YGF-Tech Scan Service — API Documentation

Item	Content
Document version	v3.3
Date	2026-9-16
Service package	`com.eastaeon.scandecoderservice`
Basis	Verified line by line against the actual service source (`ScanDecoderService/`)
Sample project	`ScanDemo/` (buildable test APK + source, bilingual UI)
Replaces	The previous *Scan service interface definition (Scan_service.doc)*

0. Summary

The scan service exposes broadcast interfaces only — no SDK, no AIDL, no .so.

```
Your app ──sendBroadcast(control / settings)──▶ Scan service ──sendBroadcast(result)──▶ Your app
```

Three steps to integrate:

```
// 1) Register the result receiver (must be registered in code, NOT in AndroidManifest.xml)
IntentFilter f = new IntentFilter("com.android.server.scannerservice.broadcast");
registerReceiver(receiver, f, Context.RECEIVER_EXPORTED);

// 2) To trigger a scan from software, send one broadcast
Intent i = new Intent("com.ygf.scan_service_control_action");
i.putExtra("command", "startScanOneTimes");      // single scan
sendBroadcast(i);

// 3) Read the barcode inside the receiver
String code   = intent.getStringExtra("scannerdata");
int    codeId = intent.getIntExtra("codeId", 0);   // barcode type code
String type   = intent.getStringExtra("codeType"); // symbology name, e.g. "Code 128"
String aim    = intent.getStringExtra("aimId");    // AIM ID, e.g. "]Cm"
int    x1     = intent.getIntExtra("x1", 0);       // barcode corner coordinates
```

`ScanDemo` wraps all of this in **one file, `ScanApi.java`** — copy it into your project and you are done.

1. Service basics

Item	Value
Package	`com.eastaeon.scandecoderservice`
Identity	`sharedUserId=android.uid.system`, `persistent=true`, `coreApp=true` (system-level always-on app)
minSdk / targetSdk	19 / 23
Permissions required by your app	**None.** Sending and receiving broadcasts needs no permission

No configuration is required on your side. As long as the service is running on the device, the interfaces below are available.

2. Receiving scan results

2.1 The result broadcast (the only one)

After decoding a barcode the service sends **exactly one** result broadcast:

Item	Value
action	`com.android.server.scannerservice.broadcast` (default, configurable)
barcode extra	`scannerdata` (default, configurable)
Sent when	`barcode_send_mode` is `BROADCAST` or `FOCUS_BROADCAST`

2.1.1 All extras of the result broadcast

extra	Type	Description
`scannerdata`	String	**Barcode content** (may include prefix / suffix / end character). Key name is set by `key_barcode_broadcast`
`dataType`	String	**Symbology name**, e.g. `Code 128`, `QR Code`, `EAN-13`; empty string for unknown types
`codeType`	String	Same as `dataType` (new in v3.1 — a clearer alias for customers)
`codeId`	int	**Barcode type code** (= ASCII code

		of the CodeID character), e.g. `106` = Code 128, `115` = QR. **Table in §2.5**
`aimId`	String	**AIM ID**, e.g. `]Cm`, `]Qm`, `]Am`. **Table in §2.5**; empty string on non-SYC devices or for unknown types
`x1 y1`	int	Coordinate of barcode corner 1
`x2 y2`	int	Coordinate of barcode corner 2
`x3 y3`	int	Coordinate of barcode corner 3
`x4 y4`	int	Coordinate of barcode corner 4

About the coordinates

- The four points are the barcode corners, in the order returned by the decoder library; connect them 1→2→3→4→1 to draw the barcode outline;
- The coordinate system is that of the **image fed to the decoder**: **width 1280 × height 800**, origin at the **top-left**;
- To map them onto your own preview / image, just scale: `x_yours = x1 * your_width / 1280`;
- This device runs only one decode channel (1280×800), so all coordinates share the same system;
- In rare cases (result produced by another decode path) the coordinates are `0`.

Service-side implementation (`ShunfengControlService.onDecodeTwoResultCallback`):

```
Intent intent = new Intent(settings.getString("action_barcode_broadcast",
                                              "com.android.server.scannerservice.broadcast"));
intent.addFlags(Intent.FLAG_RECEIVER_FOREGROUND);
intent.putExtra(settings.getString("key_barcode_broadcast", "scannerdata"), resultString);
intent.putExtra("dataType", CodeTypeID.getCodeType(codeId));
intent.putExtra("codeId",   codeId);
intent.putExtra("codeType", CodeTypeID.getCodeType(codeId));
intent.putExtra("aimId",    CodeTypeID.getAimId(codeId));
intent.putExtra("x1", x1);  intent.putExtra("y1", y1);
intent.putExtra("x2", x2);  intent.putExtra("y2", y2);
intent.putExtra("x3", x3);  intent.putExtra("y3", y3);
intent.putExtra("x4", x4);  intent.putExtra("y4", y4);
sendBroadcast(intent);
```

Recommendation: keep the default action and key.

If you really need your own action name:

```
Intent i = new Intent("com.android.service_settings");
i.putExtra("action_barcode_broadcast", "com.myapp.SCAN_RESULT");
i.putExtra("key_barcode_broadcast",    "barcode");
sendBroadcast(i);
```

After changing them you **must re-register your receiver with the new action/key**, otherwise you will receive nothing

(the "Apply broadcast name" button in `ScanDemo` demonstrates this).

> ⚠ **Only this main broadcast carries the v3.1 fields `codeId` / `codeType` / `aimId` / coordinates.**
>
> For backward compatibility with domestic courier customers the service also fires several historical alias broadcasts
>
> (`cn.sto.pda.scan.result`, `nlscan.*`, `com.sunmi.*`, ...) — those do **not** contain the new fields.
>
> Customer apps should listen to the main broadcast above only.

2.2 Three things you must get right on the receiving side

Point	Detail
Register in code only	Custom implicit broadcasts are not delivered to receivers declared in `AndroidManifest.xml` since Android 8.0
Android 13+ requires the exported flag	`registerReceiver(r, filter, Context.RECEIVER_EXPORTED)`, otherwise it throws at runtime
Process must be alive	The result is delivered to the process that registered the receiver; a killed app receives nothing

2.3 No result? Check these three items in order

1. Is `barcode_send_mode` set to `FOCUS` / `FOCUS_OVERWRITE` / `CLIPBOARD` / `EMUKEY`? Those four **do not broadcast**;
2. Does your registered action/key match the service-side setting?
3. Is the scan service running? (Send `ACTION_START` once — see §3.1.)

2.4 Scan state broadcast (optional)

To know whether the scanner is currently scanning:

Item	Value

action	`com.eastaeon.receive_scan_control_action`
extra `command`	`onDecoderServiceReady` / `onDecoderConnected` / `onStartDecoderConnect` / `onStartDecoderDisconnect` / `onDecoderDisconnected` / `onDecoderConnectFailed` / `isScaning`
extra `isScaningvalue`	Boolean, only for the `isScaning` event
extra `data`	String, usually empty

2.5 Barcode type: `codeId` / `codeType` / `aimId` table

This device uses the **SYC (Shengyuan / HS) decoder library**. The `codeId` in that library's callback is simply the

ASCII code of the CodeID character (for example the CodeID of Codabar is the character `a`, so `codeId = 97`).

The table below corresponds line by line to the "supported symbologies" list in the vendor document

解码库 Android 版本(Java)AO.pdf.

`codeId`	CodeID char	`aimId`	`codeType` / `dataType`
97	`a`	`]Fm`	Codabar
104	`h`	`]Hm`	Code 11
106	`j`	`]Cm`	Code 128 (generic ID, also covers GS1-128 / ISBT 128)
60	`<`	`]X0`	Code 32
98	`b`	`]Am`	Code 39
84	`T`	`]L2`	TLC39
105	`i`	`]Gm`	Code 93 / 93i
71	`G`	`]X0`	BC412
110	`n`	`]X0`	Plessey (UK / Anker)
100	`d`	`]E0`	EAN-13 (incl. ISBN, incl. Add-On)
68	`D`	`]E4`	EAN-8 (incl. Add-On)
121	`y`	`]em`	GS1 DataBar

123	{	]em	GS1 DataBar Limited
125	}	]em	GS1 DataBar Expanded
73	I	]C1	GS1-128
33	!	]em	GS1 Composite
81	Q	]X0	China Post (Chinese 2 of 5)
101	e	]Im	Interleaved 2 of 5
109	m	]X0	Matrix 2 of 5
89	Y	]X0	NEC 2 of 5
102	f	]Rm	Straight 2 of 5 IATA
102	f	]S0	Straight 2 of 5 Industrial (shares CodeID with IATA)
103	g	]Mm	MSI
116	t	]Bm	Telepen
99	c	]E0	UPC-A (incl. Add-On)
69	E	]E0	UPC-E
69	E	]X0	UPC-E1 (shares CodeID with UPC-E)
122	z	]zm	Aztec Code
72	H	]X0	Han Xin
119	w	]dm	Data Matrix
120	x	]Um	MaxiCode
115	s	]Qm	QR Code
115	s	]Qm	rMQR (shares CodeID with QR)
46	.	]X0	Dot Code
88	X	]X0	Grid Matrix
86	V	]O6	Codablock A
113	q	]Om	Codablock F
114	r	]Lm	PDF417
82	R	]Lm	Micro PDF417

Notes on usage

Point	Detail
Shared CodeIDs	`102` (`f`) is both Straight 2 of 5 IATA and Industrial; `69` (`E`) is both UPC-E and UPC-E1; `115` (`s`) is both QR and rMQR. `codeId` alone cannot tell them apart — the preceding row is reported by convention
Unknown symbology	A `codeId` not in the table (including Micro QR, which has no CodeID assigned in the vendor table) → `codeType` = `""`, `aimId` = `""`; use the numeric `codeId` instead
Historical `dataType` bug	Before v3.1 `dataType` was **always an empty string** on this device (the decoder adapter only had hsm/sdc mappings). Fixed in v3.1 — it now returns the `codeType` value above
AIM ID format	Output verbatim as in the vendor "supported symbologies" list (including the `]` prefix and the modifier), e.g. `]Cm`. Character 1 is always `]`, character 3 is the modifier
Example test	"Is this a 2D code?" → `codeId` ∈ {115, 122, 119, 120, 72, 88, 114, 82, 86, 113}; "is it 1D?" → everything else

3. Controlling the scanner

3.1 Master switch (essential)

Action	Broadcast action	What the service does
Scan ON	`com.ygf.scandecoderservice.ShunfengControlService.ACTION_START`	Starts the service + sets `scanner_enable=true`
Scan OFF	`com.ygf.scandecoderservice.ShunfengControlService.ACTION_PAUSE`	Sets `scanner_enable=false` + stops the service
Restart	`com.eastaeon.scandecoderservice.ShunfengControlService.ACTION_RESTART`	Stops, then starts again after 80 ms

```
sendBroadcast(new Intent("com.ygf.scandecoderservice.ShunfengControlService.ACTION_START"));
```

> Once switched off, **neither the hardware key nor software triggering produces barcodes** — the whole scan feature is disabled.
>
> Recommended: send `ACTION_START` when your app starts or before entering the scan screen.

3.2 Software trigger (main interface)

> **v3.2 note**: `scanner_scan_continue` (continuous scanning) and `manual_continue_enabled` (scan-while-held) are mutually exclusive; the exclusion is maintained **by the scan service itself** — enabling one automatically disables the other. The client only needs to broadcast the parameter that actually changed, **no coupling/linked commands are required**.

Item	Value
action	`com.ygf.scan_service_control_action`
extras	`command` (see below), `packageName` (caller package, recommended)

command	Meaning	Notes
`startScanOneTimes`	**Single scan**	Stops after one decode or timeout; most used
`startScan`	**Start continuous scanning**	Runs until `stopScan` is received
`stopScan`	**Stop scanning**	—
`startScanOpenLaser`	Laser always on (aim mode)	For hardware supporting continuous laser
`setScanningParameter`	Set low-level engine parameter	Extra `param`(int), `value`(int)
`settings_changed`	Ask the service to reload settings	Optional after changing parameters

```
// Single scan
Intent i = new Intent("com.ygf.scan_service_control_action");
i.putExtra("command", "startScanOneTimes");
i.putExtra("packageName", getPackageName());
sendBroadcast(i);

// Continuous / stop
i.putExtra("command", "startScan");
i.putExtra("command", "stopScan");
```

The timeout of `startScanOneTimes` equals `scanner_timeout` (default 5000 ms).

If `manual_continue_enabled = true` (**factory default**), the single-scan timeout is forced to 60 s.

> **Stopping continuous scan**: after `startScan` a single `stopScan` now stops it immediately (fixed on the service side —
>
> between decodes in continuous mode there was an intermediate state where "not decoding but continuous scan still active"
>
> and the original code dropped the stop request, so several key presses were needed to stop).

3.3 Simulating the hardware scan key (alternative)

Item	Value
action	`com.ygf.floatingbutton.KEY_DOWN`
Option A	with `command` = `startScan` / `stopScan` / `startScanOneTimes`
Option B	with `keycode` = scan key code

Scan key codes recognized by the service: `227`, `284`, `285`, `286`, `287`, `293`, `317`, `318`, `84`.

> **`keycode` is not recommended:** that branch *toggles* (stops if scanning, starts if idle) and is affected by
>
> `manual_continue_enabled` / `lift_scan_enabled`, so the behaviour is hard to predict.
>
> Use the `command` form in §3.2 for precise control.

4. Settings

4.1 Which action to use

action	Parameter naming	Recommendation
`com.android.service_settings`	`scanner_xxx` / `sym_xxx`	✅**Use this one** — complete parameter set
`com.android.scanner.service_settings`	`sound_play` / `viberate` / `interval` …	Legacy only. Its translation table is incomplete: `scanner_timeout`, `manual_continue_enabled`, `lift_scan_enabled`, `shortest_barcode`, `longest_barcode`, `floating_button_enabled`,

		`reading_pane_enabled` **cannot be passed** through it

One broadcast may carry several parameters; the service processes them one by one.

Always include `packageName`.

4.2 Parameter reference

A. Feedback

Parameter	Type	Default	Description
`scanner_sound_play`	boolean	`true`	Beep on scan
`scanner_vibrate`	boolean	`false`	Vibrate on scan (80 ms)
`scan_indicatorlight_enabled`	boolean	`true`	Scan indicator LED (not blinking in continuous mode)
`scan_flash_enabled`	boolean	`true`	Illumination LED
`flashlight_mode`	String	`"150"`	Illumination level

> The beep is played by the pre-installed sound service after receiving a `com.eastaeon.android.PLAY_SOUND_EFFECT` broadcast. If that app is removed, `scanner_sound_play=true` stays silent.

B. Scan behaviour

Parameter	Type	Default	Description
`scanner_scan_continue`	boolean	`false`	`true` = continuous scanning; `false` = stop after one decode
`manual_continue_enabled`	boolean	`true`	Keep scanning while the hardware key is held
`lift_scan_enabled`	boolean	`false`	Scan when lifted up
`scanner_timeout`	int	`5000`	Single-scan timeout (ms), only applied if **≥1000**
`scanner_interval`	int	`1000`	Continuous-scan interval (ms), only applied if **≥10**

`boot_start`	boolean	`true`	Auto-start scan service on boot
`floating_button_enabled`	boolean	`false`	Floating button

C. Result handling

Parameter	Type	Default	Description
`barcode_send_mode`	String	`FOCUS_BROADCAST`	How the result is delivered, see §4.3
`scanner_prefix`	String	`""`	Barcode prefix
`scanner_suffix`	String	`""`	Barcode suffix
`scanner_endchar`	String	`ENTER`	End character, see §4.4
`filter_prefix_suffix_blank`	boolean	`false`	Trim leading/trailing blanks
`action_barcode_broadcast`	String	`com.android.server.scannerservice.broadcast`	Result broadcast action
`key_barcode_broadcast`	String	`scannerdata`	Result broadcast extra key
`barcode_chinese_coding`	String	`""`	Encoding for Chinese codes, e.g. `"GBK"`
`shortest_barcode`	int	—	Minimum barcode length (1–9999)
`longest_barcode`	int	—	Maximum barcode length (1–9999)
`result_prefix_control`	int	—	Drop the first N characters of the result
`result_output_control`	int	—	Drop the last M characters of the result

divider_settings	boolean	—	Auto-remove a trailing _ / -

D. Saving scan images

Parameter	Type	Default	Description
save_pic_enabled	boolean	false	Save the raw scan image
scan_save_pic_days	int	3	Retention in days (1–30)

E. Symbology switches (complete list, new in v3.1)

Naming rule: sym_<symbology>_<suffix>, **sent via the `com.android.service_settings` broadcast**:

Parameter	Type	Description
sym_<code>_enable	**boolean**	Enable/disable one symbology, e.g. sym_qr_enable = false
sym_<code>_min	**int**	Minimum length for that symbology
sym_<code>_max	**int**	Maximum length for that symbology
sym_all_code_enable	boolean	Enable (true) / disable (false) **all** symbologies at once
sym_reset_all_code_enable	boolean	Restore all symbology defaults + reset business defaults (send mode, end character, ...) + scanner_enable=true

> ⚠ **Types matter**: _enable must be putExtra(String, boolean), _min / _max must be putExtra(String, int).
>
> A string value is treated as the default, which looks like "the broadcast was sent but had no effect".

The 35 main switches (all symbologies supported on this device)

#	Parameter	Symbology		#	Parameter	Symbology
1	sym_aztec_enable	Aztec Code		19	sym_upca_enable	UPC-A
2	sym_codabar_enable	Codabar		20	sym_upce0_enable	UPC-E0/UPC-E

3	sym_code11_enable	Code 11		21	sym_upce1_upce1_enable	UPC-E1 ⚠ doubled suffix here
4	sym_code128_enable	Code 128		22	sym_iata25_enable	Straight 2 of 5 IATA
5	sym_code39_enable	Code 39		23	sym_codablock_a_enable	Codablock A
6	sym_code93_enable	Code 93		24	sym_codablock_f_enable	Codablock F
7	sym_datamatrix_enable	Data Matrix		25	sym_msi_enable	MSI
8	sym_ean8_enable	EAN-8		26	sym_tlcode39_enable	TLC39
9	sym_ean13_enable	EAN-13		27	sym_trioptic_enable	Trioptic Code 39
10	sym_int25_enable	Interleaved 2 of 5		28	sym_strt25_enable	Straight 2 of 5 Industrial
11	sym_maxicode_enable	MaxiCode		29	sym_matrix25_enable	Matrix 2 of 5
12	sym_micropdf_enable	Micro PDF417		30	sym_nec25_enable	NEC 2 of 5
13	sym_pdf417_enable	PDF417		31	sym_telepen_enable	Telepen
14	sym_qr_enable	QR Code		32	sym_usps4cb_enable	USPS 4 State
15	sym_rss_rss_enable	GS1 DataBar		33	sym_gs1_128_enable	GS1-128
16	sym_rss_rsl_enable	GS1 DataBar Limited		34	sym_hanxin_enable	Han Xin
17	sym_rss_rse_enable	GS1 DataBar Expanded		35	sym_hk25_enable	China Post (HK25)
18	sym_gridmatrix_enable	Grid Matrix				

There are also several fine-grained switches (normally no need to touch). ⚠ Note: fine-grained keys do **NOT** all follow the `sym_*_enable` boolean rule —

the inverse switches are **String-typed without the `_enable` suffix** (`sym_c128_inverse`, `sym_code93_inverse`, `sym_int25_inverse`, `sym_pdf417_inverse`,

values `"0"/"1"/"2"`, send with `--es`), and rectangular DataMatrix is a **boolean without suffix** (`sym_datamatrix_rectangle`, send with `--ez`).

Boolean fine-grained keys include: `sym_chinapost_enable`, `sym_koreapost_enable`, `sym_code32_enable`, `sym_code49_enable`,

`sym_composite_enable`, `sym_composite_upc_enable`, `sym_isbt_enable`, `sym_auspost_bar_output_enable`,

`sym_postnet_check_transmit_enable`, `sym_code39_check_enable`, `sym_code39_check_transmit_enable`, `sym_code39_fullascii_enable`,

`sym_ean13_2_digit_addenda_enable`, … (a mistyped key is silently ignored by the service — no error, no effect).

Examples

```
// Decode QR + Code 128 only, everything else off
Intent i = new Intent("com.android.service_settings");
i.putExtra("packageName", getPackageName());
i.putExtra("sym_all_code_enable", false);      // disable all first
i.putExtra("sym_qr_enable",       true);       // then enable QR
i.putExtra("sym_code128_enable",  true);       // then enable Code 128
i.putExtra("sym_qr_min", 1);                   // QR min length
i.putExtra("sym_qr_max", 500);                 // QR max length
sendBroadcast(i);

// Restore factory defaults (all symbologies + business parameters)
Intent r = new Intent("com.android.service_settings");
r.putExtra("packageName", getPackageName());
r.putExtra("sym_reset_all_code_enable", true);
sendBroadcast(r);
```

> Nothing else is needed afterwards: the service detects the symbology change and reloads the decoder engine itself.

F. Auxiliary

Parameter	Type	Description
`packageName`	String	Caller package name; recommended on every request

> The `pda_*` family (disable keys, hide status bar, silent APK install, …) is device-customization level and requires
>
> system privileges. Regular apps cannot use them; not covered here.

4.3 `barcode_send_mode` values

Value	Where the result goes	Broadcast?

`FOCUS_BROADCAST`	Focused input field **and** broadcast	✅factory default
`BROADCAST`	**Broadcast only**	✅recommended for your app
`FOCUS`	Focused input field only (append)	❌
`FOCUS_OVERWRITE`	Focused input field, overwrite	❌
`CLIPBOARD`	System clipboard	❌
`EMUKEY`	Typed as keyboard input, character by character	❌

> ⚠ Unless the mode is `BROADCAST` / `FOCUS_BROADCAST`, your app receives **no result broadcast at all**.
>
> If your app needs the data, **set `BROADCAST`**.

4.4 `scanner_endchar` values

Value	Appended to the result
`ENTER`	`\n` (**factory default**)
`NONE`	nothing
`TAB`	`\t`
`SPACE`	a space

Case sensitive — must be upper case.

4.5 When do settings take effect?

- Parameters are written to shared preferences and take effect on the next scan **immediately**;
- Engine-related parameters (symbology switches, timeout, interval, image saving)

trigger an automatic reload inside the service — no extra action needed;

- To force a full reload, send:

```
Intent i = new Intent("com.ygf.scan_service_control_action");
i.putExtra("command", "settings_changed");
sendBroadcast(i);
```

4.6 Complete example (several parameters in one broadcast)

```
Intent i = new Intent("com.android.service_settings");
i.putExtra("packageName",             getPackageName());
i.putExtra("scanner_sound_play",      true);        // beep
i.putExtra("scanner_vibrate",         true);        // vibrate
i.putExtra("scanner_scan_continue",   false);       // no continuous scan
```

```
i.putExtra("barcode_send_mode",       "BROADCAST"); // results via broadcast only
i.putExtra("scanner_endchar",         "ENTER");     // append newline
i.putExtra("scanner_timeout",         5000);        // ms, >= 1000
i.putExtra("scanner_interval",        1000);        // ms, >= 10
i.putExtra("scanner_prefix",          "");
i.putExtra("scanner_suffix",          "");
i.putExtra("sym_qr_enable",           true);        // enable the common symbologies only
i.putExtra("sym_code128_enable",      true);
i.putExtra("sym_code39_enable",       true);
sendBroadcast(i);
```

5. Other broadcast interfaces (optional)

5.1 Read the PDA serial number

```
sendBroadcast(new Intent("com.android.receive_get_pda_sn"));

// Result: action = com.android.receive_pda_sn, extra: pda_sn (register a receiver in code)
```

5.2 Temporarily mute the scan beep (global)

```
Intent i = new Intent("com.eastaeon.sdk.CLOSE_SCAN_SOUND");
i.putExtra("EXTRA_CLOSE_CTRL", true);      // true = mute, false = unmute
sendBroadcast(i);
```

Unlike `scanner_sound_play`, this is a global mute with higher priority.

6. Quick reference

I want to...	Send
Turn scanning on	`…ShunfengControlService.ACTION_START`
Turn scanning off	`…ShunfengControlService.ACTION_PAUSE`
Scan once	`com.ygf.scan_service_control_action` + `command=startScanOneTimes`
Start continuous scan	`com.ygf.scan_service_control_action` + `command=startScan`
Stop	`com.ygf.scan_service_control_action` + `command=stopScan`
Receive results	register `com.android.server.scannerservice.broadcast` in code, read `scannerdata`
Get the barcode type	same broadcast: `codeId`(int) / `codeType`(String) /

	`aimId`(String), see §2.5
Get the barcode position	same broadcast: `x1,y1 ~ x4,y4`, coordinate system 1280×800
Beep on/off	`com.android.service_settings` + `scanner_sound_play`
Vibrate on/off	`com.android.service_settings` + `scanner_vibrate`
Continuous mode	`com.android.service_settings` + `scanner_scan_continue`
Change send mode	`com.android.service_settings` + `barcode_send_mode`
Change end character	`com.android.service_settings` + `scanner_endchar`
Change result action/key	`com.android.service_settings` + `action_barcode_broadcast` / `key_barcode_broadcast`
Enable/disable a symbology	`com.android.service_settings` + `sym_<symbology>_enable` (list in §4.2-E)
Allow only specific symbologies	`sym_all_code_enable=false` then `sym_xxx_enable=true` one by one
Restore symbology defaults	`com.android.service_settings` + `sym_reset_all_code_enable=true`

7. FAQ

Q1. The hardware key decodes, but my app receives nothing.

A: Check `barcode_send_mode`. Only `BROADCAST` / `FOCUS_BROADCAST` broadcast. If it was changed to `FOCUS` / `CLIPBOARD`, nothing is sent — just set `barcode_send_mode=BROADCAST`.

Q2. `startScan` does nothing.

A: ① Is the master switch on (`ACTION_START`)? ② Is the screen on and unlocked? (No response while the screen is off or locked.) ③ Is the camera occupied by another app?

Q3. The result has an extra (or missing) newline.

A: Controlled by `scanner_endchar`, default `ENTER` (`\n`). Set it to `NONE` to remove.

Q4. `scanner_timeout` has no effect.

A: ① The value must be ≥1000. ② If you use the legacy action `com.android.scanner.service_settings`, this parameter is not forwarded — use `com.android.service_settings`. ③ If `manual_continue_enabled=true` (default), the single-scan timeout is forced to 60 s.

Q5. After changing `action_barcode_broadcast` I no longer receive results.

A: You must re-register your receiver with the new action/key.

Q6. `dataType` / `aimId` are empty strings.

A: ① The `codeId` of that result is not in the §2.5 table (for example Micro QR, which has no CodeID in the vendor table); ② `aimId` is currently only supported on SYC (Shengyuan) decoder devices — other devices return `""`. In that case use the numeric `codeId` instead.

Q7. What is the relationship between `codeId` and `dataType`?

A: `codeId` is the **raw type code from the decoder library** (ASCII code of the CodeID character) — stable and language-independent, so it is the recommended value for program logic; `codeType` / `dataType` is the **readable name** of the same symbology, better for logs or UI. They map one-to-one; see §2.5.

Q8. The coordinates `x1..y4` are all 0.

A: ① In rare cases the result comes from a non-callback decode path that carries no coordinates; ② coordinates are returned as 0 when the barcode was recognized but corner information is unavailable; ③ make sure the firmware on the device actually contains the v3.1 changes.

Q9. A symbology switch broadcast has no effect.

A: ① Check the extra type: `_enable` must be boolean, `_min`/`_max` must be int; ② make sure you use `com.android.service_settings` (the legacy action does not forward `sym_*`);

③ the easiest way to verify is `sym_all_code_enable=false` followed by enabling one specific symbology.

Q10. A broadcast seems to have no effect — how to isolate the problem?

A: Send the very same broadcast from a shell first to separate "interface problem" from "app problem":

```
adb logcat -s ShunFengReceiver ServiceReceiver ScanService ShunfengControlService
SYCDecoderServiceHelper

adb shell am broadcast -a com.ygf.scan_service_control_action --es command startScanOneTimes
adb shell am broadcast -a com.ygf.scandecoderservice.ShunfengControlService.ACTION_START
adb shell am broadcast -a com.android.service_settings --es barcode_send_mode BROADCAST
adb shell am broadcast -a com.android.service_settings --ez sym_qr_enable true
```

If the shell command produces a barcode, device and interface are fine and the problem is in your app; if not,

the problem is on the device/firmware side.