



Success Story

On-Demand Data Generation at a Large Financial Institution

From 5-Day Provisioning
to Self-Service Data

Learn more at curiositysoftware.ie

© Curiosity Software Ireland

On-Demand Enterprise Data



A centralized test data team required between

1 hour and 5 days to fulfil data requests, hunting for data in complex DB2 mainframe and a Google BigQuery warehouse.



Curiosity introduced automated, self-

service data provisioning, with engineers and CI/CD tools running “find and makes” using self-service forms and APIs calls.



Curiosity's Enterprise Test Data

automatically generates repeating JSON into BigQuery, and has found and made consistent data within the DB2 mainframe.

4 Benefits at a Glance



Data provisioning time reduced from between 1 and 5 days, to instant and self-



Unlimited quantities and variety of data drive rigorous testing and development.



Complete test data coverage finds bugs early.



Parallelization of tests and engineers shortens release cycles.



Contents

On-Demand Enterprise Data	- 2 -
4 Benefits at a Glance	- 2 -
The Challenge.....	- 4 -
Manual Testing vs Delivery Speed and Quality	- 4 -
Vastly Complex Data and a Data Migration to Support.....	- 4 -
Lengthy Waits for Data.....	- 5 -
The Solution	- 7 -
Self-Service Data “Find and Makes”	- 7 -
Building a Test Data Mart	- 7 -
Self-Service Forms and API Calls	- 7 -
A Reusable Catalogue of Data Requests	- 9 -
Generating Missing Data On-The-Fly	- 10 -
Find Similar Data	- 10 -
Data Cloning.....	- 10 -
Synthetic Data Generation	- 10 -
Complex JSON Message Generation	- 11 -
Data Reservation and Parallelization.....	- 11 -
Solution Summary: Self-Service Data “Find and Makes”	- 12 -
From 5-Day Provisioning to Self-Service	- 13 -
Transform Your Data Provisioning!.....	- 14 -

curiosity

The Challenge

Manual Testing vs Delivery Speed and Quality

One of the world's largest credit bureau's faced lengthy test data provisioning bottlenecks, undermining both the speed and quality of their testing and development.

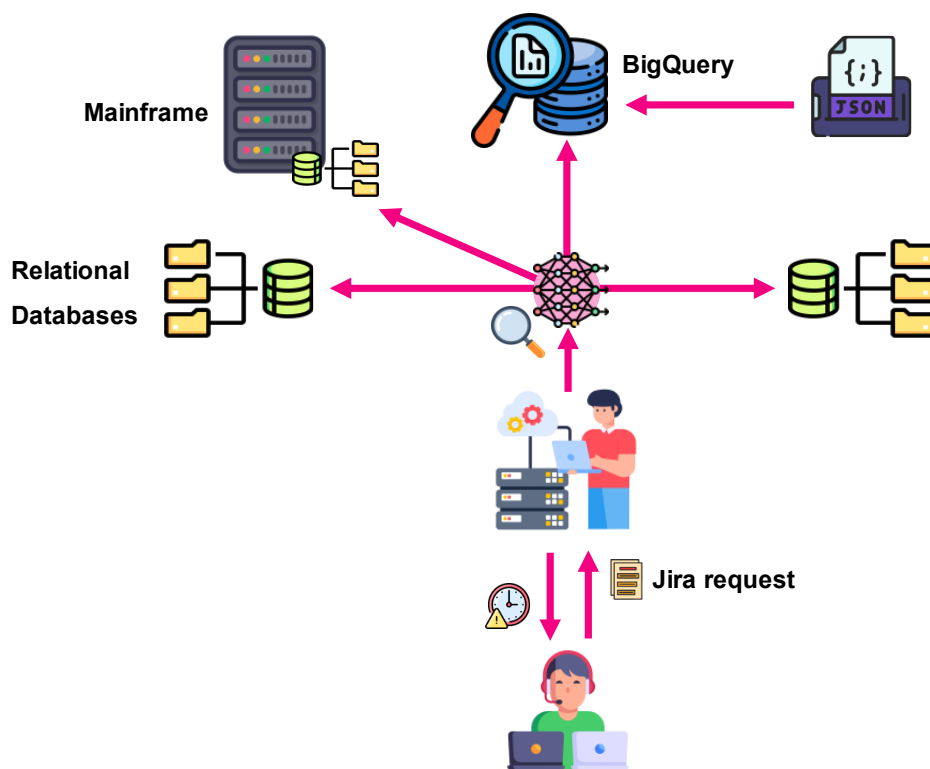
Testers and developers had to submit Jira tickets to an overworked team of test data subject matter experts (SMEs). The SMEs then hunted for the combinations needed in testing and development, searching for referentially intact data across a complex technology stack.

Vastly Complex Data and a Data Migration to Support

The sheer size and complexity of the data made manual data provisioning a significant bottleneck.

When Curiosity started working with the credit bureau, data had to be found in a DB2 Mainframe, upon which a credit reporting system was built. Credit files were stored in a fixed file format, with intricate interrelations. The data provisioning team had to hunt for data to create files, validating it, and sending files back to test and development teams.

The credit bureau then migrated to Google BigQuery. This large database has complex arrays within its relational data. It moreover ingests complex JSON files, some as large as 4gb, and including repeated elements. Data provisioning team must create referentially-intact data for this complex system, while fulfilling the diverse and fast-flowing requests sent by test and development teams.



Lengthy Waits for Data

Before Curiosity began working with the credit bureau, there were never enough SMEs to find relevant data quickly enough. For simple data requests, an SME required 1-4 hours to find data, validate it, and send it in a file. Complex requests took 4-5 days, hindering development agility.

The volume of available data was further limited by the number of Jira requests an SME could fulfil in a work week. This hindered data-intensive testing and development, further undermining software quality.

To deliver quality software faster, the bureau needed to modernize their “request and receive” approach to test data provisioning.

Key Problems to Solve:



Time spent searching for, maintaining and creating data.



Difficulty in finding relevant data in large data sets.



Cross-team dependency on SMEs for finding and creating data.



Up to 5 days to fulfil a test data request.



Insufficient volumes of data for testing and development.

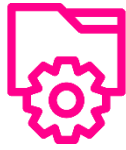


Test automation limited by manual bottlenecks.



Delivering data throughout and after a migration.

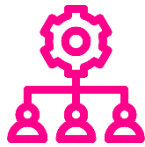
Key Technology Challenges:



A complex, legacy mainframe application and fixed files.



A complex BigQuery warehouse, ingesting large, repeating JSON files.



Growing numbers of dependent application teams.



Lack of subject matter experts.

The new standard in test data management

Simplify complex application landscapes and provide confidence and clarity at every step of your test data management journey with our intuitive, AI-driven Enterprise Test Data® platform.

[Book a Meeting](#)

The Solution

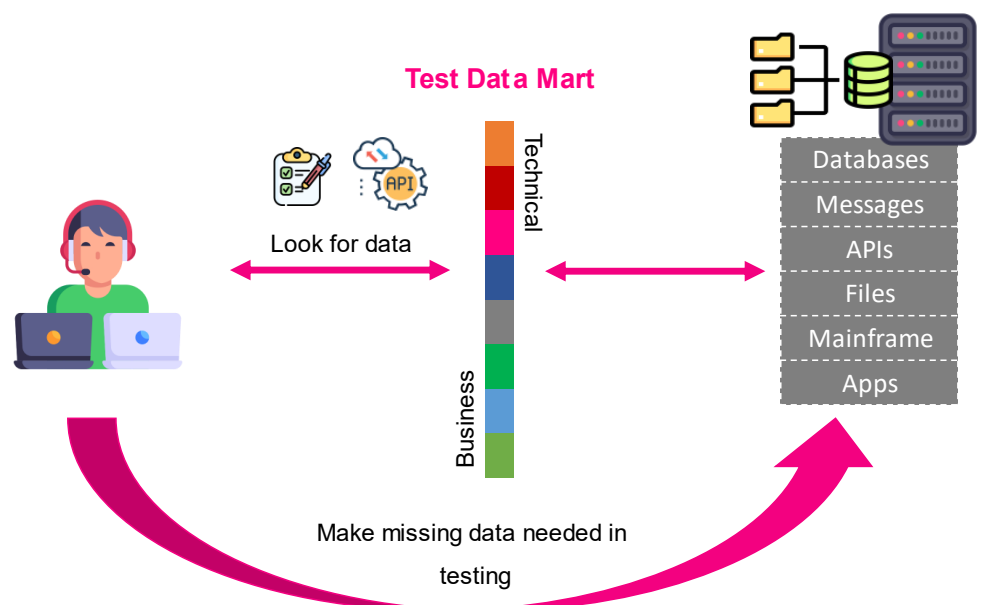
Self-Service Data “Find and Makes”

Through an initial 5-day proof of concept (POC) and subsequent support by test data engineers, Curiosity have used Enterprise Test Data to introduce a rapid approach to finding and making data.

Testers, developers and CI/CD tools now submit data requests using intuitive web forms and API requests. Enterprise Test Data then hunts automatically for data needed to fulfil the test case or user story definition, automatically making data where relevant combinations cannot be found.

Building a Test Data Mart

These self-service “find and makes” are built on top of a test data mart. The mart provides a central, aggregated view of the source data’s technical attributes. These technical attributes are mapped to the intuitive “business language” used by testers.



Self-Service Forms and API Calls

This separation of the business and technical enables self-service data lookups. Testers and developers can enter parameters into forms constructed in business language. The forms, and API requests, cast queries to look for data. This initially found data in the DB2 mainframe database. Post-migration, they generated and reserved data in the BigQuery warehouse.

Using the forms and API calls, testers can input hundreds of possible parameters. The parameters search for data on demand, while changing data to meet specific requests.

Create Commerce Orders - Job Parameters

Create Commerce Orders

☒ Parameters

☐ Schedule

Country

USA

Low Price

10

High Price

100

How Many Products

1

How Many Customers

1

How Many Orders

1

How Many Items

10

</>

📄

Create List

▶ Execute

⌛ Cancel

Self-service forms expose a choice of parameters to end users. They are constructed in intuitive business language, using drop-downs, checkboxes, text and numerical fields to generate data on demand. CI/CD tools can do the same, using APIs.

The forms present a configurable subset of parameters in checkboxes, drop-downs, number and text fields. This allows testers and developers to locate the data they need using the smallest possible set of inputs.

The forms, test data dictionary and data mart additionally group parameters. This allows testers or developers to look for a set of data based on an initial set of parameters, adding more parameters if they need to perform a more granular search. For example, a section of forms is dedicated to credit ratings. This contains even more targeted groupings of parameters, such as a group focused just on user location. Testers can expand the parameters, circling out to find the data they need.

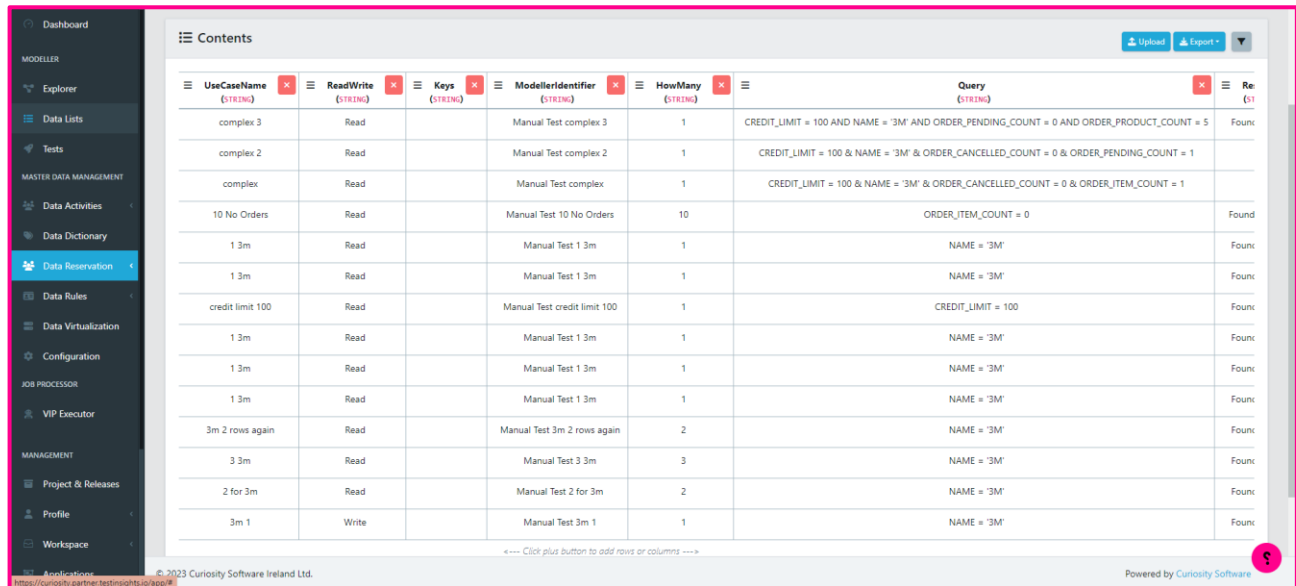
The new standard in test data management

Simplify complex application landscapes and provide confidence and clarity at every step of your test data management journey with our intuitive, AI-driven Enterprise Test Data® platform.

[Book a Meeting](#)

A Reusable Catalogue of Data Requests

Each query submitted to Enterprise Test Data becomes reusable and repeatable on demand. Meanwhile, new queries add attributes to the data mart:

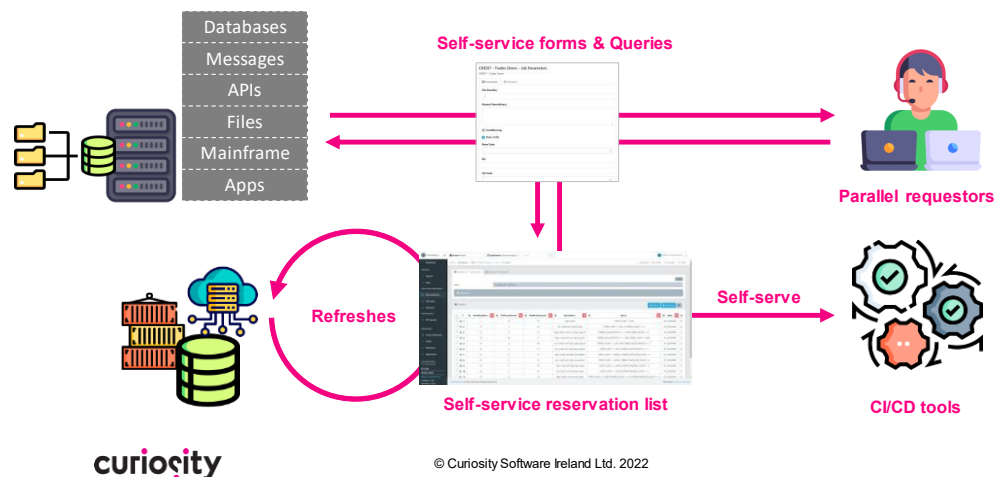


The screenshot shows a web application interface with a sidebar on the left containing navigation links like Dashboard, Explorer, Data Lists, Tests, Data Activities, Data Dictionary, Data Reservation (highlighted), Data Rules, Data Virtualization, Configuration, Job Processor, VIP Executor, and Management. The main area displays a table titled 'Contents' with columns: UseCaseName (STRING), ReadWrite (STRING), Keys (STRING), ModelIdentifier (STRING), HowMany (STRING), Query (STRING), and Re (STRING). The table lists various test data queries such as 'complex 3', 'complex 2', 'complex', '10 No Orders', '1 3m', 'credit limit 100', '3m 2 rows again', '3 3m', '2 for 3m', and '3m 1', each with associated Read/Write permissions, model identifiers, counts, and specific query strings.

UseCaseName (STRING)	ReadWrite (STRING)	Keys (STRING)	ModelIdentifier (STRING)	HowMany (STRING)	Query (STRING)	Re (STRING)
complex 3	Read		Manual Test complex 3	1	CREDIT_LIMIT = 100 AND NAME = '3M' AND ORDER_PENDING_COUNT = 0 AND ORDER_PRODUCT_COUNT = 5	Found
complex 2	Read		Manual Test complex 2	1	CREDIT_LIMIT = 100 & NAME = '3M' & ORDER_CANCELLED_COUNT = 0 & ORDER_PENDING_COUNT = 1	
complex	Read		Manual Test complex	1	CREDIT_LIMIT = 100 & NAME = '3M' & ORDER_CANCELLED_COUNT = 0 & ORDER_ITEM_COUNT = 1	
10 No Orders	Read		Manual Test 10 No Orders	10	ORDER_ITEM_COUNT = 0	Found
1 3m	Read		Manual Test 1 3m	1	NAME = '3M'	Found
1 3m	Read		Manual Test 1 3m	1	NAME = '3M'	Found
credit limit 100	Read		Manual Test credit limit 100	1	CREDIT_LIMIT = 100	Found
1 3m	Read		Manual Test 1 3m	1	NAME = '3M'	Found
1 3m	Read		Manual Test 1 3m	1	NAME = '3M'	Found
1 3m	Read		Manual Test 1 3m	1	NAME = '3M'	Found
3m 2 rows again	Read		Manual Test 3m 2 rows again	2	NAME = '3M'	Found
3 3m	Read		Manual Test 3 3m	3	NAME = '3M'	Found
2 for 3m	Read		Manual Test 2 for 3m	2	NAME = '3M'	Found
3m 1	Write		Manual Test 3m 1	1	NAME = '3M'	Found

A “reservation list” of queries constructed from parameters previously submitted to Enterprise Test Data’s “find and make” activity.

Stored user queries can be rerun automatically, providing on demand data that fulfils every past request. Executing the reusable queries during environment refreshes then produces fresh data for every past test and development scenario, ready at the start of each iteration. Automation frameworks and CI/CD tools can likewise trigger reusable queries from the central “reservation list”.



The on demand refreshes can populate traditional instances, containers, virtual databases and more. Enterprise Test Data can furthermore “explode” data coverage based on the catalogue of past queries, automatically creating varied test data to find bugs earlier, and at less cost to fix.

Generating Missing Data On-The-Fly

When no suitable data is found, Enterprise Test Data automatically generates the missing combinations on demand. This maintains the optimal data coverage needed for rigorous testing and development. It additionally feeds “gold copy” data for fulfilling future requests.

A range of techniques generate the accurate data on demand:



Find Similar Data

If exact matches cannot be found for a request, SQL parsing generates functions that find similar data.



Data Cloning

Where more data is needed to fulfil a request, data cloning creates equivalent data combinations with unique identifiers. This enables parallelized testing and development, “crawling” across tables to create consistent data that fulfils all the relationships needed.



Synthetic Data Generation

Where no similar or matching data exists, Enterprise Test Data’s comprehensive data generation activity creates the requisite combinations from scratch. This passes parameters entered into the self-service forms or API requests into its data generation engine, producing data that accurately reflects definitions stored in a central data dictionary. The rich data is generated directly to databases, or via message queues, files, and APIs.

When working with the credit bureau’s DB2 system, the automated “makes” would break down the fixed format credit files into data points. Where no matching data could be found, similar data would then be located, or new data generated on demand.

Following the credit bureau’s switch to BigQuery, Enterprise Test Data’s flexible connectors and extensive message generation capabilities now produce synthetic JSON messages on demand.

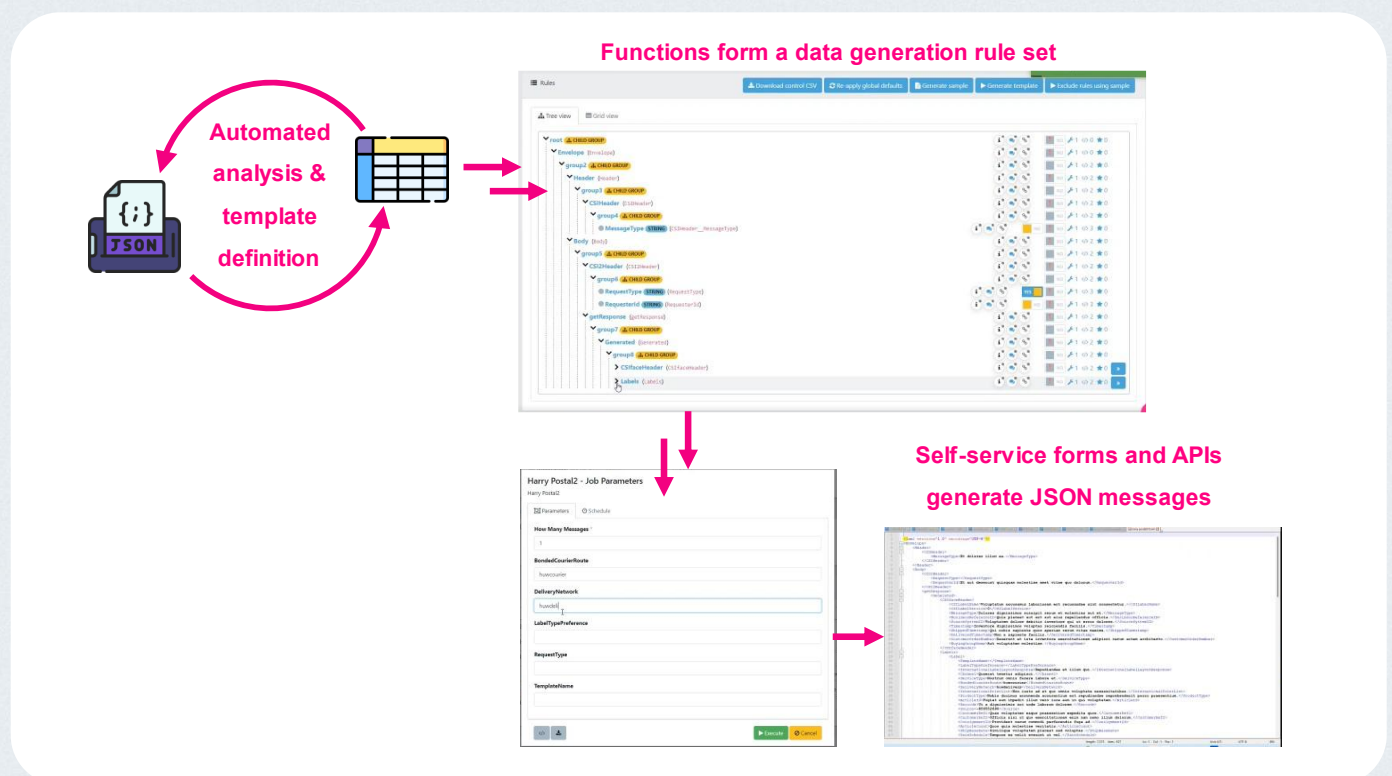


Complex JSON Message Generation

As the credit bureau migrated from DB2 to BigQuery, test data provisioning needed to provide complex repeating JSON files, some as large as 4gb in size. Curiosity leveraged Enterprise Test Data's extensive synthetic message generation capabilities and connectors to enable a seamless transition to the new technology, with no loss in the test data provisioning service.

The automated “find and makes” now generate JSON messages, which are ingested by the BigQuery warehouse. This leverages message templates, which are created by automated message analysis. The templates define the structure of the JSON files.

Enterprise Test Data's comprehensive data generation functions then generate data consistently for each data point in the JSON file template, based on the parameters inputted by the data request:



The template-driven JSON message generation creates the complex JSON files with nested elements, while its high-performance rapidly produces JSON files of the volumes needed in testing and development. This ensures that data is generated consistently for each on demand data request.

Data Reservation and Parallelization

Data cloning additionally removes bottlenecks created by constrained data, providing on demand data for parallel testers and tests. This test autonomy and parallelization is further supported by test data “allocation” and reservation. Data allocation automatically locks data when it is requested in a shared

environment. It assigns “Read Only” or “Read/Write” privileges to the data, ensuring that no other engineer, test or framework can transact against it. Today, testers and developers can use forms and API calls to perform automated “Find and Reserves” within the BigQuery warehouse, avoiding test failures created when useful data is edited or used up by another test or tester.

Solution Summary: Self-Service Data “Find and Makes”

This approach eradicated test data provisioning bottlenecks, implementing an automated, self-service approach to data “find and makes”:



Enterprise Test Data provides a common way of identifying and making test data.



It attaches non-ambiguous data definitions to test cases.



It automates test data searches through web forms and APIs.



It refines the searches in cases of no exact match, generating missing data on-the-fly.



It calls legacy application APIs to generate complex reports.



It auto-generates repeating JSON files to feed in BigQuery.



It modifies the report to match search criteria.

From 5-Day Provisioning to Self-Service

Self-service Enterprise Test Data removes the data provisioning bottlenecks that were undermining the credit bureau's delivery quality and agility. Automated "find and makes" eradicate the time lost waiting for data, saving between 1 hour and 5 days per test data request.

The standardized, automated approach breaks the dependency on a small team of SMEs to fulfil each data request. Instead, engineers and CI/CD tools can self-serve *all* the data they need for complete test and user story coverage.

This provides data of the variety and volumes needed in rigorous testing and rapid development. From "fail fast" experiments and unit testing, to high-volume stress testing, parallel teams and frameworks can deliver quality faster, enjoying truly on demand data.

Curiosity have supported the credit bureau through their migration from one data format to another, leveraging our flexible technologies and connectors to provide on demand data from evolving sources. This has enabled continued time savings:

Problem solved	Problem before Curiosity	Solution with Curiosity
The time it takes an SME to complete a data request from start to finish (find, validate and send the file)	1-4 hours	Instant - an SME does not need to be involved
The time it takes SME to complete a complex data request	4-5 days	Instant
Quantity of data	Limited to how many Jira tickets SME can complete in a work week	Infinite – limited only by the available infrastructure
Speed and accuracy of data	Limited to SME's available resources at a given time	Once test data is shaped , you get immediate results whenever you run the query or test case
Speed of implementation	Homegrown data generation – ~6 months	5 days

Transform Your Data Provisioning! —



Visit www.curiositysoftware.ie to learn more or [book a demo](#) with a Curiosity expert today!

Additionally, you can also email us at info@curiositysoftware.com



Call USA:

+1 914 218 0180



Curiosity Software Ireland

Unit 6 The Mill, The Maltings, Bray, Co. Wicklow, A98 XV40, Ireland

Curiosity Software USA

4136 Del Ray Ave. Suite 658, Marina Del Rey, CA 90292, USA

