



business integration
JOURNAL

This article is a PDF version of the one that appeared in a recent issue of *Business Integration Journal*, the resource for BPM, Web services, and EAI.

 **WRQ verastream**

RAPID, LOW-RISK
HOST INTEGRATION.

WRQ

www.wrq.com

Enabling Composite Applications by Reusing Legacy Data and Logic

By Scott Rosenbloom

How do you create new applications that reuse information locked away in your legacy systems? A cost-effective, reliable, and quick solution is to leverage the existing legacy data and logic in the form of composite services.

New Initiatives, Legacy Assets

An increasing need for timely information, propelled by Web technology, is driving a range of new initiatives in IT departments. Customers and partners experienced with the convenience of Web transactions increasingly look for this Web capability from all the companies they do business with. Businesses benefit from providing their employees with an integrated view of the customer, often through a Web portal. Meeting these imperatives is a challenge for organizations that have critical data and logic on legacy systems.

Legacy data frequently needs to be presented and updated online. For example, when customers access a Web self-service order processing application, they require information on availability, order status, and shipping options. An online purchasing application must also be able to update the host with information on purchases and new customers. Each of these data stores potentially resides on a legacy system, and it's unlikely that the legacy systems have Application Programming Interfaces (APIs) capable of supplying data directly to a Web application.

business integration journal takeaways

BUSINESS

- The methods to create new business initiatives with legacy assets include replacing, replicating, or reusing the assets.
- Reusing legacy assets takes advantage of an existing investment and leaves proven business rules in place.
- Legacy integration provides extraordinary benefits such as reusable data or data online in days or weeks rather than months, all at a fraction of the cost of a new system.

TECHNOLOGY

- The presentation layer is the only way to access 80 percent of legacy systems.
- Reusing legacy assets minimizes the need for new coding, requires less testing and eliminates data synchronization problems.
- Providing security and performance are the top challenges to bringing legacy assets online, but are solvable with the right planning and technology.

Web portals, through which employees access information about customers, face nearly the same issues as Web self-service applications. A key strength of portals is their potential for increasing employee productivity by presenting multiple sources of customer information in a single view. These sources may include technical support records, account activity, and contact information, and some of this information can reside on legacy systems.

The presence of critical data and logic in legacy systems increases the challenge IT departments face when they pursue new Web-driven initiatives. Yet, replacing the legacy systems isn't their only option and often isn't the best one.

Options for Legacy Systems

As shown in Figure 1, the challenge presented by legacy systems can be overcome in several ways:

- Replace the legacy system with a packaged application
- Replicate legacy data and logic in a new system
- Reuse legacy data and logic by Web-enabling the legacy host.

Replace

Replacing an existing system has the advantage of letting the system designer select an application that specifically supports the required initiative. If the initiative requires providing inventory and logistics information through a portal, the designer can select a packaged ERP application that includes all the essential features in a single product. This fully integrated design simplifies management and maintenance, provided the ERP application is stable and doesn't require extensive customization.

Replacement has drawbacks, despite the attraction of migrating to new technology. It can be a costly approach, since buying a packaged solution to replace existing core business systems requires spending money on new software and the services to customize and deploy it. Often, it doesn't support the company's business processes, which have been embedded in the existing systems over the years. Implementing the new solution can also be time-consuming.

Replicate

Replicating an existing system requires copying information from existing data stores and re-creating business logic. The duplicated information is synchronized with the information in existing systems.

This approach has the advantage of creating a responsive application that's sized and customized to work with the new data store. It also retains the use of the legacy system for functions other than those covered by the replicated system, avoiding some of the risk of replacing a smoothly running system.

Some of the pitfalls of replication arise from the risk of re-creating business logic that formerly resided on the legacy system. Often, the legacy business logic is poorly understood and is intertwined with the presentation logic on the host. Re-creating this logic on a new system is prone to errors, resulting in data corruption from incorrectly validated data. The lack of knowledge of the legacy logic can also present problems for planning the replication work, leading to project schedules slipping.

Reuse

Reusing the logic and data in a legacy system makes use of an existing investment. You leave the existing information and business logic in place and expose business functions as services. These services can then be composed with other services, from additional legacy systems and from new systems, to build composite applications that are Web-enabled. The legacy business logic can be used unchanged, with its proven business rules intact. Less testing is required than with replacement or replication, since there's far less new code. With no code or data replication, there are no problems with data synchronization; all information is synchronized by definition. Overall, the requirements for less new code and less testing make reusing the legacy system the fastest option to implement and the least costly. In addition, data from the legacy system can be exposed without changing any of its core functionality or operations.

Some challenges arise when reusing legacy systems. Security can be a concern, in particular when host security models need to be resolved with those of the new Web application. Performance can also be an issue when the connection between the legacy system and the service enablement layer presents a bottleneck. It's important to choose an integration server product that addresses these issues. When these challenges are overcome, the cost, speed, and reliability advantages of reusing legacy assets can be realized.

How to Reuse Legacy Data and Logic

Reusing legacy assets has clear advantages over replacement or replication; it reduces the amount of new coding and takes advantage of the proven reliability of the existing system. To reuse the legacy assets, the data and logic of the legacy system need to be exposed through an interface. The interface can be found in one of three layers: the data layer, the transaction layer, and the presentation layer. The choice of which layer to expose is dictated by the design of the legacy system and by your requirements for the data:

- **Data layer:** When read-only information is desired and the underlying data store is accessible, expose the system through the data layer for optimal performance.
- **Transaction layer:** The transaction layer may not exist as an explicit feature if business logic is intertwined with presentation logic. However, if the data layer isn't an option and the transaction layer is available, you should access the system through the transaction layer.
- **Presentation layer:** When there are no APIs for the legacy system, as is often the case, you need to access host data and logic by creating your own API at the presentation layer.

The presentation layer is the only way to access more than 80 percent of legacy systems. Concerns about data integrity often prevent using the data layer, and ill-defined business logic prevents using the transaction layer. In effect, a green-screen user interface must be adapted to provide both data and logic as a service to other applications, turning the presentation layer into an API. With the right technology, this can be an efficient process that provides secure, high-performance access to the legacy system.

Your choice of tool for exposing the legacy presentation layer is critical to the success of the integration project. You must be able to navigate the green-screen interface effectively, extract information from it with minimal time and effort, and automatically generate services that can be provided to other applications. To have the flexibility you need for achieving successful host integration — on time, within budget, and up to specifications — it's essential that you have the following capabilities

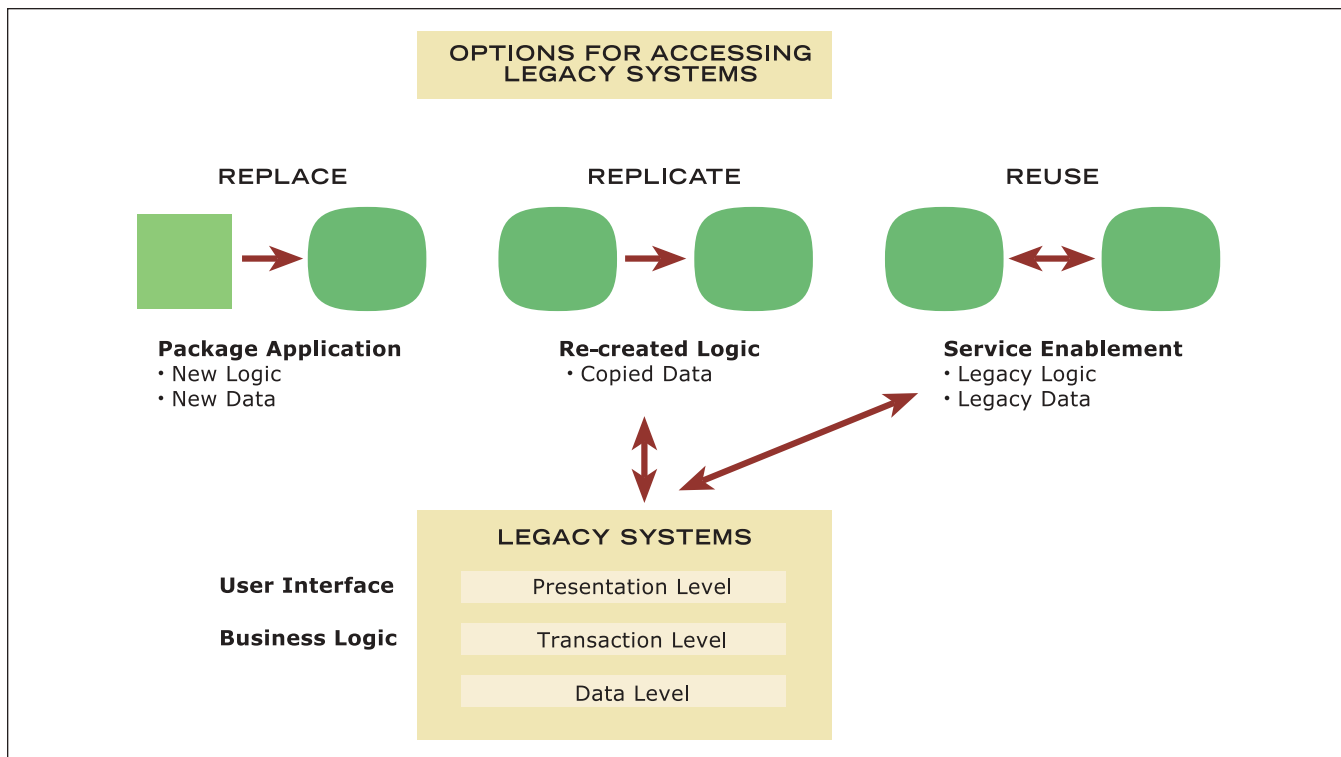


Figure 1 — The options for enterprises with legacy systems include replacing, replicating, or reusing the legacy data and logic.

when exposing the host business rules:

- An ability to expose the services via various technologies
- The ability to build the composite application, using the legacy composite services in the Integrated Development Environment (IDE) of your choice (.NET or Java).
- Auditing and logging capabilities (to keep the host system secure).
- Failover because hosts are typically reliable and you must ensure that the Web infrastructure is just as reliable.

Security and performance are two challenges you need to consider adequately when exposing legacy information to the Web via composite applications.

Security becomes an issue when you open a host system to the Internet through Web applications, or when the flexibility permitted by service enablement leads to new groups of users with new security requirements. For a secure service-enabled system, check for the following in your integration project:

- Do you need to map the security context of your Web application with the security context of the host? This would allow authentication of users all the way through the system from Web to host, sometimes important

for auditing purposes. Your integration tool should make this mapping possible.

- Do you need a different security granularity for your new Web application from that on the host? If so, this must be implemented in a middle layer. For example, the host application may have been written for internal users. Once logged on, these internal users have access to information about all your customers and orders. The new services expose the host to customers who access it through a Web self-service application, but you want the customers to have access only to their own orders, not anyone else's. This change in security granularity needs to be addressed in the access layer, between the host and the Web application. Your integration tool should help you design the granularity you need as you build the access layer.

Performance is affected when the presentation layer of the legacy application creates a bottleneck for accessing the host data. This can happen if the host system requires logging in and out for each session, or if data transfers generate many round trips between the host system and Web application. Look for these performance issues as you plan for host integration:

- Do you have excessive overhead for logging in and out? If so, this can be solved with session pools — groups of sessions pre-connected and already navigated to the proper request screen.
- Do you need auditing? Auditing can affect performance if it requires connecting to the host more than once. It's essential for the server to pay the connection penalty only once, an efficiency that can be achieved by caching and maintaining secure sessions for each user.
- Does the number of round trips between the requester and host affect your performance? Keep the number of round trips to a minimum with server-side processing of interactions with the host. You can facilitate server-side processing through abstraction of host data into an easily accessed database, which is inserted as a table-like layer between the host and the requester. An additional benefit of data abstraction is the ease of using the host services through standard database queries. You can do that by allowing the Web developer to construct queries through SQL, which also accelerates the development process.

To visualize some of the challenges faced during legacy integration and how to solve them, consider the case of a fictitious company that would like to provide online ordering.

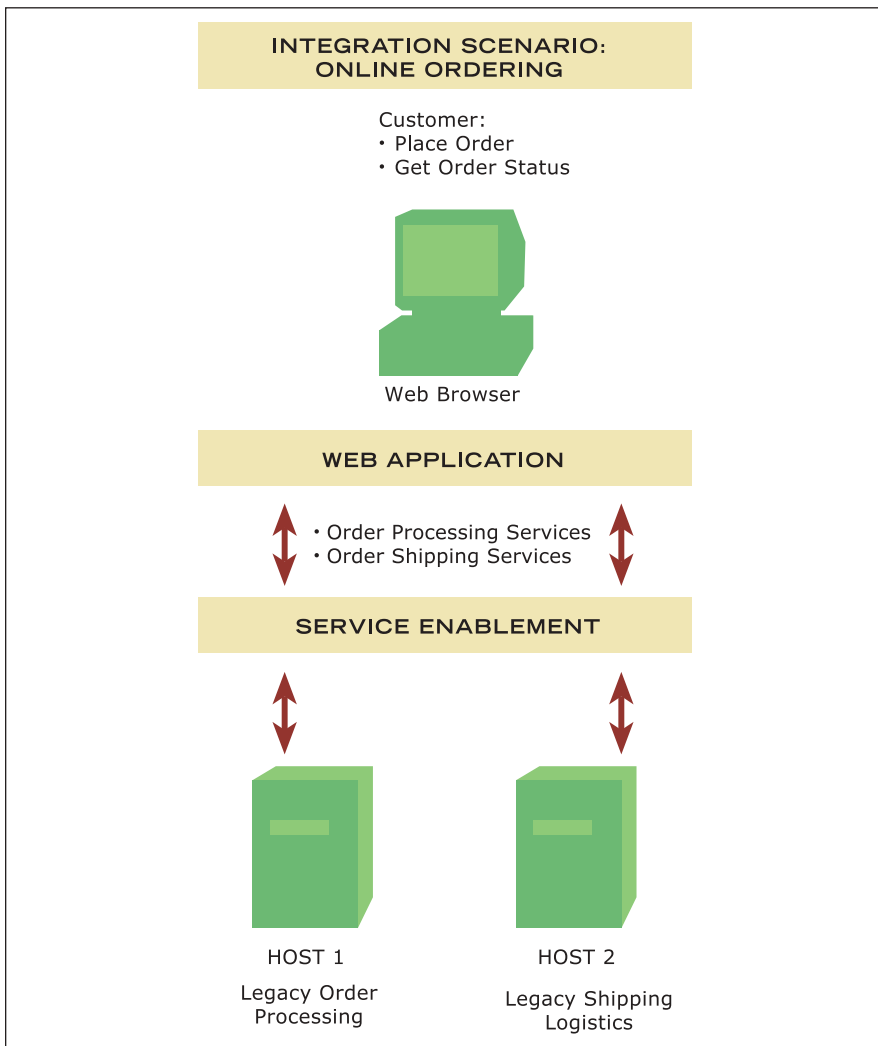


Figure 2 — Online ordering through a Web self-service application may require data and logic from multiple legacy systems.

Online Ordering Scenario

When businesses make the transition from a traditional ordering system to an online ordering system, they often face the need to integrate legacy assets with a Web application. Consider a supplier whose existing ordering system has involved taking customer calls, faxes, or written orders and keying them into a host system through terminals. Now the supplier would like to let customers place orders around the clock, using a self-service Web application to save time for the customer and the company. The Web application addresses the design of an online catalog and shopping cart with a user-friendly interface.

The order processing and shipping logistics systems at the supplier reside on host computers. Order processing could be re-created in the Web application but would need to replicate the extensive logic already available on the host system

and would need to synchronize with the back-office order processing systems. Calculation of shipping costs and tracking of the shipments is likewise already available; replicating these functions would be unnecessarily costly and would risk introducing errors into a shipping system that has been extensively tested and fine-tuned.

As the quickest, most cost-effective way to provide online ordering, the supplier has decided to reuse legacy assets by service-enabling them (see Figure 2). The Web application will consume the services.

Several challenges emerge on closer inspection of the supplier's legacy systems:

- There's no service interface, not even an API. The business logic is entwined with the data and presentation layers.
- There's significant overhead in establishing a host session, which affects

performance. Performance is also hurt by the lack of scalability in the creation of sessions on the host system.

- Security on host systems is limited to employees only. There's no way to segregate information by customer login.

In the end, the supplier's IT department satisfies these challenges by:

- Using Graphical User Interface (GUI) tools to learn how to navigate the host application, encapsulate its business logic, and expose it as an interface such as a Web service.
- Using session pools to pre-establish sessions and navigate to appropriate transactions on the host. The sessions return to a start transaction when the current transaction completes, ready for the next client request.
- Abstracting the interface and exposing specific information based on the security context of a client request. For example, the design lets a customer submit orders only for themselves and not for others, and view the status of only their own orders.

Conclusion

The methods to create new business initiatives with legacy assets include replacing, replicating, or reusing the assets. The most expedient, least costly method is to reuse the legacy system — to service-enable existing applications, connecting host systems with Web applications. There are some challenges to bringing legacy assets online, such as providing security and performance, but these problems are solvable. The benefits are extraordinary — data online in days or weeks rather than months and at a fraction of the cost of a new system. **BJ**

About the Author



Scott Rosenbloom is a strong proponent of service-oriented integration. He seeks ways to help customers address current integration needs while building a foundation for future initiatives. At WRQ, he helps shape the future of the WRQ Verastream® product line. Previously, he spent 10 years at IBM in software design, development, and consulting. Voice: 206-217-7500; e-Mail: scottr@wrq.com; Website: www.wrq.com