Technical Documentation — ADB Device Automation & C2-Based System Surveillance

This document describes a **comprehensive, all-in-one Android automation system** built using the Android Debug Bridge (ADB).

The suite provides:

- Remote screenshot capture
- Screen recording
- Media extraction (images + videos)
- UI hierarchy dumping
- Contacts retrieval
- Full remote camera control (photo, video, camera switching)
- OEM-adaptive behavior (Transsion, Xiaomi, Samsung, Oppo, Vivo, Google…)
- Automated UI parsing and fallback strategies

1. Device Connection, Detection & Capability Profiling

When launched, the system begins by analyzing the connected device:

✓ 1.1 Device Identification

It reads:

- Brand
- Manufacturer
- System build information
- Android version
- Whether the device belongs to special OEM families

This detection step is essential because certain vendors — such as **Transsion**, **Xiaomi (MIUI)**, and **Oppo/ColorOS** — use custom camera apps and UI traits that require tailored logic.

2. Camera Control System (Photo, Video, Switching)

This part acts as a **remote-controlled camera bot**, capable of:

- Launching the camera
- Switching between front/back lenses
- Taking photos
- Recording videos
- Saving output
- Extracting files automatically

2.1 Camera Launch Logic (OEM Adaptive)

Different manufacturers require different launch methods:

A. Standard Android devices

Can be launched using their standard system intents.

B. Transsion OS (Tecno / Infinix / Itel)

Camera apps cannot be opened with normal intents.

The tool uses alternative methods that reliably trigger the camera UI.

C. Xiaomi MIUI

Has delayed UI load and repositioned controls.

The tool waits longer before interaction.

Each OEM has:

- Custom UI layouts
- Different button placements
- Different “confirm/accept” workflows

The system adapts automatically.

2.2 UI Auto-Detection With UIAutomator

To interact with the camera app without touching the device, the system:

- Captures a UI dump file from the device
- Parses it as XML
- Searches for:
    - Shutter button
    - Confirm/save button
    - Lens toggle (front/back switch)
- Inspects element attributes:
    - Position on screen
    - Resource ID
    - Content description
    - View class
    - Display text

This enables:

- OEM-agnostic detection
- Adaptation to custom camera layouts
- Correct tapping regardless of screen resolution

If the UI changes or buttons are moved, the script still finds them.

2.3 Hardcoded Overrides (Optional)

For speed and precision, users may define:

- A fixed shutter button location
- A fixed switch-camera button location
- A fixed save/check button location

When these are provided:

- UI parsing is skipped

- Tapping is instantaneous
- More reliable for repetitive captures

2.4 Photo Capture Workflow

When taking a photo, the system performs:

- **Pre-scan** of all photos currently on the device
- **Safe camera close** (to ensure clean state)
- **Camera launch** (method depends on OEM)
- **Lens selection** (front or back)
- **Shutter button press**
- **Confirm / Save**
- **File extraction (pull) to computer**
- **Duplicate avoidance**
- **Unique file naming** using timestamps + random suffixes

This ensures:

- No overwriting
- No missing files
- Consistent output on all devices

2.5 Video Recording Workflow

Video capture works similarly to photo capture, but with:

- Video mode switching
- Recording start/stop logic
- Longer UI wait times
- MP4 file creation and extraction

The system ensures:

- Recording starts only after the UI is ready
- Recording stops cleanly
- Video file is finalized before extraction

2.6 Screen Resolution Awareness

The system retrieves the device's screen size and uses it to:

- Calculate fallback button coordinates
- Detect UI element regions
- Place taps accurately on any display

This guarantees multi-device compatibility.

2.7 Error Handling

The system gracefully manages:

- Slow UI loads
- Missing elements
- OEM layout issues
- Failed taps
- Delayed camera startup
- File transfer interruptions

It logs:

- All button detections
- UI parsing results
- Fall-back scenarios
- Device-specific behavior

3. Screenshot Capture (Simple, Instant, Lossless)

This function captures the exact pixel buffer the display compositor is showing at that moment.

How It Works Internally

The system requests a frame from Android's display service
The device encodes the frame into a PNG format
The image is streamed directly to the computer
The tool saves the file with a timestamped name

Why PNG?

- Lossless
- Sharp edges
- Perfect for documentation and UI analysis

Why this method?

- Fastest approach
- No temporary files stored on the device
- Ideal for rapid debugging and documentation

4. Screen Recording

Screen recording uses Android's built-in recording infrastructure.

Internal Workflow

The device starts capturing every rendered frame
Hardware encoders convert frames to a video format (usually H.264)
The video is written to a file on the device
After recording ends:
The file is finalized
The tool pulls it to the computer
The device-side file is removed

Why save the video on the device first?

Reliable writing
Prevents corruption
Smooth encoding even under load

5. Media Extraction (Images & Videos)

To list and retrieve media files, the system uses **MediaStore**, Android's internal media index.

How Media Discovery Works

MediaStore maintains a structured database of images and videos
The system queries the database
Media entries are returned with:

- ID
- File path
- MIME type
- Timestamp

Why MediaStore instead of file scanning?

- Devices store media in different folders
- OEMs use custom directory structures
- MediaStore is consistent on all Android devices

Media Pull Workflow

User selects items
System retrieves file paths
Each file is streamed to the computer
Duplicates are prevented

6. UI Dumping (XML-based UI Hierarchy)

A UI dump is a complete structural representation of the current screen.

Internal Steps

Android's accessibility inspector traverses the UI hierarchy
An XML file is generated

The tool retrieves this file

The XML contains:

- All visible elements
- Coordinates for each element
- Class names
- Text values
- Resource identifiers

Use Cases

- UI automation
- Documentation
- Reverse engineering layout
- Debugging
- Finding clickable targets

7. Contact Queries (Non-Sensitive)

The tool retrieves contact list entries using Android's **Contacts Provider**.

Returned Data Includes

- Contact ID
- Display name

Why only these fields?

- Safe
- Non-sensitive
- No phone numbers, emails, or addresses
- Suitable for documentation and verification

8. File Organization & Naming

All output files — screenshots, screen recordings, photos, videos — use unique timestamped names ensuring:

- No overwriting
- Easy sorting
- Clean archival

9. Error Handling & Stability

The suite ensures:

- OEM-specific recovery paths
- UI-based fallback logic
- Retry strategies
- Timeout detection
- Logging for each action

It remains reliable even when:

- UI elements move
- Camera apps change
- OEM layers delay rendering
- Device storage is busy

10. Summary — What This Unified System Achieves

This merged system is a **complete ADB-powered automation and documentation toolkit** capable of:

Documentation Features

✓ Screenshots

✓ Screen recordings

✓ UI dumps

✓ Media extraction

✓ Contact retrieval

Advanced Camera Automation

✓ Launching any OEM camera reliably

✓ Front/back camera switching

✓ Photo capture

✓ Video capture

✓ UI auto-parsing

✓ Hardcoded override support

✓ Unique file naming

✓ Intelligent waiting & timing

✓ Fault-tolerant execution

OEM-Adaptive Behavior

- Transsion OS
- Xiaomi MIUI
- Samsung One UI
- Oppo/ColorOS
- Vivo/FuntouchOS
- Google Pixel

Overall Purpose

This suite allows a user to operate an Android device almost entirely remotely for:

- Software QA
- Automated testing
- Documentation
- Research
- Device analysis
- Forensics-style captures
- Remote photography/video use cases