



Integrating Mainframe-Intensive Environments

How reusable services accelerate the process

CONTENTS

Mainframe-Intensive Environments: Different Needs, Different Challenges	1
Most of Your Work Has Already Been Done	1
Composite Apps Need Interoperable Services	2
Why Interoperable Services Can Be Difficult to Achieve	3
Unlocking Mainframe Applications	3
Creating Services for Screen-Based Mainframe Applications	4
Creating Services from Other Enterprise Systems ...	4
Pulling It Together: The Centralized Service Repository	5
About Attachmate	5

Integrating Mainframe-Intensive Environments

How reusable services accelerate the process

This paper introduces concepts that could change the way you look at integration projects. It presents advanced approaches for simplifying integration by using building blocks that improve performance on an ongoing basis. You'll learn how to breathe new life into existing systems—including mainframe applications—and use them as valuable assets to help reduce the time and cost of new business initiatives.

You'll also learn how to make business functions—currently locked up in mainframe legacy applications—available as reusable services. And you'll see how to do it without risky re-engineering or replacement. This concept of service-enablement paves the way for a dramatically accelerated approach to integration.

Mainframe-Intensive Environments: Different Needs, Different Challenges

New standards are continually emerging around system architecture, application development, and enterprise processing. But for most large organizations, one fact prevails: mainframe applications are still running many mission-critical aspects of the business. In the meantime, IBM z/OS (OS/390), IBM i5/OS (OS/400), HP OpenVMS, UNIX, and HP MPE/iX operating systems continue to thrive.

These mainframe-intensive computing environments have unique needs for integration. Mainframes are an integral part of enterprise computing—but not the only part. Mainframe systems are surrounded by newer generations of technology, and with each new application comes a need for integration with the core systems that run the business. But because older mainframe applications don't adhere to modern programming standards and interfaces, integration can be difficult.

Up to now, in fact, mainframe systems have often been viewed as an obstacle—something that impedes our ability to move quickly on new business initiatives. In today's reality, however, they can be a tremendous

asset. Thanks to advanced technology, it's easier to integrate mainframe systems and repurpose them to accelerate new projects.

Most of Your Work Has Already Been Done

Many enterprises have made substantial investments in business applications and databases. Millions of lines of code have been written. One could even argue that almost every important business function is already coded and in production. In an ideal world, all of these business functions would be individually packaged and fully interoperable; building new applications would simply be a matter of putting the functions you need in the proper order.

The term composite application describes such a solution. A composite application requires very little new code; instead, it relies on other systems to do most of its work (see Figure 1). In effect, it is a composite of many applications, and the majority of its business logic is stored and executed on other enterprise systems.

For composite applications to be feasible, business functions need to be interoperable and reusable. So while it's true that most enterprises have thousands of business functions already in daily use, they run on highly disparate systems and are far from interoperable. What's more, many of them are embedded in mainframe applications that were written long before there was any indication of multi-tier programming structures that segregate data, business logic, and presentation.

New technology allows you to break through those barriers. Using the right set of tools, it is possible to isolate business functions across disparate platforms, encapsulate them as interoperable services, and quickly build composite applications that reuse existing code in ways never thought possible. Application development is not only faster, but the new composite applications are inherently more bug-free because they execute code that has already been tested and proven in production.

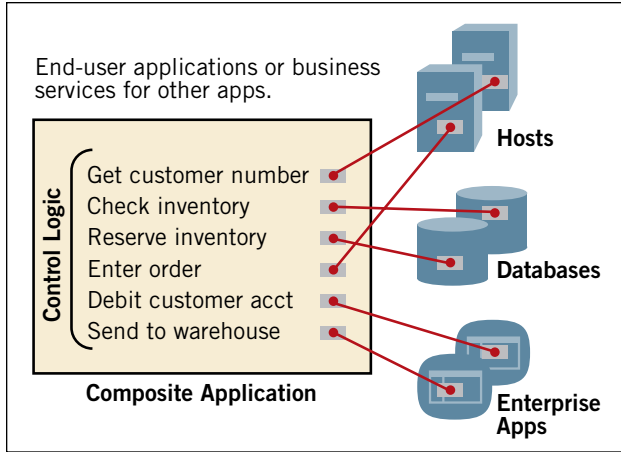


Figure 1. A composite application contains minimal logic. Instead, it relies on external systems to carry out business functions.

Composite Apps Need Interoperable Services

A composite application can take many forms, from comprehensive end-user applications to individual business services that support other systems. For example, a CRM application might take advantage of a business service called “change customer address.”

The composite application combines business services that represent the individual business functions available in the enterprise. A business service could perform a logical operation, or simply save or retrieve information. In the “change customer address” example above, the composite application might call upon a series of business services to synchronize the address update across several different systems. The ability to reuse these functions in various combinations becomes the basis for rapid results and fewer programming errors.

Existing enterprise systems are rich in business functions that can be harvested for reuse as services. It usually doesn’t take long to identify and encapsulate the short list of functions needed to support a given project, and you can always add new ones later as you need them. Over time, your library of available services will grow, making it progressively easier and faster to create new composite applications.

A composite application always contains some amount of logic, even though the majority of business functions are performed on external systems. For each external function performed, the composite application must

make a decision about the next step. For example, a composite application might check available inventory before placing an order; the function it executes next will depend on whether the item is in stock.

The practicality of composite applications, therefore, depends on these two prerequisites:

- **An ample source of interoperable business services.** This could simply be a list of what is available, or a more sophisticated central repository for managing services (see Figure 2). It is essential, however, for all services to use common interfaces and conventions. Otherwise, your composite application will become a complex matrix of point-to-point integration scripts.
- **Tools to create and manage additional logic.** Visual tools can simplify this process, and sometimes eliminate the need for manual programming altogether. Higher-level Business Process Management (BPM) tools are useful for combining multiple business services into composite applications. They also make it possible to integrate human processes.

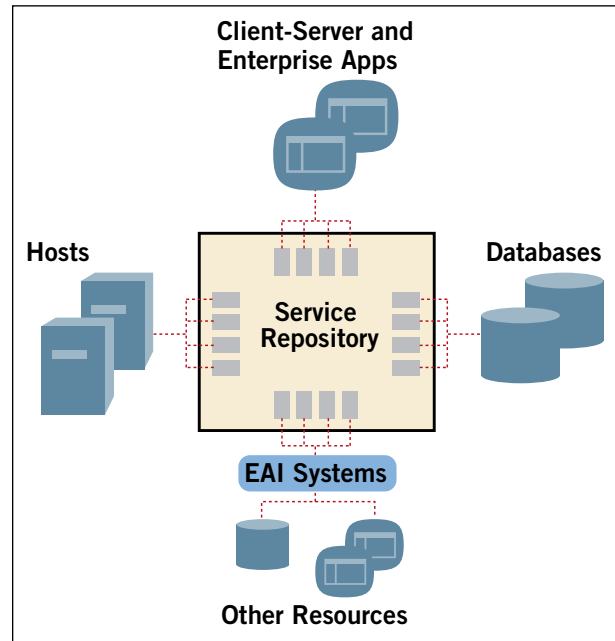


Figure 2. A central service repository manages the interchangeable business components necessary to create composite applications.

Why Interoperable Services Can Be Difficult to Achieve

The most attractive option, of course, is to create a library of interoperable, reusable services that represent the business functions in an enterprise. Without this centralized repository, it would be difficult to know which business functions are available and what they do. It would also be challenging to take advantage of visual tools to consume services, without a repository to view.

In the past, creating such a repository of disparate services has been impractical. One hurdle was the difficulty of making services appear and behave consistently when the underlying business functions all run on incompatible systems.

An early (and still common) approach to solving this problem was to replicate the business functions and data on a common platform to achieve consistent operation. This approach has major drawbacks, however. The replicated business functions on the new system might not work exactly the same way, thereby missing out on the advantage of already-tested code.

Data replication is an even bigger problem. Databases must be synchronized, which creates significant network traffic. And unless synchronization occurs frequently, there is a substantial margin for error. Wrong information—like whether an item is actually in stock—could damage the customer relationships that the application was created to improve.

For reliable operation, not to mention scalability, the services should always point to—not replicate—the business functions. All logic and data should remain on their original applications and databases. The services should simply provide the metadata that describes the business function, what parameters are needed to execute it, and where it is carried out. By invoking these services, the composite application can execute business functions on the original, unchanged systems.

Keep in mind that it is impractical—and often risky—to make any changes to original applications, especially mainframe applications. That's because they typically run mission-critical operations, and any disruption in service could directly impact business results.

Unfortunately, these applications can also be brittle. Because they usually do not isolate business functions or data, a change in one area of the program can easily break something else. (This problem often occurs when mainframe code is “wrapped” with additional

programming to create services.) Add the fact that many host applications are poorly documented (with the original programmers having moved on), and it becomes clear that any attempts to integrate must be 100-percent noninvasive.

Unlocking Mainframe Applications

The success of a new project can hinge on the ability to integrate with legacy mainframe applications because they contain large amounts of customer information and related business logic. Without access to these essential resources, CRM applications cannot deliver on their promise of a single view of the customer.

It's no accident that mainframe applications are among the best sources for harvesting reusable business functions. Even for companies that have installed comprehensive ERP systems, many legacy mainframe applications still support unique, mission-critical operations. There are compelling advantages to making these business functions available for new initiatives.

Newer mainframe applications are sometimes more accessible than old ones. For example, high-performance, direct transactions are sometimes possible through CICS/TS or IMS/TM on IBM mainframes. Even in these cases, however, work must be done to identify and create business services that can be used by other systems.

But most mainframe applications present a tougher challenge. Created long before multi-tier programming structures became popular, they are monolithic programs intertwined with data, business logic, and user interfaces. There is no delineation between individual business functions, and often the only access is directly to the transactions (in the case of CICS or IMS based applications) or through terminal screens.

It may be tempting to go around existing business rules, executing read and write functions directly to the mainframe database, but this carries a high risk of irrecoverable data corruption and it might crash the application itself. For a given operation, mainframe applications commonly write to several fields within a database, and arbitrarily changing one entry without going through the application logic could introduce errors into the system.

But even the toughest mainframe applications can be reused for purposes other than their original intent. Today, individual business functions in mainframe applications can be quickly identified, encapsulated, and reused in composite applications. More important,

this process can occur without changing the mainframe program in any way. And mission-critical applications can run exactly as they always have.

Creating Services for Screen-Based Mainframe Applications

The objective for integrating screen-based legacy mainframe applications (as with any other enterprise system) is to isolate business functions and turn them into interoperable, reusable services.

One approach is to emulate the end user, which has proven to be practical, fast, and effective. For example, to credit a payment to a customer's account an end user might take these steps:

1. Select from a menu: "get customer number."
2. Enter the customer's name; this returns the customer number.
3. Write the number down.
4. Back up to the main menu.
5. Select a new menu item: "update a customer record."
6. Enter the customer number.
7. Select a new menu item: "credit payment."
8. Enter the payment amount to credit.
9. Back up to the main menu.

Taken together, these steps perform the business function of "credit customer payment." To create a corresponding business service that can be executed by other programs, the same steps are followed and recorded. The resulting service is a meta description of:

- The function performed (credit a customer payment).
- Where it occurs (this host application).
- The information needed to complete the function (customer name and payment amount).
- The possible results produced (successful or not successful).

This service-creation process, sometimes called modeling, can be used for any business function the mainframe application can perform (see Figure 3). Because the services are all fully abstracted and interoperable, business functions from different types

of mainframe systems can be combined in any order along with those from any other enterprise application or database. The new application requires no mainframe-specific programming; all services present consistent, reusable interfaces.

The concept of emulating a real user began several years ago and was commonly called screen scraping because the programs "scraped" the screen, looking for specific information at specific coordinates. The process worked, but some fundamental problems prevented widespread adoption. Its use of an early programming interface called HLLAPI, for example, required each step of the process to be programmed into the new application. And with its reliance on specific screen coordinates, even the slightest change to the mainframe application meant reprogramming.

Most of those original problems have been overcome, and screen-based integration with mainframe applications is now practical. The ability to create fully abstracted services means new applications no longer require mainframe-specific code. Modeling now looks for information in regions of the screen, so minor changes (like an additional menu item) don't affect operation. There is still a chance that changes to the mainframe application will require a change to the service, but updates are now easy and only the model is affected—never the applications using the affected services.

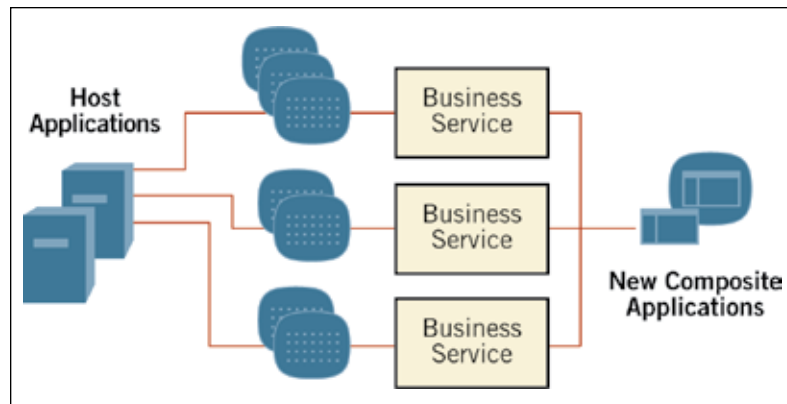


Figure 3. Reusable business service can be easily isolated within screen-based legacy applications by recording the actions necessary to complete a given function.

Creating Services for Other Enterprise Systems

Service creation for newer enterprise applications is usually easier, since the applications provide APIs for communicating with other programs. And the latest versions of these applications provide services out of the box. For applications that don't provide services, you'll need a specific adapter to layer a service interface on top of it. The adapter can communicate with the unique API of the underlying application.

In many cases, the services provided (either natively or via an adapter) by enterprise applications are too low-level to be meaningful to a developer creating a composite application. It may be necessary to combine a number of low-level services to create understandable business services.

Pulling It Together: The Centralized Service Repository

The ultimate objective is a centralized facility for managing all reusable enterprise services. As mentioned above, it is usually not necessary to create large numbers of services to get started; create only those you need to finish the project at hand. (Modest mainframe integration projects won't even require a central repository.) Your library of reusable business functions will grow over time and accelerate each new project.

From this central repository, it becomes possible to mix and match business functions originating anywhere in the enterprise, from mainframe systems to the newest application servers. Your new applications will be substantially faster and much more stable as a result of using already-proven business logic.

When evaluating service-oriented integration solutions, consider these questions:

- Can all enterprise services, including mainframe-based services, be made visible and accessible in a central repository?
- Are all services conforming to a standard so they can be interoperable, regardless of original platforms or protocols?
- Do the services point to business functions running on the original systems, or have logic and data been replicated?
- Are the business functions executed in real time when the composite application requests them?
- Are visual tools provided to create composite applications?

With new technologies, you can improve customer service quickly and economically—using fewer resources. That means you can play by the rules of today's economy and win.

About Attachmate

Attachmate helps businesses extend, manage, and secure their IT investments. We offer a broad range of solutions—from terminal emulation, legacy integration, and PC lifecycle management products to innovative systems and security management tools. With our technology, more than 65,000 businesses worldwide are putting their IT assets to work in new and meaningful ways. Learn more at www.attachmate.com.



Corporate Headquarters
1500 Dexter Avenue North
Seattle, Washington 98109
TEL 206 217 7500
800 872 2829
FAX 206 217 7515

EMEA Headquarters
The Netherlands
TEL +31 71 368 1100
FAX +31 71 368 1181

Asia Pacific Headquarters
Australia
TEL +61 3 9825 2300
FAX +61 3 9825 2399

Latin America Headquarters
Mexico
TEL +52 55 9178 4970
FAX +52 55 5540 4886

WEB attachmate.com
E-MAIL info@attachmate.com

For regional office information, visit www.attachmate.com.