

---

# Understanding Microsoft Integration Technologies

## **A Guide to Choosing a Solution**

June 2005

© Copyright Microsoft Corporation 2005. All rights reserved.

---

# Contents

<b>STYLES OF INTEGRATION.....</b>	<b>3</b>
<b>INTEGRATING APPLICATIONS DIRECTLY.....</b>	<b>3</b>
ASMX, .NET REMOTING, AND ENTERPRISE SERVICES.....	4
INDIGO.....	4
<i>Describing Indigo</i> .....	4
<i>When to Use Indigo</i> .....	5
<i>Indigo and Other Integration Technologies</i> .....	5
<b>INTEGRATING APPLICATIONS THROUGH QUEUES.....</b>	<b>5</b>
MICROSOFT MESSAGE QUEUING.....	6
<i>Describing MSMQ</i> .....	6
<i>When to Use MSMQ</i> .....	6
<i>MSMQ and Other Integration Technologies</i> .....	6
SQL SERVICE BROKER.....	6
<i>Describing SSB</i> .....	6
<i>When to Use SSB</i> .....	7
<i>SSB and Other Integration Technologies</i> .....	7
INDIGO AND QUEUED COMMUNICATION.....	8
<b>INTEGRATING WITH APPLICATIONS AND DATA ON IBM SYSTEMS.....</b>	<b>9</b>
HOST INTEGRATION SERVER 2004.....	9
<i>Describing HIS 2004</i> .....	9
<i>When to Use HIS 2004</i> .....	10
<i>HIS 2004 and Other Integration Technologies</i> .....	11
<b>INTEGRATING APPLICATIONS THROUGH A BROKER.....</b>	<b>11</b>
BIZTALK SERVER 2006.....	11
<i>Describing BTS 2006</i> .....	11
<i>When to Use BTS 2006</i> .....	12
<i>BTS 2006 and Other Integration Technologies</i> .....	13
<b>INTEGRATING DATA.....</b>	<b>13</b>
SQL SERVER INTEGRATION SERVICES.....	13
<i>Describing SSIS</i> .....	13
<i>When to Use SSIS</i> .....	14
<i>SSIS and Other Integration Technologies</i> .....	14
SQL SERVER REPLICATION.....	15
<i>Describing SQL Server Replication</i> .....	15
<i>When to Use SQL Server Replication</i> .....	15
<i>SQL Server Replication and Other Integration Technologies</i> .....	16
<b>CONCLUSION.....</b>	<b>16</b>
<b>APPENDIX: SUMMARIZING MICROSOFT INTEGRATION TECHNOLOGIES.....</b>	<b>17</b>

## Styles of Integration

There is no silver bullet for application integration. Different situations call for different solutions, each targeting a particular kind of problem. While a one-size-fits-all solution would be nice, the inherent diversity of integration challenges makes such a simplistic approach impossible. To address this broad set of problems, Microsoft has created several different integration technologies, each targeting a particular group of scenarios. Together, these technologies provide a comprehensive, unified, and complete integration solution.

Microsoft's integration technologies can be grouped into several categories:

Technologies for integrating applications directly, including ASP.NET Web Services (ASMX), .NET Remoting, and Enterprise Services. All of these will soon be subsumed by a unified foundation for service-oriented applications, code-named *Indigo*.

Technologies for integrating applications through queues, including Microsoft Message Queuing (MSMQ) and SQL Service Broker (SSB). Both of these will also be usable via Indigo.

Technologies for integrating Windows applications with applications and data on IBM systems. This diverse set of solutions is provided by Host Integration Server (HIS) 2004.

Technologies for integrating applications through a broker, the approach taken by BizTalk Server (BTS) 2006.

Technologies for integrating data, including SQL Server Integration Services (SSIS) and SQL Server Replication.

Each of these technologies has its own distinct role to play in integrating applications. All of them also have important things in common, however. All can be used from a single development environment, Visual Studio 2005, and all rely on a common foundation, the .NET Framework. Given this, combining these technologies is straightforward, making it easier to solve complex integration challenges.

This paper looks at these varied technologies, describing the scenarios for which each one is the best choice. The goal is to simplify the process of choosing the most appropriate Microsoft technology or technology combination for solving a particular problem.

## Integrating Applications Directly

The defining characteristic of application logic, the thing that most clearly distinguishes it from data, is that it is active—it does something. The simplest way to connect this active logic is to directly connect one part of an application to another using some kind of remote procedure call (RPC). .NET Framework applications today have three options for doing this: ASMX, .NET Remoting, and Enterprise Services. Once it's released, Indigo will subsume all three of these technologies, providing a common solution for this and other scenarios.

## ASMX, .NET Remoting, and Enterprise Services

The communication technologies included in today's .NET Framework are well known. Still, it's worth briefly summarizing the role each one plays:

ASP.NET Web Services (ASMX) lets Windows applications communicate directly with applications running on Windows or other operating systems via SOAP, the foundation protocol for web services.

.NET Remoting lets Windows applications communicate directly with other Windows applications using the traditional distributed objects approach.

Enterprise Services lets Windows applications communicate directly with other Windows applications, letting those applications use distributed transactions, object lifetime management, and other functions.

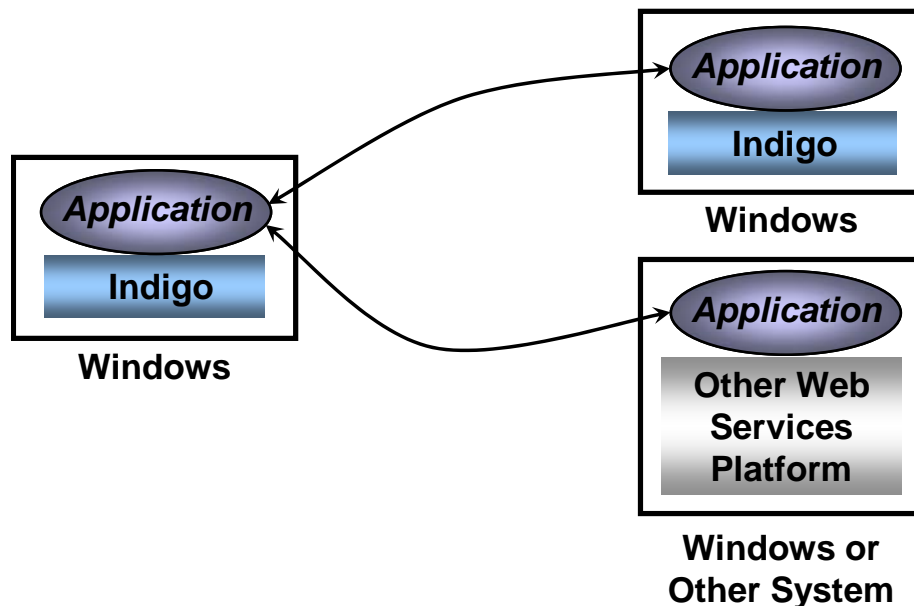
While all of these technologies are useful, why not have just one that can be used in all of these situations? As described next, this is exactly what Indigo provides.

## Indigo

Indigo is the code name for Microsoft's forthcoming extension to the .NET Framework for building service-oriented applications. Scheduled to be released in 2006, it will be available for the next release of the Windows client, code-named *Longhorn*, for Windows XP, and for Windows Server 2003.

### Describing Indigo

The agreement by all major vendors to standardize on web services for application-to-application communication is a watershed in integration. Web services technologies, based on SOAP, provide a direct way to connect software running on platforms from multiple vendors. Indigo implements SOAP and the associated group of multi-vendor agreements known as the WS-\* specifications. Other major vendors are implementing these same technologies, allowing reliable, secure, and transactional communication between applications running on diverse systems.



As the figure on the previous page shows, applications built on Indigo can communicate directly with other Indigo-based applications or with applications built on other web services platforms, such as a J2EE application server. When communicating with non-Indigo applications, the wire protocol is standard text-based SOAP, perhaps with additions defined by one or more of the WS-\* specifications. When communicating with other Indigo applications, the wire protocol can be an optimized binary version of SOAP. A single Indigo application can use both options simultaneously, allowing high performance for homogeneous partners together with cross-platform interoperability.

Indigo applications send and receive SOAP messages over one or more *channels*. An HTTP channel is provided, for instance, that allows communication according to the agreements defined by the Web Services Interoperability (WS-I) Organization. Indigo also provides other channels, including channels that support MSMQ and SSB. As described later in this paper, this gives Indigo applications the benefits of queued messaging when they're communicating with other Windows applications.

### When to Use Indigo

The primary integration scenarios for Indigo are:

Direct web services communication between a Windows application and an application built on another web services platform.

Direct communication between a Windows application and another Windows application.

Communication via message queues between a Windows application and another Windows application using Indigo over MSMQ or SSB. Indigo provides a common programming interface for both queued and direct communication, and it also allows a single application to expose endpoints supporting both communication styles.

### Indigo and Other Integration Technologies

Indigo will subsume ASMX, .NET Remoting, and Enterprise Services, the earlier .NET Framework technologies for direct communication. (It's important to realize, however, that applications built with these earlier technologies will continue to work unchanged on systems with Indigo installed.) Indigo will also supersede System.Messaging, the .NET Framework's standard interface to MSMQ. How Indigo fits with queued communication is described in the next section.

## Integrating Applications through Queues

Direct communication between parts of an application is simple to understand and straightforward to implement. It's not always the best solution, however. Rather than communicating directly with another application, it's often better to use queued messaging. This style of communication relies on the presence of one or more queues between the sender and receiver, with applications sending messages to and receiving messages from these queues.

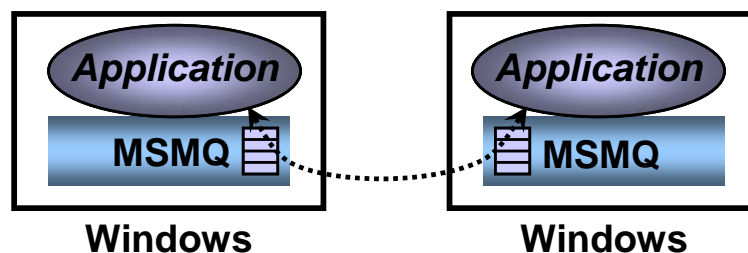
Queued communication lets applications interact in a flexible, adaptable way. One major benefit of this approach is that the receiving application need not be ready to read a message from the sender at the time that message is sent. In fact, the receiver might not even be running when the message is sent. Instead, messages wait in a queue, usually stored on disk, until the receiver is ready to process them.

## Microsoft Message Queuing

Microsoft Message Queuing (MSMQ) is the built-in technology in Windows for application-to-application communication using queued messaging. The current release, MSMQ 3.0, runs on Windows XP and Windows Server 2003, while earlier releases run on older versions of Windows. MSMQ is also available for mobile devices running Windows CE.

### Describing MSMQ

As the figure below illustrates, MSMQ lets Windows applications communicate via message queues. The messages it sends can contain any type of information, and because they're sent asynchronously, the sender need not block waiting for a response. Using asynchronous messaging can be somewhat more complex for a developer than using RPC, but it's nonetheless the right solution in many cases.



### When to Use MSMQ

The primary integration scenarios for MSMQ are:

When asynchronous communication is required between two or more Windows applications.

When the sender and receiver might not be running at the same time.

When message-level logging is required.

### MSMQ and Other Integration Technologies

Once Indigo ships, new applications that today use MSMQ directly will typically access it via the Indigo programming model, as described later in this section. Also, the services provided by MSMQ are similar in some ways to those provided by SSB. Choosing between the two requires understanding SSB, which is described next.

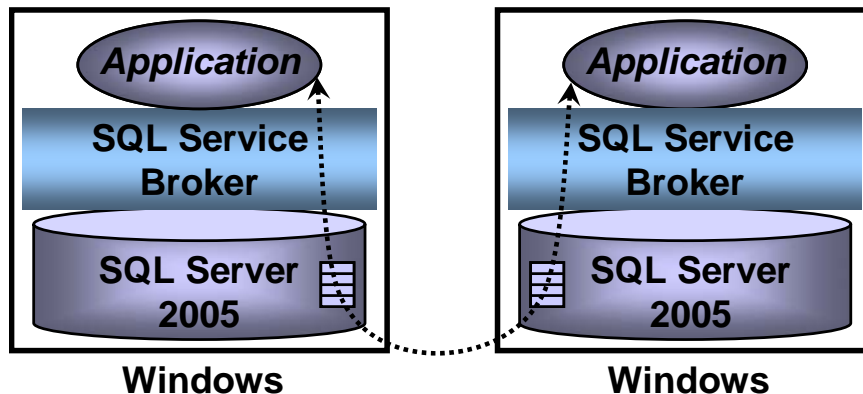
## SQL Service Broker

SQL Server Broker (SSB) is a new communication technology provided as part of SQL Server 2005. Scheduled to be released in late 2005, SQL Server 2005 will be available for Windows XP, Windows Server 2003, and Windows CE. SSB is included with all versions of the product, including Express, Workgroup, Standard, and Enterprise.

### Describing SSB

Most enterprise applications use a database management system (DBMS) in some way. Some applications, such as those written as stored procedures, run within the DBMS itself. Others, such as .NET Framework applications that use ADO.NET, access the DBMS externally. If these applications need to communicate via message queues, why not rely on the DBMS itself to provide those queues? This is exactly what's done by SSB.

Rather than create a standalone queuing infrastructure, as MSMQ does, SSB provides queued communication using SQL Server 2005. The figure below shows how this looks.



To let applications use its queues, SSB adds several verbs to SQL Server's T-SQL language. These verbs allow applications to start a relationship called a *conversation*, then send and receive messages using that conversation. The SQL verbs defined by SSB are intended to be accessed by software created in various ways, including:

- Applications written as stored procedures in T-SQL.

- Applications written as stored procedures in a language based on the Common Language Runtime (CLR), such as C#. This option relies on SQL Server 2005's built-in support for the CLR.

By using SQL Server 2005 to persistently store queued messages, SSB can provide efficient, full-featured communication, including high-performance transactional messaging, integrated backup and recovery mechanisms, and more.

#### When to Use SSB

The primary integration scenarios for SSB are:

- Connecting logic built as stored procedures in one or more separate instances of SQL Server 2005.

- Connecting logic built as a .NET Framework application using SQL Server 2005 with a stored procedure in the same or another instance of SQL Server 2005.

Before Indigo is available, some applications might use SSB directly through ADO.NET. Indigo will become the core interface for applications using message queuing, however, and so most queued applications will use SSB through an Indigo channel, as described later in this section.

Any organization that relies heavily on SQL Server 2005 for building applications, especially when those applications are implemented as stored procedures, will likely use SSB for communication. And because it's part of SQL Server, SSB allows users to have a single product to install, configure, and monitor, together with a single approach to failover, for both a DBMS and a queuing technology.

#### SSB and Other Integration Technologies

The functionality of SSB overlaps to some degree with what MSMQ provides. Both rely on queued messaging, and so either can be used when this style of communication is needed. Its integration with the database means that SSB will normally be the preferred

queuing option for database-oriented applications, but there are also cases where MSMQ is preferable. These include the following:

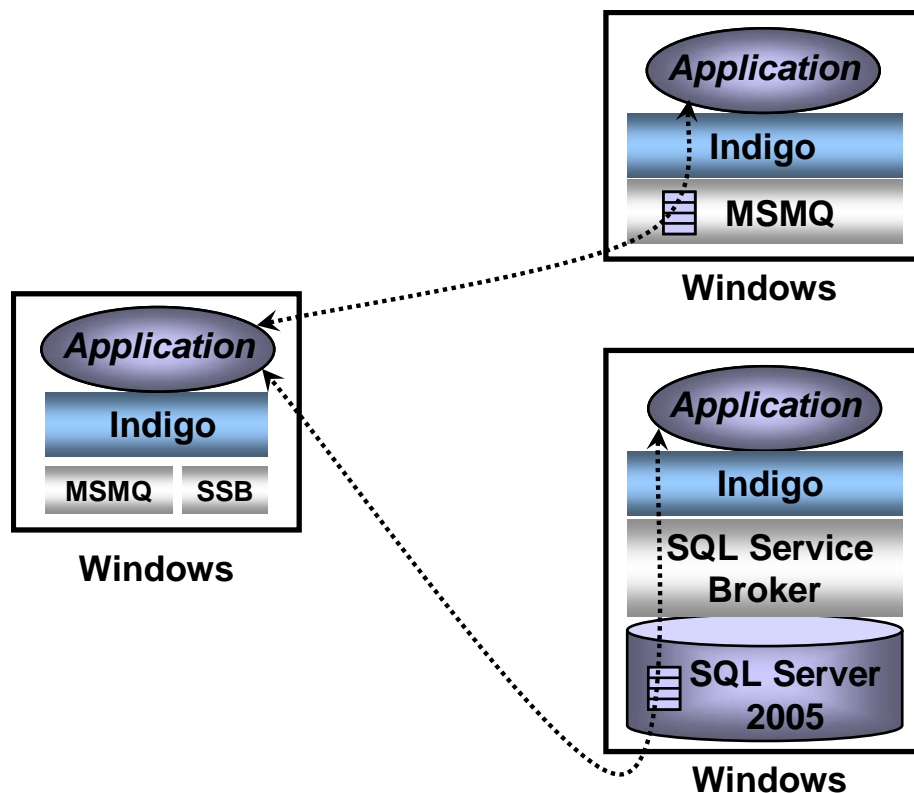
Communication between applications that need only in-memory queuing, i.e., those that don't require messages to be stored persistently in transit. MSMQ supports memory-based queues, while SSB does not.

Situations where the extra license cost of SQL Server 2005 isn't acceptable. Unlike SQL Server 2005, MSMQ is part of Windows, and so there's no extra charge for using it. Even though SSB is included with the Express edition of SQL Server 2005, which doesn't require a paid license, every message sent via SSB must traverse at least one licensed copy of SQL Server 2005.

Once Indigo is available, applications will typically access both MSMQ and SSB via the Indigo programming model. How Indigo works with these queuing technologies is described next.

### Indigo and Queued Communication

While Indigo is based on web services, its channel architecture allows it to send SOAP messages over diverse protocols. For direct communication with non-Windows systems, Indigo will typically send SOAP over HTTP. For queued communication between Windows applications, Indigo will also be able to send SOAP messages over MSMQ and SSB, as the figure below shows.



Even though MSMQ and SSB will be accessible via Indigo, an application will still be able to access these queuing technologies directly. Here's how to make the choice:

Indigo supersedes System.Messaging, the .NET Framework's standard interface to MSMQ, and so most queued applications should use Indigo once it's available. A few MSMQ services aren't available through Indigo, however. For example, Indigo applications that communicate via the MSMQ channel can't peek into a queue of received messages, can't use journaling or queued receipts, and aren't able to let the sending system specify the response queue that a reply message should be sent to. Applications that require these services should still access MSMQ directly using System.Messaging.

As with MSMQ, applications that access SSB through Indigo will see only a subset of SSB's capabilities. An application that needs access to all of SSB's functions might choose to use SSB directly.

Ultimately, the simplicity, flexibility, and ubiquity of the Indigo programming model will lead most developers to use it for the majority of queuing applications implemented outside the database.

## Integrating with Applications and Data on IBM Systems

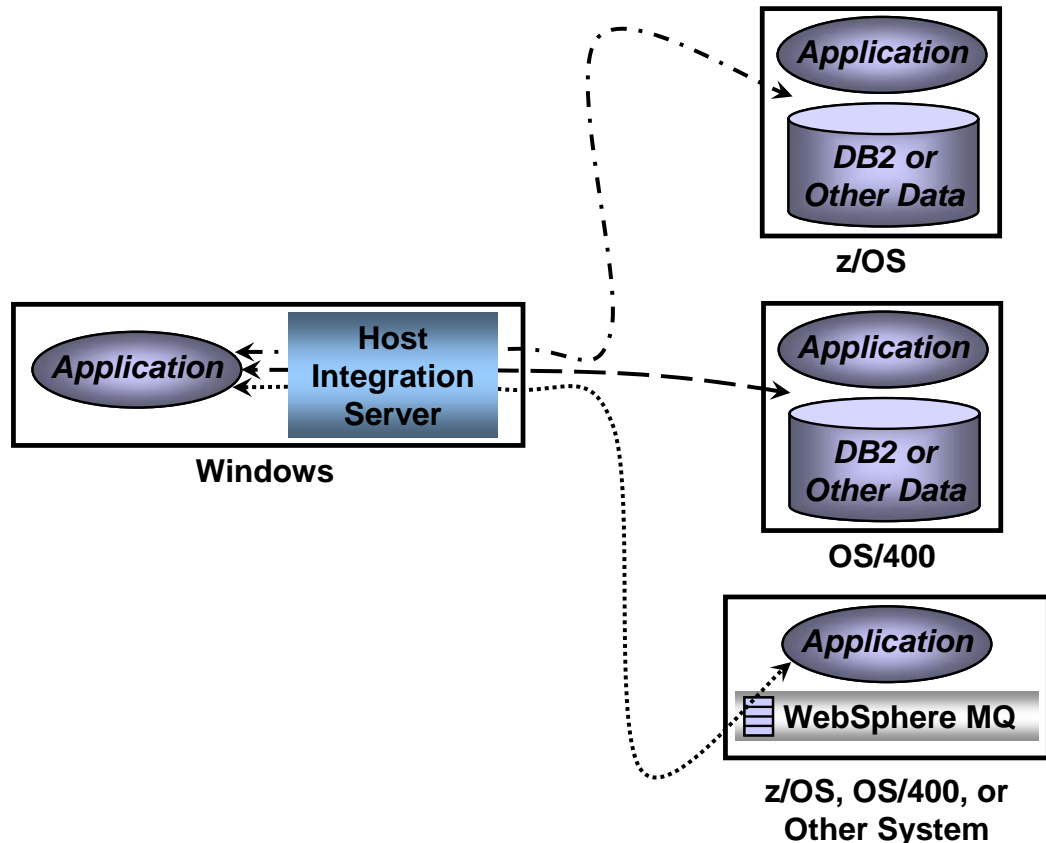
### Host Integration Server 2004

Host Integration Server (HIS) 2004 is a set of technologies focused on connecting to applications and data on IBM mainframes and mid-range systems. The various components it includes run on a variety of systems, including Windows XP, Windows 2000, Windows Server 2003, and Windows 2000 Server.

#### Describing HIS 2004

Many enterprises have substantial investments in IBM mainframe and midrange systems. When new applications are written on Windows, integrating with existing applications and data on these older systems is often essential. Yet doing this can be challenging, since these environments support applications and store data in several different ways. Effectively linking Windows software to these existing IBM systems requires a variety of approaches.

As the figure on the next page illustrates, HIS 2004 contains components that address these diverse requirements. Using various parts of the product, Windows software can access applications and data on IBM zSeries mainframes running z/OS, along with applications and data on IBM iSeries mid-range systems running OS/400. HIS 2004 also includes an MSMQ-MQSeries Bridge, allowing queued messaging between MSMQ and IBM's WebSphere MQ (formerly known as MQSeries).



#### When to Use HIS 2004

The primary integration scenarios for HIS 2004 are:

Connecting Windows systems to IBM zSeries mainframes and iSeries midrange systems using Systems Network Architecture (SNA) and other IBM communication technologies, including SNA over TCP/IP.

Integrating Windows security with IBM mainframe or midrange security systems, including IBM's Resource Access Control Facility (RACF) and Computer Associates' ACF2 and Top Secret.

Accessing existing CICS and IMS applications, either directly from .NET Framework applications using HIS 2004's Transaction Integrator or via Web services.

Creating Windows applications that access data stored on zSeries and iSeries systems, including VSAM data and relational data stored in DB2.

Connecting MSMQ to IBM's WebSphere MQ, allowing messages to be transferred between these two message queuing technologies.

HIS 2004 includes a broad range of functions. As its name suggests, however, all of them are focused on accessing application logic and data stored on IBM mainframe and midrange systems.

## HIS 2004 and Other Integration Technologies

As described later in this paper, some components of HIS 2004 can be used together with other integration technologies such as BTS 2006 and SSIS. The key fact to remember is that whenever a complete solution requires integration with IBM systems, HIS 2004 has a role to play.

### Integrating Applications through a Broker

Rather than integrating applications directly, it sometimes makes more sense to connect them via a *broker*. A broker is software that sits between the applications being integrated, interacting with all of them. By providing a common connection point, brokers avoid the complexity that can arise when several applications are connected directly to one another. Brokers can provide a range of integration services, including transformations between different message formats and support for diverse communication technologies. A broker can also act as a platform for its own application logic, providing the intelligence to control a business process. For Windows, connecting applications through a broker means using BizTalk Server 2006<sup>1</sup>.

#### BizTalk Server 2006

Like its predecessor BizTalk Server (BTS) 2004, BTS 2006 is an integration and business process platform. Available for Windows Server 2003, Windows 2000 Server, and Windows XP, BTS 2006 is Microsoft's solution for brokered application-to-application integration.

#### Describing BTS 2006

As the figure on the next page illustrates, BTS 2006 sits in the middle of a group of applications. By providing adapters for various communication mechanisms, including MSMQ, Indigo, EDI, and many more, BTS 2006 can communicate with Windows and non-Windows applications in a number of different ways. BTS 2006 also provides other integration services, including:

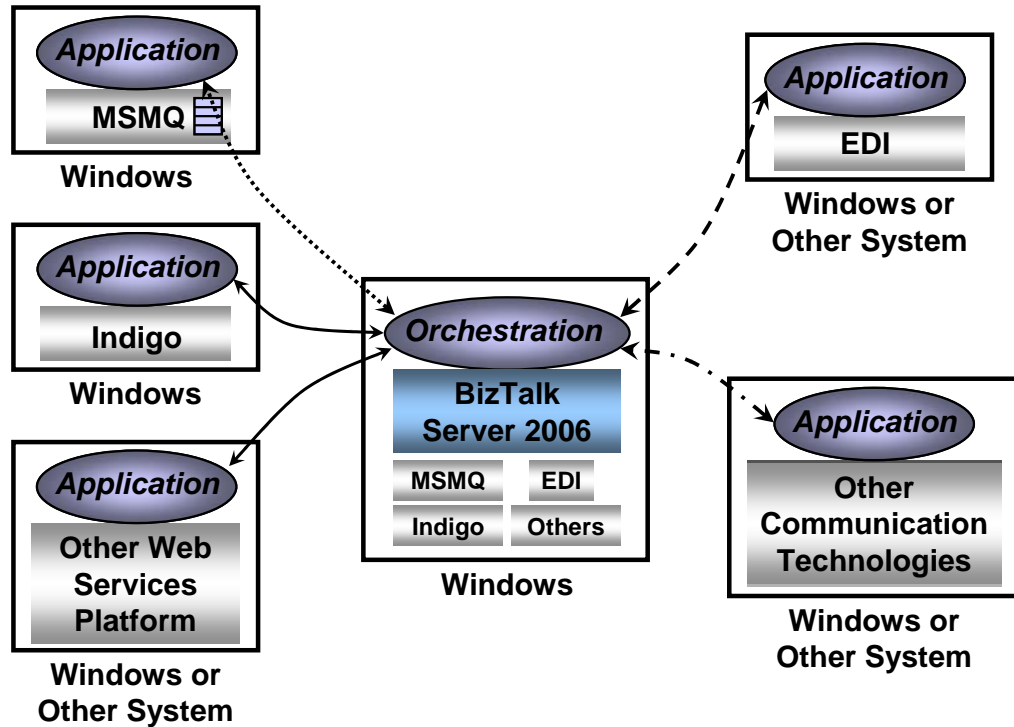
- The ability to graphically define *orchestrations*, logic that interacts with applications on other systems to drive an integrated process, together with runtime services for orchestrations, such as state management and support for long-running transactions.

- Graphical definition of XML schemas for messages, along with the ability to define transformations between incoming and outgoing messages that use those schemas.

- Business-to-business (B2B) integration features, including support for Electronic Data Interchange (EDI), RosettaNet, HL7, and other standard interchange formats. It also includes services for managing interactions with trading partners.

---

<sup>1</sup> Despite its name, SQL Service Broker doesn't provide a "broker" in this sense of the term. In the terminology of SSB, a broker is a message queue.



BizTalk Server 2006 is also a business process server. Viewed from this perspective, it provides features such as:

Support for graphical definition of business processes using an Orchestration Designer hosted in Visual Studio. A lightweight Visio-hosted Orchestration Designer for Business Analysts is also included.

Business Activity Monitoring (BAM), providing real-time displays of business process information to the information workers that rely on those processes.

A Business Rules Engine (BRE), letting complex business rules be defined, accessed, and maintained in a single place.

#### When to Use BTS 2006

The primary integration scenarios for BTS 2006 are:

Creating brokered application-to-application message-based integration, especially when data mapping and support for diverse communication mechanisms is required.

Implementing integration processes, including long-running processes that take hours, days, or weeks to complete, and processes with complex business rules.

Addressing B2B integration, including situations with many trading partner interactions and those that require industry standards such as RosettaNet and HL7.

Creating business processes that give information workers real-time visibility into an integrated process.

Using a brokered approach can add cost to an integration solution. Still, for scenarios such as those just listed, the additional overhead of a broker is more than made up for by the

value it provides. In these situations, BTS 2006 is the best approach to integrating diverse applications.

### BTS 2006 and Other Integration Technologies

BTS 2006 can work with many other integration technologies from Microsoft and other vendors. For example, as mentioned earlier, adapters are available for MSMQ and Indigo, as well as adapters for non-Microsoft integration technologies such as IBM's WebSphere MQ. BTS 2006 also supports integration using web services, including a SOAP adapter and the ability to import and export definitions created using the Business Process Execution Language (BPEL).

It's also possible to create custom adapters. For example, an organization could build a custom adapter based on components in HIS 2004 that connects to CICS and DB2. (These adapters will be included as a standard part of HIS 2006, the product's next release.) And while an adapter for SSB is likely to be available in the future, a custom adapter could be created today.

One of the main benefits of a brokered approach to integration is the ability to communicate with a diverse set of applications. Given this, it shouldn't be surprising that BTS 2006 is able to use virtually any integration technology that provides direct application-to-application communication.

## Integrating Data

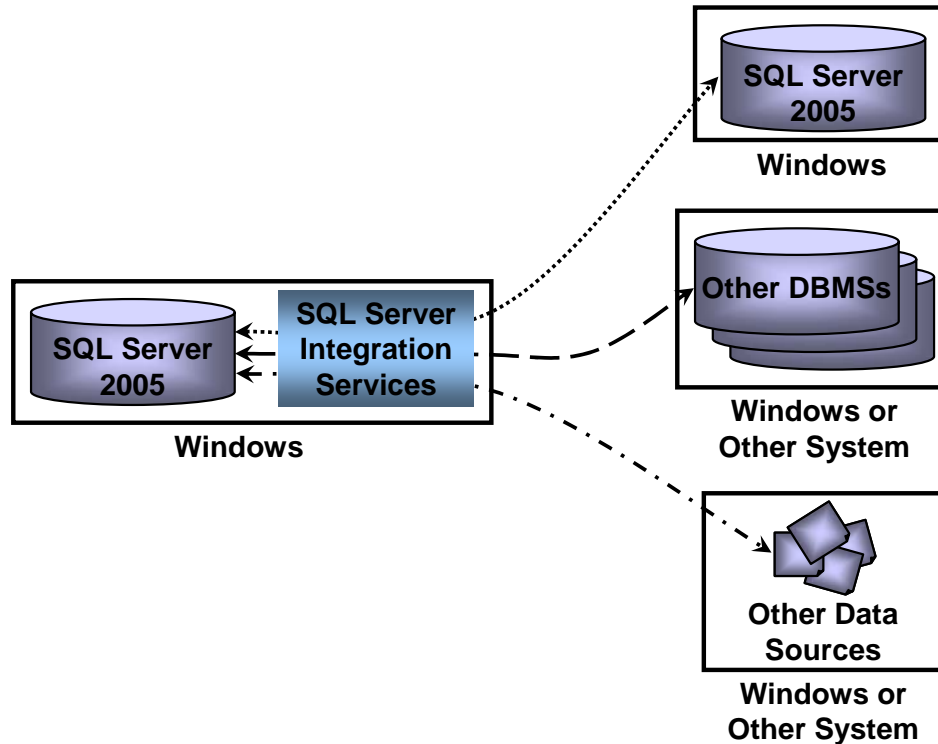
Integrating applications requires connecting active logic. Integrating data, by contrast, means moving and manipulating passive information. While the software that integrates data from different sources is not passive, the data itself—the thing that's actually being integrated—doesn't have any intelligence of its own. Because of this, data integration happens between data stores, not applications. Accordingly, both of Microsoft's technologies for data integration are associated with SQL Server, its flagship product for data management.

### SQL Server Integration Services

SQL Server Integration Services (SSIS) provides tools for combining data from diverse data sources into a SQL Server 2005 database. The successor to SQL Server 2000's Data Transformation Services (DTS), SSIS is the extract, transform, and load (ETL), service for SQL Server 2005. SSIS is included as part of SQL Server 2005 Standard Edition, with some advanced components for data cleansing and text mining added in the Enterprise Edition.

#### Describing SSIS

It's become increasingly common for organizations to create databases that contain large amounts of historical data obtained from a group of operational databases. Creating this kind of data warehouse implies integrating a broad and diverse set of information. Yet doing this effectively also requires making the information consistent, coherent, and comprehensible. A primary goal of SSIS is to make this possible.



The figure above shows how SSIS is typically used. Data from various DBMSs, including SQL Server 2005 and others, can be combined with data from other sources, such as semi-structured data and binary files. The result, stored in SQL Server 2005, can then be used as a foundation for historical reporting, data mining, and online analytical processing (OLAP).

#### When to Use SSIS

The primary integration scenarios for SSIS are:

Combining information from a group of operational databases into a data warehouse. Along with powerful support for data transformations, SSIS provides graphical tools for defining the ETL process, fuzzy logic for data cleansing, error handling, and other features to make integration of diverse data easier.

Transferring data from one DBMS to one or more other DBMSs. Because SSIS supports heterogeneous data sources, the products involved might or might not be SQL Server 2005.

Loading data into SQL Server databases from flat files, spreadsheets, and other diverse data sources.

#### SSIS and Other Integration Technologies

Like other Microsoft integration technologies, SSIS relies on the data access components included in HIS 2004 for access to data on IBM mainframe and midrange systems. And whatever systems are involved, it's conceivable that other integration approaches could be used to solve the core problems that SSIS addresses. For example, it's possible to create an orchestration using BTS 2006 that accesses diverse data sources and builds a common integrated database from a variety of data sources. In nearly every case, however, SSIS is a better choice for solving this problem. BTS 2006 focuses on

integrating application logic, not data, and so it's better suited to real time communication of information between applications and among trading partners. SSIS, by contrast, is optimized for bulk data loading from diverse data sources. Given this, SSIS is a much better choice for creating data warehouses than the application-oriented BTS 2006. (In fact, BTS 2006's Business Activity Monitoring component uses SSIS to build the BAM data warehouse.)

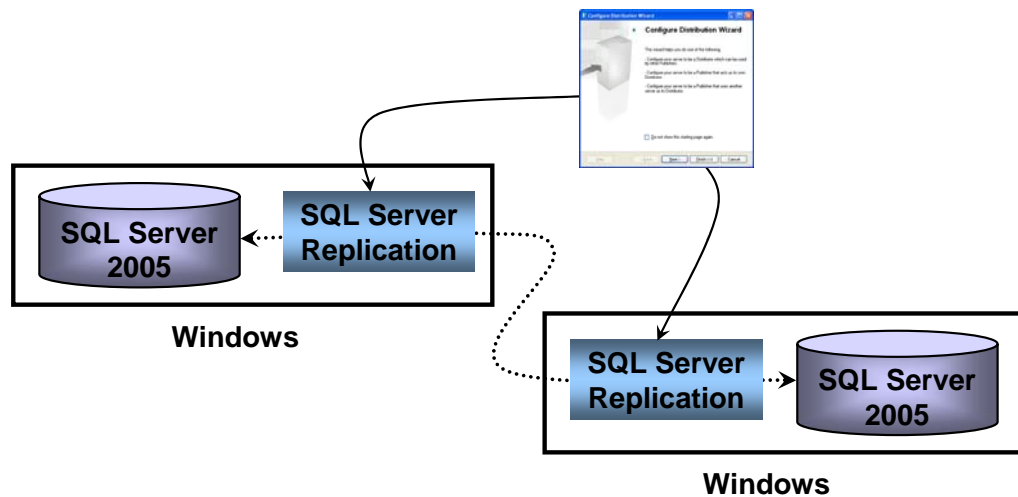
SSIS can also be used to replicate identical data across different systems rather than to combine diverse data. As described next, however, SQL Server Replication is usually a better solution for this problem.

## SQL Server Replication

As its name suggests, SQL Server Replication allows replicating data across two or more SQL Server databases. Available with SQL Server 2000, this technology is also contained in all versions of SQL Server 2005 and in SQL Server CE.

### Describing SQL Server Replication

It's often useful to have a copy of the same data in multiple databases, then have that data automatically kept in sync. For example, letting applications running on a group of web servers spread their read requests across a group of identical databases, each on its own machine, can improve the application's scalability and availability. For this arrangement to work, all updates must go to a single database instance, then be propagated to the read-only copies. SQL Server Replication is designed for situations like this.



The figure above shows a simple illustration of SQL Server Replication. As the diagram suggests, one of the most important benefits of this technology is a powerful user interface that lets database administrators easily define what data should be replicated, see differences between tables, and more. A replication conflict viewer is also provided, allowing data conflicts that occur during replication to be resolved.

### When to Use SQL Server Replication

The primary integration scenarios for SQL Server Replication are:

Replicating data between tables in one or more SQL Server instances. Those instances might be running on servers, clients, or even mobile devices that are only occasionally connected. Rather than copying entire tables, SQL Server Replication replicates incremental row-level changes, letting updates be propagated at near real-time speeds.

Using SQL Server as a source for data that is replicated to IBM and Oracle databases.

Using Oracle as a source for data that is replicated to SQL Server, IBM, and Oracle databases. In this case, data is first replicated to a SQL Server database, then replicated to the other databases.

Data replication is useful in a variety of different scenarios, and so SQL Server Replication can be applied in numerous ways. In fact, even though this technology can be correctly seen as a tool for integration, it's just as accurate to view SQL Server Replication as a solution for data synchronization.

### SQL Server Replication and Other Integration Technologies

If synchronization of identical data sources in near real time is required, SQL Server Replication is the best choice. No other integration technology provides support for rapidly tracking and delivering changes as they occur. While it's possible to write a custom replication application using BTS 2006 or other integration technologies, it's challenging to do correctly. The result also wouldn't have the performance or the features that are included in SQL Server Replication.

SQL Server Replication replicates data identically using either all columns of the selected tables or a subset of the table columns, while maintaining the data in the same format. SSIS has the additional capability to transform the data as it is moved, and so it's a better choice for copying entire tables that have different structures. Yet unlike SQL Server Replication, SSIS doesn't support replicating incremental row-level changes—it copies the whole table—which means SSIS typically has lower performance. As an ETL tool, SSIS focuses on integrating diverse data rather than replicating identical data, the problem for which SQL Server Replication was designed.

## Conclusion

Different integration problems require different solutions, and it's important to use the right tool for the job. To address the diversity of applications and data that must be connected, Microsoft has produced a range of integration products and technologies. These solutions sometimes overlap, and so more than one choice might be applicable to a given situation. While these ambiguous cases are relatively uncommon, the most straightforward way to make a decision is by examining the fundamental scenarios for each integration technology.

The goal of integration is to connect diverse pieces of software into a coherent whole. If the tools for doing this are themselves built in diverse ways, such as when products built by several different firms are grouped together under a single marketing umbrella, reaching this goal becomes even more difficult. By providing a unified set of solutions, all built on the .NET Framework and all accessible using Visual Studio 2005, Microsoft provides a comprehensive, unified, and complete solution for today's integration challenges.

## Appendix: Summarizing Microsoft Integration Technologies

<b>Integration Style</b>	<b>Technology</b>	<b>Primary Integration Scenarios</b>
<i><b>Integrating applications directly</b></i>	<b>ASP.NET Web Services (ASMX)</b>	Connecting Windows applications with Windows and non-Windows applications via SOAP
	<b>.NET Remoting</b>	Connecting Windows applications with other Windows applications via distributed objects
	<b>Enterprise Services</b>	Connecting Windows applications with other Windows applications that use distributed transactions, object lifetime management, etc.
	<b>Indigo</b>	Connecting Windows applications with Windows and non-Windows applications using web services, distributed transactions, lifetime management, etc. (subsumes ASMX, .NET Remoting and Enterprise Services)
<i><b>Integrating applications through queues</b></i>	<b>Microsoft Message Queuing</b>	Connecting Windows applications with other Windows applications using queued messaging
	<b>SQL Service Broker</b>	Connecting SQL Server 2005 applications with other SQL Server 2005 applications using queued messaging
	<b>Indigo</b>	Connecting Windows applications with other Windows applications using queued messaging (via MSMQ and/or SSB)
<i><b>Integrating with applications and data on IBM systems</b></i>	<b>Host Integration Server 2004</b>	Connecting Windows applications with IBM zSeries and iSeries applications and data Connecting MSMQ with IBM WebSphere MQ
<i><b>Integrating applications through a broker</b></i>	<b>BizTalk Server 2006</b>	Connecting Windows applications and non-Windows applications using diverse protocols Translating between different message formats Controlling business processes with graphically-defined orchestrations Connecting with business partners using industry standards, such as RosettaNet and HL7 Providing business process services, such as Business Activity Monitoring and a Business Rules Engine
<i><b>Integrating data</b></i>	<b>SQL Server Integration Services</b>	Combining and transforming data from diverse sources into SQL Server 2005 data
	<b>SQL Server Replication</b>	Synchronizing SQL Server data with copies of that data in other instances of SQL Server, Oracle, or DB2