INTEGRATION GUIDE

Veris Protocol

Cryptographically signed identity, real-time capability enforcement, and tamper-evident audit for autonomous AI agents.

Base URL https://verisaegis.com/api/public/aegis

SDK npm install @verisaegis/sdk

Source github.com/Afrik-bot/veris-protocol · MIT

Support dev@verisaegis.com

Overview

Veris adds three things to an AI agent workflow. Your existing agent code does not change — Veris wraps the actions that carry consequences.

01
Issue
One call at deployment. The agent receives a signed passport declaring exactly what it may and may not do.

02
Verify
One call before each sensitive action. Scope checked, trust scored, policies applied.

03
Decide
ALLOW proceeds with a scoped token. DENY blocks. REVIEW escalates to a human.

04
Prove
Every decision written to a hash-chained audit log automatically. No code required.

Contents

Integration effort

Under one day of engineering, once. You wire Veris into your deployment pipeline a single time; every agent you deploy afterwards receives a passport automatically. Integration cost does not scale with the number of agents.

01 Before you start

Get an API key

Create a key in the Veris Console under **Settings → API Keys**, or email **dev@verisaegis.com** and we will provision one for your organisation. Keys are shown once at creation and stored only as a SHA-256 hash on our side.

```
# Store as an environment variable. Never commit to source control.
export AEGIS_API_KEY=aegis_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

> **A note on naming**
>
> Route paths, the `aegis_` key prefix, and the `ap_` passport prefix retain the protocol's original codename. They are frozen for backwards compatibility and will not change.

Install the SDK

```
npm install @verisaegis/sdk
```

Node 18 or later. Zero runtime dependencies. Python and other languages can call the REST API directly — every example in this guide includes a raw HTTP form.

```
import { AegisClient } from '@verisaegis/sdk';

const veris = new AegisClient({ apiKey: process.env.AEGIS_API_KEY });
```

Endpoints at a glance

Method	Endpoint	Purpose
POST	`/issue`	Issue a signed AI Passport
POST	`/verify`	Trust decision before an action
POST	`/revoke`	Revoke a passport immediately
POST	`/attest`	Submit behavioural evidence
GET	`/audit`	Retrieve the audit log
GET	`/passports`	List your organisation's passports
GET	`/.well-known/jwks.json`	Public signing keys — **no auth**
POST	`/verify-passport`	Verify a passport offline — **no auth**

02 Issue an AI Passport

Do this once per agent type, at deployment — not per request. The passport defines the agent's identity and the exact boundaries of what it may do.

SDK

```
const { passport, passportJwt } = await veris.passports.issue({
  agentName:    'CustomerSupportAgent',
  agentVersion: '2.1.0',
  modelFamily:  'claude-sonnet',
  deploymentEnv: 'production',
  jurisdiction: 'US',

  // What this agent CAN do
  permittedActions: [
    'read:customer_records',
    'send:support_response',
  ],

  // What it can NEVER do — explicit denials always win
  deniedActions: [
    'write:billing_records',
    'delete:customer_data',
  ],

  // Actions that force a REVIEW decision
  requiresHumanApproval: ['process:refund'],
});

// Store passport.passport_id — you will use it on every verify call
console.log(passport.passport_id);   // ap_oq5ZqfWWKGtpdFdkbM6c24yAHF
```

HTTP

```
curl -X POST "https://verisaegis.com/api/public/aegis/issue" \
  -H "Authorization: Bearer $AEGIS_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "agentName": "CustomerSupportAgent",
    "deploymentEnv": "production",
    "permittedActions": ["read:customer_records","send:support_response"],
    "deniedActions": ["write:billing_records","delete:customer_data"],
    "requiresHumanApproval": ["process:refund"]
  }'
```

Response

```
{
  "passport": {
    "passport_id":   "ap_oq5ZqfWWKGtpdFdkbM6c24yAHF",
    "agent_id":      "agt_18167b8c3c5252aeb0eb852350861d4e",
    "agent_name":    "CustomerSupportAgent",
    "status":        "ACTIVE",
    "trust_score":   850,
    "trust_tier":    "TRUSTED",
    "permitted_actions": ["read:customer_records","send:support_response"],
    "denied_actions":    ["write:billing_records","delete:customer_data"],
    "issued_at":     "2026-08-03T21:23:30Z",
    "expires_at":    "2027-08-03T21:23:30Z",
    "signing_key_id": "kid_937ce495367ebd11",
    "passport_jwt":  "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6..."
  },
  "passportJwt": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6..."
}
```

Capability format

Capabilities use `verb:resource` — for example `read:customer_records`. Wildcards are supported in the verb position (`delete:*` matches every deletion capability).

> **Precedence rule**
>
> Explicit denials always take precedence over any grant, including wildcards. An agent permitted `write:*` but denied `write:billing_records` will always be denied billing writes. This rule is absolute and cannot be overridden by any policy.

03 Verify before sensitive actions

This is the one call you add to existing agent code. Place it before any action that touches customer data or carries downstream consequences.

```
async function handleCustomerQuery(passportId, customerId) {
  const decision = await veris.verify({
    passportId,
    requestedCapabilities: ['read:customer_records'],
    riskLevel:     'MEDIUM',
    relyingParty:  'crm-system',
  });

  if (decision.decision === 'ALLOW') {
    // Proceed. decision.capabilityToken is scoped to this interaction.
    return crm.getCustomer(customerId);

  } else if (decision.decision === 'REVIEW') {
    // Human approval required, or trust too low for this risk level
    return escalateToHuman(decision.warnings, decision.auditReceiptId);

  } else {
    // DENY — blocked before the action occurred, and logged
    throw new Error(`Access denied: ${decision.warnings.join(', ')}`);
  }
}
```

Two real responses

Captured from the live API. Same agent, two different requests.

ALLOW

`read:customer_records`

Trust score **910** · TRUSTED
Capability token issued

DENY

`write:billing_records`

Blocked before execution
No token · logged at HIGH severity


```
{
  "decision":       "ALLOW",
  "trustScore":     910,
  "trustTier":      "TRUSTED",
  "validUntil":     "2026-08-03T21:38:11Z",
  "capabilityToken":"captoken.eyJwYXNzcG9ydElkIjoi...",
  "auditReceiptId": "ar_lbe8qgvfJn6D5Xdjt7HoRf0f",
  "warnings":       [],
  "scoreComponents": {
    "identity": 900, "behavioral": 960, "compliance": 1000,
    "historical": 700, "environmental": 950
  },
  "matchedPolicies": [],
  "trustModel": {
    "weights": { "identity": 0.30, "behavioral": 0.25, "compliance": 0.20,
                 "historical": 0.15, "environmental": 0.10 },
    "thresholds": { "trusted": 850, "elevated": 650, "standard": 400, "low": 200 },
    "isDefault": true
  },
  "latencyMs": 34
}
```

```
{
  "decision":       "DENY",
  "trustScore":     0,
  "trustTier":      "CRITICAL",
  "capabilityToken": null,
  "auditReceiptId": "ar_kcSBcykU8VpnZOm7HgFJ0oPZ",
  "warnings":       ["capability_denied:write:billing_records"],
  "scoreComponents": null
}
```


Risk levels

Risk level	Effect on the decision
`LOW`	Standard thresholds apply
`MEDIUM`	Default. Slight environmental score penalty
`HIGH`	Requires a TRUSTED-tier score, otherwise REVIEW
`CRITICAL`	Always REVIEW — human approval required regardless of score

04 Signed passports & offline verification

Every passport is issued as an ES256-signed JWT. Anyone can verify it offline, without calling our API and without trusting our database.

Structure

Standard JWS compact serialisation — `header.payload.signature`.

```
// Header
{ "alg": "ES256", "typ": "JWT", "kid": "kid_937ce495367ebd11" }

// Payload
{
  "iss": "https://verisaegis.com",
  "sub": "ap_oq5ZqfWWKGtpdFdkbM6c24yAHF",
  "aud": "veris-relying-party",
  "iat": 1785792210,
  "exp": 1817328210,
  "veris": {
    "version":       "1.0",
    "agentId":       "agt_18167b8c3c5252aeb0eb852350861d4e",
    "agentName":     "CustomerSupportAgent",
    "orgId":         "org_HD8v04mU8gLJ",
    "deploymentEnv": "production",
    "jurisdiction":  "US",
    "capabilities": {
      "permitted": ["read:customer_records","send:support_response"],
      "denied":    ["write:billing_records","delete:customer_data"],
      "requiresHumanApproval": ["process:refund"]
    },
    "behavioralBaselineHash": "54f9ad691b90de49f6819a7bed2be5c565d738d3...",
    "policyFramework": "NIST_AI_RMF_1.0",
    "initialTrustScore": 850,
    "initialTrustTier":  "TRUSTED"
  }
}
```

Verify it yourself

Using any standard JOSE library — no Veris SDK required:

```
import { jwtVerify, createRemoteJWKSet } from 'jose';

const JWKS = createRemoteJWKSet(new URL(
  'https://verisaegis.com/api/public/aegis/.well-known/jwks.json'
));

const { payload } = await jwtVerify(passportJwt, JWKS, {
  issuer: 'https://verisaegis.com',
});

console.log(payload.veris.capabilities.permitted);
```

The public key set

Unauthenticated. Retired keys remain published so passports issued before a key rotation stay verifiable.

```
GET /api/public/aegis/.well-known/jwks.json

{ "keys": [{
    "kty": "EC", "crv": "P-256", "use": "sig", "alg": "ES256",
    "kid": "kid_937ce495367ebd11",
    "x": "LHC4xNVFIjO6iCpkURL-tAQD_CyWnIAwHZsy_9eOz7c",
    "y": "a3Zg7DBl_4joZA7w2tJzs5Lv6gc9SS61bbwGdjw2sMY"
}] }
```

What signature verification does and does not prove

A valid signature proves the passport was issued by Veris and has not been altered. It does **not** check revocation status or current trust score.

For a live decision including revocation, dynamic trust, and policy evaluation, call `POST /verify`. Use offline verification when you need to confirm provenance and scope without a network round trip.

05 Policies

Organisations can define rules evaluated on every verification — enforcing standards that apply across all agents, independent of any individual passport.

> **Policies restrict. They never grant.**
>
> There is no ALLOW effect. A policy can deny an action, force human review, or raise the required trust score. It can never grant a capability an agent's passport does not already carry, and it can never override an explicit denial. The capability scope check always runs first and is absolute.

Effects

Effect	Behaviour
`DENY`	Blocks the action outright
`REVIEW`	Forces a REVIEW decision requiring human approval
`REQUIRE_SCORE`	Raises the minimum trust score for matching actions

Conditions

All conditions are optional and combined with AND. An empty condition matches anything.

Condition	Matches on
Capabilities	Requested capabilities, with `verb:*` wildcard support
Risk levels	LOW / MEDIUM / HIGH / CRITICAL
Deployment environments	production / staging / development
Jurisdictions	US / EU / UK / APAC / GLOBAL
Agent name	Case-insensitive substring match
Relying parties	The calling system identifier
Time window	UTC hours and days of week

Example

Policy: Financial actions require high trust
Effect: REQUIRE_SCORE 950 · **Capabilities:** `write:billing_records` , `process:refund`
An agent scoring 910 that is otherwise permitted the capability now receives:


```
{
  "decision":   "REVIEW",
  "trustScore": 910,
  "warnings":   ["policy_min_score:Financial actions require high trust:950"],
  "matchedPolicies": [
    { "policyId": "pol_finhigh01",
      "name":     "Financial actions require high trust",
      "effect":   "REQUIRE_SCORE" }
  ]
}
```

Policies are configured in the Veris Console under **Policies**. Every create, update, and deletion is written to the audit log.

06 Trust evidence & scoring

Veris standardises the trust evidence. The scoring model is a reference implementation you can replace.

The five signal components

Standardised by the protocol and returned raw with every decision:

Component	What it measures
`identity`	Certificate validity, key storage, rotation age, revocation freshness
`behavioral`	Deviation from the declared behavioural baseline
`compliance`	Policy checkpoint adherence, human-approval compliance
`historical`	Time-decayed incident and attestation history (90-day half-life)
`environmental`	Deployment context, jurisdiction, request risk level

Reference scoring model

```
T(a) = 0.30·identity + 0.25·behavioral + 0.20·compliance
     + 0.15·historical + 0.10·environmental

TRUSTED ≥ 850   ELEVATED ≥ 650   STANDARD ≥ 400   LOW ≥ 200   CRITICAL < 200
```

> **This is a default, not a requirement**
>
> Protocols that achieve broad adoption standardise formats, identities, signatures, and verification methods — not reputation or risk models. TLS specifies how a certificate is structured and verified; it does not tell a browser which authorities to trust.
>
> Veris follows that pattern. If your organisation already has a risk engine, ignore the composite and apply your own model to the raw `scoreComponents`. If you prefer to use ours with different weights, configure them in the Console — the weights in effect are returned in the `trustModel` field of every response.

Standardised by the protocol	Chosen by your organisation
The five components and how each is measured	The weight of each signal
Signed passport format and verification	Tier thresholds
Capability scope model and precedence	Policies layered on top
Audit event format and hash chain	Whether to use the composite at all
Request and response contract	

07 Audit trail

Every issuance, decision, revocation, and policy change is written to an append-only, hash-chained log — automatically, with no code on your side.

```
const log = await veris.audit.getLog(passportId, { limit: 100 });

log.events.forEach(e =>
  console.log(`[${e.severity}] ${e.event_type} — ${e.occurred_at}`)
);
```

How tamper-evidence works

Each entry stores a hash of its own contents plus the hash of the entry before it. Altering any historical record breaks its link to every subsequent record, and the break is detectable by walking the chain.

```
seq 44   PASSPORT_ISSUED       prev 62d8a26e…   row 3b53773d…
seq 45   VERIFICATION_ALLOW    prev 3b53773d…   row 230689fb…
seq 46   VERIFICATION_DENY     prev 230689fb…   row 4376f9d3…
             ↑                      ↑
       each prev_hash equals the previous entry's row_hash
```

Database rules additionally reject `UPDATE` and `DELETE` on audit records, so modification fails rather than merely being detected.

> **What this gives your customers**
>
> When a client asks "*what did your AI agent do with our data, and was it authorised?*" — the answer is a structured, cryptographically verifiable export covering every access, every decision, and every trust score at the moment of decision. Production-ready compliance evidence, not application logs reconstructed after the fact.

08 Revocation & attestations

Revocation — the kill switch

```
await veris.passports.revoke({
  passportId,
  reason: 'SECURITY_INCIDENT',
  notes:  'Anomalous access pattern detected at 14:23 UTC',
});
```

All subsequent verify calls for that passport return DENY. Revocation propagates in under 60 seconds and is written to the audit log at HIGH severity.

Reasons: `SECURITY_INCIDENT` · `POLICY_VIOLATION` · `SCOPE_CHANGE` · `DECOMMISSIONED` · `ORGANIZATION_OFFBOARDING` · `OTHER`

Attestations — feeding the trust score

Submit evidence about an agent's behaviour. Positive attestations raise the historical component over time; incidents lower it sharply with a 90-day decay.

```
await veris.attest({
  passportId,
  attestationType: 'POLICY_CHECKPOINT',
  severity: 'INFO',
  evidence: {
    checkpoint: 'quarterly_access_review',
    result:     'PASSED',
    reviewedBy: 'ComplianceSystem-v1',
  },
});
```

Attestation type	Score impact
`POLICY_CHECKPOINT`	+15
`BEHAVIORAL_SAMPLE`	+10
`CAPABILITY_EXERCISE`	+8
`HUMAN_APPROVAL`	+5
`INCIDENT_REPORT`	−150

09 Deployment patterns

Healthcare — patient record access

```
const decision = await veris.verify({
  passportId,
  requestedCapabilities: ['read:patient_records'],
  riskLevel:       'HIGH',
  interactionType: 'phi_data_access',
  relyingParty:    'ehr-system',
});
```

Pair with a policy requiring TRUSTED tier for any PHI capability, and an organisation policy forcing REVIEW on `delete:*`. The audit export becomes the evidence trail for access reviews.

Financial services — regulated data and transactions

```
const decision = await veris.verify({
  passportId,
  requestedCapabilities: ['read:market_data', 'write:research_reports'],
  riskLevel:       'HIGH',
  interactionType: 'regulated_data_access',
});
```

Transaction capabilities belong in `deniedActions` or `requiresHumanApproval`, never in the permitted set for an autonomous agent. A REQUIRE_SCORE policy on financial capabilities adds a second layer.

Workflow automation — document processing

```
const decision = await veris.verify({
  passportId,
  requestedCapabilities: ['read:contract_documents','extract:contract_data'],
  riskLevel: 'MEDIUM',
  context:   { documentClassification: 'CONFIDENTIAL' },
});
```

Set `jurisdiction: 'EU'` at issuance for data-residency-sensitive deployments, and deny `send:external_api` to prevent cross-border transfer without human approval.

10 Field reference

Issue request

Field	Required	Description
`agentName`	Yes	Human-readable agent name
`permittedActions`	Yes	Array of allowed capabilities
`agentVersion`	No	Default `1.0.0`
`modelFamily`	No	e.g. `claude-sonnet` , `gpt-4o`
`deploymentEnv`	No	production / staging / development
`jurisdiction`	No	Default `US`
`deniedActions`	No	Explicitly forbidden — always wins
`requiresHumanApproval`	No	Forces a REVIEW decision

Verify request

Field	Required	Description
`passportId`	Yes	The `ap_…` passport identifier
`requestedCapabilities`	Yes	Actions the agent wants to perform
`riskLevel`	No	LOW / MEDIUM / HIGH / CRITICAL (default MEDIUM)
`interactionType`	No	Free-text label for the interaction
`relyingParty`	No	Identifier of the calling system
`context`	No	Arbitrary metadata recorded in the audit event

Verify response

Field	Description
`decision`	ALLOW / DENY / REVIEW
`trustScore` · `trustTier`	Composite under the model in effect
`scoreComponents`	The five raw signal values
`trustModel`	Weights and thresholds used; whether they are defaults
`matchedPolicies`	Organisation policies that fired
`capabilityToken`	Scoped token, issued on ALLOW only
`auditReceiptId`	Reference to the audit record for this decision
`warnings`	Reasons and advisories
`latencyMs`	Server-side processing time

Errors

Status	Meaning
400	Missing or malformed required field
401	Missing, invalid, or revoked API key
403	Organisation suspended or inactive
404	Passport not found within your organisation
409	Conflict — for example, revoking an already-revoked passport

11 Security posture

We state our position openly, including what is not yet hardened. The full threat model is published in the public repository.

Live today

- ES256-signed passports, verifiable offline against a public JWKS
- Hash-chained, append-only audit log with database-enforced immutability
- API keys stored only as SHA-256 hashes; immediate revocation
- Capability scope enforcement with absolute deny precedence
- Organisation policy engine that can only restrict, never grant
- Fail-closed evaluation — ambiguity resolves to REVIEW or DENY, never ALLOW

On the near-term roadmap

- HSM-backed signing keys — replacing environment-variable key storage
- Independent penetration test
- SOC 2 Type II
- Production latency optimisation toward the sub-50 ms p99 target

> **What this means for your deployment**
>
> Veris is ready today for pilots and non-regulated production data. We do not recommend routing regulated data — PHI, payment data, or material non-public information — through the platform until HSM-backed signing and an independent penetration test are complete. Design partners are notified as each item lands.

Threat model: `github.com/Afrik-bot/veris-protocol/blob/main/docs/THREAT_MODEL.md`
Security disclosures: **security@verisaegis.com** — please do not open public issues for vulnerabilities.

12 Design partner programme

A scoped, fixed-price engagement to prove value against one real workflow — without a procurement cycle.

Scope

Phase	Activity
Week 1	Scoping and integration — under a day of your engineers' time, with our direct support
Weeks 2–7	Running in your environment. Weekly check-ins, priority support, direct roadmap input
Week 8	Executive readout: findings, risks, agent inventory, and recommendations

What you receive

- Agent inventory across the target workflow
- AI Passports issued for every agent in scope
- Runtime authorisation live in your environment
- Organisation policies configured to your rules
- Audit dashboard with exportable compliance evidence
- Direct influence over the protocol roadmap
- Co-developed case study, subject to your approval

> **Commercial terms**
>
> A 60-day pilot is **$25,000**, credited in full toward a first-year contract should you continue. Ongoing pricing is usage-based and scales with verification volume rather than seat count.

Getting started

Email **dev@verisaegis.com** and we will provision an API key the same day. Your engineers can issue a passport and run a live verification in about ten minutes using the examples in this guide.

Website	verisaegis.com
Repository	github.com/Afrik-bot/veris-protocol · MIT licensed
SDK	npm install @verisaegis/sdk
Integration support	dev@verisaegis.com
Security disclosures	security@verisaegis.com