Comprehensive Analysis of Firebase Authentication Domain Routing and Antigravity AI Agent Skill Implementation

Introduction to the Authentication Routing Architecture

The integration of federated identity providers—such as Google, Facebook, or Apple—into modern web applications relies fundamentally on the OAuth 2.0 authorization framework and the OpenID Connect (OIDC) protocol.[1] When an application utilizes Firebase Authentication as its identity management backend, the Firebase platform automatically provisions a default hosting subdomain for the project.[2] This automatically generated domain takes the format of https://.firebaseapp.com and acts as the secure intermediary, or the default authentication handler, for all OAuth, OIDC, and Security Assertion Markup Language (SAML) sign-in redirects.[2]

The utilization of this default handler presents a significant architectural advantage for complex deployments. It permits developers to use multiple domains with the same federated providers, allows the sharing of a single callback URL across disparate services, and seamlessly integrates with identity providers that restrict applications to a single authorized redirect URI per application.[2] However, the primary drawback of this default configuration is an interrupted and inconsistent user experience. During the sign-in flow, users navigating away from a primary application domain—in this instance, thefunfanreporter.com—will briefly observe the underlying thefunfanreporter.firebaseapp.com URL in their browser's address bar before being redirected back to the primary application state.[2]

Beyond mere branding inconsistencies, failing to align the authentication domain with the application's primary domain introduces severe technical liabilities related to browser security policies. Modern web browsers have implemented stringent storage partitioning and third-party cookie blocking mechanisms to protect user privacy. Starting with Google Chrome version M115, Mozilla Firefox version 109, and Apple Safari version 16.1, implementing a custom authentication domain is strictly required for redirect sign-in flows to function properly.[3] The Firebase Authentication JavaScript SDK relies on a cross-origin hidden iframe that connects to the application's Firebase Hosting domain to manage the authentication state seamlessly.[3] If the application operates on thefunfanreporter.com but the hidden iframe attempts to establish a session via thefunfanreporter.firebaseapp.com, the browser will classify the interaction as an illicit third-party storage access attempt and block it, resulting in a silent

failure of the authentication state.[3] To resolve both the branding discrepancy and the critical storage partitioning constraints, developers must customize the authentication handler to operate entirely on the application's native domain.[2]

Pricing Tier Implications on Custom Domain Capabilities

A pervasive hypothesis when encountering functional constraints within the Google Cloud or Firebase environments is that the desired feature is gated behind a premium billing tier. It is imperative to systematically address whether the Firebase "Spark" plan—the platform's no-cost tier—restricts the ability to utilize a custom domain for authentication routing.[4]

Spark Plan Entitlements versus Blaze Plan Consumption

The Firebase pricing model is bifurcated into two primary tiers: the Spark Plan, which is free but enforces strict resource quotas intended for early development, and the Blaze Plan, which is a pay-as-you-go model calculated on actual resource consumption.[4] An exhaustive analysis of the Spark plan entitlements reveals that the configuration of custom domains and the provisioning of associated SSL certificates are explicitly included at no cost.[4] The inability to display thefunfanreporter.com during the Google Sign-In flow is strictly a configuration omission, not a billing or tier-level restriction.[4]

Under the Spark plan, the Firebase Hosting service provides a robust set of features without requiring a billing instrument. The platform allows for multiple sites per project and permits custom domain mapping (e.g., routing thefunfanreporter.com or a dedicated subdomain like auth.thefunfanreporter.com).[4] Furthermore, Firebase Hosting provides automated SSL certificate provisioning and renewals via Let's Encrypt and Google Trust Services across a global Content Delivery Network (CDN).[5]

The following table delineates the specific hosting limitations enforced on the Spark plan, none of which prevent the implementation of a custom authentication domain:

Firebase Hosting Resource	**Spark Plan (No-Cost) Limitation**	**Blaze Plan (Pay-as-you-go) Scaling**
Custom Domain & SSL	Included at no cost [4]	Included
Multiple Sites per Project	Included at no cost [4]	Included

Total Storage	Capped at 10 GB [4]	Billed per GB over quota
Data Transfer (Bandwidth)	Capped at 360 MB per day (approx. 10 GB/month) [4]	Billed per GB over quota
Subdomains per Apex Domain	Limited to 20 subdomains due to SSL minting limits [5]	Limited to 20 subdomains

Authentication Quotas and the Identity Platform Upgrade

While the custom domain infrastructure itself is provided free of charge, the authentication volume under the Spark tier is tightly regulated, particularly if the project has been upgraded to utilize advanced enterprise features. Standard Firebase Authentication permits an unlimited number of registered user accounts and unlimited email/password sign-ins.[7] However, Firebase offers an optional upgrade known as "Firebase Authentication with Identity Platform".[8]

If thefunfanreporter.com relies on standard Firebase Authentication without the Identity Platform enabled, the service remains entirely free with no upper limit on monthly active users.[11] Conversely, if the project has been upgraded to the Identity Platform, the Spark tier limits the application to 3,000 Daily Active Users (DAUs).[7] Surpassing these Identity Platform limits requires linking a Google Cloud billing account to transition to the Blaze plan, where pricing scales based on Monthly Active Users (MAUs). On the Blaze plan, the first 49,999 MAUs remain free, with subsequent MAUs billed incrementally (e.g., $0.0055 per MAU between 50k and 100k).[12]

The following table outlines the stringent daily and monthly limits applied to authentication operations, emphasizing the constraints of the Spark plan when Identity Platform is active:

Authentication Operation / Metric	**Spark Plan Limit**	**Blaze Plan Limit**
Standard Monthly Active Users	50,000 MAU [4]	Pay-as-you-go after 50,000 [12]
Identity Platform Daily Active Users (Tier 1)	3,000 DAU [7]	Unlimited (Billed by MAU) [9]
Identity Platform Daily Active Users (Tier 2 -	2 DAU [7]	Unlimited (Billed by MAU)

SAML/OIDC)		
Anonymous User Accounts	Capped at 100 million [7]	Capped at 100 million
New Account Creation Rate	100 accounts/hour per IP address [7]	100 accounts/hour per IP address
Account Deletion Rate	10 accounts/second [7]	10 accounts/second
Address Verification Emails	1,000 emails/day [7]	100,000 emails/day [7]
Password Reset Emails	150 emails/day [7]	N/A

A highly restrictive aspect of the Spark plan relates to Phone Number Verification (SMS). The Spark plan does not support production-level phone authentication, offering only a negligible testing quota of 10 free SMS messages per day.[4] If the application requires SMS verification, an immediate upgrade to the Blaze plan is mandatory. Once on the Blaze plan, SMS costs are highly variable and strictly determined by the destination region.[4] The following table illustrates the vast disparity in regional SMS pricing:

Destination Region	**Cost per SMS (USD)**	**Supported Carriers (where applicable)**
North America (US/Canada)	$0.01 [4]	All major carriers
United Kingdom	$0.04 [4]	All major carriers
Germany (Tier 1)	$0.0743 to $0.10 [4]	Deutsche Telekom [4]
Indonesia (Tier 1)	$0.1350 to $0.33 [4]	Telkomsel [4]
Brazil	$0.02 [4]	All major carriers
Russian Federation	$0.33 [4]	All major carriers

Burundi / Madagascar	$0.43 [4]	All major carriers
Default Global Rate	$0.46 [4]	Unlisted regions [4]

In summary, the assumption that the Spark tier prevents the removal of the firebaseapp.com domain is demonstrably false. As long as thefunfanreporter.com operates within the defined 3,000 DAU limit (if upgraded to Identity Platform) or relies on standard Firebase Authentication, the platform fully supports domain masking without incurring financial charges.[4]

Architectural Mechanisms of Custom Domain Routing

To successfully implement domain masking and ensure the Google Sign-In popup displays the correct URL, it is critical to understand the four disparate systems that must be intricately synchronized. A failure in any single layer will result in broken authentication flows or 404 routing errors. The architecture relies on the orchestration of Firebase Hosting, the Firebase Authentication Authorized Domains whitelist, the Google Cloud Platform (GCP) OAuth 2.0 Credentials, and the client-side Firebase Initialization SDK.[13]

The Role of Firebase Hosting and DNS Propagation

Even if the primary web application is hosted on an external platform—such as Vercel, Netlify, or Amazon Web Services—Firebase Hosting must be utilized to establish the custom domain within the Firebase ecosystem.[13] Firebase Hosting acts as the primary routing engine. When a custom domain is connected, Firebase mints an SSL certificate and prepares its global CDN to intercept traffic directed at a specific, internal virtual route: /__/auth/handler.[5]

Connecting a domain requires proving cryptographic ownership. The developer must access their domain provider's DNS management console and deploy a specific TXT record provided by the Firebase console.[5] This TXT record, formatted as fah-claim=, acts as an ACME challenge token, authorizing the Cloud Certificate Manager to provision and automatically renew SSL certificates via authorities like Let's Encrypt.[5] Following successful verification, the developer must update the DNS A records to point to Firebase's IPv4 addresses, or utilize a CNAME record if establishing a dedicated authentication subdomain.[5]

The Google Cloud Console OAuth 2.0 Redirect URI

When a user initiates a Google Sign-In, the application redirects the user to Google's centralized authorization servers. This HTTP request includes a specific redirect_uri parameter, which tells Google exactly where to send the user back after successful authentication.[1] Google's security model strictly enforces that this redirect_uri must exactly match a pre-approved, whitelisted URL listed within the project's GCP OAuth 2.0 Client ID

configuration.[1]

If the client application requests a redirect to https://thefunfanreporter.com/__/auth/handler, but only the default https://thefunfanreporter.firebaseapp.com/__/auth/handler is whitelisted in the GCP console, Google's servers will immediately reject the request. This security mismatch typically manifests as a redirect_uri_mismatch error or an auth/unauthorized-continue-uri exception, preventing the sign-in popup from loading.[1]

The Client-Side authDomain Parameter Override

The Firebase client Software Development Kit (SDK)—whether implemented in JavaScript for the web, Swift for iOS, or Kotlin for Android—dictates which domain the application utilizes to open the authentication popup or instantiate the cross-origin iframe.[2] By default, when a developer copies the initialization snippet from the Firebase Console, the authDomain field is hardcoded to the firebaseapp.com address.[2] If this variable is not explicitly overridden in the source code, the application will continually default to the Firebase-branded domain, rendering all backend DNS and GCP configurations completely inert.[2]

Exhaustive Technical Remediation Strategy

The following protocol details the precise, step-by-step technical configuration required to route Firebase Authentication through the custom domain thefunfanreporter.com.

It is highly recommended by domain experts to use a dedicated subdomain—such as auth.thefunfanreporter.com—exclusively for Firebase Authentication if the primary application is hosted on an external provider like Vercel.[13] Mapping the root apex domain simultaneously to Vercel for frontend hosting and to Firebase for authentication routing creates unresolvable DNS conflicts, often resulting in 404 Not Found errors when the OAuth provider attempts to hit the /__/auth/handler endpoint.[14] If the entire application is hosted natively on Firebase Hosting, the root apex domain (thefunfanreporter.com) can be used directly.[5] The instructions below assume the use of a dedicated auth subdomain, which is the safest and most robust architectural approach.

Phase 1: Configuring Firebase Hosting and DNS Verification

The foundational step is introducing the custom domain to the Firebase Hosting infrastructure to trigger the generation of the necessary SSL certificates and establish the CDN routing paths.[5]

Action Step	Detailed Instruction	System Interface

1. Navigate to Hosting	Access the Firebase Console, select the target project, and click **Build > Hosting** in the left-hand menu.[5]	Firebase Console
2. Initialize Wizard	If Hosting has not been initialized, execute the "Get Started" wizard to provision the default site.[5]	Firebase Console
3. Add Domain	Click the **Add custom domain** button and input auth.thefunfanreporter.com .[5] Do not check the redirect option.[18]	Firebase Console
4. Extract Verification	Copy the generated TXT record string provided by the setup wizard.[5]	Firebase Console
5. DNS Verification	Access the domain registrar's DNS management portal (e.g., GoDaddy, Cloudflare). Create a TXT record where the **Host** is auth and the **Value** is the Firebase verification string.[5]	Domain Registrar
6. DNS Routing	Create a CNAME record where the **Host** is auth and the **Value** points directly to thefunfanreporter.firebaseapp.com.[13] *Note: If using the root domain instead of a subdomain, you must use A records pointing to the provided Firebase IPv4 addresses.*	Domain Registrar

Phase 2: Whitelisting the Domain in Firebase Authentication

Once the domain is connected to Firebase Hosting, the Authentication backend must be explicitly instructed to accept requests originating from this newly established domain.

Action Step	Detailed Instruction	System Interface
1. Navigate to Auth Settings	In the Firebase Console, navigate to **Build > Authentication**.[13]	Firebase Console
2. Locate Authorized Domains	Select the **Settings** tab (or **Sign-in method** tab) and scroll down to the **Authorized domains** list.[13]	Firebase Console
3. Add the Custom Domain	Click the **Add Domain** button. Enter auth.thefunfanreporter.com exactly as configured in Phase 1 and click Add.[13]	Firebase Console

Phase 3: Updating Google Cloud Platform (GCP) OAuth Restrictions

Because Google Sign-In operates as an independent federated OAuth provider, Google's core infrastructure must be updated to permit the custom domain to send and receive secure authorization tokens. Failure to complete this phase guarantees a broken sign-in popup.

Action Step	Detailed Instruction	System Interface
1. Access GCP Credentials	Navigate to the Google Cloud Console (console.cloud.google.com). Ensure the correct Firebase project is selected from the top dropdown. Navigate to **APIs & Services > Credentials**.[13]	Google Cloud Console
2. Edit Web Client	Under the **OAuth 2.0 Client IDs** header, locate the auto-generated	Google Cloud Console

	credentials typically named "Web client (auto created by Google Service)" and click the pencil icon to edit.[16]	
3. Add JavaScript Origin	Scroll to the **Authorized JavaScript origins** section. Click **Add URI** and input https://auth.thefunfanreporter.com.[19]	Google Cloud Console
4. Add Redirect URI	Scroll to the **Authorized redirect URIs** section. Click **Add URI** and input the precise callback path: https://auth.thefunfanreporter.com/__/auth/handler.[3] The trailing /__/auth/handler is mandatory.[3]	Google Cloud Console
5. Save Configuration	Click **Save** at the bottom of the page to apply the OAuth policy updates.[20]	Google Cloud Console

Phase 4: Modifying the Client-Side SDK Configuration

The backend infrastructure is now fully prepared to handle custom domain routing. The final critical phase is directing the client application to utilize this new pathway instead of the default configuration. The developer must locate the Firebase initialization code within the web or mobile application's source files.

The configuration syntax varies significantly depending on the platform and SDK version in use. The following table provides the exact code modifications required across various environments to override the default authDomain:

Platform / SDK Version	**Required Code Modification for Custom Domain Override**

Web JavaScript (Version 9 Modular)	Update the configuration object before initialization:
	
	const firebaseConfig = {
	
	apiKey: "...",
	
	authDomain: 'auth.thefunfanreporter.com',
	
	};
	
	const app = initializeApp(firebaseConfig); [2]

Web JavaScript (Version 8 Namespaced)	Update the initialization call directly: firebase.initializeApp({ apiKey: '...', authDomain: 'auth.thefunfanreporter.com' }); [2]
Android (Java)	Set the custom domain on the instance: FirebaseAuth.getInstance().setCustomAuthDomain("auth.thefunfanreporter.com"); [2]
Android (Kotlin + KTX)	Set the custom domain using Kotlin extensions: Firebase.auth.setCustomAuthDomain("auth.thefunfanreporter.com") [2]
iOS (Swift)	Modify the Auth object property: let auth = Auth.auth() auth.customAuthDomain = "auth.thefunfanreporter.com" [2]
iOS (Objective-C)	Modify the Auth object property:

	FIRAuth *auth =; auth.customAuthDomain("auth.thefunfanreporter.com"); [2]

Phase 5: Customizing Email Templates for Brand Consistency

To ensure comprehensive brand consistency, the custom domain should also be utilized for all outbound authentication emails, such as password recovery flows and email address verification messages.[22] Without this step, users will receive emails originating from a generic Firebase address, which damages trust and increases the likelihood of the communication being flagged as spam.

1. In the Firebase Console, navigate to the **Authentication** section and open the **Templates** page.[22]
2. For each active email template, click the edit icon (represented by a pen).[22]
3. Click the **customize domain** option and enter the custom domain (thefunfanreporter.com).[22]
4. Firebase will generate a new set of TXT and CNAME records. These records represent Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM) configurations.[22]
5. These records must be added to the domain registrar's DNS settings. This cryptographic signature proves to receiving mail servers (like Gmail or Outlook) that Firebase is authorized to send emails on behalf of thefunfanreporter.com, ensuring high deliverability.[22] Furthermore, if the project utilizes Identity Platform multi-tenancy, the tenant metadata must be explicitly updated to allow the tenant to inherit these custom domains and SMTP settings.[22]

Advanced Troubleshooting and Error Resolution

Domain configuration interfaces directly with distributed Domain Name System (DNS) networks and global CDN caching layers, meaning errors can manifest in non-obvious ways. A comprehensive understanding of potential failure states is required to finalize the implementation.

SSL Minting Statuses and Certificate Errors

After updating the DNS records in Phase 1, the Firebase Hosting console will display various statuses indicating the progress of the SSL provisioning process.[5]

- **Needs Setup:** This status indicates that the DNS A or CNAME records have not successfully propagated from the domain provider to the Firebase Hosting servers.[5] DNS propagation can take up to 24 hours globally. Developers should utilize tools like the Google Admin Toolbox Dig service to verify external DNS propagation.[5]

- **Pending:** This suggests that the DNS routing is correct, but Firebase has not yet finished provisioning the SSL certificate.[5] A common blocking factor at this stage is a highly restrictive CAA (Certificate Authority Authorization) record on the root domain. If a CAA record exists, it must explicitly be modified to allow letsencrypt.org and pki.goog to mint certificates on behalf of the domain.[5]
- **Minting Certificate:** An SSL certificate is currently being produced. During this transient stage, the site may serve an invalid certificate.[5]
- **NET::ERR_CERT_COMMON_NAME_INVALID:** If a user encounters this severe browser security warning when attempting to initiate the Google Sign-In flow, it confirms that the SSL certificate for the custom domain has not yet been fully provisioned across the global edge network.[18] The authentication flow will strictly fail until the status updates to **Connected**.

The 404 Not Found Routing Error

Developers frequently report encountering a 404 Not Found error when the popup window redirects to https://thefunfanreporter.com/__/auth/handler.[17] This error occurs almost exclusively when the developer attempts to use their root apex domain for both their primary web hosting (utilizing an external provider like Vercel or Netlify) and their Firebase Authentication routing simultaneously.[14]

If the A records for the root domain point to Vercel's servers, the Vercel routing engine receives the OAuth callback request aimed at /__/auth/handler. Because Vercel's infrastructure does not possess this internal, Firebase-specific route, it correctly throws a 404 error, terminating the authentication sequence.[17] The definitive resolution to this architectural conflict is the implementation of the dedicated auth subdomain detailed in the step-by-step guide. The root domain points to the external host, while the auth subdomain points exclusively to Firebase via a CNAME record, cleanly separating the traffic.[13]

Firebase Authentication Error Code Index

When the configuration layers are misaligned, the Firebase Authentication SDK will emit specific error codes. The following table maps these common errors to their corresponding resolutions:

Error Code	Diagnostic Description	Required Resolution
auth/unauthorized-continue-uri	The domain where the application is hosted has not been whitelisted by the Firebase backend.[15]	Return to Phase 2 and add the exact custom domain to the Authorized Domains list in the Firebase

		Console.[15]
redirect_uri_mismatch	A Google OAuth-specific error indicating that the exact URL requested by the client does not match the URL configured in the GCP console.[1]	Return to Phase 3. Ensure the Authorized redirect URI ends exactly with /__/auth/handler and uses the https:// scheme.[1]
auth/too-many-requests	The application has exceeded the Spark plan's daily active user limits or the IP-based account creation limits.[7]	Upgrade to the Blaze plan or wait for the daily quota to reset.[7]
auth/session-cookie-expired	The provided Firebase session cookie has expired, invalidating the current state.[15]	Force the user to re-authenticate to generate a fresh session token.[15]
auth/session-cookie-revoked	The user's session token has been explicitly revoked by the backend administrator.[15]	Force the user to re-authenticate.[15]

The Antigravity AI Agentic Platform

To fulfill the user's objective of automating this entire diagnostic and remediation process without manual intervention, the workflow must be integrated into Google Antigravity. Introduced alongside the powerful Gemini 3 model architecture on November 18, 2025, Antigravity represents a fundamental paradigm shift in software engineering tools.[25] While it is built upon a heavily modified fork of the open-source Visual Studio Code foundation, it abandons the traditional concept of an Integrated Development Environment (IDE) in favor of an "agent-first" or "agentic" development platform.[27]

Unlike traditional inline coding assistants (such as GitHub Copilot or earlier iterations of AI tools) which primarily focus on autocompleting individual lines of text, Antigravity presupposes that the Artificial Intelligence is an autonomous actor.[28] The platform bifurcates the developer experience into two distinct primary windows: the familiar Editor View for synchronous, manual coding, and the newly introduced Manager Surface, or Agent Manager.[26]

The Agent Manager acts as a "mission control" center.[28] From this interface, developers shift their role from manual typists to architectural managers, orchestrating a workforce of digital

agents by delegating high-level, complex, end-to-end tasks.[25] These agents operate asynchronously across multiple computational surfaces—they can edit files, execute commands within the terminal, and navigate a headless browser to visually verify their work.[26] To establish trust and provide an audit trail without forcing the developer to read through thousands of lines of raw terminal logs, the agents rely on "Artifacts." Artifacts are tangible, verifiable deliverables, such as markdown implementation plans, task checklists, code diffs, and browser screenshots, generated at every stage of the execution flow.[26]

Antigravity is highly versatile, operating as a cross-platform solution compatible with MacOS, Windows, and Linux.[26] While it defaults to utilizing Google's proprietary Gemini 3 Pro and Gemini 3 Deep Think models with generous free rate limits (resetting every 5 hours), the platform embraces model optionality, offering full support for Anthropic's Claude Sonnet 4.5 and open-source variants like GPT-OSS.[26]

Progressive Disclosure and the Architecture of Agent Skills

A critical engineering challenge in developing autonomous AI agents is mitigating "Context Saturation" and "Tool Bloat".[33] Loading a Large Language Model's (LLM) active context window with every possible tool, entire framework documentations, and exhaustive codebase standards at the beginning of a session leads to severe latency and financial waste.[33] More importantly, it causes "context rot," a phenomenon where the model becomes confused by the sheer volume of irrelevant data, leading to hallucinations or the misapplication of tools.[33]

To circumvent this limitation, Antigravity utilizes an architectural pattern called "Progressive Disclosure," implemented via "Agent Skills".[33] An Agent Skill is a lightweight, open-format, highly modular unit of procedural knowledge.[33] Instead of forcing the model to memorize every specific workflow—such as database migrations, security audits, or, in this case, complex Firebase Domain routing—the capabilities are packaged into discoverable units.[33] The core engine is initially provided only with a lightweight metadata "menu" of available skills. When the agent identifies that the user's natural language intent specifically matches a skill's description, it dynamically retrieves and loads the heavy, step-by-step procedural instructions contained within that skill's definition.[33]

Constructing a SKILL.md File

Skills in Antigravity are constructed using standardized formats, ensuring they are portable and easily parsed by the agent. The brain of every skill is a single file named SKILL.md, which must reside within the .agent/skills/ directory of the active workspace for local project skills, or the global directory for cross-project functionality.[35]

The SKILL.md file is strictly bifurcated into two sections [36]:

1. **YAML Frontmatter:** Located at the absolute top of the file, delineated by ---. This metadata section defines the name (a unique identifier using hyphens) and, crucially, the

description.[36] The description acts as the semantic trigger that prompts the AI to activate the skill. It must be written clearly in the third person to optimize the model's triggering logic.[37]
2. **Markdown Body:** The procedural instructions. This section dictates the "When to use" and "How to use" directives. It establishes the guardrails, code conventions, and exact step-by-step logic the agent must autonomously execute or guide the developer through.[36]

The markdown body can also define strict workflows. If a workflow step includes the // turbo annotation alongside the run_command tool, the agent is authorized to automatically execute that specific terminal step without waiting for explicit user approval, significantly speeding up repetitive scaffolding tasks.[39]

The Firebase Authentication Custom Domain Expert Skill Implementation

To endow the Antigravity agent with the expert-level capabilities required to systematically resolve the firebaseapp.com routing issue for thefunfanreporter.com (and abstract the entire 5-phase configuration process outlined previously), developers must construct a custom skill payload.

The following documentation represents the exact, technically exhaustive payload required to train the Antigravity agent.

Deployment Protocol

To activate this capability within the Antigravity environment, the developer must execute the following sequence:

1. Open the Google Antigravity IDE and navigate to the root directory of the application workspace.
2. Ensure the hidden .agent configuration directory exists. This can be created via the integrated terminal by executing the command: mkdir -p.agent/skills/firebase-auth-domain-fix.[37]
3. Create a new markdown file at the exact path: .agent/skills/firebase-auth-domain-fix/SKILL.md.[37]
4. Copy and paste the entirety of the following code block into the SKILL.md file exactly as formatted. No alterations to the YAML structure are permitted.

The SKILL.md Payload

---

name: firebase-auth-domain-fix description: Expert system for configuring and deploying a custom authDomain for Firebase Authentication. Use this skill whenever a user reports that the default firebaseapp.com URL is showing during Google Sign-in or OAuth redirects, or when explicitly requested to configure a custom domain for Firebase Auth.

Firebase Authentication Custom Domain Configuration Expert

Mission Statement

This skill instructs the agent to systematically resolve the issue where Firebase Authentication displays the default .firebaseapp.com domain instead of the user's custom domain (e.g., thefunfanreporter.com) during federated OAuth flows. It enforces a strict, 5-phase architectural workflow to update the client SDK, the Firebase Auth whitelist, Firebase Hosting DNS, GCP OAuth credentials, and email templates.

When to use this skill

- The user asks how to remove firebaseapp.com from their Google Sign-In popup or iframe.
- The user is experiencing auth/unauthorized-continue-uri, redirect_uri_mismatch, or NET::ERR_CERT_COMMON_NAME_INVALID errors after attempting to change their authentication domain.
- The user asks to configure thefunfanreporter.com (or any custom domain) as the authDomain in their Firebase SDK configuration.

Core Knowledge & Architectural Constraints

- **Pricing Reality:** Custom domains for Firebase Authentication and Firebase Hosting are completely FREE under the Spark (no-cost) tier. Do not tell the user they need to upgrade to fix this issue.
- **Routing Conflict Prevention:** If the main frontend application is hosted OUTSIDE of Firebase Hosting (e.g., on Vercel, Netlify, or AWS), you MUST instruct the user to create a

dedicated subdomain (like auth.thefunfanreporter.com) to prevent 404 routing errors. If the app is hosted natively ON Firebase, the root apex domain (thefunfanreporter.com) is acceptable.
- **OAuth Callback Strictness:** The exact trailing path /__/auth/handler is unconditionally mandatory for Google Cloud Platform redirect URIs.

How to execute the configuration (Step-by-Step)

When triggered, you must perform or guide the user through the following phases precisely in order. Utilize the editor to make code changes autonomously where applicable, and output clear, formatted markdown checklists (Artifacts) for console-based steps that require manual user intervention.

Phase 1: Client-Side SDK Autonomous Update

1. Scan the workspace to locate the Firebase initialization code (search for firebase.js, firebase.ts, or the firebaseConfig object).
2. Identify the authDomain property within the configuration object.
3. Autonomously refactor the code to replace the default .firebaseapp.com with the custom domain requested by the user (e.g., thefunfanreporter.com or auth.thefunfanreporter.com).
4. Generate a Code Diff Artifact presenting the change to the user for approval.

Phase 2: Firebase Hosting DNS Mapping (Guide the User)

Generate an Artifact instructing the user to perform the following in their browser:

1. Navigate to Firebase Console -> Build -> Hosting.
2. Click **Add custom domain** and input their specific domain (e.g., auth.thefunfanreporter.com). Ensure they do not check the redirect box.
3. Copy the provided TXT record and add it to their domain registrar's DNS settings to verify ownership.
4. Add the provided CNAME (or A records if using apex) to point the domain to Firebase. *Agent Advisory:* Explicitly warn the user that SSL provisioning (Pending status) may take up to 24 hours. They may see a NET::ERR_CERT_COMMON_NAME_INVALID error during this propagation window.

Phase 3: Firebase Auth Authorized Domains (Guide the User)

Instruct the user via the Artifact:

1. Navigate to Firebase Console -> Authentication -> Settings (or Sign-in method tab).
2. Scroll down to the **Authorized domains** list.
3. Click **Add Domain**.
4. Type the exact domain configured in Phase 1 (e.g., auth.thefunfanreporter.com) and click

save.

Phase 4: Google Cloud Console OAuth 2.0 Credentials (Guide the User)

This is the most critical step to prevent redirect_uri_mismatch errors. Instruct the user carefully:

1. Navigate to the Google Cloud Console (console.cloud.google.com).
2. Navigate to **APIs & Services** -> **Credentials**.
3. Locate and edit the "Web client (auto created by Google Service)" under the OAuth 2.0 Client IDs section.
4. Under **Authorized JavaScript origins**, click Add URI and input: https://auth.thefunfanreporter.com
5. Under **Authorized redirect URIs**, click Add URI and input exactly: https://auth.thefunfanreporter.com/__/auth/handler
6. Click Save.

Phase 5: Brand Consistency Verification

Ask the user if they wish to update their Authentication Email Templates (Password Resets, Verifications) to also use the custom domain. If yes, instruct them to navigate to Authentication -> Templates -> Customize Domain, and guide them through adding the generated SPF and DKIM TXT records to their DNS provider.

Final Verification Protocol

Ask the user to trigger a Google Sign-In flow in their local or production environment. Verify that the popup URL now displays the custom domain. If a 404 error occurs immediately upon redirect, diagnose if they are hosting on Vercel/Netlify using the root domain, and forcefully advise switching to a CNAME auth subdomain.

By leveraging this meticulously crafted skill within the Antigravity platform, the developer transforms the AI from a generalized coding assistant into an autonomous, domain-specific cloud architect capable of flawlessly executing complex infrastructure synchronization.

Works cited

1. Using OAuth 2.0 for Web Server Applications | Authorization - Google for Developers, accessed February 17, 2026, https://developers.google.com/identity/protocols/oauth2/web-server
2. Showing a custom domain during sign in | Identity Platform | Google ..., accessed February 17, 2026, https://docs.cloud.google.com/identity-platform/docs/show-custom-domain
3. Best practices for using signInWithRedirect on browsers that block ..., accessed

February 17, 2026, https://firebase.google.com/docs/auth/web/redirect-best-practices
4. Firebase Pricing, accessed February 17, 2026, https://firebase.google.com/pricing
5. Connect a custom domain | Firebase Hosting - Google, accessed February 17, 2026, https://firebase.google.com/docs/hosting/custom-domain
6. Exploring Firebase's Free Tier: How Much Can You Get for Free? - DEV Community, accessed February 17, 2026, https://dev.to/iredox10/exploring-firebases-free-tier-how-much-can-you-get-for-free-3971
7. Firebase Authentication Limits - Google, accessed February 17, 2026, https://firebase.google.com/docs/auth/limits
8. Differences between Identity Platform and Firebase Authentication, accessed February 17, 2026, https://docs.cloud.google.com/identity-platform/docs/product-comparison
9. MFA, Blocking functions, and more come to Firebase Authentication, accessed February 17, 2026, https://firebase.blog/posts/2022/07/new-firebase-auth-features/
10. What is the difference between Identity Platform and Firebase Authentication with Identity Platform - Stack Overflow, accessed February 17, 2026, https://stackoverflow.com/questions/73661376/what-is-the-difference-between-identity-platform-and-firebase-authentication-wit
11. Is Firebase Auth Free Without Using Identity Platform whem hitting over 50k MAU? - Reddit, accessed February 17, 2026, https://www.reddit.com/r/Firebase/comments/1ga4878/is_firebase_auth_free_without_using_identity/
12. 2025 Firebase Authentication's latest pricing explained and the best alternatives, accessed February 17, 2026, https://blog.logto.io/firebase-authentication-pricing
13. Quick Guide: Setting Up Google OAuth2 Login with a Custom Domain in Firebase Auth, accessed February 17, 2026, https://medium.com/@citi_zen/quick-guide-setting-up-google-oauth2-login-with-a-custom-domain-in-firebase-auth-fc86c328682d
14. Problems with custom authDomain and NextJS : r/Firebase - Reddit, accessed February 17, 2026, https://www.reddit.com/r/Firebase/comments/1kl42eu/problems_with_custom_authdomain_and_nextjs/
15. Admin Authentication API Errors - Firebase - Google, accessed February 17, 2026, https://firebase.google.com/docs/auth/admin/errors
16. Help with custom auth domain on signing in with Google consent screen : r/Firebase, accessed February 17, 2026, https://www.reddit.com/r/Firebase/comments/1l5ncn7/help_with_custom_auth_domain_on_signing_in_with/
17. Firebase App Hosting authentication with GoogleProvider does not work - Reddit, accessed February 17, 2026, https://www.reddit.com/r/Firebase/comments/1iu354d/firebase_app_hosting_authentication_with/

18. Firebase Auth - customized redirect domain prompts NET::ERR_CERT_COMMON_NAME_INVALID warning - Stack Overflow, accessed February 17, 2026, https://stackoverflow.com/questions/52218777/firebase-auth-customized-redirect-domain-prompts-neterr-cert-common-name-inv
19. How to replace the myApp-123.firebaseapp.com with my custom domain myApp.com, accessed February 17, 2026, https://stackoverflow.com/questions/44815580/how-to-replace-the-myapp-123-firebaseapp-com-with-my-custom-domain-myapp-com
20. Google Cloud Console: Add authorized redirect URLs for authentication - Stack Overflow, accessed February 17, 2026, https://stackoverflow.com/questions/74764510/google-cloud-console-add-authorized-redirect-urls-for-authentication
21. How to get firebase auth (google login) to display custom domain instead of default firebase project domain (eg. project-123.firebaseapp.com) - Reddit, accessed February 17, 2026, https://www.reddit.com/r/Firebase/comments/1j1y0g7/how_to_get_firebase_auth_google_login_to_display/
22. Use a custom domain for Authentication emails - Firebase, accessed February 17, 2026, https://firebase.google.com/docs/auth/email-custom-domain
23. Configure Hosting behavior - Firebase - Google, accessed February 17, 2026, https://firebase.google.com/docs/hosting/full-config
24. Firebase Custom Domain 404 Error on subdomain - Stack Overflow, accessed February 17, 2026, https://stackoverflow.com/questions/44532088/firebase-custom-domain-404-error-on-subdomain
25. What is Google Antigravity?, accessed February 17, 2026, https://medium.com/@tahirbalarabe2/what-is-google-antigravity-49872c58305f
26. Build with Google Antigravity, our new agentic development platform, accessed February 17, 2026, https://developers.googleblog.com/build-with-google-antigravity-our-new-agentic-development-platform/
27. Google Antigravity - Wikipedia, accessed February 17, 2026, https://en.wikipedia.org/wiki/Google_Antigravity
28. Google Antigravity Tool (IDE): What It Is and How Developers Benefit: ExpertAppDevs.Com, accessed February 17, 2026, https://medium.com/@expertappdevs/google-antigravity-tool-ide-what-it-is-and-how-developers-benefit-50119f8d886c
29. Getting Started with Google Antigravity, accessed February 17, 2026, https://codelabs.developers.google.com/getting-started-google-antigravity
30. Tutorial : Getting Started with Google Antigravity | by Romin Irani - Medium, accessed February 17, 2026, https://medium.com/google-cloud/tutorial-getting-started-with-google-antigravity-b5cc74c103c2
31. accessed February 17, 2026,

https://developers.googleblog.com/build-with-google-antigravity-our-new-agentic-development-platform/#:~:text=Antigravity%20isn't%20just%20an,editor%2C%20terminal%2C%20and%20browser.
32. Google Antigravity AI Coding: Building My Portfolio Site from Scratch | Tech With Sam, accessed February 17, 2026, https://techwithsam.dev/articles/google-antigravity-ai-coding
33. Authoring Google Antigravity Skills, accessed February 17, 2026, https://codelabs.developers.google.com/getting-started-with-antigravity-skills
34. Create Your First SKILL.md File (Make AI Agents Do Exactly What You Want) | #claude #antigravity, accessed February 17, 2026, https://www.youtube.com/watch?v=Fh-aBKrG5CI
35. Antigravity Global Skills Configuration: Where is the correct path and how to activate them?, accessed February 17, 2026, https://www.reddit.com/r/google_antigravity/comments/1qn48a0/antigravity_global_skills_configuration_where_is/
36. What are Google Antigravity Skills? Build 24/7 AI Agents | VERTU, accessed February 17, 2026, https://vertu.com/lifestyle/mastering-google-antigravity-skills-the-ultimate-guide-to-extending-agentic-ai-in-2026/
37. Agent Skills - Google Antigravity Documentation, accessed February 17, 2026, https://antigravity.google/docs/skills
38. Tutorial : Getting Started with Google Antigravity Skills - Medium, accessed February 17, 2026, https://medium.com/google-cloud/tutorial-getting-started-with-antigravity-skills-864041811e0d
39. Google Antigravity Prompts - GitHub Gist, accessed February 17, 2026, https://gist.github.com/CypherpunkSamurai/f16e384ed1629cc0dd11fea33e444c17
40. How do you create skills with AG? : r/google_antigravity - Reddit, accessed February 17, 2026, https://www.reddit.com/r/google_antigravity/comments/1qg177z/how_do_you_create_skills_with_ag/