

Leaving PowerCenter

Migrating an on-premises Informatica estate to IDMC or cloud-native services — and where AI-assisted engineering genuinely earns its place

Published August 2026

Reading time 22 minutes

Audience Data platform leads, enterprise architects, finance partners

CONTENTS

Executive summary

1. The clock, and what is actually known about it

2. Two Informatica products, constantly confused

3. Choosing the target

4. The assessment is the whole product

5. What no converter does for you

6. The numbers you will be quoted, and how to read them

7. What re-platforms separately, and why programmes miss it

8. Where AI-assisted engineering genuinely earns its place

9. Proving the numbers match

10. Sequencing the programme

11. A readiness checklist

12. Summary

Executive summary

PowerCenter 10.5.x left standard support on 31 March 2026. There is no 10.6 and no 11.x. Every organisation still running an on-premises Informatica estate is now on a clock it does not control, and most of them are being quoted an automation percentage by somebody who has not seen their repository.

This paper is written for the people who have to make the decision defensible: the data platform lead, the enterprise architect, and the finance partner who will be asked why the number moved. It sets out how a PowerCenter exit actually works — not how it is marketed.

Five things it argues, and the rest of the paper is the evidence:

1. **The product decision comes before the migration decision.** Informatica sells two different things that are constantly confused. One re-hosts your estate and stops the clock. The other converts it into native cloud assets. Choosing the first while believing you bought the second is the most expensive mistake available in this programme.
2. **The assessment is the only estate-specific number that exists.** Every percentage quoted before a repository has been read is a marketing number. The vendor's own tooling will tell you which of your assets it cannot convert — get that output before you sign anything.
3. **The residue is the project.** Automated conversion handles the standard transformation shapes. What it does not handle is documented, consistent across converters, and almost always underestimated: SQL overrides, parameter files, workflow commands, scheduler dependencies, and a specific list of transformation types with no cloud equivalent.
4. **AI-assisted engineering is the largest genuine acceleration available on this work — in the reading and the drafting, not in the deciding.** Pointed at an undocumented estate it converts an unknowable problem into a costed plan. Pointed at conversion it drafts faster than a human can. It does neither safely without a review gate and a value-level reconcile, and the honest claim is that it compresses drafting while *increasing* review effort.
5. **A converted pipeline that runs is not a converted pipeline that is correct.** The failure that matters is not the one that errors in week two. It is the one that silently produces different values and surfaces at a quarter-end close four months later. Row counts and checksums do not catch it. A five-stage reconcile gate does.

1. The clock, and what is actually known about it

Date	Event	Confidence
18 November 2025	Salesforce completed its acquisition of Informatica	Published
31 March 2026	PowerCenter 10.5.x left standard support — all 10.5.x.x minor versions, patch level irrelevant	Published

Date	Event	Confidence
March 2027	Paid extended support reported to end	Verify — see below
2029	Sustaining support reported to end; no guaranteed fixes for critical defects	Verify — see below

A note on the last two rows, because it matters more than it looks.

Informatica's public lifecycle page is access-gated. The extended and sustaining dates circulate widely in partner and practitioner material, and they are consistent, but they are secondary-sourced. **Confirm them from your own support portal before either appears in a budget paper.** A precise-looking date that turns out to be wrong is worse than a range that is honest, and any vendor putting those dates in a proposal without that caveat is telling you something about how they handle facts.

A useful precedent: 10.4 left standard support on 31 March 2024, on the same pattern. The runway Informatica gives a version is not a surprise, and it is not negotiable at the end of it.

What the acquisition changes. The direction reported through 2026 is a decomposition of IDMC into headless services callable from Salesforce's agent platform. There is no published commitment on Cloud Data Integration pricing or packaging under the new ownership. That absence is itself material if you are budgeting a multi-year renewal: you are being asked to commit to a pricing model whose owner has not committed to it.

The second clock, which runs faster. Support dates are the visible deadline. The one that actually determines whether your migration is deliverable is the labour market. PowerCenter expertise is scarce, expensive and thinning every year, and the people who built your estate have mostly already left. Every year you wait, the pool of people who can read what you own gets smaller and the cost of reading it goes up.

2. Two Informatica products, constantly confused

Getting this distinction right in the first meeting is the single highest-value thing on this page. Both are sold under "cloud modernization". They do entirely different

things.

CDI-PC — Cloud Data Integration for PowerCenter		PC2CDI asset conversion
What it does	Re-hosts the PowerCenter binaries. The domain registers to IDMC and workflows run as-is under a Secure Agent	Converts repository metadata into native CDI assets — mappings, mapplets, mapping tasks, taskflows
What you end up with	The same PowerCenter mappings, running, in a managed place	A real IDMC estate you can develop against
What it is for	Buying time past end of support with no rewrite	Actual modernization
The trap	Objects you largely cannot edit in their new home	Genuine conversion effort, honestly scoped

The conversion engine lives *inside* CDI-PC, as an agent application. The mechanical flow is: register the domain → repoint configuration → **assess the repository** → download the output → configure connection mapping and conversion properties → convert → bulk metadata update.

Why this decides the budget. An organisation that takes the re-host path has PowerCenter objects running in the cloud that its own engineers largely cannot change. It has paid, the support clock has stopped, and nothing has modernized. Twelve months later it is having the same conversation with a smaller runway and a larger bill. Re-hosting is a legitimate choice — as a deliberate bridge, with a dated plan behind it. It is a catastrophic choice made by accident.

On product names. "CMT" and "Cloud Modernization Tool" are not current Informatica product names. The current ones are CDI-PC, PC2CDI modernization, and Cloud Data Validation. If a vendor is using the retired names in a proposal, they are working from material several years old.

3. Choosing the target

There are three destinations, and the assessment should recommend between them rather than assert one. Any partner who arrives already knowing the answer is selling their bench, not reading your estate.

IDMC (Cloud Data Integration)

The shorter path. The conversion utility exists, the transformation vocabulary is broadly familiar to your team, and the operational model is close enough that people can be redeployed rather than rehired.

Choose it when the estate is predominantly classic batch ETL, when the team's Informatica skills are an asset worth preserving, and when the licensing relationship is one you want to keep.

Understand what you are accepting: consumption-based metering, a pricing model whose new owner has published no commitment, and the same vendor concentration you have now.

Cloud-native integration services

A genuine rebuild, and usually far cheaper to run. The estate is re-implemented on the cloud platform's own integration primitives — managed workflow orchestration, a message bus, an API gateway, a secret store, a B2B/EDI service where trading-partner traffic is involved.

Choose it when the estate carries significant non-ETL surface (see §7), when the run cost of per-core licensing has become the problem, or when the strategic direction is away from vendor concentration.

Understand what you are accepting: it is a rebuild, not a conversion, and the effort model is different in kind. The compensation is that the replacement usually funds development and test environments the licence never covered, and that the skills required are mainstream and hireable rather than niche and shrinking.

A split

Usually the right answer for a large estate, and rarely the one proposed, because it is harder to sell and harder to plan. Batch ETL to IDMC where the conversion utility earns its keep; trading-partner, file-transfer and event-driven surface to cloud-native services where it belongs.

The reason it is right is that the two halves of a typical estate have genuinely different economics, and forcing them onto one destination means overpaying for one of them.

The comparison that should be in the paper

Dimension	On-premises, licensed per core	Metered cloud platform
Cost model	Fixed licence on a multi-year commitment. You pay for peak capacity every hour, used or not	Metered plans plus consumption. Resize or retire within a month
Capacity	The licensed core count is a hard ceiling. More throughput means a purchase order and a contract amendment	Scale up or out in minutes; give a heavy application its own plan when it earns one
Upgrades	Version upgrades are projects: compatibility matrices, server rebuilds, regression, downtime windows	The runtime is patched underneath you. There are no version-upgrade projects to fund
Environments	Production plus a licence-limited development lab. Test competes for the same cores	Full development, test and production parity. A new environment costs a plan and a storage account, not a licence
Release management	Repository export and import, hand-carried between environments. Drift is hard to see and harder to prove	Source control and pipelines with environment-parameterised settings — repeatable, reviewable, diffable against what is running
Resilience	Single-site servers, manual failover, recovery by restore from backup	Redundant compute and storage, a platform SLA, and per-execution run history for replay and evidence
Observability	Session logs read after the fact. No live metrics, no alert routing	Platform metrics and log analytics with alert rules — every action of every run inspectable
The people who run it	A skillset that is scarce, expensive and thinning	Mainstream skills your existing team can be brought onto

4. The assessment is the whole product

Never quote a migration price for an estate nobody has inventoried. Any fixed fee offered before a repository has been read is either padded to cover the unknown or is about to become a change-order conversation.

Register the PowerCenter domain to the modernization tooling and run a full repository assessment. It produces a machine-generated inventory that buckets every asset:

- **Automated** — the utility expects to convert it without help.

- **Partial** — it will convert, with human completion.
- **Manual** — **Informatica's own tooling declaring there is no direct cloud support and the asset must be rebuilt by hand.**

That third bucket is the honest core of your estimate, and it comes from the vendor rather than from a consultancy with an incentive.

The output files that decide the number

The assessment output is a ZIP. Most of it is inventory. These are the files that change a price:

File	What it tells you
MANUAL_FLAGS.csv, PARTIAL_FLAGS.csv	The actual work. Score flag frequency, not asset count — one mapping carrying nine manual flags is not one unit of work
Command_List.xlsx	Workflow commands: <code>pmcmd</code> invocations, shell scripts, external scheduler hooks. The most under-estimated line item in every migration
Parameter_Files.xlsx	Parameterisation, which is its own conversion task type with its own path requirements
Sql_List.csv, the SQL/ELT summary	SQL overrides. See §5 — these are not auto-translated
Connection_Summary_Report.xlsx	Supported and unsupported connections. Read the second half first
File_List.xlsx, XML pre-processing configuration	Flat-file and XML surface area

Sizing the assessment itself

The agent application wants roughly 5–6 GB of heap for a ~9 GB repository — about a 600 MB XML export. One Secure Agent handles one assessment plus two concurrent conversion jobs. Ask for the repository size before anyone promises a timeline; this is where "the assessment takes a week" quietly becomes three.

What an assessment should hand you

Deliverable	Accepted when
Estate inventory	It accounts for 100% of objects in the repository export, classified into agreed complexity bands

Deliverable	Accepted when
Complexity and risk register	Every <code>Manual</code> and <code>Partial</code> flag is explained and costed
Hidden-work schedule	SQL overrides, parameter files, workflow commands, scheduler dependencies, connections with no cloud equivalent
Target architecture recommendation	Options with trade-offs stated, not a single answer asserted
Wave plan	Waves grouped by folder and connection, sequenced by risk
Effort model and price	Per wave, with the assumptions that would change it stated as falsifiable
Parity approach	The specific reconcile gate for this estate — \$8

5. What no converter does for you

These are documented limits rather than opinions, and they are consistent across vendors. This list is the difference between an estimate and a guess.

- **SQL overrides are not auto-translated.** SQL is extracted to spreadsheets and a human types the target dialect into a converted-SQL column. It is invisible in an asset count and it is frequently one of the largest single line items in the programme. Budget it explicitly.
- **Parameter files are their own conversion task**, with their own input mapping and their own path requirements on the Secure Agent.
- **Advanced conversion properties are undocumented.** The vendor's own guidance is to contact support for the supported list. Plan on a support case, and plan on it taking a week.
- **Transformation types with no cloud equivalent.** Cloud Data Integration has no Custom transformation, no Source Qualifier, no XML Parser/Generator and no External Procedure. XML and hierarchy work moves to Hierarchy Parser, Builder or Processor, to Structure Parser, or to templating. Java, Normalizer, Transaction Control, Sequence, SQL, Rank, Router and Union do exist.
- **Unconnected lookups are supported** via the `:LKP` syntax — this is a widely repeated myth worth correcting in a design review. What is true is that they cannot be called from a Joiner or a Java transformation.

- **The legacy PowerCenter-task fallback is far more constrained than people assume:** one workflow, one session, one mapping, no non-session task types, and **no reusable transformations or shortcuts**, because there is no repository in the cloud product.
- **Known utility defects worth testing for early:** indirect flat-file connections not becoming file lists; connected flat-file lookups losing the lookup object and its connection; mapplets with source or target connections failing to publish.

The independent picture agrees

Where a completely different converter has been examined by practitioners — an LLM-and-rules engine rather than the vendor's — the residue is **the same residue**:

Converts reliably	Unreliable or unsupported
Aggregator, Filter, Joiner, Sorter	Java transformation — frequently not converted at all
Straightforward expression logic	Custom transformation — no target equivalent
Mapplets to reusable functions	Transaction Control
Structural mapping shape	SQL transformation and stored-procedure calls
	Sequence Generator — mis-implemented; use the platform-native identity construct
	Lookups translated to a different SQL dialect — often incorrect, needing real refactoring
	Source Qualifier SQL — copied verbatim, dialect untranslated
	Hardcoded file paths and parameter syntax

That the two lists agree is the important finding. It means the residue is a property of PowerCenter's semantics rather than of any one converter's quality, and no procurement decision between converters will make it go away. Everything in the right-hand column is a **review trigger**: an asset touching any of it does not merge on a green build alone.

6. The numbers you will be quoted, and how to read them

This is the most useful table in the paper, and none of it is ours.

Source	Claim	What kind of number it is
Informatica marketing	"Reuse up to 100% of PowerCenter assets", "up to 8X faster ", cost "reduced by up to 50% "	Vendor marketing
Informatica case study	99% of assets migrated in six months with zero defects	Vendor case study, best case
Informatica case study	"up to 70% of applications" migrated	Vendor case study
Informatica blog and partner material	"over 90% of existing mappings, mapplets, sessions and workflows" migrate automatically	Vendor marketing
Practitioner reports of the PC2Cloud utility	60% converted; the remaining 40% requires redevelopment or reimplementation	Practitioner report
A comparable AI conversion product (SAS to PySpark — the same playbook)	timelines cut "up to 80%", "5X faster", " 70% out-of-the-box accuracy "	Vendor
Another services vendor's AI conversion offering	timelines reduced 30–40%	Vendor
A major analyst house, on generative AI in application modernization	timelines 40–50% faster; technical-debt cost down ~40%	Analyst estimate

Three observations a buyer can act on:

1. **The spread is 60% to 100%, and it is not a disagreement about facts.** It is a disagreement about what is being counted — assets, mappings, "applications", or effort — and over what estate. A number without a denominator is not a number.
2. **Vendor best-case and practitioner median are roughly 40 points apart.**
Plan against the practitioner figure; be pleasantly surprised by the vendor one.
3. **The most honest comparison available is a vendor against itself.** One converter vendor advertises "up to 95%" automation while its own published case study reports **78%** on a real Informatica estate. That single comparison makes the whole point without anyone needing to argue about methodology.

The only number that matters is yours, and the assessment produces it.

Anyone quoting you an automation percentage before reading your repository is quoting you a marketing figure — and that includes anyone who quotes you one after reading this paper.

7. What re-platforms separately, and why programmes miss it

The modernization tooling covers the classic Repository and Integration Services. It does **not** cover B2B Data Exchange, Managed File Transfer, MDM, ILM or Data Quality estates. Those re-platform on their own path, with their own design decisions.

This is the single most common scoping failure in an Informatica exit, because those components are invisible in a mapping count and highly visible at cutover. If your estate carries any of the following, it needs its own workstream and its own line in the budget:

What you are retiring	What it actually did	What replaces it	The part that bites
B2B / EDI translation	X12 translation, document routing, endpoint profiles, transaction logging	A cloud integration-account service with X12 encode/decode, plus a shared translation library for maps, flat-file conversion and control-number issuing	Control numbers. They are stateful, partners reject out-of-sequence documents, and the issuing logic is usually undocumented inside the translator being retired
Managed file transfer	AS2, SFTP and FTPS partner transport, PGP, keystores, MDN handling, landing folders	Partner and agreement definitions behind an API gateway, with certificates in a managed secret store and rotation as a platform operation	Every partner has to re-point or re-agree at the network edge. That is a scheduling problem across dozens of external organisations, and it is the long pole on the plan
Synchronous web services	SOAP services exposed to line-of-business applications calling integration logic directly	An API gateway fronting a dispatcher that routes to the same backends	Preserving the contract exactly. Keep the operation names and the consuming systems need no change; get it wrong and the migration spreads into every application that calls it

What you are retiring	What it actually did	What replaces it	The part that bites
Change data capture	Licensed CDC out of CRM and midrange databases, feeding master data downstream	Native change events onto a message bus, with session-ordered consumers writing downstream	Ordering. The old CDC guaranteed it per source; a topic only guarantees it per session, so the session key has to be chosen per entity and got right first time
Scheduler, repository, log analyzer	Job scheduling and dependencies, export/import promotion, session logs read after the fact	Platform recurrence and event triggers, CI/CD with environment-parameterised settings, platform monitoring with alert rules and per-run history	Promotion by repository export is how environments drift. Replacing it with pipelines is the change that makes everything after it repeatable

Note the fourth column. It is where the programmes go wrong, and none of it appears in an asset inventory.

Also out of scope unless explicitly written in: source or target system changes, schema redesign, data-quality remediation, infrastructure and identity work in your own tenant, and licence procurement. And one more, which deserves its own sentence:

A migration that "fixes things while we are in there" is not a migration. It destroys the one thing that makes correctness provable: a known-good baseline. Fix it afterwards, on purpose, with the parity evidence already banked.

8. Where AI-assisted engineering genuinely earns its place

This is the part of the programme where the last three years have actually changed what is possible. It is also where the marketing is worst, so it is worth being precise about which claims survive contact with an estate.

The position that holds up:

AI reads the estate and drafts the conversion. A human reviews every asset. The reconcile gate decides.

Three properties follow, and together they are the whole argument:

- **Speed comes from the drafting**, which is the part with a fast feedback loop.
- **Safety comes from the gate**, which is mechanical and does not get tired.
- **Nobody has to trust the model** — not the client, and not the engineers.

8.1 Reading the estate: the unambiguous win

This is where the acceleration is real, and it is where even the sceptics agree it is real.

A twenty-year-old PowerCenter estate is an unknowable object. The documentation is stale where it exists. The people who made the decisions have gone. Somewhere in it are mapplets nobody has opened since 2014, a Router with eleven branches, and a shell script invoked from a workflow command that turns out to be load-bearing.

Pointed at the exported metadata, AI tooling produces, at volume and in hours rather than months:

- a dependency graph across mappings, mapplets, sessions, workflows and parameter files;
- a difficulty classification per object, cross-referenced against the vendor's own flags;
- **documentation for an estate that has not been documented since the people who built it left;**
- an inventory of the hidden surface — SQL overrides, workflow commands, hardcoded paths, scheduler hooks — with each occurrence located rather than estimated.

The burden being removed here is processing volume, not judgement. That is exactly why this application is uncontroversial: reading ten thousand XML objects consistently is a task humans do badly and expensively, and a machine does well. It is also the step that converts an unknowable problem into a costed plan, which is what makes the rest of the programme financeable.

At the scale this matters — thousands of translation documents, hundreds of partner configurations, a repository export measured in hundreds of megabytes — reading the estate by hand is not slow. It is not done at all, which is why so many programmes are scoped on a mapping count and a hope.

8.2 Drafting the conversion: real, and conditional

AI drafts the residue the utility will not handle: the transformation types with no direct equivalent, the SQL overrides needing dialect translation, the error-handling logic that was reimplemented rather than converted.

It is good at this in the way it is good at any translation task with abundant examples and a tight feedback loop. It is also, and unavoidably, wrong sometimes — and the way it is wrong is the thing to design against.

8.3 The four failure modes, and only one of them matters

Failure mode	Caught by	How dangerous
Output unrelated to the input	Anyone glancing at it	Harmless. Obvious, immediate
Lifting, compilation and runtime errors	The build	Harmless. That is what a build is for
Silently wrong output — it compiles, it runs, it produces different values	Only a value-level reconcile	This is the only class that matters
False failures — a correct translation rejected by a bad harness	Investigation, eventually	Dangerous differently: it teaches the team to distrust the gate, and a gate people override is worse than no gate

The first two are why a green build is not sufficient evidence. The third is why the reconcile gate in §9 exists at all. The fourth is why the harness must be maintained as carefully as the code it judges.

8.4 The review gate

No machine-converted asset merges without all three. Published human-in-the-loop guidance for ETL conversion is thin, which is itself a reason to write the standard down and hold to it.

1. **A reviewer diff against the source PowerCenter XML.** Not a read of the output — a *diff against the source*. The question is "does this do what the original did", and it cannot be answered by reading the new code alone.
2. **A green reconcile run**, at least through the keyed value diff (§9, stage 3).
3. **Explicit sign-off on any transformation on the unreliable list** in §5, named in the commit, so the record exists when somebody asks in month nine.

And one sequencing rule that teams skip when a deadline approaches: **sample before scaling**. Convert a representative subset, validate it end to end, understand the failure pattern for *this* estate, and only then run at volume. Every converter vendor's own guidance says some version of this.

8.5 What it does to the estimate — the honest version

AI changes the shape of the work rather than the size of the risk:

- The **Automated** bucket gets faster. That is where the leverage is.
- The **Partial** bucket gets somewhat faster, with more variance.
- The **Manual** bucket does not benefit and is **priced as new build**. It is new build.
- **Review effort goes up, not down**, because every asset now needs a diff against source.
- The parity harness is a fixed cost that amortises across waves.

AI compresses the drafting — and the reviewing is what makes the drafting safe. A firm claiming otherwise is describing a risk transfer, not a saving.

That sentence is the test to apply to any AI-accelerated migration proposal you are handed. If the saving is claimed to come out of the review, ask who is reading the converted code. The answer is either "a named engineer" or "nobody".

8.6 What should never appear in a proposal

- **An automation percentage for an estate nobody has assessed.** Including its disguises: a consolidation ratio ("nine thousand objects became four hundred"), an "N:1", an "8X".
 - **A named model or version.** The practice does not depend on which model is current, and naming one dates the material within months.
 - **A claim that AI replaces the engineer.** It does not, and a firm that says otherwise will hand you converted code nobody has read.
-

9. Proving the numbers match

Everything above is available somewhere at some price. *Row-level evidence that the replacement produces the same output as the system it retired* mostly is not — and it is the only thing that makes an AI-assisted conversion defensible.

The premise

A converted pipeline that runs is not a converted pipeline that is correct. The failure that matters is not the one that throws an error in week two. It is the one that **silently produces different values and surfaces at a quarter-end close, four months later, when nobody can tell whether the number was ever right.**

So the gate is not "did it run". It is **"does it produce the same rows"**.

Row counts and checksums are not a gate

They are the first stage of one. Aggregate matches do not guarantee row-level accuracy, and they hide precisely the cases that hurt:

- **Nulls** — empty string versus null, null handling in aggregates, null-safe joins.
- **Type coercion and precision** — decimal scale, floating point, implicit casts that round.
- **Ordering** — not a defect in a set, but it breaks naive comparison and wastes a day.
- **Date and time** — time zones, boundary values, date-versus-timestamp truncation.
- **Errors that cancel** — two rows wrong in opposite directions leave the sum correct.

A migration signed off on row counts has been tested for the failures nobody was worried about.

The five-stage gate

Each stage is cheap enough to run continuously, and each catches what the previous one cannot.

1. Schema diff. Column names, types, nullability, precision and scale. Fast, and it eliminates whole classes of defect before a single row is compared.

Expect false positives on wide-numeric, time and nested types — the same logical type is spelled differently on different platforms. **Catalogue the known-equivalent pairs once per target platform** and keep the list in the engagement repository. Re-litigating them every run is how a team learns to ignore the gate.

2. Row-hash diff. A hash per row, compared as a set. **No join key required**, which makes it the cheapest full-estate sweep available and the right default for a first pass over everything.

Normalise before hashing — trim, case, null representation, numeric scale — and **record the normalisation**, because it is a decision about what counts as "the same". Some source platforms need a custom hash function; budget for it.

3. Keyed value diff. Join on business keys and compare column values, reporting **per-column mismatch samples** rather than pass/fail. This is the stage that tells an engineer *where* the defect is instead of *that* one exists. Sample during development; **a full pass before sign-off, and the contract should say so.**

4. Edge-case suite. Written per estate, and **this is where the real defects live.** It is also the stage a converter vendor does not run, because it needs somebody who understands the business logic.

At minimum: nulls and empty strings · type casts and precision boundaries · date boundaries, month-ends and time zones · empty result sets · duplicate keys · the maximum and minimum values actually present in the data · every code path in a Router or a Decode · error and reject handling, which is frequently reimplemented rather than converted.

5. Parallel run. Old and new together, reconciling every cycle, for **N clean consecutive cycles** — with N agreed in the contract rather than negotiated at the end when everyone is tired.

Parallel running catches what static comparison cannot: scheduling and dependency behaviour, late-arriving data, incremental and CDC logic, and volume behaviour under real load.

Make the gate the acceptance criterion

The reconcile gate belongs in the contract, not in the method deck. A workable form:

Converted mapping set. Accepted when each mapping executes to completion in the Client-provided test environment against the agreed test dataset and produces row counts and checksums matching the legacy output within the tolerance in Exhibit C.

Exhibit C is where the tolerance lives, and it must be explicit. Zero is right for most things and wrong for floating-point aggregates. Agreeing it before the work starts is worth a great deal more than arguing about it during acceptance.

What the harness has to do

1. **Cross-platform.** The source is a legacy database and the target is a cloud platform. Both sides must be first-class.
2. **Reports rather than assertions.** A pass/fail is useless to the engineer fixing it. Emit per-column mismatch samples.
3. **Runs in CI**, so every wave reconciles automatically rather than when somebody remembers.
4. **Output the client can read.** The reconciliation evidence is a deliverable, and it is frequently the thing that gets an invoice approved.
5. **Deterministic and re-runnable**, so a fix can be proved rather than asserted.

10. Sequencing the programme

Wave one is a calibration purchase

The first wave should be deliberately small and deliberately *representative* — a mix of Automated, Partial and Manual assets rather than the easiest ones. Choosing the easy ones produces a velocity figure that is wrong in the direction that hurts.

Its real purpose is **calibrating the effort model**. Until one wave is done, any per-wave fixed fee is a borrowed number. After it, the estimate is measured — and only then is fixed-price contracting safe for either side.

Also stood up once in wave one and reused for the whole programme: the runtime environment and Secure Agent sizing, the connection map, the parameterisation approach, source control and deployment, and the parity harness.

Group waves by folder and connection, never by mapping count

Connection mapping is configured per conversion task, so waves that share endpoints are materially cheaper to run than waves that do not. **This is the highest-leverage planning decision in the whole programme and it is invisible on a Gantt chart.** A wave plan organised by "200 mappings each" is a plan built for a status report rather than for the work.

The wave loop

1. **Convert** — vendor tooling for what it handles, AI-assisted conversion for the residue, under the review gate in §8.4.
2. **Review** — a human diff against the source XML. Every asset, no exceptions.
3. **Reconcile** — the gate in §9. Schema, row hash, keyed value diff, edge cases.
4. **Promote** — through environments via source control, not by hand.
5. **Parallel-run** — old and new together, reconciling every cycle.

Cutover and decommission

Phased, with a documented rollback at every step. **Decommission only after N clean parallel cycles.** Nothing is switched off until its replacement has run clean alongside it — which is also the honest answer to why the old channel is still there while the new one settles.

Handover is a deliverable, not a courtesy: the runbook, the parity harness and how to run it, the residual-risk register, and **what was deliberately not migrated, and why.**

11. A readiness checklist

Before you go to market, you should be able to answer these. If you cannot, the assessment is the thing to buy first — from anyone.

The estate

- ☐ Can you produce a PowerCenter repository export, and do you know its size?
- ☐ Do you know how many workflow commands, shell scripts and external scheduler hooks exist?

- ☐ Do you know where your SQL overrides are, and in which dialect?
- ☐ Does your estate include B2B, MFT, MDM, ILM or DQ components? (§7 — separate workstreams)
- ☐ Which connections have no direct cloud equivalent?

The decision

- ☐ Are you buying a re-host or a conversion? Does everyone in the room agree? (§2)
- ☐ If a re-host, what is the dated plan for what happens after it?
- ☐ Have you compared run cost, not just migration cost?
- ☐ Have you priced staying — including extended support and the labour market?

The proof

- ☐ What is your acceptance criterion, in writing, and where does the tolerance live?
- ☐ How many clean parallel cycles before decommission, and who decides?
- ☐ Who reads the converted code? Name them.

The proposal in front of you

- ☐ Does it quote an automation percentage for an estate nobody has assessed?
 - ☐ Does it name the transformation types it cannot convert?
 - ☐ Does it price the Manual bucket as new build, or as a discount on conversion?
 - ☐ Does the senior name on the proposal appear on the engagement?
 - ☐ Is there a written list of what the vendor does *not* do?
-
-

12. Summary

A PowerCenter exit is not a conversion project with some residue. **It is a residue project with some conversion**, and every part of a defensible plan follows from accepting that.

Get the assessment first, because it is the only estate-specific number that exists. Decide between re-host and conversion deliberately, because the two are sold under one banner and only one of them modernizes anything. Scope the

components that re-platform separately, because they are invisible in an asset count and decisive at cutover. Use AI where it is genuinely transformative — reading an estate nobody understands, and drafting the code — and do not let it near the deciding. Then prove the result, row by row, because a pipeline that runs is not a pipeline that is correct.

None of this is a secret and none of it is proprietary. It is written down here because the material a buyer is usually handed is a percentage and a logo, and a percentage is not a plan.

ABOUT THIS PAPER

DataCode is a data integration consultancy that moves enterprises off Informatica PowerCenter onto IDMC and cloud-native services. A senior engineer leads every engagement. This paper sets out how the work is done as a standard rather than as a sales document, which is why the method, the review gate and the reconcile gate are given in full: publishing them costs nothing and demonstrates the claim better than asserting it.

The mechanics described here are drawn from the vendor's published product behaviour, from practitioner reports, and from the founders' own delivery experience on enterprise PowerCenter estates. Third-party figures are attributed to their source and are not ours. DataCode publishes no automation percentage of its own, for the reason given in section 6.