Abstract

This article demonstrates how to integrate the Google Workspace Extension for Gemini CLI with Google Antigravity. It addresses a Model Context Protocol (MCP) tool naming incompatibility using a custom proxy script, enabling seamless, authenticated automation of Google Workspace tasks directly within the Antigravity IDE environment.

Introduction

Since its release, the **Gemini CLI** has been rapidly adopted across various development scenarios. Ref Its utility increased significantly with the introduction of **Gemini CLI Extensions**, which simplify the installation and management of Model Context Protocol (MCP) servers. Ref Most recently, the **Google Workspace Extension for Gemini CLI** was released by Google, providing an MCP server specifically designed to manage Workspace automation. Ref A distinct advantage of this extension is its streamlined authorization process — authentication runs automatically when the Gemini CLI is launched, making it highly efficient.

Simultaneously, **Google Antigravity** has arrived, promising to revolutionize the IDE landscape. Ref Because Google Antigravity supports MCP servers, it opens the door to complex task processing involving external data. I previously explored this potential in my article, "Agentic Automation in Google Workspace: Bridging Antigravity and Gemini 3.0," where I utilized a custom-built MCP server by me. Ref

However, leveraging the official Google Workspace Extension within Antigravity offers a superior experience due to its pre-configured client ID and simplified authorization flow. This article details the technical steps required to bridge these two powerful tools.

Prerequisites

Before proceeding, ensure your environment meets the following requirements:

1. **Gemini CLI:** Must be installed and authenticated.
2. **Extensions:** The following Gemini CLI Extension must be installed and verified:

Google Workspace Extension: Confirm this works via the Gemini CLI. Launching the CLI after installation should trigger the automatic authorization process.

This guide assumes that your Gemini CLI environment allows the use of extensions.

Integration Workflow

To implement this workflow, we must establish a communication bridge between Google Antigravity and the extension's MCP server.

1. Google Antigravity Setup

First, ensure your IDE environment is ready. Please refer to the official release and installation guide at https://antigravity.google/.

2. Connecting the MCP Server

Identifying the Compatibility Issue

First, locate the installation directory of your Google Workspace Extension using the command:

```
gemini extensions list
```

The output directory should look similar to: /home/{username}/.gemini/extensions/google-workspace

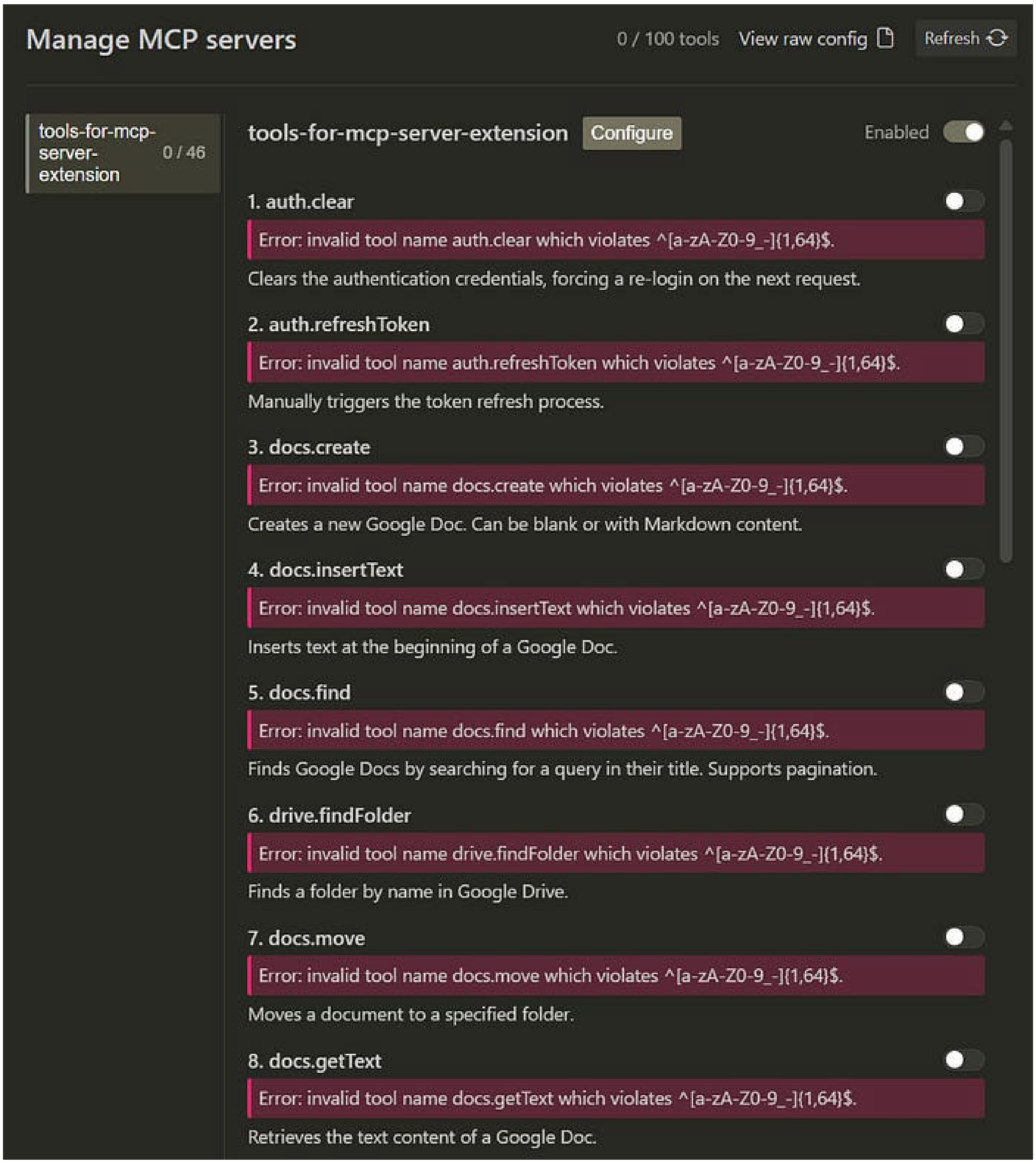
The actual MCP server script is located here: /home/{username}/.gemini/extensions/google-workspace/dist/index.js

To register this server with Google Antigravity, you would typically update the mcp_config.json file as follows:

```
{
  "mcpServers": {
    "tools-for-mcp-server-extension": {
      "command": "node",
      "args": [
        "/home/{username}/.gemini/extensions/google-workspace/dist/index.js"
      ]
    }
  }
}
```

However, after refreshing the MCP server in Antigravity, you will likely encounter an error:

Press enter or click to view image in full size



tools-for-mcp-server-extension

1. auth.clear

Error: invalid tool name auth.clear which violates ^[a-zA-Z0-9_-]{1,64}$.

Clears the authentication credentials, forcing a re-login on the next request.

2. auth.refreshToken

Error: invalid tool name auth.refreshToken which violates ^[a-zA-Z0-9_-]{1,64}$.

Manually triggers the token refresh process.

3. docs.create

Error: invalid tool name docs.create which violates ^[a-zA-Z0-9_-]{1,64}$.

Creates a new Google Doc. Can be blank or with Markdown content.

4. docs.insertText

Error: invalid tool name docs.insertText which violates ^[a-zA-Z0-9_-]{1,64}$.

Inserts text at the beginning of a Google Doc.

5. docs.find

Error: invalid tool name docs.find which violates ^[a-zA-Z0-9_-]{1,64}$.

Finds Google Docs by searching for a query in their title. Supports pagination.

6. drive.findFolder

Error: invalid tool name drive.findFolder which violates ^[a-zA-Z0-9_-]{1,64}$.

Finds a folder by name in Google Drive.

7. docs.move

Error: invalid tool name docs.move which violates ^[a-zA-Z0-9_-]{1,64}$.

Moves a document to a specified folder.

8. docs.getText

Error: invalid tool name docs.getText which violates ^[a-zA-Z0-9_-]{1,64}$.

Retrieves the text content of a Google Doc.

The Error: Error: invalid tool name ###.$$$ which violates ^[a-zA-Z0-9_-]{1,64}$.

The Cause: The Google Workspace Extension uses tool names containing dots (e.g., docs.create). Currently, the MCP client in Google Antigravity strictly enforces a regex that only allows alphanumeric characters, underscores, and hyphens. Consequently, the tools fail to load.

Implementing the Proxy Solution

While renaming all tools in the source code would resolve this, it is not a sustainable solution as it breaks with every extension update. Instead, I developed a **Proxy Script**. This Node.js script acts as a middleware:

1. **Sanitization:** It intercepts tool definitions sent from the server and replaces forbidden characters (like .) with hyphens for the Antigravity client.
2. **Restoration:** When Antigravity calls a tool, the proxy intercepts the request and restores the original name (e.g., converting docs-create back to docs.create) before passing it to the real MCP server.

Create a file named proxy.js in the same directory as the extension (/home/{username}/.gemini/extensions/google-workspace/dist/) and paste the following code:

```
/**
 * Proxy for the MCP server of Google Workspace Extension for Gemini CLI.
 * It converts characters in tool names that do not match the regex rule ( ^[a-zA-Z0-9_-]{1,64}$ )
 * into hyphens for the client. During execution, it restores the original name before sending it to
the server.
 * Author: Kanshi Tanaike
 * https://github.com/tanaikech
 *
 * version 1.0.0
 */

const { spawn } = require("child_process");
const readline = require("readline");
const path = require("path");

const targetScript = process.argv[2];

if (!targetScript) {
  console.error(
    "Error: Please provide the target script file. e.g., node proxy.js index.js"
  );
  process.exit(1);
}

const toolNameMap = new Map();

const serverProcess = spawn("node", [targetScript], {
  stdio: ["pipe", "pipe", process.stderr],
});

function sanitizeToolName(originalName) {
  const sanitized = originalName.replace(/[^a-zA-Z0-9_-]/g, "-");
  if (sanitized !== originalName) {
    toolNameMap.set(sanitized, originalName);
  }
  return sanitized;
}

function restoreToolName(sanitizedName) {
  return toolNameMap.get(sanitizedName) || sanitizedName;
}

const inputRl = readline.createInterface({
  input: process.stdin,
  output: null,
  terminal: false,
});

inputRl.on("line", (line) => {
  try {
    const request = JSON.parse(line);
    if (
      request.method === "tools/call" &&
      request.params &&
      request.params.name
    ) {
      const originalName = restoreToolName(request.params.name);
      request.params.name = originalName;

      serverProcess.stdin.write(JSON.stringify(request) + "\n");
    } else {
      serverProcess.stdin.write(line + "\n");
    }
  } catch (e) {
    serverProcess.stdin.write(line + "\n");
  }
});

const outputRl = readline.createInterface({
  input: serverProcess.stdout,
  output: null,
  terminal: false,
});

outputRl.on("line", (line) => {
  try {
    const response = JSON.parse(line);
    if (response.result && Array.isArray(response.result.tools)) {
      response.result.tools = response.result.tools.map((tool) => {
        if (tool.name) {
          tool.name = sanitizeToolName(tool.name);
        }
        return tool;
      });
      console.log(JSON.stringify(response));
    } else {
      console.log(line);
    }
  } catch (e) {
    console.log(line);
  }
});

serverProcess.on("close", (code) => {
  process.exit(code);
});

process.on("SIGINT", () => {
  serverProcess.kill("SIGINT");
  process.exit();
});
```

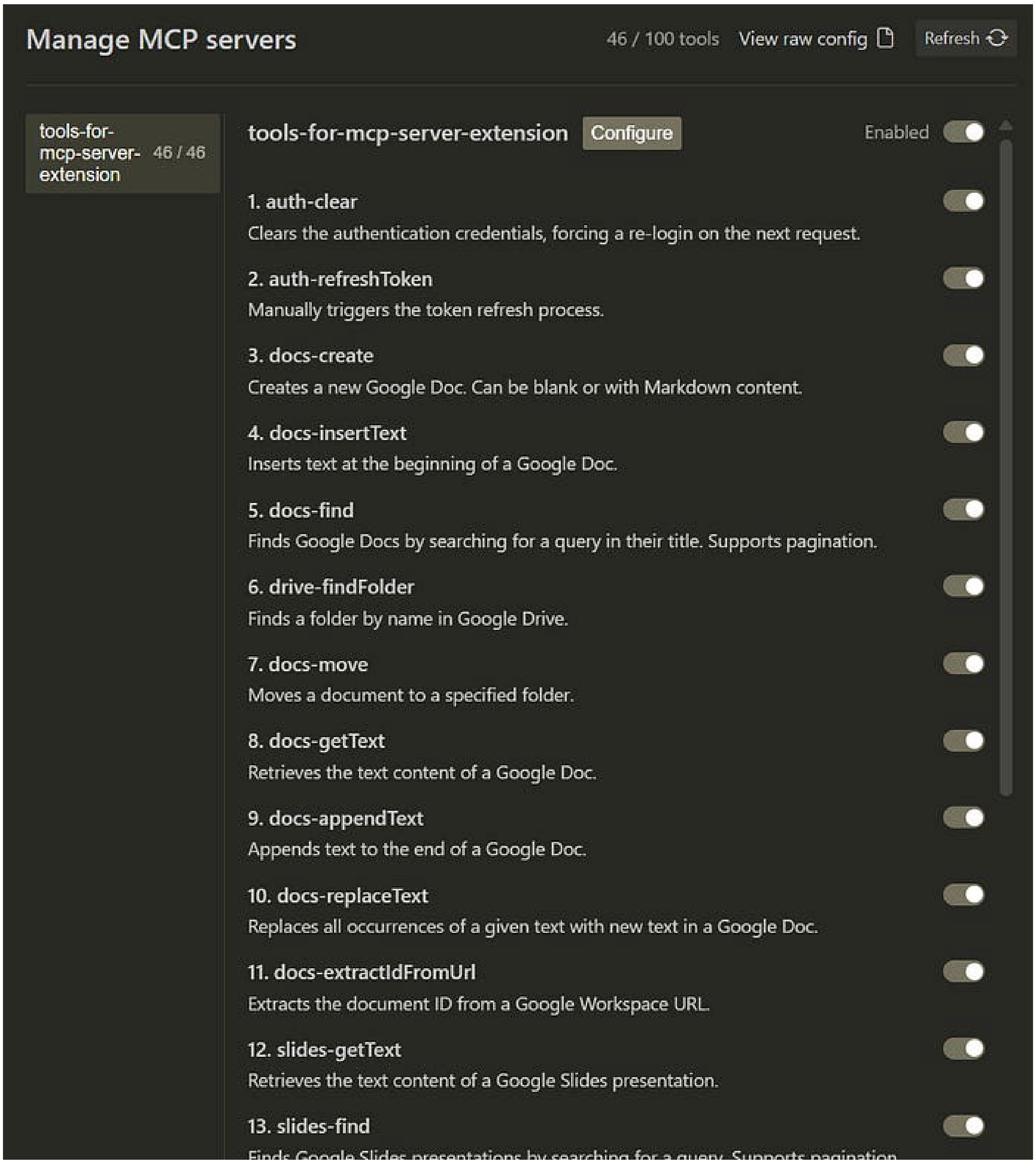
Configuration Update

Now, update your mcp_config.json to route traffic through the proxy:

```
{
  "mcpServers": {
    "tools-for-mcp-server-extension": {
      "command": "node",
      "args": [
        "/home/{username}/.gemini/extensions/google-workspace/dist/proxy.js",
        "/home/{username}/.gemini/extensions/google-workspace/dist/index.js"
      ]
    }
  }
}
```

Upon refreshing the MCP servers, the error will disappear, and the Google Workspace tools will load correctly as shown below:

Press enter or click to view image in full size



Manage MCP servers

46 / 100 tools View raw config Refresh

tools-for-mcp-server-extension 46 / 46

tools-for-mcp-server-extension Configure Enabled

1. auth-clear
Clears the authentication credentials, forcing a re-login on the next request.

2. auth-refreshToken
Manually triggers the token refresh process.

3. docs-create
Creates a new Google Doc. Can be blank or with Markdown content.

4. docs-insertText
Inserts text at the beginning of a Google Doc.

5. docs-find
Finds Google Docs by searching for a query in their title. Supports pagination.

6. drive-findFolder
Finds a folder by name in Google Drive.

7. docs-move
Moves a document to a specified folder.

8. docs-getText
Retrieves the text content of a Google Doc.

9. docs-appendText
Appends text to the end of a Google Doc.

10. docs-replaceText
Replaces all occurrences of a given text with new text in a Google Doc.

11. docs-extractIdFromUrl
Extracts the document ID from a Google Workspace URL.

12. slides-getText
Retrieves the text content of a Google Slides presentation.

13. slides-find

. was replaced with -.

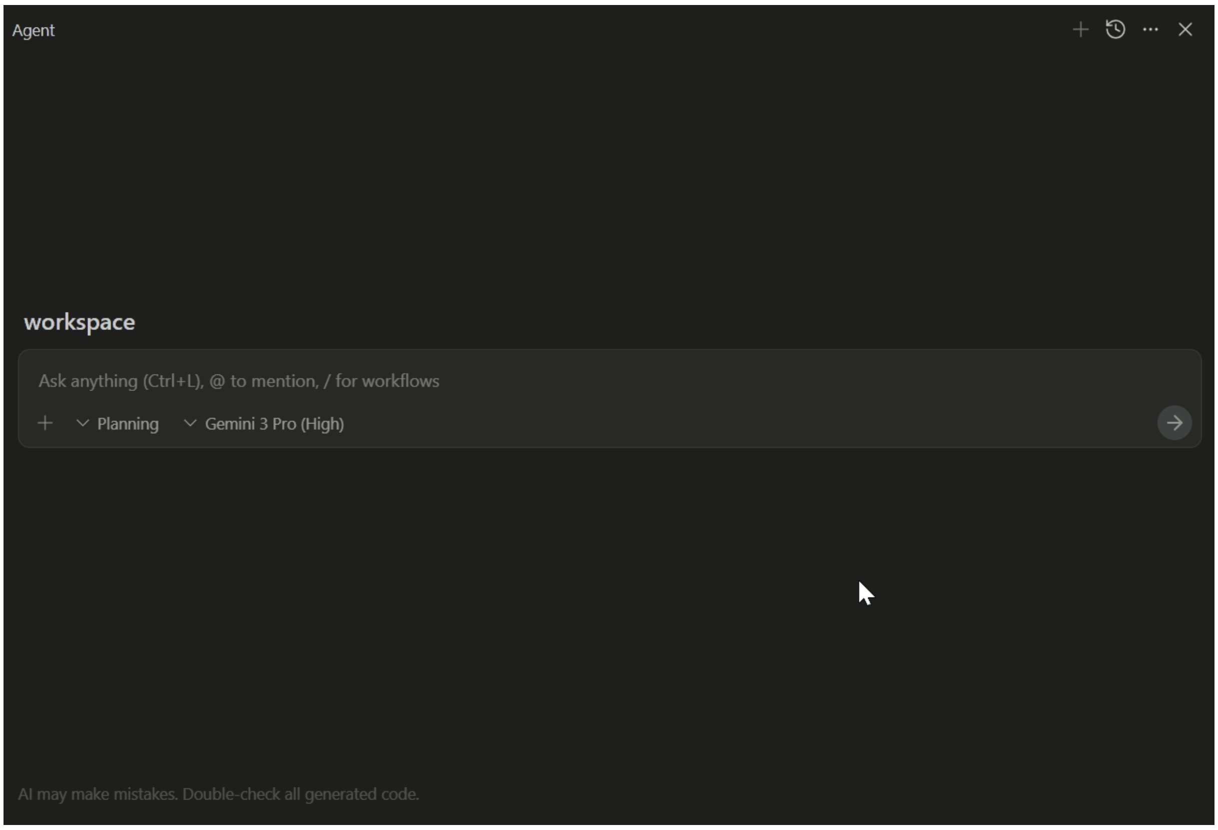
3. Validation and Testing

To verify the integration, I performed the following tests using natural language prompts within Antigravity.

Test 1: Google Drive Search Prompt:

Search a file "sample file for searching" on Google Drive.

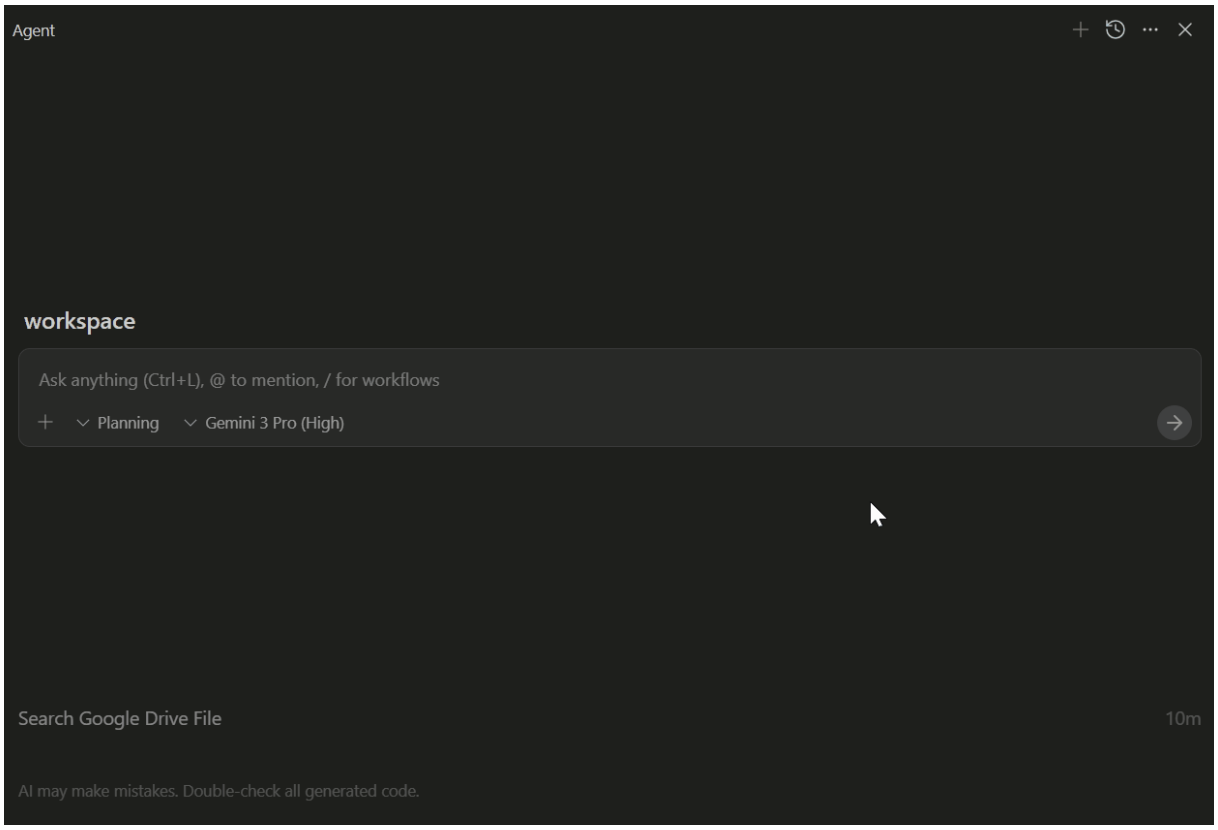
Press enter or click to view image in full size


Agent
workspace
Ask anything (Ctrl+L), @ to mention, / for workflows
Planning
Gemini 3 Pro (High)
AI may make mistakes. Double-check all generated code.

Test 2: Google Docs Retrieval Prompt:

Get a text from a Google Document named "sample Google Document".

Press enter or click to view image in full size


Agent
workspace
Ask anything (Ctrl+L), @ to mention, / for workflows
Planning
Gemini 3 Pro (High)
Search Google Drive File
10m
AI may make mistakes. Double-check all generated code.

As demonstrated, the tools function seamlessly through the pre-authorization at Gemini CLI, bridging the gap between Antigravity and the Workspace Extension.

Summary

- **Integration of Ecosystems:** This solution successfully combines the Google Workspace Extension for Gemini CLI with the Google Antigravity IDE.
- **Simplified Authentication:** By leveraging the Gemini CLI extension, users benefit from a robust and pre-configured authorization flow compared to custom MCP servers.
- **Regex Incompatibility Solved:** A technical conflict regarding tool naming conventions (Antigravity's regex vs. Extension's dot notation) was identified.
- **Proxy Middleware:** A Node.js proxy script was implemented to sanitize tool names dynamically, ensuring compatibility without modifying the source extension code.
- **Proven Functionality:** Tests confirm that complex Workspace tasks, such as searching Drive and reading Docs, can be executed directly from Antigravity.

The rule of the tool name for the MCP server might be updated in the future. At that time, the current workaround using a proxy script might be unnecessary.