EPONA SOLUTIONS

Legacy Estate Assessment & Modernisation Roadmap

Prepared for Harwick Mutual Insurance Society

Document	Estate assessment: final report
Version	1.0
Author	Darren Bowles, Epona Solutions Ltd
Engagement	Fixed price, three weeks, read-only
Distribution	Harwick Mutual executive team and engineering leads
Confidentiality	Client confidential

ABOUT THIS SAMPLE. Harwick Mutual doesn't exist. I wrote this to show the shape, depth and tone of the real thing; every system, name and number in it is made up. A real report is built from your estate, your code and your people.

1. Summary

I spent three weeks in Harwick Mutual's estate: fourteen systems built between 1996 and 2022, running policy, claims and broker operations for around 90,000 members. This section is what I found and what I'd do about it; the detail sits behind it in sections 2 to 7.

First, the estate works. That needs saying, because the rest of this page is about risk and it would be easy to come away thinking the sky is falling. It isn't. But two problems are getting steadily worse and neither will fix itself.

- Premium rating for every product line runs on RATE/400, an RPG/400 system from 1996, and exactly one person can maintain it: a contractor, two days a week, who retires next year. The rating rules exist in his head and in the code, nowhere else. Of everything in this report, this is the risk with a date on it.
- The core policy and claims databases sit on SQL Server 2014 and Windows Server 2012 R2. Both are out of support and neither has received a security patch for some time. For a regulated firm that is an awkward conversation waiting to happen.
- Less urgent, but costing you every week: delivery is slow because the estate makes it slow. Keystone releases quarterly because its build only works on one developer's machine, and the nightly batch has crept past six hours with nobody entirely sure of the run order.

None of this needs a rewrite. The estate has reasonable seams, and a phased approach retires the worst risks early without betting the company on a big bang. AI-assisted methods help most with the ugliest part of the work, recovering undocumented behaviour, and your requirement that code stays in the building can be met with locally hosted models (section 5).

Options at a glance

Opt	Shape	Duration	Indicative cost	What it buys
A	Stabilise only: supported platforms, repeatable builds, batch sorted, quick wins	1 quarter	£40–60k	Removes the unsupported-platform risk and most of the delivery drag. The rating engine problem remains.
B	**Stabilise, then strangle the rating engine (my recommendation)**	3–4 quarters	£180–260k	Everything in A, plus the rating rules recovered, tested and served from a supported platform. The key-person risk goes away.
C	Full re-platform to .NET 8 and Azure	6–8 quarters	£400–600k	A modern estate end to end. I wouldn't do this now; it concentrates spend and risk without a matching near-term return.

My recommendation is Option B, and I'd start the quick wins in section 7 immediately whatever you decide about the rest. They come to roughly fifteen days of your own team's effort, and one of them pays for this report several times over, every year.

2. Scope, method and estate inventory

Scope. Three weeks, read-only. I had access to source control (two TFS collections, one Git organisation, and four shared folders which turned out to contain production code), a restored copy of the policy and claims databases, infrastructure inventory exports, and interviews with the head of engineering, the Keystone lead developer and the operations manager. I didn't request production access and didn't need it.

Method. Automated inventory and dependency extraction over everything I could find; static analysis for framework versions, dead code and end-of-life components; schema and workload analysis on the database copy; and the batch dependency graph rebuilt from the SQL Agent and SSIS definitions. I then cross-checked all of it in the interviews. Where what I was told didn't match what the code says, I've gone with the code, and flagged the one case where that mattered (section 3).

Estate inventory

System	Purpose	Technology	Born	Risk	Key people
RATE/400	Premium rating, all product lines	RPG/400, IBM i 7.2	1996	CRIT	1 (contractor, 2 days/wk)
Keystone	Policy administration	VB.NET WinForms, .NET Framework 4.5.2	2009	HIGH	2 (1 retiring)
ClaimsDesk	Claims handling	C# WinForms, .NET Framework 4.8	2014	MED	3
BrokerLink	Broker portal	ASP.NET WebForms 4.5	2012	HIGH	2
DocFactory	Policy document production	VB6 + Word COM automation	2004	HIGH	1
Nightly batch	Renewals, bordereaux, finance feeds	SQL Agent + SSIS 2014, 61 jobs	2010	HIGH	1 (ops)
FIN-BRIDGE	Finance interface to IBM i	Flat-file EBCDIC transfer, C# console	2011	MED	1
Member portal	Customer self-service	ASP.NET MVC 5	2016	MED	2
MI suite	Management reporting	SSRS 2014, 214 reports	2013	MED	1
CRM	Contact and address management	Vendor package, v2019, unpatched	2019	MED	vendor
Quote API	Aggregator quotes (pilot)	C# .NET 6, Azure App Service	2022	LOW	0 (orphaned)
Core databases	Policy, claims, finance staging	SQL Server 2014 / Windows Server 2012 R2	2015	CRIT	1 (DBA, part-time)

Two further minor systems (intranet and telephony integration) are in the working papers; neither changes anything here. The Quote API deserves a mention though: it's the newest and best code in the estate, and nobody has owned it since its author left in 2023. That's a waste, and I've picked it up in the roadmap.

3. The five findings that matter

R1. One person can maintain the rating engine — CRITICAL

RATE/400 calculates every premium Harwick charges. It's 214,000 lines of RPG/400, last restructured in 2011, and it's maintained by one contractor on two days a week who has told you he finishes next year. Nobody else at Harwick reads RPG. The rating rules aren't written down anywhere else either; I checked the pricing team's spreadsheets against the engine on twenty sampled cases and they disagreed on three, and in all three the engine was right, which tells you where the truth actually lives. If this man becomes unavailable before the rules are recovered, you can't safely change prices. Most of the risks in this report can be bought down whenever you choose. This one has a clock on it.

R2. The core databases are out of support — CRITICAL

SQL Server 2014 left extended support in July 2024; Windows Server 2012 R2 stopped getting updates in October 2023. The policy and claims databases, which is to say the entire member base, sit on both, unpatched. I'd treat the move to SQL Server 2022 as hygiene rather than something needing a business case; it's well-trodden work, and several of the quick wins hang off it.

R3. Nobody can draw the nightly batch — HIGH

Sixty-one jobs run in an order that lives partly in the schedules and partly in the operations manager's head. It took me the better part of week two to reconstruct the dependency graph; it's now appendix C, which I'd gently suggest is not where it should have had to come from. Elapsed time has grown from just under four hours in the 2022 logs to six hours ten minutes now. At that rate it starts colliding with the online day inside eighteen months. And when a run fails, recovery depends on one person knowing the order, at the cost of the following business day.

R4. Keystone can't be built repeatably — HIGH

The policy system builds on one developer's workstation and nowhere else, courtesy of locally installed components nobody has ever inventoried. There's no CI. Releases average one a quarter, and two of the last six needed a same-day patch. In my view this is the single biggest reason the business experiences IT as slow, and it makes every other change to Keystone riskier than it ought to be.

R5. The finance bridge fails quietly — MEDIUM, rising

FIN-BRIDGE moves money-relevant data to and from the IBM i as fixed-width EBCDIC files, and its character conversion silently truncates two field types. Finance reconciles the resulting differences by hand every month end; when I asked about it, nobody knew the cause, it had simply been absorbed into the process. It works until the day it doesn't, and it's coupled to the same machine as R1.

> **What didn't make the top five.** The VB6 document system and the unpatched CRM are real problems but contained ones, and the broker portal is dated rather than dangerous. All are sequenced in the roadmap instead. The full register, 23 items, is appendix A.

4. Key-person exposure and end-of-life summary

Knowledge concentration

Area	Depth	Exposure	First mitigation
RATE/400 rating rules	1	Contractor, retiring next year, 2 days/wk	Rule recovery with a parallel-run harness (phase 2), started while he's still around to answer questions
Keystone internals	2	Lead developer retires Q2 next year	CI plus documentation recovery (phase 1), paired handover
Batch run order	1	Single operations engineer	The dependency map now exists (appendix C); automate the run book from it
DocFactory (VB6)	1	One developer, who also owns FIN-BRIDGE	Contain: no new features; replace in phase 3
DBA function	1 (part-time)	Contract DBA, 3 days/month	Fold a monitoring baseline into the SQL 2022 migration; increase cover during phase 1

End-of-life register

Component	Where used	Support status	Action
SQL Server 2014	Core databases, SSIS, SSRS	Ended Jul 2024	Phase 1
Windows Server 2012 R2	Database and batch hosts	Ended Oct 2023	Phase 1
.NET Framework 4.5.2	Keystone, BrokerLink	Ended Apr 2022	Phase 1 (uplift to 4.8.1 first)
VB6 runtime + IDE	DocFactory	IDE unsupported since 2008; runtime tolerated on Windows 11, no promises beyond that	Phase 3
IBM i 7.2	RATE/400, FIN-BRIDGE target	Ended Apr 2021, and no extended support contract is in place	Phase 2 retires the dependency
TFS 2017 / TFVC	Source control (partial)	Ends 2027; already blocking modern tooling	Phase 1 (consolidate to Git)
TLS 1.0/1.1 endpoints	BrokerLink, FIN-BRIDGE transport	Deprecated everywhere that matters	Quick win

Support dates checked against the vendors' lifecycle pages during the assessment. The register in the working papers also covers the minor components, including two third-party .NET libraries with known CVEs; those two are on the quick-wins list because they're internet-facing.

5. AI-readiness assessment

You will have heard a lot of claims about AI on legacy code, and some of them are even true. My view, from using these tools daily: they are genuinely useful, occasionally startling, and unsafe without supervision. The practical question is never "should we use AI"; it's which systems, for what, with what checks. I applied three tests to each part of the estate: can we pin the current behaviour down with tests before changing anything, are we contractually allowed to process the code this way, and is the payoff worth the setup. The estate splits fairly cleanly.

Rating	Area	My take
GREEN	Documentation recovery, estate-wide	The highest-value, lowest-risk use there is. Generate current-state documentation of Keystone and ClaimsDesk from the source. The output is prose a human reviews, not code that executes, so the worst case is an editing job.
GREEN	Characterisation tests (Keystone, ClaimsDesk service layer)	The C# and VB.NET service code has clear enough seams that behaviour-pinning tests are cheap to generate and quick to review. Everything else in the roadmap stands on this.
GREEN	SQL tuning and batch analysis	Query plans, index proposals, SSIS analysis. Easy to keep honest: measure before adopting anything.
AMBER	BrokerLink and the 2,400 stored procedures	Doable, but the portal's logic is tangled into the UI, and the procedures are a mix of generated and hand-written. AI output here needs harder review and a slower pace. Do it after the green work has taught the team what good looks like.
AMBER	SSIS package modernisation	The packages are XML the tools handle badly. Convert with assistance, but verify every package against recorded outputs.
RED	RATE/400 direct translation	Machine-translating 214,000 lines of RPG without a harness would just launder unknown behaviour into a new language. Build the parallel-run harness first (phase 2), then translate rule by rule with every output checked against the live engine.
RED	FIN-BRIDGE and DocFactory	EBCDIC conversion and COM automation timing are precisely where these tools produce plausible nonsense. Human-led, with AI in an advisory role at most.
RED	CRM vendor package	The licence forbids decompilation and processing by third parties. Out of scope, full stop.

Tooling that fits your security posture

Your rule is that member data and source code do not leave the building, which is fair enough. It can be met two ways, and I tried both against a copy of the estate during week two:

- Locally hosted models on a single GPU workstation. Entirely self-contained; nothing leaves your network. Slower and less capable than the frontier models, but comfortably good enough for all the green-rated work above.
- A UK-region cloud deployment under contractual data-processing terms, with no training on your inputs. Frontier capability, worth having for the amber work, but only if your DPO signs the terms off, and with source only, never member data, as a hard rule.

Either way, the guardrails are the same: characterisation tests before any change, a human review on every merge, AI-assisted commits labelled as such, and a written list of what the tools must not touch (the red rows above). I don't regard any of these as negotiable.

6. The roadmap (Option B, my recommendation)

Phase 0 — Quick wins (weeks 1–6, alongside anything else)

The seven items in section 7: about fifteen days of effort, no dependencies on anything below, immediate return. Worth doing for morale alone; the team could use proof the estate can be changed safely.

Phase 1 — Stabilise (quarter 1) · 60–80 days

- Get all the source into one Git organisation, including the four shared folders, and retire TFVC.
- CI pipelines for Keystone, ClaimsDesk and BrokerLink, so builds stop depending on one workstation. Take Keystone from 4.5.2 to 4.8.1 while you're in there; it's a low-risk, in-place uplift.
- SQL Server 2014 to 2022, Windows Server to 2022, with Query Store switched on from day one to catch plan regressions.
- Turn the batch dependency map (appendix C) into an orchestrated, restartable run book, and apply the reordering from the quick wins.
- Documentation recovery on Keystone, AI-assisted, while the retiring lead is still there to correct it. This is time-boxed by his leaving date, so it goes in phase 1 even though it serves phase 3.

Phase 2 — Strangle the rating engine (quarters 2–3) · 90–130 days

- Harness first: replay a rolling window of real quote and renewal inputs through RATE/400 and record the outputs as the golden master. The pricing team's disagreement spreadsheet from section 3 becomes a test asset here.
- Recover the rating rules product line by product line, AI-assisted extraction with the contractor reviewing, into a documented, tested rule set.
- Serve rating from a new .NET 8 service behind an API. Run it in parallel with RATE/400 until the match rate holds at 100% over a full renewal cycle per product line, then cut over, line by line.
- Fix FIN-BRIDGE's truncation defects as its feeds re-point to the new service. Finance's month-end manual reconciliation currently costs a day and a half a month; I'd expect most of that back.
- Exit criterion: the IBM i decommissioned, or reduced to a read-only archive, and the key-person dependency gone.

Phase 3 — The edges (quarter 4 onwards) · 60–90 days

- Replace DocFactory with a modern document pipeline and retire the VB6 estate.
- BrokerLink: modernise the service layer under the existing UI. A full rewrite can wait until the argument is about capability rather than risk.
- Adopt the orphaned Quote API as the pattern for new services. It's already the best code you own; someone should be in charge of it.

> **Why this order.** Phase 1 before phase 2 because the strangler needs CI, supported platforms and recovered documentation underneath it. Phase 2 before phase 3 because R1 has a deadline and everything in phase 3 can wait. Put bluntly, it's risk retired per pound, earliest.
>
> **On the numbers.** The effort bands are informed estimates, and I'd rather you treated them that way. A four-to-six-week pilot on the first phase 1 subsystem would turn them into measured figures from your actual codebase before you commit programme money.

7. Quick wins: things your team can do now

#	Action	Effort	Impact
1	Add the two missing indexes on claims search and pin the regressed query plan (appendix B has the details)	2 days	Claims search from 31 seconds to under one. Measured on the database copy. This is the screen the contact centre rings up about.
2	Reorder the nightly batch per the dependency map; move statistics maintenance off the critical path	3 days	Batch from 6h 10m to roughly 2h 20m, modelled from the job timings. Removes the online-day collision risk for years.
3	Archive pre-2018 document blobs out of the primary policy database	3 days	Database from 1.9TB to about 600GB; backup window from 4 hours to under one. Restores become something you can actually rehearse.
4	Turn off TLS 1.0/1.1 on BrokerLink and the FIN-BRIDGE transport	1 day	Closes an audit finding that has been open for two years.
5	Decommission the three unused SQL Server instances I found during inventory	2 days	About £14,000 a year in licences and support, plus a smaller patch surface.
6	Stand up CI for ClaimsDesk, the easiest candidate, as the template for phase 1	4 days	The first repeatable build in the estate.
7	Patch the two third-party libraries with known CVEs in the member portal	1 day	Removes the two worst known vulnerabilities in your internet-facing code.

Call it fifteen days of work all in. Items 1, 2, 3 and 5 are not guesses; I measured or modelled them on the restored database copy during the assessment. Each one is written up as an implementation-ready ticket in appendix B, sized for your own team, with rollback steps and a verification measure. None of them needs me.

> **Why quick wins are in an assessment at all.** A report that only asks for money invites a long conversation. These seven give you something back inside the quarter, item 5 alone covers most of my fee every year, and they buy the engineering team some visible momentum before anyone has to make a bigger decision.

8. Appendices and close-out

Delivered alongside this report

Appendix	Contents
A. Full risk register	All 23 risks with likelihood, impact, owner and first mitigation, as a spreadsheet you can import into your risk system
B. Quick-win tickets	The seven items from section 7 as implementation-ready tickets: steps, rollback, verification measure
C. Batch dependency map	The 61-job graph, as a diagram and in machine-readable form, with the proposed reordering
D. Inventory detail	Per-system component versions, line counts, dependencies, end-of-life dates, licence notes, and the assumptions behind the effort bands
E. AI tooling evaluation	The two configurations I tested, with setup notes, how each performed on samples from your estate, and a proposed usage policy including the red-rated exclusions

A few things I want on the record

- The effort bands are planning figures with their assumptions stated in appendix D. They are not quotations.
- Read-only meant read-only; nothing was changed during the assessment. Where a quick win says "measured", I measured it on the restored copy, not production.
- Interview findings are labelled as such, and corroborated in code or logs where I could. The one material conflict I found, the pricing spreadsheets versus RATE/400, is reported in section 3.
- Nothing in this report requires buying more of my time. Where I've mentioned the pilot, the same work could be done by your own team, or another supplier, working from this document.

Close-out

The 90-minute walkthrough covers sections 1, 3 and 6, and I'll hand the quick-win tickets to your engineering lead at the same session. My access was surrendered at the end of week three. I keep no copy of your source or data beyond this report and its appendices, as the NDA requires.

END OF SAMPLE. Harwick Mutual is made up; the report you'd get isn't. Three weeks, £7,950 fixed, about your estate.
Darren Bowles, Epona Solutions Ltd · linkedin.com/in/bowlesdarren