*Ce document a pour but de*

* *Définir les objectifs de la gestion de configuration logicielle d'une Application*
* *Déclarer les moyens et les configurations utiles à cette gestion*
* *Décrire l'utilisation de ces configurations*

*Sa rédaction peut être précédée d'une formation de 2 heures détaillant chacune de ses parties, et proposant aux participants (au plus 4 RGC – Référent de Gestion de Configuration -) d'imaginer durant la session, des instanciations spécifiques à leurs Applications*

***Recommandations importantes***

*Pour la rédaction et l'utilisation de ce Document*

* *Conservez tous les guides de rédaction comme celui-ci, même après finalisation de ce document,*

*Car ils peuvent encore aider un lecteur sur la compréhension des objectifs d'un paragraphe.*

* *Le non affichage des marques de paragraphe (bouton [¶]) les cache*
* *N’utilisez pas la couleur des commentaires (ou une couleur approchante, comme le rouge) dans le texte standard du document, pour éviter toute confusion. Cachez les commentaires fréquemment pour vérifier la lisibilité finale de votre document*
* *Les commentaires <encadrés> ainsi sont des options et alternatives :*
  + *Soit des <"textes"> prédéfinis, mais ce ne sont que des suggestions,*
  + *Soit des identifiants à préciser, comme par exemple <Nom du RA>*
* *Ces commentaires ne seront jamais imprimables (sauf en décochant l'option Police cachée pour l'ensemble du document).*
* *Modifiez le titre sur le pied de page : PGCL (Plan de Configuration Logicielle) - <Nom de l'Application>*
* *Avant la première revue de ce PGCL, merci de le transmettre au RGCT (Référent des Gestions de Configurations Transverse) par mail, afin qu'il puisse le sauvegarder dans le site C'You GCL. Ce fichier sera alors modifiable depuis le C'You GCL par le RGC/RGCT et le RGCT (utilement lors des revues de ce document)*
* *S’il est nécessaire d’imprimer, ne pas oublier de mettre à jour les n° de lignes de la table du Contenu après avoir cachés ces marques de paragraphes, car la pagination changera alors*

*Si lors de la rédaction de ce document, le rédacteur se trouve devant une incompréhension,*

*Il est indispensable qu'il appelle le RGCT disposé à lui fournir tout complément d'information*

*Pour se faire expliquer le concept bloquant, l'objectif d'un paragraphe, etc.*

*Cette trame de document doit évoluer selon les retours d'usage des rédacteurs,*

*Toutes les critiques constructives sont les bienvenues,*

*Donc elles doivent être précises avec des propositions de modifications*

*La dernière version de ce document est disponible sur le site C’You*

*Gestion de Configuration*

*Stratégie et Plan*

*Préciser ici*

*De l'Application*

*Des Applications*

*ESB*

*<Nom de l'Application>*

*<Nom de l'Application>*

*Du Patrimoine*

*ECHANGES et Habilitations*

*<Nom du Patrimoine>*

**Liste de Diffusion**

| Nom | Fonction | RACI |
| --- | --- | --- |
| Lamine NDIAYE | RPA *(Responsable de Patrimoine Applicatif)* | R |
| Abir TAHRI | RA *(Responsable d’Application)* | R |
| Pierre Bédué | RGCT*(Référent de Gestion de Configuration Transverse)* | A |
| Equipe ESB TIBCO | Contributeur(s). *nommément si ils sont peu nombreux, sinon leur entité, comme par exemple "CDEV", "TMA IBM à tel endroit, …* | I |
|  | *Si un éditeur est concerné par le PGCL car il doit se conformer à certaines obligations qui y sont décrites, donner son nom et sa fonction "Editeur"*  *Une TRA devant récupérer des source pour construire elle-même sa version à tester est concernée alors par ce PGCL.* | *I* |
|  | *Pas la peine d'ajouter ici le RGC (Référent de Gestion de Configuration) car c'est le rédacteur, dont le nom va apparaitre dans C’You* |  |

**Evolutions du document**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version  Si le document est géré sous C'You, il n'est pas nécessaire d'indiquer ici les mini ajustements (donnant lieu à une version automatique de C'You) qui sont plutôt à préciser dans la zone commentaire à chaque archivage dans C'You | Date de Validation  Pas celle de parution de la version, mais celle d'acceptation par l'acteur de la colonne Validation | Rédacteur  Par défaut le RGC, mais cela peut être aussi le RGCT, le RPA, le RA si il n'est pas aussi RGC | Nature des modifications | Validation  Qui valide le document  RGCT au moins  RA si ce n'est pas le RGC  RPA possiblement |
| 0.1 | 24/01/2020 | RGCT | Initialisation du document | RPA, RA et/ou RGC, |
|  |  |  |  |  |

Le versionnement sous C'You

* Commence à 0.1, et se poursuit par 0.2, 0.3, etc., pendant la mise au point de la Stratégie
* Passera à 1.0 à la publication par le RGCT de la partie Stratégie (exploitable pour effectuer la migration)
* Se poursuit en 1.1, 1.2, etc., pendant la mise au point du Plan
* Passera à 2.0 à la publication par le RGCT de la version complète Stratégie et Plan

**Le cycle de révision de ce document est annuel car faisant partie de la Revue Patrimoniale**

Contenu

[I. Stratégie de Gestion de Configuration 4](#_Toc526261842)

[A. Besoins d’exploitations des configurations 4](#_Toc526261843)

[1. Besoins de constituer des configurations 4](#_Toc526261844)

[2. Parties prenantes des configurations 4](#_Toc526261845)

[3. Configurations : Types, Cas d'usage, Dépôts 5](#_Toc526261846)

[4. Gestion de Configuration des Environnements (GCE) 7](#_Toc526261847)

[B. Eléments à organiser 7](#_Toc526261848)

[1. Types & identifications des éléments dans les configurations 7](#_Toc526261849)

[2. Éléments externes liés 10](#_Toc526261850)

[C. Contexte d’organisation 10](#_Toc526261851)

[1. Méthodes 10](#_Toc526261852)

[2. Sous-traitances 12](#_Toc526261853)

[3. Migration de configurations 12](#_Toc526261854)

[D. Contexte technique 12](#_Toc526261855)

[1. Moyens techniques de gestion de configurations 12](#_Toc526261856)

[2. Dimensionnement des moyens 13](#_Toc526261857)

[3. Adaptation du contexte technique 13](#_Toc526261858)

[II. Plan de Gestion de Configurations 13](#_Toc526261859)

[A. Organisation des éléments 13](#_Toc526261860)

[1. Agrégations et Ségrégations 13](#_Toc526261861)

[2. Règles de gestion des configurations 13](#_Toc526261862)

[3. Règles de nommage 14](#_Toc526261863)

[B. Politiques de Versions et d'Identifications 14](#_Toc526261864)

[1. Occurrences des historisations et restaurations 14](#_Toc526261865)

[2. Règles de versionnement 14](#_Toc526261866)

[3. Surcharge d'identification 15](#_Toc526261867)

[4. Occurrences de création et d'utilisation de branches 15](#_Toc526261868)

[5. Traitements des Fusions 15](#_Toc526261869)

[C. Exploitation des éléments 16](#_Toc526261870)

[1. Cohérences Projets & Applications 16](#_Toc526261871)

[2. Gouvernance 17](#_Toc526261872)

[3. Activités, Actions et Acteurs 18](#_Toc526261873)

[III. Annexes 18](#_Toc526261874)

[A. Abréviations 18](#_Toc526261875)

[B. Références documentaires 19](#_Toc526261876)

[C. Rôles et Activités GC 19](#_Toc526261877)

[D. Revue GC 21](#_Toc526261878)

Ce document a pour but de

* Définir les objectifs de la gestion de configuration logicielle d'une application
* Déclarer les moyens et les configurations utiles à cette gestion
* Décrire l'utilisation de ces configurations

# Stratégie de Gestion de Configuration

QUELQUES RAPPELS SUR LA GESTION DE CONFIGURATION

Une Configuration est un assemblage d'éléments. Ce groupement doit être en final complet et cohérent, pour un objectif commun.

* Complet? La présence de chacun des éléments est indispensable
* Cohérent? L’état de chacun des éléments est en accord avec l’ensemble
* Objectif commun? La composition de l’ensemble et l’état individuel de ses constituants ne sont justifiés que par un objectif auquel tous participent : répondre au besoin, inventorier un contexte …
* En final, ce groupement doit être validé vis-à-vis de l’objectif et remanié le cas contraire, pour que finalement que la configuration corresponde à l'Objectif

A. Activité à organiser

La GC (Gestion de Configuration) est une activité permettant de contrôler et d’utiliser des ensembles d'éléments identifiés et enregistrés. Cette activité s'appuie sur les 5 sous-activités :

1. Planification d’un processus, avec une organisation, des techniques et des moyens

2. Identification des éléments individuellement, puis en ensembles complets et cohérents (listes)

3. Contrôle des Actions (identifier, assembler, sélectionner, modifier et exploiter)

4. Conservation de l’historique des évolutions des configurations au cours des Activités (traçabilités)

5. Vérification de disponibilité et des validités (identifications, complétudes, cohérences)

B. De la Stratégie au Plan

Pour simplifier la démarche, il est nécessaire de décrire le binôme Besoins-Moyens :

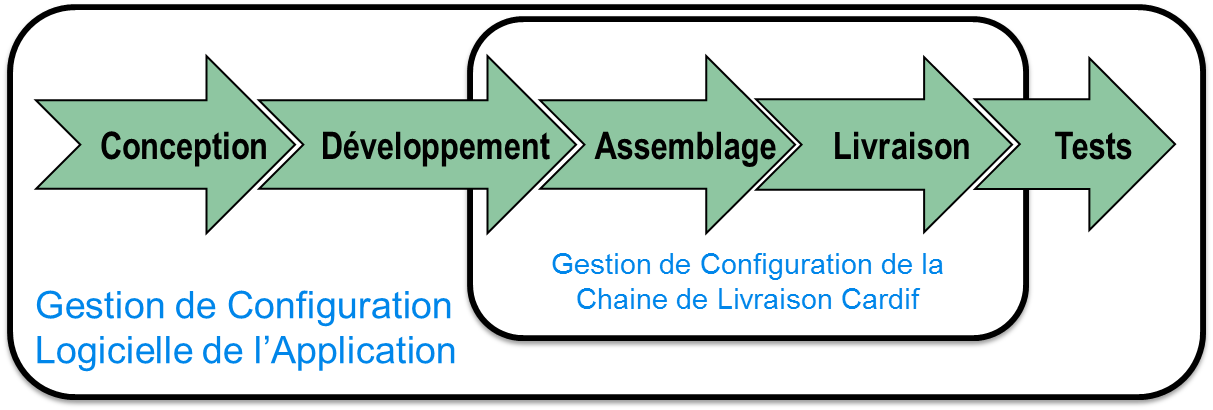
Le QUOI, c’est définir les objectifs et leur cadre selon une Stratégie

Le COMMENT, c’est répondre aux objectifs contraints, par des solutions dans un Plan

Ce document a pour objectifs de partir du contexte et des objectifs de la GC pour décrire la solution répondant à ces besoins.

Ce document va se focaliser sur la Gestion de Configuration Logicielle (GCL)

Avec ses liens avec d'autre GC (Gestion de configurations documentaires, des tests, des Environnements, etc…) en précisant leurs "accostages" c’est-à-dire les conditions et synergies de leurs relations.



Pour l’Application ESB TIBCO <nom de l'application> sinon "les Applications" suivi de la <liste des applications concernées>, cette gestion de configuration doit permettre de fiabiliser et de systématiser :

* Les développements logiciels optionnel : « devant utiliser à <court/moyen/ long> terme (cible 2021) la <Chaine de Livraison Cardif (CLC) / Toolchain Group (TCG)> pour les technologies qu'elle supporte (Cf. § B.1), ce qui nécessite une migration technologique » => supprimer si besoin la fin de la phrase à partir d' "optionnel.", sinon faire les choix de d’usage à +/- long terme d’une des 2 chaines d’outillages
* Les conceptions, développements, constructions des packages versionnés gérés à terme par la < CLC / TCG >, => supprimer si besoin la fin de la phrase à partir de "gérés…", sinon faire un choix entre CLC et TCG
* Les associations des configurations construites et déployables avec les gestions de configurations complémentaires : gestion documentaire, sans intégrer la gestion des tests

POINT D'ATTENTION: Diverses règles de GC sont définies et consultables sur le site communautaire C’You "GCL" (Global Configuration League) par le lien ci-dessous :

<https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx>

Les règles sont groupées sous le menu "Règles" du bandeau du site

Ce PGCL s'appliquera à référencer les quelques règles plutôt que les recopier afin de bénéficier de leurs évolutions (demandes prises en compte lors de revues régulières réalisées entre le RGC et le RGCT).

Tout au long de ce PGCL, le renvoi à l'une des règles sera noté [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx), avec l'hyper lien intégré

## Besoins d’exploitations des configurations

### Besoins de constituer des configurations

**Evoquer ici**

* Les besoins de constituer des ensembles d'éléments,
* Typiquement
  + La cohérence et complétude des développements, constructions et livraisons
  + Les technologies ayant besoin de gérer leurs configurations
  + Les acteurs devant coordonner la création et l'exploitation de configurations
  + Les versionnements nécessaires

Les textes sont déjà proposés ci-dessous)

**Ne pas spécifier ici**

* Ni les moyens de gérer ces configurations (du type traçabilité, complétude, cohérence, …)
* Ni le comment utiliser ces moyens
* Développements : les fichiers source de l'Application ont vocation à évoluer et la maitrise de leurs évolutions nécessite de s'assurer de leur cohérence et complétude, depuis leurs développements jusqu'à leurs livraisons. Corriger/compléter si besoins ces propos

NB: les fichiers sources peuvent être aussi des fichiers de paramétrages directement associés à une livraison d'un tierce partie (éditeur) sans avoir à modifier cette solution technique externe. Par exemple ces fichiers doivent pouvoir particulariser cette livraison.

Compléter si c'est le cas, en précisant les configurations par niveaux, par exemple :

* Un package est une configuration d'éléments
* L'Application est une configuration de packages
* Un projet\* est une configuration d'Applications\* le projet ici est en terme d'objectifs et n'est pas à confondre avec le terme de projet utilisé dans certains outils (Eclipse, Jira, Ultraedit…)

Il est aussi possible de disposer de configurations techniques collaborantes (par exemple une structure de base de données + ses modules d'exploitation externes) donc non hiérarchiques, bien que liés par un versionnement.

Pour le cas d'un éditeur, imaginer un dépôt avec les tags des versions de l'éditeur contenant les releases Notes avec les tags des versions Cardif

* Technologies : L'Application ou <les Applications> si c'est le cas est <sont> <mono>multi-technologies <s> et la cohérence des développements, constructions et livraisons associées.

Seules sont utiles les technologies associées aux fichiers sources qui seront mis en configuration

L'utilité est dans la répartitions des fichiers sources dans des configurations prises en compte automatiquement par la CLC ou la TCG, ou au moins par des acteurs associés à ces technologies.

Attention, bien préciser les technologies pour les différentes couches du SI : langage, OS, ordonnanceur, base de données,

Donc dans le cas de solution plus ou moins complètes fournies par une tierce partie (éditeur par exemple) il est inutile d'évoquer leurs technologies

* + <Technologie 1> <cadre : développement, paramétrages, scripts, …>
  + <Technologie 2> <cadre : développement, paramétrages, scripts, …>
  + Java code générés manuellment et automatiquement
  + Xml Modèle données
  + Xsd Shéma de données
  + Wsdl description d’un service 🡺 contrat d’interface contenant contenant toutes les infos relatives au web service :
    - Les headers
    - Les paramètres entrées/sorties
    - les opérations
    - les end point
  + Dtl librairie
  + Sql requètes d’administrations et d’exploitations, appelés par des ksh
  + Ksh scripts exploitation et d’installation
  + Txt properties, varaibles d’environnement et applicatives

NB : il existe un pattern générique pour les développements gérés par le framework TIBCO. L’agencement des fichiers est déclarés spécifiquement pour chaque web Service et Container. Cependant ce pattern peut être adapté pour chaque Service et Container

* Acteurs : les modes Projet et Run (ou bien que des évolutions : utiliser ici le vocabulaire habituel) (trains de maintenances) peuvent être dissociés (ou pas=> préciser le cas) (acteurs identiques). Cependant la cohérence et la traçabilité voire la coordination doivent passer par une Gestion de Configuration (GC) commune et centralisée sur certains aspects.
* Versionnements : toutes les configurations d'éléments sont distinguables par des numéros de versions utilisant un format générique. Ces versions sont utilisées de la même façon (format) et communément (occurrences d'application) par les Parties Prenantes du Patrimoine pour la Documentation, le Développement, les Livraisons, les Tests. (Supprimer/corriger les étapes impropres, comme par exemple les développements uniquement par un éditeur, voire même des tests complètement sous-traité ou bien non réalisés, la confiance et le mode relationnel avec un éditeur ne le nécessitant pas. Dans ces cas indiquer succinctement le système de versionnement avec l'Editeur)

### Parties prenantes des configurations

Présenter les différents intervenants de cette gestion de configuration: organisateurs, utilisateurs, clients, fournisseurs, constructeurs, exploitants. Indiquer leurs rôles de façon macroscopique avec un RACI (Réalise - Autorité - Consulté - Informé). Les rôles associés à la gestion de configurations seront décrits dans l'annexe de ce document (Cf. § III.AB)

Les Configurations gérées <au cours d’un Projet ou d’une Maintenance > l'un ou l'autre ou bien les deux> pour l'Application ESB sont à la disposition de contributeurs internes à l'entreprise et ceux externes autorisés. Des règles d'accès garantissent à la fois la confidentialité et le respect des rôles créateurs vs exploitants. Des règles doivent encadrer les droits et responsabilités d'une TMA (Cf. § I.C.2).(TMA optionnelle)

Contributeurs identifiés

Supprimer les lignes inutiles

|  |  |
| --- | --- |
| Développeurs | situer l'équipe, internes Cardif |
| RPA | <responsable de patrimoine applicatif> Lamine NDIAYE |
| CDT : Centre de Tests Cardif | le cas échéant |
| Métier | <nom de l'entité> chez Cardif Patrimoines exploitants l’ESB |
| RA | <Nom du RA de l’/des Application(s) concernée(s)> Abir TAHRI |
| RGC : Référent de la Gestion de Configuration de l'Application | <normalement le RA> Gerard KOUASSI |
| RTC : Responsable Technique des Configurations de l'Application | <acteur technique de GC> Gerard KOUASSI |
| Exploitant des environnements   * Prod * Prod - 1 (Préprod) | (par exemple IPS) |
| Exploitant Cardif des environnements non gérés par la production | (SOD) |

Les rôles des acteurs sur les Configurations sont décrits de façon macroscopique avec un RACI (*Réalise - Autorité - Consulté - Informé*)

Leurs rôles, associés aux différents actions de gestion de configurations, seront précisés dans l'annexe de ce document (Cf. § III.C – Annexes – Rôles et Activités GC). (En effet les actions doivent être décrites avant d'être attribuées)

Cas du Mergeur (responsable de la réalisation technique des fusions sous Git)

* C'est celui qui réalise les fusions des travaux collaboratifs et parallèles) : il ne décide pas des occurrences de merge mais une même personne peut prendre ces décisions (RA faisant fonction de RTC) et effectivement ensuite réaliser la fusion technique entre plusieurs fichiers sources.
* Ce rôle est attribué au à priori au RTC ou bien à tout autre contributeur compétent pour réaliser les fusions sous Git, et aussi compétent dans la compréhension fonctionnelles et techniques de ces fusions dans le contexte du Projet ou de la Maintenance.
* Dans tous les cas, le RGC le ou les désigne (ou le fait désigner par la TMA)

Préciser ici quels sont ceux qui assurent la fonction de mergeurs : personne spécifique (RA, RTC, développeurs) mais non nominativement.

Rôles spécifiques de la Gestion de Configuration

* Le rôle de RGC est assumé par <RA> sauf si rôle délégué, donc alors préciser ici son <Nom> et sa <fonction>
* Le décideur des occurrences de fusion sur une branche projet et Master est le RA. C’est une proposition. Sur la branche projet, cela peut être que sous la responsabilité du CP. Si c'est quelqu'un d'autre, préciser son <Nom> et sa <fonction>.
* Le RA est l'intermédiaire vis-à-vis de(s) l’équipe projet <"Interne"> et/ou <Externe> :
* Valide les recettes de l'équipe projet pour autoriser les fusions de MEP
* Valide les documents projets gérés en configuration dans un C'You Projet (ou par un système autre à nommer ici) par une équipe Projet
* Valide et récupère les documents applicatifs pour les gérer en configuration dans un C'You Application (ou par un système autre à nommer ici)
* Dans le cas d'une maintenance, le RA devra avoir les mêmes obligations de GC vis-à-vis d'un C'You de maintenance (ou par un système autre à nommer ici) et d’une équipe de maintenance (Int/Ext) proposition.

La CLC/TCG <l’une ou l’autre> sera à terme (2021) la consommatrice principale (voire unique) des configurations de fichiers sources C'est une proposition => Phrase inutile à supprimer si la ni la CLC, ni la TCG ne sont utilisées par l'Application.

Le CDT (Centre de Tests Cardif) peut être consommateur des configurations de spécifications afin de préparer les tests correspondants. (à ces spécifications gérées en GCL par le RA). C'est une proposition =>

* Phrase inutile à supprimer si le CDT n'est pas concerné par les tests, ou bien
* Phrase à corriger si le CDT est concerné par les tests, mais ne prend pas en compte les configurations de spécifications gérées en configuration par l'Application

Les concepteurs et développeurs, contributeurs à la conception d'éléments de configurations, sont à la fois consommateurs et producteurs des configurations de spécifications et de fichiers sources.

### Configurations : Types, Cas d'usage, Dépôts

L'objectif de ce paragraphe est

1. De déterminer les Types de Configuration pour les Usages essentiels, selon les Besoins évoqués dans le § I.A.1,
2. Puis de lister en conséquence les Dépôts nécessaires à créer, gérer et/ou utiliser (Notion : les dépôts sont les structures techniques de stockage et d'administration d'éléments regroupés et organisés en configurations)

Les façons d'utiliser ces configurations seront précisées dans la partie "Plan" de ce PGCL

Pourquoi préciser " les Dépôts nécessaires à créer, gérer et/ou utiliser " ? :

La Gestion de configuration Logicielle va

* + Gérer des dépôts associés aux développement : fichiers sources, fichiers de paramétrages, fichiers construits, assemblés, livrés, déployés
  + Utiliser et se coordonner avec des gestionnaires de configurations existants
    - De documents
    - De Tests
    - D'environnements
    - etc.

Une liste générique de types de configurations est proposée :

1. Configurations de développement
2. Configurations de construction/assemblage et de Livraison
3. Configurations de documents liés à l'Application
4. Configurations de documents liés au Projet
5. Configurations de tests
6. Configurations d'Audits (Cast)
7. Configurations de services

Dans cette liste, ne garder que les Configurations appropriées au contexte à gérer et les adapter si besoin vis-à-vis de ce contexte.

Les éléments que ces configurations gèrent seront précisés dans le § B.1

Dans le cas de Cardif, pour les développements Open,

* Seules les configurations de développements gérés sous Git vont faire l'objet de spécificités à préciser dans ce PGCL
* Les autres configurations seront à choisir dans la liste générique ci-dessus, pour les utiliser sans les gérer

Ci-dessous, un petit texte introductif est proposé :

Les Configurations gérées au cours d’un Projet ou d’une Maintenance pour l’Application <les Applications> objet(s) de ce PGCL, sont de plusieurs types, décrits ci-dessous avec :

* Leurs types d’éléments gérés
* Les occurrences d’alimentation et d’exploitation (faire le distinguo des cas de création/modifications d'éléments, de leur usage en lecture seule)
* Les contraintes particulières gérées pour l’Application <les Applications>

Ces caractéristiques doivent permettre de définir précisément les configurations nécessaires

1. Configurations de développement
   * Les fichiers développés y sont référencés de façon asynchrone par les développeurs à travers le framework ou directement
   * La CLC/TCG <l’une ou l’autre> utilisera ces fichiers pour construire les livrables (packages) à déployer. Phrase inutile à supprimer si la ni la CLC, ni la TCG ne sont utilisées par l'Application.
   * Pour les technologies non gérées par la CLC et la TCG, des exploitants utilisent des fichiers pour développer, fichiers qui à termes auront vocation à être gérés par Git, mais sans être déployés par la CLC oui bien la TCG.
   * Les concepteurs les exploitent pour les études d’impact.
   * La règle de nommage des dépôts Git est consultable par ce lien [C'You - Communauté GCL - Règle de nommages des dépôts Git](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/Règle-de-nommage-des-dépôts-Git.aspx)

IL EST INDISPENSABLE DE LIRE CETTE REGLE AVANT DE CONTINUER LA REDACTION car elle oriente très fortement les choix des dépôts Git nécessaires

Les fichiers sources seront par défaut placés dans des dépôts de l'outil de GCL "Git"

Ces dépôts Git ne peuvent contenir, ni documents, ni tests (à part des tests unitaires)

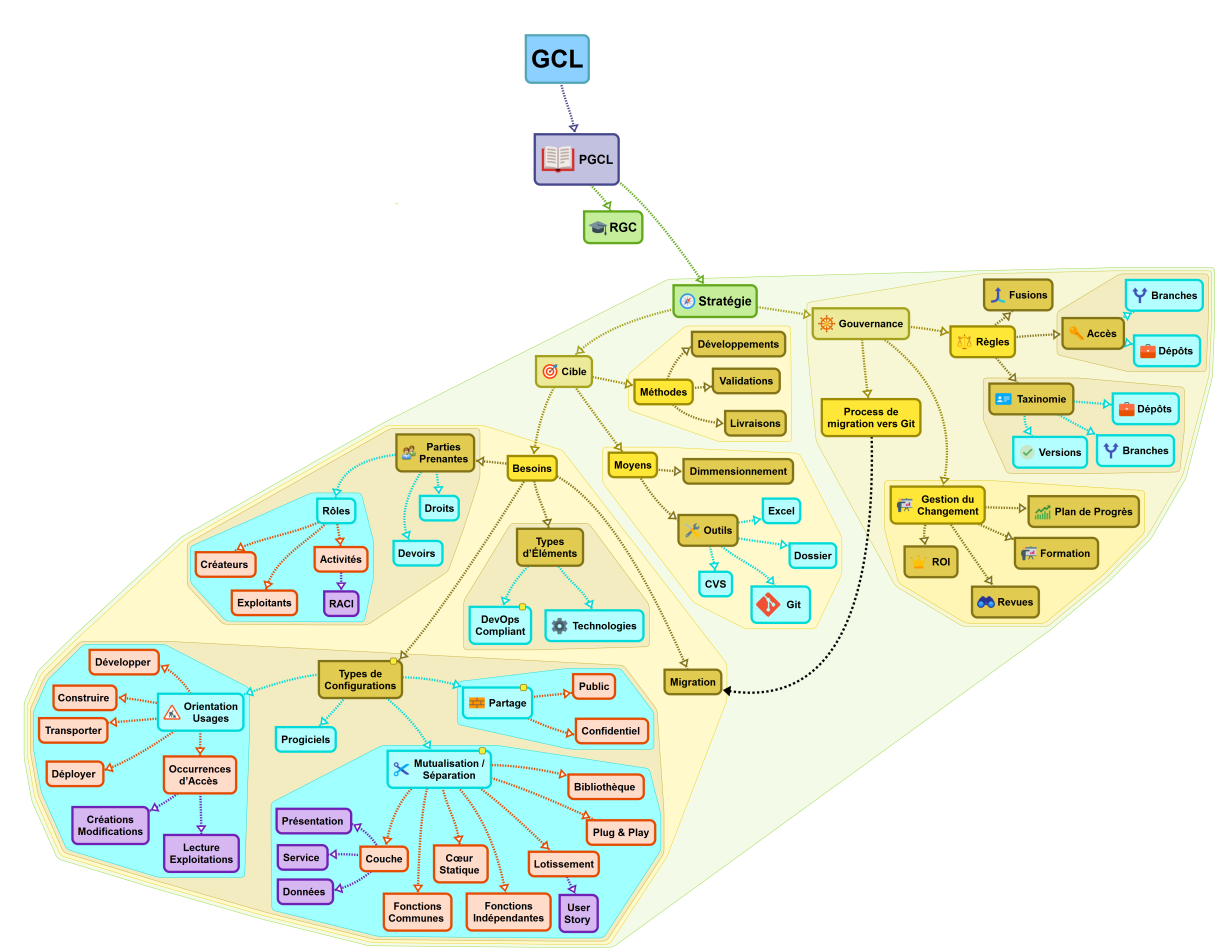
La répartition des fichiers sources dans des dépôts différents est orienté par :

1. Les technologies des fichiers sources
   * Certaines technologies pouvant être associées à la fois dans leurs développements et leurs déploiements et usages (ex : Ksh et .sql)
2. Les usages de ces fichiers sources
   * Développer, Construire, Transporter, Déployer
   * Les occurrences d'accès : Création, Modification, Lecture, Exploitation
3. La mutualisation / séparation pour des usages identiques

Cela permet de la ségrégation d'accès, car moins il y a d'accédant à un dépôt, moins les fusions risquent d'être conflictuelles (fichier modifié par plus d'un développeur)

* + Fonctions communes
  + Fonctions indépendantes
  + Cœur statique ((kernel stable)
  + Couches applicatives : Présentation, Service, Données
  + Lotissement (possible aussi avec des User Story si construites avec du code très spécifique)
  + Composants Plug & Play
  + Bibliothèques de fonctionnalités (de services)
  + Certains dépôts contiennent des fichiers source permettant de construire une solution binaire
  + D'autres dépôts peuvent ne contenir que des fichiers à transporter, comme des scripts

1. Les partages autorisés, du Public au très confidentiel
2. Les progiciels (développements externes (livraisons éditeur)



Les noms des dépôts doivent respecter les règles d'organisations prévues pour les dépôts Git de Cardif, sur la page consultable par ce lien: [GlobalConfigurationLeague : Règle-de-nommage-des-dépôts-Git](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/Règle-de-nommage-des-dépôts-Git.aspx). Il est possible de s'inspirer des options proposées dans cette page

* Lister ci-dessous les dépôts Git nécessaires
  + En justifiant leur nécessité (cas de répartitions évoqués ci-dessus) et l'usage
  + En respectant les règles de nommage
    - Si le taxon Group est utilisé, préciser auparavant le ou les groupes nécessaires et les population d’utilisateurs liées à ces groupes Il n’est pas nécessaire de préciser l’URL de GitLab => commencer par le code REFI de l’application
    - Si le ou les Application(s) n’ont pas de code REFI, préciser le cas et substituer par le code App qui servira pour créer un pseudo code : "APX" + 4 digits décimaux, la valeur de ces digits étant fixée en accord avec SOT
  + Cas des Releases Notes : si elles sont techniques (scripts de déploiement) elles ont pleine légitimité à être historiées dans Git
  + Dans le cas de l'Application ESB ICE <des Applications> dans le cas d'un PGCL mutualisé faisant l'objet de ce PGCL, les dépôts sources suivants doivent être exploités autant d'item que de dépôt Git à créer
  + Les noms de groupes ESB et ICE seront utilisés pour des groupes d’utilisateurs devant accéder aux dépôts des services Synchrone (ESB) ou bien (exclusivement) Batch (ICE); Phrase à supprimer si aucun groupe n’est utilisé

**Le tableau ci-dessous présente la liste des dépôts Git nécessaire pour l’Application** :

(Il n’est pas nécessaire de préciser l’URL d’accès à GitLab

|  |  |
| --- | --- |
| Nom des dépôts | Contenant et Usage |
| Respecter la syntaxe suivante (le taxon Groupe étant optionnel):  [Code REFI]/[Groupe]/<Technologie>.<artefact id> | Types de fichiers et/ou fonctionnalités pour lesquelles les fichiers sont gérés en configurations dans ce dépôt |
| AP10792/ESB/services/tibco.pbc\_service\_<nom du service> | Chaque service disposant de son propre dépôt Git |
| AP10792/ESB/containers/tibco.pbc\_container\_<nom du container> | Chaque container disposant de son propre dépôt Git |
| AP10792/ESB/libraries/tibco.pbc\_library\_<nom de la librairie> | Chaque librairie disposant de son propre dépôt |
| AP10792/ESB/core/tibco.pbc\_core\_<nom du core> | Chaque core disposant de son propre dépôt |
| AP10792/ESB/framework/tibco.pbc\_framework | Pour conserver les evolution spécifiques à l’ESB |

1. Configurations de construction/assemblage et de Livraison

Ces configurations sont de l'ordre de la gestion de configurations des environnements (GCE) car précisant des interfaces et des contextes de serveurs

* + Gère-les packages créés par le framework dans les dépôts Container/TCG <l’une ou l’autre> => supprimer ce paragraphe si non utilisée
  + La CLC/TCG <l’une ou l’autre> l’alimente à partir de la configuration de développement, avec une contrainte de mise à disposition que doit respecter le Projet ou la Maintenance (les sources impliquées dans la construction d’un package sont ceux en bout de branche) supprimer ce paragraphe si ni la CLC ni la TCG ne sont utilisées
  + La CLC/TCG <l’une ou l’autre> l’exploite pour réaliser ses déploiements supprimer ce paragraphe si ni la CLC ni la TCG ne sont utilisées
  + Certaines technologies non gérées par la CLC et la TCG doivent passer obligatoirement par ces configurations pour construire et déployer (ou bien déployer sans construire)
  + Contraintes d'accès aux sources permettant la construction et la livraison

1. Configurations de documents liés à l'Application
   * Ensembles documentaires correspondant au fonctionnement de l'Application en Production, dont la gestion est décrite dans le PGCD (Plan de Gestion de Configuration Documentaire) de l'Application Localiser ici ce document, ou bien préciser l’intention d’en créer un (avec la date projetée) ou bien justifier ici son inexistence : besoin, mutualisation PGCD, …
   * Ces documents sont gérés par le SharePoint C'You Application Si ce n'est pas le cas, justifier une gestion documentaire spécifique et préciser l'organisation des dépôts documentaire, leur localisation et leurs moyens de gestion
   * Le RA l’exploite pour initier la documentation projet
   * La configuration développée lui impose ses numéros de version applicative (donc cela implique une colonne ‘Version Applicative’ pour TOUS les documents dans le C’You Application)
   * Mis à jour à chaque nouvelle Mise En Production (MEP) de l'Application
   * Les documents associés à l'Application sont décrits dans la Cartographie Documentation validé par <préciser l'entité> (Cf. § III.B Références Documentaires)
2. Configurations de documents liés au Projet/Maintenance l'un ou l'autre ou les deux
   * Ensembles documentaires correspondant au Projet/Maintenances</Maintenance> l'un ou l'autre ou les deux, dont la gestion est décrite dans le PGCD (Plan de Gestion Documentaire) du Projet Localiser ici ce document ou bien justifier ici son inexistence : besoin, mutualisation PGCD, …
   * Ces documents sont gérés dans un dossier spécifique par le SharePoint C'You Application Si ce n'est pas le cas, justifier une gestion documentaire spécifique et préciser l'organisation des dépôts documentaire, leur localisation et leurs moyens de gestion
   * Le RA l'a initié à partir de la documentation Applicative
   * La configuration développée lui impose ses numéros de version applicative.
   * Les documents associés au Projet sont décrits dans la Cartographie Documentation validé par <préciser l'entité> (Cf. § III.B Références Documentaires)
3. Configurations de tests
   * Au niveau Tests de code, gestion en configuration de ces tests dans C’You
   * Au-delà des développements, (Tests fonctionnels, tests techniques de performances)<cacher les options inutiles>, les tests sont sous laresponsablités des usages des services exposés par l’ESB
   * Ces Tests sont (ou seront, en précisant une date cible) gérés par l'outil HP ALM (Application Life Management) Si ce n'est pas le cas, justifier une gestion des tests spécifique et préciser l'organisation des dépôts des tests, leur localisation et leurs moyens de gestion
4. Configurations d'Audits (Cast)

Les technologies suivantes qui sont utilisées chez CARDIF sont supportées par CAST: Java, JEE, Informatica, .Net (C#, ASP, VB), Cobol, Angular (.js et .ts), C/C++, Python, Php, Flex, Shell, Oracle PLSQL, SQL Server T-sql

L’application est éligible à une analyse CAST si elle utilise au moins l’une des technologies ci-dessus

* + Sans objet <si Cast ne s’applique pas, ne garder que ce point>
  + Analyse et résultats d'analyse des fichiers sources
  + Ensemble des moyens (versions et paramétrages spécifiques) et résultats d'audits qualité

1. Configurations de services

Quand l’Application, en tant que cliente d’un Service, **doit s’adapter pour être en phase avec ce Service qui évolue**, il est nécessaire d’associer le n° de version du service avec le n° de version de l’Application cliente = dépendance forte. Cela peut n’être qu’un simple label sous Git, portant le n° de version du Service, étiquette posée sur le commit versionné de l’Application

* + Sans objet <si il n’y a pas dépendance de services, ne garder que ce point>
  + Certains serices exposés peuvent dépendres d’autres services ne faisant pas partie de l’ESB compléter si c'est le cas par "Dont la dépendance aux évolutions des modes et des formats de sollicitation de ces services nécessite de les gérer en configuration" ou pas => justifier ce cas contraire
  + Ces services sont les suivants: supprimer cet item si inutile

Lister dans le tableau le cas échéants chaque service et leurs sollicitations, sinon supprimer le tableau

|  |  |
| --- | --- |
| Service | Sollicitations |
| Service d’authentification SESAME | Token de connexion |

**REMARQUE :**

**Le PGCL ne décrit QUE**

* **La Gestion des Configurations de développement Logiciels,**
* **Avec ses relations avec les autres configurations utilisées (lecture/écriture),**

(Ces autres configurations ayant des règles de gestion non décrites ici (CLC, TCG, C'You, ALM) mais si possible référencées au cas par cas)

### Gestion de Configuration des Environnements (GCE)

Une liste générique de configurations d'Environnements est proposée :

1. Configuration documentaire de l'environnement Projet (cadre de spécifications),
2. Configurations de l'environnement de développement et de TU
3. Configurations des environnements de tests
4. Configurations des environnements de Production

**Indiquer ici la prise en compte (ou pas!) en configuration de ces environnements**

* Pour chacun de ceux non pris en compte, ne pas l'enlever de la liste mais commenter "Environnement non géré en configuration par l'Application"

Pour les environnements gérés, décrire si besoin (compléter les items déjà proposés) succinctement:

* Les objectifs de la gestion de configuration de l'environnement et éventuellement globale des environnements partagés (gérer la cohabitation des applications)
* Leurs éléments communs, le relevé des cohérences entre ces configurations,
* La traçabilité des évolutions, les pivots de flux unique et commun – données dont le format est sous la responsabilité de l'application créant ces données –
* Le cas échéant, les besoins de contrôles des configurations transverses inter-applications (selon des adhérences inter-applicatives).

Préciser aussi les contraintes appliquées à des parties prenantes externes à Cardif, pour accéder aux environnements et référentiels de développements et de tests.

1. Configuration documentaire de l'environnement
   * Ni les DI « projet » ni le plan de déploiement n’ont vocation à être géré en GC par défaut. Dans le cas contraire, expliquer comment
   * Par contre, un DI de type applicatif a vocation à être géré en configuration Dans le cas contraire, expliquer pourquoi
2. Configurations de l'environnement de développement et de TU (Tests Unitaires)
   * Modèle commun d’un framework déployé sur chaque poste de dévelopements
   * Le RA est responsable de ce modèle et de ses évolutions, gérées en configuration dans Git (dépôt de type Framework)
   * Profils ODIN fournissant les listes des prérequis d’outillage
   * Complément aux profils ODIN selon les spécificités des projets/maintenances
   * Rassemble caractéristiques des plateformes techniques (exemples : ETL-MUTU, BO-MUTU, SAS-MUTU, Oracle…) utilisées par l’application dans l’environnement Projet/Maintenance et leur version associée
   * Ce type de configuration est réparti et géré dans plusieurs référentiels (ODIN, …)
3. Configurations des environnements de tests
   * Le contexte logiciel et matériel utilisé pour les tests n’a pas besoin d’être suivi en configuration car le contexte est relativement stable et ses évolutions sont gérées en configuration par les usages impliqués (exemples : Informatica, BO, SAS, CFT, Control/M, Oracle…)
   * Suivre les cas où les données sont générées pour des tests particuliers (moyens d'extraction, de filtrages, d’anonymisation,). La traçabilité des modes de génération de ces jeux de données doit être assurée
   * Suivre les changements dans le code et le paramétrage des outils de tests développés spécifiquement. <Ou bien préciser qu'il n'y a pas d'outils de tests développés spécifiquement>. Ces suivis doivent à minima s'appuyer sur le numéro de version que l'outil teste (version du script de tests – identifiée par le numéro de la version applicative testée -, parfois associé à la version de l'outil de test)
   * Ces informations sont à préciser pour chaque test et/ou campagne de tests
   * Gérer les évolutions des scripts de tests de performance et leurs résultats

Les configurations d'environnements doivent permettre principalement d'identifier des causes extérieures à l'Application, pour des défauts de construction ou de comportement apparaissant malgré la reconduction à l'identique des configurations de développement et de tests.

Les gestionnaires de l’environnement projet/maintenance (SOD et IPS) doivent respecter les mêmes règles de transparence et de communication sur l’état et la gestion en version de leurs configurations d’environnement.

1. Configurations des environnements de Production <ou bien n'est pas géré>
   * Tous les paramètres de l'environnement de Production
   * Sa composition technique et logicielle
   * Ces états sont à relever pour la MEP et à comparer pour chaque défaut de fonctionnement

## Eléments à organiser

### Types & identifications des éléments dans les configurations

Selon les configurations listées précédemment (Cf. § I.A.3)

* Lister les types d'éléments suivis en configurations dans lesquelles ils sont rassemblés et gérés.
* **Ne garder que les items appropriés** au contexte à gérer et les adapter si besoin vis-à-vis de ce contexte Applicatif

1. Configurations de développement :
   * **Fichiers source Java**
     + La gestion des fichiers sources Java est géres au niveau framework déléguée aux équipes du CDEV.
     + Les composants Java appelés par l'ETL sont gérés dans le système d'arborescence Unix (que la CLC ou la TCG peut aussi prendre en charge)
     + Les composants Java d'IHM Web (ou batch Java) sont gérés comme tous les développements WAS que gère la CLC ou la TCG <l’une ou l’autre> (construction et déploiements)
   * **Fichiers source Unix/Linux**
     + Les fichiers source Unix (ksh, sql…) des traitements de l’application (répertoire $CONF\_APPLI et ses sous-répertoires, répertoire $SOFT\_APPLI et ses sous-répertoires)
     + Les fichiers source Unix (ksh, sql…) des traitements de déploiement de l’application
     + Les fichiers de profil Unix de tous les environnements et couloirs concernant l’application
   * **Fichiers Oracle**
     + Scripts DDL de livraison mettant à jour le schéma (tables, vues, procédures stockées, packages …) de la version n-1 pour obtenir la version cible n
     + Scripts DDL du schéma cible en version n (tables, vues, procédures stockées, packages …), qui est un relevé après exécution du script de livraison mettant à jour le schéma depuis la version n-1)
     + Requêtes externes
     + Certains paramètres sont applicables à chaud (donc associables au déploiement de la nouvelle version de la base) d'autres nécessitent un reboot donc concernent la GCE (taille disque, gestion mémoire, logs, mises en cache, services d'accès, jeu de caractères, partitions …)
2. Configuration de construction et de déploiement

Récupération/Mise à jour des données de configurations externes (fichiers mutualisés…). Ces configurations ne sont pas gérées ni en version ni en contrôle d’accès, mais historiées (historisation non systématique) et sauvegardées dans le référentiel GCE. Se rapporter à la Règle de paramétrages des environnements [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

1. Configurations de documents
   * Documents liés à l'Application, et à ses évolutions (Avant-Projet, Projet, Maintenances)
   * Sont gérés en configuration dans le site C'You nécessaires : (C'You Application, C'You Projet, C'You Avant-Projet, pas forcément tous = > ne garder que ceux réellement existant ou bien créés dans un futur proche)
   * Se conformer à la stratégie documentaire du patrimoine pour les documents Applicatifs
   * Se référer au site Harmonie pour tous les documents Projet Cf. § III.B Références Documentaires)
   * La seule obligation prévue par le PGCL est la cohérence des versions documentaires, applicatives et tests
2. Configurations de Tests (Normalement listées dans la Stratégie de Test Applicative – particulièrement indispensable pour les TNR ! –. Si oui, alors la référencer ici et supprimer les items de ce paragraphe en justifiant la non redite)
   * Gérer par les usages
   * Exigences de test + Ordonnancements
     + Stratégie de tests
     + Spécifications de tests (issues du C'You Projet ou <Application> pour les tests de non régression – TNR -) en phase avec les spécifications de développement)
     + RBT (Risk Base Testing) = risque produit (fonctionnel et non fonctionnel) + risque technologique
   * Tests + Données de tests + Scénarios + Campagnes module "Cycle & Releases" d'ALM
     + Plan de tests de vérification applicative (VA)
     + Plan de tests de recette
   * Résultats Selon le vocabulaire de référence
     + Traces des Tests Unitaires (log, traces d’exécution…)
     + Compte-Rendu de tests de VA
     + Compte-Rendu de tests de recette
     + PV de recette
3. Configurations d'Audits :
   * Sans Objet <à ne garder que si CAST n’est pas solicité>
   * CAST : analyse formelle, propre au patrimoine, amendable, par application

Une règle décrit comment les traitements génériques de Cast permettent la persistance de l'identification des éléments analysés [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)]

* + Les fichiers sources fournit à CAST doivent obligatoirement avoir auparavant été identifiés dans Git par un même label de version (posé sur un Commit d’un ou plusieurs dépôts contenant ces sources de la version analysée)
  + Pour les modèles de qualité formelle ou fonctionnelle du produit :
    - Mesures brutes
    - Résultats (mesures interprétées)
    - KPI (seuils) Projet ou maintenance des RAQ

1. Configurations de Données Pivot :
   * Sans Objet <à ne garder que si il n’y a pas de données pivot>
   * Format xsd (« <nom objet>\_<statut objet>.xsd ») des données du fichier Pivot (flux standard) et sa version intrinsèque (dans une balise XML) du modèle du format
   * Remarque : les flux techniques ne sont pas des données Pivot
2. Configurations de l'environnement de développement et de TU :
   * Liste des versions des outils utilisés
   * Liste des versions des plateformes utilisées [future norme]
   * Autorisations d’accès gérées par ODIN/WGD
3. Configurations des environnements de tests :
   * Hors scope
   * En fonction des applications accédées :
     + Les versions
     + Les connexions inter-applicatives (CFT…)
   * Les standard Unix (modèle d’arborescence…)
   * Référentiel à créer <A PRECISER>
4. Configurations des environnements de Production :
   * Configuration externe gérée par ITSM (+ idelphes)
   * Les logs d’installation [Futur]
   * Outil : Portail 360

### Éléments externes liés

Si des types d'éléments sont déjà gérés dans des configurations externes, il faut préciser ici que cette gestion sera (ou pas!) réutilisée donc liée. Par exemple, la gestion documentaire, des binaires, des tests. (Le contexte technique des outils associés sera détaillé dans le § 3. Migration de configurations)

Les configurations documentaires, de tests et d'environnements sont gérées par des référentiels spécialisés.

Les formats, les règles de gestion et l'organisation de ces référentiels sont des contraintes de l’application qui délègue ces gestions de configurations spécialisées.

Cependant, des règles d'accostages existent afin de labéliser/récupérer des versions d'éléments stockés sur des référentiels séparés, mais appartenant à une même configuration commune (Exemple : Propager l’identifiant de la version jusqu’aux configurations d’environnements afin de le récupérer au moment des tests ou de découverte d’anomalie)

La mise en œuvre des actions de GC décrites dans le § II.D.3 doivent permettre le respect de ces règles.

## Contexte d’organisation

### Méthodes

Énoncer (simplement et référencer) ici les règles du cadre méthodologique

Ce sont les Règles de

* Documentation,
* Développement
* Construction
* Test
* Déploiement

Elles doivent être appliqués selon leurs incidences avec la gestion des configurations. Par exemple,

* Les obligations de cycles de documentations liées,
* Un mode de développement (V ou Agile),
* Une formalisation de couverture de tests,
* Une parallélisation des tests,
* Un processus d'intégration continue,
* Des actions et des actes associés aux promotions des versions,
* Les sous-étapes des méthodes projet appliquées incluant un acte de GC, etc…

#### Mode de développement :

Ce qui oriente la GC : supprimer les items inutiles, en ajouter d'autres si besoin

* Stratégies projets/évolutions correctives
* Mode de développement en cycle en V.
* Mode de développement Agile.
* Projets/évolutions techniques

Eviter si possible les modifications multi-technologiques simultanément au sein d’un même projet/évolution : modification d’une technologie à la fois (mode asynchrone) !

L’objectif est de faciliter l’identification des causes des anomalies rencontrées et éviter les big-bangs d’intégration. Quand c’est possible, privilégier les MEPs individuelle par technologie. (Pas de "BigBang")

Cette méthode peut induire un lotissement des projets/évolutions multi-technologiques. Exemple : tuilage de projets ou de lots intra-projet. Induit une politique de branches par Projet/lot.

Projets/évolutions fonctionnelles :

* Les développements sont organisés par domaine fonctionnel et/ou fonctionnalité
* Politique de branche par projet/évolution puis par fonctionnalité ou domaine fonctionnel

#### Méthode de documentation

Le mode de fonctionnement recommandé par C'You est l'utilisation exclusive des référentiels supprimer les items inutiles, en ajouter d'autres si besoin. Si sans C’You propre pour certains, préciser les modes non exclusifs.

* Avant-projet,
* Projet,
* Maintenance
* Application

Prévus à cet effet, à l'exclusion de toute étape intermédiaire (Partage réseau, Wiki, …)

Se rapporter aux Règles de Gestion Documentaire [[1]](#footnote-1)

Dans le cas où C'You, n'est pas utilisé, justifier d'une méthode de documentation officielle et donc validée par le RPA

Choix INTERDIT :

* D’un répertoire bureautique pour le développement de la version n+1 de la documentation
* D’un répertoire bureautique pour la récupération en lecture seule de la version n (suffixe du fichier selon propriété intrinsèque « version de configuration » renseigné par C’You)

#### Méthode en relation avec les technologies utilisées :

Certaines méthodes sont des règles déjà/bientôt référencées => localisation précisées au Chapitre III.B Références Documentaires

* + - Versionnement transverse :

Le workflow de développement impose des versions par technologie restant internes aux développements et une version applicative imposée aux tests et à la documentation (Baseline)

* + - Unix

Les développeurs développent les composants Unix sur leur poste de travail

Les tests unitaires (TU) peuvent se réaliser que sur les postes de dévelopements. Cela impose la synchronisation de ces tests entre les différents développeurs. Cela implique soit des TU commun, soit des TU individuels séquentiels

Pour des raisons de sécurisation du code, les développeurs doivent sauvegarder une fois par jour (à repréciser comment dans la partie Plan de ce document)

* + - Informatica

Le contexte de travail et les méthodes de GC associées à cette technologie font l'objet d'une règle générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

* + - Control/M

Le contexte de travail et les méthodes de GC associées à cette technologie font l'objet d'un règle générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

* + - BO

Les binaires BO (Biar) de contenu et de structures ne peuvent être gérés en configuration qu'associés à la référence de la base de données applicative pour lesquels ils sont adaptés.

Les connexions à la base de données applicative sont spécifiques à l'environnement. Le Biar ne change pas, mais cela nécessite des paramétrages post livraison. <Indiquer si ces paramètres sont manuels ou à intégrer dans un fichier de paramétrage géré en configuration>

Les droits d'accès précisés par le Biar de structure peuvent être aussi modifiés post livraison (et de plus les droits peuvent être spécifiés dans WGD)

* + - SAS

Les fichiers programmes SAS contiennent des développements et des requêtes.

Les fichiers programmes SAS sont exécutés avec des enchainements précisés dans des fichiers d'ordonnancements.

L'organisation des fichiers SAS est <méthode d'organisation>

Le mode de développement des fichiers SAS est <méthode de développement>

Des fichiers programmes sont communs à plusieurs ordonnancements

* + - Oracle

Les scripts de transformation de schéma sont à gérer en configuration, associés aux requêtes dépendant de ce changement de schéma <préciser les cas particuliers>

* + - JAVA
      * Soit générés automatiquement à travers l’ihm TIBCO (TIBCO Designer)
      * Soit directement par un éditeur Java

<Préciser si les développements Java nécessitent des méthodes spécifiques allant au-delà des référencements et fonctions de bases offertes par l'outil de GC Git>

#### Principes de versionnement des configurations :

**Les numérotations de versions sont applicables identiquement, toutes technologies confondues participant à la même version applicative**.

La règle de versionnement [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx) distingue les niveaux d'évolutions,

* Majeures pour des évolutions de l'application dans le cadre d'un projet,
* Mineures pour des maintenances,
* Un suffixe final (indice) pour les changements unitaires pour un projet ou une maintenance.

#### Occurrences de modification des configurations :

Les modifications et validations des éléments sont encadrées par des méthodes.

Ces méthodes sont celles définies pour développer, construire et déployer.

Elles sont appliquées à travers des étapes et sous-étapes précisant les cas de ces modifications et validations de ces éléments. Les cas expliquent les raisons et conditions d'applications (appelées aussi occurrences) donc impliquant ces actions sur les configurations.

Des actions de GC y sont associées selon les acteurs. Ces actions sont décrites dans le § II.D.3.

Des règles encadrent certaines des pratiques [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

#### Constats de progrès :

Définir ici ce qui doit être vérifié par la Gouvernance

Cette Gouvernance est menée par le RGC

Ce sont les éléments à estimer pour savoir si le dispositif de gestion de configuration tend vers l'efficacité car répondant de mieux en mieux aux besoins GC exprimés. Donc des constats de progrès. Les moyens de cette Gouvernance (l'estimation des éléments décrits ici) sont précisés dans le § II.D.2

* Le bon fonctionnement passe par le respect de règles de gestion des configurations, règles assistées par les outils de GC, eux-mêmes ayant des règles s'accès à ses règles
* La revue formelle de GC permet par exemple de vérifier l'application du dispositif de GC prévu dans le PGC. La création centralisée des numéros de versions et les moyens d'en hériter sur les autres référentiels de GC est un des éléments forts de la gouvernance.
* Cela doit s'accompagner par exemple par des droits d'accès en lecture très ouverts sur les configurations mais assez restrictifs en modifications
* Préciser aussi les capacités à comparer des versions successives d'une configuration

La liste simplifiée proposée ci-dessous n'est pas exhaustive et peut reprendre des points ci-dessus et aussi préciser certains aspects à contrôler impliquant des tierces parties :

Les éléments utiles à analyser pour constater les progrès sur les pratiques de GC sont :

* Les moyens et méthodes mis en œuvre pour gérer les configurations
* La traçabilité des configurations, leurs identifications, complétudes et cohérences
* Les organes de décisions, d'information et de surveillance du dispositif de GC

### Sous-traitances

Si des parties prenantes sont extérieures à l'organisation – par exemple une TMA, une TRA

* Préciser leurs contraintes (à contractualiser) vis-à-vis de la gestion de configurations encadrée par ce document. Typiquement
  + Les obligations de moyens associés (accostages des gestions de configuration du client et du sous-traitant) et/ou intégrés (modalités d'accès à la gestion de configuration du client)
  + Les obligations de réversibilité.

Sans objet

Une TMA aura au minimum l'obligation de respecter 2 contraintes :

* La capacité de réversibilité : moyens prouvés de reverser les éléments gérés par la TMA dans les référentiels de gestion de configurations du client, selon les organisations et les formats du client. <Inutile si la TMA utilise les dépôts de l'application gérés par Cardif>
* L’usage de la CLC: l'utilisateur interne ou externe doit respecter certaines contraintes des outillages de cette chaine de traitements. Ces contraintes sont contractualisées : item et sous items à supprimer si CLC non utilisée
  + Processus de construction
  + Référentiels associés
  + Sources de constructions à l'extrémité de branches d'un référentiel Git
  + …
* L’usage de la TCG <décrire les contraintes spécifiques à l’Application dans ce cas, vis-à-vis de l’exploitation des fichiers par la TMA>
* Les droits et devoir des parties prenantes sont expliquées au § I.A.2

### Migration de configurations

**Dans le cas d'une migration de configurations existantes vers Git,**

* Préciser la localisation actuelle des éléments à organiser. Attention, le terme de configurations pour un existant est parfois trop avancé, car cela peut se résumer qu'à cette localisation sans forcément une organisation en GC
* Cette localisation doit être accessible par les personnes chargées de cette migration (SOT)
* Préciser ce qui doit être récupéré, transformé, migré, lié,
* Préciser les règles de gestion, reconduites ou non. = les dépôts Sources dans le cas de la migration vers Git
* Lister les dépôts et mettre en face de chacun, le dépôt Git receveur
* Dans le cas où un dépôt source est réparti vers plusieurs dépôts GIT, préciser la règle de migration pour sélectionner les fichiers à récupérer :
  + - * En compréhension (type et/ou taxon commun des noms de fichiers)
      * In extenso (liste exacte des fichiers, chacun associé au dépôt Git cible)

La description Source-Cible est indispensable dans ce §, que les dépôts sources soit des dépôt CVS, ou bien de simple dossiers; ou même encore une structure de fichiers dans un outil de développement

Le « comment migrer » sera expliqué dans la partie Plan (§ II.A.1 et II.A.2)

Les configurations du projet existent déjà, dépôts gérées par l'outil CVS revoir cette assertion s’il n'y a pas d'outil de GC, ou bien si celui-ci n'est pas CVS (Excel par exemple)

Ces dépôts <"dossiers"> sont listés ci-dessous avec le dépôt Git cible et les règles de migration :

|  |  |
| --- | --- |
| Emplacement des fichiers à migrer dans Git  (et éventuellement la règle de migration) | Dépôt Git cible (du § A.3.1) |
| Localisation Source : <URL, dossier> <Dépôt CVS> | [Code REFI]/[Groupe]/<Technologie>.<artefact\_id> |
| D’autres localisations non CVS sont possibles. |  |
| La récupération des élements de l’application ESB se fera en bînome sur un poste Tibco, pour que SOT puissent accéder au partage sécurisé des fichiers à migrer, donc avec l’assistance du RGC de l’Application | AP10792/ESB/services/tibco.pbc\_service\_<nom du service> |
| AP10792/ESB/containers/tibco.pbc\_container\_<nom du container> |
| AP10792/ESB/libraries/tibco.pbc\_library\_<nom de la librairie> |
| AP10792/ESB/core/tibco.pbc\_core\_<nom du core> |
| AP10792/ESB/framework/tibco.pbc\_framework |

Le contexte de travail et les méthodes de GC associées aux migrations de configurations sous Git font l'objet d'un règle générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

## Contexte technique

### Moyens techniques de gestion de configurations

On explicite ici les conteneurs des éléments gérés en configuration et les gestionnaires de ces éléments,

Lister les outils dédiés à la gestion de configuration et leur contexte technique. Inclure les outils de création et d'utilisation des éléments, ces outils devant être liés aux gestionnaires de configuration.

Le § I.A.3 a listé **les types configurations** à gérer, le § I.B.1 a listé **les éléments à y organiser**, faire ici la relation entre ces configurations peuplées et les référentiels techniques les accueillant.

Préciser les moyens de sauvegardes et les conditions de continuité de service (principalement dans les cas de GC distribuée – documentation, spécifications, sources, binaires, tests –).

Par défaut, il n'y a rien à modifier dans ce paragraphe, si Git, C'You et ALM sont utilisés de façon standard.

Les moyens techniques de GC sont à la fois les conteneurs des éléments gérés en configuration et les gestionnaires de ces éléments, procédures plus ou moins complètement outillées. Ces deux rôles sont généralement assumés par un outil de GC :

* SharePoint : outil de gestion documentaire (avec un PGD décrivant ses liens avec cette GC)
* Gère automatiquement la préemption dans la modification (Check-out), un incrément de version documentaire (Check-in), des champs complémentaires (métadonnées), un workflow d'état
* Deux instances sont utilisables pour la GC : C'You Application et C'You Projet. Se référer au PGD C'You pour l'utilisation des référentiels documentaires (Cf. § III.B Références Documentaires)
* Git : outil de gestion configuration de fichiers : contient les configurations de développements. Une base commune de référence pour tous les projets est la solution GitLab qui est installée sur un serveur VM de test (solution provisoire) et en définitive sur le serveur S00VL9968035 redondé sur le S00VL9968012, sauvegardée par IPS et BP2I (sauvegarde journalière incrémentale)
* HP ALM : outils de gestions des tests. Un PGC spécifique doit décrire ses liens avec cette GC.

NB : Des informations récupérables de référentiels de configurations techniques ne sont pas pris en compte – typiquement des paramétrages d'environnement tels que des identifiants et/ou caractéristiques de flux CFT – car ils seront gérés dans le cadre de la GCE. Par exemple ODIN/WGD, ITSM, IDelphes, Clarity, Formel, Portail 360°

Le rapprochement entre les moyens techniques de GC et les configurations gérées est précisé dans le § II.A.2. Il est possible que des éléments d'une même configuration soient gérés séparément sur deux référentiels.

### Dimensionnement des moyens

Jauger la volumétrie d'accueil des configurations et de leurs éléments, actuelle et future.

Préciser les capacités et contraintes d'accès aux moyens de gestion de configurations, selon droits et devoirs (lecture seule, intervenant extérieurs principalement)

Le dimensionnement des moyens de développement dans le cadre de la CLC/TCG <l’une ou l’autre> (besoins des référentiels associés aux capacités techniques des bases de données les accueillant, pour GitLab, Delivery Manager, Nexus, ARA, Jenkins, Maven, ALM, C'You, autres outils de la TCG à préciser ici…) est réputé par leur gestionnaire pour s'adapter aux besoins et performances des applications et de leurs utilisateurs.

### Adaptation du contexte technique

Si le contexte par défaut de l'organisation actuelle ne satisfait pas les ambitions de la gestion de configurations envisagée (actuelle et future), décrire les adaptations et évolutions nécessaires, dont celles possibles selon certaines conditions (occurrence, contexte, coûts, délais, ressources).

Le contexte technique est globalement satisfaisant</insatisfaisant>, excepté préciser les exceptions (par exemple ne disposer que d’un C’You Application) en négatif si le satisfaisant est majoritaire, en positif si l'insatisfaisant est majoritaire

# Plan de Gestion de Configurations

Le QUOI gérée en configuration a été décrit dans le chapitre Stratégie

Le COMMENT gérer en configuration est décrit dans ce chapitre Plan

## Organisation des éléments

### Agrégations et Ségrégations

Selon les types d'éléments à organiser (Cf. § I.B.1), expliquer la logique de leurs agrégations en configurations, selon les types de configurations nécessaires (Cf. § 0). Par exemple des configurations distinctes selon des techniques et/ou des objectifs fonctionnels.

Les configurations de tests sont partagées entre les C'You et ALM :

* Les spécifications sont héritées du C'You Projet vers ALM
* Les résultats de tests sont transmis d'ALM vers le C'You Projet et/ou le C'You Application selon le type de tests (tests d'intégration Projet, tests de régression de l'Application)

Décrire et expliquer l'organisation interne des configurations selon une arborescence standardisée de sous-dossiers spécialisés (par exemples, une reproduction de l'arborescence cible du déploiement, des sous-dossiers pour les spécifications, les fichiers sources séparés par technologies, les paramétrages, les tests et leurs résultats, la documentation, …)

La composition standard des configurations documentaires gérées par C'You sont définies dans le PGCD C'You pour l'utilisation des référentiels documentaires (Cf. § III.B Références Documentaires) les documents.

Les développements dans le monde Unix nécessitent des arborescences des sources globalement organisées à l'identique de l'arborescence du serveur. En effet cette arborescence est utilisée à la fois pour développer et tester directement sur le serveur Unix dans les conditions de la structure Unix cible.

Cette organisation est décrite de façon générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

Si nécessaire, préciser les cas d'exclusions (selon l'acteur ou la phase projet) de certains éléments (accès limités à certains sous-dossiers, éléments utilisables mais non modifiables, existences transitoires, …), l'emploi de filtres de discrimination techniques (séparer des technologies, des types de documents, …)

### Règles de gestion des configurations

Lister les configurations gérées et/ou accédées (Cf. § I.B.2), leurs contenus (logique), **leurs localisation techniques,** leurs liens, leurs utilisateurs et gestionnaires, leurs occurrences d'exploitation. Préciser les liens d'accès aux configurations

Les configurations de développements sont accessibles et gérées selon les règles génériques [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

L'URL d'accès au dépôt GitLab est standard pour tous:

<https://cip-gitlab-assurance.is.echonet/>

La configuration de test correspondante sur ALM est identifiée selon des règles génériques [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

### Règles de nommage

Enoncer les normes pour désigner les configurations. Cela sera très lié à leurs contenus et aux cas d'utilisations pour permettre une identification associée aux usages exprimés (Cf. § 0)

Attention ne pas confondre versionnement et nommage des configurations, les deux étant associés pour désigner le concept de "Version" banalisé entre le contenu et une instance spécifique de ce contenu

Les configurations gérées par Git sont identifiées selon des règles génériques [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

Les configurations liées à un projet ou à une maintenance se distinguent par une politique de branches, les éléments participant étant spécialement initiés puis développés sur une branche spécifique de versions (Cf. § II.C).

## Politiques de Versions et d'Identifications

### Occurrences des historisations et restaurations

**Encadrer les cas d'historisation** des configurations

– Normalement doit apporter une valeur ajoutée en termes de complétude et/ou de cohérence potentielle entre les éléments de la configuration

–. Indiquer ces cas permettant de **faire avancer l'état qualitatif de la configuration** (capacité de l'ensemble de ses éléments à spécifier un besoin, à construire une solution, à valider la satisfaction du besoin, …)

Signaler **si certaines sauvegardes peuvent êtres techniques** (sans objectif de complétude ou de cohérence) afin de centraliser toutes les sauvegardes. Justifier ou interdire (possible pollution du référentiel)

Important : l'objectif d'une historisation n'est pas une sauvegarde, mais le contexte technique peut obliger une historisation pour faire une sauvegarde. Voir ci-dessous

Les **cas de restaurations** d'un contexte doivent être justifiés par une situation : retourner à une version stable, profiter d'une avancée annexe, …

Il existe des obligations de pré-identification des packages dans des fichiers pom.xml, qui sont expliquées dans le document Recommandations et Normes pour les ETUDES (Cf. §III.B)

Dans le cas d'Informatica <supprimer cette ligne si le projet n'est pas concerné>, la gestion de version des éléments, interne à cet outil n'est plus utilisée. Une gestion de versionnement des Folders fait l’objet d’une règle spécifique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

Le RefGC-Local sous Git permet des historisations des fichiers des configurations de développement, avec une obligation de cohérence qui n'est vérifiée que par les TU.

Les configurations transférées du RefGC-Local vers le RefGC-Central, sont potentiellement utilisables pour participer à une solution Projet ou Maintenance testable.

Le RefGC-Local n'étant pas à priori sauvegardé (sauf si RefGC-Local sur U: avec des moyens et volumétrie de sauvegarde dimensionnées de manière adéquate), seul un transfert des configurations de développement sur le RefGC-Central permettront une sauvegarde de celle-ci. Attention ces configurations ne sont pas exploitables car non garantie. Une possibilité est de disposer en Local d'une branche spécifique aux sauvegardes reportées sur le GitLab

Les développements dans le monde Unix <supprimer ce § si le projet n'est pas concerné>, se font soit sur un RefGC-Local d’un PC, soit à travers un RefGC-Local sur le serveur Unix. Ce qui implique que les développeurs peuvent s'y relayer en y transférant leur branche personnelle depuis le RefGC-Central (en n'y conservant aucun des fichiers du précédent occupant ou bien se coordonner avec lui). L'organisation des dépôts Git pour développer dans le monde Unix fait l'objet d'une règle [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

**Cas de restaurations possibles** (dans le sens origine "**Depuis** …" vers destination "**Sur**…") :

1. Sur le Working Directory (WD) depuis le RefGC local :

* Pour poursuivre son travail (le WD pouvant être sur Windows ou Unix)

1. Sur le Working Directory (WD) depuis sa sauvegarde physique traditionnelle

* Ne donne que l'état physiquement sauvegardé du WD (dont le RefGC local)
* Si on veut récupérer un état logique (état du dernier commit), mieux vaut repartir de son RefGC local (cas n°1), ou bien compléter par un check-out

1. Sur le RefGC-Local depuis le RefGC-Central:

* Pour ré-initier son RefGC-Local
* Pour succéder à un autre développeur sur un WD Unix commun

1. Sur le RefGC-Central depuis les sauvegardes physiques du GitLab

* Pour ré-initier le RefGC-Central techniquement perdu/corrompu

### Règles de versionnement

Présenter la norme d'identification des états successifs sauvegardés d'une configuration.

Versionnement des sources : gérés sous Git®, un n° de version original est associé aux éléments à chaque sauvegarde (au commit). Cependant, ce n° est commun à tous les éléments sauvegardés au même moment depuis l'espace de travail du développeur.

Une version globale applicative est numérotée selon la même règle [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

Le moyen outillé respecte le besoin exprimé §C.1.d

Présenter la norme d'identification des versions d'application, de projet et d'évolution, norme qui doit intégrer des règles d'incréments. Si besoin la compléter pour qu'elle soit applicable à cette gestion de configurations. Détailler le processus, la procédure, le moyen et les droits de ces versionnements.

En cas de d'application multi technologiques, expliquer la façon de versionner des évolutions spécifiques à une technologie et l'utilisation de version transverse à toutes les technologies.

Dans le cas d'un outil de développement, le mieux est de le dessaisir (si possible) de ses prérogatives de versionnement, ou bien au minimum du choix et de la syntaxe des numéros de versions et du choix des occurrences d’identification par des numéros de version.

Les règles de nommage et de versionnement des éléments documentaires sont décrites dans le Plan de Gestion de Configuration Documentaire Projet et Patrimoines PGCD (Cf. §III.B).

L'application d'un numéro de version sans le suffixe '–snapshot' à une configuration de fichiers source se fait en cas de complétude et de cohérence prouvées de cette configuration, c’est-à-dire la capacité à minima la construction d'une solution testable déployable appelée "release".

Le versionnement des éléments et configurations (utilisés en aval de la création des éléments : packages et applications) est une règle générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

Ce numéro de version Applicative est alors propagé aux autres configurations partageant la même cohérence, au sein du gestionnaire de sources. C’est-à-dire les dépôts Git locaux qui réimportent les étiquettes associées du référentiel central.

Les configurations applicatives gérées par des Référentiels externes au Projet (typiquement des fichiers de paramétrages Cf. § I.B.2) mais gérés sous Git, perçoivent la même propagation de ce n° de version. Ce versionnement concerne aussi les artefacts de tests associés (cycles, scénarios, tests, résultats de tests, données de tests, défauts constatés), mais non gérés sous Git.

### Surcharge d'identification

Encadrer l'usage d'identificateurs complémentaires aux étiquettes de versions :

Par exemple : Environnement où est actuellement la version (label "glissants" car à reporter sur la prochaine version occupant cet environnement).

Autre exemple, la version d’un kernel (binaire non modifiable) provenant d’un éditeur, version qui a été utilisée pour valider des développements Cardif associés. Donc deux labels sur le même commit : celui de l’éditeur et celui des sources Cardif

Décrire éventuellement un workflow qui permet d'agréger des éléments selon des règles de gestion formalisées par des labels d'états,

Une version de composant (package, document, …) utilisé (pas forcément livré) identiquement pour plusieurs versions applicatives doit être identifiée comme associé aux labels de ces versions. Cela est possible en cumulant les labels (Tag Git) sur ce même composant.

Les gestionnaires de configurations accostées (Documentation, Tests, Environnement) doivent permettre de gérer la version applicative commune selon les mêmes occurrences. Politiques de Branches

### Occurrences de création et d'utilisation de branches

Exposer comment appliquer des évolutions différentes et simultanées à une configuration (tous types d'éléments sont concernés), nécessitant des modifications parallélisées sur des branches divergentes.

Lister ces besoins (plusieurs modificateurs sur la même évolution, même modificateur sur plusieurs évolutions, …) par type d'éléments, préciser les occurrences et les acteurs (créateur, utilisateur. Cf. § III.A)

* + - **Référentiels de développement**
* Référentiels locaux : chaque développeur développe ses sources dans un répertoire de travail géré en configuration par une base Git dite "locale" installée sur son disque local C:
* Référentiel absolu : c'est une base Git dite « de référence », commune à toutes les applications et gérée par le référentiel GitLab
* Les bases Git « locale » ne sont pas sauvegardées….
* La base Git "de référence" est redondée et sauvegardée par IPS et BP2I (sauvegarde journalière incrémentale)

Lister ici les types de branches utilisées, avec leurs occurrences de création, de fusion, de rebase, de suppression. : Branches Développeur, Branches Projets, Branches HotFix, Branche Debug, Branches de corrections avant/après MEP

* + - **Types de branches**
* Branche Master pour les versions de production
* Branche Projet pour les développements dans le cadre d'un Projet (possiblement pour plusieurs Applications, mais la branche ne concerne que les développements gérés par un dépôt spécifique à chaque Application => identité de nommage de cette branche projet pour toutes les Applications concernées
* Branche Maintenance : même logique que pour les branches Projets
* Branche des contributeurs d'un Projet ou d'une Maintenance, distinguant la participation de chacun des développeurs. Il est de la responsabilité du contributeur de récupérer de la branche Projet/Maintenance les fichiers nécessaires à ses développements (tout ou partie d'un des commits de la branche Projet/maintenance, sur sa branche personnelle pour le projet dans le GitLab, puis ensuite sur sa branche équivalente de son Git Local).

Cette gestion des branches fait l'objet d'une règle générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

D'autres branches peuvent être créées temporairement et localement

### Traitements des Fusions

Selon les occurrences de choix d'évolutions réalisées en parallèle, présenter les obligations et circonstances de convergence vers une solution commune (fusions au cas par cas).

Présenter les types de fusions et pour chacune, expliquer ses objectifs, ses circonstances, l'aptitude de chaque branche à y participer, sa localisation (sur quelles branches de quel dépôt Git) , ses méthodes (1 à 1, Big Bang, unilatérale, consensuelle, discriminée, …), les actions, les droits et responsabilités de l'acteur réalisant le type de fusion (Cf. § III.A).

Les cas standards sont présentés ci-dessous :

La fusion (appelée aussi "merge") est réalisée par un Mergeur (cf. I.A.2, II.D.3, III.B) :

* + - De la branche d'un contributeur vers la branche Projet (ou Maintenance) associée à son activité de développement, est initiée par le contributeur et se réalise sur la branche du contributeur
      1. Récupération dans la WD du contributeur de le dernière version de la branche Projet(ou Maintenance)
      2. Choix dans cette WD des fichiers fusionnés
      3. Report sur la branche Projet(ou Maintenance) de ces choix (fusion unilatérale)
    - D’une branche Projet sur la branche Master est initiée par le RTC, et se réalise toujours sur la branche Projet
      1. Chargement de la WD du RTC par la dernière version de la branche Projet
      2. Récupération sur la WD du RTC de le dernière version de la branche Master
      3. Choix dans cette WD des fichiers fusionnés
      4. Report sur la branche Master de ces choix (fusion "simple" unilatérale)
    - D’une (de la) branche Maintenance sur la branche Master est initiée par le RTC, et se réalise toujours sur sa branche Maintenance
      1. Chargement de la WD du RTC par la dernière version de la branche Maintenance
      2. Récupération sur la WD du RTC de la dernière version de la branche Master
      3. Choix dans cette WD des fichiers fusionnés
      4. Report sur la branche Master de ces choix (fusion "simple" unilatérale)

Cette gestion des fusions fait l'objet d'une règle générique [[Cf. Règles GC sur le C'You GCL]](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx)

## Exploitation des éléments

### Cohérences Projets & Applications

La validation d'une version d'une configuration (Cf. § B.2) intervient suite à son utilisation fructueuse dans le cadre d'un projet ou d'une maintenance. Lister les phases d'exploitation des configurations d'éléments au niveau du Projet, et de ses maintenances, puis de l'Application, avec les validations successives des complétudes et cohérences dans chaque phase.

En cas de projet multi-technologique, expliquer les cohérences locales prouvées, les éventuelles livraisons partielles, l'utilisation des labels inter-technologies.

Dans le cas d'un Projet concernant plusieurs Applications devant converger vers un objectif fonctionnel commun, expliquer comment, au moment de la phase d'intégrations inter-applications, on associe les versions de ces applications. Par exemple :

* Un tableau avec une colonne par application, une ligne par campagne de tests inter applications et à l'intersection, le numéro de version de chaque application
* Si toutes les applications sont sous Git, un tag commun de version de Projet, sur la branche du Projet sur chaque dépôt des Applications.

Types d'enchainements possibles de phases d'exploitation des éléments :

* **Assemblage puis packaging, puis livraison**
  + La CLC/TCG <l’une ou l’autre> est utilisée <option>
  + Le projet sous-traite la production de packages Java, cependant livrés en cohérence avec des packages créés par le projet sans construction <option complémentaire>
* **Packaging, puis livraison** <option à adapter>
  + Le projet package des composants Unix (.ksh, .sql, .txt, .cfg, .xml). Ces packages ont t qu'une vie transitoire jusqu'à la livraison finale en Production. En conséquence
    - Il n'y a pas de conservation de packages de versions précédentes,
    - Pas de versionnement de ces packages,
    - Pas capacité à les gérer en configuration, excepté l'unique qui sert au déploiement.
  + Cependant, dans la nécessité de livrer à nouveau un package, son stockage versionné dans Nexus est utile
  + NB: cette politique de la version unique non identifiée d'un package applicatif repose sur la reproductibilité de ce package ainsi que la rapidité de cette reconstruction (ces deux critères étant réunis par l'automatisation)
  + Ce principe est à vérifier pour les packages ETL (temps d'extraction – nombre de dossiers à explorer - pouvant être long avant la capacité de construire le package)
  + L'objectif est d'appliquer un même n° de version applicative sur tous les package participant à la livraison
  + Pour BO, comme le packaging est manuel (automatisation non opérationnelle)

La capacité à reconstruire à l'identique un package réussi précédemment est hypothétique. La conservation de packages reste indispensable, dans une logique versionnée et gérée en configuration, car il est possible qu'une livraison comporte plusieurs packages BO (fichiers BIAR) associés. Les composants de ces BIARs ne sont pas gérés par Git. Git doit gérer les BIAR de sauvegardes des dossiers BO spécifiques à l'Application (BIAR global ou BIAR de contenu + BIAR de structure) et les BIAR de livraison, éléments gérés en configurations

* + Pour SAS <proposer une solution qui peut devenir une règle générique !>
  + ControlM <proposer une solution qui peut devenir une règle !>
  + JAVA <dans le cas où la CLC/TCG <l’une ou l’autre> n'est pas utilisée>.

L'objectif est d'avoir une cohérence inter-technologique pour des raisons d'adhérences fonctionnelles entre des composants ControlM, BO, ksh, etc. Donc livrer par le même processus tracé permet de vérifier la complétude d'une version multi-technologique : version identique, complétude des packages livrés. Ce qui peut être fixé est :

* + - * Le stockage sous Git des Draft Ctrl-M applicatifs, associés éventuellement (à confirmer) à leurs paramètres de promotion
      * La prise en compte par la CLC/TCG <l’une ou l’autre> pour les transférer sur Nexus puis les livrer à la demande (via DM & ARA) <phrase optionnelle>
      * La logique de promotions Ctrl-M est appliquée lors de la livraison : draft générique livré accompagné par des scripts spécifiques associé à un paramétrage spécifique à l'environnement (paramètres de promotion déjà présents dans l'environnement :
        + Soit déployés par la CLC/TCG <l’une ou l’autre> depuis Git, les paramètres appliqués dans l'environnement cible en tant que simple exécution demandée par le DM (associer Draft et paramètre pour un environnement) <phrase optionnelle>
        + Soit en tant que paramètres saisis dans le DM <dans le cas de la CLC, sinon autre interface pour la TCG> et gérés puis appliqués par lui<phrase optionnelle>
        + Soit utiliser une IHM sur le serveur de déploiement pour appliquer les paramètres et les mettre au point
      * Le relevé des livraisons par technologies (connaissance de l'état des livraisons pour une même version sur le même environnement cible
* **Livraison directe (ni construction, ni packaging)** 
  + A priori aucun <et surtout pas recommandé car demande un contrôle manuel de complétudes et de cohérences, à travers des manifestes à établir avant livraison et à vérifier après livraison>

### Gouvernance

Préciser l'ensemble des mesures, des règles, des organes de décision, d'information et de surveillance qui permettent **d'assurer le bon fonctionnement et le contrôle du dispositif** de gestion de configuration, en vérifiant les points listés dans le § I.C.1.f Constats de progrès

La Gouvernance s'attachera à

* Mener des revues formelles sur les moyens et méthodes définies dans ce PGC.

Ces revues n'intègrent pas l'analyse des éléments gérés en configuration, leur complétude et cohérences étant assurées par l'outillage Git et ses règles de gestion techniques.

* Mener des revues techniques sur l'état technique des configurations gérées
* Analyser globalement l'organisation du dispositif de décisions, d'information et de surveillance des activités de GC
* Si nécessaire, mettre en œuvre les moyens assurant le bon fonctionnement et le contrôle de ce dispositif

Les organes de Gouvernance peuvent impliquer le RA, le RGC, le RTC et le RGCT

Une proposition de points de contrôles est disponible sur le site C’You GCL :

<https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/Uses-Cases-GCL-Git.aspx>

### Activités, Actions et Acteurs

Lister les activités de gestion et d'utilisation des configurations, avec les actions des acteurs (Cf. § I.A.2) au sein de ces activités :

* Phases,
* Activités,
* Acteurs,
* Actions de Gestion de Configurations:
  + Instanciation d'une Configuration,
  + Versionnement d'Elément,
  + Identification de la Configuration,
  + Complétude et de cohérence de la Configuration,
  + Versionnement de la Configuration,
  + Traçabilité des configurations.

Préciser pour chaque action si l'opération est automatique (donc réalisée par l'outillage) ou bien manuelle

Action de Gestion de configuration Logicielle <découvrir toute ou partie du contenu des paragraphes>

* Instanciation d'une Configuration

Création d'une structure d'accueil pour des éléments en GC. Pour Git, ce sont les dépôts à créer et à peupler à partir d'un existant. Dépôts Application à l'initialisation de l'application sous Git, dépôts Projet à partir d'une des versions du dépôt Application

* Versionnement d'Elément

Normalement inutile avec Git qui versionne que des ensembles d'éléments (ceux contenu dans la Working directory) au moment d'un commit

* Identification de la Configuration

Associée aux dépôts unitaires – repository Git - et à l'ensemble des dépôts collaborant à la livraison applicative (même racine des dépôts de l'Application) et aux livraisons applicatives des Applications collaborant au résultat d'un Projet

* Complétude et cohérence de la Configuration

Tout ce qui permet de pérenniser une complétude et une cohérence initiale

* Versionnement de la Configuration

Attention ce n'est pas l'attribution d'une étiquette de version, chaque commit Git créant une version. C’est un label appliqué à un des commit de chaque dépôt participant à une configuration

* Traçabilité des configurations

La traçabilité est assurée par les outils assurant le maintien de la cohérence initiale établie à l'entrée de l'outil, avec des fonctions d'historisations entre les différentes étapes de transformations jusqu'à la livraison. C'est l'idéal DevOps, fortement outillé

Les parties non outillées doivent passer par de la gestion de déclaratif à travers des manifestes listant tout ce qui entre et sort d'une étape. Et chaque étape doit valider (checker) ces manifestes au moment de ses réceptions et de ses livraisons vis-à-vis de ses partenaires amont et aval

Ces Actions sont replacées au sein d'Activités décrites par Phases dans le tableau ci-dessous :

| Phases | Activités | Acteurs | Actions de Gestion de Configurations | Auto et ou Manuel |
| --- | --- | --- | --- | --- |
| Développement  Construction | Choix version de départ du Projet | RA, RGC | * Création de la branche Projet depuis la version choisie sur de la branche Application | M |
| Instanciation du projet | RGC | * Instanciation de la configuration Projet pour chaque contributeur * Pose de label initial de la version projetée | M |
| Modification des sources | Développeur | * Versionnement automatique des éléments modifiés de la configuration projet | M, A |
| Référencement des sources | Développeur | * Contrôle de la complétude et de la cohérence des fichiers sources, aptes à la construction * Versionnement manuel de la configuration de fichiers sources candidats à la construction | A |
| Fusion des sources | RTC Mergeur | * Source résultant cohérent avec les objectifs cumulés des différents contributeurs | M |
| Prise en compte des fichiers sources | Constructeur | * Choix des fichiers sources * Sélection de la dernière version des sources | M, A |
| Compilations des sources | Constructeur | * Nommage du binaire créé | M, A |
| Développement  Release | Versionnement du binaires (et librairies | Constructeur | * Incrémentation du label de version attribué * Pose du label de version sur le binaire créé | A  A |
| Reconnaissance des sources de référence | RGC | * Versionnement de la configuration constructible sur le référentiel central par incrément manuel du n° de version de l'application | M |
| Reconnaissance des sources Locaux | Développeur | * Report du n° de versions des sources de référence sur son équivalent local au contributeur | M |
| Création/sélection de la FMEO |  | * Traçabilité de la FMEO   + - Règles de nommage/renommages     - Unicité des noms * Filtrages de sélection limités | M |
| Construction  Déploiement  Audit | Renseignement d'une FMEO |  | * Complétude et cohérence des saisies   + - Saisies libres, cohérences à postériori     - Transmission des infos du développeur     - Pas d'état d'utilisabilité (que l'état obsolète)     - Notion de FMEO déployée     - Informations complémentaires (Certificats Sesame, Getlog, …) pour la création de SA ou sa mise à jour   Cf. the Delivery Manager' manual | M |
| Gestion de configuration du DM |  | * Complétude et cohérence de la FMEO   + - Saisies libres, cohérences à postériori     - Sauf choix SA existant de l'Application   Cf. the Delivery Manager' manual |  |
| Contrôle qualité du DM |  |  |  |
| Exploitation des données du DM et de Nexus | Responsable des déploiements | * Référencement unique et centralisé des configurations de déploiement * Complétude et cohérence de la DL   + - C'est le déploiement réussi qui le prouve * Capitalisation sur les demandes de déploiement suivantes ? |  |
| Référencie les modèles qualités associés à leurs résultats de mesures, selon les versions qualifiées du Projet | Auditeur | * Vérifie que tous les résultats sont associés à leur modèle * Vérifie que les sauvegardes de deux versions ne supportent pas les mêmes résultats | A |

# Annexes

## Abréviations

NB: les abréviations utilisées pour les rôles en gestion de configuration ne correspondent pas à des fonctions RH

| **Abréviation** | **Signification** |
| --- | --- |
| ARA | Automic Release Automation  https://automic.com/fr/products/automic-release-automation) |
| CLC | Chaine de Livraison Cardif |
| TCG | Toolchain Group |
| GC | Gestion de Configurations |
| GCE | Gestion de Configuration des Environnements |
| PGC | Plan de Gestion de Configuration (Projet et/ou Application) |
| PGCL | Plan de Gestion de Configuration Logicielle (hors GCE) |
| RACI | Réalise - Autorité - Consulté - Informé |
| RefGC-Central | Référentiel Central de Gestion de Configurations |
| RefGC-Local | Référentiel Local de Gestion de Configurations |
| RFA | Responsabilité Fonctionnelle de l'Application |
| RGC | Référent de Gestion de Configuration (de l'Application) |
| RTC | Responsable Technique de Configuration |
| RGCT | Référent de Gestions de Configurations Transverse |
| TMA | Tierce Maintenance Applicative |

## Références documentaires

Supprimer les références documentaires inutiles

| **Document** | **Localisation** |
| --- | --- |
| Cartographie Documentaire | [https://cyou-cardif-assurance.is.echonet/rooms/PROQual/Methodes/Guides/FR/Cartographie\_Documentaire.pdf](https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/CardifDocuments/Forms/AllItems.aspx?RootFolder=%2fcommunities%2fGlobalConfigurationLeague%2fCardifDocuments%2fMod%c3%a8les&FolderCTID=0x012000B6C0A2BDC059E746B515ED346107DFD0) |
| GESTION DE CONFIGURATIONS  Règles, Normes & Standards | <https://cyou-cardif-assurance.is.echonet/communities/GlobalConfigurationLeague/Pages/home.aspx>  Menu "Règles" |
| PGCD : Plan de Gestion de Configuration Documentaire | https://cyou-cardif-assurance.is.echonet/rooms/PROQual/Methodes/Template/FR/Plan\_Gestion\_Documentaire.docx |
| Liste des Livrables | <http://cardif-harmonie.is.echonet/main.php?ctx=LIV_N&lg=FR> |
| Etude Automatisation CTLM - EdB Dev |  |
| Recommandations et Normes pour les ETUDES.doc |  |
| Normes de dénomination des Applications sous ALM |  |

## Rôles et Activités GC

Les rôles des acteurs de la gestion de configurations sont décrits ici selon leurs responsabilités et activités. Les acteurs prenant en charge un ou plusieurs de ces rôles sont précisés au § I.A.2

Vérifier la cohérence et complétude avec le tableau du §II.C.3

NB: les rôles en gestion de configuration ne correspondent pas à des fonctions RH.

| **Parties**  **Prenantes** | **Rôle**  **GC** | **Activités GC**  **Et**  **Responsabilité GC** | RACI par rapport aux Configurations  (Réalise - Autorité - Contributeur - Informé). |
| --- | --- | --- | --- |
| RA | RGC | * Choisi la version de départ du Projet parmi les versions de l’Application * Décide de l'intégration à l'Application d'un Projet (Evolution, Maintenance) développé * Décide des fusions de Projet à l'Application, * Incrémente le N° de l'Application pour chaque nouveau Projet | A, C, I |
| * Désigne les personnes devant assumer les rôles techniques GC clés : RGC, Contributeurs, Mergeur * Décide des occurrences des fusions des travaux des différents Contributeurs | C, I |
| * Rédige, diffuse et maintient le PGC du Projet (pouvant dépendre d'un PGC de l'Application ou plus largement de l'Entité) * Contrôle l'application des choix de GC au sein de l’Application * Interlocuteur du RGCT (notamment pour les revues GC) | R, C, I |
| Pour les merges sur la branche Master, le développeur expérimenté désigné sous la responsabilité du RA | RTC | * Gestionnaire des branches sur le RefGC-Central * Réalisateur des fusions des travaux des contributeurs dans RefGC-Central * Pose les labels de versions sur les configurations validées du RefGC-Central * Garanti que la fusion coïncide aux objectifs de la version envisagée * *(NB: rôle technique, ne décide pas des cas et occurrences de fusions)* | R, I |
| Pour les merges sur la branche Projet ou Maintenance, toute personne désignée par l'équipe Projet ou TMA |
| Développeur (Développeurs, internes Cardif, Equipe TMA non Cardif, Equipe(s) Projet non Cardif) | Contributeurs à la conception d'éléments gérés en configuration | * Initie ses branches de développements sur son RefGC-Local * Tire les fichiers nécessaires du RefGC-Central, vers son RefGC-Local * Modifie les fichiers sous sa responsabilité dans son RefGC-Local * Pousse ses contributions, de son RefGC-Local, vers le RefGC-Central * Participe aux fusions multi contributeurs selon les sollicitations du Mergeur | R, I |
| RA ou toute personne habilitée aux déploiements,   * La CLC/TCG <l’une ou l’autre sinon item inutile>, * Centre de Tests Cardif * Métier ou MOA ou AMOA | Consommateur de Configurations | * Accède en lecture seule à des configurations identifiées du RefGC-Central * Valorise par l'utilisation les configurations du RefGC-Central   (valorise => capacité à réutiliser, construire, à tester, à exploiter la solution) | R, I |
| Personne désignée par Cardif pour réaliser ce rôle centralisé inter Patrimoine & Projets | Référent de Gestions de Configurations Transverse (RGCT) | * Guide la rédaction du PGC du Projet/Application * Accompagne le RGC * Réalise la Revue de GC du Projet/Application * Apporte une vision transverse des GC pratiquées par les Projets (multi Application) | R, I |

## Revue GC

Les points de revue (limités à moins d'une vingtaine) et faisant l'objet d'un document particulier) doivent par exemple permettre de vérifier :

* L'application des choix de gestion (organisation, actions) du PGC, sans préjuger de leurs efficacités
* L'atteinte de chacun des objectifs/besoins initiaux retenus solutionnés par les moyens décrits par ce PGC
* La vision des parties prenantes (Cf. § 0) sur l'efficacité du dispositif

Dans ce document, pour chaque point, on peut préciser comment apprécier la revue vis-à-vis de ce PGC :

* Ce qui est vérifié et la mesure, le seuil d'atteinte de l'objectif et la valeur actuelle mesurée
* Les dispositions à mettre en œuvre si la valeur mesurée n'atteint pas le seuil
* Les points de vue des parties prenantes

Ce tableau accompagnera le résultat de la revue

Le tableau du type de celui ci-dessous accompagnera à postériori le résultat brut de chaque revue GC.

Il permettra d'analyser les moyens mis en œuvre (la partie Plan) vis-à-vis des objectifs fixés (la partie Stratégie) et de proposer un plan d'amélioration de l'application de ce PGC et/ou des objectifs et/ou des moyens associés à ces objectifs :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Point de revue  (date de la revue) | Pourcentage atteint  de l'objectif du point de la revue | Niveau d'application des choix du PGC | Nivaux d'atteinte des Objectifs et Besoins initiaux | Point de vue du RGC (satisfaisant ou progrès nécessaires) | **Actions correctrices à mettre en œuvre** |
| I - Eléments versionnés, Configurations versionnées |  |  |  |  |  |
| II - Complétude et Cohérence vérifiées |  |  |  |  |  |
| III - Occurrences de gestion, rôles et responsabilités |  |  |  |  |  |
| IV - Politique de branches |  |  |  |  |  |
| V - PGC et sa maintenance |  |  |  |  |  |
| VI - Standardisation |  |  |  |  |  |
| VII - Plan d'amélioration |  |  |  |  |  |
| VIII - Pilotage des projets |  |  |  |  |  |

1. Au Chapitre III.B Références Documentaires [↑](#footnote-ref-1)