

# Régularisations de fin de période

## Rôle

Les régularisations transforment les données accumulées pendant le module en écritures de fin de période. Elles concernent notamment amortissements, contrats, stocks et comptes explicitement configurés pour être régularisés.

## Préférences structurantes

Les constantes By Module exposent plusieurs flags :

- `uses_regularizations;`
- `uses_new_regularizations;`
- `regularization_accounts_count;`
- `affected_regularizations_reports;`
- `uses_stock_regularizations;`
- `has_new_immobilizations;`
- `has_new_contracts.`

## Calcul générique

`utilsByModule.getAmountToRegularize()` calcule la portion à régulariser en fonction du montant, de la date d'acquisition/début, de la fin de vie/contrat et des bornes de l'exercice. Les helpers spécialisés contrats et immobilisations s'appuient sur cette logique temporelle.

## Static data

`contract_regularizations.json` fournit les types de régularisation de contrats. D'autres comportements sont pilotés par les settings de modules/pages et les comptes définis dans les données Accounting.

# Runtime Unity

- `RegularizationDataBase`;
- `RegularizationBillEntryBase`;
- `ContractRegularizationBillEntry`;
- `BillDataContractRegularizationsReport`.

# UI

- `ContractRegularizationsReportControl`;
- `ContractWithInterestRegularizationsReportControl`;
- rows spécialisées correspondantes.

flowchart TD

DATA[Immobilizations / Contracts / Stock] --> PERIOD[End of fiscal period]

PERIOD --> CALC[Regularization calculations]

CALC --> REPORT[Regularization Reports]

CALC --> ENTRY[Journal entries]

ENTRY --> LEDGER[Ledger]

LEDGER --> FS[Financial statements]

# Pourquoi cette page est séparée

Une régularisation n'est ni “un contrat”, ni “une immobilisation”, ni “un bill” au sens architectural. C'est une étape transversale de fin de période qui consomme ces domaines et produit des écritures.

# Ajouter une nouvelle régularisation

## 1. Identifier la source

Une régularisation doit venir d'une donnée métier précise : contrat, immobilisation, taxe, stock ou compte explicitement configuré. Ne pas créer une écriture “magique” directement dans l'état financier.

## 2. Définir le calcul

Réutiliser `getAmountToRegularize()` lorsqu'il s'agit de répartir un montant sur une période. Le helper prend acquisition, fin de vie éventuelle, exercice fiscal, précision et sens `inAdvance`.

## 3. Définir les comptes

Ajouter les relations debit/credit dans la couche Account State appropriée. L'ordre compte : `getRegularizationsAccountsOrdersAndRelations()` est exécuté avant certaines étapes de closing/états.

## 4. Ajouter le report si l'utilisateur doit voir/calculer la régularisation

Créer ou étendre la Bill/report data serveur, le modèle Unity et le contrôle UI. Configurer la page dans les JSON.

## 5. Answer Key + Grading

Le même calcul de montant doit être utilisé des deux côtés. Le grading possède des branches pouvant utiliser les régularisations saisies par l'étudiant (`useUserRegularizations`) : vérifier si la nouvelle règle appartient à ce chemin.

# Test minimal

- montant entièrement dans l'exercice;
- prorata début/fin d'exercice;
- in advance vs to pay/receive;
- arrondi selon `numDecimals`;
- impact trial balance;
- impact income statement/balance sheet;
- context suivant.

