2026

SipLive SBC Manager

API Reference

2026 — Version 12.14

Contents

1 Introduction

The SipLive SBC Manager REST API provides programmatic access to all configuration, operational, and diagnostic features of the platform. Every write the GUI performs is backed by the same validated service layer the API calls, ensuring that automated workflows have identical guarantees to GUI workflows.

The API is versioned at `/api/v1/`. A SOAP compatibility facade is available at `/soap/v1` for legacy integrations requiring XML-over-HTTP.

1.1 What the API Can Do

- Provision and manage SIP trunks, DID inventory, routing policies, HMR rule sets, and fraud controls.
- Operate the staged apply workflow: stage a config version, check cluster readiness, queue apply or rollback, and monitor per-node apply history.
- Manage HA nodes, maintenance mode, and per-node config overrides.
- Run route simulation dry runs without originating calls.
- List, create, and download system backups.
- Request and retrieve packet captures, SIP ladders, call traces, and diagnostic bundles.
- Search CDR records and pull call-volume, failed-call, and trunk-usage reports.
- Review the audit log for compliance and security events.
- Manage GUI users, roles, and scoped API tokens.

1.2 Design Principles

- **Stage-then-apply**: write endpoints stage database changes and flag generated config as needing Apply. Asterisk is not reloaded until an explicit apply call is made.
- **Secrets never returned**: trunk passwords, API token bearer values (after initial creation), and password hashes are never present in API responses.
- **Path traversal prevention**: file-serving endpoints (backup downloads, PCAP downloads) verify that resolved paths remain inside managed directories before streaming.
- **Audit trail**: every write records an audit event. Responses include `audit_id` when applicable.
- **Idempotency by convention**: import endpoints (`/dids/import`, `/endpoints/provision`) match on unique identifiers and update existing records rather than creating duplicates.

2 Authentication

2.1 Bearer Token

Every API request must include an `Authorization` header:

```
Authorization: Bearer sbc_xxxxxxxxxxxxxxx
```

Tokens are created and managed in the GUI under **Security → API Tokens**, or via the `/api/v1/security/api-tokens` endpoint itself. Each token carries one or more **scopes** that restrict which endpoints it may call.

2.2 Token Scopes

Scopes follow a `action:resource` pattern. The full catalog is returned by `GET /api/v1/security/permissions`.

Scope	Access Granted
`read:trunks`	List and view SIP trunks (no secrets)
`write:trunks`	Create, update, delete trunks
`read:dids`	List DID inventory
`write:dids`	Import, update, and sync DIDs
`read:routes`	List routing policies
`write:routes`	Create, update, delete routes
`read:hmr`	List HMR sets and rules
`write:hmr`	Create, update, delete HMR sets and rules
`read:fraud`	List fraud/allow-block rules
`write:fraud`	Create, update, delete fraud rules
`read:ops`	View apply status, nodes, config versions, alerts
`write:ops`	Stage config, queue apply/rollback, manage nodes
`read:backups`	List backups and metadata
`write:backups`	Create backups
`read:diagnostics`	List captures, download artifacts, trace calls
`write:diagnostics`	Start packet captures
`read:cdr`	Search CDR records and reports
`read:audit`	Search audit log
`read:security`	List users, roles, API tokens
`write:security`	Create/update/delete users, roles, API tokens

Scope	Access Granted
`simulate:routes`	Run route simulation dry runs
`read:api-docs`	Access API coverage matrix and SOAP WSDL

2.3 IP Allowlists

Tokens may be restricted to one or more source IP addresses or CIDR networks via the `allowed_ips` field. Requests from unlisted addresses are rejected with `403` even if the token is otherwise valid.

2.4 Token Expiry

Tokens accept an optional `expires_at` ISO-8601 timestamp. Expired tokens return `401`. Tokens with no expiry are permanent until deleted.

---

3 Base URL and Versioning

`https://<sbc-hostname>/api/v1/`

Replace `<sbc-hostname>` with the fully qualified domain name or IP of your SBC node. All examples in this document use `https://sbc.example.com/api/v1/`.

The SOAP facade is at:

`https://<sbc-hostname>/soap/v1`

The OpenAPI machine-readable coverage matrix is available at:

```
GET /api/v1/api-coverage          (scope: read:api-docs)
```

---

4 Rate Limiting

The API applies per-token rate limiting. When the limit is exceeded the server returns `429 Too Many Requests`. The response includes standard `Retry-After` guidance. Batch import and bulk-update endpoints enforce per-batch size limits documented with each endpoint.

---

5 Request and Response Format

All REST endpoints accept and return `application/json`. File-download endpoints (`GET /backups/{backup}`, `GET /diagnostics/captures/{id}/{type}`, `POST /diagnostics/bundle`) return binary streams with appropriate `Content-Type` and `Content-Disposition` headers.

Every successful write response includes:

- The saved or operated-on resource (or an operation summary for bulk/apply calls).
- `audit_id` when an audit event was recorded.
- `needs_apply: true` when the change requires an Asterisk reload to take effect.

---

6 Error Codes

HTTP Status	Meaning
400	Malformed request (missing required field, invalid JSON)
401	Missing, expired, or invalid bearer token
403	Token lacks required scope, or source IP not in allowlist
404	Resource not found
409	Duplicate resource, idempotency conflict, or unsafe apply conflict
422	Validation error (field-level details in `errors` key)
429	Rate limit exceeded
500	Unexpected server error

Validation errors follow Laravel's standard JSON envelope:

```
{
  "message": "The given data was invalid.",
  "errors": {
    "name": ["The name field is required."]
```

```
  }
}
```

---

7 API Coverage

7.1 GET /api-coverage

Scope: `read:api-docs`

Returns the machine-readable GUI-to-REST/SOAP parity matrix, current coverage status, and shared safety controls. Use this endpoint to confirm which features are available on a given node version before scripting against them.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/api-coverage
```

---

8 SIP Trunks

SIP trunk writes stage changes and set `needs_apply`. Apply the configuration via the Operations endpoints before Asterisk reflects the change.

8.1 GET /trunks

Scope: `read:trunks`

Returns the full trunk inventory. Trunk secrets (passwords, auth keys) are never included in any response.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/trunks
```

8.2 GET /trunks/{id}

Scope: `read:trunks`

Returns a single trunk by numeric ID. Secrets are excluded.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/trunks/5
```

8.3 POST /trunks

Scope: `write:trunks`

Creates a SIP trunk. Secrets are accepted but never returned.

```
curl -X POST https://sbc.example.com/api/v1/trunks \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Bandwidth-Primary",
    "host": "sip.bandwidth.com",
    "port": 5060,
    "transport": "udp",
    "codecs": ["ulaw","alaw"],
    "max_channels": 50
  }'
```

8.4 PUT /trunks/{id}

Scope: `write:trunks`

Updates an existing trunk. Only supplied fields are changed. Secrets are accepted but never returned. Setting `enabled: false` disables the trunk without deleting it.

```
curl -X PUT https://sbc.example.com/api/v1/trunks/5 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"max_channels": 100}'
```

8.5 DELETE /trunks/{id}

Scope: `write:trunks`

Deletes a trunk. Trunks referenced by active routes or DID assignments are rejected with `409`. Remove the references first, or re-route traffic before deleting.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/trunks/5
```

8.6 GET /trunks/export

Scope: `read:trunks`

Streams the full trunk inventory as CSV. Secrets are excluded from the export.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/trunks/export -o trunks.csv
```

8.7 GET /trunks/{id}/status

Scope: `read:trunks`

Returns live registration and channel status for a single trunk, sourced from Asterisk PJSIP state.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/trunks/5/status
```

8.8 GET /carrier-templates

Scope: `read:trunks`

Returns the full carrier template library with readiness scores, defaults, and firewall/media/security guidance for supported carriers: Microsoft Teams Direct Routing, Bandwidth, SIPLive, Telnyx, Twilio Elastic SIP Trunking, Skyetel, Flowroute, generic PBX, and generic TLS.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/carrier-templates
```

8.9 POST /trunks/{id}/maintenance

Scope: `write:trunks`

Toggles maintenance mode on a trunk. Maintenance trunks are excluded from normal routing and from apply queues targeting healthy trunks.

```
curl -X POST https://sbc.example.com/api/v1/trunks/5/maintenance \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"maintenance": true}'
```

8.10 POST /trunks/{id}/sip-trace

Scope: `write:trunks`

Toggles verbose SIP tracing for a trunk in Asterisk PJSIP. Tracing is ephemeral and survives until the next Asterisk reload.

```
curl -X POST https://sbc.example.com/api/v1/trunks/5/sip-trace \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"enabled": true}'
```

---

9 Endpoint Provisioning (NAP Wizard)

The endpoint provisioning endpoint replicates the full New Endpoint wizard in a single transactional call. It creates the trunk, optional DID assignments, optional HMR set, an initial route, and optional per-node overrides in one atomic workflow.

9.1 POST /endpoints/provision

Scope: `write:trunks`

Accepts the same fields as the GUI wizard. Supports `save_only`, `apply_one` (specify `node_id`), `apply_all`, and `schedule` (with `not_before` ISO timestamp) as the `action` field. Preflight checks for DID conflicts, route collisions, capacity warnings, and node eligibility run before saving.

```
curl -X POST https://sbc.example.com/api/v1/endpoints/provision \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "trunk": {
      "name": "Skyetel-Secondary",
      "host": "sip.skyetel.com",
      "transport": "tls",
      "codecs": ["ulaw"],
      "max_channels": 25
    },
    "dids": ["+19195550100", "+19195550101"],
    "route": {
      "name": "Skyetel-Inbound",
      "match_field": "source_ip",
      "match_value": "205.0.113.0/24"
```

```
    },
    "action": "apply_all"
  }'
```

---

10 DID Management

DID writes stage changes. Apply before Asterisk dialplan reflects the new routing.

10.1 GET /dids

Scope: `read:dids`

Returns paginated DID inventory with assigned trunk and failover trunk. Supports `trunk_id`, `status`, `number`, `page`, and `per_page` query parameters.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/dids?trunk_id=3&page=1"
```

10.2 POST /dids/import

Scope: `write:dids`

Creates or updates DID routing records in batch. Matches on E.164 number and updates existing records rather than rejecting duplicates. Returns counts of created, updated, and failed rows.

Accepted fields per DID: `number` (E.164), `trunk_id`, `failover_trunk_id`, `strip_digits`, `prepend_digits`, `description`.

```
curl -X POST https://sbc.example.com/api/v1/dids/import \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "dids": [
      {"number": "+19195550100", "trunk_id": 3, "failover_trunk_id": 4},
      {"number": "+19195550101", "trunk_id": 3}
    ]
  }'
```

10.3 POST /dids/bulk-update

Scope: `write:dids`

Updates routing fields on existing DID records in bulk by E.164 number. Numbers not found in inventory are reported as missing rather than created.

```
curl -X POST https://sbc.example.com/api/v1/dids/bulk-update \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "dids": [
      {"number": "+19195550100", "trunk_id": 5},
      {"number": "+19195550101", "trunk_id": 5}
    ]
  }'
```

10.4 GET /dids/export

Scope: `read:dids`

Streams DID inventory as CSV with the same filters as `GET /dids`.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/dids/export?trunk_id=3" -o dids.csv
```

10.5 DELETE /dids/{id}

Scope: `write:dids`

Deletes a single DID record. The corresponding dialplan entry is removed on the next Apply.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/dids/42
```

10.6 POST /integrations/bandwidth/sync

Scope: `write:dids`

Triggers an immediate Bandwidth DID inventory sync using the `.env` credentials on the local node (`BANDWIDTH_USERNAME`, `BANDWIDTH_PASSWORD`, `BANDWIDTH_ACCOUNT_ID`). Returns attempted, synced, and failed counts.

```
curl -X POST -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/integrations/bandwidth/sync
```

10.7 GET /integrations/siplive/dids

Scope: `read:dids`

Returns SIPLive DID sync status and the last sync result using the local node's `.env` credentials (`SIPLIVE_SYNC_ENABLED`, `SIPLIVE_API_BASE_URL`, `SIPLIVE_API_TOKEN`).

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/integrations/siplive/dids
```

---

11 Routing Policies

Route writes stage changes. Apply before Asterisk dialplan reflects the updated routing.

11.1 GET /routes

Scope: `read:routes`

Returns all routing policies with match conditions, priority, and trunk assignments.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/routes
```

11.2 POST /routes

Scope: `write:routes`

Creates a routing policy. Supports simple single match (`match_field`, `match_operator`, `match_value`) or complex multi-condition matching via the `match_conditions` array. Time windows, failover trunk, and HMR assignments are optional.

Route types: `static`, `source`, `destination`, `emergency`, `blocked`, `failover`.

```
curl -X POST https://sbc.example.com/api/v1/routes \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Inbound-Bandwidth",
    "type": "source",
```

```
    "match_field": "source_ip",
    "match_operator": "cidr",
    "match_value": "216.82.224.0/19",
    "trunk_id": 3,
    "priority": 10
  }'
```

11.3 PUT /routes/{id}

Scope: `write:routes`

Updates a routing policy. Emergency route changes that would demote or remove emergency coverage require the `confirm_emergency_change: true` field.

```
curl -X PUT https://sbc.example.com/api/v1/routes/12 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"priority": 20, "failover_trunk_id": 4}'
```

11.4 DELETE /routes/{id}

Scope: `write:routes`

Deletes a routing policy. Deleting an emergency route requires `confirm_emergency_change: true`.

```
curl -X DELETE https://sbc.example.com/api/v1/routes/12 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"confirm_emergency_change": true}'
```

11.5 GET /routes/export

Scope: `read:routes`

Streams routing policy inventory as CSV.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/routes/export -o routes.csv
```

---

12 Route Simulation

Route simulation evaluates call-routing decisions against the live policy database without originating any traffic.

12.1 POST /route-simulate

Scope: `simulate:routes`

Runs a single dry-run route simulation. Returns the matched route, trunk, applied transforms, and any fraud-rule hits.

```
curl -X POST https://sbc.example.com/api/v1/route-simulate \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "source_ip": "203.0.113.10",
    "caller_id": "+19195550100",
    "destination": "+19195550101"
  }'
```

12.2 POST /route-simulate/batch

Scope: `simulate:routes`

Runs up to 200 dry-run simulations in one call. Each item in `cases` follows the same fields as the single simulate endpoint. Results are returned in the same order as the input array.

```
curl -X POST https://sbc.example.com/api/v1/route-simulate/batch \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "cases": [
      {"source_ip": "203.0.113.10", "caller_id": "+19195550100", "destination": "+1919555010
      {"source_ip": "10.0.0.5", "caller_id": "+18005551212", "destination": "911"}
    ]
  }'
```

13 Header Manipulation Rules (HMR)

HMR sets are reusable rule collections that can be assigned to trunks or routes. HMR writes stage changes and require Apply before Asterisk reflects them.

13.1 GET /hmr

Scope: `read:hmr`

Returns all HMR sets with their rules and trunk/route assignments.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/hmr
```

13.2 POST /hmr

Scope: `write:hmr`

Creates an HMR set with an optional inline `rules` array. Assignments to trunks (`assigned_trunk_ids`) and routes (`assigned_route_ids`) are applied transactionally.

```
curl -X POST https://sbc.example.com/api/v1/hmr \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Teams-Normalize",
    "description": "Normalize PAI and strip Privacy for Teams Direct Routing",
    "rules": [
      {"action": "set",    "header": "P-Asserted-Identity", "value": "<sip:${CALLERID(num)}@$
      {"action": "remove", "header": "Privacy",
    ],
    "assigned_trunk_ids": [7]
  }'
```

13.3 PUT /hmr/{id}

Scope: `write:hmr`

Updates an HMR set. Supplying `rules` replaces the full rule list transactionally. Supplying `assigned_trunk_ids` or `assigned_route_ids` syncs the assignment list.

```
curl -X PUT https://sbc.example.com/api/v1/hmr/3 \
  -H "Authorization: Bearer sbc_xxx" \
```

```
  -H "Content-Type: application/json" \
  -d '{"assigned_route_ids": [10, 11]}'
```

13.4 DELETE /hmr/{id}

Scope: `write:hmr`

Deletes an HMR set and removes all trunk and route assignments.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/hmr/3
```

13.5 GET /hmr/export

Scope: `read:hmr`

Streams HMR set inventory as CSV.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/hmr/export -o hmr.csv
```

13.6 POST /hmr/{id}/preview

Scope: `read:hmr`

Returns a rendered preview of the HMR set rules as they would appear in generated Asterisk dialplan. Does not apply or stage anything.

```
curl -X POST -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/hmr/3/preview
```

---

14 Fraud Controls

Fraud rules are destination allow/block policies. Non-regex patterns are normalized to digits. Fraud rule writes stage changes and require Apply.

14.1 GET /fraud-rules

Scope: `read:fraud`

Returns all destination fraud rules with type, pattern, and enabled state.

```
curl -H "Authorization: Bearer sbc_xxx" https://sbc.example.com/api/v1/fraud-rules
```

14.2 POST /fraud-rules

Scope: `write:fraud`

Creates a fraud rule. `type` is `allow` or `block`. `pattern` accepts E.164 prefixes, digit strings, or regular expressions when `is_regex: true`.

```
curl -X POST https://sbc.example.com/api/v1/fraud-rules \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"type": "block", "pattern": "900", "description": "Block 900-prefix calls"}'
```

14.3 PUT /fraud-rules/{id}

Scope: `write:fraud`

Updates a fraud rule.

```
curl -X PUT https://sbc.example.com/api/v1/fraud-rules/8 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"enabled": false}'
```

14.4 DELETE /fraud-rules/{id}

Scope: `write:fraud`

Deletes a fraud rule.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/fraud-rules/8
```

14.5 GET /fraud-rules/export

Scope: `read:fraud`

Streams fraud rule inventory as CSV.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/fraud-rules/export -o fraud-rules.csv
```

15 Operations and Apply Workflow

The operations API controls the staged-config apply lifecycle. The general workflow is:

1. **Stage**: `POST /ops/config-versions` to snapshot current database state.
2. **Preview**: `GET /ops/apply-readiness` to confirm node eligibility.
3. **Queue**: `POST /ops/apply-queue` to queue the config version for one node or all healthy nodes.
4. **Monitor**: `GET /ops/config-versions/{id}` or the HA node health page to track per-node apply results.

15.1 GET /ops/apply-status

Scope: `read:ops`

Returns the current generated-config state: whether apply is needed, the active config version, per-node apply status, and cluster-level apply health.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/apply-status
```

15.2 GET /ops/apply-readiness

Scope: `read:ops`

Returns per-node and cluster-level apply eligibility. Reports reasons a node would be skipped (maintenance mode, unhealthy, no config version staged). Use this before queuing an apply.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/apply-readiness
```

15.3 GET /ops/config-versions

Scope: `read:ops`

Lists config version metadata and per-node apply counts without exposing full generated file content.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/config-versions
```

15.4 POST /ops/config-versions

Scope: `write:ops`

Creates or reuses a config version snapshot of the current database state. Does not apply, reload, or write any Asterisk files. Returns the version ID for use with `POST /ops/apply-queue`.

```
curl -X POST https://sbc.example.com/api/v1/ops/config-versions \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"description": "Pre-maintenance checkpoint"}'
```

15.5 POST /ops/apply-queue

Scope: `write:ops`

Queues an existing config version (or the current database state) for apply on one eligible node or all eligible nodes. The optional `not_before` field (ISO-8601) schedules the apply for a future maintenance window.

```
curl -X POST https://sbc.example.com/api/v1/ops/apply-queue \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"config_version_id": 14, "target": "all"}'
```

Scheduled apply example:

```
curl -X POST https://sbc.example.com/api/v1/ops/apply-queue \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "config_version_id": 14,
    "target": "all",
    "not_before": "2026-06-22T03:00:00Z"
  }'
```

15.6 POST /ops/rollback-queue

Scope: `write:ops`

Queues a prior config version as a rollback. Rollback operations are recorded separately in the audit log with a `rollback` event type.

```
curl -X POST https://sbc.example.com/api/v1/ops/rollback-queue \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"config_version_id": 11, "target": "all"}'
```

15.7 POST /ops/config-versions/{id}/approve

Scope: `write:ops`

Approves a staged config version for queueing through the change-approval workflow. Approved versions may be queued for apply.

```
curl -X POST -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/config-versions/14/approve
```

15.8 POST /ops/config-versions/{id}/reject

Scope: `write:ops`

Rejects a staged config version. Requires `approval_note`. Rejected versions cannot be queued for apply.

```
curl -X POST https://sbc.example.com/api/v1/ops/config-versions/14/reject \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"approval_note": "Trunk host typo - needs correction before apply."}'
```

15.9 DELETE /ops/apply-queue/{id}

Scope: `write:ops`

Cancels a future scheduled apply by apply-history record ID. Only future (not-yet-due) applies can be cancelled.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/apply-queue/55
```

15.10 GET /ops/scheduled-applies

Scope: `read:ops`

Returns future, due, or all queued scheduled apply jobs with a queue summary.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/ops/scheduled-applies?status=future"
```

---

16 HA Nodes

16.1 GET /ops/nodes

Scope: `read:ops`

Returns all registered HA nodes with enabled state, maintenance state, last heartbeat, apply health, and current config hash.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/nodes
```

16.2 PUT /ops/nodes/{id}

Scope: `write:ops`

Toggles maintenance mode or enabled state for an HA node without reloading Asterisk. Setting `maintenance: true` excludes the node from future apply-queue runs targeting healthy nodes.

```
curl -X PUT https://sbc.example.com/api/v1/ops/nodes/2 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"maintenance": true}'
```

16.3 POST /ops/nodes/{id}/evacuate

Scope: `write:ops`

Initiates a safe node-evacuation workflow. The node is placed in maintenance mode, active calls are allowed to complete (up to a configured drain timeout), and the node is excluded from future applies and routing.

```
curl -X POST -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/nodes/2/evacuate
```

16.4 GET /ops/cluster-health

Scope: `read:ops`

Returns active-active cluster health: per-node reachability, apply sync state, config-hash agreement, and overall cluster health score.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/cluster-health
```

16.5 GET /ops/config-drift

Scope: `read:ops`

Reports per-node config drift: expected hash vs. on-disk hash, last apply timestamp, and drift severity. Use this to detect nodes that have had their Asterisk config modified outside the GUI.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/config-drift
```

16.6 GET /ops/health

Scope: `read:ops`

Returns runtime health checks: database connectivity, Asterisk command access, PJSIP transport state, firewall SIP/RTP rules, generated config paths, CDR readability, and Laravel storage health.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/health
```

16.7 GET /ops/alerts

Scope: `read:ops`

Returns operations alert summary and alert-delivery readiness (email/webhook configuration) without exposing delivery secrets.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/alerts
```

---

17 Per-Node Config Overrides

Node overrides allow specific trunk or route configuration values to differ on individual SBC nodes without creating separate trunk or route records.

17.1 GET /ops/node-overrides

Scope: `read:ops`

Returns all per-node overrides with node, scope, target, and the overridden field set.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/node-overrides
```

17.2 POST /ops/node-overrides

Scope: `write:ops`

Creates a per-node override. `scope` is `trunk` or `route`. `target` is the trunk or route name. `overrides` is the map of fields to override. Changes require Apply before Asterisk config reflects them.

```
curl -X POST https://sbc.example.com/api/v1/ops/node-overrides \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "node": "sbc01",
    "scope": "trunk",
    "target": "TeamsDirect",
    "overrides": {"max_channels": 40, "passthrough_headers": ["X-MS-Teams"]}
  }'
```

17.3 GET /ops/node-overrides/effective/{node}

Scope: `read:ops`

Returns the effective merged configuration for all trunks and routes on a specific node after all per-node overrides are applied.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/node-overrides/effective/sbc01
```

17.4 POST /ops/node-overrides/import

Scope: `write:ops`

Imports an array of per-node overrides in batch. Existing overrides for the same node/scope/target are replaced.

```
curl -X POST https://sbc.example.com/api/v1/ops/node-overrides/import \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "overrides": [
      {"node": "sbc01", "scope": "trunk", "target": "TeamsDirect",
       "overrides": {"max_channels": 40, "passthrough_headers": ["X-MS-Teams"]}}
    ]
  }'
```

17.5 DELETE /ops/node-overrides/{id}

Scope: `write:ops`

Deletes a per-node override. The next Apply will restore the base trunk or route configuration on the affected node.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/ops/node-overrides/7
```

---

18 Backups

18.1 GET /backups

Scope: `read:backups`

Returns backup freshness metadata and safe backup inventory: name, size, and creation timestamp. Local filesystem paths are never exposed.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/backups
```

18.2 POST /backups

Scope: `write:backups`

Creates a managed recovery backup ZIP containing a MariaDB dump and generated Asterisk config. Returns safe backup metadata. Local filesystem paths are never returned.

```
curl -X POST https://sbc.example.com/api/v1/backups \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"reason": "pre-maintenance"}'
```

18.3 GET /backups/{backup}

Scope: `read:backups`

Downloads a managed backup ZIP by filename. The filename must be one returned by `GET /backups` and must match the `sbc-backup-YYYYMMDD-HHMMSS.zip` pattern. Local filesystem paths are never accepted. Every download is audited.

```
curl -L -H "Authorization: Bearer sbc_xxx" \
  -o sbc-backup.zip \
  https://sbc.example.com/api/v1/backups/sbc-backup-20260521-001500.zip
```

---

19 Diagnostics

19.1 GET /diagnostics/captures

Scope: `read:diagnostics`

Returns paginated packet-capture job metadata. Supported query parameters: `status`, `interface`, `date_from`, `date_to`, `page`, `per_page`. The response exposes artifact availability flags but never exposes local PCAP, summary, or ladder file paths.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/diagnostics/captures?status=finished"
```

19.2 POST /diagnostics/captures

Scope: `write:diagnostics`

Requests a short targeted packet capture. Required fields: `name`, `interface`, `filter` (BPF), `duration_seconds` (5–120). The server validates the interface is currently visible and compiles the BPF filter before starting a privileged `tshark` process. The response returns job status and artifact availability flags, never local paths.

```
curl -X POST https://sbc.example.com/api/v1/diagnostics/captures \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "PBX options probe",
    "interface": "any",
    "filter": "host 203.0.113.10 and udp port 5060",
    "duration_seconds": 10
  }'
```

19.3 GET /diagnostics/captures/{id}/{type}

Scope: `read:diagnostics`

Downloads a finished capture artifact. `type` must be `pcap`, `summary`, or `ladder`. The server verifies the resolved artifact path remains inside the managed PCAP directory before streaming.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/diagnostics/captures/12/ladder -o ladder.txt
```

19.4 POST /diagnostics/bundle

Scope: `read:diagnostics`

Generates and streams a sanitized diagnostic ZIP bundle. Includes: selected runtime status, health checks, RTP quality summary, capture inventory, generated config includes, tailed logs, and a manifest with SHA-256 hashes. PCAP payloads are not bundled automatically. Common passwords, bearer tokens, API keys, private keys, and trunk secrets are redacted.

```
curl -X POST -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/diagnostics/bundle -o sbc-diagnostic.zip
```

19.5 GET /diagnostics/trace

Scope: `read:diagnostics`

Searches stored CDR records, Asterisk CEL CSV files, capture metadata/summaries/ladders, and recent logs by Call-ID, unique ID, linked ID, caller ID, destination, or SBC test token. Returns a unified `timeline` array, evidence matrix, confidence label, missing-evidence checklist, and next-step recommendations. When Asterisk log lines contain SIP methods, statuses, or

Call-IDs, the response includes `log_ladder` events for call tracing without a PCAP. Local filesystem paths are never returned.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/diagnostics/trace?needle=15551235555"
```

19.6 GET /diagnostics/rtp-quality

Scope: `read:diagnostics`

Returns best-effort RTP quality analysis from recent logs and capture summaries. Includes jitter, packet loss, RTT, RTP-timeout evidence, a MOS-style estimate, severity, issue count, redacted evidence samples, and operator recommendations. For packet-level proof, use a targeted PCAP capture.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/diagnostics/rtp-quality
```

19.7 POST /diagnostics/test-plan

Scope: `read:diagnostics`

Builds a safe test-call worksheet from route simulation without originating calls. Returns expected routing, risk level, go/no-go decision, expected SIP outcome, live trunk status, capacity assertions, suggested PCAP settings, optional test headers, pre-call checks, Asterisk verification commands, stop conditions, validation matrix, post-call evidence criteria, trace search values, and SIPp starter guidance. Optional `test_type` values: `inbound`, `outbound`, `failover`, `emergency`.

```
curl -X POST https://sbc.example.com/api/v1/diagnostics/test-plan \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "source_ip": "203.0.113.10",
    "caller_id": "+19195550100",
    "destination": "+19195550101",
    "test_type": "inbound"
  }'
```

20 CDR and Reporting

20.1 GET /cdr

Scope: `read:cdr`

Returns paginated CDR records. Supported query parameters: `caller_id`, `destination`, `trunk_id`, `disposition`, `date_from`, `date_to`, `page`, `per_page` (max 500 per page).

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/cdr?destination=1555123&date_from=2026-05-01"
```

20.2 GET /cdr/export

Scope: `read:cdr`

Streams up to 10,000 filtered CDR rows as CSV. Supports the same filters as `GET /cdr`.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/cdr/export?date_from=2026-05-01" -o cdr.csv
```

20.3 GET /reports/call-volume

Scope: `read:cdr`

Returns call totals grouped by `day` or `hour` (set `bucket` query parameter). Supports the same CDR filters.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/reports/call-volume?bucket=day&date_from=2026-05-01"
```

20.4 GET /reports/failed-calls

Scope: `read:cdr`

Returns failed call totals grouped by SIP disposition code. Supports the same CDR filters.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/reports/failed-calls?date_from=2026-05-01"
```

20.5 GET /reports/trunk-usage

Scope: `read:cdr`

Returns call totals, answered calls, failed calls, and total call duration grouped by trunk.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/reports/trunk-usage?date_from=2026-05-01"
```

---

21 Audit Log

21.1 GET /audit-logs

Scope: `read:audit`

Returns paginated audit events for security review and compliance export. Supported query parameters: `action`, `user_id`, `target_type`, `date_from`, `date_to`, `page`, `per_page` (max 500 per page). Credential-shaped fields in `before` and `after` payloads are redacted in responses.

```
curl -H "Authorization: Bearer sbc_xxx" \
  "https://sbc.example.com/api/v1/audit-logs?action=trunk&date_from=2026-05-01"
```

---

22 Security Management

Security write endpoints create audit log entries. The active token cannot delete itself or disable its own owner.

22.1 GET /security/users

Scope: `read:security`

Returns GUI user inventory with role names, enabled state, and last-login time. Password hashes, remember tokens, and session data are never returned.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/users
```

22.2 POST /security/users

Scope: `write:security`

Creates a GUI user with a strong password, role assignment, and enabled state. The password is accepted but never returned.

```
curl -X POST https://sbc.example.com/api/v1/security/users \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "NOC Technician",
    "email": "noc@example.com",
    "role_id": 2,
    "password": "Use-A-Long-Strong-Secret-123!",
    "enabled": true
  }'
```

22.3 PUT /security/users/{id}

Scope: `write:security`

Updates a GUI user. Omit `password` to preserve the current password. The API prevents disabling the user that owns the active token.

```
curl -X PUT https://sbc.example.com/api/v1/security/users/4 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"name": "NOC Lead", "role_id": 3}'
```

22.4 DELETE /security/users/{id}

Scope: `write:security`

Deletes a GUI user. The API prevents deleting the user that owns the active token.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/users/4
```

22.5 GET /security/roles

Scope: `read:security`

Returns role names, permission arrays, and assigned user counts.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/roles
```

22.6 POST /security/roles

Scope: `write:security`

Creates a GUI role with validated permission keys. Use `GET /security/permissions` as the canonical permission catalog before submitting.

```
curl -X POST https://sbc.example.com/api/v1/security/roles \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"name": "noc-readonly", "permissions": ["dashboard.view", "trunks.view", "cdr.view"]}
```

22.7 PUT /security/roles/{id}

Scope: `write:security`

Updates a GUI role with validated permission keys. Supplying `permissions` replaces the full permission list.

```
curl -X PUT https://sbc.example.com/api/v1/security/roles/3 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{"permissions": ["dashboard.view", "trunks.view", "routes.view", "cdr.view"]}'
```

22.8 DELETE /security/roles/{id}

Scope: `write:security`

Deletes a role. Roles with assigned users are rejected with `409`; reassign users first.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/roles/3
```

22.9 GET /security/permissions

Scope: `read:security`

Returns valid GUI permission keys grouped by functional area and a flattened list. Also returns the canonical API scope catalog and recommended token presets for least-privilege integration setup.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/permissions
```

22.10 GET /security/api-tokens

Scope: `read:security`

Returns API token metadata for access review. Bearer secrets and token hashes are never returned.

```
curl -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/api-tokens
```

22.11 POST /security/api-tokens

Scope: `write:security`

Creates a scoped API token. The `token` bearer value is returned **once** in this response; store it immediately. Later calls return only metadata. `allowed_ips` accepts an array or comma-separated CIDR/IP strings.

```
curl -X POST https://sbc.example.com/api/v1/security/api-tokens \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Inventory sync",
    "scopes": ["read:trunks", "read:dids"],
    "allowed_ips": ["203.0.113.10", "10.10.10.0/24"],
    "expires_at": "2026-12-31T23:59:59Z"
  }'
```

Response (excerpt):

```
{
  "token": "sbc_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "id": 15,
  "name": "Inventory sync",
  "scopes": ["read:trunks", "read:dids"],
  "expires_at": "2026-12-31T23:59:59Z"
}
```

22.12 DELETE /security/api-tokens/{id}

Scope: `write:security`

Deletes an API token. The active bearer token cannot delete itself.

```
curl -X DELETE -H "Authorization: Bearer sbc_xxx" \
  https://sbc.example.com/api/v1/security/api-tokens/15
```

---

23 SOAP Compatibility

The SOAP facade provides feature parity with the REST API for legacy integrations that require XML-over-HTTP. It runs over the same validated service layer as REST and does not bypass token authentication or business-logic guards.

23.1 WSDL and Operation Catalog

Retrieve the full WSDL (requires `read:api-docs`):

```
GET /soap/v1/wsdl
```

Retrieve a machine-readable operation catalog (name, required scope, category, write/read classification):

```
GET /soap/v1/operations
```

23.2 Sending SOAP Requests

All SOAP requests go to:

```
POST /soap/v1
```

Include the same `Authorization: Bearer sbc_xxx` header as REST. SOAP bodies are size-limited; XML External Entity (XXE) and DOCTYPE payloads are rejected; local paths and secrets are not returned.

Example — route simulation:

```
curl -X POST https://sbc.example.com/soap/v1 \
  -H "Authorization: Bearer sbc_xxx" \
  -H "Content-Type: text/xml" \
  -d '<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SimulateRoute>
```

```
      <source_ip>203.0.113.10</source_ip>
      <caller_id>19195550100</caller_id>
      <destination>19195550101</destination>
    </SimulateRoute>
  </soap:Body>
</soap:Envelope>'
```

23.3 SOAP Operations Reference

The table below lists all SOAP operations, the required scope, and the equivalent REST endpoint.

SOAP Operation	Scope	REST Equivalent
GetApiCoverage	read:api-docs	GET /api-coverage
GetTrunks	read:trunks	GET /trunks
CreateTrunk	write:trunks	POST /trunks
UpdateTrunk	write:trunks	PUT /trunks/{id}
DeleteTrunk	write:trunks	DELETE /trunks/{id}
GetDids	read:dids	GET /dids
ImportDids	write:dids	POST /dids/import
BulkUpdateDids	write:dids	POST /dids/bulk-update
SyncSipLiveDids	write:dids	POST /integrations/siplive/dids/sync
GetFraudRules	read:fraud	GET /fraud-rules
CreateFraudRule	write:fraud	POST /fraud-rules
UpdateFraudRule	write:fraud	PUT /fraud-rules/{id}
DeleteFraudRule	write:fraud	DELETE /fraud-rules/{id}
CreateRoute	write:routes	POST /routes
UpdateRoute	write:routes	PUT /routes/{id}
DeleteRoute	write:routes	DELETE /routes/{id}
CreateHmrSet	write:hmr	POST /hmr
UpdateHmrSet	write:hmr	PUT /hmr/{id}
DeleteHmrSet	write:hmr	DELETE /hmr/{id}
SimulateRoute	simulate:routes	POST /route-simulate
GetApplyStatus	read:ops	GET /ops/apply-status
GetApplyReadiness	read:ops	GET /ops/apply-readiness

SOAP Operation	Scope	REST Equivalent
GetApplyPreview	read:ops	GET /ops/config-versions/{id}
GetScheduledApplies	read:ops	GET /ops/scheduled-applies
StageConfigVersion	write:ops	POST /ops/config-versions
QueueApply	write:ops	POST /ops/apply-queue
QueueRollback	write:ops	POST /ops/rollback-queue
CancelScheduledApply	write:ops	DELETE /ops/apply-queue/{id}
ApproveConfigVersion	write:ops	POST /ops/config-versions/{id}/approve
RejectConfigVersion	write:ops	POST /ops/config-versions/{id}/reject
GetNodes	read:ops	GET /ops/nodes
GetClusterHealth	read:ops	GET /ops/cluster-health
GetConfigDrift	read:ops	GET /ops/config-drift
GetNodeOverrides	read:ops	GET /ops/node-overrides
GetEffectiveNodeOverrides	read:ops	GET /ops/node-overrides/effective/{node}
UpdateNode	write:ops	PUT /ops/nodes/{id}
EvacuateNode	write:ops	POST /ops/nodes/{id}/evacuate
SaveNodeOverride	write:ops	POST /ops/node-overrides
DeleteNodeOverride	write:ops	DELETE /ops/node-overrides/{id}
GetHealth	read:ops	GET /ops/health
GetOpsAlerts	read:ops	GET /ops/alerts
GetDiagnosticCaptures	read:diagnostics	GET /diagnostics/captures
TraceCall	read:diagnostics	GET /diagnostics/trace
PlanTestCall	read:diagnostics	POST /diagnostics/test-plan
SearchCdr	read:cdr	GET /cdr

SOAP Operation	Scope	REST Equivalent
GetCallVolumeReport	read:cdr	GET /reports/call-volume
GetFailedCallsReport	read:cdr	GET /reports/failed-calls
GetTrunkUsageReport	read:cdr	GET /reports/trunk-usage
GetAuditLogs	read:audit	GET /audit-logs
GetUsers	read:security	GET /security/users
GetRoles	read:security	GET /security/roles
GetPermissions	read:security	GET /security/permissions
GetApiTokens	read:security	GET /security/api-tokens
CreateUser	write:security	POST /security/users
UpdateUser	write:security	PUT /security/users/{id}
DeleteUser	write:security	DELETE /security/users/{id}
CreateRole	write:security	POST /security/roles
UpdateRole	write:security	PUT /security/roles/{id}
DeleteRole	write:security	DELETE /security/roles/{id}
CreateApiToken	write:security	POST /security/api-tokens
DeleteApiToken	write:security	DELETE /security/api-tokens/{id}
GetBackups	read:backups	GET /backups
CreateBackup	write:backups	POST /backups

Note: SOAP does not stream binary files. Use the REST `GET /backups/{backup}` and `GET /diagnostics/captures/{id}/{type}` endpoints for file downloads.

23.4 SOAP Field Conventions

SOAP write bodies use the same field names as their REST counterparts. For compatibility with simple SOAP clients:

- List fields (`allowed_ips`, `codecs`, `passthrough_headers`, `hmr_rule_set_ids`, `assigned_trunk_ids`, `assigned_route_ids`) may be sent as comma or whitespace-

separated text.

- Complex route `match_conditions` and `time_windows` may be supplied as JSON strings.
- Simple route clients may use `match_field`, `match_operator`, and `match_value` instead of the `match_conditions` array.
- HMR rule lists may be supplied as a JSON string in `rules_json`.
- `UpdateTrunk`, `DeleteTrunk`, `UpdateRoute`, `DeleteRoute`, `UpdateHmrSet`, `DeleteHmrSet`, `UpdateFraudRule`, and `DeleteFraudRule` accept either `id` or `name` to identify the target record.

---

24 Operational Guidelines

24.1 Write-then-Apply Pattern

Every write endpoint stages changes in the database and sets `needs_apply: true` on the response. Asterisk is not reloaded until an explicit apply call is made. This separates the risk of a bad configuration change from the act of applying it, allowing review and rollback before reloading.

Recommended flow for automated provisioning:

```
POST /trunks           → stages trunk
POST /dids/import      → stages DIDs
POST /routes           → stages route
POST /ops/config-versions → snapshot
GET  /ops/apply-readiness → verify nodes are eligible
POST /ops/apply-queue   → apply to all healthy nodes
```

24.2 Partial Apply Results

When a queued apply targets multiple nodes, the result is per-node. One node can succeed while another fails. Check `GET /ops/apply-status` or `GET /ops/cluster-health` after an apply to confirm all nodes reflect the new config hash.

24.3 Secrets Policy

- Trunk passwords and auth credentials are accepted on write but never returned.
- API token bearer values are returned once on `POST /security/api-tokens`. Store immediately.

- Diagnostic bundles, audit logs, and PCAP summaries are sanitized: bearer tokens, private keys, trunk passwords, and common credential patterns are redacted before any API or download response.

24.4 Safe Defaults

- Apply calls target healthy, enabled, non-maintenance nodes unless a specific `node_id` is provided.
- Emergency route delete and downgrade operations require `confirm_emergency_change: true`.
- Bulk operations (DID import, batch simulation) enforce per-request size limits.
- All file-serving endpoints validate that resolved paths remain inside managed directories before streaming any content.

---

25 Appendix: Scope Quick Reference

Scope	Read / Write	Primary Endpoints
`read:trunks`	Read	`GET /trunks`, `GET /trunks/{id}`, `GET /trunks/export`, `GET /trunks/{id}/status`, `GET /carrier-templates`
`write:trunks`	Write	`POST /trunks`, `PUT /trunks/{id}`, `DELETE /trunks/{id}`, `POST /trunks/{id}/maintenance`, `POST /trunks/{id}/sip-trace`, `POST /endpoints/provision`
`read:dids`	Read	`GET /dids`, `GET /dids/export`, `GET /integrations/siplive/dids`

Scope	Read / Write	Primary Endpoints
write:dids	Write	POST /dids/import, POST /dids/bulk-update, DELETE /dids/{id}, POST /integrations/bandwidth/sync
read:routes	Read	GET /routes, GET /routes/export
write:routes	Write	POST /routes, PUT /routes/{id}, DELETE /routes/{id}
read:hmr	Read	GET /hmr, GET /hmr/export, POST /hmr/{id}/preview
write:hmr	Write	POST /hmr, PUT /hmr/{id}, DELETE /hmr/{id}
read:fraud	Read	GET /fraud-rules, GET /fraud-rules/export
write:fraud	Write	POST /fraud-rules, PUT /fraud-rules/{id}, DELETE /fraud-rules/{id}
read:ops	Read	GET /ops/* (all read endpoints), GET /ops/health, GET /ops/alerts
write:ops	Write	POST /ops/config-versions, POST /ops/apply-queue, POST /ops/rollback-queue, DELETE /ops/apply-queue/{id}, PUT /ops/nodes/{id}, POST /ops/nodes/{id}/evacuate, POST /ops/node-overrides, DELETE /ops/node-overrides/{id}

Scope	Read / Write	Primary Endpoints
`read:backups`	Read	`GET /backups`, `GET /backups/{backup}`
`write:backups`	Write	`POST /backups`
`read:diagnostics`	Read	`GET /diagnostics/captures`, `GET /diagnostics/captures/{id}/{type}`, `POST /diagnostics/bundle`, `GET /diagnostics/trace`, `GET /diagnostics/rtp-quality`, `POST /diagnostics/test-plan`
`write:diagnostics`	Write	`POST /diagnostics/captures`
`read:cdr`	Read	`GET /cdr`, `GET /cdr/export`, `GET /reports/*`
`read:audit`	Read	`GET /audit-logs`
`read:security`	Read	`GET /security/users`, `GET /security/roles`, `GET /security/permissions`, `GET /security/api-tokens`
`write:security`	Write	`POST/PUT/DELETE /security/users`, `POST/PUT/DELETE /security/roles`, `POST/DELETE /security/api-tokens`
`simulate:routes`	Read	`POST /route-simulate`, `POST /route-simulate/batch`

Scope	Read / Write	Primary Endpoints
`read:api-docs`	Read	`GET /api-coverage`, `GET /soap/v1/wsdl`, `GET /soap/v1/operations`