OCEAN6

Ocean6 Technical Documentation

Architecture and Developer Reference

English edition 1.1

Technical review specification | Reference implementation 0.2

19 September 2026

Purpose

A developer and operator reference for Ocean6 task execution, node admission, delivery verification and service accounting. It combines an executable local reference with proposed network contracts and release requirements.

Scope

The implemented reference provides deterministic text retrieval and independent local replay. Network scheduling, authenticated receipts, model execution and settlement are specified as proposed capabilities with explicit validation requirements.

Readers

Application developers, infrastructure engineers, node operators, security reviewers and technical partners.

O6-TECH-EN-001

Contents

Availability appears beneath each chapter title. Start with the local quickstart to reproduce implemented behavior.

01 Overview and release status 3
02 System architecture 4
03 Roles and trust boundaries 5
04 Core objects and identifiers 6
05 Run the local reference 7
06 Local task specification 8
07 Results and local receipts 9
08 Verification and troubleshooting 10
09 Node admission 11
10 Capability profiles and resource claims 12
11 Scheduling and execution leases 13
12 Network task contract 14
13 Execution lifecycle 15
14 Acceptance and settlement lifecycle 16
15 Idempotency retries and cancellation 17
16 API design reference 18
17 Errors and client recovery 19
18 Identity keys and receipt signatures 20
19 Delivery verification 21
20 Model and agent workloads 22
21 Data security and isolation 23
22 Deployment and operations 24
23 Metering billing and settlement 25
24 Observability and benchmarking 26
25 Conformance and fault testing 27
26 Release criteria and open decisions 28
27 Release evidence and version history 29
28 Terminology and reading paths 30
29 Implementation and industry references 31
30 Standards and documentation governance 32

01 Overview and release status

Start here | Overview

Ocean6 is developing a service network for coordinating AI workloads, compute providers and external systems. Its proposed protocol connects a defined task to an authorized executor, an inspectable delivery record, a separate acceptance decision and an attributable charge.

This documentation describes the system architecture and the engineering requirements for that network. It also provides an executable local reference that developers can inspect and reproduce today. The reference implements deterministic text retrieval; it establishes a narrow, testable foundation for input validation and result verification.

Capability	Availability	Evidence
Local text retrieval	Implemented in reference 0.2	Python and JavaScript implementations; task schema; fixtures
Local replay and byte integrity	Implemented in reference 0.2	Independent Python replay and SHA-256 checks
Node admission and scheduling	Proposed	Protocol requirements and acceptance cases
Remote AI and GPU execution	Not demonstrated by this release	Requires a deployed runtime and workload records
Authenticated receipts and payments	Proposed	Requires identity infrastructure, signed records and reconciliation
TEE or zero-knowledge guarantees	Research option	Requires a defined trust model and implementation evidence

How to use this documentation

Start with the architecture and trust model, then run the local quickstart. The network chapters use the label Proposed specification. These chapters define requirements for a future implementation, not a public API contract. Reference 0.2 chapters describe the released local behavior.

MUST, MUST NOT, SHOULD and MAY express requirement levels when capitalized, following BCP 14. A requirement in a proposed chapter is not a claim that the control is deployed. Documentation version 1.1 and software reference version 0.2 are separate version lines. [S5]

This edition is intended for application developers, infrastructure engineers, node operators and technical reviewers. Hardware pricing, token issuance, staking and reward formulas are outside its scope.

02 System architecture

Architecture | Proposed specification

The architecture separates task coordination from workload execution. The control plane authenticates requests, evaluates policy and maintains authoritative state. Executors receive narrowly scoped leases. Acceptance and accounting services retain records independently of executor claims.

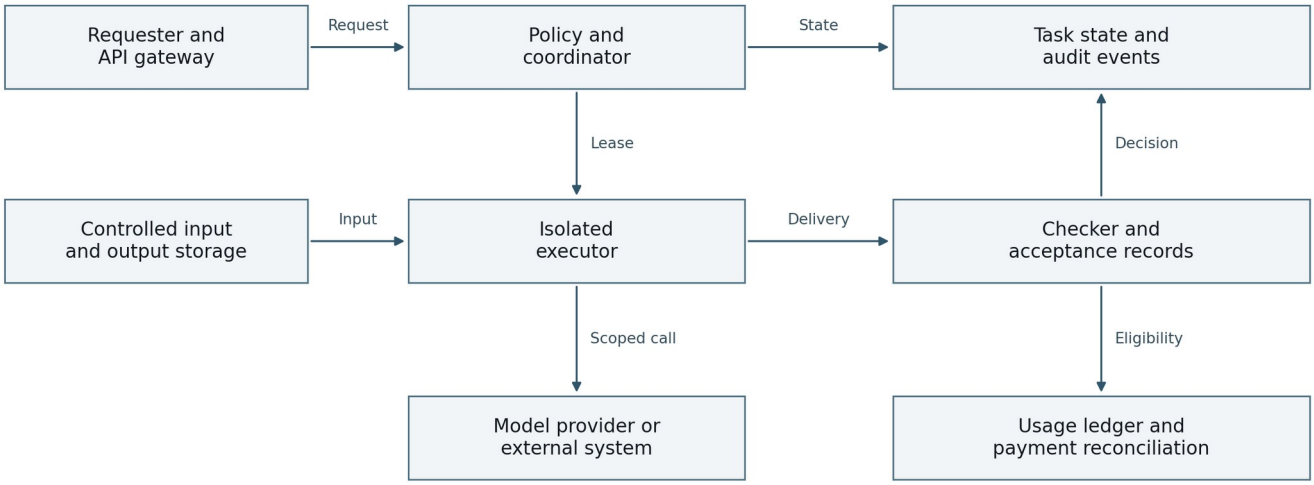


Figure 1 Proposed logical architecture. Arrows show selected interactions; they do not describe an existing deployment or a complete network topology.

Request flow

The gateway validates the request and binds it to a tenant. The coordinator freezes the workload, input digest, acceptance profile and quote, reserves eligible capacity and issues a lease. The executor retrieves authorized inputs, runs the specified workload and submits an output reference. The checker evaluates that delivery before the accounting service applies the agreed charging rules.

Deployment and chain boundary

An initial pilot can operate within one administrative domain while keeping privileges and records separate. Logical separation does not establish independent control. Any shared ownership of coordination, execution and checking must be disclosed.

Inputs, outputs, queues and operational logs remain offchain by default in this design. An onchain commitment or payment record is an optional extension subject to privacy, cost and finality review. A transaction alone cannot establish that a model ran correctly. No chain identifier, contract address or network consensus mechanism is specified as deployed.

03 Roles and trust boundaries

Architecture | Proposed specification

Every privileged operation MUST be attributable to a tenant, service principal or operator. A wallet address may support an authentication flow, but it does not by itself authorize data access, workload execution or settlement.

Role	Responsibility	Authority limit
Requester	Submit, inspect and request cancellation of tenant tasks	Cannot read another tenant's objects
Coordinator	Enforce policy, allocate capacity and maintain task state	Cannot substitute allocation for delivery evidence
Executor	Run leased work; submit outputs and usage claims	Cannot approve its own delivery or confirm payment
Checker	Evaluate a frozen acceptance profile and record evidence	Cannot change the request or transfer customer funds
Accounting service	Reconcile usage, quotes, invoices and payment evidence	Cannot infer output quality from a payment
Operator	Deploy, investigate, rotate keys and recover services	Cannot alter historical records without an audit trail

Authorization rules

AUTH-01 The server MUST derive tenant ownership from authenticated context and verify it on every object operation. Client-supplied ownership fields are untrusted. AUTH-02 Credentials MUST restrict operations, resources and validity periods; executor credentials MUST be scoped to the active task or lease.

Human administrators and service accounts SHOULD use separate credentials and audit categories. Changes to production artifacts, billing policy, settlement permissions or quarantined-node eligibility SHOULD require independent review. Review records identify the actor, approver, affected version and reason.

Trust assumptions

The proposed coordinator is authoritative for state ordering and lease generations. A trusted registry binds provider identities to keys. External model providers remain trust dependencies. Evidence signed by an executor establishes an attributable statement; it does not independently establish hardware identity, model quality or correct execution.

04 Core objects and identifiers

Architecture | Proposed specification

Separate identifiers prevent retries, checks and payments from being counted as additional customer tasks. Identifiers are opaque and scoped by the service. Clients MUST NOT derive authorization or ordering from their textual format.

Object	Purpose	Required relationship
Task	One logical customer request	Owns its attempts and frozen contract
Attempt	One execution opportunity	Belongs to exactly one task
Lease	Temporary execution authority	Binds attempt, node, generation and expiry
DeliveryReceipt	Executor statement about an output	Binds attempt, input, output and runtime
AcceptanceRecord	Evaluation of a delivery	Binds receipt, rule version and checker
UsageRecord	Measured or claimed consumption	Binds attempt, meter and deduplication key
SettlementRecord	Evidence of a payment outcome	Binds invoice, channel and external identifier
DisputeCase	Investigation or adjustment	References original records without replacing them

Immutable and mutable data

A task contract freezes execution inputs and commercial conditions. Operational fields, such as the active attempt and current state version, change through ordered events. A change to a frozen field MUST create a new task or an explicitly versioned amendment; it MUST NOT silently rewrite a request that has already been accepted.

Records use explicit units. Resource quantities are integers in named units, such as bytes. Amounts use a declared decimal scale or integer minor units. Timestamps identify UTC and their source. Digests identify both the algorithm and the exact bytes being hashed.

Relationship invariants

OBJ-01 A task has at most one attempt authorized to produce its current delivery. OBJ-02 An accepted delivery has one primary acceptance outcome under the frozen profile. OBJ-03 Repeated notifications MUST NOT create additional charges or payments. Review and reversal records are appended to the original evidence chain.

05 Run the local reference

Build | Reference 0.2

Run a complete task on your computer, inspect the result and verify it independently. This path requires no account, wallet or network endpoint. The program uses the Python standard library and performs deterministic line retrieval on a local CPU.

Prerequisites

The release declares Python 3.9 or later; the recorded release environment is Python 3.12.14. Other versions are not certified by that record. Download the engineering reference from the documentation site and extract it, or use the reference-v0.2 directory in the accompanying documentation package. [E1][E2]

```
cd reference-v0.2
python3 reference_runtime.py run task.json result.json receipt.json
python3 reference_runtime.py verify task.json result.json receipt.json
python3 test_reference.py
```

The first command writes a result and an unsigned receipt. The second recomputes the task and checks the original input and output byte digests. The test command exercises validation and failure conditions. Successful run and verification commands exit with code 0; handled validation or file errors exit with code 1.

Inspect the files

File	What to inspect
task.json	Operation, input document, search terms and result limit
result.json	Ranked matches, original line numbers and total matching lines
receipt.json	Input and output digests; local times; signature and settlement are null
task.schema.json	Structural constraints; runtime checks add further restrictions

The run command overwrites the specified result and receipt files. Use distinct input, result and receipt paths. Keep the original task bytes if you intend to verify the receipt later.

Compare a browser result

```
python3 reference_runtime.py compare task.json browser-result.json
```

Export the task and result from the local task lab first. compare checks semantic equivalence through Python replay; it does not require a receipt or authenticate the executor. No model inference, GPU execution or remote node operation is exercised by this guide.

06 Local task specification

Build | Reference 0.2

The input is a JSON object with exactly five fields. The local runtime performs explicit validation rather than invoking a general JSON Schema engine. Its structural schema uses JSON Schema Draft 2020-12. [S1]

Field	Type	Constraint
version	String	Exactly "0.2"
operation	String	Exactly "literal-line-search"
document	String	Valid Unicode scalar text; 1 to 20,000 UTF-8 bytes
terms	Array of strings	1 to 16 entries; each raw term contains 1 to 80 Unicode code points
top_k	Integer-valued JSON number	1 to 10; booleans are rejected
Additional fields	Not permitted	Missing or unknown fields fail validation

Normalization and ranking

1. Reject isolated Unicode surrogate code points. Convert CRLF and CR line endings to LF before splitting the document. Line numbering begins at 1; empty lines retain their positions.
2. Reject a term containing CR or LF. Trim only leading and trailing ASCII spaces and tabs, then fold ASCII A through Z to lowercase. Reject empty normalized terms and remove duplicates in first-occurrence order.
3. Match each normalized term as a literal substring of each line after ASCII case folding. Other scripts remain literal. There is no tokenization, Unicode normalization or language-dependent case folding.
4. Set score to the number of distinct normalized terms matched in the line. Sort by descending score, then ascending original line number. Return at most top_k entries.

Input parsing

JSON object member names must be unique for interoperable input. The Python parser rejects duplicates; the browser editor uses native JSON.parse, which retains the last duplicate. Python also rejects nonstandard NaN and Infinity constants.

The schema cannot by itself enforce the UTF-8 byte limit or the nonempty-after-trimming rule. A schema-valid object can still fail runtime validation. The CLI limits the complete task file to 150,000 bytes, separately from the 20,000-byte document limit.

07 Results and local receipts

Build | Reference 0.2

Save this task as task-demo.json in the reference directory, or use the supplied file in the documentation package. Every line matches at least one term; the second line ranks first because it matches two distinct terms.

```
{
  "version": "0.2",
  "operation": "literal-line-search",
  "document": "Alpha\nalpha delivery\ndelivery",
  "terms": ["alpha", "delivery"],
  "top_k": 2
}
```

Rank	Line	Score	Matched terms
1	2	2	alpha, delivery
2	1	1	alpha

total_matches is 3 before truncation. The line field is one-based. Each match includes the original line text and matched_terms in normalized term order. No match produces an empty matches array and total_matches equal to 0.

Receipt fields

Field	Value or meaning
version	"0.2"
execution	"local-cpu"
implementation	"ocean6-python-reference/0.2"
task_sha256 and result_sha256	SHA-256 of the exact original file bytes
started_at and finished_at	Local UTC claims with six fractional digits and a Z suffix
signature and settlement	Both null

verify first compares the result with a replay, then checks the exact receipt field set, fixed values, file digests, timestamp format and time ordering. JSON whitespace and member order do not affect semantic comparison, but any byte change affects the relevant receipt digest.

A local receipt is not an authenticated execution certificate. An attacker who replaces the task, result and receipt together can construct a self-consistent bundle. These records do not identify a remote executor or prove a payment.

08 Verification and troubleshooting

Build | Reference 0.2

Reproduce an integrity failure

```
python3 reference_runtime.py run task-demo.json result-demo.json receipt-demo.json
python3 tamper_result.py
python3 reference_runtime.py verify task-demo.json result-tampered.json receipt-demo.json
```

The documentation package includes tamper_result.py. It copies result-demo.json, increases the first match score and writes result-tampered.json. The final command is expected to exit with code 1 and REPLAY_MISMATCH. The original result remains unchanged.

Error or symptom	Cause and corrective action
INVALID_FIELDS	Supply exactly the five task fields; remove unknown fields.
INVALID_DOCUMENT	Check scalar Unicode and the UTF-8 byte length.
INVALID_TERM or EMPTY_TERM	Check raw length, line breaks and content after ASCII trimming.
INVALID_TOP_K	Use an integer-valued number from 1 through 10.
REPLAY_MISMATCH	The parsed output differs from deterministic replay. Preserve the mismatch for investigation.
TASK_DIGEST_MISMATCH	Task bytes changed. Restore the original bytes or create a new run.
RESULT_DIGEST_MISMATCH	Result bytes changed even if its parsed meaning did not.
INVALID_RECEIPT_SCOPE	The receipt has an unsupported scope or non-null signature or settlement.
INVALID_TIME_ORDER	The declared finish time precedes the declared start time.

Cross language verification

```
python3 test_cross_runtime.py
```

This optional command requires Node.js on PATH and compares 80 generated inputs with fixed seed 6. The recorded release environment used Node.js 24.19.0. It checks agreement between implementations; it is not a load test or a proof that the specification is free of defects.

verify and compare limit result files to 150,000 bytes. verify limits receipts to 10,000 bytes. The CLI also requires distinct resolved file paths. Operating-system errors can include environment-specific text and are not a stable network error taxonomy.

09 Node admission

Operate | Proposed specification

A node becomes schedulable only after its identity, runtime and workload capabilities have been evaluated. A hardware declaration is a claim to inspect. A successful challenge demonstrates the properties tested by that challenge, not the execution history of every customer task.

Stage	Required evidence	Decision
Registration	Provider identity, node identifier, accountable operator and registered key	Create a declaration under review
Environment inspection	OS, CPU or GPU, memory, drivers and runtime artifact digest	Match a versioned support profile
Workload qualification	Fixed test inputs, outputs, runtime version and failure logs	Qualify specific workload profiles
Isolation review	Privilege, mount, network and resource-limit checks	Confirm the execution boundary
Continuing checks	Health observations, service probes, capacity changes and delivery failures	Retain, restrict or suspend eligibility

Eligibility lifecycle

The proposed states are declared, under_review, eligible, draining, quarantined and revoked. Only eligible nodes may receive new assignments. Draining stops new work while permitted work finishes. Quarantine suspends eligibility pending investigation. Revocation invalidates admission; reinstatement requires a new review record.

NODE-01 Runtime changes, key revocation, stale critical observations and resource changes MUST trigger an eligibility review. A previous pass MUST NOT substitute indefinitely for current evidence. NODE-02 Admission results MUST identify the test version, observed environment, decision and limitations.

Scope of checks

The organization of this process draws on io.net's resource-verification documentation and Aethir's separation of registration, standby and in-service checks. Ocean6 does not inherit their algorithms, operating intervals, device requirements or verification guarantees. [R1][R2]

Minimum hardware, challenge design, check frequency and oversubscription policy remain release decisions. They must follow workload measurements and failure analysis. No certified Ocean6 hardware support matrix is published by reference 0.2.

10 Capability profiles and resource claims

Operate | Proposed specification

A capability profile states what a node is configured to run and links that claim to admission evidence. Keep installed capacity, currently allocatable capacity and observed workload performance as separate measurements.

Field group	Required content	Validation rule
Identity	provider_id, node_id, key_id	Resolve through the trusted registry
Runtime	runtime_version, artifact_digest	Pin the executable artifact
Location	region and permitted data scopes	Match task policy; record the source of the claim
Hardware	CPU or GPU model, count, memory and VRAM	Use explicit units and inspection evidence
Workloads	Supported profile names and versions	Admit each profile independently
Capacity	Available resources and reservations	Reconcile against active leases
Connectivity	Network policy and service probes	Distinguish reachability from service readiness
Evidence	Test identifiers, timestamps and evidence references	Reject expired or incompatible evidence

Resource accounting

CAP-01 Capacity reservations MUST be updated atomically with lease allocation or through an equivalent consistency mechanism. Two coordinators MUST NOT reserve the same exclusive resource concurrently. Shared accelerators require an explicit partitioning and interference policy.

A device count is not a portable unit of compute. Qualification SHOULD name the model, runtime, precision, input size and concurrency used in a measurement. A node qualified for a small retrieval task is not thereby qualified for model inference or training.

Operator actions

Operators should drain a node before planned maintenance, preserve failure evidence before redeployment and rerun affected qualification tests after changing drivers or runtime artifacts. Emergency isolation may interrupt work; the coordinator must retain the interrupted attempt and its billing context.

The reference bundle contains an unpopulated capability template. It is a documentation aid, not a registered provider, a completed hardware test or evidence of available network capacity.

11 Scheduling and execution leases

Protocol | Proposed specification

Scheduling first filters for policy compliance, then ranks eligible capacity. Mandatory filters include tenant authorization, data location, workload compatibility, resource requirements, network access, budget and deadlines. Unavailable capacity MUST NOT cause a silent relaxation of these constraints.

Lease contract

Field	Required meaning
task_id and attempt_id	The logical request and this execution opportunity
node_id and lease_id	The selected node and unique lease
fencing_token	Monotonically increasing generation for the task
expires_at	Authority expiry evaluated by the coordinator
scope	Permitted objects, operations and resource reservation

LEASE-01 Allocation MUST create the reservation, attempt and lease within one consistency boundary. LEASE-02 State-changing submissions MUST match the current attempt and fencing token. A recovered executor with an older token cannot overwrite the current delivery or cause a payment.

Renewal and partitions

Only a current, unexpired lease may be renewed. The authority records the prior expiry, new expiry and renewing identity. Missing heartbeats stop new allocation and initiate a health review; lease expiry determines the existing authority to commit work.

An executor without valid authority MUST NOT begin a new external side effect. It should stop or isolate ongoing output according to its workload profile. Downstream systems must also enforce generation or idempotency controls to prevent stale actions; a coordinator cannot guarantee global exactly-once behavior across systems it does not control.

Late delivery

A receipt admitted after lease expiry or supersession MUST NOT become the active delivery. The coordinator records the admission time, state version and generation. Later expiry does not invalidate a delivery admitted under valid authority. Duplicate submission returns the original record. Lease duration, renewal margin and clock tolerance require versioned configuration and fault tests.

12 Network task contract

Protocol | Proposed specification

The proposed network contract is distinct from the five-field local task. Declarative workload definitions allow validation before resources are reserved; Nosana's task definition and validation guides provide a useful documentation pattern. [R3][R5]

Field	Owner	Contract
protocol_version	Client; validated by server	An explicitly supported version
workload_profile	Client	Immutable name and version
input_ref and input_digest	Client; authorized by server	Controlled object reference and digest of original bytes
runtime_digest	Policy or client profile	Pinned executable artifact
constraints	Client; bounded by policy	Resource, region, egress and deadline requirements
acceptance_profile	Client or product profile	Frozen evaluation rules and version
quote_ref	Accounting service	Valid quote with budget and failure-charge policy
max_attempts	Client; bounded by policy	Positive, capped attempt count
idempotency_key	Client	Unique within tenant and operation scope
task_id and tenant_id	Server	Opaque identity and authenticated ownership
active_attempt_id	Server	Current execution selection, or null
resolution	Server	open, completed, failed, cancelled or disputed

Validation and immutability

TASK-01 Reject unknown protocol versions, unsupported workload profiles and unrecognized fields. Resolve ownership and object access before acceptance. TASK-02 Freeze the workload, inputs, runtime, acceptance profile and quote before the first executor acceptance.

A request fingerprint covers every client-controlled field that affects execution, access, acceptance or cost. Its canonicalization profile must be versioned. Raw input digests continue to bind the original object bytes, independently of the task-object encoding.

Network field types, maximum sizes, profile registries and canonical encodings must be frozen in a reviewed schema before integration. The proposed semantics require engineering approval before a stable API contract is published.

13 Execution lifecycle

Protocol | Proposed specification

Each attempt has its own execution state. Retrying creates a new attempt identifier under the same task; it never reopens the failed attempt. Delivery, acceptance and payment are separate axes so that operational dashboards do not confuse these outcomes.

Attempt state	Meaning	Permitted next states
submitted	Attempt recorded; executor acceptance pending	accepted, rejected, cancelled
accepted	Executor accepted valid work and authority	running, failed, cancelled, expired
running	Work has started under the lease	delivered, failed, cancelled, expired
delivered	An authorized output submission was durably recorded	None on this execution axis
rejected	Executor did not accept the attempt	None
failed	Execution failed with a recorded reason	None
cancelled	Cancellation confirmed for the attempt	None
expired	Execution authority or deadline expired	None

State invariants

STATE-01 A transition MUST compare the current state version, check the active lease where applicable and commit its event atomically. STATE-02 Terminal execution records are immutable. Later evaluation or dispute records reference them. STATE-03 A delivered attempt is not automatically an accepted result or a paid task.

Delivery submission checks structural validity, authorization, current attempt and accessible output metadata. Acceptance then checks identity evidence, content integrity and the frozen evaluation rules. Content that later fails evaluation remains associated with the delivered attempt for investigation.

Event ordering

Every state event includes event_id, task_id, attempt_id where applicable, entity_type, previous_state, next_state, sequence, recorded_at and actor identity. Sequence is strictly increasing within its declared entity stream; timestamps do not define a global event order. Consumers deduplicate by event_id and reconcile gaps against authoritative state.

This separated state model refines the earlier design draft, which presented validated and settled in one linear sequence. It changes proposed network semantics only; reference 0.2 has no network state machine.

14 Acceptance and settlement lifecycle

Protocol | Proposed specification

Acceptance outcomes

An acceptance record evaluates one delivery under one frozen rule set. Its processing status is pending until evidence is sufficient to produce passed, failed or disputed. Missing evidence remains pending until the applicable deadline; it must not default to passed.

Task resolution	Condition	Effect
open	Work, recovery or acceptance is still permitted	A policy-compliant attempt may be scheduled
completed	An eligible delivery passed acceptance	Close execution; account for charges separately
failed	No accepted delivery and retry policy is exhausted	Close with a reason and usage record
cancelled	Cancellation won the state transition and is confirmed	Close further execution authority
disputed	An unresolved acceptance or responsibility issue closes normal processing	Open a case; retain evidence and fee holds

RES-01 Closing a task MUST atomically prevent new attempts and retire its remaining execution authority. A later complaint or refund creates a separate case or adjustment; it does not rewrite a closed task's history. An acceptance failure may permit a new attempt only if the frozen retry policy allows it.

Payment outcomes

Maintain a separate accounting status, such as not_due, eligible, pending, confirmed, failed or reversed. eligible means contractual conditions for a charge have been evaluated; confirmed requires the payment channel's confirmation evidence. A successful workload may still have a failed payment, and a cancelled workload may incur an agreed usage charge.

PAY-01 The executor MUST NOT create acceptance or payment confirmations. PAY-02 Duplicate payment notifications MUST have one ledger effect. PAY-03 Adjustments and refunds MUST preserve the original invoice and external payment reference.

Acceptance ownership

The checker records its identity, rule version, code digest, evidence references and decision. When the same organization operates the executor and checker, documentation must identify that relationship. A separate software process is not an independent third-party verification service.

15 Idempotency retries and cancellation

Protocol | Proposed specification

Submission idempotency

IDEM-01 Scope the idempotency key by authenticated tenant and operation. Persist the key, request fingerprint and task identifier atomically. Repeating the same key and fingerprint returns the original task. Reusing the key with different content returns a conflict and leaves the original contract unchanged.

The key retention period must cover the supported request-retry, task and reconciliation windows. Before retrying after a transport timeout, a client should query the task or repeat the original request with the same key. A new key can create a new task.

Retry eligibility

Failure class	Required handling
Temporary transport or runtime failure	Retry only a declared safe workload within attempt, deadline and budget limits.
Invalid input or denied permission	Do not automatically retry unchanged input.
Unknown external side effect	Query or reconcile the external system before repeating an action.
Superseded executor returns	Retain late evidence; reject state changes from an old generation.
Delivery acknowledgment lost	Return the recorded delivery; do not create another acceptance or charge.

RETRY-01 A new attempt MUST have a new identifier and lease generation. The old attempt remains visible. Resource consumption on a stale node can continue during a partition, but its output cannot replace the current authorized result.

Cancellation races

CANCEL-01 Acknowledging a cancellation request means it has been recorded, not that execution has stopped. The coordinator orders cancellation confirmation against delivery using atomic state transitions. If delivery wins, the cancellation response must expose that state and apply the frozen acceptance and fee policy.

For a confirmed cancellation, retire execution authority and retain any incurred usage. A device gateway must distinguish command acknowledgment from observed physical completion. Emergency stops and safety interlocks remain the responsibility of the local control system.

Cross-system exactly-once execution is not a guarantee of this design. External actions require their own idempotency, reconciliation and compensation rules.

16 API design reference

Protocol | Proposed specification

The paths below define an interface review surface. No production host, access token flow or callable Ocean6 network API is supplied by this release. An integration release must publish its OpenAPI contract, authentication process, limits and compatibility policy together.

Operation and proposed path	Success semantics
POST /v1/tasks	202: request admitted; returns a task identifier and state, not execution success
GET /v1/tasks/{id}	200: authorized state snapshot with a state version
POST /v1/tasks/{id}/cancel	202: request recorded; 200 may return a previously recorded decision
GET /v1/tasks/{id}/events	200: ordered events with an opaque continuation cursor
GET /v1/tasks/{id}/deliveries	200: authorized delivery metadata; object access remains separately controlled
POST /v1/providers/nodes	202: registration under review, not scheduling eligibility
POST /v1/attempts/{id}/receipt	201: newly stored receipt; 200: an identical previously stored receipt

Transport and request controls

API-01 Public transport MUST authenticate the service and protect credentials in transit. The server MUST validate content type, parse limits, nesting depth and duplicate JSON member names before applying business rules. Unknown fields or unsupported versions must produce explicit errors.

API-02 State-changing operations MUST enforce authorization and a defined concurrency policy. Task creation uses idempotency; receipt submission uses delivery identity and the active lease generation. A request identifier supports support and audit correlation without exposing credentials.

Pagination and asynchronous delivery

Event cursors are opaque and bound to the authorized stream. The contract must document cursor expiry, ordering and recovery from an expired cursor. Clients should reconcile a snapshot and resume events without treating repeated notifications as new work.

Webhooks, if introduced, require destination validation, signed payloads, event identifiers, retry limits and a replay window. A receiver acknowledges durable receipt before asynchronous processing. Callback delivery order is not a substitute for the event stream's sequence.

17 Errors and client recovery

Protocol | Proposed specification

Use a stable machine-readable error code and a human-readable explanation. The proposed error envelope follows RFC 9457 Problem Details, with error_code and request_id extensions. Details must be sanitized and must not reveal another tenant's objects. [S3]

HTTP status	Condition	Client action
400	Malformed encoding, JSON or duplicate member names	Correct the request
422	Well-formed request violates a supported contract	Inspect the field error and revise the input
401	Missing or invalid authentication	Obtain valid credentials through the supported flow
403	Authenticated caller lacks permission	Request the required scope
404	Object absent or not visible to this caller	Check the identifier without inferring tenant existence
409	Idempotency conflict, stale generation or invalid transition	Read authoritative state; resolve the conflict
429	Rate limit exceeded	Honor Retry-After and use bounded backoff
503	Temporary service or capacity unavailability	Retry only within the task and budget policy

Illustrative error object

```
{
  "type": "about:blank",
  "title": "Conflict",
  "status": 409,
  "detail": "The idempotency key is bound to another request.",
  "error_code": "IDEMPOTENCY_CONFLICT",
  "request_id": "illustrative-request-id"
}
```

This object illustrates the draft format. Network error codes remain subject to contract approval. The local CLI error strings in the troubleshooting chapter are a separate reference interface.

Recovery without duplicate effects

A timeout does not show whether the server committed a request. Recover with the original idempotency key or recorded task identifier. Retry delays should include jitter and a ceiling; a 429 or 503 response does not authorize exceeding a deadline or creating a second payment.

Preserve the request identifier and sanitized diagnostic context when escalating an error.

18 Identity keys and receipt signatures

Security | Proposed specification

An authenticated receipt requires a trusted binding between provider identity and public key. The verifier MUST obtain keys from that registry or another approved trust source; a public key embedded in an untrusted receipt is not sufficient.

Key lifecycle

KEY-01 A key record includes provider_id, key_id, algorithm, public key, permitted scope, activation time, expiry and revocation state. Rotation creates a new record. Historical verification retains the applicable key version and revocation decision. A claimed signing time cannot independently establish that a compromised key was safe at that time.

Candidate signature profile

Ed25519 and the JSON Canonicalization Scheme are candidates for protocol review. Reference 0.2 implements neither. Its helper named canonical normalizes parsed numeric values for comparison and is not an RFC 8785 implementation. [S2][S4]

```
unsigned_envelope = {
  "profile": "ocean6-receipt-draft-1",
  "key_id": registered_key_id,
  "algorithm": "Ed25519",
  "payload": receipt_payload
}
signing_bytes = ASCII("OCEAN6_RECEIPT_DRAFT_1")
              + byte(0x00)
              + JCS_UTF8(unsigned_envelope)
```

The notation above is pseudocode. The proposed signature covers the profile, key identifier, algorithm and full payload. The final envelope adds a signature field. The protocol review must freeze encoding, Base64url rules, field limits, library choices and positive and negative test vectors before implementation claims are made.

Binding and replay resistance

SIG-01 The payload MUST bind network_domain, task_id, attempt_id, lease_id, fencing_token, request digest, output digest and validity conditions. Verification checks authority at recorded delivery admission and deduplicates receipt identity even when the signature is valid.

SIG-02 If key lookup or revocation evaluation is unavailable, defer identity-dependent acceptance. Do not silently downgrade to a digest-only check. A signature establishes an attributable statement within its trust assumptions; hardware provenance and computation correctness require additional evidence.

19 Delivery verification

Security | Proposed specification

Verification is a sequence of distinct checks with recorded outcomes. A checker must state what it verified and which trust sources it relied on. A single verified badge should not conceal whether identity, integrity, execution environment or business quality was evaluated.

1. Validate the protocol version, field set, types and size limits. Reject ambiguous encodings and unsupported profiles.
2. Resolve tenant, task and attempt. Check the coordinator's recorded delivery admission against the lease and generation valid at that point; reject cross-task or stale submissions.
3. Resolve the trusted key record and verify the complete signature profile, including revocation and replay rules.
4. Retrieve authorized input and output bytes and recompute the declared digests. Record unavailable evidence separately from a mismatch.
5. Evaluate the frozen acceptance profile: deterministic replay, model evaluation, customer approval or another explicitly defined method.
6. Persist an acceptance record with the checker identity, rule version, code digest, outcome, reasons and evidence references. Send only the appropriate accounting eligibility event.

Check	What it establishes	What remains outside scope
Digest comparison	Consistency with specified bytes	Publisher or executor identity
Registered-key signature	An attributable signed statement	Truth of every execution claim
Deterministic replay	Conformance to a fixed algorithm	Arbitrary model inference or GPU use
Model evaluation	Performance on a declared evaluation method	Universal output correctness
Environment attestation	Properties covered by the attestation trust model	Business suitability of the output

Missing evidence and disputes

VERIFY-01 An unavailable output, registry or evaluation dataset cannot yield a pass. Retain a pending decision until the defined deadline, then follow the failure or dispute policy. A successful HTTP response or well-formed hash string is not a completed integrity check.

If the same actor can replace the input, output, receipt and manifest, self-consistency alone provides no independent trust anchor. Pin the original task and approved release through an authenticated channel. The same limitation applies to the local reference checksum file.

20 Model and agent workloads

Security | Proposed specification

AI workloads require a workload-specific acceptance method. Deterministic retrieval can be replayed exactly under its published rules. Generative inference may vary with sampling, hardware, numerical precision and engine behavior, even when a seed is recorded.

Model execution profile

Profile element	Required record
Model identity	Model name, immutable weight reference or provider version, and usage rights
Runtime	Engine version, artifact digest, precision and quantization
Request	Input reference, prompt-template digest and sampling configuration
Tools	Allowed tools, argument constraints, authorization scope and budget
Evaluation	Dataset version, method, thresholds, adjudication and failure handling
External service	Provider request identifier, model version and reconcilable usage

AI-01 Freeze acceptance criteria before execution. Evaluate output structure, business quality and execution provenance separately. Exact string equality is appropriate only where the workload profile justifies it. An external model response remains inside the external provider's trust boundary.

Agent authorization

Agent-generated content MUST NOT expand tool permissions. Treat prompts, retrieved documents and model outputs as untrusted data. A tool gateway enforces allowed operations, destinations, data scope and spending limits independently of the model. High-impact external actions require the workflow's explicit authorization and reconciliation policy.

Confidential and verifiable execution

TEE attestation and zero-knowledge proofs are possible extensions, not guarantees of this release. Any extension must identify the code or computation covered, input binding, hardware or cryptographic trust assumptions, verifier behavior, freshness and failure handling.

Ritual's current documentation explains replicated and delegated execution and the handling of asynchronous work. Ocean6 uses this as a reference for documenting execution boundaries; it does not claim Ritual integration or inherit its deployed mechanisms. [R4]

For physical equipment, local safety systems retain authority over interlocks, emergency stops and manual takeover. A software task result must distinguish command receipt from observed completion.

21 Data security and isolation

Security | Proposed specification

The threat model covers customer data, credentials, task state, execution capacity and financial records. Potential adversaries include malicious requesters, compromised executors, stolen service credentials and colluding operators. Admission does not make a workload or node permanently trustworthy.

Threat	Required control	Residual dependency
Cross-tenant access	Check ownership on every object operation; use scoped credentials	Negative authorization tests
Workload escape	Nonprivileged runtime, constrained mounts, quotas and restricted egress	Isolation testing for each runtime
Prompt-driven tool misuse	Policy enforcement outside the model; validate tool arguments	Application-specific action review
SSRF and arbitrary downloads	Controlled object references; validate destinations and redirects	DNS changes, metadata endpoints and egress enforcement
Forged or replayed receipts	Trusted key registry, complete binding and generation checks	Key custody and revocation policy
Duplicate charges	Stable usage and payment identifiers; reconciliation	External payment-channel behavior
Executor and checker collusion	Disclose control; retain evidence; use independent review where required	Actual organizational independence
Artifact compromise	Pinned artifacts, dependency inventory and vulnerability response	Build and distribution trust

Data lifecycle

DATA-01 Declare processing location, permitted providers and retention before accepting sensitive data. Keep raw inputs and outputs off public chains by default. Logs should contain identifiers, classifications and digests instead of customer content or credentials; even a digest may reveal low-entropy content through enumeration.

Secrets MUST NOT appear in task JSON, prompt text, browser source or routine logs. Issue short-lived, task-scoped access where possible. Define encryption in transit, storage encryption and key custody for the actual deployment; this specification does not certify a provider's controls.

Deletion policy must cover primary storage, caches, logs and backups with explicit time bounds. Record the difference between removing active access and expiring recoverable backups. No immediate-erasure or regulatory certification claim follows from this design.

22 Deployment and operations

Operate | Proposed specification

Pilot topology

A controlled pilot may use a coordinator, transactional state store, object storage, work queue and isolated executors. The database and queue products are implementation decisions. Selection must satisfy atomic state changes, tenant isolation, backup recovery and audit requirements.

OPS-01 Authoritative state lives in durable records. Queue messages are notifications, not the sole source of task truth. Use a transactional outbox or equivalent method to recover when a state commit succeeds but notification delivery fails. Assume at-least-once delivery and deduplicate consumers.

Incident	Required response
Coordinator restart	Recover durable task state; reconcile leases and pending outbox events.
Executor disconnect	Stop new allocation; evaluate lease expiry and safe retry conditions.
Object store outage	Keep acceptance pending; retry authorized reads within the deadline.
Checker failure	Preserve delivery; do not let the executor self-approve.
Payment notification loss	Query the original external payment identifier before any new action.
Registry or key incident	Quarantine affected authority, preserve evidence and apply revocation policy.

Change management

OPS-02 Pin every released executable to a version, digest and dependency inventory. Publish migration requirements, compatibility impact and rollback limits with the artifact. In-flight tasks retain their frozen workload and acceptance profiles during an upgrade.

Drain nodes before planned maintenance. Recovery exercises must include state, authorization, evidence objects and accounting relationships; restoring a database alone does not establish complete service recovery.

Production operating decisions

Before offering a paid service, define on-call ownership, incident severity, customer notification, backup retention, recovery point and recovery time objectives. Set values from the actual deployment and exercises. Static website availability or a local test pass does not establish these service objectives.

23 Metering billing and settlement

Operate | Proposed specification

The billing model separates resource consumption, acceptance, payment confirmation and network incentives. This specification does not introduce or modify Ocean6 token supply, staking, reward or release parameters.

Quote and budget

BILL-01 A quote MUST fix the asset or currency, unit, rate, validity period, budget limit, treatment of failures and cancellations, and dispute terms before execution is accepted. GPU time, model tokens and completed service calls are distinct units and cannot be combined without an explicit conversion.

The coordinator must reserve or otherwise control budget before authorizing work. Budget exhaustion rejects new allocation or stops further workflow steps according to the accepted policy. It does not create an unlimited credit facility.

Usage records

A UsageRecord identifies task, attempt, meter, measurement version, interval, unit, quantity and deduplication key. Executor-reported usage is a claim until evaluated. External-model usage should reconcile to provider request records. GPU measurement must state its observation method and treatment of shared resources; online wall-clock time alone is not effective compute.

Condition	Accounting behavior
Delivery passes acceptance	Apply the frozen quote to confirmed billable usage.
Failure or cancellation	Apply the agreed policy and retain reason and usage separately.
Automatic retry	Preserve attempt costs and avoid duplicate logical-task charges.
Repeated payment notice	Apply one ledger effect for the same external payment identifier.
Dispute or refund	Append a case and adjustment; preserve the original record.

Payment finality

BILL-02 Use fixed decimal precision or integer minor units with explicit rounding rules. Do not accumulate monetary balances in unconstrained binary floating point.

For a future onchain payment path, publish chain, contract, asset, upgrade authority and confirmation policy. Broadcast, inclusion and finality are separate states; reorganization and failed transactions require reconciliation. The local reference always writes settlement as null and makes no payment.

24 Observability and benchmarking

Operate | Proposed specification

Measure logical tasks, execution attempts, node health and financial events separately. Retrying a task must not inflate customer demand, and subsidized prices must not be presented as the full cost of delivering a workload.

Metric	Definition	Required context
Task acceptance rate	Passed tasks divided by admitted tasks closed in the measurement set	Include failed, expired, cancelled and disputed outcomes; list pre-admission rejection separately
Delivery latency	Time from submission to eligible delivery	p50 and p95, sample count, failures and incomplete tasks
Queue and execution time	Waiting and running intervals recorded separately	Clock source, skew and missing events
Inference performance	Time to first token and output token rate	Model, precision, engine, input and output lengths, concurrency
Observed node health	Successful probes divided by expected probes	Probe definition, interval, observation window and missing observations
Cost per task	Allocated service cost and customer charges reported separately	Retries, bandwidth, model fees, subsidies and currency

Avoid biased reports

METRIC-01 Report a fixed submission cohort alongside completed-task statistics, including still-running and pending-acceptance tasks. Results based only on completed work can hide slow or stalled tasks. Do not merge different models, hardware or acceptance profiles into an unexplained ranking.

Closed resolution is immutable; later disputes should be reported through separate correction or dispute metrics. Publish the time window, inclusion rules, denominator and exclusions for every rate. A task failure caused by a retry limit still belongs to the original logical task.

Benchmark record

Include date, artifact digest, hardware, driver, runtime, model or dataset digest, warm-up policy, concurrency, run count, error categories and raw per-run records. Report unsuccessful runs alongside latency percentiles.

The reference bundle's benchmark-template.csv contains headers only. Its 9 tests and 80 cross-runtime comparisons are functional checks, not measurements of throughput, GPU capacity, cost savings or production availability.

25 Conformance and fault testing

Assurance | Proposed specification

The cases below are release requirements for a network pilot. They are not recorded passes. Each result must identify the build, configuration, environment, input, expected behavior, observed behavior, raw logs and reviewer.

ID	Scenario	Pass condition
NET-01	Identical request repeated	Same task identifier; no additional reservation
NET-02	Same key with changed content	Conflict; original task remains unchanged
NET-03	Concurrent coordinators allocate	One current attempt and consistent resource reservation
NET-04	Partitioned executor returns late	Old generation cannot commit or trigger payment
NET-05	Delivery acknowledgment is lost	Original delivery returned; no duplicate acceptance
NET-06	Cancellation races with delivery	Atomic ordering; no contradictory closed outcomes
NET-07	Cross-tenant object access	Denied without leaking private object existence
NET-08	Replayed or invalid-key receipt	Rejected with a recorded verification reason
NET-09	Queue messages duplicate or reorder	State constraints hold; sequence gaps reconcile
NET-10	Commit succeeds and notification fails	Recovery republishes the event without losing work
NET-11	Retry and payment notifications repeat	Attempt costs reconcile; ledger effect occurs once
NET-12	Untrusted workload or download target	No unauthorized mount, egress or internal-network access
NET-13	Rules change during execution	Existing tasks retain their frozen profiles
NET-14	Restore or rollback is exercised	State, permission, evidence and accounting links recover

Review and release evidence

Functional review belongs to the implementing team. Authorization, isolation and key-handling cases require a security review appropriate to the deployment. Physical actions and real payments require the responsible operator to confirm recovery and reversal behavior.

TEST-01 Preserve failed cases and exclusions in the release record. Passing a subset must not be described as complete network conformance. TEST-02 Independent audits must state auditor, scope, version, date, findings and remediation status. No independent audit is asserted for reference 0.2.

26 Release criteria and open decisions

Assurance | Proposed specification

Release stages are defined by evidence. Calendar targets do not replace an installable artifact, an operating endpoint or a reproduced acceptance result.

Stage	Required evidence	Current scope
Local reference	Source, explicit contract, runnable fixtures and verification	Reference 0.2 available
Controlled network pilot	Authenticated endpoint, node runtime, support matrix and NET test results	Proposed; not supplied by this release
Paid service	Pricing, data terms, metering reconciliation, incident and recovery procedures	Requires implementation and operating evidence
Open provider or chain extension	Published control model, admission rules, deployed contracts where relevant and security review	Separate approval and evidence required

Decisions to close before a pilot

Decision	Required basis	Suggested accountable function
Initial remote workload	Customer need, acceptance dataset, cost and access constraints	Product and engineering
Hardware support	Compatibility tests and raw workload records	Infrastructure engineering
Lease and retry limits	Partition tests and external side-effect analysis	Backend architecture
Identity and signatures	Threat model, registry design, library review and test vectors	Security engineering
Billing and data policy	Metering source, payment reconciliation and retention requirements	Service operations
Chain integration	Specific business requirement, privacy, finality and control costs	Architecture and governance

The functions above are responsibility proposals, not appointments of named personnel. Each approved decision needs a record identifier, owner, rationale, affected protocol version, approval date and supporting evidence.

A public release should connect each claimed capability to a stable documentation page, a versioned artifact and its validation record. Preview features must retain their availability label until their acceptance conditions have been met.

27 Release evidence and version history

Assurance | Reference 0.2 and documentation 1.1

The baseline is the engineering reference published on 19 September 2026. Its VALIDATION.md records self-published engineering checks. The record is evidence of the stated test scope, not an independent assessment of the network. [E2][E3]

Recorded check	Result	Scope
Python unit and CLI checks	9 tests passed	Validation, boundaries, Unicode, ordering and tampering
Python and JavaScript comparison	80 cases passed; seed 6	Agreement for the generated sample set
Extracted release quickstart	run, verify and compare passed	Local release workflow
Browser interactions	Execution, modified input, replay pass and tamper failure exercised	Cloud Chrome interaction
Layout inspection	Desktop and responsive views checked	No physical-mobile or Safari certification

Artifact identity

```
reference_runtime.py
3188b54c93976182704eef2fa4b96a4496f9aa81d965b2fd142ad550c62ea55c

reference-runtime.mjs
d116023bcf252025887cf693ad1a7d5bba92c3d4c26746b333040823d0079c24
```

Source commit: 39d1291bc181aba31ce348b3e9f2389657618c35. The release record identifies Python 3.12.14 and Node.js 24.19.0. Checksums detect changes relative to the manifest; the manifest itself needs an authenticated distribution source.

Documentation change history

Version	Change
Reference 0.2	Executable local retrieval, independent replay, structural schema and release checks
Chinese documentation 1.0	Architecture and network protocol review draft
English documentation 1.1	Developer-oriented structure; executable English example; independent execution, acceptance and payment states; expanded recovery guidance

Documentation 1.1 does not change the v0.2 runtime or establish network deployment. The earlier v0.1 authored receipt fixture is not an execution event generated by the v0.2 workload. Network capacity, model inference, payment and production load testing remain outside the recorded reference evidence.

28 Terminology and reading paths

Reference

Term	Meaning in this documentation
Task	One customer request and the complete set of its attempts
Attempt	One execution opportunity with its own identity and history
Workload profile	Versioned execution method and input and output constraints
Acceptance profile	Evaluation rules frozen before execution
Lease	Time-limited authority to perform a scoped execution
Fencing token	Monotonic generation used to reject stale executors
Idempotency	Repeated requests do not repeat the defined business effect
Receipt	An executor's record or claim about a delivery
Digest	A cryptographic summary of explicitly defined bytes
Attestation	Evidence about stated properties under a defined trust root
TEE	Trusted execution environment; guarantees depend on the hardware and threat model
Settlement	Payment confirmation under the selected channel's conditions
SLO	Service level objective with a metric, window and target
JCS	JSON canonicalization defined by RFC 8785

For application developers

Read the overview, run the local reference, then inspect the local task, result and error contracts. Review the network task and API chapters before designing a future integration; their availability remains Proposed specification.

For operators and reviewers

Operators should focus on admission, resource claims, leases and incident recovery. Security reviewers should inspect roles, signature binding, verification order and isolation. Technical diligence should begin with release status, artifact identity, recorded tests and the evidence required for the next release stage.

Use Task for the logical request and Attempt for an individual execution. Reserve accepted for the explicitly named context: executor acceptance or delivery acceptance. Use payment confirmed only when independent channel evidence is present.

29 Implementation and industry references

Reference | Sources reviewed 19 September 2026

The implementation links identify Ocean6 reference materials. Industry links inform the organization of this documentation and specific engineering questions. They do not indicate a partnership, integration, audit or endorsement.

[E1] Ocean6 developer reference

Published reference site and browser task lab. The public brand domain is ocean6.ai; this baseline uses the linked reference deployment.

[E2] Ocean6 Engineering Reference 0.2

Versioned source, schema, fixtures, tests and engineering notes.

[E3] Ocean6 reference validation record

Recorded environment, test outcomes and limits of the release evidence.

[R1] io.net Proof of Work

Resource verification, verification outcomes and operator troubleshooting. Used as a pattern for documenting a check and its failure modes.

[R2] Aethir The Checker

Registration, standby and in-service checking responsibilities. Used to organize admission and continuing qualification.

[R3] Nosana Job Definition

Declarative workload descriptions with explicit execution requirements.

[R4] Ritual Chain Developer Documentation

Architecture, delegated execution and asynchronous lifecycle documentation. Used to frame execution and verification boundaries.

[R5] Nosana Job Definition Validation

Strict validation before submission and actionable field-level errors.

No competitor device count, test interval, performance figure or security guarantee is adopted as an Ocean6 property. Referenced pages may change; formal design decisions should retain the reviewed version and relevant evidence.

30 Standards and documentation governance

Reference

[S1] JSON Schema Draft 2020 12

Structural schema language used by the local task schema. Runtime constraints still apply.

[S2] RFC 8785 JSON Canonicalization Scheme

Candidate network canonicalization profile. This RFC is Informational, not Standards Track; the local reference does not implement it.

[S3] RFC 9457 Problem Details for HTTP APIs

Reference format for the proposed network error envelope.

[S4] RFC 8032 Edwards Curve Digital Signature Algorithm

Ed25519 candidate signature algorithm. A complete Ocean6 profile still requires review and interoperability tests.

[S5] RFC 8174 and BCP 14 requirement language

Clarifies the uppercase requirement keywords defined by RFC 2119.

Version policy

Maintain separate versions for documentation, protocol contracts, workload profiles and executable artifacts. A documentation revision records its reason and affected requirements. A protocol change records compatibility impact and migration behavior. Editorial improvements do not imply a new runtime capability.

Publication requirements

Each developer page should state availability, applicable version and required prerequisites. A guide should include a runnable path and an expected outcome. A reference page should define fields, constraints and errors. An operational page should define evidence, failure handling and recovery.

Network specifications require engineering review before being described as stable. Promotion to a production contract requires an approved schema, deployed implementation, current operational ownership and linked acceptance evidence. If documentation and code disagree, classify and resolve the discrepancy openly; do not rewrite historical validation records.

This English edition is a technical review specification with an implemented local reference. It is suitable for engineering review and integration planning. Production readiness must be assessed against the release criteria, not the completeness of the document.