GENROCKET ENTERPRISE TEST DATA AUTOMATION

April 2020

GenRocket is taking a leadership role in the evolution of traditional **Test Data Management.** We built the industry's most comprehensive **Test Data Generation** platform and have extended it to become an **Enterprise Test Data Automation** platform.

Test Data Automation requires 3 essential ingredients that GenRocket now offers: Precise control over data quality, automated self-service provisioning and a fully scalable enterprise test data platform.

HIGH QUALITY DATA

- Accurate Test Data Generation
- Blended / Queried Prod Data
- Combinations & Permutations
- Assured Referential Integrity
- 100% Data Privacy & Security

SELF-SERVICE PROVISIONING

- Fast, Powerful Self-Service
- Test Data Cases & Rules
- Presets & Wizards
- Intelligent Automations
- Cloud Collaboration / On Prem Data

SCALABLE PLATFORM

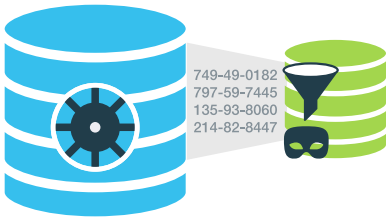
- Zero Touch On-Demand Delivery
- High Volume & High Performance
- Dynamic API-Driven Test Data
- CI/CD & Testing Tool Integration
- Adaptable to Any Test Data Need

Test Data Automation is a comprehensive solution that eliminates industry-wide problems that compromise testing efficiency and each one is describe below.

Problem: Manual Test Data Creation

The World Quality Report found that 69% of companies are still creating data manually, principally with spreadsheets, in order to provision the data needed for testing. Manual test data creation is both time-consuming and cumbersome, but necessary if testing requirements call for data value combinations and permutations that are not readily available from other sources.

Solution: Test Data Automation can dramatically accelerate the manual data creation process by 1,000% or more by defining and generating any volume of test data with total control over the required data variations (patterns, combinations, and permutations) and the desired output format.

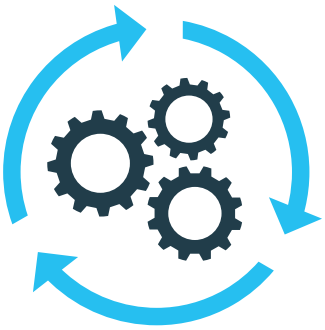


Problem: Production Data Mining and Masking

When the need for test data arises, many QA teams mine the application production database for relevant subsets that can be used for testing. The challenge is the random nature of the data they extract. It's not controlled data and usually requires manual modification by testers to be useful. Additionally, sensitive customer information must be carefully masked to secure the data and conform to relevant privacy laws.

Solution: With **Test Data Automation** testers can generate the precise data they need for any type of positive, negative, edge case or combinatorial testing with real-time synthetic test data that is generated at 10,000 rows per second and is 100% secure.

Problem: Test Data for End-to-End Testing



One of the most demanding QA challenges is testing a business process across multiple applications and their APIs with end-to-end integration testing. This places a special demand on test data for validating workflows that process information and make decisions based the results. Test data must be dynamic in order to validate different outcomes for a variety of input values.

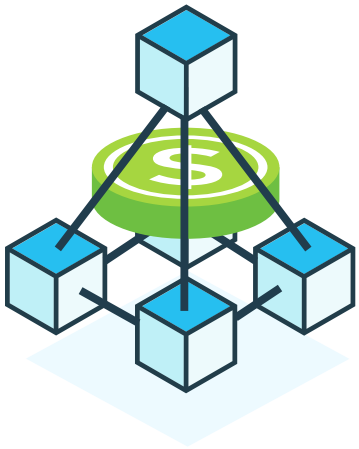
Data accuracy is critical if the test involves the use of real transaction or account codes to test different paths of the workflow. End-to-end integration testing may also involve multiple data structures and formats to ensure compatibility between diverse systems.

Solution: GenRocket's **Test Data Automation** solves complex end-to-end testing challenges with its ability to define test data based on business rules and combine synthetic data with production data that is queried during the testing process and injected into an automated test procedure.

Problem: Simulating Complex Data Feeds

Data feeds are batch or real-time streams of data that represent an exchange of electronic information like credit transactions, money transfers, insurance claims or supply chain orders and fulfillment. They are often transmitted and processed in high volume with an expectation of total accuracy and precision. Testing applications that process complex data feeds can be very demanding.

Solution: The GenRocket **Enterprise Test Data Automation** solution is designed for provisioning test data for any complex data feed in industries like healthcare, financial services, eCommerce, and supply chain logistics. Same day provisioning of high-volume data feeds (in the millions of rows) is an easy and automated process with self-service modules and a wizard-based user interface.



Problem: TDM System Cost and Complexity

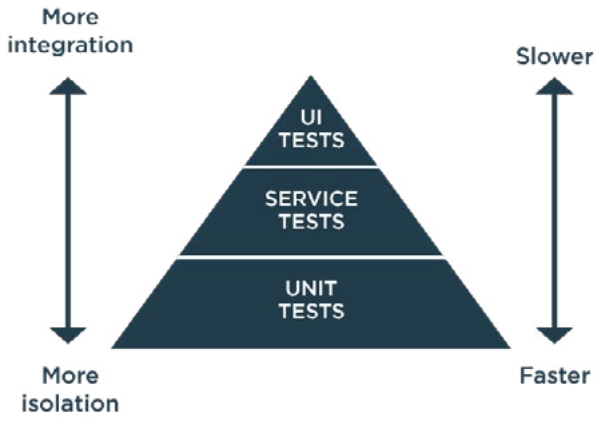
In addition to manual data creation and the mining and masking of production data, another data provisioning alternative is the use of a traditional Test Data Management platform. The option is often considered, but infrequently implemented because of its high cost and complexity. TDM systems pull, copy, mask and refresh production data. This is a slow, complex process and has come to represent a bottleneck for testing.

The preferred model is distributed self-service where testers can provision their own test data when they need it. As the next generation of Test Data Management, *Enterprise Test Data Automation* combines the use of *Real-Time Synthetic Test Data* with queried production data to maximize speed and data quality with a solution that offers simplicity at a much lower cost.

Solution: With its self-service modules and a wizard-based user interface, GenRocket's **Test Data Automation** solution radically improves the simplicity and efficiency of test data provisioning. Adaptable methods for integrating GenRocket with test automation tools and frameworks complete the picture and maximize the return on investment in test automation technology.

TEST DATA AUTOMATION DEPLOYMENT STRATEGY

GenRocket's deployment strategy for *Enterprise Test Data Automation* follows Agile best practices and uses the *Software Testing Pyramid*, a useful model to visualize the relationship between various test categories and their implications for automated test execution.

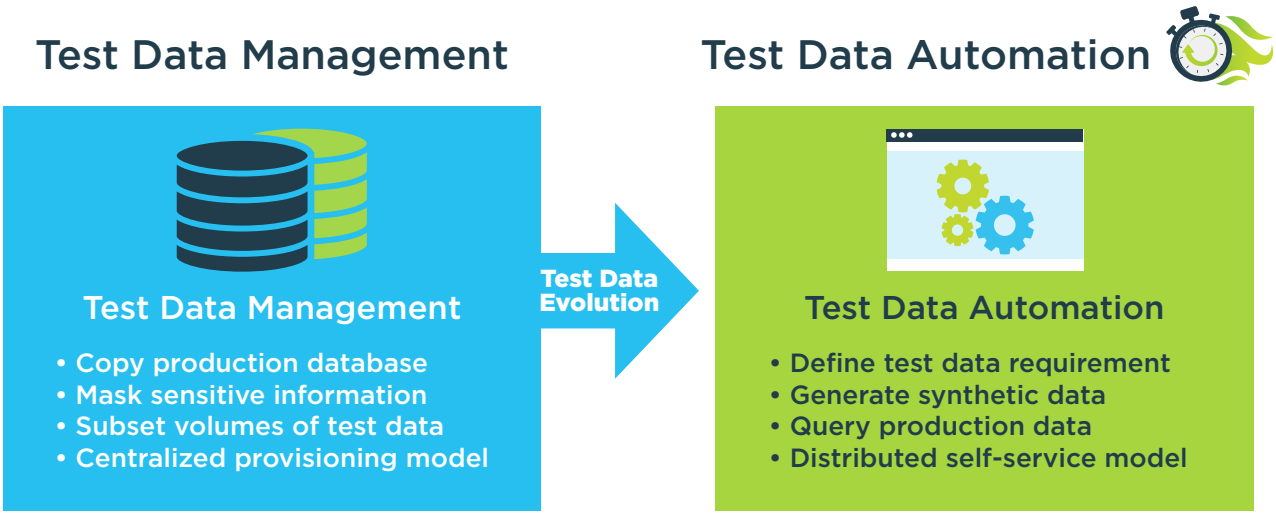


Toward the bottom of the pyramid are tests performed on more isolated units of code which are executed more frequently and take less time to complete. Toward the top of the pyramid are tests performed on more integrated sections of code which are performed less frequently and take longer to complete. The three tiers represent Unit Testing, Service Testing, and User Interface (UI) or end-to-end testing.

Our *Test Data Automation Deployment Strategy* starts at the lowest level of **Unit Testing** and progresses through the higher levels of **Service Testing** and **UI Testing**. The test data challenge is different for each stage and the implementation of GenRocket's **Test Data Automation** solution adapts to meet the various demands of Agile testing.

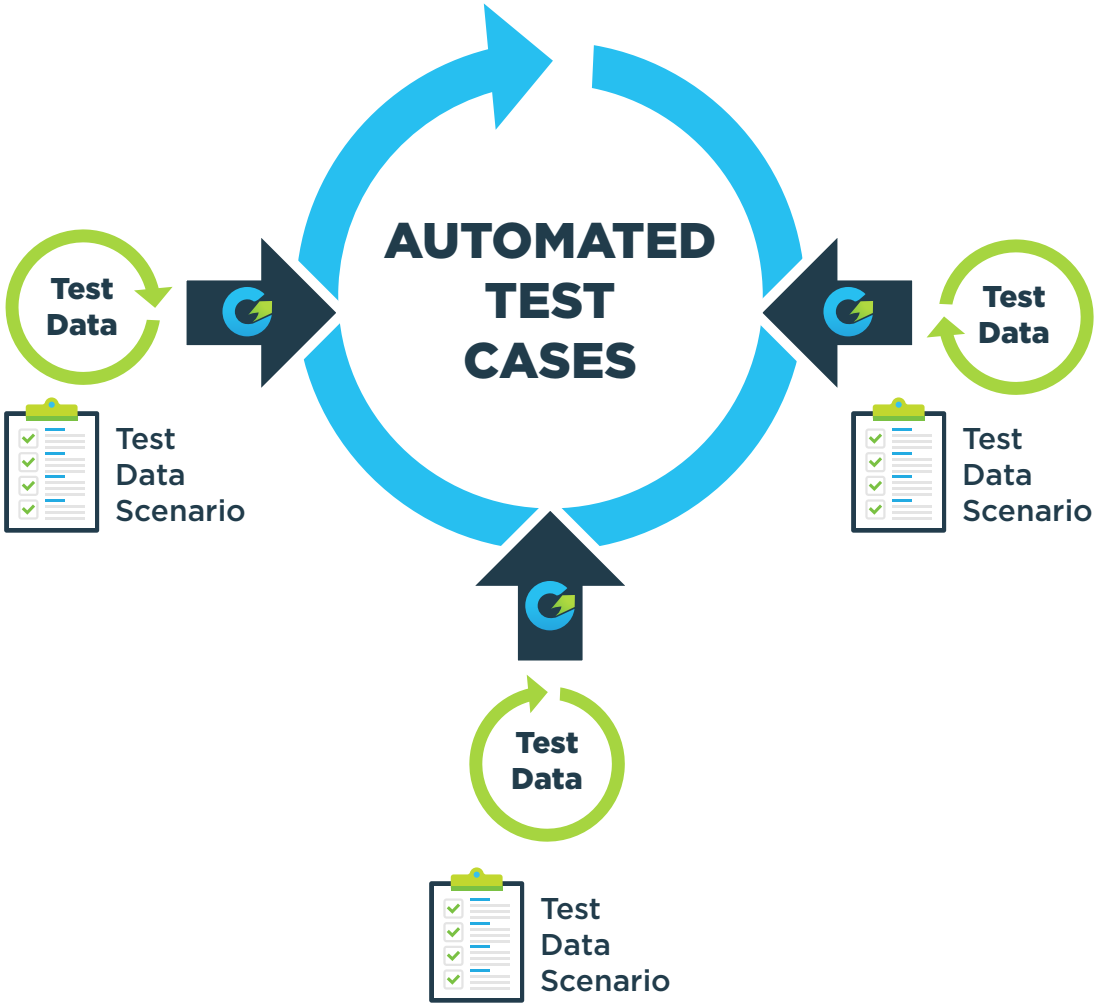
Follow the Test Data Provisioning Evolution

The model for provisioning test data is evolving from traditional *Test Data Management* (TDM) to real-time and highly integrated *Test Data Automation*. The traditional TDM model is based on copying production data, mining that data for appropriate test data sets, masking sensitive information that must remain private, subsetting, refreshing and versioning the data for test cases and working through a centralized data provisioning process to accomplish all of the above. The traditional TDM approach is too cumbersome for an Agile environment.



The TDM model is evolving to a new and highly efficient model where controlled and accurate test data is provisioned instantly and on-demand in a distributed self-service environment. With *Test Data Automation*, the process is streamlined into a 3-step process: Import the data model, create a *Test Data Scenario* matching the needs of each test case and generate test data in real-time during test operations.

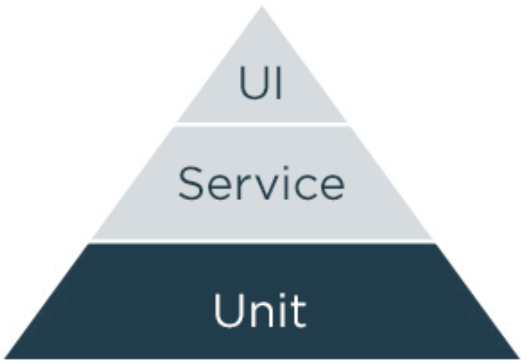
Real-Time Synthetic Test Data Generation



Real-Time Synthetic Test Data Generation is designed for the accelerated pace of Agile testing and maximizes the benefits of continuous integration and testing. GenRocket can be used for any level of testing from basic unit testing, to more complex API testing, to the sophisticated requirements of end-to-end integration testing.

The GenRocket platform seamlessly integrates with test automation tools. It provides an intuitive self-service layer for defining any volume or variety of test data and a powerful API that injects data into the testing process at any stage of a complex workflow.

Unit Testing at the Agile Task Level



SOFTWARE TEST PYRAMID

Unit testing is performed at the Agile *Task Level* on the smallest unit of code. These isolated tests are often performed by a developer for testing a single function and data domain. Unit testing is sometimes described as *in memory* testing because it is isolated from the external environment and involves no database access.

Unit tests represent the highest volume of test operations and requires lowest volume of data, making it an excellent candidate for test automation. However, test data must be carefully controlled to meet the needs of each test case and generated in real-time to support the accelerated pace of continuous integration and testing.

The component-based architecture of the GenRocket Test Data Automation platform is an ideal technology for defining and provisioning test data for high volume, automated unit testing. Simply define the required data using a Test Data Scenario and generate it during test execution in any format required (JSON, XML, CSV, etc.).

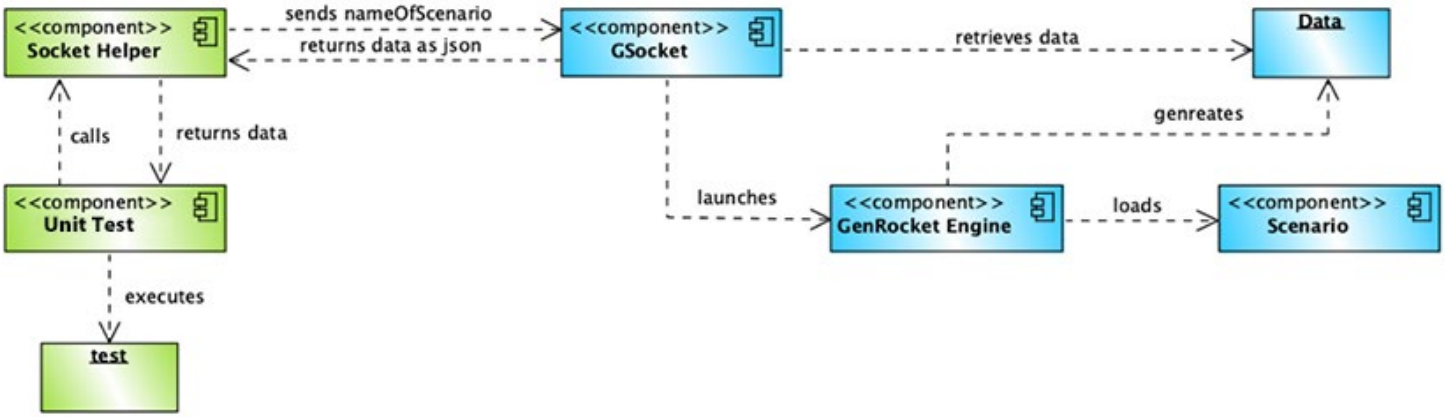
Unit Testing With GenRocket Test Data Automation

Here is a summary of the steps for unit testing in an Agile environment with GenRocket:

- Define Test Data Scenarios for each task-level test
- Generate single-domain test data in the desired format
- Invoke Test Data Scenarios from a test script via socket interface
- Execute unit tests with automated test tools and the GenRocket platform

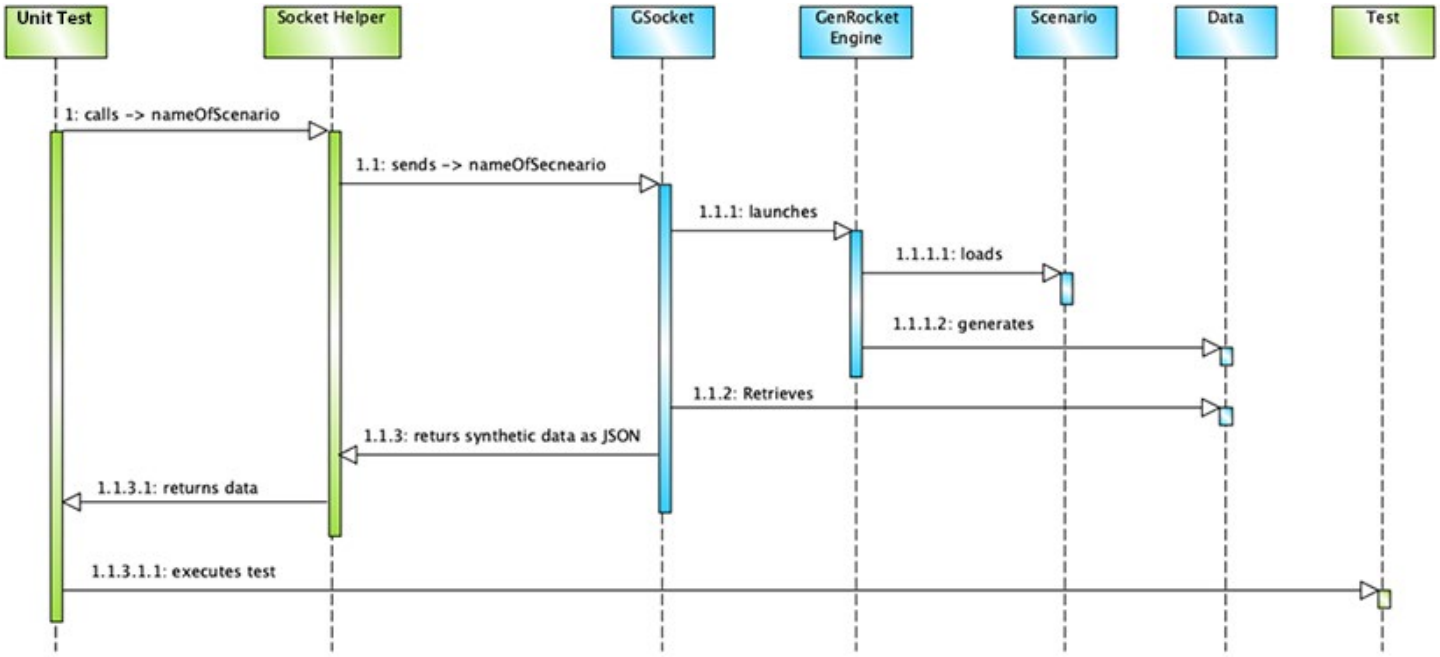
Below is a component diagram to illustrate the elements used for this automated testing approach. Unit tests use a Socket Helper to call the GenRocket Engine using its GSocket interface. The GenRocket engine loads the appropriate Test Data Scenario and generates predefined test data in the required format (JSON in this case) in less than 100 milliseconds. The GSocket interface then retrieves the data and returns it via the Socket Helper for use during the test procedure. The components of the test script are color-coded in green while the GenRocket components are color-coded in blue.

Unit Testing Component Diagram



The sequence diagram below illustrates the process as that takes place during test execution.

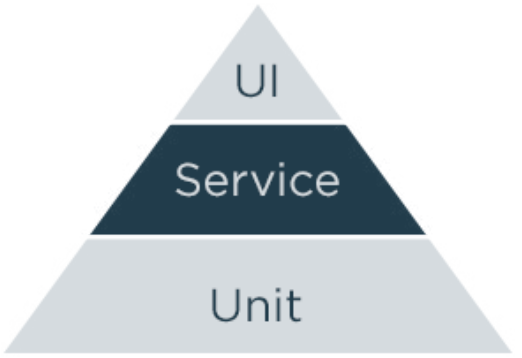
Unit Testing Sequence Diagram



After the test is complete, the test data is purged. The Test Data Scenario is stored for future use by this, or any other test, and ready to generate a fresh copy of pre-defined test data.

SERVICE LEVEL TESTING AT THE AGILE STORY LEVEL

Service level testing is the next level of complexity represented by the software pyramid. Service level tests focus on the interaction of Agile Tasks via one or more APIs in a module of code that represents an Agile User Story.



SOFTWARE TEST PYRAMID

Service level testing, also called API testing, usually involves database access and one or more data domains. When using multiple domains, it's critical to maintain referential integrity between key fields in the data tables involved in testing.

Service level tests can involve any category of testing including functional, integration, performance, security or regression testing. In an Agile environment, service level tests are usually automated and run through the release pipeline.

GenRocket is a perfect match for meeting the test data challenges of Service/API testing. Test Data Scenarios can be defined to generate any variety of test data in any volume to meet the needs of any category of testing. With 258 Test Data Generators, GenRocket TDG can simulate any combination of data values and its 61 Test Data Receivers can format data in any output format. This provides testers with total control over the data they need for service/API testing with comprehensive test data that maintains referential integrity and data privacy.

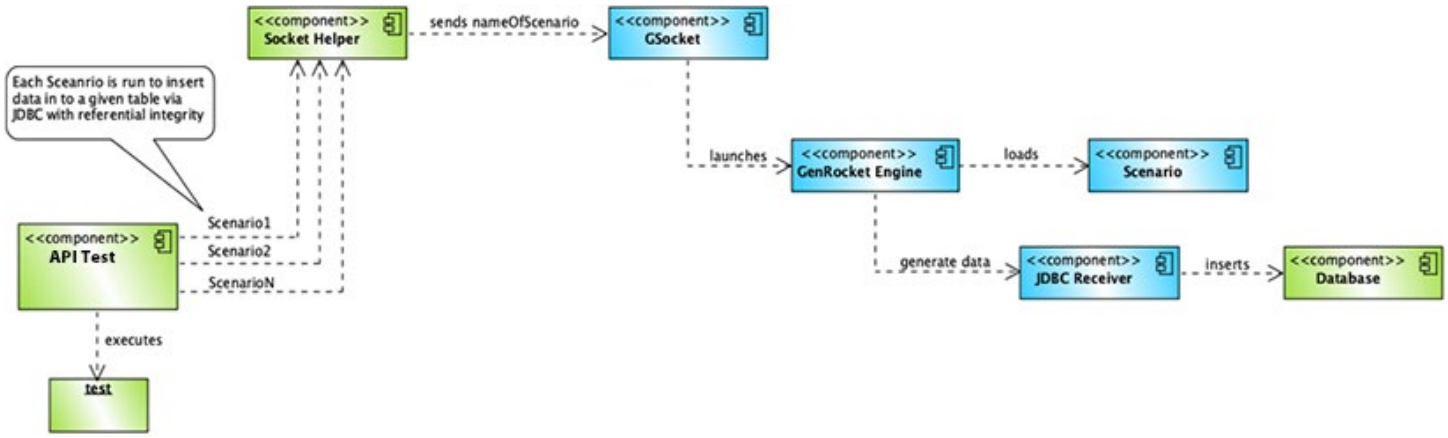
Service Level Testing with GenRocket Test Data Automation

Here is a summary of the steps for using GenRocket when performing service level testing in an Agile environment:

- Define *Test Data Scenarios* for each story-level test
- Maintain referential integrity between key fields in data domains
- Invoke *Test Data Scenarios* from a service level test script via socket interface
- Execute service level tests using automated test tools and the GenRocket platform

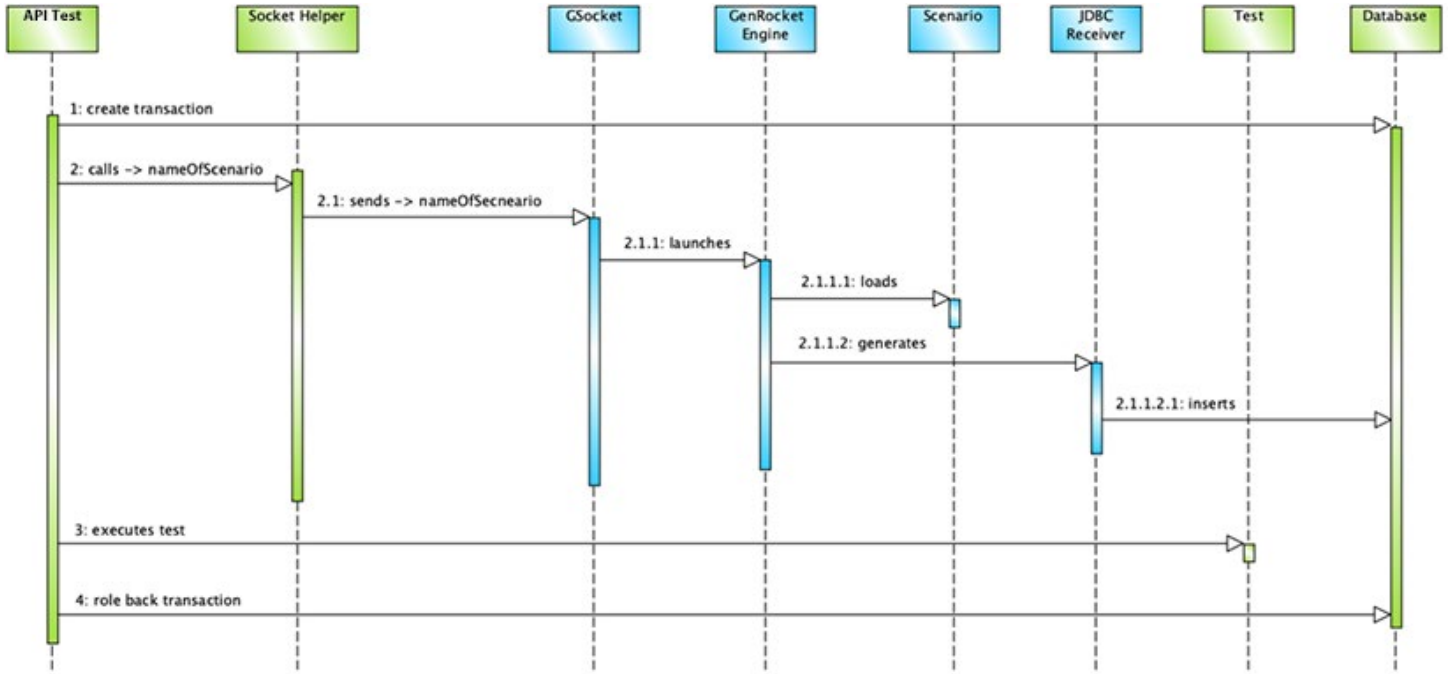
Below is a component diagram to illustrate the various elements used for automated service/API testing. During test operation, GenRocket *Test Data Scenarios* are run to insert data into a given table via JDBC with referential integrity. A *Socket Helper* in the test script calls each *Test Data Scenario* via the *GSocket* interface and launches the *GenRocket Engine* to run the scenario, generate the data, and insert the data into database table(s).

Service Level Testing with GenRocket Test Data Automation



Service Level Testing Sequence Diagram

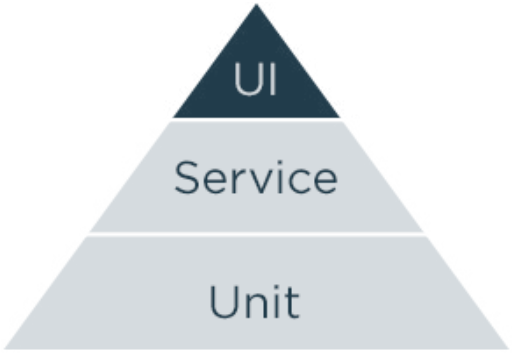
The sequence diagram below illustrates the process that takes place during service-level test execution.



After the test is complete, all test data is purged. Test Data Scenarios are then stored for future use by this, or any other test, and ready to generate a fresh copy of the test data when needed.

UI TESTING AT THE AGILE FEATURE LEVEL

The highest and most complex level of the Agile software testing pyramid is Feature-level testing at the User Interface, or UI layer. Feature-level tests are complex end-to-end test procedures for large blocks of code designed to execute a business solution from a user's perspective.



SOFTWARE TEST PYRAMID

End-to-end testing typically involves complex workflows that require comprehensive test data that is controlled (for data combinations and permutations), rules-based (for testing business logic), private (no personally identifiable information), accurate and realistic (real production data blended with synthetically controlled data). As in the service layer, a variety of testing categories may also be involved in UI testing, including functional, integration, performance, security, and regression testing.

GenRocket is especially powerful for end-to-end testing. Its component architecture allows any combination of data tables and complex data feeds to be modeled, defined and generated on-demand and during test execution.

GenRocket's self-service layer, called ***GSelf-Service*** allows any tester to define, create and manage comprehensive, enterprise-class test data that is seamlessly integrated with test automation tools. GenRocket's ability to replace sensitive customer data with realistic synthetic data ensures data privacy. Its ability to query a production database and retrieve accurate data values that are blended with controlled synthetic data maximizes test case effectiveness.

UI Testing With GenRocket Test Data Automation

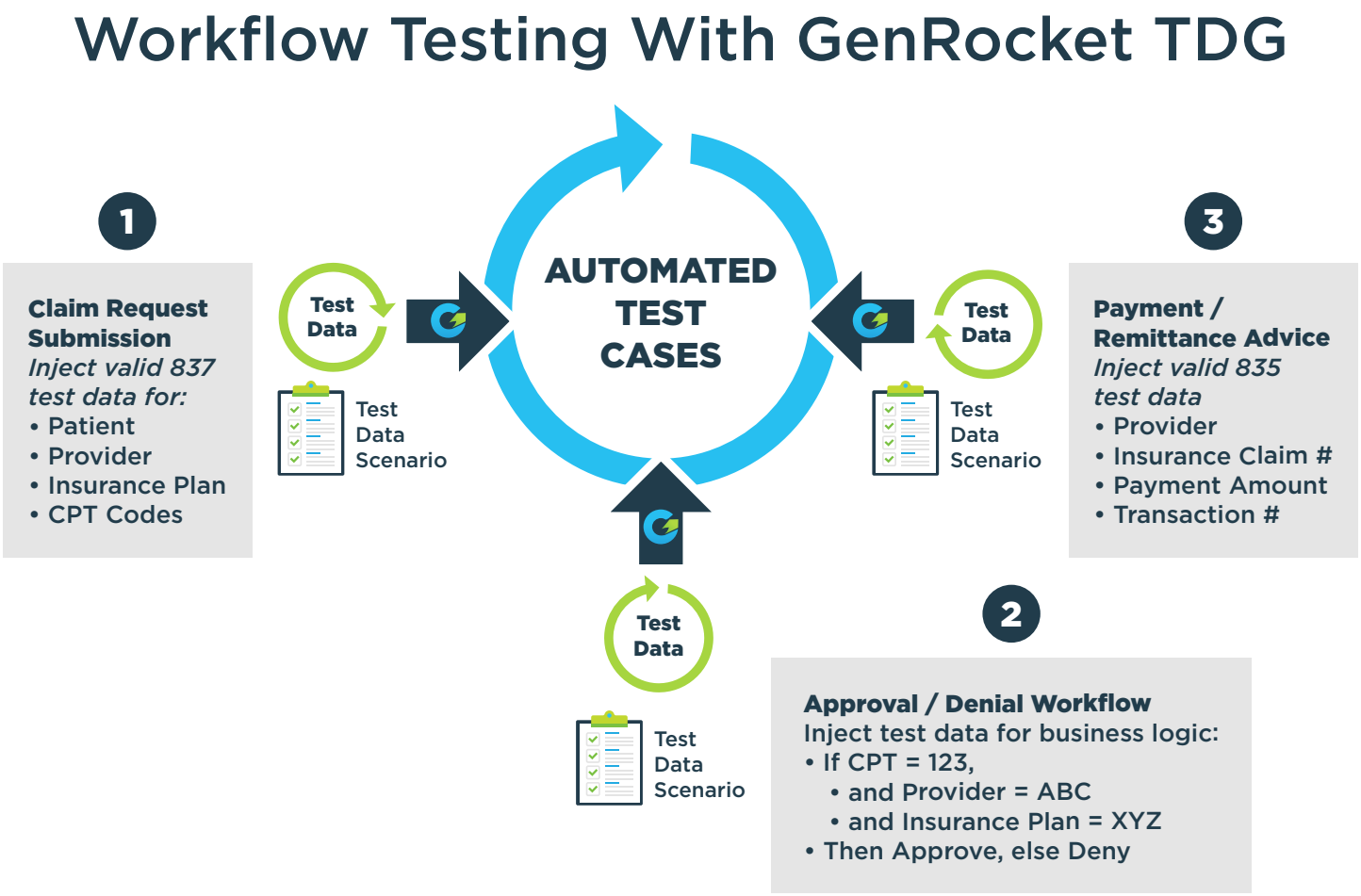
Here is a summary of the steps for using GenRocket for UI testing in an Agile environment:

- Define Test Data Scenarios for each feature-level test
- Test complex workflows with blended synthetic and production data
- Invoke Scenarios in multiple ways: Bulk loading, Restful web services, and GSocket
- Execute UI level tests using automated test tools and the GenRocket platform

End-to-End Testing with GenRocket Test Data Automation

Here is an example of end-to-end testing with GenRocket. Imagine testing the workflow in a healthcare insurance claims processing application. Test data for a submitted claim must contain synthetic data in place of personally identifiable information for patients and providers.

However, the codes used to represent diagnostic and treatment procedures, called CPT codes, must be based on production data to be clinically accurate. CPT codes are the basis for returning a correct determination for approval or denial of benefits based on the patient's insurance plan. In order to test the claim processing business logic, the tester must blend synthetic test data to replace sensitive information with production data to ensure accuracy. See the workflow testing diagram below.



MAXIMIZE THE BENEFITS OF CONTINUOUS INTEGRATION AND TESTING

The Agile test strategy outlined in this article will streamline the Agile testing process as it improves the effectiveness of test cases used to prevent bugs during the development process.

Greater ROI: QA teams realize the full benefit of their investment in test automation tools with a TDG platform that provides real-time, enterprise-class test data on-demand

Higher Quality: More control over the volume and variety of test data used for testing allows QA teams to achieve greater code coverage and drive higher software quality

Improved Efficiency: A distributed self-service model for provisioning test data eliminates the wait states and enables true continuous integration and testing in an Agile environment

If you would like to learn more about implementing a *Test Data Transformation Strategy* for your QA organization, contact us to request a live demonstration of **GenRocket Test Data Automation.**

REQUEST A DEMO