identium

Creating identities for all

CX1500N UHF Handheld Reader

Complete Developer SDK Guide

Keyboard Emulator App • Flutter Plugin • Flutter Sample App

Version	Date	Publisher
1.2.0	March 2026	Identium Tech Solutions

Section 1: Introduction 4
1.1 What is the CX1500N UHF Reader? 4
1.2 Package Contents 4
1.3 Hardware Specifications 4
1.4 Architecture Overview 5
Section 2: Keyboard Emulator App 6
2.1 Overview 6
2.2 Architecture Diagram 6
2.3 Java File Structure 6
2.4 Installation & Build Instructions 7
2.5 First-Time Device Setup 7
2.6 Settings Reference 7
2.7 Barcode Scanner Note 8
Section 3: Flutter Plugin — identium_uhf_plugin 9
3.1 Overview 9
3.2 Plugin Architecture 9
3.3 Installation 9
3.4 Complete API Reference 9
3.5 Data Models 11
3.6 Enum Reference 11
InventoryMode 11
Gen2Session & Gen2Target 11
RfRegion 11
LockObject & LockType 12
RfMode — Encoding Modes 12
3.7 Code Examples 13
1. Connect to Reader 13
2. Read Single Tag 13
3. Multi-Tag Inventory Stream 13
4. Read EPC Only 13
5. Read EPC + TID (with filter) 13
6. Read EPC + TID + User Bank 14
7. Barcode / QR Stream 14
8. Set RF Power 14
9. Configure RF Mode, Session, Target & Region 14
10. Write EPC to Tag 14
11. Lock a Memory Bank 14
12. Kill (Permanently Destroy) a Tag 14
Section 4: Flutter Sample App — identium_uhf_sample 16

4.1 Overview 16
4.2 Screen Wireframes 16
Screen 1: Home Screen 16
Screen 2: Multi-Tag Inventory 16
Screen 3: Single Tag Read 17
Screen 4: Read / Write / Lock / Kill 17
Screen 5: Barcode / QR Scanner 17
Screen 6: RF Settings 18
4.3 Navigation Flow 18
4.4 How to Build & Run 18
4.5 Dependencies 18
Section 5: SDK Quick Reference Card 20
5.1 UHF RF Power Reference 20
5.2 Tag Memory Bank Reference 20
5.3 RF Mode Reference 20
5.4 Error Code Reference 21
Section 6: Troubleshooting 22
Additional Resources 22

Section 1: Introduction

1.1 What is the CX1500N UHF Reader?

The Identium CX1500N is a professional-grade UHF RFID handheld reader running Android 8.0 or higher. It integrates a high-performance UHF RFID engine (EPC Class1 Gen2 / ISO 18000-6C), a 1D/2D barcode scanner, and a full Android OS in a rugged, ergonomic form factor designed for warehouse, retail, and industrial use.

Communication with the UHF engine is handled over UART at /dev/ttysWK0 using Identium's proprietary UHF SDK (UHF_SDK-debug.aar). This SDK package provides three ready-to-use software components that wrap the SDK for different integration scenarios.

1.2 Package Contents

Component	Description	Language	Use Case
TestUHFAPI	Reference Android app demonstrating raw SDK calls	Java / Android	Testing & validation
Keyboard Emulator App	Standalone Android service app that types RFID/barcode data into any focused text field	Java / Android	Immediate deployment without app changes
Flutter Plugin (identium_uhf_plugin)	Dart/Flutter plugin wrapping the UHF SDK via MethodChannel + EventChannel	Dart + Java	Custom Flutter app development
Flutter Sample App (identium_uhf_sample)	Full-featured demo Flutter app showcasing all 12 plugin capabilities	Dart / Flutter	Developer reference & demo

1.3 Hardware Specifications

Specification	Value
UHF Frequency	902–928 MHz (North America) / 865–868 MHz (EU) / 920–925 MHz (China)
Protocol	EPC Class1 Gen2 / ISO 18000-6C
Read Range	Up to 10 meters (at 33 dBm)
RF Power	5–33 dBm (adjustable in SDK)
UART Path	/dev/ttysWK0
Minimum Android	8.0 (API Level 26)
Barcode Scanner	1D / 2D, QR Code, PDF417

1.4 Architecture Overview

The three-layer architecture isolates hardware access from application logic:

Flutter App (Dart)	Your custom Flutter app or the sample app
↕ MethodChannel / EventChannel	
Flutter Plugin (Java)	identium_uhf_plugin — bridges Dart to Android Java
↕ JNI / AAR	
UHF SDK (AAR)	UHF_SDK-debug.aar — UHFEngine + UHFParamsOperator
↕ UART /dev/ttysWK0	
CX1500N Hardware	UHF RFID engine + Barcode module

Section 2: Keyboard Emulator App

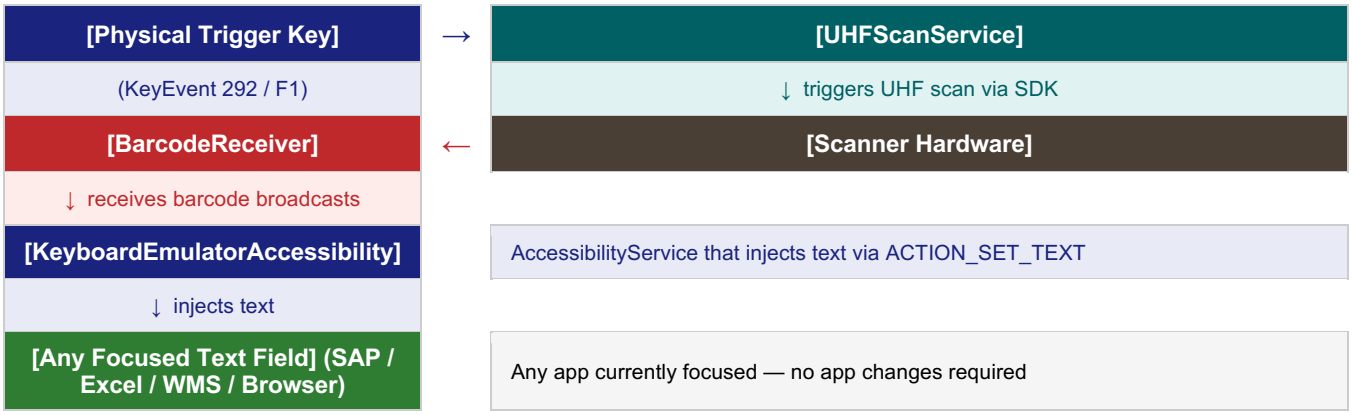
2.1 Overview

The Keyboard Emulator App transforms the CX1500N into a 'virtual keyboard' device. When a user presses the physical scan trigger, the app scans an RFID tag or barcode and injects the result text directly into any focused text field — as if the user had typed it on a keyboard. This requires zero modifications to the target application.

Key benefits:

- Works with any Android app: SAP, Excel, WMS, browser, etc.
- No network required — pure local operation
- Configurable prefix, suffix (Enter/Tab/Space/None), and output format
- Supports both UHF RFID and barcode scanning
- Auto-start on device boot (optional)

2.2 Architecture Diagram



2.3 Java File Structure

File Name	Purpose
MainActivity.java	Main UI with on/off toggle; intercepts the physical trigger key (KeyEvent 292 / F1) and dispatches scan requests
KeyboardEmulatorAccessibility.java	Android AccessibilityService that injects scanned text into the focused view using ACTION_SET_TEXT
UHFScanService.java	Foreground service that keeps the UHF module powered and performs single-tag reads on demand
BarcodeReceiver.java	BroadcastReceiver listening for barcode/QR results from the CX1500N scan SDK

File Name	Purpose
BootReceiver.java	BroadcastReceiver for ACTION_BOOT_COMPLETED — auto-starts the service on device reboot (if enabled)
ScanConfig.java	Persistent configuration helper backed by SharedPreferences (prefix, suffix, RF power, mode, etc.)
SettingsActivity.java	Settings UI with controls for scan mode, prefix/suffix, RF power SeekBar, and auto-start toggle

2.4 Installation & Build Instructions

1. Open the IdentiumKeyboardEmulator/ folder in Android Studio (Flamingo or newer)
2. SDK files are pre-placed: app/libs/UHF_SDK-debug.aar and app/libs/scans.jar — no additional setup needed
3. Select Build → Generate Signed Bundle / APK from the menu
4. Choose APK, then create a new keystore or select an existing one; finish the build wizard
5. Transfer the generated APK to the CX1500N device via USB cable or cloud storage
6. On the device: open Settings → Security → enable 'Unknown Sources' (or 'Install Unknown Apps')
7. Locate the APK with a file manager and tap to install

2.5 First-Time Device Setup

> After installation, the Accessibility Service MUST be enabled manually. This is a one-time setup per device.

8. Open the 'Identium UHF Keyboard' app on the device
9. Tap 'Open Accessibility Settings' — this opens Android's Accessibility settings
10. Scroll to find 'Identium UHF Keyboard Emulator' in the list of installed services
11. Tap it, then toggle the service ON and confirm the system warning dialog
12. Press the Back button to return to the main app screen
13. Toggle 'Enable Keyboard Mode' to ON in the main app UI
14. Open any target app (e.g., Notes, your WMS system, SAP, or a browser)
15. Tap inside a text field to place keyboard focus in it
16. Press the physical scan trigger — the scanned RFID/barcode data types automatically into the field

2.6 Settings Reference

Setting	Options	Default	Description
Scan Mode	UHF / Barcode / Both	Both	Which scanner hardware to use when trigger is pressed
Prefix	Any text string	(empty)	Text prepended to every scan result before output

Setting	Options	Default	Description
Suffix	None / Enter / Tab / Space	Enter	Terminator character appended after each scan result
Output Format	EPC Only / EPC+TID	EPC Only	What tag data to output — EPC alone or EPC with TID bytes
RF Power	5–33 dBm	33 dBm	UHF transmit power; higher = longer range, more power consumption
Vibrate	On / Off	On	Haptic feedback pulse on successful scan
Beep	On / Off	On	Audio beep on successful scan
Auto-start	On / Off	Off	Automatically start the scanning service when device boots

2.7 Barcode Scanner Note

The barcode module broadcasts results using the Android Intent action: com.android.server.scannerservice.broadcast The scanned data is available in the Intent extra with key: scannerdata This is the standard CX1500N scan SDK broadcast action. BarcodeReceiver.java registers for this broadcast automatically.

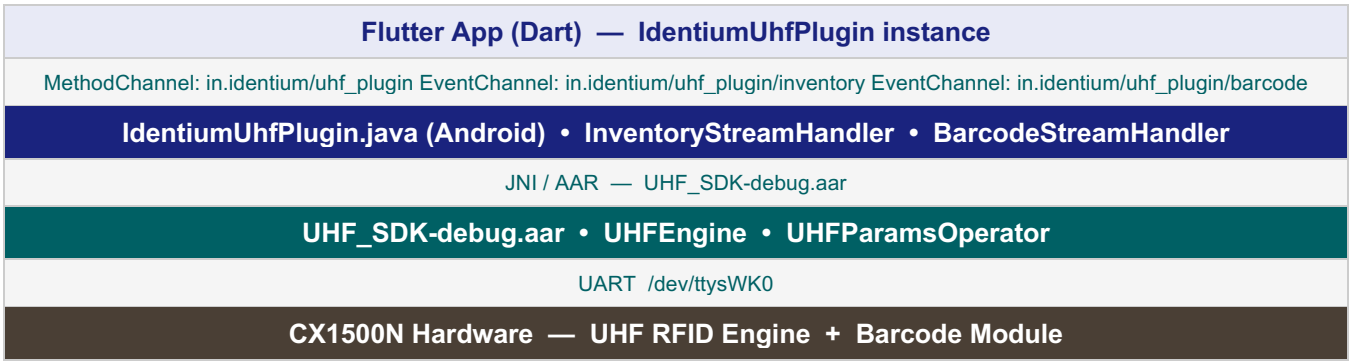
Section 3: Flutter Plugin — identium_uhf_plugin

3.1 Overview

The identium_uhf_plugin is a Flutter plugin that wraps the CX1500N UHF SDK for use in Flutter applications. It exposes a clean Dart API through a MethodChannel for one-shot commands and two EventChannels for continuous streaming of RFID tag and barcode scan data.

- MethodChannel: in.identium/uhf_plugin — for commands like connect, read, write, lock, kill
- EventChannel: in.identium/uhf_plugin/inventory — continuous stream of UhfTag objects
- EventChannel: in.identium/uhf_plugin/barcode — continuous stream of BarcodeResult objects
- Full async/await API — all calls return Futures
- Dart data models: UhfTag, BarcodeResult, RfPower, UhfResult

3.2 Plugin Architecture



3.3 Installation

Add the plugin to your Flutter project's pubspec.yaml:

```
dependencies:
  identium_uhf_plugin:
    path: ../identium_uhf_plugin
```

Also copy UHF_SDK-debug.aar into your app's android/libs/ directory and ensure it is referenced in android/app/build.gradle: implementation(name: 'UHF_SDK-debug', ext: 'aar')

3.4 Complete API Reference

Method	Parameters	Returns	Description
connect()	—	Future<bool>	Power on UHF module and connect via UART /dev/ttysWK0
disconnect()	—	Future<void>	Stop inventory, disconnect from module, power off
readEpc()	—	Future<UhfTag?>	Read EPC from a single tag in the field
readEpcTid()	filterEpc: String?	Future<UhfTag?>	Read EPC + TID (12 bytes) with optional EPC filter
readEpcTidUser()	filterEpc: String?	Future<UhfTag?>	Read EPC + TID + User bank data
readSingleTag()	filterEpc: String?	Future<UhfTag?>	Read single tag with optional EPC filter
startInventory()	mode, readTid, readUser, readRssi, readFrequency	Future<bool>	Start continuous multi-tag inventory with options
stopInventory()	—	Future<void>	Stop active inventory session
inventoryStream	—	Stream<UhfTag>	Live stream of scanned tags during inventory
barcodeStream	—	Stream<BarcodeResult>	Live stream of barcode/QR scan results
readBarcode()	—	Future<BarcodeResult?>	One-shot barcode scan trigger
setRfPower()	readPower: int, writePower: int (5–33 dBm)	Future<bool>	Set RF transmit power for read and write operations
getRfPower()	—	Future<RfPower?>	Get current read and write power settings
setRfMode()	RfMode enum	Future<bool>	Set Gen2 RF encoding mode (FM0 / M2 / M4 / M8)
setGen2Session()	Gen2Session enum	Future<bool>	Set Gen2 session parameter (0–3)
setGen2Target()	Gen2Target enum	Future<bool>	Set Gen2 target (A / B / A→B / B→A)
setRfRegion()	RfRegion enum	Future<bool>	Set RF regulatory region (NA / EU / China / Global)
writeEpc()	targetEpc, newEpc, password (hex strings)	Future<UhfResult>	Write a new EPC value to a tag
writeUserData()	targetEpc, startBlock, data, password (hex strings)	Future<UhfResult>	Write data to the User memory bank
lockTag()	targetEpc, lockObject, lockType, password	Future<UhfResult>	Lock or permanently lock a memory bank
killTag()	targetEpc, password (hex strings)	Future<UhfResult>	Permanently destroy / kill a tag (IRREVERSIBLE)
readTagData()	targetEpc, bank, startBlock, blockCount, password	Future<String?>	Raw memory bank read, returns hex string

3.5 Data Models

Model	Fields	Description
UhfTag	epc, tid, userData, rssi, frequency	Represents a scanned RFID tag with all available data fields
BarcodeResult	data, symbology	Barcode scan result with raw data and symbology type (e.g., CODE_128, QR_CODE)
RfPower	readPower, writePower	Current RF power settings in dBm for read and write operations
UhfResult	success, error, data	Generic operation result with success flag, error message, and optional data

3.6 Enum Reference

InventoryMode

Value	Description
normal	Standard inventory mode
e7Fast	E7 fast inventory — optimised for dense tag environments
singleTag	Single tag mode — reads only one tag
e7ReadStop	E7 read-stop mode
e7Smart	E7 smart mode — adaptive algorithm
multiTagFast	Multi-tag fast mode — recommended for general use

Gen2Session & Gen2Target

Enum	Values	Notes
Gen2Session	session0, session1, session2, session3	Session 1 recommended for most applications
Gen2Target	a, b, aToB, bToA	aToB cycles through all tags; use a for normal operation

RfRegion

Value	Frequency	Region
northAmerica	902–928 MHz	FCC / North America
europe	865–868 MHz	ETSI / Europe
china	920–925 MHz	SRRC / China
global	Adaptive	Global / multi-region

LockObject & LockType

Enum	Values
LockObject	accessPassword, killPassword, epcBank, tidBank, userBank
LockType	unlock, lock, permanentLock

RfMode — Encoding Modes

Value	Link Freq	Modulation	Notes
fm_203	250 kHz	FM0	High speed
fm_111	250 kHz	M2	—
fm_220	300 kHz	FM0	—
fm_101	300 kHz	M2	—
fm_45	40 kHz	FM0	Dense tags
fm_115	40 kHz	M4	—
fm_112	40 kHz	M8	—
fm_103	40 kHz	M2	—
fm_105	40 kHz	M4	Default recommended
fm_107	40 kHz	M8	Long range
fm_113	40 kHz	M4	—

3.7 Code Examples

1. Connect to Reader

```
final uhf = IdentiumUhfPlugin();
final connected = await uhf.connect();
if (connected) {
  print('Reader connected successfully');
} else {
  print('Failed to connect — check UART path');
}
```

2. Read Single Tag

```
final tag = await uhf.readSingleTag();
if (tag != null) {
  print('EPC: ${tag.epc}');
  print('RSSI: ${tag.rssi} dBm');
  print('TID: ${tag.tid}');
} else {
  print('No tag found in field');
}
```

3. Multi-Tag Inventory Stream

```
final subscription = uhf.inventoryStream.listen((tag) {
  print('Found: ${tag.epc} | RSSI: ${tag.rssi} dBm');
});

await uhf.startInventory(
  mode: InventoryMode.multiTagFast,
  readTid: true,
  readRssi: true,
);

// Later — stop inventory:
await uhf.stopInventory();
await subscription.cancel();
```

4. Read EPC Only

```
final tag = await uhf.readEpc();
print('EPC: ${tag?.epc}');
```

5. Read EPC + TID (with filter)

```
final tag = await uhf.readEpcTid(
  filterEpc: 'E20034120123456789ABCDEF',
);
print('EPC: ${tag?.epc},  TID: ${tag?.tid}');
```

6. Read EPC + TID + User Bank

```
final tag = await uhf.readEpcTidUser();
print('EPC: ${tag?.epc}');
print('TID: ${tag?.tid}');
print('User bank: ${tag?.userData}');
```

7. Barcode / QR Stream

```
uhf.barcodeStream.listen((bc) {
  print('Barcode: ${bc.data} (${bc.symbology})');
  // e.g. "6901234567890 (CODE_128)"
});
```

8. Set RF Power

```
final ok = await uhf.setRfPower(readPower: 30, writePower: 30);
print(ok ? 'Power set to 30 dBm' : 'Failed to set power');
```

9. Configure RF Mode, Session, Target & Region

```
await uhf.setRfMode(RfMode.fm_105);            // M4 @ 40 kHz (default)
await uhf.setGen2Session(Gen2Session.session1); // Session 1
await uhf.setGen2Target(Gen2Target.aToB);       // Cycle A → B
await uhf.setRfRegion(RfRegion.china);          // 920–925 MHz
```

10. Write EPC to Tag

```
final result = await uhf.writeEpc(
  targetEpc: 'E20034120123456789ABCDEF',   // current EPC (hex)
  newEpc:    'E2003412000000000000001',    // new EPC to write (hex)
  password:  '00000000',                   // 8-char hex access password
);
print(result.success ? 'EPC written successfully!' : 'Error: ${result.error}');
```

11. Lock a Memory Bank

```
final result = await uhf.lockTag(
  targetEpc:  'E2003412000000000000001',
  lockObject: LockObject.epcBank,
  lockType:   LockType.lock,
  password:   '00000000',
);
print(result.success ? 'EPC bank locked' : 'Lock failed: ${result.error}');
```

12. Kill (Permanently Destroy) a Tag

Warning: IRREVERSIBLE — killing a tag permanently disables it. It cannot be recovered.

```
// WARNING: This operation is IRREVERSIBLE
final result = await uhf.killTag(
  targetEpc: 'E2003412000000000000001',
  password:  '00000000',   // 8-char hex kill password (must not be all zeros in
production)
);
print(result.success ? 'Tag permanently killed' : 'Kill failed: ${result.error}');
```

Section 4: Flutter Sample App — identium_uhf_sample

4.1 Overview

The identium_uhf_sample is a complete reference Flutter application demonstrating all 12 plugin capabilities. It provides a polished UI with 6 main screens, real-time tag streaming, full read/write/lock/kill operations, RF settings configuration, and barcode scanning.

- Screen 1: Home — connection status and navigation tiles
- Screen 2: Inventory — real-time multi-tag streaming with statistics
- Screen 3: Single Tag — EPC/TID/User bank read with mode picker
- Screen 4: Read/Write/Lock/Kill — tabbed interface for tag operations
- Screen 5: Barcode/QR — continuous barcode stream with history list
- Screen 6: RF Settings — power slider, mode chips, session, region

4.2 Screen Wireframes

Screen 1: Home Screen

Identium UHF Reader	
Brand Card	identium | Creating identities for all | CX1500N SDK v1.2
Status	Reader: Disconnected
Action	[Connect Reader] (button)
Tile 1	Inventory (blue)
Tile 2	Single Tag (blue)
Tile 3	Read / Write / Lock / Kill (red)
Tile 4	Barcode / QR (teal)
Tile 5	RF Settings (purple)

Screen 2: Multi-Tag Inventory

Multi-Tag Inventory [Clear]	
Controls	[Read TID] [Read User] [START] (green)
Stats	Tags Found: 0 | Mode: Multi Fast
————————————————————	

Tag 1	E20034120123456789ABCDEF | -65 dBm
Tag 2	E200341201234567890ABCDE | -72 dBm
Tag 3	E20034120ABCDEF012345678 | -68 dBm

Screen 3: Single Tag Read

Single Tag Read	
Mode	(EPC Only) (EPC+TID) (EPC+TID+User)
Action	[Read Tag] (button)
—— Result ————————————	
EPC	E20034120123456789ABCDEF
TID	E200340305004019A54F0000
RSSI	-65 dBm
Freq	920.625 MHz

Screen 4: Read / Write / Lock / Kill

Tag Operations	
Tabs	[Write EPC] [Write User] [Lock] [Kill]
—— Write EPC Tab ————————	
Target EPC	E20034120123456789ABCDEF
New EPC	E2003412000000000000001
Password	00000000
Action	[Write EPC] (button)
Result	(result appears here)

Screen 5: Barcode / QR Scanner

Barcode / QR Scanner	
Info	Press the physical trigger to scan barcode or QR code
—— Scan History ————————	
Scan 1	6901234567890 | CODE_128

Scan 2	https://identium.in | QR_CODE
Scan 3	PDF417 data sample... | PDF_417

Screen 6: RF Settings

RF Settings	
RF Power	5 ——————●——— 33 dBm (30 dBm)
Inventory Mode	[Normal] [E7 Fast] [Single] [Multi Fast ✓] [E7 Smart]
Gen2 Session	[Session 0] [Session 1 ✓] [Session 2] [Session 3]
Gen2 Target	[A] [B] [A→B ✓] [B→A]
Region	[NA] [EU] [China ✓] [Global]
Action	[Apply Settings] (button)

4.3 Navigation Flow

Home	Inventory	Single Tag	Read/Write/ Lock/Kill	Barcode/QR	RF Settings
Main hub	Stream scan	One tag	Operations	Scanner	Configure

4.4 How to Build & Run

```
# Navigate to the sample app directory
cd identium_uhf_sample

# Get Flutter dependencies
flutter pub get

# Run in release mode on connected CX1500N device
flutter run --release

# Or build APK for distribution
flutter build apk --release
```

> The device must be connected via USB with USB debugging enabled. Ensure the Flutter SDK and Android toolchain are properly configured (flutter doctor).

4.5 Dependencies

Package	Version	Purpose
identium_uhf_plugin	1.0.0	UHF hardware interface — local path dependency
provider	^6.1.1	State management for reactive UI updates
google_fonts	^6.1.0	Inter font family for professional typography

Section 5: SDK Quick Reference Card

5.1 UHF RF Power Reference

dBm	cdBm (SDK)	Typical Range	Use Case
5	500	< 0.5 m	Proximity / single-item anti-collision
10	1000	~1 m	Short range / conveyor belt
15	1500	~2 m	Dense tag environments
20	2000	~3 m	Standard warehouse operation
25	2500	~5 m	Medium range / shelving
30	3000	~8 m	Long range / dock door
33	3300	~10 m	Maximum range / bulk reading

SDK Note: The SDK internally uses centidBm (cdBm). The plugin automatically converts dBm to cdBm. Actual read range depends on antenna design, tag type, and environment.

5.2 Tag Memory Bank Reference

Bank	ID	Address Notes	Readable	Writable
Reserved	0x00	Kill password @ word 0; Access password @ word 2	Yes*	Yes*
EPC	0x01	PC+CRC @ words 0–1; EPC starts @ word 2	Yes	Yes
TID	0x02	Tag manufacturer / chip model info	Yes	No
User	0x03	User-defined data, starts at word 0	Yes	Yes

** Reserved bank requires correct access password to read/write.*

5.3 RF Mode Reference

Index	FM Value	Link Freq	Modulation	Notes
0	203	250 kHz	FM0	High speed
1	111	250 kHz	M2	—
2	220	300 kHz	FM0	—

Index	FM Value	Link Freq	Modulation	Notes
3	101	300 kHz	M2	—
4	45	40 kHz	FM0	Dense tags
5	115	40 kHz	M4	—
6	112	40 kHz	M8	—
7	103	40 kHz	M2	—
8	105	40 kHz	M4	Default — recommended for most use cases
9	107	40 kHz	M8	Long range / difficult environments
10	113	40 kHz	M4	—

5.4 Error Code Reference

Error Code	Meaning	Recommended Action
MT_OK_ERR	Operation completed successfully	None — check result data
MT_RFID_NO_TAG_ERR	No tag found in the RF field	Retry, move tag closer, or increase RF power
MT_CMD_FAILED_ERR	SDK command failed unexpectedly	Check USB connection; call disconnect() then connect()
MT_PARA_ERR	Parameter error — invalid input	Verify EPC/password are valid hex strings with correct length
MT_NOT_CONNECT	Module not connected	Call connect() before any read/write operation

Section 6: Troubleshooting

The following table covers the most common issues encountered when developing with the CX1500N SDK:

Issue	Likely Cause	Solution
connect() returns false	UART path /dev/ttysWK0 not found, or UHF module power issue	Verify the UART device exists on the physical device; ensure the module is powered on before calling connect()
No tags found in field	RF power too low, wrong frequency region, or tag out of range	Increase power to 33 dBm via setRfPower(); set the correct RfRegion for your country; ensure tag is within range
Keyboard mode not working	Accessibility Service not enabled, or service was killed by Android	Enable 'Identium UHF Keyboard Emulator' in Android Settings → Accessibility; disable battery optimization for the app
Barcode not triggering	Wrong broadcast action or BarcodeReceiver not registered	Verify broadcast action com.android.server.scannerservice.broadcast is correct for the device firmware version
Write EPC fails	Incorrect access password or EPC bank is locked	Use correct 8-character hex access password; check lock state with readTagData() on Reserved bank
killTag() fails	Kill password is wrong or tag has not had kill password set	Provide the correct 8-character hex kill password; note that all-zero kill password ('00000000') means kill is disabled
Flutter build fails — AAR not found	UHF_SDK-debug.aar missing from android/libs/ directory	Copy UHF_SDK-debug.aar into android/libs/; ensure build.gradle includes: implementation(name: 'UHF_SDK-debug', ext: 'aar')

Additional Resources

- Email support: info@identium.in
- Website: identium.in | indiarfidstore.com | rfidindia.in
- Phone: 011-47147839 / 7011001472
- Address: Plot No. 5, First Floor, Santnagar, East of Kailash, New Delhi 110065
- GSTIN: 07AAGCI6275E1ZH

identium

Creating identities for all

Identium Tech Solutions Private Limited
Plot No. 5, First Floor, Santnagar, East of Kailash, New Delhi 110065
Phone: 011-47147839 / 7011001472 | Email: info@identium.in
identium.in | indiarfidstore.com | rfidindia.in
GSTIN: 07AAGCI6275E1ZH