Charter

Composio APM Integrations - Product submission

21 October 2026

Prototype: https://charter-oauth-guide.vercel.app

Source and README: https://github.com/Pal-prakash2/charter (private)

1. Problem and target user

A working OAuth connection does not establish that an integration can launch. Developers still need to understand permission requirements, provider review, distribution rules, customer-administrator controls and who must act next.

The primary user is a developer shipping a customer-facing integration with an AI coding agent. The job is to identify what can be built now, what could delay launch and the next actionable step. This is a source-informed problem hypothesis; no user interviews or measured time savings are claimed.

2. Product approach and Composio context

Charter turns scattered requirements into a scenario-specific launch plan. Audience, capability and launch stage shape the guidance. Developer effort and external waiting stay separate, and requirements retain their sources and unresolved questions.

Existing products informed the approach:

- Composio managed OAuth can remove registration work for supported methods. The first decision is whether a developer-owned OAuth app is needed.
- Nango review guidance shows why authentication setup and approval work need separate treatment.
- Airbyte AI-assisted configuration demonstrates the value of converting documentation into implementation context.

Use Composio-managed OAuth where it fits. Where an OAuth-capable toolkit needs a developer-owned app, Charter explains the setup and launch path. Automatic managed-versus-custom routing is a proposed next step, not a shipped feature.

Product bet: make the next decision clear, preserve the evidence and let the same context travel into the coding workflow.

3. Prototype walkthrough: a consequential scope choice

For Gmail and external customers, a generic OAuth checklist hides a material feature decision:

- **Send email only (gmail.send):** sensitive-scope verification; this scope alone does not trigger a restricted-scope security assessment. Published review estimates differ. Open the send-only scenario.
- **Read a mailbox (gmail.readonly):** restricted-scope verification; assessment depends on the complete data flow and applicable exemptions. Several weeks may be needed. Open the mailbox-reading scenario.

The Gmail scope reference supports this distinction. Complete app behavior and current policy still determine applicability.

Three reviewed examples demonstrate different obstacles:

- GitHub: user authorization does not clear organization access restrictions.
- Gmail: feature scope changes the review and assessment path.
- Slack: early-customer pilots and commercial distribution follow different paths.

In the prototype, expand a gate to inspect its reason, open the launch plan for ordered steps, then switch to Agent mode. The raw Gmail Markdown preserves the same scenario and sources.

4. Agent workflow and ownership

For an unfamiliar app, Charter discovers and reads public documentation, asks Gemini for structured synthesis, validates the schema, source references and dependencies, then saves a shareable guide. It does not operate provider consoles.

- **Charter agent:** public-document research, synthesis, validation and persistence.
- **Developer and coding agent:** choose exact scopes, implement and test. The account owner handles credentials and console actions.
- **Provider or customer administrator:** review and access decisions. Charter cannot approve these gates.

Human guidance, Markdown, JSON and four MCP tools use the same guide record. Agent discovery and MCP access.

5. Validation and findings

A real Gemini run generated and persisted a Linear guide from three fetched documentation pages in about 44 seconds. It remained partial because provider review, customer-admin and marketplace requirements were unresolved. Inspect the research job and evidence, Markdown or JSON.

The Linear OAuth documentation and app manifests documentation informed the guidance. These sources were inspected; the OAuth flow was not executed.

One important finding changed the product: the original waiting label, "None documented," could imply zero waiting despite unknown approval gates. The overall wait is now marked **"End-to-end wait not established."** New output follows this rule. Historical views receive an explicit policy revision, retaining the original estimate and stored evidence.

- **15/15 public contract checks passed**, including saved-guide and job consistency across REST, Markdown and MCP.
- **8 waiting-policy checks passed**, including stage estimates, immutability and historical corrections. TypeScript and production builds passed.

A later live attempt fetched its sources, then Gemini returned 503 UNAVAILABLE for high demand. No unvalidated guide was published; saved results remained available. These software checks do not establish semantic accuracy of every generated claim or customer value.

6. Scope, trade-offs and next validation

Shipped: three reviewed examples, contextual guidance, live research, sharing and agent access. Submission tracking was deferred because provider-status access was not established and manual updates would introduce a separate workflow. Expansion should use the shared schema and reviewed batches, rather than 100 unverified articles.

The next product test is to watch five developers investigate unfamiliar providers with and without Charter, using a counterbalanced task order. Measure **time to a correct next action**, consequential factual errors and lost assumptions during agent handoff. Establish a baseline before setting improvement targets.

Limits: no user interviews, provider registration, token exchange or approval submission were completed. The public registration URL was fetched, but its authenticated form was not inspected. Current browser capture is unavailable. Built with AI assistance.

Evaluate: Prototype | Private source and setup. Evaluators need repository access.