marshal

TECHNICAL WHITE PAPER

Marshal Agents

A technical white paper describing the design of Marshal Agents that autonomously execute defined SMB workflows across a customer's own tools, governed by human approvals, gateways, and audit-ready control.

AUDIENCE
Founder-led SMBs

SCOPE
Managed AI Ops

DESIGN CENTER
Existing stack, human judgment

Not a chatbot. Not a Zap with nicer shoes. A managed operating layer for work that has consequences.

marshal

00 / CONTENTS

Table of contents

ESSENTIAL CLAIM

Marshal Agents are production workflow systems for SMB teams. They read from the customer's tools, reason over a defined workflow, route through approval gates, execute bounded actions, and leave a trace humans can inspect when reality does what reality enjoys doing.

marshal

01 / INTRODUCTION

From chat to work

The first wave of business AI taught everyone to ask questions. The next wave is less polite. It executes work.

For SMBs, the useful question is not whether a model can write a charming paragraph about pipeline hygiene. The useful question is whether it can notice a new lead, enrich the record, classify urgency, draft the right response, route it to the right human, update the CRM, and prove what happened afterward.

Marshal Agents are built for that second problem. They are not enterprise AI scaled down until the slide deck stops groaning. They are workflow-level systems designed for founder-led businesses that already run on CRM, email, calendar, Slack, spreadsheets, project management tools, billing systems, and a heroic amount of copy-paste.

Design thesis

An agent should be autonomous only inside a bounded workflow. The wider the autonomy, the tighter the controls. This is not cowardice. This is how adults ship systems that touch customers, revenue, and records of truth.

03	09	6-12
PRODUCTION LINES	AGENT TYPES	DAY DEPLOYMENT TARGET

Marshal's Agent Factory is organized around three production lines and nine agent types for lead capture, revenue generation, and operational throughput.[1]

marshal

02 / DEFINITION

What a Marshal Agent is

A Marshal Agent is a managed AI workflow executor that converts a defined business goal into a controlled sequence of tool calls, approvals, and recorded outcomes.

1

Goal

A specific job to be done, such as respond to a qualified inbound lead inside five minutes.

2

Context

Relevant data retrieved from the customer's tools, permissions, business rules, history, and current state.

3

Workflow

A constrained process map with branching, looping, routing, and stopping conditions.

4

Tools

Read, write, analysis, communication, scheduling, enrichment, and notification interfaces.

5

Control

Policy checks, approval gates, exception queues, audit logs, monitoring, and kill switches.

6

Learning

Evaluation, feedback, corrections, and workflow improvements over repeated runs.

The best agent architectures stay simple, composable, transparent, and measurable before they get clever. Anthropic's agent engineering guidance makes the same point: start with the simplest useful pattern, then add agentic complexity only where flexibility beats predictability.[3]

marshal

03 / SMB DESIGN CENTER

Small businesses do not need an agent zoo

Marshal does not design for a Fortune 100 maze where every action needs a committee, a sandbox committee, and a committee to approve the sandbox committee.

SMBs need agents that take recurring operational load off real teams. The stack is smaller, but the consequences are not. A lead can still be lost. A client can still be onboarded badly. A CRM can still become archaeological evidence of poor management.

That is why Marshal starts with narrow workflows, clear owners, measurable outcomes, and existing tools. The goal is not to replace the business system. The goal is to make the current system behave like someone competent is watching it all day.

Existing stack first

No rip-and-replace. The agent works across the CRM, inbox, calendar, project tools, docs, and finance systems already in use.

Human judgment stays human

Approvals, exceptions, and high-impact decisions route to people before action.

Workflow before autonomy

Agents execute defined process maps and adapt only inside the approved boundaries.

Measured in outcomes

The useful metrics are response time, meeting booked, record updated, cycle time reduced, and human review rate.

The SMB advantage is that the business is still knowable. Marshal turns that knowledge into an operating system before the duct tape becomes load-bearing infrastructure.

marshal

04 / AGENT FACTORY

Three production lines, nine agent types

Marshal's architecture is organized around the workflows founder-led teams feel first: inbound demand, outbound revenue work, and operational handoff after the sale.[1]

Lead Capture System

Make inbound demand survivable.

Speed-to-Lead

Responds to new inbound leads, enriches, qualifies, drafts, routes, and logs.

Qualification & Routing

Scores against ICP and buying readiness, assigns ownership, and prepares clean handoff.

Booking & Follow-Up

Coordinates scheduling, reminders, no-show recovery, and CRM updates.

Revenue Generation System

Turn prospecting from artisanal suffering into a system.

Prospect Identification

Finds and enriches candidates from ICP rules while suppressing duplicates.

Account & Personalization

Researches account context, triggers, buying signals, and one-screen briefs.

Outbound Execution Support

Drafts sequences, classifies replies, logs touches, and escalates qualified responses.

Operational Throughput System

Stop using humans as API glue.

Client Intake & Onboarding

Collects intake data, creates tasks, tracks gaps, and prepares internal handoff.

Data Sync & Admin Relay

Synchronizes records across CRM, spreadsheets, project tools, and finance systems.

Reporting & Decision Support

Assembles KPI rollups, trend reads, recap drafts, and decision briefs.

marshal

05 / REFERENCE ARCHITECTURE

Marshal agentic operating layer

Marshal Agents sit above the customer's tools and below the human decision layer. The agent is not the system of record. It is the controlled execution layer.

Human owners
founder, sales, ops, client success

Approval gates
review, approve, reject, edit

Audit console
runs, traces, exceptions, metrics

Marshal Agentic Execution Engine

- Trigger router
- Context builder
- Workflow planner
- Policy gateway
- Tool executor
- Evaluator

CRM
HubSpot, Salesforce

Inbox
Gmail, Outlook

Calendar
Calendly, Google

Comms
Slack, SMS, chat

Docs
Notion, Drive

Ops
Asana, Sheets, QuickBooks

Every side effect is governed by scope, credential permissions, workflow rules, and human approval requirements. This is how an agent gets useful without becoming a tiny unsupervised intern with API keys.

marshal

06 / EXECUTION ENGINE

From trigger to recorded outcome

A Marshal run is a bounded transaction: detect, plan, check, act, verify, log, and learn. The words are boring because the work should be boring. Boring is what you want near customer records.

1

Trigger

Event, schedule, manual request, webhook, inbox pattern, CRM state change.

2

Assess

Classify workflow, owner, priority, confidence, and whether direct execution is safe.

3

Retrieve

Pull context from approved sources, respecting access and business rules.

4

Plan

Generate a step plan from the workflow spec, including branches and gates.

5

Execute

Call tools with fixed schemas and record every input, output, and decision.

6

Gate

Pause for human review when required by policy, confidence, or impact.

7

Commit

Write approved changes to systems of record using idempotent operations.

8

Evaluate

Score outcome, capture feedback, flag drift, and update operating memory.

The architecture follows the common agent loop of plan, act, observe, adjust, and repeat, but it narrows the loop inside explicit workflow constraints and human checkpoints.[4]

marshal

07 / WORKFLOWS

Workflows are the agent's rails

A workflow specification is the executable contract between the business process and the model.

It names the trigger, required context, allowed tools, branching rules, approval gates, side effects, retry policy, exception routes, and success metrics. This turns a vague instruction like "handle inbound leads" into a process that can be tested, monitored, and improved.

Prompt chaining, routing, parallel checks, evaluator loops, and orchestrator-worker patterns are useful because they make complex tasks inspectable. They also create places to place gates, which is the part humans pretend they do not need until something expensive happens.

[3]

```
workflow_id: lead_capture.speed_to_lead.v2
owner: revenue_ops
trigger: inbound.demo_request.created
read_scopes:
  - crm.leads.read
  - enrichment.company.read
  - inbox.thread.read
write_scopes:
  - crm.leads.update
  - slack.notify
  - calendar.link.create
human_gates:
  - external_message.send when confidence <
0.92
  - route_change when account_value = high
idempotency_key: lead_id + workflow_version
success_metrics:
  - first_response_time
  - meeting_booked
  - approval_rate
  - exception_rate
```

Durable execution patterns, especially idempotent side effects, make safe retries possible when tools fail, rate limits bite, or the internet behaves like a group project.[10]

marshal

08 / TOOLS

Tools make reasoning useful

Models reason. Tools do. The tool layer is where Marshal turns language into controlled interaction with the customer's stack.

Read tools

Retrieve CRM records, inbox threads, calendar state, form submissions, documents, tickets, tasks, call notes, spreadsheets, and billing data.

Write tools

Create or update CRM objects, draft messages, assign tasks, create project records, notify Slack, schedule meetings, and move workflow state.

Analysis tools

Score fit, dedupe records, summarize threads, compare fields, generate KPI tables, calculate deltas, and produce decision briefs.

Gateway tools

Request approval, route exceptions, lock a record, log a run, issue rollback instructions, and escalate to a human owner.

Tool contract

- Name and purpose
- Input schema
- Output schema
- Allowed scopes
- Credential owner
- Side-effect level
- Timeout and retries
- Idempotency key
- Rollback path
- Audit fields

MCP's resource, prompt, and tool model is a useful reference pattern for connector design because it separates context, reusable prompts, and executable functions.[8]

marshal

09 / CONTEXT AND MEMORY

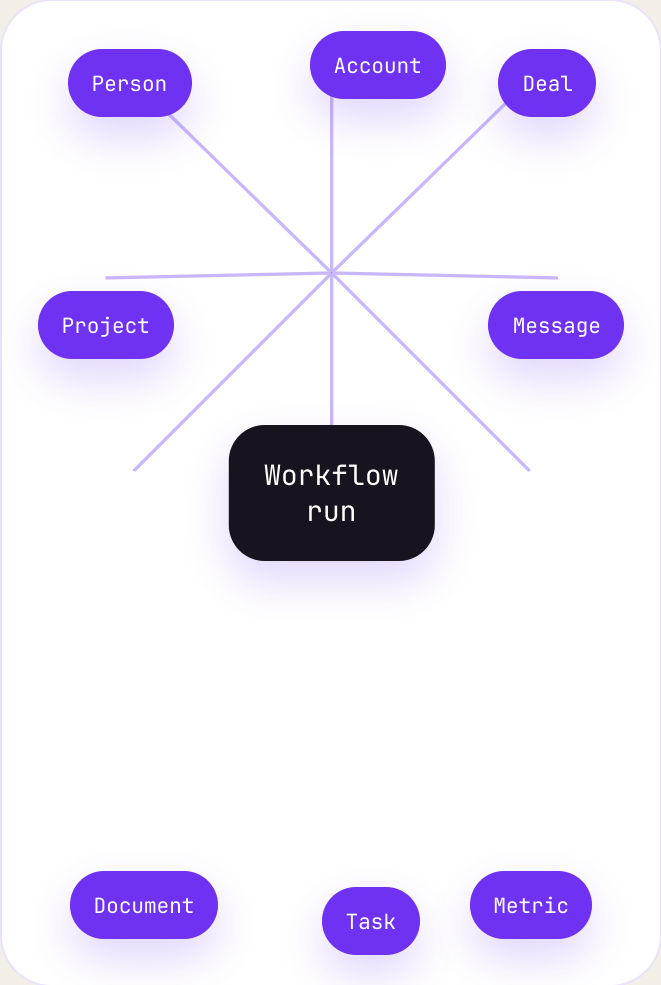
A living map of the business

SMBs do not need a moonshot knowledge graph. They need their CRM, inbox, docs, calendar, and project tools to stop acting like divorced parents.

Marshal builds task-specific context from the customer's existing systems. The agent retrieves only what the workflow needs: account history, recent messages, deal stage, meeting availability, onboarding checklist, project ownership, support state, and any business rules attached to the workflow.

Memory is not a mystical brain jar. It is structured feedback from runs: approved drafts, rejected routes, recurring exceptions, owner preferences, tool failures, and performance deltas. Retrieval-augmented generation keeps responses and decisions grounded in the customer's current data instead of model folklore.

[14]



marshal

10 / GOVERNANCE

Control is not a feature. It is the product boundary.

Marshal governance is built around three questions: what can the agent see, what can it do, and when must a human sign off?[2]

Permissions

Inherited access, least privilege, credential scopes, owner-controlled revocation.

Policy gateway

Checks workflow scope, tool scope, user access, data class, and action class before execution.

Human approval

Required for external messages, irreversible writes, high-value routing, ambiguous records, or low confidence.

Exception queue

Routes anomalies to the responsible person with context, recommendation, and next action.

Autonomy ladder

0	Observe only
1	Draft for review
2	Execute after approval
3	Execute low-risk actions
4	Execute with sampling review
5	Full autonomy inside scope

NIST and OWASP both emphasize that trustworthiness and risk management have to be designed into AI systems, not sprinkled on afterward like compliance parsley.[6][7]

marshal

11 / GATES AND GATEWAYS

Where humans stay in command

A gate is a deliberate pause. A gateway is an enforceable control. Marshal uses both.

Control	When it triggers	Human action	System record
Draft approval	External-facing message, proposal note, client recap, outbound sequence	Approve, edit, reject, request rewrite	Prompt, source context, draft, reviewer, final text
Write approval	CRM stage movement, task creation, owner change, finance sync, data overwrite	Approve or reject write	Old value, new value, tool call, idempotency key
Risk gateway	Low confidence, missing field, policy conflict, high-value account, sensitive data	Resolve exception	Reason code, confidence, recommended next step
Commit gateway	After all required approvals are satisfied	No manual action unless policy requires	Final write, timestamp, result, rollback note

This is the operating pattern: let the machine handle repetition, search, drafting, routing, and synchronization. Keep humans on judgment, exceptions, high-impact changes, and final accountability.

marshal

12 / NINE AGENT TYPES

Lead Capture System

Inbound demand is expensive to create and embarrassingly easy to waste. Lead Capture agents keep the revenue front door from becoming a haunted mailbox.

Speed-to-Lead

Watches forms, chat, demo requests, inbound emails, or partner referrals. It enriches the lead, checks fit, drafts or sends an on-brand response, notifies the owner, creates the CRM activity, and prepares the meeting context.

- Triggers: new inquiry, demo form, reply, chat event
- Reads: form fields, CRM, enrichment, inbox history
- Writes: CRM update, Slack alert, draft response
- Gates: external send, high-value account, low fit ambiguity

Qualification & Routing

Scores fit and intent, applies routing rules, detects ownership conflicts, assigns the next human, and packages the handoff. This replaces the sacred SMB ritual of arguing in Slack about who owns a lead.

- Triggers: lead created, status changed, enrichment complete
- Reads: ICP rules, territory, account ownership, past activity
- Writes: owner, score, lifecycle stage, handoff note
- Gates: territory conflict, strategic account, duplicate uncertainty

Booking & Follow-Up

Coordinates scheduling, reminders, reschedules, no-show recovery, and post-booking CRM hygiene. The agent does not close deals. It prevents the calendar from quietly murdering them.

- Triggers: qualified lead, meeting not booked, no-show
- Reads: calendars, routing owner, lead preferences
- Writes: booking link, reminder, CRM task, recap
- Gates: unusual request, custom scheduling constraint

marshal

12 / NINE AGENT TYPES

Revenue Generation System

Outbound work fails when research, personalization, sequencing, and reply handling live in four tabs and someone's optimism.

Prospect Identification

Builds candidate lists from ICP rules, enriches contacts and companies, suppresses existing records, detects conflicts, and creates reviewable prospect pools.

- Triggers: campaign brief, schedule, list refresh
- Reads: ICP, CRM, enrichment, exclusions, territory
- Writes: prospect list, score, source, suppression reason
- Gates: uncertain match, duplicate, high-risk source

Account & Personalization

Turns accounts into briefs: firmographic context, trigger events, likely pain, existing relationships, and draft personalization hooks. It gives humans leverage without requiring them to become LinkedIn archaeologists.

- Triggers: prospect approved, target account added
- Reads: website, CRM history, public signals, notes
- Writes: account brief, personalization line, talking points
- Gates: unsupported claim, missing source, sensitive inference

Outbound Execution Support

Drafts email and LinkedIn copy, tracks touches, categorizes replies, logs actions, and escalates interest, objections, and meeting-ready responses.

- Triggers: approved prospect batch, reply received
- Reads: sequence rules, brief, CRM, reply thread
- Writes: draft message, touch log, reply category, task
- Gates: external send, objection requiring judgment

12 / NINE AGENT TYPES

Operational Throughput System

After a deal closes, the business either runs a system or it runs a scavenger hunt. These agents reduce the scavenger hunt.

Client Intake & Onboarding

Collects intake data, checks completeness, creates onboarding tasks, prepares internal handoff, tracks missing assets, and keeps the client from falling into the void between sales and delivery.

- Triggers: deal closed, agreement signed, invoice paid
- Reads: CRM, contract, form, project templates
- Writes: project, tasks, checklist, client note
- Gates: missing contract, custom scope, unclear owner

Data Sync & Admin Relay

Keeps records aligned across CRM, sheets, project tools, finance tools, and dashboards. Humans should not be the integration layer unless the business is trying to reenact medieval bookkeeping.

- Triggers: record change, scheduled reconciliation, discrepancy
- Reads: source systems, mapping rules, audit history
- Writes: updates, alerts, reconciliation report
- Gates: destructive overwrite, conflicting system values

Reporting & Decision Support

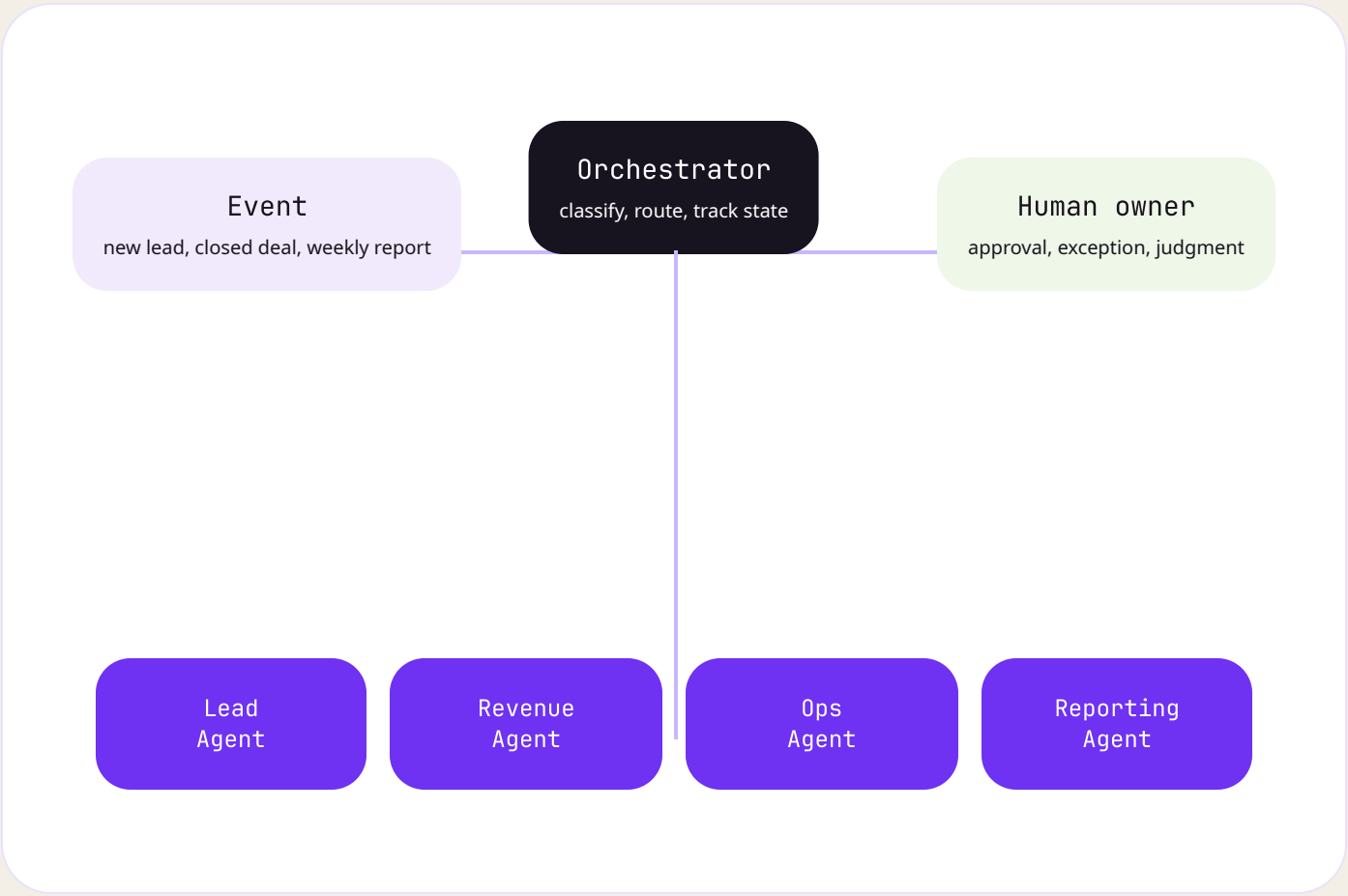
Assembles KPI rollups, trend reads, variance explanations, client recap drafts, and decision briefs. It prepares the room. Humans still make the call.

- Triggers: weekly report, client update, owner request
- Reads: CRM, project tool, finance, analytics, notes
- Writes: report draft, dashboard table, summary, alerts
- Gates: recommendation, ambiguous metric, financial interpretation

13 / ORCHESTRATION

Agents collaborate like a small team

The goal is not one giant genius agent. That way lies latency, confusion, and a troubleshooting session that resembles an exorcism. Marshal composes specialized agents with clear ownership.



Sub-agents allow specialization. A lead routing agent should not also invent onboarding checklists and reconcile finance fields. Small agents with explicit contracts are easier to test, monitor, replace, and improve.

Open agent communication standards such as Agent2Agent are emerging to let agents coordinate across tools and vendors. Marshal's architecture treats this as a design direction while keeping SMB workflows grounded in today's actual stack, not tomorrow's conference booth.

[9]

marshal

14 / SIDE EFFECTS

Writes require discipline

Reading is reversible. Writing is where the bill comes due.

Draft

No system-of-record mutation. Human reviews text or recommendation.

Propose

Agent suggests a change with old value, new value, source, and reason.

Approve

Human accepts, edits, rejects, or reroutes the action.

Commit

Tool executor performs the write with idempotency key and audit fields.

Verify

Agent reads back the destination system to confirm effect.

Compensate

If needed, the run records rollback or follow-up instructions.

The rule

The more visible, irreversible, financial, customer-facing, or data-destructive the action is, the more conservative the gateway. This is the difference between automation and a bot with a loaded crossbow.

OWASP's excessive agency category exists because agents with too much functionality, permissions, or autonomy can cause damage when instructions are ambiguous or manipulated.[7]

marshal

15 / MODEL ROUTING

Use the right model for the job

Model choice is an execution decision, not a personality quiz. A routing task does not need the same model as a nuanced client recap or a multi-source account brief.

Workflow step		
Classification	fast model	Cheap, low-latency, repeatable labels.
Retrieval synthesis	general model	Summarize grounded context with citations and constraints.
Complex reasoning	strong model	Plan branches, resolve ambiguity, draft high-stakes text.
Evaluation	judge model	Score correctness, completeness, grounding, and policy adherence.

Marshal designs for model portability. Workflows should not be fused to one provider, one prompt style, or one benchmark moment. Models change. Business processes should not panic every time a leaderboard updates.

Routing and specialization are standard patterns for reducing cost and latency while preserving quality. The agent should escalate only when the workflow actually needs deeper reasoning or more context.

[3]

marshal

16 / EVALUATION

Evaluate the work, not the vibes

An agent can sound competent while quietly taking the wrong step. This is also true of humans, but at least humans buy lunch sometimes.

Golden tasks	Sandbox runs	Step graders	Outcome graders	Regression suite	Production monitoring
known scenarios	full transcripts	tool calls and gates	business result	before promotion	drift and failures

Tool selection accuracy	Approval rate	Exception rate	Latency
Cost per run	First response time	Meeting booked rate	Record sync accuracy
Human edit distance	Rollback rate	Source coverage	Grounding score

Strong agent evals combine end-state checks, tool-call inspection, transcript review, human grading, model grading, and regression baselines. Small task sets can be useful early when they are precise and representative.[5][13]

marshal

17 / OBSERVABILITY

Every run should explain itself

When an agent touches business systems, "the model decided" is not an acceptable postmortem. That sentence belongs in the same landfill as "we'll circle back."

```
run_id: 2026-06-06T12:00:33Z-lead-8432
workflow_id: lead_capture.speed_to_lead.v2
trigger: demo_request.created
context_sources:
  - hubspot.contact: read ok
  - enrichment.company: read ok
  - gmail.thread: no prior thread
plan_steps: [enrich, score, draft, gate, notify, update]
tool_calls:
  - enrich_company: success, 821 ms
  - crm_update: pending approval
human_gate:
  reviewer: sales_owner
  decision: approved_with_edits
outcome:
  first_response_time: 00:03:41
  meeting_status: pending
  eval_score: 0.91
```

Observability covers traces, metrics, logs, transcripts, tool calls, policy decisions, approvals, errors, and outcome metrics. It lets Marshal find the exact step that failed instead of staring at the final output like a pilgrim awaiting revelation.

OpenTelemetry defines a vendor-neutral pattern for traces, metrics, and logs, while modern agent frameworks increasingly include tracing for model calls, tools, handoffs, and guardrails. The point is the same: emit telemetry or enjoy guessing.

[11][12]

marshal

Security model for SMB agent operations

SMBs do not need enterprise theater. They need practical controls that map to the actual risk of agents reading and writing across business systems.

Least privilege

Each workflow gets only the read and write scopes it needs. Credentials remain owner-controlled and revocable.

Permission checks

Every request is checked against user access, workflow scope, tool scope, and data class before action.

Prompt injection defense

Untrusted content is treated as data, not instruction. Tool calls are gated by policy, not by whatever text appeared in an email.

Output handling

Generated text and structured outputs are validated before use in downstream systems.

Human review

Required for sensitive data, financial actions, public messages, high-value accounts, and low-confidence decisions.

Run controls

Audit logs, alerting, exception queues, rollback notes, and kill switches support managed operation.

Trustworthy agent design spans user control, values, security, transparency, and privacy. OWASP specifically calls out prompt injection, sensitive information disclosure, improper output handling, and excessive agency as LLM application risks.[4][7]

marshal

Deployment lifecycle

Marshal ships one workflow first. Not because ambition died. Because production systems should earn their territory.

1

Scope

Pick a recurring workflow with visible pain, measurable outcome, clear owner, and accessible tools.

2

Map

Document triggers, systems, fields, business rules, approvals, exceptions, and outputs.

3

Build

Create tool contracts, workflow spec, prompt units, routing logic, and sandbox dataset.

4

Test

Run golden scenarios, edge cases, permission checks, failure paths, and human review flows.

5

Pilot

Deploy with conservative gates, monitor run logs, gather feedback, measure outcomes.

6

Operate

Move autonomy upward only when metrics and human trust justify it. Managed means someone watches the system after launch.

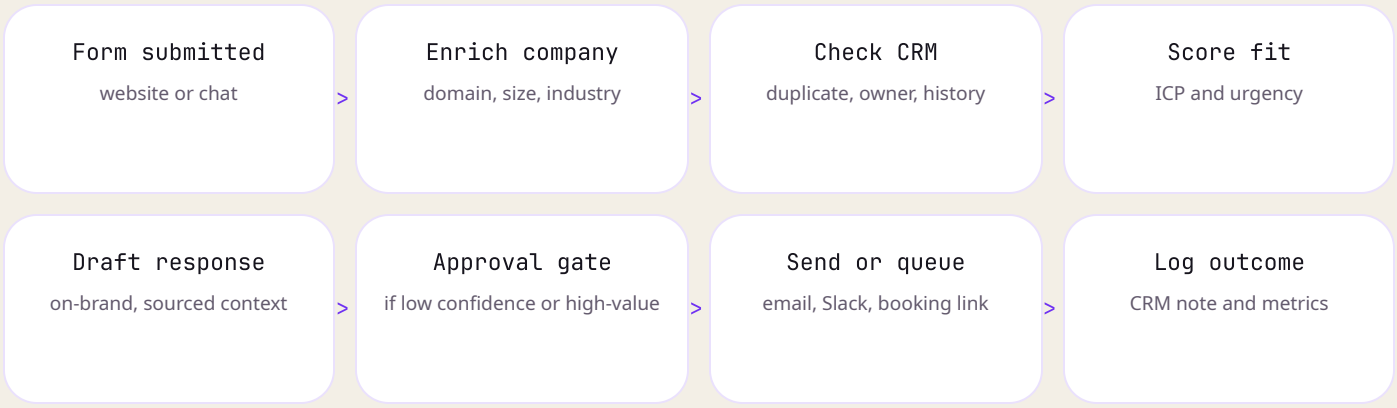
Fit criteria matter: clear workflow, operational pain, a real growth model, practical tool readiness, measurable outcome, and an internal owner. Without those, the agent becomes theater. Very modern, very useless.

marshal

20 / SMB USE CASE

Use case: Speed-to-Lead

Scenario: a founder-led services firm gets a demo request from a qualified buyer while the team is in meetings. The agent handles the first response path without pretending it is the sales team.



Inputs

Lead form, CRM, enrichment, routing rules, calendar availability, response templates.

Human checkpoints

Strategic account, unusual request, uncertain fit, external message below confidence threshold.

Business metrics

First response time, meeting booked rate, lead acceptance, approval rate, edit distance.

marshal

21 / SMB USE CASE

Use case: Client Intake & Onboarding

Scenario: a new client signs. The handoff from sales to delivery can either be a system or a Slack archaeology expedition.

Deal closed
CRM stage change

>

Collect context
contract, scope, notes

>

Create project
template and owner

>

Request intake
forms and assets

Check gaps
missing fields and conflicts

>

Approval gate
custom scope or missing contract

>

Notify team
handoff brief

>

Monitor status
checklist and exceptions

Inputs

Closed-won deal, agreement, kickoff notes, project template, intake questionnaire.

Outputs

Project created, onboarding checklist, client request, internal brief, missing-data queue.

Metrics

Time to kickoff, checklist completeness, missing asset age, owner response time, exception rate.

What SMBs should measure

A good agent program does not measure how many tokens got incinerated in the name of progress. It measures whether work moved.

System	Primary outcomes	Control metrics	Failure signals
Lead Capture	First response time, meetings booked, lead acceptance, conversion to opportunity	Approval rate, edit distance, route accuracy, duplicate rate	Missed SLA, wrong owner, low-quality response, orphan lead
Revenue Generation	Approved prospect volume, reply rate, positive reply rate, qualified opportunities	Suppression accuracy, source coverage, personalization quality, bounce rate	Bad fit list, unsupported claim, sequence fatigue, CRM conflict
Operational Throughput	Cycle time, records synchronized, reports delivered, onboarding completion	Exception rate, write accuracy, rollback rate, missing-field age	Conflicting source data, stale dashboard, unapproved write, unowned task

Outcome-based operations need two kinds of measurement: business results and control health. If the result improves while the control metrics degrade, the system is borrowing risk from the future. The future charges interest.

marshal

23 / BEST PRACTICES

Best-practice architecture patterns

Start with workflows, not personalities

Define the job, trigger, system state, allowed actions, and stop conditions before choosing model behavior.

Keep tools narrow and explicit

Give every tool one job, a fixed schema, boundaries, examples, and failure behavior.

Gate before side effects

Use policy checks and human approvals before writes, external communications, destructive actions, or sensitive decisions.

Use routing over one giant prompt

Specialized sub-agents and model routing reduce cost, latency, and mystery.

Make retries safe

Use idempotency keys and durable run state so failures do not create duplicate actions.

Evaluate trajectories

Judge final outcomes, intermediate tool choices, approval behavior, and full run transcripts.

Instrument everything

Logs, traces, metrics, source context, tool calls, exceptions, and approvals should be visible.

Promote autonomy gradually

Move from observe to draft to approved execution to limited autonomy only after the evidence earns it.

These patterns synthesize current guidance from agent engineering, evaluation, observability, security, and durable workflow research. [3][5][7][10][11]

marshal

24 / CONCLUSION

The SMB operating layer

Marshal Agents exist because founder-led companies have a strange talent for building revenue machines and then feeding them manual coordination until everyone looks tired.

The technical answer is not a chatbot. It is not a no-code toy. It is not enterprise AI cosplaying as salvation. It is a managed agentic operating layer: defined workflows, customer-owned tools, bounded autonomy, human approvals, exception queues, audit logs, evaluations, and continuous operation.

The agent's job is to do the repeatable work, surface the ambiguous work, and make the business more legible every time it runs. That is the real promise: not that AI replaces judgment, but that judgment stops drowning in clerical sludge.

1. **Ground in the stack**

   Use the tools and records the business already runs on.

2. **Constrain the workflow**

   Autonomy lives inside defined process boundaries.

3. **Preserve human authority**

   Humans approve, override, and own judgment.

4. **Evaluate continuously**

   Every run teaches the system, or it should not have run.

Machines handle the work. Humans handle the judgment. Marshal operates the system.

25 / REFERENCES

References and source notes

[1] Marshal Agent Factory
runmarshal.com/agent-factory and linked agent pages, accessed June 2026. Defines three production lines, nine agent types, and the SMB Agent Factory framing.

[2] Marshal platform, governance, and orchestration pages
runmarshal.com/platform/marshal-agents, /platform/agent-governance, /platform/agent-orchestration, and /ai-agent-development-services, accessed June 2026.

[3] Anthropic
Building effective agents, December 2024. Patterns for workflows, routing, prompt chaining, orchestrator-workers, evaluator-optimizer loops, and tool design.

[4] Anthropic
Trustworthy agents in practice, April 2026. Principles for human control, security, transparency, privacy, permission modes, and agent safeguards.

[5] Anthropic
Demystifying evals for AI agents, January 2026. Evaluation concepts, traces, transcripts, outcome checks, model-based and human grading, regression testing.

[6] NIST
Artificial Intelligence Risk Management Framework and Generative AI Profile, 2024. Voluntary framework for trustworthy AI design, deployment, and evaluation.

[7] OWASP
Top 10 for Large Language Model Applications 2025. Risks include prompt injection, sensitive information disclosure, improper output handling, and excessive agency.

[8] Model Context Protocol
MCP specification, 2025-06-18. Standard concepts for resources, prompts, and tools.

[9] Google
Agent2Agent Protocol announcement and documentation, April 2025. Patterns for agent collaboration across frameworks and vendors.

[10] Temporal
Idempotency and durable execution guidance. Safe retries and side-effect management for workflow systems.

[11] OpenTelemetry
Documentation for traces, metrics, logs, and vendor-neutral observability.

[12] OpenAI Agents SDK
Tracing and guardrails documentation for model calls, tools, handoffs, and runtime validation.

[13] LangSmith
Agent evaluation documentation for final response, single-step, and trajectory evaluation.

[14] Lewis et al.
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2020.

marshal

Marshal Agents

The managed agentic operations layer for founder-led businesses that play to win.

Existing stack | Human approvals | Audit trails | Outcome-driven

runmarshal.com