

THE BROKER TIMES

GUIDE 3 OF 5

OpenRouter

AI WhatsApp Agent Engine · thebrokertimes.com.au

OpenRouter is the gateway between your scenario and the AI model. One account, one API key, and access to essentially every major model — so you can change your mind later without touching the WhatsApp integration.

Time: 10 minutes.

Step 1 — Account and key

1. Go to openrouter.ai and sign up
2. **Settings** → **Credits** → add a small amount to start. Twenty dollars is plenty to build and test with.
3. **Settings** → **API Keys** → **Create key**
4. Name it "WhatsApp Agent". Copy it — it starts with `sk-or-v1-`

Optionally set a **credit limit** on the key. Do it. If something loops, the limit is what stops it becoming an expensive lesson.

Step 2 — Put the key in Make

In the scenario, open **module 32 (HTTP — Make a request)**, find the `Authorization` header and replace the placeholder so it reads:

```
Bearer sk-or-v1-your-actual-key-here
```

The word `Bearer` and the space stay. Only the key changes.

Step 3 — Choosing a model

The blueprint ships pointed at a fast, cheap model. Change it in the same HTTP module — it's the `"model"` line at the top of the request body.

What matters for this job:

- **Speed.** This is a live chat. A model that takes eight seconds feels broken.

- **Instruction-following.** Your guardrails are only as good as the model's willingness to obey them. This is where cheap models fail — not on writing ability, but on quietly ignoring "never quote a rate".
- **Reliable JSON.** The scenario parses the response as JSON. The request already sets `response_format: json_object`, which most current models honour.

Practical approach: build and test on a fast cheap model. Then run your ten nastiest test messages past a mid-tier model and compare. If the cheap one holds the guardrails, keep it. If it slips even once, pay the extra — a broker's licence is worth more than the token difference.

Model names on OpenRouter look like `anthropic/claude-haiku-4.5` or `openai/gpt-4.1-mini`. The current catalogue and live per-token pricing is at openrouter.ai/models. Check it rather than trusting any figure printed in a guide, including this one.

Step 4 — Controlling cost

Three dials, in order of effect:

1. **History length.** Module 28's limit is set to 15 messages. Every message you keep is re-sent to the model on every turn. Dropping to 10 cuts your token bill noticeably and rarely hurts quality on short enquiries.
2. **Knowledge base size.** It is sent on every single message. Keep it tight and factual. Delete anything the agent doesn't actually need.
3. `max_tokens`. Set to 700 in the blueprint. WhatsApp replies should be short anyway. Lowering it caps the worst case.

Step 5 — Check it works

You can test the exact call before wiring up WhatsApp:

```
curl https://openrouter.ai/api/v1/chat/completions \
  -H "Authorization: Bearer sk-or-v1-your-key" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "anthropic/claude-haiku-4.5",
    "response_format": {"type": "json_object"},
    "messages": [
      {"role": "system", "content": "Reply only with JSON: {\"reply\": \"...\", \"stage\": \"WARM\", \"ha"},
      {"role": "user", "content": "hi, do you do first home buyers?"}
    ]
  }'
```

If that returns JSON, your key and model name are both good, and any later failure is in Make rather than OpenRouter.