Next-Generation Test Data Management

How to deliver the right test data, to the right teams, at the right time
Welcome

Applications are critical to the success of modern businesses, enabling them to optimize internal processes, gain and retain customers, and outperform competitors. But to consistently innovate and deliver at the cutting edge of their industries, companies must execute regular rollouts of fresh application features, security updates, and other improvements. Central to these efforts is data. In particular, managing nonproduction environments and their requisite data needs is of paramount importance to better, more timely application development.

At Delphix, many of our customers look to improve Test Data Management (TDM) to streamline processes across development, test, and operations teams. Too often, antiquated technologies and complex processes stand in the way of fast access to high-quality test environments. But by helping customers better manage and deliver test data, Delphix:

- Accelerates test cycles with environment provisioning, refresh, and reset in just minutes.
- Eliminates the need for subsets or synthetic data, increasing testers’ ability to catch bugs and improving overall code quality.
- Protects sensitive data in nonproduction with automatic masking of test data.
- Dramatically reduces storage requirements for nonproduction environments.

As you read Gartner’s report, “Determining the ‘Right’ Number of Nonproduction Environments,” we encourage you to envision an enterprise strategy that takes advantage of these features. Delphix represents the next generation of test data management solutions, giving customers more flexibility in deciding what the “right” number of nonproduction environments actually is, while also allowing them to provision those environments with both ease and efficiency. All of these capabilities come together to help customers transform TDM and, ultimately, outpace the competition.

Sincerely,

Charles Moore
Director of Product Marketing, Delphix
Research from Gartner:

Determining the ‘Right’ Number of Nonproduction Environments

Production environments must be physically segregated from development and testing environments. Determining how many additional development and testing environments are “enough” is critical to quality and agility.

Overview
Adequate environmental separation helps development teams move code smoothly and consistently to the operations team. Do your environments meet your needs?

Key Findings
• There are distinctions in access, service levels and use cases that clearly differentiate development, test and staging environments.
• Additional logical environments may be needed to support particular processes.
• Risk assessment of production readiness should belong to the organization responsible for the operational service level.

Recommendations
• Establish, at a minimum, distinct development, test and staging environments.
• Access of development and test staff to staging environments should be minimal and closely supervised.
• Incorporate explicit regression test steps to deal with differences from one environment to the next.

What You Need to Know
The “right” number of development and test environments differs from organization to organization, and by project or application within organizations. A clear definition, supplemented by the use of a configuration management tool that automates the choices, effectively reduces code and process errors introduced in production.

Analysis
It’s a wise practice to separate development, testing and training activities from the real work of production. For example, in many situations where COBIT, the Sarbanes-Oxley Act, or the U.S. Food and Drug Administration (FDA) regulations are at issue, there is a legal or audit requirement to limit access to production application instances and their data. Testing and other nonoperational activities in production have regularly led to outages, data losses and other problems. Most organizations understand the risks of touching production, and are asking: How many additional environments are needed?

The answer is “just enough,” but that raises more questions than it answers. Here is a framework for determining the right number of environments, as well as some guidance on sizing and policy.

Minimum Number of Environments Needed
At a minimum, your organization should have three additional nonproduction environments for each major application:
• Development — owned and managed by a designee within the programming team; it is used to create, debug and test code
• System test — owned and managed by the application development organization; it is used to assess functional completeness and adherence to models and design
• Acceptance test (also called assurance, staging or preproduction) — typically owned by an organization outside of development (such as production control or quality assurance); it is used to assess compliance with end-user requirements and risk of release, and may, in some cases, also be used for system integration testing by the production group
The first two environments clearly should be under the control of the development team. Practices in these environments are intended to produce a functionally complete product. Development teams are typically measured by the completeness, internal quality, timeliness and budget compliance of their work. If the release cycle is slow, then organizations may attempt to do acceptance testing in the same environment used for system tests. Many development environments are heavily loaded with profiling and monitoring tools that deeply affect performance, making them ill-fitted for this dual usage. Often, these encumbrances and the need to restrict access to the assurance region require that separate system testing and acceptance testing environments be established.

The focus changes to risk assessment with the turnover of a release candidate to the acceptance or preproduction environment. In the preproduction stage, we see activities such as system regression testing, load and other performance testing, security scanning, service desk training, deployment and rollback testing, and user acceptance testing. These activities all aim to establish confidence that the release can be deployed without compromising production service levels. In other words, the objective is to establish that the risk of putting the candidate into production is acceptable. Increasingly, ownership of this stage belongs to a group reporting to whoever is responsible for the production service levels. This might be a quality assurance group or a turnover group reporting to operations. This stage often also provides the separation of duties required for audit compliance.

The acceptance environment may also be used for parts of the ITIL release process, such as system integration testing, checking the ability of the application to coexist with other infrastructures and applications that potentially will be sharing resources. Other uses can include testing of deployment and rollback scripts, and checking the various processes that will update configuration data, application monitoring and other operational artifacts.

The acceptance and system integration test environments may be separated for organizational reasons, but usually are similar in their configuration and service-level requirements.

**Do You Need Additional Environments?**

Address these questions to determine whether you need to add environments:

- *Do we need a separate environment to recreate production problems?* Typically, the answer to this is yes. Many production issues can’t be recreated in a programmer environment, because this environment is inherently unstable, and, even if it isn’t, it potentially has new or revised code that could disrupt an attempt to recreate production. Programmer access to production for emergency or hot fixes creates security risks, and is difficult to manage. The developer access has to be temporary and revoked immediately upon completion of the fix. Alternatively, some organizations grant temporary access to a separate fix-test environment or to the preproduction environment. Once the patch is ready, production control personnel can deploy it through standard mechanisms to production. If the need to recreate production problems happens infrequently (for example, once every six months), then it may be more appropriate to have a process in place for a temporary pseudo-production environment to be created with a few hours’ notice. This facility will sometimes also serve as a way to test application recovery procedures. This environment is typically owned by the programming staff.

- *Do we need a separate environment to train users?* For major new applications or revisions to established applications, a model office that enables the business to simulate a day (or other time period) or to test staff may be needed. Determine whether the need to train or do other business-driven work is great enough to require a separate environment. Typically, this environment is owned and operated by the business, and its code mimics production-level code.
• **Do we need a separate environment for each release team?** Many organizations have set up multiple release teams that overlap. Each team requires its own environment at the programmer level, with its code then migrating up the release chain to production. If a development organization has three overlapping release teams, each requiring its own environment, then the programmer environment should include three separate versions, with migration between the levels handled as each team moves its release to the system test environment. In this case, the number of added environments would be two, and the release manager is the owner of his or her team’s environment.

• **Do we need a separate environment for usability testing?** Most organizations aren’t sophisticated enough to do formal usability testing early in a project. Those that are need to use the appropriate programmer version to do early screen and process flow checks, or they need to set up an environment specifically for usability testing. The major driver for a formal and distinct environment is the instability of the programmer environment — that is, if the business is scheduled for a usability test, then the version needs to be available.

• **Do we need a separate certification environment?** In some large organizations, testing the interaction between major applications and the hardware, software and middleware that run them is critical. Best practices demand that infrastructure changes be managed independently of application changes. Therefore, the two classes must have separate environments or serialize the use of one environment between them.

• **Do we need an environment to act as a definitive library?** An organization’s release processes may call for a library of the source codes or executables that are actually in production. This is called a definitive software library (DSL) or a definitive media library (DML) in some of the ITIL processes.

• **What if I want to do continuous deployment?** Continuous deployment automates many test and promotion processes, but generally uses similar sets of environments, as we’ve discussed. Automation speeds the cycle, and may allow some staging to be enforced by the processes, rather than by establishing physically distinct environments.

**Virtual or Real?**

The principle reason for the physical separation of activities is to prevent compromises of the goals of each environment; therefore, production should be strictly separated. The test and preproduction areas can be more closely linked, but usually have different access lists. The more-fragile development and test regions are often physically segregated so that their restarts, builds and other activities don’t interfere with preproduction test schedules. The degree of physical separation must be higher when virtualization can’t successfully limit the interactions among systems. Successful virtualization enables more sharing of physical hardware. We see limited use of virtualized performance test regions (preproduction), unless production is also exploiting virtualization. In cases where production is virtualized, virtualized performance testing can be a good predictor of production performance, and becomes acceptable.
Other Concerns to Consider

Environment size. Sizing an environment is not an exact science. Generally, the programmer environments won’t exceed 20% to 40% of production, and the test environments won’t exceed 50% of production size. There may be times, however, where performance tests are critical, and full-size production databases will be contained at the test levels. It’s a trade-off: disk space for risk. Simulation tools can reduce size requirements in many contexts.

Environment migration. The software change management process controls migration patterns (that is, how a given set of code moves from one environment to another). Large organizations, or those with rapid release rates, will need to have an automated solution to control the migration process. Others do this manually. In either case, it’s critical that the migration path be carefully defined. In this day and age, with so many moving parts, most organizations should be moving toward automation of these migrations. It’s much easier to implement early and make the process progressively more complex than to delay and have to mount a major project to bring complex processes under control.

Complexity. The more environments you own, the more complex the process of making changes is, the more organizational challenges there are and the more difficult it is to be flexible to changes in the project, because shifting requirements or defects are discovered.

Service levels for nonproduction environments. If there are substantial differences between the preproduction and development and test environments, then appropriate testing to protect against the consequences of those differences needs to be in the plan. Support levels for preproduction approach those that are needed for production. If the production support interferes with the ability to maintain sufficient similarity among environments, then a separate support team will need to be created to focus on support of the preproduction, test and development environments.

Best practices for the transition process in the stages of testing (integration, acceptance and preproduction). Approval workflows are typically established that determine when a product can move from stage to stage. While these can be manual, they are best-implemented in change management or application life cycle management (ALM) tools, like IBM Rational ClearQuest, Microsoft Visual Studio Team System (VSTS), CollabNet TeamForge and Atlassian Jira. Criteria for promotion to the next stage should include the successful completion of some range of tests, and concurrence from the appropriate stakeholders.

Source: Gartner RAS Core Research Note G00210456, Jim Duggan, Matthew Hotle, Thomas E. Murphy 6 November 2014
Delphix Represents the Next Generation of TDM

An exercise in determining the “right” number of environments carries with it an unstated assumption that businesses must balance quantity of environments with availability of IT resources. After all, businesses are careful in meting out dev/test environments since they are constrained by expensive legacy infrastructure or the complex processes with which that infrastructure is marshaled.

But what if those constraints could be removed? Specifically, what would nonproduction landscapes look like if project teams could effortlessly deliver new environments, while consuming a fraction of the resources?

Delphix is a test data management solution that provides exactly that capability, delivering space-efficient dev/test environments on demand. Delphix software captures and stores a copy of production application data—including ongoing changes—then delivers virtual copies for development, testing, reporting, or other purposes, without any data duplication or movement. New environments can provisioned in 1/100th of the time using 1/10th of the infrastructure.

In contrast to traditional TDM solutions, Delphix provisions datasets that represent full copies of the production source rather than subsets and contain real data instead of synthetic data. Furthermore, Delphix can automatically apply masking to virtual datasets to secure sensitive information, while still preserving the value of the data for testing purposes.

Delphix Redefines the Notion of “Right-Sized” Nonproduction Landscapes

With Delphix, customers can meet and exceed the threshold requirements for providing sufficient environments for major applications. Virtual data copies consume less storage than even physical subsets while requiring essentially no labor to provision and refresh. This means project teams have the slack to scale to more expansive landscapes, as needed. Moreover, Delphix positions businesses to satisfy future state requirements for Continuous Delivery or DevOps, methodologies that demand elevated levels of environment availability and fluidity.

**FIGURE 1** Delphix delivers lightweight, virtual nonproduction environments on demand

Source: Delphix
**Threshold Requirements for Nonproduction:**

*Maximum Productivity Within Minimum Landscapes*

By virtualizing application data derived from production systems, Delphix allows customers to efficiently provision environments dedicated to development, testing, and validation. While this trio of environments represents table stakes—the minimum amount of IT workspace necessary to build software—Delphix also provides agility benefits enabling project teams to accelerate the software development life cycle (SDLC) within even the leanest of nonproduction landscapes.

For example, test environments leveraging virtual copies of data can be instantly refreshed. Rather than waiting hours for IT administrators to restore datasets from backup copies following destructive test runs, QA teams can perform data refreshes or resets in minutes, and as a self-service.

The key benefit stemming from these efficiency gains is a “shift left” in testing: application teams execute more productive test cycles, earlier in the SDLC when bugs are easier and less expensive to fix. Contrast this with TDM solutions based on data subsetting or synthetic data. Legacy approaches not only force testers to work with limited datasets that eventually erode application quality, but also introduce process overhead that slows them down.

Example: One F100 customer used Delphix to reduce test environment refresh time from 2 days to 20 minutes, resulting in an overall project schedule reduction of 30%.

**Objective Requirements for Nonproduction:**

*When More Is Better Than Less*

While separate environments for dev, test, and UAT may be sufficient for simple applications, today’s complex enterprise software typically demands additional flexibility. Small landscapes result in either serialization of effort that constrains project capacity, or resource sharing schemes that yield unpredictable, collision-prone testbeds.

Since Delphix removes the constraints that limit the number of environments that businesses deploy, it allows them to easily meet more aggressive requirements. For example, customers can move towards parallel development models in which release teams are given separate, multi-environment project lines. This allows organizations to execute multiple projects concurrently, even for complex applications—ERP chief among them—that prescribe rigid approaches to development and testing.

---

**FIGURE 2** Delphix enables “shift left” testing

<table>
<thead>
<tr>
<th>Before</th>
<th>With Delphix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5 HR</strong></td>
<td><strong>5 MIN</strong></td>
</tr>
<tr>
<td><strong>TEST</strong></td>
<td><strong>TEST</strong></td>
</tr>
<tr>
<td><strong>20 MIN</strong></td>
<td><strong>20 MIN</strong></td>
</tr>
<tr>
<td><strong>RESTORE</strong></td>
<td><strong>RESTORE</strong></td>
</tr>
<tr>
<td><strong>DATA1 SYS ADMIN</strong></td>
<td><strong>DATA1 SYS ADMIN</strong></td>
</tr>
</tbody>
</table>

**Current challenges:**
- Long hours to restore an application
- Setup, teardown, wait times
- Limit testing
- Errors, bugs more costly to fix later in the SDLC

**With Delphix:**
- 1000% increase in test cycles
- Reset, restore in minutes or less

Source: Delphix
Project teams can quickly provision environments dedicated to not only application development, but also training, integration testing, reporting, or break/fix processes. The efficiency with which Delphix creates new temporary environments increases IT responsiveness, lowers barriers to new pilot programs, and curbs organizational resistance to experimentation.

Example: One F500 manufacturer used Delphix to create 20 parallel dev/test project lines. This allowed them to increase the number of SAP projects executed per year from 2 to 11, while reducing their storage footprint by 90%.

Future State Requirements:
Towards DevOps and Continuous Delivery

The ability to spin up and tear down data environments in minutes instead of days lends itself to more flexible, iterative development practices. Delphix is a natural fit for organizations embracing DevOps or other methodologies that put a premium on fast environment standup and enhanced environment reproducibility.

Tools like Docker, Puppet, and Chef may be able to quickly stand up lightweight application instances with the right configurations, using minimal hardware and no operations time. They do little, however, to mobilize the test data needed to provision full environments at the same pace, and with the same level of automation.

Delphix helps customers meet these requirements by delivering copies of production data as rapidly as a containerization solution delivers code or configurations.

By filling this “data gap,” Delphix allows teams achieve new levels of self-sufficiency, facilitating non-production landscapes in which every developer or tester provisions their own individual environment, complete with secure test data. In addition, Delphix enables new models of collaboration, including the ability to version control, bookmark, and share test data with a few mouse-clicks. Application teams already have these tools for managing source code through the SDLC. Delphix empowers developers to manage data in the same way that they already manage source code.

Example: One F100 retailer used Delphix to expand from 5 shared nonproduction environments to 200, one for every developer and tester. This facilitated a transition to a Continuous Delivery model involving daily releases instead of monthly releases.

While businesses must take a measured approach to architecting their non-production environments, they should not be limited by legacy TDM solutions. Delphix redefines the notion of the “right-sized” nonproduction landscape. With virtualized data, Delphix gives customers more flexibility in sizing their landscape along with the agility and self-service features to maximize its value.

Source: Delphix
About Delphix

Business is increasingly digital, and applications enable digital business. The companies that manage their application portfolios well will be the disruptors and leaders of their industries. Data is the fuel for application projects, and Delphix transforms the way that organizations manage data for their application projects. Delphix software enables Data as a Service, within a firm’s on-premise, private, or public cloud infrastructure. With a unique focus on applications and databases, and features that exist nowhere else, the Delphix solution brings enterprise data into the modern age.