL Server as its primary data storage mechanism. SQL Server provides BizTalk with the ability to cluster database installations to achieve fault tolerance and high availability, as well as providing a supported and reliable way to ensure transaction stability and overall performance.

Deep SQL Server knowledge is not required for low-volume application designs; however, for systems that will have large volumes of messages being processed (in the order of greater than 20 messages per second), SQL Server knowledge on the team is a definite must-have. In high-volume scenarios, it becomes increasing important to understand how to properly distribute the BizTalk Server databases to achieve maximum performance as well as maintain reliability and scalability options. SQL Server also includes several performance-tuning tools such as the SQL Profiler, Tracing, and the SQL Server Enterprise Management Console, which allow DBAs to tune and monitor traffic within the database. This performance-tuning knowledge becomes increasingly more important as the volume of transactions a system is designed to accommodate increases.

What Are the Project Timelines? Will the Team Be Using BizTalk Exclusively to Decrease Development Time?

If the only reason you are looking at BizTalk is because you hope it will allow you to hit a project milestone or decrease overall project risk, then you need make sure you are using BizTalk for the right reasons. BizTalk is not a cure-all solution. Projects that are running behind schedule do not just have "BizTalk thrown in there" to make the unrealistic deadline go away. Most project managers want to find the "magic bullet" that can solve all their projects' problems. BizTalk has a place in these types of scenarios, but it is a means to an end, and not the end in and of itself.

Implementing BizTalk within any project carries with it a certain level of risk and challenges. In the end, what may happen is that you trade one risk for another. In order to be successful in this type of scenario, the solution must have features or subsystems on the projects' critical path that can be easily inserted and replaced with the appropriate BizTalk tool while at the same time not increasing the overall project risk.

If your project is in danger of not meeting a deadline, and you think that BizTalk may be just the tool for implementing some of the project requirements, the easiest way to evaluate whether BizTalk will work is to implement a series of simple proof of concepts. In most situations, software does not need to be purchased for this, and a 120-day demo copy of the product can be downloaded from www.microsoft.com/biztalk/evaluation/trial/default.mspx for free. During this proof-of-concept phase, Microsoft field staff can be engaged to help ensure the prototypes are successful and address any technical issues that may arise. Involving Microsoft field staff in these types of situations is generally the easiest way to help alleviate the risk of taking on the creation and success of any proof-of-concept or prototype activities while engaged in an already risky project.

Is There Enough in the Budget to Implement BizTalk?

Whether you have the budget to implement BizTalk will depend on the size and complexity of the solution. Typical implementations cost anywhere from $15,000 on the low end and upwards of $500,000+ on the high end for hardware and software costs. These are exclusive of any custom development, support, or hosting costs.

BizTalk Server Editions

BizTalk as a solution provides many options for configuring the product to help ensure the right mix of hardware and software is purchased. The product is available in the following editions:

- **Standard Edition**: BizTalk Server 2006 Standard Edition is designed for small and medium-sized organizations. It enables you to integrate up to 10 internal applications with up to 20 external trading partners. This edition supports only one- or two-processor machines and does not support clustered deployments. If the product is installed on a multiprocessor machine, it will only use up to two of the available processors. Since Standard Edition does not support scaling-out solutions using multimachine BizTalk Server group processing or scaling to multiple processor machines, it is generally intended for stand-alone applications with moderate transaction volumes. Standard Edition still allows for scaling out the SQL Server that BizTalk Server uses. For example, it is possible to install BizTalk Standard Edition on a single-processor machine but have it access SQL Server Enterprise Edition on a multiprocessor, clustered environment.
- **Enterprise Edition**: BizTalk Server Enterprise Edition is the full-featured version of the product. It is able to use multiprocessor machines and supports BizTalk Server application clustering via BizTalk Server Groups.
- **Developer Edition**: BizTalk Server Developer Edition is the enterprise edition of the product licensed to run in development and testing environments only. This is ideal for installation on development team members' workstations and for installation in staging or quality assurance environments for application testing purposes only; it cannot be used in any production scenarios. Developer Edition is available as part of a Microsoft Developer Network (MSDN) subscription.

SQL Server Editions

Like BizTalk, SQL Server also has separate editions to meet the varying needs of customer requirements and installation situations. The two main versions of SQL Server are **Standard Edition** and **Enterprise Edition**. The key difference between the two versions in the context of a BizTalk Server environment is their support for database clustering. Only SQL Server Enterprise Edition supports database clustering, and it requires Microsoft Windows Server Enterprise Edition as its installed operating system to install the cluster server software.

Required BizTalk Server and SQL Server Hardware

The minimum hardware requirements for BizTalk Server as published by Microsoft are as follows:

- Minimum configuration:
  - 900 megahertz (MHz) or higher Intel Pentium–compatible CPU
  - 1024 megabytes (MB) of RAM
  - 6 gigabyte (GB) hard disk
  - CD-ROM or DVD-ROM drive

Please note that this is a minimum configuration to allow the product to run. In actuality, a typical BizTalk Server machine will have some variant of the following hardware:

- Common configuration:
    - 2.6 GHz or higher Pentium IV
    - 1536MB of RAM
    - RAID 5 hard disk array for application data
    - RAID 0+1 hard disk array for operating system
    - CD-ROM or DVD-ROM drive

As stated before, every BizTalk installation needs a SQL Server instance. In low-volume scenarios, it is possible to install both BizTalk Server and SQL Server on the same physical machine. This configuration will work quite well and is fully supported. Issues with this configuration tend to arise when transaction volume increases and additional processing resources are required to service incoming requests.

In high-volume scenarios, it is not realistic to expect to be able to run both BizTalk and SQL Server on the same physical machine. From a performance perspective, SQL Server traditionally is a "RAM hungry" application. This is due mainly to the memory management system used internally within the product. The upside to this is that query executing time within SQL Server is generally extremely fast, and the server engine is able to cache query results to increase performance. It is possible to configure SQL Server to use less memory, but the performance of the database engine could degrade accordingly.

From a fault-tolerance perspective, it would be disastrous to expect maximum uptime and reliability from a shared BizTalk/SQL Server hardware platform. SQL Server enterprise provides clustering support to allow for failover occurring across separate physical machines in the event of a hardware failure. Database clustering is a key component to ensure that the environment achieves maximum uptime despite failures in hardware or software.

A typical SQL Server Enterprise Edition clustered environment is shown in Figure 1-4 and will have a configuration similar to the following:

- Common high-volume database server configuration:
    - 3.0 GHz or higher Pentium IV
    - 4096MB of RAM
    - RAID 5 hard disk array for application installations
    - RAID 0+1 hard disk array for operating system
    - CD-ROM or DVD-ROM drive
    - Shared fiber channel array or storage area network (SAN) for SQL Server database files

---

■ **Note** In a clustered environment, two database server machines are required. In most configurations, the first server accepts database requests, while the other is considered a "hot standby." This configuration is referred to as **active/passive clustering**.

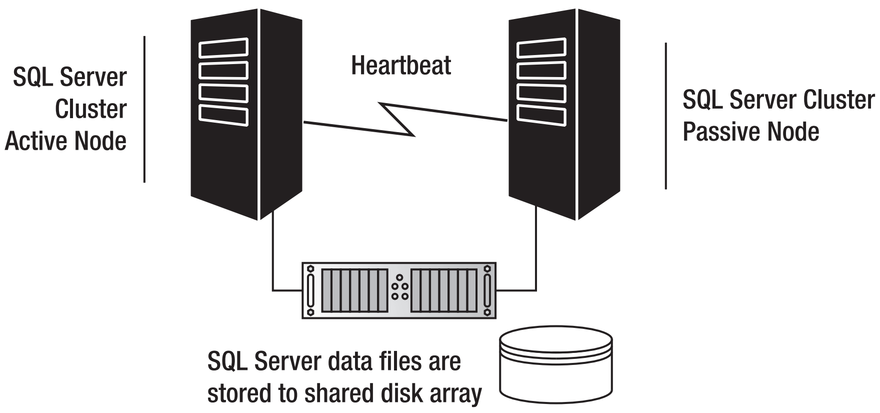

SQL Server Cluster Active Node
Heartbeat
SQL Server Cluster Passive Node
SQL Server data files are stored to shared disk array

Figure 1-4. *SQL Server clustering configuration*

Software Costs

When estimating costs for a new BizTalk project, it is imperative to consider the total cost of what the software will be. Don't forget, too, that you must consider the cost of both BizTalk *and* SQL Server.

BizTalk Costs

When project managers begin estimating the cost for the solutions hardware environment, many often forget one thing—BizTalk is not freeware. Each edition has it's own per-processor cost. The software licensing costs are shown in Table 1-1.

Table 1-1. *BizTalk Server Pricing*

Edition	BizTalk Server 2006
Enterprise Edition	$29,999 US per processor
Standard Edition	$8,499 US per processor
Developer Edition	$499 US per developer, or as part of an MSDN Universal subscription

SQL Server Costs

SQL Server can be licensed in two modes: per client access license (CAL) or per processor. Per-CAL licensing requires that you purchase a separate license for each concurrent connection to the SQL Server database, whereas per-processor licensing licenses the software based on the number of processors installed in the machine. Given the complex nature of BizTalk and SQL Server processing, it is generally not advisable to license SQL Server based on the CAL model. Developer and testing servers are permitted to use SQL Server Developer Edition. Like BizTalk Developer Edition, this is a full-featured version of the product licensed for use on development and testing machines only and is available through MSDN subscriptions. Per-processor licensing costs are outlined in Table 1-2.

Table 1-2. *SQL Server Pricing*

Edition	SQL Server 2000
Enterprise Edition	$24,999 US per processor
Standard Edition	$5,999 US per processor
Developer Edition	Part of MSDN Universal subscription

How Many Servers Are Required to Implement a BizTalk Solution?

Low-volume systems can be implemented with one BizTalk server machine and one SQL Server machine. High-volume solutions can use eight or more machines depending on the type and volume of processing required.

Typical Low-Volume Solution Configuration

Figure 1-5 shows a configuration that will work for a typical low-volume transaction system.

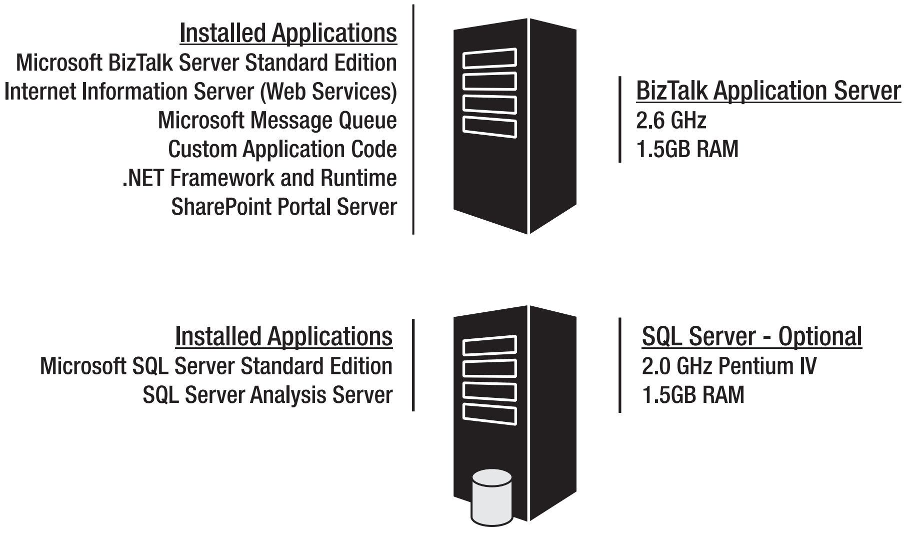

Installed Applications
Microsoft BizTalk Server Standard Edition
Internet Information Server (Web Services)
Microsoft Message Queue
Custom Application Code
.NET Framework and Runtime
SharePoint Portal Server
BizTalk Application Server
2.6 GHz
1.5GB RAM
Installed Applications
Microsoft SQL Server Standard Edition
SQL Server Analysis Server
SQL Server - Optional
2.0 GHz Pentium IV
1.5GB RAM

Figure 1-5. *Low-volume BizTalk Server configuration*

Typical High-Volume System Configuration

High-volume transaction systems require a more robust configuration than is shown in Figure 1-5. Figure 1-6 gives one example of how you might configure BizTalk to handle a high volume of transactions.

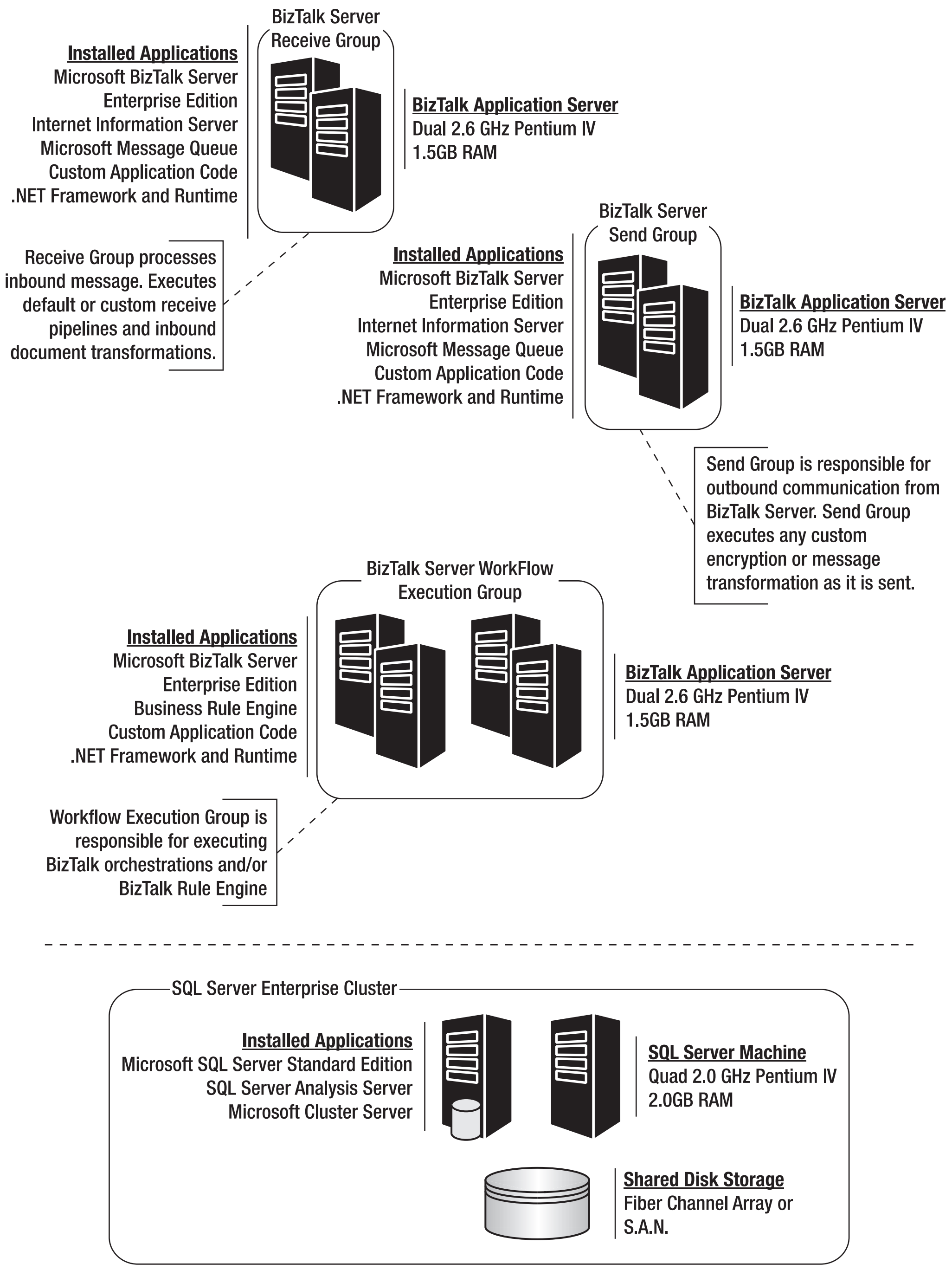

BizTalk Server
Receive Group
Installed Applications
Microsoft BizTalk Server
Enterprise Edition
Internet Information Server
Microsoft Message Queue
Custom Application Code
.NET Framework and Runtime
BizTalk Application Server
Dual 2.6 GHz Pentium IV
1.5GB RAM
Receive Group processes inbound message. Executes default or custom receive pipelines and inbound document transformations.
BizTalk Server
Send Group
Installed Applications
Microsoft BizTalk Server
Enterprise Edition
Internet Information Server
Microsoft Message Queue
Custom Application Code
.NET Framework and Runtime
BizTalk Application Server
Dual 2.6 GHz Pentium IV
1.5GB RAM
Send Group is responsible for outbound communication from BizTalk Server. Send Group executes any custom encryption or message transformation as it is sent.
BizTalk Server WorkFlow
Execution Group
Installed Applications
Microsoft BizTalk Server
Enterprise Edition
Business Rule Engine
Custom Application Code
.NET Framework and Runtime
BizTalk Application Server
Dual 2.6 GHz Pentium IV
1.5GB RAM
Workflow Execution Group is responsible for executing BizTalk orchestrations and/or BizTalk Rule Engine
SQL Server Enterprise Cluster
Installed Applications
Microsoft SQL Server Standard Edition
SQL Server Analysis Server
Microsoft Cluster Server
SQL Server Machine
Quad 2.0 GHz Pentium IV
2.0GB RAM
Shared Disk Storage
Fiber Channel Array or
S.A.N.

Figure 1-6. *High-volume BizTalk Server implementation*

How Much Custom Code Are You and Your Team Willing to Create? Would You Rather Use Completely Out-of-the-Box Functionality?

Most BizTalk solutions do not require significant amounts (>20,000 lines) of code to implement. Simple projects may not require any. The amount of custom code required will depend greatly on the type of system and the complexity of the business logic that is required.

There are several key types of processing that require a BizTalk solution to implement custom code:

- **Custom message processing logic**: This generally refers to custom pipelines and custom pipeline components. If you need to perform things like custom encryption or decryption, custom digital signature or message verification, compression routines, or custom envelope processing, you will need to write .NET code.
- **Custom transformation logic**: Transformation logic is handled within BizTalk maps. Most transformations can be implemented using the default BizTalk **functoids**. Functoids are pieces of application code that are used to manipulate how values from an incoming message are mapped to the outgoing message within a BizTalk transformation. BizTalk ships with over a hundred default functoids that are able to accommodate most standard tasks. In the case where custom logic needs to be created, most projects will want to code custom functoids to facilitate code reuse. You will want to code custom functoids when you notice that the same scripts or routines are implemented several times across multiple maps.
- **Custom business logic**: .NET components are usually called within orchestrations and contain common business procedures and classes. This is not a BizTalk item per se as the project will need to implement this logic regardless of the platform.
- **Workflow automation**: These are BizTalk orchestrations. In most situations, there is one BizTalk orchestration for each workflow to be automated. In many cases, complicated workflow is broken out into multiple orchestrations so that common pieces can be used in several orchestrations and be coded by multiple developers.
- **Custom adapters**: One of the key selling features of BizTalk is the vibrant adapter partner community that supports it. Adapters allow BizTalk applications to communicate with different transport mechanisms and applications without requiring that the coder understand the details of how this communication will work. Adapters exist for over 300 platforms, operating systems, and ERP applications. In certain circumstances, either an adapter is not available for a desired platform or it is more cost effective to develop a custom adapter from scratch due to licensing costs. In these cases, you will need to create a custom adapter using the BizTalk Adapter Framework classes. Depending on the transaction volume requirements, transport specifics, and protocols involved, this may be a difficult task for the novice developer.

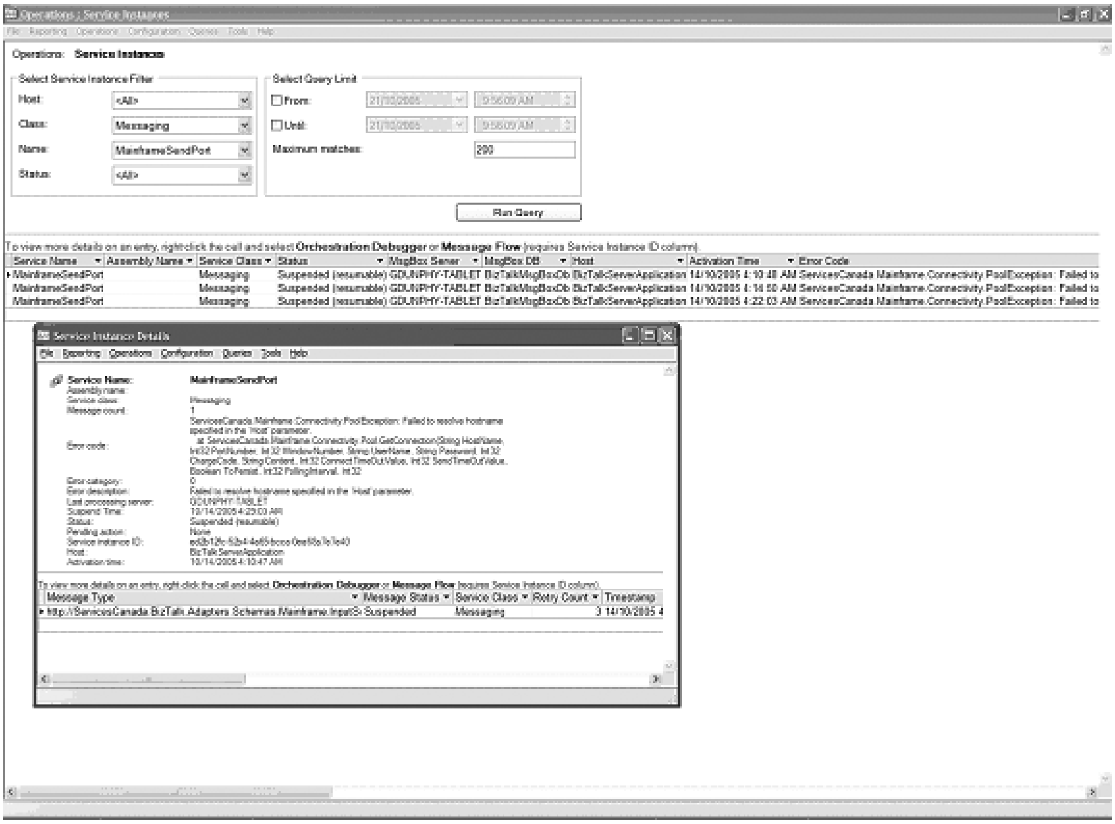
Is BizTalk Suited for the Application in Question?

Refer to the four main types of BizTalk projects—workflow automation, legacy application integration, trading partner exchange, and organizational message broker scenarios. If your application has any of these types of pieces, BizTalk may be a fit. If not, you need to examine where and why BizTalk is being evaluated.

The key thing to remember in this type of situation is to define early in the solution's design phase what data will be exchanged with BizTalk, what the schema definitions are, and how data will be returned to the calling application. Another key decision is to decide whether the main application requires a **synchronous** or an **asynchronous** response from BizTalk and what the threshold values are for an acceptable transaction. If the calling application needs a subsecond synchronous response from BizTalk to process the message, it is imperative that the system be sized properly to ensure that response times are acceptable. If the communication can be asynchronous, it is easier to restrict the flow of messages to an acceptable level either by using a queuing mechanism or some sort of batch processing.

Will Every Transaction in the System Need Monitoring? Will the Tracked Data Need Saving for Archiving Purposes?

This issue is often overlooked when a new project begins. BizTalk provides a very simplistic user interface to view the message specifics and transaction history within the product. This interface, called the Health and Activity Tracker (HAT), is shown in Figures 1-7 and 1-8.


Operations : Service Instances
File Reporting Operations Configuration Queries Tools Help
Run Query

Operations: **Service Instances**

Select Service Instance Filter

Host:	<All>
Class:	Messaging
Name:	MainframeSendPort
Status:	<All>

Select Query Limit

☐ From:	21/10/2005	9:56:09 AM
☐ Until:	21/10/2005	9:56:09 AM
Maximum matches:		200

To view more details on an entry, right-click the cell and select **Orchestration Debugger** or **Message Flow** (requires Service Instance ID column).

Service Name	Assembly Name	Service Class	Status	MsgBox Server	MsgBox DB	Host	Activation Time	Error Code
MainframeSendPort		Messaging	Suspended (resumable)	GDUNPHY-TABLET	BizTalkMsgBoxDb	BizTalkServerApplication	14/10/2005 4:10:48 AM	ServicesCanada.Mainframe.Connectivity.PoolException: Failed to
MainframeSendPort		Messaging	Suspended (resumable)	GDUNPHY-TABLET	BizTalkMsgBoxDb	BizTalkServerApplication	14/10/2005 4:16:50 AM	ServicesCanada.Mainframe.Connectivity.PoolException: Failed to
MainframeSendPort		Messaging	Suspended (resumable)	GDUNPHY-TABLET	BizTalkMsgBoxDb	BizTalkServerApplication	14/10/2005 4:22:03 AM	ServicesCanada.Mainframe.Connectivity.PoolException: Failed to

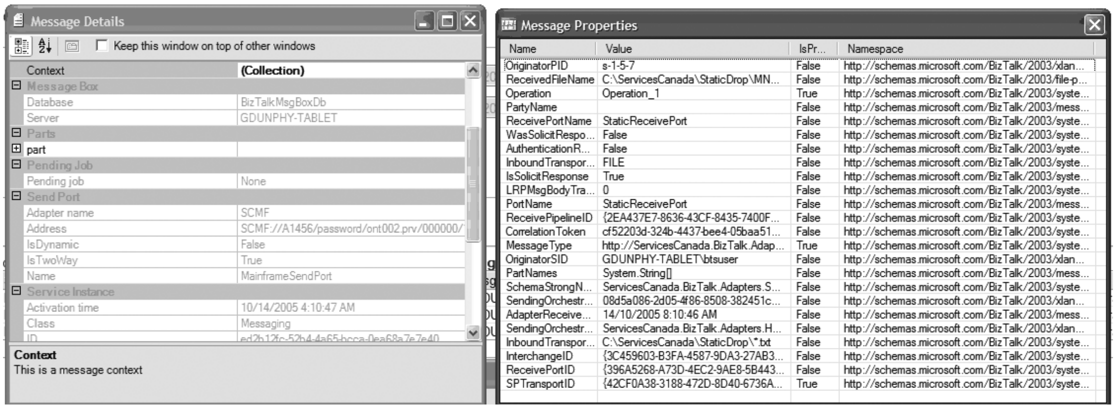
Service Instance Details

Service Name:	**MainframeSendPort**
Assembly name:	
Service class:	Messaging
Message count:	1
Error code:	ServicesCanada.Mainframe.Connectivity.PoolException: Failed to resolve hostname specified in the 'Host' parameter. at ServicesCanada.Mainframe.Connectivity.Pool.GetConnection(String HostName, Int32 PortNumber, Int32 WindowNumber, String UserName, String Password, Int32 ChargeCode, String Content, Int32 ConnectTimeOutValue, Int32 SendTimeOutValue, Boolean ToPersist, Int32 PollingInterval, Int32
Error category:	0
Error description:	Failed to resolve hostname specified in the 'Host' parameter.
Last processing server:	GDUNPHY-TABLET
Suspend Time:	10/14/2005 4:29:03 AM
Status:	Suspended (resumable)
Pending action:	None
Service instance ID:	ed2b12fc-52b4-4a65-bcca-0ea68a7e7e40
Host:	BizTalkServerApplication
Activation time:	10/14/2005 4:10:47 AM

To view more details on an entry, right-click the cell and select **Orchestration Debugger** or **Message Flow** (requires Service Instance ID column).

Message Type	Message Status	Service Class	Retry Count	Timestamp
http://ServicesCanada.BizTalk.Adapters.Schemas.Mainframe.InputS	Suspended	Messaging	3	14/10/2005 4

Figure 1-7. *Health and Activity Tracker*



Context	(Collection)
Message Box	
Database	BizTalkMsgBoxDb
Server	GDUNPHY-TABLET
Parts	
part	
Pending Job	
Pending job	None
Send Port	
Adapter name	SCMF
Address	SCMF://A1456/password/ont002.prv/000000/
IsDynamic	False
IsTwoWay	True
Name	MainframeSendPort
Service Instance	
Activation time	10/14/2005 4:10:47 AM
Class	Messaging
ID	ed2b12fc-52b4-4a65-bcca-0ea68a7e7e40

Context
This is a message context

Name	Value	IsPr...	Namespace
OriginatorPID	s-1-5-7	False	http://schemas.microsoft.com/BizTalk/2003/xlan...
ReceivedFileName	C:\ServicesCanada\StaticDrop\MN...	False	http://schemas.microsoft.com/BizTalk/2003/file-p...
Operation	Operation_1	True	http://schemas.microsoft.com/BizTalk/2003/syste...
PartyName		False	http://schemas.microsoft.com/BizTalk/2003/mess...
ReceivePortName	StaticReceivePort	False	http://schemas.microsoft.com/BizTalk/2003/syste...
WasSolicitRespo...	False	False	http://schemas.microsoft.com/BizTalk/2003/syste...
AuthenticationR...	False	False	http://schemas.microsoft.com/BizTalk/2003/syste...
InboundTranspor...	FILE	False	http://schemas.microsoft.com/BizTalk/2003/syste...
IsSolicitResponse	True	False	http://schemas.microsoft.com/BizTalk/2003/syste...
LRPMsgBodyTra...	0	False	http://schemas.microsoft.com/BizTalk/2003/syste...
PortName	StaticReceivePort	False	http://schemas.microsoft.com/BizTalk/2003/mess...
ReceivePipelineID	{2EA437E7-8636-43CF-8435-7400F...	False	http://schemas.microsoft.com/BizTalk/2003/syste...
CorrelationToken	cf52203d-324b-4437-bee4-05baa51...	False	http://schemas.microsoft.com/BizTalk/2003/syste...
MessageType	http://ServicesCanada.BizTalk.Adap...	True	http://schemas.microsoft.com/BizTalk/2003/syste...
OriginatorSID	GDUNPHY-TABLET\btsuser	False	http://schemas.microsoft.com/BizTalk/2003/xlan...
PartNames	System.String[]	False	http://schemas.microsoft.com/BizTalk/2003/mess...
SchemaStrongN...	ServicesCanada.BizTalk.Adapters.S...	False	http://schemas.microsoft.com/BizTalk/2003/syste...
SendingOrchestr...	08d5a086-2d05-4f86-8508-382451c...	False	http://schemas.microsoft.com/BizTalk/2003/xlan...
AdapterReceive...	14/10/2005 8:10:46 AM	False	http://schemas.microsoft.com/BizTalk/2003/mess...
SendingOrchestr...	ServicesCanada.BizTalk.Adapters.H...	False	http://schemas.microsoft.com/BizTalk/2003/xlan...
InboundTranspor...	C:\ServicesCanada\StaticDrop\*.txt	False	http://schemas.microsoft.com/BizTalk/2003/syste...
InterchangeID	{3C459603-B3FA-4587-9DA3-27AB3...	False	http://schemas.microsoft.com/BizTalk/2003/syste...
ReceivePortID	{396A5268-A73D-4EC2-9AE8-5B443...	False	http://schemas.microsoft.com/BizTalk/2003/syste...
SPTransportID	{42CF0A38-3188-472D-8D40-6736A...	True	http://schemas.microsoft.com/BizTalk/2003/syste...

Figure 1-8. *Health and Activity Tracker message details*

The HAT can provide the data that is stored within the message and give the message context properties; however, this data is essentially system-level data. There is no "business-relevant" information in the context other than the message type and any exception information. If there is a requirement to see the transaction from an end-user perspective (i.e., the message is received, processed by system XXX, and failed at updating system YYY), the HAT interface will not be sufficient. Often the type of application that needs to see a transaction from an end-user perspective is integrated into an existing SharePoint portal site using BAS. Developing this application is generally not a trivial task and can add time onto the final product's delivery date. If the type of information is volumetric data (e.g., how many transactions failed, how many were successful, what was the volume at peak processing time), the data can be created using BAM and accessed with Microsoft Excel or another similar tool. Implementing a BAM-based solution is generally less time consuming than creating a customized BAS SharePoint application. Starting in BizTalk 2006, the BAM service ships with a default SharePoint portal application that allows you to view all BAM-related information within the system. This base application is usually then modified or integrated with another main SharePoint site to provide the relevant business-level metrics required.