

Putting it together: real engagements, a worked project and the glossary

How a real TDM engagement is planned and signed off, a worked end-to-end project, and a glossary of every term used in the guide.

Planning a real TDM engagement: use cases, POC, assessment and sign-off

Knowing how to run a masking job is one skill. Getting a company to trust you with its data, agreeing what "done" means, and leaving behind something the audit team accepts is another, and it is the one that decides whether a TDM project survives. This section covers the documents of a TDM engagement: a use-case questionnaire, a data inventory, a sensitivity assessment, a tool-selection matrix, a proof-of-concept (POC) plan, an assessment report, a risk register, a deployment plan, a sign-off checklist, a RACI table and KPIs.

Everything is worked through with ShopSphere, the fictional online retailer (example data).

10.1 Words to know first

Engagement

One piece of TDM work for one client or business unit, from first conversation to hand-over. ShopSphere's "protect QA data" project is an engagement.

Use case

A specific situation where test data causes a problem, written in terms of who needs what data and why. "QA needs 90 days of orders without real customer identities" is a use case.

TDG

Test data generation: creating synthetic data instead of copying real data.

LOB

Line of business: one business area with its own applications and data, such as ShopSphere's order management.

Technology landscape

The list of systems, databases, files and interfaces that hold or move the data in scope. The "map" you draw before you choose tools.

POC (proof of concept)

A small, time-boxed trial of the chosen approach on a slice of real systems, to show it works before a full rollout.

Assessment

A structured review of an application and its data environment, ending in a written report with findings and recommendations.

Sign-off

Formal written approval by named people that agreed criteria have been met.

Success criteria

Measurable statements that decide pass or fail, agreed before the POC starts.

Risk register

A living table of things that could go wrong, with likelihood, impact, owner and mitigation.

RACI

A table that marks, for each task, who is Responsible, Accountable, Consulted or Informed.

KPI

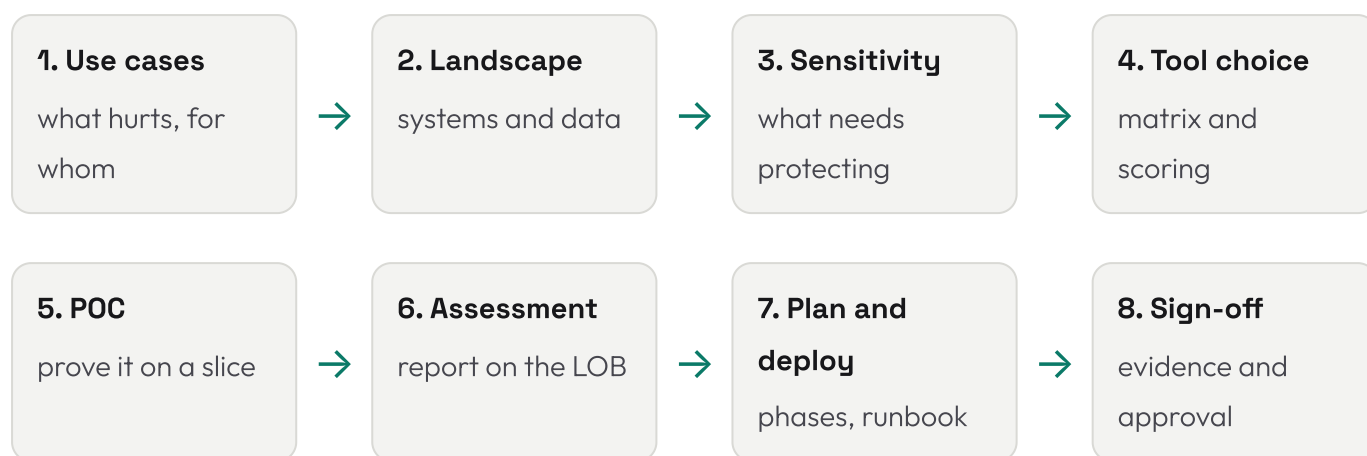
Key performance indicator: a number you track to show whether TDM is helping.

Evidence pack

The queries, screenshots and reports that prove each success criterion is met.

10.2 The life cycle in one picture

A real engagement moves through the same stages every time, whether the goal is masking or generation. The stages differ in depth, not in order.



WHY PLAN BEFORE TOUCHING A TOOL

The costly mistakes in TDM are decided early: masking a column that is never used, missing a file that carries the same SSN, picking a tool that cannot read the main database, or promising "all sensitive data masked" without agreeing how to measure it. Each of those is cheap to fix on paper and very expensive after the first refresh has reached 200 testers.

10.2.1 Two plans: generation and masking

Masking plan

Starts from *existing* data. The central question: which real data can we safely make available, and how do we prove it is protected?

Heavy on: landscape, sensitivity rating, algorithm choice, referential integrity, evidence.

Generation (TDG) plan

Starts from a *model* of the data. The central question: what shape of data do the tests need, and how much?

Heavy on: data model, rules and patterns, volumes, edge cases, repeatability.

The tools themselves draw the same line. In Informatica TDM a data generation operation needs a separate plan: the guide says generation cannot be combined with masking or subsetting in the same plan (User Guide 10.5.10, p.33). GenRocket describes its own method as Model, Design, Deploy, Manage (The GenRocket Methodology; the page itself carries only a video). Many engagements need both: mask the customer data, generate the data that never existed.

10.3 Finding the use cases

A tool can only solve a problem someone has named. Collect the problems with the questionnaire below, in short interviews with QA, development, DBA, security and compliance people, and record the answers in the same table every time.

#	Question	Why you ask	Ask
U 1	Which test environments exist (dev, QA, UAT, performance, training), and what data does each hold today?	Shows where real data may already be sitting	QA lead, DBA
U 2	How does data reach each environment (full copy, backup restore, extract, manual script)? How long does it take and how often?	Gives the baseline for provisioning and refresh time	DBA, TDM engineer
U 3	Who can see each environment's data (employees, contractors, vendors, offshore teams)?	Decides the severity of exposure	Security, compliance
U 4	Which fields are personal, financial, health or secret? Which regulations apply (GDPR, HIPAA, PCI, others)?	Sets what must be protected (see the sensitive data chapter)	Compliance officer
U 5	What test cases fail or are skipped today because the right data is missing?	Finds generation use cases	QA engineers
U 6	Which tests need volume (performance, load) or rare conditions (edge cases, negative cases)?	Shows where synthetic data beats a copy	Performance tester, QA
U 7	Which data must stay realistic and linked (same customer across systems, valid-looking IDs)?	Defines the integrity requirements for masking	Developers, data steward
U 8	How many defects in the last year were caused by wrong, stale or missing test data?	Gives the business case and a KPI baseline	QA lead
U 9	What else will need the data: demos, training, third-party sandboxes, analytics?	Widens the scope early	Product owner
U 10	What already exists: scripts, masking tools, virtualization, licences?	Avoids duplicating or fighting an existing tool	IT manager

#	Question	Why you ask	Ask
U11	What would "success" look like in six months, in numbers?	Seeds the success criteria	Sponsor

10.3.1 ShopSphere's answers, turned into use cases

After the interviews the TDM engineer writes each finding as a use case with an identifier, an owner and a type.

SHOPSPHERE USE-CASE REGISTER (EXAMPLE DATA)				
ID	Use case	Type	Driver	Priority
UC 1	QA and UAT need realistic customers and orders, with no real names, SSNs or phone numbers	Masking	Compliance (U4), exposure to contractors (U3)	High
UC 2	QA refreshes take about 3 days and block testers; a smaller, linked copy would do	Subset (and masking)	U2, U8	High
UC 3	The new gift-card feature has no production history to copy	Generation	U5	Medium
UC 4	Performance test needs about 5 million orders, 10 times what QA holds	Generation	U6	Medium
UC 5	Checkout validation must be tested with null names, 6-digit and 5-digit zips, and odd tax IDs	Generation (edge cases)	U5, U6	Medium
UC 6	A nightly customer export file goes to a marketing partner's test system	Masking (files)	U3, U9	High

The priorities are proposals for the sponsor to confirm, because priority is a business decision, not a technical one. The register is the input for every later step: if a use case is not on it, it is not in scope.

10.4 Mapping the technology landscape

The technology landscape is a table: every place where in-scope data lives or moves, who owns it, and which environments it feeds. Draw it before choosing tools, because a tool that cannot reach one of the rows leaves a hole in the protection.

SHOPSPHERE LANDSCAPE INVENTORY (EXAMPLE DATA)						
ID	System or store	Type	Holds	Feeds	Refresh	Owner
L1	SHOPSPH ERE_PRO D	Relational database (engine to confirm)	CUSTOMERS, ORDERS, ORDER_ITEMS, PRODUCTS	QA, UAT, PERF copies	Monthly full copy, about 3 days	DBA team
L 2	QA and UAT databases	Relational database	Copies of L1	Testers, contractors	On refresh	QA lead
L 3	customers .dat	Delimited file	Customer id, names, SSN, phone	Marketing partner's test system	Nightly	Marketi ng IT
L 4	billing_f ixed.txt	Fixed-width file	Customer id, name, tax id	Finance test system	Monthly	Finance IT

Then check each row against what the tools can read. The sources say: Informatica TDM builds subsets from relational, flat file or XSD sources (User Guide 10.5.10, p.112). Delphix lists Fixed Width, Delimited, XML, JSON and Parquet files as supported (Files connector) and databases through its connectors. GenRocket's G-Subset subsets and masks MySQL, MSSQL, PostgreSQL, Oracle, DB2, Sybase and Databricks, but only between databases of the same type (G-Subset overview). A row that no candidate tool reads goes straight into the risk register.

The same SSN appears in L1, L3 and L4. That is the central integrity requirement: whatever masks it must give the same result everywhere, or the partner's test data no longer joins to ShopSphere's. The masking concepts are in the masking concepts chapter.

10.5 Sensitivity assessment

For each column in scope, decide whether it is sensitive, how badly, and what to do. Do not trust column names alone: both Informatica and Delphix profile the data itself as well as the names. Informatica discovers data domains by patterns in data or metadata (User Guide 10.5.10, p.31). Delphix's ASDD profiler adds data-level classifiers, which found fields called DATA_00 and DATA_01 as phone and email in an official screenshot (see the Delphix labs chapter). A simple, defensible rating is impact if leaked (1 to 3) times exposure in test (1 to 3).

SENSITIVITY ASSESSMENT FOR CUSTOMERS (EXAMPLE RATINGS)						
Column	Category	Impact	Exposure	Score	Decision	Integrity need
customer_id	Key	1	3	3	Keep (do not mask)	Primary key; foreign key in ORDERS
first_name, last_name	Personal	2	3	6	Mask	Realistic; same input, same output
email	Personal	2	3	6	Mask	Valid shape; unique if used as a login
phone	Personal	2	3	6	Mask	Same digit count
ssn	National ID	3	3	9	Mask	Format ddd-dd-dddd; same value everywhere
tax_id	Tax identifier	3	3	9	Mask	Format kept; consistent across L1 and L4
date_of_birth	Personal	2	3	6	Mask (shift)	Still a valid date and plausible age
zip_code	Location	1	3	3	Mask or coarsen	6 digits; region may need to stay for tax tests
total_amount	Business	1	3	3	Keep	Sums must still reconcile

The decision rule used here: score 6 or more gets masked; below that is a business discussion. The compliance officer, not the TDM engineer, owns the final "keep" decisions. Record each decision with a reason, because it will be questioned in the sign-off.

10.6 Choosing between Informatica TDM, GenRocket and Delphix

The three tools overlap but have different centres of gravity. A gap in a tool's column means *the documentation consulted does not cover it*, not that the tool cannot do it.

Capability	Informatica TDM	GenRocket	Delphix Continuous Compliance
Core idea	One console, Test Data Manager, for discovery, subset, masking and generation, plus a test data warehouse (User Guide, p.21-22)	Synthetic data from domains, generators and scenarios; G-Subset adds subsetting and masking of existing databases	Discover and mask sensitive data, integrated with virtual copies (VDBs) of databases (Delphix introduction page)
Discover	Primary key, entity and data domain profiles (p.31)	G-Subset can match column names to find PII columns (G-Subset profiling page)	ASDD profiler: path, type, regex and list classifiers in profile sets
Subset	Entities: a driving table plus related tables by constraints, with filter criteria (p.114)	G-Subset: where clause, % or number of rows; composite keys not supported	Per-table filter or custom SQL in a rule set; the docs do not describe following related tables
Mask	Techniques such as substitution, shuffle, key, SSN, credit card; repeatable output with a seed (p.152 to 154)	Synthetic data replacement: sensitive values replaced by generated ones, not reversible; by default the real values are not read	Algorithm frameworks (secure lookup, mapping, character and segment mapping); tokenization is reversible
Generate	Separate data generation plans (p.33)	The core strength	Not documented
Files	Flat file sources (p.112)	Many output formats through receivers (the five-components page)	Fixed-width, delimited, XML, JSON, Parquet, mainframe

Capability	Informatica TDM	GenRocket	Delphix Continuous Compliance
Delivery	Load to target, or store versioned data sets in the test data warehouse (p.22)	Generate straight into the target	Masked VDB provisioning and refresh, for Oracle, SQL Server and Sybase only
Automation	ilmcmd command line (p.23)	Runtime API and CI/CD pipeline articles exist	Masking REST API pages exist
Known watch-outs	SSN masking cannot return all-unique SSNs (p.175)	G-Subset: same database type on both sides	Non-conformant values are left unmasked by default; masking and virtualization must run on separate engines
Best at (summary)	Breadth and governance in one product	Volume, edge cases and no need for production data	Consistent masking plus fast delivery of masked copies

Delphix pages: Introduction to Delphix masking, Provision masked VDBs, Managing rule sets. GenRocket: Five key components and G-Subset data column profiling.

10.6.1 Turning the matrix into a decision

A matrix informs; a weighted score decides, and the scores must come from the POC, not from opinion. List criteria, give each a weight, score each tool 1 to 5 from POC evidence, and multiply. The weights below are a template.

Criterion	Weight	Informatica	GenRocket	Delphix
Reaches every row of the landscape	5	score after POC	score after POC	score after POC
Masking quality and integrity (format, consistency)	5			
Generation of volume and edge cases	3			
Delivery speed to testers	4			
Team skills, licences and cost	3			
Automation, fit with your CI tools and audit evidence	3			
Support from the vendor and any partner	2			

For ShopSphere, UC1, UC2 and UC6 are masking and subset use cases, so the masking tool is chosen by the score, with existing licences and the VDB question (U10) usually deciding close results. UC3 to UC5 are generation use cases, which the matrix points to GenRocket. Mixed answers are normal: engagements often use one tool to protect existing data and another to create new data.

10.7 The proof of concept

A POC answers one question: will this approach work here, on our data, with our people? It is small on purpose. It uses a copy, never production, and it ends with a pass or fail against criteria written *before* it started. A POC without agreed criteria turns into a demo that everyone declares a success.

10.7.1 POC plan for ShopSphere

Section	Content for ShopSphere (example)
Objective	Show that the chosen tool can mask the eight sensitive CUSTOMERS columns and one export file on a lab copy, keep keys and formats intact, and run without manual fixes
In scope	CUSTOMERS and ORDERS in a lab copy (a VDB or restored backup) and <code>customers.dat</code> ; use cases UC1 and UC6
Out of scope	ORDER_ITEMS and PRODUCTS, UAT, volume tests, generation (UC3 to UC5 have their own POC)
Environment	Isolated lab, masking user with only the rights the tool documents, no route to production
Method	The sequence practised in the Delphix labs (connect, profile, assign, run, verify), or the Informatica equivalent in the Informatica masking chapter
Time box	Ten working days: 2 setup, 3 profile and rule design, 2 masking runs and fixes, 2 verification, 1 report
Team	TDM engineer (runs it), DBA (copy and rights), data steward (rules), compliance officer (accepts evidence)
Exit	Pass if every "must" criterion is met; otherwise a written list of gaps and a decision: fix, change tool, or stop

10.7.2 Success criteria

Each criterion states how it is measured and whether it gates the result. Measured means a query, a count or a log line, not an opinion.

ID	Criterion	Measured by	Threshold	Gate
S1	Every sensitive column found and masked	Inventory export plus before/after query	8 of 8 columns changed	Must
S2	Keys survive	Orphan query on ORDERS	0 orphans; row counts unchanged	Must
S3	Formats preserved	Pattern queries (SSN, tax id, phone, zip)	100% match the original pattern	Must
S4	Same value, same masked value everywhere	Compare the SSN in the database and in the file	0 mismatches	Must
S5	No unexplained warnings	Job summary: tables or columns with non-conforming data	0, or each explained in writing	Must
S6	No real value left behind	Search the masked output for five known real test values	0 found	Must
S7	Evidence accepted	Compliance officer reviews the evidence pack	Signed	Must
S8	Time to a usable masked copy	Stopwatch from start of run to verified copy	Report against the 3-day baseline	Reported

S5 exists because of how the tools behave. Delphix leaves non-conformant values unmasked by default and records a warning, so a job can finish while data stays exposed (see Delphix masking terminology). Informatica's SSN masking is documented to be unable to return all-unique values (User Guide 10.5.10, p.175), which is exactly the sort of limit a POC should expose before rollout.

10.7.3 How a services partner structures the whole path

The POC is the middle of a longer path. In the 2020 GenRocket and Kualitatem webinar, a representative of Kualitatem, a software-testing services company and GenRocket partner, described how his company

structures a TDM engagement in three phases. This is one company's method, told in a sales webinar, not a standard.

Partner's phase	What the partner described	Matching section
1. Assessment	Build the business case. Scan the client's environments, tools and skills. Recommend a TDM tool and give an implementation plan for it.	The use-case questionnaire (section 10.3), the landscape inventory (section 10.4), the tool matrix (section 10.6) and the assessment report (section 10.8)
2. Pilot, then roadmap and rollout	Pick a medium-sized project, neither the biggest nor the smallest, that covers the key scenarios, and produce results the client can see as measurable improvement. Then lay out a roadmap and roll out project by project, based on the data model.	The POC plan and its success criteria (section 10.7), then the deployment plan (section 10.10)
3. Support	After go-live, a set period of production support in which the client can bring problems and questions back to the team.	The Operate phase of the deployment plan, and the conditions and owners recorded at sign-off (section 10.11)

Two points are worth borrowing. First, choose the pilot for coverage, not for ease: a project that touches the key scenarios shows the client the real benefit, while a trivial one proves little. The ShopSphere POC above is deliberately smaller (a lab copy, ten days); the partner's pilot is a real medium-sized project. Both produce measurable results before a wider rollout. Second, plan the support period before go-live, so the client knows who to call after the project team leaves.

The partner also listed what it weighs when recommending a tool: complexity and cost, speed of providing data, adaptability to the client's existing tools (it named Jenkins, Selenium, CI pipelines and programming languages), and the support available. These map onto the scoring table in section 10.6. The partner added that GenRocket meets all four factors. That is a claim from a vendor's partner, so take scores from your own POC instead.

The webinar ended with the vendor's route to a POC: a personal demo, then a no-cost proof of concept delivered by the vendor, the partner or both. Vendors commonly offer a similar route. Accept it if it helps, but still agree your own success criteria and exit decision first.

10.8 The assessment of a line of business

The POC proves the tool; the assessment describes the line of business it will be applied to, and what must change for a full rollout. For ShopSphere the LOB is online order management. Record evidence for each dimension below.

Dimension	What to find out
Application and interfaces	Which applications use the data, which files and APIs leave the LOB, which hold copies
Data	Tables, volumes, growth, real keys and undeclared relationships (a join the application uses but the database does not declare breaks masking)
Environments and refresh	How many, how often, who triggers it, current duration
Existing controls	Current masking scripts, access rules, past incidents
Compliance	Applicable rules and what evidence an auditor will ask for
Fit with tools	Landscape rows against tool support, from the matrix
Infrastructure	Capacity and network. Delphix's best-practice page, for example, gives a standard dedicated engine as 8 vCPUs and at least 64 GB of RAM, about one vCPU per concurrent masking job and at least 1 GB per job (Best practices)
People and process	Who will own rules, keys and reruns after the project

10.8.1 Assessment report outline

1. **Summary:** scope, verdict (ready, ready with conditions, not ready), the three findings that matter most.

2. **Scope and method:** LOB, systems, interviews held, evidence reviewed, POC results.
3. **Current state:** the landscape inventory and the use-case register.
4. **Sensitivity results:** the column ratings and decisions.
5. **Findings:** each with evidence, severity and a recommendation, for example "F3, High: the partner file L3 carries unmasked SSNs; mask before the next nightly run".
6. **Tool recommendation:** the filled scoring table and why.
7. **Deployment plan:** phases, effort and dependencies (next section).
8. **Risks:** the register.
9. **Appendices:** evidence pack, queries, configuration exports.

10.9 Risk register

List risks as soon as the landscape is drawn and revisit them at every checkpoint. Rate likelihood (L) and impact (I) from 1 to 3; the product ranks them.

SHOPSPHERE RISK REGISTER (EXAMPLE DATA)

ID	Risk	L	I	Mitigation	Owner
R1	A job succeeds with warnings and leaves values unmasked (Delphix default for non-conformant data)	3	3	Review warnings every run; fail on encounter for sensitive columns; success criterion S5	TDM engineer
R2	A file or system outside the plan carries the same SSN (L3, L4)	2	3	Landscape inventory; leak search (S6)	Data steward
R3	Different algorithm or key per system, so the same SSN masks differently	2	3	One algorithm instance and key for SSN; test S4	TDM engineer
R4	Masking breaks keys or joins	2	3	Never mask keys unless the same deterministic algorithm is used on every side; orphan query	DBA
R5	Tool limit discovered late (for example G-Subset needs the same database type on both sides and no composite keys; masked VDB provisioning covers only some databases)	2	2	Check limits in the landscape step; test in POC	TDM engineer
R6	In-place job fails midway and leaves semi-masked data (the Delphix docs warn about this on cancel)	2	3	Snapshot or backup before every run; restore procedure tested	DBA
R7	Masked data too unrealistic for tests	2	2	Tester review of a sample; adjust algorithms	QA lead
R8	Schema changes add new sensitive columns that nobody profiles	3	2	Re-profile on each release; rule set refresh; change control	Data steward
R9	Engine capacity too small for the real volume	2	2	Sizing in the assessment; measure in POC (S8)	Infrastructure

ID	Risk	L	I	Mitigation	Owner
R1 0	Sponsor expects "100% secure" with no definition	2	3	Agree the success criteria and sign-off checklist in writing	Project manager

10.10 The deployment plan

A deployment plan turns a successful POC into a repeatable service. It names phases, the condition for entering and leaving each one, and who owns it.

Phase	Main activities	Exit condition
P0 Mobilise	Approve scope and success criteria; accounts and approvals; baseline KPIs measured	Signed charter and baseline
P1 Prepare	Size and install the tool; network routes; masking database users with only needed rights; backup and restore tested	Smoke test: connect, profile one table
P2 Configure	Connectors, rule sets, profiling, inventory review, algorithms, keys or seeds, roles; for Informatica, policies, projects and plans	Inventory approved by the data steward
P3 Pilot the LOB	Run the full LOB in one environment; verify with the POC criteria	Evidence pack accepted
P4 Roll out	Remaining environments and files; schedule jobs; connect to the refresh process; train users	Sign-off checklist complete
P5 Operate	Monitor jobs, re-profile on releases, report KPIs, review risks	Quarterly review

Some tool features support promotion. Delphix lets you export an environment, with its connectors, rule sets and jobs, and import it into another masking engine; the docs say moving environments between engines is the main reason (Managing environments). It also offers an inventory approval workflow that makes an approver sign off a database inventory before any masking job can run. Informatica generates

a workflow from a plan and runs it (User Guide 10.5.10, p.32). Use these to move work from the pilot to production setups without re-typing it.

10.10.1 The refresh runbook

The safest ordering is: copy into a closed area, mask, verify, and only then open the copy to testers. Nobody sees unmasked data at any point. Delphix's masked provisioning enforces a version of this for supported databases by masking during VDB provisioning (see Provision masked VDBs).



10.10.2 The generation (TDG) deployment plan

For generation the same phases apply with different content. GenRocket describes a hybrid design: modelling and design happen in the cloud component, and the data itself is generated locally in the on-premises component (GenRocket architecture). Its five components are domain, attribute, generator, receiver and scenario (five key components).

Step	For ShopSphere
Model	Domains for CUSTOMERS, ORDERS, ORDER_ITEMS, PRODUCTS; attributes per column; parent-child links
Design	A generator per attribute, scenarios for UC3 to UC5, volumes (5 million orders), boundary and null cases (see the data patterns chapter)
Deploy	Runtime on the generation machine, a receiver that loads the QA database or writes files, access to the target
Manage	Version the scenarios, rerun on schema change, track generation time and test coverage gained

10.11 Sign-off

Sign-off is a decision with names attached. Each line of the checklist below needs evidence a reviewer can open.

#	Check	Evidence	Signer
1	Scope equals the use-case register and landscape inventory	Register and inventory, with changes listed	Project manager
2	Every in-scope sensitive column has an approved, verified treatment	Exported inventory; Informatica Data Masking Report (p.322) or Delphix inventory CSV	Data steward
3	Keys intact, formats kept, same value masked the same across systems	Queries S2 to S4 and their output	DBA, QA lead
4	No unexplained warnings and no real value left	Job summaries; leak search S6	TDM engineer, compliance officer
5	Rollback and restore tested	Test record	DBA
6	Access limited to named roles; audit logging on	Role list; audit log extract	Security
7	Runbook and rules owned by named people	Runbook, RACI	QA lead
8	KPI baseline captured and targets agreed	KPI table	Sponsor
9	Business acceptance of the masked data's usefulness	Tester sample review	Business owner

A written decision, signed and dated, with one of three outcomes: accepted, accepted with listed conditions and due dates, or rejected with reasons. Informatica's guide notes that its Data Validation

Option can give test data "verifiable proof" that no sensitive values were left (User Guide 10.5.10, p.24). Plan audit reports are intended for auditors (p.322). Both are examples of tool-made evidence you can attach.

10.12 Governance and audit

Governance is what keeps the masked data trustworthy after the project team leaves.

Control	Documented behaviour	Practice to adopt
Who can change rules	Delphix roles such as IT Security Analyst, DBA, SME, Operator and Environment Owner (Users and roles); Informatica roles and privileges decide allowed tasks (p.23)	Least privilege; separate who edits rules from who runs jobs
Keys and seeds	Most Delphix algorithms include a key, and changing it changes the output (Algorithms); Informatica key masking is repeatable for the same seed (1 to 1,000 for strings, p.168)	Treat keys and seeds as controlled configuration; change only through change control, because a change breaks cross-system consistency
Change control	Delphix has an inventory approval workflow (database rule sets)	Review every rule change; re-profile on each release
Audit trail	Delphix logs user actions and keeps the last one million entries by default, recommending periodic API export (Audit logs); Informatica's Audit Trail Report lists recent activity as a CSV (p.322)	Export on a schedule and retain per policy
Evidence retention	Plan audit and row count reports (Informatica p.322, 324); job reports (Delphix)	Keep the pack with the release it protects

10.13 Roles and RACI

R = does the work, A = answers for it (one per row), C = consulted, I = informed.

Activity	TDM eng.	DBA	Data steward	Compliance	QA lead	Proj. mgr
Use-case discovery	R	C	C	C	R	A
Landscape inventory	R	R	C	I	C	A
Sensitivity decisions	C	I	R	A	C	I
Tool selection and POC	A/R	C	C	C	C	I
Masking rules and keys	R	C	A	C	I	I
Copy, restore and rights	C	A/R	I	I	I	I
Run and verify refresh	A/R	C	C	I	C	I
Sign-off	R	C	C	A	C	R
KPI reporting	R	I	I	I	C	A

10.14 KPIs: how you show it worked

Measure before you start, or you can only say "it feels better". The four KPIs below are a common starting set. Capture the baseline in the first phase of the plan.

SHOPSPHERE KPIS (EXAMPLE BASELINE AND TARGET)

KPI	Definition	How measured	Baseline	Target
Provisioning time	Time from a tester's request to usable, verified, masked data	Ticket timestamps	3 days	4 hours
Refresh time	Duration of the whole refresh run, from copy to release	Runbook timestamps; job executions	3 days	1 day or less
Defects caused by bad data	Defects closed as data-related, divided by all defects in the period	Defect tracker label	18%	8% or less
Sensitive data masked	Sensitive columns masked in non-production divided by sensitive columns in the approved inventory	Inventory export and verification queries	0%	100%, verified

Two supporting measures catch quiet failures: **unexplained warnings per run** (target 0; the tools can finish a job with non-conformant values left unmasked) and **columns not in the inventory** found by the latest re-profile. A KPI is only useful with a named owner and a review date; put both in the RACI.

10.15 Common failure modes

Failure	How it shows up	Prevention
Unclear scope	New systems added weekly; no end date	Use-case register; change requests
One system masked, its copies not	A partner file or log still carries real SSNs	Landscape inventory; leak search
Inconsistent masking across systems	Joins on SSN or tax id return nothing	One algorithm and key per data element; cross-system test
Masked data too unrealistic	Testers bypass the masked copy and ask for production	Tester sample review; formats and distributions kept
Warnings ignored	"Succeeded" jobs leave bad values unmasked	Explain every warning; fail on encounter for sensitive columns
Rules go stale	A new column appears after a release and is never profiled	Re-profile each release; change control
Tool picked before the problem	The tool cannot read a main data source	Landscape and POC before purchase
No baseline	Cannot prove benefit; sponsor loses interest	KPI baseline in the first phase
Knowledge in one head	The rules and keys are lost when someone leaves	Runbook, RACI, keys in controlled configuration

10.16 Recap

RECAP

- Every engagement follows one life cycle: use cases, landscape, sensitivity, tool choice, POC, assessment, deployment, sign-off. Generation and masking differ in emphasis, not order.
- Find use cases by interviewing; record them in a register with a type and priority.
- Inventory the landscape and rate each column; check every row against what the tools can read.
- The three tools have different centres of gravity: Informatica TDM is one console for discover, subset, mask and generate; GenRocket is synthetic generation, with G-Subset for subsetting and masking; Delphix pairs masking with virtual copies. Gaps in the documentation mean "not covered", not "not possible".
- A POC needs measurable criteria, a copy, and a pass or fail decision made in advance.
- The assessment, risk register, deployment plan and sign-off checklist turn the POC into a service.
- Keep keys and seeds under change control, keep audit logs, define RACI and KPIs.

Worked project: masking and subsetting ShopSphere end to end

This worked project applies the earlier techniques to ShopSphere in three parts: profile and mask FirstName and LastName; profile and subset; profile and mask SSN, alphanumeric values, TaxID and ZipCode. Each part has a goal, a tool, configuration steps, validation criteria and test queries. The main design question is how to mask SSN, TaxID and ZipCode while keeping format and integrity; documented facts are kept apart from design choices.

10.17 Words to know first

Acceptance criterion

A measurable statement that decides whether a piece of work is done, such as "0 orphan orders".

Deterministic masking

The same input always gives the same masked output (for the same algorithm and key). It lets one SSN mask identically in a database and a file.

Format-preserving

The masked value has the same shape as the original: same length, same dashes, digits stay digits.

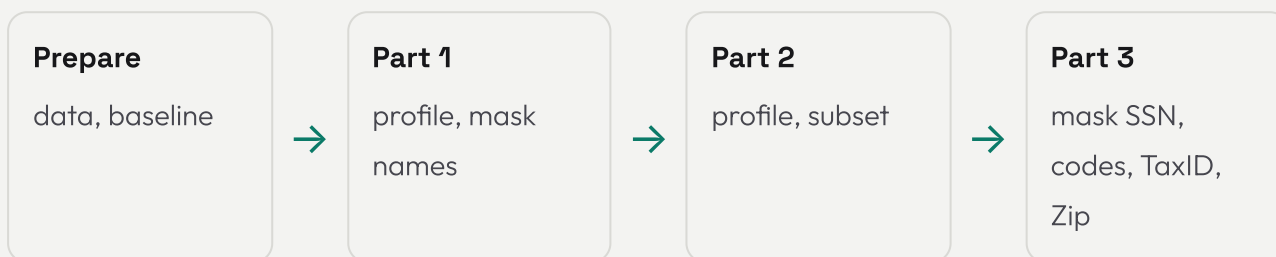
Referential integrity

Every foreign key still points at an existing row, so joins still work (data basics).

10.18 The project at a glance

The running example is ShopSphere again: CUSTOMERS, ORDERS, ORDER_ITEMS and PRODUCTS, plus two additions, `membership_code` (an alphanumeric value such as `SS-A7K29X`) and a customer extract file. The sample rows use six-digit postal codes (560001) and a US-style SSN (123-45-6789), and TaxID is a nine-digit value such as `98-7654321`.

Part	Goal	Columns or tables	Primary tool	Why this tool
1	Profile and mask FirstName, LastName	first_name, last_name	Delphix; Informatica as the cross-check	The Delphix Name algorithms are documented
2	Profile and subset	All four tables, subset by a criterion	Informatica TDM	Its entities and driving tables are built for referentially intact subsets (User Guide 10.5.10, p.114)
3	Profile and mask SSN, alphanumeric values, TaxID, ZipCode	ssn, membership_code, tax_id, zip_code	Delphix; Informatica as the cross-check	Both tools document format-keeping algorithms
Support	Generate source rows and edge cases	Source data, edge rows	GenRocket	It generates fake rows in volume and edge cases without production data (Why synthetic test data)



10.19 Preparing the source

Set up the lab database and files, with enough rows to make a subset visible (generate more if needed). Take a "before" snapshot, which the test queries compare against as `customers_before`, record baseline counts, and confirm the orphan queries return 0 on the source. All masking runs on a restorable copy; production is never touched.

10.20 Part 1: profile and mask FirstName and LastName

10.20.1 Scope

Inputs. CUSTOMERS with first_name and last_name; the "before" snapshot; two added edge rows: (1005, 'Ann-Marie', 'von Wilke') and (1006, 'S.', 'K.'), plus the duplicate-name row 1004 (Asha Iyer) from the labs. **Tool.** Delphix primary; Informatica as a cross-check. **Output.** Masked names and an inventory showing the two assignments.

10.20.2 Steps

1

Connect and profile.

Follow Labs 4 and 5 in the Delphix labs; concepts in the Delphix profiling chapter. For Informatica, the equivalent is a data domain profile (see the Informatica profiling chapter).

Expected result: first_name is assigned FIRST_NAME and last_name LAST_NAME, as the official profiling screenshots show; record each assignment.

2

Choose the algorithms and write down why.

Delphix: d\px-core:FirstName and d\px-core:LastName . Both are instances of the Name framework; they are deterministic, require string input, return null, empty and whitespace-only values unmasked, and are not unique, so a masked value may equal the input (FirstName, LastName). A single character is treated as an abbreviation and masked to a single character; hyphenated words count as one word. The default FirstName instance returns at most two names; the LastName instance is configured with a particles file and masks only the first word after removing particles. Informatica alternative: substitution masking from a name dictionary, with Repeatable Output for the same replacement each run (User Guide 10.5.10, pp.154, 176 to 177).

3

Mask only these columns.

Use a rule set whose inventory has just these two assignments (duplicate, clear, assign), as in Lab 7, and run an in-place job on a restored or snapshotted copy.

4

Verify using the queries below and the job's execution summary.

10.20.3 Acceptance criteria and test queries

ID	Criterion	Test
AC1-1	Both columns profiled and assigned	Inventory export lists both with domain and algorithm
AC1-2	Names changed	Query 1: rows where the name equals the original (expect 0; a rare equal value is allowed by the docs and must be explained)
AC1-3	Same input, same output	Query 2: both Asha rows share one masked first name
AC1-4	Nothing else changed; row count same	Query 3
AC1-5	Edge rows behave as documented	Query 4: initials stay single characters; the hyphenated name is masked as one word
AC1-6	No warnings for these columns	Execution summary: nonconforming data 0

```
-- 1 unchanged names (compare with your before-copy table)
SELECT COUNT(*) FROM customers c JOIN customers_before b USING
(customer_id)
WHERE c.first_name = b.first_name OR c.last_name = b.last_name;
-- 2 consistency
SELECT COUNT(DISTINCT first_name) FROM customers WHERE customer_id IN
(1001, 1004);
-- 3 nothing else changed
SELECT COUNT(*) FROM customers c JOIN customers_before b USING
(customer_id)
WHERE c.phone <> b.phone OR c.ssn <> b.ssn OR c.zip_code <> b.zip_code;
-- 4 edge rows
SELECT customer_id, first_name, last_name FROM customers WHERE customer_id
IN (1005, 1006);
```

10.21 Part 2: profile and subset

10.21.1 Scope

Goal. Produce a smaller copy of ShopSphere that is still referentially intact: a chosen set of customers with all their orders, all the items of those orders, and the products those items use.

Inputs. A source with enough rows to make a subset visible: at least 30 customers across several postal prefixes, with orders and items. If your source is only the three lab rows, generate more: GenRocket produces fake rows without touching production, using generators such as NameGen, PhoneNumberGen and SSNGen (five key components; SSNGen sequential or random: SSNGen).

Criterion (design choice). Customers whose zip_code starts with 560 .

10.21.2 Steps with Informatica TDM (primary)

1

Create a project and import the metadata.

The process (User Guide 10.5.10, pp.30 to 32) starts with a project and imported source metadata. See the Informatica tour.

2

Profile keys and relationships.

Run primary key and entity profiles where keys are not declared, and apply the results (p.31; the Informatica profiling chapter). A subset follows relationships, so missing or undeclared ones (for example ORDER_ITEMS to PRODUCTS through product_code) give a broken subset.

3

Create an entity with CUSTOMERS as the driving table.

The guide says an entity is a driving table plus related tables, found by physical and logical constraints; you add filter criteria on a column, which can be a value or a parameter changed at run time (pp.113 to 114). Put the criterion on zip_code and check the Entity Map: confirm ORDERS, ORDER_ITEMS and PRODUCTS are present and the relationships are enabled. See the Informatica subset chapter.

4

Create a subset plan, generate the workflow and run it.

Add the entity to a plan with a target connection (plans and workflows). The guide notes a masking and a subset operation can share one plan (p.32), which is what a real project would do so the subset is never exposed unmasked.

5

Run the Row Count Report on the plan: it lists the tables and rows each plan affects

Alternatives. Delphix rule sets accept a per-table filter (`customers.zip_code LIKE '560%'` , with the table prefix on the column, as the docs show) or custom SQL that must include the key columns; the docs do not describe following related tables automatically, so related-table filters would be yours to write (Managing rule sets). GenRocket's G-Subset subsets by a where clause, percentage or row count, supports no composite keys, and migrates only between databases of the same type (G-Subset overview).

10.21.3 Acceptance criteria and test queries

ID	Criterion	Test
AC2-1	Relationships profiled or declared and recorded	Relationship map or profile results
AC2-2	Only the selected customers are present	Query 1: customers outside the criterion (expect 0)
AC2-3	Complete family: every order of a selected customer is present	Query 2: compare order counts per customer with the source
AC2-4	No orphans	Queries 3 to 5
AC2-5	Subset is smaller, with a stated ratio	Row counts target against source, per table
AC2-6	Source unchanged	Source row counts equal the baseline

```
-- 1 customers outside the criterion (run on the target)
SELECT COUNT(*) FROM customers WHERE zip_code NOT LIKE '560%';
-- 2 orders per customer, run on target and source, then compare
SELECT customer_id, COUNT(*) FROM orders GROUP BY customer_id ORDER BY
customer_id;
-- 3 orders without a customer
SELECT COUNT(*) FROM orders o LEFT JOIN customers c ON c.customer_id =
o.customer_id WHERE c.customer_id IS NULL;
-- 4 items without an order
SELECT COUNT(*) FROM order_items i LEFT JOIN orders o ON o.order_id =
i.order_id WHERE o.order_id IS NULL;
-- 5 items without a product
SELECT COUNT(*) FROM order_items i LEFT JOIN products p ON p.product_code
= i.product_code WHERE p.product_code IS NULL;
```

A SUBSET IS REAL DATA

A subset copies real rows. Until it is masked (Parts 1 and 3), treat the subset target as sensitive. A real run puts masking and subsetting in one plan so the subset is never exposed unmasked.

10.22 Part 3: profile and mask SSN, alphanumeric values, TaxID and ZipCode

10.22.1 Scope

Goal. Mask four kinds of identifying value so the data stays useful: formats are kept, the same real value always becomes the same masked value, and keys and joins still work. **Inputs.** CUSTOMERS with ssn, membership_code, tax_id and zip_code; the customer extract `customers.dat` (which carries the SSN); the "before" copy; and the edge row 1004 with SSN N/A. **Tool.** Delphix primary; Informatica as a cross-check. **Output.** Four verified columns and an algorithm sheet.

The word "integrity" has two meanings here, and you must show both. *Referential integrity*: foreign keys still match. None of these four columns is a key, so the safest decision is not to mask any key at all. *Consistency*: the same SSN or TaxID appears in several places (the database and the extract file) and must mask to the same value everywhere, or joins between systems fail. Deterministic algorithms, with the same algorithm instance and key used everywhere, give consistency.

10.22.2 Algorithm design, column by column

The table separates what the docs state from the design choice. The Delphix pages do not state the default algorithm for the SSN, ZIP and TAX_ID domains; check Settings, Domains on the engine and

decide whether to keep or replace each default.

Column	Delphix choice	What the docs state	Design choice and cautions
ssn 123-45-6789	dLpx-core:CM Digits	Masks every ASCII digit anywhere in the string and keeps letters and punctuation; it is an instance of Character Mapping, which is deterministic until rekeyed, and gives no two text inputs the same output (CM Digits, Character Mapping)	Chosen for format keeping and uniqueness. Caution: on a <i>numeric</i> column leading zeros are not significant, so collisions can occur ("7" and "007"); the docs advise CM Numeric for numeric types. The docs do not say the output obeys US SSN validity rules
ssn, alternative	Segment Mapping instance, dashes ignored	The docs name SSNs as a use for Segment Mapping, say it produces no repeats if the allowed output range is at least as large as the input range, and allow ignoring dashes and preserving segments such as the last four digits (Segment Mapping)	Use if you need position-specific rules. Configuration sketch below
members hip_code SS-A7K29X	dLpx-core:CM Alpha-Numeric	Masks ASCII digits and upper and lower case letters, preserves other characters; for non-numeric types; at least one maskable character per value, else a warning; "ABCxyz123" becomes "HANwhp391" (CM Alpha-Numeric)	Digits stay digits and letters stay letters, but the "SS-" prefix is masked too. To keep it, a Character Mapping instance with a preserve range of length 2 (see zip below)

Column	Delphix choice	What the docs state	Design choice and cautions
tax_id 98- 7654321	A Character Mapping instance with group [0-9] (or CM Digits)	Digits are mapped within the group and the hyphen, being outside it, is kept	A separate instance gives TaxID its own key, so that SSN and TaxID values with equal digits do not mask alike. Use the same instance for the database and any file that carries TaxID
zip_code 560001	A Character Mapping instance: group [0-9] , preserve range start 0 length 3, forward	The preserve-range parameter exists: positions count only maskable characters, start at 0, and cannot be combined with preserve-leading-zeros	Keeping the first three digits keeps the region, which is a privacy trade-off: postal codes are partly identifying, so the compliance officer must approve it. Store as text. For US ZIP+4 values, RepeatFirstDigit masks only the "+4" part (RepeatFirstDigit)

To create the custom instances in the UI (Settings, Algorithms, + Algorithm): enter a unique name, choose the Character Mapping framework, add the character group, tick Case Sensitive if relevant, set Minimum Masked Positions, and add the preserve range. Through the API the shape is as below. The framework ids differ between engines, so look them up first with `GET /algorithm/frameworks` ; the field names are from the documentation examples.

```

{ "algorithmName": "SS_ZipKeepRegion", "algorithmType": "COMPONENT",
  "frameworkId": <Character Mapping id>,
  "algorithmExtension": { "caseSensitive": true, "characterGroups":
["0123456789"],
    "minMaskedPositions": 1, "preserveLeadingZeros": false,
    "preserveRanges": [ { "start": 0, "length": 3, "direction": "FORWARD"
} ] } }

{ "algorithmName": "SS_SSN_Seg", "algorithmType": "COMPONENT",
  "frameworkId": <Segment Mapping id>,
  "algorithmExtension": { "autoIgnoreCharacters": true,
"allowShortSegments": false,
    "segments": [ { "length": 3, "segmentType": "MASK_NUMERIC" },
                  { "length": 2, "segmentType": "MASK_NUMERIC" },
                  { "length": 4, "segmentType": "MASK_NUMERIC" } ] } }

```

The Segment Mapping sketch masks the three groups of digits separately and ignores the dashes. Left at its default ranges it may produce area numbers such as 000, which real SSNs never use. Restricting ranges is possible (the docs describe input and mask value ranges), but if input and mask ranges differ, values outside the input range are left unmasked and the mapping is no longer reversible for tokenization, so test it before relying on it.

10.22.3 Informatica cross-check

Column	Technique (User Guide 10.5.10)	Parameters and cautions
ssn	SSN masking (p.175): a built-in format for string columns that generates valid SSNs and avoids numbers on the Social Security Administration's High Group List	Accepts any nine-digit format with any delimiters; Repeatable Output with a seed gives deterministic results. The guide says it cannot return all-unique SSNs, so check uniqueness if the column has a unique constraint
member ship_cod e	Key masking, string (pp.164 to 168)	Mask Format characters: A alphabetic, D digit, N alphanumeric, X any, + no masking. For SS-A7K29X a format such as <code>+++NNNNNN</code> keeps the prefix and hyphen. N allows a letter to become a digit, so it keeps length but not letter/digit type; seed 1 to 1,000 for repeatability
tax_id	Key masking, string	Mask format <code>DD+DDDDDDDD</code> : digits replaced by digits, hyphen kept. Same seed wherever the value appears. The guide does not promise unique output for key masking, so check with a distinct count
zip_code	Key masking, string	Mask format <code>+++DDD</code> keeps the first three characters. For a numeric column the numeric key masking options differ

The mask format meanings and the seed range are from the key masking pages (pp.164 to 168); the concrete formats for ShopSphere are design choices. See the Informatica masking chapter.

10.22.4 Steps

1

Profile and review.

Run a profiling job over the rule set (Lab 5). By the published ASDD patterns, `ssn` and `tax_id` are found by column name and type, `zip_code` by name (its data rule targets US 5-digit and ZIP+4 values, so six-digit codes will not match by value), and `membership_code`, with no matching name, probably not at all. Assign it by hand.

Expected result: An inventory listing four columns with a domain or a manual note, and a log of what the profiler missed.

2

Create the custom algorithm instances (`zip`, `tax_id`, and optionally `membership_code` with prefix kept), and record the exact configuration in the algorithm sheet.

3

Assign the algorithms in a rule set that holds only these four columns, with Enable Automatic Updates off.

4

Mask the extract file with the same SSN algorithm.

Use the file route of Lab 6 (format file, flat-file rule set, inventory) and assign the *same algorithm instance* to the `ssn` field. Using the identical instance is what makes the database and the file agree.

5

Snapshot, run the job, then run the file job. Read the execution summary: the planted N/A SSN should produce a non-conformant warning (CM Digits needs at least one digit; see the algorithms and masking jobs chapter).

6

Verify with the queries below and save the output.

10.22.5 Acceptance criteria and test queries

ID	Criterion	Test
AC3 -1	Four columns profiled and decided	Inventory export
AC3 -2	SSN keeps its format and stays unique, none equals the original	Queries 1 and 2
AC3 -3	TaxID keeps its format	Query 3
AC3 -4	ZipCode keeps six digits and region, full value changed	Query 4
AC3 -5	membership_code keeps its pattern; NULL stays NULL	Query 5
AC3 -6	SSN identical in the database and the file	Shell comparison
AC3 -7	Keys and counts intact; other columns unchanged	Orphan query; counts
AC3 -8	Every warning explained	Execution summary, log patterns for row 1004
AC3 -9	No original value left anywhere	Search all masked outputs for the original SSNs and tax IDs

```

-- 1 SSN shape and uniqueness (real rows only)
SELECT COUNT(*) FROM customers WHERE customer_id ≤ 1003 AND ssn !~ '^[0-9]{3}-[0-9]{2}-[0-9]{4}$';
SELECT COUNT(DISTINCT ssn) FROM customers WHERE customer_id ≤ 1003;
-- expect 3
-- 2 nothing equals the original
SELECT COUNT(*) FROM customers c JOIN customers_before b USING
(customer_id)
WHERE c.customer_id ≤ 1003 AND (c.ssn = b.ssn OR c.tax_id = b.tax_id OR
c.zip_code = b.zip_code);
-- 3 tax id shape
SELECT COUNT(*) FROM customers WHERE customer_id ≤ 1003 AND tax_id !~
'^[0-9]{2}-[0-9]{7}$';
-- 4 zip: six digits, first three kept
SELECT COUNT(*) FROM customers c JOIN customers_before b USING
(customer_id)
WHERE c.customer_id ≤ 1003 AND (c.zip_code !~ '^[0-9]{6}$' OR
LEFT(c.zip_code,3) <> LEFT(b.zip_code,3));
-- 5 membership code pattern (prefix masked) and NULL kept
SELECT COUNT(*) FROM customers WHERE customer_id ≤ 1003 AND
membership_code !~ '^[A-Z]{2}-[A-Z0-9]{6}$';
SELECT membership_code FROM customers WHERE customer_id = 1004;
-- expect NULL

# 6 consistency (shell; your SQL client may differ)
psql -At -F'|' -c "SELECT customer_id, ssn FROM customers WHERE
customer_id ≤ 1003 ORDER BY 1" > db_ssn.txt
awk -F'|' '{print $1"|" $4}' customers.dat | sort > file_ssn.txt
diff db_ssn.txt file_ssn.txt #
expect no output
# 9 leak search
grep -rn -e "123-45-6789" -e "98-7654321" masked_outputs/ #
expect nothing

```

Expect the "bad" queries for AC3-2 to AC3-5 to count 0. Query 2 shares the earlier caution: an equal value is not strictly impossible for every algorithm. For Character Mapping the docs say the masked value never matches the input when at least one maskable character is present, so any match there is a finding.

10.23 Common mistakes and gotchas

COMMON MISTAKES

- **Masking a key to "be safe".** It breaks joins. Leave keys alone unless the same deterministic algorithm is applied on both sides.
- **Different algorithms for the same value.** An SSN masked differently in the database and the file ruins consistency; reuse one instance.
- **Numeric columns and leading zeros.** Digit masking on a numeric zip or tax id can collide; keep such values as text or use the numeric algorithm.
- **Trusting profiling for six-digit postal codes.** The documented ZIP data patterns target US formats; the column name does the work.
- **A subset left unmasked.** It is real data; mask it in the same plan.
- **Presenting design choices as product facts.** CM Digits is a suitable choice for SSN, not the product's SSN algorithm.

10.24 Recap

RECAP

- The project has three parts: names, subset, and SSN, alphanumeric values, TaxID and ZipCode.
- Use Delphix for the masking parts and Informatica for the subset, with the other tool as a cross-check and GenRocket for fake source rows.
- Every part has acceptance criteria that a query or report can prove.
- Format-preserving, deterministic algorithms (CM Digits, CM Alpha-Numeric, a Character Mapping instance with a preserve range, or Informatica key masking with a mask format and seed) keep shape and consistency. Segment Mapping and Informatica SSN masking are documented alternatives with their own limits.
- Integrity means both intact foreign keys and the same value masked the same way in every system.
- A subset is real data until it is masked.
- Keep documented facts apart from design choices, and explain every warning.

Glossary

Plain-English definitions of the terms used in the guide, each with a pointer to the section that explains it. Terms are alphabetical within each group: general ideas first, then privacy, ETL, and one group per tool. Tool-specific terms carry the tool name in brackets, for example "(Delphix)".

SAME WORD, DIFFERENT TOOL

A *domain* is a named kind of data found by patterns in Informatica, a kind of sensitive data with one default algorithm in Delphix, and a table-like thing you generate rows for in GenRocket. A *rule* changes data in Informatica, but a Delphix *rule set* only lists tables or files. Check the bracketed tool name before you trust a definition.

10.25 General TDM and data terms

Acceptance criterion

A measurable statement that decides whether a piece of work is done, such as "0 orphan orders". Section 10.17

Assessment

A structured review of an application and its data environment, ending in a written report of findings and recommendations. Section 10.8

Audit trail

A record of who ran what, when and with which settings, kept so compliance questions can be answered later. Section 3.35

Boundary value

A value at, just below or just above a limit the application enforces, such as ages 17, 18 and 19 for an 18-plus rule. Section 3.3

Collision

Two different inputs masked to the same output. Fine for names, harmful in a column that must be unique. Section 2.34

Column / field / attribute

One fact held for every row of a table, such as first name or order date. Section 1.22

Composite key

A primary key made from more than one column. Section 1.23

COTS

Commercial off-the-shelf software: bought ready-made from a vendor rather than built in house. Section 1.5

Coverage matrix

A grid of test conditions against the data that covers each one, so gaps show at a glance. Section 3.1

Data

Raw facts such as "1001" or "Asha", before anyone says what they mean. Section 1.21

Data pattern

One of five shapes for values: boundary conditions, sequential, random, edge cases and null. They serve test purposes rather than form a separate classification. Section 3.4

Data profiling

Scanning a data source with software and reporting facts about it, such as nulls, uniqueness and patterns. Section 2.16

Data subset

A smaller copy of a database that still hangs together: fewer rows, same structure, referential integrity intact. Section 2.20

Data type

The kind of value a column holds (number, text, date), which controls what is allowed. Section 1.22

Database

An organised store of data that a program can search and change reliably. Section 1.22

Database administrator (DBA)

The person who grants the TDM tool access to source and target databases and helps with performance and storage. Section 1.13

Deterministic masking

Masking that turns the same input into the same output every time, so a customer is masked identically in every table and run. Section 2.33

Dictionary / lookup file (masking)

A list of replacement values, such as thousands of first names, that substitution picks from. Section 2.32

Discovery (data discovery)

Finding out what a data source holds: which tables and columns exist, which are sensitive and how tables relate. Section 2.16

Driving table

The table you start from when choosing a subset; related rows are pulled in from it. Section 2.22

Dynamic masking

Masking at the moment of reading, depending on who asks. It protects production views, not test copies, and the guide's sources do not cover it. Section 2.35

Edge case

A rare but possible input outside normal use, such as a surname with an apostrophe or a date on 29 February. Section 3.3

Encryption

Scrambling data with a key so key holders can unscramble it. Reversible, so it is not masking. Section 2.30

Engagement

One piece of TDM work for one client or business unit, from first conversation to hand-over. Section 10.1

Environment

A complete running copy of the application with its own servers and database: dev, QA, UAT, staging or production. Section 1.7

Evidence pack

The queries, screenshots and reports that prove each success criterion is met. Section 10.1

False positive / false negative

Discovery mistakes: flagging a column that is not sensitive, or missing one that is. The second is more dangerous. Section 2.19

Flat file

A plain file holding records as lines of text, with no database engine, keys or data types behind it. Section 1.24

Foreign key (FK)

A column whose values must match the primary key of another table; it is the link between tables. Section 1.23

Format-preserving

A masked value that keeps the original's shape: a 10-digit phone stays 10 digits, a card number keeps its checksum. Section 2.32

Hand-crafted data

Test rows typed or edited by a person: slow, hard to repeat and tiny in volume. Section 3.3

Hash (MD5)

A one-way scramble: the same input always gives the same output, but the output cannot be turned back into the input. MD5 is one hash function, with a 128-bit result. Section 6.18.11

Hybrid test data

Masked or queried production data with generated rows added on top. Section 3.3

Information

Data with meaning: "customer 1001 is Asha, living in postal area 560001". Section 1.21

In-place masking

Masking that changes the values inside the database you point the job at. Never aim it at production. Section 2.35

Irreversible

You cannot recover the original value from the masked one. Section 2.34

KPI

Key performance indicator: a number tracked to show whether TDM is helping. Section 10.14

LOB

Line of business: one business area with its own applications and data. Section 10.1

Logical key (general)

A column the tool is told to treat as a key, with no change to the real database. Section 1.23

Luhn check

The checksum formula that real payment card numbers follow, used by payment forms to catch mistyped numbers. A Luhn-valid number is not necessarily a real card. Section 6.18.11

Masking

Replacing real sensitive values with realistic but fictitious ones so test data cannot identify real people. Section 2.30

Metadata

Data about data: table and column names, types, lengths and key definitions, not the values. Section 1.22

Metadata profiling / data profiling

Metadata profiling reads names and types; data profiling reads the values. They find different things. Section 2.17

Negative test / negative data

A test, or data, the application should reject or handle safely, such as an order total of -5.00. Section 3.3

Non-deterministic (random)

Giving a different output each time for the same input. Section 2.29

Non-production environment

Any copy of the application that is not live: development, QA, UAT or staging. Section 1.5

Null

"No value at all" in a column, different from an empty string, which is a value of zero characters. Section 7.7

Obfuscation

Umbrella word for making data hard to read by any method; masking is one kind. Section 2.30

On-the-fly masking

Reading unmasked data from a source and writing only masked data to a different target. Section 2.35

Orphan

A child row whose foreign key points at no existing parent; it means the data is broken. Section 1.23

Parent table and child table

The table pointed at is the parent; the table holding the foreign key is the child. Section 1.23

Pilot (services-partner sense)

In one services partner's approach, a real medium-sized project chosen to cover the key scenarios and show the client measurable improvement before a roadmap and rollout. It is larger than a lab POC. Section 10.7

POC (proof of concept)

A small, time-boxed trial of an approach on a slice of real systems, run before full rollout. Section 10.7

Positive test / positive data

A test, or data, the application should accept and process normally. Section 3.3

Primary key (PK)

A column, or columns, whose value is unique in every row and never empty. Section 1.23

Production (prod)

The live system real customers use; it holds real data. Section 1.7

Production support period

A set period after go-live in which the client can bring problems and questions back to the project team; the last phase of the partner's approach. Section 10.7

Provisioning

Delivering a prepared data set to a test environment so testers can use it. Section 3.35

RACI

A table marking who is Responsible, Accountable, Consulted or Informed for each task. Section 10.13

Random data

Values picked unpredictably from a set or range, giving realistic variety. Section 7.4

Record

One row of a table, or one line of a flat file: one customer, one order. Section 1.24

Referential integrity

The rule that every foreign key value matches an existing primary key value. Section 1.23

Refresh

Replacing the data in a test environment with a newer copy. Section 1.5

Regression testing

Re-checking that old features still work after a change; it wants the same data every run. Section 1.7

Repeatable

Giving the same results when re-run. Informatica uses the word for deterministic masking switched on by a seed. Section 2.31

Risk register

A living table of things that could go wrong, with likelihood, impact, owner and mitigation. Section 10.9

Sampling

Reading or picking only some rows; efficient, but a rare value may be missed. Section 2.17

Schema

The description of the data (tables, columns, types, keys) as opposed to the data itself. Section 1.22

Seed

A starting number for a random picker. The same seed gives the same "random" values every run. Section 3.7

Sequential data

Values that follow a fixed order such as 1, 2, 3, 4; the usual choice for ids. Section 7.3

Shuffle

Moving a column's existing values between rows. The values stay real, so check it is safe enough. Section 2.32

Sign-off

Formal written approval by named people that agreed criteria have been met. Section 10.11

SQL

Structured Query Language, used to ask a relational database questions and change its data. Section 1.25

Stateful data

Rows in a particular condition that a workflow needs, such as an order already shipped whose payment has settled. Section 3.3.3

Static masking

Creating a masked copy once; anyone using the copy sees masked data. The normal TDM approach. Section 2.35

Subsetting

Taking a small, consistent slice of a big database instead of all of it. Section 2.20

Substitution

Replacing a value with a believable, unrelated one picked from a list. Section 2.32

Success criteria

Measurable pass-or-fail statements agreed before a POC starts. Section 10.7

Synthetic data

Data created by a program from rules, not copied from real people. Section 3.5

TDG (test data generation)

Creating synthetic test data with a tool rather than copying production. Section 10.1

TDM engineer

The person who discovers sensitive data, designs subsets, configures masking and generation and fixes failures. Section 1.13

TDM pipeline

The chain from production (or a generator) to usable test data: find sensitive data, cut down, mask, load, check. Section 3.39

TDM (Test Data Management)

The planning, tools and habits that give testers safe, realistic, right-sized data when they need it. Section 1.9

Technique (Informatica) / algorithm (Delphix)

The method that changes a value, such as substitution or date shift. Section 2.29

Technology landscape

The list of systems, databases, files and interfaces that hold or move the data in scope. Section 10.4

Test case

One specific check: "do this, with this data, expect that result". Section 1.5

Test condition

A specific situation a test wants to check, such as a customer with no phone number. Each needs data that creates it. Section 3.1

Test data

The data a test case needs: rows that exist before it starts and values typed during it. Section 1.5

Test data as a service

Test data that teams request and receive like a service-desk ticket, owned by a small group with the rules. Section 3.11

Tokenization (general)

Replacing a value with a token while keeping a protected way to reverse it. Section 2.30

UAT (user acceptance testing)

Testing where the business confirms the application does what it asked; it needs believable data. Section 1.7

Unique permutations

Every combination of values worth testing; one of the kinds of data GenRocket's CEO says masked production copies lack. Section 3.3.3

Unique / uniqueness

A column where no value repeats; masking must not create duplicates there. Section 2.34

Validation of masked data

Checking a finished job: re-run discovery, test joins, formats, uniqueness, repeatability and job warnings. Section 2.38

World Quality Report (vendor-cited)

An industry quality report that GenRocket's CEO cites for two figures: 69% of organisations use spreadsheets to generate test data and 59% still create it manually. Section 6.2.1

10.26 Privacy and regulation

Anonymisation

A goal: data that identifies no one, even in combination. Legal tests are strict, so masking alone does not guarantee it. Section 2.30

Business associate

Under HIPAA, a vendor that handles PHI for a covered entity, such as a firm testing a hospital's billing system. Section 2.5

CCPA

California Consumer Privacy Act: gives California residents rights over personal information businesses hold. Section 2.5

Compliance officer

The person responsible for making sure the organisation follows laws and internal data rules. Section 2.9

Consent / lawful purpose

Ideas shared by privacy laws: data needs a lawful reason, and testing is a different purpose from running the shop. Section 2.5

Covered entity

Under HIPAA, a health plan, healthcare provider or healthcare clearinghouse. Section 2.5

Data classification

Labelling data by how much damage a leak could do (for example public, internal, confidential, restricted) so protection fits. Section 2.6

Data minimisation

Collecting and keeping no more data than needed. It is the legal argument for subsetting. Section 2.5

Data steward

The person who knows a data set's meaning and quality and says what a column holds and who may see it. Section 2.9

De-identification

Making data no longer point to a person. HIPAA names two ways: remove 18 listed identifier kinds, or get an expert to certify tiny risk. Section 2.5

Detection signals

The clues tools use to find sensitive columns: column name, data pattern, list of values, plus type and length. Section 2.7

Direct identifier

Data that points to one person on its own, such as an SSN. Section 2.4

DPDP Act

India's 2023 consent-centred law on digital personal data. Rules are phased in, so check current status. Section 2.5

Especially sensitive data

Data many laws treat as extra delicate, such as ethnic origin, religion, health or biometrics. Blanking it can beat faking it. Section 2.6

GDPR

General Data Protection Regulation: EU law, applying since May 2018, covering personal data of people in the EU wherever the company sits. Section 2.5

HIPAA

US law from 1996 whose Privacy and Security Rules protect health information that identifies a person. Section 2.5

Identifier

A piece of data that points to one person, alone (direct) or in combination (indirect).
Section 2.1

Indirect identifier (quasi-identifier)

Data that identifies only in combination, such as zip code plus birth date plus gender.
Section 2.4

Minimum necessary

HIPAA idea: people and systems get only the least PHI their job needs. Testers rarely need the real thing. Section 2.5

PCI data

Payment card details such as card number and expiry date. Section 2.3

PCI DSS

Data Security Standard from the card brands for anyone handling card data. A contractual standard, not a law. Section 2.3

PHI (Protected Health Information)

Health data that can be tied to a person. The link to an individual, not the diagnosis alone, makes it PHI. Section 2.3

PII (Personally Identifiable Information)

Data that identifies a person alone or combined with other data, such as name, email, phone or government ID. Section 2.3

Pseudonymisation

Replacing identifiers with pseudonyms while a way back exists elsewhere; still personal data under GDPR. Section 2.30

Regular expression (regex)

A compact text recipe for a pattern, for example three digits, a hyphen, two digits, a hyphen, four digits. Section 2.7

Re-identification

Working out who a "nameless" record belongs to by combining it with other information. Section 2.4

Sensitive data

Data that could hurt a person or company if leaked or misused. "Sensitive" is a business and legal judgement. Section 2.2

10.27 ETL**All-or-nothing load**

Loading inside a transaction so a failure undoes everything. Section 3.27

Batch processing

Collecting data for a period and processing it as one group; easy to re-run. Nearly all test data provisioning is batch. Section 3.23

Bulk load

Loading rows in the database's native bulk-load format instead of row by row, used for very large volumes. GenRocket's bulk load receivers write that format. Section 3.22.1

Change data capture (CDC)

Any technique that detects which source rows were inserted, updated or deleted, so only those are processed. Section 3.19

Cleaning

Making values consistent so equal things look equal, such as "MENON" and "Menon ". Section 3.21

Data quality dimensions

Completeness, validity, uniqueness, consistency and timeliness: five questions to check a load. Section 3.28

Data warehouse

A separate database built for reporting across lots of history, with a few big reads.
Section 3.17

Deduplication

Keeping one row for each real-world thing. Section 3.21

ELT

Extract, Load, Transform: load raw data first, then transform in the target. Raw sensitive data lands unmasked. Section 3.23

ETL

Extract, Transform, Load: copy data out of sources, reshape and clean it, write it to a target. Section 3.16

Extract

Reading data out of a source system without changing it. Section 3.19

Full extract / incremental extract

Reading everything every time, versus only what is new or changed since last time.
Section 3.19

Integration (ETL) engine

The program that actually reads, transforms and writes rows. Section 3.24

Load

Writing transformed rows into the target correctly, repeatably and fast enough. Section 3.22

Lookup (ETL)

Enriching a row by finding a related value in a reference table. Section 3.21

Mapping (ETL)

The definition of how source columns become target columns; also, narrowly, translating codes like S to Shipped. Section 3.21

Mapplet (ETL)

Informatica's reusable group of transformations, built once and used in many mappings. Section 3.26

Metadata repository (ETL)

A database storing mapping and connection definitions and run history. Section 3.24

Operational database (OLTP)

The database an application uses to run the business, built for many small fast reads and writes. Section 3.17

Reconciliation

Comparing counts and totals between two stages to show nothing was lost or invented. Section 3.28

Reject (error) row

A row the job could not process, written aside with a reason instead of stopping the run. Section 3.27

Scheduler

Starts jobs at the right time or after another job finishes. Section 3.24

Session (ETL)

A runnable instance of one mapping with its run settings. Section 3.25

Slowly changing dimension (SCD)

A way to keep history in a warehouse table when descriptive data, like an address, changes. Section 3.22

Staging area

A temporary landing zone where extracted data sits unchanged before transformation. Section 3.20

Streaming

Processing each event within seconds of arrival. Section 3.23

Transform

Turning staged raw data into the clean, correctly typed shape the target needs. Section 3.21

Transformation

One operation on rows, such as filter, lookup or aggregate. Section 3.25

Truncate and load

A full refresh: empty the target table, then insert everything. Section 3.22

Type conversion

Turning text into real dates and numbers so sorting and arithmetic work. Section 3.21

Upsert

"Update if the row exists, insert if it does not." Section 3.22

Watermark

A saved marker, such as the last order id loaded, telling the next run where to start. Section 3.19

Workflow / job (ETL)

A set of sessions and steps with an order and conditions, started by a schedule or trigger. Section 3.25

10.28 Informatica TDM

Actions menu (Informatica)

A menu that changes with the view; almost every task starts with "click Actions". Section 4.10

Ad hoc rule (Informatica)

An unnamed rule made directly on a column inside one project; it cannot move to another project. Section 5.17

Advanced rule (Informatica)

A rule that combines techniques using input, variable and output columns, to mask or generate several columns together. Section 5.3

Application service (Informatica)

A background program in the installation that does one job, such as running profiles or workflows. Section 4.1

Assignment (Informatica)

The link between a rule and a column, made directly or through a data domain. Section 5.2

Audit Trail Report (Informatica)

A report of all recent activities (type, object, user, time); it ignores permissions. Section 5.40

Auto cascade (Informatica)

Masks a parent key and its child foreign keys with the same value; applied by default where key constraints exist. Section 5.4

Column profile (Informatica)

Reports range, values and patterns of chosen columns so generation rules can be inferred. Section 4.32

Concatenation pipe operator (||) (Informatica)

In a PowerCenter expression, `||` joins values into one string, as in `first_name || ' ' || last_name`; the concatenation function does the same. Section 5.9

Conditional constraint (Informatica)

A table-level rule limiting one column by another, such as `To_City` not equal to `From_City`. Section 5.23

Conformance (Informatica)

The percentage of rows that fit a rule or pattern; a threshold lets a few disagree. Section 4.25

Connection (Informatica)

Saved access to a database, repository or service; plans choose source, target, staging, lookup and storage connections. Section 4.4

Constraint (Informatica)

A rule linking two tables, usually a foreign key; physical if declared in the database, logical if added in TDM. Section 4.39

Content panel and Details panel (Informatica)

The two screen areas: one lists items, the other shows more about the selected row. Section 4.10

Criteria parameter (Informatica)

A named placeholder in an entity filter so the value can change at run time. Section 4.39

Curation (Informatica)

Reviewing suggested profile results and approving or rejecting each. Section 4.25

Custom rule (Informatica)

A global rule that uses the logic of an imported PowerCenter maplet. Section 5.17

Data coverage analysis (Informatica)

Checks whether a data set has enough rows for each combination of two chosen columns, shown as a grid. Section 5.41

Data domain (Informatica)

A named kind of data, such as Social Security Number, found by data patterns or column-name patterns, with a sensitivity level and preferred rules. Section 2.8

Data domain profile (Informatica)

A profile that assigns data domains to columns by searching data, column names, or both. Section 4.31

Data generation (Informatica)

Creating new realistic but fictional rows from rules instead of copying real ones. Section 5.15

Data Integration Service (Informatica)

The application service that performs discovery profiles and data coverage tasks. Section 4.4

Data integrity for subset (Informatica)

A plan setting for how far subsetting follows relationships: Transactional (default), Referential Only, One-way or Two-way. Section 4.44

Data pack (Informatica)

A data set published to the self-service portal for testers who do not use Test Data Manager. Section 5.29

Data pattern and metadata pattern (Informatica)

Two regular expressions on a data domain: one matches values, the other matches column names. Section 4.22

Data set (Informatica)

A named collection of tables in the test data warehouse; storing it again creates a new version. Section 5.44

Data Validation Option (Informatica)

A separate Informatica solution used with PowerCenter to check that target data is accurate. Section 4.3

Dictionary (Informatica)

A file or table of ready-made values that a rule picks from; an authorised user must add it to TDM first. Section 5.22

Driving table (Informatica)

The table chosen as an entity's starting point. Section 4.39

Effective Dates technique (Informatica)

Fills start and end date columns so periods follow one another without overlap. Section 5.22

Encryption masking (Informatica)

Standard (AES, CRC, MD5) or format-preserving encryption; the latter can be decrypted with the same pass phrase. Section 5.6

Entity (Informatica)

A driving table plus all tables related to it by constraints; the unit copied to a subset. Section 4.42

Entity profile (Informatica)

A profile that finds possible entities by comparing column names across source tables. Section 4.30

Exception handling (Informatica)

Per-rule settings for nulls, blanks and errors, plus optional pre- and post-processing expressions. Section 5.5

Exception test data (Informatica)

Deliberate nulls and invalid values, set as percentages, mixed into generated output. Section 5.21

Expression Builder (Informatica)

A dialog for building a formula by picking functions and columns. Section 4.17

Expression masking (Informatica)

Applies a formula built in the Expression Builder; repeatable output needs a storage table. Section 5.6

Expression transformation (Informatica)

A PowerCenter transformation that computes a new value for every row; the mapplet example uses one to build `full_name`. Section 5.9

Generation rule (Informatica)

A saved recipe for one kind of value: a technique plus its parameters. Section 5.14

Global rule (Informatica)

A named rule created once in Policies and reusable in every project. Section 5.17

Group (Informatica)

Unrelated tables, or tables to copy unfiltered, added to a plan as they are. Section 4.43

ilmcmd, tdwcmd, tdwquery (Informatica)

Command-line programs: `ilmcmd` runs some TDM tasks, `tdwcmd` lists warehouse details, `tdwquery` runs SELECT queries on a data set. Section 5.42

Impacted Objects dialog (Informatica)

A warning listing what breaks when you change or delete something already in use. Section 4.10

Informatica Administrator

The web client where a domain administrator manages application services, users and groups. Section 4.4

Informatica domain

The family of Informatica services installed and managed together, with a shared set of users. Not a data domain. Section 4.1

Job (Informatica)

Something TDM runs, such as a profile, workflow generation or workflow run, listed in Monitor with a state like In Queue, Running, Success, Error or Terminated. Section 5.39

Key masking (Informatica)

Deterministic masking from input, rule and seed; give key columns in both tables the same seed. Section 5.6

Logical key or constraint (Informatica)

A key relationship defined only inside TDM; the source tables are not updated. Section 4.25

Major / minor constraint (Informatica)

A foreign-key severity: major pulls rows both ways, minor only child to parent. Section 4.44

Mapplet Designer (Informatica)

The part of PowerCenter Designer where mapplets are built. Section 5.9

Mapplet (Informatica)

A reusable group of PowerCenter transformations; TDM can import one as a custom masking or generation rule. Section 5.9

Mapplet Input transformation (Informatica)

The mapplet box that lists the input ports, such as `first_name` and `last_name`, that receive values from the mapping. Section 5.9

Mapplet Output transformation (Informatica)

The mapplet box that returns the output port, such as `full_name`; without it the mapplet has nothing to return. Section 5.9

Masking rule (Informatica)

A technique plus its parameters, saved under a name. Section 5.1

Masking technique (Informatica)

The logic that replaces a value, such as substitution or shuffle. Section 5.6

Model repository (Informatica)

Holds table metadata for discovery profiles and connection information. Section 4.4

Monitor (Informatica)

The view for checking job status, stopping jobs and reading logs. Section 5.39

Normal load (Informatica)

The load setting chosen in Workflow Manager when the target has a primary key; bulk loading skips some database checking and logging. Section 5.9

Nullification (Informatica)

Replaces the value with NULL. The strongest privacy result, but the target column must allow NULL. Section 5.6

Override (Informatica)

Letting whoever assigns a rule change its parameters for one column. Overridden seeds on matching columns break the match. Section 5.3

Parameter (Informatica)

A named placeholder whose value you supply at run time, so one plan serves many runs. Section 5.29

Plan (Informatica)

The saved recipe of one operation: components, source and target connections, settings. It does nothing until a workflow is generated and run. Section 5.30

Policy (Informatica)

A named bundle of data domains saying what is sensitive and how to protect it. It holds no data. Section 2.8

Policy pack (Informatica)

A ready-made policy shipped with TDM: PII, PHI or PCI. Section 2.8

PowerCenter

Informatica's ETL product, the engine under TDM. Section 4.4

PowerCenter Integration Service (Informatica)

Runs the subset, generation and masking workflows. Section 4.4

Preferred rule and default rule (Informatica)

Rules attached to a domain are preferred; one is the default for all its columns unless you pick another. Section 2.8

Profile (Informatica)

A saved search definition (tables, kind of analysis, sampling limits) you can run again. Section 4.25

Profiling warehouse (Informatica)

The database where profile results are loaded. Section 4.4

Project (Informatica)

The workspace for one job; holds imported source tables, profiles, assignments, subset definitions and plans. Section 4.18

Random masking (Informatica)

Non-repeatable output within a range or blur; no seed. Section 5.6

Reference Lookup technique (Informatica)

Takes values from an existing parent table so child keys are valid. Ad hoc only. Section 5.22

Repeatable output (Informatica)

Same input gives the same masked value in every table and run; switched on with a seed. Section 5.5

Repository (Informatica)

A relational database where Informatica keeps its own definitions and results, not your business data. Section 4.1

Reset (Informatica)

Copying a data set version from the warehouse back to a target so a test environment returns to a known state. Section 5.45

Residual tables (Informatica)

Tables not yet in any group or entity; tick them into a group to cover the whole source. Section 4.43

Restricted column (Informatica)

A column property that stops the value being written to the target. Section 2.6

Row Count Report (Informatica)

Shows the tables in a workflow run and how many rows were affected. Section 5.40

Rule (Informatica)

The logic that masks a value or generates test data: a technique plus parameters. Section 2.8

Rule Simulator (Informatica)

A built-in preview showing original beside masked values for a standard rule. Section 5.3

Seed (Informatica)

A start number, 1 to 999 and default 1, giving repeatable output. Note: sources differ; one guide table says 1,000. Section 5.5

Sensitive flag (Informatica)

A per-column label for reference only; nothing is masked until a rule is assigned. Section 2.6

Sensitivity level (Informatica)

A label on a data domain, from a list an administrator defines. Section 4.10

Sequence technique (Informatica)

Generates ordered numbers or dates from a start value and increment. Section 5.22

Set of Values technique (Informatica)

Generates a short typed list of values, each with a percentage of rows. Section 5.22

Shuffle masking (Informatica)

Swaps a column's values between rows; random, representative or constrained variants. Not allowed for XSD sources. Section 5.6

SSN, SIN, phone, credit card, email, IP and URL masking (Informatica)

Built-in techniques that keep each format valid, for example a card number keeps a valid checksum. Section 5.6

Staging connection (Informatica)

A database connection for temporary working tables; recommended as the source for cyclic subsets. Section 5.33

Standard rule (Informatica)

A built-in technique with parameters: one technique, one input column and one output column. Section 5.3

Storage connection (Informatica)

A database holding storage tables for repeatable substitution and expression masking. Section 5.33

Substitution masking (Informatica)

Replaces a value with similar unrelated data from a dictionary; unique substitution fails if the dictionary is too small. Section 5.6

Template (Informatica)

An optional container for entities and groups; the guide gives little detail. Section 4.43

Test Data Manager (Informatica)

The web client you work in; the name is also used for the whole product. Section 4.1

Test Data Manager Service (Informatica)

Creates and manages the TDM repository and runs Test Data Manager. Section 4.4

Test Data Warehouse (Informatica)

A central store for versions of test data, so teams can save and restore a known-good state. Section 5.43

Uniqueness percentage (Informatica)

The share of values in a column that are unique; 100 means no duplicates. Section 4.25

Value cascade (Informatica)

Gives similar columns not linked by a key the same masked value, via a chosen driving column. Section 5.4

Version and tag (Informatica)

Each re-store of a data set makes a new version; tags are comma-free words used to search data sets. Section 5.44

View (Informatica)

One of the main Test Data Manager screens, chosen from the tab bar. Section 4.10

Workflow (Informatica)

The runnable job TDM generates from a plan; PowerCenter runs it. Section 4.1

Workflow Manager (Informatica)

The PowerCenter tool where a workflow is created to run a mapping. Section 5.9

10.29 GenRocket

Attribute (GenRocket)

One characteristic of a domain, such as firstName; for a database it is a column. Section 6.3

Attribute property key (GenRocket)

A receiver setting for one attribute, such as columnName or include. Section 6.22

Bulk load receiver (GenRocket)

A receiver writing a database's native bulk-load file, much faster than JDBC inserts for millions of rows. Section 6.23

DDL import (GenRocket)

Creates domains and attributes from CREATE TABLE text. It does not create relationships, receivers or scenarios. Section 6.14

DelimitedFileReceiver (GenRocket)

Writes values to a delimited file such as CSV, with optional header and quoting. Section 6.24

Domain Dashboard (GenRocket)

The screen where you work on one domain's attributes, variables, receivers and scenarios. Section 6.14

Domain (GenRocket)

One "thing" you want data for. For a database it is a table; it can also be a file, message, JSON or XML document. Section 6.3

Domain Referencing Wizard (GenRocket)

Sets parents and siblings for many domains at once, with a diagram that redraws as you click. Section 6.15

Domain variable (GenRocket)

A named domain setting, `global.DomainName.variableName`, that generator parameters can use. Every domain has `id`, `loopCount` and `seed`. Section 6.16

EdgeCaseGen (GenRocket)

Slips tricky values, such as a negative age every fifth record, into otherwise normal data. Section 7.6

Encrypted scenario file (GenRocket)

The design sent from the cloud web platform to the customer side as an encrypted file (`.grs`), which the runtime executes to write data inside the customer's network. Section 6.5

Enforce Type/Limit (GenRocket)

Makes a generator keep values within the attribute's declared type and size. Section 6.18

EvalCaseGen (GenRocket)

Evaluates one equation using up to five variables and returns a true or false value. Section 7.6

FixedFileReceiver (GenRocket)

Writes fixed-width columns padded to set lengths. Section 6.24

FlexibleDateRangeGen / DateDeltaGen (GenRocket)

Date generators: one produces dates in a range and format, the other the gap between two dates. Section 7.10

G-Case / Test Data Case (GenRocket)

Changes components inside a scenario so one scenario gives many permutations for different test cases. Section 3.11

Generator (GenRocket)

The recipe that decides what value an attribute gets, such as a name from a list or a counter. Generators only generate; receivers format. Section 6.18

Generator parameter (GenRocket)

A setting on a generator, such as `startRange`. Required ones carry an asterisk. Section 6.13

Generator tag (GenRocket)

A rule: if an attribute name matches this text, assign this generator. Set per project or organisation. Section 6.18

Generator Tuning Wizard (GenRocket)

Step 3 of the XTS wizard: adjusts the generators behind each attribute across a whole project version, with statistics on how well they match. Section 6.18.3

GenericSQLInsertReceiver (GenRocket)

Inserts generated rows straight into a table through JDBC; a truncate option can empty the table. Section 6.24

GenRocket Runtime (GenRocket)

Java programs on your own machine that read a downloaded scenario and generate data inside your firewall. Section 6.6

G-Portal (GenRocket)

A self-service platform where requesters search for, request and download test data. Section 3.11

G-Repository (GenRocket)

The component that keeps scenarios and configuration files in sync across a company's machines. Section 6.5

.grs file (GenRocket)

The downloaded, encrypted scenario file you run with `genrocket -r`. Section 6.22

Import from DDL / XTS (GenRocket)

Two ways to create many domains from a schema: paste CREATE TABLE text, or extract structure from a live database. Section 6.14

Linked generators (GenRocket)

Several generators in one attribute, run left to right; later ones can use earlier values. Section 6.18

ListGen (GenRocket)

Picks from a fixed list of values, in order or at random. Section 7.5

Loop count (GenRocket)

How many rows a domain generates. A child's count is per parent row. Section 6.25

Masking receivers (GenRocket)

Receivers that take an existing file and write a new one with targeted values replaced. Section 6.24

Master project (GenRocket)

A project option for sharing master domains with other projects. Section 6.8

Model, Design, Deploy, Manage (GenRocket)

The four work phases of GenRocket's methodology: import the model, design the data, generate and deliver it, manage versions. Section 6.4

Offline certificate (GenRocket)

A file letting the runtime run scenarios without internet; organisation-specific and valid 24 hours. Section 6.6

Organization resource (GenRocket)

A named folder setting, such as `resource.output.directory`, each user fills with their own path. Section 6.23

Output format (GenRocket)

The shape of the result: delimited, fixed-width, JSON, XML, Excel, SQL statements and many more. Section 6.22

Parent, child and sibling domains (GenRocket)

A child has exactly one parent whose key it uses; a "describing" lookup table is marked a sibling. Section 6.15

PermutationGen (GenRocket)

Generates all possible combinations of the values it is given. Section 7.6

Preview (GenRocket)

Sample generated values shown in the web platform so you can check before downloading. Section 6.13

Primary attribute (GenRocket)

The attribute marked as the key, which children refer to. Section 6.13

Primary domain and root domain (GenRocket)

The primary domain is the one a scenario is built for; the root is the first parent at the top of the chain. Section 6.22

Project and project version (GenRocket)

A project holds the data model for one application or database; a version is one release's copy of it. Section 6.8

Project type (GenRocket)

Standard (default), Micro (domain-limited, such as files) or IPM (in-place masking of databases). Section 6.8

QueryOnceGen (GenRocket)

A query generator that reads real data from a database once, for example the current maximum id. Section 6.18

Quick Scenario (GenRocket)

Builds a scenario named after a domain, adding its parents and siblings for valid keys. Section 6.25

RangeGen / RandomGen (GenRocket)

Number generators: RangeGen counts up (or down); RandomGen picks a random number in a range. Section 6.18

Receiver (GenRocket)

The component that turns raw generated values into a usable output: file, SQL, message and more. Section 6.23

Receiver parameter (GenRocket)

A setting for the whole receiver, such as where to write; a star marks required ones. Section 6.22

Reference parameter (GenRocket)

A parameter value pointing at another attribute or generator, so one value is built from others. Section 7.14

ReferenceGen / ProxyReferenceGen (GenRocket)

The foreign-key pair: the proxy marks "a parent key goes here" on the domain; the real ReferenceGen replaces it in a scenario. Section 6.18

Scenario chain (GenRocket)

Linked scenarios run in order by one command; a chain set links chains. Section 6.25

Scenario domain (GenRocket)

A scenario's copy of a template domain; template changes flow down unless locked. Section 6.13

Scenario (GenRocket)

A saved instruction sheet: which domains to generate, how many rows, in what order. Running it makes the data. Section 6.25

Scratch Pad (GenRocket)

A way to create a domain from a domain name and a list of attribute names, optionally with type and size; generators are assigned automatically. Section 6.14.3

Seed (GenRocket)

A starting number for random generators; the same seed gives the same "random" data. Every domain has a seed variable. Section 6.18

Segment-and-merge receivers (GenRocket)

Pairs that build nested JSON or XML: each domain writes segments, a merge receiver combines them. Section 6.24

SimplePatternGen (GenRocket)

The fallback generator making values like name1, name2. Treat it as a to-do marker.
Section 6.18

SQLFileInsertReceiver (GenRocket)

Writes a file of SQL insert statements without touching a database; a safe first choice.
Section 6.24

Stateful scenario (GenRocket)

A scenario that remembers generator positions between runs, so a counter continues.
Section 6.25

Synthetic Data Replacement (GenRocket)

GenRocket's masking method: replacing sensitive values with synthetic ones. Section 6.24

Template domain (GenRocket)

A domain as defined in a project version, the master copy scenarios copy from. Section 6.14

Test Data Automation (GenRocket)

GenRocket's phrase for defining test data once as components and generating it automatically, whenever a test needs it. Section 6.4

Test data case suite (GenRocket)

In the 2022 demo, a suite of unit, negative, load and integration cases built by a self-service module, with volume set per domain and rules controlling the output. Section 6.4.2

Test Data Engineer (GenRocket)

A trained person who models test data projects and designs the data for others. Section 3.9

Test data project (GenRocket)

The data model for one application or database (domains, attributes, generators, receivers, scenarios) with no customer data; also the unit of licensing. Section 6.8

Test data rules, G-Rules (GenRocket)

Condition-and-action rules attached to cases rather than single attributes, suited to complex if/then/else logic. Section 6.18.5

Test plan (GenRocket)

A set of scenarios run with one command, as when a Jenkins job runs GenRocket against a test plan. Section 6.25.10

User profile and server profile (GenRocket)

A .grp file in the .genrocket folder that ties a person, or a CI/CD server, to a licence. Section 6.6

Visible (GenRocket)

An attribute setting; if false, receivers do not see it, so it works as a helper. Section 7.14

Web platform (GenRocket)

GenRocket's cloud web application, where you model and design. Also called GenRocket Cloud. Section 6.1

WeightGen / MultiWeightGen (GenRocket)

Percentage-based choice: WeightGen splits between two values, MultiWeightGen among any number. Section 7.5

XTS (GenRocket)

Extract Table Schema: a runtime option that writes a live database's structure, not data, to an encrypted file. Section 6.14

10.30 Delphix

Algorithm (Delphix)

The recipe that replaces a real value with a safe one. Profiling only chooses it; the masking job runs it. Section 8.39

Algorithm framework (Delphix)

A kind of algorithm you configure into instances, such as Secure Lookup or Segment Mapping. Section 9.1

Algorithm instance (Delphix)

A named framework plus settings, such as `dlpx-core:FirstName`. This is what you assign to a column. Section 9.1

Algorithm key (Delphix)

A secret value inside most algorithms; changing it changes every output. Section 9.1

Algorithm syncability (Delphix)

Copying algorithms between engines so both mask alike. Non-deterministic ones cannot be synced. Section 8.35

Application (Delphix)

A tag naming the system whose data you manage; create one before an environment. Section 8.16

ASDD profiler (Delphix)

Automated Sensitive Data Discovery, the current profiler, using classifiers instead of the legacy search expressions. Section 8.39

ASDD Standard (Delphix)

The recommended built-in profile set, with path, type, regex and list classifiers. Section 8.43

Assignment threshold (Delphix)

The minimum confidence before the profiler writes a domain and algorithm into the inventory. Section 8.39

Async Task (Delphix)

A background task whose ID you can search under Monitor, Async Tasks. Section 8.4

Character Mapping (Delphix)

A keyed one-to-one swap of characters within groups you define; unlisted characters stay. Suits key columns. Section 9.4

Checklist (Delphix)

An action checking that every table in a rule set has a primary key, an indexed identity column or a logical key. Section 8.34

Classifier (Delphix)

One recognition rule tied to one domain, such as a column name that looks like ssn. Section 8.39

Classifier framework (Delphix)

A kind of classifier: PATH (table and column name), TYPE (data type and length), REGEX (data) or LIST (values in a list). Section 2.7

Collision (Delphix)

Two different real values masked to the same fake value. Section 8.7

Confidence (Delphix)

A 0 to 100 score for how sure the profiler is about a match. Section 8.39

Connector (Delphix)

Saved details the engine uses to reach one database or set of files. Section 8.17

Containerized engine (Delphix)

The engine run as a Kubernetes pod of three containers; meant for ephemeral use, and without FTP file masking. Section 8.8

Continuous Compliance engine (Delphix)

The product that discovers sensitive data and masks it; also called the masking engine or Compliance Engine. Section 8.1

Continuous Data engine (Delphix)

The virtualization product that creates VDBs; never run it in the same machine as the masking engine. Section 8.23

Copybook (Delphix)

A rule set type for mainframe data sets described by a copybook. Section 8.30

Date Shift (Delphix)

Moves a date within a range, never back onto itself. Fixed shifts are deterministic; variable ones are not syncable. Section 9.9

Dependent Date Shift (Delphix)

Masks two related dates together, keeping their order and interval. Section 9.4

Domain (Delphix)

A named kind of sensitive data, such as FIRST_NAME, with exactly one default masking algorithm. Section 8.1

dSource (Delphix)

The engine's compressed copy of a source database from which VDBs are made. Section 8.23

Encoding (Delphix)

How characters become bytes. ASCII and UTF-8 for most file types; mainframe data must be EBCDIC. Section 9.17

End-of-record (Delphix)

The characters ending a line: LF on Linux, CR+LF on Windows. Section 9.15

Environment (Delphix)

A folder-like container inside an application holding connectors, rule sets and jobs. Its purpose is Mask or Tokenize/Re-Identify. Section 8.16

Extended connector (Delphix)

A connector on an uploaded JDBC driver, for databases not supported natively. Section 8.15

File connector and file rule set (Delphix)

The connector says where files are; the rule set lists which files or name patterns and which format each uses. Section 9.15

File format (Delphix)

A small uploaded description of a file's field names (and lengths for fixed width). Metadata only. Section 9.19

Free Text Redaction (Delphix)

Blanks sensitive words and patterns inside notes columns; needs expertise to configure. Section 2.32

Identity column (Delphix)

A column that numbers rows automatically; the engine can use one as the key or add a temporary one. Section 8.27

In-place / on-the-fly (Delphix)

In-place changes data where it sits; on-the-fly reads a source and writes masked data to a target. Section 9.11

Inventory (Delphix)

The table of every column or field in a rule set with its domain and algorithm: profiler output, masking input. Section 8.39

JDBC (Delphix)

Java Database Connectivity. The engine uses a JDBC driver per database and a JDBC URL to say where to connect. Section 8.15

Job (Delphix)

A saved runnable task: profiling, masking, tokenization or re-identification. Section 8.1

Logical key (Delphix)

A unique, non-null column (or columns) you tell Delphix to treat as the row identifier; the database is unchanged. Section 8.33

Lookup file (Delphix)

A text file with one replacement value per line; the algorithm picks a line from the input. Section 9.1

Mapping (Delphix)

You supply replacement values; each input gets a distinct output, stored encrypted. Needs enough values. Section 9.9

Masked VDB (Delphix)

A VDB whose data was masked by a masking job during provisioning; it cannot be unmasked. Section 8.23

Masking engine (Delphix)

The Continuous Compliance engine, where connectors, rule sets, profiling jobs and masking jobs are defined. Section 9.30

Masking job (Delphix)

A saved set of options to apply a rule set's inventory, with a method, threads, memory and scripts. Section 9.1

Masking user (Delphix)

A dedicated database account for the connector, with only the permissions masking needs. Section 8.15

Match strength (Delphix)

A weight on one classifier match, such as 0.7 for a dashed SSN; several weak clues combine. Section 2.7

Meta Data Source (Delphix)

The connector screen grouping: Database, File or Mainframe. Section 8.15

Min Max (Delphix)

Pulls extreme values toward the middle, since the top salary may reveal a person. Section 9.4

Multi-Tenant (Delphix)

A job option letting one rule set serve identical schemas through different connectors chosen at run time. Section 8.23

Non-conformant data (Delphix)

A value not fitting the expected format. By default it stays unmasked and only a warning is recorded, so check job warnings. Section 8.7

Out-of-the-box instance (Delphix)

An instance shipping with the engine; it cannot be edited or deleted. Section 9.1

Payment Card (Delphix)

Keeps the first digits and the card's Luhn validity, masking the rest. Section 9.4

Profile set (Delphix)

A bundle of classifiers; choosing one decides which kinds of sensitive data a job can detect. Section 8.39

Profiling (Delphix)

Scanning column names and optionally data to find sensitive columns. Also called sensitive data discovery. Section 8.39

Profiling job (Delphix)

A saved instruction to scan a rule set using a profile set. Section 8.39

Record type and qualifier (Delphix)

When one file mixes layouts, a record type names each layout and up to three qualifiers recognise it. Section 9.20

Role (Delphix)

A set of allowed actions on object kinds; each user has one role and a list of environments. Section 8.9

ROWID (Delphix)

An Oracle pseudocolumn holding each row's address; recommended as the logical key for speed. Section 8.27

Rule set (Delphix)

The list of tables or files of one connector that jobs may look at. Four types: Database, Semi-Structured, Flat Files, Copybook. Section 8.30

SDD (Selective Data Distribution) (Delphix)

Replicating a masked VDB to another engine in non-production without sending unmasked data. Section 8.15

Secure Lookup (Delphix)

Picks a replacement from a lookup file by hashing the input, so output is consistent. Collisions are possible. Section 8.7

SecureShuffle (Delphix)

Moves each value to a different row; the values stay real and the result is not syncable. Section 2.32

Segment Mapping (Delphix)

Masks short values by segments with no collisions, if the allowed set is big enough; suits SSNs and keys. Section 8.7

Snapshot and TimeFlow (Delphix)

A snapshot is a dataset's state at one moment; the TimeFlow is the timeline of those states. Section 8.23

Stream (Delphix)

One parallel lane of work; more streams let the engine process several tables at once.
Section 9.1

String Algorithm Chain (Delphix)

Runs two to eight string algorithms one after another. Section 2.32

Text enclosure and escape (Delphix)

The enclosure (often a double quote) wraps a value that may contain the delimiter; the escape writes a literal enclosure. Section 9.15

Tokenization (Delphix)

The only reversible family, using AES-128 and Base64; output is longer than input.
Reversing it is re-identification. Section 9.9

Update threads (Delphix)

Parallel threads writing to the target database; too many can deadlock and each table needs an index. Section 9.11

VDB (virtual database) (Delphix)

A full read-write copy of source data provisioned from a dSource or another VDB, created in minutes. Section 8.23

Virtual appliance (Delphix)

A ready-made machine image of the engine imported into a hypervisor or cloud.
Section 8.8

Whole-file masking (Delphix)

Sending a file to one algorithm in one piece, using a format with one field of length 0.
Section 9.26

Definitions are drawn from the guide's own sections; see each linked section for its sources.