



Composite applications

Break the delivery logjam with
Hewlett Packard Enterprise Service Virtualization

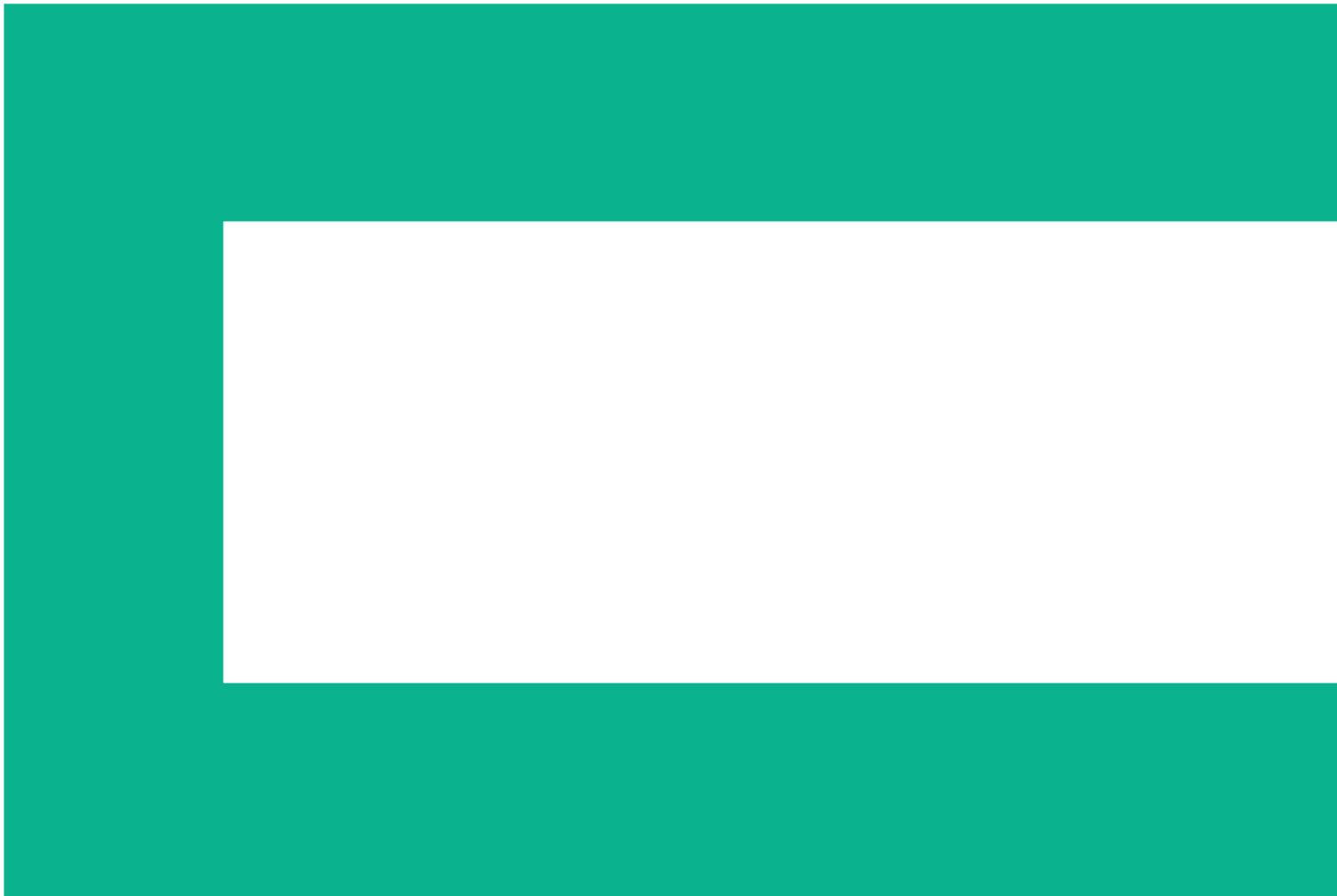


Table of contents

- 2 Executive summary**
 - Speed's the name of the game but it's not a "hall pass" for quality
- 2 Introduction**
 - Enter composite application architecture as a catalyst for speed
- 2 The powerful role of HPE Service Virtualization**
 - Scenario #1: Building a composite application? What if all the components aren't ready at the same time?
 - Scenario #2: Testing when the composite application contains a call to a third-party cloud service
 - Scenario #3: Developing and testing a multi-step business process that must access a constrained mainframe application in production
 - Scenario #4: Achieving performance testing of a composite application when a service is constrained
- 8 HPE Software Services**

Executive summary

Speed's the name of the game but it's not a "hall pass" for quality

In today's increasingly competitive and impatient world, success in delivering software of value hinges on speed. First, to market often means first to revenue or first to adoption. And, in order to keep pace with business and user demands, development and testing teams are under increased, almost unrelenting pressure to deliver software updates much more frequently than ever before. However, with great speed comes great responsibility. Being first and fastest does not equate to value when the delivered solution does not perform, lacks in user experience or responsiveness. Risks of delivering poor quality software can even affect the brand and business viability, especially if the issue pertains to security breaches or data privacy. Applications teams must move quickly but must also balance the need to keep functional quality, performance, and application security levels aligned even more closely with demanding compliance and business expectations. Add to the urgency the fact that impact to business continuity and brand is rising even more as increasingly business operations and customer engagement takes place online and through mobile devices. Imagine the risk to business if vital customer-facing applications such as online bill pay or customer health records access is subjected to quality, performance, or security failures. The dilemma facing organizations today is how to be agile, but not be fragile with application delivery.

Introduction

Enter composite application architecture as a catalyst for speed

Adopting composite application architecture is becoming a proven path to speed of application delivery. Building composite applications or redesigning existing applications into flexible components promises easier, faster application update and delivery, but can challenge even the most nimble and skilled development and testing organizations. Composite applications do not fly in formation; often certain components become available before others or other components needed to be a part of the integrated application are constrained in production. On top of the push to componentize, organizations are also looking to scale and focus on building what matters most to manage their restricted resources against an ever-increasing backlog of customer and business requirements. This leads them to acquire application components or services from cloud service providers instead of building in-house and then integrating these cloud components with the rest of the application. However, this is often easier said than done as developing against and testing against a third-party provided cloud service is often time consuming, introduces unplanned access costs, or may not be possible at all for security reasons.

The powerful role of HPE Service Virtualization

What you have as a result is an uneasy tension between the agility benefits of adopting composite application architecture and leveraging cloud-based services and the ability to develop and test against these services in a timely manner. This white paper explores these challenges, the obstacles to development and testing with cloud and composite applications, and how the judicious application of a solution known as HPE Service Virtualization can mitigate these obstacles and break the logjam to faster application delivery.

The paper will explore the benefit of applying Service Virtualization in four challenging scenarios:

1. Developing and testing an end-to-end business process when components are not available
2. Developing and testing a call to a Cloud Service
3. Developing and testing a multi-step business process that must access a constrained mainframe application in production
4. Achieving performance testing of a composite application when a service is constrained

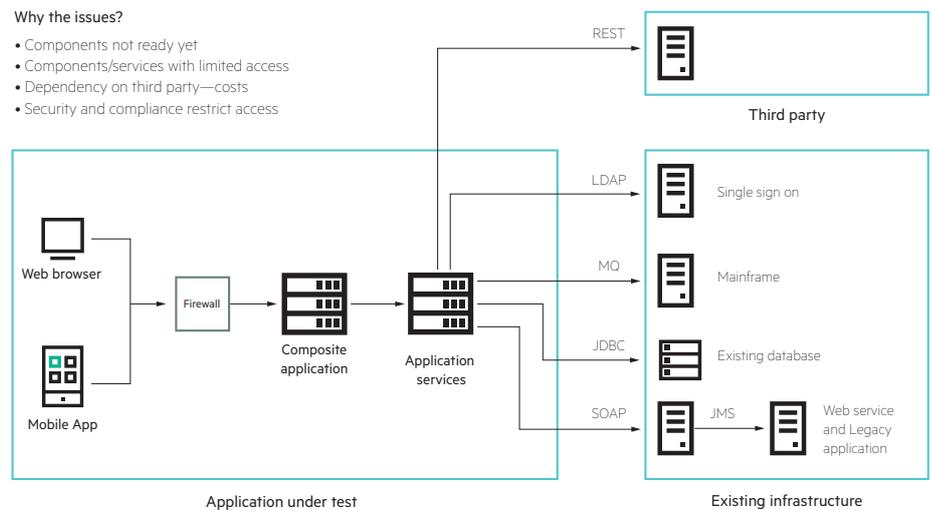


Figure 1: Composite applications equal constraints to developers and testers

Scenario #1: Building a composite application? What if all the components aren't ready at the same time?

Composite application architecture is here to stay. It represents a fundamental set of patterns behind just about every type of application modernization effort being embraced by today's enterprises. Examples live in all forms, from service-oriented architecture (SOA) and Representational State Transfer (REST)-based shared business services, to orchestrated business processes and emerging cloud-sourced solutions. Widely used enterprise application providers are also embracing composite application architecture. Packaged application software continues to be redesigned into flexible components. These components form the basis of the enterprise application business processes. And, in addition, for Web-based and social applications, Web-service application programming interfaces (APIs) are the de facto integration model for mash-ups.

The development and testing of modern composite applications often involves multiple steps to address the lifecycle of component application assembly. It requires early unit and service testing of individual shared services, often without the general user interface (GUI) to support the test; testing of integrations between orchestrated services and data sources; and ultimately end-to-end testing to accommodate the business user experience and transactional integrity. Thus, development and testing teams must have ongoing access to all the needed components.

But, what happens if the components are not all ready or available at the same time? What if one component hasn't been developed yet but another is ready to test? Or, if a component has a critical defect found in early unit testing and is undergoing a change but is needed for an integration test? These constraints cause delays, or worse, skipped testing in order to hit the delivery schedule. Acknowledging and planning for these unavailability constraints does not have to mean building in developer unit test, functional, or integration testing delays. There is another solution—investing in virtual services with a solution called Service Virtualization.

Functional and performance testing with virtualized services

- Instantly available services
- Test early and often
- Lower testing and infrastructure costs

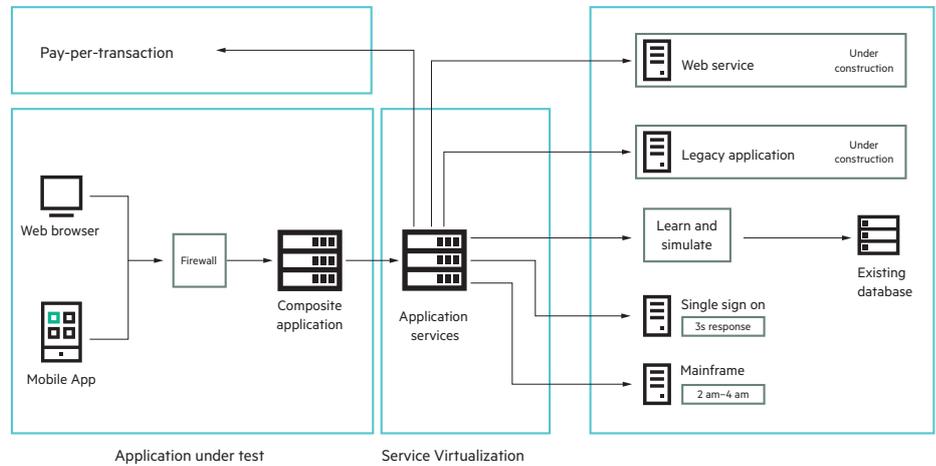


Figure 2: Service Virtualization eliminates delays caused by unavailable components and services

Virtual services act as proxies for the eventual delivered service. They can be identified, modeled, and made available to development and testing teams to maintain forward progress on compositional development, unit testing, functional, and integration testing. And, because they are virtual, running on a virtualization server in the development and testing lab, they can become a shared service, always available to the project teams. In addition, forward-thinking organizations can set up a shared-service repository with built-in governance that can include virtual, as well as actual shared services for ongoing management and to reduce duplication of effort even regarding virtual services.

With Service Virtualization, a development or testing team can create a virtual version of a service or application component without scripting or having to develop in-depth knowledge of the services programming language, protocol, or message payload format. For services under development and not ready yet or undergoing change, Service Virtualization software can learn and simulate the service from its interface definition and data model, the actual service does not have to be running. Once a service is virtualized, it's readily available for developers and testers, and as the service evolves, the virtual version is easy to maintain without scripting. The Service Virtualization software uses a simple wizard model to guide the virtual service creator through the process easily.

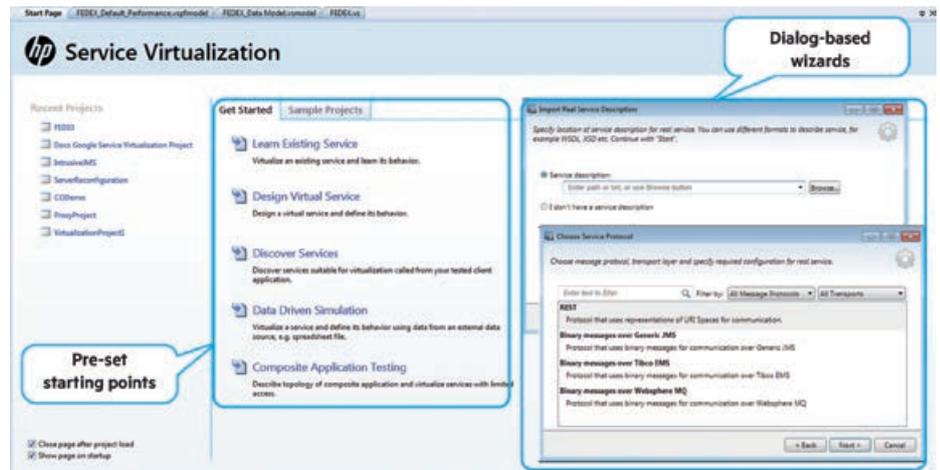


Figure 3: Creating Virtual Services using HPE's simple Service Virtualization Wizard

Scenario #2: Testing when the composite application contains a call to a third-party cloud service

Many business processes and composite applications contain transactions and call-outs to third-party services. Consider a simple e-commerce application. Often the customer order application contains a mobile or GUI front-end that enables a customer to locate a product, order it, and pay for it with credit card. Behind the scenes however, the customer order application is a composite and to complete the transaction, a credit check is done, calling out to an external credit check service. What if the application team made a change to the application such as adding an additional product line or supporting customers in other geographies? How can the change be tested for functionality and performance? Running the test would include the call-out to the third-party credit check service and the third party will charge the company a fee each time their service is accessed. These additional costs are often too high and as a result, testing corners are cut, or even skipped. This is especially problematic with high volume load testing. So, what's the testing team to do? With Service Virtualization, a virtual version of that service can be created by attaining the interface definition and data model from the third-party service provider. The virtual service then stands in for the third-party service and testing can continue with only the cost of creating the virtual service instance being absorbed by the application project team.

Scenario #3: Developing and testing a multi-step business process that must access a constrained mainframe application in production

System of engagement is the current terminology for applications that engage and delight end-users such as online shopping, media and entertainment, and information services. However, behind the scenes, many customer-facing applications consist of a vibrant Web or Mobile user interface front ending a multi-step business process that may often access a back-end system of record. Customer records management, inventory management, payment processing are common examples of back-end systems of record that have been designed for reliability, stability, and scale. And, many of these systems of record have existed to support the organization for years, even decades, and as a result are run on legacy platforms such as mainframes, are implemented in highly adopted but legacy protocols and programming languages such as CICS and COBOL respectively, and are managed under tight monitoring and control by production operations.

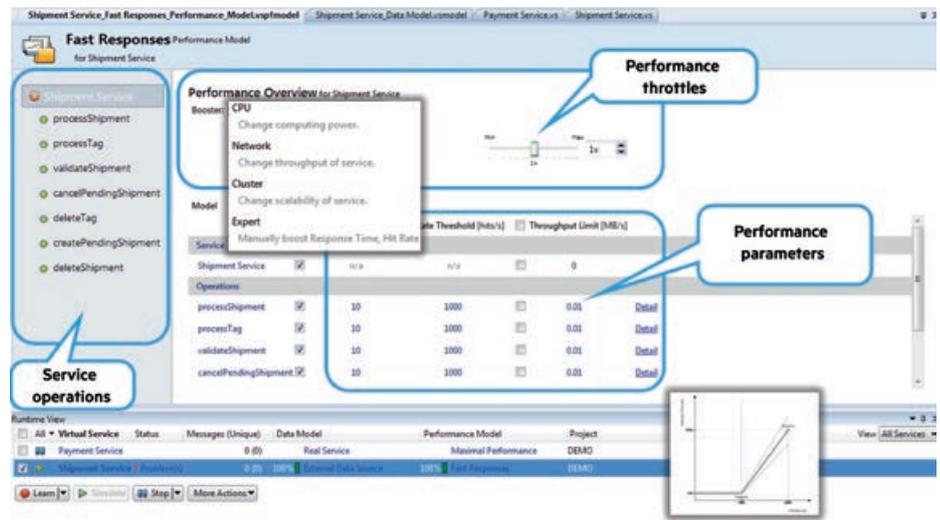


Figure 4: Modifying performance behavior of virtual services

Systems of engagement are designed to be nimble and change frequently to delight their ever impatient consumers. However, systems of record are just the opposite—developed for stability and reliability with infrequent change. As a result, development and testing teams of these systems of engagement struggle to be able to fully develop against and test changes when the change requires access to the back-end system of record. Often times, you hear of delays caused by limited access windows to the back-office system or the time required to provide and then reset the test data from these back-office systems.

With Service Virtualization, access to these systems is only needed once to create the virtual version of the back-office system and then ongoing development and testing occurs against the virtual version. Service Virtualization is able to virtualize the behavior of application components written in legacy programming languages such as COBOL and implemented with widely adopted legacy architectures such as IBM Information Management System Transaction Manager (IBM IMS TM) and IBM Customer Information Control Transaction Server (IBM CICS TS). And, the best part of implementing a Service Virtualization solution is that the application team does not need to locate rare development experts who know COBOL, IMS, or CICS.

Scenario #4: Achieving performance testing of a composite application when a service is constrained

Unavailable, costly, highly regulated, or infrequently available services can impart costly delays and charges for developers and testers needing access. However, there is a fourth scenario where a virtual service can greatly increase the ability to rapidly deliver an application that meets the expectations of the business stakeholders. This scenario is related to load and performance testing. New composite applications and business process implementations require alignment with a performance SLA that meets or exceeds user expectations. And, especially with the advent of mobile consumption, user patience for poor performing applications is shrinking rapidly, with 12 percent of users switching to a competitor on first encounter of a poor performance experience.¹

¹"Harris Interactive Mobile Transactions Survey for Tealeaf," Harris Interactive, March 2011



So, how does a testing team validate an SLA if a back-end service is constrained and cannot deliver the response time needed for the required performance scenarios? Often, services in production become constrained due to consumption patterns or when not heavily loaded, cannot provide delays in response time to mimic a performance scenario that occurs during expected peak consumption. Using a virtual service provides complete control of the performance behavior of the virtual service component. Need a fast response time? Service Virtualization lets the user control the response to meet this scenario? Need to simulate a slower response such as what would occur on peak utilization to see the end result to the user of the composite application? Service Virtualization lets you create simulated peak load conditions with the quick configuration of a performance parameter in the virtual service designer.

Conclusion: Composite Applications and Service Virtualization—the one-two punch for speed and quality

This paper is designed to help development and testing teams facing challenges with ensuring rapid and high quality delivery of composite applications and applications utilizing third-party cloud services consider the role and value of a Service Virtualization approach. Service Virtualization as a strategy and a software solution is designed to relieve the uneasy tension between the agility benefits of adopting composite application architecture and leveraging cloud based services and the ability to develop and test against these services in a timely manner. With Service Virtualization software, teams developing cloud and composite applications are equipped with a powerful solution to mitigate constraints, costs and delays, and break the logjam to faster application delivery.

About HPE Service Virtualization Software

HPE Service Virtualization is a software solution consisting of a virtual server and designer expressly created to remove constraints that cause development and testing delays for validating the different parts of a composite application—the services and the data—that integrate to form a business process. HPE Service Virtualization has extensive support for the protocols, programming, and data models widely used by today's modern and legacy application teams. In addition, HPE Service Virtualization is fully integrated with HPE functional and load testing solutions to quicken and simplify the ongoing process of validation of composite applications in ever shrinking agile delivery schedules. This enables IT organizations to reduce costs and enterprise risks associated with development and testing to accelerate quality application time-to-market.



HPE Software Services

Get the most from your software investment. We know that your support challenges may vary according to the size and business-critical needs of your organization.

HPE provides technical software support services that address all aspects of your software lifecycle. This gives you the flexibility of choosing the appropriate support level to meet your specific IT and business needs. Use HPE cost-effective software support to free up IT resources, so you can focus on other business priorities and innovation.

HPE Software Support Services gives you:

- One stop for all your software and hardware services saving you time with one call 24x7, 365 days a year
- Support for: VMware, Microsoft®, Red Hat, and SUSE Linux as well as HPE Insight Software
- Fast answers giving you technical expertise and remote tools to access fast answers, reactive problem resolution, and proactive problem prevention
- Global Reach Consistent Service Experience giving global technical expertise locally for more information goes to hp.com/services/softwaresupport.

Learn more at
hp.com/go/servicevirtualization
hp.com/go/alm

You can also follow us on Twitter
[@HPSoftwareALM](https://twitter.com/HPSoftwareALM).

Sign up for updates

★ Rate this document



© Copyright 2012, 2015 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Microsoft is a U.S. registered trademark of Microsoft Corporation.

4AA4-4189ENN, November 2015, Rev. 1