ARCHITECTURE STANDARD

TIÊU CHUẨN KỸ THUẬT — TIÊU CHUẨN KIẾN TRÚC

Document the Boundaries • One-Way Dependencies • Design for the Failure You Expect

Tổ chức / Organization	AI Biết Tuốt — Khung Quản trị Phát triển Phần mềm
Chủ sở hữu / Document Owner	Chief Architect / Kiến trúc sư trưởng
Phê duyệt / Approved by	Engineering Governance Council (hoặc ủy quyền cho Kiến trúc sư trưởng/Head of Engineering)
Ngày hiệu lực / Effective Date	17/09/2026
Phiên bản / Version	1.0
Chu kỳ rà soát / Review Cycle	Hằng năm, hoặc ngay khi công cụ/thực tiễn ngành thay đổi đáng kể
Tài liệu cấp trên / Parent Documents	Engineering Constitution v1.0 (Tầng 1); Architecture Governance Policy v1.0 (Tầng 2)
Phân loại / Classification	INTERNAL

SAMPLE DOCUMENT

A. Kiểm soát tài liệu

DOCUMENT CONTROL

Phiên bản	Ngày	Người soạn	Mô tả thay đổi
1.0	17/09/2026	Chief Architect / Kiến trúc sư trưởng	Ban hành lần đầu

B. Mục lục nội dung

CONTENTS

- 1. Mục đích
- 2. Phạm vi áp dụng
- 3. Liên kết với Policy liên quan
- 4. Định nghĩa
- 5. Yêu cầu kỹ thuật bắt buộc
- 6. Công cụ khuyến nghị
- 7. Đo lường tuân thủ
- 8. Cơ chế ngoại lệ
- 9. Chủ sở hữu tài liệu và chu kỳ rà soát
- 10. Tài liệu liên quan
- 11. Phê duyệt

1. Mục đích

Standard này cụ thể hóa **Architecture Governance Policy (AGP — Tầng 2)** thành các quy tắc kỹ thuật CỤ THỂ, đo lường được và kiểm tra được, áp dụng khi thiết kế và xây dựng hệ thống/dịch vụ phần mềm. AGP quy định NGUYÊN TẮC quản trị kiến trúc (Architecture Decision Record — ADR bắt buộc cho quyết định quan trọng, cơ chế rà soát Architecture Review Board/kiến trúc sư trưởng, danh mục công nghệ đã duyệt, đánh giá reuse trước khi xây mới). Standard này không lặp lại các nguyên tắc đó — giả định người đọc đã đồng ý với AGP — mà trả lời câu hỏi: khi thiết kế/xây một hệ thống cụ thể, chính xác phải tuân theo quy tắc kỹ thuật nào.

2. Phạm vi áp dụng

Áp dụng cho mọi dịch vụ/hệ thống phần mềm mới được thiết kế và mọi thay đổi kiến trúc đáng kể (theo tiêu chí "quyết định kiến trúc quan trọng" tại AGP) trên hệ thống hiện có, không phân biệt ngôn ngữ hay nền tảng công nghệ. Đối với hệ thống legacy đang vận hành và không có thay đổi kiến trúc đáng kể trong phạm vi rà soát hiện tại, Standard này áp dụng ở mức khuyến nghị cải thiện dần (xem Điều 5.4 và Điều 8), không hồi tố bắt buộc toàn bộ ngay lập tức.

3. Liên kết với Policy liên quan

- **Architecture Governance Policy (AGP):** Standard này cụ thể hóa trực tiếp AGP — đặc biệt các điều khoản về quyết định kiến trúc quan trọng cần ADR, nội dung tối thiểu của ADR, C4 Model được khuyến nghị làm ngôn ngữ sơ đồ, đánh giá reuse/buy/extend, và cơ chế Architecture Review Board/kiến trúc sư trưởng. Tài liệu kiến trúc tạo ra khi áp dụng Điều 5 của Standard này (sơ đồ C4, API contract...) chính là bằng chứng đính kèm ADR theo AGP.
- **Software Development Governance Policy (SDGP):** mở rộng thực hành từ Điều 5.8 (quyết định dựa trên bằng chứng — ADR) và Điều 5.9 (tái sử dụng trước khi xây mới) của SDGP.
- Khi có mâu thuẫn giữa Standard này và AGP/SDGP về quy trình phê duyệt hoặc vai trò/trách nhiệm, AGP/SDGP là tài liệu cấp trên và có hiệu lực cao hơn; Standard này chỉ bổ sung chi tiết kỹ thuật, không thay thế quy trình quản trị đã quy định.

4. Định nghĩa

- **C4 Model:** đã định nghĩa tại AGP — phương pháp trực quan hóa kiến trúc phần mềm theo bốn mức chi tiết tăng dần (System Context, Container, Component, Code) do Simon Brown phát triển. Standard này chỉ quy định mức tối thiểu bắt buộc phải có (Điều 5.1), không định nghĩa lại khái niệm.
- **API contract:** đặc tả hình thức, có thể đọc bằng máy, mô tả giao diện của một API trước khi triển khai — ví dụ đặc tả OpenAPI cho REST, hoặc file `.proto` (Protocol Buffers) cho gRPC — dùng làm nguồn tham chiếu chung, thống nhất giữa bên cung cấp và bên tiêu thụ API. Cách tiếp cận thiết kế API bắt đầu từ contract trước khi viết code thường được gọi là **API-first design** hoặc **contract-first development**.
- **OpenAPI Specification (OAS):** đặc tả mở, độc lập ngôn ngữ để mô tả REST API, hiện do **OpenAPI Initiative** — một dự án hợp tác dưới Linux Foundation — duy trì, đang ở nhánh phiên bản 3.x (bản mới nhất tính đến 2026 thuộc dòng 3.2.x).

- **AsyncAPI:** đặc tả tương tự OpenAPI nhưng dành cho giao tiếp bất đồng bộ qua message broker/event stream (Kafka, RabbitMQ, MQTT...), do AsyncAPI Initiative duy trì — dùng làm tham chiếu khi Điều 5.2.4 áp dụng cho giao tiếp bất đồng bộ.
- **Dependency Inversion Principle (DIP):** một trong năm nguyên tắc SOLID, được Robert C. Martin ("Uncle Bob") phổ biến qua các bài viết và cuốn sách *Agile Software Development: Principles, Patterns, and Practices*. Nguyên tắc gồm hai phần: module cấp cao không nên phụ thuộc vào module cấp thấp — cả hai nên phụ thuộc vào abstraction; abstraction không nên phụ thuộc vào chi tiết, chi tiết nên phụ thuộc vào abstraction.
- **Twelve-Factor App:** phương pháp luận gồm 12 yếu tố cho xây dựng ứng dụng SaaS hiện đại, dễ mở rộng, dễ vận hành trên nền tảng cloud — công bố năm 2011 bởi Adam Wiggins, đồng sáng lập Heroku, đúc kết kinh nghiệm vận hành hàng trăm nghìn ứng dụng trên nền tảng Heroku, hiện là tài liệu mã nguồn mở tại 12factor.net.
- **Circular dependency (phụ thuộc vòng tròn):** tình trạng hai hoặc nhiều module/service phụ thuộc lẫn nhau tạo thành vòng khép kín, khiến không thể build, triển khai hoặc thay đổi độc lập từng phần.
- **Single point of failure (SPOF):** một thành phần trong hệ thống mà nếu gặp sự cố sẽ làm toàn bộ hoặc một phần quan trọng của hệ thống ngừng hoạt động, không có phương án dự phòng thay thế.
- **Trace ID / Correlation ID:** định danh dùng để nối các bản ghi log/sự kiện thuộc cùng một luồng xử lý xuyên nhiều dịch vụ, phục vụ truy vết (distributed tracing) — chuẩn phổ biến hiện nay là W3C Trace Context (header `traceparent` / `tracestate`), được hệ sinh thái OpenTelemetry và phần lớn nhà cung cấp APM hỗ trợ.

5. Yêu cầu kỹ thuật bắt buộc

MANDATORY TECHNICAL REQUIREMENTS

5.1 Tài liệu hóa kiến trúc tối thiểu bằng C4 Model

Mọi dịch vụ/hệ thống mới, hoặc thay đổi kiến trúc đáng kể trên hệ thống hiện có (theo tiêu chí tại AGP), **PHẢI (MUST)** được tài liệu hóa tối thiểu ở hai mức C4 Model:

- **System Context** — hệ thống trong bối cảnh với người dùng và các hệ thống khác nó tương tác.
- **Container** — các khối triển khai độc lập bên trong hệ thống (ứng dụng, dịch vụ, cơ sở dữ liệu, hàng đợi...) và cách chúng giao tiếp với nhau.

Hai sơ đồ này **PHẢI (MUST)** được lưu cùng ADR liên quan theo AGP (docs-as-code) — không lưu rời rạc trong công cụ trình chiếu cá nhân không kiểm soát phiên bản. Mức Component và Code **CÓ THỂ (MAY)** được bổ sung tùy độ phức tạp hệ thống, không bắt buộc — kinh nghiệm ngành cho thấy hai mức này thường thay đổi quá nhanh để vẽ tay và duy trì cập nhật liên tục.

5.2 API contract bắt buộc trước khi triển khai

1. Mọi API nội bộ (giữa các service trong tổ chức) hoặc API bên ngoài (public/đối tác) **PHẢI (MUST)** có API contract xác định TRƯỚC KHI triển khai code (API-first/contract-first) — ví dụ đặc tả OpenAPI cho REST, hoặc file `.proto` cho gRPC.
2. API contract **PHẢI (MUST)** được lưu trong kho mã nguồn (docs-as-code), có kiểm soát phiên bản, và là nguồn tham chiếu duy nhất (single source of truth) cho hợp đồng giao diện — không dùng tài liệu rời rạc mô tả lại API bằng văn xuôi làm nguồn chính thức.
3. Thay đổi có khả năng phá vỡ khả năng tương thích ngược (breaking change — ví dụ xóa/đổi tên field bắt buộc, đổi kiểu dữ liệu, đổi hành vi endpoint hiện có) **PHẢI (MUST)** có chiến lược versioning rõ ràng (ví dụ: version hóa theo đường dẫn hoặc header, phát hành song song phiên bản cũ trong giai đoạn chuyển tiếp), công bố trước cho bên tiêu thụ API. **KHÔNG ĐƯỢC PHÉP (MUST NOT)** thay đổi ngầm gây vỡ client hiện có.
4. Đối với giao tiếp bất đồng bộ qua message broker/event stream, đội kỹ thuật **NÊN (SHOULD)** áp dụng nguyên tắc tương tự: xác định event schema trước khi triển khai (ví dụ đặc tả AsyncAPI hoặc JSON Schema/Protobuf cho payload) và có chiến lược tương thích ngược/xuôi khi thay đổi.

5.3 Dependency một chiều giữa các module/service

- Dependency giữa các module/service chính **PHẢI (MUST)** tuân theo một hướng rõ ràng, hợp lý theo lớp/domain (ví dụ: lớp trình bày phụ thuộc lớp nghiệp vụ, lớp nghiệp vụ phụ thuộc abstraction của lớp hạ tầng — tinh thần Dependency Inversion Principle — không phụ thuộc ngược lại).
- **KHÔNG ĐƯỢC PHÉP (MUST NOT)** tạo circular dependency giữa các module/service chính (ví dụ: service A gọi service B, service B lại phụ thuộc ngược vào service A theo cách không thể tách rời).
- Đội kỹ thuật **NÊN (SHOULD)** tích hợp công cụ phân tích kiến trúc/dependency-lint vào pipeline CI khi hệ sinh thái công nghệ đang dùng có công cụ phù hợp (ví dụ các công cụ phân tích phụ thuộc theo ngôn ngữ tương ứng), để phát hiện tự động vi phạm hướng phụ thuộc hoặc circular dependency — thay vì chỉ dựa vào rà soát thủ công. Đây là khuyến nghị "khi khả thi", không bắt buộc với mọi ngôn ngữ/nền tảng nếu chưa có công cụ trưởng thành tương ứng.

5.4 Áp dụng Twelve-Factor App tương xứng bối cảnh

Dịch vụ chạy trên hạ tầng cloud/container **NÊN (SHOULD)** tuân theo các yếu tố phù hợp của Twelve-Factor App tương xứng với bối cảnh công nghệ, ví dụ (không giới hạn):

- **Config:** cấu hình (kết nối cơ sở dữ liệu, API key, tham số môi trường) qua biến môi trường, không hard-code trong mã nguồn hoặc commit kèm secret.
- **Processes/Disposability:** xử lý stateless ở tầng ứng dụng khi có thể, để hỗ trợ scale ngang và khởi động/tắt nhanh, an toàn.
- **Logs:** ghi log ra stream chuẩn (stdout/stderr) thay vì tự quản lý và xoay vòng file log thủ công trong container.

> **Lưu ý:** đây là khuyến nghị áp dụng CÓ CHỌN LỌC theo mức độ phù hợp — không phải mọi hệ thống legacy đều bắt buộc tuân thủ toàn bộ 12 yếu tố ngay lập tức. Hệ thống mới xây trên nền tảng cloud-native **NÊN (SHOULD)** áp dụng đầy đủ hơn; hệ thống legacy áp dụng dần theo lộ trình cải thiện, ưu tiên yếu tố chi phí thấp/rủi ro thấp trước (ví dụ tách cấu hình khỏi mã nguồn thường dễ thực hiện và nên làm sớm).

5.5 Mục tiêu phi chức năng cho hệ thống quan trọng

Hệ thống được phân loại quan trọng (theo tiêu chí phân loại mức độ quan trọng do tổ chức quy định, ví dụ theo mức ảnh hưởng kinh doanh, số người dùng, hoặc mức độ nhạy cảm dữ liệu xử lý) **PHẢI (MUST)** xác định rõ mục tiêu phi chức năng tối thiểu ngay từ giai đoạn thiết kế, ghi rõ trong tài liệu kiến trúc (kèm ADR liên quan), tối thiểu gồm:

- **Availability** — ví dụ mục tiêu uptime hoặc SLO cụ thể theo ngữ cảnh hệ thống.
- **Latency** — ví dụ ngưỡng thời gian phản hồi chấp nhận được cho các luồng chính.
- **Throughput** — ví dụ khối lượng giao dịch/yêu cầu dự kiến cần đáp ứng.

Các mục tiêu này **KHÔNG ĐƯỢC PHÉP (MUST NOT)** để ngầm định hoặc chỉ tồn tại trong đầu người thiết kế. Con số cụ thể do tổ chức/đội phụ trách hệ thống tự quy định tương xứng với bối cảnh nghiệp vụ và rủi ro — Standard này không áp đặt một con số chung cho mọi hệ thống.

5.6 Đối chiếu Technology Radar trước khi chọn công nghệ mới

Lựa chọn công nghệ mới (ngôn ngữ lập trình, framework, cơ sở dữ liệu, dịch vụ bên thứ ba) cho hệ thống production **PHẢI (MUST)** được đối chiếu với danh mục công nghệ đã duyệt (Technology Catalog/Technology Radar) tại AGP trước khi áp dụng. Công nghệ nằm ngoài danh mục đã duyệt **PHẢI (MUST)** đi qua quy trình ngoại lệ hoặc rà soát ADR bổ sung theo AGP trước khi dùng cho production — không được tự ý áp dụng rồi hợp thức hóa sau.

5.7 Xác định và giảm thiểu Single Point of Failure

Đối với hệ thống được phân loại quan trọng (Điều 5.5), đội thiết kế **PHẢI (MUST)** xác định rõ các single point of failure tiềm ẩn (hạ tầng, dịch vụ bên thứ ba, thành phần dữ liệu...) và có phương án giảm thiểu (redundancy, failover, circuit breaker, triển khai đa vùng sẵn sàng...) tương xứng với mục tiêu availability đã đặt ra tại Điều 5.5 — không nhất thiết loại bỏ hoàn toàn mọi SPOF nếu chi phí không tương xứng rủi ro, nhưng rủi ro còn lại **PHẢI (MUST)** được ghi nhận có chủ đích (ví dụ trong ADR), không được bỏ sót do chưa từng được xác định.

5.8 Truy vết xuyên dịch vụ (distributed tracing)

Khi một hệ thống được cấu thành từ nhiều dịch vụ giao tiếp với nhau (qua HTTP, gRPC, hoặc message broker), các dịch vụ **NÊN (SHOULD)** tiếp nhận và truyền tiếp (propagate) một Trace ID/ Correlation ID xuyên suốt luồng xử lý — khuyến nghị theo chuẩn W3C Trace Context khi nền tảng/thư viện đang dùng hỗ trợ — để có thể truy vết một request/sự kiện qua nhiều dịch vụ khi điều tra sự cố. Yêu cầu này ở mức NÊN (không phải PHẢI) vì mức độ cần thiết phụ thuộc số lượng dịch vụ tham gia một luồng xử lý; hệ thống chỉ có một dịch vụ duy nhất chưa cần áp dụng.

6. Công cụ khuyến nghị

RECOMMENDED TOOLING

Mục đích	Ví dụ công cụ/kỹ thuật
Vẽ sơ đồ C4 Model	Structurizr, PlantUML (thư viện C4-PlantUML), Mermaid (cú pháp C4 diagram)
Viết/kiểm tra API contract REST	Trình soạn thảo OpenAPI (Swagger Editor), công cụ lint đặc tả OpenAPI
Kiểm tra API contract gRPC	Công cụ lint/kiểm tra breaking-change cho file `.proto`
Đặc tả event/message bất đồng bộ	AsyncAPI, JSON Schema, Protobuf cho payload
Phát hiện circular dependency / dependency-lint	Công cụ phân tích phụ thuộc phù hợp theo ngôn ngữ/nền tảng đang dùng — đội tự chọn theo hệ sinh thái công nghệ của mình
Truy vết xuyên dịch vụ	Thư viện/nền tảng tương thích OpenTelemetry và chuẩn W3C Trace Context

Bảng trên là ví dụ tham khảo, không bắt buộc dùng đúng công cụ liệt kê — đội kỹ thuật lựa chọn công cụ phù hợp với hệ sinh thái công nghệ của mình, miễn đạt được mục tiêu kiểm soát tại Điều 5.

7. Đo lường tuân thủ

COMPLIANCE METRICS

- Tỷ lệ hệ thống/dịch vụ mới có đủ tài liệu C4 Model (System Context + Container) kèm ADR liên quan tại thời điểm ra mắt (Điều 5.1).
- Tỷ lệ API (nội bộ/bên ngoài) có API contract lưu trong kho mã nguồn, kiểm soát phiên bản (Điều 5.2).
- Số lượng breaking change API được phát hiện KHÔNG có chiến lược versioning trước khi phát hành — mục tiêu bằng 0, hoặc giảm dần nếu đã từng phát hiện.
- Số lượng circular dependency được công cụ lint phát hiện trong các module/service chính (Điều 5.3), theo dõi xu hướng giảm dần.
- Tỷ lệ hệ thống quan trọng có ghi rõ mục tiêu availability/latency/throughput trong tài liệu kiến trúc (Điều 5.5).
- Tỷ lệ lựa chọn công nghệ mới cho production có đối chiếu Technology Radar/AGP trước khi áp dụng (Điều 5.6).
- Tỷ lệ hệ thống quan trọng đã xác định SPOF và có phương án giảm thiểu tương ứng (Điều 5.7).

8. Cơ chế ngoại lệ

EXCEPTIONS

Ngoại lệ với bất kỳ yêu cầu nào tại Điều 5 (ví dụ: hệ thống quá nhỏ không cần đủ hai mức C4, hệ thống legacy chưa thể tách API contract ngay, thời hạn gấp không kịp đối chiếu Technology Radar trước khi thử nghiệm ở môi trường không phải production...) áp dụng đúng quy trình ngoại lệ đã quy định tại AGP (và

SDGP ở mức tổng quát hơn) — ghi nhận bằng văn bản, có thời hạn, người phê duyệt phù hợp (kiến trúc sư trưởng/Architecture Review Board tùy phạm vi ảnh hưởng). Standard này không định nghĩa lại quy trình ngoại lệ hay vai trò và trách nhiệm liên quan — tham chiếu AGP để biết chi tiết đầy đủ.

9. Chủ sở hữu tài liệu và chu kỳ rà soát

DOCUMENT OWNER & REVIEW CYCLE

Chủ sở hữu: **Chief Architect / Kiến trúc sư trưởng**. Rà soát hằng năm, hoặc ngay khi có thay đổi đáng kể về công nghệ nền tảng, phương pháp luận kiến trúc phổ biến trong ngành, hoặc phát sinh sự cố lớn liên quan trực tiếp đến vi phạm một trong các yêu cầu tại Điều 5.

10. Tài liệu liên quan

RELATED DOCUMENTS

- Engineering Constitution v1.0 (Tầng 1)
- Software Development Governance Policy (SDGP) v1.0
- Architecture Governance Policy (AGP) v1.0
- Technology Catalog / Technology Radar, ADR Template (tài liệu vận hành đi kèm AGP)

"Vẽ ranh giới trước khi viết dòng code đầu tiên. Để phụ thuộc chảy một chiều. Và giả định thành phần quan trọng nhất của hệ thống — rồi hỏi: nếu nó chết, điều gì xảy ra tiếp theo?"

11. Phê duyệt

APPROVAL

Vai trò	Họ tên	Chữ ký	Ngày
Chief Architect / Kiến trúc sư trưởng (Document Owner)			
Head of Engineering			
Đại diện Engineering Governance Council			