

GenRocket (1): the building blocks

What GenRocket is, then its core pieces in order: domains, attributes, generators, receivers and scenarios.

GenRocket: what it is and how it works

GenRocket is built around the third way of making test data: generation. It does not copy production data at all. You describe the shape of the data you want, and it manufactures as many rows as you ask for, already consistent with each other.

The product is made of five building blocks. Some pieces run in a cloud web site and some on your own machine, and GenRocket's own method is *Model, Design, Deploy, Manage*. The building blocks are covered in detail in domains, attributes and generators, then receivers and scenarios.

6.1 Words to know first

Synthetic test data

Data manufactured by an algorithm instead of captured from real events. GenRocket's page defines it in nearly those words. Nobody's real name or card number is in it.

Domain

One "thing" you want data for, such as a customer. For a database it is a table. For output it can also be a file, message, JSON or XML document.

Attribute

One characteristic of a domain, such as `firstName` or `email`. For a database it is a column.

Generator

The small engine attached to an attribute that decides what value to produce, for example "a first name from a list" or "count up from 1".

Receiver

The component that takes the raw generated values and puts them into a usable form: a CSV file, a JSON file, SQL insert statements, a database table and many more.

Scenario

A saved instruction sheet: which domains to generate, how many rows of each, in what order. Running it is what actually produces data.

Test Data Automation (TDA)

The phrase GenRocket's pages use for its approach: test data defined once as components and then generated automatically, in real time, whenever a test needs it.

Web platform

GenRocket's web application in the cloud, where you model and design. Also called GenRocket Cloud.

GenRocket Runtime

A set of Java programs installed on your own computer or server. It reads a downloaded scenario and generates the data there, inside your firewall.

Project and project version

A project holds the data model for one application or database. A version is one release's copy of that model, with its own domains, receivers and scenarios.

Referential integrity

The promise that every foreign key points to a row that exists. See the data basics chapter.

6.2 What GenRocket is and why synthetic data

GenRocket's welcome pages are short and to the point. They define synthetic test data as data "manufactured or generated by an algorithm rather than real-world events", and argue that companies can maximise test coverage by using it to test all permutations and combinations. The CI/CD page

describes the product as a component-based platform for defining and provisioning test data for high-volume, automated testing.

Think of a bakery supplying cupcakes for a hundred tasting events. One approach is to buy finished cakes from a rival shop and scrape off the logos (masked production data). The GenRocket approach is to own the recipe and the oven: "chocolate, 12 per tray, one tray with a deliberately burnt top", made on demand. The analogy breaks in one place: a recipe makes cakes that taste alike, while a GenRocket recipe can vary values, include edge cases and keep related trays consistent.

WHY IT MATTERS

A tester for ShopSphere's "Customer Orders" app needs a customer with 0 orders, one with 1, one with 200, one whose email is missing and one whose surname is 60 letters long. Production rarely contains all of these, and copying production brings real names along (see why TDM exists).

Because GenRocket's data is manufactured to a rule, you can ask for the odd cases directly and rebuild the same set tomorrow.

The documentation is also frank about the work involved. The FAQ says the harder data-design tasks are assigned to a small group of trained "Test Data Engineers", while testers and developers mainly search for the data they need, deploy it into their test environment and tweak it. The Test Data Workflow page starts from the same person: it says the workflow begins with a Power User, a Test Data Engineer.

One more design choice matters for privacy. The Test Data Projects page says a project *contains no customer data*. It holds your data model and the five components that describe how to generate data. The architecture diagram (below) labels the cloud part "No Customer Data Stored".

6.2.1 The problem the vendor says it solves

GenRocket's 2020 webinar (the CEO and CTO, with a partner company) opens with the market problem: test automation is spreading fast, but test data is not keeping up. Citing an industry quality report that he names as the World Quality Report, the CEO gives two figures: 69% of organisations use spreadsheets to generate test data, and 59% still create it manually. He adds stories from unnamed customers: a large financial firm with a TDM tool finds the data it needs in production only about a third of the time, and a healthcare insurer's fastest delivery of test data is 34 days. The complaints he hears each week are that data is too slow, does not match the test cases, and is too manual.

These are the vendor's claims: the report, the two percentages (which the talk does not reconcile) and the customer stories are unverified. The 2022 overview video promises the matching fix, cutting days to minutes by replacing spreadsheets and manual methods. Its examples (25 hours of data preparation down

to five to ten minutes per cycle, 16 days down to two hours, over 1,200 hours saved in nine months) are also vendor claims, with no method shown.

6.2.2 Traditional TDM versus synthetic test data automation

Both videos describe the contrast; the 2022 video draws it most clearly. The CEO does not call traditional test data management (TDM) wrong. He says it helps with privacy law and faster provisioning, and that GenRocket adds to it. The differences are in the starting point and in what each can produce.

Traditional TDM: copy, mask, subset, refresh

Start from a production database or application. Copy the data, run a masking algorithm to make a secure copy in a lower (non-production) environment, and refresh it from time to time. Testers then search the copy for data that fits their test case. The data is realistic and keeps referential integrity.

The gaps the vendor says its enterprise customers report: no unique data, no negative data (deliberately invalid values), not every permutation and combination, not enough volume, nothing in a required condition, and no dynamic data spanning many applications.

Synthetic test data automation

Start from the *metadata* (table and column definitions), not the data. The metadata becomes a test data project that acts as the template. It is versioned (version 1, version 2) and lifecycle-managed so it follows the application. Against the project you build test data cases for unit, integration, load and performance tests, and negative cases. Each gives realistic data with referential integrity plus the variety that copies lack.

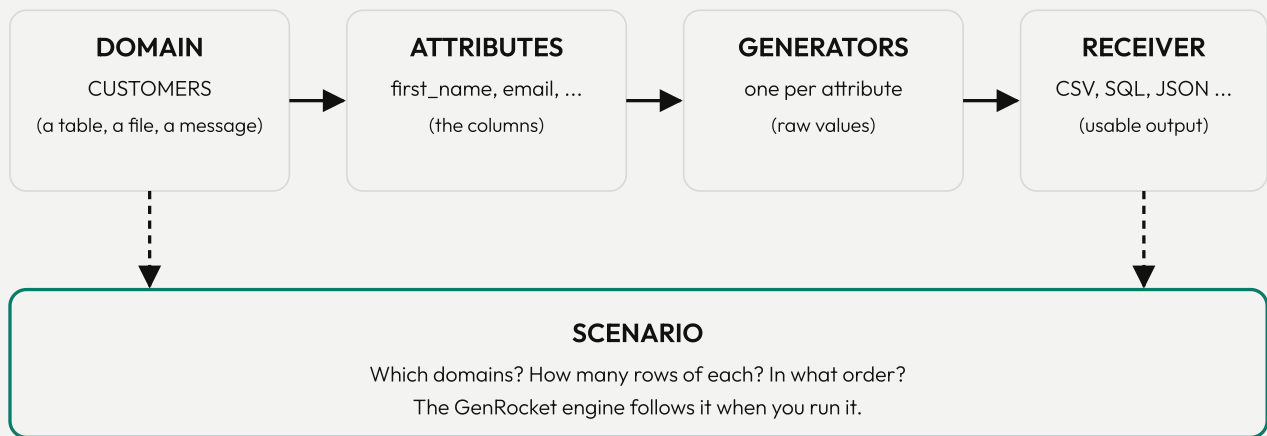
The 2020 webinar adds that "query generators" can still blend a few real production values, such as an account number, into generated rows.

"Stateful" data, one of the gaps, is data in a particular condition that a workflow needs; for ShopSphere, an order already shipped whose payment has settled. On masking, the CTO said in the 2020 Q&A that when GenRocket moves production data to staging it replaces sensitive columns with synthetic values instead of obfuscating them. The help centre also lists masking receivers and an in-place masking project type (sections 6.8.2 and 19.3.9), so check the current product. The concepts are in the chapters on subsetting, masking and generation.

6.3 The five key components

The help centre says the whole platform is built on five key components, and every other feature is a derivative of them. Learn these five words and the rest of GenRocket becomes vocabulary. The table gives the reference with ShopSphere examples.

THE FIVE KEY COMPONENTS, WITH SHOPSPHERE EXAMPLES			
Component	Think of it like	Official examples	ShopSphere example
Domain	A noun: a person, place or thing	Database table, file, image, message, JSON, XML	CUSTOMERS , ORDERS , ORDER_ITEMS
Attribute	A characteristic of the noun	Id, FirstName, LastName, Phone, SSN, DOB	first_name , email , zip_code
Generator	The recipe for one attribute's values	NameGen, PhoneNumberGen, SSNGen, DateGen, QueryGen	A name generator for first_name , a counter for customer_id
Receiver	The packaging that turns raw values into a usable format	JDBC, SQL, CSV, Image, XML, JSON, VSAM, REST, SWIFT	A CSV file of customers, SQL inserts for orders
Scenario	The instruction sheet: what to generate and how much	"Generates data into a typical test case in about 100 milliseconds" (the vendor's claim)	"50 customers, then their orders"



How the five components relate. A domain has attributes, each attribute has a generator, a receiver packages the values, and a scenario (dashed arrows) draws on the domains and receivers to tell the engine what to produce.

One rule to remember from the Generators Overview: generators only *generate* values; receivers *format* them. The split is deliberate. You can change CSV to JSON by swapping the receiver, without touching any generator.

The five-component page adds a small note: the components can also be managed through the Runtime Application Interface (API). The CI/CD page, quoted later, uses that to change a scenario while a pipeline runs.

WORKED EXAMPLE: ONE CUSTOMER ROW

Take the ShopSphere row 1001 | Asha | Menon | asha.menon@example.com | 9876543210 | 123-45-6789 | 1990-04-12 | 560001 from the earlier chapters. In GenRocket terms: the **domain** is CUSTOMERS. The **attributes** are customer_id, first_name, last_name, email and so on. A **generator** on first_name produces "Asha" (then a different name for the next row), a generator on customer_id counts 1001, 1002, 1003. A **receiver** writes the lines into a file. A **scenario** says "make 3 of these". Every value is fake, so the row can go to any test environment.

6.3.1 The five components in plain words

The 2020 webinar explains the same five pieces before any demo, and its wording is a good way to remember them. A **domain** is a noun: a person, place or thing, such as organisation, department, user or address, with no limit on how many or how they are arranged. Its **attributes** are its characteristics (name, password, phone number, date of birth, card number). Each attribute gets a **generator**, and a generator

has *parameters* that control exactly what comes out; a name generator, for example, can be told to give first, last, male or female names, and the CTO says its lists allow around 40 million combinations. A generator's output is raw, unformatted data, and the **receiver** formats it. The **scenario** is the instruction set that runs and produces the data. His rule of thumb is one scenario per domain, linked into a chain with a loop count each; section 6.25.7 works through his example with the arithmetic.

6.4 Test Data Automation and "Model, Design, Deploy, Manage"

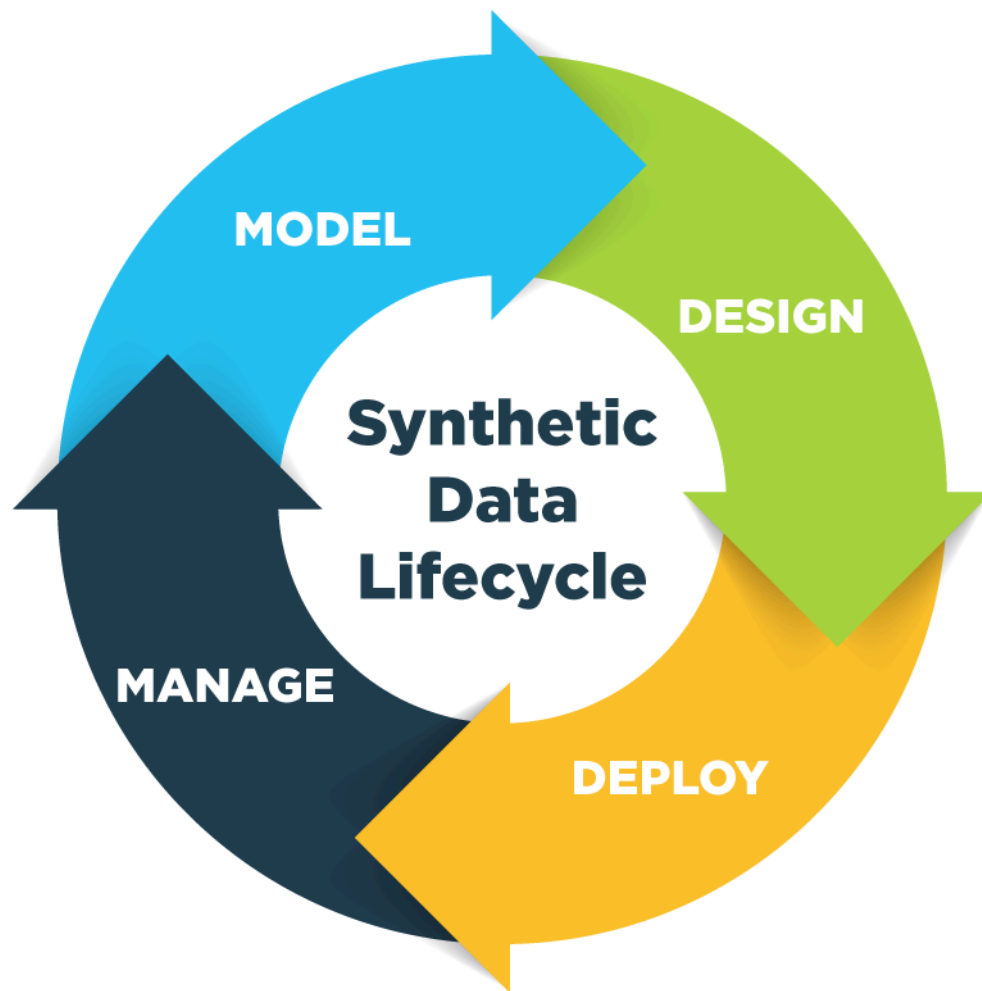
GenRocket calls its platform a *Test Data Automation* platform. The CI/CD page explains the idea with a short list of things any generator of data must know, and then shows how GenRocket captures each one:

What the generator must be told	Where GenRocket records it
The object to create (a User, a Customer)	Domain
The properties of the object (name, id number, phone)	Attributes
What each property should look like (Swedish names, a national id)	Generators
The format of the result (CSV, JSON, XML)	Receivers
How many, in what order (implied)	Scenario
Something to carry out the instructions (added later)	GenRocket Runtime

Notice that the runtime has now joined the list as a sixth piece. The five components are the *design*; the runtime is the *engine* that executes it. The docs describe the runtime as a secure Java program that executes encrypted scenarios inside the customer's firewalled environment.

6.4.1 The four steps

GenRocket's own name for its method is "Model, Design, Deploy, Manage". The Methodology page uses exactly these words, the CEO states it as "the GenRocket methodology" in the 2022 overview video, and the CI/CD page gives each step a plain meaning.



Official screenshot: The vendor's "Synthetic Data Lifecycle" ring: Model, Design, Deploy and Manage chase each other in a circle. The circle matters: after Manage, the next release of your application sends you back to Model.

Source [page](#) ↗

Step	What the docs say	Where it happens in the product	ShopSphere version
Model	Import the data model. Done in the web platform (many import and create methods) or by API at runtime.	Project Dashboard: <i>New Domain</i> menu (import from DDL, XTS, CSV, JSON and others) or manual creation	Import the CUSTOMERS, ORDERS and ORDER_ITEMS definitions as three domains
Design	Create test data designs that match each test case: how much data, which variations. Self-service features exist for positive, negative, pattern, permutation, boundary, dynamic and load data.	Generators, receivers, scenarios, and the self-service "Test Data Case" family	Pick generators per column, a CSV receiver, a scenario for "50 customers"
Deploy	Generate and deliver the data in real time. Over 100 receivers format and deliver it. Triggered from a CI/CD script or a REST request to a GenRocket server component.	Download the scenario, run it with the runtime	A pipeline job runs the scenario before the order tests start
Manage	Manage multiple releases of test data projects with version control. A repository component keeps managed artifacts in sync.	Project versions, categories, G-Repository	Release 2.0 of the shop gets its own project version

Counts differ by source and date, so treat them as approximate and growing. The Deploy row says "more than 100" receivers and the receivers overview says "120+". The 2020 webinar spoke of about 258 generators and 61 to 63 receivers (the company started with 13 and 4), and the 2022 video of 680+ generators and 91 receivers. The current help centre is the reference.

The same page also lists the self-service vocabulary used in the Design step, in a table from the vendor's screenshot below. These names are used in the data patterns chapter.

GenRocket Term	Think of it like...	Example
 Test Data Case	How you specify the exact test data needed for a test case	100 customers (unit), 5 Branches with 10 Accounts with 1,000 Customers (Integration)
 Test Data Rule	How to apply rules against your Test Data Case to only get the data you want	Platinum customer starting bank balance: \$10,000, Gold customer starting balance: \$5,000 - \$9,999
 Test Data Query	Ability to query specific production data values from a file or database	Query Platinum and Gold customer Account #'s to blend with synthetically generated data
 Test Data Stories	Stories combine a group of Test Data Cases	Combine Type Tables Test Data Cases with Account Table Test Data Cases
 Test Data Epics	Epics combine a group of Test Data Stories	Combine Integration Test #1 with Integration Test #2

Official screenshot: The vendor's table of design-time terms. A Test Data Case specifies the exact data one test needs (for example 100 customers), a Rule filters the data to what you want, a Query pulls specific production values to blend with generated data, Stories combine cases, and Epics combine stories.

[Source page ↗](#)

DO NOT CONFUSE THE STAGES WITH THE FIVE COMPONENTS

Model, Design, Deploy and Manage are *phases of work*. Domain, attribute, generator, receiver and scenario are *things in the product*. Several components appear in more than one phase: generators are chosen while designing, but a generator's output is only produced while deploying.

6.4.2 The four steps in the 2022 demo

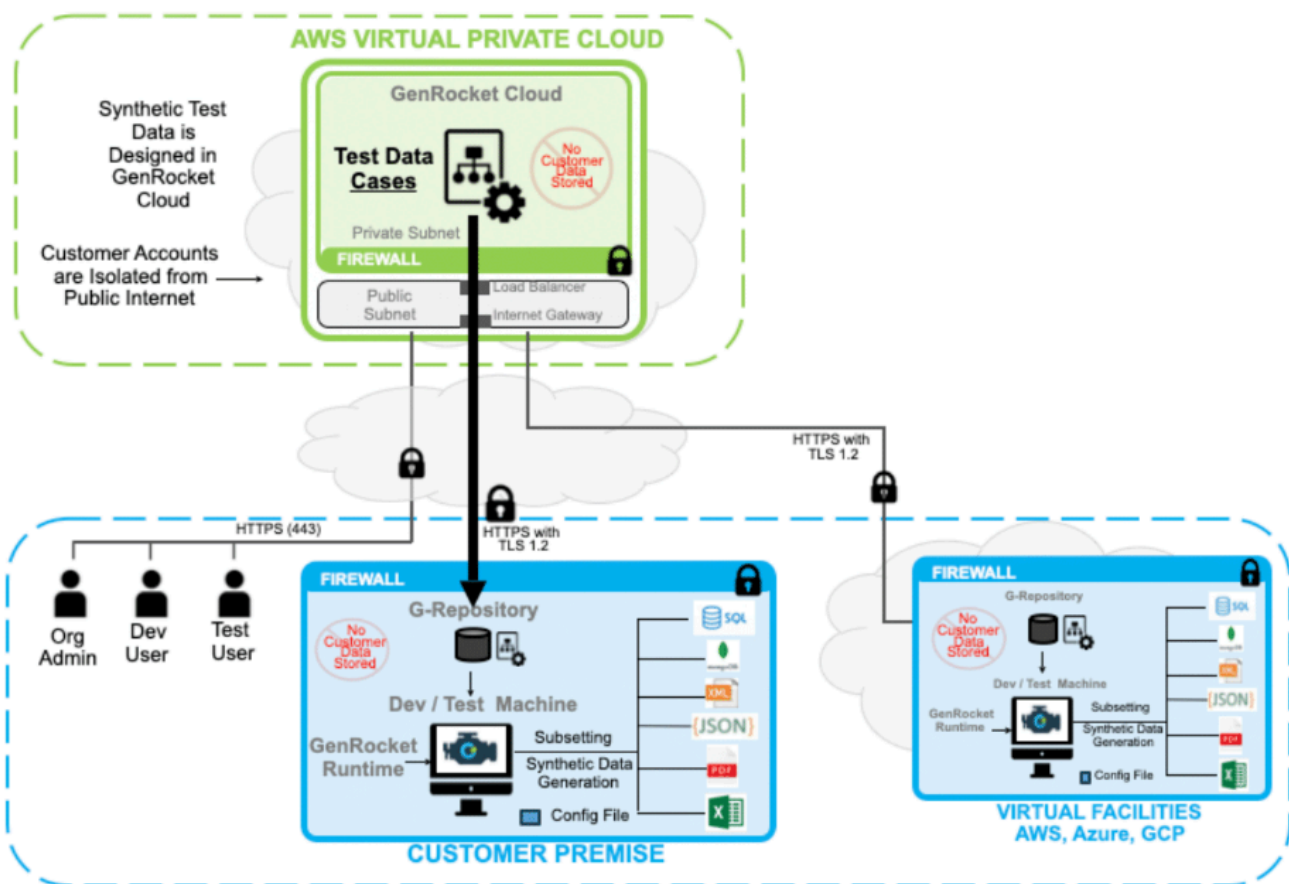
The 2022 overview video, spoken by GenRocket's CEO, sketches the method like this: a customer uploads a data model to the GenRocket cloud (Model), a test data engineer designs the data each test needs (Design), and a developer or tester wires the data into a CI/CD pipeline or test case and generates it on demand (Deploy). For Manage it names versioning and lifecycle management of the project template. Its short live demo then touches the steps:

Step	What the demo shows
Mode l	A new project is created with the XTS wizard (extract table schema), which reads the <i>metadata</i> , not the data. A domain referencing wizard then sets the parent and child relationships, and a generator tuning wizard adjusts the generators behind each attribute.
Design	A self-service module builds a test data case suite. The examples are unit, negative, load and integration cases, with the volume set per domain (the demo names a branch and an account domain), small and large setups, and rules that control the output.
Deploy	Generation is launched from Jenkins, a batch file or a shell script. In the demo the rows go into a SQL database, including a rule-controlled value such as an account balance.
Manage	A navigation view lets users move between projects, project versions, domains, attributes and individual generators. The video shows less of this step than the others.

The same wizards appear among the Project Versions icons in section 6.8.4. The 2020 webinar adds that there are nine ways to set up a project, XTS among them, and that XTS connects to a database over JDBC.

6.5 Architecture: a cloud for design, your machine for generation

The Architecture page says GenRocket is built on a *hybrid* architecture. The **cloud component** is where synthetic data is *modelled and designed*. The **on-premise component** is where synthetic data is *generated locally*. The cloud itself is offered as a Virtual Private Cloud (VPC) or a Dedicated Private Cloud (DPC).



Official screenshot: The architecture diagram. Top: the GenRocket cloud, where test data cases are designed, marked "No Customer Data Stored". Bottom: your premises (and your own cloud facilities), behind your firewall, where the GenRocket Runtime on a dev or test machine produces SQL, XML, JSON, PDF or Excel output. Connections are labelled HTTPS with TLS 1.2, and users (Org Admin, Dev User, Test User) reach the web platform over HTTPS 443.

[Source page](#) ↗

Read the diagram from top to bottom. A designer logs in to the web platform and builds the design. The design is then sent to the customer side as an encrypted scenario file. On the customer side, the runtime runs it and writes data into databases and files that never leave your network.

VIRTUAL PRIVATE CLOUD



Official screenshot: The same idea as three steps: create designs in the cloud, download them securely, generate synthetic data on your premises on Windows, Mac or Linux.

[Source page](#) ➤

WHY IT MATTERS

For a bank or hospital, the question "does our data leave the building?" decides whether a tool may be used at all. In this design, the cloud holds only the *design* (a data model and rules, not data). The only place rows are created is on a machine you control. Compare this with a masking tool (Informatica masking chapter), where real data is read and transformed.

The Architecture page links onward to security, team permissions, the runtime overview and "G-Repository", the component that keeps scenarios and configuration files synchronised across a company's machines. It also offers a downloadable PDF, "Components and Ports Configuration Guide", for deployments.

6.5.1 What the 2020 webinar adds to the architecture

In the webinar the CTO and CEO describe the same hybrid design in plain terms. The CEO lists the hosting choices as on-premises, a dedicated private cloud, and a virtual private cloud (VPC), and says about 90% of customers use the VPC; the current docs name the VPC and the dedicated private cloud. In the live demo the CTO builds a project, a domain and a receiver in the cloud, then downloads the result as an *encrypted scenario*. The file stays the same size (22 KB in his demo) whether it generates one row or a billion, because it holds instructions, not data. He runs it with the runtime, a set of Java programs, so it works on any machine with Java: Windows, Linux, Mac and even some mainframes. The CEO's conclusion is that all generation can happen on a test automation server inside the corporate network, which is what the "No Customer Data Stored" label in the diagram above means.

SPEED, SCALE AND PATENT: THE VENDOR'S CLAIMS

The 2020 webinar claims that a typical automated test case gets its data in about 100 milliseconds, that a run reaches thousands of rows per second, and that parallel runs produce millions or billions of rows in minutes. The CTO adds that importing metadata for 10,000 tables of 100 attributes takes around eight minutes. For contrast, the help centre says JDBC inserts average about 4,000 records per second (section 6.23.5). Real speed depends on the receiver, the database and the machine.

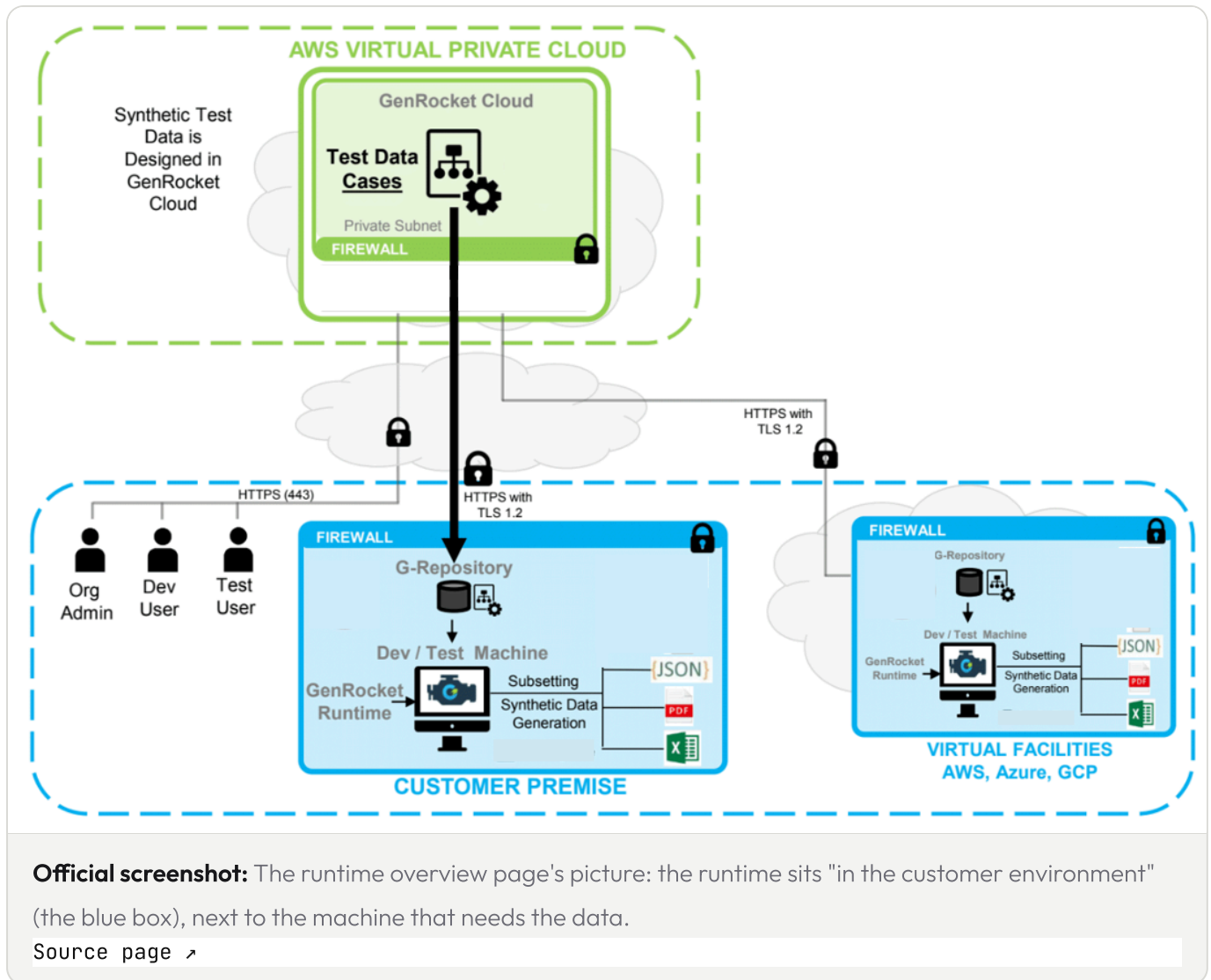
Both videos say GenRocket holds the only patent for synthetic test data generation (the 2020 webinar dates it to 2017). That is the vendor's statement.

6.6 The GenRocket Runtime

6.6.1 What it is

The runtime is a set of Java jar files, downloaded from GenRocket Cloud and installed wherever data generation is needed: a laptop, a shared machine, a server, a virtual machine or a container. The pages list these jars.

Jar	Job (docs wording, shortened)
<code>genrocket.jar</code>	Reads the command line, checks the user's licence, launches the engine
<code>gr-engine.jar</code>	The core engine that runs scenarios (and supports components such as the multi-user server)
<code>gr-generators.jar</code>	The generator classes
<code>gr-receivers.jar</code>	The receiver classes
<code>gr-grepository-servers.jar</code>	Creates and runs a G-Repository server
<code>gr-grepository-clients.jar</code>	Creates a G-Repository client



The page advises installing the runtime as close as possible to where the data is needed. Three setup styles are listed: a **local user machine**, a **shared central machine** where many people log in, and a **local or remote server**, including one used for automated testing in CI/CD.

6.6.2 What it needs (as the docs state)

Item	Documented value
Operating system	64-bit Linux (recommended), Mac OS or Windows
CPU and memory	At least 4 CPUs; at least 8 GB RAM (16 GB recommended)
Disk	At least 500 GB SSD (this is the page's "recommended hardware")
Java	8 (except 1.8u20), 11, 17, 21 or 24. Check with <code>java -version</code>
Browser for the web platform	Chrome, Microsoft Edge, Firefox or Safari
Rights	Admin rights, to set environment variables, install the runtime and install Java
Network	Port 443 (HTTPS) open from the machine to GenRocket Cloud, used for licence checking. The page also says URL or IP allow-listing may be needed

The runtime needs a few environment variables (settings your operating system hands to programs): `GENROCKET_HOME` , `PATH` , `JAVA_HOME` and `JAVA_OPTS` on the overview page. The Windows setup and troubleshooting pages spell the first one `GEN_ROCKET_HOME` . The right spelling depends on the version, so copy the spelling from the setup article for your operating system.

6.6.3 Profiles: how the runtime knows who you are

A scenario file is encrypted and tied to your organisation. To run it, the runtime must find a *profile* file (extension `.grp`) in a hidden folder named `.genrocket` in your home directory. The profile ties the machine to a GenRocket licence.

Profile	Used for
User Profile	Every person who runs the runtime. It matches the scenario to your organisation and checks that you have an enabled account
Server Profile	A server with no particular person, typically a CI/CD machine. An organisation administrator must first add the server in the web platform

Both are downloaded from the web platform. Whenever a resource value changes, the docs say you must download your profile again and replace the old one.

6.6.4 Port 443, online mode and the offline certificate

The runtime runs in *online mode* by default: when you run a scenario, it contacts GenRocket Cloud over port 443 to confirm the licence. The "Verify Port 443" page gives a quick test: open

`https://app.genrocket.com/license/check` in a browser, or run `curl` on the same address. If the port is open you see a short JSON message saying a valid user or server was not found. That odd-looking "failure" is the proof that the connection works.

If your company blocks that port, the docs offer an *offline certificate*: download a file from the web platform, put it in the `.genrocket` folder, and run scenarios without internet. The certificate is specific to one organisation and lasts 24 hours, so you download a fresh one each day.

OFFLINE CERTIFICATE PITFALL FROM THE DOCS

If the home directory stored in your `resource.home.directory` resource does not match the machine's real home directory, you get "Unable to validate license". The fix on the page is: correct the resource value, download the certificate again and retry.

6.6.5 Setting it up: the shape of the steps

The outline of the Windows setup follows; folder paths and exact variable names belong to the vendor page for your system.

1

Check the prerequisites.

An active user account, the cloud address reachable on port 443, admin rights, and Java installed.

Expected result: What you should see: `java -version` prints a supported version.

2

Download the runtime zip from the web platform (Options menu) and extract it into your home directory as a folder named `genrocket` .

The page warns against a nested folder such as `genrocket\genrocket-3.5.x` .

Expected result: What you should see: a `genrocket` folder with `bin` and `lib` inside.

3

Set the environment variables, open a new terminal and run `genrocket` once.

The first run downloads the other jars into `lib` and creates the hidden `.genrocket` folder.

Expected result: What you should see: no error. "genrocket is not a recognized command" means the folder name, home variable or `PATH` is wrong.

4

Download your user profile into `.genrocket` and fill in your organisation resources.

Resources hold folder paths for home, JDBC configuration and output (enter `N/A` for JDBC if you connect to no database).

Expected result: What you should see: a `.grp` profile file in `.genrocket` and your own paths in the Resources pane.

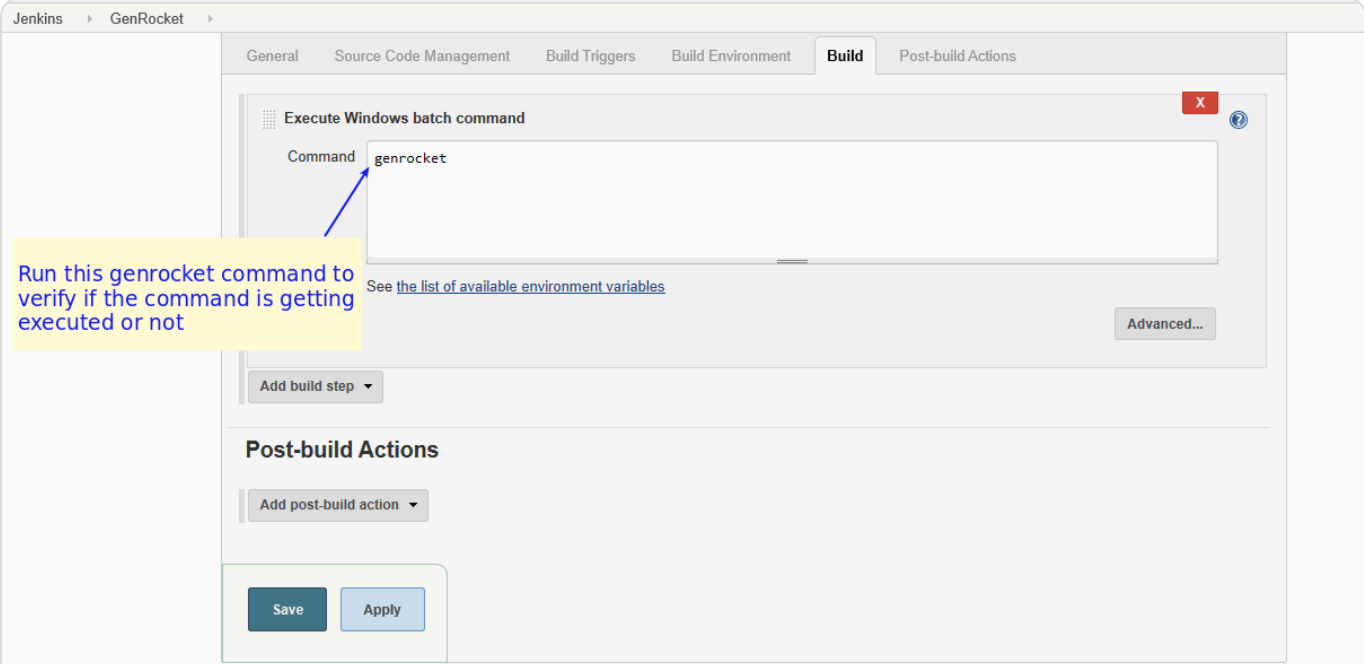
Typing `genrocket` alone, or `genrocket -h` , prints the runtime's command-line options for your version.

6.6.6 Updates, Docker and Jenkins at overview level

Updates. A full runtime update replaces the whole `genrocket` folder; a jar-only update is done from the command line. Release notes say which is needed.

Docker. GenRocket provides a Docker image you download from the web platform and load with `docker load` . A container forgets everything when it stops, so you *mount* host folders for the runtime, scenarios, configuration files, the `.genrocket` profile folder and the output.

Jenkins (continuous integration). Jenkins is a popular tool that runs build and test jobs automatically. The Windows and Mac/Linux Jenkins pages run a scenario as a build step. The outline: install the runtime on the Jenkins server and set the variables where Jenkins reads them (builds do not see your personal environment); register the server in the web platform and download its Server Profile into the server's `.genrocket` folder (no person is logged in on a build server); then create a Freestyle job whose build step first runs bare `genrocket` to prove Jenkins can find it, and then `genrocket -d -r SCENARIO_NAME.grs` . You should see a SUCCESS message in the console and your files in the output folder. Section 6.25.10 shows how the 2020 webinar combines this with Selenium tests.



Official screenshot: The vendor's Jenkins example: the build step first runs plain `genrocket` to check that Jenkins can find the runtime. Later in the same article the command is replaced by one that runs a named scenario file.

[Source page ↗](#)

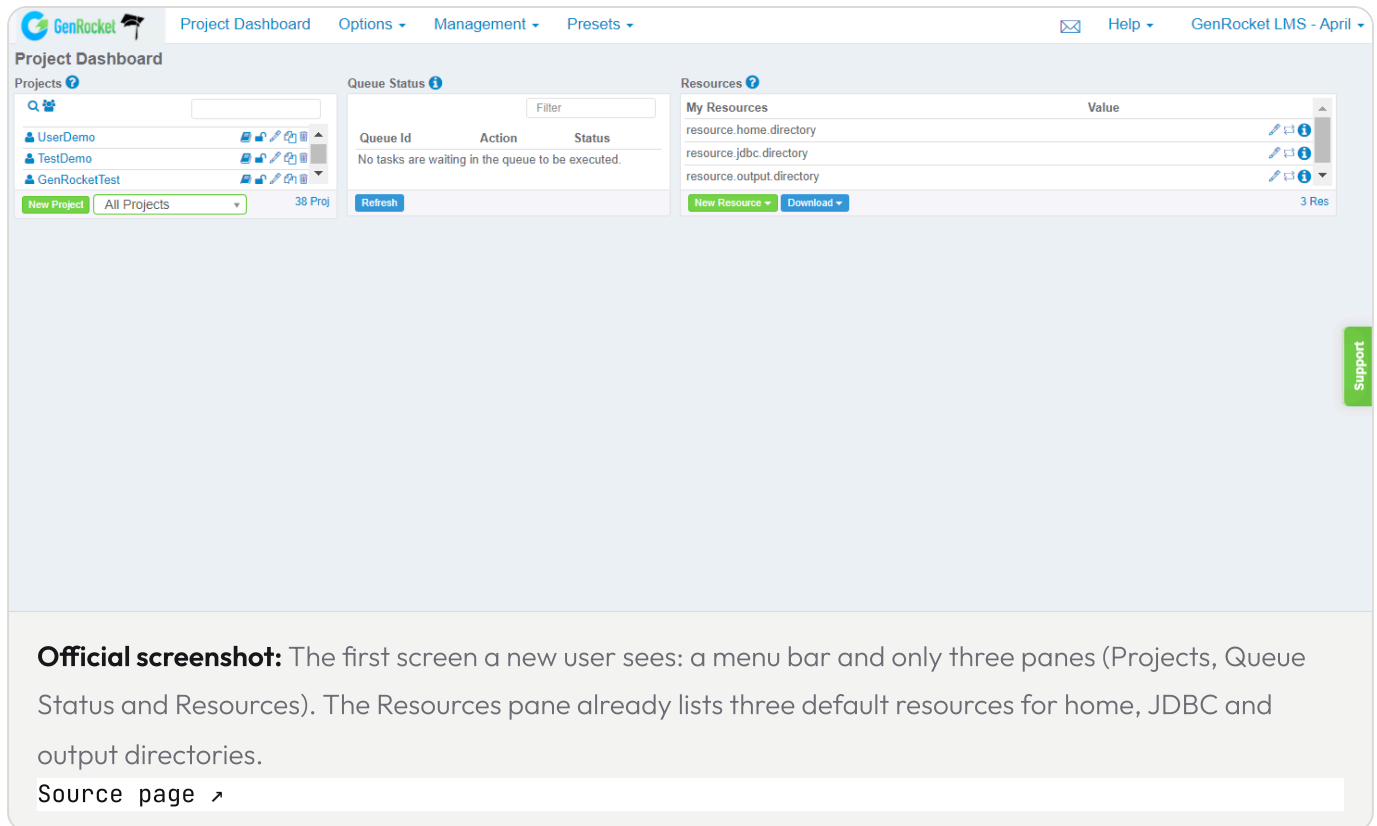
The CI/CD page adds two ways to trigger generation from a pipeline: a shell or batch command running the runtime, or a REST request to a GenRocket server component. It also describes Java and REST APIs that let a pipeline create or modify a scenario in real time, for example to change a loop count (how many rows) per test.

NOT COVERED HERE

Linux and Mac setup, the multi-user server, G-Repository administration and API method details are separate topics in the help centre and are not described here.

6.7 The web platform and the Project Dashboard

You reach the web platform with a link, a user name and a temporary password from a welcome email. The first-login page lists password rules and, after you accept the licence agreement, may ask you to change the temporary password. Organisations that use single sign-on or multi-factor authentication may log in differently; ask your administrator.



The docs describe those three panes this way: **Projects** lists and manages the organisation's test data projects; **Queue Status** shows tasks waiting to run, such as a project deletion or a data import (a Refresh button updates it); **Resources** holds organisation resources, which are named variables whose value differs per user (typically a folder path), so one scenario works on everyone's computer.

Select a project and more panes appear. This one screen is where most work happens. The complete screen, with a small demo project loaded, is below.

GenRocket

Project DashboardOptionsManagementPresets

HelpGenRocket FlightSchool - April

Project Dashboard

Projects

Filter

DemoProject

New ProjectAll Projects

Queue Status

Filter

Queue Id	Action	Status
No tasks are waiting in the queue to be executed.		

Refresh

Resources

My Resources

	Value
resource.home.directory	Users\April\Downloads
resource.jdbc.directory	N/A
resource.output.directory	Users\April\Downloads

Add ResourceDownload

3 Res

DemoProject - Version 1.0

Project Versions

VerDescriptionManage

1.0	Default 1.0 ...	
-----	-----------------	--

Manage

Domains

All Domains

Department	
Organization	
Address	
User	

New Domain4 Domains

Scenarios

All Scenarios

AddressScenario	
DepartmentScenario	
OrganizationScenario	
UserScenario	

New ScenarioBatch Download4 Scenarios

Scenario Chains

All Scenari...

No data available in table

New Scenario Chain

Scenario Chain Sets

All Scenari...

No data available in table

New Scenario Chain Set

Management

Project Version VariableConfiguration ManagementMigration ManagementGenerator Tags

New Project Version Variable Set

Name	Value	Actions
No data available in table		

Filter

Official screenshot:

The complete dashboard for a demo project. Top row: Projects, Queue Status, Resources. Below: Project Versions, Domains, Scenarios, Scenario Chains, Scenario Chain Sets, and a Management pane with four tabs.

Source page ↗

TDM Guide · Chapter 06: GenRocket (1): the building blocks

21 / 100

Pane	What it is for (docs)
Projects	View, create, filter and manage projects; filter by owner; open a browse window; assign categories and tags; create a project preset
Queue Status	Tasks waiting to be processed
Resources	Organisation resources; download your profile, a server profile or an offline certificate
Project Versions	Create, copy and manage versions; generator search; the Domain Referencing Wizard; a status report; generator tuning wizard; self-serve menu
Domains	The domains of the selected version; create or import domains; filter by category; domain report
Scenarios	Create, batch-create, batch-download and batch-delete scenarios
Scenario Chains and Chain Sets	Link scenarios so one command runs several
Management	Four tabs: Project Version Variables, Configuration Management (merge-configuration files for nested output), Migration Management, Generator Tags

The page also lists common icons. An open padlock locks an item: locked items cannot be deleted, and the lock stops changes made higher up from flowing down (you will meet locks again in the receivers and scenarios chapter). The main menu bar offers Options (downloads such as the runtime), Management, Presets and Help.

READING ORDER FOR THE PROJECT DASHBOARD

Work left to right, top to bottom: pick a *project*, pick a *version*, look at its *domains*, look at its *scenarios*. That is also the order in which you build things.

6.8 Projects, project versions and categories

6.8.1 What a project is, and how many you need

A **Test Data Project** contains the data model for one application or database you want data for. It contains aspects of your model (domains, attributes, generators, receivers, scenarios) but no customer data. This is also the unit of licensing: the Test Data Projects page says GenRocket is licensed based on the number of projects a customer needs.



Official screenshot: The vendor's simplest rule of thumb: one application with one underlying database is typically one project.

[Source page ↗](#)

The page's way to count: list the applications and databases you are testing. Database size, complexity of related tables, unrelated functions and special message formats may split one into several projects.

Situation	Minimum projects (docs)
One application, one database	1
Several unrelated modules, each with its own database	1 each
Multi-schema database whose tables are related	1 for all schemas (depending on complexity); unrelated schemas need a project each
Next release of the same application	The same project, with a new <i>project version</i>
A workflow across applications (for example customer, agent, quote, payment, policy)	At least 1 per application, database or complex file type, so that referential integrity holds between applications
Closed or packaged systems such as SAP or Salesforce	1 per module or group of related functions
SWIFT or EDI messages	1 per message type
Subsetting or masking a SQL database	1 per related group of tables

For ShopSphere's "Customer Orders" app with its single `SHOPSPHERE_PROD` schema, the docs' rule suggests one project, because the four tables are related.

6.8.2 Project types

The Project Types page lists three, and warns that GenRocket must enable the feature before you see the choice. If it is not enabled, every project is a Standard project.

Type	Meant for	Limits stated
Standard	The default; synthetic data directly for applications, APIs and systems	Up to five project versions; no limit on domains or attributes
Micro	Domain-limited uses: flat or nested files (CSV, JSON, XML), multi-page PDFs, structured or unstructured data, masking or generation	Behaves as an independent project; up to five versions
IPM	In-place masking of databases	Counts as two standard projects; one IPM project per database masked; up to five versions

6.8.3 Creating a project

New Project

Name*

SampleProject

Project Type*

Micro (Remaining: 37) ▼ i

Description

Micro project for file generation

Industry Type

-- Select Industry Type -- ▼ i

Team

-- Select Team -- ▼ i

Master Project

☐ i

Save

Cancel

Official screenshot: The New Project dialog from the vendor's example. Name and Project Type are marked required (the type only appears when enabled for the organisation); description, industry type, team and the Master Project checkbox are optional.

[Source page ↗](#)

1

Open the Projects pane in the Project Dashboard and click *New Project*.

If you cannot see this button, the docs say you lack permission; ask your organisation's administrator to assign the right role.

2

Enter a name, a type if offered, and a description.

The docs recommend a description. Choose the industry type if you can: they say it helps GenRocket assign the right generator to attributes.

3

Optionally pick a team or tick Master Project, then click *Save*.

A master project is a way to share master domains with other projects; teams are explained in the user section below.

Expected result: What you should see: the project appears in the list and extra panes appear for configuring it. A default project version is created automatically.

The screenshot shows the 'Project Dashboard' interface. The 'Projects' pane is highlighted with a red box and contains a list of projects: 'PortalDemoAH' and 'JSON-Test'. Below the list is a dropdown menu set to 'All Projects'. To the right, the 'Queue Status' section shows 'No tasks are waiting in the queue to be executed.' and a 'Refresh' button. Below the Projects pane, the 'PortalDemoAH - Version 1.0' section is visible. It includes a 'Project Versions' table with one row for version '1.0' and a 'Manage' button. To the right of this is the 'Domains' section, which has a search bar set to 'All Domains' and a list of domains: 'Account', 'Address', 'Customer', and 'Transaction'. At the bottom of the Domains section is a 'New Domain' button and a count of '4 Domains'.

Official screenshot: After saving, the new project appears in the highlighted Projects pane, and the Project Versions and Domains panes below it show its default version 1.0.

[Source page](#)

The next step after creating a project is to import or create the domains, which is the subject of the domains, attributes and generators chapter.

6.8.4 Project versions

A project is born with a version numbered 1.0. A new version is normally created when the release your domains and scenarios test goes to production, and the next release needs testing. Each version has its own domains, receivers, scenarios and configuration.

There are two ways to make one, both from the *Manage* menu in the Project Versions pane.

New Version

A blank slate. No domains or scenarios from before. Use it when the data model changes so much that starting fresh is easier.

The dialog asks for a version number and description, an optional default date format (for the whole version; a more specific format at attribute or generator level wins), and an optional *Enforce Type/Limit* checkbox.

Copy Version

Replicates the whole version, or a subset of domains, so the new version matches the next release of your software.

The docs state a limit: permitted users can copy a version five times in 24 hours, counted from the first copy.

Create Project Version

Version Number:

1.1 ←

Description:

Test Project ←

Default Date Format:

e.g. yyyy-MM-dd

Enforce Type/Limit:

☐

Save

Cancel

Official screenshot: The Create Project Version dialog: version number, description, optional default date format, and the Enforce Type/Limit checkbox, which keeps generated values within each column's declared type and size (explained in the generators chapter).

[Source page ↗](#)

Project Versions

Version	Manage 
1.7	         
1.6	         
1.4	         
1.2	         

Manage ▼

Official screenshot: A Project Versions pane with several versions. Each row has a set of icons for tools such as the wizards and reports mentioned earlier. The green Manage button opens the New Version and Copy Version menu.

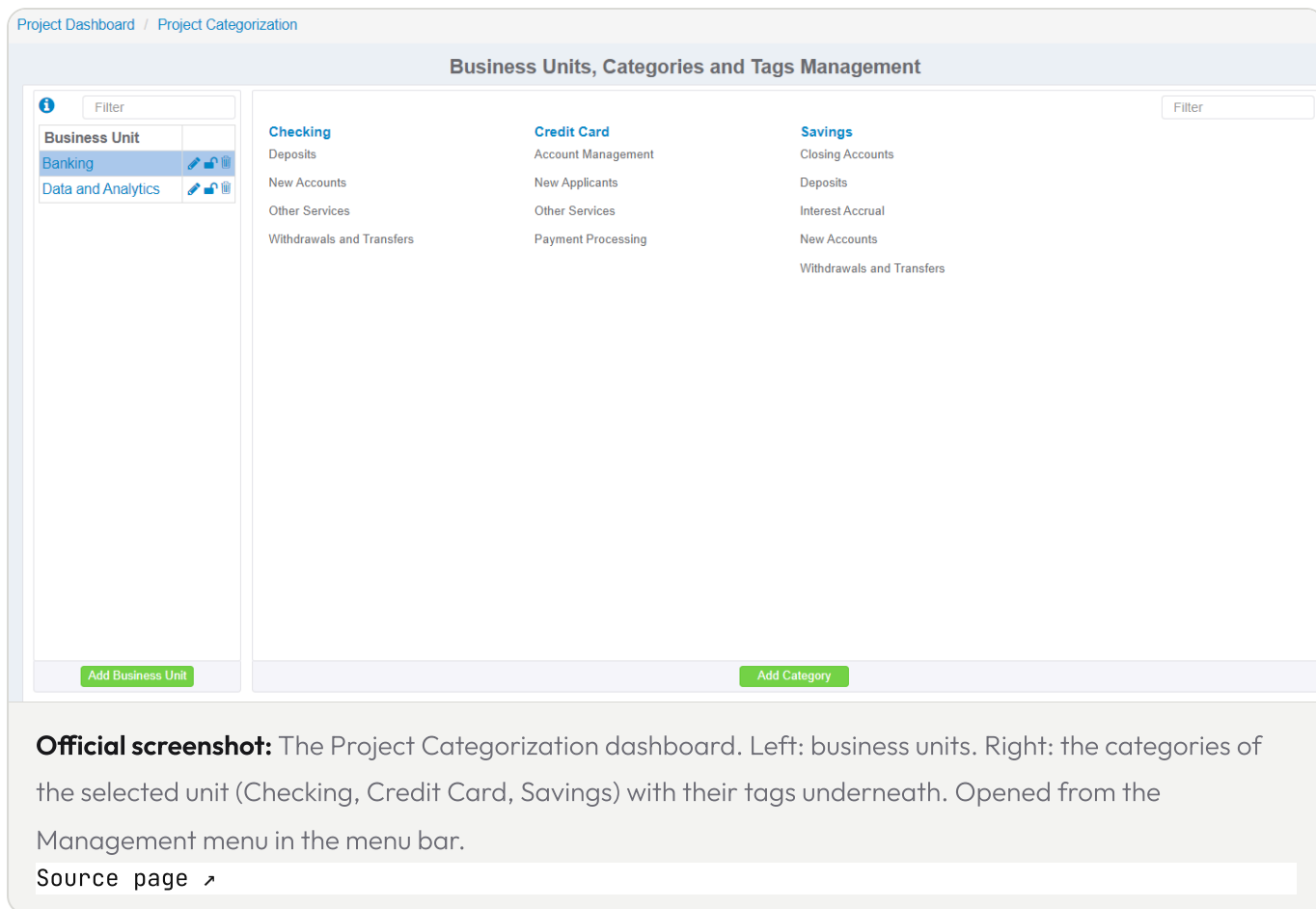
[Source page](#) ↗

The Project Dashboard page lists some of those icons: generator search, the Domain Referencing Wizard, a project-version status report (it flags attributes with no generator, among other gaps), a global seed assigner and a generator tuning wizard.

6.8.5 Project categorization

A company with hundreds of projects needs a way to find them. Project categorization lets an organisation administrator build a three-level list, then lets users filter projects by it:

- **Business Unit:** the top level, such as a department or team (the docs' example is "Bank").
- **Category:** a group of similar projects within a unit (Credit Card, Checking, Savings). No limit on how many.
- **Tag:** the lowest level, a keyword for a category (New Accounts, Payments Received). A category can have up to five tags.



Only an Org Admin creates the units, categories and tags; an Org Admin or the project's creator assigns them to projects; any user can then filter projects by them. The docs recommend using this only when you have many projects, and say a small organisation can find projects by name. A user can find a matching project, copy it, and adapt the copy: reuse, not rebuilding.

6.9 Users, roles and licensing basics

Licensing facts stated in the docs (prices and contract terms are not in the help centre):

- Licence is counted in *test data projects* (see above).
- Users are *named users*: a real, registered person whose email address is their unique ID. The number of users per organisation is unlimited.
- A named user may have a desktop and a laptop, but should run scenarios on only one at a time. The runtime checks for a valid, enabled profile on each run.

Role	What the docs say it can do
Organization Admin	Creates users, resets passwords, sets password expiry, accesses all projects, domains and scenarios, and can add or delete resources and organisation variables
Team Manager	Creates teams, adds projects and members to them and sets each member's permissions per project
General User	The default role; changes only their own account and works with projects, domains, attributes, receivers and scenarios not placed in a team. Inside a team, permissions depend on what the team manager granted
Enable Web Service API	Lets the user call GenRocket's web API

A *team* is how an organisation restricts who sees which projects: if a project is placed in a team, members get exactly the permissions they were given, and a user without permission to manage, for example, receivers sees them read-only. Other administration topics (multi-factor login, IP restrictions, SCIM provisioning) are in the help centre's User Management section.

6.10 ShopSphere plan in GenRocket vocabulary

The Model and Design thinking for the running example, before touching any screen. The product does not prescribe this design; it is built out in the following two chapters.

SHOPSPHERE PLAN IN GENROCKET VOCABULARY (EXAMPLE DATA)

Item	Decision
Project	One project, "ShopSphere Orders", one default version 1.0
Domains	CUSTOMERS, ORDERS, ORDER_ITEMS (PRODUCTS is left out of this example for brevity)
Relations hips	CUSTOMERS is the parent of ORDERS; ORDERS is the parent of ORDER_ITEMS
Generat ors	A counter for each key, a name generator for names, an email generator, a date generator for birth dates and order dates (details in the domains chapter)
Receivers	A delimited-file receiver for customers (CSV), a SQL-insert-file receiver for orders
Scenario	One scenario per domain, run in order, 50 customers to begin with
Where it runs	The GenRocket Runtime on a QA machine; later, optionally, as a pipeline step

The full build is in the receivers and scenarios chapter, and the hands-on exercise that generates 50 records is in the GenRocket hands-on chapter.

6.11 Common mistakes and gotchas

COMMON GOTCHAS

- **Expecting data to appear in the web platform.** The cloud holds designs, not data. Rows are created only when a downloaded scenario runs under the runtime on your machine. The "Preview" you see in the web platform is a sample for checking configuration.
- **Forgetting the profile.** Without a valid `.grp` profile (or offline certificate) in `.genrocket`, the runtime refuses to run. Download a fresh profile whenever a resource value changes.
- **Environment variables set but not picked up.** A terminal opened *before* the change keeps the old values; open a new one.
- **Port 443 blocked.** The runtime cannot validate the licence. Use the browser or `curl` test, then ask the network team or use the offline certificate.
- **Treating the number of generators or receivers as fixed.** Docs and videos give different, dated counts. Look the name up in your version.
- **Creating a new project for every release.** Use a new *version*; project count is a licensing unit.
- **Confusing the five components with the four phases.** See the warning in the four-steps section.

6.12 Recap

KEY POINTS

- GenRocket makes synthetic data from a model: no production rows are copied, and projects contain no customer data.
- Five components do the work: domain (table or file), attribute (column), generator (value recipe), receiver (output format), scenario (instruction sheet).
- "Model, Design, Deploy, Manage" is GenRocket's own methodology and the Test Data Automation four-step process; the vendor's videos present it as a demo and their statistics and speeds as claims.
- Traditional TDM copies, masks, subsets and refreshes production data; synthetic test data automation starts from metadata and adds unique, negative, high-volume and stateful data.
- Design happens in the cloud web platform; generation happens on your machine with the GenRocket Runtime, which needs Java, a profile, and usually port 443.
- Docker and Jenkins are two ways to run the same runtime in automated environments.
- A project is one data model and a licensing unit; a version is one release of it; categories help find projects.
- Roles and teams control who may change what; named users run scenarios on one machine at a time.

GenRocket: domains, attributes and generators

The first three of the five building blocks are **domains** (the tables), **attributes** (the columns) and **generators** (the recipes that decide each value). Together they are the Model and part of the Design phase. The worked example describes the ShopSphere CUSTOMERS, ORDERS and ORDER_ITEMS tables to GenRocket, links them so every foreign key is valid, and chooses a generator for every column.

The output side, receivers and scenarios, is in the receivers and scenarios chapter. Patterns built with generators (boundary, sequential, random, edge cases, null) are in the data patterns chapter.

6.13 Words to know first

Template domain

A domain as defined in a project version. Think of it as the master copy; a scenario gets a copy called a *scenario domain*.

Scenario domain

The per-scenario copy of a template domain. Changes to the template flow down to it unless it is locked.

Domain variable

A named setting that belongs to a domain, written `global.DomainName.variableName`, which generator parameters can refer to. Every domain gets three: `id`, `loopCount` and `seed`.

loopCount

How many rows (records) a domain generates.

Seed

A starting number for random generators. The same seed gives the same "random" data on every run.

Parent, child, sibling

GenRocket's way to express relationships between domains. A child has exactly one parent; "describing" tables are marked as siblings.

Primary attribute

The attribute marked as the key (shown with a key icon). Children refer to it.

Generator parameter

A setting on a generator, such as `startRange`. Required ones carry an asterisk in the web platform.

Linked generators

Several generators placed in one attribute so that later ones can use the values of earlier ones.

Preview

A sample of generated values shown in the web platform so you can check a configuration before downloading anything.

DDL

Data Definition Language: the SQL text that creates tables (`CREATE TABLE ...`). See the data basics chapter.

6.14 Domains: the tables of your design

6.14.1 What a domain is

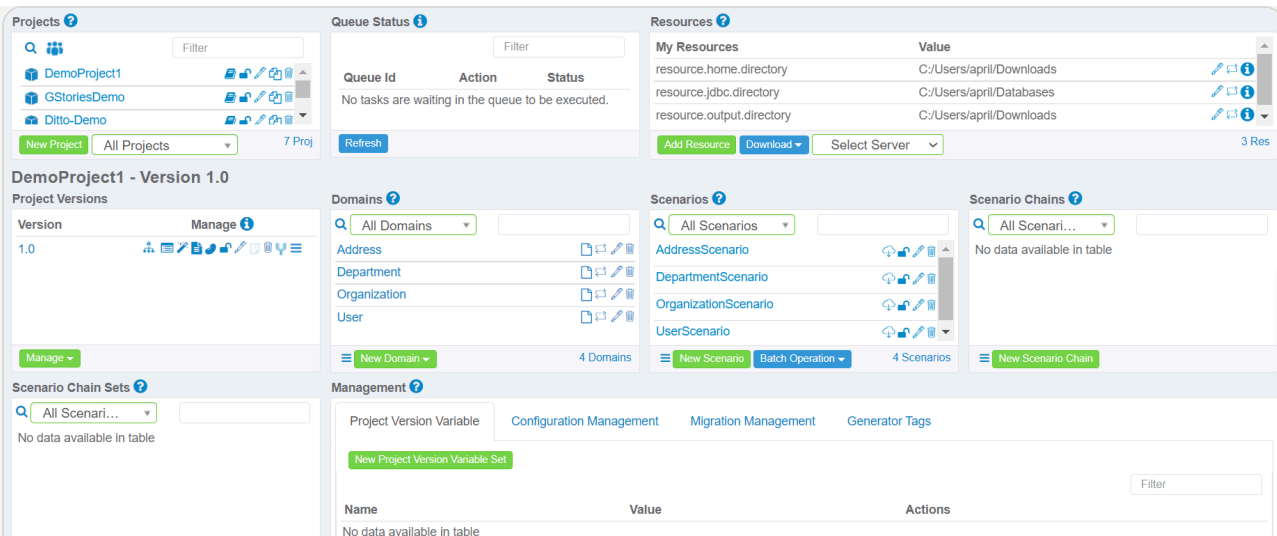
A domain "defines real-world objects", a noun such as a person, place or thing. When you model a database, a domain is a table and its attributes are the columns. The docs' own example has four tables, Company, Department, User and Address, each becoming a domain with its columns as attributes. At least one domain must exist in a project to generate any data.

A domain inside a project version is a *template*: it holds the attributes, generators and relationships. When you create a scenario, GenRocket copies the template into a *scenario domain*. Changes you make to the template cascade down to every scenario unless the scenario (or the scenario domain) is locked. It is like a letter template, except that you can pin a printed letter so later edits leave it alone.

The docs also state a rule: a template domain works independently and cannot really reference another domain's attributes. Relationships at the domain level are "just probable". The real references are made at the scenario level, where related domains are put together. You will see this again with ProxyReferenceGen below.

Each time a domain is created or imported, it gets three default variables, all initially 1 (details below): `id` , `loopCount` and `seed` .

The "New Domain" page says a domain name should contain only letters and numbers (spaces are now also allowed), so ORDER_ITEMS becomes a domain named `OrderItems` with `ORDER_ITEMS` in the *Table Name* field. The page does not say what happens with an underscore.

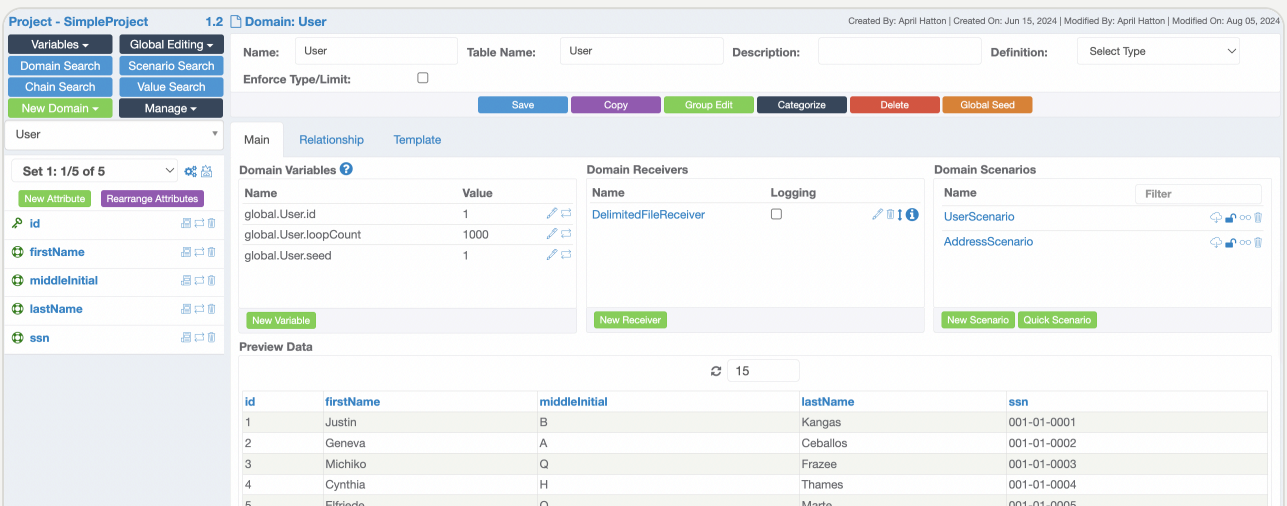


Official screenshot: Where domains live: the Domains pane of the Project Dashboard lists the domains (here Address, Department, Organization and User) of the selected project version. Clicking a domain name opens its Domain Dashboard.

[Source page](#)

6.14.2 The Domain Dashboard

Click a domain's name and the Domain Dashboard opens. It is the main working screen; the table below explains its areas.



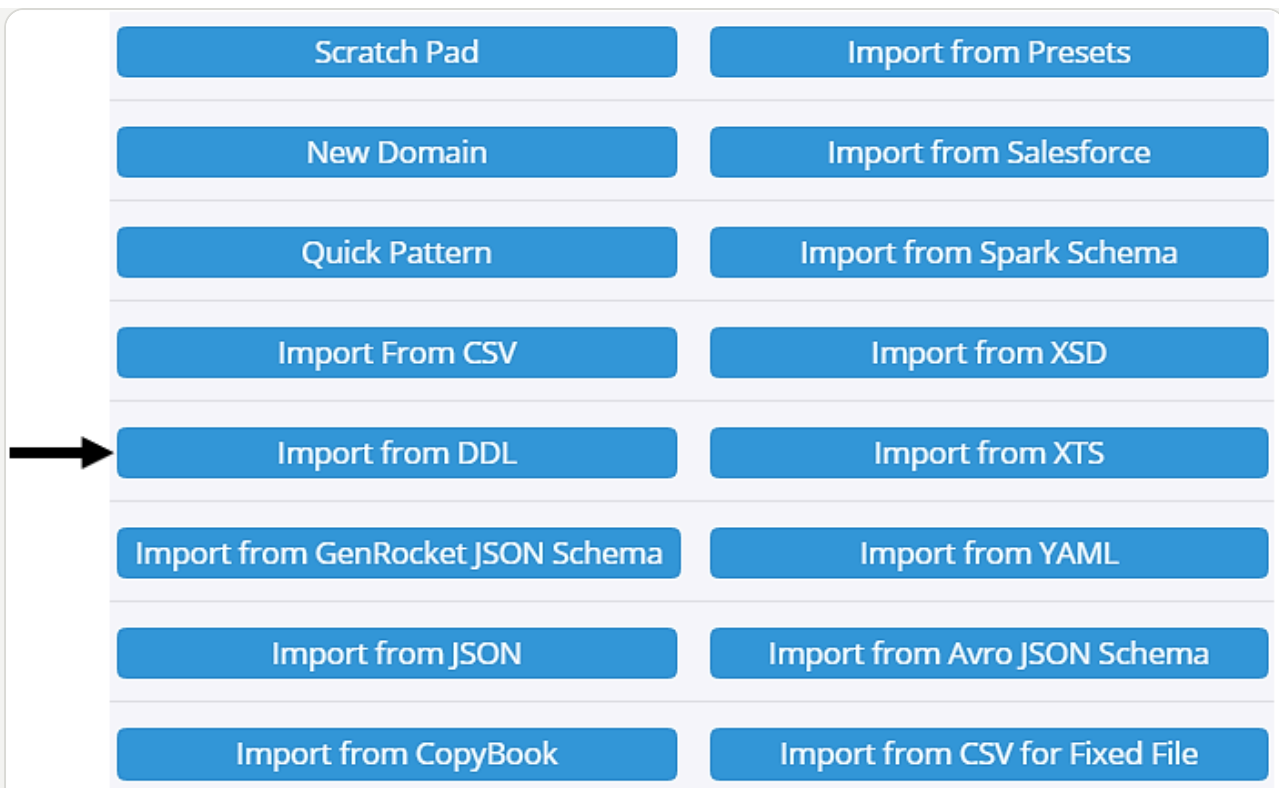
Official screenshot: The Domain Dashboard. Left: the attributes (a key icon marks the primary attribute). Centre: three panes for domain variables, receivers and scenarios. Bottom: a Preview Data grid that shows sample generated rows (the "15" box sets how many).

[Source page](#)

Area	What you do there (docs)
Top fields and buttons	Name, table name and description; <i>Save</i> , <i>Copy</i> the domain to another project version, <i>Group Edit</i> (change many attributes at once, or delete many), <i>Categorize</i> , <i>Delete</i> , and <i>Global Seed</i> (give all generators of the domain the seed value)
Attributes pane (left)	New Attribute, Rearrange Attributes, a gear icon (Browse and Manage Attributes), per-attribute icons. A set selector appears when a domain has more than 15 attributes, which are shown in groups of 15
Domain Variables	View, create and edit the variables (id, loopCount, seed and your own)
Domain Receivers	Add a receiver, enable logging (useful above 10,000 records), reorder receivers. The order decides which output is produced first
Domain Scenarios	Create a New or Quick scenario, download, lock and unlock, delete
Relations hip tab	Add or remove the referential parent and siblings, and see a diagram
Template tab	XML template files for the domain (not covered here)

6.14.3 Four ways to create domains

The page "How do I import my data model into GenRocket?" counts **seventeen primary methods**: DDL (single or multiple), XTS, CSV, CSV for fixed file, JSON, GenRocket JSON schema, standard JSON schema, XSD, YAML, presets, Spark schema, Avro JSON schema, CopyBook, Salesforce, Scratch Pad, Quick Pattern and New Domain. The count changes between pages, so do not memorise it. All are reached from the green *New Domain* button in the Domains pane.



Official screenshot: The New Domain menu: every way of creating domains in one list. The arrow in the vendor's picture points at Import from DDL, the method used below.

Source [page](#) ↗

For ShopSphere, four of these matter most.

Method	You supply	You get	Best when
New Domain	A name	One domain with only an <code>id</code> attribute	You are adding one small domain by hand
Scratch Pad	A domain name and a list of attribute names (optionally with type and size)	A domain with attributes and automatically assigned generators	Fast prototyping, or adding new columns to an existing domain
Import from DDL	The <code>CREATE TABLE</code> text of one table, or a DDL file of many	Domains and attributes, with type and size information	You have the SQL for the schema
Import from XTS	An encrypted schema file extracted from a live database	Domains, attributes and generators for as many tables as you extracted	The schema is large, or you can connect to the database

6.14.4 Creating a domain by hand

1

Select your project in the Projects pane.

The Domains pane only appears once a project is selected.

2

Click *New Domain*, then choose *New Domain*.

This is the single-domain method.

3

Type a name, for example `Customers` , and click *Save*.

Use letters and numbers.

Expected result: What you should see: the Domain Dashboard for the new domain, with just one attribute called `id` and three domain variables (`global.Customers.id` , `.loopCount` and `.seed` , each 1).

Project - DemoAH 1.0 Domain: User Created By: April Hatton | Created On: Dec 21, 2019 | Modified By: April Hatton | Modified On: Dec 21, 2019

Variables ▾ Global Editing ▾
 Domain Search Scenario Search
 Chain Search Value Search
 New Domain ▾ Manage ▾

User ▾

Set 1: 1/1 of 1
 New Attribute Rearrange Attributes

id

Main Relationship Template

Domain Variables ?

Name	Value
global.User.id	1
global.User.loopCount	1
global.User.seed	1

New Variable

Domain Receivers

Name	Logging
------	---------

New Receiver

Domain Scenarios

Name

New Scenario Quick Scenario

Preview Data

15

id
1
2
3
4
5

Official screenshot: What a freshly created domain looks like: one `id` attribute, the three default variables set to 1, no receivers or scenarios yet, and a preview that simply counts 1, 2, 3.

[Source page](#)

You would then add attributes. That is fine for two or three columns; for a whole database use an import.

6.14.5 Importing from DDL

You can paste one table's DDL, or upload a file with many. The vendor's sample single-table DDL creates a `user` table with an `id` primary key and columns such as `first_name varchar(25) not null`. Choosing the database vendor from a list tells GenRocket how to read the SQL dialect.

Multi-table import:

1

Open *New Domain* and choose *Import from DDL*, then *Import Multiple Domains from DDL*.

The other choice in that first dialog is single-table import.

2

Pick the vendor (for example `mysql`) and choose the DDL file.

Optionally tick *Enforce Type/Limit*, which makes every assigned generator respect each column's declared type and size (explained below).

Create New Domain from DDL

Vendor:

DDL File:

Enforce Type/Limit: ☐

Save

Cancel

Official screenshot: The multi-table DDL dialog: choose the database vendor, choose the file, and decide whether to enforce type and size.

[Source page](#)

1

Click **Save** and wait.

The file goes into a queue (you can watch the Queue Status pane). The user who started the import gets an email when it starts and when it ends.

Expected result: What you should see: new domains with their attributes, a primary key marked, and generators assigned automatically.

2

Optionally open the DDL audit from the Project Versions pane.

The "GenRocket Table Schema (GTS) Audit Details" option shows what was imported and any errors.

WHAT THE DDL IMPORT DOES NOT DO

The page is explicit: relationships between domains are *not* created automatically, receivers are *not* assigned and scenarios are *not* created. You must do these yourself (relationships with the Domain Referencing Wizard, below). It also lists SQL keywords the importer does not support, such as `DEFAULT 'N' :: bpchar`, `WITHOUT TIME ZONE` and `ON UPDATE CURRENT_TIMESTAMP(N)`. Remove them from the DDL before importing.

6.14.6 Importing a live database schema (XTS)

"XTS" stands for Extract Table Schema. This is the documented way to pull a database's structure straight from the database. Two steps:

1

Extract the schema with the runtime.

Run the runtime's `-xts` option with the path to a JDBC properties file and, optionally, the table names (`genrocket -xts /path/config.properties company,department,user`). With no names, every table is extracted. Options exist to exclude tables (`-xt` , `-xf`), and tables in a specific schema are written `schema1:tablename` . The properties file holds four required entries: `driver` , `user` , `password` and `url` .

Expected result: What you should see: an encrypted schema file, tied to your organisation. The page calls it a schema extract; it does not say more about what is read.

2

Import the file with the XTS Import Wizard.

Open *Self Serve Options* then *XTS Wizard* (or choose *Import from XTS* in the New Domain menu), choose the file, optionally enable Enforce Type/Limit, click Save and wait. The project is frozen until the import finishes.

Expected result: What you should see: domains, attributes and generators created for every extracted table, and the wizard's step 1 ready to be marked completed.

XTS Management Wizards

Project XTSWizardDemo Version 1.0

Step-1 ⓘ



XTS Import

Step-2 ⓘ



Domain Referencing Wizard

Step-3 ⓘ



Generator Tuning Wizard

Step-4 ⓘ



Receiver Assignment Wizard

Step-5 ⓘ



Scenario Assignment Wizard

Step-6 ⓘ



Domain Family Wizard

Edit ⓘ



Attribute Modification Wizard

Edit ⓘ



Receiver Modification Wizard

Delete ⓘ



Receiver Delete Wizard

Delete ⓘ



Scenario Delete Wizard

XTS Step Completion Checklist

Step 1 - XTS Import	Mark Completed
Step 2 - Domain Referencing Wizard	Mark Completed
Step 3 - Generator Tuning Wizard	Mark Completed
Step 4 - Receiver Assignment Wizard	Mark Completed
Step 5 - Scenario Assignment Wizard	Mark Completed
Step 6 - Domain Family Wizard	Mark Completed

Official screenshot: The XTS wizard hub: six numbered steps (import, referencing, generator tuning, receiver assignment, scenario assignment, domain families) with a completion checklist, plus edit and delete wizards for later changes.

[Source page ↗](#)

Each step has its own wizard that works on many domains at once, because the setup after importing hundreds of tables is also repetitive; you tick each step as complete so that the next unlocks.

SHOPSPHERE: WHICH METHOD?

ShopSphere has three tables. If the team gives you `CREATE TABLE` scripts, a DDL import takes minutes and keeps the primary key and sizes. If you can reach a dev copy of the database with a read-only account, an XTS extract avoids typing anything. Either imports a schema, not rows.

6.14.7 A live Scratch Pad demo from the 2020 webinar

In the 2020 webinar, GenRocket's CTO builds a domain on screen in a few minutes, which shows what the Scratch Pad method feels like. He creates a project called "live demo", opens the domain page and chooses Scratch Pad. He types attributes that nearly every application has (first name, last name, middle initial, password, username), then adds social security number, card number and phone number. GenRocket looks each name up in what he calls its data warehouse, a store of the

generators most used for common attribute names, and attaches a likely generator to each. It even matched "middle" after he misspelled it. A preview of sample rows appears at once.

The docs describe the same behaviour: *Create with Generators* assigns generators, guessing from the attribute name, domain name, industry type and what other users chose (section 6.18.1). They say nothing about tolerating misspellings, so type names carefully. They also say an attribute with "password" in its name is given the EncryptGen generator automatically.

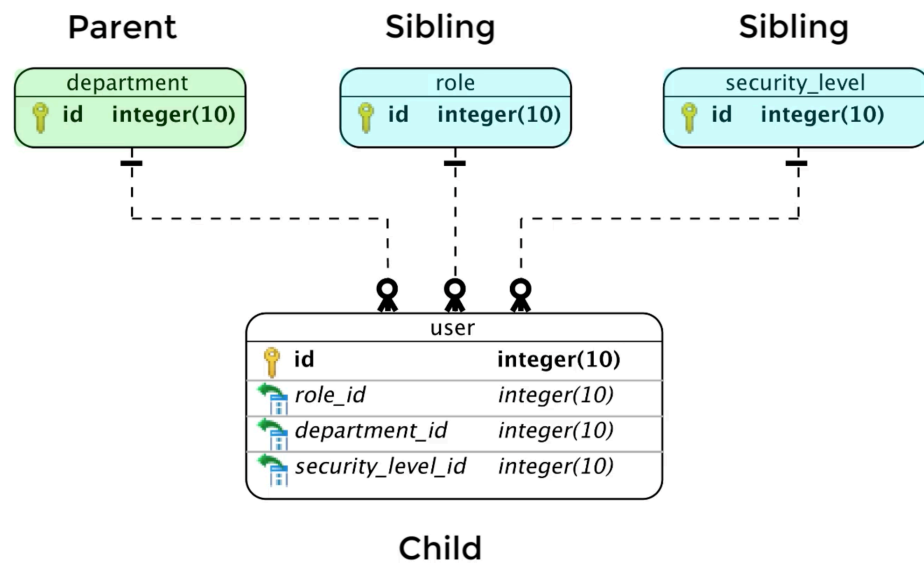
Two details from the same session. He says there were nine ways to set up a project in 2020; the docs now count seventeen (section 6.14.3). And in the Q&A he describes the schema extract (XTS, section 6.14.6) as reading metadata only (table and attribute names, types, foreign keys), after which a wizard lets you mark each domain's one parent and the rest become siblings. The docs match. His claim of 10,000 tables of 100 attributes loaded in about eight minutes is a 2020 vendor figure that the docs do not repeat.

6.15 Relationships: parent, child and sibling

Real tables point at each other: `ORDERS.customer_id` is a foreign key to `CUSTOMERS` (data basics chapter). If you generate 50 customers and 80 orders with random customer numbers, some orders will point at customers that do not exist, which breaks *referential integrity*. GenRocket's answer is to generate data *in order, from parent to child*, so a child can only pick a key a parent has already made.

The documentation says GenRocket "generates synthetic data with full referential integrity" because every domain has an id or primary attribute, and children refer to the parent's primary attribute. It also says GenRocket looks at relationships differently from a database. A database table can have several foreign keys. A GenRocket domain may have **only one parent but many siblings**, because generating sequentially along several parents would produce "a large amount of unnecessary data".

An entity may have only 1 parent but can have many siblings



Official screenshot: The vendor's picture of the rule: the `user` table has three foreign keys, but in GenRocket only `department` is the parent; `role` and `security_level` are siblings (the vendor's earlier diagram shows all three as plain parents, the way a database sees it).

[Source page](#)

Term	Meaning	Rule of thumb from the docs	ShopSphere
Parent	The domain whose rows are created first and which children point to	A real relationship between two tables, such as department and user	CUSTOMERS for ORDERS; ORDERS for ORDER_ITEMS
Child	A domain that refers to its single parent's primary attribute	Exactly one parent	ORDERS, ORDER_ITEMS
Sibling	A domain that only "describes" another, used for lookups	If the table is an adjective of another (role is an adjective of user), mark it a sibling	PRODUCTS describes an item: a sibling of ORDER_ITEMS

The scenario page adds a vocabulary pair: the *primary domain* is the one a scenario is created for, and the *root domain* is the first parent up the chain. For an ORDER_ITEMS scenario, ORDER_ITEMS

is primary and CUSTOMERS is root, so the scenario creates customers, then orders, then items, and every key matches.

WHY IT MATTERS

Valid foreign keys are the reason a test can load generated data into a real database and not see constraint errors. The principle (parents before children) is in the generation concepts chapter; GenRocket builds it into the domain model.

6.15.1 Setting a parent or sibling by hand

1

Open the *child* domain's dashboard and click the *Relationship* tab.

The relationship is stored on the child.

2

Click *Add Referential Parent* and tick the parent's radio button, then *Save*.

Radio buttons enforce the one-parent rule.

Add Parent of User

Filter

Primary Domain	☐
Address	☐
Department	☐
Organization	☒

Showing 1 to 3 of 3 entries

Previous 1 Next

Save Cancel

Official screenshot: Choosing the parent: the dialog lists the other domains with one radio button each, so only one can be chosen.

Source page ↗

1

For a lookup table, click **Add Referential Sibling**, tick one or more domains, and add them to the scenarios you want.

Expected result: What you should see: the Relationship tab draws a small diagram of the parent, the child and its siblings. If scenarios already exist, you are asked which scenarios should receive the new relationship.

6.15.2 The Domain Referencing Wizard

Setting parents one by one is tedious after a big import. The wizard shows every domain with parent radio buttons, sibling checkboxes and a diagram that updates as you click. It opens from the Project Versions pane or step 2 of the XTS wizard.

XTS Wizards / Domain Referencing Wizard

Domain Referencing Wizard

--Search Criteria-- ⓘ

☐ Filter Out Root Domains

☐ All ☐ Reviewed ☐ Not Reviewed

Root Domains	Reviewed
Account	☐
AccountLevel	☐
AccountType	☐
Branch	☐
CardProduct	☐
CardType	☐
Customer	☐
CustomerAccount	☐
Transaction	☐
TransactionType	☐

10 Total Domains

References for Account

Domains	Parent	Siblings
AccountLevel	☒	☐
AccountType	☐	☐
Branch	☐	☐

10 Domains to Review - 100% to complete
0 Domains Reviewed - 0% completed

Domain Relationship Diagram

Official screenshot: The Domain Referencing Wizard. Left: domains and a Reviewed checkbox. Middle: for the selected domain (Account), the parent radio buttons and sibling checkboxes, and a progress pie. Right: a relationship diagram that redraws as you choose.

[Source page](#)

1

Select a domain on the left.

Its possible parents appear in the middle pane.

2

Click a parent's radio button.

Siblings are ticked automatically for the other domains, and the domain is marked Reviewed because you changed it.

3

If a domain is already right, click **Mark as Reviewed**.

This is how you track progress when hundreds of tables exist.

Expected result: What you should see: the progress pie fills as domains are reviewed. In the XTS wizard, finish by clicking **Mark Completed** for step 2.

6.16 Domain variables, loopCount and categories

A **domain variable** is an instance variable of a domain, named `global.DomainName.variableName`. Generator parameters throughout the project can refer to it, and changing the value instantly changes everything that references it. Three exist by default:

Variable	Controls	Default	Example from the docs
<code>global.X.id</code>	The start of the <code>id</code> attribute. The <code>id</code> attribute's RangeGen reads it as <code>startRange</code>	1	Set to 100: ids run 100, 101, 102...
<code>global.X.loopCount</code>	How many rows the domain generates	1	Set to 10,000: ten thousand rows
<code>global.X.seed</code>	The seed for random generators	1	Same seed in two domains gives the same random numbers in both

You can add your own, such as `global.User.startDate`. Defaults cannot be deleted.

6.16.1 Changing how many rows

The docs describe three levels at which the row count can be set, from broad to narrow: the domain (template) level, which applies everywhere; the *test data case* level, which they recommend as the most flexible because different tests can have different counts; and the scenario level, for special cases. The steps below change it at the domain level.

1

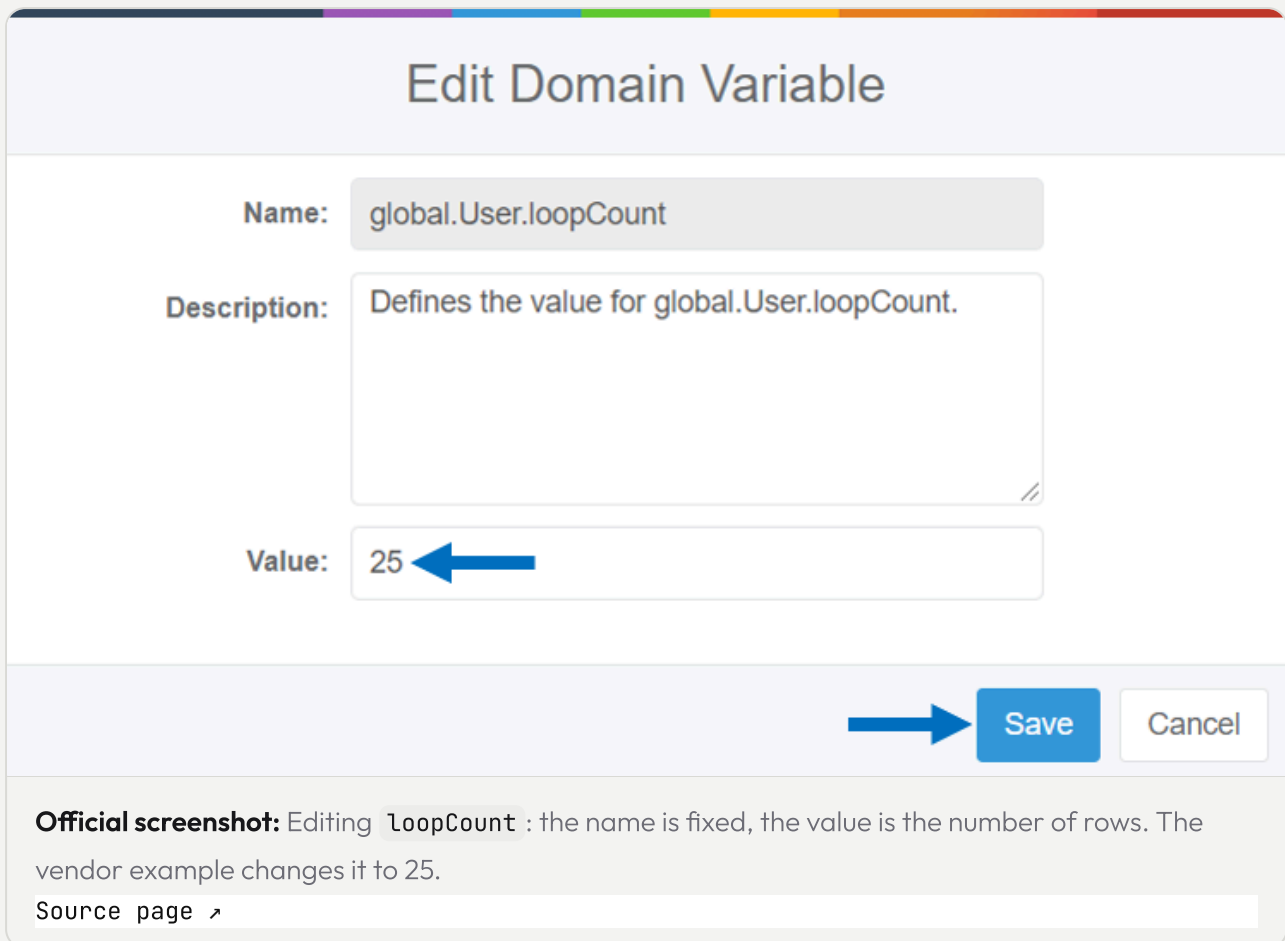
Open the domain and find the `loopCount` variable in Domain Variables.

It starts at 1, which is why a first run produces a single row.

2

Click its pencil icon, type the new value and click **Save**.

Expected result: What you should see: the Value column changes (in the vendor example, from 1 to 25).



Edit Domain Variable

Name: global.User.loopCount

Description: Defines the value for global.User.loopCount.

Value: 25

Save **Cancel**

Official screenshot: Editing `loopCount` : the name is fixed, the value is the number of rows. The vendor example changes it to 25.

[Source page ↗](#)

For many domains at once, the *Global Editing* menu has an "Edit Domains LoopCount" screen. One quirk: if a domain's first generator is a ListGen with `setLoop` true, the list length overrides the loop count.

6.16.2 Domain categories

When a version has dozens of domains, open a domain, click *Categorize*, create a category and save; then filter the Domains pane by category or use Advanced Search. Scenarios, chains and chain sets can be categorised too.

6.16.3 The id variable and loopCount, as shown in the 2020 demo

In the same demo, the CTO notes that a new domain already has three variables and starts with `id`. He calls it the starting point from which GenRocket guarantees referential integrity. The `id` attribute reads it, and so does the `username` generator, which builds its value from the `id`; when he types 1,000 into the variable, both change together. He adds that the `id` attribute can be made invisible so receivers do not output it while it keeps working; the docs' *Visible* setting (section 6.17) is that switch.

The docs agree on the mechanics: the variable is the start of the `id` range, every parameter that refers to it follows a change, and integrity comes from children referring to the parent's primary attribute (section 6.16). They differ on one point. The demo scenario generated 10,000 users, with the loop count described as the first thing a scenario reads, but the docs say a new domain's `loopCount` starts at 1. The demo does not show where 10,000 was entered.

6.17 Attributes: the columns

An attribute defines a characteristic of a domain, and a domain needs attributes for any data to exist. Importing creates them from the schema; otherwise you add them by hand. Every new domain starts with an `id` attribute that produces a unique number per row.

Click an attribute to open the **Attribute Dashboard**. The top half holds the attribute's own parameters; the bottom half holds its generators.

Domain: User / Attribute: firstName

Created By: | Created On: Apr 17, 2021 | Modified By: | Modified On: Apr 17, 2021

Attribute: **Properties**

Name: firstName ⓘ Type: ⓘ

Original Name: firstName ⓘ Size: ⓘ

% Null: 0 ⓘ Table Reference: ⓘ

Null Value: ⓘ * ⓘ Date Format: ⓘ

Output Condition: None ⓘ Description: ⓘ

☒ Visible ⓘ ☐ Primary ⓘ

☒ Required ⓘ ☐ Unique ⓘ

Save Copy Delete

Preview - Loop Count: 10 ⓘ

Dorinda

Karon

Rolanda

Sabra

Sherilyn

Jacquelynn

Celia

Dionna

Kristian

Previous 1 2 Next

Official screenshot: The top half of the Attribute Dashboard: the attribute's own parameters and, on the right, a preview of values it would generate.

Source page ↗

Attribute parameter	Meaning (docs wording, shortened)
Name / Original Name	Name is the camel-case name shown in the web platform. Original Name is the real column name as imported, and is what appears in output
% Null and Null Value	The percentage of rows that get a null (for negative tests) and what "null" looks like
Output Condition	Whether to leave the attribute out of a receiver's output when it is null, blank or both
Visible	If false, receivers do not see the attribute (useful for helper attributes)
Primary	Marks the key that children reference (a key icon appears)
Type, Size, Required, Unique, Table Reference	Imported from DDL or XTS: the column's declared type, length, null-ability, uniqueness and foreign-key target
Date Format	Format when the attribute generates a date

NAME VERSUS ORIGINAL NAME

ShopSphere's `customer_id` column becomes the attribute `customerId` (camel case), while its Original Name stays `customer_id`. Receivers such as the SQL-insert one use a `columnName` property key to write the real column name.

6.17.1 Adding an attribute

1

In the Domain Dashboard, click *New Attribute*, type a name and *Save*.

The name matters, because GenRocket uses it to guess a generator.

Create New Attribute

Domain

Name



Official screenshot: Adding an attribute needs only a domain and a name.

[Source page](#) ➤

1

Look at the generator that was assigned, change parameters if needed and click **Save Generator**.

Expected result: What you should see: for `firstName`, the NameGen generator with parameters `nameType`, `caseType`, `sortOrder`, `seed` and `waitAmount`, and sample names in the preview.

To add columns that appeared later in the real database, the docs use Scratch Pad: type the existing domain name exactly as it is in GenRocket (not the table name), list the new column names, and choose *Create with Generators*.

6.17.2 Referencing one attribute from another, and the ordering rule

Many generators have a `reference` parameter that takes another attribute's value. EmailGen, for example, can reference the `id` so each email is unique (`user1@example.com`, `user2@example.com...`). Double-click the reference field and pick the attribute with the plus button.

Select Attribute to Reference

Domain Attributes

Attribute Generator

Global Variables

Resources

Organization Variables

Attribute

Select

id

+

firstName

+

lastName

+

Cancel

Official screenshot: The reference picker: double-click a reference parameter and choose from domain attributes, other generators in the same attribute, global variables, resources and organisation variables.

[Source page](#)

Name: gen1

Generator: EmailGen 

prefix *: user 

reference *: #{User.id} 

suffix *: example 

extension *: com 

Save Generator

Remove Generator

Refresh Preview

Project Tag

Official screenshot: The result: the `reference` parameter now shows `#{User.id}`. Save the generator, then check the preview.

[Source page](#)

ONLY LOOK UPWARD

An attribute can only reference attributes *above* it in the list, because the engine fills attributes from top to bottom. Referencing one below returns an empty value. If you need to, click *Rearrange Attributes* and drag the attribute lower, then close the dialog to save.

Other attribute tasks, in brief: *Copy* an attribute to another domain or project version (rename it if copying within the same domain); use *Group Edit* to change one parameter on many attributes, or remove many attributes at once; and use the gear icon (Browse and Manage Attributes) to see which generators are in use.

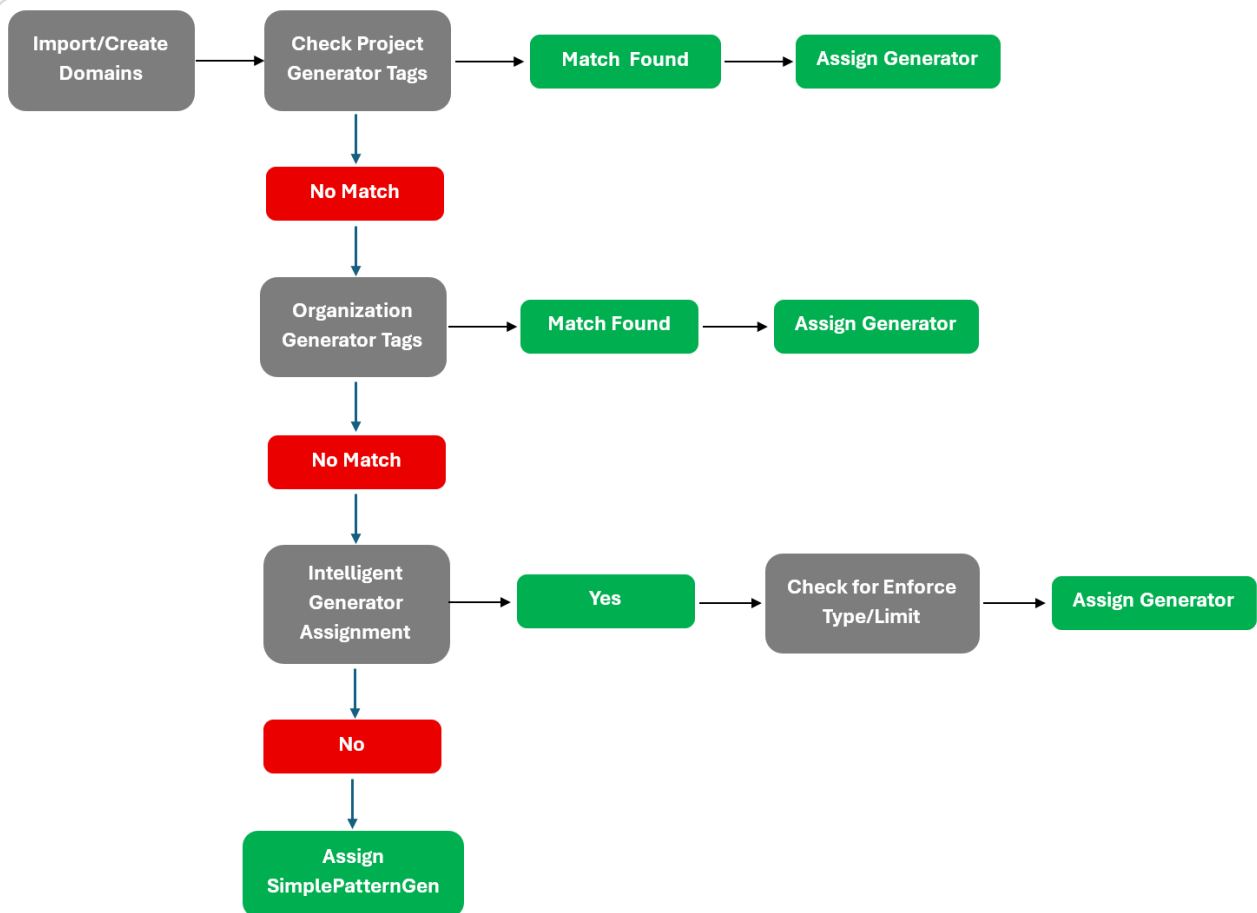
6.18 Generators: the recipe for each value

A generator creates synthetic data (or queries existing data) for one attribute. Every attribute needs one. GenRocket advertises over 765 generators (a growing number), grouped by purpose and, for many countries, by country. Generators produce raw values only; receivers format them (receivers and scenarios chapter).

6.18.1 How a generator gets assigned automatically

You rarely pick generators from scratch. For each new or imported attribute GenRocket tries these steps in order, according to the "How are Generators Assigned" page:

1. Is there a matching *Project generator tag*? If yes, use it.
2. Otherwise, a matching *Organization generator tag*?
3. Otherwise, an intelligent guess from the domain name, attribute name, the project's industry type and usage patterns of other users. If Enforce Type/Limit is on, the column's type is considered too (a `date` column gets `FlexibleDateRangeGen`, a `uniqueidentifier` gets `UUIDGen`).
4. If nothing fits, **SimplePatternGen**, which builds a value like `name1, name2, name3` from the attribute name and the row's id.



Official screenshot: The assignment flowchart: project tags first, then organisation tags, then the intelligent guess, and SimplePatternGen as the last resort.

[Source page ↗](#)

TREAT SIMPLEPATTERNGEN AS A TO-DO MARKER

It generates values like `street1`, which are unique but not realistic. When you see it on an attribute, the tool could not guess; pick a better generator. The Project Version status report flags attributes without a generator.

6.18.2 Replacing a generator on one attribute

1

Open the attribute and use the *Quick Generator Replacement* menu below the generator.

Typing part of a name filters the list (typing "town" offers `BulgariaTownGen` and `UKTownGen`).

2

Pick the generator and click *Replace*, then set its parameters and click *Save Generator*.

Expected result: What you should see: the new generator's name and parameters, and a fresh preview. Not sure of the name? Use *Remove Generator* and the *Add a Generator* dialog, which has categories and a search box.

Attribute Preview

Name: gen1

Generator: SimplePatternGen *i*

Enforce Type/Limit : ☐ *i*

prefix : Double Click to Add a Reference *i*

#[User id] *i*

town

BulgariaTownGen

UKTownGen

Quick Generator Replacement

Replace

Official screenshot: Quick Generator Replacement: type any part of a generator's name, choose it, then click Replace.

[Source page](#) ➤

6.18.3 Assigning one generator to many attributes

If `first_name` and `last_name` both need NameGen, do it in one go through the gear icon above the attribute list.

1

Click the gear icon (Browse and Manage Attributes) and tick the attributes.

Selected rows turn blue.

2

Type the generator name in *Select Generator*, click *Set Generator* and confirm with *Yes*.

The dialog warns that existing generators on those attributes will be replaced.

Confirm Apply Generator

Are you sure you want to apply the NameGen generator to 2 Attribute(s). This will replace any Generators currently assigned to Attributes.

No

Yes

Official screenshot: The confirmation: NameGen will be applied to two attributes, replacing what they have.

[Source page ↗](#)

1

Click **Modify Generator Parameters** to set a parameter for several attributes together.

Here you can give `firstName` the value `firstName` in `nameType` and `lastName` the value `lastName`, each in its own row, or tick rows and set a shared parameter such as `sortOrder`.

Global Edit Generator Parameters for - NameGen

Select Generator Parameter ▼

Save

☐ Check all

X	Domain	Attribute	Generator Name	name Type(*)	case Type(*)	sortOrder(*)
☐	User	firstName	gen1	lastName	capitalize	ascending
☐	User	lastName	gen1	lastName	capitalize	ascending
				firstName		
				maleName		

Official screenshot: Global Edit of generator parameters: one row per attribute, one column per parameter.

[Source page ↗](#)

The Generator Tuning Wizard (step 3 of the XTS wizard) does the same across the whole project version, with statistics on how well generators match attributes. Copying a configured generator to other attributes is possible too: *Simple Copy* to chosen attributes (up to 20 per bucket; it replaces the target's generator) and *Pattern Copy* to all attributes whose names follow a pattern.

6.18.4 Generator tags: steering the automatic choice

A *generator tag* is a rule "if an attribute's name matches this text, assign this generator". Each has a value, a match type (starts with, ends with, contains or matches) and an optional ignore-case flag. Project-level tags apply to all versions of a project; organisation-level tags apply everywhere. The page's example: every attribute named like `firstName` in a Canadian project should get `CanadaNameGen`. Tags live in the *Generator Tags* tab of the Management pane (project) or the Organization menu.

New Project Generator Tag

Name:

first name

Value:

firstName

Match:

Matches ✕ ▼

Ignore Case:

☒

Generator:

CanadaNameGen ✕ ▼

Save

Cancel

Official screenshot: Creating a project tag: when an attribute name matches `firstName`, assign `CanadaNameGen`.

[Source page ↗](#)

To see which generators GenRocket thinks fit an attribute, click the small icon beside it in the attribute list. A ranked list appears (in the vendor's example NameGen 84%, AustraliaNameGen 5%, and so on); the top one is the default, and a tag icon lets you promote another.

6.18.5 Generator families

The generator overview pages group generators into families; each generator has a "How do I use..." page with parameters.

Family	What it does	Examples named in the docs
Numbers and ranges	Counting, random numbers	<code>RangeGen</code> (increasing numbers; <code>startRange</code> , <code>endRange</code> , <code>jumpAmount</code> , <code>waitAmount</code> , <code>sortOrder</code>), <code>RandomGen</code> (<code>startRange</code> , <code>endRange</code> , <code>uniqueLimit</code> , <code>seed</code>)
Names and identity	People data	<code>NameGen</code> (<code>nameType</code> , <code>caseType</code> , <code>sortOrder</code>), <code>FullNameGen</code> , <code>EmailGen</code> (<code>prefix</code> , <code>reference</code> , <code>suffix</code> , <code>extension</code>), <code>PhoneNumberGen</code> (<code>areaCode</code> , <code>prefix</code>), <code>SSNGen</code> (<code>generationType</code> , <code>includeDash</code>)
List	Pick from a fixed set, in order or random	<code>ListGen</code> (up to 15 values; <code>list</code> , <code>random</code> , <code>unique</code> , <code>setLoop</code>), <code>ListCSVGen</code> and <code>ListExcelGen</code> (read a file), <code>PermutationGen</code> (all combinations)
String	Combine and alter text	<code>ConcatGen</code> , <code>SubstringGen</code> , <code>PadGen</code> , <code>StringCaseGen</code> , <code>StringRegexGen</code> , <code>RandomStringGen</code> , <code>StringReplacementGen</code>
Date and time	Dates in ranges and formats	<code>FlexibleDateRangeGen</code> (<code>format</code> , <code>startDate</code> , <code>startRange</code> , <code>endRange</code> , <code>nextBy</code> , <code>randomize</code>), <code>SystemDateGen</code> , <code>DateFormatGen</code>
Conditional (business logic)	Output depends on a rule or percentage	<code>EvalCaseGen</code> , <code>EvalCaseActionGen</code> , <code>SwitchGen</code> , <code>IfThenElseGen</code> (conditions); <code>CalcGen</code> (formula); <code>WeightGen</code> , <code>MultiWeightGen</code> (percentages); <code>EdgeCaseGen</code>
Query	Read real data from a database or file and blend it with generated data	<code>QueryOnceGen</code> , <code>QueryBeforeLoopV2Gen</code> , <code>QueryEachLoopV2Gen</code> , <code>QueryToMapGen</code> (they need a JDBC connection)
UUID	128-bit unique identifiers	<code>UUIDGen</code> (random), <code>UUIDSeedGen</code> (patterned), <code>UUIDTimeBasedGen</code>

Family	What it does	Examples named in the docs
Reference	Copy a value generated elsewhere	<code>ReferenceGen</code> , <code>ProxyReferenceGen</code> , <code>LoopSetGen</code> (sets the loop count)

Conditional generators can express rules such as "if age is under 18, return Child; between 18 and 60, Adult" or "50% Regular, 30% Premium, 20% VIP". The docs contrast them with test data rules (G-Rules), which attach conditions to cases instead of single attributes. The patterns they enable are taken further in the data patterns chapter.

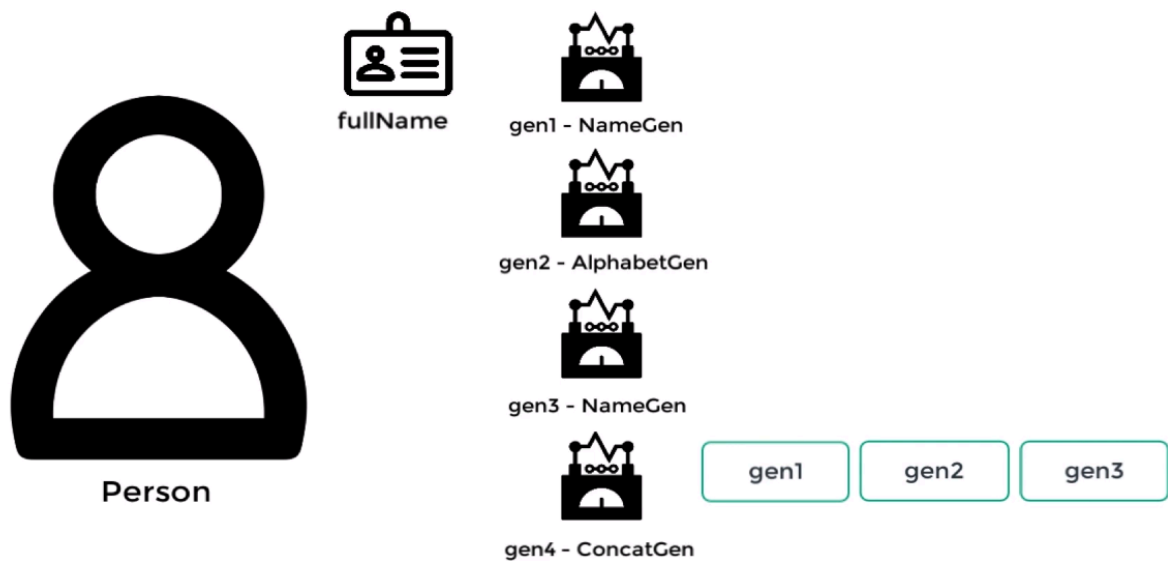
6.18.6 Seeding: repeatable randomness

If two runs of a "random" generator give different numbers, test failures cannot be reproduced. A *seed* fixes this: with the same seed, a generator gives the same set every run. A seed can be set on an attribute's generator or through the domain's `seed` variable (button *Global Seed* applies it to the whole domain). The data patterns chapter shows the docs' example.

6.18.7 Linked generators

One attribute can hold several generators in a row (`gen1` , `gen2` ...). They run left to right, a generator may use any generator to its *left*, and the *last* one gives the value. The docs' example builds a full name in a single `fullName` attribute: `NameGen`, `AlphabetGen` (middle initial), `NameGen` (last name), then `ConcatGen` joining the three. Without linking you would need four attributes and would have to hide three of them with Visible turned off.

Single Attribute Solution With Linked Generators



Official screenshot: Linked generators: one `fullName` attribute holds four generators; the last one joins the first three.

[Source page](#) ↗

6.18.8 ReferenceGen and ProxyReferenceGen: how a child finds its parent key

This pair is the mechanism behind foreign keys, and it confuses everyone once. The docs explain it as follows.

ProxyReferenceGen (domain level)

A placeholder. Because a template domain cannot see another domain, the foreign-key attribute holds this generator, whose only parameter is `startRange` (default 1000). It marks "a parent's key goes here".

Assigned automatically when you import with DDL or XTS and the column has a foreign key.

ReferenceGen (scenario level)

The real thing. When a scenario is created and the parent relationship exists, GenRocket swaps the proxy for ReferenceGen. Its one parameter `reference` points at the parent's primary attribute, such as `#AddressScenario.User.id`.

It can also be set up manually, as in the docs' "User and Address" example.

Domain: Address / Attribute: referenceUser Created By: April Hatton | Cre

Attribute		Properties	
Name	referenceUser	Type	Reference
Original Name	referenceUser	Size	
% Null	0	Table Reference	User
Null Value		Date Format	
Output Condition	None	Description	
☒ Visible ☐ Primary		☒ Required ☐ Unique	
<div> <div>Save</div> <div>Copy</div> <div>Delete</div> </div>			

Official screenshot: How a manual proxy is set up: the attribute has Type = Reference and Table Reference = User (the parent domain), then the ProxyReferenceGen generator is assigned.
[Source page](#)

This is why the project version status report flags "foreign key attributes that do not have the ProxyReferenceGen": without the proxy, the child will not find its parent when a scenario is made. The referenced attribute is the parent's *primary* attribute; to target another column, tick its Primary checkbox.

6.18.9 Enforcing data type and size

A generator does not know that `first_name` is `varchar(5)` unless you tell it. The **Enforce Type/Limit** option makes the generator keep values within the attribute's *Type* and *Size*. It is on automatically for generators assigned by a DDL or XTS import, can be enabled for a whole project version when you create it, and can be set manually per generator. The docs list integer limits it applies, for example 127 for `tinyint`, 32767 for `smallint` and 2147483647 for `int`.

- 1 **Open the attribute, set *Type* (for example `varchar`) and *Size* (for example 5) and save.**
These tell GenRocket the limits.
- 2 **Tick *Enforce Type/Limit* in the generator section and save the generator.**
Expected result: What you should see: the preview now shows names of at most five characters.

Attribute
Properties

Name
firstName
Type
varchar

Original Name
firstName
Size
5

% Null
0
Table Reference

Null Value
Date Format

Output Condition
None
Description

☒ Visible
☐ Primary

☒ Required
☐ Unique

Save
Copy
Delete

Preview - Loop Count: 10

Willa
Carli
Chant
Earth
Farra
Guill
Lucil
Darli
Glynd

Previous
1
2
Next

Attribute
Preview

Name: gen1
Generator: NameGen
Enforce Type/Limit : ☒
nameType *: firstName
caseType *: capitalize
sortOrder *: random

Linked Generators

Name
Alias
Generator

gen1
NameGen

Add Generator
Copy Generators
Delete All Generators

Start typing to select Generators
Add

Official screenshot: Type and size set on the attribute, and Enforce Type/Limit ticked on the generator.

Source [page](#)

6.18.10 Primary keys after the current maximum

When you insert into a table that already holds rows, new keys must not collide with old ones. The docs' recipe links two generators in the `id` attribute. `gen1` is `QueryOnceGen` with `resourceName` pointing at the JDBC resource, `query = select max(id) + 1 from <tableName>` and a `sampleValue` (5001 in the example) used for previews. `gen2` is `RangeGen` whose `startRange` references `gen1`.

Attribute **Properties**

Name: Type: Preview - Loop Count:

Original Name: Size:

% Null: Table Reference:

Null Value: * Date Format:

Output Condition: Description:

☒ Visible ☒ Primary ☐ Required ☐ Unique

[Save](#) [Copy](#) [Delete](#)

5001 5002 5003 5004 5005 5006 5007 5008 5009

Previous **1** 2 3 Next

Attribute **Preview**

Name:

Generator:

Enforce Type/Limit: ☐

startRange *:

endRange:

jumpAmount *:

waitAmount *:

iterate:

sortOrder *:

[Save Generator](#) [Remove Generator](#) [Project Tag](#)

Linked Generators

Filter:

Name	Alias	Generator
gen1		QueryOnceGen
gen2		RangeGen

[Add Generator](#) [Copy Generators](#) [Delete All Generators](#)

[Add](#)

Official screenshot: The finished setup: **gen2** (RangeGen) starts at **gen1** (QueryOnceGen) and the preview begins at 5001. In a real run the start value comes from the database.

[Source page](#)

The General FAQ adds a second pattern: write the parent's ids to a delimited file as you insert, then read them back with ListCSVGen at scenario level so the keys match.

6.18.11 Inside a few generators, as shown in the 2020 demo

The CTO opens individual attributes to show that every generator works the same way: parameters decide its behaviour. Each example below is followed by what the docs say.

- **A counter (RangeGen).** The **id** uses a range generator with a start, an end and a "jump". Starting at 1,000 it counts 1000, 1001, ...; setting the jump to 3 makes the preview step by 3. The RangeGen page confirms **startRange**, **endRange** and **jumpAmount**, and that with no end it runs to the largest long number.
- **Names (NameGen).** He says the name generator draws on lists of first, last, male and female names for about 40 million combinations. The NameGen page confirms this and gives the list sizes (1,000 male, 3,000 female, 10,000 last names).

- **A constant plus the id (ConcatGen).** He replaces the `username` generator with a concatenation generator, types the constant "user" and picks the `id` attribute as the second part. The ConcatGen page builds `BR01-1000` the same way, from a constant and a RangeGen value.
- **A hashed value.** He passes the id through an MD5 setting and the hash appears at the end of each output row. A *hash* is a one-way scramble: the same input always gives the same output, but the output cannot be turned back into the input. The EncryptGen page lists MD5, SHA-1, SHA-256, SHA-384 and SHA-512.
- **SSN and card numbers.** He says the SSN generator avoids numbers that are never issued, such as area number 666. The SSNGen page does not mention such a rule, so check the preview. For cards he picks Discover, MasterCard and Visa and says the numbers pass the *Luhn check*, the checksum formula payment forms use to catch mistyped numbers. The CreditCardGen page confirms Luhn-valid output and one on/off parameter per card brand. Luhn-valid does not mean real: no person owns these numbers.

6.18.12 Linked generators and "patterns, not randomness" in the 2020 Q&A

Asked how to generate values defined by a regular expression, the CTO answers in three parts. A regular-expression generator exists for that. More broadly, he says most generators do not produce random data, because random data is not good for testing; they produce patterns, and because linked generators can talk to each other, simple patterns combine into complex data. His examples are if/then/else, case, switch and a multi-way generator that works in percentages. For very complex rules there is a separate feature, test data rules, which takes if/then/else conditions. The docs name matching generators (IfThenElseGen, EvalCaseGen, SwitchGen, MultiWeightGen, StringRegexGen; see section 6.18.5) and describe test data rules (G-Rules) as the condition-based alternative. The data patterns chapter builds these patterns one by one.

6.19 ShopSphere design: generators for every column

The design below uses the documented generators; for `StringRegexGen` only the overview description is available. Domain names avoid underscores; the table name keeps them. "Domain variable" shows the setting that gives a sensible start.

DOMAIN CUSTOMERS (TABLE CUSTOMERS): `GLOBAL.CUSTOMERS.ID` = 1001, `LOOPCOUNT` = 50
(EXAMPLE DATA)

Attribute (original name)	Generator	Key settings	Why
customerId (customer_id), primary	RangeGen	<code>startRange</code> = the id variable (1001)	Unique, increasing, matches the sample row 1001
firstName (first_name)	NameGen	<code>nameType</code> firstName, <code>caseType</code> capitalize, <code>sortOrder</code> random, a <code>seed</code>	Realistic and repeatable
lastName (last_name)	NameGen	<code>nameType</code> lastName	Same generator, different type
email	EmailGen	<code>prefix</code> customer, <code>reference</code> the id, <code>suffix</code> example, <code>extension</code> com	Unique (the id is in it) and on a safe domain
phone	PhoneNumber Gen	<code>areaCode</code> 987	Sequential numbers, fake
ssn	SSNGen	<code>generationType</code> random, <code>includeDash</code> true	Valid-looking, not real people's numbers
dateOfBirth (date_of_birth)	FlexibleDateR angeGen	<code>format</code> yyyy-MM-dd, <code>startDate</code> 1955-01-01, an end range of about 45 years, <code>randomize</code> true	Adults born across 1955 to 2000
zipCode (zip_code)	ListGen	<code>list</code> 560001, 600017, 682001 ..., <code>random</code> true	A small fixed set, as in the sample rows
taxId (tax_id)	StringRegexG en	a regular expression for the tax-ID pattern	Pattern-shaped text

DOMAINS ORDERS AND ORDERITEMS (PARENTS: CUSTOMERS, THEN ORDERS)

Domain.attribute	Generator	Key settings	Why
Orders.orderId (primary)	RangeGen	id variable 5001	Unique order numbers
Orders.customerId (foreign key)	ProxyReferenceGen, replaced by ReferenceGen in the scenario	reference = the parent customer's id	Every order points to a customer that exists
Orders.orderDate	FlexibleDateRangeGen	format yyyy-MM-dd, recent start date	Dates in a window
Orders.totalAmount	RandomGen	startRange 10, endRange 500	Plausible amounts
OrderItems.itemId (primary)	RangeGen	id variable 9001	Unique
OrderItems.orderId (foreign key)	ProxyReferenceGen, then ReferenceGen	reference the parent order id	Valid links
OrderItems.productCode	ListGen	up to 15 codes	A fixed product set (PRODUCTS can be a sibling later)
OrderItems.quantity	RandomGen	1 to 5	Small whole numbers
OrderItems.unitPrice	RandomGen	range of prices	Positive prices

Row counts: Customers `LoopCount` 50. How many orders per customer and items per order is set per scenario, which is covered next. Because `total_amount` is generated independently, it will not equal the sum of its items. If a test needs that, use the conditional and calculation generators from the table above, or a G-Rule (the docs describe both).

LAB: MODEL SHOPSPHERE IN A PRACTICE PROJECT

Goal: Build the three domains with relationships and a preview of each. (Needs a GenRocket account and a practice project.)

1

Create the domains by DDL import or by hand.

Use the sample DDL of your team, or create `Customers` , `Orders` and `OrderItems` by hand with attributes from the tables above.

2

Run the Domain Referencing Wizard (or Relationship tabs).

Set Customers as the parent of Orders and Orders as the parent of OrderItems.

Expected result: Expected: a three-box chain in the relationship diagram.

3

Replace generators where the table says so, set parameters, and save each generator.

Expected result: Expected: previews show names, ids from 1001, emails like `customer1001@example.com` , and dates in your range.

4

Set the Customers loopCount to 50.

Expected result: Expected: the variable shows 50.

6.20 Common mistakes and gotchas

WATCH FOR THESE

- **Expecting a DDL import to set relationships.** It does not. Run the Domain Referencing Wizard.
- **Giving a table several parents.** Use one parent and mark describing tables as siblings.
- **Referencing an attribute below the current one.** The value is empty. Reorder first.
- **Leaving SimplePatternGen in place.** It means no better guess was found.
- **Forgetting Enforce Type/Limit.** Values may not fit the target column.
- **Forgetting that a new domain's loopCount is 1.** A first run gives one row.
- **Using ListGen for large lists.** It supports up to 15 values. The docs recommend ListCSVV2Gen or another file-based list generator for more. ListCSVGen loads the whole file into memory.
- **Exhausted unique values.** RandomGen with a small range and `uniqueLimit 1` stops with "Exhausted attempts to find the next valid random value" when the loop count is too high (the page's own example is a range of 1 to 20, a limit of 1 and 20 rows). Widen the range, raise the limit or lower the row count.

6.21 Recap

KEY POINTS

- A domain is a table (template), an attribute is a column, and each attribute needs a generator.
- Create domains by hand, with Scratch Pad, from DDL, or from a database with XTS. Imports do not set relationships or receivers.
- Each domain has three variables: `id` , `loopCount` and `seed` .
- One parent, many siblings. Parents generate first, so child foreign keys always find a parent.
- ProxyReferenceGen is the domain-level placeholder; ReferenceGen is the scenario-level real reference.
- GenRocket assigns generators by tags, then by guess, then SimplePatternGen. Replace them one at a time or in groups.
- Seeds make random data repeatable; linked generators build complex values; Enforce Type/Limit keeps values inside column limits.

GenRocket: receivers and scenarios

Besides describing the data (domains, attributes and generators), two jobs remain: say *what format* the rows must arrive in, and say *how many of what, in which order*. The first job belongs to **receivers**, the second to **scenarios**. Once both exist, you download the scenario, run it with the GenRocket Runtime, and files or database rows appear.

Receivers come first (the common ones in detail, the rest by family), then scenarios, scenario chains and related controls, an end-to-end ShopSphere build and a troubleshooting table. Patterns and the 50-record exercise are in the data patterns chapter and the GenRocket hands-on chapter.

6.22 Words to know first

Receiver

The component that takes a domain's raw generated values and writes them in a usable format: file, database insert, message and so on.

Receiver parameter

A setting for the receiver as a whole (where to write, what to name the file). A star in the web platform marks required ones.

Attribute property key

A setting for one attribute *within* a receiver, such as `columnName` (the header or column) or `include` (leave it in or out).

Output format

The shape of the result: delimited file, fixed-width file, JSON, XML, Excel, SQL statements and many others.

Organization resource

A named folder setting that each user fills in with their own path, for example `resource.output.directory`. Receivers refer to it instead of a hard-coded path.

Scenario

The saved instruction sheet: a primary domain, its parents, how many rows of each, and which receivers run.

Primary domain and root domain

The primary domain is the one the scenario is built for. The root is the first parent at the top of the chain.

Scenario chain

Several scenarios linked so that one command runs them in order.

.grs file

The downloaded, encrypted scenario file that you run with `genrocket -r`.

Stateful scenario

A scenario that remembers generator positions between runs, so a counter continues where it stopped.

6.23 Receivers

6.23.1 What a receiver is, and what it supports

A receiver receives unformatted raw data from a generator and "morphs it into a usable output format". The overview says 120+ types exist (the count grows), and groups what they reach: SQL databases, NoSQL databases such as MongoDB, web-service formats (REST, SOAP, XML, JSON), fixed-file formats (CSV, Excel, VSAM, NACHA, BAI2), big-data formats (Parquet, Avro), message formats (Kafka, MQ, SWIFT), packaged applications (Salesforce, SAP, Guidewire, Duck Creek), complex feeds (X12 EDI, FHIR), documents and images (PDF, JPEG, TIFF, PNG), and cloud services.

The vendor's receiver browser lists sixteen **categories**:

Category	What it holds (examples from the list)
Common	DelimitedFileReceiver, FixedFileReceiver, JSONFileReceiver, XMLFileReceiver, ExcelFileReceiver, SQLFileInsertReceiver, PDFFileReceiver
JDBC and MySQL	GenericSQLInsertReceiver, GenericSQLInsertV2Receiver, SQLUpdateReceiver, SQLBatchUpdateReceiver, StoredProcedureCallReceiver, MySQLInsertV2Receiver
Data File	OracleSQLFileInsertReceiver, SQLFileUpdateReceiver, ExcelFileMultiSheetReceiver, RandomAccessFixedFileReceiver
Segment And Merge	SegmentDataCreatorReceiver plus the JSON, XML, Avro, Parquet, Delimited, EDI and SWIFT merge receivers (nested data)
Masking	DelimitedFileMaskReceiver, JSONFileMaskReceiver, XMLFileMaskReceiver and others (replace values in an existing file)
Partition	Bulk-load receivers such as MySQLPartitionReceiver and DelimitedPartitionReceiver
Web Service, Message Broker, Compression	REST and SOAP callers, Kafka, zip-style receivers
Image, PDF, Template, In Memory, Parquet, Miscellaneous	Images, PDFs, template-driven output, memory formats, Parquet files, Groovy and Python script receivers

The "Supported Output Formats" page is a large checklist rather than prose. For each format it marks whether GenRocket supports it, whether the vendor validated it, whether a bulk-load receiver exists, and for databases whether a JDBC driver exists. It also notes that a missing format can be requested from the vendor. Use it when a project asks "can you produce X?".

Counts depend on the date and the source. The 2020 webinar spoke of about 61 to 63 receivers (the company started with four), the 2022 overview video of 91, and the current overview page of 120+; the help centre is the reference. On how new ones appear, the 2020 webinar's CTO said that for a proprietary format or endpoint GenRocket could not yet reach, it would add a receiver in two to three weeks, while the CEO put new generators at one to three business days. That is the vendor's 2020 statement, not a promise for today; the supported-formats page is the current way to ask for a missing format.

WHY IT MATTERS

Receivers are why the same design can feed a CSV for a spreadsheet user, SQL inserts for a database tester and JSON for an API test. You never change the generators for that, only add or swap a receiver. More than one receiver may be attached to a domain: they run in the order listed, which is useful when you need several outputs or several folders.

6.23.2 Adding a receiver to a domain

1

Open the domain's dashboard and click *New Receiver* in the Domain Receivers pane.

A receiver must be on a domain before its values reach any output.

Add A Receiver

A receiver is required for this domain to produce data. Click  or double click receiver in right area to add one. If you're unsure which receiver to add you can always add one later.

Receiver Categories

Tag Filter

- All Receiver Categories
- Common Receivers
- Compression Receivers
- Data File Receivers
- In Memory Receivers
- JDBC Receivers
- Masking Receivers
- Miscellaneous Receivers
- MySQL Receivers
- Partition Receivers
- Segment And Merge Receivers
- Template Receivers
- Web Service Receivers

Receivers

Tag Filter Receivers Filter

AmazonS3Receiver 	ArrayReceiver 	AvroFileReceiver 
AvroSegmentMerge... 	BAI2MappingReceiver 	DelimitedFileMaskR... 
DelimitedFileReceiver 	DelimitedFileSegme... 	DelimitedFileV2Rec... 
DelimitedPartitionR... 	DelimitedReceiver 	EDISegmentMergeR... 
EmailTestData Snipp... 	ExcelFileMulti Sheet... 	ExcelFileReceiver 
FileMaskMultiBlock... 	FixedDelimitedSeg... 	FixedFileReceiver 

[Cancel](#)

Official screenshot: The Add A Receiver dialog: pick a category on the left or type in the filter, then click the plus on the receiver you want.

[Source page ↗](#)

1

Click the plus on the receiver, optionally rename it, and click *Save*.

The name only has to be unique within the domain.

2

If scenarios already exist for the domain, tick the ones that should get the receiver and click *Add Receiver*.

Receivers copied into scenarios are separate from the template. Without this step, old scenarios will not have the new receiver.

Add DelimitedFileReceiver to Scenarios

Select All | Deselect All

Filter

	Scenario Name	Scenario Domain
☒	UserScenario	User

Add Receiver

Cancel

Official screenshot: Adding the receiver to existing scenarios: tick the scenarios that should receive it.

Source [page](#) ↗

1

Fill in the required parameters (marked with a star) and click *Save*.

Expected result: What you should see: the receiver listed in the Domain Receivers pane of the Domain Dashboard.

Many domains at once. In the Domain Dashboard open the *Global Editing* menu and choose "Add a New Receiver to Multiple Domains". This opens the Receiver Assignment Wizard: take ownership, choose the receiver and its parameters, filter to the domains, and execute. The vendor's example adds a DelimitedFileReceiver to the Department, User and Address domains in one pass.

6.23.3 The Receiver Dashboard

Click a receiver's name to open its dashboard. Four tabs matter.

Tab	Use
Parameters	The receiver's own settings. Starred ones are required to download and run a scenario
Attributes Property Keys	Per-attribute settings such as column name and include. Drag attributes to reorder output columns, use <i>Sync Receiver Attribute Order</i> , or the name-conversion menus (camel case is the default)
File Config	When available: when to start a new output file (for example by a count) and how to name files dynamically
Directory Config	When available: how many files per sub-folder and how to name sub-folders

Parameters
Attributes Property Keys
File Config
Directory Config

* path	<code>#{resource.output.directory}</code>	
subDir		
* fileName	Text.txt	
uniqueAttribute		
* delimiter	\t	
headerFilePath		
headerFileName		
* quoteTextData	None	
* booleanValue	true/false	
* nullValue	(null)	
* includeHeader	True	
* appendToFile	false	

Save

Official screenshot: A receiver's Parameters tab (DelimitedFileReceiver). Note the default `path : #{resource.output.directory}` , a reference to the organisation resource, not a typed folder.
[Source page ↗](#)

The column order of an output file normally follows the attribute order in the domain, which is shaped by the rule that attributes only reference attributes above them. If that order is not what the consumer wants, drag the attributes into the desired order in the Attributes Property Keys tab.

6.23.4 Output folders: organisation resources

Receivers do not hold a user's folder name. They use resource references. A new account has three: `resource.home.directory` , `resource.jdbc.directory` and `resource.output.directory` (the dashboard screenshot in the overview chapter shows them). The resource page recommends the home directory be the real home folder, and the other two be different folders, preferably sub-folders of home, because using the same path for all three can cause errors "or, in some cases, a process that never finishes". After

changing any resource value you must download your profile again and replace the old file in `.genrocket` ; otherwise you get the error "Unable to Download Users Profile" if a resource has no value, or stale paths if you forget.

6.23.5 Configuration files, V2 receivers and bulk load

- **Configuration file.** A file that tells a *merge* receiver how to combine data from different domains into one nested output. It defines file name segments, the segment files and the element layout. It is created in the Configuration Management tab and its name goes in the merge receiver's `configName` . Some imports (JSON, XML) create one for you.
- **V2 receivers.** Names contain "V2" just before "Receiver", for example `GenericSQLInsertV2Receiver`. They add two parameters, `resourcePath` and `resourceSubDir` , so a receiver can point to different configuration folders without creating more global resources. The vendor says V1 receivers will eventually be hidden from the picker; scenarios that already use them keep working.
- **Bulk load receivers.** Inserting through JDBC averages about 4,000 records per second, according to the page, which is slow for millions of rows. Bulk-load (partition) receivers write the database's native bulk-load file format instead. The page lists receivers for MySQL, Oracle, PostgreSQL, MS SQL, DB2, Sybase, Snowflake, Databricks, Parquet and delimited files.

6.24 The common receivers in detail

These are the common receivers. Parameters below are those the vendor pages list; a star means the page marks it required. Defaults and exact spellings can differ between product versions, so check the Parameters tab in your own project.

6.24.1 DelimitedFileReceiver (CSV and other delimited files)

Writes attribute values, separated by a delimiter, to a file. This is your CSV maker.

Parameter	Meaning
<code>path *</code> , <code>subDir</code>	Folder (and optional sub-folder) for the file
<code>fileName *</code>	Name of the file
<code>delimiter *</code>	The separator character. The screenshots show a tab (<code>\t</code>) as the default; type a comma for CSV
<code>includeHeader *</code>	Whether to write a header row of column names. Set to False for no header
<code>quoteTextData</code>	None, Single or Double quotes around text
<code>nullValue *</code> , <code>booleanValue *</code>	How null and true/false appear
<code>appendToFile *</code>	True adds to an existing file; false overwrites it
<code>uniqueAttribute</code>	An attribute used to stop a value being written twice
<code>headerFilePath</code> , <code>headerFileName</code>	Optional XML file that defines a multi-row header

Property keys per attribute: `columnName` (the header text) and `include` . With no header file, each attribute's *Original Name* becomes the header.

Parameters
Attributes Property Keys
File Config
Directory Config

Attribute name conversions ▼
Column name conversions ▼
Sync Receiver Attribute Order

S.No	*attribute ⓘ	columnName ⓘ	include ⓘ	order ⓘ
1	id	id	true	↑
2	firstName	firstName	true	↑
3	lastName	lastName	true	↑
4	ssn	ssn	true	↑
5	dateOfBirth	dateOfBirth	true	↑

Official screenshot: Property keys: one row per attribute with the column name that will appear in the header and an include flag.

Source [page ↗](#)

HEADER STILL SHOWING AFTER SETTING INCLUDEHEADER TO FALSE?

Two vendor reminders: re-download the scenario after any change, because the old file contains the old settings; and if it still shows, your receiver jar may be out of date (the jar-update page explains how to update).

6.24.2 FixedFileReceiver

Writes fixed-width columns (every value padded to a set length), the shape of many mainframe and legacy files. Parameters: `outputPath *`, `outputFileName *`, `subDir`, `lineSeparator` (for example `\n`), `delimiter`, `quoteTextData *` and `includeHeader *`. Property keys: `columnName`, `include` and `length`, the width of each attribute. Entering `# {BLANK}` or `# {EMPTY}` as a column name blanks a header cell.

6.24.3 JSONFileReceiver

Writes a flat JSON file (*not* nested). Parameters: `path *`, `subDir`, `fileName *`, `rootElementName` and `alwaysArray` (true forces a JSON array even for a single record). Property keys: `columnName`, `include`, `useQuotes` (numbers such as SSNs are often quoted on purpose). One discrepancy worth flagging: the parameter list says `path`, while the page's use case mentions `resourcePath`. They are probably the same setting in different versions; trust the Parameters tab you see.

6.24.4 XMLFileReceiver

Writes flat XML. Parameters include `path *`, `fileName *`, `rootElementName *`, `rowElementName *` and `exportStyle *`, which chooses how values appear: `ATTRIBUTES_ONLY` (each value as an XML attribute of the row element), `ELEMENTS_ONLY` (child elements with text) or `ELEMENT_VALUE` (child elements with a `value` attribute). Property keys: `name` and `include`.

6.24.5 ExcelFileReceiver

Writes an Excel file. The `type` parameter selects `xlsx` or `xls`; `path *`, `fileName *` and `type *` are required, along with quote, boolean and null formatting settings. The page warns that Excel generation is generally slower than delimited files, and that on Linux a "Permission denied" error can mean the temporary `profiles` folder under `/tmp` is not writable.

6.24.6 SQLFileInsertReceiver (SQL statements in a file)

Creates a file of standard SQL insert statements, ready to run in any SQL client. Nothing touches a database while generating, which makes it a safe first choice.

Parameter	Meaning
<code>path *</code> , <code>subDir</code> , <code>fileName *</code>	Where the <code>.sql</code> file goes and what it is called
<code>tableName *</code>	The real table name in the database
<code>batchCount *</code>	How many insert statements per batch written
<code>statementSeparator *</code> , <code>rowDelimiter *</code>	End-of-statement text (for example a semicolon) and line endings (UNIX/Linux/Mac-LF or Windows-CR/LF)
<code>quoteTextData *</code> , <code>nullValue *</code> , <code>booleanValue *</code> , <code>escapeCharacter</code>	Quoting and special-value formatting
<code>appendToFile *</code> , <code>uniqueAttribute</code>	Append or overwrite; avoid duplicates

Property keys: `columnName` (**must match the real column name**, such as `first_name`), `include` and `useQuotes`. This is where the attribute name versus original name distinction from the generators chapter pays off.

Parameters
Attributes Property Keys
File Config
Directory Config

Attribute name conversions
Column name conversions
Sync Receiver Attribute Order

S.No	*attribute <i>i</i>	columnName <i>i</i>	include <i>i</i>	useQuotes <i>i</i>	order <i>i</i>
1	id	id	true	true	↑
2	firstName	first_name	true	true	↑
3	lastName	last_name	true	true	↑
4	dateOfBirth	date_of_birth	true	true	↑
5	phoneNumber	phone_number	true	true	↑
6	username	username	true	true	↑
7	password	password	true	true	↑

Official screenshot: The SQL-insert receiver's property keys: the attribute `firstName` is written to the column `first_name`. Make the `columnName` values match the database.

[Source page ↗](#)

Example output from the vendor page: `insert into user(id,first_name,last_name,middle_initial,emailAddress) values ('1','Penelope','Spaeth','F','user@email.com');` . If a nullable column should show a real NULL rather than a quoted string, set `nullValue` to NULL and `quoteTextData` to Double (or the style your database wants).

6.24.7 Writing straight into a database: the JDBC receivers

GenericSQLInsertReceiver inserts generated rows directly into a table through JDBC. Its parameters include `resourceName` * (the resource that points to the connection information), `databaseName` *, `tableName` *, `truncate` *, `statementSeparator` *, `nullValue` * and `quoteTextData` *. Property keys are `columnName` , `columnType` and `include` . The newer **GenericSQLInsertV2Receiver** adds `resourcePath` , `resourceSubDir` , `continueOnError` and `logErrorDetails` . The V2 page says it suits "up to 10,000 rows into a table with 25 columns or fewer", and recommends the `GenericSQLInsertThreadReceiver` for larger loads.

TRUNCATE EMPTIES THE TABLE

The V2 page warns that its `truncate` parameter can truncate or delete all existing rows before inserting, and that this cannot be undone. The default is "notApplicable" (nothing deleted). Never point a truncating receiver at a database you have not backed up.

Direct database receivers need a *JDBC configuration file* (`config.properties`) on the machine running the scenario. It holds `driver` , `user` , `password` , `url` and, for receivers,

`batchCount` . Its folder comes from `resource.jdbc.directory` . You can check a connection with the runtime's `--test-jdbc` option before running a scenario.

Update receivers. *SQLUpdateReceiver* runs batched `UPDATE` statements whose values come from attributes (up to 20 attributes mapped to variables `var1` to `var20`), which is the basis of replacing real values with synthetic ones in a table. *SQLBatchUpdateReceiver* reads one or more update statements from an XML configuration file that can address several databases, using placeholders such as `#{firstName}` . The FAQ also names *StoredProcedureCallReceiver* for calling procedures, while the FAQ says *SQLUpdateReceiver* can run DDL and DML scripts before or after a load, and another page uses *SQLUpdateV2Receiver* to create a database table.

6.24.8 Nested data: segment-and-merge receivers (overview)

JSONFileReceiver and *XMLFileReceiver* are flat. For *nested* output (customers containing orders containing items) GenRocket uses two receiver types together:

- 1 Add *SegmentDataCreatorReceiver* to each domain except the merge domain.**
Each domain writes its rows as a small XML "segment" file, tagged with its parent's id so relationships survive.
- 2 Create a merge domain and add a merge receiver, such as *JSONSegmentMergeReceiver* or *XMLSegmentMergeReceiver*.**
The merge receiver reads the segments and the configuration file, and writes nested files. Others exist for Avro, Parquet, delimited, fixed, EDI, HL7, SWIFT and SOAP.
- 3 Put the merge scenario last in a scenario chain, create the configuration file, and download both.**
Merging only works once every segment exists. Imports of JSON, XSD and some other models set most of this up for you.

The JSON merge page lists parameters such as `outputPath` , `configName` , `segmentPath` and `recordsPerFile` . If the root domain's loop count is 50 and `recordsPerFile` is 8, seven files are made (six with 8 records, one with 2). The vendor pages walk through a full nested build.

6.24.9 Masking receivers (overview)

GenRocket also masks: it replaces sensitive values with synthetic ones, a method the file-masking overview calls Synthetic Data Masking (SDM). The file-masking receivers take an existing file and write a new one with the targeted values replaced:

DelimitedFileMaskReceiver, ExcelFileMaskReceiver, JSONFileMaskReceiver, XMLFileMaskReceiver, ParquetFileMaskReceiver, ORCFileMaskReceiver, EDIFileMaskReceiver and FileMaskMultiBlockReceiver (fixed files such as VSAM). The same documents describe database masking in place (IPM). The vendor states that production data is never sent to GenRocket, since only metadata is used. How this compares with masking in other tools is in the Informatica masking chapter and the masking concepts chapter.

6.25 Scenarios

6.25.1 What a scenario is

A scenario contains instructions that tell the GenRocket engine what synthetic data to generate and how much. You select the domains, say how many rows each produces, decide their order, download the scenario to a computer, and run it with the runtime. Domains, attributes, generators and receivers are its building blocks, so all four must exist first.

A scenario is for *one primary domain*. If the primary domain has parents, the scenario pulls them in as helper domains so that foreign keys are valid: a scenario for Address with parents User and Department also creates departments and users. To output data for several related domains, create a scenario per domain and link them in a chain (below). The docs recommend a scenario for every domain, except pure helper domains.

6.25.2 Creating a scenario

There are three ways: *New Scenario* (with or without a primary domain), *Quick Scenario* (named after the domain and built from its relationships) and *Batch New Scenarios* (a quick scenario for each of several domains).

1

Open the domain's dashboard and click *Quick Scenario*.

Quick is the fastest way when the domain's parent relationships are already right, because it builds the parent domains in for you.

Domain: User Created By: April Hatton | Created On: Apr 17, 2021 | Modified By: April Hatton | Modified On: Apr 18, 2021

Name: User Table Name: User Description: Definition: Select Type

Enforce Type/Limit: ☐

Save Copy Group Edit Categorize Delete Global Seed

Main Relationship Template

Domain Variables

Name	Value
global.User.id	1
global.User.loopCount	1
global.User.seed	1

New Variable

Domain Receivers

Name	Logging
DelimitedFileReceiver	☐

New Receiver

Domain Scenarios

Name Filter

New Scenario Quick Scenario

Official screenshot: Quick Scenario is in the Domain Scenarios pane of the domain.

[Source page](#)

1

Accept the name (it matches the domain), add an optional description and click **Save**.

Expected result: What you should see: the Scenario Dashboard for the new scenario.

Name Your Scenario

This action will automatically create a Scenario for you including Parent and Sibling relationships with referential integrity for the User Domain.

Name: User

Description: Generates User Data

Save Cancel

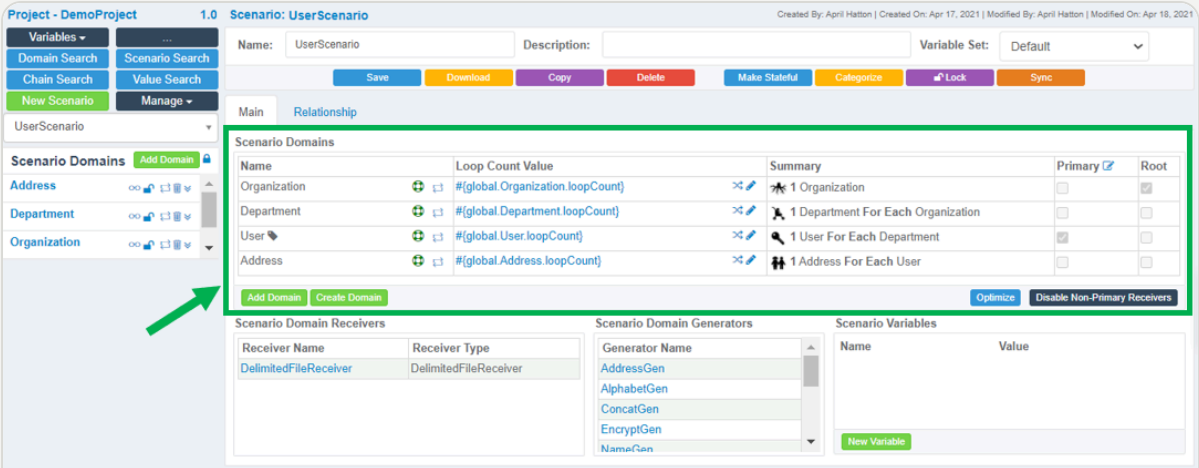
Official screenshot: The Quick Scenario dialog states what it does: it builds in parent and sibling relationships so referential integrity holds.

[Source page](#)

For *New Scenario*, you enter a name and then choose to create it with or without a primary domain; with one, a wizard asks which domain and whether to add referential integrity automatically. Choose "without" when you want a different parent for this scenario only (see "Scenario domains and relationships" below).

6.25.3 The Scenario Dashboard

Open a scenario from the Scenarios pane of the Project Dashboard (or from the domain's Domain Scenarios pane).



Official screenshot: The Scenario Dashboard. The Scenario Domains table shows each domain, its loop count and a summary such as "1 User For Each Department". Receivers, generators and variables for the selected scenario domain are underneath.
[Source page](#)

Area	What it does (docs)
Top buttons	Save name and description, <i>Download</i> the scenario, <i>Copy</i> , <i>Delete</i> , make stateful or stateless, <i>Categorize</i> , <i>Lock</i> and <i>Sync</i>
Variable Set drop-down	Chooses a project-version variable set for the downloaded scenario
Scenario Domains	Name, loop count, summary, which domain is primary and which is root. Add a template domain, remove a domain, randomise or change the loop count, choose the primary domain, <i>Optimize</i> , or <i>Disable Non-Primary Receivers</i>
Scenario Domain Receivers	Enable or disable each receiver for this scenario, and add references to its parameters

The loop count of a child domain is counted *per parent row*. So to get 100 orders for 50 customers, you set the Orders loop count to 2. Section 6.25.7 works through a three-level example with larger numbers.

A REMINDER ABOUT WHERE TO CHANGE THE NUMBER OF ROWS

The domain page recommends test data cases for varying row counts. Here the loop count is edited in the Scenario Domains table or on the domain. One quirk from the docs: if the first generator of an attribute is ListGen with `setLoop true`, the list length decides the count and the pencil icon is absent.

6.25.4 Scenario domains, relationships and locks

A scenario holds *copies* of template domains (scenario domains). Template changes flow into scenarios unless the scenario, a scenario domain or an attribute is *locked*. A lock stops changes made higher up from cascading down; while a scenario is locked the Sync button is hidden, and a locked item cannot be deleted. Use *Sync* to pull in template changes deliberately.

You can also change a relationship for one scenario only, leaving the template alone. The vendor's story: for the Student scenario, make Teacher the parent of Student instead of Subject. The steps are to add the Teacher domain to the scenario, remove the Student domain's current parent, and add Teacher as its parent. Changes at scenario level do not flow back up. Relationships cannot be changed at the test data case level, so this is the place to do it.

6.25.5 Scenario variables and scenario state

Variables. Scenario variables are name-and-value pairs usable only inside the scenario that created them. In the Scenario Variables pane click *New Variable*, give a name and value and save, then reference it from a parameter (double-click the parameter field and pick it).

State. Normally every run of a scenario produces the same data, for example a RangeGen starting at 10000 starts there every time. Sometimes you want the next run to continue (ids 10000 to 10099 first, then 10100 and on). Clicking *Make Stateful* saves each generator's position at the end of every *successful* run; *Make Stateless* turns it off. By default scenarios are stateless. The state is kept in a small H2 database in the `.genrocket` folder of the machine, and two runtime options list or clear it: `--state-list` and `--state-clear` (with an id or `all`).

6.25.6 Scenario chains and chain sets

A **scenario chain** links two or more scenarios so that one downloaded file runs them in order. The vendor example has Department, User and Address scenarios that produce three files from one command. A **chain set** links chains. A chain holds up to 25 scenarios and a chain set up to 25 chains. Optionally set a *delay* in seconds after a scenario.

1

In the Scenario Chains pane click **New Scenario Chain**, enter a name and description and click **Save**.

Naming makes the chain findable among many.

Create New Scenario Chain

Name: Sample

Description: Generate Department, User, and Address data

Save Cancel

Official screenshot: Naming the chain.
Source page

1

Click the plus beside each scenario to add it, in the order they should run.

Parents must come first so that their output exists before children use it.

2

Set delays with the pencil icon if you want one, then **Save and Download**.

Expected result: What you should see: a chain file you run just like a scenario.

6.25.7 Worked example: how loop counts multiply along a chain

In its 2020 webinar, GenRocket's CEO explains chains with a three-level example: Organization, Department and User, with one scenario per domain, linked so that every user belongs to a real department and every department to a real organization (referential

integrity). Each scenario has a loop count, and a child's count applies to every parent row, so the numbers multiply down the chain.

THE WEBINAR'S CHAIN, WITH AN ARITHMETIC CHECK IN THE LAST COLUMN

Scenario (one per domain)	Loop count	Counted	Rows made	Check
Organization	3	once	3	3
Department	3	per organization	9	$3 \times 3 = 9$
User	3,000	per department	27,000	$9 \times 3,000 = 27,000$

A second route gives the same answer: each organization has 3 departments of 3,000 users, so 9,000 users per organization, and 3 organizations make 27,000. That matches the total the CEO states.

The ShopSphere lab in section 6.26 is the same idea one level shorter: 50 customers x 2 orders = 100 orders. One difference to remember: the webinar describes each scenario as producing its own domain, while the help centre says a child's scenario also pulls in its parents as helper domains (section 6.25.1), which is why non-primary receivers are disabled (section 6.28). Follow the help centre, and compare each file's row count with your calculation.

6.25.8 Running a scenario with the runtime

The scenario file is run from a command window on a machine that has the runtime and your profile.

1

Download the scenario (or chain).

The Download button on the dashboard, or the cloud icon in the Scenarios pane. Several scenarios can be downloaded together as a ZIP with *Batch Download* (up to 15).

2

Open a command window or terminal in the folder where the file was saved.

The command needs to find the file.



Official screenshot: Start in the folder that holds the downloaded scenario.

[Source page ↗](#)

1

Run `genrocket -r <ScenarioName>.grs .`

The `-r` option means "run". Chains use the same command.

Expected result: What you should see: lines saying the licence was validated, the runtime versions, "Running Scenario ..." for each scenario, and finally "All done!" with the elapsed time.

```
>genrocket -r AllScenarioChain.grs
GenRocket Runtime belongs to https://app.genrocket.com
License has been validated
GenRocket Runtime Version: 3.9.0
GenRocket Version: 3.9.0
GenRocket Engine Version: 3.9.0.0
GenRocket Generator Runtime Version: 3.9.0.0
GenRocket Receiver Runtime Version: 3.9.0.0
GenRocket GRepositoryClient Runtime Version: 3.9.0.0
GenRocket GRepositoryServer Runtime Version: 3.9.0.0
Validating the Scenario Checksum for Integrity.
Starting Engine...
Running Scenario DepartmentScenario...
Building headers from Domain attributes...
Waiting 10 seconds before running next scenario. ←
Running Scenario UserScenario...
Building headers from Domain attributes...
Waiting 10 seconds before running next scenario. ←
Running Scenario AddressScenario...
Building headers from Domain attributes...
All done! Time elapsed: 24s
```

Official screenshot: A chain run in a terminal. The two green arrows mark the delays between scenarios.

[Source page ↗](#)

When data is inserted into a database and loading fails, the runtime throws an exception to the command line (or to the testing framework when it runs in an automated pipeline), so no error means success. The vendor's note on stopping a run: press `Ctrl+C`, and on Windows answer the "Terminate batch job (Y/N)?" prompt with Y to abort.

The 2020 webinar's live demo shows a simple way to check a run. The CTO builds a user domain with an XML receiver, downloads the scenario (about 22 KB, and he points out it would stay that size whether it generates one row or a billion, because it holds instructions, not data), and runs it from a command line with the Java-based runtime. He then opens the XML output and checks that the ids start at 1,000 and rise by one, as the id generator was set to do, that the other values follow their generators' patterns, and that scrolling to the end shows 10,000 rows, the loop count he had set.

Scenarios can also be run through a repository server or an API, which are not covered here. Section 6.25.10 shows the command inside a pipeline.

6.25.9 The Scenario Thread Engine (overview)

The thread engine speeds up chains and chain sets by running several of their scenarios at the same time: `genrocket -r Chain.grs -threaded 10`. The docs caution that threads may run in any order, so use it only where order does not matter, for example many segment files that a merge receiver combines later (the engine waits for every segment before merging). Do *not* use it where SQL inserts depend on parent rows already being present, because the order matters for referential integrity.

6.25.10 Scenarios inside an automated test pipeline

Later in the 2020 webinar, the CEO plays a short vendor video about combining GenRocket with Jenkins (a tool that runs jobs automatically) and Selenium (a tool that drives a web browser to run functional tests). The point is that data is created at the moment a test needs it, not prepared days earlier. In plain steps:

1

A Jenkins job starts.

A trigger such as a build normally starts it; the video starts it by hand.

2

The job opens a shell and runs GenRocket against a test plan.

A test plan here is a set of scenarios to work through, so one command covers the whole run.

Expected result: What you should see: the Jenkins console showing the test plan loading and executing.

3

When a test needs data, GenRocket generates it.

Each test gets fresh data shaped for it, not a shared set that earlier tests may have changed.

4

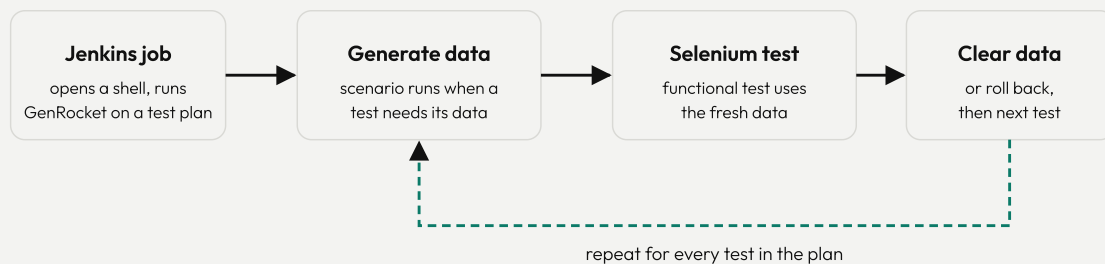
GenRocket then calls a Selenium step that performs the functional test.

The test consumes the data it was just given.

5

The data can be cleared, and the next test gets fresh data.

The CEO says it need not sit around; in the 2022 overview video he adds that the transaction can be rolled back before the next test case runs.



The Jenkins, GenRocket and Selenium flow described in the 2020 webinar's video. The dashed loop is the repeat for each test.

The Jenkins page in the help centre (see the overview chapter) runs one named scenario per build step; the test-plan runner in the video is not documented on those pages, so check the current documentation before copying this layout.

WHERE SHOULD THE RUNTIME RUN WHEN THE TEST SERVERS ARE IN THE CLOUD?

A viewer asked about cloud-hosted applications in the 2020 webinar's Q&A. The CTO's answer: for a quick data set on your own computer, download the scenario and run it with the local runtime (two or three minutes, he said, against an hour of hand-written Python or Groovy). For a full regression suite on a cloud test server, such as one on Amazon or Azure, install Java and the GenRocket jars on that server and generate the data there. Do not generate it elsewhere and carry it up, because that is slow and moves a lot of data into the cloud. The rule of thumb is: run the runtime where the tests run.

6.26 ShopSphere end to end: CUSTOMERS to CSV and ORDERS to SQL inserts

The build below uses the design from the domains, attributes and generators chapter. SQLFileInsertReceiver is used for orders because it needs no database connection.

Lab: one chain, two outputs

Goal: produce `Customers.csv` with 50 customers and `Orders.sql` with 100 insert statements whose `customer_id` values all exist in the CSV.

1

Check your resources.

In the Resources pane the home and output directories must point to real folders on the machine that will run the scenario, and your profile must be in `.genrocket`. If you change one, download the profile again.

Expected result: Expected: three resource rows with paths (JDBC may be N/A).

2

Add a DelimitedFileReceiver to the Customers domain.

Set `fileName` to `Customers.csv`, `delimiter` to a comma, `includeHeader` `True`, `quoteTextData` `None`, `appendToFile` `false`. Leave `path` as the output resource.

Expected result: Expected: the receiver appears in Domain Receivers.

3

In its Attributes Property Keys tab, set `columnName` to the table's column names.

Set the id, names and other columns to `customer_id`, `first_name` and so on, so the CSV header matches the source database. Set `include` `false` for anything the consumer does not want.

4

Add a SQLFileInsertReceiver to the Orders domain.

Set `tableName` to `ORDERS`, `fileName` to `Orders.sql`, `batchCount` `100`, `statementSeparator` a semicolon, `quoteTextData` `Single`, `nullValue` `NULL`, `appendToFile` `false`. Set the column names to `order_id`, `customer_id`, `order_date` and `total_amount`.

Setting `useQuotes` `false` on the numeric columns is a reasonable choice, but the vendor page only says "sometimes a numeric value should be quoted".

Expected result: Expected: the property-key tab shows the real column names.

5

Create the scenarios.

Use *Quick Scenario* on Customers and on Orders. The Orders scenario brings Customers in as its parent. Both will normally have their receivers, so in the Orders scenario open the Scenario Domain Receivers pane and disable the Customers receiver (or use *Disable Non-Primary Receivers*), so the CSV is written once, by the Customers scenario.

Expected result: Expected: the Orders Scenario Dashboard lists Customers and Orders, with Orders as primary and Customers as root.

6

Set the row counts in the Scenario Domains table.

Customers: 50. Orders: 2, meaning two orders for each customer, so 100 in total.

Expected result: Expected: the summary for Orders reads like "2 Orders For Each Customer" (the product's wording).

7

Create a chain called ShopSphereChain with CustomersScenario first and OrdersScenario second, then download it.

Order matters: parents first.

8

Run `genrocket -r ShopSphereChain.grs` from the download folder.

Expected result: Expected: "Running Scenario CustomersScenario", then "Running Scenario OrdersScenario", then "All done". Output folder holds Customers.csv and Orders.sql .

► Show how to check the result

VARIATION: INSERT STRAIGHT INTO A DATABASE

Replace the `SQLFileInsertReceiver` with `GenericSQLInsertV2Receiver`. Create a `config.properties` file with the connection settings (`driver` , `user` , `password` , `url` , `batchCount`) in the JDBC resource folder, set the receiver's table settings, leave `truncate` on "notApplicable", and put the JDBC driver jar in the runtime's `lib` folder. Test with `--test-jdbc` , then run the chain. This needs a database and credentials from the DBA.

6.27 Troubleshooting

The vendor publishes several pages titled "ERROR: ..." plus related receiver and runtime notes. They are condensed here.

Message or symptom	Cause (docs)	What the docs advise
"Could not find the required profile: GRXXXXXXXXProfile.grp" (may add "Unable to validate license. Reason: INVALID ACCESS KEY")	The user profile is missing or outdated in <code>.genrocket</code>	Download your profile again and place it in the <code>.genrocket</code> folder (replace any old one), then re-run
"Unable to find domain <Domain> from <Domain.Attribute>"	An attribute refers to another domain's attribute at test-data-case level while no relationship exists between the domains	Define the relationship at the template domain level; add the parent to the scenario
<code>java.lang.OutOfMemoryError: GC overhead limit exceeded</code>	Not enough heap: unique limits on generators, huge files read into memory, or large database queries	Set <code>JAVA_OPTS</code> (for example <code>-Xms1024m -Xmx2048m</code>), relax unique limits, use a batch-reading generator (CSVToMapGen instead of ListCSVGen), or raise the heap further
"Communication link failure" when connecting to a database	Network, firewall, server down, wrong host, port or credentials in the JDBC file, or an incompatible driver	Check connectivity, that the server runs, firewall and VPN, the <code>config.properties</code> values and the driver (the page lists a dozen fixes, including MySQL settings)
PostgreSQL: "column ... does not exist"	PostgreSQL folds unquoted names to lower case	Quote the identifier or change the receiver's column names to lower case

Message or symptom	Cause (docs)	What the docs advise
"'genrocket' is not recognized" or "command not found"	Runtime folder, home variable or <code>PATH</code> is wrong	Check that the folder is named <code>genrocket</code> , the variables are set, and open a new terminal
<code>UnsupportedClassVersionError</code> , "JAVA_HOME is set to invalid directory"	Java too old, or the variable points to the wrong place	Check <code>java -version</code> , set <code>JAVA_OPTS</code> for Java above 1.8, correct <code>JAVA_HOME</code> or install the JDK
"Could not find or load main class ..."	The dependent jars did not download	Run <code>genrocket -a</code> , or download the jars manually into <code>lib</code>
"Unable to Download Users Profile"	A resource has no value	Give every resource a valid path, then download again

6.28 Common mistakes and gotchas

WATCH FOR THESE

- **Changing a receiver or loop count and re-running the old file.** The scenario file is a snapshot. Download it again after every change.
- **Wrong chain order.** Children before parents means keys do not exist when you load the data.
- **Column names that do not match the database.** In SQL receivers, `columnName` must be the real column.
- **Parent data written twice.** A child scenario brings its parents in; disable the non-primary receivers if you only want the child's file.
- **Truncating a table by accident.** Leave `truncate` on "notApplicable".

6.29 Recap

KEY POINTS

- A receiver turns generator output into a format. Sixteen categories, 120+ receivers; more than one can sit on a domain.
- Receivers use organisation resources for folders. Download your profile again whenever a resource changes.
- Delimited, fixed, JSON, XML, Excel and SQL-file receivers cover most needs; segment-and-merge receivers make nested output; masking receivers replace values in existing files.
- JDBC receivers insert directly and need a JDBC config file; V2 adds resource path settings; bulk-load receivers are for very large volumes.
- A scenario is built for a primary domain and pulls in its parents. A child's loop count is per parent row.
- Download, then run with `genrocket -r`. Chains run several scenarios in order. Locks, state and variables control behaviour between runs.
- Loop counts multiply down a chain (3 organizations x 3 departments x 3,000 users = 27,000 in the 2020 webinar), so work out the expected row count before you run, then compare it with the output.
- In a pipeline, run the runtime where the tests run, generate data when a test needs it, and clear it afterwards.
- When something fails, the commonest causes are a missing profile, wrong paths or variables, relationship or ordering mistakes, and Java memory settings.