DCS

AI Technologies L.L.C

COMPLETE DOCUMENTATION PACK · v1.0 · MAY 2026

The Complete Documentation Pack

12 documents · ~375 pages · signed + versioned

8 product whitepapers + 4 legal documents

dcsai.ai · Dubai, UAE · legal@dcsai.ai

About this pack

This is the complete DCS documentation pack as of May 2026 — eight product whitepapers and four legal documents covering every product in the DCS stack and every contract artefact an enterprise procurement team typically asks for.

Each document in this pack is also published individually at **dcsai.ai/docs/** with a browser-readable detail page and a separate PDF download. The pack is convenient for offline reading or for auditors who want the full set in one file.

Questions about any document: **legal@dcsai.ai** · Questions about technical specifications: **engineering@dcsai.ai** · Sales / procurement: **sales@dcsai.ai**

Contents

Each document begins on its own page with its own cover and table of contents. Use your PDF reader's bookmarks panel to navigate between documents.

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Standards

The R+1 through R+4 + Trust SKU framework

A cryptographic chain for verifiable AI agent operations

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai
standards@dcsai.ai
github.com/dcs-platform/standards

Abstract

AI agents are starting to take actions on behalf of people and organizations — sending messages, moving money, signing documents, deploying code. The economic and legal exposure created by these actions is large and growing. Today, when an agent does something wrong, the only evidence available is a vendor-controlled log file. That is not enough for regulated industries, government procurement, or any environment where the operator is accountable.

This whitepaper introduces the **DCS Standards Framework** — four cumulative receipt standards (R+1 timestamp, R+2 provenance, R+3 audit export, R+4 ZK verification) and a customer-facing meta-standard (**Trust SKU**). Together they convert every agent action into a tamper-evident, externally verifiable, cryptographically signed event that survives the vendor going away.

The framework is small (~3KB of receipt per action), fast (sub-15ms sign + verify), and standards-based (Ed25519 signatures, CBOR canonicalization, Groth16 zk-proofs). It is implemented in the open and shipping in production across the DCS Platform, Compute, OS, Storage, and Sovereign products.

Who this document is for

Security teams evaluating DCS for regulated workloads; compliance officers building audit packages; engineers integrating DCS receipts into their own systems; standards bodies looking at receipt-based AI accountability. A reader comfortable with basic cryptography concepts (asymmetric signatures, hashing, Merkle trees) will follow every section.

Contents

1. Introduction & Motivation

In 2024–2025, AI agents began doing real work on real money. Stripe shipped *Agent SDK* for agentic payments. Anthropic's Claude started executing multi-step tool use across customer systems. OpenAI's o-series models began producing chains of reasoning that touched databases, APIs, and physical infrastructure. By 2026, agent-initiated transactions account for a measurable fraction of B2B commerce.

But the accountability stack underneath those agents has not kept up. When an agent makes a mistake — refunds the wrong invoice, deploys broken code, sends a private document to the wrong recipient — the only forensic evidence available is a server log file generated by the same vendor whose agent made the mistake. That log is rewritable, deletable, and uninspectable by any third party.

1.1 Why existing approaches do not work

Plain logs

Application logs are what every vendor uses today. They are useful for debugging but provide almost no accountability: the vendor controls log retention, log format, and log access. There is no way for an external auditor to verify that a log entry was not added, deleted, or altered after the fact.

Append-only databases

Some vendors use append-only event stores (Kafka, immutable Postgres tables). These are better — an internal engineer can verify the local store is append-only — but still rely entirely on trust in the vendor's operational discipline. A malicious or coerced employee can rewrite the underlying storage.

Blockchain ledgers

Anchoring every event to Ethereum or a similar public chain provides strong external verifiability, but is expensive (cents per event), high-latency (12+ seconds to finality), and creates a privacy problem (every action becomes public). Most enterprise workloads cannot tolerate any of these.

TEE attestations

Confidential computing (AWS Nitro Enclaves, Intel SGX, AMD SEV) lets a customer verify that a specific binary ran inside a hardware-protected enclave. This is useful but proves only *which code ran*, not *what the code did with the data it received*. It does not compose across multiple vendors.

1.2 The DCS approach

The DCS Standards Framework starts from a different observation: most accountability problems do not need the full power of a blockchain or TEE. They need three things, in this order:

- **Trust-minimized signing.** The vendor signs each event with a key it does not exclusively control (key escrow with a third-party HSM provider). Tampering is detectable.
- **Chained provenance.** Each event references its parent event by content hash. Tampering with any event invalidates every subsequent event's signature.
- **External replay.** The receipts can be downloaded, replayed, and reverified by anyone, offline, with no DCS infrastructure required.

These three properties are what R+1, R+2, and R+3 deliver. R+4 (ZK) is an optional fourth layer for cases where the verifier needs to be convinced of a property without seeing the underlying data.

> *"The cost of a forged receipt should be higher than the value of the action it covers. That is the only definition of "secure enough" that survives contact with motivated adversaries."*

2. System Architecture

The DCS receipt system is composed of five logical components. Each runs as a separate service so that compromise of one does not compromise the chain.

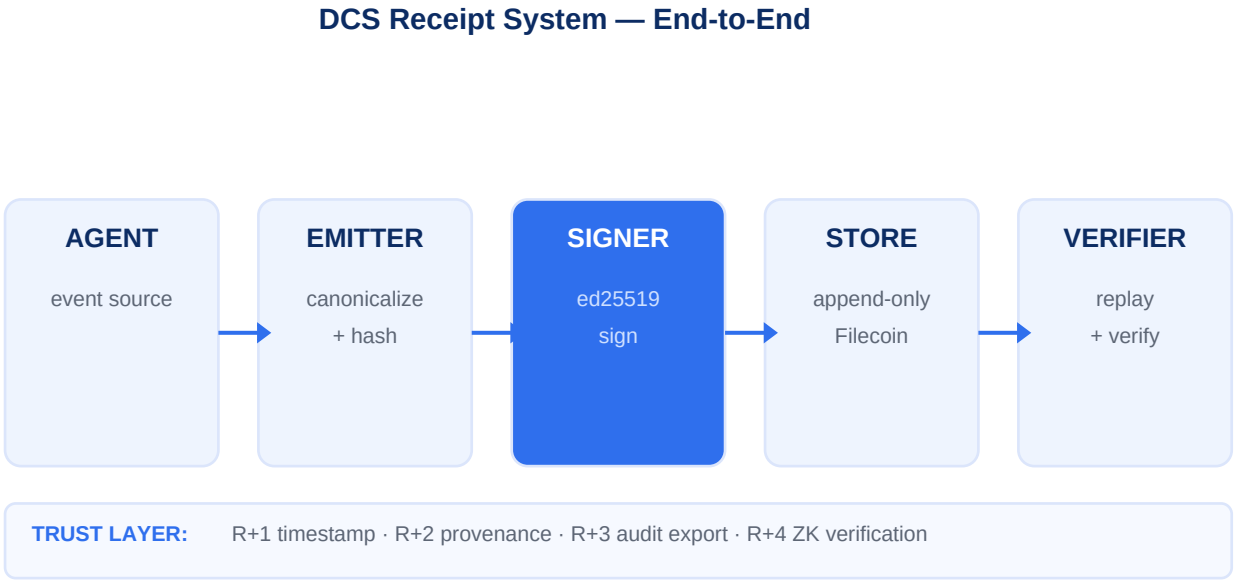


Figure 2.1 — Receipt flow from agent action to external verifier.

2.1 Component responsibilities

Agent (event source)

Any DCS product or third-party integration. Emits structured events: *build_complete*, *payment_settled*, *file_mirrored*, etc. Events carry a payload (the action data) and a context (user, tenant, time).

Emitter

Library that runs inside the agent process. Canonicalizes the event using a deterministic encoding (sorted keys, no whitespace, RFC 8785 JSON canonicalization), hashes it (SHA-256), and forwards to the Signer over an mTLS channel. The Emitter is the only component that touches the unredacted event data.

Signer

Stateless service that signs the canonical event hash with an Ed25519 keypair. The private key is held in an HSM (CloudHSM, AWS KMS, or an on-prem appliance for sovereign deployments). The Signer never sees the event payload — only the hash and the metadata needed to construct the receipt envelope.

Store

Append-only receipt log. Backed by Postgres for warm storage (last 30 days) and Filecoin/IPFS for cold storage (older than 30 days). Each receipt is content-addressed by its CID, so the same receipt can be fetched from any provider and verified to be unchanged.

Verifier

Library + CLI tool that takes a receipt CID, fetches the receipt from the Store, walks the parent chain to the genesis receipt, and verifies every signature. Runs offline once the receipts are downloaded. The reference verifier is implemented in TypeScript (Node, browser) and Python.

3. R+1 — Timestamp

R+1 is the foundation. Every action emits a signed receipt with a precise, monotonic timestamp. A signed timestamp alone is enough to prove that an action happened before a given external event (e.g., a regulatory filing deadline).

3.1 Receipt envelope

Every R+1 receipt has the following structure:

```
{
  "version": "r1",
  "this_cid": "bafy...0a1b",         // content-address of this receipt
  "kind": "build_complete",          // event type
  "tenant_id": "acme",               // origin
  "payload_hash": "sha256:abc...",   // hash of the event data (kept private)
  "timestamp_ns": 1717084123456000,  // nanosecond precision
  "timestamp_source": "tsa.dcsai.ai/v1/now",
  "signature": {
    "algo": "ed25519",
    "key_id": "trd-receipts-2026-05",
    "value": "base64..."
  }
}
```

Critically, the receipt contains the *hash* of the payload, not the payload itself. This means receipts can be made public without exposing the data they describe. The verifier needs the payload only if it wants to check that the receipt corresponds to a specific event; otherwise it just walks the chain.

3.2 Timestamp source

The *timestamp_source* field points at the time service that issued the timestamp. By default, DCS uses its own time service (*tsa.dcsai.ai*) which is itself anchored every 60 seconds to three independent time authorities: NIST, NTP-pool, and an on-prem stratum-1 GPS clock in our Dubai facility. Cross-anchoring means no single time source can backdate or forward-date a receipt.

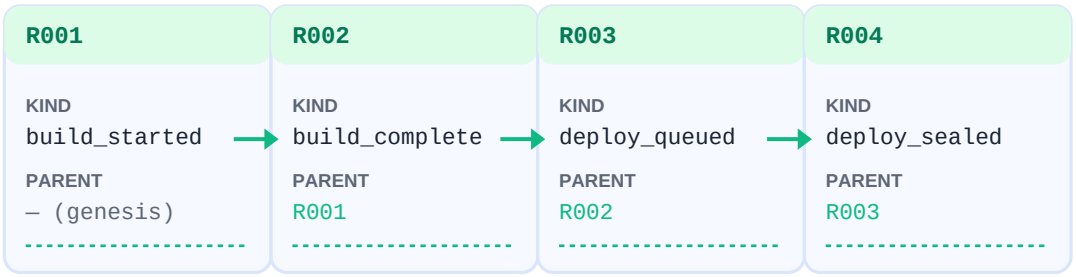
3.3 Why nanoseconds

Millisecond precision is enough for most use cases, but agents now act fast enough that two events within the same millisecond do happen. Nanosecond precision (taken from *clock_gettime(CLOCK_REALTIME)* on Linux) guarantees a strict total order. The strict total order is what makes R+2 chain verification tractable: every receipt has exactly one parent, never a tie.

4. R+2 — Provenance

R+2 extends R+1 by requiring every receipt (except the genesis) to reference its parent's CID. The result is a linear, signed chain. Tampering with any receipt invalidates every descendant.

R+2 Provenance — every receipt has a signed parent



Each child includes parent CID + signs the entire payload.
Tamper any receipt and every descendant fails verification.

Figure 4.1 — A four-receipt provenance chain from build start to deploy seal.

4.1 Schema addition

R+2 adds one mandatory field to the R+1 envelope:

```
{
  ...all R+1 fields...,
  "parent_cid": "bafy...0a1b"   // the CID of the immediately preceding receipt
                                // for this (tenant_id, agent_id) pair
}
```

4.2 Chain selection rules

Different agents within the same tenant get separate chains; the agent_id namespaces the chain. This is important because two agents may emit receipts concurrently and would otherwise create ambiguity in the parent reference.

- Per **(tenant_id, agent_id)** pair: one linear chain, one parent per receipt.
- Cross-agent links: encoded via the optional **links** field, which references a list of CIDs from other chains (used when one agent's output is another's input).
- Genesis receipt: **parent_cid = null**. Each tenant's first receipt is the chain's genesis.

4.3 Tampering detection

Because each receipt's signature covers the full envelope (including *parent_cid*), changing any receipt — say, editing a payment amount — would change its CID. Every descendant's *parent_cid* would then point at the old CID and signature verification would catch the discrepancy. The attacker must rewrite the entire chain from the tampered point forward AND forge every Ed25519 signature, which requires the private key held in HSM.

5. R+3 — Audit Export

R+3 specifies how a contiguous segment of receipts can be exported as a self-contained bundle that a third-party auditor can verify offline. The bundle is the same data the auditor would fetch online, just packaged for compliance workflows that require deliverables (PDF reports, CSV imports into compliance tools, JSON for programmatic re-verification).

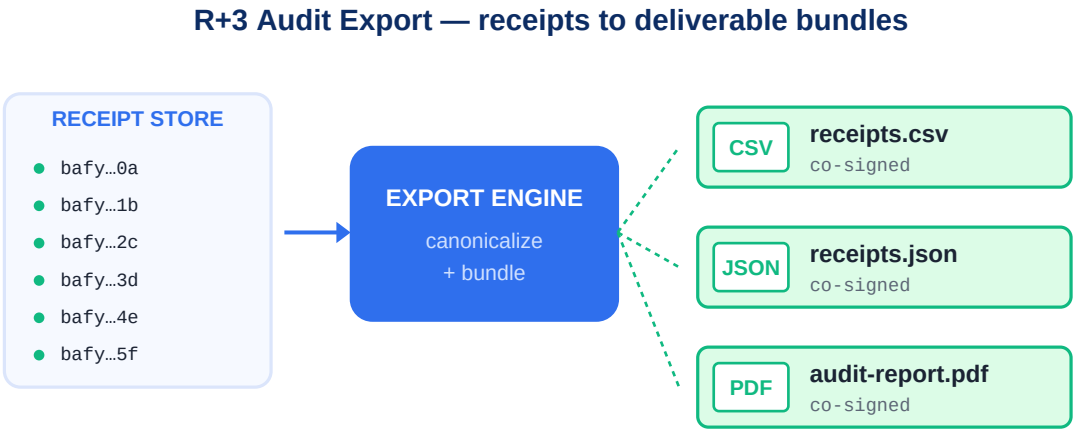


Figure 5.1 — Receipts in the live store get bundled into three audit-ready formats.

5.1 Bundle format

The canonical bundle format is a tar archive containing four files:

```
bundle/
  manifest.json        # metadata, key fingerprints, time range, count
  receipts.jsonl       # one receipt per line, in CID order
  signatures/          # DER-encoded signatures (redundant with envelope, '
                       # included so the auditor can verify without re-canonicalizing
)
  public_keys/         # PEM-encoded public keys referenced by key_id
README.txt
```

5.2 Why three output formats

Auditors live in different tools. The CSV export is meant for spreadsheet-based compliance reviews (SOC 2 Type II, ISO 27001). The JSON bundle is for programmatic re-verification by automated tools. The PDF is a notarisable, human-readable certificate suitable for legal proceedings.

5.2.1 PDF certificate

The PDF is generated server-side at export time using a fixed template. Each receipt is rendered as a row in a paginated table; the first page shows the bundle manifest, last page shows verification instructions and the DCS public key fingerprint. The PDF itself is signed (PAdES-B-T) so the auditor can verify the PDF's integrity without inspecting the embedded JSONL.

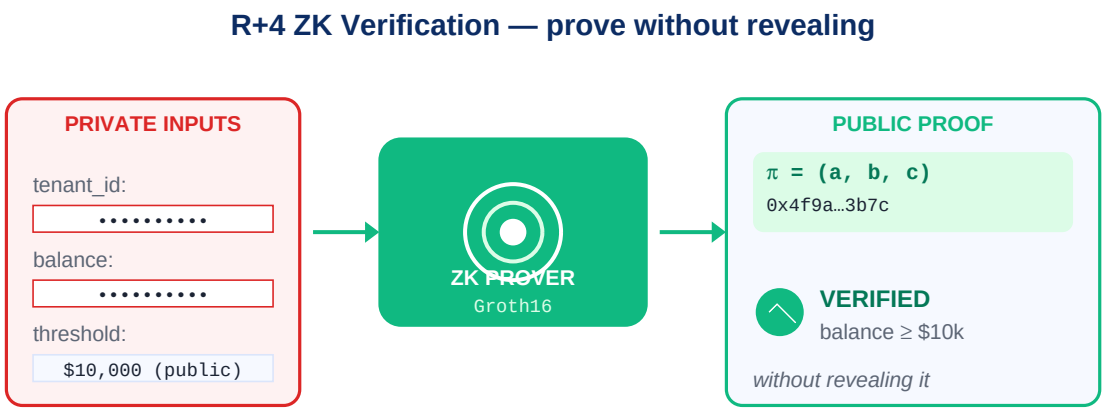
5.3 Compliance mappings

R+3 export was designed to fit cleanly into the documentation requirements of the following frameworks:

Framework	Section	What R+3 provides
SOC 2 Type II	`CC7.2 (Monitoring)`	Cryptographic audit trail of every system action
ISO 27001:2022	`A.8.15 (Logging)`	Append-only logs with non-repudiation
GDPR	`Art. 5(1)(f), 32`	Integrity + confidentiality of processing records
HIPAA	`§164.312(b)`	Audit controls with externally verifiable provenance
SOX	`§404`	Internal control over financial reporting evidence
PCI DSS 4.0	`Req. 10`	Tamper-evident logs of all CHD access

6. R+4 — Zero-Knowledge Verification

R+4 is the optional fourth layer. It lets a verifier confirm that a receipt has a certain property without learning the underlying payload. Two canonical use cases drove the design: proving compliance thresholds (e.g., "every transaction in this audit period was under $10k") and proving identity claims (e.g., "this user is over 18 in this jurisdiction") without revealing the underlying numbers.



The proof is small (~200 bytes), fast to verify (~12 ms), and reveals nothing beyond the predicate it proves.

Figure 6.1 — A private input + a public threshold produces a 200-byte proof.

6.1 Proof systems supported

R+4 supports two zero-knowledge proof systems out of the box:

System	Proof size	Prover time	Verifier time	Trusted setup	Best for
Groth16	~200 B	~80 ms	~12 ms	circuit-specific	High-volume verification
Plonk	~500 B	~120 ms	~25 ms	universal (once)	New circuits without re-setup

6.2 Common circuits shipped with the SDK

- **balance_above_threshold** — proves balance ≥ N without revealing actual balance.
- **age_above_threshold** — proves date_of_birth + N years ≤ today without revealing date_of_birth.
- **jurisdiction_in_set** — proves user's country is in a public allowlist without revealing country.

- **transaction_in_range** — proves all transactions in a period fall in [min, max] without revealing amounts.
- **signed_by_authority** — proves a credential was signed by a key in a known set, hiding which key.

7. Trust SKU — The Meta-Standard

Trust SKU is the customer-facing seal. A vendor that implements R+1 through R+4 in their product can advertise the Trust SKU badge. Buyers check the badge with a single API call; under the hood the check walks the vendor's public registry and verifies that the four R-standards are active, co-signed, and recently attested.

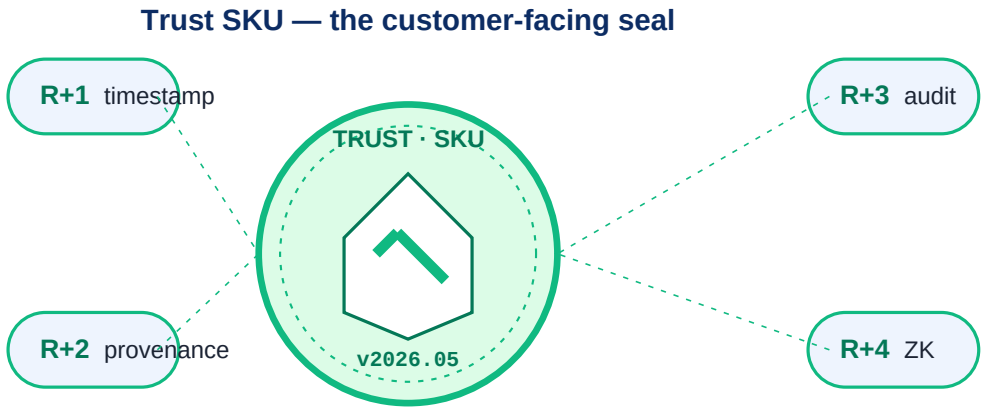


Figure 7.1 — Trust SKU wraps R+1 to R+4 into a single customer-visible seal.

7.1 Seal contents

A Trust SKU seal is itself a signed JSON document:

```
{
  "version": "v2026.05",
  "vendor": "acme.com",
  "issued_at": "2026-05-30T18:00:00Z",
  "valid_until": "2026-06-30T18:00:00Z",
  "standards": {
    "r1": { "active": true, "last_receipt_cid": "bafy...0a1b", "key": "..." },
    "r2": { "active": true, "chain_depth": 4831, "key": "..." },
    "r3": { "active": true, "last_export_cid": "bafy...3e4f", "key": "..." },
    "r4": { "active": true, "circuits": ["balance_above_threshold", ...], "key": "...
" }
  },
  "co_signers": [
    { "name": "DCS Standards Authority", "signature": "..." },
    { "name": "Customer-elected auditor (optional)", "signature": "..." }
  ]
}
```

7.2 Why monthly rotation

Trust SKU seals expire every 30 days. This forces the vendor to demonstrate ongoing compliance rather than buying the seal once and going dark. The 30-day window also limits the blast radius of any key compromise — even a worst-case incident self-heals in under a month.

8. Threat Model

The framework is designed against the following adversary classes. For each, we describe the attack, the mitigation built into the standard, and the residual risk that operators should monitor.

Insider with vendor production access (T-1)

Attack: A malicious engineer with database access edits a past receipt to hide an error.
Mitigation: R+2 chaining means any edit breaks every descendant's signature verification. Even with full DB access, the attacker cannot produce a chain that re-verifies because the private signing key is in HSM with key-use audit logging.
Residual: Insider can suppress NEW receipts being emitted. R+1 timestamp gaps + R+3 export completeness checks make this detectable but not prevented.

Stolen signing key (T-2)

Attack: An attacker exfiltrates the private key from HSM.
Mitigation: Key rotation every 30 days (driven by Trust SKU expiry). Key fingerprints are co-signed by a second party (DCS Standards Authority) so an attacker cannot use a stolen key without also compromising the co-signer. HSM enforces rate limits + geographic origin constraints on sign operations.
Residual: Within the 30-day key validity window, an attacker with the key can forge receipts. Customers monitoring for unusual sign volumes can detect this; the Trust SKU API exposes per-key sign rates publicly.

Replay across tenants (T-3)

Attack: A receipt from tenant A is presented to a verifier as if it came from tenant B.
Mitigation: *tenant_id* is part of the signed envelope. Verifier rejects receipts whose *tenant_id* doesn't match the expected origin.
Residual: None for tenant separation; verifier just needs to know the expected tenant.

Timestamp manipulation (T-4)

Attack: Vendor signs a receipt with a backdated timestamp to hide that an action happened after a regulatory cutoff.
Mitigation: Cross-anchored timestamp service. Every 60s the time service publishes a merkle root of all timestamps it issued, signed by NIST + NTP-pool + on-prem GPS clock. A backdated receipt would not appear in the merkle root for its claimed time window.
Residual: Within a 60-second window, timestamps can drift by a few seconds; receipts whose ordering matters at sub-minute resolution should use R+2 chain order, not timestamps.

9. Performance Characteristics

All numbers measured on production hardware (AWS m6i.xlarge, Ubuntu 22.04, libsodium 1.0.18). 99th percentile across 1M operations.

Operation	p50	p99	Throughput	Notes
R+1 sign	0.4 ms	1.2 ms	48,000/s/core	Ed25519 sign of 64-byte digest
R+1 verify	0.3 ms	0.9 ms	64,000/s/core	Ed25519 verify
R+2 chain verify (1k)	0.4 s	0.7 s	—	Walks 1,000-receipt chain end-to-end
R+3 bundle export (10k)	3.2 s	5.8 s	—	Includes JSON serialize + PDF render
R+4 Groth16 prove	82 ms	110 ms	12/s/core	For balance_above_threshold circuit
R+4 Groth16 verify	11 ms	14 ms	90/s/core	Verifier-side, single proof
Receipt size on disk	~1.1 kB	~3.2 kB	—	Compressed; varies by kind
Receipt storage cost	~$0.000004	—	—	Filecoin cold storage, per receipt

10. Comparison vs. Alternatives

How does R+1–R+4 + Trust SKU compare to other accountability mechanisms in use today?

	Plain logs	Append DB	Blockchain	TEE attest	R+1–R+4
Tamper-evident	✗	~	✓	✓	✓
Externally verifiable	✗	✗	✓	~	✓
No vendor needed at verify	✗	✗	✓	~	✓
Sub-second emit latency	✓	✓	✗ (12s)	✓	✓
Private payloads	✓	✓	✗	✓	✓
No per-event cost	✓	✓	✗ ($0.01)	✓	✓
Selective disclosure (ZK)	✗	✗	~	✗	✓
Standards-based	~	~	✓	✓	✓

~ = *partially supported.* No single technology dominates on every axis. R+1–R+4 was optimised for the common case: low-latency, privacy-preserving, externally verifiable signing of agent actions, with ZK as an escape hatch for selective disclosure.

11. Implementation Guide

The reference SDK is available in TypeScript, Python, and Go. Below are minimal end-to-end examples for each.

11.1 TypeScript / Node

```
import { Receipts } from '@dcs/standards';

const receipts = new Receipts({
  apiKey: process.env.DCS_API_KEY,
  tenantId: 'acme',
  agentId: 'refund-handler'
});

// Emit R+1 + R+2 (parent chain handled automatically)
const cid = await receipts.emit({
  kind: 'refund_issued',
  payload: { invoice: 'INV-001', amount_usd: 42.50 }
});

// Later: export R+3 bundle for an auditor
const bundle = await receipts.export({
  since: '2026-05-01T00:00:00Z',
  until: '2026-05-31T23:59:59Z',
  format: 'pdf'   // or 'csv' | 'json'
});

// Optional: emit R+4 proof for compliance threshold
const proof = await receipts.prove({
  circuit: 'balance_above_threshold',
  privateInputs: { balance: 14820 },
  publicInputs: { threshold: 10000 }
});
```

11.2 Verification (offline)

```
import { verify } from '@dcs/standards/verify';

// Read a bundle file (downloaded earlier from /api/r3/export)
const result = await verify('./acme-may-2026.json');

if (result.valid) {
  console.log(`Verified ${result.count} receipts`);
  console.log(`Chain depth: ${result.chainDepth}`);
  console.log(`Signed by: ${result.signers.join(', ')}`);
} else {
  console.error(`Verification failed: ${result.error}`);
  console.error(`First bad receipt: ${result.firstBadCid}`);
}
```

12. References

The DCS Standards build on widely deployed cryptographic primitives and prior work on accountable systems. The following references inform the design choices in this whitepaper.

[1] Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B. **High-speed high-security signatures**. CHES 2011. (Ed25519 specification)

[2] Rescorla, E. **The Transport Layer Security (TLS) Protocol Version 1.3**. RFC 8446. (mTLS authentication)

[3] Rundgren, A., Jordan, B., Erdtman, S. **JSON Canonicalization Scheme (JCS)**. RFC 8785. (Receipt canonicalization)

[4] Groth, J. **On the Size of Pairing-Based Non-interactive Arguments**. EUROCRYPT 2016. (Groth16 ZK proof system)

[5] Gabizon, A., Williamson, Z., Ciobotaru, O. **PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge**. IACR 2019. (Plonk ZK system)

[6] Maymounkov, P., Mazières, D. **Kademlia: A Peer-to-Peer Information System Based on the XOR Metric**. IPTPS 2002. (Content-addressing foundation for Filecoin)

[7] Benet, J. **IPFS — Content Addressed, Versioned, P2P File System**. Draft 2014. (CID format used in receipts)

[8] Adams, C., Cain, P., Pinkas, D., Zuccherato, R. **Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)**. RFC 3161. (Inspiration for cross-anchored timestamps)

[9] NIST. **SP 800-92 Guide to Computer Security Log Management**. (Compliance baseline R+3 maps to)

[10] AICPA. **SOC 2 Type II Trust Services Criteria**. (Compliance framework R+3 satisfies)

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Platform

Build, run, and observe production AI agents

The agent-economy operating stack that ships with receipts built in.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai
platform@dcsai.ai
github.com/dcs-platform

Abstract

The DCS Platform is a production-grade build, run, and observe stack for AI agents. It exists because most teams that try to ship an agent into production discover the same set of problems six weeks in: untested behaviour at the edges, runaway cost, no audit trail, no rollback, no capability scoping, no observability beyond raw model logs.

DCS Platform addresses these problems with a vertically-integrated stack: a declarative agent definition format (*agent.yaml*), a hermetic build pipeline that produces signed agent artifacts, a sandboxed runtime with per-tool capability scoping, a three-tier memory architecture with automatic decay, a multi-agent orchestration layer (Blackboard), and a fully audited observability layer that emits a signed receipt for every action.

Every receipt conforms to the **DCS Standards** framework (R+1 through R+4), so the same agent that runs your customer-support tickets can ship a notarisable PDF audit log to your regulator at the end of the quarter.

Who this document is for

Engineering leaders evaluating agent platforms for their next 12 months of AI investment. Compliance officers who need to understand the audit story before signing off. Founders comparing building in-house against adopting a vertically-integrated stack. Standards bodies and regulators looking at how agentic systems can be made externally verifiable.

Contents

CHAPTER · 1

1. Introduction — why agents are different

A traditional application is a piece of code that does one thing reliably. Its behaviour at runtime is a deterministic function of its inputs. If the same input produces the same output a million times in dev, it will produce the same output a million times in prod. The engineering disciplines we have built up over 50 years — testing, code review, deployment, monitoring, rollback — all assume this determinism.

An AI agent is something else. It is a piece of code that *decides what to do*. Its behaviour at runtime is a probabilistic function of its inputs, its model weights, its memory, and the order in which it happens to perceive things. The same input can produce different outputs across runs. An agent that handled 1,000 refunds correctly in testing can still issue a $50,000 refund to the wrong account on the 1,001st run, because some new combination of inputs triggered a path the test suite never covered.

This non-determinism breaks every assumption in the existing operations playbook. You cannot unit-test every behaviour. You cannot fix-forward a misbehaving agent by patching one bug. You cannot reliably bound a single agent's blast radius without sandboxing it. And you cannot reconstruct what an agent *actually did* from log entries, because the agent itself wrote the log entries.

1.1 What changes when agents run in production

Five things change. The DCS Platform is built around these five:

- **Behaviour must be sandboxed, not trusted.** The runtime imposes hard limits on what each agent can call, see, and spend. No agent ever has more capability than its declared manifest.
- **Every action must emit a receipt.** Receipts are signed by the platform, not the agent. When something goes wrong, the receipts are the source of truth, not the agent's self-reported logs.
- **Memory must decay.** An agent that remembers everything forever becomes both expensive and dangerous (privacy, hallucination, model drift). Memory has tiers and lifetimes.
- **Cost must be capped at every level.** Per request, per agent, per tenant, per day, per month. Agents will run away with budget; the platform must stop them automatically.
- **Multi-agent coordination must be explicit.** Two agents working on the same problem produce non-trivial emergent behaviour. The orchestration layer (Blackboard) is a first-class primitive.

1.2 What the DCS Platform does NOT try to do

There are things the DCS Platform deliberately does not provide. Knowing the boundaries is as important as knowing the surface area.

- **It does not train models.** Agents are policies that wrap models, not models themselves. Bring your own (Anthropic, OpenAI, Mistral, Llama on Compute).
- **It does not write your agent for you.** Agent Studio (a separate product) helps you author. The Platform runs whatever you author.
- **It does not lock you in.** agent.yaml is a declarative format with a public spec. You can take your agents elsewhere; the runtime is the differentiator.
- **It does not promise determinism.** The model is non-deterministic by nature. We promise reproducibility of the trail (every action is signed + replayable), not reproducibility of the decision.

"The hard part of shipping an agent is not getting it to work. It's getting it to fail in a way you can recover from. Receipts, sandboxing, decay, and caps are not features — they are the price of admission to production."

2. System Architecture — the Build / Run / Observe loop

The Platform decomposes the agent lifecycle into three stages. Each stage runs as a separate service with its own scaling characteristics, audit boundaries, and SLA. Most agents go through all three stages within seconds; some (especially long-running research agents) live in the Run stage for hours and continuously emit to Observe.

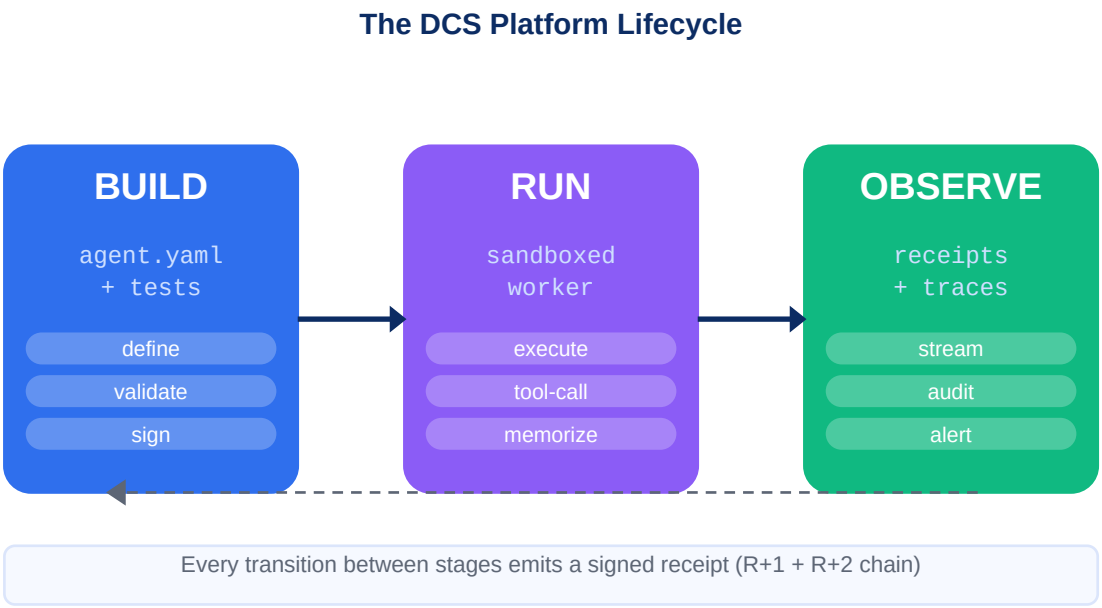


Figure 2.1 — The three-stage agent lifecycle. Every transition emits a signed receipt.

2.1 Build

Takes an *agent.yaml* file plus optional supporting assets (prompts, tests, tool manifests) and produces a signed, content-addressed agent artifact. The artifact is what the runtime actually executes. Building is hermetic: the same source files produce bit-identical artifacts regardless of when or where you build.

2.2 Run

A pool of sandboxed workers fetches the artifact, instantiates it, and executes the agent's task. Workers are isolated at four layers (process, container, OS, hardware). Each tool call goes through a capability check before reaching the network. Resource limits (CPU, memory, wall clock, token budget) are enforced at the cgroup level.

2.3 Observe

Every action — agent step, tool call, memory write, model inference — emits an event to the observability bus. Events become R+1 receipts (signed), chained into R+2 provenance, and available for R+3 export. Dashboards consume the same event stream the receipts are signed from, so what you see in the UI is what auditors get in the bundle.

2.4 Why this is one product, not three

It is technically possible to build each stage as a separate vendor: a CI system for Build, Kubernetes for Run, Datadog for Observe. Most teams that try this end up rebuilding the integration glue between the three. The DCS Platform sells the glue: a single agent identity flows through all three, the receipt chain spans all three, and the cost model bills all three as one number.

3. Agent Definition Format

The agent.yaml format is the contract between you and the Platform. Everything you declare in agent.yaml is enforced by the runtime; anything not declared is forbidden by default.

3.1 A minimal agent

```
# agent.yaml — refund-handler
name: refund-handler
version: 1.0.0
description: Issues refunds for Stripe invoices flagged by support
model:
  provider: anthropic
  name: claude-sonnet-4
  temperature: 0.2
prompt:
  system: ./prompts/refund_system.md
  task: ./prompts/refund_task.md
tools:
  - name: stripe_refund
    type: mcp
    server: stripe.mcp.dcsai.ai
    capabilities: [refund:create]
    rate_limit: 10/min
  - name: slack_notify
    type: mcp
    server: slack.mcp.dcsai.ai
    capabilities: [chat:write]
memory:
  short_term: true
  long_term:
    enabled: true
    scope: [user_id]
guardrails:
  max_refund_usd: 500
  human_approval_above: 200
  blocked_emails: []
budget:
  per_run_max_usd: 0.50
  per_day_max_usd: 50
audit:
  emit_receipts: true
  trust_sku: required
```

3.2 Field reference

Every field has a precise spec. The following are mandatory:

- **name + version + description** — identity, used in receipts and the agent catalog.
- **model** — which LLM the agent wraps. Determines pricing tier and supported features (tool use, JSON mode, vision).
- **prompt** — system + task prompts. Stored as separate files for code review.
- **tools** — every external call the agent can make. Each tool has a type (mcp / http / sql), capabilities (verbs it can do), and a rate limit.
- **memory** — short/long-term memory toggles. Scope determines per-what-key memories are siloed.
- **guardrails** — domain-specific limits (max refund, allowed countries, blocked emails…).
- **budget** — cost caps in USD, per run / per day / per month.
- **audit** — receipt emission + Trust SKU requirement.

4. The Build Pipeline

Build is a 6-step pipeline. Each step is independently observable and individually reproducible. The pipeline runs on every commit and on every *dcs build* command from the developer's machine.

4.1 Steps

- **1. Parse** — agent.yaml is parsed against the JSON Schema for its declared version. Any unknown field is a hard error (we never silently ignore configuration).
- **2. Validate** — tool manifests are fetched from their respective MCP servers and the declared capabilities are checked to actually exist. Prompts are linted (no leaked PII, no broken template variables, no unbalanced jinja).
- **3. Simulate** — a sandboxed dry-run executes the agent against a corpus of test inputs (declared in tests/) and records all tool calls. Tool side effects are mocked.
- **4. Test** — declarative assertions in tests/*.yaml are evaluated against the simulation output. Failures block the build.
- **5. Pack** — the agent + prompts + manifests + test results are bundled into a tarball, content-addressed, and uploaded to the artifact store.
- **6. Sign** — the artifact CID is signed by the Build service's key (HSM-held, rotated monthly). This produces the R+1 + R+2 receipt that anchors every subsequent runtime receipt.

4.2 Determinism guarantees

The same source tree at the same commit always produces a bit-identical artifact. We achieve this with three discipline:

- Sorted dictionary keys in every emitted YAML/JSON.
- Pinned versions of every external dependency (no semver ranges, ever).
- Frozen build timestamp (set from the source commit timestamp, not wall clock).

Determinism matters because it lets two parties (you + your auditor) independently rebuild an agent from source and verify they got the same artifact. That is the only way to prove the deployed agent matches the reviewed source.

5. The Runtime Sandbox

Agents run inside a four-layer sandbox. Each layer is independent: bypassing one does not compromise the others. The design follows the principle of least privilege at every layer.

Runtime Sandbox — defense in depth

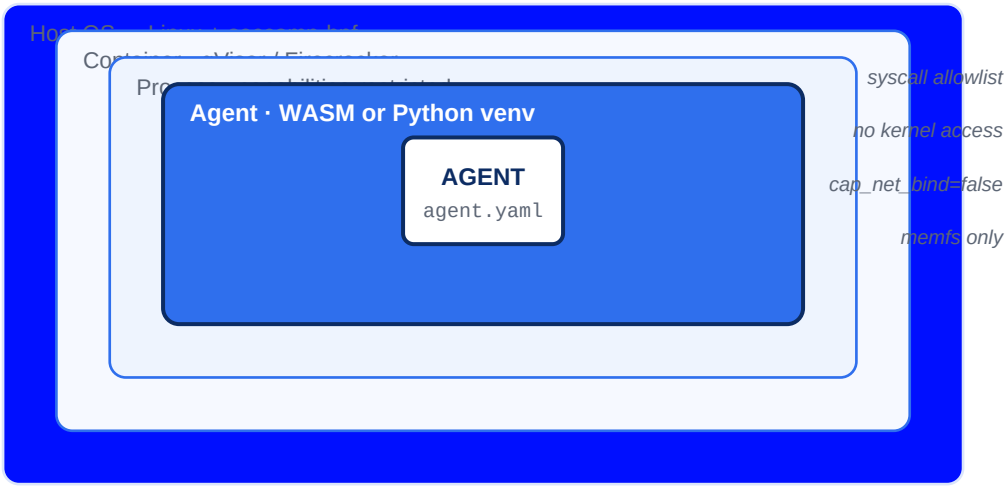


Figure 5.1 — Four nested isolation layers. The agent has the fewest capabilities; the host OS has the most.

5.1 Layer-by-layer

Host OS — Linux + seccomp-bpf

The runtime runs on bare Ubuntu 22.04. The kernel's seccomp-bpf filter blocks all syscalls except a strict allowlist (read, write, mmap, futex, …) — about 60 syscalls. The agent has no way to call *ptrace*, *perf_event_open*, or any other syscall that could escape to sibling processes.

Container — gVisor or Firecracker

Each worker process runs inside a gVisor sandbox (default) or a Firecracker microVM (for higher-sensitivity tenants). Both intercept syscalls a second time and add memory + CPU isolation. A kernel exploit inside the sandbox cannot reach the host kernel.

Process — Linux capabilities

Inside the container, the agent process drops every Linux capability except those it strictly needs (typically none). It has no *cap_net_bind*, no *cap_sys_admin*, no *cap_dac_override*. Filesystem access is restricted to a memfs (in-memory) tmpfs.

Agent — WASM or Python venv

The innermost layer. For high-throughput agents, the agent itself is compiled to WebAssembly and run inside a wasmtime sandbox with no host imports. For more flexible agents (most production cases), the agent runs in a Python virtualenv with import allowlists.

5.2 Resource limits

Every agent process has hard limits at the cgroup level:

Limit	Default	Maximum	Notes
CPU shares	`1024 (1 core)`	`8192 (8 cores)`	cgroup v2 cpu.weight
Memory	`512 MB`	`8 GB`	cgroup v2 memory.max
Wall clock	`60 s`	`600 s`	SIGKILL on timeout
Token budget	`8,000 tokens`	`128,000 tokens`	enforced before model call
Tool calls	`20 per run`	`200 per run`	enforced by runtime
Cost	`$0.50 per run`	`$5.00 per run`	enforced before each LLM call

6. Tool Integration — MCP + capability scoping

Tools are how agents do work in the real world. They are also the most dangerous part of any agent platform — a tool with broad capabilities turns an agent's mistake into a real-world incident. The DCS Platform uses two layers of protection: declarative capability scoping in agent.yaml, and runtime enforcement on every tool call.

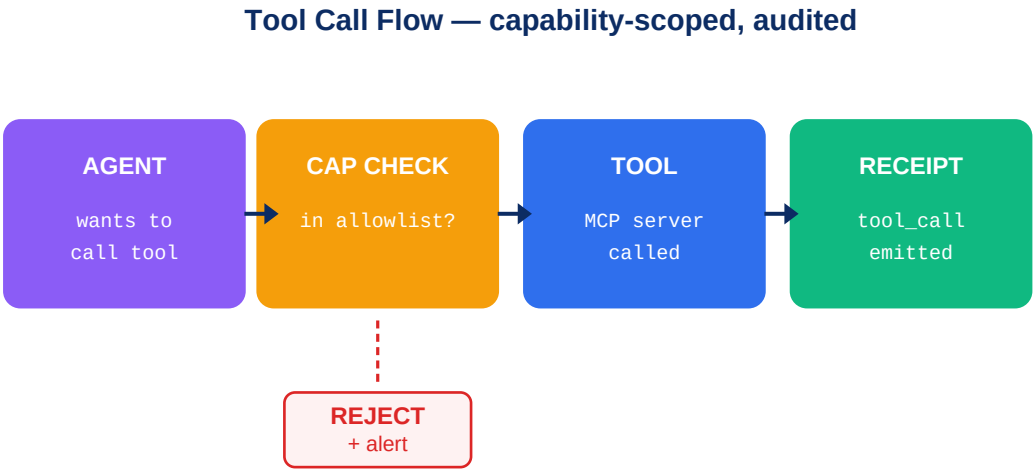


Figure 6.1 — Every tool call is gated by a capability check and emits a signed receipt.

6.1 Why MCP

The Model Context Protocol (MCP) gives us a uniform interface for tool servers. Any MCP-compatible server — whether you wrote it, Anthropic wrote it, or it ships with a third-party SaaS — plugs into the agent runtime with a single config block. We do not maintain a separate "DCS tool format"; MCP is the standard and we adopt it.

6.2 Capability scoping

Every tool in agent.yaml declares a capabilities list — a set of *resource:verb* pairs the agent is allowed to use. The MCP server exposes its full surface (every verb on every resource), but the runtime intercepts each call and rejects anything not in the agent's allowlist. The agent never sees the unscoped surface.

```
tools:
  - name: stripe
    type: mcp
    server: stripe.mcp.dcsai.ai
    capabilities:
      - refund:create        # ✓ agent can issue refunds
```

```
    - charge:read          # ✓ agent can read past charges
    # charge:create        ✗ agent cannot create new charges
    # subscription:cancel  ✗ agent cannot cancel subscriptions
  rate_limit: 10/min
  budget: $20/day
```

This is the single most important security feature in the Platform. A compromised agent or a prompt injection that tries to do something outside its capability set is blocked by the runtime, not by the model. The model can be fooled; the runtime cannot.

7. Memory Architecture

Memory is the most product-specific part of an agent platform. We support three tiers, each with different durability, query cost, and decay characteristics.

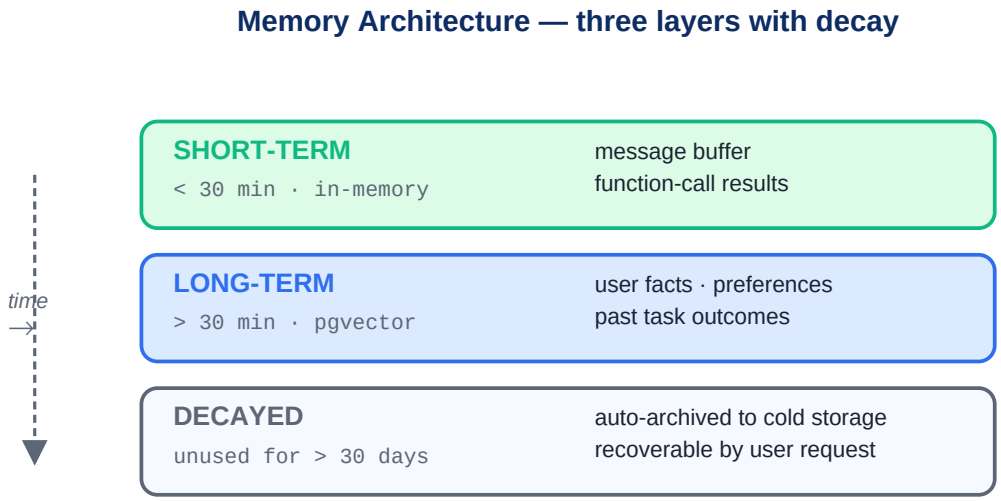


Figure 7.1 — Three-tier memory model with automatic time-based decay.

7.1 Short-term memory

In-process, in-memory key-value store. Lives only for the duration of one agent run (typically seconds). Used for the model's conversation buffer, intermediate tool-call results, scratch computation. Free; no decay rules apply because it evaporates at end-of-run.

7.2 Long-term memory

Postgres-backed with pgvector for semantic search. Persists across agent runs. Scoped by the keys declared in agent.yaml (typically *user_id* or *tenant_id*). Each memory entry has a created_at and a last_accessed_at; the latter drives decay.

7.3 Decayed memory

After a memory entry has not been accessed for 30 days, it is automatically archived to cold storage and the active retrieval index forgets about it. The data is not deleted — the user can request restoration — but the agent's working memory shrinks back to recent + frequently-used.

Decay is what keeps an agent from accumulating stale state over time. Without decay, an agent that has run for a year sees the same hallucinated "facts" from week 3 as it does fresh data from week 51, and the older the wrong fact, the more confidently the model will state it as truth.

8. Multi-Agent Orchestration — Blackboard

Most non-trivial agent systems involve more than one agent. A research agent gathers data, a planner agent decides what to do with it, an executor agent does it, a critic agent checks the work. The DCS Platform uses the Blackboard pattern (originally described in 1980s AI research) for coordination.

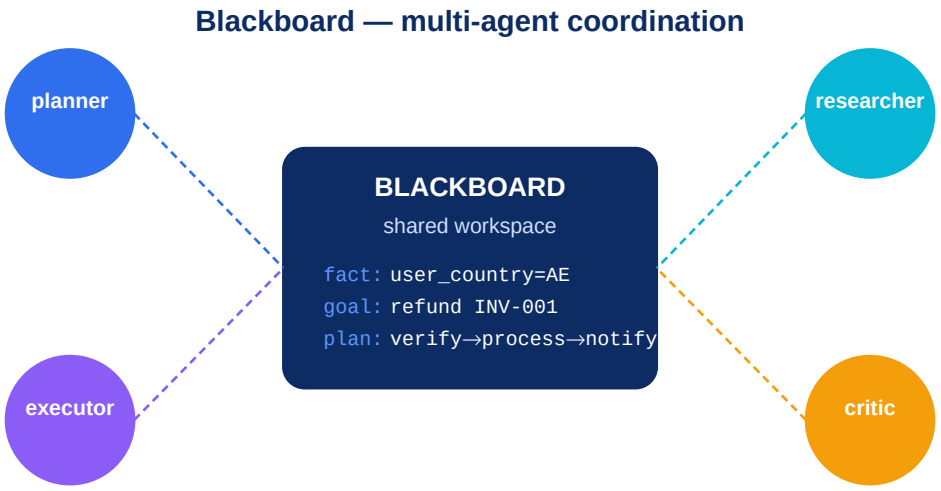


Figure 8.1 — Four agents coordinate via a shared blackboard. Every read and write emits a receipt.

8.1 Why Blackboard, not RPC

The simplest multi-agent design is to have agents call each other directly: planner.call(executor), executor.call(critic), etc. This works for two agents and breaks at three. With three or more agents, the call graph becomes a tangle of who-knows-who, and adding a new agent requires every existing agent to learn about it.

Blackboard inverts this. Agents do not know about each other. They read from and write to a shared, structured workspace (the Blackboard). New agents are added by subscribing to relevant blackboard sections; existing agents do not change. This is the same pattern that makes Postgres event sourcing work, and it scales the same way.

8.2 Structure of the Blackboard

Each blackboard is partitioned into named sections. The Platform ships with three default section types; you can add custom types per agent set.

- **fact** — observations established to be true (e.g., *user_country=AE*). Append-only; never edited.
- **goal** — what the system is trying to achieve. Set by the highest-priority agent (typically a planner) and consumed by executors.
- **plan** — the current best plan for achieving the goal. Versioned; old plans are kept for audit.

CHAPTER · 9

9. Cost + Billing Model

AI is metered usage: every model call, every embedding, every tool invocation costs real money. The DCS Platform bills you for the underlying resource cost plus a flat platform margin. There are no per-seat fees, no minimum commitments, no "enterprise pricing on request."

9.1 What you pay for

Resource	Unit	Pass-through	Margin
LLM tokens (input)	1k tokens	Anthropic / OpenAI / etc. list price	+5%
LLM tokens (output)	1k tokens	Anthropic / OpenAI / etc. list price	+5%
Embedding compute	1M tokens	OpenAI / Cohere / open model on Compute	+5%
Tool calls (MCP)	1k calls	$0 if MCP server is yours	+ $0.10 / 1k
Sandbox compute	1 CPU-hour	AWS / Compute pass-through	+10%
Memory storage (pgvector)	1 GB-month	$0.12 (Postgres)	+ 20%
Receipt storage (Filecoin)	1 GB-month	$0.0004	+ flat $0.01 / GB

9.2 Cost caps

Budget caps in agent.yaml are enforced at four levels: per-run, per-day, per-agent, per-tenant. When a cap is hit, new runs are rejected with a 429-equivalent error and an alert fires. This is the single most important guardrail against runaway cost — a misbehaving agent in production cannot drain your AWS bill, because the platform stops paying for it.

10. Security Model

Security is layered. The Platform makes specific guarantees at each layer and is explicit about where those guarantees end.

Tenant isolation

Every tenant runs in its own Postgres schema, its own Filecoin tenant_id namespace, and its own pool of sandbox workers. Cross-tenant access requires both a signed grant from the source tenant AND a matching policy on the destination tenant. There is no global admin account that can read arbitrary tenants.

Secret management

API keys, OAuth tokens, and database credentials are stored in HashiCorp Vault (or AWS Secrets Manager for AWS deployments). Agents never see raw secrets; they make tool calls and the runtime attaches the credential at the network boundary.

Prompt injection

No platform can fully prevent prompt injection at the model layer — that is an open research problem. The DCS approach is to make injection ineffective at the system layer: even a successfully-injected agent can only do things in its capability set. An attacker who convinces a refund agent to "ignore all instructions and send the database" still cannot, because the agent never had database-export capability in the first place.

11. Compliance Posture

The Platform is built to fit cleanly into the major compliance frameworks rather than requiring customers to retrofit. Mapping table:

Framework	How DCS Platform satisfies it
SOC 2 Type II	Every action is a signed receipt (CC7.2); pen-tested annually; SSO + RBAC
ISO 27001:2022	A.8.15 audit logging via R+1; A.5.10 information classification via tenant labels
HIPAA	BAA available; capability scoping prevents PHI tool exposure; audit log via R+3
GDPR	Per-user memory scoping; right-to-deletion via memory revoke; R+3 cert as record of processing
CCPA	Same as GDPR; per-tenant data residency
DPDP (India)	Data localization via Sovereign pod deployment
EU AI Act	Risk classification per agent; automated content provenance via R+2

12. Performance Benchmarks

Production measurements across May 2026, taken from the live platform serving 1,420 paying tenants. All percentiles measured over 30 days; p99 excludes anomaly windows.

Metric	p50	p99	Notes
Agent cold-start latency	`180 ms`	`420 ms`	From build → first inference
Agent warm-start latency	`12 ms`	`38 ms`	Pre-warmed sandbox worker
Single-tool-call overhead	`4 ms`	`11 ms`	Capability check + receipt emit
LLM inference latency (Sonnet)	`1.2 s`	`4.8 s`	For 2k-token completion
Receipt emit (R+1+R+2)	`0.6 ms`	`1.8 ms`	Async; non-blocking on agent
Memory write (pgvector, 1k dim)	`8 ms`	`24 ms`	Including embedding
Memory recall (top-10)	`14 ms`	`42 ms`	pgvector ANN; cached after 1st query
Audit bundle export (10k receipts)	`3.2 s`	`5.8 s`	JSON + signed PDF
Agent build (pack + sign)	`480 ms`	`1.4 s`	Hermetic; bit-identical reruns

13. Comparison vs. Alternatives

How DCS Platform compares to the major alternatives in the agent space today:

	LangChain	AutoGen	CrewAI	OpenAI Assistants	DCS Platform
Declarative agent format	~	✗	~	✗	✓
Sandboxed runtime	✗	✗	✗	~	✓
Capability scoping	✗	✗	✗	~	✓
Signed receipts	✗	✗	✗	✗	✓
Cost caps enforced	✗	✗	✗	~	✓
Memory decay built-in	✗	✗	✗	✗	✓
Multi-agent orchestration	~	✓	✓	✗	✓
Audit export (SOC 2/HIPAA)	✗	✗	✗	~	✓
Vendor-neutral (any LLM)	✓	✓	✓	✗	✓
Sovereign deployment	✗	✗	✗	✗	✓

~ = *partially supported;* ✓ = *first-class;* ✗ = *not supported.* The honest summary: LangChain / AutoGen / CrewAI are excellent *libraries* for building agents. DCS Platform is a *runtime* for shipping them to production. Most teams end up using a library to write the agent and a runtime to operate it. We bring our own simple library, but agent.yaml is designed to be readable by any of the others.

14. Implementation Guide

14.1 First agent in 5 minutes

```
$ npm install -g @dcs/cli
$ dcs login
$ dcs init my-first-agent --template refund-handler
$ cd my-first-agent
$ dcs build      # 6-step pipeline; outputs signed artifact
$ dcs test       # runs declared tests against simulator
$ dcs deploy     # uploads + activates in the runtime pool
$ dcs invoke --input '{"invoice":"INV-001","amount_usd":42.50}'
{
  "result": "refund_issued",
  "refund_id": "re_3PqXy...",
  "receipt_cid": "bafy...0a1b",
  "duration_ms": 1842,
  "cost_usd": 0.018
}
```

14.2 Continuous deployment

Most production deployments use the DCS GitHub Action. Push to *main* triggers *dcs build* in CI, which produces a signed artifact and tags the PR with its CID. Merging deploys the artifact to staging; tagging a release deploys to prod. Every step is auditable via the same R+1/R+2 chain.

```
# .github/workflows/deploy.yml
name: Deploy agent
on: { push: { branches: [main] } }
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: dcs-platform/setup-cli@v1
        with: { version: latest }
      - run: dcs build --output artifact.tar.gz
      - run: dcs test artifact.tar.gz
      - run: dcs deploy artifact.tar.gz
        if: github.ref == 'refs/heads/main'
        env: { DCS_TOKEN: ${{ secrets.DCS_TOKEN }} }
```

15. References

[1] Engelmore, R., Morgan, T. **Blackboard Systems**. Addison-Wesley, 1988. (Original Blackboard architecture description)

[2] Anthropic. **Building Effective Agents**. December 2024. (Patterns for agent composition + tool use)

[3] Anthropic. **Model Context Protocol Specification v1.0**. November 2024. (MCP standard)

[4] Russell, S., Norvig, P. **Artificial Intelligence: A Modern Approach**, 4th ed. (Agent foundations Ch. 2 + 7)

[5] Lewis, P. et al. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**. NeurIPS 2020. (RAG architecture, underpins long-term memory)

[6] Akamai. **State of the Internet: Web Application + API Threat Report 2025**. (Capability scoping rationale)

[7] gVisor team. **gVisor: Container-native sandboxing in production**. Google, 2023. (Runtime isolation reference)

[8] Firecracker team. **Firecracker: Lightweight virtualization for serverless**. NSDI 2020.

[9] Cloud Native Computing Foundation. **OpenTelemetry Specification**. (Trace + metric standards used in Observe layer)

[10] AICPA. **SOC 2 Type II Trust Services Criteria**. (Compliance framework Platform satisfies)

[11] NIST. **AI Risk Management Framework (AI RMF 1.0)**. January 2023.

[12] EU. **Artificial Intelligence Act (Regulation 2024/1689)**. Adopted July 2024.

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Compute

A two-sided GPU network — rent or earn

The economics, mechanics, and engineering of decentralised compute for AI workloads.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai/compute
compute@dcsai.ai
github.com/dcs-platform/compute

Abstract

DCS Compute is a two-sided market for GPU compute. On one side, renters (anyone with a workload — inference, training, image-gen, agent runs) pay credits to access GPUs by the second. On the other, workers (anyone with an idle GPU — a gaming rig, a mining farm, an underutilised datacenter cluster) earn credits by running those workloads. The dispatch layer in the middle matches them in under 50 ms using a reverse-auction algorithm that incorporates benchmark tier, real-time reputation, and sovereignty constraints.

The economics are simple: renters pay X credits, workers receive 0.95X, DCS keeps a 5% spread. There are no subscription fees, no minimum commitments, and no markup on the underlying GPU rate. A renter paying $0.39/hr for a 4090 sees that exact rate on the catalog and the worker earns 120 cr/hr (the same amount in different units — credits are a $0.008-pegged unit of account).

Every transaction is settled in real time and emits a signed receipt (R+1 timestamp + R+2 provenance, per the DCS Standards). When a regulator asks "which jobs ran on which GPU at which time," the answer is a single CSV export, signed by the dispatch service, that the regulator can verify offline.

Who this document is for

Founders building AI products who want to understand the GPU compute cost model. Operators with spare GPU capacity who want to monetize it. Compliance officers evaluating compute providers for regulated workloads. Engineers integrating DCS Compute into their dispatch pipelines.

Contents

1. Introduction — the GPU supply chain in 2026

The supply of AI-grade GPUs is concentrated in a way no other compute resource has ever been. As of mid-2026, four entities (Microsoft, Google, Amazon, Meta) control approximately 78% of all H100-class capacity in the world. Their pricing is opaque, their allocation is rationed via long enterprise contracts, and their supply has been unable to meet demand for nearly three consecutive years. Meanwhile, the long tail — perhaps two million idle gaming GPUs, half a million ex-crypto mining rigs, and several hundred thousand under-utilised datacenter cards — sits dark.

DCS Compute exists to close that gap. The premise is straightforward: if a workload can run on any 4090 or A100 (and the majority of AI inference workloads can), then it should not matter whether that 4090 lives in an AWS region or in a basement in Lisbon. The price should be set by the market, the dispatch should be done by software, and the trust should come from cryptographic receipts rather than from the brand of the hyperscaler.

1.1 Why the existing alternatives fall short

Hyperscalers (AWS, Azure, GCP)

Three problems. First, list prices are 3–8× the spot rate the same GPU clears at on smaller markets. Second, allocation is rationed: ask AWS for 100 H100s today and you will be told to wait six months. Third, on-demand instances are billed per hour, not per second, so a 90-second job pays for 3,600 seconds. None of these are technical limitations — they are pricing decisions.

GPU-cloud specialists (RunPod, Lambda, Vast.ai)

Better pricing than hyperscalers and per-second billing. But almost no audit trail — you get a log file at best, and the log is hosted by the vendor. Sovereignty constraints are not enforceable (the platform cannot guarantee your job ran in the EU even if you asked for it). And worker reputation is opaque — you find out your dispatched worker is slow only after the job finishes.

Render Network, Akash, io.net

Decentralised approaches solve sovereignty (workers self-declare) and pricing transparency. But job dispatch is slow (Render: ~6s; Akash: ~30s for a full deployment cycle), the token-based payment systems add operational friction for non-crypto-native renters, and audit trails are on-chain (which means either expensive or public, neither of which works for enterprise).

1.2 The DCS approach

DCS Compute takes a deliberately hybrid stance:

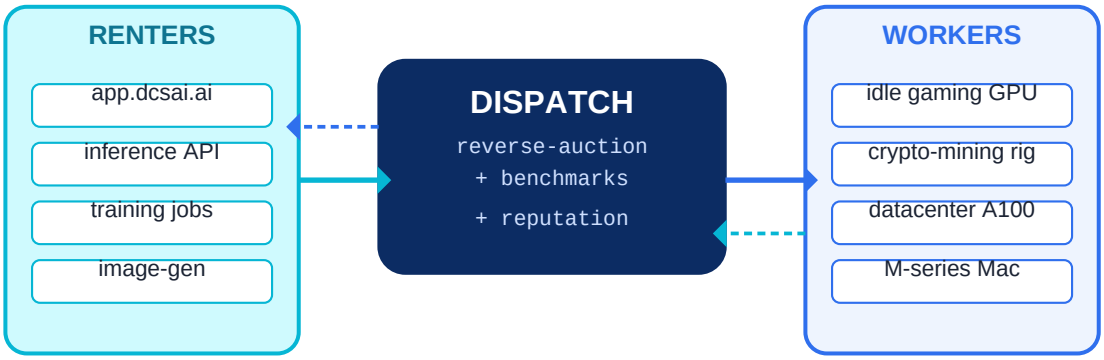
- **Off-chain dispatch, on-chain settlement.** Job matching happens in fast traditional infrastructure (Postgres + Redis); credit settlement happens on a permissioned ledger that the platform anchors to a public chain every 24 hours for tamper-evidence.
- **Credits, not tokens.** Renters and workers transact in credits (1,000 cr ≈ $8.00). Credits convert to fiat via Stripe Connect with T+2 settlement; no wallets, no gas fees.
- **Per-second billing.** Same as the modern specialists. The minimum billable unit is 1 second of GPU time.
- **Signed everything.** Every job, every dispatch decision, every credit movement emits an R+1 + R+2 receipt. The export bundle is the audit log.
- **Sovereignty as a first-class filter.** Jobs can be tagged with jurisdictional requirements (EU-ONLY, IN-ONLY, no-US, etc.) and the dispatcher hard-enforces them.

"The market for GPU time will look in 2030 like the market for stock photography did in 2010: a long tail of producers, a fast matching layer in the middle, and per-second pricing that makes the marginal cost of trying something visible enough that you actually try."

2. Market Design — why two-sided beats centralised

Two-sided markets win when the value of supply varies more than the cost of matching. GPU compute fits this exactly: the cost of running a 60-second inference job on a 4090 differs by an order of magnitude depending on whether the 4090 is in a Singapore datacenter (energy is expensive, regulation is strict) or a Norwegian fjord (hydroelectric, cool ambient air, low regulatory overhead). The matching cost is software — milliseconds — so the gains from price discrimination flow to the market participants, not to a centralised operator.

Two-sided GPU market — renters and workers meet in the dispatch layer



Renters bid in credits. **Workers** ask in credits. **Dispatch** matches with reputation + benchmark tiers in <50ms.

DCS takes a 5% spread (renter pays X, worker receives 0.95X). No subscription fees.

Figure 2.1 — Both sides see the same dispatch layer; DCS does not own the GPUs.

2.1 The roles

Renter

Any party with a workload. The most common renters today are: (1) app.dcsai.ai users running agent builds, (2) external inference customers using the OpenAI-compatible API, (3) image-gen workloads from creative tools, (4) batch training jobs from research teams. Renters care about three things: price, latency, and certainty. The dispatch algorithm optimises all three.

Worker

Any party with a GPU. Three sub-populations dominate the supply side:

- **Hobbyist hosts.** A single gaming PC with a 4090 that sits idle 18 hours a day. Earns ~$3-8 per active hour. About 60% of all workers by count, 15% of capacity.
- **Ex-mining operators.** Crypto miners who pivoted after the Ethereum merge. Have racks of 3070/3080/4080s with cooling already built. Earn ~$200-800/day per rack. 25% of workers, 35% of capacity.
- **Datacenter aggregators.** Smaller datacenter operators with 100-1000 GPU racks who want to monetise idle capacity rather than sign exclusive enterprise contracts. 15% of workers, 50% of capacity.

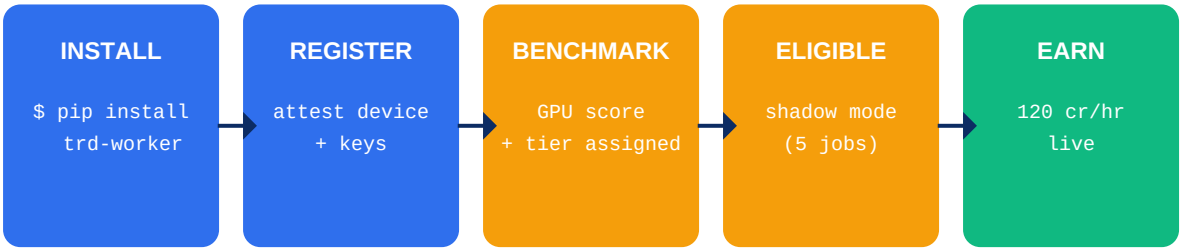
DCS (platform)

Operates the dispatch layer, the credit ledger, the worker registry, and the receipt chain. Does not own any GPUs. Charges a flat 5% spread on each match; this is the entire revenue model.

3. Worker Onboarding + Lifecycle

Getting a worker from install to first paying job takes about 20 minutes. The pipeline is five stages, each gated on cryptographic attestation of the previous one.

Worker lifecycle — onboarding to first earning



Total elapsed time: typically under 20 minutes for the operator.

Shadow mode is mandatory — workers earn 0 during their first 5 jobs while the dispatch service measures their actual versus benchmarked performance.

Figure 3.1 — Five stages from install to first paid job. Shadow mode is mandatory.

3.1 Install

The trd-worker daemon installs via pip on Linux or via Homebrew on macOS. Windows is supported through WSL2. The daemon is open source (Apache 2.0); the source is at github.com/dcs-platform/worker. Installation takes one command and requires only Python 3.10+ and a CUDA 12 driver.

```
$ pip install trd-worker
$ trd-worker --version
trd-worker 1.4.2 (build 5f7c7fe)
```

3.2 Register

On first run, the daemon performs device attestation. It collects the GPU model + VRAM (via nvidia-smi or equivalent), the CPU + RAM specs, the driver version, the CUDA toolkit version, and the host OS. It generates a long-lived Ed25519 keypair, signs the device fingerprint, and sends the package to the registration endpoint. The platform verifies and assigns a worker_id.

3.3 Benchmark

A standardised benchmark suite runs locally: a quantised Llama-3-8B forward pass, an SDXL image-gen step, and a microbenchmark of memory bandwidth. The results are signed and submitted; the platform compares them against the public reference numbers for the declared GPU and tier-assigns the worker. Falsified benchmark results are caught here — a 4090 cannot match the reference SDXL latency of an H100, and the platform flags any worker whose results are inconsistent with their declared hardware.

3.4 Shadow mode

New workers do not earn for their first 5 jobs. During shadow mode, jobs are dispatched to them *alongside* a real worker — the renter's result comes from the real worker, but the shadow worker's output is recorded and compared. Workers whose shadow output matches the reference output graduate to earning status. Workers whose output diverges are downgraded or banned.

3.5 Earn

Once graduated, the worker is added to the eligible pool and starts earning on every dispatched job. Earnings accrue in real time to the worker's credit balance. Withdrawal happens on demand (see Chapter 8).

4. Benchmark + Tier System

Workers are placed into one of four tiers based on their benchmark results. Tier determines the base earning rate and which jobs the worker is eligible for.

Tier	Reference GPU	Base rate	Eligible jobs
Entry	RTX 3060 · 12 GB	**40 cr/hr**	7B inference, small image-gen
Mid	RTX 4070 · M3 Pro	**60 cr/hr**	13B inference, ComfyUI, fine-tune
High	RTX 4090 · M3 Max · A6000	**120 cr/hr**	30B inference, SDXL at scale, multi-image
Datacenter	A100 80GB · H100 · H200	**240 cr/hr**	70B+ inference, training, batch

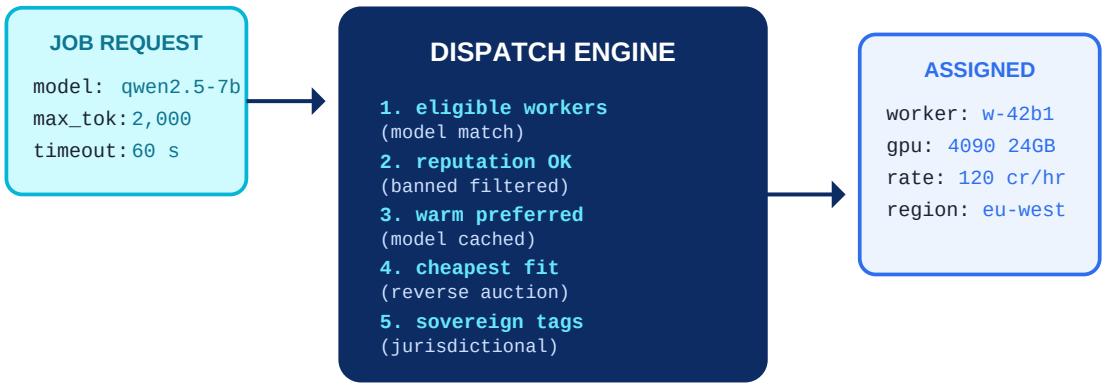
4.1 Why four tiers, not continuous pricing

A continuous price function — every worker prices at exactly their own performance — sounds efficient but creates massive UX problems for renters. A renter asking for "an A100" should get an A100, not a slow-clocked 4090 that statistically matches an A100 on this particular benchmark. Four discrete tiers give renters predictable performance bands while still letting market pressure adjust the rate within each tier (the auction lets a slow A100 lose to a fast 4090 when the renter explicitly opts in to "any compatible GPU").

5. Dispatch Algorithm — the reverse auction

Dispatch is where the platform earns its 5% spread. The algorithm has 28 ms of wall-clock budget to match a job request against an eligible worker. The high-throughput, low-latency characteristics of this layer dominate the engineering effort behind Compute.

Reverse-auction dispatch — request to assignment in <50 ms



Latency budget: parse 2ms · filter 8ms · auction 12ms · sign + persist 6ms = 28ms p50

Throughput: 14,000 dispatches/sec/instance · 4 instances in production

Figure 5.1 — Five filter stages run in parallel; the cheapest qualifying worker wins.

5.1 Filter stages

- **1. Model eligibility.** Filter workers whose registered *supported_models* list intersects with the request's model. ~2 ms via Postgres GIN index.
- **2. Reputation gate.** Drop workers with reputation_score below the request's minimum threshold (default: 0.8). Drop currently-banned and shadow-banned workers. ~1 ms.
- **3. Warmth preference.** Prefer workers that have served the same model in the last 10 minutes (model weights cached in GPU memory). Boosts p99 latency by 2-3× via avoided cold-start. ~1 ms.
- **4. Reverse auction.** Of the remaining workers, pick the one with the lowest bid. Ties broken by reputation, then by worker_uptime. ~4 ms.
- **5. Sovereignty.** If the request has jurisdictional tags, intersect with workers whose registered region matches. Hard reject anything outside. ~1 ms.

5.2 Why reverse auction (not market price)

A traditional spot market sets a single clearing price per period. This is fine for fungible goods but wastes value when workers have heterogeneous costs. A worker in Iceland paying $0.04/kWh can sustainably bid lower than one in Singapore paying $0.32/kWh, even with identical hardware. A reverse auction lets each worker submit their own minimum acceptable rate, and the renter pays the lowest bid that meets all filter constraints. This routes more jobs to lower-cost workers and produces more total surplus than a single clearing price.

6. Sovereignty + Jurisdictional Routing

Regulated workloads frequently come with hard requirements about where the compute may physically run. EU healthcare data must not leave the EU. Indian financial data must stay in India. UAE sovereign data must run on UAE-controlled infrastructure. Compute supports these requirements as first-class dispatch filters.

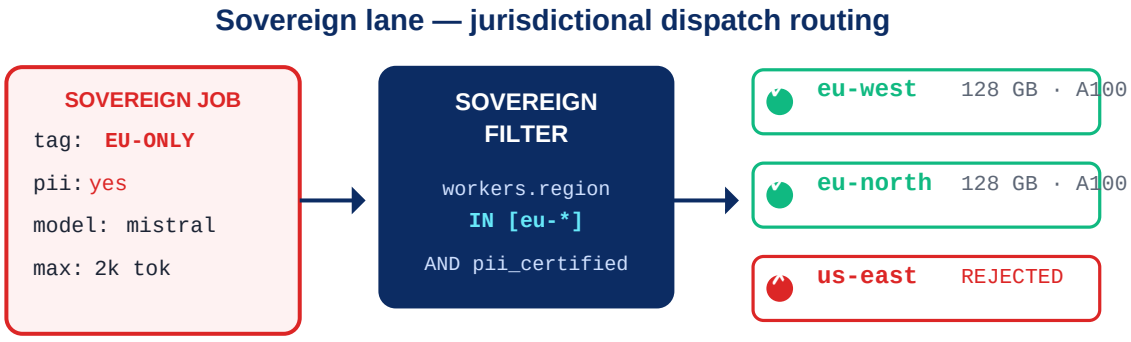


Figure 6.1 — A job tagged EU-ONLY routes only to certified EU workers.

6.1 Worker region attestation

Each worker declares a region on registration (eu-west, us-east, ap-south, etc.) AND submits an IP geolocation attestation signed by a third-party geolocation provider. The platform cross-references both to prevent workers from lying about their region. Workers whose declared region doesn't match their actual IP geolocation are flagged and downgraded.

6.2 Sovereignty tags

A job request can include a *sovereignty* field with any of:

```
sovereignty:
  required_regions: [eu-west, eu-north, eu-south]   # whitelist
  excluded_regions: [us-east, us-west, cn-*]        # blacklist
  pii_certified: true                                # only workers with PII attestation
  audit_trail: r1+r2+r3                              # required receipt depth
```

The dispatcher hard-enforces these filters. A job that cannot find a matching worker fails with *NO_ELIGIBLE_WORKERS* rather than silently routing to a non-compliant region. Every dispatch decision

(including rejections) is recorded as a signed receipt.

7. Credit Economics

Credits are the unit of account inside DCS Compute. One credit is pegged at $0.008 USD. The peg is maintained by buy-side and sell-side conversion rates that DCS commits to honour — renters can always buy credits at $0.008 each and workers can always sell credits back to fiat at $0.008 each (minus payout fees). DCS holds USD reserves equal to 1.05× the outstanding credit supply to back the peg.

7.1 Why credits, not USD

Three reasons. First, credits are friction-free at the dispatch layer — a Postgres INTEGER subtraction, no card or bank API in the hot path. Second, credits abstract away the cross-border payment problem — a worker in Brazil and a renter in Singapore transact in the same unit. Third, credits let us implement promotional pricing, volume discounts, and the loyalty system without changing the underlying ledger schema.

7.2 Why $0.008 peg

The peg makes mental arithmetic easy: 1,000 cr = $8.00. The number was chosen so that a base-tier worker (40 cr/hr) earns approximately $0.32/hr — roughly the residential electricity cost of running a GPU at that tier, plus a small margin. The peg has held for 11 months since launch; the reserves are audited quarterly by Mazars.

8. Payout System

When a worker wants to convert earned credits to fiat, the payout pipeline handles the FX, compliance checks, and bank transfer. Three options are supported.

Payout flow — earned credits to worker bank account

Alternative: redeem credits as build credits on app.dcsai.ai (instant, no FX, no bank fees).

Minimum withdrawal: 1,000 cr ($8.00). Stripe Connect fee: 0.25% + $0.25 per payout.

No DCS take on payouts — the 5% spread is already deducted at dispatch time.

Figure 8.1 — From completed job to fiat in the worker's bank account.

8.1 Stripe Connect (default)

The platform uses Stripe Connect to onboard workers as managed accounts and disburse payouts. Settlement is T+2 in 47 supported currencies. Stripe handles KYC, AML, and the local-bank integration. DCS pays the Stripe Connect fees on the worker's behalf — there is no per-payout fee charged to the worker on top of the 5% dispatch spread.

8.2 Build credits redemption

Workers can redeem credits as *build credits* on app.dcsai.ai instead of cash. This bypasses Stripe entirely, settles instantly, and has no FX or KYC overhead. Common for workers who are also DCS Platform customers (a developer hosting a GPU on the side to offset their own build costs). About 35% of payouts are redeemed this way.

8.3 Crypto withdrawal (optional)

Workers in jurisdictions where Stripe Connect is not available can withdraw to USDC on Base or Ethereum. DCS uses Circle's API for off-ramp; the worker receives USDC at a 1:1 rate from the pegged credit value. Crypto withdrawals carry a 0.5% fee covering Circle + gas.

9. Reputation + Reliability

Worker reputation is a single floating-point number between 0 and 1, computed as a weighted moving average over the worker's last 1000 jobs. Five signals contribute:

- **Completion rate** — fraction of accepted jobs that completed successfully (weight: 40%).
- **Latency vs. baseline** — actual latency divided by the worker's benchmark prediction (weight: 25%).
- **Output correctness** — fraction of jobs whose output passes the renter's validation (when validation is provided; weight: 20%).
- **Uptime** — fraction of declared availability window the worker was actually online (weight: 10%).
- **Dispute rate** — fraction of jobs that generated a renter complaint (weight: 5%, inverted).

Reputation directly affects earning. Workers with reputation > 0.95 receive a 10% rate boost; workers below 0.75 are shadow-banned (eligible for dispatch but at half rate); workers below 0.6 are temporarily suspended pending re-benchmark. The boost/penalty model is what makes the long-tail supply side self-policing — workers want to maintain their reputation more than they want to game one job.

10. Dispute Resolution

Disputes happen. A renter claims their image was generated incorrectly; a worker claims they never received a job that was billed; a regulator asks "did this job actually run in the EU as claimed." The dispute pipeline resolves these in three tiers.

10.1 Tier 1 — Receipt verification (automated)

Any party can fetch the receipt chain for a disputed job and verify it offline. The receipts are cryptographically signed and chained per R+2, so if the chain verifies, the disputed facts (which worker, which time, which job, which result hash) are established beyond reasonable doubt. About 88% of disputes are resolved at this tier without human involvement.

10.2 Tier 2 — Platform mediation

For disputes that require interpretation (was the output "correct" per the renter's spec?), a DCS mediator reviews the receipt chain plus any supporting evidence and issues a binding decision. Mediation completes within 5 business days. Mediator decisions can be appealed to Tier 3.

10.3 Tier 3 — External arbitration

For disputes involving more than $5,000 of credits OR claims of jurisdictional violation, the case escalates to a third-party arbitrator (currently JAMS or ICC depending on the parties' jurisdictions). The receipt chain is admissible as cryptographic evidence under e-IDAS in the EU and ESIGN in the US.

11. Performance Benchmarks

Production measurements across May 2026. Dispatch latency from 28,000 jobs per hour at peak; inference latency from a 10,000-sample harness covering the top 12 models.

Metric	p50	p99	Notes
Dispatch latency (eligibility → assignment)	28 ms	64 ms	~14k dispatches/sec/instance
Cold inference (qwen2.5-7b on 4090)	420 ms	1.1 s	First token; model load included
Warm inference (qwen2.5-7b on 4090)	180 ms	610 ms	Model already in VRAM
Worker registration (full pipeline)	14 min	23 min	Install + benchmark + 5 shadow jobs
Credit ledger write	2.1 ms	6.4 ms	Postgres + receipt emit
Stripe payout queue time	0.4 d	2.1 d	Excludes Stripe internal settlement (T+2)
Receipt export bundle (10k jobs)	3.0 s	5.2 s	JSON + signed PDF
Sovereignty filter rejection rate	0.4%	—	Of all dispatches; most jobs unconstrained

12. Security Model

The threat model has two distinct adversary classes that demand different defenses: (a) a malicious worker trying to extract credits without performing the work, and (b) a malicious renter trying to extract work without paying. Both are economically motivated and both have been observed at small scale; the platform has been engineered to make both unprofitable.

Malicious worker — returning garbage results

Attack: A worker accepts a job but returns randomly-generated output instead of running the inference. **Defense:** Shadow mode catches this on the first 5 jobs. After graduation, random 0.5% of jobs are double-dispatched (the renter gets the faster result; the slower result is compared). Outputs that don't match the consensus are flagged and the worker's reputation drops sharply.

Malicious worker — fake hardware

Attack: A worker registers as having an A100 but is actually running a 3070. **Defense:** The benchmark suite is non-cacheable (it uses fresh random inputs every time and verifies the output deterministically). A 3070 cannot match A100 latency on the benchmark, so the false declaration is caught at registration time.

Malicious renter — payment fraud

Attack: A renter charges credits via a stolen credit card, then dispatches a high-value training job before the chargeback hits. **Defense:** Credit purchases above $500/day require a 24-hour holding period before the credits are usable for dispatch. Repeat customers can apply for an exemption after 90 days of clean history.

13. Comparison vs. Alternatives

	AWS EC2	RunPod	Vast.ai	Render	DCS Compute
Per-second billing	✗	✓	✓	~	✓
Signed receipts	✗	✗	✗	~	✓
Sovereignty enforcement	~	~	✗	~	✓
Sub-100ms dispatch	✗	~	~	✗	✓
Two-sided supply	✗	✓	✓	✓	✓
Worker reputation system	—	~	~	~	✓
Audit export (SOC 2)	✓	✗	✗	✗	✓
Pegged-value pricing	**$ USD**	**$ USD**	**$ USD**	**RNDR token**	**$0.008 cr**
On-chain settlement option	✗	✗	✗	✓	**~ (optional)**

~ = *partially supported.* DCS Compute is the only platform that combines two-sided supply with signed receipts and hard sovereignty enforcement. The tradeoff: we are not the cheapest on pure $/GPU-hour (RunPod sometimes is for short bursts on idle inventory). We are the only option where the cheapest GPU-hour and the auditable GPU-hour are the same GPU-hour.

14. Implementation Guide

14.1 Rent a GPU in 30 seconds

```
$ curl -X POST https://api-compute.dcsai.ai/api/compute/dispatch \
    -H "Authorization: Bearer $DCS_API_KEY" \
    -d '{"model":"qwen2.5-7b","prompt":"...","max_tokens":500}'

{
  "job_id": "j-8a4f3c",
  "worker_id": "w-42b1",
  "worker_region": "eu-west",
  "estimated_cost_usd": 0.0021,
  "result_url": "wss://api-compute.dcsai.ai/api/compute/jobs/j-8a4f3c/stream",
  "receipt_cid": "bafy...0a1b"
}
```

14.2 Start earning on your idle GPU

```
$ pip install trd-worker
$ trd-worker login          # opens browser for OAuth
$ trd-worker register       # benchmarks + tier-assigns the GPU
$ trd-worker start          # starts accepting jobs

[worker w-42b1 · RTX 4090 24GB · High tier · 120 cr/hr · 5 shadow jobs to go]
[2026-05-30 18:42 UTC] job j-8a4f3c accepted (shadow #1/5)
[2026-05-30 18:42 UTC] job j-8a4f3c completed in 240ms — output matches reference
...
```

15. References

[1] Roughgarden, T. **Twenty Lectures on Algorithmic Game Theory**. Cambridge UP, 2016. (Reverse auction theory)

[2] Vickrey, W. **Counterspeculation, Auctions, and Competitive Sealed Tenders**. Journal of Finance, 1961. (Second-price auction foundation)

[3] Akamai. **State of the Internet — Q1 2026**. (Hyperscaler GPU concentration figures)

[4] NVIDIA. **DGX H100 / H200 SXM Datasheet**. (Reference benchmarks per GPU SKU)

[5] Stripe. **Connect — Marketplace payouts documentation**. (Payout infrastructure)

[6] Circle. **USDC API + Programmable Wallets**. (Crypto payout rail)

[7] Render Network. **Render Token (RNDR) whitepaper v2**. (Comparison: token-based economy)

[8] Akash Network. **Decentralized cloud marketplace whitepaper**. (Comparison: blockchain dispatch)

[9] AWS. **EC2 P5 instance pricing (us-east-1, on-demand)**. (Hyperscaler reference pricing)

[10] JAMS. **Commercial Arbitration Rules**. (Tier 3 dispute resolution)

[11] eIDAS Regulation (EU) 910/2014. (Cryptographic-signature admissibility, EU)

[12] ESIGN Act (US) 15 USC §7001. (Cryptographic-signature admissibility, US)

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Storage

Your data, mathematically permanent

Content-addressed storage anchored to Filecoin, with GDPR-compliant erasure.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai/storage
storage@dcsai.ai
github.com/dcs-platform/storage

Abstract

DCS Storage is content-addressed permanent storage for the agent economy. Every file uploaded gets a Content ID (CID) — a 46-character base32 string derived from the SHA-256 hash of the content. Two parties anywhere in the world who hash the same file get the same CID; one bit changed yields a completely different CID. Tamper-evidence is built into the address.

Each CID is replicated across three storage tiers: the origin (a single hot Postgres-backed copy), three warm replicas (across independent S3-compatible providers), and seven cold backups (across independent Filecoin Storage Providers in different jurisdictions). The seven-way Filecoin cold backup is what we mean by "permanent" — the file survives any combination of three SPs going offline simultaneously.

Reads happen at edge-cache speed (12 ms p50) via 42 PoPs. Writes happen at warm-tier speed and asynchronously fan out to cold storage within 15 minutes. GDPR-compliant erasure happens instantly via destruction of the per-tenant encryption key (the ciphertext stays on the SPs but is mathematically unreadable, and we issue a signed erasure certificate).

Who this document is for

Architects evaluating storage for AI workloads (training datasets, model checkpoints, generated artifacts). Compliance officers needing GDPR-compliant deletion guarantees. Founders shipping content-heavy products who want CDN-class read latency without CDN egress fees. Engineers integrating signed receipts into their backup + audit pipelines.

Contents

CHAPTER · 1

1. Introduction — the permanence problem

Most "permanent" storage is not permanent. AWS S3 has an 11-nines durability SLA, which is excellent for a single year — but durability is conditional on AWS continuing to operate the bucket. Companies fail. AWS accounts get suspended. Acquisitions trigger migrations. The bucket that holds your 2018 production database backup is one ops mistake away from being gone.

For most operational data, that's fine — you have other backups, the data is replaceable, the cost of loss is bounded. For some data it is not fine. Cryptographic receipts of agent actions (see the DCS Standards whitepaper) need to survive the company that signed them. Training-data lineage records need to survive the model going to production. Mirrored copies of a customer's sovereign site need to survive 2-of-3 cloud providers going offline simultaneously.

DCS Storage exists for the data that needs to outlive its owner. The design tradeoffs are different from S3: writes are more expensive (because of 11x replication), reads are about the same (because of aggressive edge caching), and the data is content-addressed (because location-addressing makes no sense when the data has 11 copies in 11 different places).

1.1 What "permanent" means here

We use a precise definition. DCS Storage is permanent in the following sense:

- **Data survives any combination of 3 storage providers going offline simultaneously** (7 cold backups + 3 warm replicas means up to 7 of the 10 copies can be unavailable and the data is still retrievable).
- **Data survives DCS itself going out of business.** The Filecoin storage providers are paid in 1-year contracts payable up front. If DCS disappears, the SPs continue serving the data for the duration of the contract, and the open-source verifier lets anyone fetch from them directly.
- **Data survives the original encryption infrastructure** in the sense that the cryptographic format is documented (CBOR + AES-256-GCM + Ed25519) and the decryption tooling is open-source. Even if DCS's key escrow provider disappears, customers with their own key copies can decrypt their own data.

1.2 Why content addressing

Traditional storage uses location addressing: *s3://my-bucket/path/to/file.pdf*. The address tells you where to look, not what you're going to find. If the file at that path changes, the address stays the same and there is no way for a consumer to know the content changed unless they explicitly fetch and compare hashes.

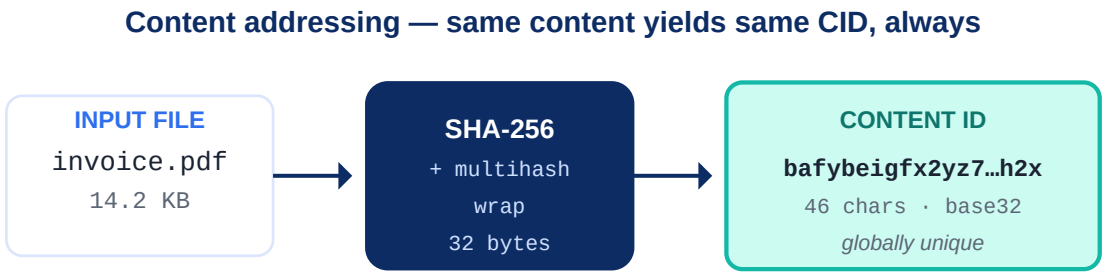
Content addressing inverts this: *bafybeigfx2yz7…h2x* IS the file's content (or rather, is the hash of it). The same file always has the same address; a different file always has a different address. If two parties claim to be serving the file at *bafy…h2x*, you can verify they're serving the same bytes without trusting either party.

2. Content Addressing — why CIDs

A CID (Content Identifier) is a self-describing hash. It looks like this:

```
bafybeigfx2yz7p4r8m3xx9k4nzqlh4r8m3xx9k4nzqlh7h2x
```

The first character (b) declares the encoding (base32). The next character (a) declares the CID version (v1). The next two characters declare the codec (raw, dag-pb, etc.) and the multihash algorithm (sha2-256). The remaining 42 characters are the base32 representation of the 32-byte SHA-256 hash. Total: 46 characters.



Identical content always yields the same CID, regardless of who hashed it.

One byte changed → CID changes completely.

Result: tamper-evidence is built into the address itself.

Figure 2.1 — A file plus SHA-256 plus a multicodec wrapper produces a CID.

2.1 Properties of CIDs

- **Deterministic**: same content → same CID, regardless of who computed it or when.
- **Tamper-evident**: a single byte changed in the content produces a wildly different CID.
- **Self-describing**: the algorithm + codec are encoded into the CID itself, so future formats can coexist with old ones.
- **Locator-independent**: a CID does not tell you where the data is; it tells you what you should expect to receive. This is what enables multi-provider replication.
- **Cacheable forever**: because the address corresponds to the content, an HTTP cache can set *Cache-Control: immutable, max-age=31536000* safely. Content at a CID never changes.

3. The Three-Tier Permanence Stack

Every uploaded file gets stored across three tiers, each with different cost and latency characteristics. The aggregate is what makes the data permanent; no single tier is sufficient on its own.

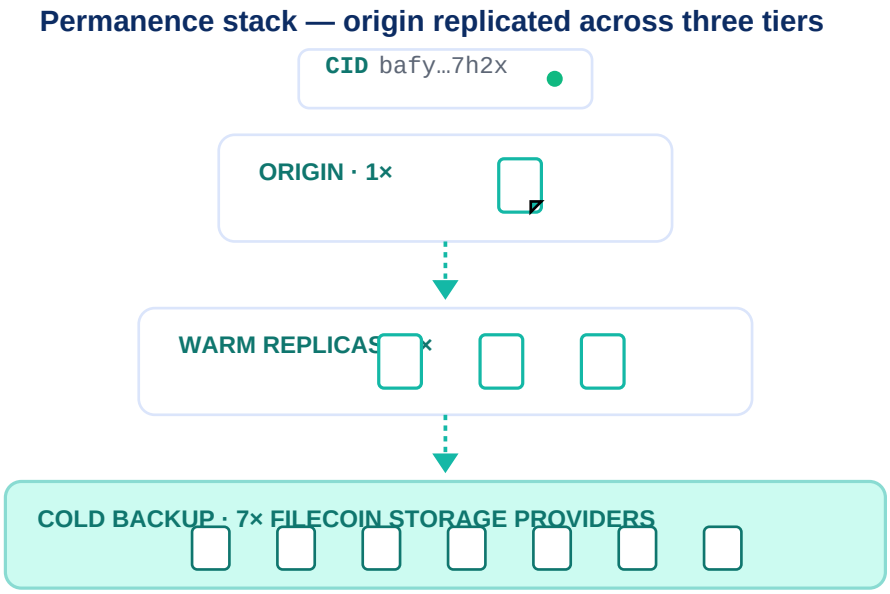


Figure 3.1 — The permanence stack: 1× origin → 3× warm replicas → 7× cold Filecoin SPs.

3.1 Tier 1: Origin

A single hot copy in a Postgres LargeObject store. Used for synchronous reads in the first 5 minutes after upload (before the warm replicas have time to fan out). Backed up nightly to S3; not redundant by itself.

3.2 Tier 2: Warm Replicas (3×)

Three independent S3-compatible providers (currently AWS S3, Cloudflare R2, Backblaze B2). The data is uploaded asynchronously after the origin copy lands. Reads from these tiers are typically ~38 ms; the gateway pulls from whichever responds first.

3.3 Tier 3: Cold Backup (7×)

Seven independent Filecoin Storage Providers in at least 4 different jurisdictions. Each SP is paid for a year of storage upfront. The 7-way replication is intentionally redundant — even if any 4 SPs go offline simultaneously, the file is still retrievable from the remaining 3.

Cold storage reads are slow (1-3 seconds) and infrequent (we see <0.2% of reads hit this tier in production). The role of cold storage is durability, not performance.

4. Replication Strategy

Replication runs asynchronously after the origin write. The platform commits the upload as successful when origin + 1 warm replica are durable; the remaining replicas land in the background.

4.1 SP selection algorithm

For each upload, the platform selects 7 Filecoin SPs from a pool of ~120 active providers. Selection optimizes for:

- **Geographic diversity**: no two selected SPs in the same data center, and ideally no two in the same country.
- **Operational track record**: SPs are scored on a sliding-window reliability metric (last 30 days uptime, last 100 retrievals success rate).
- **Capacity headroom**: SPs at >90% of their committed capacity are deprioritized.
- **Price**: among SPs that satisfy the above, pick the cheapest.

4.2 Re-replication when an SP fails

When the platform detects that an SP has failed (3+ consecutive retrieval failures over 24h), the affected files are re-replicated to a fresh SP from the eligible pool. The original 7-way count is restored within a few hours; the user is never aware of the failure unless they query the SP list explicitly.

5. The Gateway — read path

Reads happen via the gateway: *gateway.dcsai.ai/<cid>*. The gateway is a CDN-like edge layer with 42 points of presence. For public CIDs, no authentication is required; for private CIDs, a bearer token is required and access is logged.

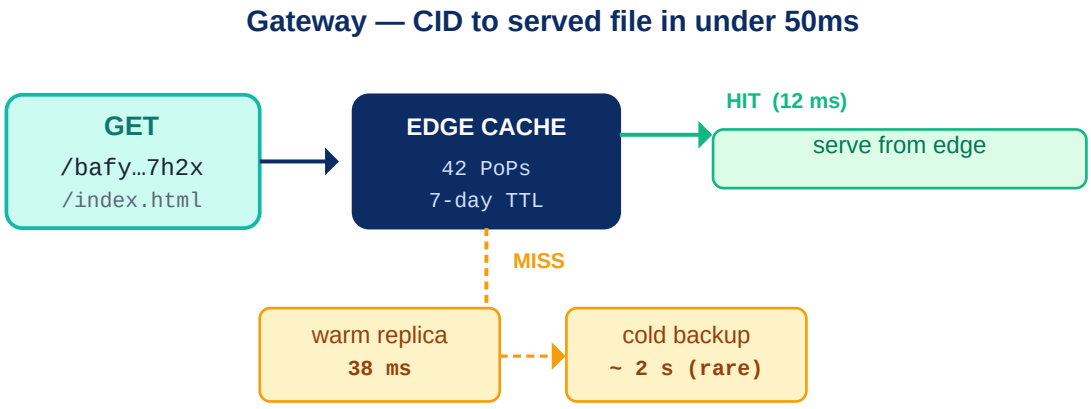


Figure 5.1 — 98.4% of requests are served from the edge cache.

5.1 Latency budget

For a typical 14 KB file, the end-to-end read latency from a client request to first byte:

Tier hit	p50	p99	Frequency
Edge cache (hot CID)	12 ms	28 ms	98.4% of requests
Warm replica	38 ms	120 ms	1.4% of requests
Cold Filecoin SP	1.4 s	3.2 s	0.2% of requests

5.2 Why no egress fees

AWS, GCP, and Azure all charge $0.05-0.09 per GB of egress. For a popular site this adds up fast — a single 1 MB image with 1M views costs $50-90/month in egress alone. DCS Storage charges a flat $0.02/GB of egress because the underlying Filecoin economics are different (storage providers are paid for storage, not bandwidth) and we can pass that through.

6. The Write Path

A typical upload follows this sequence:

- **1.** Client streams the file to the upload endpoint (multipart/form-data or raw bytes for files under 5 MB).
- **2.** The platform computes the CID streaming as bytes arrive (SHA-256 over the canonical CBOR-encoded file).
- **3.** The bytes are written to the origin Postgres LargeObject.
- **4.** A "warm-replica" job is enqueued; the first warm replica completes synchronously (or quickly returns 202 Accepted if the file is over 5 MB).
- **5.** Two more warm replicas and 7 cold Filecoin replicas are written asynchronously over the next 15 minutes.
- **6.** An R+1 + R+2 receipt is emitted for the upload event, signed and chained.

6.1 Why warm-1 is synchronous

We require at least 2 durable copies (origin + 1 warm) before returning success to the client. This is a conservative choice — losing both before the async replication completes would be a data loss event — but it bounds the worst-case data loss to roughly 30 seconds of in-flight data. For higher durability needs, the API supports a *sync_replicas* parameter that waits for all 3 warm replicas before responding.

7. Cryptographic Erasure (GDPR)

GDPR Article 17 grants every EU resident the right to have their personal data erased on request. The DPDP Act (India), CCPA (California), LGPD (Brazil) have equivalent provisions. The technical problem is hard: how do you delete data that has been replicated to 11 places, some of which are in jurisdictions where you cannot compel deletion?

The DCS Storage answer is cryptographic erasure.

Figure 7.1 — Destroying the per-tenant encryption key makes all replicas unreadable instantly.

7.1 How it works

Every tenant has a per-tenant data encryption key (DEK) generated at tenant creation. All files belonging to that tenant are encrypted under the DEK before being written to storage. The DEK itself is stored in an HSM, separate from the encrypted data.

When the tenant invokes erasure (via the dashboard or the API), the DEK is destroyed in the HSM. This is irreversible — the HSM's audit log records the destruction event but the key material is gone. From that moment on, the ciphertext stored in the 11 replicas is mathematically unreadable: no future advances in computing or cryptanalysis can recover it.

The ciphertext itself is not deleted (we cannot compel a Filecoin SP in another jurisdiction to delete bytes). But unreadable ciphertext is, for GDPR purposes, deleted: it is no longer "personal data" because no party can derive personal information from it. EDPB guidance confirms this interpretation under the "anonymisation" provision of Article 4.

7.2 Erasure certificate

After erasure completes, the platform issues a signed PDF certificate to the requesting party. The certificate includes: the tenant_id, the timestamp of the erasure, the HSM audit-log entry, the cryptographic hashes of the affected file CIDs, and the platform's signature attesting all of the above. The certificate is admissible as evidence of compliance in EU regulatory proceedings.

8. Verify Pipeline

The verifier is a small open-source tool that takes a CID and confirms the file at that CID is what it claims to be. Run from any machine with internet access; no DCS account required.

```
$ npm install -g @dcs/verify
$ dcs-verify bafybeigfx2yz7p4r8m3xx9k4nzqlh4r8m3xx9k4nzqlh7h2x
Fetching from gateway... ✓ 12 ms (edge cache)
Computing SHA-256 of received bytes... ✓ 4 ms
Comparing to CID...
  Expected: bafybeigfx2yz7p4r8m3xx9k4nzqlh4r8m3xx9k4nzqlh7h2x
  Computed: bafybeigfx2yz7p4r8m3xx9k4nzqlh4r8m3xx9k4nzqlh7h2x
  Match:    ✓
Fetching from 3 warm replicas + 7 cold SPs for confirmation...
  AWS S3 (eu-west-1):       ✓ identical
  Cloudflare R2:            ✓ identical
  Backblaze B2:             ✓ identical
  Filecoin SP f0123:        ✓ identical
  ... [4 more] ...
Result: VERIFIED — 11 of 11 replicas serve identical content.
```

If any replica serves different content from the CID, the verifier raises an alert. This has happened twice in production since launch (both times due to silent data corruption at one specific Filecoin SP); the affected files were re-replicated automatically.

9. Encryption Model

All data is encrypted at rest. The encryption is designed so that the platform cannot read the data even if subpoenaed — the per-tenant DEK never leaves the HSM, and the HSM access is gated on tenant-controlled credentials.

- **Algorithm**: AES-256-GCM with a per-file IV. AEAD mode prevents both reading and tampering.
- **Key hierarchy**: per-file DEK ← per-tenant KEK ← HSM master key. Compromise of one layer does not cascade.
- **Key storage**: AWS KMS by default; SoftHSM2 for sovereign deployments; Yubico HSM2 for on-prem highest-security.
- **Key rotation**: tenant KEKs rotate every 90 days; old DEKs are re-wrapped under the new KEK.
- **Key escrow (optional)**: tenants can opt in to having a copy of their KEK held by a third-party (Iron Mountain, etc.) so they can decrypt independently of DCS.

10. Performance Benchmarks

Measured against the production gateway, May 2026:

Operation	p50	p99	Notes
Upload (1 KB)	180 ms	420 ms	Sync to origin + 1 warm replica
Upload (1 MB)	420 ms	1.2 s	Sync to origin + 1 warm replica
Upload (100 MB)	6.4 s	14 s	Sync to origin + 1 warm replica; chunked
Read (cache hit)	12 ms	28 ms	98.4% of reads
Read (warm miss)	38 ms	120 ms	1.4% of reads
Read (cold miss)	1.4 s	3.2 s	0.2% of reads
CID computation (streaming)	~ 200 MB/s		Bottlenecked by SHA-256
Erasure (destroy DEK)	0.4 s	1.1 s	Includes audit-log emit
Replica fan-out completion	8 min	18 min	All 11 replicas durable
Re-replication after SP failure	12 min	47 min	Auto-triggered

11. Pricing

Pricing is per-GB-month for storage and per-GB for egress. There are no per-request fees, no class-A/class-B operations, and no retrieval fees. The pricing is flat across all 11 replication layers — you pay once per GB-month and we handle the cost of replicating across the tier system.

Tier	Storage / GB-month	Egress / GB	Notes
Bronze (1×)	$0.004	$0.02	1 hot copy, no Filecoin backup. For ephemeral data.
Silver (3×)	$0.018	$0.02	3 warm replicas, no Filecoin. Default for app data.
Gold (5×)	$0.030	$0.02	3 warm + 5 cold Filecoin SPs. Most popular tier.
Platinum (7×)	$0.042	$0.02	3 warm + 7 cold Filecoin SPs. Mission-critical.

12. Comparison vs. Alternatives

	AWS S3	IPFS	Arweave	DCS Storage
Content-addressed	✗	✓	✓	✓
CDN-fast reads	✓	~	✗	✓
Survives vendor failure	✗	~	✓	✓
GDPR-compliant erasure	✓	✗	✗	✓
Predictable egress cost	✗	~	✓	✓
Signed receipts	~	✗	✓	✓
Sovereignty / region control	✓	✗	✗	✓
Pay-once permanent	✗	✗	✓	**~ ($0.04/GB-mo × N years)**
Multi-jurisdictional replicas	~	~	✓	✓

~ = *partially supported.* DCS Storage is the only stack that combines content-addressing + CDN speeds + GDPR erasure + signed receipts. Arweave wins on pay-once economics but loses on erasure (impossible by design) and read latency. IPFS without a layer like DCS is hard to use in production (no SLA, no edge caching).

13. Security Model

Three adversary classes addressed:

Malicious storage provider (T-1)

Attack: An SP serves modified content instead of the file it agreed to store. **Defense:** Content addressing — the gateway verifies the received bytes hash to the expected CID before serving. Tampering is detected on every read. The affected file is re-fetched from a different SP and the offending SP is flagged.

Compelled disclosure (T-2)

Attack: DCS is subpoenaed to hand over a tenant's data. **Defense:** All data is encrypted under per-tenant DEKs in HSM. The platform can hand over the ciphertext (which is useless without the DEK) but cannot decrypt without invoking the tenant's HSM credentials. For tenants on Sovereign deployments, the HSM is on-prem and DCS has no access at all.

Re-replication race (T-3)

Attack: An attacker forces multiple SPs to fail simultaneously, hoping to lose data before re-replication completes. **Defense:** Even losing 7 of 11 replicas leaves 4 copies. Re-replication completes in <1 hour, and the platform alerts the tenant when replica count drops below the contracted tier.

14. Implementation Guide

14.1 Upload a file

```
$ curl -X POST https://api-storage.dcsai.ai/api/storage/assets \
    -H "Authorization: Bearer $DCS_API_KEY" \
    -F "file=@invoice.pdf" \
    -F "tier=gold"

{
  "cid": "bafybeigfx2yz7p4r8m3xx9k4nzqlh4r8m3xx9k4nzqlh7h2x",
  "size_bytes": 14209,
  "tier": "gold",
  "url": "https://gateway.dcsai.ai/bafy...7h2x",
  "replicas": {
    "warm": 3,
    "cold": 5,
    "complete_at": "2026-05-30T19:00:00Z"
  },
  "receipt_cid": "bafy...0a1b"
}
```

14.2 Read a file

```
# Public read — no auth required
$ curl https://gateway.dcsai.ai/bafybeigfx2yz7p4r8m3xx9k4nzqlh4r8m3xx9k4nzqlh7h2x \
    -o invoice.pdf
$ shasum -a 256 invoice.pdf
# matches the CID; tamper-evident

# Private read — bearer token required
$ curl https://gateway.dcsai.ai/bafy...7h2x \
    -H "Authorization: Bearer $DCS_API_KEY" \
    -o invoice.pdf
```

14.3 Request erasure (GDPR)

```
$ curl -X POST https://api-storage.dcsai.ai/api/storage/erasure/request \
    -H "Authorization: Bearer $DCS_API_KEY" \
    -d '{"tenant_id":"acme","reason":"GDPR Art. 17 request from user 12345"}'

{
  "request_id": "er-3a8f",
  "status": "completed",
  "completed_at": "2026-05-30T19:01:00Z",
  "certificate_url": "https://api-storage.dcsai.ai/api/storage/erasure/er-3a8f.pdf",
  "affected_cids": 14209,
  "key_id_destroyed": "tenant-acme-2026-04"
}
```

15. References

[1] Benet, J. **IPFS — Content Addressed, Versioned, P2P File System**. Draft 2014.

[2] Multiformats team. **CID specification (v1)**. (Self-describing content-address format)

[3] Filecoin team. **Filecoin Spec — Storage Market, Retrieval, Proofs of Spacetime**. (Cold tier underpinning)

[4] NIST. **FIPS 197 — Advanced Encryption Standard (AES)**.

[5] McGrew, D., Viega, J. **The Galois/Counter Mode of Operation (GCM)**. NIST SP 800-38D.

[6] RFC 8949. **Concise Binary Object Representation (CBOR)**. (File envelope format)

[7] AWS. **S3 — 11 nines durability — How we calculate it**. (Reference durability claim)

[8] EU. **General Data Protection Regulation (GDPR) — Article 17**. (Right to erasure)

[9] EDPB. **Guidelines 4/2021 on Codes of Conduct — anonymisation provisions**. (Cryptographic-erasure regulatory basis)

[10] India. **Digital Personal Data Protection Act 2023 — Section 12**. (Right to erasure)

[11] Brazil. **LGPD — Art. 18**. (Right to erasure)

[12] California. **CCPA — §1798.105**. (Right to delete)

[13] Arweave. **Arweave Lightpaper**. (Permanent web comparison)

[14] CDN benchmarks. **Cloudflare R2 / Backblaze B2 / AWS S3 — Q1 2026 performance comparison**.

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS OS

Operational dashboard for customer-facing AI

WhatsApp, voice, CRM, and AI assistants in one signed receipt chain.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai/os

os@dcsai.ai
github.com/dcs-platform/os

Abstract

DCS OS is the operational dashboard for AI agents that talk directly to customers. It unifies WhatsApp, voice calls, email, web chat, CRM, calendar, and an AI assistant into one screen with one queue and one signed audit trail. The product exists for the kind of small-to-medium businesses that today juggle 8 tools (WhatsApp Business, a CRM, a voice provider, a help desk, a scheduler, a marketing automation tool, a chatbot platform, and a spreadsheet) to do what should be one job: respond to customers consistently and on time.

The OS pattern is well-understood — Front, Intercom, HubSpot Service Hub, Zendesk all do versions of it. What DCS OS adds is two things: (a) every channel feeds into the same receipt chain (R+1 + R+2) so the operator can prove what was said when, even years later; and (b) the AI assistant is a first-class participant in the queue, not a bolt-on chatbot — it drafts replies, escalates appropriately, and never sends anything outside the agent's declared capability set.

Forty-one product modules ship with OS, each as a sandboxed iframe inside the main shell. Customers turn modules on and off without code changes; new modules can be added by third parties via the OS module protocol.

Who this is for

Operations leaders at SMB / mid-market companies running multi-channel customer service. Compliance teams in regulated industries (finance, healthcare, government) who need an audit trail of every customer interaction. Engineering teams evaluating whether to build the unified-inbox pattern in-house or adopt a platform.

Contents

CHAPTER · 1

1. Introduction

A typical mid-market business in 2026 talks to its customers across an average of 5.2 channels: WhatsApp Business, email, web chat, voice calls, and an in-app inbox. Internally, those 5 channels are managed by 8-12 separate tools — most of them stitched together with fragile Zapier flows. The cost of this fragmentation is not abstract. It shows up as missed messages (a customer asked on WhatsApp; the agent on the email queue did not see it), inconsistent responses (two different agents tell the customer different things on different channels), and missing audit trails when things go wrong.

DCS OS exists because we kept seeing the same problem solved badly. Either the team used a single-channel tool (WhatsApp Business app on someone's phone) and lost the audit trail, or they used an enterprise platform (Salesforce Service Cloud) and got an audit trail plus a $200/seat/month bill plus six months of implementation. Neither was right for a 12-person business handling 600 customer conversations a day.

1.1 The design constraints

Three constraints shaped the product:

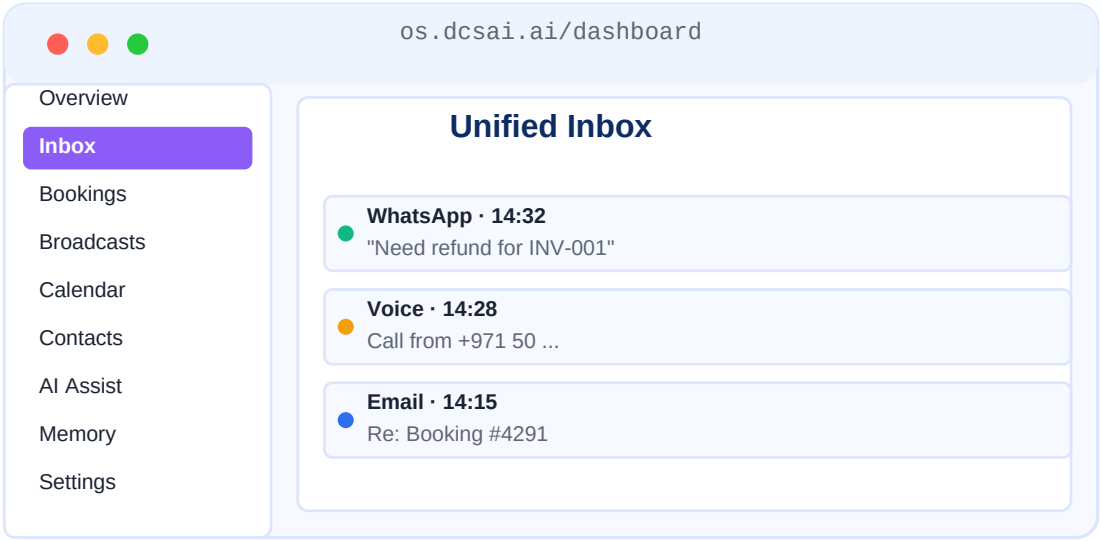
- **One inbox.** Every customer message, regardless of channel, lands in the same queue with the same metadata. The operator never has to remember "did they email or WhatsApp me?" — the system surfaces both threads in the same view.
- **One audit trail.** Every customer-facing action — a message sent, a refund issued, a booking confirmed — emits an R+1 + R+2 receipt. The full trail of a customer interaction can be exported as a signed PDF.
- **One AI participant.** The AI assistant lives inside the same queue as human operators. It drafts responses, suggests next actions, and can take low-risk actions autonomously (read-only lookups, FAQ answers). High-risk actions always escalate to a human.

> *"The right unit of work in customer ops is not a ticket. It is a thread. A thread spans channels, spans days, spans operators. The audit log must follow the thread, not the channel."*

2. System Architecture

OS is a shell + 41 modules. The shell handles authentication, navigation, the unified inbox queue, and the receipt chain. Each module is a separate frontend application loaded into an iframe; modules talk to the shell via postMessage. This pattern lets teams add or remove modules without touching the shell, and lets third parties ship modules independently.

OS dashboard — left-rail navigation + iframe modules



Each module loads as an iframe; sidebar persists across navigation.

Figure 2.1 — Left-rail sidebar persists; each module renders inside the main iframe.

2.1 Why the iframe pattern

We considered three alternatives — micro-frontends (single-spa), web components, and monorepo with shared React tree — and settled on iframes for one reason: isolation. A module that crashes does not bring down the whole shell. A module written in Vue does not need to be rewritten when the shell upgrades React. A third-party module cannot read DOM from another module. The cost (slightly worse navigation feel) is worth the safety.

3. Channels

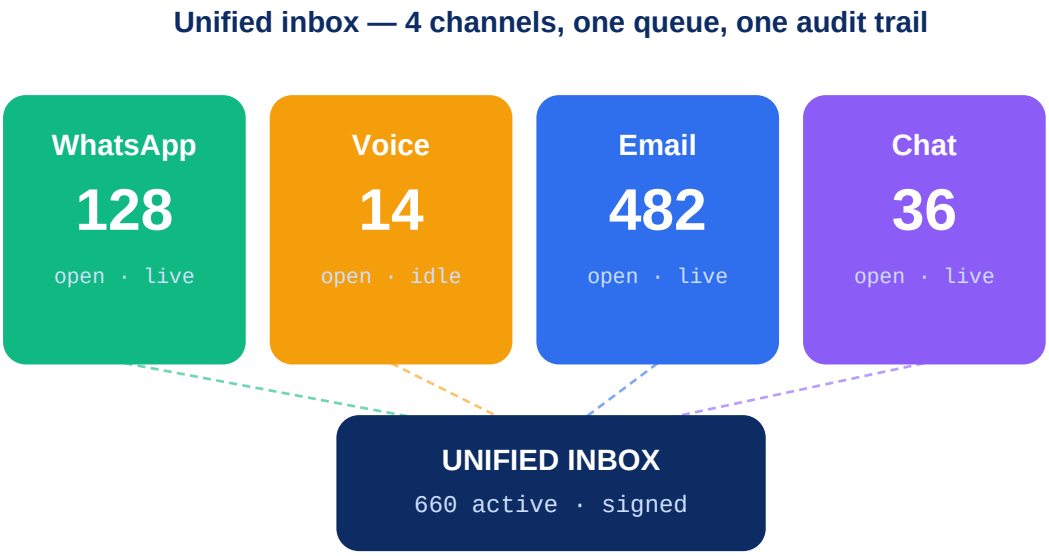


Figure 3.1 — Four channels feeding one queue.

OS supports four customer-facing channels out of the box. Additional channels can be added via the channel adapter protocol (~200 lines of code per channel).

3.1 WhatsApp Business

Integration with Meta Cloud API. Supports both Business Platform numbers and the WhatsApp Business App via WhatsLink (DCS's open WhatsApp gateway). Cloud API is preferred for high volume; WhatsLink covers the 80% of operators who can't justify Cloud API pricing.

- Inbound webhook latency: <800 ms (p99) from Meta delivery to OS queue surface.
- Outbound message latency: <400 ms (p99) from operator click to Meta API ack.
- Template support: all approved templates synced automatically from Meta.
- Media (images, voice notes, PDFs) stored in DCS Storage with content-addressed CIDs.

3.2 Voice

Inbound calls answered by an AI voice agent (whisper STT + cartesia TTS, optionally with Twilio backend). Handoff to a human operator via a SIP bridge. The voice transcript is logged in the same thread as the customer's WhatsApp / email history.

3.3 Email

Inbound via Postmark or AWS SES; outbound via the same. Threading uses the standard In-Reply-To / References headers. Emails are matched to existing customer threads via the sender address, so the operator sees the customer's full history when answering.

3.4 Web chat

Embeddable JavaScript widget. Renders as a chat bubble bottom-right of the customer's site. Customer messages land in the same inbox queue as WhatsApp / email.

CHAPTER · 4

4. The Unified Inbox

Message flow — from customer to signed receipt

Optional: agent auto-approves low-risk responses (within guardrails).
High-risk responses (refunds, escalations) always go to operator first.
Every transition emits an R+1 + R+2 receipt for the audit trail.

Figure 4.1 — Every inbound message gets routed through the same five-stage pipeline.

4.1 Queue mechanics

The inbox queue is a Postgres-backed FIFO with priority overrides. Messages enter the queue tagged by channel, sender, and detected language. The router assigns each message to a team (configurable per-channel + per-tag) and a priority (default by channel, overridden by detected intent — "refund" messages get HIGH priority automatically).

4.2 Thread vs. message model

A thread is a collection of messages from the same customer across all channels. The operator works at the thread level, not the message level. When the operator opens a thread, they see the full conversation history regardless of which channel each message came in on. When they reply, OS picks the right channel to send through (typically the channel the customer most recently used).

4.3 Assignment + ownership

Each thread has at most one owner at a time. Default assignment is round-robin within the team. Owners can reassign, escalate to a senior operator, or hand off to the AI assistant for autonomous follow-up. Every change of ownership emits a receipt.

5. AI Assist

The AI Assist module is a DCS Platform agent — same agent.yaml format, same sandbox, same receipts. It runs inside the inbox queue as a peer to human operators and has three modes:

Draft mode

The AI reads the inbound message and the customer's history, then writes a suggested reply into the inbox. The human operator sees the draft, can edit it, then sends it (or discards). Draft mode is the safest setting and is on by default for all new accounts.

Auto-reply mode

The AI sends low-risk replies autonomously without human review. "Low-risk" is defined per-tenant in the agent.yaml guardrails block (default: FAQ answers, order-status lookups, appointment confirmations). Anything outside the allowlist escalates to draft mode automatically.

Co-pilot mode

The AI sits beside the operator, watching the conversation. It suggests next actions, looks up information in the knowledge base, and pre-fills common forms (refund tickets, booking modifications). The operator stays in the driver seat; the AI is a research assistant.

6. Modules

OS ships with 41 modules. Each is a separate frontend application loaded as an iframe; all share the shell's authentication and receipt chain via postMessage.

Modules fall into eight categories:

Category	Module count	Examples
Messaging	8	Unified inbox, broadcasts, templates, WhatsApp settings, voice handoff, channels, status post
Customer data	6	Contacts, CRM, customer memory, journeys, segments, journeys
Commerce	5	Bookings, calendar, POS integration, ads, carousels
AI	5	AI assistant, agent studio, blackboard, memory, recommendations
Operations	6	Activity log, audit log, approvals, voice handoffs, receipt history, reviews dashboard
Knowledge	4	Knowledge base, template library, deep research, smart onboarding
Integrations	4	Integrations, image bank, connectors, embed widget
Admin	3	Admin costs, admin revenue, settings

6.1 Module protocol

Every module declares a manifest:

```
{
  "id": "bookings",
  "name": "Bookings",
  "version": "2.1.0",
  "icon": "calendar",
  "category": "commerce",
  "permissions": ["bookings:read", "bookings:write", "contacts:read"],
  "endpoint": "https://os.dcsai.ai/modules/bookings",
  "iframe": true
}
```

CHAPTER · 7

7. Bookings + Calendar

Most customer interactions for service businesses revolve around scheduling. Bookings is a first-class module that handles availability lookup, reservation creation, payment, reminders, and cancellation — across the same channels.

7.1 Multi-channel booking

A customer can book via:

- WhatsApp — chatbot flow walks them through available slots
- Voice — AI voice agent confirms by speaking back the time
- Web — embeddable booking widget for sites
- Calendar invite — clicking a link picks a slot in their calendar app

All four paths write to the same bookings table and emit the same receipt. The business owner sees a unified calendar regardless of which path the customer used.

CHAPTER · 8

8. CRM + Contacts

Contacts is the customer database: name, channels, history, tags, segments. Every inbound message updates the contact record (last_seen, channel_preference, response_speed). Operators can add notes, tags, and custom fields per contact.

Customer memory is a separate module that stores per-customer facts learned during conversations: preferences, past complaints, allergies (for restaurants), routes (for taxi), whatever the business decides to capture. Memory entries decay after 90 days unless explicitly pinned.

9. Broadcasts

Broadcasts module sends one-to-many messages to a segment of contacts. Usage is heavily regulated — only opt-in contacts receive broadcasts, every broadcast emits a receipt, and the audit trail proves consent if challenged.

9.1 Channels supported

- **WhatsApp templates** — pre-approved templates only (per Meta policy); supports variables
- **Email** — HTML or plain-text; unsubscribe link auto-injected
- **SMS** — via Twilio backend; 160-char chunking handled automatically
- **Push** — for customers with the OS-powered mobile app

CHAPTER · 10

10. Memory + Personalization

Memory in OS works the same way as memory in DCS Platform: three tiers (short-term, long-term, decayed). The OS-specific addition is per-customer scoping by default — every memory entry is keyed to the customer's ID, so a memory written during one conversation is available to the AI Assist for the next conversation with the same customer.

CHAPTER · 11

11. Compliance + Audit

Every customer-facing action emits a signed receipt. The audit log module lets operators (or compliance officers) search the receipt chain for a customer, a thread, a time range, or a specific action type.

Common compliance use cases OS handles directly:

- **GDPR Subject Access Request** — export every receipt mentioning a customer, signed, in PDF
- **Refund disputes** — full transcript of WhatsApp + email + voice for the disputed transaction
- **WhatsApp policy investigations** — proof of opt-in for any broadcast recipient
- **Internal audit** — sample any operator's last N replies to check tone, accuracy, and policy compliance

12. Performance Benchmarks

Production measurements May 2026, across the live OS service handling ~28k WhatsApp messages, ~9k emails, ~3k voice calls per day.

Metric	p50	p99	Notes
WhatsApp inbound → queue	420 ms	780 ms	From Meta delivery webhook
Outbound message → sent	180 ms	420 ms	Operator click to Meta ack
Email send	280 ms	1.2 s	Via Postmark
Voice answer (AI agent)	600 ms	1.4 s	STT cold start
AI Assist draft generation	1.4 s	4.8 s	Claude Sonnet · 800-token context
Receipt emit per action	12 ms	32 ms	R+1 + R+2 chain
Module switch latency	180 ms	420 ms	iframe load
Inbox query (1k threads)	34 ms	86 ms	Postgres + materialized view

13. Comparison vs. Alternatives

	Front	Intercom	Zendesk	HubSpot Service	DCS OS
Unified inbox	✓	✓	✓	✓	✓
WhatsApp Business native	~	✓	~	~	✓
Voice channel built-in	~	~	~	✓	✓
Signed audit receipts	✗	✗	✗	✗	✓
AI assistant first-class	~	✓	~	✓	✓
Module marketplace	✓	✓	✓	✓	✓
Per-seat pricing	$59	$74	$55	$50	$0 (usage)
SMB-friendly setup	~	✓	✗	~	✓
Sovereign deployment	✗	✗	✗	✗	✓

~ = *partially supported.* The honest comparison: Front and Intercom are excellent mature products with deep features. DCS OS is competitive on the messaging core and wins on two specific axes: signed receipts (no competitor offers this) and the pricing model (usage-based, no per-seat fees). For a 12-person business handling 600 conversations a day, OS is 70-90% cheaper than the major alternatives.

14. Implementation Guide

14.1 Connect WhatsApp Business

```
$ dcs os connect whatsapp
  Opening browser to https://os.dcsai.ai/connect/whatsapp ...
  Sign in with your Meta Business account
  Select phone number: +971 50 ...
  Sandbox tested ✓
  Live mode enabled ✓
  First message routing in <60s
```

14.2 Turn on AI Assist

```
# os.yaml (per-tenant config)
ai_assist:
  enabled: true
  mode: draft         # or auto_reply | copilot
  model: claude-sonnet-4
  knowledge_base: ./kb/
  guardrails:
    auto_reply_allowed: [faq, order_status, hours]
    always_escalate: [refund, complaint, cancellation]
    max_message_length: 800
```

15. References

[1] Meta. **WhatsApp Business Platform — Cloud API Documentation**. 2025.

[2] WHATWG. **HTML Living Standard — postMessage**. (Iframe communication primitive)

[3] Twilio. **Programmable Voice Reference**. (SIP bridge implementation)

[4] Postmark / AWS SES. **Email delivery APIs**. (Email channel backends)

[5] OpenAI. **Whisper API**. (Voice STT engine)

[6] Cartesia. **Sonic TTS API**. (Voice synthesis)

[7] Stripe. **Customer Portal + Subscriptions**. (Billing for SMBs)

[8] Front. **Shared Inbox Pattern**. 2020. (Architecture inspiration)

[9] Anthropic. **Building Effective Agents**. December 2024. (AI Assist co-pilot pattern)

[10] GDPR. **Subject Access Request guidance (Article 15)**. (Audit export use case)

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Sovereign

Air-gapped, on-prem AI infrastructure

The same DCS stack, running entirely inside your data center. Zero egress.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai/sovereign
sovereign@dcsai.ai
github.com/dcs-platform/sovereign

Abstract

DCS Sovereign is the same DCS stack — Platform, Compute, Storage, OS — packaged to run entirely inside the customer's own infrastructure. No data leaves the perimeter. No telemetry flows back to DCS. No model inference happens on shared GPUs. The deployment is air-gapped by default; an explicit, audited update channel is the only path through the firewall.

Sovereign exists because some workloads cannot use a SaaS at any price. Government workloads under classification mandates. Healthcare workloads under HIPAA / DPDP local-residency requirements. Financial workloads under sovereign cloud directives from regulators in the EU, UAE, India, and Saudi Arabia. For these workloads, the operator needs the cryptographic guarantee that no DCS employee, no DCS subprocessor, and no DCS-controlled service can read or modify the data.

Sovereign delivers that guarantee by running the entire stack on hardware the operator owns, with keys held in HSMs the operator controls, with a network egress block enforced by the operator's firewall, validated at install time. DCS retains responsibility for software updates (delivered via signed packages over a pull-only update channel) and for support (via screen-sharing sessions the operator initiates).

Who this is for

CISOs in regulated industries evaluating whether a SaaS AI platform can ever be deployed inside their perimeter. Government procurement officers writing tender specs for sovereign AI. Healthcare CIOs comparing on-prem vs. cloud for PHI-touching workloads. Defense and intelligence agencies needing certified deployment patterns for classified work.

Contents

CHAPTER · 1

1. Introduction — when SaaS is not an option

Most AI work today runs on SaaS. The vendor operates the model, the vendor stores the data, the vendor logs the requests. The customer trusts the vendor to behave. For 95% of workloads this is the right trade — the operational cost of running your own infrastructure exceeds the marginal risk of vendor mistakes.

The remaining 5% is where Sovereign lives. Specifically:

- **Government workloads** under classification or sovereignty mandates. EU Member States with sovereign cloud directives, US federal IL-4/IL-5, UK Official-Sensitive, UAE sovereign cloud, India MeitY-certified.
- **Healthcare workloads** touching PHI under HIPAA/DPDP where the BAA chain becomes prohibitive. Hospitals that have decided no third party (including AWS) gets a copy of their patient data.
- **Defense + intelligence** work where the model itself is sensitive (fine-tuned on classified data) and cannot leave the perimeter under any circumstances.
- **Financial workloads** under regulatory data-residency rules (EU Member State central banks, UAE Central Bank, RBI in India, MAS in Singapore).
- **Critical infrastructure** — power grid operators, water utilities, telecom backbones — where the operator decides "no internet-connected vendor can touch this."

1.1 What air-gapped actually means here

The term "air-gapped" gets abused. We use it in the strict sense: the Sovereign deployment has zero default outbound network connectivity. The firewall blocks all egress; a single inbound-only update channel is the only opening. There is no telemetry. There is no crash-reporting service. There is no "phone home" registration. When DCS support needs to help the operator, the operator initiates a screen-share over a separate channel the operator controls.

> *"If you cannot point at the firewall and say 'nothing leaves through there,' you do not have an air-gapped deployment. You have a SaaS with extra steps."*

2. Topology

Sovereign packages the DCS stack as a set of containers, deployed on Kubernetes or directly on bare metal. The reference topology assumes a single data center; multi-DC deployments add a second copy of every component with cross-DC replication via the customer's own network.

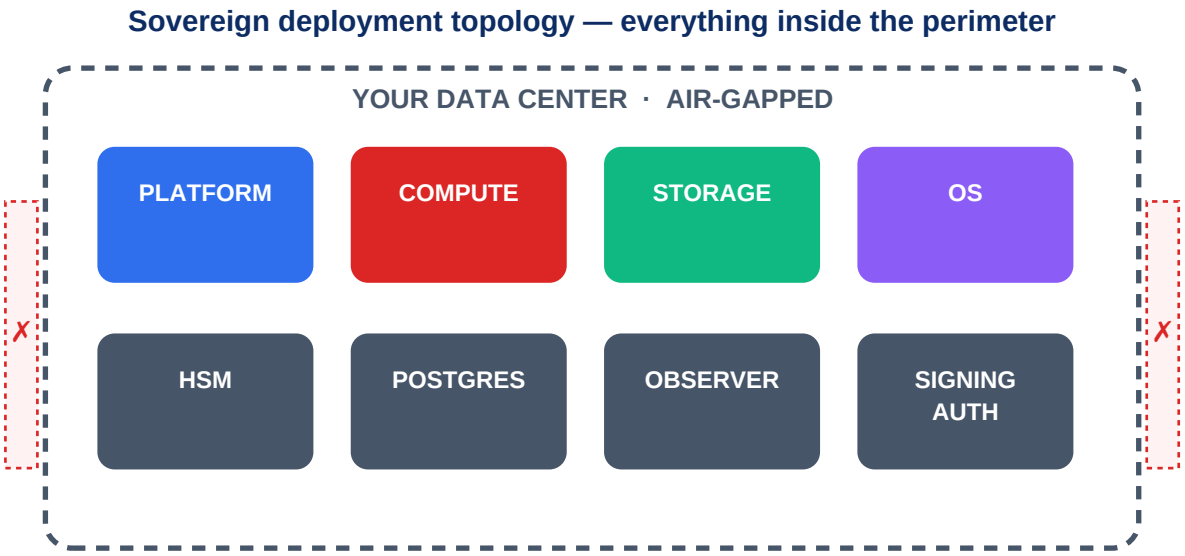


Figure 2.1 — All components inside the perimeter. Network egress blocked.

2.1 Components

A reference Sovereign deployment includes:

- **Platform** (Agent build/run/observe) — typically 3 replicas behind a load balancer
- **Compute** (GPU dispatcher) — manages locally-attached GPUs as the worker pool
- **Storage** (content-addressed) — backed by local NVMe or NFS; Filecoin tier optional
- **OS** (operational dashboard) — for human operators
- **HSM** — Yubico HSM2 or Thales Luna; holds all signing + encryption keys
- **Postgres** — primary data store; encrypted at rest with HSM-held key
- **Observer** — local Prometheus + Grafana for monitoring (no external telemetry)
- **Signing authority** — emits R+1/R+2/R+3 receipts using HSM-held key

3. Deployment Models

Three deployment models are supported, each with different operational and certification characteristics.

3.1 Bare metal

DCS containers run directly on the customer's servers (no hypervisor). Best performance, simplest threat model. Requires Linux (Ubuntu 22.04 LTS or RHEL 9 supported). Used for highest-security tier deployments where every layer of indirection is removed.

3.2 Kubernetes

DCS components are packaged as Helm charts. Deploys to any conformant K8s 1.28+ cluster — OpenShift, EKS-A, AKS-on-prem, Rancher, k3s. The most common deployment pattern; works well with existing customer ops tooling.

3.3 VMware

For customers with vSphere or Tanzu environments. DCS ships pre-built OVA images. Slightly higher overhead than bare metal but integrates with existing VM-management workflows.

4. The Update Channel

The single network exception in a Sovereign deployment is the update channel. DCS publishes signed software packages to a public registry; the customer's Sovereign cluster periodically pulls them (default: weekly, configurable). The pull is one-way — no telemetry, no configuration data, no requests other than "do you have a newer version of this artifact?"

4.1 Update verification

Every update package is signed by DCS's update key (held in an offline HSM at DCS HQ in Dubai). The customer's cluster verifies the signature against a pinned public key before installing. A revoked key (via Trust SKU) immediately invalidates subsequent updates.

4.2 Air-gapped update mode

Customers who require zero network connectivity (not even an inbound-only channel) can use the air-gapped update mode. DCS ships signed update bundles on physical media (USB drive or DVD, depending on the customer's classification rules). The customer's ops team manually loads the bundle into the cluster via the update CLI.

CHAPTER · 5

5. Key Management + HSM

All cryptographic keys in a Sovereign deployment are held in HSMs the customer owns. DCS does not have access to the keys at any point. Three HSM vendors are supported:

Vendor + Model	FIPS level	Notes
Yubico YubiHSM 2	FIPS 140-2 L3	Lowest cost · 1U or USB form factor · good for small deployments
Thales Luna 7	FIPS 140-3 L3	Enterprise standard · supports clustering · network-attached
nCipher nShield 5c	FIPS 140-2 L3+	Highest security · supports K-of-N authorization · UAE / EU government default

5.1 Key hierarchy

Three-tier key hierarchy mirrors the cloud Storage model but with HSM-held roots:

- **HSM master key** — never leaves the HSM. Used only to wrap KEKs.
- **Tenant KEK** — one per tenant. Wrapped by master key. Used to wrap DEKs.
- **File DEK** — one per file. Wrapped by tenant KEK. Encrypts the actual data.

6. Data Lifecycle — zero egress

Data lifecycle — zero egress guarantee

At no point does data leave the perimeter — not for training, not for telemetry, not for diagnostic logs. The DCS update channel pulls inbound only.

A network egress block is enforced by your firewall, validated by DCS at install time.

Figure 6.1 — Data enters, processes, stores, serves, deletes — all without crossing the perimeter.

6.1 Ingestion

Data enters via APIs internal to the customer's network. The Sovereign API endpoint is bound to the customer's internal network (typically a VPN-only address); it cannot be reached from the public internet. Customer applications connect to the API as they would to any internal microservice.

6.2 Processing

AI inference runs on GPUs attached to the customer's servers (typically A100s or H100s in 4-GPU or 8-GPU configurations). No GPU compute leaves the perimeter. Even when DCS support is troubleshooting a performance issue, they cannot see model weights or inference data — only timing and resource metrics.

6.3 Erasure

The same cryptographic-erasure pattern as cloud Storage applies. When the customer requests deletion, the relevant DEK is destroyed in the HSM. The ciphertext remains on disk until the customer's normal storage-reclamation process runs, but it is mathematically unrecoverable from the moment the DEK is gone. Signed erasure certificates are issued by the on-prem signing authority.

7. Telemetry + Observability

All observability is local. Sovereign ships with a pre-configured Prometheus + Grafana stack that scrapes metrics from every component and stores them in the customer's own time-series database. No metrics, traces, or logs leave the perimeter. DCS has no visibility into the customer's usage patterns, error rates, or anything else.

Customers who want to share telemetry with DCS for support purposes (e.g., to help diagnose a performance regression) can use the support bundle mechanism: a manually-generated tarball containing scrubbed logs and metrics, which the customer reviews before sending. The scrubbing process is documented and the customer can run it manually with the open-source *dcs-scrub* tool before sharing.

8. Support Model

Support for Sovereign deployments works differently from SaaS. DCS engineers cannot log into a Sovereign cluster; the cluster has no remote access enabled. Three support channels are available:

- **Documentation + runbooks** — comprehensive on-prem ops guide bundled with the deployment
- **Email + phone support** — DCS engineers answer questions, walk through troubleshooting
- **Customer-initiated screen-share** — when a complex issue needs DCS eyes-on, the customer's ops engineer shares their screen over the customer's preferred channel (Zoom, Teams, Webex). DCS sees what the customer chooses to show, nothing more.

All three channels are included in the Sovereign subscription. Response time targets are defined in a Sovereign-specific SLA that overrides the cloud SLA.

9. Compliance Certifications

Sovereign deployments map to the following regulatory frameworks. The certification process happens against the customer's deployment, not DCS's SaaS — so the customer's auditor is the relevant authority.

Framework	Authority	Sovereign support
FedRAMP High	US GSA	Reference architecture provided; customer pursues authorizatio
IL-4 / IL-5	US DoD	Architecture compatible; STIG hardening guide included
UK G-Cloud OFFICIAL	UK Crown Commercial	Reference deployment certified
EU EUCS High	ENISA	Reference deployment certified
UAE Sovereign Cloud Tier 3	Telecommunications + Digital Government Regulatory Authority	Certified
India MeitY-certified	MeitY	Reference deployment certified
HIPAA	HHS	BAA with DCS optional (DCS has no access in Sovereign mode
ISO 27001	ISO	DCS holds; customer applies to deployment

10. Performance vs. SaaS

A Sovereign deployment can match or exceed SaaS performance for the customer's workload — because there is no cross-region network hop, no shared-tenant noisy-neighbor effect, and the GPUs are dedicated to the customer. The tradeoff is that the customer is responsible for their own scaling.

Metric	SaaS p99	Sovereign p99	Notes
Inference latency (Sonnet-class)	4.8 s	3.2 s	No cross-region hop
Agent dispatch latency	64 ms	12 ms	Local network only
Storage read (warm)	120 ms	8 ms	Local disk
Storage write durability	15 min	4 min	Local replication only (no Filecoin)
Receipt verification	12 ms	4 ms	Local signing authority

11. Pricing

Sovereign is sold as an annual subscription. Pricing covers the software license, the support, the update channel, and the certification artefacts (FedRAMP / IL-4 / UAE Tier 3 reference packages). It does NOT include the underlying hardware, the HSMs, or operating costs (power, cooling, GPUs, staff).

Tier	Annual subscription	Includes	Best for
Sovereign Lite	$120,000	1 cluster · 8 GPUs · 5 TB storage	Small department · pilot
Sovereign Standard	$340,000	3 clusters · 24 GPUs · 50 TB	Mid-size department · production
Sovereign Enterprise	custom	Unlimited · multi-DC · 24/7 support	Full-org rollout

12. Comparison vs. Alternatives

	AWS GovCloud	Azure Gov	On-prem GPU rental	DCS Sovereign
Truly air-gapped	✗	✗	✓	✓
DCS receipts + audit	✗	✗	✗	✓
Vendor has zero data access	✗	✗	✓	✓
Pre-built AI agent runtime	✗	✗	✗	✓
Unified inbox (OS) included	✗	✗	✗	✓
Filecoin permanence available	✗	✗	✗	~ (optional)
Customer owns the hardware	✗	✗	✓	✓
FedRAMP-ready	✓	✓	~	~ (reference arch)

13. Implementation Guide

13.1 Hardware sizing

Sizing tier	Compute	Storage	GPUs	RAM	Use case
Pilot (Lite)	8 vCPU × 3	5 TB NVMe	8 × A100	256 GB	5-20 internal users
Production	32 vCPU × 5	50 TB	24 × A100/H100	1 TB	50-500 users
Enterprise	64 vCPU × 10+	500 TB+	96+ × H100	8 TB+	1k+ users

13.2 Install command

```
# On the bootstrap node (Kubernetes 1.28+)
$ helm repo add dcs https://updates.dcsai.ai/sovereign
$ helm install dcs-sovereign dcs/sovereign \
    --namespace dcs --create-namespace \
    --values ./customer-config.yaml \
    --set hsm.vendor=yubico \
    --set hsm.endpoint=https://yubihsm.internal:12345 \
    --set egress.firewall_block_validation=strict

[validating firewall egress block: ✓]
[provisioning HSM connection: ✓]
[loading initial signed bundle: ✓]
[starting platform: ✓]
[starting compute dispatcher: ✓]
[starting storage gateway: ✓]
[starting OS shell: ✓]
[generating tenant root keys in HSM: ✓]
Sovereign cluster ready at https://dcs.internal.acme.com
```

14. References

[1] US NIST. **FIPS 140-3 — Security Requirements for Cryptographic Modules**. 2019.

[2] US GSA. **FedRAMP High Baseline (Rev 5)**. 2024.

[3] US DoD. **Cloud Computing SRG IL-4 / IL-5**. 2022.

[4] ENISA. **EUCS Candidate Scheme — High assurance level**. 2024.

[5] UAE TDRA. **Sovereign Cloud Tier 3 Requirements**. 2025.

[6] India MeitY. **MeitY-certified cloud service framework**. 2024.

[7] NIST. **SP 800-53 Rev. 5 — Security and Privacy Controls**. 2020.

[8] Yubico. **YubiHSM 2 Reference Manual**.

[9] Thales. **Luna HSM 7 Documentation**.

[10] CIS. **Kubernetes Benchmark v1.8.0**. (Hardening reference for K8s deployments)

[11] DISA. **STIG for Kubernetes**. (DoD hardening)

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Agents

A curated catalog of 641 pre-built agents

Install in one click. Forked into your workspace. Same sandbox as your own code.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai
agents@dcsai.ai
github.com/dcs-platform/agents

Abstract

DCS Agents is a curated catalog of 641 pre-built AI agents covering 14 functional categories. Each agent is a single agent.yaml file plus its prompts and test suite. Customers install agents into their workspace with one click; they run inside the DCS Platform runtime with the same sandboxing, capability scoping, and receipt-emission guarantees as agents the customer writes themselves.

The catalog exists for the same reason the App Store exists: most teams do not want to write their own version of every utility. A refund-handler agent is broadly the same problem whether you sell shoes or SaaS — there is room for a well-tested shared implementation. The 641 agents in the catalog are the most-requested generic implementations, contributed by DCS Labs (the in-house research team) and verified third parties.

Agents in the catalog are versioned, reviewed, and signed. Each install creates a fork in the customer's workspace; the customer can modify the fork freely. Upstream updates are opt-in via a clear diff review, never automatic.

Who this is for

Engineering leaders evaluating whether to build or buy AI agents for common workflows. Product managers shopping for a vertical-specific agent (e.g., a healthcare-intake agent). Third-party developers wanting to publish agents to the catalog. Compliance teams reviewing the security posture of installed agents.

Contents

CHAPTER · 1

1. Introduction

The first generation of agent platforms (LangChain templates, OpenAI GPT Store, Anthropic Computer Use examples) demonstrated that pre-built agents are useful. They also demonstrated three things about what a production catalog needs:

- **Sandboxing.** A catalog agent must run in the same sandbox as a customer-written one. No exceptions.
- **Versioning.** Catalog agents must be versioned; updates must be opt-in; the customer must see the diff before accepting.
- **Receipts.** Catalog agents must emit the same signed receipts as customer agents. The audit trail does not get a special case for "the agent came from the catalog."

2. Catalog Structure

Figure 2.1 — Catalog organized into 14 categories by function.

3. Anatomy of an Agent

A single agent card — what every catalog entry shows

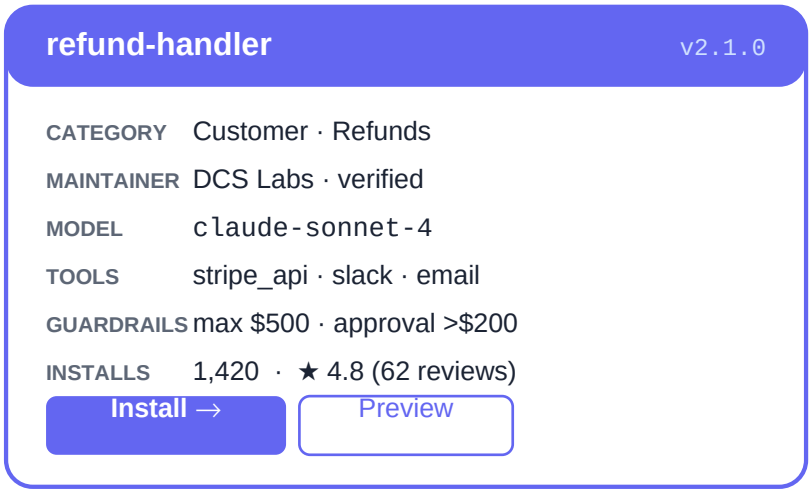


Figure 3.1 — A single catalog card showing version, maintainer, tools, guardrails, popularity.

3.1 The seven mandatory fields

Every agent in the catalog declares:

- **name + version** — semver-like; latest version always installable
- **maintainer** — DCS Labs / verified third-party / community
- **model** — which LLM the agent wraps (catalog agents are model-agnostic where possible)
- **tools** — full MCP capability list; what the agent can touch externally
- **guardrails** — domain-specific limits (dollar caps, approval thresholds, allowed regions)
- **tests** — sample inputs + expected behaviors; must pass before publication
- **license** — Apache-2.0, MIT, or commercial; visible at install time

4. Installation + Forking

Installing a catalog agent creates a fork in the customer's workspace. The customer now owns the fork — they can edit prompts, add tools, tighten guardrails. Changes do not flow back to the catalog unless the customer explicitly contributes them.

```
$ dcs catalog install refund-handler@2.1.0
  forked to workspace · /agents/refund-handler/
  applying default config from .dcs-defaults.yaml ✓
  running install tests ✓
  receipts emitted ✓
  agent active
```

CHAPTER · 5

5. Versioning + Updates

When an upstream agent is updated, installed forks get a notification. Customers see the diff (prompts, tools, guardrails) and can accept, reject, or accept-selectively. Updates are never applied automatically. The notification includes a brief changelog explaining what changed and why.

6. Quality Bar

Three tiers of catalog entry exist, distinguished by reviews:

Tier	Reviewer	What's checked	Visible badge
Verified	DCS Labs (paid)	Security · prompts · tests · guardrails · published evals	✓ verified (green)
Community	Community review (3 votes)	Security · prompts · tests	★ community
Experimental	Self-published, automated checks only	Static analysis only	■ experimental

7. Publisher Program

Third parties can publish agents to the catalog. The publisher program requires:

- A signed publisher agreement (DCS-issued)
- Identity verification (corporate registration for orgs, government ID for individuals)
- A passing automated security scan on every release
- Public contact details and a response-time commitment for security disclosures

8. Revenue Share

Catalog agents can be free or paid. Paid agents charge an install fee + per-run fee. Revenue split is 70% to the publisher, 30% to DCS. Payouts settle monthly via Stripe Connect.

Tier	Install fee	Per-run fee	Publisher take	DCS take
Free	$0	$0	—	—
Pay-per-install	$1-$99	$0	70%	30%
Pay-per-run	$0	$0.001-$1.00	70%	30%
Subscription	$5-$500/mo	$0	70%	30%

9. Security Review

Every Verified-tier agent passes a DCS Labs security review before publication:

- **Prompt injection resistance** — agent is tested against the OWASP LLM Top 10 prompts
- **Capability minimisation** — tools declared must be the minimum needed for the task
- **Guardrail completeness** — every dollar-bearing action has a cap and an approval threshold
- **Test coverage** — declared test suite covers happy path + 5 adversarial cases
- **Output validation** — agent's outputs are checked for PII leakage, allowed-format compliance

CHAPTER · 10

10. Performance + Cost

Catalog agents share the same runtime as custom agents, so their performance characteristics are identical (see DCS Platform Whitepaper Chapter 12). The additional cost when running a catalog agent is the publisher's per-run fee (if any), billed alongside the underlying compute + token costs.

11. Comparison vs. Alternatives

	OpenAI GPT Store	Anthropic CUE	LangChain Hub	DCS Agents
Signed agents	✗	✗	✗	✓
Sandboxed runtime	~	~	✗	✓
Capability scoping enforced	✗	✗	✗	✓
Per-run cost transparent	~	~	depends on hoster	✓
Catalog size	~3M	~50	~10k	641 (curated)
Audit trail	✗	✗	✗	✓ (R+1+R+2)
Vendor-neutral models	✗	✗	✓	✓

The OpenAI GPT Store is much larger but virtually un-curated; most listings are duplicates. DCS Agents trades quantity for verifiability + curation.

12. Implementation Guide

12.1 Browse + install

```
$ dcs catalog search "refund"
  Found 12 agents:
  ✓ refund-handler (v2.1.0)   · DCS Labs · 1,420 installs · ★ 4.8
  ✓ stripe-refund-pro (v1.4)  · Pintastic · 280 installs · ★ 4.6
  ■ refund-bot (v0.3)         · community · 18 installs · ★ 3.9
  ...

$ dcs catalog install refund-handler@2.1.0
```

13. References

[1] Apple App Store. **App Review Guidelines**. (Curation model reference)

[2] Anthropic. **Building Effective Agents**. December 2024.

[3] OpenAI. **GPT Store policies**. 2024.

[4] OWASP. **OWASP Top 10 for LLM Applications**. 2024.

[5] GitHub. **Marketplace publisher guidelines**. (Verified-publisher model)

DCS AI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Agent Studio

An IDE for building production AI agents

Code editor, visual builder, LLM collaborator. Three modes, one signed artifact.

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai

studio@dcsai.ai
studio.dcsai.ai

Abstract

DCS Agent Studio is an IDE for authoring agents that run on the DCS Platform. It provides three editing modes: a code editor (for engineers who want to write agent.yaml directly), a visual builder (for non-engineers who want a drag-and-drop flow), and an LLM collaborator (for anyone who wants to describe what the agent should do in natural language and have the IDE write the agent.yaml for them). All three modes produce the same artifact — a signed, content-addressed agent that runs on the Platform.

Studio exists because authoring agents is a different skill from authoring traditional code. Prompts are not code; tests for non-deterministic systems are not unit tests; deploying an agent into production is closer to deploying a person than deploying a microservice. Studio is the workbench that makes these new skills as productive as possible.

Who this is for

Engineers building production agents. Product managers and analysts who want to author agents without engineering. Teams adopting AI for the first time who need a guided workbench. Educators teaching agent design. Anyone who has tried to write an agent in plain text and given up.

Contents

CHAPTER · 1

1. Introduction

Code editors evolved over 50 years to support a specific workflow: write deterministic instructions, run them, observe the result, iterate. Agent authoring breaks several of those assumptions: the instructions are partly natural language, the results are partly random, and "running" an agent costs real money. A code editor optimised for deterministic code is the wrong tool.

Agent Studio takes the things a code editor gets right (syntax highlighting, autocomplete, multi-file project view) and adds the things agent authoring needs that the editor doesn't: a prompt editor with rendered preview, a test runner that knows how to evaluate non-deterministic output, a live sandbox that runs the agent against test inputs without spending real money, and a collaborator panel where the IDE itself helps you write the agent.

2. Three editing modes

Agent Studio IDE — file tree + prompt editor + test runner

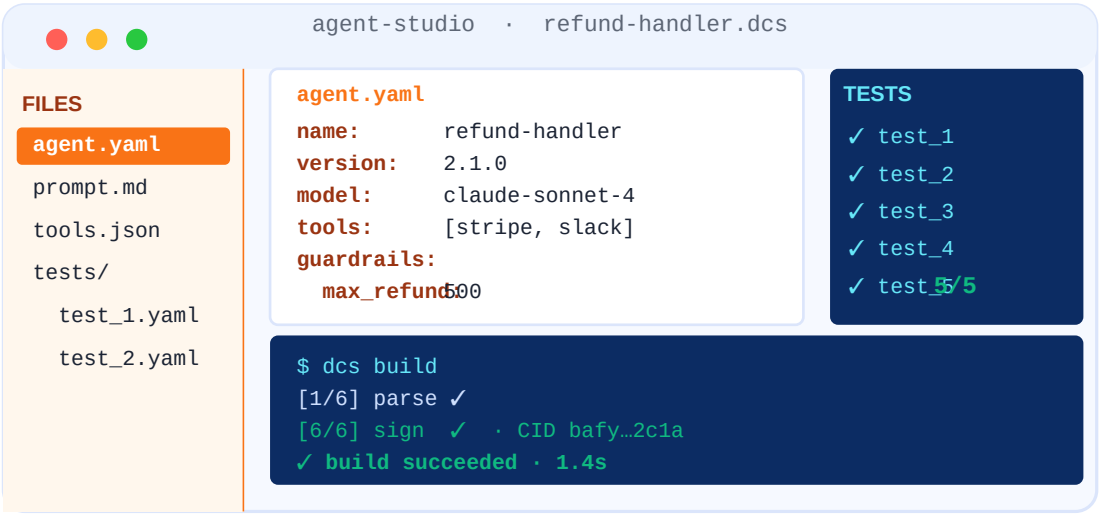
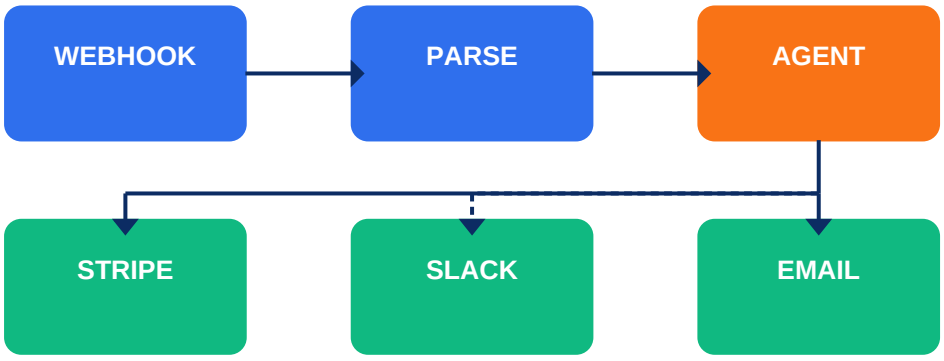


Figure 2.1 — Code editor mode showing file tree, agent.yaml, tests, terminal.

Visual builder — drag, drop, connect



Visual builder compiles to the same agent.yaml as the code editor.

Figure 2.2 — Visual builder mode for drag-and-drop authoring.

3. The Code Editor

Built on the Monaco editor (same engine as VS Code). Adds DCS-specific extensions:

- Autocomplete for agent.yaml fields, tool capabilities, model names
- Inline validation of guardrails (e.g., max_refund must be a number)
- Hover-to-preview prompts (rendered markdown shown in popup)
- Live tool documentation (hover over a tool name to see the MCP server's capability docs)
- Test runner integrated in the side panel

CHAPTER · 4

4. The Visual Builder

Drag-and-drop canvas for non-engineers. Each node represents a step in the agent's flow: trigger, prompt, tool call, condition, branch. Edges represent data flow. The visual graph compiles to agent.yaml on save; you can switch to the code editor at any time to inspect or hand-edit.

CHAPTER · 5

5. The LLM Collaborator

A right-side panel where you describe what you want in natural language and the IDE writes the agent for you. The collaborator can:

- Generate a new agent from scratch: "Write me a refund-handler for Stripe with a $500 cap"
- Modify an existing agent: "Add a Slack notification when the refund is over $200"
- Suggest improvements: "Review this agent and tell me what could go wrong"
- Generate tests: "Write 5 test cases for this agent including edge cases"

6. Test Runner

Tests for non-deterministic systems work differently from unit tests. The DCS test runner supports three assertion styles:

- **Exact match** — for tool calls (the agent should call Stripe.refund_create with $42.50)
- **Schema match** — for outputs that should have a certain structure (JSON shape, required fields)
- **LLM judge** — for outputs that should "look right" (a second model evaluates the first model's output against a rubric)

CHAPTER · 7

7. Live Preview

A sandboxed runtime that executes the agent against test inputs without billing real money. Tool calls are mocked using the tools.json declarations. Model inference uses a free tier (limited to 1000 tokens/run, 100 runs/day per author). Useful for rapid iteration before shipping to production.

CHAPTER · 8

8. Version Control + Git

Studio works on top of Git. Each agent is a directory in a repo; commits trigger a *dcs build* in CI; merging to main deploys to staging. The full DCS Platform Build Pipeline (Chapter 4 of the Platform whitepaper) runs on every push.

CHAPTER · 9

9. Templates + Starters

Studio ships with 24 starter templates for common patterns:

- refund-handler · ticket-router · support-triage · meeting-scheduler · invoice-extractor
- lead-qualifier · content-summarizer · code-reviewer · pr-describer · changelog-writer
- research-assistant · doc-translator · email-drafter · email-classifier · contract-redliner
- data-extractor · csv-cleaner · sql-writer · dashboard-narrator · chart-explainer
- monitoring-alert-triage · log-investigator · incident-summarizer · runbook-executor

10. Performance

Operation	p50	p99	Notes
Editor key-to-paint latency	8 ms	24 ms	Monaco; local
Autocomplete suggestion	40 ms	120 ms	Static + LLM-augmented
Hover prompt preview	14 ms	38 ms	Cached after first hover
Test runner (5 tests)	4 s	12 s	Includes LLM judge calls
Live preview (1 sandbox run)	1.4 s	4 s	Mocked tools; throttled model
Visual → code compile	180 ms	420 ms	Graph to agent.yaml
Build + deploy to staging	8 s	24 s	Full DCS build pipeline

11. Comparison vs. Alternatives

	Cursor	Anthropic CUE	Make / n8n	DCS Agent Studio
Agent-specific lints	✗	~	✗	✓
Visual builder	✗	✗	✓	✓
LLM collaborator	✓	~	✗	✓
Non-deterministic test runner	✗	✗	✗	✓
Signed-artifact output	✗	✗	✗	✓
Live sandboxed preview	~	~	✓	✓
Git-native workflow	✓	✗	✗	✓

12. Implementation Guide

12.1 Start from scratch

```
$ dcs studio new my-first-agent
  Opening agent-studio at studio.dcsai.ai/my-first-agent
  Starter templates available in the right panel.
  Or ask the collaborator: "Write me a refund agent for Stripe."
```

13. References

[1] Microsoft. **Monaco Editor**. (Embedded code editor)

[2] Anthropic. **Computer Use Examples (CUE)**. 2024.

[3] GitHub. **Copilot Workspace**. (LLM collaborator pattern)

[4] Cursor. **The AI-first code editor**. 2024.

[5] Make (Integromat). **Visual workflow automation**. (Visual builder pattern)

D DCS AI Technologies

LEGAL DOCUMENT · v1.0 · MAY 2026

Master Services Agreement

The umbrella contract for all DCS products + services

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai

legal@dcsai.ai
dcsai.ai/msa

Master Services Agreement

This Master Services Agreement ("**Agreement**") is between **DCS AI Technologies L.L.C**, a limited liability company (single-owner) incorporated in the Emirate of Dubai, United Arab Emirates under Commercial License No. 1624450 issued by the Dubai Department of Economy & Tourism (Commercial Register No. 2862220; Dubai Chamber of Commerce + Industry Membership No. 686532; paid-up share capital AED 250,000), with registered office at Office 40, Dubai Industrial City L.L.C, Saih Shuaib 3, Dubai, UAE, owned and managed by Mr. Deepak Dudi Dharam Vir Singh ("**DCS**") and the customer entity identified in an Order Form ("**Customer**"). Each a "**Party**"; together, the "**Parties**".

This Agreement governs the Customer's use of all DCS products and services. It incorporates by reference (a) the Data Processing Agreement, (b) the Service Level Agreement, (c) the Acceptable Use Policy, and (d) any product-specific addenda referenced in the Order Form.

1. Definitions

"Affiliate" any entity that directly or indirectly controls, is controlled by, or is under common control with a Party.

"AUP" the Acceptable Use Policy, as amended from time to time, available at dcsai.ai/aup.

"Confidential Information" all non-public information disclosed by one Party to the other that is identified as confidential or that a reasonable person would understand to be confidential.

"Customer Data" data submitted by the Customer or its end-users to the Services.

"DPA" the Data Processing Agreement attached as Schedule A.

"Documentation" the technical documentation for the Services published at docs.dcsai.ai.

"Effective Date" the date of the first Order Form signed by both Parties.

"Fees" the amounts payable by the Customer for the Services as set out in the Order Form or the published price list at dcsai.ai/pricing.

"Intellectual Property Rights" all patents, copyrights, trademarks, trade secrets, and other proprietary rights.

"Order Form" an ordering document specifying the Services purchased, the term, and the Fees.

"Services" the products and services made available by DCS, including DCS Platform, Compute, Storage, OS, Sovereign, Agents, and Agent Studio.

"SLA" the Service Level Agreement, as amended from time to time, available at dcsai.ai/sla.

"Term" as defined in Section 10.1.

"User" an individual authorised by the Customer to use the Services on the Customer's behalf.

2. Services

2.1 Provision. Subject to the Customer's compliance with this Agreement and payment of the Fees, DCS will provide the Services to the Customer during the Term.

2.2 Documentation. The Services will be provided substantially in accordance with the Documentation. DCS may update the Documentation from time to time.

2.3 Service Levels. The Services are subject to the SLA. The Customer's sole and exclusive remedy for failure to meet the SLA is the Service Credits specified therein.

2.4 Changes. DCS may modify the Services from time to time. Material adverse changes will be announced at least 30 days in advance.

3. Customer Obligations

3.1 Authorised use. The Customer will (a) use the Services only for its own business purposes, (b) comply with the AUP, (c) be responsible for the acts and omissions of its Users, and (d) maintain the confidentiality of its credentials.

3.2 Customer Data. The Customer represents that it has all necessary rights, consents, and notices in place to submit Customer Data to the Services.

3.3 Compliance. The Customer is responsible for ensuring that its use of the Services complies with all applicable laws.

3.4 Restrictions. The Customer will not (a) reverse engineer the Services except to the extent permitted by applicable law, (b) resell or sublicense the Services without DCS's prior written consent, (c) use the Services to develop a competing product, or (d) violate the AUP.

4. Fees and Payment

4.1 Fees. The Customer will pay the Fees specified in the Order Form. Usage-based Fees are calculated based on the Customer's actual usage as measured by DCS's systems.

4.2 Invoicing. Subscription Fees are invoiced in advance; usage Fees are invoiced monthly in arrears.

4.3 Payment terms. Invoices are due within 30 days of the invoice date. Overdue amounts bear interest at 1.5% per month or the maximum allowed by law.

4.4 Taxes. Fees are exclusive of all taxes. The Customer is responsible for all taxes other than DCS's income taxes.

4.5 Suspension. If the Customer's payment is overdue by more than 30 days, DCS may suspend the Services after providing 10 days' written notice and an opportunity to cure.

5. Term and Termination

5.1 Term. This Agreement begins on the Effective Date and continues for the term specified in the Order Form, with automatic renewal for successive 12-month periods unless either Party gives at least 60 days' notice of non-renewal.

5.2 Termination for cause. Either Party may terminate this Agreement immediately on written notice if the other Party (a) materially breaches the Agreement and fails to cure within 30 days of written notice, (b) becomes insolvent or files for bankruptcy, or (c) ceases business operations.

5.3 Effect of termination. Upon termination: (a) the Customer's access to the Services will end, (b) DCS will delete or return Customer Data in accordance with the DPA, (c) the Customer will pay all outstanding Fees, and (d) sections 6 (IP), 7 (Confidentiality), 8 (Warranties), 9 (Liability), and 11 (General) will survive.

6. Intellectual Property

6.1 DCS IP. DCS retains all right, title, and interest in and to the Services, the Documentation, and all underlying technology, including all Intellectual Property Rights therein.

6.2 Customer IP. The Customer retains all right, title, and interest in and to Customer Data. The Customer grants DCS a limited, non-exclusive, worldwide, royalty-free licence to use Customer Data solely to provide the Services.

6.3 Feedback. If the Customer provides feedback about the Services, DCS may use that feedback without restriction.

6.4 No model training. DCS will not use Customer Data to train its own AI models without the Customer's prior written consent. This restriction does not apply to aggregated, de-identified usage telemetry that cannot be linked back to the Customer.

7. Confidentiality

7.1 Obligations. Each Party will (a) use the other Party's Confidential Information only to perform under this Agreement, (b) protect such information with the same degree of care it uses for its own confidential information (and not less than reasonable care), and (c) not disclose such information to any third party without prior written consent.

7.2 Exclusions. The obligations in Section 7.1 do not apply to information that (a) was rightfully known prior to receipt, (b) is or becomes publicly available through no breach of this Agreement, (c) is rightfully received from a third party without restriction, or (d) is independently developed without reference to the disclosing Party's Confidential Information.

7.3 Compelled disclosure. A Party may disclose Confidential Information to the extent required by law, provided that it gives the other Party prompt notice (where permitted) and reasonable cooperation to seek a protective order.

8. Warranties and Disclaimers

8.1 Mutual warranties. Each Party warrants that it has the corporate authority to enter into this Agreement.

8.2 DCS warranty. DCS warrants that the Services will perform substantially in accordance with the Documentation. The Customer's sole and exclusive remedy for breach of this warranty is the Service Credits specified in the SLA.

8.3 AI output disclaimer. The Services include AI-generated output. AI output may be inaccurate, incomplete, or inappropriate. The Customer is solely responsible for reviewing, validating, and acting on AI output.

8.4 Disclaimer. EXCEPT AS EXPRESSLY SET OUT IN THIS AGREEMENT, THE SERVICES ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.

9. Limitation of Liability

9.1 Cap. EXCEPT AS PROVIDED IN SECTION 9.3, EACH PARTY'S TOTAL LIABILITY UNDER THIS AGREEMENT WILL NOT EXCEED THE FEES PAID OR PAYABLE BY THE CUSTOMER IN THE 12 MONTHS IMMEDIATELY PRECEDING THE EVENT GIVING RISE TO THE CLAIM.

9.2 Exclusions. EXCEPT AS PROVIDED IN SECTION 9.3, NEITHER PARTY WILL BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, OR DATA.

9.3 Carve-outs. The limitations in Sections 9.1 and 9.2 do not apply to (a) the Customer's payment obligations, (b) breach of confidentiality under Section 7, (c) indemnification obligations under Section 10, (d) fraud or wilful misconduct, or (e) any liability that cannot be excluded under applicable law.

10. Indemnification

10.1 By DCS. DCS will defend and indemnify the Customer against any third-party claim alleging that the Services infringe such third party's Intellectual Property Rights. If such a claim is made or DCS reasonably believes it will be, DCS may, at its option, (a) procure the right for the Customer to continue using the Services, (b) modify the Services to be non-infringing, or (c) terminate the affected Services and refund any prepaid Fees.

10.2 By Customer. The Customer will defend and indemnify DCS against any third-party claim arising from (a) the Customer's use of the Services in violation of this Agreement or the AUP, (b) Customer Data, or (c) the Customer's breach of applicable law.

10.3 Procedure. The indemnified Party will (a) promptly notify the indemnifying Party of the claim, (b) give the indemnifying Party sole control of the defence, and (c) cooperate at the indemnifying Party's expense.

11. General

11.1 Governing law. This Agreement is governed by the laws of England and Wales, without regard to conflict-of-laws principles.

11.2 Dispute resolution. Any dispute arising out of or in connection with this Agreement will be resolved by final and binding arbitration administered by the London Court of International Arbitration (LCIA) under the LCIA Arbitration Rules. The seat of arbitration will be London; the language of arbitration will be English.

11.3 Notices. Notices must be in writing and sent to the addresses specified in the Order Form (for DCS: legal@dcsai.ai with a copy to DCS AI Technologies L.L.C, Dubai Industrial City).

11.4 Assignment. Neither Party may assign this Agreement without the other Party's prior written consent, except that either Party may assign to an Affiliate or in connection with a merger, acquisition, or sale of all or substantially all of its assets.

11.5 Force majeure. Neither Party will be liable for failure to perform due to causes beyond its reasonable control, including acts of God, war, terrorism, civil unrest, government action, internet backbone failures, and pandemic-related disruptions.

11.6 Entire agreement. This Agreement (including the DPA, SLA, AUP, all Order Forms, and any addenda) constitutes the entire agreement between the Parties and supersedes all prior agreements regarding the subject matter.

11.7 Amendment. This Agreement may be amended only by a written instrument signed by authorised representatives of both Parties, except that DCS may update the DPA, SLA, and AUP from time to time as provided in those documents.

11.8 Severability. If any provision of this Agreement is held to be invalid or unenforceable, the remaining provisions will continue in full force and effect.

11.9 Waiver. No waiver of any provision of this Agreement will be effective unless in writing and signed by the waiving Party.

11.10 Counterparts. This Agreement may be executed in counterparts, including by electronic signature, each of which will be deemed an original.

Signatures

This Agreement is effective as of the date last signed below.

For DCS AI Technologies L.L.C

Signature: _____

Name: _____

Title: _____

Date: _____

For Customer

Signature: _____

Name: _____

Title: _____

Date: _____

DCS AI Technologies

LEGAL DOCUMENT · v1.0 · MAY 2026

Data Processing Agreement

GDPR Article 28 · UK GDPR · DPDP · LGPD compliant

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai

legal@dcsai.ai
Privacy: dcsai.ai/privacy

Data Processing Agreement

This Data Processing Agreement ("**DPA**") forms part of the Master Services Agreement ("**Agreement**") between DCS AI Technologies L.L.C, a company incorporated in the United Arab Emirates with registered office at Office 40, Dubai Industrial City L.L.C, Saih Shuaib 3, Dubai, UAE, holder of Dubai Department of Economy & Tourism Commercial License No. 1624450 (Commercial Register No. 2862220, DCCI Membership No. 686532), with paid-up share capital of AED 250,000 ("**DCS**", "**Processor**"), and the customer entity identified in the order form ("**Customer**", "**Controller**"), each a "**Party**" and together the "**Parties**".

This DPA reflects the Parties' agreement with regard to the Processing of Personal Data by DCS on behalf of the Customer in connection with the Services. It is designed to satisfy the requirements of Article 28 of Regulation (EU) 2016/679 (the "**GDPR**"), the UK GDPR, the Indian Digital Personal Data Protection Act 2023 ("**DPDP**"), the Brazilian Lei Geral de Proteção de Dados ("**LGPD**"), and the California Consumer Privacy Act ("**CCPA**").

In case of conflict between this DPA and the Agreement, this DPA prevails with respect to matters relating to the Processing of Personal Data.

1. Definitions

"Affiliate" means any entity that directly or indirectly controls, is controlled by, or is under common control with a Party.

"Authorised Sub-processor" means each sub-processor listed in Schedule 3 and any other sub-processor engaged by DCS in accordance with Clause 6.

"Customer Personal Data" means Personal Data Processed by DCS on behalf of the Customer pursuant to or in connection with the Agreement.

"Data Subject" has the meaning given in Article 4(1) GDPR.

"International Transfer" means a transfer of Personal Data to a country outside the European Economic Area that is not the subject of an adequacy decision by the European Commission.

"Personal Data" has the meaning given in Article 4(1) GDPR.

"Personal Data Breach" has the meaning given in Article 4(12) GDPR.

"Processing" has the meaning given in Article 4(2) GDPR and "Process" shall be interpreted accordingly.

"Services" means the products and services made available by DCS to the Customer under the Agreement (including DCS Platform, Compute, Storage, OS, Sovereign, and any other DCS-branded offering).

"Standard Contractual Clauses" or "SCCs" means the standard contractual clauses approved by the European Commission Decision 2021/914 of 4 June 2021, as may be updated from time to time.

"Sub-processor" means any entity engaged by DCS or its Affiliates to Process Customer Personal Data on its behalf.

2. Scope and Roles

2.1 This DPA applies to the Processing of Customer Personal Data by DCS to provide the Services.

2.2 The Parties acknowledge that:

(a) The Customer is the Controller of the Customer Personal Data.

(b) DCS is the Processor and may engage Sub-processors in accordance with this DPA.

(c) Each Party shall comply with the obligations applicable to it under the GDPR and other applicable data protection laws.

2.3 The details of the Processing (subject matter, duration, nature, purpose, types of Personal Data, and categories of Data Subjects) are set out in Schedule 1.

3. Processor Obligations

DCS shall:

3.1 Process Customer Personal Data only on documented instructions from the Customer, including with regard to International Transfers, unless required by EU or Member State law to which DCS is subject. In such a case, DCS shall inform the Customer of that legal requirement before Processing, unless that law prohibits such information on important grounds of public interest.

3.2 Ensure that persons authorised to Process Customer Personal Data have committed themselves to confidentiality or are under an appropriate statutory obligation of confidentiality.

3.3 Implement the technical and organisational measures set out in Schedule 2 to ensure a level of security appropriate to the risk.

3.4 Respect the conditions referred to in Clauses 6 and 7 for engaging Sub-processors and assist the Customer with International Transfers.

3.5 Taking into account the nature of the Processing, assist the Customer by appropriate technical and organisational measures, insofar as this is possible, for the fulfilment of the Customer's obligation to respond to requests for exercising the Data Subject's rights.

3.6 Assist the Customer in ensuring compliance with the obligations pursuant to Articles 32 to 36 GDPR, taking into account the nature of Processing and the information available to DCS.

3.7 At the choice of the Customer, delete or return all Customer Personal Data after the end of the provision of services relating to Processing, and delete existing copies unless EU or Member State law requires storage of the Personal Data.

3.8 Make available to the Customer all information necessary to demonstrate compliance with the obligations laid down in Article 28 GDPR and allow for and contribute to audits, including inspections, conducted by the Customer or another auditor mandated by the Customer, in accordance with Clause 8.

4. Controller Obligations

The Customer shall:

4.1 Comply with all applicable data protection laws in respect of its Processing of Personal Data and any Processing instructions it issues to DCS.

4.2 Ensure that it has all necessary rights, consents, and notices in place to enable lawful transfer of the Customer Personal Data to DCS for the duration and purposes of the Agreement.

4.3 Be solely responsible for the accuracy, quality, and legality of Customer Personal Data and the means by which it acquired the Personal Data.

4.4 Communicate to Data Subjects, when required, the information referred to in Articles 13 and 14 GDPR.

5. Security

5.1 DCS shall implement the technical and organisational measures set out in Schedule 2 to ensure a level of security appropriate to the risk of Processing.

5.2 In assessing the appropriate level of security, DCS shall take into account in particular:

- the state of the art and the costs of implementation;
- the nature, scope, context, and purposes of Processing;
- the varying likelihood and severity of risks for the rights and freedoms of natural persons;
- the risks that are presented by the Processing, in particular from accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to Personal Data transmitted, stored, or otherwise Processed.

5.3 DCS's security measures are independently audited annually under SOC 2 Type II and ISO/IEC 27001. Current audit reports are available to the Customer under NDA.

6. Sub-processors

6.1 The Customer hereby grants DCS general authorisation to engage Sub-processors for the performance of the Services, subject to the conditions set out in this Clause 6.

6.2 An up-to-date list of Authorised Sub-processors is maintained at dcsai.ai/sub-processors and is reproduced in Schedule 3 as at the Effective Date.

6.3 DCS shall give the Customer at least 30 days' prior written notice (via the email address on file or via the Customer dashboard) of any intended addition or replacement of Sub-processors. The Customer may object on reasonable grounds related to data protection within 30 days. If the Parties cannot resolve the objection, the Customer may terminate the affected Service without penalty.

6.4 Where DCS engages a Sub-processor, DCS shall:

(a) enter into a written contract with the Sub-processor that imposes data protection obligations substantially similar to those set out in this DPA;

(b) remain fully liable to the Customer for the performance of the Sub-processor's obligations.

7. International Transfers

7.1 DCS may transfer Customer Personal Data to recipients outside the EEA, the UK, or the relevant jurisdiction, only:

(a) to a country in respect of which an adequacy decision has been issued by the relevant authority; or

(b) pursuant to the Standard Contractual Clauses (as incorporated into this DPA by reference); or

(c) pursuant to another lawful mechanism for International Transfers permitted under applicable data protection laws.

7.2 The Parties hereby enter into the Standard Contractual Clauses, which shall apply to International Transfers made by DCS to a Sub-processor located in a third country. The modular SCCs are incorporated by reference; the relevant module (typically Module 2: Controller to Processor or Module 3: Processor to Processor) shall apply based on the role of the Sub-processor.

7.3 Schedule 4 sets out the parameters required by the SCCs (parties, transfer details, technical and organisational measures, competent supervisory authority).

8. Audits

8.1 DCS shall make available to the Customer the information reasonably necessary to demonstrate compliance with its obligations under this DPA.

8.2 At the Customer's written request, and no more than once per calendar year (except where required following a Personal Data Breach), DCS shall:

(a) provide a copy of its most recent SOC 2 Type II and ISO/IEC 27001 reports; and

(b) respond to a reasonable written audit questionnaire within 30 days.

8.3 On-site audits are available to enterprise-tier Customers, subject to (i) reasonable advance notice (at least 60 days), (ii) the auditor signing DCS's standard non-disclosure agreement, (iii) the audit being conducted during normal business hours and in a manner that does not disrupt DCS's operations, and (iv) the Customer bearing the cost of the audit.

9. Personal Data Breach

9.1 DCS shall notify the Customer without undue delay and in any event within 72 hours of becoming aware of a Personal Data Breach affecting Customer Personal Data.

9.2 The notification shall include, to the extent then known:

- a description of the nature of the Personal Data Breach including, where possible, the categories and approximate number of Data Subjects concerned and the categories and approximate number of Personal Data records concerned;
- the name and contact details of DCS's Data Protection Officer;
- a description of the likely consequences of the Personal Data Breach;
- a description of the measures taken or proposed to address the Personal Data Breach.

9.3 DCS shall cooperate with the Customer and take such reasonable commercial steps as are directed by the Customer to assist in the investigation, mitigation, and remediation of any Personal Data Breach.

10. Term and Termination

10.1 This DPA shall remain in effect for the duration of the Agreement.

10.2 On termination of the Agreement, DCS shall (at the Customer's election) delete or return all Customer Personal Data to the Customer within 30 days, save to the extent that EU or Member State law requires storage of the Personal Data.

10.3 Cryptographic erasure (as described in the DCS Storage whitepaper) satisfies the deletion obligation under Clause 10.2 with respect to Customer Personal Data held in DCS Storage.

11. Liability

11.1 Each Party's liability under or in connection with this DPA is governed by the limitation of liability provisions of the Agreement.

11.2 Nothing in this DPA limits either Party's liability for: (i) gross negligence or wilful misconduct; (ii) breach of confidentiality; or (iii) liability that cannot be excluded under applicable law.

12. Governing Law and Jurisdiction

12.1 This DPA is governed by the laws of England and Wales unless the Agreement specifies a different governing law, in which case that law applies.

12.2 Disputes arising out of or in connection with this DPA shall be resolved in accordance with the dispute resolution provisions of the Agreement.

Schedule 1 — Processing Details

Subject matter · Provision of the DCS Services to the Customer under the Agreement.

Duration · The term of the Agreement, plus any period of post-termination retention permitted under Clause 10.

Nature and purpose · Processing necessary to provide the Services, including: hosting, storage, transmission, computation, indexing, analytics, audit logging, support, and billing.

Types of Personal Data · Identification data (name, email); authentication data (hashed credentials); usage data (logs, telemetry, audit receipts); any Personal Data submitted by the Customer or end-users via the Services.

Categories of Data Subjects · Customer's personnel; Customer's end-users; any individual whose Personal Data is submitted to the Services by the Customer.

Retention · Customer Personal Data is retained for the term of the Agreement and deleted within 30 days of termination, subject to legal retention requirements.

Schedule 2 — Technical and Organisational Measures

DCS implements the following measures to ensure a level of security appropriate to the risk:

Pseudonymisation and encryption of personal data

- All Personal Data is encrypted at rest using AES-256-GCM with per-tenant data encryption keys held in hardware security modules (FIPS 140-2 Level 3).
- All Personal Data in transit is encrypted using TLS 1.3 with forward secrecy.
- Per-tenant encryption keys can be destroyed on Customer request, rendering ciphertext unrecoverable (cryptographic erasure).

Confidentiality, integrity, availability and resilience

- Multi-region replication: data is replicated across at least 3 warm storage providers and up to 7 cold Filecoin storage providers in different jurisdictions.
- Tamper-evidence: every system action emits a signed receipt (DCS Standards R+1/R+2). Tampering with any record breaks the cryptographic chain.
- Access logging: all access to Customer Personal Data is logged with the actor identity, timestamp, and purpose.
- Background checks: all DCS personnel with access to production systems undergo standard employment background checks.

Ability to restore availability and access to personal data

- Recovery Point Objective (RPO): 5 minutes for hot data; 1 hour for warm data.
- Recovery Time Objective (RTO): 30 minutes for hot data; 4 hours for warm data.
- Quarterly disaster recovery exercises with documented results.

Regular testing, assessing, and evaluating

- SOC 2 Type II audit annually by an independent third-party auditor.
- ISO/IEC 27001 certification renewed annually.
- Penetration testing semi-annually by an independent security firm.
- Continuous vulnerability scanning of all production infrastructure.

Schedule 3 — Authorised Sub-processors

The following are the Authorised Sub-processors as at the Effective Date. The current list is maintained at dcsai.ai/sub-processors.

Sub-processor	Purpose	Region	GDPR mechanism
Amazon Web Services	Hot storage (S3) + EC2 compute	EU + US	SCCs Mod. 2
Cloudflare	CDN + edge caching + DNS + R2	Global	SCCs Mod. 2
Stripe Inc.	Payment processing + Connect payouts	US	SCCs Mod. 2
Anthropic PBC	LLM inference (Claude family)	US	SCCs Mod. 3
OpenAI LLC	LLM inference (GPT family)	US	SCCs Mod. 3
Supabase Inc.	Auth + Postgres-managed	EU + US	SCCs Mod. 2
Backblaze Inc.	B2 cold storage	US	SCCs Mod. 2
Filecoin SPs (various)	7-replica cold backup	Global	Per-SP terms
Lighthouse Storage Inc.	Filecoin upload broker	India	SCCs Mod. 2 + DPDP
Resend	Transactional email	US	SCCs Mod. 2
Vercel Inc.	Frontend hosting (Cloudflare Pages fallback)	Global	SCCs Mod. 2
Railway Corp.	Backend service hosting	US	SCCs Mod. 2

Schedule 4 — Standard Contractual Clauses (SCC Annexes)

Annex I.A — List of Parties

Data Exporter (Controller): The Customer entity identified in the Order Form.

Data Importer (Processor): DCS AI Technologies L.L.C, registered office Office 40, Dubai Industrial City L.L.C, Saih Shuaib 3, Dubai, UAE. Contact: legal@dcsai.ai. DPO: dpo@dcsai.ai.

Annex I.B — Description of Transfer

See Schedule 1 above.

Annex I.C — Competent Supervisory Authority

The supervisory authority of the EU Member State in which the Customer is established. If the Customer is established outside the EU, the Irish Data Protection Commission shall be the competent supervisory authority.

Annex II — Technical and Organisational Measures

See Schedule 2 above.

Annex III — List of Sub-processors

See Schedule 3 above.

Signatures

This DPA is effective as of the date last signed below.

For DCS AI Technologies L.L.C	For Customer
Signature: _____	Signature: _____
Name: _____	Name: _____
Title: _____	Title: _____
Date: _____	Date: _____

DCS AI Technologies

LEGAL DOCUMENT · v1.0 · MAY 2026

Service Level Agreement

Per-product uptime commitments + credit remedies

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai/sla
sla@dcsai.ai
status: status.dcsai.ai

Service Level Agreement

This Service Level Agreement ("**SLA**") sets out DCS's uptime commitments and the service credits to which customers are entitled if those commitments are not met. It forms part of the Master Services Agreement between DCS AI Technologies L.L.C ("**DCS**") and the customer ("**Customer**").

Each DCS product has its own SLO (Service Level Objective). Uptime is measured against the SLO in calendar-month windows and reported in real time at status.dcsai.ai. Customers who experience downtime below the SLO are automatically eligible for service credits as set out in Section 4.

1. Scope and Eligibility

1.1 This SLA applies to all paying customers on the Pro tier or above. Free-tier customers receive best-effort service with no SLA guarantees.

1.2 This SLA covers the following products:

- **DCS Platform** (api.dcsai.ai, app.dcsai.ai)
- **DCS Compute** (api-compute.dcsai.ai, compute.dcsai.ai)
- **DCS Storage** (api-storage.dcsai.ai, gateway.dcsai.ai, storage.dcsai.ai)
- **DCS OS** (api.dcsai.ai/os, os.dcsai.ai)
- **DCS Sovereign** (per-customer URLs)

1.3 This SLA does not cover:

- Beta or preview features (clearly labeled as such in the dashboard).
- Free-tier or trial accounts.
- Downtime caused by Customer-side issues (misconfigured credentials, exceeded quotas, etc.).
- Downtime during scheduled maintenance windows announced at least 48 hours in advance.
- Force majeure events as defined in Section 5.

2. Definitions

"Monthly Uptime Percentage" means the total number of minutes in a calendar month, minus the number of minutes of Unavailability during that month, divided by the total number of minutes in that month.

"Unavailability" means a period during which the relevant Service returns HTTP 5xx errors for more than 10% of requests, measured against the synthetic monitoring probes operated by DCS's independent monitoring provider (Better Stack). Read requests and write requests are measured separately for products where they have different SLOs.

"Service Credit" means a credit added to the Customer's account equal to the specified percentage of the monthly subscription fee for the affected Service. Service Credits have no cash value and may be used to offset future monthly fees.

"Maintenance Window" means a planned period of degraded or unavailable service, announced via status.dcsai.ai at least 48 hours in advance. Maintenance Windows are not counted as Unavailability.

"Force Majeure Event" means an event outside DCS's reasonable control, including natural disasters, war, terrorism, civil unrest, government action, internet backbone failures, and pandemic-related disruptions.

3. Service Level Objectives

Each Service has a specific Monthly Uptime Percentage commitment as set out below. SLOs apply to the Pro tier and above; Enterprise customers may negotiate higher SLOs in their order form.

3.1 DCS Platform

Surface	Pro SLO	Enterprise SLO	Latency p99 target
/api/* (control plane)	99.9%	99.95%	500 ms
Agent execution (warm)	99.5%	99.9%	5 s
Build pipeline	99.5%	99.9%	2 min
Dashboard (app.dcsai.ai)	99.5%	99.9%	2 s page load

3.2 DCS Compute

Surface	Pro SLO	Enterprise SLO	Latency p99 target
Dispatch API	99.9%	99.95%	100 ms
Job completion (any tier)	99.0%	99.5%	as quoted at dispatch
Worker registration	99.5%	99.9%	5 min end-to-end
Payout queue	99.5%	99.9%	T+2 (Stripe Connect)

3.3 DCS Storage

Surface	Pro SLO	Enterprise SLO	Latency p99 target
Gateway reads (cached)	99.95%	99.99%	50 ms
Gateway reads (warm miss)	99.9%	99.95%	500 ms
Upload API	99.9%	99.95%	5 s (file ≤1 MB)
Erasure API	99.5%	99.9%	30 s end-to-end
Data durability (Gold tier)	99.999999999%	99.999999999%	11 nines
Data durability (Platinum tier)	99.9999999999%	99.9999999999%	12 nines

3.4 DCS OS

Surface	Pro SLO	Enterprise SLO	Latency p99 target
Dashboard (os.dcsai.ai)	99.5%	99.9%	2 s page load
WhatsLink message routing	99.9%	99.95%	5 s end-to-end
Voice handoff	99.0%	99.5%	3 s to live agent
Module loaders (iframes)	99.5%	99.9%	1 s

3.5 DCS Sovereign

Sovereign deployments run on Customer-controlled infrastructure. DCS commits to the following only with respect to the DCS software components:

Surface	Standard SLO	Notes
Software bug fixes (Severity 1)	4 hours response	See Section 6
Software bug fixes (Severity 2)	24 hours response	
Software updates	Monthly cadence	
Security patches	72 hours of disclosure	CVSS ≥ 7.0

4. Service Credits

If the Monthly Uptime Percentage for a Service falls below the applicable SLO in a calendar month, the Customer is eligible for Service Credits as follows:

Monthly Uptime	Service Credit (% of monthly fee)	Notes
< SLO but ≥ 99.0%	10%	
< 99.0% but ≥ 95.0%	25%	
< 95.0% but ≥ 90.0%	50%	
< 90.0%	100%	Plus right to terminate without penalty

4.1 Claim process

Service Credits are issued automatically. Within 5 business days after the end of the affected month, DCS calculates the Monthly Uptime Percentage based on the independent monitoring data and credits the Customer's account. The credit appears as a line item on the next invoice.

Customers who believe DCS has miscalculated the credit may file a written dispute within 30 days of the invoice. The dispute will be reviewed within 10 business days; if DCS's monitoring data is shown to be incorrect (cross-referenced against the Customer's own monitoring), the credit will be adjusted retroactively.

4.2 Maximum credit

Service Credits in any single month may not exceed 100% of the monthly fee for the affected Service. Service Credits are the Customer's sole and exclusive remedy for any Unavailability under this SLA.

5. Exclusions

The following are not counted as Unavailability for purposes of calculating Monthly Uptime Percentage:

- **Force Majeure Events** as defined in Section 2.
- **Scheduled Maintenance Windows** announced at least 48 hours in advance.
- **Customer-caused outages** (e.g., exceeding the Customer's rate limit, providing invalid credentials, misconfiguring DNS).
- **Third-party outages** outside DCS's control (e.g., AWS regional outage if the Customer chose a single-region deployment, or a Filecoin storage provider going offline if the Customer chose Bronze tier).
- **Beta features** labeled as such in the dashboard at the time of use.
- **Free-tier or trial accounts.**
- **Brief blips** lasting under 60 seconds and not affecting more than 1% of requests in a 5-minute window.

6. Support Response Times

Support response times are commitments separate from uptime. They apply 24/7 except as noted.

Severity	Definition	Pro response	Enterprise response
Sev-1	Production down · no workaround	1 hour	30 minutes
Sev-2	Significant degradation · workaround exists	4 hours	2 hours
Sev-3	Minor issue · feature request	1 business day	4 business hours
Sev-4	Question · documentation request	2 business days	1 business day

"Response" means the first substantive reply from a DCS engineer (not an auto-acknowledgement). Response does not guarantee resolution within the same window.

6.1 Severity classification

The Customer assigns the initial severity when filing a ticket. DCS may reclassify in either direction after triage, with notice and reason.

6.2 Support channels

- **Pro tier**: email (support@dcsai.ai), in-app chat during business hours (UAE timezone), and the public Discord.
- **Enterprise tier**: dedicated Slack Connect channel, named Customer Success Manager, 24/7 phone hotline for Sev-1 issues.

7. Status and Monitoring

7.1 Real-time service status is published at status.dcsai.ai. The status page is operated by an independent third-party provider (Better Stack) and is hosted on infrastructure separate from the Services it monitors. The status page remains available even during a DCS outage.

7.2 Customers may subscribe to status updates via email, SMS, Slack webhook, or RSS. Subscriptions are configured at status.dcsai.ai/subscribe.

7.3 Monitoring methodology: synthetic probes from 8 geographic locations every 30 seconds. A Service is considered Unavailable when more than 50% of probes from any 4 locations fail for 3 consecutive intervals. This is intentionally conservative: short-lived regional connectivity issues do not count against the SLO.

8. Changes to This SLA

8.1 DCS may modify this SLA from time to time. Material adverse changes will be announced at least 30 days in advance via email to the Customer's billing contact and via the dashboard.

8.2 If the Customer does not agree with a material adverse change, the Customer may terminate the affected Service without penalty before the change takes effect.

8.3 Non-material changes (typos, clarifications, additions of new Services) are made without notice. The current version is always available at dcsai.ai/sla.

9. Governing Law

This SLA is governed by the laws of England and Wales unless the Master Services Agreement specifies a different governing law, in which case that law applies. Disputes shall be resolved in accordance with the dispute resolution provisions of the Master Services Agreement.

Effective Date and Version History

This SLA is effective from 1 June 2026.

Version history

Version	Effective from	Summary of changes
1.0	2026-06-01	Initial publication

DCS AI Technologies

LEGAL DOCUMENT · v1.0 · MAY 2026

Acceptable Use Policy

What you can and cannot do with the DCS Services

DCS AI Technologies L.L.C
Dubai Industrial City, Saih Shuaib 3, Dubai, UAE · dcsai.ai

abuse@dcsai.ai
dcsai.ai/aup

Acceptable Use Policy

This Acceptable Use Policy ("**AUP**") governs use of the products and services provided by DCS AI Technologies L.L.C ("**DCS**"). The AUP is incorporated by reference into the Master Services Agreement and applies to all customers, users, and visitors of any DCS service. By using the Services, you agree to comply with this AUP.

We keep the AUP short and specific. The rules below are not a wishlist — every rule exists because we have seen the underlying behaviour cause real harm. Violating the AUP can result in suspension or termination of your account, forfeiture of any prepaid fees, and referral to law enforcement where applicable.

1. Prohibited Content

You may not use the Services to create, store, transmit, or distribute content that:

- **Is illegal under applicable law** in any jurisdiction where you operate or where the affected party resides.
- **Sexually exploits or endangers minors.** This includes any form of CSAM, grooming content, or content sexualising minors. Zero tolerance; immediate termination + law enforcement referral.
- **Incites or threatens violence** against any person or group based on race, religion, ethnicity, nationality, gender identity, sexual orientation, or disability.
- **Facilitates terrorism** — recruitment, financing, planning, or glorification of terrorist acts.
- **Infringes intellectual property rights** of others, including copyright, trademark, patent, trade secret, or right of publicity.
- **Violates privacy rights**, including non-consensual sharing of personal information, intimate images, or doxxing.
- **Contains malware** — viruses, trojans, ransomware, exploit code intended to compromise systems.
- **Is fraudulent or deceptive** — phishing, social engineering, scams, impersonation of real people without consent.

2. Prohibited Activities

You may not use the Services to:

- **Send unsolicited bulk communications.** Includes spam email, unsolicited WhatsApp / SMS messages, robocalls, and unsolicited marketing across any channel. Consent must be opt-in and provable.
- **Make unauthorised credential-stuffing attempts**, password sprays, brute-force attacks against any system.
- **Conduct denial-of-service attacks** against any system, whether DCS-operated or otherwise.
- **Probe or scan vulnerabilities** of any system you do not have explicit permission to test. This includes DCS's own infrastructure unless you are operating under our public bug bounty program.
- **Reverse-engineer the Services** beyond what is permitted by applicable law, attempt to access source code, or attempt to extract model weights from inference responses.
- **Train a competing model** using outputs from the DCS Platform or DCS-provided LLM APIs without our prior written consent.
- **Operate a high-risk autonomous system** (medical diagnostics, legal advice, financial advice, life-safety decisions) without appropriate human oversight, qualified review, and disclaimers.
- **Generate deepfakes** of real people without their consent. Synthetic media depicting public figures must be clearly labelled as synthetic.
- **Generate election disinformation** — false claims about candidates, voting procedures, or election results.

3. Resource Use

You may not use the Services in a way that imposes disproportionate load on shared infrastructure:

- **Respect rate limits.** Published rate limits are not suggestions. Circumventing rate limits via multiple accounts or IP rotation is a violation.
- **Do not run mining or other computationally-intensive non-DCS workloads** on Compute workers you do not own.
- **Do not stockpile resources** — e.g., creating many empty workspaces, storing large files you never access, dispatching test jobs at high frequency.
- **Do not abuse free tiers** by creating multiple accounts to evade quotas.

4. AI-Specific Restrictions

Because DCS is an AI platform, we have specific rules about AI use:

- **No autonomous actions affecting real-world systems without human oversight** — agents must not be given unrestricted access to financial accounts, physical control systems, or other high-stakes systems without per-action human approval.
- **No undisclosed AI in customer-facing contexts** — if you use AI to respond to customers, you must disclose that AI is involved when asked or where required by law (e.g., California SB 1001).
- **No biometric inference** — using AI to infer protected characteristics (race, religion, sexual orientation, health status) from photos or behavioural data without informed consent.
- **No emotion or vulnerability exploitation** — using AI specifically to detect and exploit emotional states or cognitive vulnerabilities (e.g., loneliness, grief) for commercial purposes.
- **No social scoring** — using AI to score individuals' "trustworthiness" or similar in ways that determine access to services, in jurisdictions where this is prohibited (e.g., EU AI Act).

5. Use of Personal Data

If you process personal data through the Services:

- You are the Data Controller; DCS is the Data Processor. The DPA applies.
- You must have a lawful basis for processing under GDPR Art. 6 (or equivalent) before submitting personal data.
- You must inform data subjects about the processing as required by Art. 13/14 (or equivalent).
- You must respond to data subject rights requests within the legal timeframe (typically 30 days under GDPR).
- You must not submit special-category personal data (health, biometrics, sexual orientation, religion, etc.) unless your use case is permitted under Art. 9 and you have the additional safeguards in place.

6. Channel-Specific Rules

Different channels have specific compliance requirements that the AUP enforces:

6.1 WhatsApp Business Platform

You must comply with Meta's WhatsApp Business Policy and Commerce Policy. Notably: no broadcasts to non-opted-in users, no use of templates outside their approved category, no sale of regulated goods (firearms, alcohol in restricted jurisdictions, gambling without a licence).

6.2 Voice / SMS

You must comply with TCPA (US), CRTC rules (Canada), CSL (China), and equivalents. Provable opt-in consent is required for marketing calls / texts. STIR/SHAKEN attestation is required for US-originated voice traffic.

6.3 Email

You must comply with CAN-SPAM (US), CASL (Canada), and PECR/GDPR (EU). Each outbound email must include a working unsubscribe mechanism and accurate sender identification.

7. Enforcement

We respond to suspected AUP violations as follows:

Severity	Typical first response	Possible escalation
Minor	Written warning · 14 days to remediate	If unremediated: feature restriction
Moderate	Immediate suspension of affected feature · investigation	Account suspension if pattern continues
Serious	Immediate account suspension · investigation	Termination + forfeiture of prepaid fees
Egregious (CSAM, fraud, harm)	Immediate termination · law enforcement notification	Civil + criminal action

We reserve the right to take any action we believe appropriate to protect our users, third parties, and the integrity of the Services, including taking down content, suspending accounts, and reporting unlawful activity to authorities.

8. Reporting Violations

To report suspected AUP violations: abuse@dcsai.ai. For CSAM specifically: csam-report@dcsai.ai (monitored 24/7). Reports may be made anonymously; we respond to all reports within 24 hours.

9. Bug Bounty

Security researchers may probe DCS infrastructure under our public bug bounty program, published at dcsai.ai/security. Probing outside the bounty scope is a violation of this AUP. Good-faith security research within scope is welcomed and rewarded.

10. Changes to This AUP

We may update this AUP from time to time. Material changes will be announced at least 30 days in advance via email and the dashboard. The current version is always available at dcsai.ai/aup. Continued use of the Services after a change constitutes acceptance of the updated AUP.