



JAVA SDK

Integration Guide

—

Version 0.0.1 2025

support@gripid.eu

This documentation covers the initial integration of the GripID biometric device. While the device is equipped with secure communication and operational standards, certain deployment environments—such as those with physical theft risk or requiring advanced access control—may benefit from additional protection layers. For high-security scenarios, GripID offers a dedicated service to help organizations design, manage, and secure a custom GripID Wallet. This wallet-based infrastructure enhances identity assurance and device security against theft or misuse. Our team is available to provide tailored support for wallet-based identity management and advanced cryptographic protection.

1. Contents

1. Contents	2
2. Overview	2
3. Bluetooth Features.....	2
4. System Requirements	3
5. Integration Steps.....	3
5.1 SDK Integration steps	3
6. Usage Examples	4
6.1 Checking Bluetooth Status	4
6.2 Starting Bluetooth Server	4
6.3 Sending Data	5
6.4 Stopping the server	5
7. Best Practices.....	5

2. Overview

The GripID IRC Bluetooth SDK provides a Java-based solution for Bluetooth **RFCOMM** communication. This document focuses exclusively on the Bluetooth functionality of the SDK.

3. Bluetooth Features

- RFCOMM server implementation
- Device discovery and connection management
- Bidirectional data transfer
- QR code generation for device pairing
- Event listeners for connection states

4. System Requirements

- Java 11 or higher
- Bluetooth adapter with JSR-82 support
- Windows/Linux with BlueCove compatibility (tested on windows 10)

5. Integration Steps

5.1 SDK Integration steps

- Add SDK Dependency

```
<dependency>
<groupId>com.gripid</groupId>
<artifactId>irc-sdk</artifactId>
<version>0.0.1</version>
<scope>system</scope>
<systemPath>${project.basedir}/libs/irc-sdk-0.0.1.jar</systemPath>
</dependency>
```

- Initialize the SDK

```
IrcSdk sdk = IrcSdk.getInstance();
```

- Configure Bluetooth Listener

```
sdk.setBluetoothListener(new BluetoothListener() {
    @Override
    public void onBluetoothConnected(String deviceName) {
        System.out.println("Connected to: " + deviceName);
    }
})
```

```
@Override  
  
public void onBluetoothDisconnected(String deviceName) {  
    System.out.println("Disconnected from: " + deviceName);  
}  
  
@Override  
  
public void onBluetoothDataReceived(String message) {  
    System.out.println("Received: " + message);  
}  
  
});
```

6. Usage Examples

6.1 Checking Bluetooth Status

```
if (sdk.checkBluetoothStatus()) {  
    System.out.println("Bluetooth available");  
    System.out.println("Device address: " + sdk.getBluetoothAddress());  
} else {  
    System.err.println("Bluetooth not available");  
}
```

6.2 Starting Bluetooth Server

```
sdk.startBluetoothServer();
```

6.3 Sending Data

```
if (sdk.sendMessageToBluetooth("Hello device!")) {  
    System.out.println("Message sent successfully");  
} else {  
    System.err.println("Failed to send message");  
}
```

6.4 Stopping the server

```
sdk.stopBluetoothServer();
```

7. Best Practices

Always check **checkBluetoothStatus()** before operations