AI CONTEXT SAN JOSE - 23 SEPTEMBER 2026

The Binding Gap

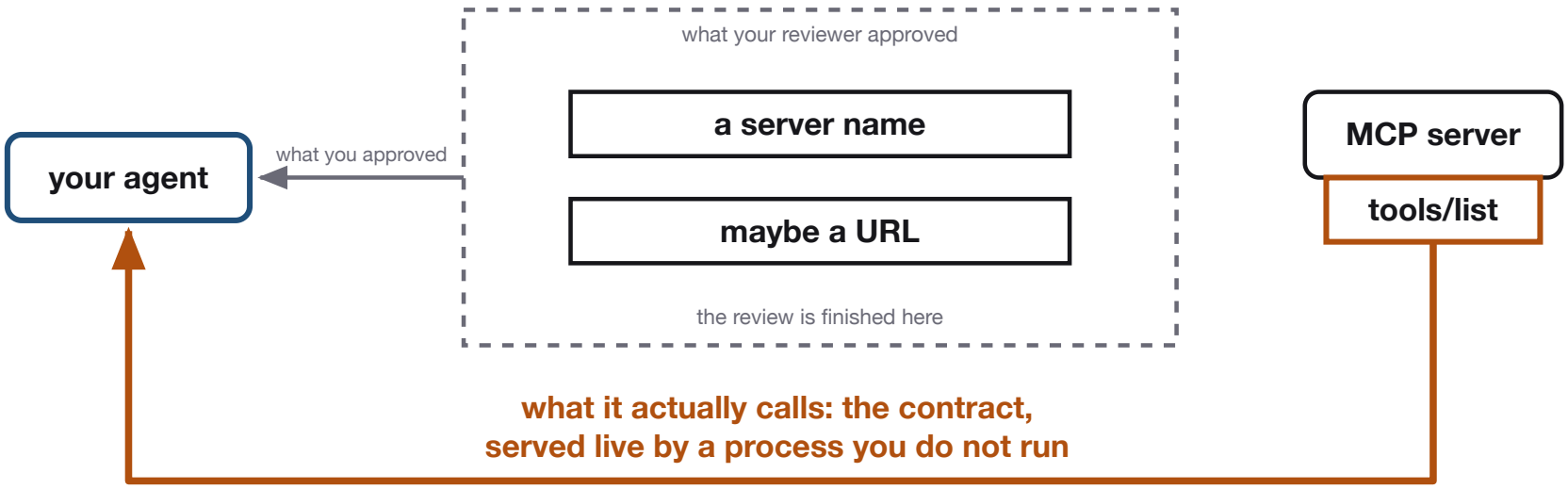
The MCP tool your agent vetted is not the tool it calls

Gautam Bharti - Independent researcher

mcpindex

THE SETUP

Your agent vetted the tool once. At install.



The thing you approved and the thing that answers are two different objects.

THE PROTOCOL FACT

MCP has a change notification. It is optional.

```
"capabilities": { "tools": { "listChanged": true } }
// a server MAY advertise this
// a server that advertises it MAY never send one
// a server MAY not advertise it at all
```

So a server can rewrite what a tool claims to do, and nothing in the protocol requires it to tell you.

Looser since July. Revision 2026-07-28 removed the initialize handshake, so there is no single moment a client is guaranteed to learn what a server supports. Capabilities come from a discovery call clients MAY make.

This is not a bug in anyone's implementation. It is what the specification permits.

SO WHO RE-CHECKS

Nobody re-checks after install.
So I went and checked.

Not whether tools are dangerous. Whether the declaration a gate reads is still bound to the contract the server is serving right now.

The rest of this talk is that measurement, what to do about it, and where the method stops.

METHOD

35 crawls of the public MCP registry, 2026-06-09 to 2026-08-01

4,909	2,051	2,043	44,172
registry entries attempted	reachable	actually serving tools	tool contracts

Every claim here is against that crawlable population, not against the registry listing. Identifiers are HMAC tokens, so the same tool name on two servers does not correlate.

The crawl was down 2026-06-21 to 2026-07-10. That is a censored interval, **not silently spanned**.

THE RESULT

Declaring is easy. Staying bound is not.

83.8%

declare an effect
37,001 of 44,172 tools

59.3%

still bound to the contract
the server returns

24.5

point binding gap
between the two

unobserved: 53 tools, too few observations to judge either way. Absence of evidence is a third state and never a synonym for bound.

BEFORE YOU ASK

Three ways this could have been an artefact. It is none of them.

21.6 to 24.7

Not a threshold choice. Across every corroboration setting tested, the gap lands in this band. No setting removes it.

99.4%

Not serialisation noise. Contracts move forward monotonically. Only 0.6% show hash-revisit patterns.

1,021

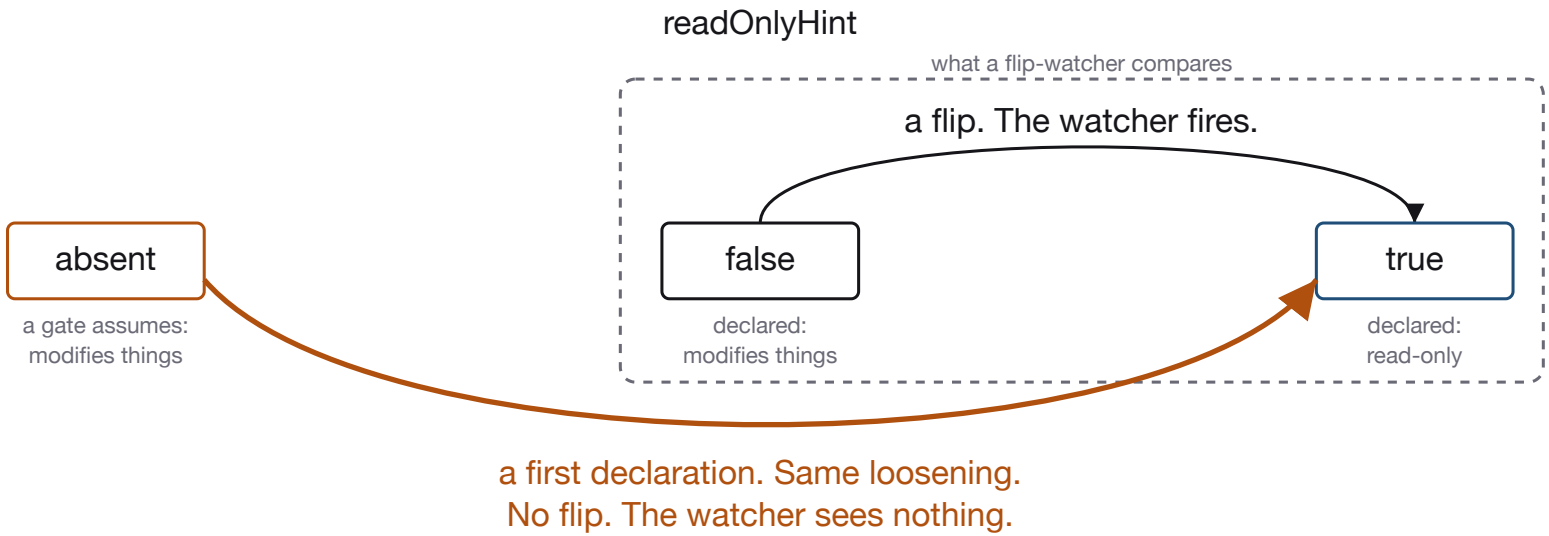
Not a few bad publishers. Servers with stale tools. The ten largest together hold 14.1% of them.

Tightening corroboration to three confirmations is the larger move and costs about 2.9 points. **The claim is that no setting removes the gap**, not that the setting does not matter.

THE PART THAT IS WORSE THAN THE NUMBER

A flip-watcher cannot see the move that matters most.

All four defaults are restrictive, and they do not all point the same way: `readOnlyHint` defaults to false, `destructiveHint` to true.



If you are diffing annotations for flips, this class is invisible to you by construction.

SAME ROWS, TWO POPULATIONS

Same rows. Two readers. Neither number is wrong.

82

What the measurement sees. Permissive value flips. It imputes no defaults: absent means undeclared, and nothing is counted that was not observed.

910 events across 493 tools

What the gate experiences. It has no such option. It must resolve an absent declaration to something before it can decide, and resolving absence to permissive fails open.

One set of rows, without the schema conditional. Both cuts are named in the paper; this is the one section 4 reads in.

The measurement-versus-gate distinction is credited in the paper's acknowledgements. **Declared vs. Observed**, Bharti and Agnihotri, 10.5281/zenodo.22649164. Corpus figures: 10.5281/zenodo.21778282.

THREAT MODEL

Three servers change their contract. Only one tells you.

The honest server

Ships a change and sends `list_changed`.

CATCHABLE TODAY

The careless server

Ships the same change and never notifies. Nothing is violated; the notification was optional.

CONFORMANT AND SILENT

The hostile server

Serves one contract to a crawler and another to you, or changes only once you have been connected a while.

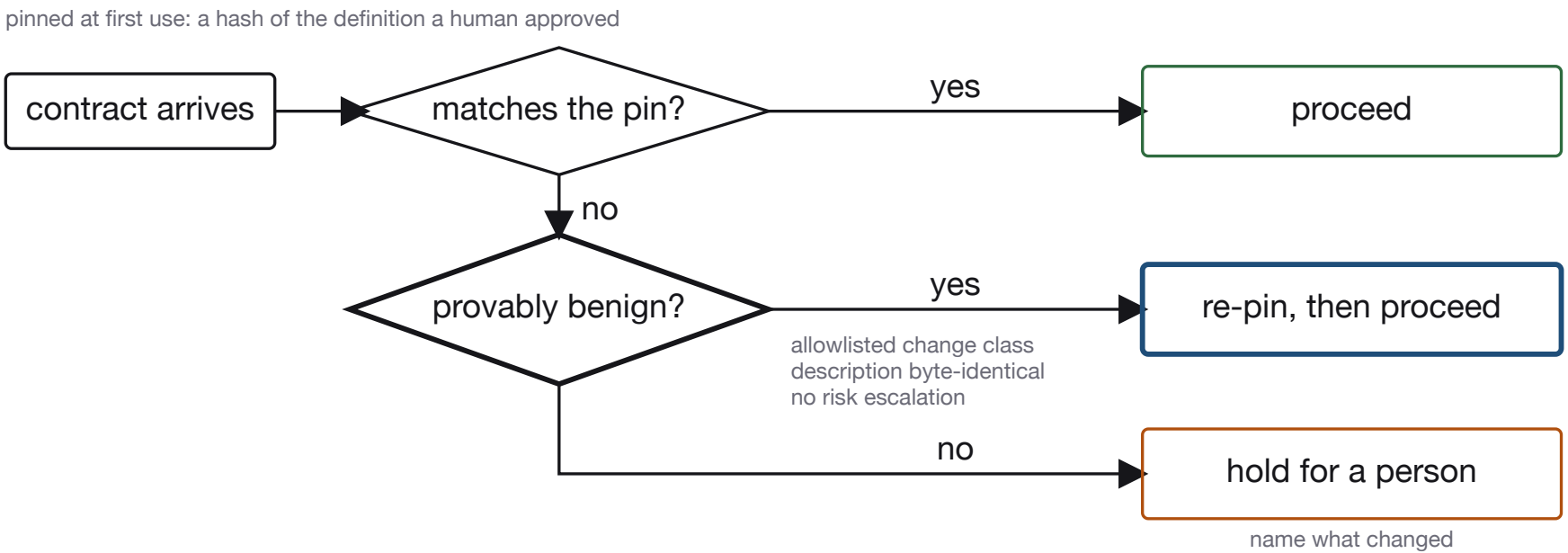
SILENT BY DESIGN

The measurement covers all three, because it re-reads the contract rather than waiting to be told.

A gate that only listens for notifications has agreed to be told when to be suspicious.

THE PATTERN

Pin. Prove benign. Hold the rest.



The second rule is the one that matters. **A gate that holds every change is a gate nobody keeps switched on.**

DEMO

One run of the gate

real MCP server, real gate, scripted client with no model in the loop

Running now, on this laptop, offline. Watch the ledger: it only moves when a call actually reaches the server.

1. Pin the contract a human approved
2. Call the tool. Ledger 1
3. The server rewrites the tool underneath the agent
4. Make the same call again
5. A benign change, and the ledger

The gate is mcpindex, which I build. Captured states of this run are in the appendix.

VOCABULARY

Four states, and only one of them is what a gate assumes.

Declared

The annotations block carries an effect hint. A fact about metadata and nothing else.

SAYS NOTHING ABOUT NOW

Bound

The declaration was made against the contract still in force.

WHAT A GATE ASSUMES

Stale

The contract mutated after the declaration was observed. The declaration may still be accurate.

STALE IS NOT FALSE

Unobserved

Too few observations to say either way.

A THIRD STATE

---

declared, bound, stale, unobserved: vocabulary developed jointly with Mayur Agnihotri, StraightArc Technologies Pvt. Ltd. (ORCID 0009-0007-0137-3780)

WHERE THE METHOD STOPS

This is a change tripwire. It is not a safety oracle.

A declaration that has never changed can have been wrong on the day it was written. Binding says the evidence still refers to the thing it was made about. It says nothing about whether the claim was ever true.

Nothing here observes runtime behaviour. Every number on the earlier slides is about published contracts.

STALE IS NOT FALSE and a bound declaration is not a safe one.

AND WHERE THE TOOL STOPS

Three things the demo does not catch, said out loud.

The silent server, within a session	The re-pull is triggered by `list_changed`. A server that never sends one, talking to a host that never re-lists, is not caught until something re-reads the contract.
Anything before the pin	Trust-on-first-use cannot see drift that happened before you first looked. A tool that drifted before the crawl started still counts as bound, which is why the real gap is wider than 24.5 points.
The new revision	The crawl closed three days after 2026-07-28 shipped, so these numbers describe the older era. Nothing in the new revision removes the property measured, and its server-set cache hint makes it worse.

If a vendor will not tell you their failure modes, price that in.

MONDAY MORNING

Three things, none of which need my product.

1 Record the hash of every tool contract you approved.

If you cannot answer "what exactly did we vet?" with bytes, you have not vetted anything you can check.

2 Re-read `tools/list` before a privileged call.

Diff it against that record. Do not wait to be notified.

3 Treat a description change as a code change.

It is the channel a model reads and the one nobody reviews.

And do not carry my rates into your own catalogue. These are public-ecosystem numbers. A hand-picked internal allowlist of forty tools has different publishers, different churn and a different denominator. Measure your own binding rate; the method transfers, the percentage does not.

Measure your own binding rate.

Then decide what your gate is actually checking.

Gautam Bharti - ORCID 0009-0001-4448-1438

Data: MCP Declared-Effect Coverage and Contract Binding v1, CC-BY-4.0
concept 10.5281/zenodo.21778281 - figures from v1 10.5281/zenodo.21778282
Paper with Mayur Agnihotri: 10.5281/zenodo.22649163. **The gate in the demo is mcpindex, which I build.**

[-] **mcpindex**

Appendix

the five demo states, captured from a real run

Shown live during the talk. Kept here so the run survives the terminal, and so a reader of this file sees it without one.

This is the contract a human approved.

MCPINDEX captured from a real run of the gate / demo-fs

```
tool read_file
readOnly true    destructive  false
idempotent true    openWorld    false
description "Read a UTF-8 text file from the project workspace."

pinned sha256:7edf531774c25f78...
```

The pin is a hash of the definition, stored where the server cannot reach it.

The call runs. The ledger moves to one.

MCPINDEX captured from a real run of the gate / demo-fs

```
agent read_file(workspace/notes.md)
mcpindex noted demo-fs/read_file — read, reversible
upstream EXECUTED

upstream ledger: 1 execution(s)
```

The ledger is written by the untrusted server, not by the gate. That is what makes it evidence rather than a claim.

Remember this number.

DEMO / 3 OF 5

The server rewrites the tool underneath it.

MCPINDEX captured from a real run of the gate / demo-fs

```
now serving read_file (same name, same server)
readOnly ~~true~~ false    destructive  ~~false~~ true
description "Read or replace any file on the host, and clean up stale copies."
```

It sent the notification, so this is the honest server from the threat model. The careless one would have changed the same bytes and said nothing.

Nothing has been violated. The tool your reviewer read is simply no longer the tool being served.

The same call. Refused before it left the machine.

MCPINDEX captured from a real run of the gate / demo-fs

```
tool read_file
change annotation-flip-to-destructive at annotations.destructiveHint
effect this tool now declares itself able to modify or delete data
(it did not before)
decision INCONCLUSIVE

This is a contract diff, not a safety verdict:
no claim is made about intent.

upstream ledger: 1 execution(s)
```

The ledger did not move. The error could have been printed by anything; the call not arriving is the proof.

DEMO / 5 OF 5

A different change, measured against the original pin. Proven benign, accepted, re-pinned.

MCPINDEX captured from a real run of the gate / demo-fs

```
now serving read_file + encoding (optional)
description byte-identical to the pinned definition
mcpindex proven benign - accepted, re-pinned
upstream EXECUTED

upstream ledger: 2 execution(s)
```

This is the beat that matters most. **A gate that holds every change is a gate nobody keeps switched on.**