



GROW SMARTER.
BROKER BETTER.



EDITION · HUBSPOT

The AI Lead Reactivation Engine

An automated email, SMS and AI phone-calling system that works your dead CRM leads for you — and hands you back the ones that are ready to talk. Import two blueprints, connect five accounts, go live.

Setup time: 20–30 minutes

Import-and-go blueprints

HubSpot edition

v1.4 · August 2026

[START HERE](#)

What you are actually building

Every broker is sitting on a graveyard. Hundreds — often thousands — of people who once enquired about finance, never converted, and have not been contacted since.

This system goes back through them, one at a time, and restarts the conversation. It writes a personal email in your voice, follows up by SMS, and for the highest-intent leads it puts through an AI phone call. When someone replies, a second scenario reads the reply, works out what they meant, and either books them in, parks them, or removes them permanently.

You do not build any of it by hand. The two scenarios ship as **blueprint files**: upload a JSON file, connect your accounts, and the whole engine appears on the canvas already wired.

2 files

Blueprints to import

One sends. One listens.
Both included in this
pack, ready to upload.

5 accounts

Connections to make

Your CRM, OpenAI, your
mailbox, your SMS
provider, and Retell AI
for the calls.

~7 ops / lead

What it costs to run

Roughly seven Make
operations plus a
fraction of a cent of
OpenAI per lead
contacted.

COMPATIBILITY

It runs on the CRM you already have

The engine is CRM-agnostic. Only the first module and the last module touch your CRM — everything in between is identical no matter what you run. That is why one product covers the whole market.



Salesforce

NATIVE



HubSpot

NATIVE



Pipedrive

NATIVE



Zoho CRM

NATIVE



Salestrekker

API



Connective

API



LMG MyCRM

API



Effi

API



Finsure Infynity

API



ActivePipe

BRIDGE



BrokerEngine

BRIDGE



AFG Suite360

BRIDGE

Native — a ready-made connector, sign in and go

API — connects over the platform's API, subject to your aggregator granting access

Bridge — runs through a Google Sheets staging layer you export to

How the bridge works, in one paragraph

Closed platforms will not let an automation read them directly. So you export your dormant leads to a Google Sheet — a five-minute job you repeat monthly — and the engine reads the sheet instead of the CRM. Everything downstream is untouched: same AI, same email, same SMS, same AI call, same reply triage. The only thing you lose is the automatic write-back, which you replace with a second sheet the engine fills in and you re-import.

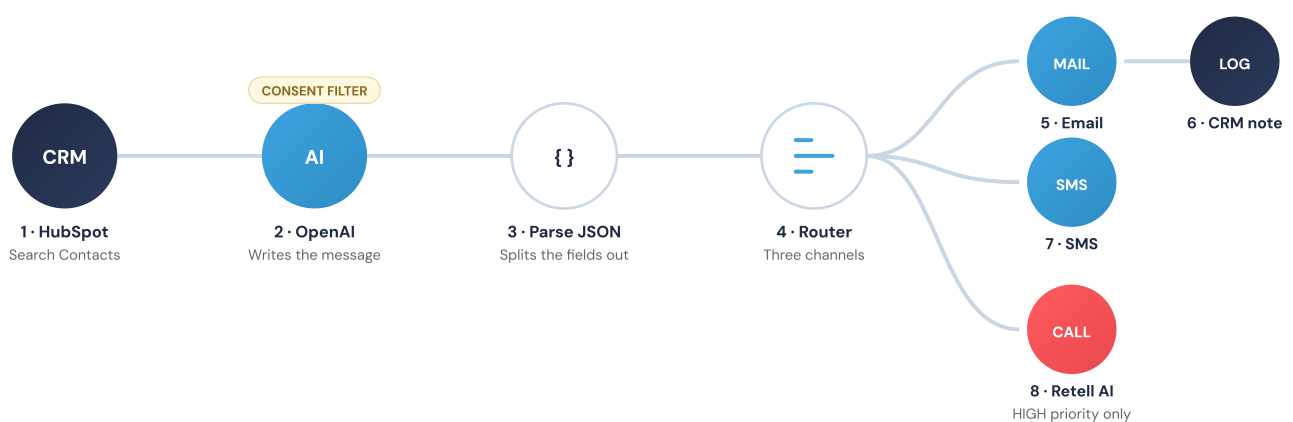
⚠ Check before you promise

API access to aggregator-owned platforms is granted by the aggregator, not the broker, and the answer differs by group and sometimes by brokerage. Confirm access is available for a given account before quoting a native build — and default to the bridge method, which works everywhere without anyone's permission.

SCENARIO 1

What lands on your canvas

Eight modules, one router, three channels. This is exactly what the blueprint builds.



SCHEDULE

Runs daily at 09:30, processing 25 leads per run — up to 750 leads a month, at a pace that protects your sender reputation.

Before you open Make

- ✓ **A Make account.** The free plan gives you 1,000 operations a month — enough to test with roughly 140 leads. The Core plan is where you will run it properly.
- ✓ **The two blueprint files** from this pack:
`01-TBT-Reactivation-Engine-HubSpot.blueprint.json` and
`02-TBT-Reply-Triage-HubSpot.blueprint.json`.
- ✓ **An OpenAI API key** with about \$10 of credit on it.
- ✓ **A HubSpot login** with permission to read contacts and write notes and deals.
- ✓ **A warmed sending mailbox** on your own domain, and an SMS account (Kudosity, Twilio or ClickSend) if you want the text channel.

⚠ Do this first — two HubSpot properties

The engine stores its campaign state on the contact record. Two custom Contact properties carry it, and everything else in this guide assumes they exist. Create them **before** you import anything, or the HubSpot modules will show as invalid.

In HubSpot go to **Settings** → **Properties** → **Create property**, object type **Contact**, and add these two. Use the internal names exactly as written — the blueprints are mapped to them.

- 1 Label **TBT reactivation state** · internal name **tbt_reactivation_state** · field type **Single-line text**. Holds one of **queued** **sent** **hot** **warm** **not_now** **opt_out**.
- 2 Label **TBT last reactivation** · internal name **tbt_last_reactivation_at** · field type **Date picker**. Enforces the 90-day frequency cap.

Why this matters more than it looks. **tbt_reactivation_state = opt_out** is the suppression flag. When somebody replies STOP, workflow 2 writes that value, and the consent gate on the outbound side refuses to pass any contact carrying it. That is the entire opt-out mechanism. Without these two properties there is nothing stopping the engine contacting someone who has already asked you not to.

⚠ One rule before you touch anything

This system contacts real people who gave you their details, and only those people. It is not a cold-outreach tool and must never be pointed at a purchased list. Every message carries an opt-out, and the opt-out is honoured automatically and permanently.

THE BUILD

Scenario 1 — the outbound engine

Import first, then connect. Do not build this by hand; the blueprint already contains every prompt, filter and mapping.

Upload the blueprint

- 1 In Make, go to **Scenarios** and click **Create a new scenario**.
- 2 On the empty canvas, click the ... **(three dots)** in the bottom toolbar.
- 3 Choose **Import Blueprint**.
- 4 Select **01-TBT-Reactivation-Engine-HubSpot.blueprint.json** and click **Save**.

The whole engine appears at once — nine modules, the router, all three filters and every prompt already written. Every module will show a small warning triangle because it has no connection yet. That is expected and is what step 2 fixes.

If the import is rejected

Make refuses blueprints that reference an app your account has not installed. Open the file, confirm it is valid JSON (no stray characters if you opened it in a text editor), and check that HubSpot CRM, OpenAI and your SMS app all appear when you search for them in Make. Install any that do not.

Connect your five accounts

Click each module in turn. Where it says *Connection*, click **Add** and sign in. Work left to right.

MODULE	APP	WHAT IT NEEDS
1 · Search Contacts	HubSpot CRM	OAuth sign-in to your portal
2 · Create Completion	OpenAI	Your API key, pasted into the API Key field
5 · Send an Email	Gmail	OAuth sign-in — pick the mailbox replies should land in
6 · Make an API Call	HubSpot CRM	Re-use the connection from module 1
7 · Send SMS	Your SMS provider	API key and secret from the provider's dashboard
8 · HTTP request	Retell AI	No connection — paste your key into the Authorization header

Gmail needs the restricted scope

When you connect Gmail, Make asks for a Google connection with send permission. Grant it — without it module 5 fails silently at run time with a permissions error that is easy to misread as a mapping problem.

Point module 1 at your dormant leads

Open **module 1 • Search Contacts**. The blueprint ships with an empty query and a limit of 25. Two things to set:

- 1 **Limit** — leave at **25** to start. This is how many leads get contacted per run. Raise it once you trust the output.
- 2 **Filter** (under advanced settings) — this is your definition of “dormant”. A good starting point: *Create Date* is before six months ago **AND** *Lifecycle Stage* is not **customer**.

The simpler alternative

If HubSpot search filters give you trouble, build a static list in HubSpot called *Dormant Leads — Reactivation*, then swap module 1 for **HubSpot → Get a List's Contacts**. Same result, and you can eyeball exactly who is in the campaign before it runs.

Make it sound like you

Open **module 2 · Create Completion**. It contains two messages. The **system** message is the rulebook — leave the compliance rules alone, but adjust the tone if you want. The **user** message is the lead's file, and it contains three things you must change:

FIND THESE LINES IN THE USER MESSAGE AND REPLACE THEM

EDIT

```
Broker business name: [YOUR BUSINESS NAME]
Broker first name: [YOUR FIRST NAME]
Opt-out wording for SMS: Reply STOP to opt out.
```

The prompt already forces the model to return a clean JSON object with five fields, which is why module 3 can split it into a subject, an email body, an SMS and a call opening line without any extra work:

WHAT THE AI RETURNS, EVERY TIME

REFERENCE

```
{
  "email_subject": "...",
  "email_body_html": "...",
  "sms_text": "...",
  "call_opening_line": "...",
  "priority": "HIGH" | "MEDIUM" | "LOW"
}
```

The priority field is doing real work

The AI reads the lead's history and scores their likely intent. Module 8 only fires when that score comes back **HIGH**, so your AI phone calls — the expensive channel — are spent on the leads most likely to answer. Change the filter on module 8 to **HIGH or MEDIUM** if you want more volume.

Set the sender details

Three quick fields across the three router branches:

MODULE	FIELD	SET IT TO
5 · Send an Email	From	Your name and sending address, e.g. <code>Matt at Broker Times <matt@yourdomain.com.au></code>
7 · Send SMS	From	Your registered sender ID or virtual number. Leave blank to use the account default.
8 · HTTP request	Headers & body	Replace <code>YOUR_RETELL_API_KEY</code> , <code>YOUR_RETELL_NUMBER</code> and <code>YOUR_RETELL_AGENT_ID</code> with the values from your Retell dashboard.

Everything else — the recipient, the subject, the message body, the SMS text, the call script variables — is already mapped from the AI output. Do not re-map them.

6

PROVE IT

Run it once, on yourself

- 1 Add yourself to HubSpot as a test contact with your real email and mobile.
- 2 Set module 1's limit to `1` and its filter to match only your test contact.
- 3 Click `Run once` at the bottom left.
- 4 Watch the bubbles appear over each module. Click any bubble to see exactly what went in and what came out.

You should receive an email and a text within a minute, and see a new note on your HubSpot contact. **Read the email properly.** If the tone is not yours, go back to module 2 and adjust the system message — that is where the voice lives.

✓ Then, and only then

Set the schedule to **Every day at 09:30**, put the limit back to 25, restore your real dormant filter, and switch the scenario **ON**.

Scenario 2 — the reply handler

Scenario 1 starts conversations. This one is where the money is: it reads every reply and decides what happens next, in seconds, at any hour of the night.

1

IMPORT

Upload the second blueprint

Same process: new scenario → ... → **Import Blueprint** → `02-TBT-Reply-Triage-HubSpot.blueprint.json`.

You get a webhook trigger, an AI classifier, and a four-branch router: **HOT**, **WARM**, **NOT NOW** and **OPT OUT**.

2

PLUMBING

Create the webhook and wire your channels into it

- 1 Open **module 1 · Custom webhook** and click **Add**. Name it `Reactivation replies`.
- 2 Make gives you a URL like `https://hook.eu1.make.com/xxxxxxx`. Copy it.
- 3 Paste it into your SMS provider's **inbound reply callback** setting.
- 4 Paste it into your Retell agent's **webhook URL** so call outcomes come back too.
- 5 For email replies, add a second scenario with a Gmail **Watch Emails** trigger that POSTs to the same URL.

THE PAYLOAD THE SCENARIO EXPECTS

REFERENCE

```
{
  "contact_id": "HubSpot contact ID",
  "first_name": "Sarah",
  "email": "sarah@example.com.au",
  "mobile": "+61400000000",
  "channel": "sms" | "email" | "call",
  "message": "the exact words they replied with"
}
```

Send one test POST with that shape, and Make will learn the data structure automatically.

Four branches, four outcomes

BRANCH	WHAT FIRES	WHAT YOU SHOULD CHANGE
HOT	Creates a HubSpot deal and emails you the reply plus the AI's suggested next action	Set the alert email address and your pipeline's first stage ID
WARM	Lead status stays <code>OPEN</code> , writes <code>tbt_reactivation_state = warm</code> and stamps today's date	Point it at your nurture list if you run one
NOT NOW	Lead status stays <code>OPEN</code> , writes <code>tbt_reactivation_state = not_now</code> and stamps today's date so your dormancy rule skips them for 90 days	Change 90 days to whatever suits your book, in module 1's filter
OPT OUT	Writes <code>tbt_reactivation_state = opt_out</code> and sets lead status <code>UNQUALIFIED</code> . Both are conditions on the consent gate, so this contact can never be selected again	Nothing. Do not weaken this branch

⚠ How suppression actually works

The OPT OUT branch does not delete anybody and it does not touch HubSpot's own `hs_email_optout` property, which the API cannot write to. It sets `tbt_reactivation_state` to `opt_out`, and the consent gate in scenario 1 refuses to pass any contact carrying that value. The SMS and phone branches re-check it independently, so a suppression cannot leak through the router.

This is why the two custom properties are not optional. If `tbt_reactivation_state` does not exist, this branch writes nothing and the gate has nothing to read.

✓ Test the opt-out branch first

POST a test payload where `message` is just "STOP", confirm it takes the OPT OUT branch, then open the contact in HubSpot and check `tbt_reactivation_state` now reads `opt_out`. Then run scenario 1 again and confirm that contact is **not** picked up a second time. That last step is the one that matters.

Turning it on without blowing anything up

- 1 **Both scenarios ON**, but module 1's filter still pointed at your test contact only.
- 2 **Reply to your own messages** — one “yes please call me”, one “not right now”, one “STOP”. Confirm all three land in the right branch.
- 3 **Open the filter to ten real leads**. Read every message that goes out. You are checking tone, not mechanics.
- 4 **Scale in steps**: 25 a day for week one, 50 for week two, then whatever your inbox can handle.
- 5 **Watch the History tab** for the first fortnight. Every run, every module, every payload is logged there.

What it costs to run 1,000 leads

Roughly 5,250 Make operations, a few dollars of OpenAI, plus whatever your SMS provider charges per message and Retell charges per call minute. The leads themselves cost nothing — they are already sitting in your CRM. Run your first batch, measure your own reply rate, and use those numbers from then on rather than anybody's claimed averages.

Troubleshooting

SYMPTOM	CAUSE	FIX
Module 3 errors on Parse JSON	The model wrapped its answer in a code fence	Confirm <i>Response format</i> on module 2 is set to <code>json_object</code>
Scenario runs, sends nothing	The consent filter on module 2 is blocking everything	Click the filter bubble in History — it shows exactly which condition failed
Email module fails with 403	Gmail connection lacks send scope	Delete the connection and re-add it, accepting all requested permissions
SMS branch never runs	Mobile numbers stored in <code>phone</code> , not <code>mobilephone</code>	Edit the filter on module 7 to match your CRM's actual field
Retell returns 401	Missing the word Bearer	The header value must read <code>Bearer sk- ...</code> , with the space
Operations burning fast	Limit set too high, or scenario scheduled every 15 minutes	Daily schedule, limit 25. That is up to 750 leads a month on roughly 5,000 operations.
Webhook never fires	Scenario 2 is switched off, or the URL was pasted with a trailing space	Turn it on, re-copy the URL, send a test POST

ONE LAST THING

Prefer this built for you?

The same engine, configured against your CRM, your voice and your compliance settings — installed and tested, with the first batch running before you hang up.

Talk to The Broker Times



AI Lead Reactivation Engine · Make.com Edition · v1.1

Third-party names and logos are the property of their respective owners and are shown to indicate compatibility only.

Not financial, legal or compliance advice. Check your own obligations under the Spam Act 2003, the Privacy Act 1988 and the Do Not Call Register Act 2006.