paymentsnz

API Standards Version & Lifecycle Management Policy

Version:	1.0
Published date:	11 April 2019
Author:	Matthew Bell, Payments NZ
Approved by:	Payments NZ Board
Approval date:	11 April 2019

Contents

Purpose

This policy will outline how the API Service will manage the versioning and lifecycle management of supported API standards. This has been agreed by the API working groups and approved by the API Council for reference when assessing the current status of supported standards.

Standards Versioning

Standards versioning refers to the change management system used to categorise and record the different versions of an API standard over time. The standards versioning system provides a framework for how an API's major, minor and patch changes are managed and published. For example versions will be displayed as per the following:

- Major **3**.0.0
- Minor 2.**4**.0
- Patch 2.4.**1**

Lifecycle Management

How the stages of an API standard's lifecycle are defined and managed, ranging from its conceptual inception at a planning stage, through to the expiration of the standard. Lifecycle management includes key consultation stages, any required timeframes, and the notification requirements as a standard moves through the lifecycle.

This is managed using the following eight stages:

- Standards development pipeline plan
- Requirements
- Standards Development
- Release candidate
- Approval
- Version Release
- Deprecation
- Expiry

Standards Version Management

The standards versioning approach for the API Service has been defined in the below table, along with a description of high level roles and processes.

Description of semantic versions	Semantic version	Roles and process
MAJOR version is when you either make backwards-incompatible API changes, or introduce significant new feature(s). A backwards incompatible change might result in Third Party applications failing (unless upgraded via a systems change process). For example, making deletions, changing mandatory fields, or any change that impacts the overall interpretation of the standard. A significant new feature might introduce a new business function, for example introducing new customer account information data types. Generally, this will be a planned and prioritised initiative developed through the standards lifecycle process. These would be a subjective decision for whether or not the new features constitute a major change.	Semantic version control is **x**.0.0. For example, **1**.0.0, or **2**.0.0 A major version: • MUST be incremented if any backwards incompatible changes are introduced to the public API. • It MAY be incremented if a significant new feature is introduced to the API standard. • It MAY include minor and patch level changes. • Patch and minor version MUST be reset to 0 when major version is incremented, e.g. 1.3.14 → 2.0.0. As an indicative guide only, a major version change to a standard might occur annually. As an indicative guide, it is expected that an 'n minus 2' approach be taken, being the latest major version and the last two	Major versions: • MUST be approved by the API Council. • The API Council MUST assess impacts, scope and risks of changes. • MUST follow the full standards lifecycle process end-to-end, including: planning and prioritisation, appropriately consulting API Standards Users, notifications to Standards Users with respect to the new major version, its deprecated date, and its expiry (refer standards lifecycle management section for more information). Note that major versions are deprecated and terminated, over time, as per the standards lifecycle management process. Expired major standards include all minor and patch versions attached to it.

Description of semantic versions	Semantic version	Roles and process
	older versions also remain available, with at least the oldest version being deprecated for expiry. For example, when version 3.0.0 is released, version 1.0.0 should be deprecated and moved towards expiry.	
MINOR: version is when you add functionality in a backwards-compatible manner, e.g. making changes to the standards, such as additions, that do not have any risk of making Third Party applications fail. A Minor version might also introduce new significant new feature.	Semantic version control is 0.**x**.0. For example, 1.**2**.0, or 1.**3**.0 A minor version: • MUST be incremented if new, backwards compatible functionality is introduced to the API standard. • It MUST be incremented if any API functionality is marked as deprecated. • It MAY be incremented if substantial new functionality or improvements are introduced. • It MAY include patch level changes. • Patch version MUST be reset to 0 when minor version is incremented, e.g. 1.3.7 → 1.4.0 As an indicative guide only, a minor version change to a standard might occur every 3 to 6 months.	Minor versions • MAY be approved by the API Council (to be discussed with the Council). • MAY be approved by management of the API Service. • The API Council MAY assess impacts, scope and risks of changes. • MAY originate via the planning stage of the standards lifecycle management process, or MAY be a result of a new issue, opportunity, or a change request being received from a Standards User. • The API Service MAY publish one or more minor releases at any time. Note that minor versions are not deprecated or expired (deprecation is applied to major versions only, and by extension all minor and patch versions within that major version).

Description of semantic versions	Semantic version	Roles and process
PATCH: version when backwards-compatible bug fixes are made, e.g. corrections of minor technical errors.	Semantic version control is 0.0.x. For example, 1.0.**1**, or 1.0.**2**. A patch version: • MUST be incremented if only backwards compatible bug fixes are introduced. A bug fix is defined as an internal change that fixes incorrect behaviour. As an indicative guide only, a patch version change to a standard might occur at any time, such as monthly.	Patch versions: • MUST be approved by management of the API Service. • MAY originate via a Standards User or API Community Member change request. • MAY originate via a recommendation from the Technical Standards Group, or any other group within the API Service.
Release candidate: Pre-release versions of any potential forthcoming patch, minor or major release. To enable the API Service to publish draft updates for Standards Users review and provide feedback. A release candidate could be a work-in-progress draft that is under development, a draft for consultation, or a final draft pending approval.	Semantic version control is 1.0.0-**rc1**, or 1.0.0-**rc2.** A release candidate: • Should NOT be relied on as confirmation of functionality in the next release. A release candidate indicates that the version might be unstable and might not satisfy the intended compatibility requirements as denoted by its associated normal version.	A release candidate could be: • a work-in-progress draft that is under development, • a draft for consultation, or • a final draft pending approval.

Lifecycle Management

The below table details how the stages of an API standard's lifecycle are defined and managed. Ranging from its conceptual inception at a planning stage, through to the expiration of the standard. Lifecycle management includes key consultation stages, any required timeframes and the notification requirements as a standard moves through the lifecycle.

The following lifecycle generally applies to Major version releases, e.g. **2**.0.0. The lifecycle could also apply to Minor version releases if deemed appropriate, e.g. 2.**1**.0, but the level of effort/involvement would be right-sized in proportion to the smaller scaled nature of the Minor change.

Lifecycle Stage	Summary Description	Process & Responsibility
Standards Development Pipeline Plan	A high level plan setting out standards development goals, strategy and priorities for upcoming standards development work. Includes expected standards deprecation plans. Consultation with Standards Users required.	Via consultation process and through applicable standards governance groups. Agreed by the API Council. Approved by the Payments NZ Board.
Requirements	Business description of the outcomes and functionality that new standards versions are to deliver.	Via the API Service's team and applicable groups. Agreed by the API Council.
Standards Development	Develop draft API standards that deliver against the Requirement's.	By the API technical group.
Release candidate	Draft API standard socialised for wider consultation and review process prior to its approval. Consultation with Standards Users required.	Option for API Standards Users, the API technical group, and potentially API Community Members to provide feedback prior to approval of the standard.

Approval	Approval of the standards version.	By the API Council.
Version Release	Publication of the approved standard. Includes adding the standard into the industry sandbox, and notifications to Standards Users.	By the API Service team, in line with the approval.
Deprecation	Approval that a given standard version will expire and no longer be valid for use in the market from a specific future date. Includes a requirement to notify Standards Users. Unless the standard has its security compromised, a notice period of at least 6 months must be given before its expiry.	Approval by the API Council. Notification by the API Service team.
Expiry	A date when a version of a standard is no longer valid or supported. Standards Users have an obligation to no longer use the standard after it has expired. Includes a requirement to notify Standards Users.	By the API Service team, as per approved deprecation date. Includes notification.