

# **Guide PDK de validation d'Unica Campaign 12.1**



# Table des matières

<b>Chapitre 1. Présentation du kit de développement de plug-in de validation (PDK).....</b>	<b>1</b>
Contenu du kit de développement de plug-in de validation.....	1
Deux modes d'utilisation de l'API de validation.....	2
Génération d'un plug-in de classe <sup>™</sup> qui est chargé dans l'application.....	3
Appeler une application pour gérer la validation.....	3
Validation des offres et des campagnes.....	4
Echantillons de valideurs inclus dans le PDK de validation.....	5
Routine de test du PDK de validation.....	6
Scripts de génération du PDK de validation.....	6
<b>Chapitre 2. Développement de plug-in de validation pour Unica Campaign.....</b>	<b>7</b>
Configuration de votre environnement en vue d'utiliser le PDK de validation :.....	7
Génération du plug-in valideur.....	8
Configuration de Unica Campaign en vue de l'utilisation d'un plug-in de validation.....	9
validationClass.....	10
validationClasspath.....	11
validatorConfigString.....	11
Test de la configuration du plug-in valideur.....	12
Création d'un plug-in valideur.....	13
Exemple validation scenario: prévention contre les éditions de campagnes.....	14
<b>Chapitre 3. Appel d'une application pour gérer la validation.....</b>	<b>15</b>
Configuration de Unica Campaign en vue de l'utilisation du plug-in exécutable exemple.....	15
Interface attendue pour l'exécutable.....	16

**Chapitre 4. Index.....**

# Chapitre 1. Présentation du kit de développement de plug-in de validation (PDK)

Utilisez le kit de développement de plug-in de validation (PDK) pour développer une logique de validation personnalisée à utiliser dans Unica Campaign.

Vous pouvez créer des plug-in pour exécuter une logique de validation personnalisée pour les campagnes et/ou les offres.

Voici quelques utilisations possibles de la logique de validation :

- Vérification des attributs étendus (personnalisés)
- Mise à disposition de services d'autorisation non fournis par Unica Platform (valider, par exemple, quels utilisateurs sont autorisés à éditer quels attributs).

Le PDK de validation est une sous-classe d'un ensemble de plug-in plus génériques fournis avec Unica Campaign.

Le PDK de validation contient des informations de référence Javadoc™ pour l'API de plug-in et pour l'exemple de code. Pour visualiser la documentation, ouvrez le fichier suivant dans votre navigateur Web :

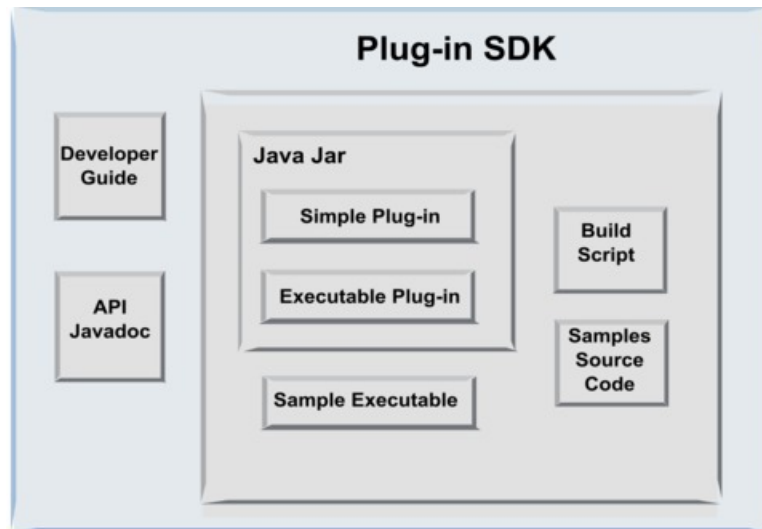
`C:\HCL\Unica\Campaign_Home\devkits\validation\javadoc\index.html`

Par exemple :

`C:\HCL\Unica\Campaign\devkits\validation\javadoc\index.html`

## Contenu du kit de développement de plug-in de validation

Le PDK de validation contient tous les composants nécessaires au développement de plug-in Java™ ou d'exécutables de ligne de commande pour effectuer des validations personnalisées dans Unica Campaign. Le PDK contient des échantillons documentés et pouvant être générés sur l'utilisation du PDK.



Le tableau ci-après présente chaque composant.

**Tableau 1. Composants du PDK de validation**

Composant	Description
Guide du développeur	Document PDF intitulé <i>Unica Campaign - Guide PDK de validation</i> .
API Javadoc	Informations de référence relatives à l'API de plug-in.
Fichier .jar Java :	Echantillon de fichier JAR qui contient les exemples de plug-in. Le fichier JAR contient : <ul style="list-style-type: none"> <li>• Plug-in simple : exemple de classe de plug-in valideur autonome.</li> <li>• Plug-in exécutable : échantillon du plug-in valideur qui exécute un exécutable de ligne de commande défini par l'utilisateur pour effectuer la validation.</li> </ul>
Exemple d'exécutable	Exécutable de ligne de commande pouvant être utilisé avec le plug-in d'exécutable sous UNIX™.
Script de génération	Script Ant qui génère le code source inclus pour en faire des plug-in valideurs utilisables.
Exemples de code source	Code source Java pour le valideur simple et le valideur exécutable.

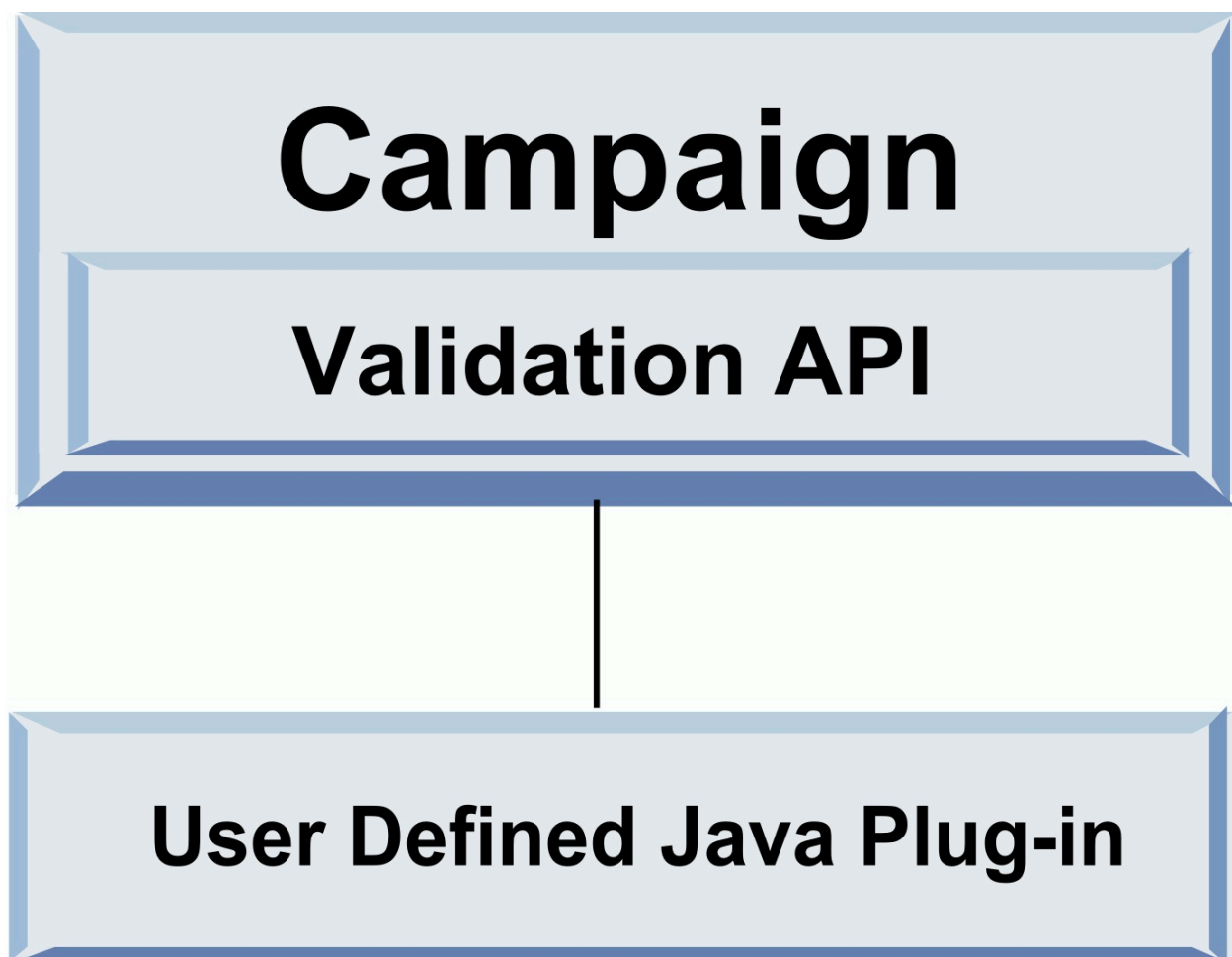
## Deux modes d'utilisation de l'API de validation

Il existe deux manières d'utiliser l'API de validation.

- L'utiliser pour générer un plug-in de classe Java qui est chargé dans l'application
- Utiliser un des plug-in inclus pour appeler une application exécutable permettant de gérer la validation

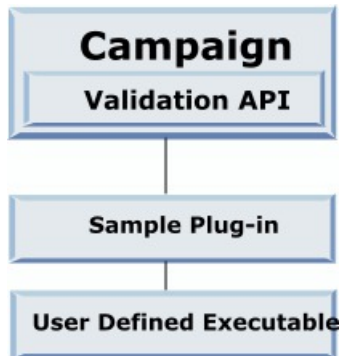
### Génération d'un plug-in de classe Java qui est chargé dans l'application

Le PDK de validation fournit les interfaces, classes auxiliaires et outils nécessaires au développement de ces classes.



## Appeler une application pour gérer la validation

Vous pouvez utiliser l'un des plug-in inclus pour appeler une application exécutable permettant de gérer la validation



L'exécutable peut être écrit dans n'importe quel langage, mais il doit se trouver et être exécuté sur le serveur Unica Campaign. Le plug-in qui appelle l'exécutable envoie un fichier XML qui contient les informations à valider (par exemple, l'utilisateur éditant l'objet ainsi que les valeurs précédentes et suivantes de tous les attributs standard et étendus de cet objet). Unica Campaign attend en retour un résultat sous la forme d'un fichier XML.

## Validation des offres et des campagnes

Un plug-in créé à l'aide du PDK de validation Unica Campaign peut exécuter une logique de validation personnalisée pour les campagnes et/ou les offres.

Le PDK de validation peut valider les offres et les campagnes. Si un plug-in de validation est défini, il est automatiquement appelé par Unica Campaign chaque fois qu'un objet d'offre ou de campagne est enregistré. Unica Campaign définit un indicateur lorsqu'il appelle la méthode de validation du plug-in. Unica Campaign transmet les indicateurs suivants :

- `ValidationInputData.CAMPAIGN_VALIDATION`, lors de l'ajout ou de la modification d'une campagne
- ou
- `ValidationInputData.OFFER_VALIDATION`, lors de l'ajout ou de l'édition d'une offre.

Vous pouvez ensuite utiliser ces indicateurs pour construire des règles de validation s'appliquant aux offres et aux campagnes.

## Echantillons de valideurs inclus dans le PDK de validation

Echantillons de valideurs inclus dans le PDK de validation Unica Campaign :

`SimpleCampaignValidator` et `ExecutableCampaignValidator`.

- `SimpleCampaignValidator` est un plug-in autonome qui présente comment effectuer des tâches telles qu'une autorisation personnalisée ou la validation de noms de campagne acceptables. Son chemin d'accès est le suivant :

```
devkits\validation\src\com\unica\campaign\core\validation\
samples\SimpleCampaignValidator.Java
```

Nous vous recommandons d'effectuer une copie de la classe avant de l'édition afin de conserver la version originale en cas de besoin.

- `ExecutableCampaignValidator` est un plug-in Java qui appelle une application exécutable pour effectuer la validation. Le code source du plug-in `ExecutableCampaignValidator` se trouve dans le même répertoire que le plug-in `SimpleCampaignValidator` :

```
devkits\validation\src\com\unica\campaign\core\validation\
samples\ExecutableCampaignValidator.Java
```

Toutefois, le véritable objectif de cet exemple est d'être utilisé comme exécutable de ligne de commande pour effectuer une validation. Ce fichier se trouve dans le chemin suivant :

```
devkits/validation/src/com/unica/campaign/core/validation/
samples/validate.sh
```

Ce fichier est un échantillon d'exécutable de bouclage, illustrant les types courants de validation.



## Routine de test du PDK de validation

Pouvoir tester le code de validation sans l'injecter dans Unica Campaign accélère le processus de développement de plug-in.

Les clients qui utilisent l'eXtreme Programming (XP) et les autres méthodologies agiles utilisent les tests d'unité de manière intensive. Le PDK de validation prend en charge ces méthodologies en proposant une routine de test pour exécuter un plug-in en dehors de Unica Campaign.

Pour utiliser la routine de test :

1. Modifiez le scénario de test d'unité pour refléter la logique de validation dans le plug-in.
2. Exécutez le script de génération :
  - Pour créer le plug-in sans effectuer de tests d'unité, exécutez les scripts de génération à l'aide de la commande `"ant jar"`.
  - Pour créer le plug-in et effectuer les tests d'unité, exécutez le script de génération à l'aide de la commande `"ant run-test"`.

## Scripts de génération du PDK de validation

Les scripts de génération du PDK de validation permettent de compiler toutes les classes dans un répertoire et de les placer dans un fichier JAR pouvant être utilisé dans Unica Campaign.

Le script de génération fourni utilise le répertoire suivant :

```
devkits/validation/src/com/unica/campaign/core/validation/samples/
```

# Chapitre 2. Développement de plug-in de validation pour Unica Campaign

Un plug-in est une classe Java qui est chargée au moment du démarrage et appelée dès lors qu'une campagne ou une offre est validée.

La validation est effectuée chaque fois qu'un utilisateur enregistre une campagne. Vous pouvez créer vos propres plug-in Java à l'aide des outils fournis dans le PDK de validation. Ce dernier contient le code source d'échantillon de plug-in ainsi qu'un fichier Ant (Apache Ant est un outil de compilation écrit en Java) que vous pouvez utiliser pour compiler des plug-in.

Les étapes suivantes indiquent comment configurer votre environnement pour développer un plug-in et créer votre propre plug-in.

1. [Configuration de votre environnement en vue d'utiliser le PDK de validation : \(à la page 7\)](#)
2. [Génération du plug-in valideur \(à la page 8\)](#)
3. [Configuration de Unica Campaign en vue de l'utilisation d'un plug-in de validation \(à la page 9\)](#)
4. [Test de la configuration du plug-in valideur \(à la page 12\)](#)
5. [Création d'un plug-in valideur \(à la page 13\)](#)

## Configuration de votre environnement en vue d'utiliser le PDK de validation :

Pour utiliser le PDK de validation avec Unica Campaign, vous devez modifier votre chemin d'accès et définir la valeur d'environnement `JAVA_HOME`.

Le PDK de validation peut être installé sur n'importe quelle machine, mais les plug-in que vous créez avec ce kit doivent être placés sur la machine exécutant Unica Campaign. Nous vous recommandons d'installer le PDK sur la machine sur laquelle vous testez vos plug-in.

Le PDK requiert que Apache Ant et un kit de développement Sun Java (JDK) soient installés sur votre machine pour créer des plug-in Java. A des fins de compatibilité, utilisez le script Ant et les modules JDK fournis avec votre serveur d'application.

Pour configurer votre environnement en vue d'utiliser le PDK de validation :

1. Ajoutez le dossier contenant l'exécutable Ant à votre chemin. Deux exemples sont fournis.
  - Par exemple, si WebLogic 11gR1 est installé dans le répertoire par défaut sous Windows™, ajoutez le chemin d'accès suivant à votre chemin : `C:\Oracle\Middleware\wlserver_10.3\common\bin`
  - Par exemple, si WebSphere® 7.0 est installé dans le répertoire par défaut sous Windows, ajoutez le chemin d'accès suivant à votre chemin : `C:\HCL\WebSphere\AppServer1\bin`
2. Affectez à la variable d'environnement `JAVA_HOME` le répertoire contenant les répertoires `bin` et `lib` du JDK. Deux exemples sont fournis.
  - Pour WebLogic 11gR1 sous Windows, donnez à `JAVA_HOME` la valeur `C:\Oracle\Middleware\jdk160_18`.
  - Pour WebSphere 7.0 sous Windows, donnez à `JAVA_HOME` la valeur `C:\HCL\WebSphere\AppServer1\java\jre`.

## Génération du plug-in valideur

Le PDK de validation Unica Campaign fournit un script Ant qui peut créer tout le code dans les exemples de fichiers.

Par défaut, le script crée un fichier JAR qui contient les classes de validation. Il est possible, en option, d'utiliser le script pour créer un Javadoc et exécuter des tests sur les plug-in valideurs pour vérifier qu'ils fonctionnent dans Unica Campaign avant de tenter de les utiliser en production.

Pour générer le valideur :

1. Accédez au répertoire PDK `<HCL_Unica_Home\Unica_Campaign_Home>\devkits\validation\build`.

Par exemple : `C:\HCL\Unica\Campaign\devkits\validation\build`

Cette propriété contient le script Ant `build.xml`.

2. Exécutez les fichiers JAR Ant depuis la ligne de commande.

- Pour créer le plug-in sans effectuer de tests d'unité, utilisez la commande `ant jar`.
- Pour créer le plug-in et effectuer des tests d'unité, utilisez la commande `ant run-test`.

Le fichier Ant exécute le script et crée un fichier JAR appelé `validator.jar` **lib** dans le sous-répertoire. Par exemple :


`C:\HCL\Unica\Campaign\devkits\validation\build\lib`

Vous disposez maintenant d'un valideur pouvant être utilisé dans Unica Campaign. L'étape suivante consiste à configurer Unica Campaign de sorte qu'il utilise ce valideur.

## Configuration de Unica Campaign en vue de l'utilisation d'un plug-in de validation

Pour configurer Unica Campaign en vue de l'utilisation d'un plug-in de validation, utilisez les paramètres de configuration dans `Unica Campaign > partitions > partition[n] > validation`.

Les propriétés de configuration indiquent à Unica Campaign comment trouver la classe de plug-in et permettent la transmission des informations de configuration aux plug-in.

 **Remarque** : La validation fonctionne avec plusieurs partitions ; `partition[n]` peut être remplacé par n'importe quel nom de partition pour fournir des routines de validation pour ces partitions également.

Vous pouvez ajuster les paramètres de configuration de validation suivants :

- [validationClass \(à la page 10\)](#)
- [validationClasspath \(à la page 11\)](#)

- [validatorConfigString \(à la page 11\)](#)

Pour utiliser `SimpleCampaignValidator`, définissez les propriétés comme suit :

- `validationClasspath` : `Unica\campaign\devkits\validation\lib\validator.jar`
- `validationClass` :  
`com.unica.campaign.core.validation.samples.SimpleCampaignValidator`
- Il n'est pas nécessaire de définir `validatorConfigString` pour utiliser `SimpleCampaignValidator`, car il n'utilise pas de chaîne de configuration.

Pour utiliser `ExecutableCampaignValidator`, définissez les propriétés comme suit :

- `validationClasspath` : `<Campaign_home>\devkits\validation\lib\validator.jar`
- `validationClass` :  
`com.unica.campaign.core.validation.samples.ExecutableCampaignValidator`
- `validatorConfigString` : `<Campaign_home>\pdk\bin\validate.sh`

## validationClass

`validationClass` indique à Unica Campaign le nom de la classe à utiliser pour la validation avec un plug-in PDK de validation.

Propriété	Description
Description	Nom de la classe à utiliser pour la validation. La valeur de la propriété <code>validationClasspath</code> indique l'emplacement de cette classe.
Détails	La classe doit être complète et doit ainsi s'accompagner de son nom de package. Si cette propriété n'est pas définie, Unica Campaign n'effectue aucune validation personnalisée.
Exemple	<pre>com.unica.campaign.core.validation. samples.SimpleCampaignValidator</pre> <p>Cet exemple permet d'affecter la classe <code>SimpleCampaignValidator</code> au paramètre <code>validationClass</code> de l'échantillon de code.</p>
Par défaut	Par défaut, aucun chemin n'est défini :

**Propriété****Description**

```
<property name="validationClass" />
```

## validationClasspath

`validationClasspath` indique à Unica Campaign l'emplacement de la classe à utiliser pour la validation avec un plug-in PDK de validation.

**Propriété****Description****Description**

Chemin de la classe utilisée pour la validation personnalisée.

**Détails**

Le chemin d'accès peut être complet ou relatif. Si le chemin est relatif, le comportement dépend du serveur d'applications qui exécute Unica Campaign. WebLogic utilise le chemin d'accès au répertoire de travail du domaine, qui est par défaut

```
c:\bea\user_projects\domains\mydomain.
```

Si le chemin se termine par une barre oblique (/ pour UNIX ou \ pour Windows), Unica Campaign considère qu'il pointe vers l'emplacement de la classe de plug-in Java à utiliser.

Si le chemin ne se termine pas par une barre oblique, Unica Campaign considère qu'il s'agit du nom d'un fichier .jar contenant la classe Java, comme indiqué dans l'exemple suivant.

Si le chemin n'est pas défini, Unica Campaign ne tente pas de charger un plug-in.

**Exemple**

```
/<CAMPAIGN_HOME>/devkits/validation/lib/validator.jar
```

Il s'agit du chemin qui, sur une plateforme UNIX, pointerait vers le fichier JAR qui est fourni avec le kit de développement de plug-in.

**Par défaut**

Par défaut, aucun chemin n'est défini :

```
<property name="validationClasspath" />
```

**Voir aussi**

Pour plus d'informations sur la désignation de la classe à utiliser, voir [validationClass \(à la page 10\)](#).

## validatorConfigString

`validatorConfigString` est transmis au plug-in valideur lorsqu'il est chargé par Unica Campaign.

Propriété	Description
Description	Chaîne transmise au plug-in valideur lorsqu'il est chargé par Unica Campaign.
Détails	<p>La manière dont le plug-in utilise cette chaîne dépend du concepteur. Vous pouvez l'utiliser pour envoyer une chaîne de configuration dans votre plug-in lorsque le système le charge.</p> <p>Par exemple, <code>ExecutableCampaignValidator</code> (qui provient de l'échantillon de plug-in exécutable inclut dans le kit de développement de plug-in) utilise cette propriété pour indiquer l'exécutable à exécuter.</p>
Exemple	<p>Pour exécuter le script de l'interpréteur de commandes Bourne comme script de validation, affectez à</p> <pre>validatorConfigString</pre> <p>à</p> <pre>/opt/unica/campaign/devkits/validation/src/com/unica/campaign / core/validation/samples/validate.sh</pre>
Par défaut	<p>Par défaut, aucun chemin n'est défini :</p> <pre>&lt;property name="validatorConfigString" /&gt;</pre>

## Test de la configuration du plug-in valideur

Après avoir généré le fichier `validator.jar` qui contient la classe `SimpleCampaignValidator` et apporté les changements de configuration requis, vous pouvez tester et utiliser le plug-in.

L'échantillon de plug-in suivant empêche les utilisateurs de Unica Campaign d'enregistrer une campagne portant le nom "badCampaign".

Pour tester votre configuration :

1. Redéployez votre serveur d'application pour que les modifications soient prises en compte. Pour connaître les instructions, consultez la documentation de votre serveur.
2. Connectez-vous à Unica Campaign et accédez à la page de création de campagne.
3. Créez une campagne portant le nom **badCampaign** et essayez de l'enregistrer.

Si tout est correctement configuré, vous ne devez pas pouvoir enregistrer la nouvelle campagne. Si vous obtenez un message d'erreur du plug-in valideur, cela signifie qu'il fonctionne correctement.

## Création d'un plug-in valideur

Ces instructions indiquent comment créer un plug-in de validation qui ressemble à `SimpleCampaignValidator`, mais empêche la création de campagnes appelées "badCampaign2".

1. Copiez l'échantillon du plug-in valideur `SimpleCampaignValidator.java` dans `<HCL_Unica_Home\Campaign_Home>\devkits\validation\src\com\unica\campaign\core\validation\samples` : Nommez la copie `MyCampaignValidator.java` et conservez-la dans le même répertoire que le fichier source. Par exemple :  
  
`C:\HCL\Unica\Campaign\devkits\validation\src\com\unica\campaign\core\validation\samples\MyCampaignValidator.java`
2. Ouvrez `MyCampaignValidator.java` dans un éditeur. Recherchez le terme "badCampaign" et remplacez-le par "badCampaign2."
3. Enregistrez le fichier et fermez l'éditeur.
4. Générez de nouveau les plug-in valideurs. Pour plus de détails, voir [Génération du plug-in valideur \(à la page 8\)](#). Si le serveur d'applications verrouille le fichier `validate.jar` pendant son utilisation, arrêtez le serveur avant de générer les valideurs.



5. Reconfigurez `campaign_config.xml` pour utiliser votre nouvelle classe : 

```
<property name="validationClass" value="com.unica.campaign.core.validation.samples.MyCampaignValidator">
```
6. Testez le plug-in valideur. Pour plus de détails, voir [Test de la configuration du plug-in valideur \(à la page 12\)](#).

Vérifiez que le valideur fonctionne : Vous ne devriez plus pouvoir enregistrer de campagnes appelées "badCampaign2."

## Exemple validation scenario: prévention contre les éditions de campagnes

Cet exemple indique comment utiliser la validation pour empêcher des éditions spécifiques d'une campagne.

Si vous tentez d'empêcher qu'une personne éditant une campagne modifie le code de campagne, vous pouvez utiliser une routine de validation de campagne personnalisée. La routine s'assure que la vérification suivante est effectuée une fois la campagne enregistrée :

```
new_campaign_code == old_campaign_code
```

Pour gérer la casse où la campagne est créée pour la première fois, transmettez à la routine un indicateur signalant si la campagne validée est nouvelle (indicateur "creation") ou existante (indicateur "edit"). Si cet indicateur est **edit**, procédez à la comparaison des codes de campagne.

L'application Campaign définit cet indicateur dans l'objet `InputValidationData` puis le transmet au plug-in. Ce dernier lit l'indicateur lorsqu'il détermine si la validation concerne une campagne nouvelle ou modifiée.

# Chapitre 3. Appel d'une application pour gérer la validation

Le PDK de validation comprend un échantillon de plug-in valideur, `ExecutableCampaignValidator`, qui lance un exécutable, `validate.sh`, à partir de la ligne de commande pour effectuer la validation.

Les sections suivantes indiquent comment :

- configurer Unica Campaign pour lancer l'échantillon de plug-in exécutable et
- créer votre propre plug-in exécutable conforme à l'utilisation de l'interface de syntaxe de l'exécutable.

## Configuration de Unica Campaign en vue de l'utilisation du plug-in exécutable exemple

Pour utiliser `ExecutableCampaignValidator`, ajustez les paramètres de configuration dans `Unica Campaign > partitions > partition[n] > validation`.

Définissez les propriétés comme suit :

- `validationClasspath` :

```
<Unica Campaign_home>\devkits\validation\lib\validator.jar
```

- `validationClass` :

```
com.unica.campaign.core.validation.samples.ExecutableCampaignValidator
```

- `validatorConfigString` :

```
<Unica Campaign_home>\pdk\bin\validate.sh
```

L'échantillon de script fourni avec le PDK de validation est un script de l'interpréteur de commandes Bourne pour UNIX. Il interdit à tout utilisateur portant le nom "badUser" de

créer une campagne. Vous pouvez visualiser le code de cet exécutable dans le répertoire suivant :

```
devkits\validation\src\com\unica\campaign\core\validation\
samples\validate.sh
```

Vous devez développer votre propre script effectuant la validation appropriée à votre mise en œuvre. Les langages de script tels que PERL et Python sont recommandés pour ce genre de scripts de traitement de texte mais n'importe quel langage pouvant être exécuté à partir de la ligne de commande est également acceptable.

## Interface attendue pour l'exécutable

Le plug-in `ExecutableCampaignValidator` appelle un fichier exécutable avec une ligne de commande contenant les arguments ci-dessous.


- `executable_name` : chaîne définie pour la propriété `validatorConfigString` dans Unica Platform.
- `data_filename` : nom du fichier lu par l'exécutable en entrée. Les données en entrée doivent être au format XML.
- `expected_result_filename` : nom du fichier devant être envoyé par l'exécutable en sortie. Les résultats attendus sont au format `data XXX.xml`, où XXX est un nombre.
  - Voici un exemple d'envoi de données dont la validation a abouti :

```
<ValidationResult result="0" generalFailureDeliver="" />
```

- Voici un exemple d'envoi de données dont la validation a échoué :

```
<ValidationResult result="1" generalFailureDeliver="">
  <AttributeError attributeName="someAttribute"
    errorMessage="something" />
  <AttributeError attributeName="someAttribute2"
    errorMessage="something2" />
</ValidationResult>
```

- Le texte du fichier XML doit être codé en ASCII ou en UTF-8.

 **Remarque** : Il est fortement recommandé d'écrire des messages d'erreur explicites pour que les utilisateurs puissent corriger le problème avant de tenter un nouvel enregistrement.