# Versioning Strategy for Web Services API

This document outlines the strategy for versioning the Web Services API to ensure backward compatibility, usability, and clarity for clients.

## 1. Introduction

API versioning is necessary to safely evolve the API without breaking existing client integrations. This strategy describes when and how to introduce new versions.

## 2. Versioning Scheme

The API will use a simple MAJOR versioning format:

```
v1, v2, v3, ...
```

Only the MAJOR version number will be exposed to clients.

## 3. Versioning Location

The API version will be included in the URL path:

```
https://api.example.com/v1/resource
```

Other schemes such as header-based versioning are not supported.

## 4. Versioning Policy

- Non-breaking changes (e.g., adding fields to responses) **do not** require a new version.
- Breaking changes (e.g., removing fields, changing data types) **require** a new MAJOR version.
- Deprecated versions will be supported for at least 12 months before removal.

## 5. Example Endpoints

```
GET /v1/users/123
GET /v2/users/123
```

## 6. Version Lifecycle

| Status | Description |
|---|---|
| Active | Supported for new and existing clients. |
| Deprecated | Supported, but will be removed after notification period. |
| Retired | No longer supported. Requests will return an error. |

## 7. Deprecation & Communication

- Deprecation notices will be communicated at least 3 months before removal.
- Announcements via email and developer portal.
- Clients are encouraged to migrate to the latest version as soon as possible.

## 8. Changelog

Detailed change logs will be published with each new version to guide client migrations.