White Paper

DELPHIX™

A MODERN APPROACH TO TEST DATA MANAGEMENT

Building a comprehensive solution to solve today's biggest application development challenges

EXECUTIVE SUMMARY

Speed is a critical business imperative for all organizations, regardless of industry. The pace at which enterprises can bring new products and services to market determines their ability to differentiate from competitors and retain market share. Now more than ever, applications are at the center of this race. As enterprises look to deliver high-quality applications at the lowest possible cost, they need to build out a more agile application infrastructure—and that includes a robust and comprehensive test data management (TDM) strategy. Once viewed as a back-office function, TDM is now a critical business enabler for enterprise agility, security, and cost efficiency.

The increasing pace of software development presents new challenges. With the proliferation of DevOps, a heightened focus on automation, and requirements to secure data across global teams of employees and contractors, IT organizations must expand the charter of traditional TDM to meet the needs of today's development and testing teams. This white paper explores the top challenges that IT organizations face when managing test data, and highlights the top evaluative criteria to consider when implementing new technology solutions as part of a TDM strategy.

UNDERSTANDING TEST DATA CHALLENGES

Historically, application teams manufactured data for development and testing in a siloed, unstructured fashion. As the volume of application projects increased, many large IT organizations recognized the opportunity to gain economies of scale by consolidating TDM functions into a single group or department—enabling them to take advantage of innovative tools to create test data. As increasing centralization began to yield large efficiency gains, the scope of TDM was expanded to include the use of subsetting, and most recently, the use of masking to manipulate production data. However, the rise of development methodologies demanding fast, iterative release cycles has led to a new set of challenges.

ENVIRONMENT PROVISIONING IS A SLOW, MANUAL, AND HIGH-TOUCH PROCESS

Most IT organizations still use a request-fulfill model, in which developers and testers often find their requests queuing behind others. Because it takes significant time and effort to create a copy of test data, environments often take days or weeks to provision. Not only does this place a strain on operations teams, it also creates time sinks during test cycles, slowing the pace of application delivery and undermining competitive advantage. In the payments industry, for example, nimble technology companies with optimized processes can release applications weeks or months faster than traditional banks with slow IT ticketing systems.

SOFTWARE DEVELOPMENT TEAMS LACK HIGH-QUALITY DATA

Software development teams often lack access to the right test data. For example, depending on the release version being tested, developers require data sets as of a specific point in time. But all too often, they must work with stale copies of production data due to the complexity of setting up a new test bed. This can result in lost productivity due to time spent resolving data-related issues. According to recent research, developers spend more than 20% of their time on test-data related activities.[1]

1 "Transforming Test Data Management for Increased Business Value." Cognizant 20-20 Insights, March 2013.

DATA MASKING IS INCREASINGLY IMPORTANT, BUT ADDS FRICTION TO RELEASE CYCLES

For many applications, such as those processing credit card numbers, patient records, or other sensitive information, data masking is critical to ensuring regulatory compliance and protecting against data breach. According to the Ponemon Institute, the cost of a data breach—including the costs of remediation, customer churn, and other losses—averages $3.8 million.[2] However, masking sensitive data often adds operational overhead; an end-to-end masking process may take an entire week, which can prolong test cycles.

TEST DATA REQUIREMENTS AND STORAGE COSTS ARE CONTINUALLY RISING

IT organizations create multiple, redundant copies of test data, resulting in inefficient use of storage. To meet concurrent demands within the confines of storage, operations teams must coordinate test data availability across multiple teams, applications, and release versions. As a result, development teams often contend for limited, shared environments, resulting in the serialization of critical application projects.

TOP CONSIDERATIONS FOR A TEST DATA MANAGEMENT SOLUTION

To address these challenges, IT organizations need to adopt the right tools and processes to efficiently make the right test data available to project teams. A comprehensive approach should seek to improve TDM in each of the following areas:

- DATA DISTRIBUTION: reducing the time to operationalize test data
- DATA QUALITY: fulfilling requirements for high-fidelity test data
- DATA SECURITY: minimizing security risks without compromising agility
- INFRASTRUCTURE COSTS: lowering the costs of storing and archiving test data

The following sections highlight the top evaluative criteria in each of these four areas.

DATA DISTRIBUTION

Making a copy of production data available to a downstream testing environment is often a time-consuming, labor-intensive process involving multiple handoffs between teams. The end-to-end process usually lags demand; at a typical IT organization, delivering a new copy of production data to a non-production environment takes days, weeks, or months in some cases.

Organizations looking to improve TDM must build a solution that streamlines this process and creates a path towards fast, repeatable data delivery. Specifically, test data managers should look for solutions that feature:

- AUTOMATION: modern software toolsets already include technologies to automate build processes, source code management, and regression testing. However, organizations often lack equivalent tools for

2 "2015 Cost of Data Breach Study: Global Analysis." Ponemon Institute, May 2015.

delivering copies of test data with the same level of effortlessness. A streamlined TDM approach must eliminate manual processes—for example, target database initialization, configuration steps, and validation checks—providing a low-touch approach to standing up new data environments.

- TOOLSET INTEGRATION: an efficient TDM approach unites the heterogeneous set of technologies that interact with test datasets along the delivery pipeline, including masking, subsetting, and synthetic data creation. This requires both compatibility across tools and exposed APIs, or other clear integration mechanisms. A factory-like approach to TDM that combines tools into a cohesive unit allows for greater levels of automation and eliminates handoffs between different teams.

- SELF SERVICE: by putting sufficient levels of automation and toolset integration in place, test data delivery can be executed via self service, directly by end users. Instead of relying on IT ticketing systems, end users should take advantage of interfaces purpose-built for their needs. Self-service capabilities should extend not just to data delivery, but also to control over test data. For example, developers or testers should be able to bookmark and reset, archive, or share copies of test data without involving operations teams.

A well-orchestrated approach to TDM has the potential to transform the overall application development process. Slashing the wait time for data means testers can execute more test cases earlier in the software development lifecycle (SDLC), enabling them to identify defects when they are easier and less expensive to fix.

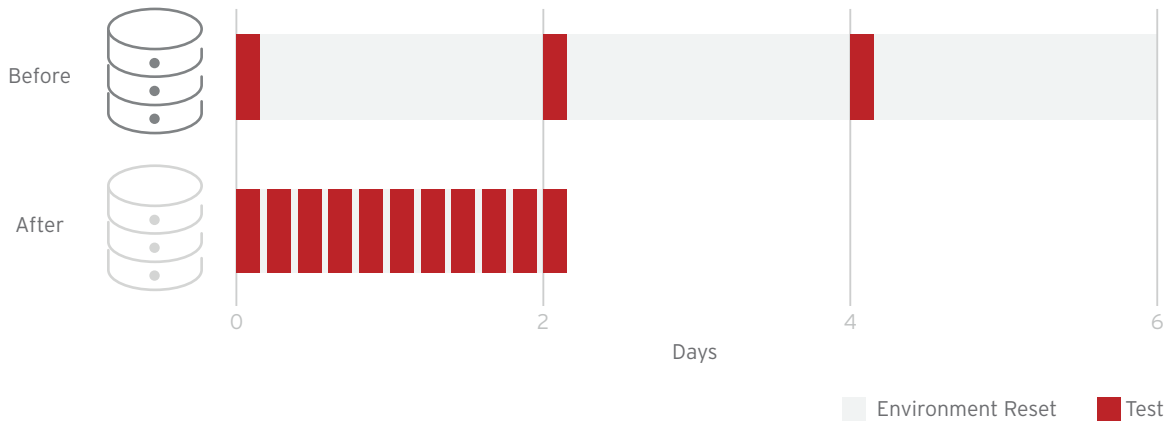

Figure 1: Testing in a traditional scenario (A) vs. a scenario with an optimized TDM approach (B).

DATA QUALITY

TDM teams go through great efforts to make the right types of test data—such as masked production data or synthetic datasets—available to software development teams. As TDM teams balance requirements for different types of test data, they must also ensure data quality is preserved across three key dimensions:

- DATA AGE: due to the time and effort required to prepare test data, operations teams are often unable to meet a number of ticket requests. As a result, data often becomes stale in non-production, which can impact the quality of testing and result in costly, late-stage errors. A TDM approach should aim to reduce the time it takes to refresh from a gold copy, making the latest test data more accessible. In addition, the latest production data should be readily available in minutes in the event that it is needed for triage.

- DATA ACCURACY: a TDM process can become challenging when multiple datasets are required as of a specific point-in-time for systems integration testing. For instance, testing a procure-to-pay process might require that data is federated across CRM, inventory management, and financial applications. A TDM approach should allow for multiple datasets to be provisioned to the same point in time and simultaneously reset to quickly validate complicated end-to-end functional testing scenarios.

- DATA SIZE: due to storage constraints, developers must often work with subsets of data, which aren't likely to satisfy all functional testing requirements. The use of subsets can result in missed test case outliers, which can paradoxically increase rather than decrease project costs due to data-related errors. In an optimized strategy, full-size test data copies can be provisioned in a fraction of the space of subsets by sharing common data blocks across copies. As a result, TDM teams can reduce the operational costs of subsetting—both in terms of data preparation and error resolution—by reducing the need to subset data as frequently.

The end-to-end process of manipulating and operationalizing test data introduces complexities that often undermine data quality. As demonstrated in the following case study, implementing the proper TDM tools can optimize this process, resulting in massive improvements in data quality across all three dimensions.

CASE STUDY

Consider the following TDM process at a large health plan provider:

1. First, a developer requests a copy of masked production data.
2. Once approved, a DBA creates a backup of production and then transfers and imports the backup copy to a staging server.
3. A test data manager validates the rowcount, any new PHI fields, and data structures.
4. The test data manager updates the subsetting artifacts, creates a subset, and executes the masking process.
5. The test data manager validates the row count, sends out an email update, and performs unit testing.
6. The DBA exports the masked data, transfers it to non-production, and updates the gold copy.
7. The DBA performs a backup and restore operation into the target Dev environment.
8. A second backup and restore process is performed to load data into a QA environment.

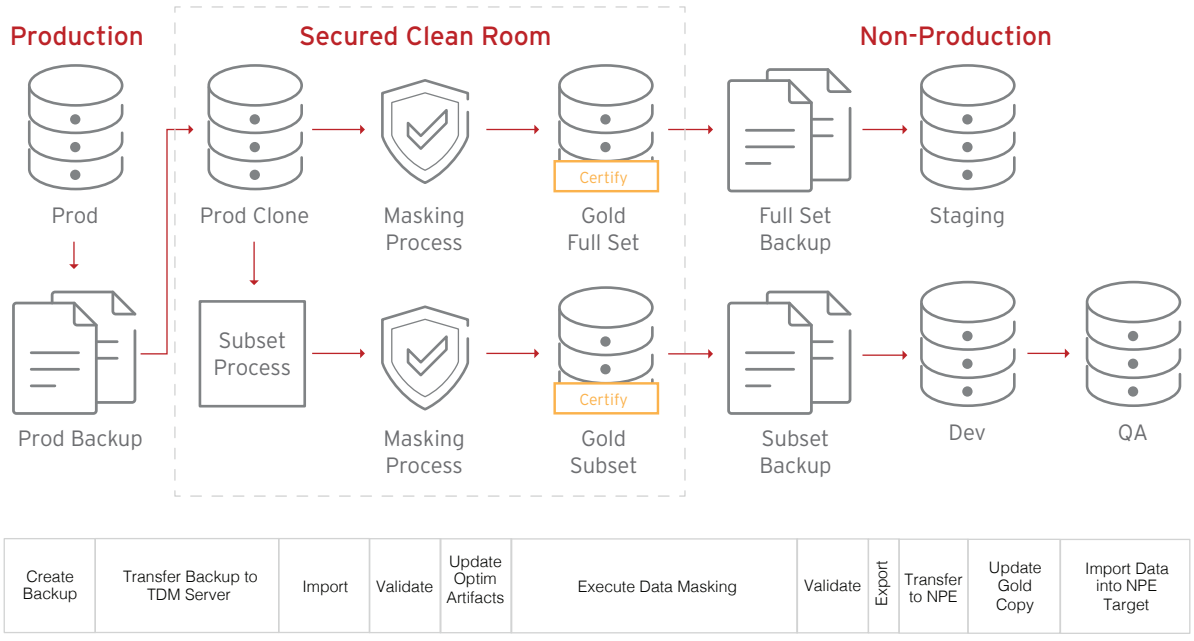

Figure 2: Example of a test data management process at a large health plan provider.

End-to-end, this process takes seven days. In an optimized process, masked data can be prepared in less than two days:

1. A TDM solution automatically and non-disruptively remains in sync with production, providing continuous access to the latest data and eliminating the need to perform a backup.
2. An admin restores data to the clean room in minutes.
3. An admin profiles and automatically assigns repeatable masking algorithms. If required, an admin subsets the data beforehand.
4. After masking is complete, the admin tests changes and validates referential integrity with the ability to quickly rollback to the initial unmasked state.
5. An admin efficiently and securely replicates masked data to non-production.
6. Instead of updating or replacing the existing gold copy, it remains as an archive in a centralized repository, where it is compressed to a fraction of the size of production.
7. Developers access masked data via self service in minutes instead of performing a manual backup and restore process.
8. QA engineers branch their own copies of development and begin testing in minutes.

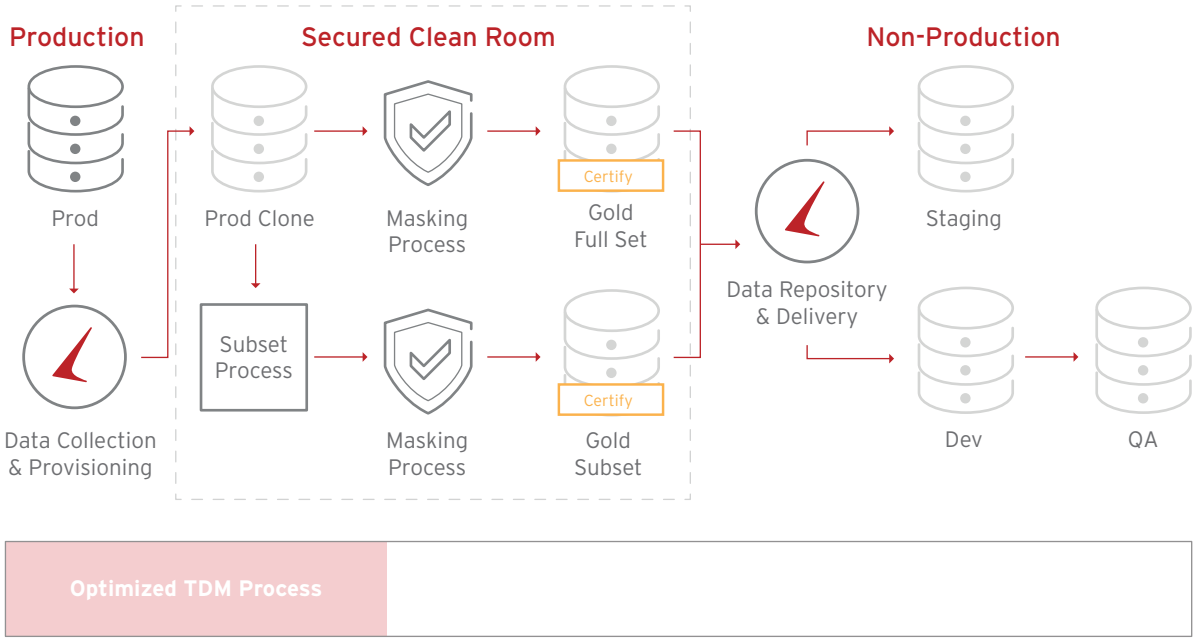

Figure 3: Example of an optimized test data management process.

DATA SECURITY

Masking tools have emerged as the de facto standard for protecting test data. By irreversibly replacing sensitive data with fictitious yet realistic values, masking can ensure regulatory compliance and completely neutralize the risk of data breach in test environments. But to make masking practical and effective, organizations must consider the following requirements:

- END-TO-END REPEATABILITY: many organizations fail to adequately mask test data because added process overhead deters them from applying masking everywhere they should. However, solutions with out-of-the-box capabilities to orchestrate a complete masking process—identifying sensitive data, applying masking to that data, and then auditing the resulting test dataset—can minimize coordination and configuration efforts.

- NO NEED FOR DEVELOPMENT EXPERTISE: organizations should look for lightweight masking tools that can be set up without scripting or specialized development expertise. Tools with fast, predefined masking algorithms, for example, can dramatically reduce the complexity and resource requirements that stand in the way of consistently applying masking.

- INTEGRATED MASKING AND DISTRIBUTION: masking processes should be tightly coupled with a data delivery mechanism. Instead of relying on separate workflows for masked data and unmasked data, an integrated approach lends itself to greater standardization of masking as a security precaution, and helps ensure that masked data can be delivered wherever it's needed. For example, many organizations will benefit from an approach that allows them to mask data in a secure zone and then easily deliver that secure data to targets in non-production environments, including those in offsite data centers or in private or public clouds.

While data masking tools eliminate the risk of exposing sensitive information in testing environments—which can represent the majority of the surface area of risk—organizations must integrate them into TDM workflows without compromising speed and simplicity objectives.

INFRASTRUCTURE COSTS

With the rapid proliferation of test data, TDM teams must build a toolset that maximizes the efficient use of infrastructure resources. Specifically, a TDM toolset should meet the following criteria:

- DATA CONSOLIDATION: it is not uncommon for organizations to maintain non-production environments in which 90% of the data is redundant. A TDM approach should aim to curb storage costs by sharing common data across environments—including those used not only for testing, but also development, reporting, production support, and other use cases.

- DATA ARCHIVING: according to Bloor Research, as many as 6 versions of a gold copy should be archived for testing different versions of an application.[3] A TDM approach should make it feasible to maintain libraries of test data by optimizing storage use and enabling fast retrieval. Data libraries should also be automatically version-controlled in the same way that tools like Git exist for code versioning.

- ENVIRONMENT UTILIZATION: at most IT organizations, projects are serialized due to contention for environments. Paradoxically, at the same time, environments are often underutilized due to the time to populate an environment with new test data. A TDM solution should decouple data from blocks of computing resources through intelligent use of "bookmarking." Bookmarked datasets can be loaded into environments on demand, making it easier for developers and testers to effectively timeshare environments. As a result, an optimized TDM strategy can eliminate contention while achieving up to 50% higher utilization of environments.

By optimizing storage and improving the elasticity of test data, TDM teams can move to a model where infrastructure is no longer a limiting cost factor in application development.

BUILDING A COMPREHENSIVE TOOLSET

No single technology exists that fulfills all TDM requirements. Rather, teams must build an integrated solution that provides all the data types required to meet a diverse set of testing needs. Once test data requirements have been identified, a successful TDM approach should aim to improve the distribution, quality, security, and cost of various types of test data. Table 1 examines these criteria across the most common types of test data.

- PRODUCTION DATA provides the most complete test coverage, but it usually comes at the expense of agility and storage costs. For some applications, it can also mean exposing sensitive data.

- SUBSETS OF PRODUCTION DATA are significantly more agile than full copies. They can provide some savings on hardware, CPU, and licensing costs, but it can be difficult to achieve sufficient test coverage. Sensitive data is often still exposed using subsets.

- MASKED PRODUCTION DATA (either full sets or subsets) makes it possible for development teams to use real data without introducing unsafe levels of risk. However, masking processes can drag on data delivery. Also, masking requires staging environments with additional storage and staff to ensure referential integrity after data is transformed.

- SYNTHETIC DATA circumvents security issues, but the space savings are limited. While synthetic data might be required to test new features, this is only a relatively small percentage of test cases. If performed manually, creating test data is also prone to human error and requires an in-depth understanding of data relationships both within the database schema or file system, as well as those implicit in the data itself.

3 "Exploring Successful Approaches to Test Data Management". Bloor Research, July 2012.

Test Data Type vs. Criteria	Distribution	Quality	Security	Cost
Production Data	Slow, manual access	Good test coverage	Sensitive data at risk	High consumption of storage, CPU, and licenses
Subset of Production	Less time to provision than full copies	Missed test case outliers	Sensitive data at risk	Some storage, CPU, and license savings
Masked Data (Full or Subset)	Extended SLAs for masked data	Must ensure referential integrity	Improved data privacy and security	Requires masking software or custom scripting and staging server
Synthetic Data	Manual process	Limited to a small percentage of test requirements	Data de-identification not required	Limited storage savings

Fails to meet criteria Partially meets criteria Meets criteria

Table 1: Evaluating Common Required Test Data Types

One technology that can improve the distribution, quality, security, and cost of all-of-the-above types of test data is data virtualization. Virtual data—which can be a copy of any data stored in a relational database or file system—shares common data blocks across copies, enabling space savings of up to 90%. Virtual data also enables rapid provisioning of test data, as of any point in time, via self service. As a result, TDM teams can accelerate the rate at which they can not only manipulate test data, but also operationalize the rollout of test data to software development teams.

CASE STUDY

One Fortune 500 financial services institution investigated the use of virtual data for the development and testing of an online platform that provides market insights to clients and enables them to make smarter financial decisions. The investigation was triggered by massive platform growth: over the span of a few years, financial data had doubled, usage had tripled, and feature development effort had quadrupled. IT was struggling to keep pace with the exploding storage costs and missed several releases due to slow environment provisioning. Moreover, a large percentage of bugs was discovered in late stage user-acceptance testing, which risked impacting customer experience.

For both Oracle and MS SQL production data sources, the firm's IT organization implemented a data virtualization technology with built-in masking. After deploying the solution in less than a few weeks, the results were immediate. For instance, rather than waiting a full day for a DBA team to restore an environment after a 20-minute test run, QA engineers leveraged secure virtual data to initiate a 10-minute reset process that brings the environment back to a bookmarked state. Less waiting enabled QA teams to execute more test cycles earlier in the SDLC—a "shift left" in testing. Ultimately, this led QA teams to discover and resolve errors when they were easier and less expensive to fix. The firm estimated that they reduced overall defect rates by 60 percent and improved productivity across 800+ developers and testers by 25 percent. They also dramatically reduced storage requirements by almost 200 TB, enabling them to accommodate massive platform growth without expanding their existing infrastructure.

ABOUT DELPHIX

Delphix enables companies—including over 30% of the Fortune 100—to complete application projects in half the time while using half the infrastructure by delivering virtual data on demand. Using built-in masking capabilities, IT organizations can improve data security while simultaneously driving dramatic productivity increases. Delphix is headquartered in Menlo Park, CA with offices globally. For more information, please visit www.delphix.com.

A Modern Approach to Test Data Management
May 2016

For more information, visit **www.delphix.com**

The Delphix Website also provides the latest product updates.
If you have comments about this documentation, submit your feedback to: **help@delphix.com**

Delphix Corp.
275 Middlefield Road, Suite 210
Menlo Park, CA 94025