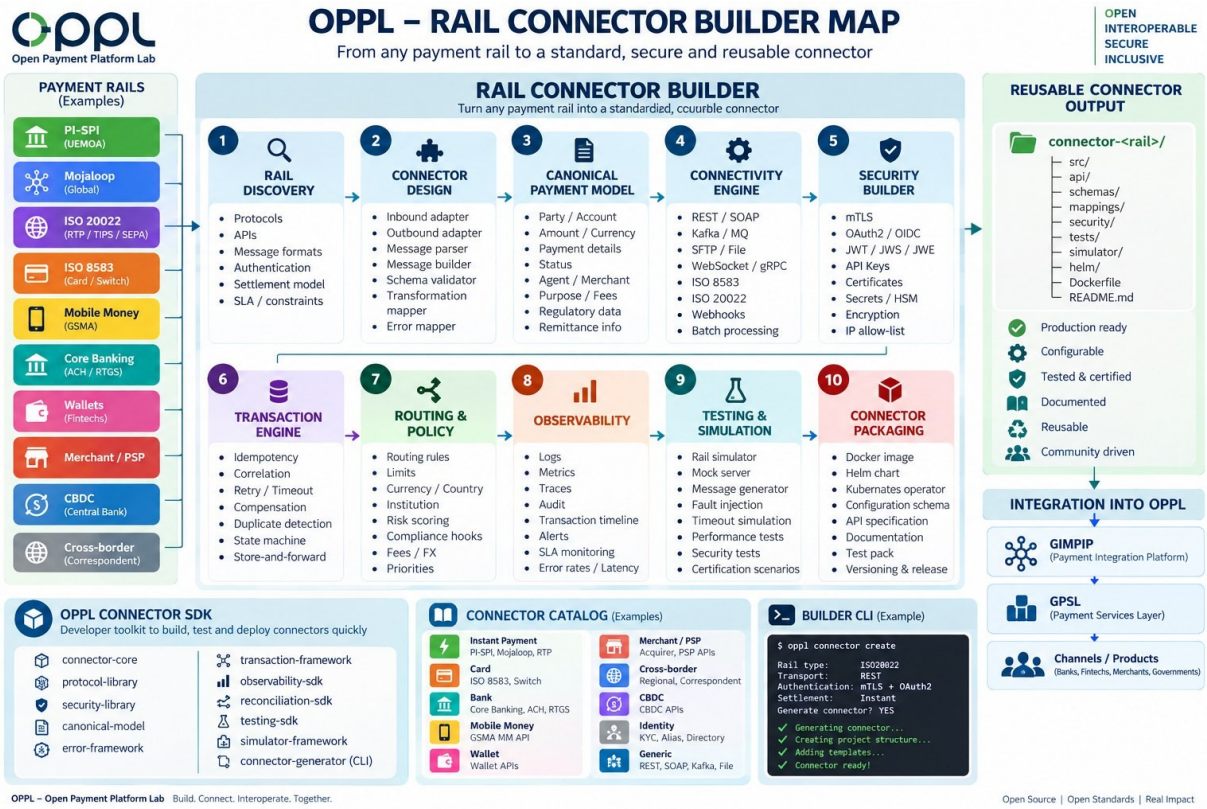
MISE EN ŒUVRE · DÉPLOIEMENT · DÉVELOPPEMENT · FORMATION

Rail Connector Framework OPPL

De n'importe quel rail de paiement à un connecteur standardisé, sécurisé et réutilisable.



Document	Guide de référence — OPPL Rail Connector Framework
Version	1.0 — 13 septembre 2026
Auteur	Lionel SAKITI — Professionnel des infrastructures numériques
Statut	Document de travail — environnement non productif, données synthétiques
Gouvernance	Rédaction assistée par IA, gouvernée par ISO/IEC 42001 ; validation humaine requise

Build. Connect. Interoperate.

Sommaire

1. Synthèse exécutive **4**
2. Le problème et la règle d'architecture **6**
2.1 Le coût caché du point à point 6
2.2 La règle fondamentale 6
2.3 La chaîne de responsabilités 7
2.4 Ce qu'est — et n'est pas — un connecteur 7
3. Le cadre en dix étapes **8**
3.1 Étape 1 — Découverte du rail 8
3.2 Étape 2 — Conception du connecteur 9
3.3 Les six verbes du connecteur, en détail 9
3.4 Ce qui n'appartient jamais au connecteur 10
4. Le modèle canonique **11**
4.1 Pourquoi il est le cœur du cadre 11
4.2 Les objets du modèle 11
4.3 ISO 20022 comme référence, pas comme obligation 12
4.4 La discipline du mapping 12
5. Connectivité et sécurité **13**
5.1 Étape 4 — Le moteur de connectivité 13
5.2 Étape 5 — Le constructeur de sécurité 14
5.3 Le sujet sous-estimé : la rotation 15
5.4 Les modèles matériels de sécurité 15
6. Le moteur transactionnel **16**
6.1 Ce qui distingue un paiement d'un appel d'API 16
6.2 La machine à états 16
6.3 Les sept mécanismes du moteur 17
6.4 L'interrogation, mécanisme de dernier recours 17
6.5 La taxonomie des erreurs 17
7. Routage, politique et observabilité **19**
7.1 Étape 7 — Où vit la décision 19
7.2 Étape 8 — L'observabilité 19
7.3 La piste d'audit 20
7.4 Le rapprochement vu du connecteur 21
8. Tests, simulation et certification **22**
8.1 Étape 9 — Le simulateur avant le connecteur 22
8.2 Le jeu de tests 23
8.3 Les six portes de certification 23
8.4 La performance, mesurée honnêtement 24
9. Le kit de développement **26**
9.1 Ce que chaque connecteur reçoit sans l'écrire 26
9.2 Le générateur 27
9.3 Versionner le kit 28
10. La fabrique de connecteurs **29**
10.1 Du projet au processus 29
10.2 Étape 10 — L'empaquetage 30
10.3 Le catalogue 30
10.4 Les dix premiers connecteurs 31
10.5 Journal de bord du troisième connecteur 31
11. Le laboratoire d'interopérabilité **33**
11.1 Faire dialoguer des rails qui ne se connaissent pas 33
11.2 Le déploiement du laboratoire 34
11.3 Les indicateurs du laboratoire 34
11.4 Ce que le laboratoire ne prouve pas 35
12. La pile technique et sa qualification **36**
12.1 Les briques candidates 36
12.2 La méthode de qualification 36
13. Mise en œuvre et déploiement **38**
13.1 Séquencement 38

13.2 Organisation des dépôts....38
13.3 Déployer un connecteur....39
13.4 Exploiter dix connecteurs comme un seul....39
13.5 Sauvegarde, reprise et réinitialisation....39
14. Former des ingénieurs de rail....40
14.1 Un parcours en six niveaux....40
14.2 Le livrable, pas la présentation....40
14.3 Progression indicative....41
15. Industrialisation et proposition de valeur....42
15.1 Le modèle cible....42
15.2 Ce qui constitue la propriété stratégique....42
15.3 La proposition de valeur....42
15.4 Offres de service envisageables....43
16. Gouvernance, risques et feuille de route....44
16.1 Organisation....44
16.2 Risques et parades....44
16.3 Feuille de route....45
16.4 Indicateurs de réussite....45
16.5 Les décisions à documenter dès le départ....45
16.6 La doctrine....46
Annexes....47
Annexe A — Glossaire....47
Annexe B — Check-list de construction d'un connecteur....47
Annexe D — Fiche de découverte type d'un rail....48
Annexe C — Références....49

1. Synthèse exécutive

Connecter un nouveau rail de paiement prend aujourd'hui des mois. Le suivant en prendra autant, puis le suivant encore. Chaque intégration repart de zéro : on redécouvre le protocole, on réécrit la sécurité, on réinvente la gestion des erreurs, on redébat de l'idempotence. Au bout de quelques rails, l'institution possède une collection d'interfaces sans rien qui les relie — et sans personne capable d'en exploiter deux à la fois.

Le **Rail Connector Framework** répond à ce problème par un déplacement simple : cesser de considérer un connecteur comme un projet, et le traiter comme un **produit fabriqué par une chaîne**. Un cadre commun, un modèle canonique, un kit de développement, un générateur, des simulateurs, un jeu de tests, six portes de certification et un catalogue. Le résultat n'est pas un connecteur de plus : c'est la capacité industrielle d'en produire un nouveau de façon prévisible.

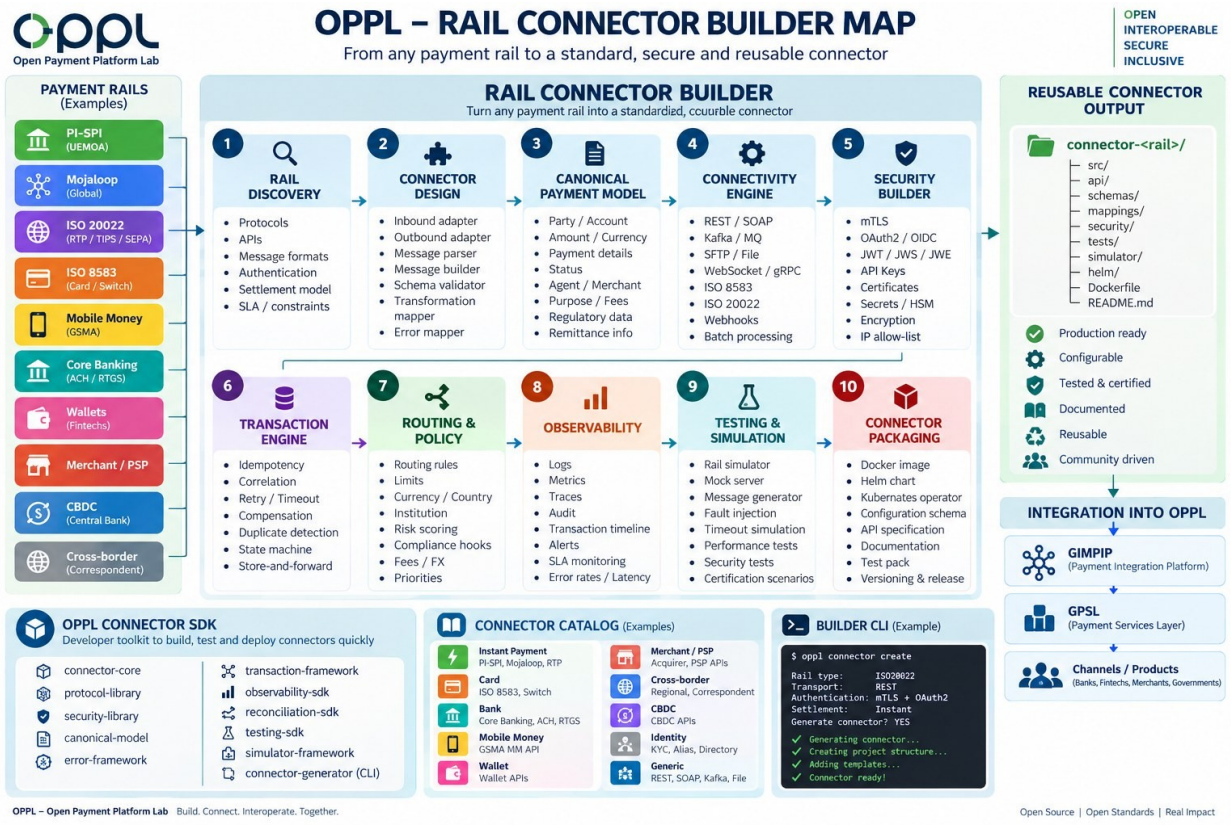


Figure 1 — Carte de construction d'un connecteur de rail — les dix étapes du cadre

La règle d'architecture qui fonde tout l'édifice tient en une phrase : **un rail ne doit jamais être directement couplé à un autre rail**. Chaque rail est traduit par son connecteur vers un modèle canonique unique ; la plateforme d'intégration route et orchestre ; la couche de services expose des produits. Cette séparation transforme un problème quadratique en problème linéaire.

Tableau 1 — Apports du cadre

Dimension	Ce que le cadre apporte
Économie d'échelle	Avec n rails couplés deux à deux, il faut n(n−1)/2 interfaces. Avec le cadre, il en faut n. Pour dix rails : 45 contre 10.
Prévisibilité	Un connecteur suit une structure imposée, une méthode en dix étapes et six portes de certification. Sa durée de production devient estimable.
Qualité	La sécurité, l'idempotence, la reprise et l'observabilité viennent du cadre : elles sont corrigées une fois pour tous les connecteurs.

Dimension	Ce que le cadre apporte
Exploitabilité	Un exploitant qui connaît un connecteur les connaît tous : même structure, même configuration, même tableau de bord, mêmes procédures.
Compétences	Un parcours en six niveaux qui produit des ingénieurs capables de construire un connecteur réel, pas d'en parler.
Propriété stratégique	La valeur durable n'est pas le code d'un connecteur : c'est la méthode, le modèle canonique, la fabrique, les simulateurs et le catalogue.

Positionnement du document

Le cadre est développé et éprouvé dans un environnement **non productif**. Les rails sont représentés par des simulateurs paramétrables ; aucune connexion à un système réel n'est établie depuis le laboratoire.

Les captures d'écran sont des **maquettes illustratives reconstituées** ; les chiffres qui y figurent sont fictifs.

Les références à GIMPIP, GPSL et PI-SPI décrivent une **cible d'intégration**. Les spécifications de ces systèmes ne sont pas publiques et toute connexion effective relève des processus d'homologation de leurs opérateurs.

2. Le problème et la règle d'architecture

2.1 Le coût caché du point à point

Lorsqu'on connecte deux systèmes de paiement, on écrit une interface. Lorsqu'on en connecte trois, on en écrit trois. À six rails, il en faut quinze ; à dix, quarante-cinq. La croissance n'est pas linéaire mais quadratique, et chaque interface doit être maintenue, sécurisée, testée, documentée et exploitée.

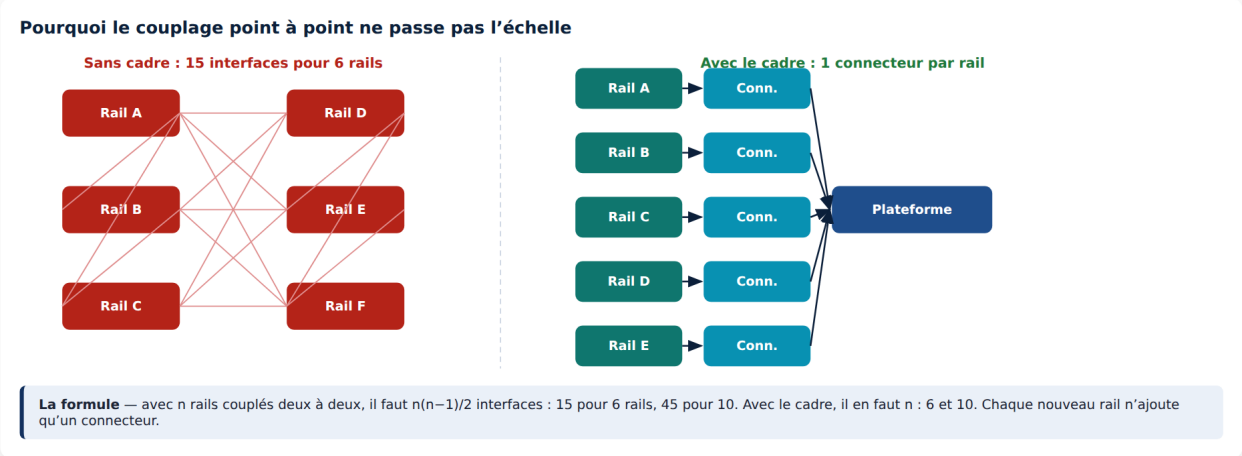


Figure 2 — Pourquoi le couplage point à point ne passe pas l'échelle

Le coût visible — le développement — n'est pas le plus élevé. Le coût réel se révèle plus tard : chaque interface a sa propre gestion d'erreurs, sa propre idée de l'idempotence, ses propres journaux, son propre mode d'exploitation. Une correction de sécurité doit être appliquée quinze fois. Un ingénieur qui part emporte la connaissance de trois interfaces que personne d'autre ne comprend.

2.2 La règle fondamentale

La règle

Un rail ne doit jamais être directement couplé à un autre rail. Chaque rail est traduit par son connecteur vers un modèle canonique unique, et c'est la plateforme d'intégration — jamais un connecteur — qui décide vers quel autre rail le paiement poursuit sa route.

Cette règle a une conséquence immédiate et libératrice : **ajouter un rail ne coûte qu'un connecteur**. Aucune interface existante n'est modifiée. Aucun connecteur ne connaît l'existence d'un autre. La complexité cesse de croître avec le carré du nombre de rails.

La différence tient en une règle

```
À éviter                          À construire

Rail A ──────► Rail B             Rail A ──► Connecteur A ─┐
Rail A ──────► Rail C             Rail B ──► Connecteur B ─┤
Rail A ──────► Rail D             Rail C ──► Connecteur C ─┼──► Plateforme
Rail B ──────► Rail C             Rail D ──► Connecteur D ─┤    d'intégration
Rail B ──────► Rail D             Rail E ──► Connecteur E ─┘          │
Rail C ──────► Rail D                                                 ▼
                                                          Couche de services
15 interfaces pour 6 rails        6 connecteurs pour 6 rails
```

2.3 La chaîne de responsabilités

Trois couches, trois responsabilités distinctes. Les confondre est l'erreur la plus fréquente et la plus coûteuse.

Tableau 2 — Les trois responsabilités

Couche	Responsabilité	Ce qu'elle ne fait jamais
Connecteur de rail	Comprendre et traduire le rail : connectivité, authentification, analyse, validation, correspondance, gestion des erreurs, corrélation	Décider vers quel rail router, appliquer la politique commerciale, porter la logique produit
Plateforme d'intégration	Intégrer, router et orchestrer : politique, sélection du rail, gestion transactionnelle, compensation	Connaître les détails protocolaires d'un rail
Couche de services	Construire et exposer les services de paiement destinés aux clients	Parler directement à un rail

La chaîne complète, du rail au produit

```
RAIL  →  CONNECTEUR  →  PLATEFORME D'INTÉGRATION  →  COUCHE DE SERVICES  →  PRODUIT

Exemples de services exposés :
  /payments        /transfers      /request-to-pay   /refunds
  /payment-status  /aliases        /beneficiaries    /merchants
  /bulk-payments
```

2.4 Ce qu'est — et n'est pas — un connecteur

Tableau 3 — Le périmètre d'un connecteur

Un connecteur EST	Un connecteur N'EST PAS
Une brique d'intégration spécialisée dans un rail	Le lieu où vit la logique métier du produit de paiement
Un traducteur entre un protocole et le modèle canonique	Un routeur : il ne choisit jamais la destination
Un composant mince, prévisible et remplaçable	Un projet sur mesure avec sa propre architecture
Un produit logiciel versionné, testé et documenté	Un script d'intégration maintenu par son auteur
Un consommateur des services du cadre commun	Un réimplémenteur de la sécurité et de l'idempotence

Le connecteur se résume à six verbes : **comprendre, connecter, valider, traduire, sécuriser, observer**. Tout ce qui dépasse ces six verbes appartient à une autre couche.

3. Le cadre en dix étapes

La construction d'un connecteur suit dix étapes ordonnées. Chacune produit un livrable vérifiable, et aucune ne peut être sautée : l'étape 3 suppose l'étape 2, l'étape 9 suppose les huit précédentes. C'est cet ordre qui rend la production prévisible.

Tableau 4 — Les dix étapes du cadre

#	Étape	Question à laquelle elle répond	Livrable
1	Découverte du rail	Comment ce rail fonctionne-t-il réellement ?	Fiche de découverte complète
2	Conception du connecteur	Quelle est sa structure interne ?	Architecture normalisée instanciée
3	Modèle canonique	Comment ce rail se traduit-il en modèle commun ?	Tables de correspondance versionnées
4	Moteur de connectivité	Par quel transport dialogue-t-on ?	Adaptateur de transport configuré
5	Constructeur de sécurité	Comment sécurise-t-on l'échange ?	Couche de sécurité activée et testée
6	Moteur transactionnel	Que se passe-t-il quand ça échoue ?	Machine à états, idempotence, reprise
7	Routage et politique	Qui décide, et sur quels critères ?	Règles déclarées dans la plateforme
8	Observabilité	Peut-on suivre un paiement de bout en bout ?	Traces, métriques, journaux, alertes
9	Tests et simulation	Comment développer sans le rail réel ?	Simulateur et jeu de tests
10	Empaquetage	Un tiers peut-il l'exploiter ?	Image, chart, documentation, version

3.1 Étape 1 — Découverte du rail

Avant toute ligne de code, le rail est documenté. Cette étape est la plus souvent bâclée et la plus coûteuse à rattraper : une contrainte découverte en phase de tests fait perdre plus de temps que la semaine consacrée à la documenter.

Tableau 5 — La fiche de découverte du rail

Domaine	Ce qu'il faut collecter
Protocoles	REST, SOAP, TCP/IP, HTTP, WebSocket, gRPC, files de messages, événements, transfert de fichiers
Formats et normes	ISO 20022, ISO 8583, JSON, XML, formats délimités, formats binaires, formats propriétaires
Sécurité	Authentification mutuelle, jetons, signatures, certificats, chiffrement, modules matériels
Cycle transactionnel	Initiation, autorisation, compensation, règlement, confirmation, annulation, remboursement, rapprochement
Contraintes d'exploitation	Engagements de service, débit, latence, disponibilité, heures limites, plafonds, devises
Comportement en erreur	Codes d'erreur, politique de nouvelle tentative, délais, comportement en cas de rejeu
Modèle de règlement	Instantané, différé, par lot, net ou brut, fenêtres de règlement

Le test de la fiche de découverte

Une fiche est complète lorsqu'un ingénieur qui n'a jamais vu le rail peut répondre à cette question : « que se passe-t-il si j'envoie deux fois le même message ? ». Si la fiche ne répond pas, le rail n'est pas documenté — et le connecteur produira des doublons.

3.2 Étape 2 — Conception du connecteur

Tous les connecteurs partagent la même architecture interne. Ce n'est pas une préférence esthétique : c'est ce qui permet à un ingénieur de reprendre le connecteur d'un collègue sans formation préalable.

L'architecture interne normalisée

```
Chemin entrant (du rail vers la plateforme)     Chemin sortant (de la plateforme vers le rail)

  Adaptateur entrant                               Modèle canonique
        ↓                                                ↓
  Authentification                                 Mapper vers le rail
        ↓                                                ↓
  Analyseur de message                             Constructeur de message
        ↓                                                ↓
  Validateur de schéma                             Sécurité (signature)
        ↓                                                ↓
  Mapper de transformation                         Adaptateur sortant
        ↓                                                ↓
  Mapper canonique                                 Rail
        ↓
  Gestionnaire de transaction
        ↓
  Client de la plateforme
```



Figure 3 — Structure de projet imposée par le générateur (capture illustrative)

3.3 Les six verbes du connecteur, en détail

Tableau 6 — Les six verbes et leurs pièges

Verbe	Ce qu'il recouvre	Erreur classique
Comprendre	Connaître le protocole, les formats, le cycle de vie et les limites du rail	Commencer à coder avant d'avoir documenté le rail

Verbe	Ce qu'il recouvre	Erreur classique
Connecter	Établir et maintenir le lien, gérer sessions, reprises et indisponibilités	Traiter la connexion comme acquise et ignorer la reprise de lien
Valider	Vérifier le message contre son schéma puis contre les règles du rail	S'arrêter à la validation de schéma, qui ne dit rien du métier
Traduire	Convertir dans les deux sens entre le rail et le modèle canonique	Glisser une décision métier dans la traduction
Sécuriser	Authentifier, signer, chiffrer, gérer les secrets et leur rotation	Réimplémenter la cryptographie plutôt que de configurer le cadre
Observer	Émettre traces, métriques, journaux et piste d'audit normalisés	Journaliser des données sensibles en clair

3.4 Ce qui n'appartient jamais au connecteur

- **Le choix du rail.** Un connecteur qui contient une condition menant vers un autre rail a reconstitué le couplage point à point.
- **La tarification.** Les frais commerciaux appartiennent à la couche de services ; seuls les frais imposés par le rail transitent par le connecteur.
- **Les plafonds commerciaux.** Les limites réglementaires et commerciales sont évaluées une fois, en amont, pour tous les rails.
- **L'évaluation du risque.** Un score de fraude calculé par connecteur donne autant de politiques que de rails.
- **La logique produit.** Ce qui distingue un virement instantané d'un paiement marchand relève du produit, pas du transport.
- **La comptabilité.** Le connecteur ne débite ni ne crédite : il transporte l'instruction et rapporte le résultat.

La règle pratique est simple : si une modification du catalogue de produits oblige à modifier un connecteur, c'est que le connecteur porte quelque chose qui ne lui appartient pas.

4. Le modèle canonique

4.1 Pourquoi il est le cœur du cadre

Le modèle canonique est la pièce qui rend tout le reste possible. Sans lui, un paiement initié par carte ne peut pas se terminer sur un rail instantané sans qu'on écrive une interface entre les deux. Avec lui, chaque connecteur ne connaît qu'un seul interlocuteur : le modèle commun.

Ce modèle n'est pas une couche de plus : il est ce qui permet de supprimer toutes les autres. Il doit donc être conçu avec soin, versionné rigoureusement et défendu contre les ajouts opportunistes propres à un rail particulier.

4.2 Les objets du modèle

Tableau 7 — Les objets du modèle canonique

Objet	Rôle	Piège à éviter
Payment	L'instruction de paiement, racine du modèle	Y ajouter des champs propres à un rail
Party	Une partie : débiteur, créancier, donneur d'ordre	Confondre la partie et son compte
Account	Un compte identifié selon un schéma donné	Supposer un format d'identifiant unique
Agent	L'institution qui tient le compte	Omettre l'agent, indispensable en interbancaire
Amount / Currency	Le montant en unités mineures entières et sa devise	Utiliser un nombre à virgule flottante
Fees / FX	Frais et conversion de devise	Mélanger frais du rail et frais commerciaux
Purpose	Motif du paiement, souvent codifié	Réinterpréter localement un code du rail
Remittance	Information de bout en bout destinée au bénéficiaire	Y loger des données techniques
Settlement	Modèle et références de règlement	Confondre exécution et règlement
RegulatoryData	Informations exigées par la réglementation	Les traiter comme optionnelles
Status	État courant dans la machine à états	Créer des états propres à un rail
Identifiers	Identifiants internes et de bout en bout	Écraser un identifiant fourni par le rail

La structure canonique

```
PaymentInstruction
├── paymentId              identifiant interne, généré par la plateforme
├── endToEndId             identifiant de bout en bout, conservé tel quel
├── transactionId          identifiant de la transaction sur le rail
├── debtor / debtorAccount / debtorAgent
├── creditor / creditorAccount / creditorAgent
├── amount { minorUnits: entier, currency: ISO 4217 }
├── fees / fx
├── purpose
├── remittanceInformation
├── settlementInformation
├── regulatoryInformation
├── status
└── timestamps
```

Deux invariants non négociables

Les montants sont des entiers d'unités mineures. Le franc CFA n'a pas de subdivision : 25 000 FCFA s'écrit 25000, jamais 25000.00 ni 2500000. Aucun nombre à virgule flottante n'est admis dans le code monétaire.

Les identifiants fournis par un rail sont conservés tels quels. On ne les réécrit pas, on ne les normalise pas, on ne les tronque pas. Ils sont la seule preuve permettant de rapprocher une transaction avec la contrepartie.

4.3 ISO 20022 comme référence, pas comme obligation

La norme ISO 20022 constitue une référence précieuse pour concevoir le modèle canonique : son vocabulaire est mûr, ses structures sont éprouvées et son adoption progresse mondialement. Il serait toutefois erroné d'en conclure que tous les rails doivent l'utiliser nativement.

Un rail par carte parle ISO 8583. Un opérateur de monnaie électronique expose une interface qui lui est propre. Un partenaire transfrontalier échange des fichiers. Le modèle canonique s'inspire d'ISO 20022 sans imposer que chaque rail y soit converti à la source : c'est précisément le rôle du connecteur que d'opérer cette traduction.

Mapper canonique — aller et retour

```
dev@oppl-lab:~/connectors/connector-pispi$ cat src/mapper/CanonicalMapper.java

public class Pispi2Canonical implements RailMapper<RailMessage, PaymentInstruction> {

  @Override
  public PaymentInstruction toCanonical(RailMessage msg) {
    return PaymentInstruction.builder()
        .paymentId(ids.newPaymentId())             // identifiant interne OPPL
        .endToEndId(msg.get("EndToEndId"))       // conservé tel quel
        .transactionId(msg.get("TxId"))
        .debtor(Party.from(msg.path("Dbtr")))
        .debtorAccount(Account.from(msg.path("DbtrAcct")))
        .debtorAgent(Agent.from(msg.path("DbtrAgt")))
        .creditor(Party.from(msg.path("Cdtr")))
        .creditorAccount(Account.from(msg.path("CdtrAcct")))
        .amount(Money.ofMinorUnits(msg.longExact("IntrBkSttlmAmt"),
                                   msg.get("@Ccy")))   // XOF : entier
        .purpose(Purpose.from(msg.path("Purp")))
        .remittance(msg.get("RmtInf"))
        .settlement(Settlement.instant())
        .status(TransactionStatus.RECEIVED)
        .build();
  }

  @Override
  public RailMessage fromCanonical(PaymentInstruction p) { /* chemin retour */ }
}

# Règle du cadre : le mapper ne contient AUCUNE décision métier.
# Il traduit. Les plafonds, le routage et la tarification vivent ailleurs.
```

Maquette illustrative · OPPL Rail Connector Framework

Figure 4 — Mapper canonique — traduction aller et retour (capture illustrative)

4.4 La discipline du mapping

- **Une correspondance est une donnée, pas du code.** Les tables sont versionnées, revues et testées indépendamment du connecteur qui les applique.
- **Les champs non mappés sont documentés.** Un champ du rail sans équivalent canonique est listé explicitement, avec la raison. Le silence est interdit.
- **Aucune décision métier dans le mapper.** Un mapper qui contient une condition sur un montant ou un pays est un mapper qui a franchi sa frontière.
- **Le chemin retour est testé autant que l'aller.** La moitié des défauts d'intégration se logent dans la construction du message sortant.
- **Les codes du rail sont conservés.** Le connecteur transporte le code d'erreur d'origine ; c'est la plateforme qui l'interprète, une fois, pour tous les rails.

5. Connectivité et sécurité

5.1 Étape 4 — Le moteur de connectivité

Un rail impose son transport : certains exposent une interface web, d'autres écoutent sur une socket, d'autres encore échangent des fichiers déposés à heure fixe. Le cadre traite le transport comme un détail interchangeable, isolé dans un adaptateur.

Tableau 8 — Transports supportés par le cadre

Transport	Rails concernés	Point d'attention
REST / HTTP	Paiement instantané, monnaie électronique, acquisition	Délais explicites, codes de statut, rejeu des requêtes
SOAP	Systèmes bancaires historiques	Gestion des espaces de noms, verbosité, outillage vieillissant
TCP persistant	Réseaux cartes	Maintien de session, messages réseau périodiques, reprise de lien
Files de messages	Interbancaire, systèmes de place	Garantie de livraison, ordre, messages non traités
Événementiel	Architectures internes modernes	Clé de partition, ordre par paiement, consommateurs idempotents
Transfert de fichiers	Compensation, lots de masse	Heures limites, fichiers partiels, accusés de réception
Rappels web	Opérateurs de monnaie électronique	Signature, doublons, absence de rappel

Adaptateur de transport — le transport devient interchangeable

```
dev@oppl-lab:~/connectors$ cat src/inbound/TransportRoutes.java

// Le cœur du connecteur ignore le transport. Seul l'adaptateur le connaît.

from("rest:post:/v1/rail/inbound")          // transport REST
    .to("direct:connector-core");

from("kafka:rail.inbound.v1?groupId=connector")  // transport événementiel
    .to("direct:connector-core");

from("sftp://rail-host/in?delay=60000")     // transport par fichier
    .split().tokenize("\n")
    .to("direct:connector-core");

from("netty:tcp://0.0.0.0:10000")           // transport TCP (cartes)
    .to("direct:connector-core");

from("direct:connector-core")              // chaîne commune à tous les transports
    .process(authenticator)
    .process(parser)
    .process(validator)
    .process(canonicalMapper)
    .process(transactionManager)
    .to("direct:platform-client");

# Conséquence pratique : changer le transport d'un rail n'impacte qu'une ligne.
# Le mapping, la sécurité et le moteur transactionnel restent inchangés.
```

Maquette illustrative · OPPL Rail Connector Framework

Figure 5 — Le transport isolé dans un adaptateur : le cœur du connecteur l'ignore (capture illustrative)

La conséquence pratique est considérable. Lorsqu'un rail change de transport — passage d'un échange de fichiers à une interface web, migration d'une file vers un bus d'événements — seul l'adaptateur est modifié. Le mapping, la sécurité, le moteur transactionnel et l'observabilité restent intacts.

5.2 Étape 5 — Le constructeur de sécurité

La sécurité est le domaine où la réécriture par connecteur cause le plus de dégâts. Chaque implémentation maison introduit ses propres faiblesses, et une faille corrigée dans un connecteur persiste dans les neuf autres. Le cadre fournit donc une couche de sécurité unique, configurée et non réécrite.

Tableau 9 — Répartition des responsabilités de sécurité

Capacité	Fournie par le cadre	Ce que le connecteur configure
Authentification mutuelle	Établissement, vérification, rotation	Certificats et autorités du rail
Jetons d'accès	Obtention, renouvellement, mise en cache	Point d'émission, portées, identifiants
Signature de message	Signature détachée ou enveloppée, vérification	Algorithme et clé attendus par le rail
Chiffrement de charge utile	Chiffrement et déchiffrement	Activation selon les exigences du rail
Gestion des secrets	Récupération, rotation, absence dans l'image	Chemins des secrets
Restriction réseau	Filtrage par plages autorisées	Plages du rail
Non-répudiation	Archivage signé des messages échangés	Durée de conservation

La chaîne de contrôles, dans l'ordre

```
Rail
  ↓
Sécurité réseau (filtrage, plages autorisées)
  ↓
Authentification mutuelle
  ↓
Authentification applicative (jeton)
  ↓
Autorisation (portées)
  ↓
Vérification de signature
  ↓
Validation du message
  ↓
Cœur du connecteur
```



Figure 6 — Couche de sécurité d'un connecteur — configurée, pas réécrite (capture illustrative)

La règle de refus

Un connecteur qui implémente sa propre cryptographie est un connecteur refusé à la certification. Non parce que l'auteur serait incompétent, mais parce qu'une correction de sécurité doit pouvoir être appliquée une fois et bénéficier immédiatement aux dix connecteurs. C'est un choix d'exploitabilité autant que de sécurité.

5.3 Le sujet sous-estimé : la rotation

Les certificats expirent, les clés tournent, les secrets changent. La plupart des connecteurs gèrent correctement le cas nominal et échouent sur la rotation : interruption de service de quelques secondes, connexions établies avec l'ancien certificat, cache de jetons non invalidé. Le cadre impose donc un test explicite — rotation sans interruption — parmi les critères de la porte de sécurité.

5.4 Les modèles matériels de sécurité

Certains rails exigent que les opérations cryptographiques sensibles — dérivation de clés, calcul de codes de vérification, chiffrement de données de carte — soient exécutées dans un module matériel dédié plutôt que dans le logiciel. Le laboratoire utilise un module logiciel de test, qui expose la même interface sans en offrir les garanties.

Tableau 10 — Module cryptographique : laboratoire et réalité

Sujet	Dans le laboratoire	En contexte réel
Module cryptographique	Module logiciel de test	Module matériel certifié
Clés	Clés de test, générées localement	Clés cérémonielles, sous double contrôle
Interface	Identique — c'est tout l'intérêt	Identique
Garanties	Aucune — usage pédagogique	Inviolabilité physique, certification
Procédures	Simplifiées	Cérémonies de clés, journal, témoins

L'intérêt de cette substitution est qu'elle permet de concevoir dès le laboratoire l'architecture d'appel au module : quelles opérations y sont déléguées, à quel moment, avec quelle latence ajoutée et quel comportement en cas d'indisponibilité. Ces décisions sont coûteuses à reprendre après coup.

Ce que le module logiciel ne dit pas

Un module matériel réel ajoute une latence non négligeable et constitue un point de défaillance unique s'il n'est pas redondé. Les campagnes de performance menées avec un module logiciel sous-estiment donc systématiquement la latence finale. Le rapport de campagne doit le mentionner explicitement.

6. Le moteur transactionnel

6.1 Ce qui distingue un paiement d'un appel d'API

Un appel d'API qui échoue se rejoue. Un paiement qui échoue peut avoir été exécuté. Cette différence, apparemment mineure, commande toute la conception du moteur transactionnel : le connecteur doit à tout moment savoir distinguer trois situations — le rail a accepté, le rail a refusé, et je ne sais pas ce que le rail a fait.

La distinction fondatrice

« Le rail a refusé » est un résultat. « Je ne sais pas ce que le rail a fait » est une absence de résultat. Les traiter de la même façon produit soit des doubles paiements, soit des clients débités sans motif. Aucune autre erreur de conception n'a des conséquences aussi coûteuses.

6.2 La machine à états

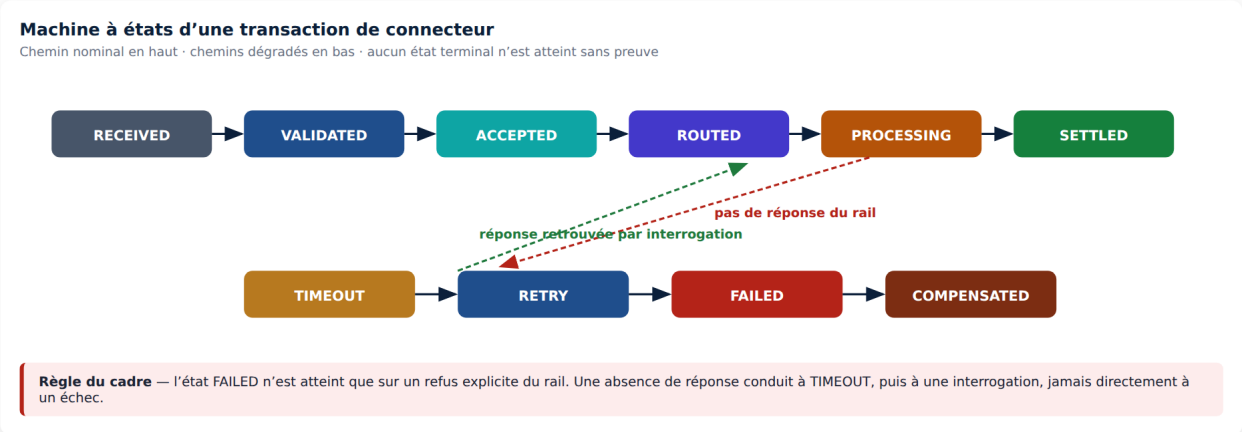


Figure 7 — Machine à états d'une transaction de connecteur

Tableau 11 — Les états et leurs transitions

État	Signification	Transitions possibles
RECEIVED	Message reçu, pas encore analysé	VALIDATED, rejet de format
VALIDATED	Message conforme au schéma et aux règles	ACCEPTED, rejet métier
ACCEPTED	Le connecteur prend la responsabilité du message	ROUTED
ROUTED	Transmis à la plateforme ou au rail	PROCESSING
PROCESSING	En attente de la réponse de la contrepartie	SETTLED, FAILED, TIMEOUT
SETTLED	Confirmation reçue, état terminal	—
TIMEOUT	Aucune réponse dans le délai imparti	RETRY, interrogation
RETRY	Nouvelle tentative après vérification	PROCESSING, FAILED
FAILED	Refus explicite, état terminal	COMPENSATED si nécessaire
COMPENSATED	Effets annulés, état terminal	—

6.3 Les sept mécanismes du moteur

Tableau 12 — Les sept mécanismes du moteur transactionnel

Mécanisme	Ce qu'il résout	Comment le cadre l'implémente
Idempotence	Le même message reçu deux fois	Clé stable, empreinte du contenu, réponse mise en cache
Corrélation	Relier une réponse tardive à sa demande	Référence enregistrée avant émission, index dédié
Nouvelle tentative	Une défaillance passagère du rail	Politique par type d'erreur, jamais sur un refus métier
Délai d'attente	Une contrepartie qui ne répond pas	Délai explicite sur chaque appel, toujours inférieur au délai du rail
Détection de doublon	Un message déjà traité	Index sur l'identifiant du rail, fenêtre de rétention
Compensation	Annuler les effets d'une étape réussie	Compensation déclarée au moment de l'étape, exécutée en ordre inverse
Stockage et réémission	Le rail est indisponible	File durable, disjoncteur, réémission ordonnée

La réservation avant émission

```
// L'ordre de ces deux opérations n'est pas négociable
const ref = await references.reserve(payment.paymentId);   // 1. réserver
const res = await rail.send(message, ref);                 // 2. émettre

// Si le connecteur s'arrête entre les deux, la reprise trouve une référence
// réservée sans émission : elle interroge le rail avant toute nouvelle tentative.
// Dans l'ordre inverse, la reprise ne saurait pas si l'émission a eu lieu.
```

6.4 L'interrogation, mécanisme de dernier recours

Tout connecteur doit implémenter une méthode d'interrogation permettant de demander au rail l'état réel d'une transaction. C'est le seul moyen de sortir d'une situation indéterminée sans prendre de risque. Un connecteur sans interrogation est un connecteur qui, tôt ou tard, produira un doublon ou perdra un paiement.

- Elle est appelée après chaque expiration, avant toute nouvelle tentative.
- Elle est appelée périodiquement sur toutes les transactions en état non terminal.
- Elle est appelée au redémarrage du connecteur, pour rattraper les transactions en vol.
- Son absence est un motif de refus à la porte fonctionnelle de la certification.

6.5 La taxonomie des erreurs

Une erreur mal classée produit un mauvais comportement. Le cadre impose donc une taxonomie unique, appliquée par les dix connecteurs, qui détermine mécaniquement ce que le système fait ensuite.

Tableau 13 — Taxonomie des erreurs du cadre

Famille	Origine	Le connecteur doit	Nouvelle tentative
Erreur de transport	Connexion refusée, résolution impossible, coupure avant envoi	Considérer que rien n'a été émis	Oui, immédiate
Erreur de sécurité	Certificat expiré, signature invalide, jeton refusé	S'arrêter et alerter — ne jamais réessayer en boucle	Non

Famille	Origine	Le connecteur doit	Nouvelle tentative
Erreur de format	Message non conforme au schéma du rail	Rejeter, journaliser le message brut, alerter	Non
Refus métier	Provision, compte inconnu, plafond, opération interdite	Conclure à l'échec et conserver le code du rail	Non — jamais
Erreur transitoire du rail	Service momentanément indisponible, saturation	Réessayer selon la politique, ouvrir le disjoncteur si persistant	Oui, avec temporisation croissante
Absence de réponse	Expiration après émission	Ne rien conclure, interroger le rail	Seulement après interrogation
Erreur de règlement	Rejet postérieur à une acceptation	Déclencher la compensation, qualifier en incident	Non
Erreur interne	Défaut du connecteur lui-même	Échouer proprement, ne pas contaminer l'état du paiement	Selon le défaut

La classe d'erreur détermine le comportement, pas le message

```
// La taxonomie est portée par le cadre, pas réinventée par connecteur
public enum ErrorClass {
  TRANSPORT(Retry.IMMEDIATE),      SECURITY(Retry.NEVER),
  FORMAT(Retry.NEVER),             BUSINESS_DECLINE(Retry.NEVER),
  RAIL_TRANSIENT(Retry.BACKOFF),   NO_RESPONSE(Retry.AFTER_INQUIRY),
  SETTLEMENT(Retry.NEVER),         INTERNAL(Retry.CONDITIONAL);
}

// Le code du rail est transporté tel quel, à côté de la classe
throw new ConnectorException(ErrorClass.BUSINESS_DECLINE,
        railCode /* "AM04" */, railMessage, correlationId);
```

Pourquoi la classification prime sur le message

Un message d'erreur est destiné à un humain ; une classe d'erreur est destinée au système. Un connecteur qui décide de réessayer en analysant le texte d'un message se brisera au premier changement de formulation côté rail. La classification se fait sur le code, jamais sur le libellé.

7. Routage, politique et observabilité

7.1 Étape 7 — Où vit la décision

La tentation est constante de loger une petite règle de routage dans un connecteur : « si le montant dépasse tel seuil, passer par l'autre rail ». Cette commodité détruit l'architecture. Un connecteur qui route connaît l'existence d'un autre rail, et le couplage point à point se reconstitue par la porte de service.

Tableau 14 — Critères de routage et lieu d'évaluation

Critère de routage	Exemples	Où il est évalué
Destination	Pays, institution, schéma d'identifiant	Plateforme d'intégration
Devise	Devise de l'opération, besoin de conversion	Plateforme d'intégration
Montant	Seuils, plafonds par type d'opération	Plateforme d'intégration
Type de paiement	Instantané, différé, masse, remboursement	Plateforme d'intégration
Disponibilité du rail	État du disjoncteur, indisponibilité déclarée	Plateforme, informée par le connecteur
Coût et latence	Tarification du rail, latence observée	Plateforme d'intégration
Risque et conformité	Score de risque, statut de conformité	Plateforme d'intégration
Priorité	Urgence déclarée, engagement de service	Plateforme d'intégration

Le connecteur ne possède qu'une seule information utile au routage : **son propre état de santé**. Il la publie ; c'est la plateforme qui en tire les conséquences.

Le repli et le doublon

Un repli vers un autre rail n'est autorisé que sur un échec **technique dont on a la certitude qu'il a précédé l'émission**. Sur un refus métier, jamais. Sur une expiration après émission, jamais avant d'avoir interrogé le rail. C'est la première cause de double paiement dans les architectures multi-rails.

7.2 Étape 8 — L'observabilité

Un paiement doit pouvoir être suivi de bout en bout, à travers toutes les couches, à partir de son seul identifiant. Le cadre impose donc des attributs normalisés que tous les connecteurs émettent de la même façon — c'est ce qui permet d'avoir un tableau de bord unique plutôt qu'un par connecteur.

Instrumentation normalisée par le cadre

```
// Attributs normalisés, identiques sur les dix connecteurs
span.setAttribute("oppl.payment_id",   payment.paymentId);
span.setAttribute("oppl.rail",         "pispi");
span.setAttribute("oppl.connector",    "connector-pispi@1.2.0");
span.setAttribute("oppl.message_type", "pacs.008");
span.setAttribute("oppl.state",        state.name());
span.setAttribute("oppl.amount_minor", payment.amount().minorUnits());
span.setAttribute("oppl.party_hash",   hash(party));   // jamais en clair
```

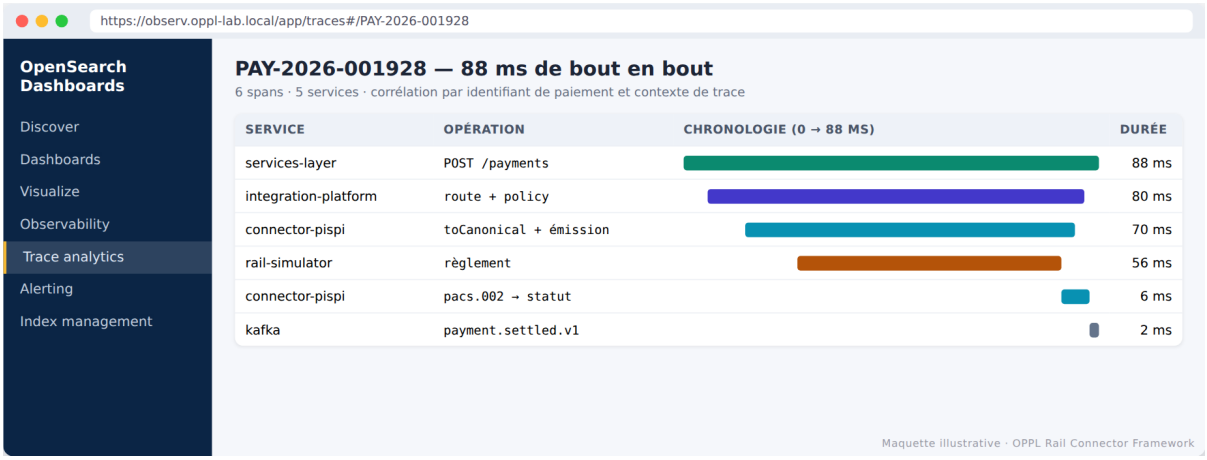


Figure 8 — Trace d'un paiement à travers la chaîne complète (capture illustrative)

Tableau 15 — Signaux surveillés pour chaque connecteur

Signal	Ce qu'il mesure	Seuil d'alerte type
Latence ajoutée par le connecteur	Temps passé hors rail	p95 supérieur au double de la référence
Taux d'erreur par code	Répartition des refus du rail	Déplacement brutal de la répartition
Transactions en état non terminal	Paiements en vol depuis trop longtemps	Plus de 15 minutes
Doublons détectés	Messages rejoués par le rail	Toute occurrence non expliquée
Disponibilité du rail	État du disjoncteur	Ouvert plus de 2 minutes
Expiration de certificat	Échéance des certificats du rail	30 jours avant
Débit et saturation	Charge et marge disponible	Au-delà de 70 % de la capacité mesurée



Figure 9 — Un tableau de bord identique pour les dix connecteurs (capture illustrative)

7.3 La piste d'audit

Au-delà des trois piliers habituels — journaux, métriques, traces — un connecteur de paiement produit une **piste d'audit** : l'archivage signé des messages échangés avec le rail, conservé pour une durée définie. Elle sert au rapprochement, à l'investigation d'incident et, le cas échéant, à la non-répudiation vis-à-vis de la contrepartie.

Ce que la piste d'audit ne doit pas contenir

Les messages archivés contiennent des données sensibles : identifiants de compte, alias, numéros de carte. L'archivage est chiffré, l'accès est tracé et restreint, et la durée de conservation est décidée avec la conformité — pas par défaut à l'infini.

7.4 Le rapprochement vu du connecteur

Le rapprochement complet appartient à la plateforme, qui confronte l'ensemble de la chaîne. Le connecteur en porte néanmoins une part indispensable : il est le seul à détenir la vérité de ce que le rail a réellement reçu et renvoyé.

Tableau 16 — La contribution du connecteur au rapprochement

Ce que le connecteur fournit	Pourquoi lui seul peut le fournir
La référence attribuée par le rail	Elle est générée par la contrepartie, jamais déduite
L'horodatage d'émission et de réception	Il situe la transaction dans la fenêtre de règlement du rail
Le code de statut d'origine	Il n'a de sens que dans le référentiel du rail
Le message brut archivé	C'est la preuve opposable en cas de litige
La liste des transactions en état non terminal	Lui seul sait ce qu'il a émis sans réponse
Le résultat des interrogations	Il est le seul à pouvoir interroger le rail

Trois écarts se détectent au niveau du connecteur et doivent faire l'objet d'un rapport quotidien : les messages émis sans réponse au-delà du délai contractuel, les réponses reçues sans demande correspondante, et les écarts entre le total émis et le total annoncé par le fichier de règlement du rail, lorsqu'il en produit un.

L'écart qui coûte le plus cher

Une réponse reçue sans demande correspondante signifie que le rail a traité quelque chose que le connecteur ne reconnaît pas. Deux causes possibles : une transaction émise puis perdue avant enregistrement local, ou un rejeu du rail. Dans les deux cas, il s'agit d'un incident majeur, pas d'un écart de rapprochement ordinaire.

8. Tests, simulation et certification

8.1 Étape 9 — Le simulateur avant le connecteur

Le principe le plus contre-intuitif du cadre, et le plus rentable : **on écrit le simulateur du rail avant le connecteur**. Attendre l'accès au rail réel pour commencer à développer est la meilleure façon de perdre trois mois ; développer sans pouvoir provoquer d'erreurs est la meilleure façon de livrer un connecteur qui ne tient pas en production.



Figure 10 — Scénarios d'un simulateur de rail (capture illustrative)

Tableau 17 — Les dix scénarios obligatoires du simulateur

Scénario	Ce que le simulateur renvoie	Ce que le connecteur doit faire
Succès	Acceptation et règlement confirmé	État terminal SETTLED, événement publié
Refus	Refus explicite motivé	État FAILED, code motif conservé, aucune nouvelle tentative
Expiration	Aucune réponse	Ne rien conclure, interroger le rail
Doublon	Réponse envoyée deux fois	Un seul effet, seconde réponse acquittée
Compte invalide	Bénéficiaire inconnu	Refus métier définitif
Message malformé	Réponse non conforme au schéma	Rejet d'analyse, alerte, pas de rejeu
Provision insuffisante	Refus pour solde	Refus métier définitif
Rail indisponible	Connexion refusée ou service arrêté	Disjoncteur ouvert, mise en attente durable
Erreur de sécurité	Certificat expiré, signature invalide	Arrêt immédiat, alerte de sécurité
Erreur de règlement	Règlement rejeté après acceptation	Compensation, incident majeur

Le dernier scénario mérite une mention particulière : un règlement rejeté après acceptation signifie que le paiement a été confirmé au client avant de se rétracter. C'est le cas le plus difficile du métier, et le seul que beaucoup d'implémentations ne traitent pas du tout.

8.2 Le jeu de tests

Chaque connecteur est livré avec un jeu de tests exécutable par un tiers. Sa composition est imposée : il compte davantage de cas négatifs que de cas nominaux, parce que c'est là que se logent les défauts coûteux.

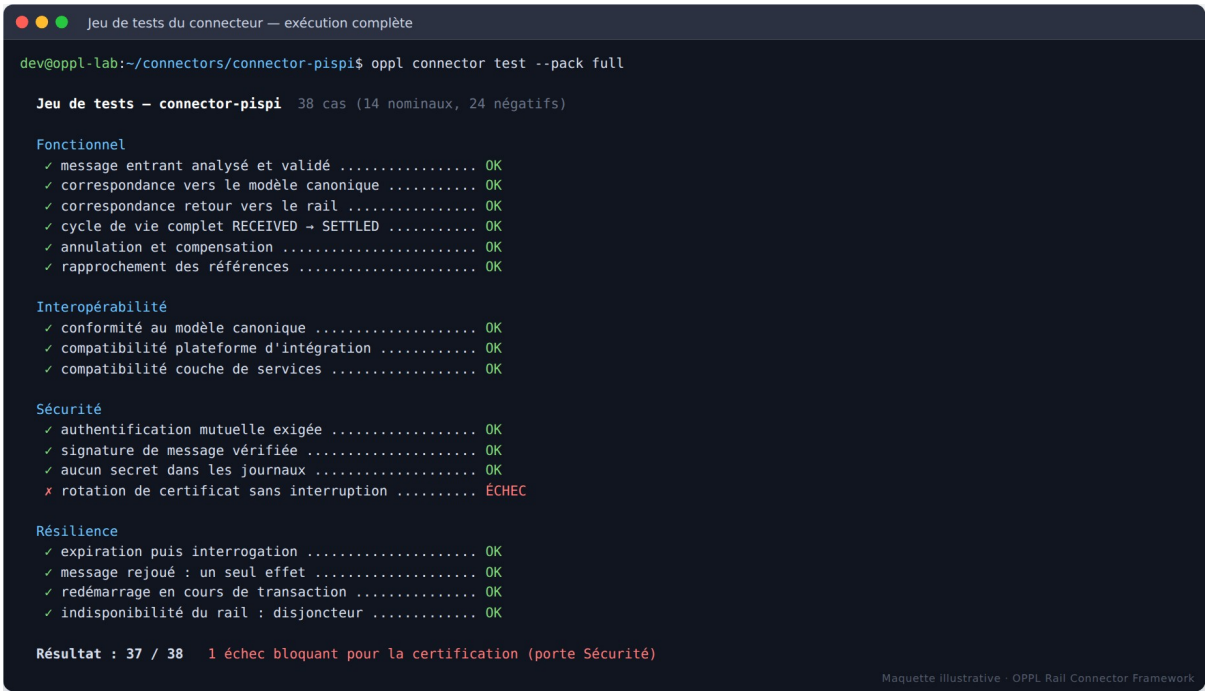


Figure 11 — Exécution du jeu de tests d'un connecteur (capture illustrative)

Tableau 18 — Composition type d'un jeu de tests

Famille de tests	Nombre indicatif	Exemples
Fonctionnels nominaux	14	Analyse, validation, correspondance aller et retour, cycle de vie complet
Erreurs du rail	8	Refus, compte invalide, provision, message malformé
Résilience	7	Expiration, rejeu, doublon, redémarrage, indisponibilité
Sécurité	5	Authentification absente, signature invalide, certificat expiré, rotation
Interopérabilité	4	Conformité au modèle canonique, compatibilité plateforme et services

8.3 Les six portes de certification

Un connecteur devient certifié après avoir franchi six portes, dans l'ordre. Une porte non franchie bloque la publication au catalogue : il n'existe pas de publication partielle, ni de dérogation.

Tableau 19 — Les six portes de certification

#	Porte	Ce qui est vérifié
1	Fonctionnelle	Messages, correspondances, cycle de vie transactionnel, erreurs, annulations, rapprochement
2	Interopérabilité	Conformité au modèle canonique, compatibilité avec la plateforme, la couche de services et le rail
3	Sécurité	Authentification, chiffrement, certificats, secrets, autorisation, piste d'audit

#	Porte	Ce qui est vérifié
4	Résilience	Expiration, nouvelle tentative, doublon, défaillance, redémarrage, reprise, indisponibilité du rail
5	Performance	Débit, latence, concurrence, consommation de ressources, stabilité dans la durée
6	Exploitation	Journaux, métriques, traces, alertes, documentation, procédures d'incident



Figure 12 — Suivi de certification d'un connecteur — une porte bloquée (capture illustrative)

Pourquoi l'ordre compte

Certifier la performance d'un connecteur dont la sécurité n'est pas validée n'a aucun sens : le correctif de sécurité changera les mesures. Les portes sont ordonnées parce que chacune suppose la précédente. Franchir les portes dans le désordre, c'est refaire deux fois le travail.

8.4 La performance, mesurée honnêtement

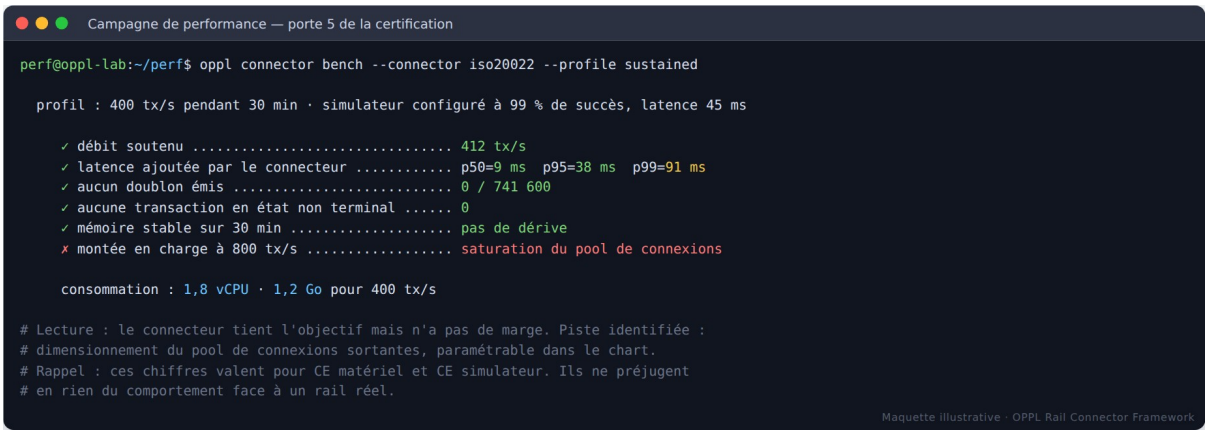


Figure 13 — Campagne de performance d'un connecteur (capture illustrative)

Les mesures portent sur ce que le connecteur ajoute, pas sur ce que le rail fait. La latence pertinente est la **latence ajoutée** : le temps passé à analyser, valider, traduire, sécuriser et journaliser, en dehors du temps d'attente de la contrepartie. C'est le seul chiffre sur lequel l'équipe a prise.

Rigueur des chiffres

Les résultats d'une campagne valent pour un matériel donné, un simulateur donné et une configuration donnée. Ils ne préjugent en rien du comportement face au rail réel, dont la latence, les limites et le comportement en saturation sont différents. Chaque campagne publie son protocole et ses résultats bruts.

9. Le kit de développement

9.1 Ce que chaque connecteur reçoit sans l'écrire

Le kit de développement est la matérialisation du cadre en code réutilisable. Sa fonction n'est pas de faire gagner du temps au premier connecteur — écrire le kit coûte plus cher que d'écrire un connecteur. Sa fonction est de rendre le deuxième, le cinquième et le dixième prévisibles, et de permettre qu'une correction bénéficie à tous.



Figure 14 — Modules du kit de développement (capture illustrative)

Tableau 20 — Les onze modules du kit

Module	Ce qu'il fournit	Conséquence si chaque connecteur le réécrit
connector-core	Cycle de vie, interfaces, configuration, démarrage, arrêt propre	Chaque connecteur invente sa structure ; aucune reprise possible
canonical-model	Objets du modèle, invariants, sérialisation, versionnage	Autant de modèles que de connecteurs — le couplage revient
protocol-library	Adaptateurs de transport, gestion de session, reprise de lien	Le transport contamine le code métier
security-library	Authentification mutuelle, jetons, signature, secrets, rotation	Le pire scénario : autant de cryptographies que de connecteurs
transaction-framework	Idempotence, corrélation, reprise, compensation, machine à états	Les doublons réapparaissent à chaque nouveau rail
error-framework	Taxonomie, conservation des codes du rail, remontée normalisée	Les erreurs deviennent illisibles en exploitation
observability-sdk	Traces, métriques, journaux, attributs normalisés	Un tableau de bord et des procédures par connecteur
reconciliation-sdk	Références, rapprochement, rapports d'écarts	Le rapprochement se découvre en production
testing-sdk	Jeu de tests type, assertions de conformité au cadre	Chacun a sa propre définition de « ça marche »

Module	Ce qu'il fournit	Conséquence si chaque connecteur le réécrit
simulator-framework	Simulateur paramétrable, injection de pannes, scénarios	Développement impossible sans accès au rail réel
connector-generator	Génération du projet complet depuis une fiche de rail	Deux semaines perdues au démarrage de chaque rail

9.2 Le générateur

Le générateur transforme une fiche de découverte en projet complet. Il ne devine rien : il applique des gabarits paramétrés par les réponses de l'ingénieur. Ce qu'il produit n'est pas un connecteur fini — c'est tout ce qui n'est pas spécifique au rail, déjà écrit, déjà testé, déjà conforme.

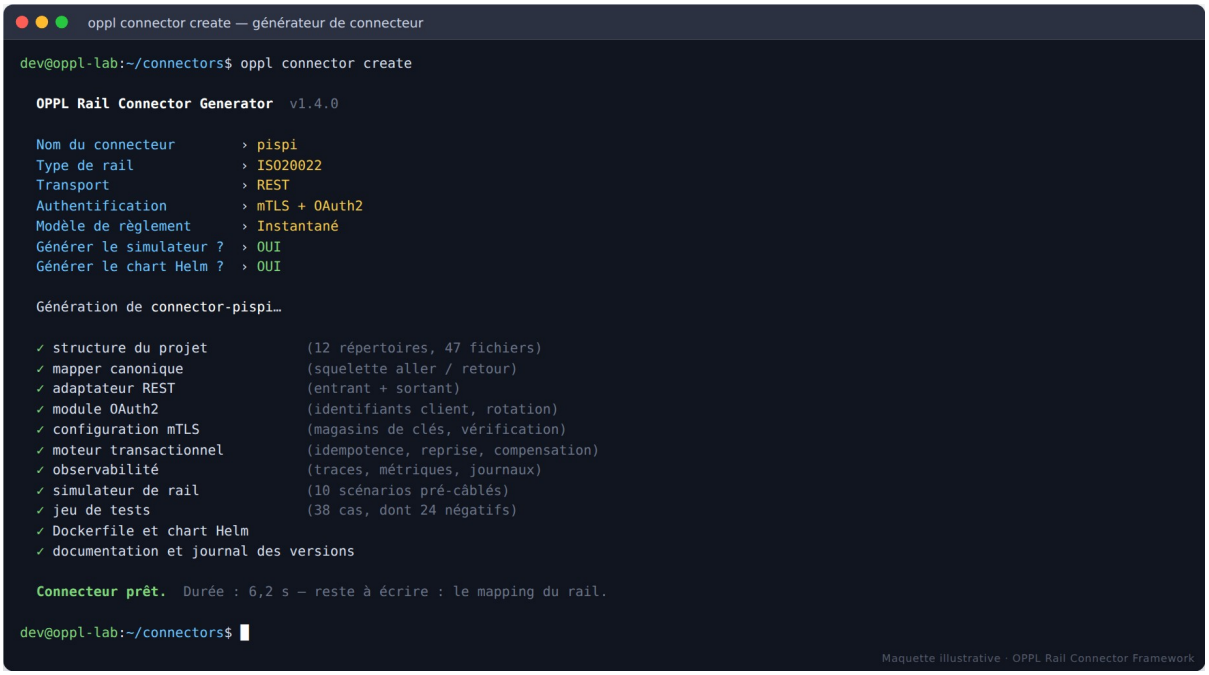


Figure 15 — Génération d'un connecteur depuis la ligne de commande (capture illustrative)

Tableau 21 — La frontière entre le généré et le spécifique

Ce que le générateur produit	Ce qui reste à écrire
Structure de projet complète et normalisée	Les tables de correspondance du rail
Squelette du mapper canonique, aller et retour	La logique d'analyse spécifique au format
Adaptateur du transport retenu	Les particularités de session du rail
Modules de sécurité configurés	Les certificats et paramètres du rail
Moteur transactionnel complet	Rien — il est générique
Observabilité complète	Rien — elle est générique
Simulateur avec dix scénarios pré-câblés	Les messages réels du rail dans les scénarios
Jeu de tests structuré	Les assertions propres au rail
Dockerfile, chart, documentation, journal des versions	Les particularités d'exploitation du rail

L'indicateur qui compte

La qualité du kit se mesure à une proportion : la part du connecteur qui est spécifique au rail. Si elle dépasse 30 %, le kit est incomplet et quelque chose de générique a été laissé à la charge de chaque équipe. Cette proportion est suivie connecteur après connecteur — elle doit décroître.

9.3 Versionner le kit

Un kit partagé par dix connecteurs devient rapidement une dépendance critique. Trois disciplines s'imposent dès le début, faute de quoi la mise à jour du kit devient impossible.

- **Versionnage sémantique strict.** Une rupture de compatibilité change la version majeure, sans exception et sans négociation.
- **Compatibilité descendante par défaut.** Un connecteur doit pouvoir rester deux versions mineures en arrière sans être bloqué.
- **Tests de non-régression sur tous les connecteurs.** Toute modification du kit déclenche l'exécution des dix jeux de tests avant publication.

10. La fabrique de connecteurs

10.1 Du projet au processus

La fabrique est ce qui transforme une méthode écrite en pratique effective. Elle enchaîne les étapes du cadre dans une chaîne automatisée, s'arrête à la première porte non franchie et ne publie au catalogue qu'un connecteur intégralement certifié.



Figure 16 — Exécution de la chaîne de fabrication (capture illustrative)

Tableau 22 — Ce que la fabrique automatise, et ce qu'elle n'automatise pas

Étape de la chaîne	Automatisée ?	Ce qui reste humain
Spécification du rail	Non	Lecture de la documentation, entretiens avec la contrepartie
Découverte et fiche	Partiellement	Validation de la complétude par un architecte
Génération du projet	Oui	—
Correspondances	Non	Le cœur du travail d'ingénierie
Configuration	Oui (validation du schéma)	Choix des valeurs
Sécurité	Oui (analyse, vérification)	Obtention des certificats
Tests automatisés	Oui	Écriture des assertions spécifiques
Simulation du rail	Oui	Construction des messages réalistes
Performance	Oui	Interprétation des résultats
Sécurité applicative	Oui	Analyse des découvertes
Certification	Oui (contrôle des portes)	Décision de dérogation — qui n'existe pas
Empaquetage et publication	Oui	—

La colonne de droite est la plus instructive. La fabrique n'automatise pas l'intelligence : elle automatise tout le reste, pour que l'intelligence se concentre sur les correspondances et sur la compréhension du rail.

10.2 Étape 10 — L'empaquetage

Un connecteur certifié devient un produit logiciel versionné. Le critère d'acceptation est précis : **un exploitant qui n'a pas écrit le connecteur doit pouvoir le déployer, le configurer et le dépanner avec le seul paquet**.

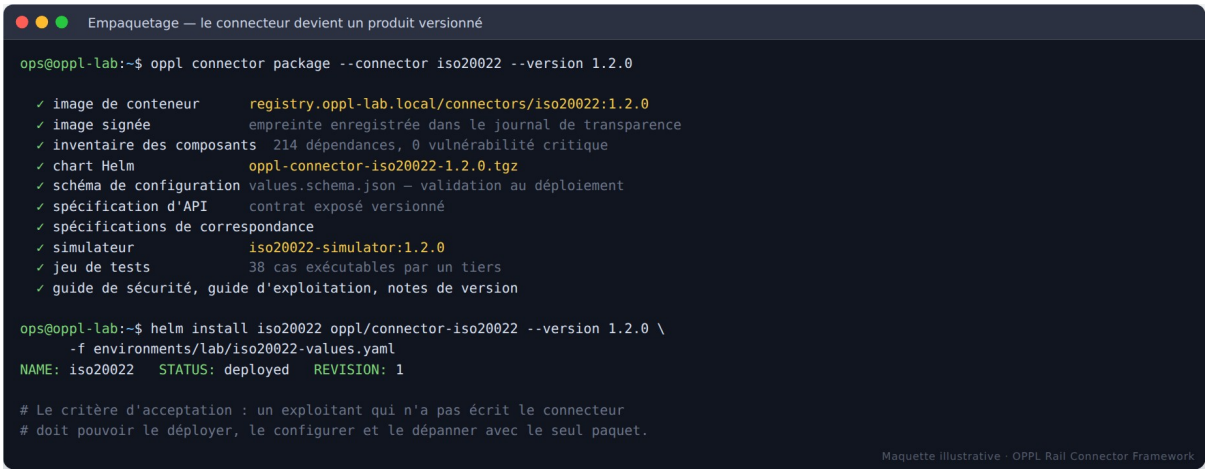


Figure 17 — Empaquetage d'un connecteur certifié (capture illustrative)

Tableau 23 — Les dix livrables d'un connecteur empaqueté

Livrable	Contenu	Pourquoi il est obligatoire
Image de conteneur signée	Le connecteur et ses dépendances	Traçabilité et intégrité de ce qui est déployé
Inventaire des composants	Liste des dépendances et de leurs versions	Réaction rapide à une vulnérabilité publiée
Chart de déploiement	Ressources, sondes, ressources requises, secrets	Déploiement reproductible sans connaissance interne
Schéma de configuration	Paramètres, types, valeurs par défaut, validation	Une erreur de configuration échoue au déploiement, pas en production
Spécification d'API	Contrat exposé par le connecteur	Intégration sans lecture du code
Spécifications de correspondance	Tables de mapping documentées	Investigation d'un écart sans l'auteur
Simulateur	Image du simulateur du rail	Reproduction d'un incident hors production
Jeu de tests	Tests exécutables par un tiers	Vérification après montée de version
Guides de sécurité et d'exploitation	Procédures, alertes, incidents connus	Exploitation par une équipe différente
Notes de version	Changements, ruptures, migration	Montée de version maîtrisée

10.3 Le catalogue

Le catalogue est la vitrine et l'inventaire de la fabrique. Il indique pour chaque connecteur sa famille, son transport, sa version, son état de certification et son simulateur associé.

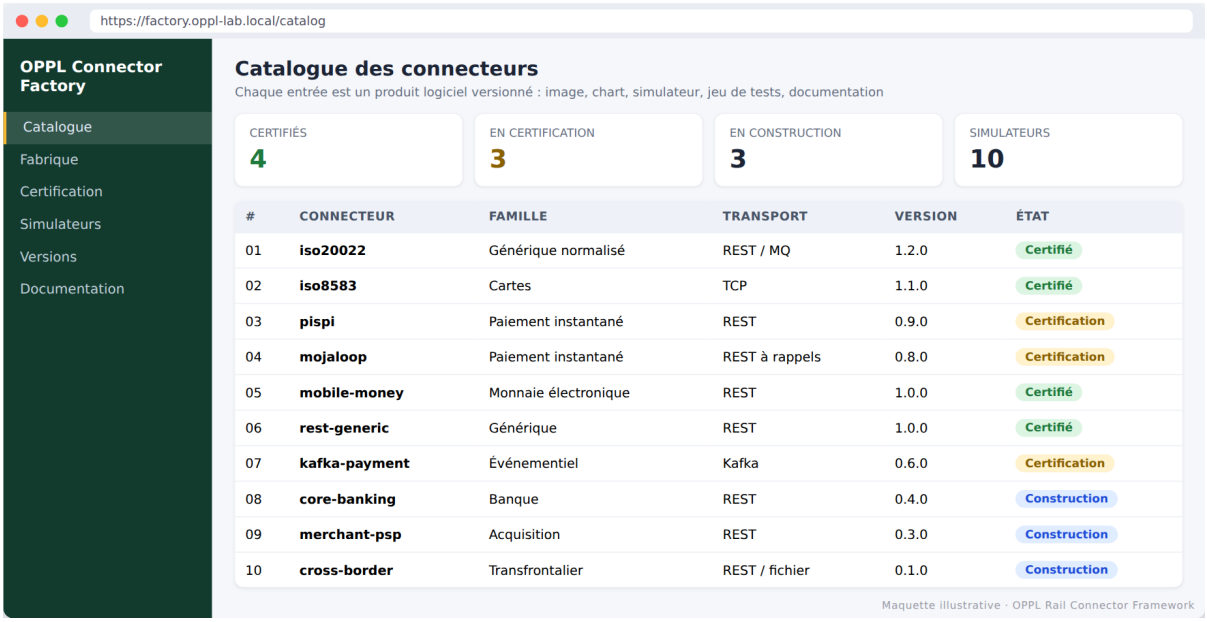


Figure 18 — Catalogue des connecteurs (capture illustrative)

10.4 Les dix premiers connecteurs

L'ordre de construction n'est pas arbitraire. Il commence par les connecteurs génériques normalisés — ceux dont les gabarits serviront ensuite aux autres — et termine par les plus spécifiques.

Tableau 24 — Ordre de construction des dix premiers connecteurs

#	Connecteur	Pourquoi à ce rang	Ce qu'il apporte aux suivants
01	ISO 20022	Norme la plus structurante, vocabulaire de référence	Le gabarit du modèle canonique
02	ISO 8583	Deuxième norme majeure, monde très différent	La preuve que le modèle canonique tient sur deux mondes
03	Paiement instantané régional	Cible prioritaire de l'écosystème	Le gabarit d'un rail à règlement instantané
04	Interopérabilité instantanée	Modèle à rappels asynchrones	La gestion des rappels et de la corrélation
05	Monnaie électronique	Rail le plus utilisé dans la zone	Le traitement des confirmations tardives
06	Générique REST	Socle de tous les rails modernes	Le gabarit réutilisable le plus fréquent
07	Événementiel	Intégration interne et bus de place	La consommation idempotente d'événements
08	Cœur bancaire	Frontière entre paiement et comptes	Le lien avec le système de comptes
09	Acquisition marchande	Entrée du commerce dans la chaîne	Les spécificités de l'acquisition
10	Transfrontalier	Le plus complexe : devises, frais, correspondants	La conversion et les frais dans le modèle

10.5 Journal de bord du troisième connecteur

La promesse du cadre — un connecteur en quelques semaines plutôt qu'en quelques mois — mérite d'être illustrée concrètement. Le déroulé ci-dessous décrit une production type une fois le kit et le générateur disponibles.

Tableau 25 — Production type d'un connecteur — environ six semaines

Jour	Activité	Produit par	Durée
1 à 3	Découverte du rail : lecture des spécifications, entretiens, fiche	L'ingénieur	3 j
4	Revue de la fiche par l'architecte, complétion des zones grises	Architecte + ingénieur	0,5 j
4	Génération du projet depuis la fiche	Le générateur	10 min
5 à 9	Construction du simulateur : dix scénarios, messages réalistes	L'ingénieur	5 j
10 à 16	Tables de correspondance aller et retour, champs non mappés documentés	L'ingénieur	7 j
17 à 19	Particularités du transport et de la session	L'ingénieur	3 j
20 à 21	Configuration de la sécurité, certificats de test	Ingénieur + sécurité	2 j
22 à 25	Jeu de tests : assertions spécifiques, cas négatifs	L'ingénieur	4 j
26 à 27	Passage des portes 1 à 4	La fabrique	2 j
28	Campagne de performance, porte 5	La fabrique	0,5 j
29 à 30	Documentation, guide d'exploitation, porte 6	L'ingénieur	2 j
31	Empaquetage et publication au catalogue	La fabrique	2 h

Deux enseignements ressortent de ce déroulé. D'abord, la part générée est marginale en temps — dix minutes — mais elle supprime les tâches où l'on se trompe : structure, sécurité, moteur transactionnel, observabilité. Ensuite, les deux postes les plus longs sont le **simulateur** et les **correspondances** : ce sont précisément les deux seules activités réellement spécifiques au rail.

Tableau 26 — Répartition du temps de production

Poste	Part du temps	Spécifique au rail ?
Découverte et fiche	11 %	Oui
Simulateur	17 %	Oui
Correspondances	23 %	Oui
Transport et session	10 %	Partiellement
Sécurité	7 %	Partiellement — configuration seulement
Tests	13 %	Partiellement
Certification	8 %	Non
Documentation et empaquetage	11 %	Partiellement

L'indicateur à suivre dans la durée

La part réellement spécifique au rail doit décroître à mesure que le catalogue s'enrichit : un nouveau rail ISO 20022 réutilisera le gabarit du premier, un nouveau rail de monnaie électronique celui du précédent. Le quatrième connecteur d'une même famille doit coûter nettement moins que le premier.

11. Le laboratoire d'interopérabilité

11.1 Faire dialoguer des rails qui ne se connaissent pas

La fabrique produit des connecteurs. Le laboratoire d'interopérabilité démontre qu'ils fonctionnent **ensemble** — ce qui n'est pas la même chose. Il assemble deux connecteurs autour de la plateforme et fait circuler un paiement d'un rail vers un autre, sans qu'aucune interface n'existe entre eux.



Figure 19 — Scénarios d'interopérabilité (capture illustrative)

Tableau 27 — Les six scénarios d'interopérabilité

Scénario	Chaîne	Ce qu'il démontre
Carte → instantané	Rail carte → connecteur → plateforme → connecteur instantané → simulateur	Deux normes très différentes convergent vers un modèle unique
Portefeuille → banque	Monnaie électronique → connecteur → plateforme → connecteur bancaire	L'interopérabilité wallet-banque sans interface dédiée
Banque → portefeuille	Cœur bancaire → connecteur → plateforme → connecteur monnaie électronique	Le sens inverse fonctionne symétriquement
Instantané → transfrontalier	Rail régional → connecteur → plateforme → connecteur correspondant	Conversion de devise et frais dans le modèle canonique
Bascule de rail	Rail principal indisponible → repli par un autre connecteur	Le repli ne produit aucun doublon
Panne en cours de chaîne	Interruption entre la plateforme et le connecteur	Reprise sans perte ni double émission

Ce que ces scénarios prouvent vraiment

Le modèle canonique n'est pas une élégance d'architecte. C'est la seule raison pour laquelle un paiement initié par carte peut se terminer sur un rail instantané sans qu'on ait jamais écrit d'interface entre les deux. Le laboratoire d'interopérabilité est la démonstration empirique de cette affirmation.

11.2 Le déploiement du laboratoire

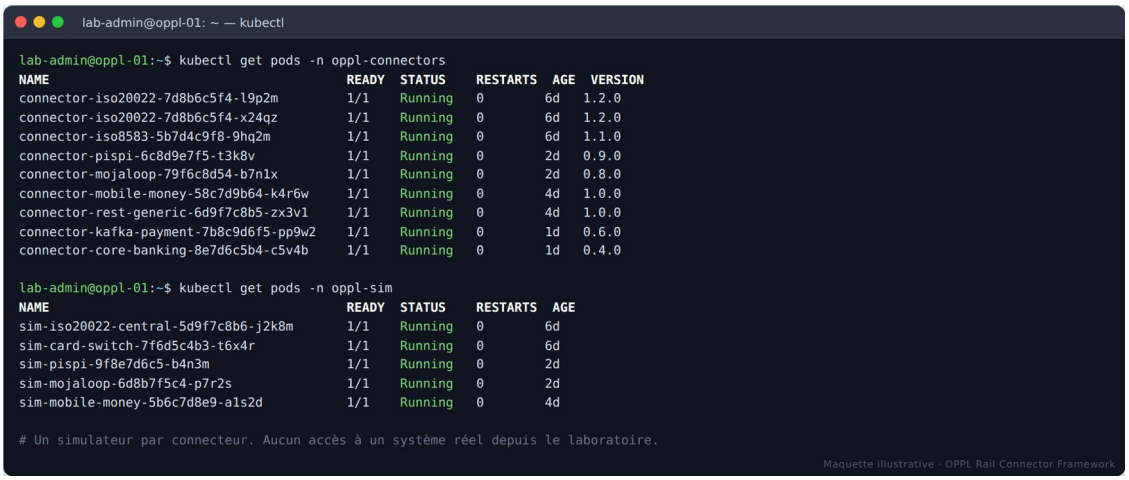


Figure 20 — Connecteurs et simulateurs déployés (capture illustrative)

Chaque connecteur est déployé avec le simulateur du rail qu'il adresse. Aucun accès à un système réel n'existe depuis le laboratoire : c'est une contrainte de sécurité, mais c'est aussi ce qui permet de provoquer des pannes sans conséquence.

Tableau 28 — Dimensionnement indicatif du laboratoire

Domaine	vCPU	RAM	Remarques
Socle (ingress, GitOps, secrets, registre)	4	8 Go	Registre local recommandé
Identité et passerelle	3	6 Go	
Plateforme d'intégration et orchestration	5	10 Go	Moteur de processus durable
Dix connecteurs (2 instances chacun pour les certifiés)	8	14 Go	≈ 0,8 vCPU et 1,4 Go par connecteur
Simulateurs de rails	4	6 Go	Un par connecteur
Bus d'événements	3	6 Go	
Observabilité	5	10 Go	Traces, métriques, journaux, tableaux de bord
Fabrique (chaîne d'intégration, registre, tests)	4	8 Go	Peut être mutualisée
Total indicatif	≈ 36	≈ 68 Go	3 nœuds de 16 vCPU / 64 Go avec marge

11.3 Les indicateurs du laboratoire

Tableau 29 — Indicateurs du laboratoire d'interopérabilité

Indicateur	Ce qu'il mesure	Cible
Scénarios croisés verts	Capacité réelle d'interopérabilité	6 sur 6
Rails connectés au catalogue	Couverture de l'écosystème	10 connecteurs certifiés
Doublons observés en campagne	Intégrité de la chaîne	0, sans exception
Temps de reconstruction du laboratoire	Reproductibilité réelle	Moins d'une journée

Indicateur	Ce qu'il mesure	Cible
Connecteurs exploités par une équipe qui ne les a pas écrits	Qualité de l'empaquetage	100 %

11.4 Ce que le laboratoire ne prouve pas

Un scénario vert dans le laboratoire démontre que la chaîne fonctionne face à des simulateurs. Il ne démontre pas qu'elle fonctionnera face aux rails réels. Trois différences persistent, qu'il faut énoncer chaque fois qu'un résultat est présenté.

- **Les simulateurs sont plus polis que la réalité.** Ils respectent leurs propres spécifications ; les rails réels s'en écartent parfois, et les écarts non documentés sont la première source de surprise à l'homologation.
- **La latence et les limites diffèrent.** Un rail réel impose des plafonds de débit, des fenêtres de maintenance et des heures limites que le simulateur n'a aucune raison de reproduire fidèlement.
- **L'homologation est un processus, pas un test.** Se connecter à un système réel suppose des accords, des certificats, des tests conjoints et une validation par l'opérateur. Le laboratoire prépare cette étape ; il ne la remplace pas.

12. La pile technique et sa qualification

12.1 Les briques candidates

Le cadre n'impose pas une pile : il impose une méthode de qualification. Les technologies ci-dessous sont des candidates crédibles, à évaluer selon le contexte de l'institution — jamais à adopter sur leur seule réputation.

Tableau 30 — Briques candidates et licences

Domaine	Candidates	Licence (à revérifier)
Intégration	Apache Camel, hubs de paiement open source, plateformes d'interopérabilité	Apache 2.0 pour Camel ; à vérifier au cas par cas pour les autres
Messagerie et événements	Apache Kafka, Apache ActiveMQ Artemis, RabbitMQ	Apache 2.0 pour les deux premiers ; MPL 2.0 pour RabbitMQ
Passerelle d'API	KrakenD, Gravitee, Apache APISIX, Kong	Apache 2.0 pour les éditions communautaires
Identité	Keycloak	Apache 2.0
Politique	Open Policy Agent	Apache 2.0
Processus durables	Temporal	MIT
Observabilité	OpenTelemetry, Prometheus, OpenSearch, Grafana, Loki	Apache 2.0 pour les trois premiers ; AGPLv3 pour Grafana et Loki
Secrets	OpenBao	Mozilla Public License 2.0
Exécution	Docker, Kubernetes, Helm, Argo CD	Apache 2.0

Un point de vigilance vérifié

Camunda 7 Community Edition, souvent cité comme moteur de processus open source, a atteint sa fin de vie. Sa dernière version de fonctionnalités est sortie en octobre 2025, son dépôt a été archivé et l'édition communautaire ne reçoit plus de correctifs de sécurité. Pour une infrastructure de paiement, retenir aujourd'hui ce composant reviendrait à bâtir sur une base qui ne sera plus corrigée. Le cadre s'oriente vers un moteur de processus durable activement maintenu ; l'édition entreprise de Camunda 7 reste supportée, mais relève d'un modèle commercial et non de l'open source.

Deux autres vigilances méritent d'être signalées. **Grafana et Loki** relèvent d'une licence à réciprocité forte : l'usage interne est sans difficulté, mais la redistribution ou l'exposition d'un service dérivé appelle une analyse. **Les passerelles d'API** disposent toutes d'une édition communautaire permissive et d'une édition entreprise : il faut vérifier que les fonctions retenues relèvent bien de la première.

12.2 La méthode de qualification

Le caractère open source d'une brique ne la qualifie pas pour une infrastructure de paiement critique. Dix étapes, chacune produisant une preuve, conduisent de la découverte à l'industrialisation.

Le tunnel de qualification

```
DÉCOUVRIR → INSTALLER → COMPRENDRE → INTÉGRER → TESTER
     → SÉCURISER → MESURER → QUALIFIER → DOCUMENTER → INDUSTRIALISER
```

Tableau 31 — Grille de qualification (pondération indicative)

Critère	Poids	Ce qui est évalué
Licence et modèle économique	15 %	Nature de la licence, offre commerciale, réversibilité, risque de changement de modèle
Gouvernance et communauté	15 %	Fondation ou éditeur unique, contributeurs, rythme des versions, cycle de vie annoncé
Sécurité	15 %	Gestion des vulnérabilités, délai de correction, historique
Exploitabilité	15 %	Sauvegarde, restauration, observabilité, montée de version, haute disponibilité
Adéquation fonctionnelle	10 %	Couverture des besoins, écarts identifiés
Performance et disponibilité	10 %	Résultats mesurés, comportement en saturation
Compétences disponibles	10 %	Disponibilité locale des profils, coût de formation
Documentation	5 %	Complétude, fraîcheur, exemples
Support	5 %	Communauté, éditeurs, intégrateurs régionaux

Le critère que l'exemple de Camunda met en lumière

Le critère « gouvernance et communauté » doit inclure explicitement le **cycle de vie annoncé**. Une brique techniquement excellente dont l'édition communautaire arrive en fin de support dans dix-huit mois est un mauvais choix pour une infrastructure qui vivra dix ans. Cette question se pose avant l'intégration, jamais après.

13. Mise en œuvre et déploiement

13.1 Séquencement

Construire la fabrique avant le premier connecteur serait absurde : on ne conçoit pas une chaîne de production avant de savoir ce qu'elle produit. Le séquencement retenu construit le premier connecteur à la main, extrait le cadre du deuxième, puis industrialise à partir du troisième.

Tableau 32 — Séquencement de la mise en œuvre

Phase	Contenu	Durée	Critère de sortie
0	Socle : orchestration, GitOps, secrets, registre, observabilité	2 sem.	Déploiement reproductible depuis Git
1	Premier connecteur, écrit à la main	4 sem.	Un rail simulé fonctionne de bout en bout
2	Extraction du modèle canonique et des invariants	2 sem.	Modèle versionné, validé par deux rails sur papier
3	Deuxième connecteur — norme très différente	3 sem.	Le modèle canonique tient sans modification majeure
4	Extraction du kit de développement	4 sem.	Les deux connecteurs reposent sur le kit
5	Cadre de simulation et jeu de tests type	3 sem.	Les dix scénarios sont exécutables
6	Générateur de connecteur	3 sem.	Un troisième connecteur généré en moins d'une journée
7	Portes de certification et fabrique	3 sem.	La chaîne bloque à la première porte non franchie
8	Connecteurs 3 à 6 et catalogue	6 sem.	Quatre connecteurs certifiés publiés
9	Laboratoire d'interopérabilité	2 sem.	Trois scénarios croisés fonctionnent
10	Connecteurs 7 à 10 et parcours de formation	8 sem.	Dix connecteurs, une promotion formée

L'erreur de séquencement la plus fréquente

Vouloir écrire le kit et le générateur d'emblée, avant d'avoir construit un seul connecteur. Le résultat est un cadre élégant qui ne correspond à aucun rail réel, et qu'il faut réécrire dès le premier contact avec la réalité. Le cadre s'**extrait** de l'expérience ; il ne se conçoit pas dans l'abstrait.

13.2 Organisation des dépôts

Quatre dépôts, quatre responsabilités

```
oppl-connector-sdk/            # le kit — dépôt unique, versionné
├── connector-core/  canonical-model/  protocol-library/
├── security-library/  transaction-framework/  error-framework/
├── observability-sdk/  reconciliation-sdk/  testing-sdk/
├── simulator-framework/
└── connector-generator/        # gabarits + ligne de commande

oppl-connectors/               # un répertoire par connecteur
├── connector-iso20022/  connector-iso8583/  connector-pispi/
├── connector-mojaloop/  connector-mobile-money/ …

oppl-factory/                  # la chaîne de fabrication
├── pipelines/  gates/  templates/  catalog/

oppl-gitops/                   # état désiré des environnements
└── environments/lab/  environments/factory/
```

13.3 Déployer un connecteur

Déploiement d'un connecteur certifié

```
# Déploiement depuis le catalogue, sans connaissance du code
helm install iso20022 oppl/connector-iso20022 --version 1.2.0 \
     -f environments/lab/iso20022-values.yaml

# La configuration est validée par son schéma : une erreur échoue ici,
# pas en production.
helm lint oppl/connector-iso20022 -f environments/lab/iso20022-values.yaml

# Le simulateur se déploie de la même façon
helm install sim-iso20022 oppl/simulator-iso20022 --version 1.2.0
```

13.4 Exploiter dix connecteurs comme un seul

L'uniformité produit son bénéfice au moment de l'exploitation. Parce que les dix connecteurs partagent la même structure, la même configuration, les mêmes attributs d'observabilité et les mêmes procédures, une équipe d'astreinte peut les traiter tous.

Tableau 33 — Alertes communes aux dix connecteurs

Alerte	Seuil	Première action
Transactions en état non terminal	> 15 min	Vérifier l'interrogation planifiée du rail concerné
Doublon détecté	Toute occurrence	Incident majeur : geler, tracer, analyser la cause
Disjoncteur ouvert	> 2 min	Vérifier le rail, informer la plateforme, suivre la file d'attente
Latence ajoutée anormale	p95 au-delà du double de la référence	Vérifier la charge et la saturation des pools
Déplacement de la répartition des codes d'erreur	Écart brutal	Souvent le signe d'un changement non annoncé côté rail
Certificat proche de l'expiration	30 jours	Planifier la rotation, la tester hors production

13.5 Sauvegarde, reprise et réinitialisation

- **Sauvegarde** : l'état transactionnel des connecteurs et la piste d'audit sont sauvegardés quotidiennement.
- **Reprise** : testée chaque mois et chronométrée — une sauvegarde jamais restaurée n'est pas une sauvegarde.
- **Réinitialisation** : le laboratoire revient à un état de référence depuis les dépôts, avant chaque session de formation ou campagne de mesure.
- **Vérification post-reprise** : une reprise se conclut par le rejeu du jeu de tests, jamais par un simple démarrage réussi des services.

14. Former des ingénieurs de rail

14.1 Un parcours en six niveaux

La fabrique constitue un environnement de formation naturel : elle produit des livrables vérifiables, elle impose une méthode, et elle rend l'échec observable sans conséquence. Le parcours conduit d'un profil junior à un ingénieur capable de construire un connecteur réel.



Figure 21 — Parcours en six niveaux (capture illustrative)

Tableau 34 — Les six niveaux du parcours

Niveau	Domaine	Contenu	Preuve attendue
1	Protocoles	HTTP, REST, SOAP, TCP, JSON, XML, files de messages, événements	Échanges réalisés sur chaque transport
2	Paiements	ISO 8583, ISO 20022, cycle de vie, compensation, règlement, rapprochement	Analyse d'un message réel et de son cycle
3	Intégration	Routes, API, architecture événementielle, correspondances, modèle canonique	Table de correspondance documentée
4	Sécurité	Jetons, authentification mutuelle, infrastructure de clés, signature, secrets	Échange sécurisé de bout en bout
5	Plateforme	Conteneurs, orchestration, charts, GitOps, observabilité, chaîne d'intégration	Connecteur déployé et observé
6	Ingénierie de rail	Construction complète d'un connecteur et de son écosystème	Le livrable complet décrit ci-dessous

14.2 Le livrable, pas la présentation

La règle du parcours

Un exercice ne se termine pas par une présentation. Il se termine par un connecteur qui fonctionne, son simulateur, ses tests automatisés, son image de conteneur, sa documentation et une démonstration exécutée devant le jury. Ce qui n'est pas exécutable n'est pas évalué.

Tableau 35 — Les sept livrables du niveau 6

Livrable	Critère d'acceptation
Code source	Structure conforme au cadre, aucune cryptographie réécrite
Image de conteneur	Construite par la chaîne, signée, sans vulnérabilité critique
Connecteur fonctionnel	Les six portes de certification franchies
Simulateur	Les dix scénarios obligatoires implémentés
Tests automatisés	Plus de cas négatifs que de cas nominaux, exécutables par un tiers
Documentation	Un exploitant qui n'a pas écrit le connecteur peut le dépanner
Démonstration	Un paiement traverse la chaîne en direct, y compris un chemin d'erreur

14.3 Progression indicative

Tableau 36 — Progression sur six mois

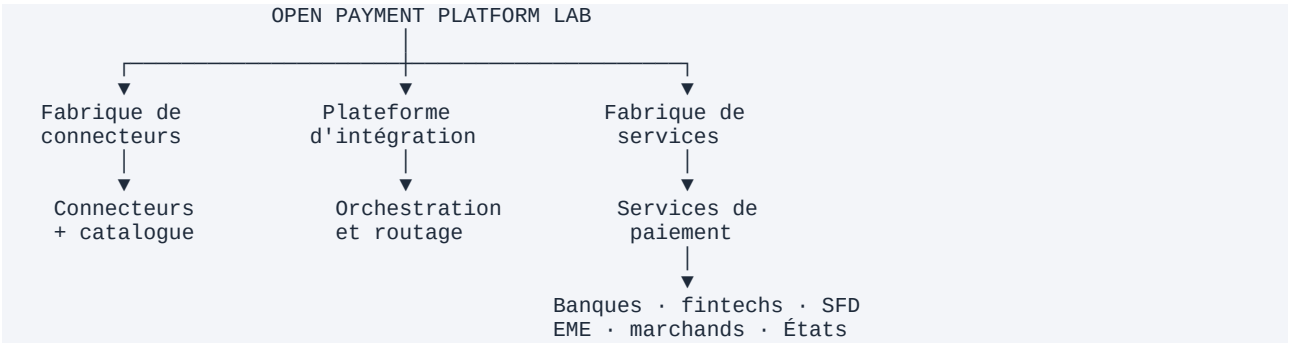
Période	Niveaux	Compétence acquise
Semaines 1 à 3	Protocoles	Dialoguer avec n'importe quel système, quel que soit le transport
Semaines 4 à 7	Paiements	Lire un message normalisé et comprendre le cycle de vie d'un paiement
Semaines 8 à 10	Intégration	Concevoir une correspondance vers un modèle canonique
Semaines 11 à 13	Sécurité	Sécuriser un échange interbancaire de bout en bout
Semaines 14 à 16	Plateforme	Déployer, observer et exploiter un connecteur
Semaines 17 à 24	Ingénierie de rail	Construire, certifier et documenter un connecteur complet

15. Industrialisation et proposition de valeur

15.1 Le modèle cible

Le cadre s'inscrit dans une organisation à trois fabriques, chacune avec son produit et son rythme propres.

Le modèle à trois fabriques



15.2 Ce qui constitue la propriété stratégique

Il serait tentant de considérer que la valeur réside dans le code des connecteurs. C'est l'inverse : un connecteur se réécrit en quelques semaines une fois que l'on sait comment faire. Ce qui ne se réécrit pas, c'est la capacité à savoir comment faire.

Tableau 37 — Ce qui se reconstitue vite, et ce qui ne se reconstitue pas

Élément	Ce qu'il représente	Temps de reconstitution
La méthode	Les dix étapes, les six portes, l'ordre	Des années — elle s'apprend par l'échec
Le modèle canonique	Le vocabulaire commun à tous les rails	Plusieurs mois, et seulement après plusieurs rails
Le kit de développement	Le code générique éprouvé	Six mois à un an
La fabrique	La chaîne automatisée et ses portes	Plusieurs mois
Les simulateurs	La capacité à développer sans le rail réel	Un mois par rail
Les jeux de tests	La définition partagée de « ça marche »	Se construit incident après incident
La certification	Le niveau d'exigence et sa constance	Se gagne par la discipline
Le catalogue	L'inventaire exploitable	Immédiat une fois le reste en place
Les compétences	Les ingénieurs formés	Des années — et elles partent avec eux
Le code d'un connecteur	Un composant remplaçable	Quelques semaines

15.3 La proposition de valeur

Le déplacement est le suivant. L'objectif n'est plus de **développer des interfaces de paiement**, activité artisanale dont le coût croît avec le nombre de rails. Il devient d'**industrialiser la capacité à connecter de nouveaux rails**, activité dont le coût marginal décroît.

Tableau 38 — Ce que change le cadre

Indicateur	Sans cadre	Avec le cadre (cible)
Durée du premier connecteur	3 à 6 mois	3 à 6 mois — identique
Durée du cinquième connecteur	3 à 6 mois	3 à 5 semaines
Part de code spécifique au rail	80 à 100 %	Moins de 30 %
Correctif de sécurité appliqué	Une fois par connecteur	Une fois pour tous
Exploitation	Une procédure par connecteur	Une procédure commune
Nouvel arrivant opérationnel	Sur un connecteur	Sur tous les connecteurs
Ajout d'un rail	Renégociation de l'architecture	Un connecteur de plus

15.4 Offres de service envisageables

Tableau 39 — Offres envisageables

Offre	Contenu	Bénéficiaires	Prérequis
Connector Factory	Production industrialisée de connecteurs selon le cadre	Banques, fintechs, opérateurs, switchs	Cadre mature, dix connecteurs au catalogue
Certification de connecteur	Passage des six portes pour un connecteur tiers	Éditeurs, intégrateurs	Référentiel publié, gouvernance de la certification
Sandbox d'interopérabilité	Environnement de test croisé entre rails	Participants d'un écosystème	Simulateurs, isolation par participant
Rail Engineering Academy	Parcours de six niveaux, livrables exécutables	Jeunes diplômés, équipes internes	Laboratoire stable, formateurs formés
Qualification de briques	Application du tunnel et de la grille	Institutions engageant un choix d'architecture	Méthode publiée, indépendance de l'évaluation

Précision indispensable

La certification délivrée par le cadre est une **conformité à un référentiel technique interne**. Elle n'a aucune valeur réglementaire et ne remplace ni l'homologation d'un opérateur de système de paiement, ni une certification de schéma, ni un agrément. Cette limite doit figurer dans tout document remis à un tiers.

16. Gouvernance, risques et feuille de route

16.1 Organisation

Tableau 40 — Organisation type

Rôle	Responsabilités	Charge indicative
Sponsor	Porte l'initiative, arbitre, protège la discipline des portes	Ponctuelle
Responsable du cadre	Méthode, modèle canonique, kit, versions, cohérence d'ensemble	100 %
Architecte d'intégration	Correspondances, revues, décisions documentées	100 %
Ingénieurs connecteurs (3)	Construction, simulateurs, tests	100 %
Ingénieur plateforme	Fabrique, chaîne, déploiement, observabilité	100 %
Responsable sécurité	Bibliothèque de sécurité, porte 3, revue des secrets	50 %
Formateurs (2)	Parcours, évaluation, accompagnement	50 %

Le rôle de **responsable du cadre** est celui qu'on omet le plus souvent, et son absence se paie immédiatement : sans quelqu'un dont la mission est de défendre le modèle canonique contre les ajouts opportunistes, celui-ci se dégrade en quelques mois jusqu'à redevenir une collection de champs propres à chaque rail.

16.2 Risques et parades

Tableau 41 — Principaux risques

Risque	Probabilité	Impact	Parade
Le modèle canonique se dilue rail après rail	Élevée	Élevé	Responsable du cadre, revue obligatoire de toute extension
Une logique de routage se glisse dans un connecteur	Élevée	Élevé	Revue de code, contrôle automatique à la porte 2
Une dérogation est accordée sur une porte	Moyenne	Élevé	Aucune dérogation. La règle ne souffre pas d'exception
Le kit est écrit avant le premier connecteur	Moyenne	Élevé	Séquencement imposé : extraire, ne pas concevoir
Un connecteur réimplémente la sécurité	Moyenne	Élevé	Refus automatique à la porte 3
Une brique en fin de vie est retenue	Moyenne	Élevé	Critère de cycle de vie dans la grille de qualification
Les simulateurs sont trop permissifs	Élevée	Moyen	Les dix scénarios sont obligatoires, pas indicatifs
Dépendance à quelques experts	Élevée	Moyen	Uniformité, documentation, rotation sur les connecteurs

16.3 Feuille de route

Tableau 42 — Feuille de route sur dix-huit mois

Période	Jalon	Livrables
Trimestre 1	Premier connecteur et modèle canonique	Socle, connecteur 1, modèle versionné
Trimestre 2	Le cadre s'extrait	Connecteur 2, kit, simulateurs, jeu de tests type
Trimestre 3	La fabrique	Générateur, portes de certification, connecteurs 3 et 4, catalogue
Trimestre 4	L'échelle	Connecteurs 5 à 8, laboratoire d'interopérabilité, première promotion formée
Trimestre 5	Consolidation	Connecteurs 9 et 10, campagnes de performance, guides d'exploitation
Trimestre 6	Ouverture	Sandbox partenaires, certification de connecteurs tiers

16.4 Indicateurs de réussite

- **Durée de production du n-ième connecteur** — l'indicateur central. Elle doit décroître de façon visible entre le deuxième et le cinquième.
- **Part de code spécifique au rail** — elle doit passer sous 30 % et continuer à décroître.
- Nombre de connecteurs ayant franchi les six portes sans dérogation.
- Nombre de correctifs de sécurité appliqués une fois et propagés automatiquement aux dix connecteurs.
- Nombre de scénarios d'interopérabilité croisés fonctionnant de bout en bout.
- Nombre d'ingénieurs ayant produit un connecteur certifié, et délai moyen pour y parvenir.
- Temps de reconstruction complète du laboratoire depuis les dépôts — objectif : moins d'une journée.

16.5 Les décisions à documenter dès le départ

Sept décisions structurent le cadre pour des années. Chacune doit faire l'objet d'une fiche datée, argumentée et, lorsque c'est possible, mesurée. Les reprendre plus tard coûte bien davantage que de les poser correctement.

Tableau 43 — Les sept décisions fondatrices

Décision	Enjeu	Conséquence d'un mauvais choix
Périmètre du modèle canonique	Ce qu'il représente et ce qu'il exclut	Un modèle qui se dilue et redevient une collection de champs par rail
Politique de versionnage du kit	Compatibilité et rythme de montée de version	Des connecteurs figés sur d'anciennes versions, impossibles à corriger
Frontière connecteur / plateforme	Qui décide, qui traduit	Le couplage point à point se reconstitue par la porte de service
Politique de nouvelle tentative	Quand réessayer, quand s'abstenir	Doubles paiements ou paiements perdus
Durée de conservation de la piste d'audit	Preuve, coût de stockage, conformité	Investigation impossible, ou conservation excessive de données sensibles
Discipline des portes	Dérogation possible ou non	Une seule dérogation accordée ruine la valeur de la certification

Décision	Enjeu	Conséquence d'un mauvais choix
Choix du moteur de processus	Cycle de vie, licence, support	Une dépendance critique arrivée en fin de vie, comme l'illustre le chapitre 12

16.6 La doctrine

Doctrine du cadre

Le rail est une infrastructure.

Le connecteur traduit le rail.

La plateforme d'intégration orchestre les rails.

La couche de services transforme les rails en services.

Le cadre industrialise le savoir-faire nécessaire pour construire l'ensemble.

Annexes

Annexe A — Glossaire

Tableau 44 — Glossaire

Terme	Définition
Adaptateur de transport	Composant isolant le protocole de communication du reste du connecteur
Cadre commun	Ensemble des règles, du modèle et du code partagés par tous les connecteurs
Compensation	Annulation des effets d'une étape déjà exécutée, par une opération inverse
Connecteur de rail	Brique d'intégration traduisant un rail vers le modèle canonique
Corrélation	Capacité à relier une réponse tardive à la demande qui l'a provoquée
Disjoncteur	Mécanisme coupant les appels vers une contrepartie défaillante
Idempotence	Propriété d'une opération produisant le même résultat qu'elle soit exécutée une ou plusieurs fois
Interrogation	Demande à un rail de l'état réel d'une transaction
ISO 8583	Norme de messages pour les transactions initiées par carte
ISO 20022	Norme universelle de messagerie financière
Latence ajoutée	Temps consommé par le connecteur, hors attente de la contrepartie
Modèle canonique	Représentation unique d'un paiement, indépendante de tout rail
Piste d'audit	Archivage signé des messages échangés avec une contrepartie
Porte de certification	Ensemble de critères qu'un connecteur doit satisfaire pour progresser
Rail de paiement	Système externe par lequel un paiement est exécuté
Repli	Rail de secours activé sur échec technique du rail principal
Simulateur de rail	Contrepartie paramétrable reproduisant le comportement d'un rail
Unité mineure	Plus petite subdivision d'une devise ; en franc CFA, le franc lui-même

Annexe B — Check-list de construction d'un connecteur

Tableau 45 — Check-list de construction

#	Contrôle	Fait
1	Fiche de découverte du rail complète et validée par un architecte	☐
2	Simulateur écrit avant le connecteur, dix scénarios implémentés	☐
3	Projet généré depuis le kit, aucune modification du squelette	☐
4	Tables de correspondance versionnées, champs non mappés documentés	☐
5	Chemin retour testé au même niveau que le chemin aller	☐
6	Aucune décision métier dans le mapper	☐
7	Aucune logique de routage dans le connecteur	☐

#	Contrôle	Fait
8	Sécurité configurée depuis la bibliothèque, aucune cryptographie réécrite	☐
9	Rotation de certificat testée sans interruption de service	☐
10	Référence réseau réservée avant émission	☐
11	Méthode d'interrogation implémentée et testée	☐
12	Rejeu du même message : un seul effet vérifié	☐
13	Montants en unités mineures entières, aucun flottant	☐
14	Codes du rail conservés sans réinterprétation	☐
15	Attributs d'observabilité normalisés émis	☐
16	Aucune donnée sensible dans les journaux ni les traces	☐
17	Jeu de tests comptant plus de cas négatifs que de cas nominaux	☐
18	Les six portes de certification franchies sans dérogation	☐
19	Paquet complet : image, chart, schéma, simulateur, tests, guides	☐
20	Un tiers a déployé et dépanné le connecteur sans son auteur	☐

Annexe D — Fiche de découverte type d'un rail

Cette fiche est le premier livrable de toute construction de connecteur. Elle est remplie avant la génération du projet et revue par un architecte. Une zone laissée vide est une zone de risque.

Tableau 46 — Fiche de découverte type

Section	Question	Réponse attendue
Identité	Nom du rail, opérateur, périmètre géographique	Texte libre
Identité	Type de rail	Instantané, différé, cartes, monnaie électronique, masse, transfrontalier
Protocole	Transport utilisé	REST, SOAP, TCP, file de messages, événements, fichier
Protocole	Sens des échanges	Sortant seul, entrant seul, bidirectionnel
Protocole	Mode de réponse	Synchrone, asynchrone par rappel, asynchrone par interrogation
Format	Norme ou format des messages	ISO 20022, ISO 8583, JSON, XML, délimité, propriétaire
Format	Versions de messages supportées	Liste précise, avec dates de bascule
Format	Schémas disponibles	Fournis, à reconstituer, absents
Sécurité	Authentification exigée	Mutuelle, jeton, clé, signature, combinaison
Sécurité	Signature des messages	Aucune, détachée, enveloppée — algorithme
Sécurité	Chiffrement de charge utile	Requis ou non
Sécurité	Gestion des certificats	Autorité, durée, procédure de rotation

Section	Question	Réponse attendue
Cycle	Étapes du cycle transactionnel	Initiation, autorisation, compensation, règlement, confirmation
Cycle	Opérations d'annulation disponibles	Annulation, remboursement, rappel — et leurs limites
Cycle	Irrévocabilité	À partir de quelle étape le paiement devient définitif
Cycle	Mécanisme d'interrogation	Existe-t-il ? Par quelle opération ?
Erreurs	Liste des codes d'erreur	Référentiel complet, avec classification
Erreurs	Comportement en cas de rejeu	Le rail détecte-t-il un doublon ? Sur quelle clé ?
Erreurs	Comportement en cas d'expiration	La transaction peut-elle aboutir après expiration côté appelant ?
Contraintes	Débit maximal admis	Transactions par seconde, par minute, par jour
Contraintes	Latence annoncée et observée	Valeurs de référence
Contraintes	Disponibilité et fenêtres de maintenance	Horaires, heures limites
Contraintes	Plafonds	Par opération, par jour, par participant
Contraintes	Devises supportées	Liste, avec précision décimale
Règlement	Modèle de règlement	Instantané, différé net, différé brut, par lot
Règlement	Fichier ou relevé de règlement	Existe-t-il ? Format, fréquence, horaire
Exploitation	Environnement de test disponible	Oui, non, conditions d'accès
Exploitation	Procédure d'homologation	Étapes, durée, interlocuteurs
Exploitation	Contacts et procédure d'escalade	Noms, horaires, délais d'engagement

Les trois questions dont l'absence de réponse bloque tout

Parmi les trente questions ci-dessus, trois sont éliminatoires : le rail détecte-t-il un doublon et sur quelle clé ; existe-t-il un mécanisme d'interrogation ; une transaction peut-elle aboutir après expiration côté appelant. Sans ces trois réponses, il est impossible d'écrire un connecteur sûr — et tenter de le faire quand même revient à décider que les doublons seront découverts en production.

Annexe C — Références

- Documentation officielle des briques candidates : Apache Camel, Apache Kafka, ActiveMQ Artemis, RabbitMQ, KrakenD, Gravitee, Apache APISIX, Kong, Keycloak, Open Policy Agent, Temporal, OpenTelemetry, Prometheus, OpenSearch, Grafana, Loki, OpenBao, Docker, Kubernetes, Helm, Argo CD.
- Spécifications : ISO 8583, ISO 20022, OAuth 2.0, OpenID Connect, OpenAPI, AsyncAPI, W3C Trace Context.
- Annonces de cycle de vie des éditeurs — à consulter systématiquement avant toute qualification, l'exemple du moteur de processus cité au chapitre 12 montrant qu'une brique populaire peut cesser d'être maintenue.
- Textes réglementaires applicables dans la zone, à obtenir dans leur version en vigueur auprès de la direction de la conformité.
- Licences citées : Apache 2.0, MIT, Mozilla Public License 2.0, AGPLv3, et licences commerciales des éditeurs concernés.

Avertissement de fraîcheur

Les technologies citées doivent être réévaluées au moment de leur qualification. Version, licence, maintenance, vulnérabilités, documentation, conditions de support et cycle de vie évoluent — parfois brusquement, comme l'a montré l'arrêt du support communautaire d'un moteur de processus largement adopté.

Signature

Open Payment Platform Lab — Rail Connector Framework.

Build. Connect. Interoperate.

Document assisté par IA, gouverné par ISO/IEC 42001. Relecture humaine et validation technique requises avant diffusion.