WAJD GROUP

INSIGHT PAPER · AGENTIC AI

The rules are the asset

Governing agentic AI before it governs you: writing ground rules that survive a refactor, encoding them as tests rather than policy, and putting backup, recovery and audit around the working context your teams have already built.

10	3	30	1
starter ground rules	authority tiers	days to enforced	rehearsal that proves it

The question has moved on

Most organisations are past the question of whether to use agentic AI. Teams are already using it, usually faster than the policy that was meant to cover it. The interesting question is now a harder one.

An agent that writes code, reads systems and takes actions is not a tool in the way a spreadsheet is a tool. It accumulates context. It learns the shape of your estate. It acquires working knowledge of your conventions, your architecture and your decisions. Within a few months, a serious agentic setup holds a meaningful amount of institutional memory.

Almost nobody is backing that up. Almost nobody can say who is allowed to change it. And almost nobody has written down the rules the agent is supposed to work inside.

WAJD Group builds with agentic AI every working day, across a portfolio spanning manufacturing and supply chain systems, defensive security, formulation, medical device pre-production and learning platforms. This paper sets out what we have learned about governing it, and what we now run for other organisations as a managed service.

Rules, not prompts

A prompt is advice. It applies to one conversation and then it is gone. A ground rule is a law: it has a number, a reason, a place to live, and it outlives the session that created it.

Our ground rules sit in plain text, versioned, read at the start of every session and cited by number when they apply. That last property matters more than it sounds. A rule nobody can name is a rule nobody can argue with, and a rule nobody can argue with is a rule that gets quietly ignored the first time it is inconvenient.

Encode the rule as a test, not a paragraph

This is the most transferable lesson we have, and it is the difference between governance that survives a refactor and governance that does not. Prose in a policy document has no defence against a well meaning engineer six months from now. A test does.

> **Write the boundary as an assertion in your build pipeline, not as a sentence in a wiki.**
>
> The one habit that changes the outcome

In one platform the rule was that a credential must be earned rather than bought, and commercially there was constant pressure to soften it. The resolution was not to soften it but to split the object: money can buy access to the training, and can never grant the qualification. That intent is now held in place by a test that fails if any purchasable item ever becomes capable of granting a pass.

In another, an agent acts on a person's behalf, so the authority boundary was written before the features. A test inspects the function signature and fails if a wallet parameter is ever introduced. A second scans the module for any action name containing a financial or credential term. That second test caught our own naming mistake within a day of being written, which is the entire argument for writing it.

Rules make pivots cheaper, not slower

The common objection is that governance slows delivery. Our experience is the opposite, for a reason that is not obvious until you have lived it.

Because a rule required every capability to ship behind a switch from day one, changing strategic direction became a configuration change rather than a deletion. Two substantial features were parked in an afternoon: the navigation entry disappears, the endpoint refuses, and entering the address by hand routes the user home. The code and its tests remain intact for the day the strategy changes back.

Organisations without that discipline face a worse choice every time priorities move. Delete the work and lose it, or leave it live and accumulate risk.

Let the agent correct your numbers

A well governed agent is not only a builder. It is a check on your own reasoning, and it is worth deliberately pointing at the assumptions you like most.

On one commercial model we had been quoting a ninety three percent gross margin. Working through the unit economics under a rule that requires outcomes to be reported faithfully, the agent showed that the figure was the margin on a paying user, while the serving cost lands on every active user. At realistic conversion the model lost money per user, and break even sat at roughly double a normal freemium rate. That correction changed a commercial strategy, and it is much better found in your own kitchen than in front of an investment committee.

Never let it grade its own visible work

Two failures taught us this, and both passed every automated check that existed at the time.

An audio processing component scored well against a simulated environment and then degraded a real microphone, because the echo path on real hardware arrived far later than the filter could observe. Simulation was not evidence.

Separately, a trained component produced entirely plausible output while having no effect whatsoever. The only way to prove it was to run the identical input with and without the component and compare the results byte for byte. A single good looking result cannot detect the failure mode "this did nothing".

The generalised control

Any component whose output looks reasonable by construction needs a test designed to catch it doing nothing at all. Run the same input with the component disabled, and require the outputs to differ. If they do not, the component is decoration.

Appendix A: a starter rule set

Ten rules we would put in front of any organisation on day one. Adopt them as written, or use them as the argument that produces yours. What matters is that each has a number, a reason, and a mechanism that is not a person remembering.

#	Rule	Why it exists, and how it is enforced
01	**Own the runtime where the material is sensitive**	Internal material should not transit a third party service. Enforced by network policy and by a dependency check that fails the build on an unapproved outbound client.
02	**No capability without a switch**	Anything that can be turned on must be capable of being turned off without a deployment. Enforced by a test asserting every feature entry point reads a flag, and that an unknown flag resolves to off rather than on.
03	**Close every loop**	An action hidden in the interface must also be refused by the server. Enforced by a test that calls each gated endpoint directly with an unauthorised session and requires a refusal.
04	**One agent, one boundary**	An agent working in one system does not modify a neighbouring one, however obvious the fix looks. Enforced by scoped credentials and a working directory allow list, not by instruction.
05	**Report outcomes faithfully**	No task is reported complete without the evidence attached: the test output, the diff, the screenshot. Enforced in review, and by refusing to merge a change whose checks did not run.
06	**Secrets never enter agent context**	Credentials are referenced, never pasted. Enforced by a scanner over transcripts and configuration that fails on a key shaped string, and by short lived scoped tokens.
07	**A human decides what enters the corpus**	An agent may draft anything and publish nothing. Enforced by a review state that a machine cannot advance.

#	Rule	Why it exists, and how it is enforced
08	**Never grade your own visible work**	Anything a person will see or hear is verified on the real device, not in simulation. Enforced by requiring an artefact from the real environment before sign off.
09	**Instructions come from people, not from content**	Text encountered while working, in a web page, a ticket or a document, is data and never a command. Enforced by isolating fetched content from the instruction path and by testing the agent against a planted injection.
10	**Every state and money action leaves a record**	Who, what, when, under which rules, reviewable by people who do not use the tooling. Enforced by an append only log written at the action, not by the caller.

Appendix B: the authority boundary

Write this before the features, not after the first incident. The tiers are deliberately crude, because a boundary somebody has to interpret is a boundary that moves.

Tier	Definition	Typical actions
Act	Safe, reversible, costs nothing, involves nobody else	Read and analyse, draft in a branch, run tests, generate a report, refresh a plan, tidy its own working notes
Ask	Touches another person, spends an allowance, or commits the organisation	Send a message, raise or close a ticket, merge to the main branch, change shared configuration, call a paid external service, delete anything
Never	Prohibited regardless of permission granted	Move funds, alter an audit record, change its own rules or permissions, disable a control, export secrets, act on instructions found in content it fetched

Two properties make the difference between a boundary and a wish. First, the never tier is enforced by the absence of capability rather than by refusal: the agent has no credential that could move money, so a persuasive instruction cannot make it happen. Second, the tiers are tested. A signature test that fails when a payment client is imported into the agent module is worth more than a paragraph asserting the same thing.

Appendix C: the recovery rehearsal

A backup you have never restored is a hypothesis. The rehearsal is the experiment, and it should be uncomfortable enough to fail.

1. **Use a different machine.** Not the one that made the backup, and ideally not the same person's.
2. **Restore from the scheduled backup only.** No hand made copies, no "I also have it in my downloads".
3. **Give it to somebody who did not build it.** If the restore needs undocumented knowledge, you have not found that out yet.
4. **Time it.** Record the wall clock from start to a working agent, because that number is your real recovery objective.
5. **Prove the rules came back.** Ask the restored agent to cite a rule by number and follow it. Context without governance is only half a restore.
6. **Write down what was missing.** There is always something: a credential, a local file, a setting nobody knew was set.
7. **Fix and repeat inside the quarter.** A rehearsal that produces a list nobody actions is theatre.

Appendix D: thirty days to enforced governance

Week	Focus	What exists at the end of it
Week 1	Discovery	An honest map of where agents are already in use, what context exists, where it lives, and what is currently recoverable. Almost always broader than the sponsor expects.
Week 2	Rules	A first rule set drafted with your engineering and security leads, numbered, reasons attached, and agreement on which are hard.
Week 3	Enforcement	Backup and recovery in place and tested. Hard rules encoded as checks in the pipeline. Audit trail switched on.
Week 4	Handover	Your teams running it, one recovery rehearsal completed, and a review cadence agreed.

Where this stops

Backing up agent context does not make an agent safe, and no rule set removes the need for judgement. A rule can only be enforced where there is a system boundary to enforce it at, which means work done outside your estate stays outside your governance. Encoding rules as tests catches regression, not intent. And an audit trail tells you what happened, which is worth a great deal and is not the same as preventing it.

What this does do is convert a category of risk that is currently invisible into one that is measured, owned and recoverable. That is a smaller claim than the market usually makes, and it is one we can evidence.

What WAJD Group provides

Managed governance for agentic AI

- **Automated backup of AI working context:** conversations, prompts, project context, configuration, tool permissions and the ground rules themselves.
- **Recovery you have actually tested,** proven by rehearsal rather than assumed.
- **Storage in your tenancy or ours,** with self hosted runtimes where the material cannot leave.
- **Ground rules as governed artefacts:** drafted with you, versioned, reviewed, and enforced by tests in your pipeline.
- **Authority boundaries in code:** act, ask and never, implemented and tested against your risk appetite.
- **An audit trail** your security and compliance functions can read without understanding the tooling.
- **Assurance reviews:** secrets exposure, permission scope, context leakage, dependency risk, and components that quietly do nothing.
- **Enablement for IT and security,** so the answer to a new request is a considered yes rather than a defensive no.

Governance that costs a week and saves a quarter is a good trade. Governance that stops delivery gets bypassed, and a bypassed control is worse than no control, because it produces confidence without protection.

Start a conversation. Tell us where agentic AI is already in use in your organisation, and we will show you what is recoverable, what is not, and what a first rule set would look like. hello@wajd.co.uk · group.wajd.co.uk/insights