# WhatsApp Automation

**Technical Implementation Summary and IT Server Requirements**

This document explains the current WhatsApp Automation application, how it operates, what infrastructure it needs in production, and what dependencies IT must provide for reliable deployment and status tracking.

**Prepared for PSI | Date: 2026-05-12**

## 1) Solution Overview

The application receives BrightCall webhook events, detects unanswered AI call scenarios, and sends a follow-up WhatsApp template message through the Commbot API. It includes a desktop UI for operations, environment toggles, AUH vs Assets routing, project-level template mapping, persistent message reporting, CSV export, and automatic retry handling for temporary Commbot failures.

> **Current public webhook endpoint:**
> `https://messages.ngrok.dev/path/to/api/callEnded`

## 2) Technologies Used

### Backend and API

- Node.js runtime (minimum supported: Node 18+)
- Express.js for webhook routes, runtime settings endpoints, and debug/reporting endpoints
- Axios for outbound API calls to Commbot
- dotenv for environment-based configuration
- HMAC validation support for webhook signatures
- Local file persistence for durable event history and retry queue

### Desktop Application

- Electron for packaging backend plus web UI into a Windows desktop app
- electron-builder for portable EXE and NSIS installer generation
- Company-branded executable and app icon configuration

### Network Exposure

- ngrok paid account with reserved domain: `messages.ngrok.dev`
- Stable HTTPS URL routing to local service on port 3000

## 3) Functional Flow

1. BrightCall sends webhook events to the ngrok URL.
2. ngrok forwards requests to the application server at `http://localhost:3000`.
3. The application validates and parses the event, resolves the target environment, and selects the correct Commbot template.
4. For eligible unanswered AI call cases, the application sends the Commbot template request. Optional repeat-follow-up and after-hours template overrides may apply.
5. If Commbot is temporarily unavailable, the outbound message is queued and retried automatically.

6. Message logs, counts, recent events, and discovered projects are available in the UI and debug endpoints.
7. Persistent history and queued retries are reloaded after restart, so reporting survives application restarts on the same server disk.

---

## 4) APIs and Main Routes

- `POST /path/to/api/callEnded` (primary BrightCall path)
- `POST /path/to/api/webphoneSummary` and related BrightCall event paths (supported route variants)
- `POST /webhooks/brightcall` (fallback generic path)
- `GET /api/settings` and `POST /api/settings` (runtime UI settings)
- `GET /debug/message-logs` (status logs and counters by environment)
- `GET /debug/message-logs.csv` (CSV export for logs)
- `GET /debug/recent-events` (recent event feed)
- `GET /debug/discovered-projects` (observed BrightCall projects)
- `GET /health` (basic service health)

---

## 5) Configuration Inputs (.env)

- Commbot API URL, API key, smart key, channel, account names, and agent metadata
- Base template IDs plus AUH/Assets-specific, repeat-follow-up, and after-hours template IDs
- Project-level template override map for routing specific projects to specific templates
- BrightCall filtering controls and duplicate-suppression timing
- Business timezone and business hours for after-hours routing
- Persistent storage paths for event history and retry queue
- Retry timing controls for temporary Commbot failures
- ngrok auth token and reserved domain
- Port and optional webhook security secret

Important: Credentials should be treated as secrets and rotated according to internal policy.

## 6) IT Requirements for Always-On Server

### Recommended Server Sizing (Production)

| Item | Recommended | Notes |
|---|---|---|
| OS | Windows Server 2019/2022 or Ubuntu 22.04 LTS | Current packaged app is Windows-first, but the backend remains standard Node.js. |
| CPU | 2 vCPU minimum (4 vCPU preferred) | Webhook load is light, but extra capacity improves resilience and monitoring headroom. |
| RAM | 4 GB minimum (8 GB preferred) | Allows Node app, ngrok, monitoring, and background retry processing comfortably. |
| Disk | 20 GB free minimum, persistent disk required | Includes application files, logs, durable event history, retry queue, backups, and installers. |
| Node.js | Node 18 LTS or Node 20 LTS | Match enterprise standard and security patching cycle. |

### Network and Firewall Requirements

- Allow outbound HTTPS (TCP 443) to:
  - ngrok cloud services
  - BrightCall services
  - Commbot API endpoint domain
- No inbound public port opening is required when using ngrok tunnel mode.
- DNS reachability for `messages.ngrok.dev` must remain valid.

### Service Persistence (Critical)

To keep the integration running 24/7, both the Node service and ngrok tunnel must auto-start after reboot and auto-recover on crash. The storage used by the application must also persist across restart and redeployment.

- Windows: run as Windows Services via NSSM, or scheduled tasks with restart policy.
- Linux: run with systemd services and restart policy ( `Restart=always` ).
- Health check and process monitoring should restart the service if unresponsive.
- Do not use ephemeral storage unless the app `data/` folder is mapped to a persistent volume or replaced with database-backed storage.

### Commbot Status Tracking Requirement

If the business wants the app to show delivered, opened, and read status inside the message log, Commbot must expose that data to this application through webhook callbacks or an API endpoint the app can call.

- The current app can store and display these statuses if they are sent back programmatically.
- The Commbot dashboard by itself is not enough. The data must be available through integration, not only visible in the browser dashboard.
- If Commbot provides status webhooks or an API, IT must allow that traffic and provide the endpoint details or credentials needed.

### Security and Compliance

- Store secrets in a secure vault or encrypted configuration pipeline.

- Restrict server access to IT admins only (least privilege).
- Enable anti-malware exclusions for the approved application path if required by endpoint security.
- Enable log retention and audit trails for operational troubleshooting.

## 7) What to Ask IT Team (Checklist)

| Request | Why it is needed |
|---|---|
| Provide an always-on VM/server with Node 18/20 installed | Runs the webhook receiver continuously |
| Install ngrok and configure paid account token | Maintains the fixed public URL for BrightCall webhooks |
| Allow outbound access to ngrok, BrightCall, and Commbot APIs | Required for webhook intake and message dispatch |
| Provide persistent disk storage for the application data folder | Keeps message history, retry queue, and reporting after restart |
| Confirm whether Commbot exposes delivered, opened, and read events through webhook or API | Needed if the app should mirror the Commbot dashboard status lifecycle |
| Configure auto-start and auto-restart service policy | Ensures no manual intervention after reboot or crash |
| Set up monitoring and alerting (service down, failed retries, failed webhooks) | Early failure detection |
| Secure secret management for .env values | Protects API keys and tokens |
| Define backup and rollback process for app updates | Safe release management |

## 8) Operational Runbook (High Level)

1. Deploy the latest build package to the server.
2. Place approved `.env` values, including template routing and persistent storage settings, and verify `NGROK_DOMAIN=messages.ngrok.dev`.
3. Start services (Node plus ngrok) through a managed service supervisor.
4. Validate endpoints:
   - `http://localhost:3000/health`
   - `http://localhost:3000/api/settings`
   - `http://localhost:3000/debug/message-logs`
   - `http://localhost:3000/debug/message-logs.csv`
5. Trigger a test webhook and confirm message log entries, retry behavior, and status updates in the UI.

## 9) Final Recommendation

For company-scale reliability, host this on a dedicated always-on server instead of a user laptop. Keep the ngrok reserved domain, use persistent server storage for the application data folder, apply service monitoring, and formalize secret management to ensure uninterrupted webhook processing and durable reporting.