

# FIVE DIRTY LITTLE SECRETS OF HIGHLY AVAILABLE INTEGRATION INFRASTRUCTURE

---

## TABLE OF CONTENTS

> Introduction	1
> Dirty Little Secret #1: The Myth of Five Nines	2
> Dirty Little Secret #2: Recovery Time	3
> Dirty Little Secret #3: Data Corruption	7
> Dirty Little Secret #4: Scheduled Maintenance	9
> Dirty Little Secret #5: Hidden Cost and Complexity	10
> Conclusion	11

---



---

## INTRODUCTION

Your enterprise software vendor says that its integration infrastructure provides high availability. Does it really?

Conventional integration infrastructure achieves high availability by combining clustering—the ability to share processing load across multiple machines working together—with commercial failover management software.<sup>1</sup>

The good news is that these conventional highly available integration infrastructures do recover from failure and restore service. The bad news is that there is often a gap between what the business expects from “highly available” infrastructure and what this infrastructure actually provides.

The purpose of this paper is to critically examine some broadly-held misconceptions of high-availability (HA) integration infrastructure. Too often, these issues do not become clear until after the infrastructure is purchased, and only then do quality gaps and hidden costs become apparent. Today, as more and more of the business world counts on the continuous availability of integration infrastructure, it is time to shine a light into this murky area and to expose the “dirty little secrets” that your enterprise software vendor might prefer you did not know.

Here, then, are the five dirty secrets of highly available integration infrastructure:

1. **The Myth of Five Nines:** traditional measures of availability don’t track what your customers care about, namely, how long will it take to process their transactions when systems fail.
2. **Recovery time:** conventional highly available integration infrastructures actually interrupt transaction processing for 5-15 minutes when they recover from failure.
3. **Data corruption:** conventional highly available integration infrastructures may deliver transactions twice or in the wrong order when they recover from failure.
4. **Scheduled maintenance:** conventional highly available integration infrastructures need to be brought down in order to upgrade hardware or software.
5. **Hidden cost and complexity:** conventional highly available integration infrastructures require third-party products, application modifications and complicated operational procedures.

Unless an infrastructure provides “continuous availability”—*meaning that transaction processing continues uninterrupted despite system failure*—infrastructure called “highly available” is actually only “mostly available.”

1. Examples of commercial failover management software include VERITAS Cluster Server, Microsoft’s Cluster Server, Sun Cluster, HP’s MC/Service Guard, or IBM’s High Availability Cluster Multi-Processing

## DIRTY LITTLE SECRET #1: THE MYTH OF FIVE NINES

We have integration infrastructure that provides five nines availability. Is that overkill? Is that enough?

What's the best way to measure uptime? It's common for vendor marketing literature to express availability as a percentage, as shown in the chart below<sup>2</sup>:

Availability %	Downtime per Year	Downtime per Month	Downtime per Week
90%	36.5 days	72 hours	16.8 hours
99%	3.65 days	7.20 hours	1.68 hours
99.9% ("three nines")	8.76 hours	43.2 min	10.1 min
99.99% ("four nines")	52.6 min	4.32 min	1.01 min
99.999% ("five nines")	5.26 min	25.9 s	6.05 s
99.9999% ("six nines")	31.5 s	2.59 s	0.605 s

So what's wrong with this?

First, not all downtime is created equal. In retail, most merchants do their most profitable business during the last few weeks of the year; an hour of downtime during peak holiday shopping is clearly more serious than an hour of downtime in July. For some businesses, seasonal peaks can be extreme. The U.S. flower industry does most of its business on two days of the year: Mother's Day and Valentine's Day. In the brokerage industry, most trading volume occurs in the first and last hours of the day, and it is during those hours that system uptime is most critical. Because the market moves every few seconds, even less than a minute of downtime can impact trading profitability.

Second, availability percentages measure the wrong thing. What your customer—whether internal or external—really cares about most is how long systems will be unable to process transactions during any given failure. Maximum system recovery time is much more important than availability as a percentage.

**Ironically, guaranteeing uptime is hardest exactly at the times when it is valued the most. This is simply because systems tend to fail more often when run with heavy loads. Disks run harder, chips run hotter, and software scales to much higher levels than seen under normal usage.**

<sup>2</sup> [http://en.wikipedia.org/wiki/High\\_availability](http://en.wikipedia.org/wiki/High_availability)

---

Finally, it's important to recognize that while the clustering of servers does improve their availability as a percentage, it doesn't reduce maximum system recovery time. The time it takes for a given server in the cluster to recover from failure and resume processing transactions it held pending doesn't change. Any transactions caught on that server will be trapped until the server recovers. So the next time someone suggests that two five-nines servers in a cluster give you ten-nines availability, beware of misleading HA arithmetic.

Planning for high availability requires more than discussion around "five nines." Analyze the impact of downtime during critical times, and set clear standards for the maximum system recovery time that can be tolerated. Understanding *when, how often and how long* systems can be down will give you a much better picture of your availability than a single average can suggest. Whether a business can or cannot tolerate this downtime will, of course, depend on its unique requirements, but with these factors in mind, you can understand the impact concretely and plan appropriately for high availability.

## DIRTY LITTLE SECRET #2: RECOVERY TIME

We have integration infrastructure that is supposedly "highly available," yet transactions get held up when there is underlying hardware, software or network failure. How can this be?

It's no secret that high-availability integration infrastructure requires some time to recover from failure: during this recovery process the failure is detected; clients disconnect from a disabled primary server to a standby server; and processing picks up from where it left off. What is a "dirty little secret" is how long this recovery can take, especially in systems that process a large volume of messages.

In fact, it is common for the core messaging components of integration infrastructure to take 5-15 minutes to recover. The time it takes depends on the size of the transaction recovery log. A typical 2 GB recovery log requires about 12 minutes to recover. Ironically, those same systems that are tuned for high volume tend to take the longest to recover because their recovery logs are the longest. In addition, failures tend to occur when the volume is greatest and, therefore, the stakes are the highest.

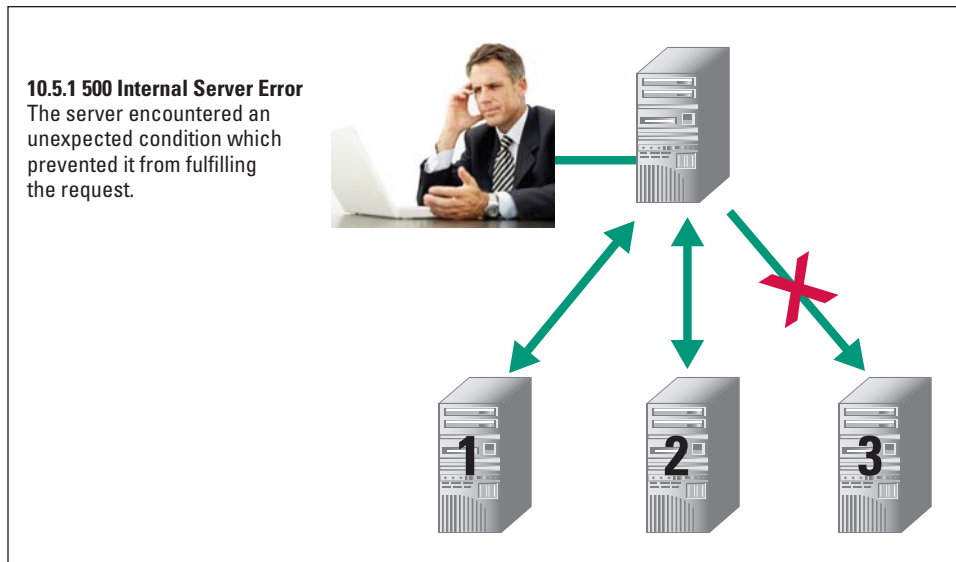
The impact of a one-minute outage on customer satisfaction is bad enough. However, a ten-minute outage would not only impact ten times as many customers, but each of them individually would experience much higher dissatisfaction because of the extended wait.

Some vendor solutions try to minimize the perception of downtime associated with recovery by providing very fast client failover. With these solutions a client may be able to disconnect from the disabled server and reconnect to a backup server in a matter of seconds. However, while these solutions reconnect clients quickly, they do not immediately resume the processing of in-flight transactions. They must first recover the transaction log. Therefore, when evaluating recovery time, it is critical to note not only how long it takes for client connections to be restored, but *how long it takes for processing of transactions to resume from the point of failure*. This latter aspect of high availability is challenging and, therefore, requires the most attention.

### What is the impact of recovery time?

The impact of recovery time depends on the integration scenario. Usually, integration infrastructure is chosen not just for a single integration scenario, but for a broad range of projects. To understand the impact of recovery time, let's look at what it means for three general cases: portal integration, straight-through processing and real-time data gathering/distribution.

### Portal Integration



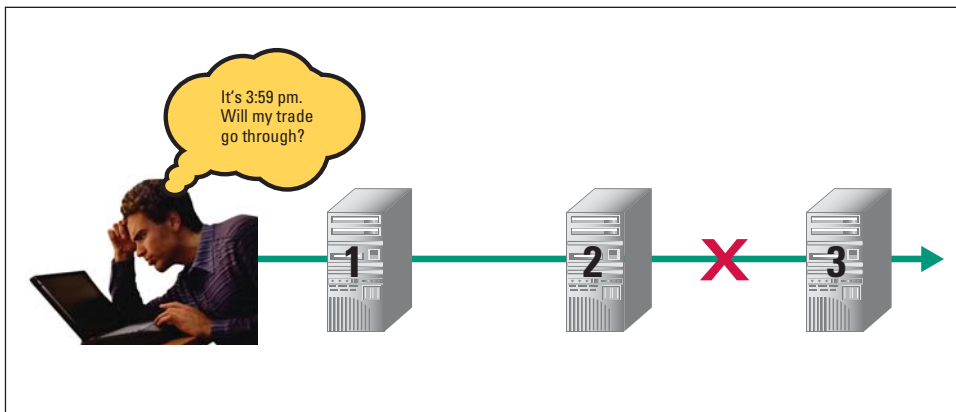
If the portal serves customers or customer service representatives directly, the high availability of its integration infrastructure is very important. This is especially so when the portal handles revenue-generating transactions or provides other critical customer services.

Most portal integration is synchronous or request/reply interaction, as represented in the diagram for the integration of resources #1 and #2. These resources are consulted to fill in a web page with such things as product or customer information. If a synchronous back-end service is unavailable, the portal will wait until it is available and, if necessary, time out and failover to a backup instance of that same service.

However, some portal integration calls for asynchronous (fire-and-forget) interaction, as seen in the diagram represented by the connection to resource #3. This type of interaction is common for the final submission of an order for processing. Because the interaction is asynchronous, final acknowledgement is provided later, by an email, for example. If the asynchronous communication link fails, then orders cannot be taken by the infrastructure during the recovery process. In this case the failure is acute because human interaction is involved. A recovery time of 5-15 minutes is unacceptable for customer-facing uses: prospects will leave for rival services if a web site is down or if the customer service representative cannot handle in-coming calls.

An integration infrastructure that provides continuous availability will cause no interruption of service of the client and will automate the connection of the client to a backup server in the event of failover. This means that the failover is effectively invisible to the end-user and also transparent to the developer, who does not need to handle the complex failure modes of conventional highly available asynchronous integration infrastructure.

### ***Straight-through Processing***



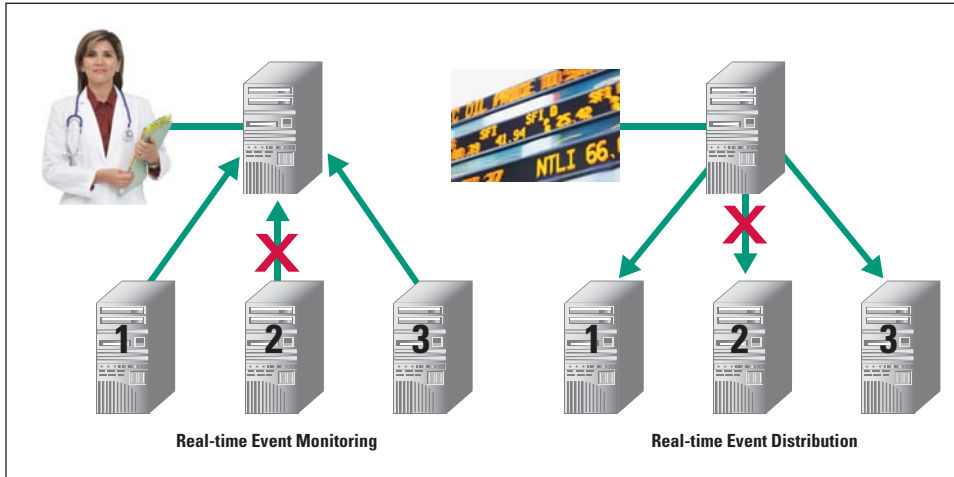
When straight-through processing must be completed quickly—either for competitive or regulatory reasons—downtime that interferes with the timely processing of transactions is critical. In these scenarios, customer interaction is not necessarily impacted immediately: the interaction with an order-taking system may proceed normally, as the integration solution's core enterprise messaging system allows the order management system to "fire-and-forget" its transaction for downstream processing. However, as far as our customer in the figure above is concerned, the failure results in downtime because transaction processing is not completed in the expected timeframe. *Therefore, the question to ask is "How long can a stuck transaction be tolerated?"* If revenue or reputation is at risk for each minute the transaction is stuck in processing, then recovery time is critical. Unsure what happened to an unacknowledged order, a user may re-submit it, resulting in duplicate transactions and complex and expensive manual

A failure to process user input or, worse, a prompt to re-enter all data could result in duplicate processing. To guard against this, developers must carefully handle the exception and roll back the transaction in order to ensure that it is submitted only once. This exception handling must be tested to ensure that it works properly no matter where in the order submission process the failure occurs.




interventions later on to reverse it. Furthermore, if transactions must be processed in order (to prevent such errors such as a customer update before a customer is created, for example), then a single stuck transaction may hold up subsequent transactions as well, which may have greater impact on the business.

### ***Real-time Event Monitoring and Distribution***



In this scenario, data or state changes (events) must be transmitted across a distributed environment in a timely and reliable way. When business operations or human safety depend on continuous operations, the high availability of the underlying integration infrastructure must be addressed. What are the potential impacts of stuck transactions?

Real-time event monitoring systems, such as health and safety monitoring, business activity monitoring (BAM) and network monitoring, require the timely and reliable collection of data from remote devices or systems. Hospital integration systems, for example, connect patient bedside devices to nursing stations, and interruption of the flow of data can put patients' lives at risk. Similarly, emergency management integration requires the continuous sharing of real-time information across multiple systems to assess and respond to accidents or premeditated attacks on infrastructure: an explosion in one power distribution transformer is an isolated incident, but closely-timed explosions at multiple points in the power grid could indicate a terrorist attack and require a coordinated response. This is possible only if the integration infrastructure that connects these multiple systems is highly available and not subject to delayed transactions during any period.



---

Directing the flow of data from one point to many (a publish-and-subscribe scenario) is a common requirement for master data distribution. Failure to distribute up-to-date pricing, product or customer information to distributed processing points typically causes downstream failures in transactions that expect consistent master data at all those points. For example, a common failure scenario is the registration of a new customer followed immediately by an order: if the notification of a new customer registration is delayed and arrives after the customer's first order, complex exceptions in the fulfillment and invoicing systems can occur. It is far better to avoid these problems entirely through continuously available integration infrastructure than to correct them after the fact.

### **DIRTY LITTLE SECRET #3: DATA CORRUPTION**

The enterprise messaging system of our integration infrastructure is supposed to guarantee transactional message delivery. However, when configured for HA, it relaxes rules about in-order and once-and-only-once delivery semantics. How can that be?

Because most people today take the transactional properties of database systems for granted (that is, they can trust that infrastructure will guarantee the integrity of transactions committed to a database), it may come as a shock to learn that typical enterprise messaging systems actually degrade their quality-of-service (QoS) guarantees when they are configured for high availability! After all, one would reasonably expect that the same systems, engineered so that they will not go down, would do so without changing their behavior. If in-order delivery is guaranteed when systems are running normally, then it should be guaranteed even in the event of a system failure. If not, your developers will have to assume that messages can be delivered out of order and code around it. This paradox—quality of service goes down when high availability is turned on—is the third “dirty little secret.”

Today, most enterprise messaging systems can be expected to provide quality-of-service guarantees. Because quality of service does impose a latency and throughput performance penalty and increases hardware costs, it is configurable and should be used judiciously. Through these controls, the behavior and performance of message delivery can be optimized to suit a given project requirements.

As systems move from batch to near real-time operation, fast and reliable data synchronization becomes critical for error-free processing. Too often, inordinate staff time and energy are spent debugging and correcting transactions that fail because master data are out-of-date or unavailable. Customer satisfaction suffers, too.

**Conventional Clustered HA Integration Infrastructure:  
Expected vs. Actual Behavior**


QoS	Expected Behavior	Actual Behavior
Best-effort	Messages may be lost or delivered out of order	<b>As expected:</b> QoS guaranteed
At-least-once	Messages may be delivered more than once or out of order	<b>As expected:</b> QoS guaranteed
Once-and-only-once	Messages may be delivered out of order	<b>QoS NOT guaranteed:</b> Duplicate messages may be delivered during recovery process.
In-order, once-and-only-once	Full integrity guaranteed	<b>QoS NOT guaranteed:</b> Messages may be delivered out of order during recovery process.

In fact, when set up for high availability in a clustered configuration, conventional enterprise messaging systems lose their ability to guarantee once-and-only-once and in-order delivery. As a result, integration infrastructure that recovers from failure may well deliver transactions twice or in the incorrect order.

Nearly all business transactions that result in an update to a database need to guard against duplicate delivery, and most transactions require processing in the correct order as well.

Duplicate delivery is a problem that most developers routinely work around, but this assumes that the target system for integration can be modified easily or at all. Too often, the team that built a system is no longer in place when needed for additional integration projects. This slows down projects and creates risks when modifying legacy systems.

In-order delivery is a trickier problem still. In this case, the developer must hold pending transactions while waiting for a transaction that should have been processed earlier to arrive. This creates new reliability and availability problems for the application developer, who now needs to deal with the possibility of the application going down while managing the pending transactions.



---

If developers are not experts in these issues, it's possible that they could be left unaddressed, leading to errors in production systems. Issues will be detected during system testing only if once-and-only-once and in-order test cases are considered up-front. Because these problems emerge only when the system is scaled up with additional servers to increase system throughput, they frequently escape notice until tested in a clustered configuration. So, in the best case, the problem will be found during pre-production testing. In the worst case, it will show up in production systems after they are scaled.

As a matter of principle, the semantics, scalability and availability of the infrastructure should be independent considerations. This is especially so for SOA infrastructure, which can be expected to be used in a large variety of projects requiring different and changing requirements, including quality of service. More than ever, it is critical to question how well an infrastructure can guarantee data integrity in highly available, clustered configurations.

#### **DIRTY LITTLE SECRET #4: SCHEDULED MAINTENANCE**

**We have a clustered system that's supposed to be highly available, yet I need to shut it down completely in order to maintain it, even for such things as applying operating system patches. How can I run 24x7?**

More and more, globalization and 24x7 operational requirements make it harder to schedule downtime in mission-critical systems. International operations (whether they serve employees or interact directly with customers) require "follow-the-sun" systems that run continuously around the clock.

Typical high-availability solutions require absolutely identical hardware and software configurations for the servers on which they run. Therefore, an upgrade on one machine requires a simultaneous upgrade on them all; every server must have identical hardware, operating systems (including patch level), and failover management software in order to work. This means that the entire cluster must come down to make even a minor change, such as a security-related patch on a given operating system. This is an often overlooked issue that can cause operational headaches after a high-availability system is deployed.

Highly available systems should have more flexibility than that and should run on heterogeneous hardware and operating systems. In this scenario, maintenance could be performed by upgrading machines in a cluster one at a time. The high-availability features of the cluster would direct work to the remaining machines in the cluster while a given machine is being upgraded. Then, when the upgraded machine is re-introduced to the cluster, full service would be restored. In this way, continuous operations could be delivered to the business and maintenance performed without disruption.

Frequently, the same business requirements that make high availability important at all (such as human safety) make scheduled downtime problematic, too.



---

## DIRTY LITTLE SECRET #5: HIDDEN COSTS AND COMPLEXITY

Most enterprise messaging solutions don't provide high availability out of the box, and, therefore, require third-party commercial failover management software and specialized hardware to be purchased separately. Furthermore, while they are robust, none of these solutions provides failover that is transparent to client applications, so application writers must code for and test complex failure scenarios. Too often, these extra costs and complexities escape notice during product evaluations.

Conventional integration infrastructure clustering distributes processing load across multiple machines that work together. This does not provide high availability out of the box. In the event of failure of a node in the cluster, transactions persistently queued up for reliable processing on that node will be stuck until the server comes back up and reconstructs the queue from the transaction log. Without a high-availability add-on product, all of this is manual work, from the detection of the failure to the restart of the failed broker or server.

To enable automatic recovery from failover, conventional integration infrastructure requires third-party, add-on products in the form of commercial failover management software, such as VERITAS Cluster Server, Microsoft's Cluster Server, Sun Cluster, HP's MC/Service Guard, or IBM's High Availability Cluster Multi-Processing. These products entail up-front and maintenance costs above and beyond those of the base integration infrastructure. In addition to the costs of the commercial failover management software, specialized hardware is required as well. A dual-ported RAID disk or SAN must be used to allow both the primary and standby servers to share access to the transaction log.

Beyond these third-party software and hardware costs, most high-availability integration solutions impose extra coding and testing burdens on development and deployment teams. These burdens come in the form of catching and handling the exceptions that occur in connected clients when a failover occurs. Because these solutions do not provide transparent failover—that is, the failover is not invisible to the client but must be explicitly handled by it—the client code must be prepared to roll back and re-try transactions. Exception pathways must be analyzed, coded and tested.

If there are human interactions depending on these transactions, then this situation must be presented in the most user-friendly way possible, although it is never easy to break the news to an interactive user that "Systems are down for 5-15 minutes; please stand by." (See "Dirty Little Secret #2—Recovery Time.") Furthermore, coders must shoulder the burden of working around the issues of potential out-of-order or duplicate transaction deliveries that are inherent in these conventional high-availability integration solutions. (See "Dirty Secret #3—Data Corruption.") Not only is this work difficult and error-prone, it is entirely systems-level programming: necessary for correct system operation, yes, but hardly the kind of development that most IT teams want to implement in-house.

At the time of publication of this paper, a dual-ported RAID controller with 500 GB of disk space cost approximately \$20,000—far more than the cost of an equivalent amount of local storage. To the extent that many of these primary/backup pairs must be deployed in an enterprise, these costs can rapidly mount.

## CONCLUSION

### Average Costs of Unplanned Outages for U.S. Industries.<sup>3</sup>

Retail brokerages	\$6.45 million / hour
Credit card sales authorization	\$2.6 million / hour
Infomercial/800-number promotion	\$199,500 / hour
Catalog sales center	\$90,000 / hour
Airline reservations	\$89,500 / hour
ATM services providers	\$14,500 / hour

Across industries, in nearly every facet of human endeavor, the increase in business velocity is unmistakable. The ubiquity of the web and the rapid adoption of standards make it easier than ever to connect customers and back-end processing. A process that might have taken a week only a few years ago today completes in a day, an hour or even minutes. Customers expect fast and certain interactions with automated systems and execution of orders.

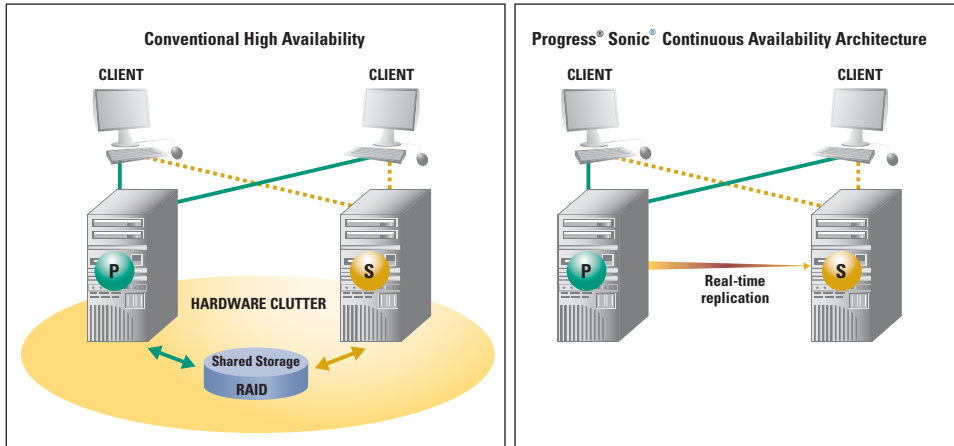
Too often, high availability is discussed in terms of server availability, which is a necessary but insufficient condition for customer satisfaction. Customers are not interested in “server response time” as much as “service fulfillment time.” A snappy web site experience is soon forgotten if a downstream failure results in a late or incorrect order. The scope of thinking about availability needs to change.

As we have seen, there are big gaps between what the business expects and what conventional “highly available” integration infrastructure actually delivers. Not only does it fail to deliver the continuous and error-free service that the business needs, it has numerous hidden costs that are often not understood until after the infrastructure is purchased and deployed.

Today, there are superior alternatives to the conventional solutions. An integration infrastructure solution that provides transactional high availability as a native capability can indeed fulfill these expectations. Not only does this approach fulfill the business

3. Source: InternetWeek 4/3/2000 and “Fiber Channel: A Comprehensive Introduction,” Kembel, 2000, p. 8, based on a survey by Contingency Planning Research.

expectations for high availability, it can avoid the hidden costs of conventional solutions by obviating the third-party add on products as well as the complexity, time and costs associated with developing in the conventional environments.



The diagrams contrast conventional HA integration infrastructure with the unique approach taken by Progress Software.

The diagram on the left shows a conventional high-availability integration solution based on an add-on failover management solution. Upon failure of the primary server, the secondary (backup) server starts up and begins the recovery process from shared storage. Such systems require costly third-party add-ons, including commercial clustering software and dual-ported disk drives.

The diagram on the right shows the Progress approach, which is based on native, real-time replication. The secondary server is continually updated with pending transactions stored on the primary server. As a result, not only does the client fail over to the secondary server quickly and automatically, but the secondary server begins processing of pending transactions immediately. This completely bypasses the usual 5-15-minute recovery process inherent in the conventional high-availability architectures and, thereby, provides your customers the availability of service they care about most—the uninterrupted flow of data and processing of transactions. It does so without degrading quality of service; in-order and once-and-only-once semantics are maintained. And, because this software-only solution runs on a variety of hardware platforms and operating systems, it does not require a complete shutdown in order to perform maintenance on underlying infrastructure. This approach provides failover that is completely transparent to the client and requires no additional clustering software or specialized hardware.



---

Progress Software calls this unique approach “Continuous Availability Architecture.” Introduced in 2006 with Progress® SonicMQ® and Progress® Sonic® ESB integration products, it provides industry-leading performance levels at lower cost than conventional highly available integration solutions. Today, hundreds of organizations—in financial services, telecommunications, government, e-business and more—depend on Continuous Availability Architecture to improve their business’ operational efficiency and reduce capital costs through the highest levels of availability, reliability and performance.

We invite you to contact Progress and learn more about how Progress high-availability integration solutions can help you deliver the service quality that your business requires.



**Worldwide Headquarters**

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA  
Tel: +1 781 280-4000 Fax: +1 781 280-4095  
[www.progress.com](http://www.progress.com)

**For regional international office locations and contact information, please refer to [www.progress.com/worldwide](http://www.progress.com/worldwide)**

Progress, Sonic, SonicMQ, and Sonic ESB are trademarks or registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the US and other countries. Any other trademarks contained herein are the property of their respective owners.

© 2009 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

Prod Code 3658



0000123014