# Vertex API Gateway Security & Configuration Specification

Version 2.4.0 – Production Release

**Author:** Vertex Security Operations
**Reviewer:** Nimbus Solutions Group
**Owner:** Synapse Engineering Division

**Date:** August 2, 2026

**Abstract**

This document defines the production security standards, traffic routing logic, and rate-limiting policies for the **Vertex API Gateway** deployed across the **Nimbus Corp** enterprise cloud infrastructure. It details configuration specifications for token authentication, path authorization, Cross-Origin Resource Sharing (CORS) security profiles, and compliance logging schemas. These settings are enforced globally to prevent unauthorized data access, mitigate distributed denial-of-service (DDoS) attempts, and maintain regulatory compliance with PCI-DSS v4.0 and SOC 2 Type II audits.

## Contents

**1 Introduction** **3**
1.1 Document Scope . . . 3
1.2 Architectural Flow . . . 3

**2 Authentication and Authorization** **3**
2.1 Token Validation Lifecycle . . . 4
2.2 Scope and Policy Mapping . . . 4
2.3 JWT Validation Configuration . . . 4

**3 Rate Limiting and Traffic Management** **5**
3.1 Token Bucket Algorithm Parameters . . . 5
3.2 Client Tier Security Profiles . . . 5
3.3 Rate Limit Filter Configuration . . . 5

**4 Routing, CORS, and Security Headers** **6**
4.1 CORS Profile Policies . . . 6
4.2 Security Response Headers . . . 7
4.3 Header Injection and Sanitization . . . 7

**5 Monitoring, Logging, and Auditing** **8**
5.1 Structured Logging and PII Masking . . . . . 8
5.2 Regulatory Compliance Mapping . . . . . 8

# 1 Introduction

In modern enterprise architectures, the API Gateway serves as the single entry point and enforcement boundary for all external and internal client interactions. For the **Nimbus Corp** digital banking ecosystem, the **Vertex Gateway** acts as the primary reverse proxy and security enforcement hub, mediating traffic between client applications and downstream microservices managed by the **Synapse Engineering** division.

This document details the security configurations, protocol enforcement rules, and traffic management policies deployed on the Vertex Gateway. The configuration ensures compliance with the enterprise security blueprint [4] and aligns with the federated identity guidelines of the Nimbus Identity & Access Management (IAM) framework [3].

## 1.1 Document Scope

This specification covers four critical dimensions of the Vertex Gateway configuration:

1. **Identity and Access Control**: Strict validation of JSON Web Tokens (JWT) issued by the federated identity provider, scope checking, and downstream claim propagation.
2. **Traffic Management and Rate Limiting**: Tiered rate-limiting rules utilizing the token bucket algorithm to prevent denial-of-service (DoS) attacks and ensure resource fairness.
3. **CORS and Routing Rules**: Path-based routing tables, cross-origin resource sharing restrictions, and secure header injection.
4. **Auditing and Compliance**: Log formats, SIEM ingestion protocols, and tracing patterns.

## 1.2 Architectural Flow

The network placement of the gateway and its interaction with the Nimbus Identity Provider (IdP) and downstream core APIs are illustrated in Figure 1.

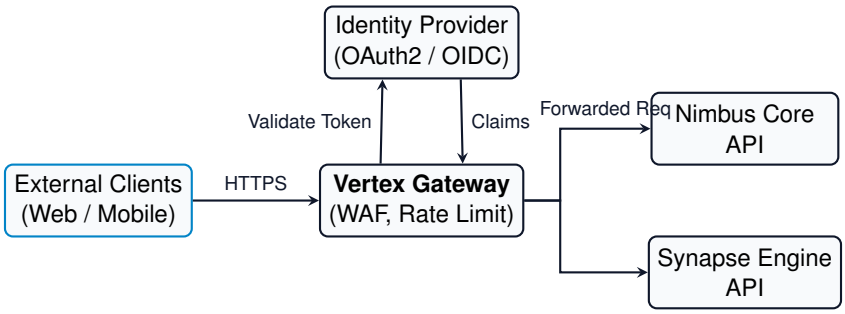


Figure 1: Vertex Gateway Network Placement and Request Execution Flow

By terminating TLS connections at the Vertex Gateway, downstream services are insulated from direct internet exposure. The gateway also intercepts malicious payloads, normalizes HTTP requests, and validates security claims before routing traffic to private subnets.

# 2 Authentication and Authorization

The Vertex Gateway implements a zero-trust boundary by requiring cryptographic authentication for all public-facing endpoints. Standardizing on OpenID Connect (OIDC) and the OAuth 2.0 framework [1], the gateway acts as a Policy Enforcement Point (PEP) while delegating token generation to the Nimbus Identity Provider (IdP) acting as the Policy Decision Point (PDP).

### 2.1 Token Validation Lifecycle

Every request targeting protected resources must present a JSON Web Token (JWT) [2] in the `Authorization` header as a bearer token. The gateway executes the validation lifecycle depicted below:

1. **Header Extraction**: Verifies the token format is compliant with `Bearer <token_string>`.
2. **Signature Verification**: Validates the cryptographic signature against the cached public keys retrieved from the Nimbus IdP's JSON Web Key Set (JWKS) endpoint. The cached keys have a time-to-live (TTL) of 24 hours. The default signature algorithms allowed are `RS256` and `ES256`.
3. **Temporal Verification**: Rejects tokens where the current time is before the issue time (`nbf` / not-before claim) or after the expiration time (`exp` claim). A clock-skew tolerance of 30 seconds is permitted.
4. **Audience and Issuer Checks**: Rejects tokens where the `iss` claim does not match `https://auth.nimbus.com/oauth2/v1` or the `aud` claim does not match the configured client identifiers of Vertex.

**JWT Signature Validation Policy**

To mitigate token-forgery attacks, the gateway is configured to explicitly reject tokens that utilize the `"alg": "none"` header value. Any request containing a signature algorithm other than those explicitly safelisted (e.g., `RS256`, `ES256`) is immediately dropped, and a `401 Unauthorized` response is returned.

### 2.2 Scope and Policy Mapping

Once token validation succeeds, the gateway performs path-level authorization using OAuth 2.0 scopes. Table 1 defines the security classification and the associated scopes for different downstream endpoints.

Table 1: Vertex Gateway Path Scope and Role Mapping

| API Context | HTTP Method | Required Scope | Security Tier |
|---|---|---|---|
| `/api/v1/retail/*` | `GET` | `retail:read` | Standard Access |
| `/api/v1/retail/*` | `POST, PUT` | `retail:write` | Elevated Confidential |
| `/api/v1/payments/*` | `POST` | `payment:write` | High Critical |
| `/api/v1/admin/*` | `GET, POST` | `admin:sys` | Core Infrastructure |

### 2.3 JWT Validation Configuration

The declarative YAML configuration for the JWT validation filter in the Vertex Gateway is shown in Listing 1.

```
filters
  - name jwt-validator
    enabled true
    config
      jwks_uri "https//auth.nimbus.com/oauth2/v1/keys"
```

```
      cache_ttl_seconds 86400
      clock_skew_seconds 30
      allowed_algorithms
        - RS256
        - ES256
      rules
        - match
            path "/api/v1/payments/**"
          required_claims
            iss "https//auth.nimbus.com/oauth2/v1"
            aud "payment-processing-engine"
            scp "paymentwrite"
```

Listing 1: YAML configuration snippet for Vertex Gateway JWT validation filter.

## 3 Rate Limiting and Traffic Management

To protect Nimbus core services from resource exhaustion, degradation, and distributed denial-of-service (DDoS) attacks, the Vertex Gateway implements distributed rate limiting. Traffic policing is enforced at the edge, utilizing a Redis-backed Token Bucket algorithm. This architecture guarantees low latency and synchronization of rate counters across the gateway cluster.

### 3.1 Token Bucket Algorithm Parameters

The rate limiting mechanism is defined by two primary parameters:

1. **Bucket Capacity** ($C$): The maximum number of tokens the bucket can hold. This defines the maximum burst size permitted for transient spikes in client requests.
2. **Refill Rate** ($R$): The rate at which tokens are added back to the bucket (tokens per second). This defines the sustained throughput limit for the client.

Every incoming request consumes exactly one token from the target bucket. If the bucket is empty, the request is rejected with a `429 Too Many Requests` status, and a standard response header is injected specifying the retry interval:

$$T_{\text{wait}} = \frac{1 - N_{\text{tokens}}}{R} \tag{1}$$

### 3.2 Client Tier Security Profiles

Rate limits are applied dynamically based on the authenticated client tier. Tiers are resolved from the JWT issuer claims and metadata. Table 2 presents the standard profiles enforced across the Vertex Gateway environment.

### 3.3 Rate Limit Filter Configuration

Listing 2 shows the YAML configuration for enforcing the client-tier rate limiting policies on the payment endpoint.

```
filters
  - name token-bucket-limiter
    enabled true
    config
```

Table 2: Dynamic Traffic Control Profiles by Client Tier

| Client Tier | Sustained Rate (R) | Burst Capacity (C) | Enforcement Key | Action on Limit |
|---|---|---|---|---|
| Anonymous Public | 10 req/sec | 20 req | Client IP Address | Delay & Reject |
| Standard Partner | 200 req/sec | 500 req | App ID (JWT Claim) | Throttle |
| Enterprise Partner | 1,000 req/sec | 2,500 req | App ID (JWT Claim) | Throttle & Alert |
| Internal Microservice | 5,000 req/sec | 10,000 req | Mutual TLS CN | Dynamic Scale |

```
      redis_cluster_nodes
        - "redis-sentinel-01.internal.nimbus.com26379"
        - "redis-sentinel-02.internal.nimbus.com26379"
      default_key_generator "claims.client_id"
      fallback_key_generator "request.client_ip"
      rate_limits
        - key "retail-mobile-app"
          refill_rate 100
          capacity 250
        - key "partner-synapse-b2b"
          refill_rate 500
          capacity 1000
```

Listing 2: YAML configuration for the Redis token-bucket rate limiter filter.

**DDoS Mitigation Override**

In the event of an active denial-of-service attack, security operations can execute an emergency override of the Vertex Gateway rate-limiting filter. This will dynamically lower the capacity of anonymous buckets to 2 req/sec and reject all requests bypassing the Cloudflare WAF, preventing traffic from reaching the internal virtual private cloud (VPC).

## 4 Routing, CORS, and Security Headers

Routing logic in the Vertex Gateway is designed to isolate downstream services from direct external paths. The gateway performs path matching, prefix stripping, and downstream header injection. Concurrently, it enforces strict Cross-Origin Resource Sharing (CORS) rules to restrict browser-based API access.

### 4.1 CORS Profile Policies

CORS policies are applied on a per-route basis. Fictional partner portals and mobile backends require distinct origins. The default security profile restricts cross-origin calls based on a strict safelist. Listing 3 illustrates the configuration for the Synapse dashboard origin.

```
cors
  enabled true
  allowed_origins
    - "https//dashboard.synapse.nimbus.com"
    - "https//portal.vertex-partner.net"
```

```
  allowed_methods
    - GET
    - POST
    - OPTIONS
  allowed_headers
    - Authorization
    - Content-Type
    - X-Requested-With
    - X-Correlation-ID
  exposed_headers
    - X-Response-Time
  allow_credentials true
  max_age_seconds 3600
```

Listing 3: CORS policy configuration block for public enterprise dashboards.

### 4.2 Security Response Headers

To protect client browsers against common vulnerability vectors such as Clickjacking, Cross-Site Scripting (XSS), and MIME-type sniffing, the Vertex Gateway injects standardized security response headers for all downstream responses. Table 3 summarizes the values enforced at the edge.

Table 3: Vertex Gateway Edge Security Headers

| Header Name | Value Enforced | Security Objective |
|---|---|---|
| `Strict-Transport-Security` | `max-age=63072000; includeSubDomains; preload` | Force TLS connection |
| `Content-Security-Policy` | `default-src 'none'; frame-ancestors 'none';` | Prevent script injection |
| `X-Frame-Options` | `DENY` | Clickjacking protection |
| `X-Content-Type-Options` | `nosniff` | Prevent MIME-type sniffing |
| `Referrer-Policy` | `strict-origin-when-cross-origin` | Avoid metadata leakage |
| `Permissions-Policy` | `geolocation=(), camera=(), microphone=()` | Disable browser hardware access |

### 4.3 Header Injection and Sanitization

In addition to generating downstream headers, the gateway executes sanitation protocols on incoming headers:

- **Strip Headers**: The gateway strips all headers matching `X-Nimbus-*` and `X-Vertex-*` from incoming client requests to prevent header injection attacks where clients fake authorization attributes.
- **Correlation ID Injection**: The gateway automatically generates a unique `X-Correlation-ID` (UUIDv4) for every request if not present. This ID is propagated down to the microservice chain to ensure trace correlation.

## 5 Monitoring, Logging, and Auditing

Security observability is critical for post-incident analysis, threat hunting, and regulatory compliance. The Vertex Gateway integrates directly with the Nimbus Enterprise Security Information and Event Management (SIEM) system to export real-time threat intelligence and audit logs.

### 5.1 Structured Logging and PII Masking

The gateway writes all log events in structured JSON format to standard output, which is collected by logging agents. To maintain compliance with privacy regulations, the gateway filters out all sensitive data.

- **Token Masking**: The `Authorization` header is stripped of its bearer token value in standard logs; only the token metadata (issuer, key ID, and expiry) is logged.
- **Request/Response Payload Masking**: Downstream debug payloads are parsed, and fields containing Personally Identifiable Information (PII) or Primary Account Numbers (PAN) are masked with asterisks before logging. For instance, card numbers are masked to only show the last four digits (`************1234`).

Listing 4 displays an example of an audit log entry emitted by the gateway after rejecting a client due to scope insufficiency.

```
{
  "timestamp": "2026-08-02T15:51:00.124Z",
  "correlation_id": "77e2992d-21e2-4c05-80f7-41c3a53ea354",
  "client_ip": "198.51.100.42",
  "http_method": "POST",
  "request_uri": "/api/v1/payments/transfers",
  "response_status": 403,
  "error_code": "INSUFFICIENT_SCOPE",
  "auth_metadata": {
    "issuer": "https://auth.nimbus.com/oauth2/v1",
    "subject": "usr_9a2b8e7c",
    "client_id": "retail-mobile-app",
    "provided_scopes": ["retail:read", "retail:write"],
    "missing_scopes": ["payment:write"]
  }
}
```

Listing 4: JSON structured audit log event representing an unauthorized request.

### 5.2 Regulatory Compliance Mapping

The configuration patterns detailed in this document map directly to standard compliance controls. Table 4 outlines the relationship between Vertex Gateway security capabilities and PCI-DSS, SOC 2, and ISO/IEC 27001 frameworks.

**Incident Response Triggers**

Any request that triggers a rate limiting block (HTTP 429) or authentication failure (HTTP 401/403) from a single IP address more than 100 times in 10 seconds is flagged by the SIEM as a brute-force or credential-stuffing attempt. This dynamically triggers a temporary IP block at the firewall level for 1 hour.

Table 4: Vertex Gateway Control Mapping to Compliance Frameworks

| Gateway Control | Functionality | PCI-DSS v4.0 | SOC 2 Type II | ISO 27001:2022 |
|---|---|---|---|---|
| JWT validation | Cryptographic auth | Req 8.3.1 | CC6.1, CC6.2 | Control A.8.20 |
| Rate Limiting | Denial of Service prevention | Req 11.4.1 | CC7.1 | Control A.8.22 |
| HSTS / HTTPS | Enforce data-in-transit encryption | Req 4.2.1 | CC6.6 | Control A.8.24 |
| CORS policies | Restrict cross-origin access | Req 6.4.3 | CC6.3 | Control A.8.28 |
| Audit logs | Masked JSON events to SIEM | Req 10.2.1 | CC7.2, CC7.3 | Control A.8.15 |

## References

[1] D. Hardt. The oauth 2.0 authorization framework. RFC 6749, IETF, October 2012.

[2] M. Jones, J. Bradley, and N. Sakimura. Json web token (jwt). RFC 7519, IETF, May 2015.

[3] Nimbus Identity & Access Team. *Nimbus Federated Identity Provider Integration Guide*. Nimbus Corp, v3.4.1 edition, 2024.

[4] Vertex Architecture Review Board. Vertex enterprise api gateway deployment blueprint. Internal Architecture Design Document V-ARCH-2025-092, Vertex Corp, 2025.