# Chapter 10

# Service API and Contract Versioning with Web Services and REST Services

Sau khi một nhóm dịch vụ được triển khai, các chương trình tiêu thụ sẽ tự nhiên bắt đầu hình thành các phụ thuộc vào nó. Khi chúng tôi buộc phải thực hiện các thay đổi vào các dịch vụ sau này, chúng ta cần phải tìm hiểu:

Chương này sẽ làm rõ ra các vấn đề

* Những thay đổi có ảnh hưởng tiêu cực đến các tiêu thụ dịch vụ hiện tại (và có thể là tương lai) không?
* Làm thế nào để triển khai và thông báo những thay đổi sẽ và không sẽ ảnh hưởng đến các tiêu thụ.
* Điều gì cụ thể tạo thành một phiên bản mới của một hợp đồng dịch vụ? Sự khác biệt giữa một phiên bản chính và một phiên bản phụ?
* Các phần của số phiên bản đề cập đến điều gì?
* Phiên bản mới của hợp đồng liệu có hoạt động với các tiêu thụ hiện tại đã được thiết kế cho phiên bản hợp đồng cũ không?
* Phiên bản hiện tại của hợp đồng liệu có hoạt động với các tiêu thụ mới có thể có yêu cầu trao đổi dữ liệu khác nhau không?
* Phương pháp tốt nhất để thêm các thay đổi vào các hợp đồng hiện tại trong khi giảm thiểu ảnh hưởng đối với các tiêu thụ là gì?
* Chúng ta cần phải lưu trữ các hợp đồng cũ và mới cùng một lúc không? Nếu có, trong bao lâu?

# 10.1 Versioning Basics

Chương này đề cập đến việc tạo ra phiên bản mới của hợp đồng dịch vụ và sự khác biệt giữa các phiên bản lớn và nhỏ.

Khi chúng ta nói rằng chúng ta đang tạo ra một phiên bản mới của một hợp đồng dịch vụ, chúng ta đang ám chỉ đến việc cập nhật các điều khoản và điều kiện mà các dịch vụ sẽ tuân thủ. Cụ thể, việc tạo ra một phiên bản mới của hợp đồng dịch vụ có thể bao gồm:

## Versioning Web Services

một hợp đồng dịch vụ Web có thể bao gồm của một số tài liệu và định nghĩa riêng lẻ được liên kết và tập hợp lại với nhau để tạo thành một giao diện kỹ thuật hoàn chỉnh.

Ví dụ: một hợp đồng dịch vụ Web nhất định có thể bao gồm:

• Một (đôi khi nhiều hơn) định nghĩa WSDL

• Một (thường là nhiều) định nghĩa Lược đồ XML

• Một số định nghĩa về Chính sách WS (đôi khi không có)

Do đó, khi chúng ta cần tạo một phiên bản mới của hợp đồng dịch vụ Web, chúng ta có thể giả sử rằng đã có một sự thay đổi trong mô tả WSDL trừu tượng hoặc một trong những nhược điểm các tài liệu đường dẫn liên quan đến mô tả WSDL trừu tượng. Hợp đồng dịch vụ Web nội dung thường có thể thay đổi là nội dung lược đồ XML cung cấp các kiểu cho các định nghĩa thông điệp của mô tả trừu tượng. Cuối cùng, một công nghệ liên quan đến hợp đồng khác vẫn có thể áp đặt các yêu cầu về phiên bản nhưng ít có khả năng thực hiện được điều đó hơn vì đây là một phần ít phổ biến hơn trong các hợp đồng dịch vụ Web là WS-Policy.

## Versioning REST Services

Nếu chúng ta tuân thủ mô hình REST bằng cách sử dụng một contract đồng nhất để diễn đạt khả năng của dịch vụ, việc chia sẻ tài liệu định nghĩa giữa các contract dịch vụ trở nên rõ ràng hơn. Ví dụ,

• Tất cả các phương thức HTTP được sử dụng trong các hợp đồng đều là tiêu chuẩn trên toàn bộ kiến trúc.

• Các định nghĩa XML Schema là tiêu chuẩn, vì chúng được bao gói trong các loại phương tiện chung.

• Cú pháp nhận dạng cho các điểm cuối dịch vụ nhẹ (được gọi là tài nguyên) là tiêu chuẩn trên toàn bộ kiến trúc.

Các thay đổi vào các mặt của hợp đồng đồng nhất mà đứng sau mỗi hợp đồng dịch vụ có thể ảnh hưởng đến bất kỳ dịch vụ REST nào trong danh mục dịch vụ.

## Fine and Coarse-Grained Constraints

1. Ràng buộc Fine-Grained

Đây là các ràng buộc chi tiết được áp dụng cho từng tài nguyên cụ thể hoặc một số lượng nhỏ các tài nguyên

Ví dụ:

* Ràng buộc về phương thức HTTP: RESTful APIs thường giới hạn việc sử dụng các phương thức HTTP nhất định như GET, POST, PUT, DELETE để thực hiện các thao tác trên tài nguyên.
* Ràng buộc về định dạng dữ liệu: API có thể yêu cầu dữ liệu được trả về hoặc gửi đi dưới một định dạng cụ thể như JSON hoặc XML.
* Ràng buộc về quản lý trạng thái: RESTful APIs thường không lưu trữ trạng thái của phiên làm việc, thay vào đó, mọi thông tin cần thiết để thực hiện yêu cầu được gửi từ máy khách.

1. Ràng buộc Coarse-Grained

Đây là các ràng buộc toàn cục được áp dụng cho toàn bộ dịch vụ hoặc một phần lớn các tài nguyên.

Ví dụ:

* Ràng buộc về đồng nhất giao diện: Tất cả các dịch vụ RESTful nên tuân thủ một số quy tắc và tiêu chuẩn chung như HATEOAS, RESTful URI, và nguyên tắc không lưu trữ trạng thái.
* Ràng buộc về phân cấp: Các tài nguyên RESTful nên được tổ chức theo cấu trúc phân cấp, giúp tăng tính tổ chức và hiệu quả trong việc quản lý dữ liệu và tài nguyên.
* Ràng buộc về tách biệt hóa: Mỗi tài nguyên RESTful nên đại diện cho một khía cạnh cụ thể của hệ thống và nên được thiết kế sao cho độc lập với các tài nguyên khác.

# 10.2 Versioning and Compatibility

Mối quan tâm số một khi phát triển và triển khai phiên bản mới của dịch vụ hợp đồng là tác động của nó đối với các bộ phận khác của doanh nghiệp đã hình thành hoặc sẽ hình thành sự phụ thuộc vào nó. Biện pháp tác động này liên quan trực tiếp đến mức độ tương thích của phiên bản hợp đồng mới với phiên bản cũ và môi trường xung quanh nó nói chung. Phần này thiết lập các loại tương thích cơ bản liên quan đến nội dung và thiết kế các phiên bản hợp đồng mới, đồng thời gắn liền với các mục tiêu và hạn chế của các chiến lược tạo phiên bản khác nhau được giới thiệu ở cuối chương này.

# 10.3 REST Service Compatibility Considerations

# 10.4 Version Identifi ers

# 10.5 Versioning Strategies

# 10.6 REST Service Versioning Considerations