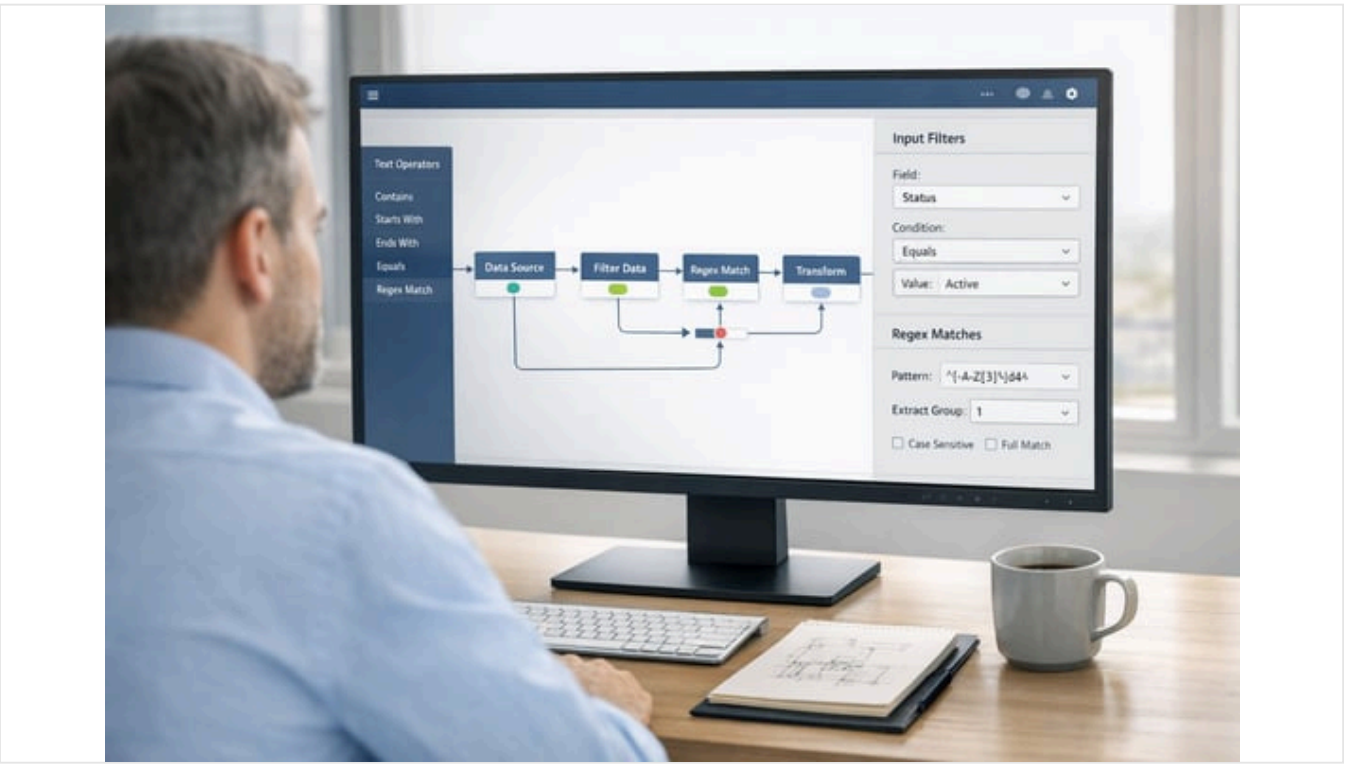
# Guide des filtres d'entrée Celigo : Regex et opérateur Matches

By houseblend.io Publié le 11 avril 2026 28 min de lecture




## Résumé analytique

Ce rapport fournit un examen exhaustif de la **plateforme d'intégration de données de Celigo**, en se concentrant sur les *filtres d'entrée*, l'opérateur « *matches* » (correspond à) et les capacités de *transformation de données* au sein de la plateforme **Integrator.io** de Celigo. Celigo est une plateforme d'intégration en tant que service (iPaaS) de premier plan qui facilite une connectivité transparente entre les systèmes SaaS et sur site (Source: docs.celigo.com) (Source: www.celigo.com). Alors que les organisations dépendent de plus en plus de systèmes interconnectés, un filtrage et une transformation efficaces des données deviennent essentiels pour garantir des intégrations précises et performantes (Source: docs.celigo.com) (Source: www.integrate.io). Ce guide explique comment les **filtres d'entrée** de Celigo autorisent sélectivement le passage des enregistrements, comment l'opérateur *matches* et les **expressions régulières (regex)** sont utilisés dans les flux Celigo, et comment la **transformation de données** de Celigo (spécifiquement Transformation 2.0) peut remodeler les données. Nous comparons la logique de filtrage et les techniques d'expressions régulières, détaillons les modes et types de champs de Transformation 2.0, et présentons des exemples et des études de cas illustrant une utilisation réelle. L'analyse s'appuie sur la documentation officielle de Celigo, des statistiques sectorielles et des perspectives d'experts pour offrir une vue complète de ces capacités, de leurs implications pour la pratique de l'intégration et des tendances futures de l'iPaaS et de l'automatisation pilotée par l'IA (Source: docs.celigo.com) (Source: www.celigo.com).

## Introduction et contexte

### Le rôle de l'iPaaS dans l'intégration moderne

Les entreprises modernes exploitent des dizaines, voire des centaines de systèmes et d'applications disparates. Une **plateforme d'intégration** (iPaaS) sert de middleware qui connecte ces systèmes, automatisant les flux de données et les processus métier à travers l'organisation (Source: docs.celigo.com) (Source: www.celigo.com). Celigo est une iPaaS basée sur le cloud conçue pour simplifier les intégrations (par exemple, **order-to-cash**, **quote-to-cash**, **hire-to-retire**) grâce à des centaines de connecteurs préconstruits et de flux personnalisables (Source: www.celigo.com). En suivant les meilleures pratiques d'intégration – telles que la définition d'objectifs clairs, le choix de modèles d'intégration reconnus et le maintien de

flux de travail « sans état » (stateless) – les organisations peuvent tirer parti de plateformes comme Celigo pour réaliser des intégrations fiables et évolutives (Source: docs.celigo.com) (Source: docs.celigo.com). Par exemple, les données provenant d'un système e-commerce peuvent être filtrées et transformées avant d'être synchronisées vers un ERP, garantissant que seules les transactions pertinentes sont transmises dans le format correct.

## Aperçu de Celigo Integrator.io

**Integrator.io** de Celigo est un constructeur de flux visuel où les utilisateurs définissent des sources de données, appliquent des filtres et des transformations, et mappent les champs vers les destinations. Les flux Integrator.io se composent d'étapes d'exportation (source) et d'importation/recherche (destination), avec des étapes de **middleware** optionnelles. Les données transitent séquentiellement par ces étapes, et des **filtres** peuvent être appliqués pour contrôler quels enregistrements continuent. Les **filtres d'entrée** sont appliqués aux étapes d'importation/recherche (afin d'ignorer les enregistrements entrants indésirables), tandis que les **filtres de sortie** sont appliqués aux étapes d'exportation ou intermédiaires (afin de ne faire passer que les enregistrements sélectionnés) (Source: docs.celigo.com) (Source: docs.celigo.com). Les filtres utilisent la logique booléenne et des opérateurs de comparaison (égal, contient, etc.) pour inclure ou exclure des enregistrements (Source: docs.celigo.com) (Source: docs.celigo.com). Pour les scénarios nécessitant une logique plus complexe que de simples règles, Celigo prend également en charge l'écriture de filtres en JavaScript (Source: docs.celigo.com).

Celigo prend en charge la transformation des données via son moteur **Transformation 2.0**, qui peut modifier, combiner ou remodeler les données d'entrée avant qu'elles ne soient mappées vers la destination. Cela inclut l'aplatissement de JSON imbriqué, la combinaison de plusieurs enregistrements sources ou la conversion de types de données (Source: docs.celigo.com) (Source: docs.celigo.com). Les transformations peuvent utiliser JSONPath pour sélectionner des champs et des modèles Handlebars pour calculer des valeurs (Source: docs.celigo.com). Dans les flux complexes, les filtres et les transformations fonctionnent ensemble : les filtres éliminent les données non pertinentes tôt, et les transformations préparent ensuite les données restantes pour l'importation.

Compte tenu de l'importance d'un traitement correct des données, la plateforme Celigo offre plusieurs façons de faire correspondre et de transformer les données :

- **Opérateur Matches :** Dans les règles de filtrage, l'opérateur « matches » teste si un champ correspond exactement à une valeur spécifiée (Source: docs.celigo.com).
- **Expressions régulières :** Celigo prend en charge les regex via les fonctions d'assistance Handlebars (`regexMatch`, `regexReplace`, `regexSearch`) pour une correspondance de motifs sophistiquée et la manipulation de données textuelles (Source: docs.celigo.com) (Source: docs.celigo.com).
- **Transformation de données :** Le moteur Transformation 2.0 (Rules 2.0) offre des modes de mappage avancés et des types de champs (Standard, Lookup, expression Handlebars, etc.) pour restructurer les données (Source: docs.celigo.com) (Source: docs.celigo.com).

Ce rapport approfondira chacun de ces domaines. Nous expliquerons comment utiliser les filtres d'entrée (y compris l'opérateur matches), comment tirer parti des assistants regex pour le filtrage et l'analyse basés sur des motifs, et comment configurer les transformations de données. Nous soulignerons également des exemples pratiques et des études de cas, tels que l'utilisation d'un filtre d'entrée pour synchroniser uniquement les enregistrements « Adjustment » d'Amazon FBA (Source: docs.celigo.com), et comment les entreprises ont appliqué Celigo à des problèmes réels (Source: www.celigo.com). Enfin, nous discuterons des implications – telles que le rôle de l'IA et de l'iPaaS dans les futures intégrations – et conclurons avec les meilleures pratiques.

# Filtres d'entrée Celigo : Concepts et utilisation

Les **filtres d'entrée** de Celigo sont un mécanisme clé pour contrôler le flux de données. Un filtre d'entrée sur une étape d'importation (ou de recherche) amène cette étape à traiter *uniquement* les enregistrements qui répondent aux critères du filtre, tout en permettant aux autres données de contourner cette étape (bien que les étapes suivantes sans filtre les verront toujours) (Source: docs.celigo.com). En termes pratiques, les filtres d'entrée agissent comme des gardiens : ils vérifient chaque enregistrement entrant et soit l'autorisent dans l'étape, soit l'ignorent. Ceci est utile pour ignorer les données indésirables sans interrompre l'ensemble du flux.

Par exemple, dans un flux Amazon vers ERP, on pourrait ne vouloir synchroniser que les commandes avec le statut « Shipped » (expédié). En ajoutant un filtre d'entrée sur l'étape pertinente, seules les commandes contenant « Shipped » dans leur champ de statut seraient traitées, et toutes les autres seraient ignorées à cette étape (Source: docs.celigo.com). Les filtres d'entrée sont configurés à l'aide d'un éditeur de règles visuel dans le Flow Builder de Celigo. Le tableau des opérateurs de filtre et leurs descriptions est présenté ci-dessous.

| OPÉRATEUR | DESCRIPTION |
|---|---|
| equals | Le champ est égal à la valeur |
| not equals | Le champ n'est pas égal à la valeur |
| is greater-than | Le champ est supérieur à (>) la valeur |
| is greater-than or equals | Le champ est ≥ à la valeur |
| is less-than | Le champ est inférieur à (<) la valeur |
| is less-than or equals | Le champ est ≤ à la valeur |
| between | Le champ est entre deux valeurs (incluses) |
| **contains** | Le champ contient une sous-chaîne donnée (Source: docs.celigo.com) |
| **does not contain** | Le champ ne contient *pas* la sous-chaîne (Source: docs.celigo.com) |
| **ends with** | Le champ se termine par la sous-chaîne spécifiée (Source: docs.celigo.com) |
| **null** | La valeur du champ est vide (aucune valeur) (Source: docs.celigo.com) |
| **is not null** | Le champ a une valeur (non vide) (Source: docs.celigo.com) |
| **matches** | Le champ a exactement la valeur spécifiée (Source: docs.celigo.com) |

*Tableau 1 : Opérateurs de filtre Celigo (au 2025) et leurs significations (Source: docs.celigo.com) (Source: docs.celigo.com).*

Dans le tableau ci-dessus, notez que l'opérateur **matches** est effectivement un test d'égalité pour les champs de type chaîne. La documentation de Celigo définit *« matches »* simplement comme « a une valeur spécifique dans le champ » (Source: docs.celigo.com). En pratique, vous utiliseriez *matches* lorsque vous souhaitez exiger une correspondance exacte (par exemple, faire correspondre un statut exactement à « Active »), tandis que *contains* permet une correspondance partielle. (D'autres opérateurs comme *equals* sont généralement utilisés avec des champs numériques ou de date, mais pourraient également s'appliquer aux chaînes.)

L'éditeur de filtres de Celigo permet des ensembles de règles complexes. Vous pouvez regrouper des conditions avec une logique ET/OU et même nier des groupes en utilisant NOT (Source: docs.celigo.com) (Source: docs.celigo.com). Une fois les règles définies, elles déterminent quels enregistrements passent. Par exemple, la documentation de Celigo illustre l'ajout d'un filtre d'entrée pour ne synchroniser que les **InventoryAdjustments** FBA de type « Adjustments » (Source: docs.celigo.com) (Source: docs.celigo.com). Dans cet exemple, l'utilisateur sélectionne `record.[Event Type]` (champ de chaîne), l'opérateur *contains*, et la valeur « Adjustments ». Seuls les enregistrements contenant ce texte dans le type d'événement passeraient (Source: docs.celigo.com) (Source: docs.celigo.com).

En plus des filtres basés sur des règles, Celigo vous permet d'écrire des filtres en JavaScript. En basculant l'éditeur de filtre de *Rules* (Règles) à *JavaScript*, toute logique peut être scriptée (Source: docs.celigo.com). Le basculement vers les règles est utile pour des scénarios tels que la concaténation de plusieurs champs ou l'exécution de tests regex au sein du filtre. La documentation note que les filtres JavaScript peuvent gérer des conditions complexes (par exemple, *« Concaténer plusieurs champs de chaîne, puis filtrer en fonction des résultats de la valeur concaténée »* ou *« Filtrer les enregistrements où un champ de date est plus ancien qu'un certain nombre de jours »* (Source: docs.celigo.com). Cette flexibilité signifie que si les opérateurs standard et les groupes booléens sont insuffisants, une fonction JS personnalisée peut implémenter une logique arbitraire.

Dans l'ensemble, les filtres d'entrée vous permettent d'**élaguer les données indésirables tôt dans le flux**, améliorant ainsi l'efficacité et la qualité des données. Par exemple, les meilleures pratiques de Celigo suggèrent de filtrer les enregistrements par critères pour réduire le traitement en aval (Source: docs.celigo.com) (Source: docs.celigo.com). Dans le cas d'une entreprise, l'ajout d'un filtre d'entrée pour limiter les enregistrements Amazon

FBA aux seuls types d'événements « Adjustments » a empêché l'envoi d'expéditions ou de transferts non pertinents vers NetSuite (Source: docs.celigo.com). Ce type de filtrage ciblé maintient les performances des flux et évite les erreurs de données ou la surcharge en aval.

# Regex Celigo et l'opérateur « Matches »

## L'opérateur « Matches » dans les filtres

Comme noté ci-dessus, l'opérateur de filtre *matches* de Celigo est utilisé pour une correspondance exacte. Contrairement à certaines plateformes qui vous permettent d'entrer des motifs regex directement dans les règles de filtrage, les règles de filtrage intégrées de Celigo utilisent *matches* comme une vérification stricte de la valeur (Source: docs.celigo.com). En d'autres termes, *« matches »* dans le contexte du filtre signifie *equals* (égal) pour une chaîne. Par exemple, si vous définissez une règle `field matches "ABC123"`, elle ne fera passer que les enregistrements dont le **champ** est exactement égal à « ABC123 ». Cela s'apparente à l'opérateur « equals », mais spécifiquement pour les champs de type chaîne (l'opérateur « equals » dans l'interface utilisateur est souvent formulé pour les champs numériques ou de date).

Étant donné que *matches* se comporte comme un test d'égalité, on peut se demander comment effectuer une correspondance de motifs à l'aide d'expressions régulières. L'interface utilisateur de filtre de Celigo n'expose pas directement la correspondance regex. Au lieu de cela, Celigo s'appuie sur ses **assistants Handlebars** pour les opérations intensives en regex (qui sont utilisées dans le mappage/les expressions, et non dans l'interface utilisateur de filtre simple). Pour effectuer un filtrage regex, vous avez quelques options :

- **Filtre JavaScript :** Basculez le filtre en mode JavaScript et utilisez `RegExp` ou `.match()` de JavaScript dans le code du filtre pour tester les motifs.
- **Prétraitement avec Transformation :** Utilisez une étape de Transformation (qui prend en charge Handlebars) pour ajouter des indicateurs ou des champs, puis filtrez en fonction de cela.
- **Handlebars dans les mappages :** Parfois, on peut effectuer une logique de filtrage indirectement en définissant des conditions dans la transformation ou le mappage.

Pour les cas simples (correspondance exacte plutôt que motif), *matches* est approprié. Par exemple, pour filtrer les enregistrements où `record.Status` est exactement égal à « Shipped », on utiliserait :

```
Champ : record.Status
Opérateur : matches
Valeur : Shipped
```

Aucune correspondance partielle ou caractère générique n'est implicite. D'un autre côté, `contains` correspondrait également à « Shipped » si le champ de statut le contient n'importe où, et `does not contain` l'exclurait. En résumé, l'opérateur *matches* impose une équivalence exacte (Source: docs.celigo.com).

## Assistants d'expressions régulières dans les flux Celigo

Bien que l'interface de filtre principale n'utilise pas de regex, Celigo fournit de puissants *assistants regex* pour une utilisation dans les transformations et les mappages de données. Ces assistants fonctionnent via des expressions Handlebars (doubles accolades) et vous permettent de rechercher, d'extraire ou de remplacer du texte en fonction de motifs. Les principaux assistants sont :

- **{{regexMatch field regex ...}}** – Correspond et renvoie le texte de la correspondance regex.
- **{{regexReplace field replacement regex}}** – Remplace le texte correspondant à une regex par un remplacement donné.
- **{{regexSearch field regex}}** – Recherche une correspondance regex et renvoie l'index de la correspondance.

Ces assistants sont documentés dans le centre d'aide de Celigo (Source: docs.celigo.com) (Source: docs.celigo.com) (Source: docs.celigo.com) et reposent sur le moteur d'expressions régulières de Node.js/JavaScript (Source: docs.celigo.com). En pratique, ils permettent une logique de filtrage beaucoup plus sophistiquée que les opérateurs de filtre de base. Examinons chacun d'eux :

- **regexMatch :** Cet assistant prend un champ texte et un modèle d'expression régulière (regex), et renvoie la ou les parties correspondantes du texte. Par défaut, il renvoie la première correspondance. Par exemple, si `record.comment = "Order ID: 12345 delivered on 2025-04-20"`, l'utilisation de

  ```
  {{regexMatch record.comment "[0-9]{5}"}}
  ```

  renverra `"12345"`, car il trouve la première séquence de 5 chiffres (Source: docs.celigo.com). Vous pouvez également spécifier un index (commençant à 0) si le champ contient plusieurs correspondances. Par exemple, si le commentaire contient « IDs: 12345 and 67890 », alors `{{regexMatch record.comment "[0-9]{5}" 1}}` renverra `"67890"` (Source: docs.celigo.com). Les drapeaux regex (comme `"i"` pour ignorer la casse) peuvent être transmis en tant que quatrième argument optionnel.

- **regexSearch :** Cet assistant recherche un modèle regex et renvoie la position (index) de la première correspondance dans la chaîne (où 0 = premier caractère). Il ne renvoie pas le texte, seulement l'index numérique. Par exemple, si `record.total = "$1499.95"`, alors

  ```
  {{regexSearch record.total "\."}}
  ```

  renverra `5`, car le premier « . » (point décimal) est le 6ème caractère (index 5) (Source: docs.celigo.com). Une option insensible à la casse peut être ajoutée (drapeau `"i"`). Si aucune correspondance n'est trouvée, `regexSearch` renvoie `-1` (Source: docs.celigo.com), ce qui peut être utilisé dans une logique conditionnelle (par exemple, pour filtrer les enregistrements où `-1` indique l'absence d'un modèle).

- **regexReplace :** Cet assistant effectue une recherche et un remplacement à l'aide d'une regex. (Bien que non explicitement documenté sur une page séparée, la documentation de Celigo le mentionne et l'assistant de remplacement y fait référence (Source: docs.celigo.com).) Il prend un champ, une chaîne de remplacement et un modèle regex (avec des drapeaux optionnels), et renvoie le texte avec toutes les correspondances remplacées. Par exemple, pour supprimer les caractères non numériques d'un numéro de téléphone :

  ```
  {{regexMatch (regexReplace phone "" "[^\\d]" "g") "\\d{10}$"}}
  ```

  (Cet exemple, tiré de la documentation de Celigo, utilise d'abord `regexReplace` pour supprimer tout ce qui n'est pas un chiffre, puis `regexMatch` pour capturer 10 chiffres (Source: docs.celigo.com).) Si votre besoin est plus simple (remplacement de sous-chaîne fixe), Celigo propose également un assistant `replace` sans regex (Source: docs.celigo.com).

Ces assistants regex s'appuient sur le comportement `RegExp` de JavaScript (Source: docs.celigo.com) (Source: docs.celigo.com), ce qui signifie qu'ils prennent en charge des modèles comme `[A-Z]`, les lookahead/lookbehind, les drapeaux global (`g`), multiligne (`m`), etc. Pour la création complète de modèles, Celigo suggère d'utiliser d'abord un testeur regex externe comme regex101 (Source: docs.celigo.com), puis d'intégrer le modèle validé dans les Handlebars.

## Exemples de cas d'utilisation des assistants Regex

- **Extraction de sous-chaînes :** Si une intégration nécessite l'extraction de données structurées à partir de texte libre, `regexMatch` est idéal. *Exemple :* Étant donné un champ `record.comment = "Order #ID12345 shipped on 2024-08-15"`, on pourrait extraire l'ID numérique via

  ```
  {{{regexMatch record.comment "[0-9]+$"}}}
  ```

  ce qui pourrait donner `"12345"` (la notation triple accolades `{{{ }}}` peut être utilisée pour éviter les guillemets de Celigo).

- **Filtrage par modèle :** Bien que les filtres n'utilisent pas directement les regex, vous pouvez utiliser `regexSearch` au sein d'une transformation pour marquer les enregistrements, puis filtrer. Par exemple, pour filtrer les noms commençant par « ENG », on pourrait créer un champ dans une Transformation avec `regexSearch(record.name, "^ENG")` puis définir le filtre d'entrée sur `newField > -1`.

- **Remplacements complexes :** Supposons que les codes SKU dans une source aient des délimiteurs variables et nécessitent une normalisation. On pourrait utiliser `regexReplace record.SKU "_" "\\W+"`, qui remplace les caractères non alphanumériques par des traits de soulignement. Combiné avec des filtres, on pourrait ensuite filtrer les SKU qui correspondent désormais à un modèle.

Chaque utilisation d'un assistant regex doit être citée dans la documentation et soigneusement testée dans son contexte. Les exemples de Celigo (Source: docs.celigo.com) (Source: docs.celigo.com) servent de bons points de départ pour les modèles courants.

# Transformation de données dans les intégrations Celigo

## Quand et pourquoi transformer les données

Dans tout scénario d'intégration, la **transformation de données** est souvent nécessaire pour réconcilier les différences de structure, de format ou de sémantique entre les systèmes. La Transformation 2.0 (Règles 2.0) de Celigo fournit un moteur puissant à cet effet (Source: docs.celigo.com). Les transformations sont appliquées aux *données d'entrée* avant le mappage vers la sortie. Cela est particulièrement utile dans deux cas :

1. **Une source, plusieurs destinations :** Lorsque vous exportez depuis une source (par exemple, un JSON complexe provenant d'une base de données) vers *plusieurs* étapes d'importation ultérieures, vous pouvez d'abord simplifier les données. Par exemple, si l'exportation inclut de nombreux champs imbriqués et tableaux, une transformation peut les aplatir ou les filtrer afin que chaque importation ne voie que le sous-ensemble pertinent (Source: docs.celigo.com). La documentation de Celigo note que les transformations sont idéales pour aplatir et simplifier les enregistrements sources destinés à plusieurs récepteurs.

2. **Plusieurs sources, une destination :** Inversement, si vous avez plusieurs sources de données alimentant une seule cible, vous devrez peut-être les consolider. Celigo permet de transformer les enregistrements de chaque source en un format commun et canonique avant de les envoyer vers la destination (Source: docs.celigo.com). Cela permet de fusionner ou de normaliser différentes entrées afin que la logique de mappage de destination soit plus simple.

En pratique, on peut utiliser la Transformation 2.0 pour effectuer des tâches telles que : aplatir des structures JSON imbriquées, convertir des tableaux en format CSV basé sur des lignes, renommer des champs, combiner des enregistrements sources ou diviser un seul enregistrement en plusieurs lignes de sortie (par exemple, créer des lignes de commande). Le moment des transformations se situe généralement juste après l'exportation et avant toute importation. Cela vous permet de « façonner » les données afin que vos mappages finaux soient simples et robustes.

## Transformation 2.0 : Modes

Le transformateur de Celigo propose trois **modes** de fonctionnement (Source: docs.celigo.com), sélectionnables via un menu déroulant :

- **Modifier l'entrée (Modify the Input) :** Ce mode apporte des modifications ciblées *au sein* de l'enregistrement original tout en conservant tous les autres champs intacts. Utilisez-le pour de petits ajustements ; par exemple, corriger le format d'un numéro de téléphone, supprimer les espaces inutiles ou ajouter un indicateur calculé, tout en laissant la structure globale de l'enregistrement inchangée (Source: docs.celigo.com).

- **Créer un enregistrement de sortie à partir de l'entrée (Create Output Record from Input) :** Ce mode permet de construire un tout nouvel enregistrement de sortie à partir de zéro en utilisant des champs (ou des calculs basés sur ceux-ci) provenant de l'entrée. Il est utile lorsque vous ne souhaitez transmettre qu'un sous-ensemble de champs ou lors de la fusion de champs dans de nouveaux champs. Par exemple, lors de la consolidation de plusieurs types de sources, vous définissez exactement comment chaque partie de la sortie doit dériver de l'entrée (Source: docs.celigo.com).

- **Créer des lignes de sortie à partir de l'entrée (Create Output Rows from Input) :** Ce mode sert à convertir un enregistrement d'entrée en *plusieurs* lignes de sortie (aplatir les éléments d'un tableau en lignes distinctes). Un exemple d'utilisation serait la transformation d'une commande unique avec une liste d'articles en plusieurs lignes d'articles pour une exportation au format CSV (Source: docs.celigo.com). Chaque élément du tableau peut devenir un nouvel enregistrement de sortie, permettant des flux de travail nécessitant des sorties basées sur des lignes.

Choisir le bon mode est essentiel. En général, « Modifier l'entrée » est rapide et simple, « Enregistrement de sortie » offre un contrôle total sur la structure finale, et « Lignes de sortie » est spécialisé pour la création d'enregistrements un-à-plusieurs.

## Types de champs de transformation

Au sein d'un mappage de transformation, chaque champ de sortie possède un **type de champ** qui détermine comment sa valeur est calculée (Source: docs.celigo.com) (Source: docs.celigo.com). Les quatre types sont :

| TYPE DE CHAMP | DESCRIPTION | EXEMPLE |
|---|---|---|
| **Standard** | Mappe directement un champ de la source vers la sortie (par défaut) (Source: docs.celigo.com). | Mapper `record.firstName` → `FirstName`. |
| **Codé en dur** | Fournit une valeur statique fixe pour le champ de sortie, indépendamment de l'entrée. | Définir `Country` = `"USA"` dans tous les enregistrements de sortie. |
| **Recherche (Lookup)** | Utilise une table de correspondance pour mapper une valeur d'entrée vers une autre valeur de sortie (Source: docs.celigo.com). | Mapper `"apple"` → `"banana"` pour toute occurrence de `"apple"`. |
| **Expression Handlebars** | Utilise un modèle ou une formule Handlebars personnalisé pour la valeur de sortie (Source: docs.celigo.com). | Concaténer `{{record.firstName}}` `{{record.lastName}}`. |

*Tableau 2 : Types de champs de transformation Celigo et utilisation (Source: docs.celigo.com) (Source: docs.celigo.com).*

Les types **Standard** et **Codé en dur** sont simples : soit on transmet la valeur, soit on définit une constante. **Recherche** est utile pour des conversions de valeurs simples (par exemple, recherche de codes de devise ou mappage de codes de statut). **Expression Handlebars** offre la plus grande flexibilité, permettant une logique complexe (concaténation, arithmétique, expressions conditionnelles, encodage, etc.) via le templating (Source: docs.celigo.com) (Source: docs.celigo.com). Par exemple, vous pourriez multiplier deux champs ( `{{ =}}` ) ou appliquer des fonctions `upper` / `lower` avec Handlebars.

Le moteur de transformation de Celigo prend en charge des types de données tels que chaîne, nombre, booléen, objet, tableaux, etc., et peut automatiquement effectuer des conversions entre les types lorsque cela est possible (Source: docs.celigo.com). Si la conversion est impossible, le mappage échouera (par exemple, mapper une chaîne vers un nombre sans analyse). Il existe également des paramètres pour gérer les valeurs nulles et documenter la logique des champs via des commentaires (Source: docs.celigo.com).

### Utilisation de JSONPath et Handlebars dans les transformations

Celigo exploite **JSONPath** pour sélectionner des champs à partir d'enregistrements d'entrée complexes (Source: docs.celigo.com). JSONPath est un langage de requête pour JSON (similaire à XPath pour XML) qui vous permet de spécifier des champs même s'ils sont imbriqués ou dans des tableaux. Le passage à JSONPath signifie que vous pouvez écrire des expressions comme `order.items[*].sku` ou `$.customer.address.city` pour choisir des valeurs. La documentation de Celigo note que les expressions JSONPath peuvent inclure des tranches de tableau et une descente récursive (syntaxe « * ») (Source: docs.celigo.com). C'est crucial lorsque les données ont des structures imbriquées ; vous pouvez naviguer précisément vers les informations nécessaires pour la transformation.

Les **expressions Handlebars** peuvent ensuite calculer ou formater ces valeurs. Par exemple, pour combiner le prénom et le nom : `{{record.customer.firstName}} {{record.customer.lastName}}`, ou pour formater des dates. Handlebars prend également en charge le branchement (via `{{#if}}`) et de nombreux assistants intégrés. Celigo fournit des outils et extensions personnalisés (par exemple, opérations mathématiques, fonctions de chaîne) à l'intérieur de Handlebars pour renforcer les transformations (Source: docs.celigo.com). Par exemple, si vous aviez besoin d'encoder un champ pour une URL, vous pourriez utiliser un assistant Handlebars comme `{{encodeField record.text}}` (hypothétique).

## Exemple de transformation

Supposons qu'un flux reçoive un JSON de commande e-commerce comme :

```
{
  "order": {
    "id": 123,
    "customer": {"firstName": "Alice", "lastName": "Smith"},
    "items": [ {"sku": "A1", "qty": 2}, {"sku": "B2", "qty": 1} ],
    "date": "2025-07-01T12:34:56Z"
  }
}
```

Nous voulons envoyer vers une destination qui attend des enregistrements avec les champs : `OrderID`, `CustomerName`, `Sku`, `Quantity`, `OrderDate`. En utilisant le mode **Créer des lignes de sortie à partir de l'entrée**, nous pourrions définir :

- `OrderID (Standard) = record.order.id`
- `CustomerName (Handlebars) = {{record.order.customer.firstName}} {{record.order.customer.lastName}}`
- Puis un mappage *Lignes de sortie* : Placez `record.order.items[*]` comme tableau d'entrée. Pour chaque article, nous mappons :
  - `Sku (Standard) = record.sku`
  - `Quantity (Standard) = record.qty`
  - `OrderDate (Standard) = record.order.date`

Celigo produirait 2 lignes de sortie : une pour SKU=A1, qty=2, une pour SKU=B2, qty=1, chacune incluant l'OrderID et le CustomerName répétés. Dans cet exemple, nous utiliserions JSONPath comme `order.items[*]` (Source: docs.celigo.com) pour les lignes, et Handlebars pour construire `CustomerName`. Le résultat est une donnée basée sur des lignes adaptée à l'exportation CSV, aplatie à partir du JSON imbriqué.

## Analyse des données et considérations de performance

Les fonctionnalités de filtrage et de transformation de Celigo sont conçues pour gérer des volumes d'entreprise. Par exemple, Celigo a rapporté le traitement de **36,5 milliards d'enregistrements** pendant le Black Friday/Cyber Monday 2025, sans ralentissements (Source: www.celigo.com). Une telle échelle souligne pourquoi un filtrage robuste est essentiel : élaguer les enregistrements non pertinents tôt réduit considérablement la charge de traitement sur les systèmes en aval. Selon Celigo, ils ont maintenu 100 % de disponibilité pendant cette période de pointe (Source: www.celigo.com). Une conception de filtre supérieure contribue à la fiabilité en garantissant que les flux ne traitent que des données valides.

Du point de vue de l'industrie, le besoin d'une intégration forte est aigu. Une enquête MuleSoft a révélé que les organisations ont en moyenne ~897 applications mais seulement ~29 % d'intégration entre elles (Source: www.integrate.io). Cette déconnexion coûte aux entreprises un retour sur investissement important – les entreprises bien intégrées voient un ROI *10,3*× supérieur de leurs initiatives IA contre *3,7*× pour les entreprises mal intégrées (Source: www.integrate.io). Bien que non spécifique à Celigo, cela souligne que des flux de données propres et bien filtrés (comme ceux que permet Celigo) sont essentiels pour débloquer la valeur commerciale. Inversement, les échecs d'intégration sont courants ; un rapport note que *84 % des projets d'intégration système échouent ou échouent partiellement* en raison de problèmes tels que des exigences médiocres ou des problèmes de données (Source: www.integrate.io). L'utilisation appropriée des filtres et des transformations dans Celigo peut aider à éviter de tels pièges en imposant la qualité des données et les attentes en matière de schéma avant d'intégrer les systèmes.

La transformation des données est tout aussi critique : 64 % des organisations citent la qualité des données comme leur principal défi en matière d'intégration (Source: www.integrate.io). En utilisant les transformations de Celigo, les intégrateurs peuvent imposer des formats de données (par exemple, normaliser les formats de date, supprimer les espaces inutiles) et détecter les anomalies. Par exemple, un flux Celigo peut utiliser un opérateur de règle **« null »** ou **« is not null »** pour filtrer les enregistrements dont les identifiants clés sont manquants (Source: docs.celigo.com). Cela évite d'envoyer des données incomplètes qui provoqueraient des erreurs dans l'ERP. Au-delà des filtres, la transformation 2.0 permet de valider ou d'enrichir les données (par exemple, via des tables de correspondance pour remplir les valeurs manquantes). En somme, un filtrage rigoureux et une mise en forme des données garantissent que seuls les enregistrements conformes et de haute qualité sont traités, répondant ainsi aux préoccupations mêmes (qualité des données, silos) qui entravent le succès de la transformation numérique (Source: www.integrate.io) (Source: www.integrate.io).

## Études de cas et exemples concrets

## Étude de cas : Filtrage des ajustements Amazon FBA

La documentation de Celigo fournit un exemple concret d'utilisation des filtres d'entrée dans un scénario de vente au détail (Source: docs.celigo.com) (Source: docs.celigo.com). Dans le flux *Amazon (FBA) Inventory Adjustments* vers NetSuite, différents types d'événements (expéditions, transferts, réceptions, etc.) sont présents, mais l'entreprise souhaitait uniquement synchroniser les entrées **« Adjustments »**. En ajoutant un filtre d'entrée sur l'étape de la source d'exportation, l'intégrateur a spécifié :

```
Champ : record.[Event Type]
Opérateur : contains
Valeur : Adjustments
```

Seuls les enregistrements où le `Event Type` contenait « Adjustments » étaient transmis. Le guide Celigo détaille cette configuration : sélection du champ, définition du type d'opérande sur Field/String, utilisation de l'opérateur AND, choix de *contains* et saisie du texte « Adjustments » (Source: docs.celigo.com) (Source: docs.celigo.com). Après avoir enregistré et exécuté le flux, le client a constaté que seuls les enregistrements « Adjustments » étaient synchronisés dans NetSuite ; les autres types d'événements étaient ignorés.

Cela illustre deux points : (1) Les filtres d'entrée peuvent être ajoutés *à n'importe quelle étape* d'un flux (y compris sur la tuile source via la boîte de dialogue « Define input filter ») (Source: docs.celigo.com). (2) Les filtres avec des opérateurs de base (contains, equals, etc.) sont souvent suffisants pour les besoins réels. Dans ce cas, les expressions régulières (regex) n'étaient pas nécessaires — une simple règle *contains* a résolu le problème. Si, toutefois, l'exigence avait été de faire correspondre un modèle (par exemple, des codes de type d'événement commençant tous par « ADJ- »), un filtre JavaScript ou une aide regex aurait pu être utilisé.

## Étude de cas : Intégration de la logistique d'entreprise

Dans un déploiement à grande échelle, **Therabody** (une entreprise d'électronique grand public) a utilisé Celigo pour intégrer les commandes, les stocks et la logistique à travers plusieurs systèmes (Source: www.celigo.com) (Source: www.celigo.com). Ils ont remplacé un iPaaS hérité et créé des flux pour la gestion des commandes et l'intégration 3PL. Grâce à Celigo, Therabody a connecté l'ERP NetSuite à plusieurs prestataires logistiques tiers (nationaux et internationaux) en temps réel. Surtout, ils ont pu synchroniser les niveaux de stock et automatiser la logistique sans solutions de contournement manuelles (Source: www.celigo.com).

Bien que l'étude de cas se concentre sur les résultats globaux (par exemple, réduction de la complexité, processus de commande efficaces), elle implique une utilisation intensive des fonctionnalités de Celigo. Par exemple, la connexion avec des 3PL implique souvent de filtrer le flux par entrepôt ou SKU, et de transformer les données dans les formats attendus par chaque système. Le succès à grande échelle (gestion des opérations en période de pointe avec des coûts de main-d'œuvre réduits) indique que les filtres et transformations de Celigo traitaient de manière fiable de gros volumes de transactions. L'histoire de Therabody souligne comment une entreprise peut s'appuyer sur Celigo pour des intégrations critiques impliquant plusieurs sources et cibles de données (Source: www.celigo.com) (Source: www.celigo.com).

## Exemple de flux de travail : Regex en action

Pour illustrer l'**utilisation des regex** dans un flux pratique, imaginez une intégration e-commerce où les SKU de produits provenant d'une place de marché incluent des suffixes que nous devons supprimer. Par exemple, les SKU entrants pourraient ressembler à `"ABC123-XL"` ou `"XYZ999-KIDS"`, et le système cible attend simplement `"ABC123"` ou `"XYZ999"`. Dans Celigo, on pourrait ajouter un champ de transformation utilisant `regexReplace` pour supprimer le suffixe après le tiret :

```
{{regexReplace record.SKU "" "-.*$"}}
```

Cette expression Handlebars utilise la regex `-.*$` pour trouver un tiret et tous les caractères suivants, en les remplaçant par une chaîne vide. Après cette transformation, le SKU est normalisé. Ensuite, un filtre pourrait utiliser des correspondances sur le SKU nettoyé, ou simplement le mapper vers le champ de destination. Cette approche démontre la combinaison du remplacement par regex dans une transformation, suivie d'un mappage standard — un flux de travail qui peut être nécessaire dans de nombreux scénarios B2C.

# Discussion : Implications et orientations futures

## Perspectives plus larges

D'un **point de vue technique**, la combinaison par Celigo d'interfaces « pointer-cliquer » et d'options avancées (filtres, JSONPath, Handlebars) atteint un équilibre idéal entre facilité d'utilisation et puissance (Source: docs.celigo.com) (Source: docs.celigo.com). Les utilisateurs moins techniques peuvent implémenter des règles avec des menus déroulants et des opérateurs logiques, tandis que les utilisateurs avancés peuvent utiliser JavaScript ou écrire des expressions Handlebars pour des cas complexes. Cela reflète une tendance générale dans les outils d'intégration : permettre à la fois le « low code » sans code et la personnalisation pilotée par le code (Source: www.celigo.com) (Source: www.celigo.com). La conséquence est que les organisations peuvent réduire le backlog des développeurs en permettant aux analystes de créer des filtres et des transformations simples, tout en laissant la place aux développeurs pour les cas particuliers.

D'un **point de vue commercial**, l'utilisation efficace des filtres d'entrée et des transformations améliore la gouvernance et la confiance dans les données. En filtrant le bruit (enregistrements de test, canaux de vente non pertinents, événements en double) et en remodelant les données dans des formats cohérents, les entreprises réduisent les erreurs de données et les réconciliations manuelles. La recherche industrielle indique que la **mauvaise qualité des données coûte des milliers de milliards** à l'échelle mondiale (Source: www.integrate.io) ; ainsi, tout ce qui minimise les données erronées (via des filtres) est précieux. La plateforme de Celigo sert donc non seulement de connecteur, mais aussi de point de contrôle pour les normes de qualité des données.

Il existe également une **perspective opérationnelle** : les flux Celigo avec filtres peuvent réduire la charge des systèmes et éviter les problèmes de limites de débit. Par exemple, filtrer les enregistrements indésirables réduit les appels API vers les systèmes en aval (coûts à l'usage). Les propres mesures de Celigo (milliards d'enregistrements traités, zéro temps d'arrêt) (Source: www.celigo.com) montrent l'importance de l'efficacité — même de petites inefficacités à grande échelle peuvent se traduire par de gros problèmes. Concevoir des filtres serrés et des transformations intelligentes est donc autant une question d'économie de coûts que de précision.

## Intégration avec l'IA et tendances futures

Pour l'avenir, Celigo se positionne comme un élément central de l'automatisation pilotée par l'IA dans les entreprises (Source: www.celigo.com) (Source: www.celigo.com). La logique des filtres et des transformations croise l'IA de deux manières :

1. **Mappage assisté par l'IA :** Celigo propose désormais des outils d'IA comme *AI Code Assistant* (qui peut générer du Handlebars ou du JavaScript à partir d'instructions en langage naturel) et *Knowledge Bot* pour obtenir des conseils sur la documentation (Source: www.celigo.com). À l'avenir, on pourrait demander au système de « filtrer les commandes où le type d'événement commence par 'ADJ' » et laisser Celigo générer automatiquement le filtre. De même, les fonctionnalités d'IA de Celigo pourraient suggérer des modèles regex ou des transformations basées sur des exemples de données. Cela peut aider à éliminer les incertitudes lors de l'écriture d'une logique de filtre ou de transformation complexe.

2. **Filtrage intelligent :** Le composant « créer ou expliquer les règles de filtrage à l'aide de l'IA de Celigo » suggère que Celigo explore l'utilisation de l'IA pour recommander des critères de filtrage (Source: docs.celigo.com). On pourrait imaginer une IA analysant les données historiques des flux et conseillant : « 90 % des événements de type 'Ship' ont également FieldX='Y', peut-être filtrer X>Y pour de meilleures performances ». Bien que spéculatif, cela indique une tendance où l'IA aide à la conception de l'intégration. L'approche de plateforme unifiée de Celigo vise à prendre en charge à la fois l'intégration et les flux de travail IA/ML sur le même canevas (Source: www.celigo.com) (Source: www.celigo.com).

Plus généralement, l'intégration d'entreprise évolue vers le **libre-service et l'orchestration**. La vision de Celigo (selon leur blog) est « une plateforme unifiée » pour les intégrations, les API et les processus d'IA événementiels ou agents (Source: www.celigo.com) (Source: www.celigo.com). Dans ce contexte, les filtres et les transformations restent des primitives fondamentales : contrôler les flux de données même dans les scénarios iPaaS avancés. Par exemple, des flux de travail extrêmement complexes (tels que des chatbots de support client pilotés par l'IA qui récupèrent des données de plusieurs systèmes) nécessitent toujours un filtrage (par exemple, ignorer les e-mails non clients) et une transformation de la sortie en un texte convivial. La cohérence de la logique de flux est ce qui rend les sorties de l'IA fiables.

## Limites et alternatives

Bien que l'approche de Celigo soit robuste, il y a quelques considérations :

- L'opérateur de filtre « matches » est limité aux correspondances exactes. Les organisations nécessitant une correspondance de modèle plus flexible *au sein même du filtre* peuvent trouver l'interface contraignante. La solution de contournement consiste à utiliser JavaScript ou un prétraitement, ce qui introduit de la complexité. Certains outils d'intégration concurrents autorisent nativement les regex dans l'interface utilisateur

de filtrage. Cependant, dans Celigo, cette lacune est atténuée par la couche de transformation.

- La dépendance à Handlebars pour une logique complexe signifie que les utilisateurs doivent apprendre sa syntaxe. Bien que la documentation et la communauté de Celigo soient utiles, il y a une courbe d'apprentissage pour maîtriser des expressions comme `{{regexMatch}}`, les boucles et les conditions.

- Les transformations de données peuvent introduire de la latence si elles ne sont pas soigneusement conçues. Bien que Celigo s'adapte automatiquement, des transformations imbriquées extrêmement volumineuses pourraient atteindre des limites de performance. Ainsi, la meilleure pratique consiste souvent à effectuer la préparation des données le plus près possible de la source (par exemple, filtrer les lignes inutiles avant de les extraire).

Les **approches alternatives** pourraient inclure l'exécution d'un filtrage en amont (dans la requête source) ou en aval (post-importation) au lieu de le faire dans Celigo. Par exemple, si la source est une base de données, l'ajout d'une clause `WHERE` pourrait réduire le besoin d'un filtre Celigo. Mais l'avantage de Celigo est l'abstraction de ces détails : l'interface utilisateur facilite les modifications sans avoir à redéployer les requêtes ou le code.

## Conclusion

La plateforme d'intégration de Celigo offre un mélange flexible de configuration visuelle et de logique avancée pour la gestion des flux de données. Les **filtres d'entrée** permettent aux utilisateurs de contrôler précisément quels enregistrements circulent à chaque étape, préservant ainsi les ressources et maintenant la pertinence des données. Les opérateurs de filtre (equals, contains, matches, etc.) couvrent la plupart des besoins courants (Source: docs.celigo.com) (Source: docs.celigo.com), et les conditions complexes peuvent être gérées par regroupement ou en passant à JavaScript pour une logique personnalisée (Source: docs.celigo.com).

Pour le traitement basé sur des modèles, Celigo fournit des **aides aux expressions régulières** (`regexMatch`, `regexSearch`, `regexReplace`) au sein de son cadre Handlebars/Transformation. Celles-ci sont à la hauteur de la puissance du moteur regex de JavaScript et permettent de découper, segmenter et nettoyer les données textuelles (Source: docs.celigo.com) (Source: docs.celigo.com). Bien que l'interface utilisateur de filtrage intégrée n'utilise pas nativement les regex, l'architecture de Celigo garantit que les capacités regex sont facilement accessibles en cas de besoin, soit dans les transformations, soit dans les filtres scriptés.

La **transformation des données** dans Celigo est complète. La Transformation 2.0 introduit plusieurs modes et types de champs pour remodeler les données selon les besoins (Source: docs.celigo.com) (Source: docs.celigo.com). Elle prend en charge l'aplatissement des documents imbriqués, la division des enregistrements et la génération de valeurs dérivées à l'aide de JSONPath et Handlebars (Source: docs.celigo.com) (Source: docs.celigo.com). Nous avons illustré comment les transformations peuvent transformer un JSON source complexe en formats de sortie plats adaptés aux applications en aval. Associées aux filtres d'entrée, les transformations permettent un large éventail de modèles d'intégration – par exemple, les clients de Celigo ont mis en œuvre des flux complets d'exécution des commandes et de synchronisation 3PL en utilisant exactement ces fonctionnalités (Source: www.celigo.com).

Pour l'avenir, Celigo améliore ces capacités avec des outils assistés par l'IA, facilitant la création et la documentation des filtres et des transformations. Le rôle de l'iPaaS s'étend d'une simple connectivité à celui de **« fondation d'automatisation »** pour les processus pilotés par l'IA (Source: www.celigo.com). Dans ce contexte, la logique de filtre et de transformation de Celigo restera centrale – même si les plateformes évoluent, les organisations devront s'assurer que les données sont exactes et correctement mises en forme à chaque étape (Source: www.celigo.com) (Source: www.celigo.com).

**En résumé**, la maîtrise des filtres d'entrée, des aides regex et des transformations de Celigo est cruciale pour construire des intégrations robustes. Ensemble, ils fournissent une boîte à outils complète : les filtres imposent quels enregistrements sont traités, les aides regex permettent une analyse fine du texte, et les transformations remodèlent les données selon les schémas cibles. L'application réfléchie de ces outils – comme dans l'exemple des ajustements Amazon (Source: docs.celigo.com) ou des flux d'entreprise comme ceux de Therabody (Source: www.celigo.com) – produit des pipelines de données efficaces et fiables. Pour tout projet d'intégration, une utilisation rigoureuse de ces fonctionnalités, soutenue par des tests et une surveillance (comme le conseillent les meilleures pratiques de Celigo (Source: docs.celigo.com), garantira que les flux de données sont corrects, performants et pérennes.

## Références

- Centre d'aide Celigo – *Apply filters* (aperçu des filtres d'entrée/sortie, opérateurs) (Source: docs.celigo.com) (Source: docs.celigo.com).

- Centre d'aide Celigo – *Regular expressions (regex)* (utilisation de regexMatch, regexReplace, regexSearch) (Source: docs.celigo.com) (Source: docs.celigo.com).

- Centre d'aide Celigo – *regexMatch helper* (exemples d'extraction de correspondances regex) (Source: docs.celigo.com).
- Celigo Help Center – *Assistant regexSearch* (exemples de recherche par regex renvoyant un index) (Source: docs.celigo.com).
- Celigo Help Center – *Assistant replace* (remplacement de sous-chaîne et note sur regexReplace) (Source: docs.celigo.com).
- Celigo Help Center – *Transformation 2.0* (modes, types de champs) (Source: docs.celigo.com) (Source: docs.celigo.com) ; *JSON path et handlebars* (Source: docs.celigo.com).
- Celigo Help Center – *Présentation des meilleures pratiques d'intégration* (Source: docs.celigo.com) (Source: docs.celigo.com).
- Blog Celigo – *Enterprise iPaaS : Pourquoi Celigo est leader* (architecture, statistiques de performance) (Source: www.celigo.com) (Source: www.celigo.com).
- Témoignage client Celigo – *Therabody* (exemple d'intégration d'entreprise) (Source: www.celigo.com).
- Celigo Help Center – *Ajouter un filtre d'entrée pour les ajustements d'inventaire FBA* (exemple de filtre étape par étape) (Source: docs.celigo.com) (Source: docs.celigo.com).
- Statistiques d'intégration et IA Integrate.io – *Statistiques sur les défis de transformation des données 2026* (données sectorielles sur l'intégration et les taux d'échec) (Source: www.integrate.io) (Source: www.integrate.io).


---

Étiquettes: celigo, filtres-dentree, operateur-matches, transformation-de-donnees, expressions-regulieres, integratorio, ipaas, helpers-handlebars

---