



ID INFO 9000 Terminal – Linux System

The new Linux-based Access Control Terminal combines **facial recognition, RFID/NFC,** and **network connectivity** in a single, intelligent device.

It is designed for seamless integration in **fitness studios, office buildings, residential complexes, and industrial environments.**

Device System

Operating System: Linux (embedded system)

Supported Language Development: JavaScript (cross-platform)

SDK & API: Free SDK and API available for custom integration

Application Development

The device supports independent application development through a lightweight JavaScript framework:

Download the dedicated VSCode plug-in.

Get the sample project from SDK Kit and run it directly in VSCode.

Develop and debug your application in real time — no recompilation required.

This simple 3-step workflow allows developers to build, test, and deploy their own UI and logic directly on the terminal, making it ideal for custom access control, attendance systems, or membership check-in solutions.

API Integration Guide – Linux Devices (ID INFO 9000)

Powered by DejaOS | Embedded Linux Platform | Custom App Development with JavaScript

System Overview: DejaOS Embedded Platform

The ID INFO 9000 runs on **DejaOS**, a lightweight JavaScript-based operating system for embedded Linux devices. It enables rapid app development, deployment, and integration through modern development tools, with full access to hardware features.

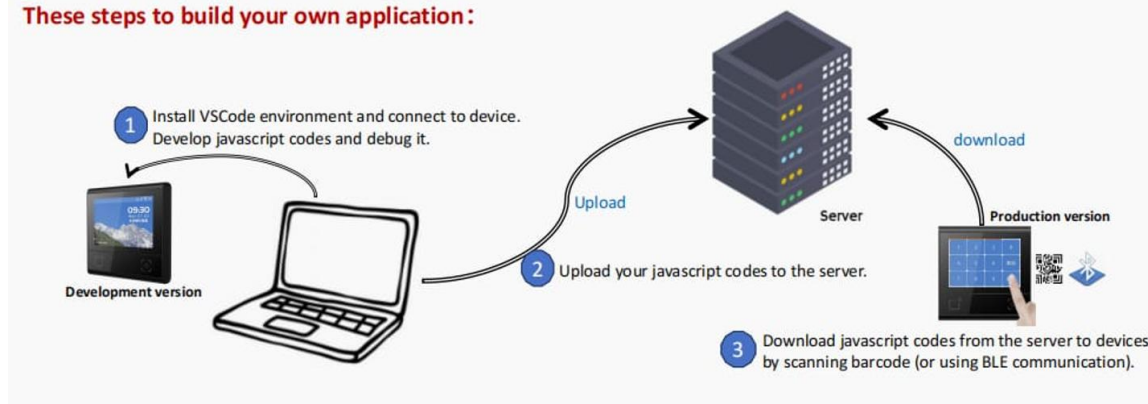
What is DejaOS?

DejaOS is a JavaScript runtime and UI framework designed for edge devices. It allows developers to build and run custom apps on the device using web technologies — without compiling or flashing firmware.

Key Benefits for Developers

-  **JavaScript App Development** (no C++ required)
-  **Real-time debugging in VSCode**
-  **Drag-and-drop UI builder**
-  **Runs directly on device (no emulation)**
-  **Cross-platform tools (Linux, Windows, macOS)**
-  **Access extended JS libraries: HTTP, MQTT, UI components, etc.**

These steps to build your own application :



How It Works:

- Develop JavaScript logic in Visual Studio Code, connected to the development version of the device
- Upload your JavaScript file to the server
- Deploy production devices by scanning a QR code or USB connection.

3-Step Development Workflow

1. Install VSCode Plugin

- Connect the ID INFO 9000 via USB
- Enable development mode in DejaOS

2. Start with a Template

- Download the attached SDK
- Prebuilt components: face auth, RFID, QR scanner, network tools

3. Develop & Deploy

- Build UI with drag-and-drop editor
- Add logic in JavaScript
- Debug and deploy live with no compile step

System Architecture

DejaOS is the core runtime system of the ID 9000 device. It acts as a virtual machine with the following layered structure:

JavaScript Layer:

- JS Applications: Access Control, Gate Control, etc.
- JS Extension Libraries: HTTP, MQTT, OSDP, UI components
- JS APIs via C/C++ bindings: GPIO, NFC, BLE, OTA, Camera, UI

Native Layer:

- JavaScript Engine: QuickJS
- GUI Engine: LVGL
- Drivers: GPIO, Camera, NFC, BLE, RS485, etc.
- Linux Kernel: Based on Ingenic/Easytech chips

You write JS code for custom scenarios, publish it via BLE or barcode from a server. Devices run identical or tailored logic per deployment.

Communication Modes

The ID 9000 supports the following modes:

1. Transparent Transmission Mode

- QR Code and Card Data are passed directly to external systems.
- Face Recognition is processed locally.
- Output: RS485, Wiegand, HTTP, or TCP.

2. Protocol Mode (Recommended for Integration)

- JS application sends data to your server (via HTTP, TCP, or RS485).
- Server validates and replies.
- Reader processes result and executes action.

3. Local Whitelist Mode

- Authorization stored on-device.
- Offline-capable.
- Managed via configuration tool or MQTT.

4. Development Mode

- For advanced device-side integration, including card reading/writing operations.

Flowchart Summary

HTTP/TCP/RS485 Mode

1. User authenticates via QR, card, or password.
2. ID 9000 sends data to server.
3. Server returns authentication result.
4. ID 9000 performs the allowed or denied action.

MQTT Access Control Mode

1. Server registers users and permissions via MQTT.
2. User presents credentials.
3. Device requests permission verification.
4. Based on server response, action is executed.

Device Configuration via MQTT

Using the MQTT Protocol (V1.0.20) you can remotely configure, control, and receive events from the ID 9000.

Topics Format:

- Downlink: 20211214/cmd/{deviceSn}/setConfig
- Uplink (Reply): 20211214/cmd/setConfig_reply

JSON Message Format:

```
{  
  "serialNo": "00000001af",  
  "deviceSn": "XXXXXXXX",  
  "data": {  
    "scannerWorkMode": 4,  
    "faceWorkMode": 4,  
    "protocolChannel": 1,  
    "httpAddr": "http://your.server.com/callback",  
    "httpTimeout": 2000,  
    "qrEn": 1,  
    "nfcEn": 1  
  }  
}
```

Authentication Mode Mapping:

1.scannerWorkMode and faceWorkMode:

- 1: Transparent mode
- 2: Protocol mode
- 3: Development mode
- 4: Local whitelist

2. protocolChannel:

- 1: HTTP
- 2: RS485
- 3: TCP

JavaScript Development with DejaOS

Key Capabilities:

- Develop in Visual Studio Code
- Use JavaScript APIs for device control
- Upload JS scripts to server
- Distribute to devices via BLE or QR code

Sample App Workflow:

1. Write application logic (e.g., door opens on valid QR).
2. Call `http.post()` or `mqtt.publish()` in JS logic.
3. Receive server response.
4. Call `ui.voice("Access granted")` or `relay.trigger()`.

Use Cases

- Fitness Studios: Face recognition + QR for member access
- Corporate Buildings: Dual authentication using ID card and face
- Warehouse Access: Offline whitelist with MQTT sync
- Parking: Wiegand output to barrier system