ADB Framework Telco Automation – Release Notes v1.0.0 (Verified Capabilities)

Item	Detail
Version	1.0.0
Release date	2025-01-15
Scope	Initial production rollout focused on desktop app + device management
Status	Production ready (features validated against code base)

Executive Overview

This release ships the first stable build of the Telco ADB Automation platform. It confirms, with implementation evidence, which capabilities are **actually delivered** in v1.0.0. The focus is on a production-ready Electron desktop application, reliable Android device management over ADB, and a documented architecture that sets the foundation for upcoming workflow automation.

Key achievements:

- Professional Windows distribution (NSIS/MSI/portable) backed by an Electron + React 18 renderer.
- FastAPI backend exposing device CRUD, health monitoring, and structured logging.
- Solid ADB-centric device lifecycle (discovery, info, status, battery/network insights, developer-mode provisioning).
- Formal documentation set (architecture, backend, frontend) plus PDF/DOCX exports for distribution.

Delivered Functionality

1. Desktop Application & Packaging

- Electron shell with preload isolation, React renderer, and localized (EN/FR) UI copy.
- Installers: NSIS wizard, MSI for enterprise deployment, and a portable binary.
- Auto-update channel configured (GitHub release source) plus system tray integration and startup shortcuts.

2. **Device Lifecycle & Telemetry**
   - Direct USB discovery through ADB with automatic refresh when devices connect/disconnect.
   - Device profile retrieval: model, manufacturer, Android/API version, battery %, charging, operator, and RAT.
   - Guided developer-mode setup (USB debugging enablement scripts) with status feedback.
   - Live card refresh on the dashboard, alias editing support, and device activity logging hooks.

3. **Backend API & Services**
   - FastAPI application (`simple-server.py`) with `/api/v1/devices`, `/api/v1/devices/{id}`, `/api/v1/devices/{id}/setup`, `/health`, and statistics endpoints.
   - SQLAlchemy persistence layer and structured logger writing JSON lines for auditability.
   - CORS policy and static asset serving to unblock Electron renderer load paths.

4. **Frontend Experience**
   - React 18 + TypeScript app using Material UI for layout, with views for Dashboard, Device Manager, Workflows, Modules, and Reports.
   - Shared hooks (`useDevices`, `useWorkflows`) powering live device grids, search, and module execution controls.
   - Workflow runner respects sequential execution (next module begins only after previous resolves) and exposes stop/cancel controls.

5. **Telco Module Framework**
   - 29 YAML module definitions, including ping, call, airplane mode, and telco-specific routines.
   - Python module runtime with input/output schema validation and synchronous execution over ADB.
   - Backend endpoints for triggering single-module "Run" operations; UI reflects in-progress state (e.g., Ping Run button turns green during backend execution).

6. **Documentation & Release Assets**
   - New documentation set in `docs/`: `architecture-overview.md`, `backend-guide.md`, `frontend-guide.md`, updated `toc.yml`, and DocFX scaffolding.
   - Export pipeline (`docs/exports/…`) with PDF/DOCX builds generated via Dockerized Pandoc (`pandoc/latex`) for architecture, backend, and

frontend guides.
- Auxiliary utilities (`generate_pdf.py`, `generate_release_pdf.bat`) to automate future doc packaging.

Fixes & Quality Improvements

- Removed obsolete Device Manager widgets (Pinned Devices, Collections, Usage Insights, Live Activity) to declutter the UI.
- Adjusted device-card layout so alias text is editable inline, optional label removed, and edit icon aligned right for better readability.
- Ensured Ping module runs entirely on the backend without modal popups; Run button indicates execution state and resets on completion.
- Workflow executor now blocks module *n+1* until module *n* fully finishes (covering multi-action modules such as Call with multiple dial attempts).
- Device connection polling intervals tightened so recently plugged/unplugged phones appear/disappear faster on the dashboard.

Documentation & Localization

- Architecture overview now captures end-to-end topology, data flow, and roadmap hooks (scheduler, live logs, multi-session locker).
- Backend and frontend guides describe stacks, state management, API surfaces, deployment steps, and testing strategy.
- French locale (`src/electron/locales/fr.json`) expanded to cover new UI strings introduced during Device Manager redesign.
- All docs referenced in `docs/toc.yml` and ready for DocFX/GitHub Pages publication.

Known Limitations / Deferred Items

Area	Status
Workflow automation (visual builder, background scheduler, cron service)	Planned – UI scaffolding exists but execution backend not yet integrated.
Advanced analytics/reports (PDF/Excel runs, historical dashboards)	Pending – basic device stats only.
Security hardening (auth, RBAC, secrets)	Not implemented in v1.0.0.

Area	Status
Live log streaming & alerting	Design documented; backend/WebSocket channels pending.
API/CLI exposure for CI pipelines	Not part of this drop.

Upgrade / Adoption Notes

1. **Prerequisites** – Windows 10/11, Node.js 18+, Python 3.11, Android SDK platform-tools (ADB) on PATH.
2. **Backend startup** – `python simple-server.py` (Electron dev harness starts it automatically).
3. **Frontend build** – `npm install && npm run build` inside `mon-projet/src/frontend` before packaging Electron.
4. **Docs publishing** – Run DocFX or regenerate PDFs via `docker run --rm -v "$PWD:/data" pandoc/latex <doc>.md -o exports/<doc>.pdf`.

Next Steps

- Wire up Scheduler/Metrics/Live Logs/Multi-session features outlined in the architecture roadmap.
- Add authentication + device reservation APIs before multi-operator deployments.
- Automate doc publishing via CI (DocFX build + GitHub Pages) and attach PDF bundles to releases.

Prepared for release 1.0.0 to give stakeholders an exact picture of shipped functionality.