Bundesverwaltungsamt

IsyFact-Standard

IsyFact – Versionierung

Version 0.2
17.10.2016

IsyFact Standards

Ansprechpartner:

Referat Z II 2
Bundesverwaltungsamt
E-Mail: isyfact@bva.bund.de
Internet: www.isyfact.de

Dokumentinformationen

Dokumenten-ID:	IsyFact-Versionierung.docx

Inhaltsverzeichnis

1. Einleitung

Versionen von Bibliotheken, Schnittstellen und Anwendungen dienen zur eindeutigen Identifizierung von Software-Artefakten mit einem bestimmten Inhalt. Dieses Konzept trifft Vorgaben und Empfehlungen zum Aufbau der Versionsnummern, um eine bessere Einheitlichkeit und Vergleichbarkeit zu gewährleisten.

2. Allgemeine Nomenklatur und Vorgaben

Versionen müssen nach dem folgenden Format aufgebaut sein (Inhalte in eckigen Klammern sind optional):

MAJOR.MINOR.PATCH[-LABEL][+BUILDMETADATA]

Die meisten der hier genannten Vorgaben sind unverändert aus „Semantic Versioning 2.0“ [SemanticVersioning20] übernommen, einige wurden leicht verändert.

Die Terme “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, und “OPTIONAL” in diesem Dokument sind, wie in RFC 2119 beschrieben, zu interpretieren.

Unter einer API wird hier alle zur öffentlichen Schnittstelle einer Bibliothek gehörenden Klassen und Konfigurationsparameter verstanden.

Bei einem finalen Tag (Veröffentlichung) besteht die Versionsnummer nur aus MAJOR.MINOR.PATCH.

1. Eine Versionsnummer einer Veröffentlichung muss (MUST) dem Format X.Y.Z entsprechen, wobei X, Y und Z Ganzzahlen größer oder gleich Null sind und eine Zahl größer als Null keine führenden Nullen enthalten darf. X ist die Major Version, Y ist die Minor Version und Z ist die Patch Version. Jedes Element muss (MUST) auf numerische Art und Weise erhöht werden. Zum Beispiel: 1.9.0 → 1.10.0 → 1.11.0.

2. Sobald eine Version eines Projektes veröffentlicht wurde, darf (MUST NOT) der Inhalt dieser Version nicht mehr verändert werden. Eine Änderung am Inhalt muss (MUST) als eine neue Version veröffentlicht werden.

3. Versionsnummern mit einer **Major** Version von 0 (0.y.z) sind für die initiale Development Phase gedacht. Änderungen können in jeder denkbaren Form und zu jeder Zeit auftreten. Eine eventuell vorhandene öffentliche API sollte nicht als *stabil* betrachtet werden.

4. Die Version 1.0.0 definiert die erste Veröffentlichung. Ab dieser Veröffentlichung hängt die Art und Weise, wie die Versionsnummer erhöht und verändert wird, von der öffentlichen API und den Änderungen, die an ihr vollzogen werden, ab.

5. Eine Vorveröffentlichung kann (MAY) durch ein **LABEL** gekennzeichnet werden, indem ein Bindestrich, gefolgt von dem Vorveröffentlichungs-Bezeichner, dessen Elemente durch Punkte voneinander getrennt werden, an die Patch Version angehängt wird. Die Elemente des Bezeichners dürfen (MUST) nur aus alphanumerischen ASCII Zeichen und dem Bindestrich ([0-9A-Za-z-]) bestehen. Sie dürfen (MUST NOT) außerdem nicht leer sein. Wenn ein

Element ausschließlich aus Ziffern besteht, darf (MUST NOT) es keine führenden Nullen enthalten. Eine Vorveröffentlichungs-Version hat einen niedrigeren Rang als die entsprechende reguläre Version. Ein Vorveröffentlichungs-Bezeichner kennzeichnet, dass die Version als *unstable* zu betrachten ist und dass sie unter Umständen nicht den Kompatibilitätsanforderungen, die für die entsprechende reguläre Version bestimmt wurden, entspricht. Beispiele: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.

6. Build-Metadaten können (MAY) ausgezeichnet werden, indem ein Plus Symbol, gefolgt von den Metadaten, deren Elemente durch Punkte voneinander getrennt werden, an die Patch Version oder den Vorveröffentlichungs-Bezeichner angehängt wird. Die Elemente der Metadaten dürfen (MUST) nur aus alphanumerischen ASCII Zeichen und dem Bindestrich ([0-9A-Za-z-]) bestehen. Sie dürfen (MUST NOT) außerdem nicht leer sein. Die Build-Metadaten haben keinerlei Einfluss auf den Rang einer Version, sodass zwei Versionen, deren Versionsnummern sich nur in den Build-Metadaten unterscheiden, denselben Rang einnehmen. Wenn eine Buildnummer angegeben wird, muss ein Verweis auf den Buildserver mitangegeben werden. Beispiele: 1.0.0-alpha+001.B1, 1.0.0+20130313144700, 1.0.0-beta+exp.sha.5114f85.

Die genaue semantische Bedeutung der Elemente und die Bedeutung der Erhöhung einer Stelle unterscheiden sich aber bei Bibliotheken, Anwendungen und Schnittstellen voneinander. Daher zeigen die kommenden Abschnitte die genauen Vorgaben pro Typ auf und liefern Beispiele.

3. Versionierung von Anwendungen

Versionen von Anwendungssystemen folgen dem folgenden Muster:

MAJOR.MINOR.PATCH[-LABEL]

Zusätzlich zu den im Kapitel 2.1 getätigten Vorgaben und Empfehlungen gelten hier die folgenden Konkretisierungen:

Auf Grundlage einer Versionsnummer von MAJOR.MINOR.PATCH werden bei Anwendungssystemen nach IsyFact die einzelnen Elemente folgendermaßen erhöht:

1. **MAJOR** wird nach Absprache erhöht, wenn signifikante Änderungen an der Anwendung veröffentlicht werden,

2. **MINOR** wird erhöht, wenn neue Funktionalitäten veröffentlicht werden, und

3. **PATCH** wird erhöht, wenn es sich um reine Bugfixes handelt oder eine irreguläre Auslieferung (bspw. zur Fehlerbehebung) ist.

Beispiele: 1.0.2, 2.5.1, 1.2.1, 0.10.0-alpha1

3.1. Versionierung im SCM

Die Benennung von Branches in einem SCM wie SVN oder Git hängt im Wesentlichen vom gewählten Workflow ab. Die Version eines finalen Tags einer Anwendung muss dem oben genannten Format folgen. Finale Versionen dürfen kein Label enthalten.

Wenn für eine Anwendung Entwicklungs- oder Integrationstags erstellt werden, beispielsweise für Continuous Delivery, müssen diese zusätzlich mit Buildmetadaten versehen werden, um möglichst genau feststellen zu können, welches Binärartefakt dem Tag zuzuordnen ist. Sie enthalten immer (MUST) einen alphanumerischen Identifier für den Buildserver und die Buildnummer.

Diese sollten daher dem folgenden Schema folgen:

MAJOR.MINOR.PATCH[-LABEL][+BUILDMETADATA]

Beispiele: 1.0.2+CG.101, 2.5.1+B1.277, 1.2.1+CG.54

3.2. Versionierung von Containern

Zusätzlich zur Version der Anwendung muss der Container oder die Installationsdatei mit Buildmetadaten versehen werden, um möglichst genau feststellen zu können, welcher Entwicklungsstand enthalten ist.

3.2.1 RPM

Bei RPM-Paketen muss die Versionsnummer des RPMs folgendes Format aufweisen:

MAJOR.MINOR.PATCH[-LABEL][+BUILDMETADATA]-RPMNR

dies entspricht auch

TAGNAME-RPMNR

Beispiele: 1.0.2+101.CG-1, 2.5.1+277.B1-1, 1.2.1+54.CG.sha.5114f85-1, 1.0.2-1

4. Versionierung von Bibliotheken

Versionsangaben von Bibliotheken folgen dem folgenden Muster:

MAJOR.MINOR.PATCH[-LABEL]

Zusätzlich zu den im Kapitel 2 getätigten Vorgaben und Empfehlungen gelten hier die folgenden Konkretisierungen.

Wenn von kompatibel gesprochen wird, ist damit gemeint, dass ein Wechsel auf die kompatible Version der Bibliothek keine Anpassungen an dem Code der nutzenden Anwendung/Bibliothek erfordert, und die bisherige Funktionalität uneingeschränkt erhalten bleibt.

Auf Grundlage einer Versionsnummer von MAJOR.MINOR.PATCH werden die einzelnen Elemente folgendermaßen erhöht:

1. **MAJOR** wird erhöht, wenn API-inkompatible Änderungen veröffentlicht werden,

2. **MINOR** wird erhöht, wenn neue Funktionalitäten, welche kompatibel zur bisherigen API sind, veröffentlicht werden, und

3. **PATCH** wird erhöht, wenn die Änderungen ausschließlich API-kompatible Bugfixes umfassen.

Das bedeutet, dass die Regeln des Semantic Versionierung 2.0 anzuwenden sind.

Zusätzlich gilt:

- Instabile Entwicklungsversionen müssen (MUST) mit dem Label -SNAPSHOT oder einem anderen Label gekennzeichnet werden. Instabile Entwicklungsversionen sollten nicht über einen längeren Zeitraum in einer Anwendung eingebunden sein, da die Gefahr besteht, dass der Build instabil wird.

Beispiele: 1.0.0, 2.3.5-SNAPSHOT, 1.3.2-alpha

5. Versionierung von Schnittstellen

Versionsangaben von Bibliotheken folgen dem folgenden Muster:

MAJOR.MINOR[-LABEL]

Zusätzlich zu den im Kapitel 2 getätigten Vorgaben und Empfehlungen gelten hier die folgenden Konkretisierungen und Abweichungen.

Bei Schnittstellen wird auf den Bugfix-Teil der Version verzichtet, da Schnittstellen keine Bugfixes im generellen Sinne enthalten können. Änderungen in einer Schnittstelle sind immer entweder API-kompatibel oder API-inkompatibel. Daher werden bei Schnittstellen nur die MAJOR und MINOR Elemente der Version genutzt.

Auf Grundlage einer Versionsnummer von MAJOR.MINOR werden die einzelnen Elemente folgendermaßen erhöht:

4. **MAJOR** wird erhöht, wenn API-inkompatible Änderungen veröffentlicht werden oder eine Schnittstelle parallel zu einer alten Schnittstellenversion angeboten werden soll,

5. **MINOR** wird erhöht, wenn ausschließlich API-kompatible Änderungen veröffentlicht werden.

Zusätzlich gilt:

- Instabile Entwicklungsversionen müssen (MUST) mit dem **Label** -SNAPSHOT oder einem anderen Label gekennzeichnet werden. Instabile Entwicklungsversionen sollten nicht über einen längeren Zeitraum in einer Anwendung eingebunden sein, da die Gefahr besteht, dass der Build instabil wird.
- Die **MAJOR** Version darf nicht (MUST NOT) Teil der Maven Versionsnummer sein, sondern muss Teil der Artefakt-ID sein. Zusätzlich muss MAJOR Teil des Packagepfades sein. Dies ermöglicht das parallele Einbinden mehrere Versionen ein- und derselben Schnittstelle.

Beispiele: 1.0, 2.33-SNAPSHOT, 1.3-alpha

6. Quellenverzeichnis

[SemanticVersioning20]

SemanticVersioning2.0.0
http://semver.org/spec/v2.0.0.html.