Unica Campaign 検証 PDK ガイド V12.1

目次

第 1 章. 検証 Plug-in Developer's Kit (PDK) の概要 1

検証 PDK の内容 1

検証 API を使用する 2 つの方法 2

アプリケーションにロードされる Java™ クラス・プラグインの作成 3

検証を処理するためのアプリケーションの呼び出し 3

オファーとキャンペーンの検証 4

検証 PDK に含まれるサンプルのバリデーター 4

検証 PDK 用のテスト・ハーネス 5

検証 PDK 用のスクリプトのビルド 5

第 2 章. Unica Campaign 用検証プラグインの開発 7

検証 PDK 使用のための環境のセットアップ 7

バリデーターのビルド 8

検証プラグインを使用するように Unica Campaign を構成する 8

validationClass 9

validationClasspath 9

validatorConfigString 10

バリデーター構成のテスト 11

バリデーターの作成 11

検証シナリオの例: キャンペーン編集を防止する 12

第 3 章. 検証を処理するためのアプリケーションの呼び出し 13

サンプル実行可能プラグインを使用するように Unica Campaign を構成する 13

想定される実行可能プログラム使用インターフェース 13

索引 .....

第 1 章. 検証 Plug-in Developer's Kit (PDK) の概要

Unica Campaign で使用するカスタム検証ロジックを開発するために、検証 Plug-in Developer's Kit (PDK) を使用します。

キャンペーン、オファー、またはその両方に関するカスタム検証ロジックを実行するプラグインを作成できます。

検証ロジックを次のように使用できます。

- 拡張 (カスタム) 属性の検査
- Unica Platform のスコープ外にある許可サービスの提供 (例えば、どのユーザーがどの拡張属性の編集を許可されるかの検証)

検証 PDK は、 Unica Campaign に付属している汎用性の高いプラグイン・フレームワークのサブクラスです。

検証 PDK には、プラグイン API およびサンプル・コードに関する Javadoc™ 参照情報が含まれています。資料を表示するには、Web ブラウザーで以下のファイルを開いてください。

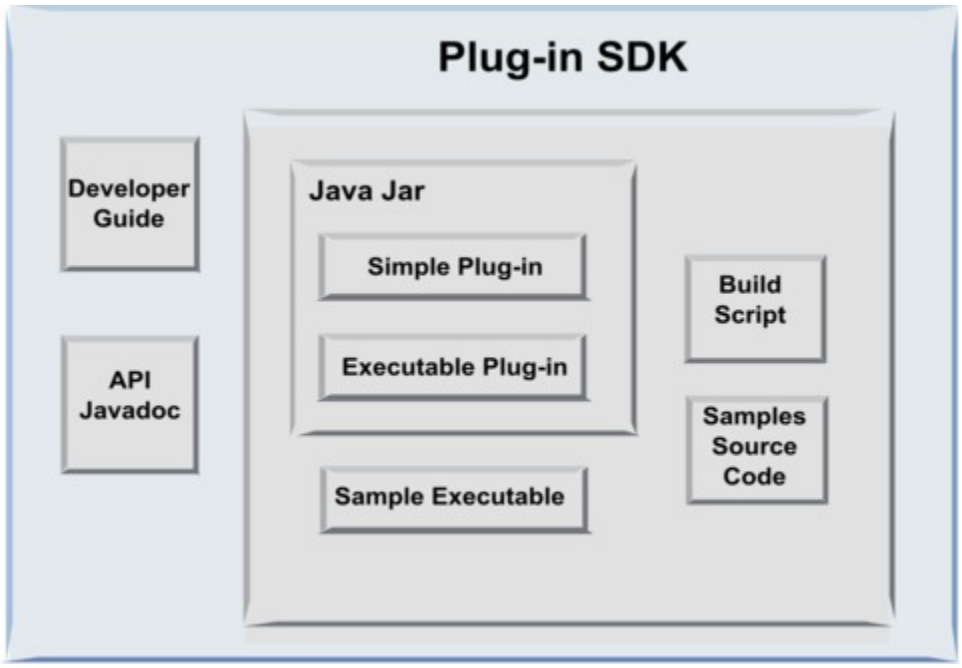
C:\HCL\Unica*Campaign_Home*\devkits\validation\javadoc\index.html

例:

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

検証 PDK の内容

検証 PDK には、 Unica Campaign にカスタム検証を追加するための Java™ プラグインまたはコマンド・ライン実行可能プログラムの開発用コンポーネントが含まれています。PDK には、PDK を使用する方法を示す、文書化されたビルド可能な例が含まれています。



次の表に各コンポーネントの説明を示します。

表 1. 検証 PDK のコンポーネント

コンポーネント	説明
開発者ガイド	Unica Campaign 検証 PDK ガイドというタイトルの PDF 文書。
API Javadoc™	プラグイン API の参照情報。
Java™ .jar ファイル	サンプル・プラグインが入っているサンプル JAR ファイル。JAR ファイルには、以下のものが含まれています。 • 単純なプラグイン: 自己完結型バリデーター・クラスの例。 • 実行可能プラグイン: 検証のためにユーザー定義のコマンド・ライン実行可能プログラムを実行するバリデーターの例。
サンプル実行可能プログラム	UNIX™ で実行可能プラグインとともに使用できるコマンド・ライン実行可能プログラム。
ビルド・スクリプト	組み込みソース・コードをビルドして使用可能なバリデーター・プラグインにする Ant スクリプト。
サンプル・ソース・コード	単純なバリデーターおよび実行可能バリデーター用の Java™ ソース・コード。

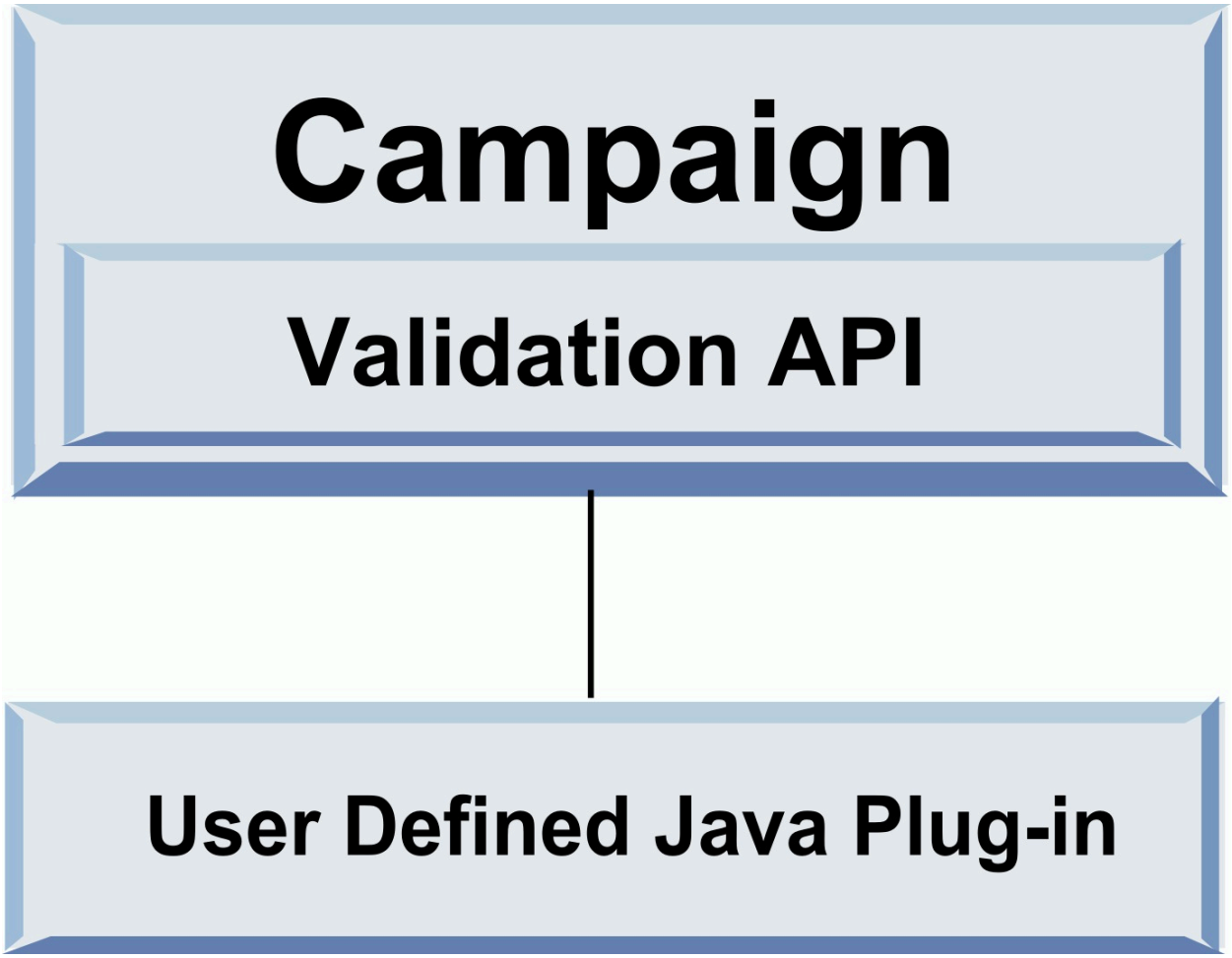
検証 API を使用する 2 つの方法

検証 API を使用するには 2 つの方法があります。

- これを使用して、アプリケーションにロードされる Java™ クラス・プラグインを作成します。
- いずれかの組み込みプラグインを使用して、検証を扱う実行可能アプリケーションを呼び出します。

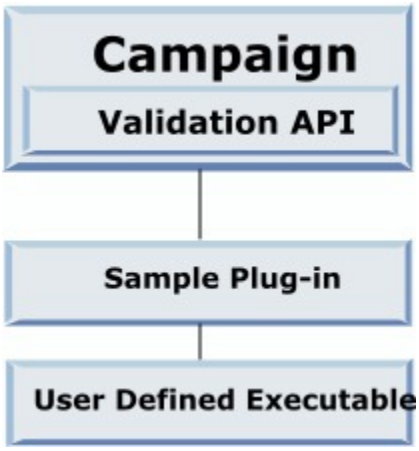
アプリケーションにロードされる Java™ クラス・プラグインの作成

検証 PDK は、インターフェース、ヘルパー・クラス、およびこれらのクラスを開発するための開発者ツールを備えています。



検証を処理するためのアプリケーションの呼び出し

いずれかの検証 PDK 組み込みプラグインを使用して、検証を扱う実行可能アプリケーションを呼び出すことができます。



任意の言語でこの実行可能プログラムを作成できますが、 Unica Campaign サーバー上にこれを格納して、サーバー上で実行されるようにする必要があります。実行可能プログラムを呼び出すプラグインは、検証される情報を含む XML ファイルを送信します。例えば、オブジェクトを編集しているユーザー、そのオブジェクトのすべての標準および拡張属性の変更前と後の値です。 Unica Campaign は、XML ファイル形式で結果情報が戻されることを想定しています。

オファーとキャンペーンの検証

Unica Campaign 検証 PDK を使って作成されたプラグインは、キャンペーン、オファー、またはその両方に関するカスタム検証ロジックを実行することができます。

検証 PDK ではオファーとキャンペーンを検証することができます。検証プラグインを定義すると、オファーまたはキャンペーンのオブジェクトが保存されるたびに、その検証プラグインが Unica Campaign によって自動的に呼び出されます。 Unica Campaign はプラグインの検証方法を呼び出すときにフラグを設定します。 Unica Campaign は次のフラグを渡します。

- `ValidationInputData.CAMPAIGN_VALIDATION`キャンペーンの追加または変更時

  または
- `ValidationInputData.OFFER_VALIDATION`オファーの追加または編集時。

その後、これらのフラグを使用して、オファーやキャンペーンに適用される検証規則を構成することができます。

検証 PDK に含まれるサンプルのバリデーター

Unica Campaign 検証 PDK には、次の 2 つのサンプルのバリデーターが含まれています。SimpleCampaignValidator および ExecutableCampaignValidator。

- SimpleCampaignValidator は、カスタム許可や許容されるキャンペーン名の検証などの方法を示す、自己完結型のプラグインです。これは、以下のパスにあります。

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

  必要な場合に元のバージョンを保持できるよう、クラスを編集する前にコピーを作成することをお勧めします。
- ExecutableCampaignValidator は、検証を実行するために実行可能アプリケーションを呼び出す Java™ プラグインです。ExecutableCampaignValidator のソース・コードは SimpleCampaignValidator と同じディレクトリーに入っています。

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

  ただし、このサンプルの本当の目的は、検証用のコマンド・ライン実行可能プログラムとして使用することです。このファイルは次のパスにあります。

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

  このファイルはサンプル・ループバック実行可能プログラムであり、一般的な種類の検証作業を示しています。

検証 PDK 用のテスト・ハーネス

Unica Campaign の中に検証コードを配置しなくてもテストできるため、プラグイン開発者のプロセスがより迅速になります。

エクストリーム・プログラミングおよびその他のアジャイル手法を使用する場合、単体テストが広く使用されています。検証 PDK は、Unica Campaign の外部でプラグインを実行するためのテスト・ハーネスを備えることにより、これらの手法をサポートします。

テスト・ハーネスを使用するには、次のようにします。

1. プラグイン内の検証ロジックを反映するよう、単体テスト・ケースを変更します。
2. ビルド・スクリプトを次のように実行します。
   - 単体テストを行わずにプラグインを作成するには、「ant jar」コマンドを使ってビルド・スクリプトを実行します。
   - プラグインを作成するとともに単体テストを行うには、「ant run-test」コマンドを使ってビルド・スクリプトを実行します。

検証 PDK 用のスクリプトのビルド

検証 PDK のビルド・スクリプトはディレクトリー内のすべてのクラスをコンパイルして、Unica Campaign での使用に適した JAR ファイルの中にそれらを格納します。

提供されているビルド・スクリプトは、以下のディレクトリーを使用します。

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

第 2 章. Unica Campaign 用検証プラグインの開発

プラグインは、起動時にロードされ、キャンペーンまたはオファーが検証されるたびに呼び出される Java™ クラスです。

ユーザーがキャンペーンを保存するたびに発生する検証です。検証 PDK に備わっているツールを使用して、独自の Java™ プラグインを作成できます。PDK には、サンプル・プラグインのソース・コードと、プラグインのコンパイルに使用する Ant ファイル (Apache Ant は Java™ ベースのビルド・ツール) が含まれます。

以下の手順で、プラグインの開発環境をセットアップする方法、および独自のプラグインを作成する方法について説明します。

1. 検証 PDK 使用のための環境のセットアップ(7 ページ)
2. バリデーターのビルド(8 ページ)
3. 検証プラグインを使用するように Unica Campaign を構成する(8 ページ)
4. バリデーター構成のテスト(11 ページ)
5. バリデーターの作成(11 ページ)

検証 PDK 使用のための環境のセットアップ

検証 PDK を Unica Campaign で使用するには、`JAVA_HOME` 環境変数のパスを変更し、設定する必要があります。

任意のマシンに検証 PDK をインストールできますが、これを使って作成するプラグインは、 Unica Campaign を実行しているマシン上に置く必要があります。プラグインのテストに使用するマシン上に PDK をインストールすることをお勧めします。

PDK で Java™ プラグインを作成するには、マシン上に Apache Ant および Sun Java™ Developer Kit が必要です。互換性を確実に得るためには、アプリケーション・サーバーに付属の Ant および JDK パッケージを使用してください。

検証 PDK の使用のために環境をセットアップするには、次のようにします。

1. Ant 実行可能プログラムが入っているフォルダーをパスに追加します。2 つの例を示します。
    - Windows™ 上のデフォルト・ディレクトリーに WebLogic 11gR1 がインストールされている場合、以下をパスに追加します。 `C:\Oracle\Middleware\wlserver_10.3\common\bin`
    - Windows™ 上のデフォルト・ディレクトリーに WebSphere® 7.0 がインストールされている場合、以下をパスに追加します。 `C:\HCL\WebSphere\AppServer1\bin`
2. `JAVA_HOME` 環境変数を、JDK の `bin` および `lib` ディレクトリーが入っているディレクトリーに設定します。2 つの例を示します。
    - Windows™ の WebLogic 11gR1 の場合、`JAVA_HOME` を `C:\Oracle\Middleware\jdk160_18` に設定します。
    - Windows™ の WebSphere® 7.0 の場合、`JAVA_HOME` を `C:\HCL\WebSphere\AppServer1\java\jre` に設定します。

バリデーターのビルド

Unica Campaign 検証 PDK に備わっている Ant スクリプトは、サンプル・ファイル内のすべてのコードをビルドすることができます。

このスクリプトは、デフォルト動作として、検証クラスを含む jar を作成します。また、実稼働でのプラグイン使用を試みる前に Unica Campaign で機能することを確認するために、バリデーターに対する Javadoc™ および run test をオプションで作成することもできます。

バリデーターをビルドするには、次のようにします。

1. PDK ディレクトリーに移動します <HCL_Unica_Home\Unica Campaign_Home>\devkits\validation\build

   例: C:\HCL\Unica\Campaign\devkits\validation\build

   このディレクトリーに Ant スクリプト `build.xml` があります。
2. コマンド・ラインで Ant jar を実行します。
   - 単体テストを行わずにプラグインを作成するには、`ant jar` コマンドを使用します。
   - プラグインを作成し、単体テストも行うには、`ant run-test` コマンドを使用します。

   Ant はスクリプトを実行して、`validator.jar` という JAR ファイルを lib サブディレクトリー内に生成します。例:

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

これで、Unica Campaign で使用可能なカスタム・バリデーターが生成されました。次の手順は、このバリデーターを使用するための Unica Campaign を構成することです。

検証プラグインを使用するように Unica Campaign を構成する

検証プラグインを使用するように Unica Campaign を構成するには、`Unica Campaign > partitions > partition[n] > validation` にある構成設定を使用します。

構成プロパティーによって、Unica Campaign がプラグイン・クラスを見つける方法が指定され、構成情報をプラグインへ渡す方法が示されます。

注: 検証は複数のパーティションで使用できます。つまり、`partition[n]` を任意のパーティション名に変更して、複数のパーティションの検証ルーチンを提供することもできます。

以下の検証構成設定を調整できます。

- validationClass(9 ページ)
- validationClasspath(9 ページ)
- validatorConfigString(10 ページ)

`SimpleCampaignValidator` を使用するには、以下のようにプロパティーを設定します。

- validationClasspath: *Unica*\campaign\devkits\validation\lib\validator.jar
- validationClass: com.unica.campaign.core.validation.samples.SimpleCampaignValidator
- 構成ストリングは使われないため、SimpleCampaignValidator の使用のために validatorConfigString を設定する必要はありません。

ExecutableCampaignValidator を使用するには、以下のようにプロパティーを設定します。

- 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 は Unica Campaign に、検証 PDK プラグインで検証に使用するクラスの名前を指定します。

プロパティー	説明
説明	検証に使用するクラスの名前です。validationClasspath プロパティーの値は、このクラスの場所を示します。
詳細	クラスは、パッケージ名で完全修飾する必要があります。このプロパティーが設定されていない場合、Unica Campaign はいかなるカスタム検証も実行しません。

例	`com.unica.campaign.core.validation.`
	`samples.SimpleCampaignValidator`
	この例は、validationClass をサンプル・コードの SimpleCampaignValidator クラスに設定します。

デフォルト	デフォルトでは、パスが設定されていません。 `<property name="validationClass" />`

validationClasspath

validationClasspath は Unica Campaign に、検証 PDK プラグインの検証で使用するクラスの場所を指定します。

プロパティー	説明
説明	カスタム検証で使用されるクラスへのパスです。
詳細	絶対パスか相対パスのいずれかを使用します。相対パスである場合、Unica Campaign を実行しているアプリケーション・サーバーによって動作が異なります。WebLogic ではドメイン作業ディレクトリーのパスが使用され、このパスはデフォルトでは次のとおりです。

プロパティー	説明
	c:\bea\user_projects\domains\mydomain。 パスの末尾がスラッシュ (UNIX™ では /、Windows™ では \) になっている場合、Unica Campaign では使用される Java™ プラグイン・クラスの位置を指しています。 パスの末尾がスラッシュでない場合、Unica Campaign では、以下の例に示されているように Java™ クラスを含む .jar ファイルの名前と見なされます。 パスが設定されていない場合、Unica Campaign はプラグインのロードを試行しません。
例	/<*CAMPAIGN_HOME*>/devkits/validation/lib/validator.jar これは、プラグイン開発者キットにパッケージ化されている JAR ファイルを指す、UNIX™ プラットフォーム上のパスです。
デフォルト	デフォルトでは、パスが設定されていません。 <property name="validationClasspath" />
関連項目	使用するクラスの指定については、validationClass(9 ページ) の情報を参照してください。

validatorConfigString

validatorConfigString は、Unica Campaign によってロードされるときにバリデーター・プラグインに渡されます。

プロパティー	説明
説明	Unica Campaign によってロードされるときにバリデーター・プラグインに渡されるストリングです。
詳細	プラグインがこのストリングをどのように使用するかは、設計によって異なります。システムがプラグインをロードするときに構成ストリングをプラグインに送るために、これを使用できます。 例えば、(PDK に含まれるサンプル実行可能プラグインの) ExecutableCampaignValidator は、実行すべき実行可能プログラムを示すためにこのプロパティーを使用します。
例	検証スクリプトとしてサンプル Bourne シェル・スクリプトを実行するには、次のように設定します。 validatorConfigString 変更後の内容: /opt/unica/campaign/devkits/validation/src/com/unica/campaign /core/validation/samples/validate.sh
デフォルト	デフォルトでは、パスが設定されていません。 <property name="validatorConfigString" />

バリデーター構成のテスト

SimpleCampaignValidator クラスを含む validator.jar ファイルをビルドし、必要に応じて構成を変更した後、プラグインをテストして、使用することができます。

次のプラグインの例では、Unica Campaign ユーザーが「badCampaign」という名前のキャンペーンを保存できないようにします。

構成をテストするには、次のようにします。

1. 変更を有効にするために、アプリケーション・サーバーを再デプロイします。手順については、サーバーの資料を参照してください。
2. Unica Campaign にログインして、キャンペーン作成ページまで移動します。
3. **badCampaign** という名前のキャンペーンを作成して、それを保存してみます。

すべての構成が正しい場合、その新しいキャンペーンを保存することはできません。バリデーターからエラー・メッセージを受け取れば、正しく処理されていることが分かります。

バリデーターの作成

SimpleCampaignValidator によく似た、しかし「badCampaign2」というキャンペーンの作成を抑止する検証プラグインを作成するには、この手順に従ってください。

1. <HCL_Unica_Home\Campaign_Home>\devkits\validation\src\com \unica\campaign\core\validation \samples にサンプル・バリデーター SimpleCampaignValidator.java のコピーを作成します。そのコピーに MyCampaignValidator.java という名前を付けて、ソースと同じディレクトリーにそれを配置します。例:

   C:\HCL\Unica\Campaign\devkits\validation\src\com \unica\campaign\core\validation\samples \MyCampaignValidator.java
2. エディターで MyCampaignValidator.java を開きます。文書内の「badCampaign」という語を見つけて、それを「badCampaign2」という語に置き換えます。
3. ファイルを保存してエディターを閉じます。
4. バリデーターを再びビルドします。詳しくは、バリデーターのビルド(8 ページ) を参照してください。アプリケーション・サーバーによって validate.jar ファイルが使用中にロックされる場合は、バリデーターをビルドする前にサーバーを停止してください。
5. 新しいクラスを使用するよう、campaign_config.xml を次のように再構成します。 <property name="validationClass" value="com.unica.campaign.core.validation.samples.MyCampaignValidator">
6. バリデーターをテストします。詳しくは、バリデーター構成のテスト(11 ページ) を参照してください。

バリデーターが機能することを確認します。これで、「badCampaign2」という名前のキャンペーンを保存できなくなります。

検証シナリオの例: キャンペーン編集を防止する

この例では、検証を使用してキャンペーンへの特定の編集を防止する方法を説明します。

キャンペーンを編集するユーザーによってキャンペーン・コードが変更されるのを防ぐには、カスタム・キャンペーン検証ルーチンを使用できます。ルーチンは、キャンペーンが保存された時に、以下のチェックが必ず行われるようにします。

```
new_campaign_code == old_campaign_code
```

キャンペーンを初めて作成する場合に対処するため、検証対象のキャンペーンが新規 (作成) または既存 (編集) のどちらであるかを示すフラグをルーチンに渡します。このフラグが**編集**を示している場合、キャンペーン・コードを比較します。

Campaign アプリケーションは `InputValidationData` オブジェクトにこのフラグを設定した後、プラグインにこのオブジェクトを渡します。プラグインは、新規キャンペーンまたはキャンペーン変更のどちらの検証であるかを判別するときに、このフラグを読み取ります。

第 3 章. 検証を処理するためのアプリケーションの呼び出し

検証 PDK にはサンプルのバリデーター `ExecutableCampaignValidator`、つまり検証を行うためにコマンド・ラインから実行する実行可能プログラム `validate.sh` が含まれています。

次のセクションでその方法を説明します。

- Unica Campaign をサンプルの実行可能プラグインを実行するように構成します。次に
- 実行可能プログラム使用インターフェースに適合する独自の実行可能プラグインを作成します。

サンプル実行可能プラグインを使用するように Unica Campaign を構成する

`ExecutableCampaignValidator` を使用するには、`Unica Campaign > partitions > partition[n] > validation` で構成設定を調整します。

以下のようにプロパティーを設定します。

- 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
  ```

検証 PDK に付属のサンプル・スクリプトは、UNIX™ 用の Bourne シェル・スクリプトです。これは、ユーザー名「badUser」を持つすべてのユーザーによるキャンペーン作成を拒否します。この実行可能プログラムのコードは、以下のディレクトリーにあります。

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

実装に合わせて適切な検証を実行する、独自のスクリプトを開発する必要があります。このようなテキスト処理スクリプトには、PERL、Python などのスクリプト言語はもちろん、コマンド・ラインから実行可能な任意の言語を使用できます。

想定される実行可能プログラム使用インターフェース

`ExecutableCampaignValidator` プラグインは、以下の引数を含むコマンド・ラインを使って実行可能ファイルを呼び出します。

- executable_name: Unica Platform の validatorConfigString に設定されたストリング。
- data_filename: 実行可能プログラムが入力として読み取るファイルの名前。入力データは XML としてフォーマット設定される必要があります。
- expected_result_filename: 実行可能プログラムが出力として送信するファイルの名前。想定される結果は data_XXX.xml という形式になります (XXX は数値)。
    - 成功したデータは、例えば次のように送信されます。

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

    - 失敗したデータは、例えば次のように送信されます。

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

    - XML ファイル内のテキストは通常の ASCII 文字または UTF-8 でエンコードされる必要があります。

注: 保存操作を再び試行する前にユーザーが問題を修正できるよう、理解しやすいエラー・メッセージを提供することを強くお勧めします。