

dlte.ch Internal Google Ads Optimization Tool

Design Documentation · Applicant: dlte.ch s.r.o. · Contact: hana.malinic@gmail.com
Manager Account ID: 959-991-9815 · Document version: 1.0 · Date: 2026-04-26

1. Purpose & Scope

This document describes an **internal-only** tool that allows our small marketing team to analyze and optimize **our own** Google Ads account through an AI-assisted natural-language interface. The tool is not offered to third parties, is not a SaaS product, and does not store, share, or process Google Ads data on behalf of any external customer.

The single purpose is to reduce manual reporting and configuration time for the two campaigns we currently run in our own account (415-313-5068): a B2C campaign for *Daruj Pesničku* (a personalized song gift service) and a B2B lead-generation campaign for *lead.vedno.it* (Slovak company contact data). Both campaigns are owned and operated solely by dlte.ch s.r.o.

Use-case classification: Internal tool for managing the applicant's own Google Ads account. No agency, reseller, or third-party access. No data leaves our infrastructure.

2. Tool Architecture

The tool is a self-hosted Model Context Protocol (MCP) server that exposes the Google Ads API to a locally-running AI assistant. There is no web frontend; all interaction happens through the assistant's desktop CLI on the operator's machine.



2.1 Component summary

Component	Role
Claude Code CLI	Anthropic's AI-assistant terminal. Sends natural-language prompts; the assistant translates them into MCP tool calls.
mcp-proxy	Bridges the stdio MCP transport into a private localhost HTTP endpoint (127.0.0.1:8085). The endpoint is never exposed publicly.
google-ads-mcp server	Python MCP server that exposes typed tool calls (e.g. <code>list_campaigns</code> , <code>add_negative_keywords</code>) over MCP. Validates input, enforces budget caps, and forwards calls to the Google Ads SDK.
google-ads-python SDK	Official Google client library. Performs API calls with our OAuth credentials and developer token.

3. Authentication & Authorization

- **OAuth 2.0 (refresh-token flow).** A single Google account (`hana.malinic@gmail.com`) authorizes the Google Ads API scope (`https://www.googleapis.com/auth/adwords`) once via a desktop-app OAuth client. The refresh token is stored in an environment variable readable only by the Docker container.
- **Developer token** issued under our Manager Account `959-991-9815` .
- **Login customer ID** set to the Manager Account; child account `415-313-5068` is linked under the manager.
- **No third-party identities.** Only employees of dlte.ch s.r.o. have access to the server, and only one Google identity is used.

4. Functionality (API Operations)

4.1 Read-only operations

Tool	Underlying API	Purpose
<code>list_campaigns</code> / <code>get_campaign</code>	GoogleAdsService.SearchStream	List campaigns and fetch metrics (clicks, impressions, cost, conversions) for the operator to review.
<code>get_keyword_performance</code>	GoogleAdsService.SearchStream	Quality score, CPC, conversion rate per keyword.
<code>search_terms_report</code>	GoogleAdsService.SearchStream	Identify wasted spend on irrelevant search queries.
<code>get_account_overview</code>	GoogleAdsService.SearchStream + CustomerService	Summary dashboard of optimization score, spend, structure.

Tool	Underlying API	Purpose
<code>compare_ad_performance</code>	GoogleAdsService.SearchStream	Side-by-side ad metrics for A/B decisions.

4.2 Write operations (only invoked after explicit human approval)

Tool	Underlying API	Purpose
<code>add_negative_keywords</code>	CampaignCriterionService.MutateCampaignCriteria	Add negatives identified from search-term reports.
<code>update_keyword_bid</code>	AdGroupCriterionService.MutateAdGroupCriteria	Adjust manual CPC bids on keywords.
<code>pause_campaign / resume_campaign</code>	CampaignService.MutateCampaigns	Activate/deactivate own campaigns.
<code>update_campaign (budget)</code>	CampaignBudgetService.MutateCampaignBudgets	Modify daily budgets — gated by hard caps (see §5).
<code>create_responsive_search_ad</code>	AdGroupAdService.MutateAdGroupAds	Add new RSA variants for A/B testing.

4.3 Estimated request volume

Daily API operations are expected to be in the low hundreds: ~50–200 read operations per audit run (one or two per business day) and ~5–30 write operations per optimization session. We expect to stay well within Basic-access quotas.

5. Safety Controls

- **Budget caps.** Two environment variables (`GOOGLE_ADS_DAILY_BUDGET_CAP_EUR=10` , `GOOGLE_ADS_LIFETIME_BUDGET_CAP_EUR=100`) define the maximum the assistant is allowed to set on any campaign. A budget update that exceeds the cap is rejected client-side before reaching the API.
- **Human-in-the-loop.** Every write call is initiated by a human operator’s prompt. The AI assistant proposes the change in plain text first; the operator must confirm before the tool call is executed.
- **Localhost binding.** The MCP server listens only on `127.0.0.1` — it is not reachable from the public internet, even from other servers on our infrastructure.
- **Audit log.** All MCP tool calls are written to the assistant’s session transcript, providing a chronological log of every read and write operation.

6. Data Handling

- No Google Ads data is persisted outside the assistant's in-memory session and short-lived API responses.
- No data is sent to any third-party analytics, logging, or BI service.
- Refresh tokens, client secrets, and the developer token are stored in a server-local `.env` file (file permissions 0600, readable only by the operating-system user that runs Docker).
- The server runs in the EU on private infrastructure operated solely by dlte.ch s.r.o.

7. User Workflow Example

1. Operator opens Claude Code CLI and types: *"Show me wasted spend on the Daruj Pesničku campaign over the last 14 days."*
2. The assistant calls `search_terms_report` (read-only) over MCP. The MCP server forwards a `GoogleAdsService.SearchStream` GAQL query.
3. The assistant summarizes results in natural language and proposes negative keywords.
4. The operator reviews the proposal and types: *"Yes, add those negatives."*
5. The assistant calls `add_negative_keywords` (write). The MCP server validates the call against budget/scope rules and invokes `CampaignCriterionService.MutateCampaignCriteria`.
6. The assistant reports the outcome.

8. Compliance & Policy Alignment

- Tool operates exclusively on accounts owned by dlte.ch s.r.o. under our manager account.
- No automated bulk-creation of campaigns. All ad copy is human-authored or human-approved.
- No scraping of competitor data, no integration with restricted verticals, no use of the API to circumvent Google Ads policies.
- The tool will use the latest stable Google Ads API version (currently v21) and the official `google-ads-python` SDK to ensure forward compatibility.

9. Source & Maintenance

- The MCP server source is based on the open-source project `grantweston/google-ads-mcp-complete` (MIT license), self-hosted in a private Docker container.
- OAuth credentials and developer token are managed by the dlte.ch operations team; rotation procedures are documented internally.
- Single point of contact: **hana.malinic@gmail.com**.