Retour au dossier de maintenance	
Sujets connexes	Cartographie fonctionnelle de LoGeAs Informations sur la version en cours Proc#145 : Procédure de Gestion des Versions Proc#168 - Chaîne de qualification et de livraison Proc#167 - Procédure de mise en ligne d'une version du client « lourd » Proc#170 - Guide pratique — Gestion des versions LoGeAs Web

Proc#169 - Cycle de release LoGeAs Web

Informations qualité

Suivi des modifications majeures		
Date	**Auteur**	**Modifications**
3 août 2026	Nicolas MARCHAND	Création du document
30 septembre 2026	Nicolas MARCHAND	Alignement sur la chaîne de qualification et de livraison (Proc#168) : la livraison est faite par les scripts `livrer-beta` et `livrer-release` derrière les portes bloquantes, ajout de la qualification et de la recette, suppression de la release sans beta ; mise au format de la maîtrise documentaire

Suivi des approbations	Cartographie fonctionnelle – fiche de la procédure
Objet	Décrire le dispositif technique de gestion des versions et de publication de LoGeAs Web, depuis l'intégration des développements jusqu'à la production des versions destinées aux utilisateurs, pour les dépôts `logeas-web` et, lorsque nécessaire, `logeas-lib`.
Destinataires	**- Validation des modifications :** Gérant **- Approbation du document :** Équipe développement

1. Objet et périmètre

Ce document décrit le dispositif technique de gestion des versions et de publication de LoGeAs Web : les scripts, les fichiers produits, le calcul des numéros de version et les éléments de preuve.

Il complète les documents suivants :

- Proc#145 - Procédure de gestion des versions : règles générales de versionnage et d'identification des versions.
- Proc#168 - Chaîne de qualification et de livraison : déroulé pas à pas, responsabilités, qualification, validation, recette et portes bloquantes.
- Guide pratique de gestion des versions : mode opératoire destiné aux développeurs.

En cas de divergence entre ce document et Proc#168 sur le déroulé d'une livraison, Proc#168 fait foi. Le présent document explique comment le dispositif fonctionne ; il ne remplace ni les décisions de validation ni les contrôles de qualification.

2. Objectifs

Le dispositif de release a pour objectifs :

- **Reproductibilité** : exécuter les opérations techniques selon une procédure identifiée et maîtrisée.
- **Traçabilité** : relier chaque version publiée à son code source, à ses commits et aux résultats des contrôles.
- **Sécurisation** : empêcher la production d'une livraison lorsque les contrôles obligatoires ne sont pas satisfaits.
- **Séparation des environnements** : distinguer les versions de test des versions destinées aux utilisateurs.
- **Maîtrise du versionnage** : éviter les collisions, les incohérences et les versions non identifiables.
- **Conservation des preuves** : permettre de retrouver les informations nécessaires à la justification d'une livraison.

3. Architecture générale du cycle

Le cycle de publication suit les huit étapes de la chaîne de qualification et de livraison (Proc#168) :

Étape	Opération	Outil	Résultat
1	Travail personnel puis intégration	`/fusion-dev`	Code intégré et contrôlé sur la branche de version
2	Qualification	`scripts\qualifier-version.bat`	Rapport qualité de la version candidate, en brouillon
3	Validation du rapport qualité	carto, écran « Rapports de version »	Rapport validé (avec ou sans réserves) et publié
4	Beta	`scripts\livrer-beta.bat`	Version de test construite, taguée, avec son rapport de release
5	Déploiement sur le serveur de test	copie manuelle	Beta disponible pour la recette
6	Recette de la beta	carto, écran « Rapports de version »	Recette conforme ou non conforme
7	Release client	`scripts\livrer-release.bat`	Version de production construite et taguée
8	Déploiement client	copie manuelle	Version en ligne

Deux **portes bloquantes** encadrent les livraisons : la porte beta (avant l'étape 4) et la porte release (avant l'étape 7). Elles sont décrites dans Proc#168.

Le passage d'une étape à l'autre reste une décision humaine. L'automatisation réalise les opérations et les contrôles prévus, mais ne décide pas de la pertinence fonctionnelle d'une livraison.

Rôle des commandes de l'assistant Claude Code. `/fusion-dev` réalise l'intégration. `/qualifier-version`, `/release-beta` et `/release-client` **ne livrent rien elles-mêmes** : elles vérifient les préconditions ou la porte, expliquent ce qui bloque, puis renvoient vers les scripts ci-

dessus, lancés par le développeur.

Il n'y a pas de release client sans beta recettée : la porte release l'exige. La seule exception est la procédure d'urgence (Proc#168, cas 8), qui produit une version affichée « Version non qualifiée » et doit être régularisée.

4. Organisation des dépôts et des environnements

4.1 Dépôts

Le cycle concerne principalement les dépôts suivants :

Dépôt	Rôle
`logeas-web`	Interface Angular LoGeAs Web et scripts de qualification et de livraison
`logeas-lib`	Bibliothèques Angular partagées utilisées par les interfaces
`logeas-cartographie`	Cartographie fonctionnelle : écran « Rapports de version » (validation des rapports qualité, recette)

Les bibliothèques de `logeas-lib` sont utilisées par `logeas-web` sous forme de paquets construits. Le changement de branche ne suffit donc pas à garantir la cohérence des dépendances : les paquets installés doivent correspondre au code de la bibliothèque. C'est pourquoi la qualification reconstruit et réinstalle `logeas-lib`, et pourquoi les portes vérifient que les paquets installés sont toujours ceux du run qualifié.

4.2 Environnements de génération

Les builds sont générés localement sur le poste de développement.

Destination	Usage
`D:\Developpements\VersionPublic\LogeasWeb-Test\browser\`	Version beta destinée aux tests
`D:\Developpements\VersionPublic\LogeasWeb\browser\`	Version destinée à la production

La séparation des destinations est assurée par les configurations Angular :

- `production,beta` pour la génération beta ;
- `production` pour la génération client.

Cette séparation évite qu'une génération de test écrase directement les fichiers de production.

4.3 Déploiement

La génération des fichiers et leur déploiement sur les serveurs sont deux opérations distinctes.

Le transfert vers les serveurs de test et de production est réalisé manuellement, par une personne

disposant des accès nécessaires au bureau à distance.

Les scripts de livraison ne réalisent pas eux-mêmes ce transfert.

5. Description technique des étapes

5.1 Travail personnel

Chaque développeur travaille sur une branche personnelle issue de la branche de version concernée.

Les commits et les sauvegardes de cette branche restent libres, afin de ne pas imposer les contrôles de publication à chaque sauvegarde intermédiaire.

Cette étape ne constitue pas une intégration validée et ne garantit pas que le code compile ou que les tests passent.

Le premier contrôle technique obligatoire du cycle intervient lors de l'intégration.

5.2 Intégration : /fusion-dev

La commande `/fusion-dev` intègre les développements personnels à la branche de version commune.

Elle réalise les opérations suivantes :

- récupération des informations du dépôt distant ;
- sélection et mise à jour de la branche de version cible ;
- identification des commits à intégrer ;
- fusion de la branche personnelle ;
- constitution d'un message de fusion contenant la liste des commits intégrés ;
- vérification du code après fusion ;
- publication de la branche commune si les contrôles sont satisfaisants.

Les contrôles techniques portent sur :

- la compilation Angular ;
- les tests unitaires ;
- l'analyse statique et le lint.

Les résultats sont examinés avant la publication de la branche commune.

En cas d'échec, la procédure ne doit pas publier une intégration non validée. En cas de conflit métier ambigu, une décision humaine est nécessaire.

La procédure n'utilise pas de `push –force` sur la branche commune.

Résultat attendu : une branche de version intégrée, contrôlée et accompagnée d'un historique de fusion exploitable.

5.3 Qualification : qualifier-version

Le script `scripts\qualifier-version.bat` produit le rapport qualité de la prochaine version, sur un code identifié sans ambiguïté.

Il réalise les opérations suivantes :

- vérification des préconditions sur `logeas-web` et `logeas-lib` : branche de version, aucune modification locale, branche à jour et poussée ;
- calcul et affichage de la **version candidate** (la prochaine beta) ;
- reconstruction de `logeas-lib` depuis son dernier commit et réinstallation dans `logeas-web` ;
- démarrage des serveurs locaux (confirmation UAC) ;
- tests unitaires des deux dépôts, puis tests d'interface (Playwright) ;
- génération du rapport qualité et de son analyse ;
- dépôt du rapport, en brouillon, sur le serveur Nono.

Le rapport est nommé `run-<version>-<AAAAMMJJ-HHMM>.html` et porte la clé du code testé (commits des deux dépôts, empreinte des paquets `logeas-lib`).

Le rapport est ensuite validé, validé avec réserves ou refusé dans la carto (Proc#168, « Validation du rapport qualité »).

5.4 Beta : livrer-beta

Le script `scripts\livrer-beta.bat` produit la version destinée à la recette. Il s'arrête au premier échec.

Les opérations techniques sont les suivantes :

1. **porte beta** (`scripts/porte-livraison.js beta`) : les deux dépôts sont propres, à jour et poussés, et un rapport de qualification de la version candidate, validé, porte exactement le dernier commit des deux dépôts et les mêmes paquets `logeas-lib` ;
2. mise à jour du numéro de version, 4e chiffre (`npm run prebuild:beta`, qui exécute `scripts/version.js beta`) ;
3. génération du rapport de release (`scripts/release-report.js beta –qualification`) : les tests unitaires et le résumé des tests d'interface y sont **repris** de la qualification, rien n'est rejoué ;
4. compilation : `ng build –configuration production,beta`, vers `D:\Developpements\VersionPublic\LogeasWeb-Test\` ;
5. commit dédié « Preparation version X.Y.Z.W », tag annoté `X.Y.Z.W_beta` et, si besoin, tag `web-X.Y.Z.W_beta` dans `logeas-lib` ;
6. après confirmation : publication du commit et du tag, **inscription de la beta dans la trace du rapport qualité** (indispensable à la porte release), reprise du rapport de release dans la carto.

Si le rapport de release ou le build échoue, la mise à jour du numéro de version est annulée : rien n'est commité ni tagué.

La beta ne vaut pas validation fonctionnelle : elle fournit l'artefact et les éléments de traçabilité destinés à la recette.

5.5 Release client : livrer-release

Le script `scripts\livrer-release.bat` produit la version destinée aux utilisateurs. Son déroulé est le même que celui de la beta, derrière la porte release.

1. **porte release** (`scripts/porte-livraison.js release`) : le tag de la beta est le dernier commit de `logeas-web` (**aucun commit depuis la beta**), `logeas-lib` est toujours au commit qualifié, le rapport qualité est toujours validé et la **recette de cette beta est « Conforme »** ;
2. mise à jour du numéro de version à 3 chiffres (`npm run prebuild`, qui exécute `scripts/version.js release`) ;
3. génération du rapport de release ;
4. compilation : `ng build –configuration production`, vers `D:\Developpements\VersionPublic\LogeasWeb\` ;
5. commit dédié, tag annoté `X.Y.Z_release` ;
6. après confirmation : publication, inscription de la release dans la trace du rapport qualité.

Le rapport qualité, publié lors de sa validation, est la pièce qualité de la release.

Tout correctif après une beta impose une nouvelle qualification et une nouvelle beta (Proc#168, cas 3) : aucune modification n'est admise entre la beta recettée et la release.

5.6 Build hors chaîne

Les commandes `npm run build:beta` et `npm run build` restent utilisables, mais elles produisent une version **hors chaîne**, affichée « Version non qualifiée » dans « À propos > Logiciel » et dans son rapport de release. Elles ne doivent pas être utilisées pour une livraison normale ; leur usage en urgence est décrit dans Proc#168, cas 8.

6. Gestion des versions

6.1 Format

Les versions de LoGeAs Web utilisent trois ou quatre composantes numériques :

- `X.Y.Z` pour une release ;
- `X.Y.Z.W` pour une beta.

Le suffixe du tag identifie la nature de la publication :

- `_beta` pour une version de test ;
- `_release` pour une version destinée aux utilisateurs.

Les règles générales de numérotation sont définies dans la procédure de gestion des versions.

6.2 Calcul automatique

Le calcul est centralisé dans :

```
scripts/version.js
```

La fonction `resolveNextVersion` détermine le numéro à utiliser en fonction :

- de la version actuelle de `package.json` ;
- du nom de la branche de version ;
- du mode de génération (beta ou release) ;
- des tags de release déjà présents dans Git.

Pour une beta, le quatrième chiffre est incrémenté à chaque nouvelle génération du même cycle.

Lorsqu'une nouvelle branche numérotée correspond à un cycle ultérieur, le script utilise son numéro comme nouvelle base de version.

Pour une release, le quatrième chiffre est supprimé. Le numéro `X.Y.Z` utilisé par les betas est repris s'il n'a pas déjà été publié. Si une release portant ce numéro existe déjà, le script avance vers un nouveau numéro conformément à la règle implémentée.

La commande `node scripts/version.js beta –simuler` affiche la version candidate sans rien modifier.

6.3 Traçabilité de la version

Le fichier `src/assets/version.ts` est régénéré lors de la préparation d'un build.

Il contient notamment les informations permettant d'identifier :

- la branche Git ;
- le commit source ;
- la date de génération.

Ces données permettent de rapprocher une version exécutée de son état source.

6.4 Cas particuliers

Les cas de changement de version mineure, de correction d'une version déjà livrée et de travail parallèle sur plusieurs versions sont décrits dans Proc#168 (cas 6 et 7) et dans le guide pratique.

Ils ne doivent pas conduire à réécrire l'historique d'une version déjà publiée.

7. Contrôles et garde-fous

Les procédures outillées appliquent des contrôles destinés à éviter les publications incohérentes ou

non traçables.

Contrôle	Étape concernée	Objectif
État des dépôts (propres, à jour, poussés)	Intégration, qualification, beta, release	Éviter de publier à partir d'un état local non maîtrisé
Branche de version	Toutes les étapes outillées	Garantir que l'opération concerne la branche attendue
Compilation, tests unitaires, lint	Intégration	Détecter les erreurs avant l'intégration
Tests unitaires et tests d'interface	Qualification	Produire le rapport qualité de la version candidate
Porte beta	Beta	N'ouvrir une beta que sur un code qualifié, avec un rapport validé
Porte release	Release	N'ouvrir une release que sur la beta recettée conforme, sans commit depuis
Paquets `logeas-lib` installés	Beta, release	Garantir que la bibliothèque livrée est celle qui a été qualifiée
Numéro de version et tags	Beta, release	Éviter les collisions et préserver la traçabilité
Séparation des répertoires de sortie	Beta, release	Empêcher la confusion entre test et production
Liste des commits intégrés	Intégration	Conserver la traçabilité des changements
Rapport de release	Beta, release	Documenter les changements et renvoyer au rapport qualité

En cas d'échec d'un contrôle bloquant, la procédure s'arrête et signale l'anomalie. **Une porte ne se contourne jamais** : si le blocage est injustifié, c'est un défaut des scripts, à signaler et corriger.

Aucune procédure ne doit résoudre silencieusement un conflit métier ou écraser des modifications non validées.

Les opérations de publication n'utilisent pas `push --force` et ne suppriment pas automatiquement des branches ou des tags.

Les échecs de tests déjà connus doivent être identifiés et distingués des régressions introduites par le changement en cours : c'est l'objet de la décision « Validé avec réserves » (Proc#168).

8. Rapports et éléments de preuve

Les éléments suivants permettent de reconstituer une publication :

- l'historique Git de la branche concernée ;
- le commit de fusion et la liste des commits intégrés ;
- le commit « Preparation version » ;
- le tag annoté de beta ou de release (et le tag `web-*` de `logeas-lib`) ;
- les informations intégrées dans `version.ts` ;
- le **rapport qualité** de la version (`run-<version>-<date>.html`), sa décision de validation et la recette de la beta ;
- le **rapport de release** HTML, enregistré dans `src/assets/releases/` et référencé dans

`manifest.json`.

Élément	Où le consulter
Rapports qualité, décisions, recettes	carto, écran « Rapports de version »
Historique d'un rapport (validation, publication, beta, recette, release)	table `ApprobDoc` du serveur Nono, trace du rapport
Rapports de release	LoGeAs Web, écran « Rapports de release »
Qualification d'une version livrée	son rapport de release et « À propos > Logiciel »

Les modalités de validation et de conservation de ces preuves relèvent de la chaîne de qualification et de livraison.

9. Opérations restant manuelles

Les opérations suivantes restent sous la responsabilité des personnes habilitées :

- le choix du moment de l'intégration ;
- le lancement de la qualification ;
- la relecture et la validation du rapport qualité ;
- le lancement de la beta ;
- la recette de la beta ;
- le lancement de la release ;
- le transfert des fichiers générés vers les serveurs ;
- la vérification du déploiement sur l'environnement cible.

La séparation entre génération, validation et déploiement permet de conserver un contrôle humain sur les étapes engageant la mise à disposition du logiciel.

10. Maintenance du dispositif

Les scripts et configurations qui mettent en œuvre le cycle de release font partie du dispositif de développement de LoGeAs Web.

Toute modification significative de ces éléments doit être contrôlée et testée avant d'être utilisée pour une publication.

Les évolutions du dispositif doivent préserver :

- la traçabilité des versions ;
- la reproductibilité des générations ;
- la séparation des environnements ;
- la fiabilité des contrôles et des portes ;
- la conservation des preuves nécessaires à la certification.

La présente procédure doit être révisée lorsque le fonctionnement effectif du cycle de release évolue.

From:
https://wiki-logeas.fr/certif/ - **dokuwiki-certif**

Permanent link:
https://wiki-logeas.fr/certif/doku.php?id=certif:procedure:develop:proceduremiseenligne:logeas-web&rev=1790756882

Last update: **2026/09/30 10:28**