VOCEA · DEVELOPER / INTEGRATION GUIDE

Integrazione sicura con Vocea

Come un applicativo terzo (HR suite, portali compliance, intranet) si collega in modo sicuro alla piattaforma di whistleblowing Vocea — preservando anonimato e **zero-knowledge**, in conformità al D.Lgs. 24/2023 e al GDPR.

Server-to-server | **API key per-tenant** | **Zero-knowledge** | **TLS obbligatorio** | **D.Lgs. 24/2023 · GDPR**

INDICE

1. Modello di fiducia

L'API Vocea è pensata per uso **server-to-server**. Un applicativo terzo si autentica con una **API key per-tenant** e può inviare segnalazioni o leggerne metadati/stato per conto del *proprio canale* — mai di altri.

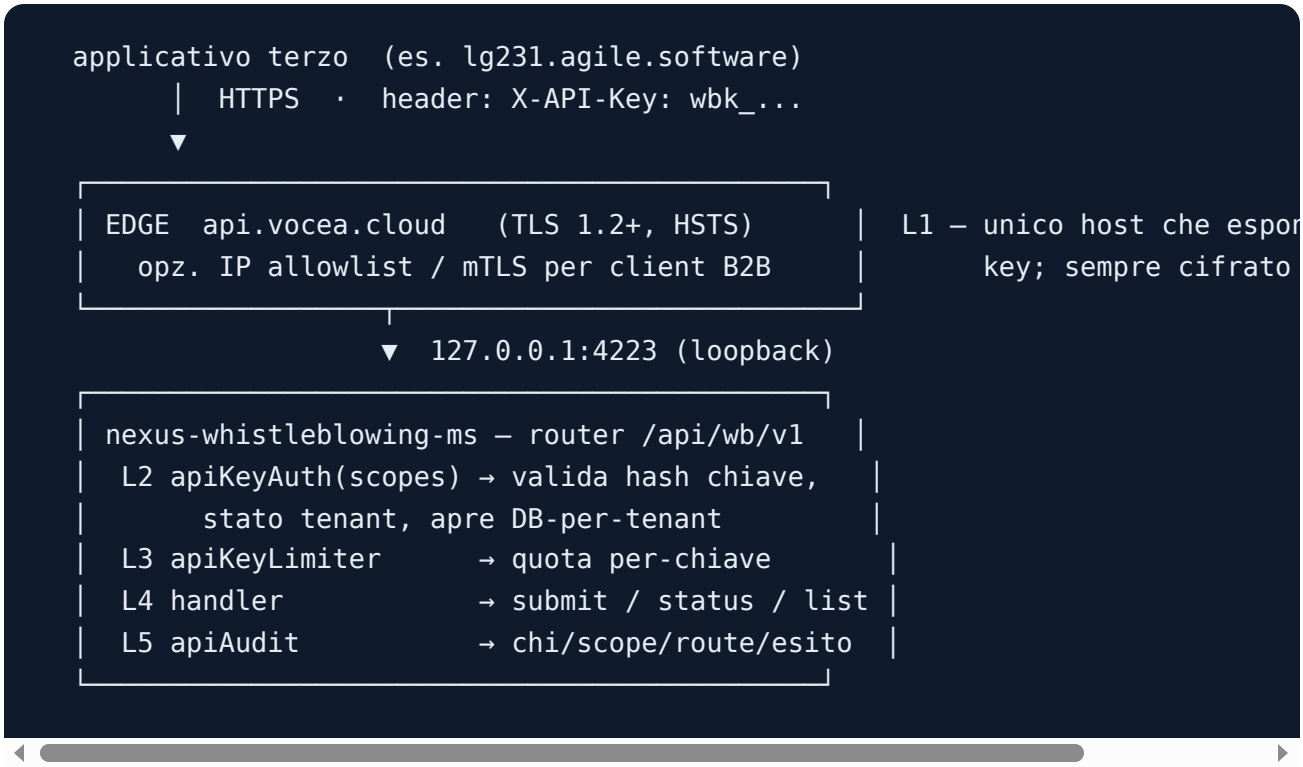
- **Multi-tenant isolato:** ogni API key è legata a un singolo canale (tenant). Risolta la chiave, il gateway commuta sul **database del singolo tenant**: una chiave non vede mai i dati di un altro canale, né il control-plane di piattaforma.
- **Zero-knowledge preservato:** il contenuto delle segnalazioni viaggia e viene archiviato **solo cifrato**. Il server non possiede le chiavi private e

non decifra mai. L'API espone metadati e, dove previsto, ciphertext — **mai** testo in chiaro.

- **Anonimato by design:** l'IP del segnalante non viene mai persistito (D.Lgs. 24/2023 art. 12). Gli audit registrano solo metadati, con IP pseudonimizzato (HMAC).

> In breve: **tutto passa cifrato attraverso Vocea, ma Vocea non può leggerlo.** L'integratore è responsabile di cifrare il payload *prima* dell'invio (vedi §5).

2. Architettura di sicurezza



Il microservizio (`:4223`) **non è mai esposto a internet**: solo l'edge proxy lo raggiunge in loopback. La chiave viene confrontata per **hash SHA-256** (nel DB non è mai salvata in chiaro). Ogni chiamata è tracciata su un audit con hash-chain HMAC, senza contenuti sensibili.

3. Autenticazione — API key

Tutte le chiamate `/v1/integrations/*` richiedono l'header:

```
X-API-Key: wbk_<prefix>_<secret>
```

Formato: prefisso `wbk_` + identificativo (prefix) + segreto ad alta entropia. Nel database Vocea conserva **solo** l'hash SHA-256 della chiave più il prefix (per identificarla): la chiave intera è mostrata **una sola volta**, alla creazione.

Ciclo di vita della chiave

Le chiavi si gestiscono dalla **console del gestore del canale** (sezione «API & Integrazioni»), non da un admin di piattaforma:

Operazione	Dove
Emissione (mostrata una volta)	Console gestore → «API & Integrazioni» → *Crea chiave* (scegli gli scope)
Elenco (prefix, label, scope, stato, ultimo uso)	Console gestore (la chiave non è mai più visibile)
Revoca	Console gestore → *Revoca* (immediata)
Rotazione	Emetti nuova → aggiorna il client → revoca la vecchia (finestra di overlap)

> **Conservazione del segreto.** Tratta la chiave come una credenziale di produzione: salvala in un **secret manager / vault**, mai in codice, in repository git o in variabili d'ambiente committate. **La chiave è solo backend-to-backend**: non usarla mai da un browser o da codice client-side (sarebbe esposta).

4. Scope & permessi

Ogni chiave porta uno o più scope; ogni endpoint richiede lo scope minimo. Principio del privilegio minimo: assegna solo ciò che serve.

Scope	Consente
`reports:submit`	Inviare una segnalazione cifrata al canale.
`status:read`	Leggere lo *stato* di una segnalazione tramite codice di tracking (solo metadati).
`reports:read`	Elencare i metadati delle segnalazioni del canale (+ ciphertext, mai plaintext).
`keyholders:manage`	Gestire i key holder del **proprio** canale (enroll/elenco/disattivazione). Solo chiave pubblica RSA (zero-knowledge).
`delegates:manage`	Gestire deleghe/gestori del proprio canale (invito/elenco).

Gli scope di gestione sono **opzionali e per-tenant**: ogni applicativo amministra solo la **propria** organizzazione; non vede né tocca altri canali, né può creare altre API key.

5. Confine zero-knowledge — come cifrare

Vocea **non decifra mai**. Per inviare una segnalazione, l'integratore deve cifrare il payload **client-side**, prima della chiamata, con le chiavi pubbliche del canale.

5.1 — Recupera le chiavi pubbliche del canale

Endpoint pubblico (nessuna API key), per `slug` del canale:

```
GET https://vocea.cloud/api/wb/tenants/{slug}/info
```

Ritorna, tra l'altro, l'elenco dei `key_holders` attivi e — fondamentale — il **compartimento** che ciascuno controlla:

```
{
  "tenant": { "id": "...", "name": "...", "slug": "acme", "default_locale": "i
  "key_holders": [
    { "id": "kh_1", "role": "...", "controls_key": "K_content",  "rsa_public_k
    { "id": "kh_2", "role": "...", "controls_key": "K_identity",  "rsa_public_
    { "id": "kh_3", "role": "...", "controls_key": "K_metadata",  "rsa_public_
  ],
  "categories": [ ... ],
  "enrollment_complete": true
}
```

> Invia solo se `enrollment_complete` è `true`: significa che i key-holder (e quindi le chiavi pubbliche) sono configurati. Altrimenti il canale non può ancora ricevere segnalazioni.

5.2 — Schema crittografico

La segnalazione è divisa in **tre compartimenti** cifrati indipendentemente, ciascuno con una chiave AES-256-GCM monouso:

- **content** — il testo della segnalazione (obbligatorio);
- **identity** — identità del segnalante, se fornita (opzionale; può essere `null` per restare anonimo);
- **metadata** — metadati applicativi (opzionale).

Per ogni compartimento:

1. genera una chiave AES-256 e un IV a 12 byte; cifra in **AES-256-GCM** → produce `ciphertext`, `iv`, `authTag` (16 byte), tutti in base64;
2. **incarta** (wrap) la chiave AES con la chiave pubblica RSA del key-holder che controlla quel compartimento, in **RSA-OAEP (SHA-256)** → un elemento in `key_wrappers`.

> **Suite:** `RSA-4096-OAEP(SHA-256)` per incartare le chiavi · `AES-256-GCM` (IV 12 byte, tag 16 byte) per i dati. La chiave privata RSA risiede solo lato gestore: Vocea archivia il ciphertext e non può aprirlo.

5.3 — Pseudocodice (WebCrypto)

```
// 1) chiavi pubbliche del canale
const info = await (await fetch(`${BASE}/api/wb/tenants/${slug}/info`)).json()
const khFor = (t) => info.key_holders.find(k => k.controls_key === t);

// 2) cifra un compartimento in AES-256-GCM + incarta la chiave in RSA-OAEP
async function sealField(plaintext, keyType) {
  const aes = await crypto.subtle.generateKey({name:"AES-GCM",length:256}, tru
  const iv  = crypto.getRandomValues(new Uint8Array(12));
  const ct  = new Uint8Array(await crypto.subtle.encrypt({name:"AES-GCM",iv},
  // GCM: gli ultimi 16 byte sono l'auth tag
  const authTag = ct.slice(ct.length - 16), body = ct.slice(0, ct.length - 16)

  const kh  = khFor(keyType);
  const pub = await importRsaPublicKey(kh.rsa_public_key_pem);              //
  const raw = await crypto.subtle.exportKey("raw", aes);
  const wrapped = await crypto.subtle.encrypt({name:"RSA-OAEP"}, pub, raw); //

  return {
    field:   { ciphertext: b64(body), iv: b64(iv), authTag: b64(authTag) },
    wrapper: { keyholder_id: kh.id, key_type: keyType, encrypted_aes_key: b64(
  };
}

const content  = await sealField(reportText, "K_content");
const identity = anonymous ? null : await sealField(JSON.stringify(id), "K_ide
const metadata = await sealField(JSON.stringify(meta), "K_metadata");
```

6. Endpoint /v1

Base URL: `https://api.vocea.cloud/v1` (dominio dedicato) — in alternativa, via gateway applicativo `https://vocea.cloud/api/wb/v1`. Tutte richiedono `X-API-Key`.

POST /integrations/reports/submit `reports:submit`

Invia una segnalazione già cifrata (vedi §5). Restituisce il **codice di tracking** — l'unico modo per seguire la segnalazione: consegnalo al segnalante.

Body (campi `*_ciphertext/_iv/_auth_tag` e `encrypted_aes_key` in base64):

```
{
  "content_ciphertext":  "BASE64", "content_iv":  "BASE64", "content_auth_tag"
  "identity_ciphertext": "BASE64|null", "identity_iv": "BASE64|null", "identit
  "metadata_ciphertext": "BASE64", "metadata_iv": "BASE64", "metadata_auth_tag
  "key_wrappers": [
    { "keyholder_id": "kh_1", "key_type": "K_content",  "encrypted_aes_key": "
    { "keyholder_id": "kh_2", "key_type": "K_identity", "encrypted_aes_key": "
    { "keyholder_id": "kh_3", "key_type": "K_metadata", "encrypted_aes_key": "
  ],
  "category": "corruption",
  "subcategory_d231": null,
  "submission_channel": "API"
}
```

curl:

```
curl -sS -X POST https://api.vocea.cloud/v1/integrations/reports/submit \
  -H "X-API-Key: wbk_<prefix>_<secret>" \
  -H "Content-Type: application/json" \
  -d @report.json
```

201 Created:

```
{
  "success": true,
  "anonymous_token": "WB-7F3K-9QA2",
  "tenant_slug": "acme",
  "message": "Segnalazione ricevuta e cifrata. Conserva il codice per verifica
}
```

GET /integrations/reports/{code}/status `status:read`

Stato di una segnalazione tramite codice di tracking. **Solo metadati**: nessun contenuto, nessun ciphertext.

```
curl -sS https://api.vocea.cloud/v1/integrations/reports/WB-7F3K-9QA2/status \
  -H "X-API-Key: wbk_<prefix>_<secret>"
```

```
{
  "code": "WB-7F3K-9QA2",
  "status": "under_review",
  "created_at": "2026-06-03T10:12:45Z",
  "last_update_at": "2026-06-05T08:30:00Z",
  "deadlines": {
    "acknowledgment_deadline_at": "2026-06-10T10:12:45Z",
    "investigation_deadline_at": "2026-09-03T10:12:45Z"
  }
}
```

Le scadenze riflettono gli obblighi del D.Lgs. 24/2023: avviso di ricevimento entro 7 giorni, riscontro entro 3 mesi.

GET /integrations/reports `reports:read`

Elenco paginato dei metadati del canale. Query opzionali: `page` (default 1), `pageSize` (1–100, default 20), `status`. Gli item includono metadati e ciphertext, **mai** testo in chiaro: la decifratura avviene solo lato gestore.

```
curl -sS "https://api.vocea.cloud/v1/integrations/reports?page=1&pageSize=20&s
  -H "X-API-Key: wbk_<prefix>_<secret>"
```

```
{ "items": [ /* metadati + ciphertext */ ], "page": 1, "pageSize": 20, "total"
```

Gestione del proprio canale (self-service)

Ogni applicativo amministra la **propria** organizzazione: key holder e deleghe del proprio canale. Richiede gli scope dedicati; isolato per-tenant.

POST /integrations/keyholders `keyholders:manage`

Enrolla un key holder. La coppia RSA si genera **lato client**: si invia **solo** la chiave pubblica (zero-knowledge). Un solo key holder attivo per `controls_key`.

```
POST /v1/integrations/keyholders
{
  "name": "Mario Rossi", "email": "odv@azienda.it",
```

```
  "role": "ODV_MEMBER",            // ODV_MEMBER | LEGAL_RPD | COMPLIANCE | EX
  "controls_key": "K_content",     // K_content | K_identity | K_metadata
  "rsa_public_key_pem": "-----BEGIN PUBLIC KEY-----\n...",
  "password": "opz. — per il portale key holder; se assente è generata"
}
→ 201 { "success": true, "id": 4, "controls_key": "K_content", "role": "ODV_ME
```

`GET` `/integrations/keyholders` elenca · `DELETE` `/integrations/keyholders/{id}` disattiva. Errori: `409 KEY_TYPE_TAKEN` (tipo già occupato → disattiva prima), `409 EMAIL_TAKEN`, `400 INVALID_PUBLIC_KEY`.

`POST` /integrations/delegates `delegates:manage`

Invita un delegato/gestore del canale (ritorna un token di invito).

```
POST /v1/integrations/delegates
{ "email": "gestore@azienda.it", "name": "Anna Bianchi", "role": "GESTORE" }
→ 201 { "success": true, "id": 1, "invite_token": "wbi_…" }
```

`GET` `/integrations/delegates` elenca le deleghe del canale.

7. Codici di errore

Tutti gli errori hanno forma `{ "error": { "code": "...", "message": "..." } }`.

HTTP	code	Significato
400	`MISSING_FIELDS` / `MISSING_KEY_WRAPPERS` / `INVALID_CATEGORY`	Body malformato o incompleto.
401	`API_KEY_REQUIRED`	Header `X-API-Key` assente.
401	`API_KEY_INVALID`	Chiave non valida o revocata.

403	`INSUFFICIENT_SCOPE`	La chiave non ha lo scope richiesto dall'endpoint.
403	`TENANT_SUSPENDED`	Canale sospeso.
404	`TENANT_NOT_FOUND`	Canale non disponibile (eliminato / in provisioning).
404	`REPORT_NOT_FOUND`	Nessuna segnalazione per il codice indicato in questo canale.
429	`RATE_LIMIT_EXCEEDED`	Quota per-chiave superata (vedi §8).

8. Rate limiting

La quota è applicata **per chiave** (non per IP): un client legittimo dietro NAT non viene penalizzato, e una chiave abusiva non aggira il limite ruotando IP. Default **60 richieste/minuto**.

Ogni risposta porta gli header standard `RateLimit-Limit`, `RateLimit-Remaining`, `RateLimit-Reset`. Su `429`, **ritenta con backoff esponenziale** rispettando `RateLimit-Reset`.

9. Hardening per integrazioni B2B

- **TLS sempre:** chiama esclusivamente in HTTPS (TLS 1.2+). Verifica il certificato del server; non disabilitare la verifica.
- **IP allowlist** (opzionale, per-chiave): si può vincolare una chiave a una lista di CIDR. Consigliato per integrazioni con IP di uscita stabili.
- **mTLS** (opzionale, enterprise): il vhost `api.vocea.cloud` può richiedere certificato client per i partner che lo necessitano.
- **Niente CORS da browser:** l'API è server-to-server. Non chiamarla da JavaScript di pagina: esporresti la chiave.

- **Rotazione:** ruota le chiavi periodicamente e dopo ogni sospetto incidente; revoca subito le chiavi inutilizzate.
- **Minimo contenuto sensibile in chiaro lato tuo:** cifra il prima possibile; non loggare ciphertext né payload.

10. Checklist sicurezza per l'integratore

- ✓ Chiave ottenuta dalla console gestore con i **soli scope necessari**.
- ✓ Chiave conservata in un **vault/secret manager**, mai in git/env committate.
- ✓ Chiamate **solo server-side**, sempre in HTTPS con verifica del certificato.
- ✓ Payload **cifrato client-side** (AES-256-GCM + RSA-OAEP) prima del submit; nessun plaintext inviato.
- ✓ Chiavi pubbliche del canale recuperate da `/tenants/{slug}/info` e usate per ogni compartimento.
- ✓ Codice di tracking `anonymous_token` consegnato al segnalante e **non** archiviato in chiaro insieme a dati identificativi.
- ✓ Gestione di `429` con **backoff**; gestione esplicita di `401/403`.
- ✓ Procedura di **rotazione/revoca** chiavi documentata e testata.

Vocea — piattaforma di whistleblowing conforme al D.Lgs. 24/2023 e al GDPR. Erogata da Agile Technology S.r.l. (P.IVA/CF IT07776161213).

Specifica macchina-leggibile: OpenAPI 3 (openapi.v1.yaml) · Versione stampabile: Specifiche API (PDF) · Sito: vocea.cloud · Supporto integrazioni: tramite la console gestore del canale.

Confine zero-knowledge: l'API espone metadati e ciphertext, mai contenuto decifrato. Vocea non possiede le chiavi private dei canali.