Procédure de Gestion des Versions

Informations qualité

Suivi des modifications majeures	5 mars 2015 - Nicolas Marchand - Création du document 1 décembre 2015 - Nicolas Marchand - Modification par rapport aux tests à passer suivant le niveau des mises à jour 14 décembre 2015 - Guillaume Natali - Modification des règles de numérotation 02 aout 2017 - Nicolas Marchand - Portage sur DoKuWiKi & Evolutions 28 août 2020 - Nicolas Marchand - Evolution raisons changement de version Janvier 2026 - Nicolas Marchand - Refonte suite à l'automatisation (Galaxie LoGeAs, version globale, schéma du processus de mise en ligne) 11 août 2026 - Nicolas Marchand - Fusion de toutes les pages relatives à la gestion de version en un seul document, dont le doublon `questionnaire:procedure:develop:gestionversion` (voir « Documents fusionnés » en bas de page) ; nouveau mécanisme de signalement des versions beta (4e chiffre porteur de la version visée, remplace le signal Y=99) ; alignement du client lourd et des serveurs sur cette même règle
Suivi des approbations	Ce document correspond à l'élément ProjeQtor Document #06 - Gestion des versions de LoGeAs (PROC-NFlog-5), fusionné le 11 août 2026 avec les procédures #30, #09, « Gestion du dépôt Git » et le doublon `questionnaire:procedure:develop:gestionversion`
Objet	Définir les types de versions mises à disposition, leurs différences, les actions à faire lors de la sortie d'une nouvelle version, les modalités d'intégration du code dans le dépôt Git, et le dépôt légal des codes sources — pour l'ensemble des composants de la Galaxie LoGeAs.
Destinataires	**- Validation des modifications :** Gérant **- Approbation du document** : Equipe dev & Equipe Ass

Objectif du Versionnage

Le versionnage constitue le mécanisme qui consiste à conserver la version d'une entité logicielle quelconque, de façon à pouvoir la retrouver facilement.

Système de Numérotation

Une version LoGeAs représente un état donné d'un produit fourni aux clients. Le numéro comporte 3 chiffres pour une version publiée, 4 pendant sa phase de test beta — ex. `11.0.3.1`.

Segment	Nom	Description
1er	Majeur (X)	Numéro « marketing » et technique ; migration technologique majeure
2e	Mineur (Y)	Nouvelles fonctionnalités ou changements de structure de base de données, changement de réglementation, nouvelle certification
3e	Release (Z)	Change à chaque publication du logiciel (obligatoire) ; ne réutilise jamais un numéro déjà publié

Segment	Nom	Description
4e	Build (W)	Uniquement pendant une phase beta : numéro d'itération de la beta en cours de test. N'apparaît jamais sur une version publiée

Règles fondamentales

1. **Immuabilité** : une version publiée ne peut jamais être modifiée ; toute correction, même mineure, nécessite une nouvelle version.
2. **Cascade** : quand le Majeur change, Mineur/Release/Build reviennent à 0. Quand le Mineur change, Release/Build reviennent à 0.

Signalement d'une version beta (X.Y.Z.W)

Une version en cours de test beta porte directement le numéro **X.Y.Z que cette beta vise à publier**, suivi d'un 4e chiffre **W** qui numérote les itérations successives de cette beta (1, 2, 3…), remis à 1 à chaque nouveau cycle.

Exemple concret :

- Dernière version publiée : `11.0.25`
- Un correctif est en préparation, sans nouvelle fonctionnalité → il vise `11.0.26`. Sa 1re beta est numérotée `11.0.26.1`, un correctif suite à retour de test donne `11.0.26.2`, etc.
- Une fois validée, la beta `11.0.26.2` devient la release `11.0.26` (le 4e chiffre est retiré, le X.Y.Z ne change pas — cohérent avec la règle de cascade : seul un changement de Majeur ou Mineur remet Release/Build à 0).
- Une évolution plus importante (nouvelles fonctionnalités) viserait `11.1.0` au lieu de `11.0.26` (cascade : Mineur incrémenté, Release repart à 0) — ses betas seraient `11.1.0.1`, `11.1.0.2`, etc.

Pourquoi ce mécanisme plutôt que l'ancien signal Y=99 (utilisé auparavant, y compris dans le code du client lourd) : avec un flag générique, deux cycles beta consécutifs qui ne changent pas le mineur — le cas le plus fréquent — produisent le même numéro de beta, sans moyen de savoir à quelle version future chacun correspond. En faisant porter directement le numéro visé, l'ambiguïté disparaît et la progression reste strictement croissante, sans trou ni collision possible.

Critères de Changement de Version

Majeur (X) : évolutions substantielles ou migrations technologiques (ex. changement de base de données).

Mineur (Y) : nouvelles fonctionnalités, modifications de structure de données (ajout de champs), changements de réglementation pris en compte, ou nouvelle certification.

Release (Z) : à chaque publication du logiciel (publication obligatoire) — correctifs, corrections de bugs, ajustements sans changement fonctionnel structurant.

Build (W) : uniquement pendant une phase beta, itération de test au sein d'un même cycle.

La Galaxie LoGeAs

Le système LoGeAs s'appuie sur plusieurs composants, chacun avec un versioning indépendant, mais suivant tous la même règle de numérotation ci-dessus.

Serveurs

- **LogeasWeb** — exécutable Windows (Delphi) donnant accès aux bases clients
- **PGI** — serveur gérant la base de données utilisateurs (Delphi)
- **Nono** — serveur annexe gérant les bases anonymisées (Lazarus)

Interfaces

- **LoGeAs (client lourd)** — interface historique, en cours de portage vers Angular (Delphi)
- **LoGeAs.fr (client full-web)** — interface utilisateur remplaçante, **c'est le présent dépôt `logeas-web`** (Angular)
- **Test.LoGeAs.fr** — interface de test (Angular)
- **Assistance** — interface pour l'équipe support (Angular)
- **Cartographie fonctionnelle** — interface de gestion qualité/certifications NF (Angular)
- **Stat-Union** — interface clients EPUdF pour consolidation comptable

Principes de gestion

- Chaque sous-ensemble possède son propre numéro de version et son propre rythme de publication.
- Les chiffres Majeur et Mineur sont, en pratique, communs à la plupart des composants (alignés historiquement sur la même ligne 11.x), à l'exception notable du serveur PGI qui suit sa propre ligne (9.x).
- Git gère les numéros de version de chaque composant indépendamment ; aucune synchronisation automatique n'est imposée entre dépôts.

Version Globale

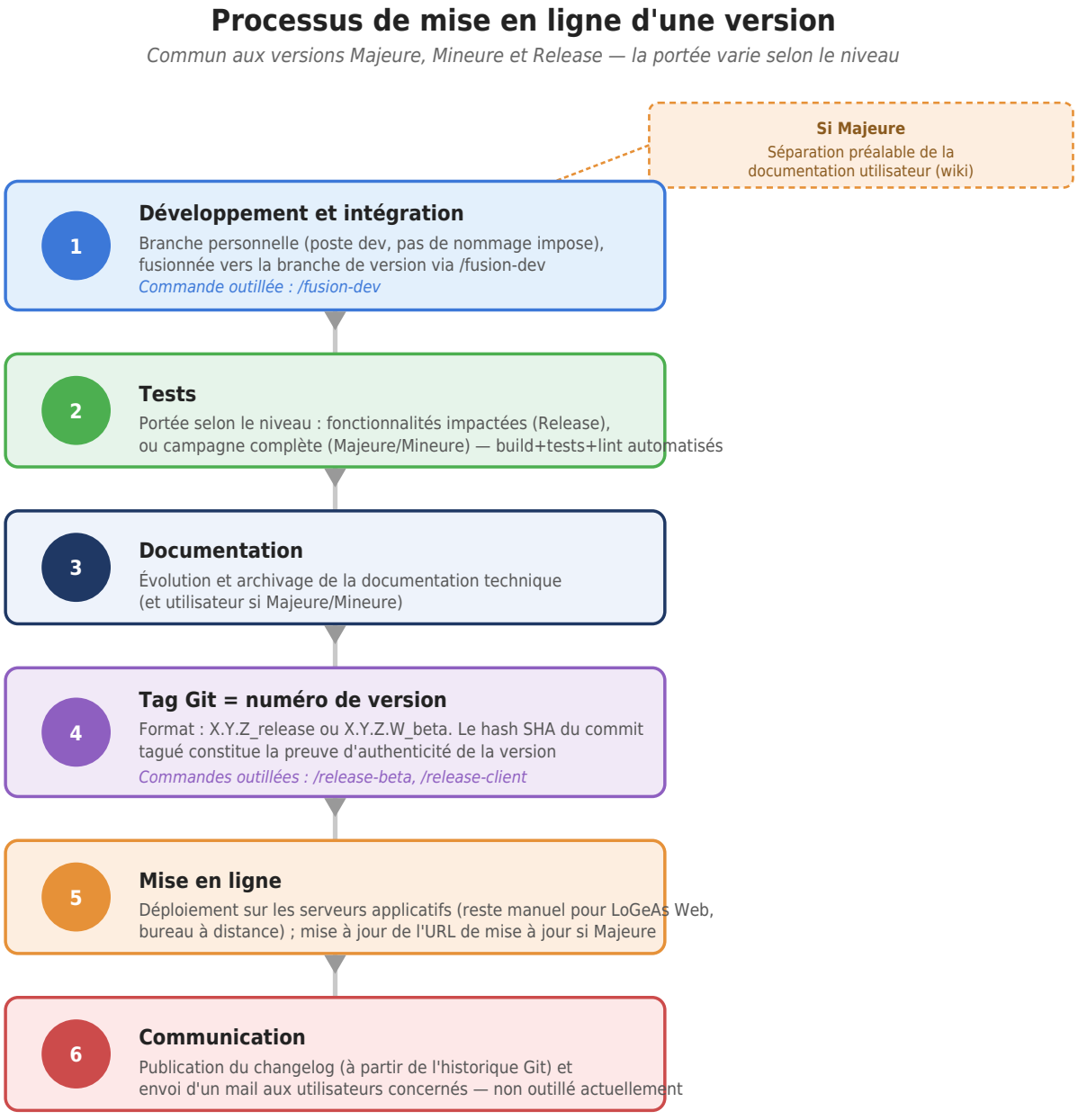
Une version « globale » synthétise l'état général du logiciel à des fins de qualité et d'assistance :

- **Majeur** : aligné sur le numéro Majeur le plus élevé parmi les sous-composants ; s'il est incrémenté, les autres chiffres reviennent à zéro.
- **Mineur** : aligné sur le numéro Mineur le plus élevé parmi les sous-composants ; s'il est incrémenté, les sous-chiffres reviennent à zéro.
- **Release** : incrémenté à chaque modification d'un sous-composant, quel qu'il soit.
- **Build** : non utilisé actuellement.

Note pratique : en l'état actuel de la Galaxie (voir matrice ci-dessous), LoGeAs Web est le

composant le plus avancé et le plus fréquemment publié — la version globale calculée suit donc, dans les faits, de très près la version de LoGeAs Web, qui sert de repère de référence au quotidien pour l'équipe.

Processus de mise en ligne d'une version



Le processus est commun aux 3 niveaux de version (Majeure, Mineure, Release) — seules la portée des tests et de la documentation varient selon le niveau :

1. **Développement et intégration** : travail sur branche personnelle (poste développeur, pas de

nommage imposé), intégrée vers la branche de version (nommée Majeur.Mineur.Release) via `/fusion-dev` — voir section « Intégration du code » ci-dessous.
2. **Tests** : portée selon le niveau (fonctionnalités impactées pour une Release, campagne complète pour une Majeure/Mineure). Compilation, tests automatiques et style vérifiés automatiquement par `/fusion-dev` à l'intégration, puis à nouveau lors de la génération beta/release.
3. **Documentation** : évolution et archivage de la documentation technique (et utilisateur si Majeure/Mineure).
4. **Tag Git = numéro de version** : l'empreinte (hash SHA) du commit tagué constitue la preuve d'authenticité de la version. Posé automatiquement par `/release-beta` ou `/release-client` — voir « Gestion des tags Git » ci-dessous.
5. **Mise en ligne** : déploiement sur les serveurs applicatifs, mise à jour de l'URL de mise à jour si Majeure. **Reste manuel** pour LoGeAs Web (copie des dossiers `LogeasWeb`/`LogeasWeb-Test` vers le serveur, accessible uniquement en bureau à distance).
6. **Communication** : publication du changelog (à partir de l'historique Git) et envoi d'un mail aux utilisateurs concernés. **Non outillé actuellement** côté LoGeAs Web.

Une fois publiée, une version n'est jamais modifiée : toute évolution repart sur une nouvelle branche / un nouveau numéro de version.

Intégration du code (dépôt Git)

Cette section fusionne l'ancienne procédure « gestion du dépôt Git » (Kevin Kopinski, 10/11/2025). Sa version originale décrivait un flux par ticket GitLab et relecture systématique par un second développeur — ce flux n'a jamais correspondu à la pratique réelle et n'est plus d'actualité (confirmé en 2026-08, suite au départ de Kevin Kopinski) : pas de ticket, pas de merge request, travail réalisé directement sur le poste du développeur.

1. **Travail sur branche personnelle** : chacun développe sur sa propre branche, créée depuis la branche de version en cours (ex. `11.0.25`). Pas de ticket GitLab ni de merge request requis.
2. **Sauvegarde régulière** : commit et push de la branche personnelle au fil de l'eau, sans vérification imposée à ce stade (procédure manuelle).
3. **Intégration = validation** : la validation d'une fonction se fait au moment de son intégration dans la branche de version courante, via la procédure outillée `/fusion-dev` (assistant Claude Code, dépôt logeas-web), qui :
    - met à jour la branche de version depuis le dépôt distant
    - fusionne la branche personnelle avec un commit de fusion portant le log complet des commits intégrés
    - vérifie compilation, tests automatiques et style après fusion — c'est ce contrôle technique qui fait office de validation, il n'y a pas de relecture séparée par un second développeur
    - publie le résultat vers le dépôt distant

Une version rassemble l'ensemble des changements intégrés sur la branche de version depuis la dernière publication.

Gestion des tags Git

Format des tags

- Beta : `X.Y.Z.W_beta` (ex. `11.0.26.1_beta`)
- Release : `X.Y.Z_release` (ex. `11.0.26_release`)

Ce format s'applique au dépôt LoGeAs Web (`logeas-web`). Les dépôts du client lourd et des serveurs (groupe Delphi) utilisent une convention de tag distincte, héritée de leur propre historique (ex. `NonoVersion11.0.1.0`, `LoGeAsVersion11.0.2.3Pré-release`) — non harmonisée avec le format ci-dessus pour l'instant.

Dans tous les cas, l'empreinte (hash SHA) du commit tagué constitue la preuve d'authenticité de la version, et peut être communiquée directement aux organismes certificateurs sans dépôt auprès de tiers.

Étapes d'exécution — LoGeAs Web

Outillées via les commandes `/release-beta` et `/release-client` (assistant Claude Code, dépôt logeas-web) :

1. **Vérification de branche** : la branche de version est propre et à jour avec le dépôt distant
2. **Génération du build** : compilation, incrémentation automatique du numéro (4e chiffre pour une beta, 3e chiffre pour une release)
3. **Commit dédié** : le changement de numéro de version est commité séparément du contenu fonctionnel
4. **Étiquetage** : tag Git annoté créé sur ce commit
5. **Publication** : push du commit et du tag vers le dépôt distant
6. **Relevé d'empreinte** : le hash de commit court est embarqué automatiquement dans `src/assets/version.ts`, consultable directement dans le logiciel livré
7. **Archivage** : conservation de la référence du tag dans le suivi de version (cartographie fonctionnelle)

Cas d'un tag posé manuellement

Respecter impérativement le format défini ci-dessus et ne jamais réutiliser un numéro X.Y.Z déjà présent dans un tag `_release` existant.

Dépôt légal des codes sources (coffre-fort numérique)

Cette section fusionne l'ancienne procédure #09 « Dépôt des Codes Sources ». Le dépôt légal intervient **après** la pose du tag `_release` (donc après exécution de `/release-client`), à chaque publication d'une version.

Les codes sont historisés via GitLab, répartis en plusieurs groupes :

- **Logeas Informatique / Divers** : AutoVideo, Composants, SFTPLogeasSync
- **Logeas Informatique / Logeas-projets-obsoletes** : BibliothequeAngular, logeas-lib, maj-serveurapp, NewInterfaces
- **Logeas Informatique / Logeas-web** : Logeas-projet-IA, logeas-web.fr, Site internet
- **NewInterfaces-group** : exemples, logeas-assistance, logeas-cartographie, logeas-lib, logeas-statunion, logeas-taches, logeas-web, questionnaire, script

Historique : avant 2024, le dépôt légal se faisait sur e-coffrefort.fr (fermé à l'automne 2024). Depuis, les dépôts se font directement sur GitLab.

Documents fusionnés dans cette page

Cette page (`procedure:develop:gestionversion`) est la version à jour de la procédure #06, enrichie de la refonte de janvier 2026 (Galaxie LoGeAs, version globale, schéma du processus de mise en ligne) et fusionnée avec :

- La procédure #30 « Versionning des codes sources », page désormais obsolète
- La procédure « Gestion du dépôt Git », page désormais obsolète
- La procédure #09 « Dépôt des Codes Sources », page désormais obsolète