Retek® 10.2 Integration Bus

Primer

The software described in this documentation is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

No part of this documentation may be reproduced or transmitted in any form or by any means without the express written permission of Retek Inc., Retek on the Mall, 950 Nicollet Mall, Minneapolis, MN 55403.

Information in this documentation is subject to change without notice.

Retek provides product documentation in a read-only-format to ensure content integrity. Retek Customer Support cannot support documentation that has been changed without Retek authorization.

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403

888.61.RETEK (toll free US)
+1 612 587 5000

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

Sales Enquiries:
+44 (0)20 7563 46 46
Fax: +44 (0)20 7563 46 10

Printed in the United States of America.

Customer Support

Customer Support hours:

Customer Support is available 7x24x365 via e-mail, phone and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance.

Contact Method	Contact Information
Internet (ROCS)	www.retek.com/support Retek's secure client Web site to update and view issues
E-mail	support@retek.com
Phone	US & Canada: 1-800-61-RETEK (1-800-617-3835) World: +1 612-587-5800 EMEA: 011 44 1223 703 444 Asia Pacific: 61 425 792 927
Mail	Retek Customer Support Retek on the Mall 950 Nicollet Mall Minneapolis, MN 55403

When contacting Customer Support, please provide:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step by step instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

Contents

Chapter 1 – Introduction

This guide is designed to introduce the reader to basic Enterprise Application Integration (EAI) concepts in general and to the Retek Integration Bus (RIB). It provides an overview of the general nature of an EAI solution and how the RIB fits into such a solution.

Chapter 2 – Enterprise Application Integration

This chapter introduces the reader to the concepts involved in Enterprise Application Integration (EAI). It describes, in general terms, what an Integration Bus is and what various EAI concepts and entities are.

EAI systems are designed to integrate disparate, dissimilar applications in a loosely coupled manner. These applications may:

- Be "Legacy" applications, designed to accomplish a specific task and requiring data input from specific channels.
- Be hosted on a diverse set of operating system platforms.
- Have their own set of Graphical User Interfaces, some of which may be antiquated in look and feel.
- Consist of isolated, standalone functionality.

Furthermore, these applications usually communicate via "point-to-point" techniques. For example, a job creates a data file from its private database and uses FTP to send a file to another system, where another batch job processes the data. Additional batch jobs are created for all other applications requiring any type of data feed from the original repository.

At first glance, point-to-point integration appears to be a simple process to implement, understand and support. Unfortunately, it has some inherent problems: lack of scalability, and latency.

Lack of scalability: It is not unusual for an enterprise to have more than 50 separate applications involved in its day-to-day business activities, and this number continually grows. As more applications become involved, the number of batch jobs increases exponentially in a point-to-point solution, blossoming into an operational headache – the probability of any batch job failing increases as the number of jobs increases. Furthermore, operations staff has reduced flexibility to perform basic tasks without affecting one or more business processes. Bringing a system down for a straightforward preventive maintenance or back up may require significant coordination between multiple batch job streams. Moreover, each application could require its own idiosyncratic data formats and/or code values, complicating the development and support of the file creation or processing program.

Latency: Besides operational concerns, the reliance on batch processing also introduces latency concerns – users want to see the latest status as of *now*, not what happened yesterday or last week. Decisions can dramatically affect the bottom line if the information they are based on is provided in a timely and accurate manner.

What is an Integration Bus?

Although there are many vendors of competing EAI solutions, they all support a core set of concepts and abstractions. One fundamental notion is that an *"Integration Bus"* carries messages between applications. The integration bus eliminates the point-to-point nature of current integration techniques. Instead of connecting each application to each other separately, each application "plugs into" a common bus. This idea is similar to how a computer uses an electronic bus for communication between a CPU, memory, disk drives, modem, keyboard, monitor, and other peripherals. Just as each piece of hardware uses a known pin-setting on its interface card to plug into a specific computer bus, so too does each application uses a specific API to plug and play with an integration bus.

The major difference between a hardware bus and an integration bus is that the operation of a data channel on a hardware bus is typically dedicated between two components, such as the CPU and memory or between memory and disk drive controller. On an integration bus, data channels may be between one source and multiple destinations, such as between an order management application and warehouse, finance, merchandizing, and marketing systems.

Once a bus exists, there needs to be something to plug into it. Although many vendors have their own names for these components, the general term *"Adapter"* describes a software module that communicates to and from an integration bus on behalf of an application. Each integration bus vendor may support many types of adapters. In addition, vendors have available a suite of adapters that connect commonly found applications into their Integration Bus. Most vendors offer a suite of specific adapters for PeopleSoft, SAP, and Oracle Financials applications.

Vendors allow a developer to extend some kind of generalized adapter and create software for integrating a customer-specific application. Many types of "generalized" adapters exist, depending on the EAI vendor. Examples include:

- Database adapters that create messages based on an SQL select statement, or process messages by extracting data and using the values within an SQL update, insert, or delete statement.
- Adapters specialized to link into a specified library or DLL. Messages are created or processed based on function calls into the library or DLL. Adapters such are these are usually specific to the programming language used, such as Perl, C, C++, or Java.
- Adapters specialized to leverage a component object model interface such as CORBA, COM, or DCOM.
- Adapters designed to interface with World Wide Web interfaces.
- File-based adapters that create files from messages subscribed on the bus or create messages based on the presence of a file. These adapters are used for integrating an existing batch-based solution with the integration bus.

SeeBeyond Technology Corporation, the EAI vendor chosen by Retek to host the Retek Integration Bus, has developed over 50 application specific adapters for their e*Gate Integrator™ EAI Integration Bus. Furthermore, there is a myriad of libraries, tools and utilities available to help develop additional application adapters.

There are two additional concepts that all EAI vendors support: A *publish and subscribe (pub/sub) messaging paradigm* with some *guarantee of delivery* for a message. In a pub/sub messaging system, an adapter publishes a message to the integration bus that is then forwarded to one or more subscribers.

Note: In much of the literature surrounding the Publish and Subscribe paradigm, the term "Event" is used instead of "Message". However, the semantic difference between the two terms for this discussion is inconsequential.

The publishing adapter does not know, nor care, how many subscribers are waiting for the message, what types of adapters the subscribers are, what the subscribers' current states are (running/down), or where the subscribers are located. Delivering the message to all subscribing adapters is the responsibility of the Integration Bus. The physical means to accomplish this is EAI vendor-specific and may involve multiple programs, network protocols, or adapter libraries.

Message subscribers subscribe to the combination of the message type and the message publisher. This is a "logical" subscription. Physically, the message must reside somewhere so that it is available until all subscribers have processed it. This typically involves the use of a *Message Queue* attached to the integration bus. The message queue accepts the message from the publisher and saves it to stable storage until it is ready to be picked up by a subscriber. In one sense, a message queue is just a special type of EAI adapter whose purpose is to save and forward messages. However, because of their importance, application independence, and transparency to both publishers and subscribers, many EAI vendors treat message queues as a completely separate kind of entity.

In most cases, message information must be kept on the message queue until all subscribers have read and processed it. However, there may be some messages that have a limited amount of time associated with their usefulness. For example, take the case where a stock ticker application publishes thousands of stock quotations throughout the day and has hundreds of subscribers. The subscribing applications are all desktop PCs. If an employee takes a vacation and turns off his/her desktop PC, it makes no sense for the message queue to hold old stock quotes just for this one employee. When the employee comes back from vacation, he/she will only probably only be interested in the current quoted values, not those from last week. For these types of messages, it makes sense for the message queue to delete them after a set amount of time.

Finally, each message found on an integration bus has an associated "type" that is separate from the message's data structure. Multiple message types may share the same message data structure or definition. Most EAI systems allow completely unstructured messages – typically defined as an array of bytes – and have length limitations that are bounded only by the available amount of disk space to store the message on the message queue. Other message types are much more structured: Examples here include:

- XML messages that have their data structure defined by Document Type Definitions (DTDs) or XML schema documents.
- Messages containing a set of comma delimited fields using an EBCDIC, ASCII, Unicode, or UTF 8 encoding.
- A set of header/detail records, where the record type is determined in the first character of the record.
- EDI messages defined by a standard such as UN/EDIFACT, where each message is a single, very long line.

In some EAI messaging solutions, there exists only an implicit association of the *message type to* the internal data structure of the message. These solutions have no explicit enforcement of the message structure – it is all up to the application to create and validate contents. However, EAI systems with centralized message definition facilities, such as SeeBeyond's e*Gate Integrator™, can validate the structure of the message outside of application code.

Pub/Sub messaging

It is important to understand Pub/Sub messaging paradigm, since it is integral to the EAI framework. Although other methods can be used to eliminate point-to-point integration, the Pub/Sub architecture has been found to be the most efficient. Elimination of point-to-point messaging provides the basis for loosely coupling the applications that are plugged into the integration bus. Almost all messages carried by an integration bus will be created and consumed using Pub/Sub techniques.

In the Pub/Sub paradigm, the terms *message* and *event* are often used synonymously. However, there is a subtle semantic difference: an *event* is an identification of something that has occurred, while a *message* contains the information about the event. Events are distinguished by their "type". For example, a system may have a "sales order" event type or a "cancel order" event type defined. Whenever a "sales order" occurs, a "sales order event" is generated. Similarly, all messages are categorized by event type, and by a format defined for this type. When a publisher publishes a message, the message content must conform to the message type definition. Note, however, that multiple event types may share the same message format.

A simple pub/sub message system is shown in Figures 1a and 1b. A logical representation is found in Figure 1a, while a corresponding physical representation is found in Figure 1b. In the diagrams, the publisher publishes a single “New Order” event whenever a new order occurs. The order management application is the publisher, with the actual message creation performed by an associated EAI adapter. The subscribers, marketing and an order fulfillment application, simply consume the message. For the sake of simplicity, the connection mechanism between the adapters and their associated applications is ignored.

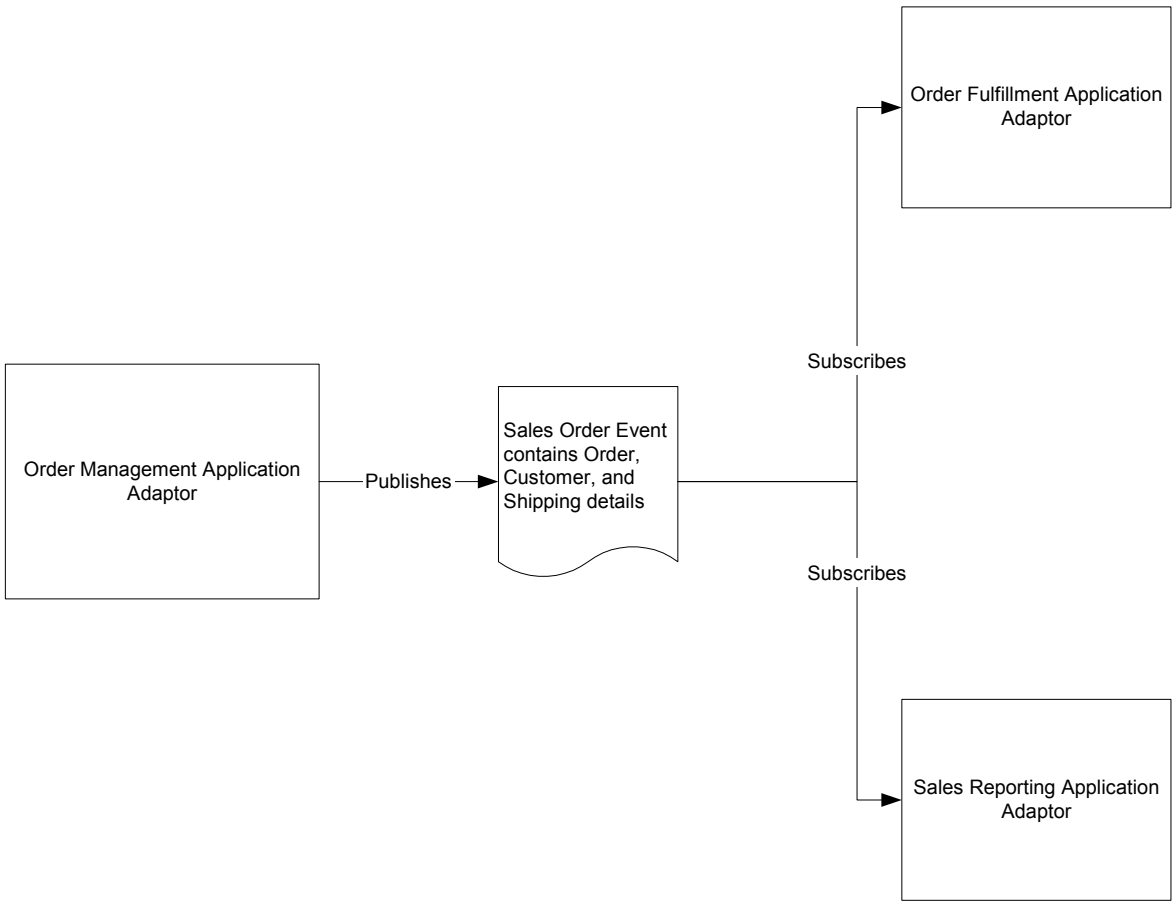

Figure 1a -- a simple Publish and Subscribe Logical Model

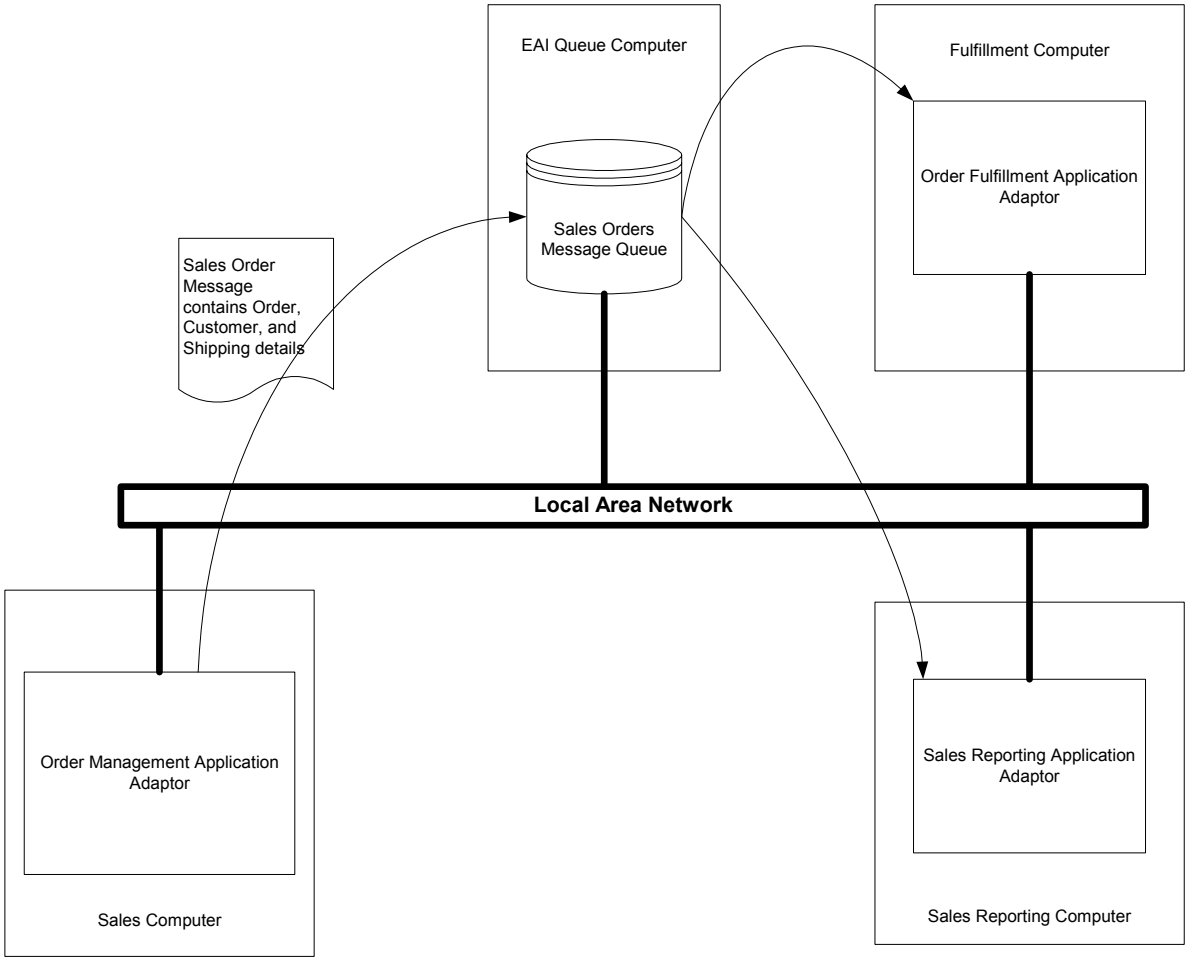

Figure 1b -- a simple Publish and Subscribe Physical Model

After a message has been delivered, the subscriber may perform any number of actions based upon the message contents and their internal logic. It may perform relatively simple actions such as inserting a single row into a database or appending the message to a file. Complex message processing is also common.

One fundamental capability of message subscribers is that they can also publish messages. For example, a message subscriber may make calls to a database to flush out missing information and then publish the new message. Subscribers may change externally created data into a canonical form before publishing by translating foreign code values or re-formatting the record's content. The idea is to form a pipeline of publishers and subscribers, where each subscriber performs a single, well defined action upon the message stream.

Deployment of EAI systems can become complicated in design. Many times the EAI framework offers a large number of deployment configuration options. These options may support a fine granularity of control over the message processing logic, enable flexibility in where components are located, increase performance, or exist for high availability considerations. Some of these options are:

- Subscribing adapters can subscribe to specific event types produced by specific publishers.
- Subscribing adapters can subscribe to multiple event types, each with their own processing logic. Subscribing adapters can thus coalesce multiple messages, if needed.
- A single publisher can publish multiple events of the same type to different message queues.
- Multiple publishers can publish to the same message queue the same or different message types.
- Message queues can be distributed on multiple hosts.
- Multiple copies of an adapter can exist for increased message bandwidth or high availability considerations.
- The same message type may be stored on multiple queues, or multiple message types may be stored within a single queue.

Request/reply messaging

Another message paradigm that EAI systems support is “Request/Reply”, sometimes also known as “Call/Response”. This paradigm is common in On Line Transaction Processing (OLTP) n-tier client/server systems or Remote Procedure Call (RPC) distributed processing frameworks. A message is created that encapsulates a request for some service to be performed. The requestor then waits for the service provider to satisfy the request and to return any computed data or status information.

Request/Reply messaging is appropriate when there are reasons *not* to publish events occurring within an application. One reason would be to maintain security or to control access to the data. Another may be that volume of the data published would be very large, while the subscribers would only use a small number of records – it would not make sense for the subscribers to allocate the resources to maintain a large collection of database tables for a few records that are infrequently accessed.

In some aspects, Request/Reply is an extension and specialization of Pub/Sub. It can be seen as two cooperating adapters that flip-flop as publisher/subscriber. The first publisher publishes a “Service Request” event that the second adapter subscribes to. Then the second adapter processes the request and publishes a “Service Reply” event that contains all of the desired information. The first adapter is the only subscriber to this event.

However, Request/Reply messaging adds additional complexity. Service requests may be made synchronously (the service requestor waits for the reply before continuing) or asynchronously (the service requestor may spawn off another thread to process the reply or may perform other actions while the request is processed). In the asynchronous case, the requestor needs to be able to track which replies have been received and to match the reply to the original request. In the synchronous case, decisions need to be made regarding how long database locks or other computing resources should be held and what processing should occur when a request “takes too long”.

Chapter 3 – The Retek Integration Bus

This chapter answers basic questions about the Retek Integration Bus and Retek's EAI solution.

What is the Retek Integration Bus?

The Retek Integration Bus (RIB) is Retek's application integration message processing for its retail application suite. Using SeeBeyond Technology Corporation's e*Gate Integrator™ EAI platform as a base, Retek has developed a set of adapters and messages that allow an enterprise to integrate both Retek and non-Retek applications together in a cohesive, scalable, and supportable fashion.

The actual software is delivered on a series of CDROM disks. One set of disks contains the e*Gate Integrator™ core EAI product, e*Gate Application e*Way™ Intelligent Adapters, and the eInsight™ Business Process Manager from SeeBeyond. Another disk will contain the adapters and sample deployment system (or schema) providing integration points with RMS, RDM, RDW and RCOM applications.

Why was the Retek Integration Bus Developed?

Prior to Retek 10, Retek's integration approach was a tightly integrated point-to-point solution. It relied heavily on file-based interfaces. Issues with this approach arose due to:

- The inherent complexity in data flow and data mapping between numerous integration points.
- The "ripple effect" that changes to applications can have with other applications. This makes future releases more difficult to develop and test. Additionally, the absence of a standard messaging definition forces third party applications to be "hardwired" directly to Retek database tables or internal/private application APIs.
- The dependence on nightly batch jobs prevented a broader support of near real-time communication.

Based on reference visits with retailers and talking with lead vendors in the EAI space, Retek recognized that many retailers were investing in EAI solutions. Retek realized that this trend was due to the benefits that EAI solutions provide.

Direct benefits from an EAI solution include:

- Reduced maintenance/upgrade overhead that is a result of replacing an enterprise's multitude of point-to-point interfaces with an integration bus solution.
- Increased integration points: EAI solutions decrease the cost it takes to integrate a single application to the rest of the enterprise. In point-to-point integration systems, many applications are not integrated due to the cost of integrating just that one system due to a lack of integration infrastructure or framework. For example, a system may not have an adequate batch scheduler, the file formats used by other applications may be unusable, or the file transfer tools may not be available on a specific platform. Hence, a significant amount of development would be needed to integrate the system unless an EAI platform was available.
- Increased visibility to the business process as engineered throughout a retailer's systems: In many cases integrating systems has left the business process in a confused state due to inconsistent integration point selection (some applications are integrated, some are not) and differing integration latencies (some applications have latencies measured in minutes between updates, while other applications wait for days for the same data). Moreover, there is typically no way to verify that the desired business model has indeed been implemented. EAI tools allow retailers to model some key business processes and to impose this model upon their applications.
- Improved integration with disparate, "one-off" systems: Many enterprises have implemented so-called "one-off" applications used for a very specific purpose by a select group of individuals within the corporation. These rogue processes or applications are difficult to incorporate into an overall systems architecture. They can also be a constant unbudgeted drain on application development resources. An EAI solution can provide additional visibility to the applications while at the same time providing a path to assimilate these applications within the enterprises standard architecture.

Indirect benefits can also be realized, depending on how the EAI solution is implemented, such as:

- Near real-time daily sales analysis – if sales are processed through a sales audit system and then published, subscribers can analyze the data throughout the day instead of the next day or, in some cases, the next week.
- Near real-time inventory visibility – for on-line markets (for example, Web stores) a true distribution center inventory representation is essential to avoid lost sales and false client commitments. An EAI solution can expose the inventory position to other applications and make this more easily accessible. Additionally, EAI systems architecture can allow other applications to post inventory changes on a near real-time basis, a requirement for accurate inventory values.

- Improved in-stock position derived from more accurate inventory values. Replenishment activities can be performed more frequently than once per day because the inventory counts are expected to be as accurate as possible during the day. Systems no longer require lengthy batch updates to refresh the inventory position before the replenishment is run.

These benefits are a strong argument for a retailer to implement an EAI system and some retailers already have. Since integration is becoming a key concern for retailers, Retek has realized that an EAI implementation is strategically necessary for Retek to maintain its technological lead. The desire is to provide a comprehensive approach to integration throughout the Retek application suite and merely continuing to provide point-to-point solutions is not a viable means to this end. Furthermore, Retek believes that additional benefits can be achieved through its RIB implementation.

Additional RIB Benefits include:

- Easy to implement alternatives to batch FTP jobs:
  - Option 1) An entire file may be published as a single message. Subscribing applications pick up the data only when they are available to handle it.
  - Option 2) The publisher can break up a file on a record-by-record basis, each record its own distinct message.

    Advantages: The data processing can be quickly interrupted between messages, additional data processing resources (adapters) can be brought on-line dynamically, and the processing application does not require explicitly coded restart/recovery mechanisms.

    One may also combine the two methods: a file can be published as a single entity and then broken up record-by-record by a subscribing adapter. This combination enables a phased implementation migrating off of a batch to a message based system: some subscribers can use the original file as one message, while others subscribe to the "splitter" adapter's messages to process each record separately.
  - Option 3) A transfer may be initiated via an XML message that specifies the source data file(s) and destination(s) to send the file to.
- Reduced dependence on RMS as the integration 'hub', reducing the development time needed for satisfying integration needs for other applications. It should be easier to integrate third-party and legacy applications.
- Enables positioning the applications for a serviced based implementation by allowing a phased transition from current batch processes to new message based implementations.
- Improves the data management of Retek applications by reducing the volume of redundant data transfers and transformations.
- Enable easier information transfer to data intensive applications. A simpler integration is achieved via a standardized message format.

- Improves application monitoring capabilities and system management tool integration via SeeBeyond e*Gate Integrator™ alert mechanisms.

What is the RIB-ETL?

The RIB-ETL is the former name for the Retek Extract, Transform and Load (RETL) product. RETL is a high volume data transfer solution and high performance framework leveraging parallel processing for large data extractions and loads. It may be used to regularly feed a data warehouse application, such as the Retek Data Warehouse (RDW) or to provide a one-time data synchronization between databases. RETL uses a data feed directly to or from a file or database(s). It is not a messaging solution per se, and is dependant upon an external scheduling mechanism.

Do I have to use the RIB?

RIB usage is not mandatory for data synchronization. Although the RIB is the recommended method of integration between multiple Retek Applications, most Retek applications are designed to be loosely coupled with the RIB and can operate completely separate from the RIB. Some business functions do not need real-time information, nor does it make sense to provide it. One may choose to develop other methods for integrating an enterprise's applications.

However, it should be noted that the RIB has been thoroughly tested for inter-Retek application integration. Using the RIB for integrating external applications with the Retek Applications such as RMS can be done with a great deal less development and reduced data latency, all in a loosely coupled fashion. One can eliminate production Batch jobs and their associated "Batch Windows" dedicated for running these jobs via a message based integration framework.

RIB usage may be mandatory for those Retek applications requesting remote services to be executed in a synchronous fashion. That is, a request/reply operation is needed between an application and a service provider whereby the application must wait for the reply from the service before continuing. In these cases, the client application may assume that an operational RIB is present.

What are my alternatives to the RIB?

The first option is to use existing batch integration techniques. All of Retek's products can still use batch methods for integration with external applications. The modifications, if any, needed for existing batch jobs to operate in the RIB-enabled release may be small enough to justify this approach. Additionally, the RIB leverages many Oracle stored procedures that are part of each product (RMS, RCOM,). These stored procedures may be called by homegrown applications as well. The stored procedures' APIs and message structure they process are documented in the "Retek Integration Guide".

If another EAI solution is already in place, one has a choice of options integrating with the RIB:

1. Leverage the stored procedures mentioned earlier as part of the existing EAI implementation. This involves developing adapters to interface with these stored procedures and to perform the translation and transformation of the RIB messages to and from the message structures currently in use.
2. Replace the existing EAI implementation with the RIB. This option should be weighed in light of the amount of integration already performed, licensing costs and development costs.
3. Create a bridge adapter between the RIB and the EAI implementation already in place. This will probably have a reduced amount of implementation associated with it. It should be considered when the applications to be integrated are running in separate administrative domains. However, it has the disadvantage of requiring two EAI solutions to be running concurrently. This is also a pragmatic approach to phased implementation of either of the other options mentioned above.

What is NOT part of the RIB?

The Retek Integration Bus provides the basic components for integrating Retek applications. These components contain the intra-Retek application message processing logic. In other words, the "out of the box" RIB is a set of components that need to be configured and tailored to a retailer's specific needs.

The types of development that will be associated with the RIB include:

- Developing adapters to integrate external applications with the RIB. This effort can leverage the 50+ e*Gate Application e*Way™ intelligent adapters already developed by SeeBeyond Technology Corporation.
- Architecting the RIB for performance/bandwidth, and high availability. This includes determining the hardware and the RIB deployment topology to be used.
- Configuring or adapting alert monitors to integrate the RIB with existing SNMP, logging, e-mail, or other systems management infrastructure.
- Creating operational support scripts for such duties as archiving or deleting old log and data files.

How do I extend the RIB for my own needs and processes?

The appropriate techniques used to extend the RIB are dependent on the specifics of the problem to be solved. However, the preferred method is to create a separate adapter that is either an additional publisher to or subscriber of one or more RIB messages. This strategy uses the RIB infrastructure to maintain a loose coupling of the external application with the Retek applications and allows these extensions to be monitored or modified without affecting the core Retek software.

SeeBeyond's e*Gate Application e*Way™ Intelligent adapters is a suite of over 50 application-specific adapters. A client may select two of as part of the RIB software, enabling a shortened development to known applications such as PeopleSoft, SAP, or Oracle Financials. Additionally, the core e*Gate Integrator™ product includes generic adapters that can call COM/DCOM, Java, Perl, C and C++ application APIs. The purpose of these adapters is to integrate with the messaging system for a specific application API. Input and output parameters to the API will be derived from or inserted into messages on the integration bus.

There may be cases where additional extensions or modifications are needed within the same database transaction that processes specific message. This is typically needed for site-specific enhancements, custom code, or referential integrity issues. In this scenario, modifications must be made to the originally supplied RIB adapter software. The specific modifications may be made in a number of places: within a database trigger, a stored procedure, or within the "Collaboration" that controls the interface to the SeeBeyond e*Gate Integrator™ platform. The specific changes are tightly coupled with the nature of the change, its risks, and considerations such as the work involved in future product updates. As in all major Retek products, all source code is available for the RIB components developed by Retek.

What do I need in order to use the RIB?

The base SeeBeyond e*Gate Integrator™ technology is supplied as part of the RIB, and no separate licensing is needed for this product component. Additional SeeBeyond components may be purchased directly from SeeBeyond. The RIB uses a standard release of e*Gate Integrator™ from SeeBeyond, and any additional SeeBeyond components can be purchased that are compatible with this release.

Integration with Retek applications can begin with the installation of the application. The RIB will be included with the purchase of the RMS, RDM, or RCOM products.

In terms of hardware and OS needs, these are derived from the specific EAI performance and high availability requirements. One or more Microsoft Windows (NT or 2000) hosts need to be available for running SeeBeyond's EAI monitoring tool. The production RIB components can be run on the Microsoft Windows, Linux, and Unix families of operating systems. The list of supported platforms may be found on SeeBeyond's Web site, http://www.seebeyond.com. Additional drivers and database requirements are also needed for the Retek 10 product suite. This includes the Oracle 9i database release.

Chapter 4 – ROI calculations for EAI

Enterprise Application Integration (EAI) is a strategic investment that can be very expensive. Naturally, a CIO or other top executive needs to support any decision regarding this technology with sound reasoning.

Is ROI relevant?

Investments in new software and technologies are made for one of two reasons: The first is that the resulting gain in efficiencies or capabilities will pay for themselves in a reasonable amount of time. The decision to implement or not is based purely on calculated savings or increased sales. The return on investment is many times confidently computed because the benefits are incremental in nature. This reasoning includes computations such as "The enterprise has X projects planned and this project or software will decrease the implementation time Y FTE days" or "The project will result in a rise in same day shipments by X percent which will result in a decrease in order cancellations by Y percent."

The second type of logic supporting a decision of this sort is based on survival and fear: by not investing in this functionality, the door is opened for the competition to gain such an advantage as to severely damage market share and profits. The benefits to the business processes provide a quantum leap in efficiencies. Examples of these include the adoption of double entry accounting, steam versus sail powered steamships, "Hub and Spoke" airline route systems, and Point of Sale systems. The question is not whether to adopt the new technology, but which implementation to adopt.

An additional question is: "When should this technology be adopted?" This involves an analysis of the costs involved now versus later, such as:

- Re-working the new interfaces added during the delay
- Loss of efficiencies by not using the new technology
- Loss of revenue due to current inefficiencies

versus other factors, such as:

- The cost of delaying other projects due to resource availability while implementing the technology immediately
- The maturity of the technology

EAI sales personnel believe that their technology provides this "quantum leap" advantage. Where would a customer rather shop: at a store that *might* have the right size, color, and selection or one that *always* has it? EAI provides the infrastructure for the correct stocking decisions because the information on sales and inventory positions is available quickly and accurately. New sales support, forecasting, warehousing, … systems can now be deployed quickly because of decreased implementation time. Pilot projects can be implemented, reviewed, cancelled, or expanded with minimum risk because the results are available in near real time. The cost and time to find the best marketing, sales, distribution, shipping, or merchandizing solution is dramatically reduced because feedback is immediate.

However, the dilemma with any technology becoming recognized as a "must have or die" technology is that usually a few deaths must occur first. On the one hand, if you wait until this is obvious, then your corporation could be among the deceased. On the other hand, the business world is also replete with projects that did not deliver expected results and had to be withdrawn, re-factored, or re-developed immediately after implementation. It is thus prudent to evaluate the expected benefits in terms of dollars and cents. While an EAI vendor is convinced of the technology's cost/benefits, a few members of your corporate board may be less enthusiastic.

What techniques can be used to compute ROI?

Computing ROI is theoretically quite simple. All you need is to compute the sum of expected expenditures to implement the system, and the amount of increased revenue and decreased costs once the system is in place. Compare the two and you are done: it sounds easy.

The problem is that, while the costs many times are simple to find, the benefits are somewhat "soft". Costs include software licensing, training, additional hardware, annual maintenance, and initial development resources. The advantages include savings in time or accuracy of information. How much money is associated with deploying new systems a week sooner? A month sooner? How does one determine the amount of time saved on a per-project basis? How many "good" decisions will be made because of better data? Computing these numbers may be easy for specific projects, but deriving them for an entire enterprise could be considered soothsaying.

Fortunately, there are a couple of ways that some idea of this can be computed. The first method of ROI calculation involves a "macro" approach: Take the total IT budget for the enterprise. Estimate the percentage spent on "integration development". Divide this number by 2 – the driver for this number is that once information is published, only the subscriber needs to be written, resulting in only half of the development needed for new integration. The total calculation results in a per-year savings once the EAI platform is implemented.

An example is as follows: suppose an enterprise has a total IT budget of 50 million dollars ($50,000,000). One study, from the Gartner Group estimates that 40% of an IT budget is spent on integration tasks. Another, from Meridien Research arrives at a 60% figure. Of this, an estimated 30% is for new integration development, while the rest is support, hardware, training, licensing, or other costs. Therefore, the total cost of integration development for this company is:

```
$50,000,000  x  0.40  x  0.30 = $6,000,000
```

Taking half of this results in an expected integration development savings of $3,000,000 per year. This should be compared to an average EAI implementation project costs of about $500,000.

One objection to using the "Macro" approach is based on the estimates of a 50% savings in the integration development work. This is based on the notion that using a batch-based system:

- Once the main data file is created, it can be sent to multiple systems.
- The software used to parse the data file is most likely the same between systems.
- The techniques of transferring data between application systems is a known problem, so development of batch FTP jobs is not large.
- Thus, the total integration new integration development that can be saved is probably only about 25% or less.

Even taken at face value, the ROI looks attractive with a 25% savings. Moreover, one may further argue that these objections are based on groundless assumptions themselves. For example, it assumes that the same data file can be used for all applications requiring data from a common source. Typically, each application has its own data requirements and field semantics, which require unique business logic integrated with an interface file format that is unique to a specific point-to-point integration.

This leads to another risk of point-to-point integration. Unless applications are specifically designed to separate local business logic from the integration point interface, items such as maintenance, support costs, and implementation risks increase every time a new interface is added and/or a business process is modified.

Note: This architecture is normally not implemented because of a variety of reasons. Creating reusable software such as this normally takes three times as much resources to develop and time/resource pressures reduce the likelihood of its adoption. The need for reuse is typically not identified upon the initial integration effort.

Time and resource constraints force developers to choose between modifying existing code and creating new code that is "slightly" different. The former risks breaking software already in production, while the latter presents problems in support and maintenance. A significant amount of analysis or "code reconnaissance" may be needed for even the smallest changes, because *any* change in a database's structure or an application's logic endangers damaging systems *already in production*.

Hence, the real cost of maintaining point-to-point systems is most likely *higher* than 50% of new integration development. This is borne out by the second technique for computing ROI, known as a "Micro" approach. In this approach, all new integration in the next year is identified and classified according to its complexity. The time it takes to design, document, develop, test, and deploy is estimated according to this classification:

- Simple: One source to one destination created with installed, reusable components. 2 – 4 weeks.
- Moderate: One source to one destination created with mostly reusable components, with one or more of these components requiring customization. 4 - 8 weeks.
- Complex: Many sources to one destination with one or more components requiring customization. 2 – 4 months.
- Elaborate: Many sources to one or more destinations with high customization requirements due to unusual communication interface, extensive additional computations, long transaction latencies, incomplete specifications. 5 – 9 months.

These computed base costs apply to point-to-point integration efforts. One then applies the same development savings of 50% to these numbers to reach an expected ROI for a project.

SeeBeyond Technology Corporation has had considerable experience with the ROI involved with EAI implementations using their e*Gate Integrator system. In the white paper, Measuring *the Value of Enterprise Application Integration,* the "Macro" and "Micro" approaches are discussed further. Also shown are the results from four corporations implementation efforts. Using the "Micro" analysis and the actual project implementation:

- Development cost savings ranged from 29% to 89%
- Maintenance costs savings ranged from 91% to 96%
- 3-year ROI ranged from 179% to 449%

Chapter 5 – Additional documentation

Refer to the following documents for more information.

Retek guides and manuals

Use the following resources to further understand the Retek Integration Bus:

Retek Integration Bus Technical Architecture Guide

An overview of the RIB architecture. It provides a description of the SeeBeyond components used and the adapters developed for the Retek 10 integration.

Retek 10.1 Integration Guide

Detailed information about the messages used between applications on the RIB.

Retek Integration Bus Deployment Guide

Deployment considerations, design patterns and strategies.

SeeBeyond Technology Corporation documentation

The following resources should be used for further understanding the Retek Integration Bus and the SeeBeyond e*Gate Integrator EAI platform:

SeeBeyond Business Integration Suite Deployment Guide

Information on how to analyze, plan, and manage an EAI deployment.

SeeBeyond Business Integration Suite Primer

An introduction to all of the available components within the SeeBeyond e*Gate product family. These include e*Ways designed to interface to specific application suites, such as PeopleSoft, SAP, and Oracle Financials.