PARASOFT

TECHNICAL WHITEPAPER

On-Demand Test Environments for Development & QA

The number and complexity of test environments that need to be accessed for an organization's various projects has become overwhelming. The challenge of providing rapid access to the necessary test environments with the appropriate configurations has escalated for a number of reasons.

- When testing transactions across today's composite applications, it's virtually impossible to avoid interaction with dependent applications that are evolving, unavailable, or difficult to access for testing.
- Organizations leveraging more iterative or Agile development methods have multiple projects evolving dependent systems with different timelines. Rapidly evolving systems creates significant complexity for staged test environments.
- Server (hardware) virtualization can alleviate some of these dependencies, but significant time and costs are still required to acquire, set up, manage, and constantly reconfigure them for various testing needs.
- Some dependent applications, such as third-party applications and mainframes, aren't feasible to virtualize or stage. Teams can only access them during tightly controlled time frames with little time or freedom to configure them for the necessary testing needs.
- Test plans commonly require that tests be exercised against a number of different scenarios (expected responses, unexpected responses, delays, and so on). Accurately assessing the application under test's (AUT) functionality typically requires testing against a myriad permutations of dependent component behaviors.

Coordinating convenient and simultaneous access to the various staged, virtualized, and live dependent components for just one test environment is a daunting task. Provisioning all the different environments required across an organization or a division—each with the necessary configuration—is often overwhelming. And when the cost and effort required to create all these test environments outweighs the perceived benefit of performing that testing, testing is shortchanged. The result is compromised quality and missed deadlines.

THE MISSING PIECE: SERVICE VIRTUALIZATION

Service virtualization alleviates these pain points by enabling easy, rapid creation of the various test environments needed with dependencies eliminated. It promotes the creation of what are essentially "disposable" test environments: environments without any impact or cost to the organization or to the other teams vying for their own test environments.

Service virtualization steps in where traditional server/hardware virtualization leaves off. When server virtualization is not feasible, service virtualization enables you to emulate the behavior of dependent applications. Unlike stubs or mocks, virtual assets are simple to create, represent a broad range of realistic behavior, and are easy to update as the dependent applications evolve.

Even when server virtualization is possible, it's not always the most practical approach in the context of a test environment. Typically, only a small percentage of the dependent application's functionality is involved in the test scenario. Yet, this low utilization does little to diminish the cost or time required to acquire these applications, configure them, and make the adjustments needed for each requested test environment.

With service virtualization, you don't have to virtualize an entire system when you need to access only a fraction of its available functionality. Instead, "virtual assets" emulate the AUT's specific interactions with the dependent applications, then stand in for the actual components in the test environment. The result is not only reduced costs, but also faster, more flexible access to the exact behavior that the team needs to test against.

With the constrained components replaced by virtual assets, the feat of accessing all of the required system components at a single, convenient time is significantly simplified.

RAPID CREATION, MANAGEMENT, & PROVISIONING OF VIRTUAL ASSETS

Parasoft's unique integration of service virtualization with test environment management helps teams accelerate and simplify the process of creating the diverse and complex environments required for thorough testing. It enables organizations to give development, QA, and performance test engineers easy access to the very specific combinations and configurations of virtual assets that they need for different test environments.

Parasoft Virtualize helps development and QA teams create and access any environment needed to develop or test an application. Through recording, teams can capture the necessary system behavior of dependent applications either by leveraging Parasoft proxy recording or converting packet sniffing technologies like Wireshark or Fiddler. Those artifacts can then be converted into reusable virtual assets in order to execute complex test scenarios. These easily configurable virtual assets then stand in place of the actual dependent applications, giving team members the freedom to perform their expected development and testing tasks whenever they want, as extensively as they want.

Through Parasoft's continuous testing platform (CTP), developers and testers can rapidly set up the specific configurations and combinations of virtual assets that they want to apply in any given test environment. From a graphical system map, users can:

- Specify whether each of the dependent applications could be replaced with a virtual asset.
- Specify what behavior that virtual asset should exhibit in the current case if the above point is yes.
- Rapidly define and provision test environments with all necessary permutations of dependent application behaviors and start testing immediately.

Virtual assets remove constraints associated with dependent applications.

For easy recording and virtualization of dependent application behavior, Parasoft Virtualize captures live dependent application traffic connected to the application under test. After a simple one-time setup, Parasoft Virtualize monitors behavior at any number of endpoints on demand, then represents this behavior in "virtual assets" that can stand in for the actual constrained components. If a downstream dependency is incomplete or unavailable, a service definition can be used to generate the virtual version. Beyond web services, Parasoft Virtualize captures the behavior of databases, messaging systems, middleware, mainframes, ESBs, legacy systems, and more.

Working in an environment where constrained components are replaced with virtual assets, developers and testers gain the freedom to perform their expected tasks whenever they want, as extensively as they want. For teams working with limited access components, this means no more struggling with inconveniently scheduled access times (such as 2 a.m. on Saturday morning), no time wasted setting the test environment to the desired state, and no access fees incurred. For teams trying to work in parallel, this means no more deadlocks or delaying testing until late in the process when defects are exponentially more difficult and costly to fix.

Virtual assets are available 24/7 for unit, functional, and performance testing—automated or manual. They can be leveraged by any test environment and deployed as a part of any CI build system such as Jenkins, Bamboo, Microsoft Azure DevOps, TeamCity, GitHub, GitLab, and so on.

Virtual assets are extremely flexible and easy to configure for different testing needs.

A staged testing environment often lacks access to the breadth of dependent applications associated with an AUT. This constraint is compounded for performance testers because staged testing environments traditionally lack the computing resources required to deliver realistic performance scenarios. More commonly, you have a limited degree of control over how the component behaves and can test only a limited range of behavior conditions (for example, the current actual response time, but not peak response time or failures). This leads to incomplete testing and exposure to significant business risks.

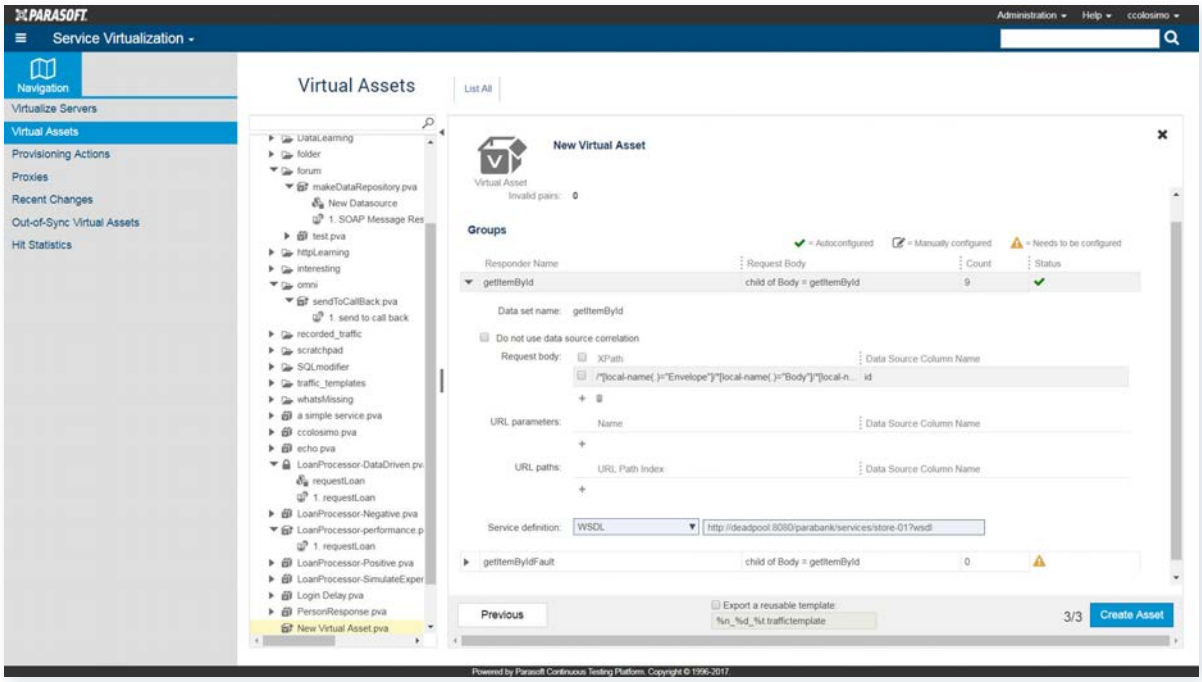

Figure 1:
Create, configure, and manage virtual assets with the service virtualization browser interface.

With Parasoft Virtualize, the simple user interface allows you to easily configure virtual assets to reproduce the specific conditions critical for completing dev/test tasks. For example, you can configure various error, failure, and performance conditions that are difficult to reproduce or rapidly configure with real systems. By adding data sources and providing conditional criteria, you can tune the virtualized asset to function as expected—or as unexpected (for negative testing). This helps the team validate the full range of system behavior, including its ability to respond correctly (or at least fail gracefully) in various exceptional situations.

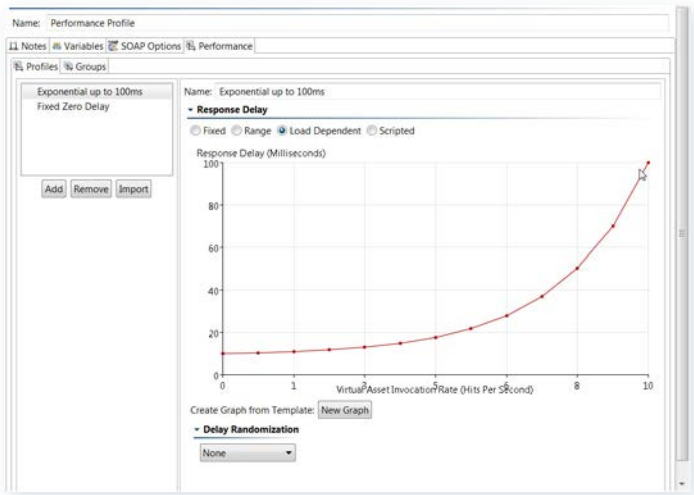

Figure 2:
Select different profiles to simulate the performance of a service under load.

In this way, your team gains complete control over a virtual asset that represents a dependent application for which you have very little control in the real world. In this environment, you can direct the behavior, data scenarios, and performance profile of the system that's your dependency. And you can set up multiple behavior profiles that can be activated or deactivated with the click of a button.

Virtual assets are easy to configure, provision, and adjust in a complete test environment.

Controlling the behavior of a single virtual asset is an important step in giving developers and testers the freedom to access the exact dependent application behavior that they need to test against.

Taking this to the next level, Parasoft allows developers and testers to instantly define and provision test environments with all necessary permutations of dependent application behaviors. This means that a user can see all the dependent components involved in a test environment, graphically set each component to the desired state, then click a button to make the specified combination of states immediately available for testing.

A complete test environment, including virtual assets, can be instantly provisioned to meet a very specific need. Then be quickly replaced with other environment configurations needed for thorough testing.

Let's take a quick look at how this is achieved with Parasoft CTP's environment manager.

Configure On-Demand Test Environment Provisioning

First, some basic configuration defines what elements and options should be made available for on-demand test environment provisioning. Follow these steps.

1. **Model the system architecture.**
   The foundation of all test environments is the complete system map, which is modeled by someone familiar with the system. This provides a high-level view of all dependent systems associated with the AUT, as well as a centralized, easily searchable repository for each component's configuration details.

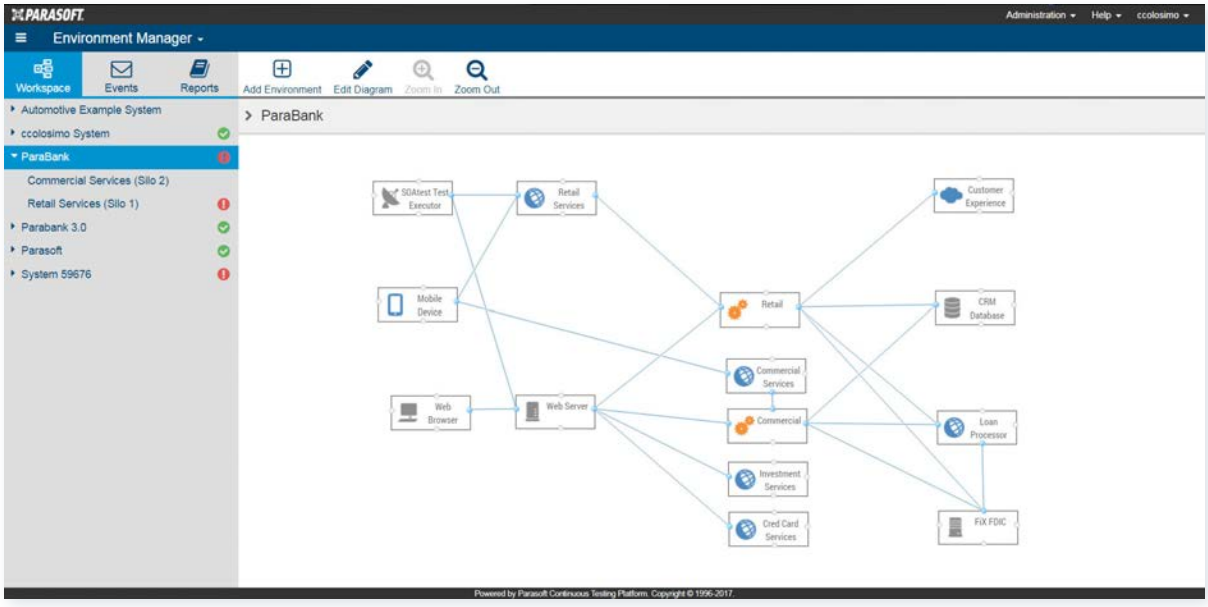

Figure 3:
View the environment model diagram of connected services and their relationships, and provision to different states (live, virtual, and so on).

2. **Define the available component states.**
   Next, you define the different states to which each component can be set in a test environment. For instance, you could configure options to have traffic routed to the actual component or to a number of different virtual assets—each of which might provide different responses, performance profiles, data sets, and so forth. You could also configure any number and scope of test cases to execute upon environment provisioning. This provides a simple way to approach test automation.

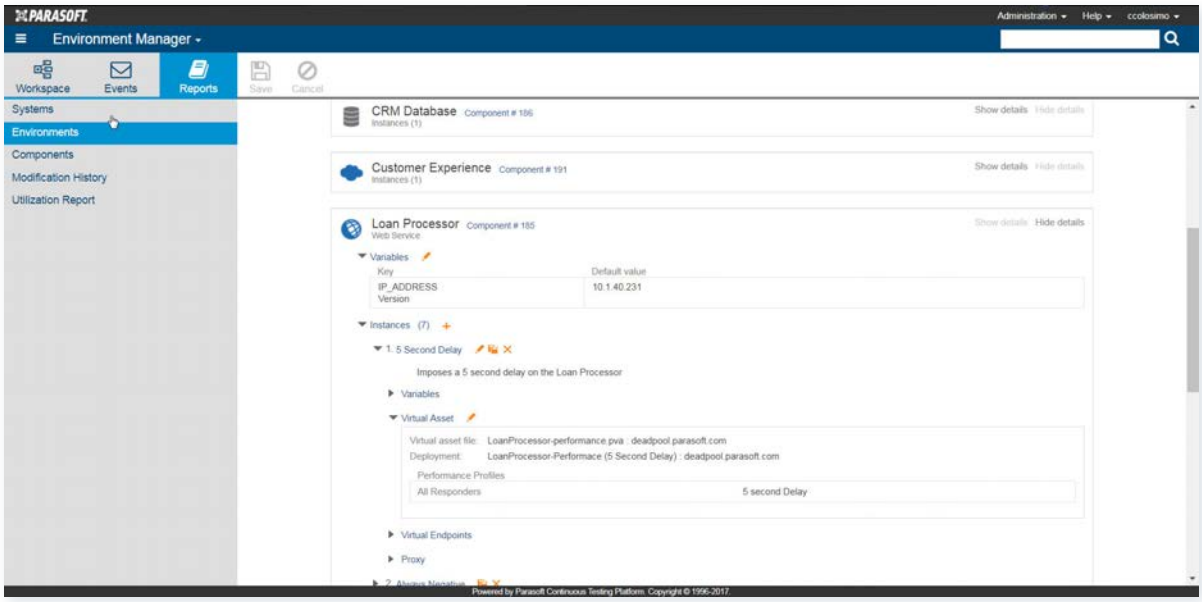

Figure 4: Associate virtual assets and proxy states with environment instances to establish configurations for provisioning.

Self-Provision Environments for Access

From the blueprints established via this basic foundation, developers and testers can then self-provision the various environments they need to access.

1. **Specify a complete environment instance.**
   Using the system definition as a template, developers and testers zero in on the subset of components that are important to a given test environment. They then use graphical controls to set each of the relevant components to the state required for a specific test environment they need to work with.

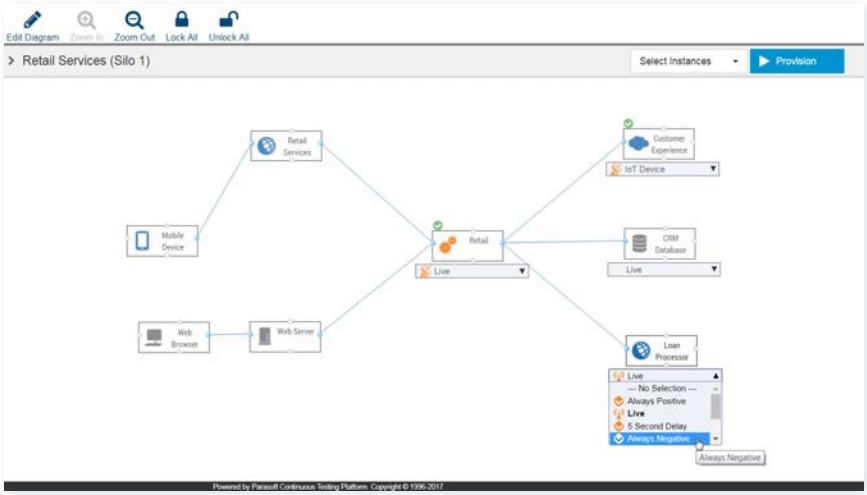

Figure 5: Configure a mix of live and virtual components and provision the resulting environment to enable testing to continue without delays.

Developers and testers can then adjust the settings for a different environment configuration (for instance, by changing the state of a few components), create a second environment instance, make additional adjustments, create another instance, and so on. In this way, they can rapidly define the different environments they need to access.

2. **Provision an environment instance to activate it.**
   The developer or tester can provision a defined "environment instance" immediately or store it for later use. With a single click, Parasoft Virtualize applies all the various configuration actions needed to set up the dependent components involved in that test environment. This allows testing to start immediately.

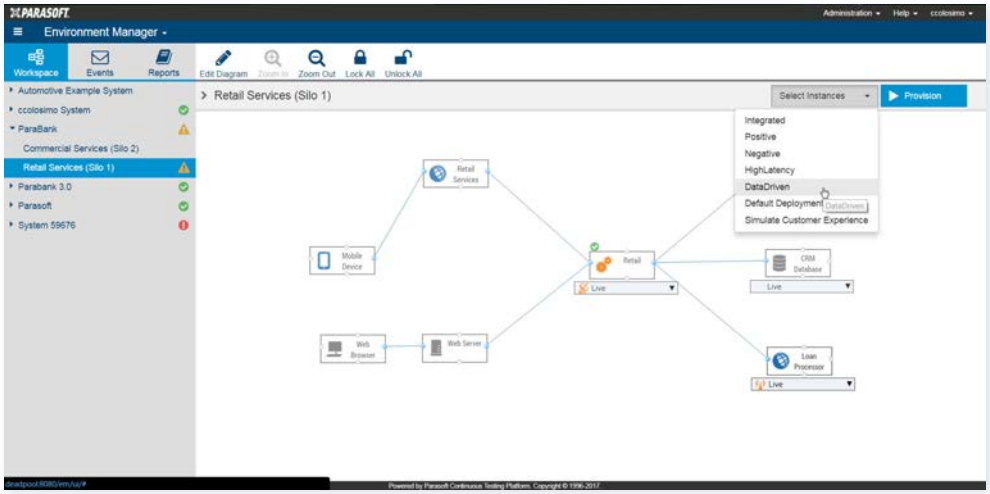

Figure 6: Choose a pre-defined configuration from select instances to quickly provision your test environment.

3. **Deploy and destroy environments.**
   Now that you have the stored various environment instances into the platform, those configurations can be deployed to remote virtualization servers on-demand or as a part of your build system. The environment configuration is a template that holds all of the connection, virtual asset, test case and data. Your build system can call out to Parasoft CTP and deploy this configuration dynamically. This allows you to spin off various versions of your required test environments as needed to support your testing. Then, when testing is complete you can tear them down to save valuable hardware resources.

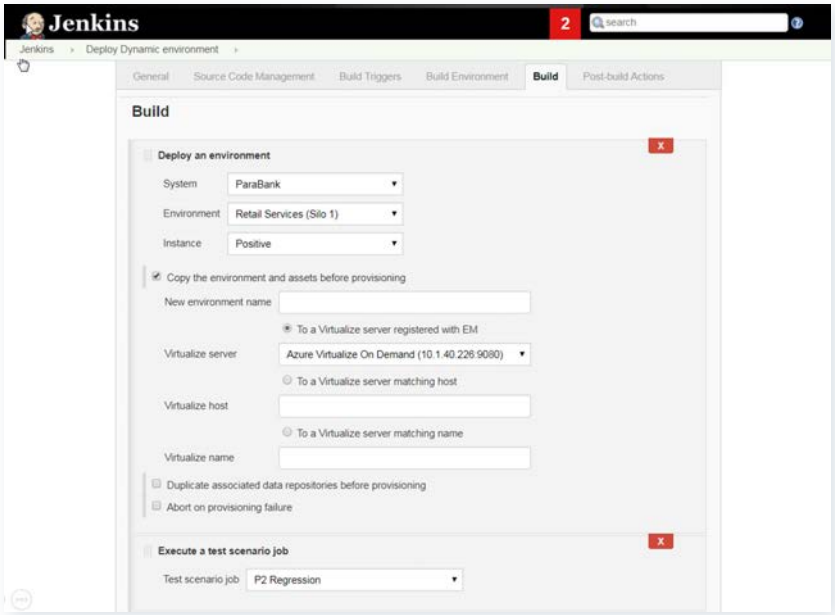

Figure 7: Integrate the deployment of your test environment into your CI/CD pipeline using your preferred build system.

SUMMARY: "DISPOSABLE" TEST ENVIRONMENTS

Parasoft enables developers and testers to easily configure a test environment from existing components to meet a specific test requirement, then rapidly reconfigure that environment with a different set of components to meet a different test requirement. In this sense, the test environments are essentially "disposable." Once the team is finished working with one test environment, they can simply adjust a few different configuration options and move on to another one in a matter of minutes. Scenario or "what if" testing can be performed as needed, on demand. Since the endpoints are an aggregation of simulated dependent system behavior, running tests against this environment doesn't impose any external business risks.

This approach also eliminates the pervasive problem of teams overriding each other's work. When multiple teams are working in a traditional test environment, all too often one team spends considerable time configuring the environment to meet their specific needs, then another team overrides this setup as they try to adjust the environment to suit their own needs. When each team can easily create their own "disposable" test environments tailored for their own needs, the potential for such conflicts is eliminated. The ability to test in parallel ensures that testing can continue unhindered, enabling earlier completion and feedback.

TAKE THE NEXT STEP

Learn more about creating, deploying, and managing virtual test environments on demand—anytime, anywhere. Talk to one of our experts today.

ABOUT PARASOFT

Parasoft helps organizations continuously deliver quality software with its market-proven, integrated suite of automated software testing tools. Supporting the embedded, enterprise, and IoT markets, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award winning reporting and analytics dashboard delivers a centralized view of quality enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives — security, safety-critical, Agile, DevOps, and continuous testing.