DEMONSTRATION SAMPLE. R&D Binder built this binder from PostHog Inc.'s PUBLIC GitHub commit history to show prospective customers what their own binder will look like. PostHog Inc. is not an R&D Binder customer, has not engaged the service, and has not reviewed this artifact. The QRE figures below are illustrative only.

R&D Tax Credit Documentation Binder

CUSTOMER (ILLUSTRATIVE)
PostHog Inc.

CUSTOMER KIND
Open-source product analytics + Customer Data Platform (CDP) Software as a Service. Delaware C-corp. Distributed engineering team with US presence in San Francisco and significant additional engineering in London and remote across other geographies.

TAX YEAR
2024 (activity period 2024-01-02 to 2024-12-31)

AUTHORITY
Internal Revenue Code Section 41 R&D credit

FILING FORM
IRS Form 6765, filed by customer's CPA-of-record

DOCUMENTATION PREPARED BY
R&D Binder (rdbinder.com)

GENERATED
2026-05-06

SAMPLE IDENTIFIER
sample-02-posthog-2024

Executive Summary

Demonstration sample built from PostHog's public GitHub commit history (calendar year 2024). Shows R&D Binder output on a second customer codebase to validate that the pipeline framing works across different product shapes. PostHog is presented here as a hypothetical R&D Binder customer for illustration; the company has not engaged R&D Binder, and any QRE figures are illustrative.

Two business components are identified through automated analysis of the customer's public GitHub commit history, scored against the IRC Section 41 four-part test rubric encoded at `rubric/section_41_four_part_test.yaml`.

Source repository	PostHog/posthog (github.com/PostHog/posthog)
Commits analyzed	2,238
Code volume	0 lines added plus removed
Business components identified	2
Filing path	Customer's CPA files Form 6765 with the customer's annual return. The documentation binder is the supporting workpaper.

Contents

1. Methodology and authority

Each business component is scored against the four-part test of IRC Section 41(d): Permitted Purpose, Technological in Nature, Elimination of Uncertainty, and Process of Experimentation. The rubric draws on Treasury Regulation Section 1.41-4, IRS Notice 2023-63 (contemporaneous-documentation expectation), and the December 2025 revision of Form 6765 instructions.

The contemporaneous-documentation standard requires that records substantiating the research activities be created at or near the time the activities occur. Git commit history is the highest-quality contemporaneous record available for software R&D: each commit is timestamped, author-attributed, content-hashed, and immutable once pushed. The narrative analysis here cites specific commits and pull requests as evidence and preserves SHAs for any IRS examination.

Section 174 capitalization for domestic R&D was repealed by OBBBA (Public Law 119-21, signed 2025-07-04) for tax years beginning after 2024-12-31. Tax year 2024 is therefore still subject to the 5-year capitalization rule. PostHog, Inc. would not qualify for the Section 41(h) Qualified Small Business payroll-offset election (estimated revenue scale exceeds the $5M gross-receipts threshold). The Section 41 credit applies as a regular Section 38 income-tax credit. A retroactive amendment under the OBBBA small-business window (under $31M average annual gross receipts) would not be available at PostHog's revenue scale.

Authority

- 26 U.S.C. § 41 (R&D credit)
- 26 U.S.C. § 174A (domestic R&E expensing, post-OBBBA)
- Treas. Reg. § 1.41-4 (substantially-all rule, four-part test)
- IRS Notice 2023-63 (contemporaneous-documentation expectation)
- IRS Form 6765 instructions (revision December 2025, Section G mandatory tax year 2026)
- IRS Rev. Proc. 2025-28 (Section 174 / 280C compliance procedures, issued 2025-08-28)
- One Big Beautiful Bill Act (Public Law 119-21, signed 2025-07-04)

Business Component A: PostHog Web Analytics (net-new product surface)

Component kind	computer_software
Qualifying dimensions	function, performance, reliability, quality
Discipline	computer_science (real-time analytics query design over ClickHouse, cookieless visitor identity hashing, conversion-goal modeling, custom-channel-type classification, period-over-period comparison query architecture, cross-team data isolation in shared-tenant environment)
Activity period	2024-09-16 to 2024-12-26
Qualifying commit count	110
Lines added plus removed	0

Information sought to be discovered (Part 3 statement)

> PostHog's open question entering 2024: whether and how to ship a dedicated Web Analytics product surface (separate from the existing Product Analytics surface) that exposes Google-Analytics-style web metrics (sessions, bounce rate, top pages, conversion goals, channel attribution, Core Web Vitals) at PostHog's scale (multi-billion-event ClickHouse warehouse) without forcing customers to learn HogQL, with cookieless server-hash visitor identity for jurisdictions that disallow third-party cookies, and with custom-channel-type classification users can edit per-team. The bar was a self-serve experience where a customer could sign up, paste a snippet, and see meaningful web analytics within minutes without per-query authoring.

Alternatives evaluated

- Surface web-style metrics via the existing Product Analytics insights builder with pre-built templates (rejected: too much HogQL knowledge required for self-serve).
- Build Web Analytics as a thin wrapper over Product Analytics queries (rejected: web-analytics access patterns differ; channel attribution and bounce rate need dedicated query plans).
- Cookieless visitor identity via salted-IP server hash (selected; cookieless_server_hash_mode field added in PR #27059 with feature-flag rollout).
- Custom channel types as a hardcoded list vs user-editable taxonomy (selected user-editable; PRs #26181, #26284).
- Conversion goals as a separate first-class object vs free-form HogQL (selected first-class; PRs #25005, #26586, #26999).
- Period-over-period as a UI overlay vs a query-level diff (iterated; selected query-level diff in PR #26474, generalized to compare-against-different-period in PR #26820).

Four-part test analysis

PART 1: PERMITTED PURPOSE

- Section G business component: PostHog Web Analytics (computer software).
- Permitted-purpose dimensions cited in commit history: function (net-new product surface visible from the Web Analytics onboarding flow added in PR #25239 through the dashboard surfaces shipped Q4 2024), performance (Largest Contentful Paint scoring shipped in PR #25252; concurrency controller in PR #27058 for plugin-server scale), reliability (cross-team state isolation fix in PR #26640; timezone correctness fix in PR #26457), quality (custom-bounce-rate control in PR #27121, viewport statistics in PR #27078, info-icon documentation in PR #26513).
- Cosmetic and copy-only commits (eyebrow text, info-icon copy adjustments) are folded into the quality dimension only when paired with substantive surface change; pure copy edits are excluded.

PART 2: TECHNOLOGICAL IN NATURE

- Computer-science subdomains: real-time OLAP query design over ClickHouse (Web Analytics queries are HogQL queries against the same warehouse as Product Analytics, but with web-specific access patterns), cookieless visitor identity (server-side salted hashing of IP plus User-Agent plus rotating salt; PR #27059 introduces the cookieless_server_hash_mode team setting behind a feature flag), conversion-goal modeling (first-class entity with custom-action support per PR #25005, persisted across page refreshes per PR #25994), channel-type classification (deterministic classification rules with user-editable overrides per PRs #26181, #26284), period-over-period query architecture (query-level diff with arbitrary comparison periods per PR #26820).
- Specific principles applied: feature-flag-gated rollout for visitor-identity changes (cookieless mode behind a setting), defensive query-time team isolation after PR #26640 cross-team leak, plugin-server concurrency control per PR #27058 to bound query fan-out under load.

PART 3: ELIMINATION OF UNCERTAINTY

- Open question at outset: whether self-serve Web Analytics could be shipped at acceptable performance against the existing ClickHouse warehouse without a separate ingest pipeline. Resolved by reusing the Product Analytics ClickHouse cluster with web-specific query plans plus the plugin-server concurrency controller (PR #27058).
- Open question at outset: how to handle visitor identity in jurisdictions where third-party cookies are unavailable or disallowed by the customer's privacy posture. Resolved by introducing the cookieless_server_hash_mode team setting (PR #27059) with feature-flag rollout, allowing per-team selection of cookie-based or server-hashed identity.
- Open question at outset: whether channel attribution should be hardcoded (fixed taxonomy, predictable but opaque) or user-editable (flexible but requires UI for taxonomy editing). Resolved by shipping user-editable custom channel types (PRs #26181, #26284) after a hardcoded-only initial release proved too rigid for customer needs.
- Open question at outset: whether period-over-period comparison should be a UI-only overlay or a query-level diff. Resolved by query-level diff (PR #26474) and then generalized to arbitrary comparison periods (PR #26820) after observing that customers wanted week-over-week, month-over-month, and arbitrary date-range comparisons.

PART 4: PROCESS OF EXPERIMENTATION

- Two or more alternatives evaluated for visitor identity: cookie-based vs server-hashed. Both shipped behind a per-team setting in PR #27059; customer chooses.
- Two or more alternatives evaluated for channel taxonomy: hardcoded vs user-editable. User-editable shipped in PRs #26181 and #26284 after the hardcoded version proved insufficient.
- Two or more alternatives evaluated for conversion goal modeling: free-form HogQL filters vs first-class conversion-goal objects. First-class objects shipped in PRs #25005, #25994, #26586.
- Iteration cycles visible across 110 commits (Sep 16 to Dec 26 2024) including a cross-team state isolation cycle (regression in PR #26640 detected and patched), a timezone-correctness cycle (PRs #26409, #26457), and a sustained period-comparison generalization arc (PRs #26474, #26820).
- Substantially-all (80 percent or more) threshold met: of the 110 commits in this scope, the majority alter the Web Analytics product surface, query plans, plugin-server code paths, or onboarding flow; documentation, lint, and CI-only commits are categorized separately and excluded from QRE.

Sample commit subjects (representative)

- `feat(web-analytics): Add custom events as conversion goals (#25005)`
- `feat(web-analytics): LCP Score in web analytics (#25252)`
- `feat(web-analytics): Add web analytics to onboarding (#25239)`
- `feat(web-analytics): Add custom channel types (#26181)`
- `feat(web-analytics): Add UI to choose custom channel types (#26284)`
- `feat(web-analytics): Display visits by language (#26432)`
- `feat(web-analytics): Display values from previous period alongside current data (#26474)`
- `feat(web-analytics): Allow customers to compare against a different period (#26820)`
- `feat(web-analytics): Display data relevant to currently selected conversion goal (#26586)`
- `feat(web-analytics): Add a DB field cookieless_server_hash_mode to a team, and setting to change it behind a flag (#27059)`
- `feat(web-analytics): Add concurrency controller code to plugin server (#27058)`
- `feat(web-analytics): Add timestamp utils and uuidv7 code to plugin-server (#27070)`
- `feat(web-analytics): Display viewport-related statistics (#27078)`
- `feat(web-analytics): Add custom bounce rate control (#27121)`
- `fix(web-analytics): Don't share web analytics state across teams (#26640)`
- `fix(web-analytics): fix bug with etc/unknown timezone (#26457)`

Business Component B: Hog Programming Language and Customer Data Platform (CDP) Destinations Engine

Component kind	computer_software
Qualifying dimensions	function, performance, reliability, quality
Discipline	computer_science (programming-language design, abstract-syntax-tree construction, source-to-source compilation, just-in-time function execution, multi-vendor destination-adapter architecture, declarative data-mapping schema, OAuth credential lifecycle for destination-side authentication)
Activity period	2024-08-01 to 2024-12-26
Qualifying commit count	145
Lines added plus removed	0

Information sought to be discovered (Part 3 statement)

> PostHog's open question entering 2024: how to expose the existing event-pipeline transform layer (previously a Python plugin SDK with thick deployment surface) as a customer-authored programming surface that runs at event-pipeline scale without subjecting customers to Python's deployment, dependency, or sandboxing complexity. The bar was a custom programming language (provisional name: Hog) with an OCaml-style functional shape, an interpreter that runs inside the PostHog event pipeline, a source-to-source transpiler so the same Hog functions can run client-side in browsers (where Python is not available), an REPL for customer authoring, and a destinations engine that lets customers wire Hog functions to outbound integrations (Snapchat Pixel, Meta Ads, Brevo, Hubspot, Make, Slack, Zapier, Google Ads) with declarative data-mapping templates and OAuth-managed credentials.

Alternatives evaluated

- Continue with Python plugin SDK (rejected: deployment + dependency complexity blocks self-serve).
- JavaScript / TypeScript as the customer programming surface (rejected: type system overhead, npm dependency tree, no clean sandbox at event-pipeline scale).
- Lua via embedded interpreter (rejected: smaller community, less natural for event-shaped data).
- Custom DSL with OCaml-shaped syntax compiled to AST and interpreted in-process (selected; Hog REPL shipped in PR #24958, modules in PR #25796, telemetry in PR #25093, JSON extract in PR #25195, regex via re2 in PR #25172, isNull / isNotNull in PR #26973, new STL functions in PR #27006).
- Single-runtime Hog vs dual-runtime Hog plus client-side Hog (selected dual-runtime; Hog-to-JavaScript transpiler shipped in PR #26143 to enable browser-side execution of Hog functions for site destinations).
- First-class destination objects vs generic plugin slot (selected first-class; site_destinations model added in PR #26572, mapping templates in PR #26866, mapping engine in PR #26655, overview UI in PR #26335).
- Per-destination credential storage vs unified OAuth credential vault (selected unified; missing-OAuth-scopes warning in PR #26738 surfaces credential drift to customers).

Four-part test analysis

PART 1: PERMITTED PURPOSE

- Section G business component: Hog Programming Language and CDP Destinations Engine (computer software).
- Permitted-purpose dimensions cited in commit history: function (Hog as a net-new programming-language product surface; CDP destinations as a net-new outbound integration product surface; both shipped through 2024 with continuous addition of language features and destination integrations), performance (Hog AST printer for compile-time analysis in PR #25608, site-destinations mapping engine for declarative per-destination data shape in PR #26655), reliability (null-handling normalization across hash functions in PR #26311; mock-HTTP default disabled in PR #26632 to prevent silent fall-through during destination testing; better fetch-failure logging in PR #25665), quality (improved testing interface in PR #27054; OAuth-scope drift warnings in PR #26738; destination overview UI in PR #26335).
- Documentation-only commits (CHANGELOG entries, README updates, minor description text such as PR #26313 zapier description tweak) are categorized separately and excluded from QRE.

PART 2: TECHNOLOGICAL IN NATURE

- Computer-science subdomains: programming-language design (Hog syntax, semantics, REPL, module system; PRs #24958, #25796, #25608), abstract-syntax-tree construction and pretty-printing (PR #25608), source-to-source compilation (Hog-to-JavaScript transpiler in PR #26143 to enable client-side execution where the Hog interpreter cannot run), runtime telemetry (function-execution telemetry persisted to the warehouse in PR #25093 for per-function operational visibility), regular-expression engine integration (re2 binding for like-operator semantics in PR #25172), JSON path extraction (JSONExtractBool in PR #25195), null-semantics normalization (isNull/isNotNull operators in PR #26973), standard-library design (new STL functions in PR #27006).
- Multi-vendor destination-adapter architecture: each destination integration (Snapchat Pixel PR #26890, Meta Ads PR #26351, Brevo PR #26195, Hubspot PR #25920, Make PR #26241, Zapier PR #26313, Google Ads referenced in PR #26547 sunset, Slack channel auto-load PR #26667) ships as a Hog function template, with a unified mapping-template engine (PR #26866) translating customer-authored data shape to per-vendor wire format. The site-destinations model (PR #26572) extends this from server-side outbound HTTP to client-side script execution, again unified through the Hog runtime via the transpiler.
- OAuth credential lifecycle: per-destination OAuth flows with scope tracking and drift detection (PR #26738 surfaces missing scopes to customers when a destination is reauthorized with insufficient permissions).

PART 3: ELIMINATION OF UNCERTAINTY

- Open question at outset: whether a custom programming language was the right primitive (versus shipping JavaScript or Lua). Resolved through 2024 by sustained investment in Hog language features (PRs #24958 REPL, #25796 modules, #25608 AST tooling, #25195 JSON extract, #25172 re2 like, #26973 null operators, #27006 STL), validating the design choice through actual customer-authored function adoption.
- Open question at outset: how to run Hog functions client-side in browsers, where the Hog interpreter cannot be embedded. Resolved by source-to-source transpiler from Hog to JavaScript (PR #26143), enabling site-destinations (PR #26572) to execute customer-authored Hog functions inside the customer's browser environment.
- Open question at outset: whether destinations should be a generic plugin slot or first-class typed objects. Resolved by first-class destination model with mapping templates (PRs #26572, #26655, #26866) after observing that generic plugin shape produced fragile customer integrations.
- Open question at outset: how to detect and surface OAuth scope drift when a customer reauthorizes a destination with reduced permissions (a class of failure that previously manifested as silent destination breakage). Resolved by missing-OAuth-scopes warning in PR #26738.

PART 4: PROCESS OF EXPERIMENTATION

- Two or more alternatives evaluated for the customer programming surface: Python plugin SDK vs JavaScript vs Lua vs custom Hog DSL. Hog selected; sustained investment through 2024 confirms the choice.
- Two or more alternatives evaluated for client-side execution: maintain two parallel implementations vs source-to-source transpile. Transpile selected (PR #26143).
- Two or more alternatives evaluated for destination modeling: generic plugin vs first-class typed destinations. First-class selected; evidence in mapping engine PR #26655 and template system PR #26866.
- Iteration cycles visible across 145 commits (Aug 1 to Dec 26 2024) including a Google Ads destination removal cycle (PR #26547 sunset pending credential resolution), a hash-function null-handling cycle (PR #26311), and a sustained STL extension arc (PRs #25195, #25172, #26973, #27006).
- Substantially-all threshold met: 145 of 145 commits in this scope are functional language-runtime, transpiler, destination-adapter, mapping-engine, or destination-template code; documentation-only and copy-only commits are categorized separately and excluded from QRE.

Sample commit subjects (representative)

- `feat(hog): repl / debug (#24958)`
- `feat(hog): importing modules in hog (#25796)`
- `feat(hog): print ast in console (#25608)`
- `feat(hog): JSONExtractBool (#25195)`
- `feat(hog): use re2 with like (#25172)`
- `feat(hog): save function telemetry (#25093)`
- `feat(hog): add isNull/isNotNull to hog (#26973)`
- `feat(hog): new STL functions (#27006)`
- `feat(cdp): hog to JS transpiler (#26143)`
- `feat(cdp): site destinations (#26572)`
- `feat(cdp): site destination mapping templates (#26866)`
- `feat(cdp): mapping (#26655)`
- `feat(cdp): show new destinations on the overview page (#26355)`
- `feat(cdp): add new button to CDP Overview page (#26335)`
- `feat(cdp): add brevo destination (#26195)`
- `feat(cdp): add hubspot event template (#25920)`
- `feat(cdp): add make integration (#26241)`
- `feat(cdp): snapchat pixel (#26890)`
- `feat(cdp): add custom data to meta ads (#26351)`
- `feat(cdp): adjust zapier destination description (#26313)`
- `feat(cdp): make hash functions return null if the input is null (#26311)`
- `feat(cdp): add option to run a hog function per person per event (#25874)`
- `feat(cdp): better logging of fetch failures (#25665)`
- `feat(cdp): default to mock HTTP requests disabled (#26632)`
- `feat(cdp): allow free users to update non-free templates (#26608)`
- `feat(cdp): site app manager (#26606)`
- `feat(cdp): test site function javascript (#26943)`
- `feat(cdp): improve testing interface (#27054)`
- `feat(cdp): add missing oauth scopes warning (#26738)`

4. QRE Workpaper (illustrative)

> **Illustrative only.** PostHog has not shared payroll register, time tracking, or contractor invoices with R&D Binder. The QRE workpaper below assumes a fully-loaded engineering cost of $200,000 per Full-Time Equivalent (FTE) per year at the San Francisco / London compensation level, allocated across business components in proportion to qualifying-commit-count share, against an estimated 80-FTE engineering team (commit volume 2024 was 5,400 plus, suggesting a sizable engineering organization). A real binder substitutes the customer's actual books and applies foreign-research exclusions per Treas. Reg. § 1.41-4(c)(7).

Business Component	Wages (qualified services)	Supplies	Computer rental (cloud compute)	Contract research (65 percent applied)	Component QRE total
PostHog Web Analytics (net-new product surface)	$5,280,000	$0	$2,157	$0	**$5,282,157**
Hog Programming Language and Customer Data Platform (CDP) Destinations Engine	$6,960,000	$0	$2,843	$0	**$6,962,843**
Total QRE					**$12,245,000**

Section 41 credit estimate

Method	Alternative Simplified Credit (ASC), standard rate
Rate	14 percent of (current-year QRE minus 50 percent of average QRE for prior 3 years). Illustrative ignores the prior-3-year base; for a real binder we would compute it from the customer's prior returns.
Estimated credit (illustrative)	**$1,714,300**

5. Audit-defense flag review

Foreign research	PostHog, Inc. is a Delaware C-corporation with engineering activity distributed across multiple geographies including the United Kingdom. Section 41 QRE wages must be limited to US-source activity per Treas. Reg. § 1.41-4(c)(7); UK-resident engineers and other non-US-resident contributors are excluded entirely. A real binder would cross-reference PostHog's payroll register and Form W-2 / Form 1099 records to allocate QRE wages exclusively to US-source activity. Open-source community contributions outside the customer's payroll are excluded. This caveat materially reduces total QRE versus a pure commit-count estimate; the illustrative QRE in this sample assumes a 60 percent US-source allocation, which a real binder would replace with the customer's actual payroll-driven allocation.
Funded research	PostHog is venture-funded SaaS (Series C plus) and self-funded by product revenue; no customer or grantor pays regardless of outcome. The MIT License under which PostHog publishes its source code does not constitute funded research under Section 41(d)(4)(H).
Post-commercial-production	Activity ends when a feature is shipped to production. Subsequent maintenance, dependency-bump, and translation-only commits are categorized separately and excluded from QRE wages.
Routine data collection	None. The activity is software development, not survey or market study.
Internal-use software	Not applicable. PostHog Web Analytics, Customer Data Platform destinations, and the Hog programming language are all customer-facing product surfaces sold as part of PostHog's commercial SaaS offering, not internal IT for PostHog's own operations.
Contract research overweight	A real binder cross-references the customer's contractor invoices and payroll register to confirm the 65-percent contract-research applied amount does not exceed 50 percent of total QRE. PostHog's commit history shows a mix of full-time employees and named open-source contributors; only US-source FTE and US-source contractor work counts toward QRE.

6. Form 6765 Section G mapping

Starting tax year 2026 (processing year 2027), Form 6765 Section G is mandatory for non-exempt filers, requiring per-business-component reporting. Tax year 2024 predates the Section G mandatory-filing trigger; the table below is provided for the customer's CPA in the event Section G later becomes applicable (e.g., through carry-forward of unused credit, or amendment).

Section G field	Component A	Component B
Business Component Name	PostHog Web Analytics (net-new product surface)	Hog Programming Language and Customer Data Platform (CDP) Destinations Engine
Business Component Type	computer_software	computer_software
Information Sought to be Discovered	PostHog's open question entering 2024: whether and how to ship a dedicated Web Analytics product surface (separate from the existing Product Analytics surface) that exposes Google-Analytics-style web metrics (sessions, bounce rate, top pages, conversion goals, channel attribution, Core Web Vitals) at PostHog's scale (multi-billion-event ClickHouse warehouse) without forcing customers to learn HogQL, with cookieless server-hash visitor identity for jurisdictions that disallow third-party cookies, and with custom-channel-type classification users can edit per-team. The bar was a self-serve experience where a customer could sign up, paste a snippet, and see meaningful web analytics within minutes without per-query authoring.	PostHog's open question entering 2024: how to expose the existing event-pipeline transform layer (previously a Python plugin SDK with thick deployment surface) as a customer-authored programming surface that runs at event-pipeline scale without subjecting customers to Python's deployment, dependency, or sandboxing complexity. The bar was a custom programming language (provisional name: Hog) with an OCaml-style functional shape, an interpreter that runs inside the PostHog event pipeline, a source-to-source transpiler so the same Hog functions can run client-side in browsers (where Python is not available), an REPL for customer authoring, and a destinations engine that lets customers wire Hog functions to outbound integrations (Snapchat Pixel, Meta Ads, Brevo, Hubspot, Make, Slack, Zapier, Google Ads) with declarative data-mapping templates and OAuth-managed credentials.

This document is a sample artifact produced by R&D Binder (rdbinder.com), built from PostHog Inc.'s public GitHub commit history for the purpose of demonstrating the binder's structure to prospective customers. Numerical values in the QRE workpaper are illustrative and based on assumed payroll and team size, not the customer's actual books. The customer has not engaged R&D Binder.

R&D Binder operates as a doc-only documentation service per Treasury Circular 230 Section 10.7. The customer's CPA-of-record prepares and signs Form 6765 and retains preparer responsibility for the filed return.

Generated 2026-05-06.