Versionamiento Semántico de APIs Gestionadas y Servicios de Integración

Grupo Bancolombia

Versión 2.0.1

**Control del Documento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registro Cambios al Documento** | | | |  |
| **Fecha** | **Motivo** | **Realizado por** | **Observaciones** | **Versión** |
| 2016/09/07 | Creación del documento | Equipo de Gobierno SOA | Elaborado por: | 0.1.0 |
| 2017/04/28 | Actualización del documento | Equipo de arquitectura empresarial emergente | Actualizado por:  Daniel Medina Arango | 1.0.0 |
| 2018/05/17 | Actualización de documento | Equipo de arquitectura empresarial emergente | Actualizado por:  Daniel Medina Arango  Se incluye información de versionamiento de APIs publicadas. | 1.0.1 |
| 2022/03/1 | Actualización de documento | Equipo de Gobierno | Se busca unificar el versionamiento y el ciclo de este en la capa de integración | 2.0.0 |

# Tabla de contenido

[Tabla de contenido 3](#_Toc125618538)

[1. Referencias 4](#_Toc125618539)

[2. Objetivo 4](#_Toc125618540)

[3. Alcance de versionamiento 4](#_Toc125618541)

[4. Premisas de Versionamiento 4](#_Toc125618542)

[5. Lineamientos de Versionamiento 4](#_Toc125618543)

[6. Restricciones de Versionamiento 5](#_Toc125618544)

[7. Definición de Versionamiento 5](#_Toc125618545)

[7.1. Definición de versionamiento para las API 6](#_Toc125618546)

[7.2. Definición de versionamiento para Servicios 6](#_Toc125618547)

[7.3. Estructuración de Versiones 7](#_Toc125618548)

[7.4. Proceso de Versionamiento 8](#_Toc125618549)

[8. Indicaciones para tener en cuenta 10](#_Toc125618550)

[9. Versionamiento en NameSpace de servicio y URL de APIs 12](#_Toc125618551)

[10. Procedimiento jubilación API´s 12](#_Toc125618552)

[11. Proceso retiro APIs en estado deprecated 12](#_Toc125618553)

[12. Glosario 14](#_Toc125618554)

[13. Referencias 15](#_Toc125618555)

# Referencias

|  |  |  |
| --- | --- | --- |
| **Referencias** | | |
| Nro | Nombre | **Descripción** |
| 1 | Estandar Nombramiento API On-Premises\_V1.0.0 | Documento de estándares de nombramiento para APIs que incluyen referencias al versionamiento |
|  |  |  |

# Objetivo

Establecer los lineamientos y estándares para la definición del versionamiento de APIs y Servicios de integración a ser registradas en el contexto del gobierno de la arquitectura orientada a servicios (SOA) para el grupo Bancolombia.

# Alcance de versionamiento

El alcance del presente documento es establecer una guía con lineamientos adecuados para el control de cambios en el tiempo de las APIs y de los servicios de la capa transaccional y empresarial a nivel de integración. Aplicado para los ambientes de desarrollo, certificación y producción.

# Premisas de Versionamiento

* Todas las APIs y productos, así como los servicios de integración de las capas empresarial y transaccional requieren gestionar su versionamiento.
* La estrategia de versionamiento abarca los ambientes de desarrollo, certificación, sandbox y producción.
* Los ciclos de vida las APIs y productos estarán ligados de manera directa.

# Lineamientos de Versionamiento

* Un API, producto o servicio de integración tendrá asignación de versión por ambiente de ejecución.
* La evolución del versionamiento será progresiva:
* Versión Anterior (1.0.0, 1.1.0, 1.1.1, etc.), versión Superior (2.0.0, 1.2.0, 1.1.2, etc.) respectivamente.
* El número de versiones de un producto a existir en cualquier ambiente será de máximo 2 versiones mayores (2 versiones activas, una de las cuales es versión activa en estado ‘deprecated’ en proceso retiro).
* En caso de llegar una nueva versión al ambiente y ya existan dos versiones conviviendo, la versión inferior pasara a estado de Jubilación y Retiro. Para lo cual se debe asegurar que todo consumidor haya sido correctamente informado del cambio para el consumo de las versiones nuevas y se realicen las gestiones necesarias para esta evolución.
* Una versión en estado ‘deprecated’ en proceso de retiro, no aceptará nuevos suscriptores y solo podrá ser consumida por los suscriptores actuales, mientras realizan la transición a la nueva versión.
* Una versión en estado de Jubilación y Retiro no aceptará registro de nuevos suscriptores ni podrá consumirse.
* Los consumidores de la versión marcada en estado de ‘deprecated’, en proceso de retiro, deberán realizar el proceso de migración a la versión superior (última versión existente). La migración deberá hacerse en el periodo de tiempo contemplando por la organización para este proceso.
* Desde el momento en que se marca el servicio o la API en estado ‘deprecated’ se debe generan los espacios correspondientes con los consumidores para realizarles la contextualización requerida y así asignar los tiempos de migración que en cualquier caso no deberá superar un año (1 año).

# Restricciones de Versionamiento

* La evolución de la versión no deberá ser retroactiva.
* La evolución de la versión superior deberá ser tomada de la versión inmediatamente inferior.
* Cuando una versión pasa a estado deprecado, esta se encuentra activa en el ambiente, pero NO acepta nuevos consumidores. Para los consumidores ya existentes el tiempo máximo que una versión estará disponible antes de su Jubilación y Retiro estará definido por un plazo de 1 año.
* La versión estará compuesta por tres dígitos: un digito seguido de un punto, un segundo dígito seguido de otro punto y un tercer dígito.

# Definición de Versionamiento

El ciclo de vida comprende una serie de etapas desde el momento de la identificación de la versión inicial, hasta su obsolescencia y retiro en la versión final, en donde la versión tiene cambios que pueden afectar su estructura.

Al actualizar la versión en ambiente de producción se puede afectar a los potenciales consumidores, debido a esto deben convivir diferentes versiones (máximo 2 [obsoleta, Superior]), dándoles tiempo a los consumidores para adaptarse a la nueva versión.

Los números de versión y la forma en que cambian entregan significado del código y lo que fue modificado de una versión a otra.

## Definición de versionamiento para las API

Las etapas para el versionamiento de las API’s son las que se listan a continuación:

* Creación
* Desarrollo
* Certificación
* Publicación en Producción
* Obsolescencia/Deprecación
* Jubilación y Retiro

Cuando una versión evoluciona es posible que la nueva versión conviva con la versión obsoleta, debido a la incompatibilidad retroactiva.

En la fase de desarrollo:

* Versión Mayor: En esta fase la versión mayor se establece en cero (‘0.Y.Z’), dado que en esta etapa está todo en cambio y no es una versión estable.

En la fase de publicación en producción:

* La versión se establece en ‘1.0.0’, definiendo el API pública. La forma en que el número de versión es incrementado después de esta publicación depende de esta API y de cómo esta cambie.
* Mayor: Inicia en 1.
* Menor: Inicia en 0.
* Parche: Inicia en 0.

## Definición de versionamiento para Servicios

Para el caso de los servicios se usará la definición de las siguientes etapas:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* Zona Construcción

Esta es la zona para las versiones que se encuentran en proceso de construcción, sea de manera lineal (una versión construida en un único ciclo) o iterativo (se requiere más de un ciclo para generar una versión que pueda estar lista para pasar a la zona productiva).

* Zona Productiva

Esta es la zona para las versiones productivas, las cuales tienen consumidores asociados.

Cuando una versión evoluciona es posible que la nueva versión conviva con la versión obsoleta, debido a la incompatibilidad retroactiva.

## Estructuración de Versiones

Las versiones pueden clasificarse según su impacto:

Imagen que contiene Diagrama

Descripción generada automáticamente

Donde X, Y, Z son enteros no negativos, donde X es la versión Mayor, Y la versión Menor, y Z la versión Parche.

|  |  |
| --- | --- |
| Versión | Descripción |
| Versión Mayor (X) | Introducen modificaciones importantes en el contrato del servicio o en la funcionalidad (Interna o Externa) del API o Producto por lo que rompen la compatibilidad de las dependencias, y que tiene impacto en los consumidores.  Ejemplo:  1.0.0, 2.0.0, 3.0.0 |
| Versión Menor (Y) | Introduce modificaciones que NO tienen impacto en los consumidores actuales del servicio.   Implementan adiciones de funcionalidades que no rompen la compatibilidad de las dependencias.  Ejemplo:  1.1.0, 1.2.0, 1.3.0 |
| Versión Parche (Z) | Cambios internos, los cuales corrigen comportamientos incorrectos en el servicio y NO tienen ningún impacto en los consumidores actuales del servicio.  Ejemplo: 1.0.0, 2.1.1. 3.2.2 |

Nota:

* Cada elemento debe incrementarse numéricamente en incrementos de 1 (una unidad).
* La primera versión siempre será 1.0.0

## Proceso de Versionamiento

Una vez que un paquete versionado ha sido liberado, los contenidos de esa versión no deben ser modificados. Cualquier modificación debe ser liberada como una nueva versión.

Se deben tener en cuenta las siguientes variables y criterios al momento de evaluar la asignación o cambio de versión:

* Variables
  + API, Producto, Servicio (Nuevo o Existente)
  + Métodos u Operaciones (Nuevo o Existente)
  + Campos (Obligatorios y Opcionales)

* Criterios Para Incremento Versión Mayor
  + Hay cambios en los contratos que afecta la compatibilidad con los consumidores.

En la siguiente tabla se relacionan algunos criterios que generan cambio de la versión mayor:

|  |
| --- |
| Todo componente nuevo con operaciones nuevas empieza en una versión mayor. |
| Cambio en el nombre de una operación del servicio. |
| Adición de un campo obligatorio al modelo de datos del mensaje de solicitud o respuesta. |
| Cambio en el nombre de un campo del modelo de datos del mensaje ya existente. |
| Modificar el tipo de un campo. |
| Cambiar una operación que va en un solo sentido a solicitud – respuesta |
| Eliminar un campo del modelo de datos del mensaje para una operación. |
| Cambiar un campo de opcional a obligatorio. |
| Quitar una operación al servicio o un método de la API. |
| Cambio en el NameSpace o en el BasePath. |
| Cambio en el nombre o tipo de un método del API. |

Imagen que contiene Texto

Descripción generada automáticamente

|  |  |
| --- | --- |
| Ejemplos cambio versión mayor | |
| Actual | Nueva |
| 1.0.0 | 2.0.0 |
| 2.1.0 | 3.0.0 |
| 2.2.0 | 3.0.0 |
| 3.5.0 | 4.0.0 |
| 4.2.0 | 5.0.0 |

* Criterios Para Incremento Versión Menor
  + Cambios en el contrato que NO afectan la compatibilidad con los consumidores actuales.
  + La versión menor debe ser ajustada a cero (0) cada vez que la versión mayor sea incrementada.

En la siguiente tabla se relacionan algunos criterios que generan cambio de la versión menor:

|  |
| --- |
| Adicionar una capacidad (método u operación) al servicio o API. |
| Adición de un campo opcional al modelo de datos del mensaje de solicitud o respuesta. |
| Cambiar un campo de obligatorio a opcional. |

A continuación, ejemplos de cómo realizar el cambio de versión menor:

|  |  |
| --- | --- |
| Ejemplos cambio versión menor | |
| Actual | Nueva |
| 1.0.0 | 1.1.0 |
| 2.1.0 | 2.2.0 |
| 2.2.1 | 2.3.0 |
| 3.5.0 | 3.6.0 |
| 4.2.2 | 4.3.0 |

* Criterios Para Incremento Versión Parche
  + Cambios internos, que corrigen comportamientos incorrectos y NO afectan la compatibilidad con los consumidores actuales.
  + La versión PARCHE debe ser ajustada a cero (0) cada vez que la versión mayor o la menor sea incrementada.

|  |
| --- |
| Ajuste al comportamiento incorrecto del servicio, el cual no implica ningún cambio a nivel de contracto. |
|  |

A continuación, ejemplos de cómo realizar el cambio de versión menor:

|  |  |
| --- | --- |
| Ejemplos cambio versión parche | |
| Actual | Nueva |
| 1.0.0 | 1.0.1 |
| 2.3.0 | 2.3.1 |
| 3.2.1 | 3.2.2 |

# Indicaciones para tener en cuenta

Las principales actividades de la Gestión de Versionamiento son:

* + Planificación: planificar el lanzamiento de una nueva versión e identificar los impactos en la implementación de nuevas versiones.
  + Desarrollo: diseñar y construir la nueva api o servicio, componente o entidad.
  + Validación: probar la nueva versión en un entorno que simule el entorno de producción, ambientes preproductivos.
  + Implementación: implementar la nueva versión en el entorno de producción. Queda activa para su uso en producción y acepta nuevos consumidores según capacidad.
  + Jubilación: Cuando nuevas versiones son implementadas, deben deprecarse las anteriores y llevar a cabo los planes de jubilación y retiro de la nueva versión si fuera necesario.
  + Comunicación: comunicar y formar a los clientes y usuarios sobre las funcionalidades de la nueva versión.

Los beneficios de una adecuada gestión de versionamiento se ven reflejados en:

* El proceso de actualización sin deterioro de la calidad de la api, componente, entidad o servicio.
* Las nuevas versiones cumplen los objetivos propuestos.
* Se reduce el número de incidentes por incompatibilidades.
* El proceso de pruebas asociado permite asegurar la calidad del servicio, api, componente o entidad a instalar si no como también la funcionalidad y usabilidad de las nuevas versiones.
* Control centralizado del servicio, api, componente o entidad desplegada.

Las principales dificultades en la gestión de versionamiento son:

* Asignación de responsabilidades en el proceso de implementación del cambio.
* Entorno de pruebas de las nuevas versiones.
* Resistencia a la centralización del proceso de cambio y estandarización del api.
* Cambios de urgencia sin gestión de versionamiento.
* Impactos de nuevas versiones en otras áreas.
* Implementación sincronizada de versiones en ambientes distribuidos
* Desconocimiento de los responsables técnicos y funcionales dueños de los consumos de las capacidades.

Los compromisos en la gestión de versionamiento son:

* Compromiso de la organización en la gestión de versiones.
* Plan de comunicaciones de las ventajas del proceso de cambio.
* Construcción lineamientos en el proceso de versionamiento.

Actualización de versión:

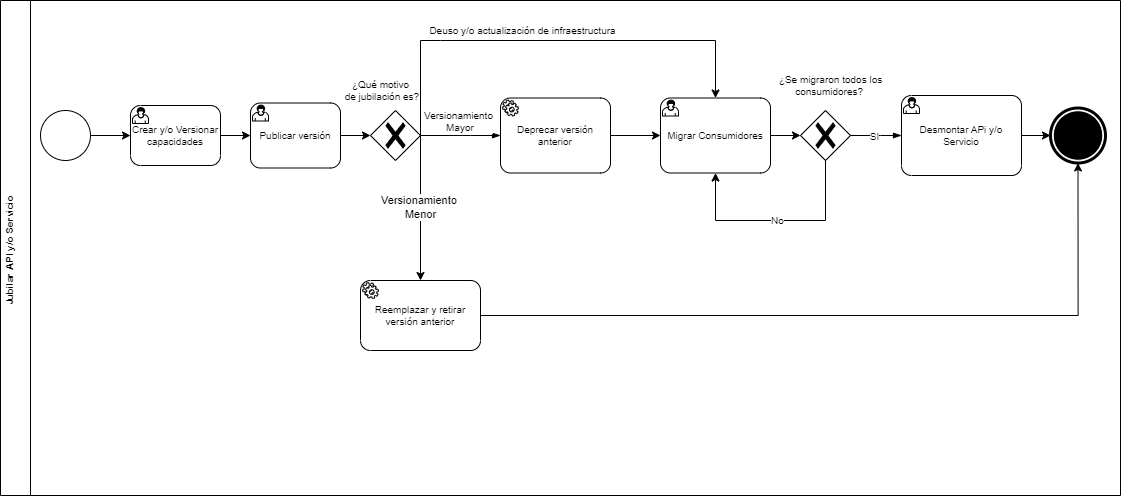
* Parche(Z): (X.Y.Z, cuando X > 0), debe incrementarse cuando se introducen arreglos compatibles con la versión anterior.
* Menor(Y): (X.Y.Z, cuando X > 0), debe ser incrementada si se introduce nueva funcionalidad compatible con la versión anterior. Debe incrementar si cualquier funcionalidad de la API o servicio es marcada como obsoleta. Puede ser incrementada si se agrega funcionalidad o arreglos considerables al código privado. La versión parche(Z) debe ser reseteada a 0 cuando la versión Menor es incrementada.
* Mayor(X): (X.Y.Z, cuando X > 0), debe ser incrementada si cualquier cambio no compatible con la versión anterior es introducido a la API o servicio. Las versiones Parche (Z) y Menor (Y) deben ser reiniciadas a 0 cuando se incrementan la versión Mayor(X).

# Versionamiento en NameSpace de servicio y URL de APIs

Debido a la relación existente entre la versión del servicio y la versión de servicio registrada en el nombre del namespace, así como en el caso de las APIs la relación de la versión y la que se establece en la URL, se establece los siguiente:

Con el fin de garantizar que los cambios de versión Menor o Parche no afecten a los consumidores la versión registrada en el namespace y en la URL solo debe hacer referencia a la versión Mayor.

# Procedimiento jubilación API´s



# Proceso retiro APIs en estado deprecated

**11.1 Crear y/o Versionar Capacidades:**

Esta actividad inicial corresponde a las definiciones realizadas en las etapas anteriores del modelo operativo de APIs y Servicios en las que el equipo estratégico y con base a los análisis realizados en el modelamiento SOA, evalúa el versionamiento de un API o Servicio según: la necesidad del negocio, resolución de incidencias o demás escenarios que involucre en su solución modificar el API y/o Servicio, o se pretenda crear una capacidad nueva que comprenda las funcionalidades de capacidades en desuso.

El analista SOA debe ser el encargado de realizar el análisis del versionamiento si corresponde a un versionamiento mayor, menor o parche de acuerdo a lo indicado en el presente lineamiento y demás responsabilidades asociadas a su rol para el cumplimiento del diseño en el versionamiento del API y/o Servicio.

**11.2 Publicar Versión:**

Esta actividad corresponde a la etapa de planear y crear, en donde el equipo táctico o de la solución es el encargado de desplegar las versiones de las capaciades o la creación de una nueva capacidad que comprenda las funcionalidades de otras capacidades en desuso.

Se deben realizar cada una de las tareas mencionadas en la etapa de planear y crear alineadas con el proceso de Habilitar y mantener soluciones en su actividad de implementar la solución.

**11.3 ¿Qué motivo de jubilación es?**

Para responder a este interrogante, se debe partir de la definición del versionamiento que haya realizado el Analista SOA y que se haya desplegado en los diferentes ambientes (Sandbox, Desarrollo, Certificación y Producción) para lo cual si es:

* Versionamiento menor
* Versionamiento mayor
* Desuso y/o actualización de infraestructura

Si es por versionamiento menor:

**11.4 Reemplazar y retirar versión anterior:**

Por ser un versionamiento menor, que no impacta el consumo de las APIS ni a sus consumidores, API connect identifica automáticamente la versión de acuerdo al nombramiento y realiza un proceso de reemplazo y retiro, dejando en API connect la versión menor Superior en la herramienta y retirando la Anterior, además de ello API connect migra los consumidores de manera autónoma.

Si es por versionamiento mayor:

**11.5 Deprecar API y/o Servicio:**

Una vez se publique la versión mayor Superior (2.0.0), la versión menor anterior (1.0.0) queda automáticamente en estado deprecado, no acepta nuevos consumidores y queda desvinculada del portal de APIS o Catalogo de Servicios, por lo que se debe realizar un plan de migración de consumidores a la API o servicio de la versión mayor Superior (2.0.0) para poder retirar la versión mayor anterior (1.0.0) de la plataforma sin impactar a los consumidores.

**11.6 Migrar consumidores:**

La migración de consumidores debe estar a cargo del Dueño técnico del producto, teniendo en cuenta el procedimiento de reconexión y reúso, teniendo como base el inventario de reconexión realizado por el Analista SOA en las fases anteriores a la públicación del API o Servicio.

Esta actividad se debe realizar para los escenarios de versionamiento mayor, o en los que se presente la creación de una nueva capacidad que por sus definiciones contempla las funcionalidades de capacidades en desuso o que van para retiro.

Para la actualización de la infraestructura es necesario realizar esta migración de consumidores y realizar procedimientos asociados al ejercicio (APICV 5 a APICV10) o (BUS 10 a ACE 11)

Nota: Es importante que se migren en su totalidad los consumidores y dejar sin consumos a la versión mayor anterior 1.0.0 para su posterior retiro de las plataformas.

**11.7 Desmontar APIS y/o Servicios:**

Esta actividad se centra en retirar las APIS y/o Servicios de API connect y el Catálogo de Servicios, para lo cual se debe tener presente lo siguiente:

* Se debe realizar un comunicado informando el retiro de las plataformas de las capaciades a sus consumidores y/o componentes que tienen impacto sobre las mismas.
* Se debe realizar una gestión del desmonte del monitoreo y soporte con los Analistas de servicio encargados de la EVC de integración o de las EVC de dónde corresponda la capacidad. Para ello deben contactar al AS.

Debe ser responsabilidad del ingeniero de software realizar el desmonte del API y/o Servicio, apoyado del Arquitecto de la solución en la identificación de componentes y otros impactos para la comunicación efectiva del retiro de las capaciades y contemplando a los consumidores en este comunicado.

# Glosario

Se relacionan términos, siglas y abreviaciones utilizadas en el documento con su correspondiente definición.

|  |  |
| --- | --- |
| Lista de términos | |
| Termino | Definición |
| Metodología ITIL | Information Technology Infrastructure Library |
| Control de Versión | Gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación. |
| API | (Application Programming Interface) Conjunto de funciones, procedimientos y políticas que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. |
| Producto | Definición que encapsula un conjunto de APIs. Configura una serie de planes con condiciones para la utilización de las APIs. |

# Referencias

* [ITIL - Gestión de Versiones Servicios TI](http://itil.osiatis.es/Curso_ITIL/index.php)
* [Versionamiento semántico](http://semver.org/)