

MOPZA26 ONLINE · SESS-59

Who changed that?

An AI watchdog for risky changes across your stack

Jimmy Piraino

Senior Manager, Commercial Technology and Platforms · Accruent

THE MORNING IT BROKE

8,400

spurious MQL notifications, one window, nobody launched anything

09:00

Slack lights up. A BDR manager asks what we launched.

09:05

We did not launch anything.

09:20

MQL count for the day is vertical.

Rest of day

Working backwards from the symptom, one guess at a time.

THE GAP

The change took four seconds. Finding it took most of a day.

1

Change made

Unauthorized edit to lead scoring, in production

2

Notifications fire

Every rescored contact crosses the MQL threshold

3

A human notices

Not from the system. From a confused BDR manager.

4

We work backwards

Symptom in pipeline, cause somewhere in the stack

And it kept happening. April, then May, then again in June, after we thought the process fix had landed. Process was not going to save us. Process is what already failed.

THE UNCOMFORTABLE PART

The whole time, HubSpot knew.

Every one of those changes was in the audit log. Timestamp, actor, email address. The information was not missing. It was sitting in a table nobody opens.

When did you last open your CRM audit log on a day when nothing was on fire?

For most of us the honest answer is never. We open the audit log after something breaks, to confirm what broke.

WHO IS TALKING

Jimmy Piraino

Senior Manager, Commercial Technology and Platforms · Accruent

1. **A two person marketing ops team**
   Me, and one HubSpot admin. That team size shapes every decision in this talk.

2. **15+ years in martech and ops**
   CrowdStrike, Coursera, Teradata, and now Accruent.

3. **A Microsoft shop**
   Teams, Power Automate, Azure, SharePoint. Not a preference, just the environment, and it explains most of what I built.



THE OBVIOUS OBJECTION

Why not just lock down permissions?

You are already thinking it, so let me say it first.

WE DID SOME OF THAT

- Tightened access to integrations and APIs after the June incident
- Fewer keys, fewer standing permissions
- It is good hygiene and you should do it

IT DOES NOT SOLVE THIS

- The contractor was authorized. They were supposed to be in there.
- The people who break your CRM almost always have legitimate access to it
- On a two person team, locking it all down makes me the bottleneck for every request

Access control answers who is allowed in. It cannot answer whether what they did was a problem. Different question, different system.

REFRAME

This is not a monitoring problem. It is a triage problem.

The events are already recorded. An audit log gives you a record, a timestamp, and an actor. What it never gives you is an answer to the only question you care about:

Does this matter?

We generate hundreds of these a day. A human reading hundreds of rows to find the two that matter is not a plan. That is a slower way of finding out when the pipeline breaks.

ONE RAW AUDIT ENTRY

```
category:       PROPERTY_VALUE
action:         UPDATE
targetObjectId: 40718322901
actingUser:     a****@vendor.com
```

Technically complete. Practically useless. Same shape whether somebody fixed a typo or rewired your scoring model.

THE WHOLE IDEA

Three boxes

I am not using AI to detect the change. HubSpot detects the change. I am using AI to do the reading I don't have the time to do.

1

Audit log

Whatever your platform already records. You are not building this part.

The events already exist

2

Reasoning layer

Summarize it, rate it HIGH or LOW, propose a next step. One record at a time.

Closes: unreadable, no risk signal

3

Triage message

Only the risky ones, pushed into a place your team already looks.

Closes: no routing

The rule that matters: no new destination. If the output lands somewhere new, it dies the same death the audit log did. Ours goes to a Teams channel we are in all day. Yours might be Slack, email, or a ticket queue. Do not make people go anywhere new.

WHAT I BUILT

Two flows, one channel

FLOW 1

HubSpot Change Watchdog

Polls the audit log every 30 minutes, sends each entry to a model, logs all of it, and posts a Teams alert for anything rated high risk.

FLOW 2

MQL Volume Monitor

Runs every 10 minutes. Counts contacts entering the MQL lifecycle stage in that window and alerts if the count crosses a threshold. No AI at all.

One Teams channel

Why two?

Flow 1 catches causes.

Flow 2 catches symptoms.

If the model misjudges a change as low risk, Flow 2 still fires. Two independent detection paths for one failure.

The original incident would have been caught by either one.

FLOW 1, STEP BY STEP

HubSpot Change Watchdog

Scheduled flow, every 30 minutes

1. **Build the window**
   Compute the start timestamp and assemble the request URL in a Compose action.
2. **Call the audit log**
   Page through every result in the window. Audit logs paginate, and you will silently miss changes if you only take page one.
3. **Reason over each entry**
   One Azure OpenAI call per entry. Three outputs come back.
4. **Log it, then gate it**
   Every entry gets written to a SharePoint list. Only the ones rated HIGH go on to post in Teams.
5. **Catch your own failure**
   The whole thing sits in a scope with error handling that posts to the same channel if the flow itself dies.

A silent watchdog is worse than no watchdog, because you think you are covered.

FOR THE HUBSPOT PEOPLE

The actual call

```
GET  https://api.hubapi.com/account-info/v3/activity/audit-logs
     ?occurredAfter={windowStart}
```

Enterprise tier only

If you are on Professional, this endpoint is not there. Do not stop listening. Flow 2 needs no audit log and no Enterprise tier, and it would have caught my incident on its own.

Scope and auth

`account-info.security.read`

Use a Service Key as the bearer token, not a legacy Private App. Service keys rotate without a rebuild and do not die when somebody offboards.

What comes back per entry: **category** · **subCategory** · **action** · **targetObjectId** · **actingUser**

Categories in play: DATA_PRIVACY_REQUEST, LOGIN, CRM_OBJECT, CRITICAL_ACTION, CONTENT.

BEFORE THE MODEL SEES IT

Look it up.
Do not ask the model.

An audit entry hands you a twelve digit object ID. Before anything reaches the reasoning layer, the flow checks whether the change touched a workflow and, if it did, makes a second HubSpot call to resolve that ID into the real name.

WHAT THE LOG GIVES YOU

```
targetObjectId: 40718322901
```



WHAT THE MODEL ACTUALLY GETS

"Lead Scoring v3"

Every fact you hand a model deterministically is a fact it cannot get wrong. Most of the AI failures I have seen in ops are not the model being dumb. They are somebody asking a model to guess at something the system already knew.

THE REASONING LAYER

Three outputs, asked for explicitly

gpt-5.4-nano through Azure OpenAI. Small, cheap, fast. You do not need a frontier model to read one structured record and answer three questions.

1

Plain English summary

Turn the enum soup into a sentence a human reads in two seconds.

"Someone updated the lead scoring property on a production workflow."

2

HIGH or LOW risk

Two options on purpose. Give it five levels and you get a wall of mediums.

HIGH

One of two. Never a maybe.

3

A specific triage step

Not "investigate." Something a person can do without thinking first.

"Confirm with the requester whether this scoring change was approved, then check MQL volume for the last two hours."

This is not a vibe check. The prompt carries an explicit rubric. Default to LOW. Rate HIGH only against a named list: a workflow edited or deleted, a workflow created by anyone outside the two named admins, data pointed at a new external endpoint, lead scoring or routing changed, a property or pipeline structure changed.

WHAT ACTUALLY SHOWS UP

The entire product

PA

HubSpot Change Watchdog

Today 09:32

HIGH

WHO

a****@vendor.com · 09:04 UTC

WHAT

Updated the lead scoring property used by an active production workflow.

GO CHECK

Confirm with the requester whether this change was approved, then check MQL volume for the last two hours.

@Jimmy

No dashboard.
No app to log into.
Nothing to remember to open.

The reason audit logs fail is not that the data is bad. It is that the data lives somewhere you do not go. All I did was move it to somewhere I already am.

THE LEDGER

The alert is what you see. The list is what you keep.

Before anything gets gated, every entry is written to a SharePoint list. HIGH and LOW, all of it. Teams is for the moment. The list is for everything that comes after it.

1. **It is queryable**

   Who changed what, when, and what the model said about it. One list, instead of scrolling back through a channel.

2. **It measures the model**

   Every row carries the rating and the reasoning, so I can go count how often it got one wrong.

3. **It leaves room for a human**

   Every row says DecisionSource: AI. The column exists so a person can overrule one and still be on the record.

ONE ROW PER CHANGE

Title	plain-English summary
OccurredAt	timestamp
Category	PROPERTY_VALUE
SubCategory	WORKFLOW
Action	UPDATE
TargetObjectId	40718322901
ActingUserEmail	a****@vendor.com
RunId	which run caught it
RiskRating	HIGH
AIReasoning	what to go check
DecisionSource	AI

FLOW 2, THE SAFETY NET

The dumb one that catches what the smart one misses

Scheduled flow, every 10 minutes → **Query the CRM Search API for contacts entering MQL in that window** → **Count them** → **Over 200? Post to the same Teams channel**

No AI in this one. No audit log. No Enterprise tier. It runs on any plan with the Search API.

Flow 1 depends on a model correctly judging that a change was risky. That is a probabilistic call and sometimes it will be wrong. Flow 2 does not care what caused the spike. It knows what normal volume looks like and yells when reality does not match.

If you only have an afternoon, build this one. It would have caught my original incident on its own.

THE HONEST VERSION

Where the AI earns its keep

And where it is an expensive way to do something simpler.

EARNS IT

- Turning a string of audit log ID soup into a sentence, eight hundred times a day, without getting bored
- Applying the same rubric to every entry the same way
- Writing a next step specific to this change, not generic advice

EXPENSIVE NOISE

- Detecting the change. The platform already did that.
- Resolving an ID you can look up. One API call beats a guess.
- Counting things. Flow 2 has no AI and never needed any.
- Five levels of risk when two will do.

The test I use: if a deterministic step can produce the answer, the model should never see the question.

WHY IT IS BUILT THIS WAY

I did not pick this stack because it is best.
I picked it because I am allowed to use it.

RULED OUT	**Homelab and Docker**	A work system touching production data does not belong on infrastructure my company cannot see or support. Not a technical limit. A judgment call.
SWAPPABLE	**Zapier, n8n, Make**	I set Zapier aside only because Power Automate was the approved path. The orchestration layer is not the interesting part. Any of them runs this same shape.
CHOSEN	**Power Automate + Azure OpenAI**	Both fully approved internally. Claude is approved for proof of concept only, not production, so Azure is the reasoning layer today.

ONE CAVEAT

These connectors need a premium Power Automate license.

If you already have one, this build adds no new line item.

SCAR TISSUE

Five things that ate real hours

1	**URI whitespace validation errors**	Build the URL in a Compose action first, then reference the Compose output in the HTTP call.
2	**Parse JSON breaking on type mismatch**	Real audit data has nulls where your sample did not. Make the schema null tolerant up front.
3	**Conditions failing on type**	A number arriving as a string does not compare the way you think. Wrap it in int() in the expression editor.
4	**Teams actions corrupting after a schema change**	No fix. Delete the action and recreate it. I lost an hour repairing one before rebuilding it in ninety seconds.
5	**"Edit in advanced mode" is gone**	The new designer removed it for conditions. You are not crazy. Use the fx expression tab on the individual operand.

Not on Power Automate? The tool changes, the failure class does not. Strings that should be numbers, schemas that assume your sample data, connectors that hold stale state. Build the URL in a variable first, tolerate nulls, rebuild rather than repair.

WHERE IT ACTUALLY STANDS

Running since June. Still rough in places.

WORKING

- Both flows have been live since June
- Unauthorized changes get caught at the source, not downstream in the pipeline
- One channel, no new tool for anyone to learn

HOW I TUNE THE FALSE POSITIVES

1. It rates one wrong, and I see it in Teams or in the list.
2. I do not touch the model. I add a line to the named rubric.
3. The ledger tells me whether it stopped happening.

I am telling you the rough parts on purpose. If I show you a finished system, you will assume you need one before you start. You do not. The rubric only covers what has burned us so far, and it grows every time this thing gets one wrong.

TAKE THIS HOME

I built it on HubSpot.

Nothing about the pattern is HubSpot.

Audit log

SWAP THIS BOX

→

Reasoning layer

UNCHANGED

→

Triage message

UNCHANGED

Salesforce

SetupAuditTrail

Marketo

Audit trail

Your CDP

Events endpoint

Anything

That records who changed what

The question to ask on Monday: which of our systems would hurt the most if somebody changed it quietly? Start there.

THE MAP, NOT THE FINISHED THING

Porting it: HubSpot to Salesforce

Four rows change. Two do not. And the two that do not are the ones that took longest to get right.

	HUBSPOT · BUILT	SALESFORCE · MAPPED
Event source	`account-info/v3/activity/audit-logs`	`SetupAuditTrail`
Auth	`Service Key, bearer token`	`Connected App, OAuth`
Window	`occurredAfter parameter`	`CreatedDate > last run`
Fields	`category, action, actingUser`	`Section, Action, CreatedBy, Display`
Reasoning layer	`Azure OpenAI + named rubric`	**Unchanged**
Ledger and alert	`SharePoint list + Teams`	**Unchanged**

Same three boxes. You are swapping the first one and renaming some fields.

THREE THINGS TO LEAVE WITH

If you remember nothing else

1

You already own the detector.

Any platform with an audit log and an API can do this. No governance tool, no new line item, nothing to rip out.

2

Use the model for reading, not for detecting.

It earns its keep translating and judging at volume. Everywhere else it is an expensive way to do something simpler.

3

Tune the rubric, not the model.

When it gets one wrong, add a line to the named list. The ledger tells you whether it worked. That is the whole loop.

Thank you

Happy to share the flow structure and the prompt. Just ask.

Jimmy Piraino

Senior Manager, Commercial Technology and Platforms · Accruent

jimmypiraino.com · **jimmy@jimmypiraino.com**

Or find me in the MO Pros Slack.