ProAgent Delivery — Capabilities Document

Plugin: proagent-delivery v0.4.0 **Date:** 2026-02-19 **Audience:** Delivery Team (SOs, PMs, Delivery Leads, and practitioners)

Part 1: Executive Summary

ProAgent Delivery is a Claude Code plugin that provides **19 capabilities** across the entire delivery lifecycle. It integrates with Slack, Google Drive, Google Workspace, Atlassian (Jira/Confluence), Excalidraw, and Rube (Composio) to automate delivery workflows from pre-sales through continuous improvement.

At a glance: 11 run modes + 8 review modes = 19 capabilities, 3 skills, 6 MCP integrations, 4 automated hooks.

Capabilities by Delivery Lifecycle Phase

Pre-Sales & Scoping

Capability	What It Does	When to Use It	What It Produces
SOW Generation	Reads client context from Slack channels and Google Drive, conducts a clarification interview with the Solution Owner, and generates a delivery-ready Statement of Work using Provectus templates.	When starting a new client engagement and you need to produce a formal SOW document.	Google Doc with 8 standard SOW sections (Purpose, Organization, Project Overview, Project Scope, Estimated Durations & Team, Payment & Fee Schedule, Project Assumptions, Signatures).
ROM Estimation	Analyzes project documents, task lists, or scope descriptions and produces granular effort estimates broken down by epic and specialty.	When you need to scope a project, estimate effort for budgeting, or plan team composition.	Semicolon-delimited CSV with effort levels (XS/S/M/L/XL) per sub-feature, plus an executive summary with FTE requirements, epic breakdown, and risk factors.
SOW Review	Audits an existing SOW for completeness, scope specificity, pricing alignment, timeline realism, team composition, and risk coverage.	Before sending a SOW to a client for signature, or when reviewing a SOW from another team member.	Structured audit report with section-by-section quality scores and actionable recommendations.
Estimate Review	Audits an existing ROM estimate CSV for completeness, sizing accuracy, missing epics, team balance, and risk coverage.	When reviewing a ROM estimate before presenting it to stakeholders or using it for SOW pricing.	Audit report with completeness checks, sizing concerns, team composition analysis, and risk factors.

Sprint Execution

Capability	What It Does	When to Use It	What It Produces
Sprint Planning	Guides teams through goal setting, capacity checking, backlog grooming, story estimation (Fibonacci), and sprint commitment with risk identification.	At the start of each sprint iteration.	Sprint plan document with committed stories, capacity allocation, risks, and definition of done.
Standup Notes	Generates AI-assisted daily standup notes by analyzing git history, Jira tickets, and calendar data. Links commits to tickets and transforms technical messages into business value statements.	Daily, to prepare standup updates or generate async standup posts.	Structured standup note (Yesterday/Today/Blockers) with Jira links and PR references. Supports daily, async, and 3P formats.
Task Planning	Creates phased task plans with progress tracking, decision logs, and error tracking. Breaks work into 3-7 phases with structured tracking sections.	When starting a complex piece of work that needs structured phases and progress tracking.	Task plan file with phases, key questions, decision log, and error tracking table.

Monitoring & Reporting

Capability	What It Does	When to Use It	What It Produces
Status Reports	Generates status reports tailored to different audiences. Supports 5 formats: daily standup, weekly team update, monthly executive summary, 3P update, and custom.	When stakeholders need a project update at any level of detail.	Formatted status report with RAG indicators, metrics, wins, issues, and next steps.
Milestone Tracking	Compares milestone progress against the baseline plan, calculates schedule drift, performs critical path analysis, and recommends corrective actions.	When checking whether the project is on track for upcoming milestones or release dates.	Milestone dashboard with planned vs. forecast dates, drift analysis, critical path visualization, and corrective action recommendations.
Sprint Health	Assesses velocity trends, burndown progress, active	Mid-sprint or at sprint end to assess	Sprint health scorecard with RAG ratings across velocity,

Review	blockers, and scope stability for the current sprint.	delivery health before the retrospective.	burndown, blockers, and scope stability.

Risk & Stakeholder Management

Capability	What It Does	When to Use It	What It Produces
Risk Assessment	Identifies risks across 5 categories (Technical, Schedule, Resource, Scope, External), analyzes using Probability x Impact matrix, and develops mitigation strategies.	At project kickoff, before major milestones, or when conditions change significantly.	Risk register with severity scores, owners, mitigation plans, and review cadence.
Delivery Risk Review	Evaluates the current risk register, identifies new risks, checks mitigation progress, and assesses overall risk posture trend.	During periodic project health checks or when you sense risk conditions have shifted.	Risk summary with trend analysis, top risks requiring attention, new risks identified, and mitigation plan health.
Stakeholder Alignment	Maps stakeholders using Power/Interest Grid, audits communication cadence, reviews open decisions, and identifies alignment gaps.	When stakeholder relationships feel strained, before major project decisions, or as part of regular governance.	Stakeholder map with engagement scores, open decision backlog, alignment issues, and RACI gap analysis.
Timeline Review	Reviews project timeline against baseline, analyzes schedule variance trends, evaluates critical path, and forecasts completion date with recovery options.	When the project feels behind schedule or before committing to new delivery dates.	Timeline review with phase progress, critical path status, variance trends, and recovery options (fast-track, crash, scope adjustment).

Continuous Improvement

Capability	What It Does	When to Use It	What It Produces
Retrospectives	Facilitates structured retrospectives using Start-Stop-Continue, 5 Whys root cause analysis, and Inversion. Produces 2-3 improvement actions with owners.	At the end of each sprint or after a significant project event.	Retrospective record with sprint metrics, Start/Stop/Continue items, root cause analysis, action items with owners, and one experiment for next sprint.
Meeting Insights	Analyzes meeting transcripts for	When you want to improve meeting	Meeting insights report with communication health

	communication patterns, decision quality, participation balance, and facilitation effectiveness.	culture or assess whether meetings are productive.	scores, decision log, facilitation recommendations, and meetings that could be async.
Comms Quality Review	Reviews the quality and consistency of team communications (status updates, 3P reports, stakeholder emails) for clarity, consistency, tone, and completeness.	When you want to raise the bar on team communications or audit communication quality.	Communications quality report with scores across 5 dimensions, common issues, best practices observed, and improvement recommendations.

Documentation

Capability	What It Does	When to Use It	What It Produces
PRD Creation	Generates a Product Requirements Document from feature ideas and JTBD (Jobs to Be Done) analysis. Reads product context and feature specs.	When defining a new feature and you need a structured requirements document.	PRD with problem statement, target audience, user stories, acceptance criteria, success metrics, scope boundaries, and open questions.
Internal Communications	Drafts internal communications using company-standard formats. Supports 6 types: 3P updates, newsletters, FAQs, incident reports, leadership updates, and project updates.	When you need to write any internal communication and want it formatted to company standards.	Formatted communication in the appropriate template with data-driven content and actionable next steps.

Part 2: Practitioner Guide

Getting Started

Installation

Copy the `proagent-delivery` directory into your Claude Code plugins directory, or reference it from your project's `.claude/plugins/` configuration.

Hub Command

Run the hub command to see all available capabilities and get routed to the right workflow:

```
/proagent-delivery:proagent-delivery-hub
```

The hub presents a quick-start routing table — describe what you need and it suggests the appropriate command.

MCP Server Setup

The plugin uses 6 MCP integrations configured in `.mcp.json`. Set the environment variables for the services your team uses. Unused servers will not be started.

Server	Package	Required Env Vars
Atlassian	`@modelcontextprotocol/server-atlassian`	`ATLASSIAN_API_TOKEN`, `ATLASSIAN_EMAIL`, `ATLASSIAN_DOMAIN`
Slack	`slack-mcp-server`	`SLACK_MCP_XOXC_TOKEN`, `SLACK_MCP_XOXD_TOKEN`
Google Drive	`@modelcontextprotocol/server-gdrive`	*(OAuth — no env vars needed)*
Google Workspace	`mcp-gsuite` (via `uvx`)	*(OAuth — no env vars needed)*
Excalidraw	Remote: `https://mcp.excalidraw.com/mcp`	*(none)*
Rube (Composio)	Remote: `https://rube.app/mcp`	*(none)*

Capabilities Reference

Pre-Sales & Scoping

1. SOW Generation

Command: `/proagent-delivery:proagent-delivery-run generate-sow`

What it does: Reads client context from Slack channels and Google Drive documents via MCP, dispatches the `sow-context-extractor` subagent for data gathering, conducts a structured clarification interview with the Solution Owner (engagement model, pricing, phases, assumptions), and generates all 8 standard SOW sections using Provectus templates. Outputs directly to Google Drive as a Google Doc.

Input:

- `--channel=<slack-channel>` — Slack channel with client conversations
- `--drive=<google-drive-url>` — Google Drive folder with project documents
- `--template=<template>` — SOW template to use (default: `sow-default`)
- `--type=<engagement-type>` — Engagement model: `t-and-m`, `fixed-price`, or `milestone-based`
- `--with-rom` — Optionally append a ROM estimate as an appendix
- `--output=<path>` — Custom output path

Output: Google Doc with sections: Purpose, Organization, Project Overview, Project Scope, Estimated Durations & Team, Payment & Fee Schedule, Project Assumptions, Signatures.

Example scenario: Use this when a new client engagement has been approved and you need to produce a formal SOW. Point it at the Slack channel where scoping conversations happened and the Google Drive

folder with discovery documents. It will extract context, ask you clarifying questions, and produce a complete SOW draft for review.

---

2. ROM Estimation

Command: `/proagent-delivery:proagent-delivery-run rom-estimate`

What it does: Analyzes project documents (local files, pasted content, or Google Drive via MCP), identifies epics, expands high-level tasks into 2-5 granular sub-features, estimates effort using XS/S/M/L/XL levels with optimistic/pessimistic durations, assigns team specialties (DevOps, BE, FE, MLE I, MLE II, DE, QA, Security, Cloud Architecture, UI/UX), and generates both a CSV file and an executive summary.

Input:

- File paths, Google Drive links, or pasted content (task lists, roadmaps, PRDs)
- `--output=<path>` — Custom CSV output path (default: `docs/rom-estimation/{slug}-rom.csv`)
- `--timeline=<weeks>` — Project timeline for FTE calculation

Output: Semicolon-delimited CSV (`epic;feature;effort_level;optimistic_duration;pessimistic_duration;specialities`) plus formatted analysis summary with effort totals, epic breakdown, effort distribution, team requirements (FTEs), risk factors, and assumptions.

Example scenario: Use this when you have a project roadmap or requirements document and need to estimate total effort, team composition, and timeline feasibility. Pass it a Google Drive folder with discovery documents or paste a feature list directly.

---

3. SOW Review

Command: `/proagent-delivery:proagent-delivery-review sow-review <path-to-sow>`

What it does: Audits an existing SOW against Provectus quality criteria: section completeness (all 8 sections present), scope specificity (every scope item passes the "could we disagree?" test), pricing alignment with engagement model, timeline realism, team composition adequacy, and risk coverage.

Input: File path or Google Drive link to the SOW document to review.

Output: Structured audit report with section-by-section quality scores, specific issues found, and actionable recommendations for improvement.

Example scenario: Use this before sending a SOW to a client for signature. Run the review on the draft to catch vague scope items, missing sections, or pricing inconsistencies before stakeholder review.

---

4. Estimate Review

Command: `/proagent-delivery:proagent-delivery-review estimate-review [path-to-csv]`

What it does: Audits an existing ROM estimate CSV for completeness (all standard epics covered, all features have sizing and specialties), sizing accuracy (flags under/over-estimated features based on complexity signals), team composition balance (workload distribution, bottleneck specialties), and risk factors (XL item count, external dependencies, testing coverage).

Input: File path or Google Drive link to the ROM CSV. If not provided, searches for `docs/rom-estimation/*-rom.csv`.

Output: Audit report with completeness checks, sizing concerns table, team composition analysis with FTE requirements, risk factors, and overall estimate health rating (Solid / Needs Refinement / Significant Gaps).

Example scenario: Use this when reviewing a ROM estimate before presenting it to stakeholders or using it to price a SOW. It catches under-scoped epics, mismatched effort levels, and team bottlenecks.

Sprint Execution

5. Sprint Planning

Command: `/proagent-delivery:proagent-delivery-run plan-sprint`

What it does: Guides teams through a structured sprint planning workflow: establish sprint parameters (number, dates, duration, team capacity), define sprint goal aligned to roadmap, groom the backlog (verify acceptance criteria, dependencies, sizing), estimate stories using Fibonacci scale (1/2/3/5/8/13), apply Eisenhower Matrix and Pareto prioritization, and commit to sprint backlog with 10-15% buffer for unplanned work.

Input: Sprint parameters (number, dates, team members), product backlog, velocity history. Gathered interactively if not provided.

Output: Sprint plan document with: sprint goal, dates, capacity, committed stories table (story, points, owner, priority, dependencies), total commitment vs. capacity, sprint risks with mitigation, and definition of done.

Example scenario: Use this at the start of each sprint. It walks the team through the full planning ceremony and produces a structured commitment document that can be shared with stakeholders.

---

6. Standup Notes

Command: `/proagent-delivery:proagent-delivery-run standup-notes`

What it does: Generates AI-assisted daily standup notes by collecting data from git commits (last 24-48h), Jira tickets (via Atlassian MCP), calendar events (via Google Workspace MCP), and local files (TO-DOS.md, CHANGELOG.md). Links commits to Jira tickets by extracting ticket IDs from commit messages, groups related commits into single accomplishment bullets, and transforms technical messages into business value statements.

Input: Format preference: `daily` (default), `async` (Slack-optimized), or `3p` (Progress/Plans/Problems). Git author name is auto-detected.

Output: Structured standup note with Yesterday/Today/Blockers sections, Jira ticket links, PR references, and follow-up tasks extracted from blockers.

Example scenario: Use this every morning before standup. It scans your recent git activity and Jira tickets to draft a standup update you can review and post, saving 5-10 minutes of manual preparation.

---

7. Task Planning

Command: `/proagent-delivery:proagent-delivery-run task-plan`

What it does: Creates a phased task plan from a goal description. Breaks work into 3-7 phases (Requirements & Discovery, Planning & Structure, Implementation, Testing & Verification, Delivery) with progress tracking. Includes key questions to answer, a decision log (Decision + Rationale), and an error tracking table (Error + Attempt + Resolution).

Input: Task objective from arguments or conversation context.

Output: Task plan file (`task-plan.md`) with phases, status tracking (`pending` / `in_progress` / `complete`), key questions, decision log, and error tracking table.

Example scenario: Use this when starting a complex deliverable that spans multiple days or involves decisions that need to be tracked. The plan file serves as a living document you update as you progress through phases.

Monitoring & Reporting

8. Status Reports

Command: `/proagent-delivery:proagent-delivery-run status-report [format]`

What it does: Generates status reports tailored to different audiences by gathering data from project tracking files, git history, and sprint backlogs. Supports 5 formats: `daily` / `standup` (Yesterday/Today/Blockers), `weekly` (team status with metrics), `monthly` / `executive` (executive summary with RAG status), `3p` (Progress/Plans/Problems in 30-60 seconds), and custom. Verifies RAG status honesty and metric accuracy.

Input: Report format (`daily` , `weekly` , `monthly` , `executive` , `3p`). Project data gathered from files and git history.

Output: Formatted status report with RAG indicators, progress, metrics (velocity, budget, timeline), wins, issues, decisions needed, and next steps.

Example scenario: Use `status-report weekly` every Friday to generate the team's weekly status email. Use `status-report 3p` for quick leadership updates. Use `status-report executive` for monthly steering committee reports.

---

9. Milestone Tracking

Command: `/proagent-delivery:proagent-delivery-run milestone-track`

What it does: Loads the project baseline plan (roadmap, timeline documents), assesses each milestone's health (planned vs. forecast date, deliverable completion, dependencies, schedule drift), performs critical path analysis, and recommends corrective actions (scope reduction, resource addition, timeline adjustment, dependency fast-tracking).

Input: Project plan or roadmap documents. Falls back to interactive input if no formal plan exists.

Output: Milestone dashboard with: milestone table (planned/forecast/drift/status/% complete/blockers), critical path visualization, schedule health summary, corrective actions table, and overall forecast variance.

Example scenario: Use this mid-project when you need to assess whether the project will hit its delivery date. It identifies which milestones are slipping and whether those are on the critical path.

---

10. Sprint Health Review

Command: `/proagent-delivery:proagent-delivery-review sprint-health`

What it does: Reviews the health of the current or most recent sprint across 4 dimensions: velocity (current vs. commitment, trend over 3-5 sprints), burndown (pace to complete by sprint end, flat spots, late scope additions), blockers (count, duration, categories, resolution time), and scope stability (stories added/removed after sprint start, net scope change). Flags scope change >10% for process review.

Input: Sprint backlog from Jira, project tracking files, or user input.

Output: Sprint health scorecard with RAG ratings across velocity, burndown, blockers, and scope stability. Includes recommendations and projected completion.

Example scenario: Use this mid-sprint to check if the team is on track, or at sprint end before the retrospective to have data-driven input on what went well and what didn't.

Risk & Stakeholder Management

11. Risk Assessment

Command: `/proagent-delivery:proagent-delivery-run risk-assess`

What it does: Runs a full project risk assessment: identifies risks across 5 categories (Technical, Schedule, Resource, Scope, External), analyzes each using Probability x Impact matrix (Low/Medium/High), prioritizes by severity (Critical/High/Medium/Low), develops mitigation strategies (Avoidance, Reduction, Transfer, Acceptance), and applies 5 Whys and SWOT for materialized risks.

Input: Project context, existing risk documentation, team input. Uses brainstorming, historical data review, expert consultation, assumption analysis, and pre-mortem techniques.

Output: Risk register with: risk table (ID, risk, category, probability, impact, score, owner, status, mitigation, review frequency), detailed mitigation plans for critical risks, risk trends, and next review date.

Example scenario: Use this at project kickoff to establish the initial risk register, before major milestones to reassess, or when project conditions change significantly (scope change, team change, dependency shift).

---

12. Delivery Risk Review

Command: `/proagent-delivery:proagent-delivery-review delivery-risks`

What it does: Evaluates the current risk register: reviews each existing risk for probability/impact changes and mitigation progress, identifies new risks using pre-mortem technique and recent project events analysis, tracks risk trends (increasing/stable/decreasing), and assesses overall risk posture.

Input: Existing risk documentation (`docs/risk-register.md` , `context/risks.md`). If no register exists, offers to create one via risk-assess.

Output: Delivery risk review with: risk summary by severity with trends, top 3 risks requiring attention, new risks identified, risks recommended for closure, mitigation plan health, and overall risk posture rating (Low / Moderate / High / Critical).

Example scenario: Use this during regular project health checks (bi-weekly or monthly) to ensure the risk register is current and mitigation plans are being executed.

---

13. Stakeholder Alignment

Command: `/proagent-delivery:proagent-delivery-review stakeholder-alignment`

What it does: Maps all stakeholders using Power/Interest Grid (Manage Closely, Keep Satisfied, Keep Informed, Monitor), audits communication cadence (last contact, frequency appropriateness, responsiveness), reviews the decision backlog (pending decisions, overdue items, impact of delays), and assesses overall alignment (scope agreement, priority conflicts, expectation consistency).

Input: Stakeholder list, communication records, pending decisions. Gathered interactively if not documented.

Output: Stakeholder alignment review with: stakeholder map table, communication health metrics, open decisions table, alignment issues, RACI gaps, and overall stakeholder health rating (Aligned / Minor Gaps / Significant Misalignment).

Example scenario: Use this when you sense stakeholder frustration, before major project decisions, or as part of monthly governance reviews. It identifies silent stakeholders and stalled decisions before they become problems.

---

14. Timeline Review

Command: `/proagent-delivery:proagent-delivery-review timeline`

What it does: Reviews the project timeline against the original baseline: loads the baseline plan, assesses current progress for each phase/milestone, analyzes schedule variance trends, evaluates the critical path, and forecasts completion date. Identifies recovery options: fast-tracking (parallel tasks), crashing (add resources), and scope adjustment (defer deliverables).

Input: Project plan documents, roadmap, milestone definitions.

Output: Timeline review with: phase progress table (planned/forecast/variance/status/% complete), critical path visualization, schedule variance trend, recovery options table (schedule impact, cost impact, risk level, recommendation), and verdict (On Track / At Risk / Behind Schedule).

Example scenario: Use this when the project feels behind schedule or before committing to new delivery dates. It provides data to support schedule negotiations with stakeholders.

Continuous Improvement

15. Retrospectives

Command: `/proagent-delivery:proagent-delivery-run retrospective`

What it does: Facilitates a structured sprint retrospective in 5 stages: set the stage (5 min — objective, ground rules, icebreaker), gather data (15 min — Start-Stop-Continue + metrics), generate insights (15 min — theme grouping, 5 Whys root cause, Inversion), decide on actions (20 min — top 2-3 improvements with owners, one experiment), and close (5 min — recap and follow-up scheduling).

Input: Sprint number, sprint goal, completion data. Interactive facilitation guides the team through each stage.

Output: Retrospective record with: sprint metrics (velocity, stories completed/carried over, bugs, blockers), Start/Stop/Continue items, 5 Whys root cause analysis, action items table (action, owner, due date, success criteria), and one experiment for next sprint.

Example scenario: Use this at the end of each sprint to facilitate the retro ceremony. It provides structure and ensures the team produces concrete, owned action items rather than vague complaints.

---

16. Meeting Insights

Command: `/proagent-delivery:proagent-delivery-review meeting-insights`

What it does: Analyzes meeting transcripts or notes for communication patterns: participation balance (who's dominating, who's silent), topic drift (agenda adherence), decision velocity (how quickly decisions

are made vs. deferred), action item clarity (owners and deadlines assigned), follow-up rate (previous action items completed), and facilitation effectiveness (agenda, time management, parking lot usage).

Input: Meeting notes, transcripts, or recordings from project files, Confluence, Google Docs, or local files.

Output: Meeting insights report with: communication health scorecard (5 dimensions), decision log, facilitation recommendations, meetings that could be async, and overall meeting health rating (Effective / Needs Improvement / Inefficient).

Example scenario: Use this when you want to improve meeting culture. Analyze a month's worth of sprint ceremonies to identify patterns — maybe standups are running long, or decisions are being deferred instead of made.

---

17. Comms Quality Review

Command: `/proagent-delivery:proagent-delivery-review comms-quality`

What it does: Reviews team communications (status updates, 3P reports, stakeholder emails, Slack messages) across 5 dimensions: clarity (readable in 30-60 seconds, bottom line upfront, data-driven metrics), consistency (format adherence, cadence met, terminology), tone/audience fit (technical abstraction, matter-of-fact language, early concern raising), completeness (progress + plans + problems, blockers with asks, wins and recognition), and timeliness.

Input: Recent communication samples from project files, Slack, email, or local documents.

Output: Communications quality report with: quality scores across 5 dimensions, common issues, best practices observed, recommendations, and overall comms health rating (Strong / Adequate / Needs Improvement).

Example scenario: Use this quarterly to audit the quality of team communications. It identifies patterns like buried bad news, inconsistent formatting, or missing stakeholder updates.

Documentation

18. PRD Creation

Command: `/proagent-delivery:proagent-delivery-run create-prd`

What it does: Generates a Product Requirements Document from feature ideas and JTBD (Jobs to Be Done) analysis. Reads product documentation, feature specs, and JTBD analysis. Focuses on product requirements and user needs (not technical implementation). Does not include time estimates.

Input: Product context (`product-development/resources/product.md`), feature description (`product-development/current-feature/feature.md`), JTBD analysis (`product-development/current-feature/JTBD.md`). Falls back to user input if files are missing.

Output: PRD with: problem statement, target audience, user stories with acceptance criteria, success metrics, scope (in/out), dependencies, and open questions. Saved to `product-development/current-feature/PRD.md` .

Example scenario: Use this when the product team has defined a feature idea and you need to formalize it into a structured requirements document before estimation or sprint planning.

---

19. Internal Communications

Command: `/proagent-delivery:proagent-delivery-run internal-comms <type>`

What it does: Drafts internal communications using company-standard formats. Gathers content from project files, git history, and sprint backlogs, then applies format-specific guidelines for tone, structure, and audience.

Input: Communication type: `3p`, `newsletter`, `faq`, `incident`, `leadership`, or `project-update`. Content context from project files.

Output:

- **3P updates:** Emoji + team name + dates, 1-3 sentences per section, readable in 30-60 seconds
- **Newsletters:** Headline, executive summary, detailed sections, call-to-action
- **FAQs:** Structured question/answer format
- **Incident reports:** Severity, timeline, impact, root cause, remediation, lessons learned
- **Leadership updates:** Bottom-line summary, key metrics, strategic alignment, decisions needed
- **Project updates:** Detailed project progress for active contributors

Example scenario: Use `internal-comms 3p` for your weekly team update to leadership. Use `internal-comms incident` after a production issue to document the post-mortem. Use `internal-comms newsletter` for quarterly team newsletters.

Integrations

What Each MCP Server Provides

Server	What It Enables
Slack (`slack-mcp-server`)	Read channel history for SOW context extraction, post standup updates, share status reports, send milestone notifications, gather team conversations for analysis.
Google Drive (`@modelcontextprotocol/server-gdrive`)	Read project documents (Docs as Markdown, Sheets as CSV, PDFs) for ROM estimation and SOW generation. Output SOWs and reports as Google Docs. List and browse Drive folders.
Google Workspace (`mcp-gsuite`)	Access Gmail for communication analysis and Google Calendar for meeting schedules, sprint ceremony scheduling, and standup context.
Atlassian (`@modelcontextprotocol/server-atlassian`)	Jira sprint management, backlog grooming, issue tracking, velocity metrics, burndown charts. Confluence documentation publishing for meeting notes, risk registers, and dashboards.
Excalidraw (`excalidraw/excalidraw-mcp`)	Interactive visual diagramming — renders canvases directly in chat via natural language. Use for project timelines, sprint flows, stakeholder maps, risk matrices, and architecture diagrams.
Rube / Composio (`rube.app/mcp`)	SaaS automation gateway providing access to Jira, Linear, Asana, ClickUp, Monday.com, Confluence, and Trello. Three-step workflow: discover tools, connect service, execute actions.

Automated Hooks

The plugin includes 4 hooks that fire automatically to support delivery discipline:

Hook	When It Fires	What It Does
Milestone Check	Before any `deploy`, `release`, or `publish` bash command	Warns you to verify milestone acceptance criteria are met and stakeholders have been notified. Suggests running `milestone-track` to check readiness.
Status Update Reminder	After any `git commit` bash command	Suggests running `status-report` to update stakeholders if the commit represents a significant batch of work.
Sprint Boundary Notification	After a Task tool invocation mentioning delivery-specialist, sprint planning, or retrospective	Suggests follow-up actions: create Jira sprint, schedule ceremonies, share summary with stakeholders.
SOW Output Notification	After writing a file with "sow" or "statement of work" in the path	Suggests sharing the SOW with stakeholders via Slack and uploading to the client Google Drive folder.

Part 3: Feedback Request

We'd love your input to shape the next version of proagent-delivery. Please leave comments directly on this document or reply with your thoughts on the following:

Questions for the Team

1. **Which capabilities are most relevant to your current projects?** Which of the 19 capabilities would you use this week if they were set up on your project?
2. **What workflows are missing?** Are there delivery activities you perform regularly that aren't covered? (e.g., change request management, vendor management, budget forecasting, client demo preparation)
3. **What would make you use this daily?** What's the single thing that would make proagent-delivery a daily habit rather than an occasional tool?
4. **Are there integrations you'd want that aren't listed?** Beyond Slack, Google Drive, Google Workspace, Atlassian, Excalidraw, and Rube — what tools do you rely on? (e.g., Harvest for time tracking, Notion, specific CI/CD tools)
5. **What's your biggest delivery pain point right now?** If you could wave a wand and automate one delivery task, what would it be?
6. **How should we handle adoption?** Would you prefer a guided onboarding session, self-service documentation, or pair-working on a real project?

This document is the version-controlled source of truth. Updates will be tracked via PR to the `provectus-marketplace` repository.