AEGIS OBSIDIAN

Version 13.7 // Immutable Kernel

The Probabilistic Project Management Protocol

System Reference, Mathematical Proofs, & User Manual

Prepared By:
Swayam, Chief Executive Officer
Biswal & Co.

Date:
February 15, 2026

Contents

1 System Definition **2**
1.1 The Aegis Philosophy 2
1.2 Core Capabilities 2

2 The Mathematics of Execution **3**
2.1 The Coordination Tax (Alpha Coefficient) 3
2.2 Scope Acceptance Sigmoid (SAS) 3
2.3 The Recalibration Engine 3

3 User Manual: Operational Protocols **4**
3.1 Acquiring the API Access Key (Mandatory) 4
3.2 Authentication Flow 4
3.3 Simulating Scope Injection 4
3.4 Daily Tactical Resolution 4

4 Technical Safeguards **5**

System Definition

1.1 The Aegis Philosophy

Aegis Obsidian is not a productivity application; it is a **Tactical Navigational Computer**. Traditional software treats tasks as binary (done/not done). **Biswal & Co.** recognizes that execution is a probabilistic stream. Aegis v13.7 uses the firm's historical data to model the friction of reality, ensuring that strategic planning is grounded in mathematical truth rather than optimistic bias.

1.2 Core Capabilities

- **Self-Healing Schema:** The system utilizes a "Self-Healing Foundation" that checks for sheet integrity every time the `INITIALIZE_SYSTEM()` function is called, preventing data corruption.
- **Probabilistic Forecasting:** Using Sample Variance (s^2) and Standard Deviation (σ), Aegis predicts delivery windows with a 95% confidence interval.
- **AI Architectural Synthesis:** Integrates directly with Google Gemini to "dream" complete project roadmaps, paginated into 30-day tactical bursts to maintain high-fidelity JSON output.

The Mathematics of Execution

2.1 The Coordination Tax (Alpha Coefficient)

As projects increase in volume, the "mental overhead" of management grows. Aegis models this via a dynamic Alpha (α):

$$\alpha = 0.02 + (\text{TotalTasks} \times 0.0002) \tag{2.1}$$

This coefficient acts as a drag on the **Scope Acceptance Score (SAS)**, simulating the real-world coordination cost of large portfolios.

2.2 Scope Acceptance Sigmoid (SAS)

When simulating new directives, Aegis uses a Logistic Sigmoid Function to map the "Absorption Capacity" of a project.

The SAS determines how much "strain" a new objective adds. If the required velocity exceeds historical peaks, the SAS drops toward zero, signaling a "Burnout Warning."

2.3 The Recalibration Engine

Unlike static tools, Aegis features a **Linear Cascade Engine**. When a delay is committed, the backend identifies the row index of the target task and performs a vector-shift on all subsequent date objects within that Project ID. This ensures that the schedule remains a "Live Document" while the final deadline remains an immutable anchor.

User Manual: Operational Protocols

3.1 Acquiring the API Access Key (Mandatory)

The AI Architect and Semantic Audit features require an external cognitive engine.

1. Access **Google AI Studio** at: https://aistudio.google.com/app/apikey
2. Log in using your authorized **Biswal & Co.** credentials.
3. Click **"Create API Key"**.
4. Copy the key. This key functions as the "Access Key" in the Aegis login portal.

3.2 Authentication Flow

The login portal is a secure gate.

- **Executive Lead:** Identify yourself (e.g., **Swayam**).
- **Access Key:** Paste your Gemini key. This key is stored in your browser's `localStorage`, meaning you only need to authenticate once per device.

3.3 Simulating Scope Injection

To use the Simulation Panel:

1. Open any active Project Card.
2. Locate the **Scope Injection Slider**.
3. Slide to the right to simulate adding up to 20 tasks.
4. Observe the **SAS Score**. If it turns red, the project is mathematically over-leveraged.

3.4 Daily Tactical Resolution

In the "Project Detail" view, the system utilizes a **Regex-Based Normalization** (`replace(/\D/g, '')`) to display clean day numbers regardless of sheet formatting. Use this view for "Ground-Level" execution.

Technical Safeguards

Aegis v13.7 implements three critical safety protocols:

1. **Type Enforcement:** Uses `getDisplayValues()` to treat all IDs as Strings, preventing the common "ID Mismatch" error that plagues standard Google Apps Script tools.
2. **Sequential Pagination:** To avoid API timeouts, the Architect creates tasks in 20-day chunks, ensuring the JSON payload never crashes the browser.
3. **Semantic Audit:** When a task is completed with evidence, the system runs a cosine-similarity check between the "Objective" and the "Evidence" to score the quality of execution.

Finalized for Deployment
Biswal & Co. // **Swayam**, CEO