Context Picker: How Adaptive Verification Picks the Right Method Without Eroding Privacy

People & In-Person / Last updated 2026-06-11 / https://www.scrambleid.com/learn/context-picker-signals-and-privacy

> **Status (April 2026):** Planned. The Context Picker is on the roadmap but is not yet shipped. This article describes the design we're building toward and is published for architects evaluating the model now so they can plan around it. If you're evaluating ScrambleID for a near-term deployment whose success depends on Context Picker behavior, contact your ScrambleID account team for the current target timeline before committing.

In one sentence: The Context Picker is an adaptive UX layer that suggests the fastest, safest verification method (QR vs code vs link) based on a small, privacy-preserving set of device, environment, and user-history signals.

TL;DR (canonical)

- Capture only the signals that materially predict success and keep raw values local whenever possible.
- Prefer event-driven signals (call state, permissions, network type) over continuous surveillance.
- Store on-device for a short window (example: 30 days) and upload aggregates (not raw data) on a low cadence (example: daily).
- Start with a "no-regret" set that requires zero new permissions.

What the Context Picker does

When multiple initiation methods exist (QR, short code, link), users pick the wrong one when:

- the camera is blocked or lighting is poor,
- audio is routed to a headset or a noisy environment makes tones unreliable,
- network quality is unstable,
- the user is busy and wants the fastest path.

The Context Picker uses lightweight signals to suggest the default method and reduce friction.

Signal catalog (forward-looking)

Below is a practical catalog of signals that can be captured on iOS/Android. The design goal is simple:

- maximize predictive value,
- minimize permissions,
- minimize retained raw data.

Device and call state

- Telephony state (idle/ringing/active)
- Audio route (earpiece/speaker/Bluetooth/car)
- Headset connect events
- Focus / Do Not Disturb state

Environmental

- Ambient noise proxy (short mic sample)
- Ambient light proxy (camera preview) or screen brightness
- Motion state (still/walking/vehicle)
- Network type and quality (Wi-Fi vs cellular; RSSI proxy)

Permissions and capability flags

- Camera permission status
- Microphone permission status
- Battery level and battery saver mode

User history (local)

- Method chosen and success/failure
- Override count (user changed the suggested method)
- Fallback triggers
- Session duration (tap to verified)

Contextual

- Calendar "busy" status (coarse)
- Local time and locale

Security and policy

- Device policy flags (MDM restrictions)
- Trusted location geofence (opt-in only)

Storage and telemetry guidance

Recommended posture:

- Keep raw values local unless necessary for immediate UX.
- Store local history for a bounded window (example: 30 days).
- Retain server-side aggregates for a bounded, tenant-configurable window (example: 90 days), then delete.
- Upload only aggregates, not raw sensor readings.
- Hash or redact any identifiers.
- Protect cross-user aggregates with calibrated noise (differential-privacy style) so no single user behavior is recoverable from any published aggregate.

Decision matrix: signals → best method (starter)

The Context Picker should be **explainable**. If the UI defaults to "Enter a code" instead of "Scan a QR", the user should see a short reason (for example: "Camera access is off" or "Low light detected").

Explainability improves completion rates and makes privacy review easier because you can document exactly which signals drive which decisions.

Key signal(s)	Likely constraint	Recommended default	Why this reduces friction	Privacy note
Camera permission **denied** or camera unavailable	QR scan unlikely to work	**Short code** (typed)	Works on any device; no camera loop	Permission state only; no images captured
Ambient light poor / glare detected	QR scan unreliable	**Short code** (typed)	Avoid failed scans and timeouts	Use coarse "good/bad"; do not store frames
Active call + audio route Bluetooth/car	Attention constrained	**Short code** (typed)	One-handed; compatible with voice guidance	Audio route is event-based; no audio recording
Active call + noisy environment	Voice prompts unreliable	**Short code** (typed)	Avoid "say your code" failure	If using noise proxy, compute locally and discard raw
Network unstable / captive portal	Remote join may fail	**QR or short code**	Fewer remote fetches; faster retries	Network type/quality only

Key signal(s)	Likely constraint	Recommended default	Why this reduces friction	Privacy note
User overrides QR→code in same context ≥3 times	Personal preference	**Learned default**	Reduces repeated frustration	Keep local; upload aggregate only if opted-in

Rule of thumb

Start deterministic, then learn. A simple rule-based picker that avoids obvious failure modes usually beats a black-box model, especially in regulated environments.

Privacy checklist (copy/paste)

- [ ] **Purpose-limited:** signals exist only to improve method selection and reliability.
- [ ] **Minimized:** prefer coarse/boolean values (e.g., `light_ok`) over raw readings.
- [ ] **Ephemeral:** raw values stay on-device; upload only aggregates.
- [ ] **Tenant-controlled:** enterprises can select a strict posture (no history, no upload).
- [ ] **Transparent:** publish a signal inventory in your App Store / Play disclosures.
- [ ] **Auditable:** document which signals drive which decisions.

Minimum "no-regret" set to capture first

If you want high value with no new prompts, start with:

1. Call state and audio route
2. Camera and mic permission flags
3. Network and battery state
4. User behavior stats (method chosen, overrides, fallbacks, duration)

This is enough to improve default suggestions in most deployments without any new sensitive telemetry.

How to use the signals (practical policies)

A deterministic starter policy:

- If camera permission is denied or ambient light is poor: default to short code or link.
- If telephony is active and audio route is Bluetooth/car: avoid audio-based methods.
- If network is unstable: prefer QR or code over remote fetches.
- If the user has overridden the suggestion multiple times for a context: learn a personal default.

Concrete scenarios (how the picker changes outcomes)

1) Contact center caller is driving

- **Signals:** active call + Bluetooth/car audio route
- **Default:** *type a short code*
- **Why it matters:** scanning a QR while driving is unsafe and fails frequently; typed code completes quickly with voice guidance.

2) Shared kiosk or locked-down device

- **Signals:** MDM policy blocks camera or camera permission denied
- **Default:** *type a short code*
- **Why it matters:** QR becomes a dead-end; a code path avoids the false "this product is broken" perception.

3) Low light / glare environment

- **Signals:** coarse light proxy indicates poor conditions
- **Default:** *type a short code* (or offer *tap a link* as secondary)
- **Why it matters:** repeated scan attempts are the #1 cause of abandonment in scan-first flows.

4) Good camera, stable network, user is stationary

- **Signals:** camera available + light OK + stable network
- **Default:** *scan a QR*
- **Why it matters:** QR is the fastest path when it works; the picker should not force a slower method.

Key Takeaway

The Context Picker uses available signals (biometric readiness, network, device posture, requested attributes) to select the optimal verification method without over-asking. Privacy by design means the picker never collects signals it won't use, and all collection is purpose-limited and auditable.

FAQ

Is this "tracking" users?

It should not be. The goal is to keep signals minimal, mostly local, and upload only aggregates.

Do we need new permissions?

Not for the minimum set. Many high-value signals are already available without new prompts.

Why publish this spec?

Because adaptive method selection is a major UX differentiator, and publishing a clear privacy posture makes the system easier to trust and cite.

Can enterprises disable signal capture?

Yes. Enterprises should be able to select a strict telemetry posture.

References (public)

- NIST Privacy Framework (Version 1.0): https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.01162020.pdf
- Apple developer guidance on privacy manifests (SDK/data disclosure): https://developer.apple.com/documentation/bundleresources/privacy-manifest-files
- Google Play Data Safety disclosure guidance: https://support.google.com/googleplay/android-developer/answer/10787469
- OECD privacy principles (collection limitation, purpose specification): https://oecdprivacy.org/

Related reading

- People Trust Checks: Overview
- People Verification Implementation Guide
- Unified ID Card: Attribute Provenance