

AGNTCON + MCPCON NORTH AMERICA · SAN JOSE

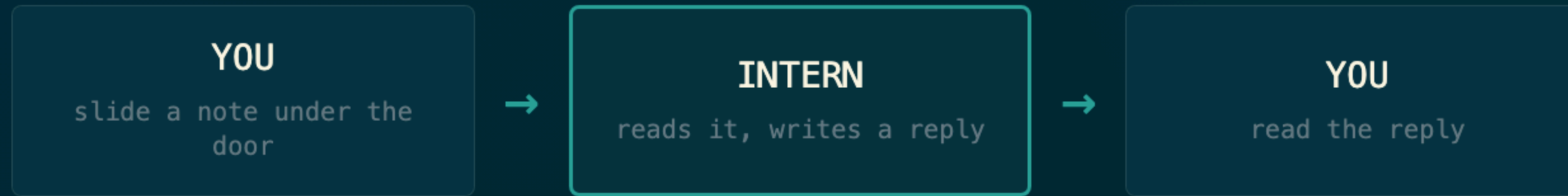
Your Tool Is in Another Castle

MCP makes tools portable. Does their meaning survive adaptation?

Thierry Damiba · [Arcade.dev](#) · 23 October 2026

START WITH THE MODEL

A model is a very capable intern in a windowless room.



- No internet, no phone, no library.
- No database, no order system; only paper fits under the door, in one exact format.
- Everything the intern knows about your problem, you wrote down first.

BUILDING ON

“Model Context Protocol in plain English”

Nate Barbettini · 9 September 2026 · caffeinate.blog/post/mcp-in-plain-english

THE WORKAROUND

You cannot fit the database under the door. You can fit instructions for asking.

```
# your note, in the only format that fits
If you need the order system, write LOOKUP(order_id).
A customer is asking where order 4417 is.

# the intern's note
LOOKUP(4417)

# your note, after you queried the database
shipped 11 Sep · arriving 16 Sep · carrier DHL
```

The model requests. Your application code executes. That pattern is tool calling. **Everything has to be flattened to fit.**

MORE TOOLS, MORE PROBLEMS

Every integration needs its own hand-written glue.

lookup_order

glue you wrote

search_docs

glue you wrote

get_calendar

glue you wrote

send_email

glue you wrote

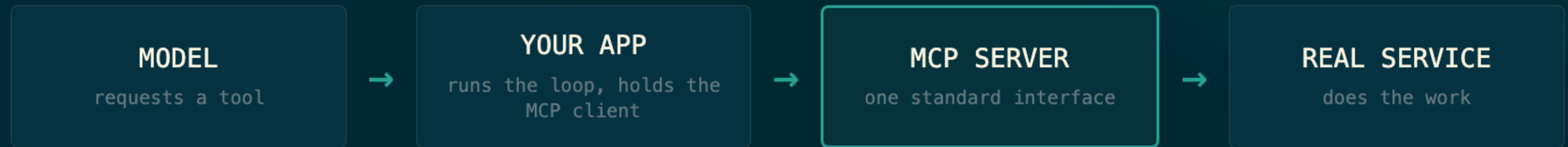
refund_payment

glue you wrote

A description the model can read, plus code that performs the call. **Once per integration.**

MCP IS A STANDARD INTERFACE

Write the integration once. Any MCP client can use it.



The standardized conversation runs between your application and the MCP server.

THE PART THAT STAYED THE SAME

From the model's point of view, nothing changed.

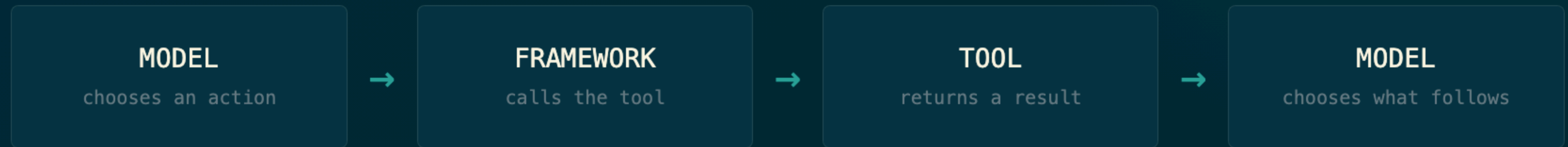
“The model still only sees the `get_weather` tool. The MCP conversation happens between the application and the MCP server.”

Nate Barbettini, “Model Context Protocol in plain English,” 9 September 2026 · caffeinate.blog/post/mcp-in-plain-english

Somebody still writes the note. In an agent application, that somebody is your framework. **MCP never standardized the note.**

FIRST: WHAT IS THE AGENT?

The agent is the loop that lets a model use tools.



In the five integrations tested, the framework maps MCP objects into its own tool and message types.

SO I TESTED THE NOTE · ONE SERVER · ONE CALL

The server returns text, image, audio, resources, and structured JSON.

T

TEXT

content[]



IMAGE

image/png



AUDIO

audio/wav



RESOURCE

embedded + link



JSON

structuredContent

Each part carries a different traceable marker so we can see where it lands.

SAME RESPONSE IN · FIVE OBJECTS OUT

Every framework received the identical MCP response.

IN text · image · audio · resource · link · structured JSON

CREWAI

`BaseTool.run`

Out: text only, as a list-shaped string

Gone: image, audio, resources, JSON

LANGCHAIN

`StructuredTool.ainvoke`

Out: nothing; conversion raised on audio

Gone: the whole result at this path

OPENAI AGENTS

`MCPUtil.invoke_mcp_tool`

Out: text, image; audio and resources as JSON
text

Gone: structured JSON, text annotations

PYDANTIC AI

`MCPToolset.direct_call_tool`

Out: text, image, audio; resources flattened to
text

Gone: structured JSON, resource metadata

MASTRA

`tool action execute`

Out: structured JSON

Gone: every content block

THE INTEROPERABILITY GAP

The connection was portable.
The agent-facing contract was not.

Protocol connectivity

can these two programs talk?
server → MCP client

≠

Contract portability

do I get the same thing to work with?
framework → agent runtime

MCP removed the glue between application and server. The glue between adapter and agent runtime is still written once per framework.

WHY APPLICATION DEVELOPERS SHOULD CARE

Code written against the MCP result does not move between adapters unchanged.

WHAT THE MCP SPEC TEACHES YOU TO WRITE

```
if result.isError:  
    retry()  
  
records = result.structuredContent["records"]
```

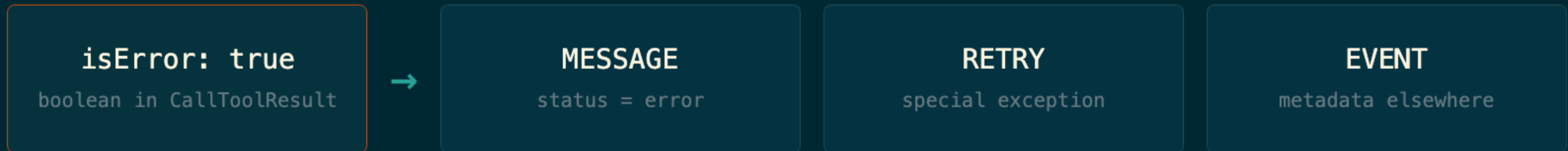
WHERE THOSE MEANINGS LANDED AFTER ADAPTATION

```
message status · exception · event  
artifact · framework output  
serialized text · omitted field
```

That changes error handling, policy inputs, parsing, and cross-framework application code.

ONE SOURCE LANGUAGE · FIVE RUNTIME LANGUAGES

MCP says `isError: true`.
Each adapter translates that
meaning into its framework's
grammar.



The fixture advertises a destructive operation.

```
Tool(  
  name="delete_account",  
  description="Permanently deletes a user account...",  
  inputSchema={  
    "account_id": {"type": "string"},  
    "confirm": {"type": "boolean"}  
  },  
  annotations={  
    "destructiveHint": true,  
    "readOnlyHint": false  
  }  
)
```

Missing confirmation becomes a tool-execution error.

```
if name == "delete_account" and not arguments.get("confirm"):  
    return CallToolResult(  
        content=[TextContent(text="confirm was not set")],  
        isError=True,  
    )
```

The protocol call succeeds. The requested operation does not run.

TWO LAYERS REPORT TWO DIFFERENT OUTCOMES

JSON-RPC says the request worked. MCP says the tool operation failed.

JSON-RPC

`"result": { ... }`

The request was
valid and processed.

CALLTOOLRESULT

`"content": [ ... "confirm was not set" ]`
`"isError": true`

The tool ran and refused.

A protocol success can contain a tool-execution failure.

`isError` is a boolean field on `CallToolResult`, sitting beside the content it explains. It is the only field MCP defines to mark a result as a tool execution error.

YOUR TURN · DESIGN THE ADAPTER

The server says the
operation failed.

What would you hand the agent?

There is no protocol-mandated framework representation.

THE REVEAL

One boolean on the MCP result becomes three different integration contracts.

PLAIN RESULT

CREWAI

ordinary text

BaseTool.run

Flag not present

PLAIN RESULT

LANGCHAIN

ordinary text
block

StructuredTool.ainvoke

Flag not present

PLAIN RESULT

OPENAI AGENTS

normal
ToolOutput

MCPUtil.invoke_mcp_tool

Flag not present

RETRY EXCEPTION

PYDANTIC AI

ModelRetry

MCPToolset.direct_call_tool

Failure as control flow

THROWN ERROR

MASTRA

thrown error

tool action execute

Failure as control flow

No measured path returned the unchanged MCP result.

WHAT THE FLAG WAS FOR

The explanation survived everywhere.
The machine-readable marking did not.

MCP SPECIFICATION · ERROR HANDLING

“Clients **SHOULD** provide tool execution errors to language models to enable self-correction.”

tool execution error
actionable feedback the model
can use to self-correct

marked by `isError: true`

ANSWERING “DID THIS CALL FAIL?” AFTERWARD

One question. A different
implementation per contract.

Plain result no typed field
3 of 5 captures

Retry exception except `ModelRetry`

Thrown error try / catch

The text still reaches the model. What moved is whether surrounding code can identify the class of result the specification singles out.

THE MISSING CONTRACT

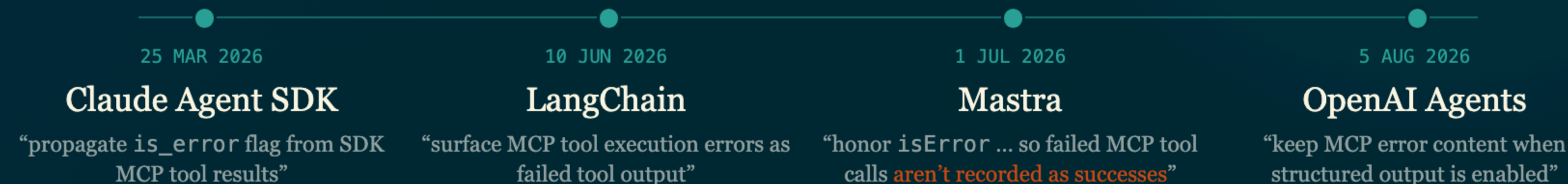
MCP standardizes
server → client.

It does not standardize
framework → agent.

“Supports MCP” confirms the connection,
not the contract the agent runtime receives.

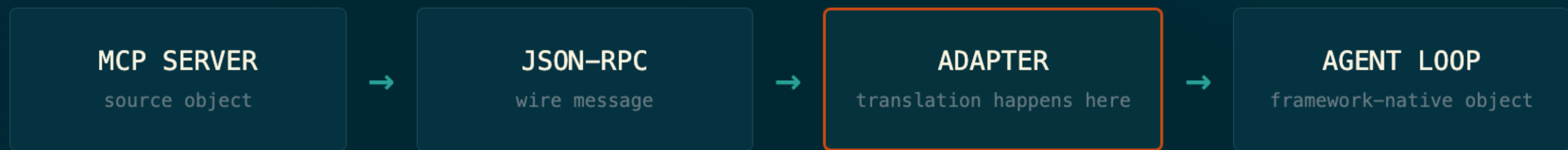
DOES IT MATTER? FOUR TEAMS ALREADY ANSWERED

Merged fixes, in their maintainers' own words.



Each fix lands at a boundary that team chose. That is the finding: where the failure signal lives is a per-framework decision, not a protocol guarantee.

THE PATH



METHOD

One test server. One protocol version. And I say exactly where I looked.

- MCP 2025-11-25, because all five frameworks speak it
- One server, handing every framework the same awkward tools and results
- Each framework's own conversion code, not my imitation of it
- Every field carries its own marker, so I can find where it went
- I read one object per framework. I did not follow it to the model

"Where I looked" has a name here: the **capture boundary**, meaning the exact function I called and the exact object it handed back. Every published result is stamped with one.

The same hint lands in different framework layers.

TEST: send the same Tool with destructiveHint: true, capture each adapted definition, then locate that exact boolean.

FRAMEWORK	OBSERVED DESTINATION OF DESTRUCTIVEHINT
CrewAI	Not attached to the adapted tool object
LangChain	Retained in StructuredTool.metadata; absent from converted provider schema
OpenAI Agents	FunctionTool retains MCP title, not behavioral hints
Pydantic AI	Retained in ToolDefinition.metadata; outside parameter schema
Mastra	mcpMetadata carries server identity, not annotations

If your safety check only reads
what gets sent to the model,
it never sees these hints.

Typed results can become content, artifacts, or framework output.

FRAMEWORK	WHERE THE JSON GOES	EVIDENCE
CrewAI	Returned as human-readable text	Measured
LangChain	<code>ToolMessage.artifact</code> , separate from model-visible content	Documented
OpenAI Agents	Model-visible JSON and custom-data context	Documented option
Pydantic AI	Depends on the surrounding content-part types	Measured
Mastra	Framework output value; content blocks omitted	Measured

INTERPRETATION

This is a record of where things landed, not a scoreboard.

- Each framework exposes a different agent abstraction.
- Transformation is sometimes intentional and sometimes configurable.
- A changed field is not automatically a bug.
- The operational question is whether applications know what changed.

WHAT ABOUT THE OTHER PROTOCOLS

The newer standards cover different boundaries. Not this one.

MCP

application ↔ server

A2A

agent ↔ agent

ACP

editor ↔ coding agent

adapter → runtime

inside one process

“A2A does not specify how an agent talks to its own sub-agents or how it invokes tools.” Its guidance instead: “**use your framework’s native primitives.**”

A2A Protocol specification, “What A2A is not” · a2a-protocol.org · verified 14 September 2026

WHAT THIS STUDY DOES NOT CLAIM

- It does not measure whether a model understands the resulting text.
- It does not measure overall framework quality.
- It does not treat MCP annotations as sufficient authorization controls.
- It does not generalize beyond the named package versions and capture boundaries.

WHAT DEVELOPERS CAN DO NOW

Five checks make adapter behavior explicit.

- Test the framework object, not only the MCP server response.
- Include failure meaning in text while also setting `isError`.
- Verify safety policy inputs independently of advisory tool annotations.
- Exercise mixed text, image, audio, and resource results in integration tests.
- Pin the adapter and MCP SDK versions used by those tests.

SINCE SEPTEMBER

Five frameworks became thirty-one.

- 31 framework adapters measured on MCP 2025-11-25, across 63 of its 64 features
- On MCP 2026-07-28, 18 of those 31 cannot open a session yet
- 16 coding-agent harnesses measured on the wire: Claude Code, Codex, Cursor, Gemini CLI and more
- Every cell links to its capture; unmeasured rows stay visible as unknown

THE QUESTION TO ADD TO EVERY MCP INTEGRATION TEST

Not only: “What did
the server send?”

Also: “What reached the agent?”



Browse the measurements: canaiuse.com

Every cell links to its capture and a reproduce command. Maintained by Arcade.dev.