

IBM Campaign
Version 9 Release 0
January 15, 2013

Validation PDK Guide

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 13.

This edition applies to version 9, release 0, modification 0 of IBM Campaign and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 1998, 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

IBM Validation Plug-in Developer's Kit (PDK) 1

Additional Validation PDK help	1
Contents of the Validation PDK	1
Components of the Validation PDK.	2
Finding reference material	2
Capabilities of the Validation PDK	2
Load plug-in into IBM Campaign	2
Offer versus campaign validation	3
Configuring the Validation PDK.	4
validationClass	4
validationClasspath	4
validatorConfigString	5
Developing a plug-in	6
Overview	6
Major steps to create plug-ins.	7
Setup	7

Build the validators	7
Configure IBM Campaign	8
Test the validator configuration	8
Create a validator.	8
Executable Sample	9
Configure Campaign for the sample executable plug-in	9
Expected executable usage interface	9
Example	10
Prevent campaign edits	10

Contacting IBM technical support . . . 11

Notices 13

Trademarks	15
Privacy Policy and Terms of Use Considerations . . .	15

IBM Validation Plug-in Developer's Kit (PDK)

The IBM® Validation Plug-in Developer's Kit (PDK) allows you to develop custom validation logic for use in IBM Campaign. The Validation PDK is a subclass of a more generic plug-in framework provided with IBM Campaign.

Key sections in this chapter include:

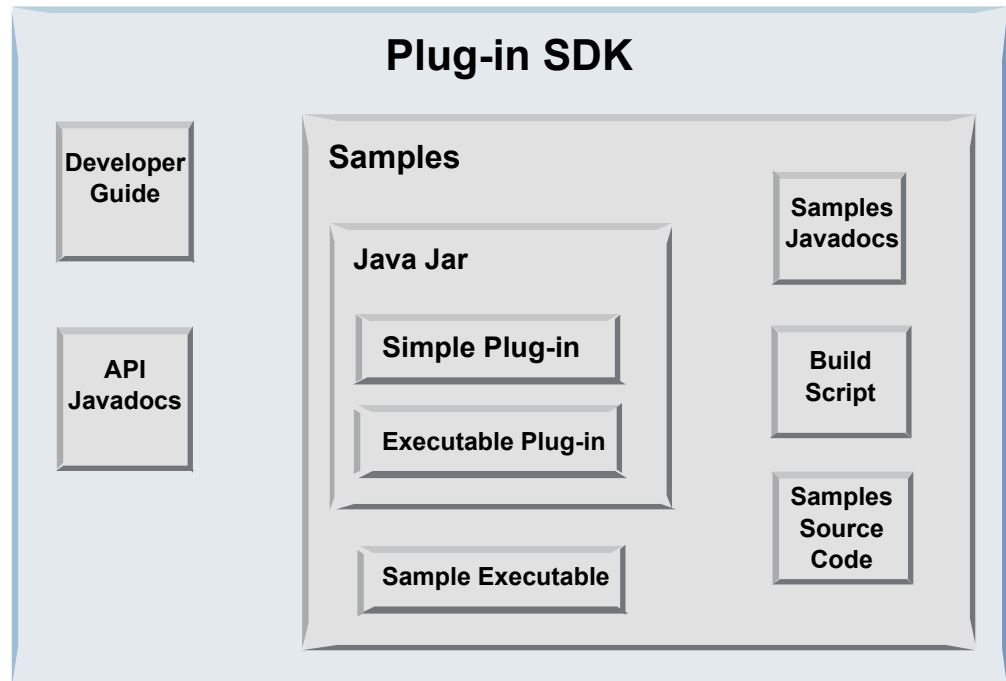
- “Contents of the Validation PDK”
- “Capabilities of the Validation PDK” on page 2
- “Configuring the Validation PDK” on page 4
- “Developing a plug-in” on page 6
- “Major steps to create plug-ins” on page 7
- “Executable Sample” on page 9
- “Example” on page 10

Additional Validation PDK help

If you require help using the Validation Plug-in Developer's Kit (PDK), contact IBM Technical Support with your questions. For details, see “Contacting IBM technical support” on page 11.

Contents of the Validation PDK

The Validation PDK contains everything that you need to develop Java™ plug-ins or command-line executables to add validation to IBM Campaign.



Components of the Validation PDK

The Validation Plug-in Developer's Kit (PDK) has the following components.

Component	Description
Validation PDK Guide	This document.
Samples	The samples are documented, buildable examples of how to use the PDK.
Java .jar file	A sample JAR file containing the sample plug-ins. The jar contains: <ul style="list-style-type: none">• Simple Plug-in: an example of a self-contained validator class.• Executable Plug-in: an example validator that runs a user-defined command line executable to perform validation.
Sample Executable	A command-line executable that can be used with the executable plug-in on UNIX.
Sample Javadocs	Javadocs describing the sample validator classes.
Build Script	An Ant script that builds the included source code into usable validator plug-ins.
Samples Source Code	The Java source code for the simple validator and the executable validator.

Finding reference material

The Validation PDK contains reference information for both the Plug-In API and the sample code. The reference information is in the form of Javadocs. To view the documentation, open the following file:

```
C:\IBM_EMM_Home\Campaign_Home\devkits\validation\javadoc\index.html
```

Where *IBM_EMM_Home* is the path to your IBM root installation directory and *Campaign_Home* is the path to your Campaign install directory.

For example, the path to the documentation might be similar to this example:

```
C:\IBM\Campaign\devkits\validation\javadoc\index.html
```

You can view the Javadocs in any web browser.

Capabilities of the Validation PDK

A plug-in made with the Validation PDK can perform custom validation logic for campaigns, offers, or both. Some possible uses of the validation logic are:

- Checking extended (custom) attributes
- Providing authorization services that are outside of the scope of IBM Marketing Platform (for example, validating which users are allowed to edit which extended attributes).

Load plug-in into IBM Campaign

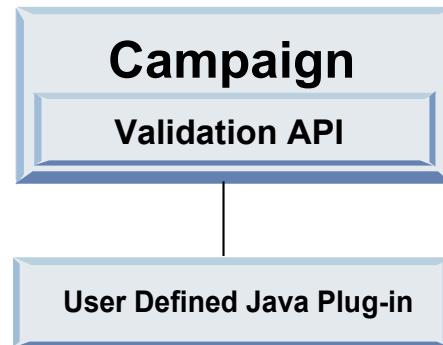
There are two ways to use the API:

- Use it to build a Java class plug-in that is loaded into the application.

- Use one of the included plug-ins to call out to an executable application to handle the validation.

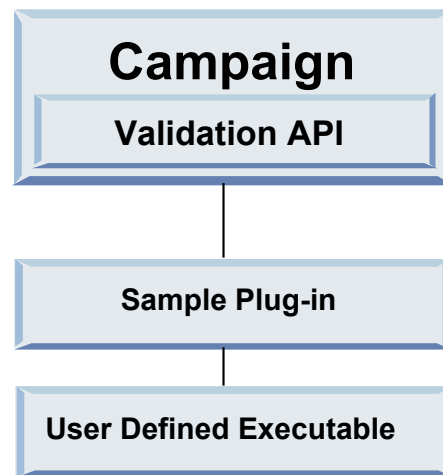
Build a Java class plug-in that is loaded into the application

The Validation PDK provides the interfaces, helper classes, and Developer's tools for developing these classes.



Call an application to handle validation

The second way to use the Validation PDK is to use one of the included plug-ins to call out to an executable application to handle the validation:



This executable may be written in any language, but must reside on the IBM Campaign server and executed on the server. The plug-in that calls the executable sends in an XML file that contains the information to be validated; for example, the user editing the object and the before/after values for all standard and extended attributes of that object). IBM Campaign expects results information in the form of an XML file in return. For more information, see "Developing a plug-in" on page 6.

Offer versus campaign validation

The Validation PDK can validate offers and campaigns. If a validation plug-in is defined, it is automatically called by IBM Campaign each time an offer or campaign object is saved. IBM Campaign sets a flag when it calls the plug-in's validate method. IBM Campaign passes the following flags:

- ValidationInputData.CAMPAIGN_VALIDATION, when adding or changing a campaign
or
- ValidationInputData.OFFER_VALIDATION, when adding or editing an offer.

You can then use these flags to construct validation rules applying to offers and campaigns.

Configuring the Validation PDK

The Validation PDK uses configuration parameters that tell Campaign how to find the plug-in class that should be used, and offers a way to pass some configuration information into those plug-ins.

All of the settings are on the Configuration page at:
Campaign > partitions > partition[n] > validation

Note: Validation works with multiple partitions; partition[n] can be changed to any partition name to provide validation routines for those partitions as well.

This section contains the following settings:

- “validationClass”
- “validationClasspath”
- “validatorConfigString” on page 5

validationClass

Property	Description
Description	This is the name of the class to use for validation. The value of the validationClasspath property indicates where this class resides.
Details	The class must be fully qualified with its package name. If this property is not set, Campaign does not perform any custom validation.
Example	com.unica.campaign.core.validation.samples.SimpleCampaignValidator This sets validationClass to the SimpleCampaignValidator class from the sample code.
Default	By default, no path is set: <property name="validationClass" />

validationClasspath

Property	Description
Description	This is the path to the class used for custom validation. It can either be a full path or a relative path.

Property	Description
Details	<p>Use either a full path or a relative path. If the path is relative, the behavior depends on the application server that is running Campaign. WebLogic uses the path to the domain work directory, which by default is</p> <p>c:\bea\user_projects\domains\mydomain</p> <p>If the path ends in a slash (forward slash / for UNIX or backslash \ for Windows), Campaign assumes that it points to the location of the Java plug-in class that should be used.</p> <p>If the path does not end in a slash, Campaign assumes that it is the name of a .jar file that contains the Java class, as shown in the example below.</p> <p>If the path is not set, Campaign does not attempt to load a plug-in.</p>
Example	<pre>/<CAMPAIGN_HOME>/devkits/validation/lib/validator.jar</pre> <p>This is the path on a UNIX platform that points to the JAR file that comes packaged with the plug-in developer's kit.</p>
Default	<p>By default, no path is set:</p> <pre><property name="validationClasspath" /></pre>
See also	<p>See "validationClass" on page 4 for information about designating the class to use.</p>

validatorConfigString

Property	Description
Description	<p>This is a string that is passed into the validator plug-in when it is loaded by Campaign.</p>
Details	<p>How the plug-in uses this string is up to the designer. You can use it to send a configuration string into your plug-in when the system loads it.</p> <p>For example, the ExecutableCampaignValidator (from the sample executable plug-in included with the PDK) uses this property to indicate the executable to run.</p>
Example	<p>To run the sample Bourne shell script as the validation script, set</p> <p>validatorConfigString to</p> <pre>/opt/unica/campaign/devkits/validation/ src/com/unica/campaign /core/validation/samples/validate.sh</pre>
Default	<p>By default, no path is set:</p> <pre><property name="validatorConfigString" /></pre>

Developing a plug-in

Overview

A plug-in is a Java class that is loaded at startup time and called whenever a campaign or offer is validated. This validation occurs whenever a user saves a campaign. You can create your own Java plug-ins using the tools provided by the Validation PDK. It contains source code for sample plug-ins and an Ant file (Apache Ant is a Java based build tool) you use to compile plug-ins. The following sections show you how to set up your environment to develop a plug-in and then walks you through the creation of your own plug-in.

Sample Validators

Two sample validators are included with the Campaign standard installation.

- `SimpleCampaignValidator` is a self-contained plug-in that shows how to do such things as custom authorization and validating allowable campaign names. It can be found in the following path:

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

We recommend that you make a copy of the class when working with it, rather than editing it directly.

- `ExecutableCampaignValidator` is a Java plug-in that calls out to an executable application to perform the validation. The source code for the `ExecutableCampaignValidator` is included in the same directory as the `SimpleCampaignValidator`.

However, the real purpose of this example is for use as a command-line executable for validation. This file is located in the following path:

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

This file is a sample loopback executable, illustrating common types of validation work.

Test Harness

Extreme programming and other agile methodologies are useful and popular. One important aspect of these methodologies is that they are test-centric. Customers who use those methodologies use unit testing extensively. The Validation PDK supports these methodologies by offering a test harness for running a plug-in outside of Campaign. Being able to test the code without putting it into IBM Campaign speeds up the plug-in Developer's process.

To use the test harness:

1. Alter the unit test case to reflect the validation logic in the plug-in.
2. Run the build script:
 - To create the plug-in without performing any unit tests, run the build scripts using the "ant jar" command.
 - To create the plug-in and also perform unit testing, run the build scripts using the "ant run-test" command.

Build Scripts

The build scripts in the PDK compile all of the classes in a directory and put them in a jar suitable for use in IBM Campaign. The directory used by the supplied build script is:

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

Major steps to create plug-ins

These are the major steps to create plug-ins:

- “Setup”
- “Build the validators”
- “Configure IBM Campaign” on page 8
- “Test the validator configuration” on page 8
- “Create a validator” on page 8

The following sections show you how to build the source to create the .jar file.

Setup

The Validation PDK can be installed on any machine, but the plug-ins you create with it must be placed on the machine running IBM Campaign. We recommend installing the PDK on the machine on which you are testing your plug-ins.

The PDK requires you to have Jakarta Ant and a Sun Java developer kit on your machine to create Java plug-ins. We recommend using the Ant and JDK packages that come with your application server to ensure compatibility.

To set up your environment to use the Validation PDK:

1. Add the folder containing the Ant executable to your path. Two examples are provided.
 - For WebLogic 10gR3 installed in the default directory on Windows, add the following to your path: C:\Oracle\Middleware\wlserver_10.3\common\bin
 - For WebSphere® 7.0 installed in the default directory on Windows, add the following to your path: C:\IBM\WebSphere\AppServer1\bin
2. Set the JAVA_HOME environment variable to the directory containing the bin and lib directories of the JDK. Two examples are provided.
 - For WebLogic 11gR1 on Windows, set JAVA_HOME to C:\Oracle\Middleware\jdk160_18
 - For WebSphere 7.0 on Windows, set JAVA_HOME to C:\IBM\WebSphere\AppServer1\java\jre

Build the validators

The PDK supplies an Ant script that can build all of the code in the sample files. The default behavior for the script is to create a jar that contains the validation classes. Optionally, it can also create Javadocs and run tests against the validators to ensure that they work in Campaign before trying to use the plug-in in production.

To build the validator:

1. Change directory to the PDK directory,
`<IBM_EMM_Home\Campaign_Home>\devkits\validation\build`
You will see the Ant script, build.xml, in this directory.

2. Run the Ant jar at the command line.

Ant runs the script and produces a JAR file called validator.jar in the directory:

```
<IBM_EMM_Home\Campaign_Home>\devkits\validation\build\lib
```

You now have a custom validator that can be used in IBM Campaign. The next section explains how to configure Campaign to use this validator.

Configure IBM Campaign

After you create a validator plug-in, you must tell IBM Campaign where it is so that you can use it.

To use the `SimpleCampaignValidator`, set the properties described in “Configuring the Validation PDK” on page 4 as follows:

- `validationClasspath`: `Unica\campaign\devkits\validation\lib\validator.jar`
- `validationClass`:
`com.unica.campaign.core.validation.samples.SimpleCampaignValidator`
- The `validatorConfigString` does not have to be set to use the `SimpleCampaignValidator` because it does not use a configuration string.

Test the validator configuration

After building the `validator.jar` file that contains the `SimpleCampaignValidator` class and making the necessary configuration changes, you are ready to test and use the plug-in. This plug-in prevents users from saving a Campaign named "badCampaign."

To test your configuration:

1. Redeploy your application server to have the changes take effect. For details on redeploying your application server, see your server documentation.
2. Log in to IBM Campaign and navigate to the campaign creation page.
3. Create a campaign with the name **badCampaign** and attempt to save it.

If everything is properly configured, you are not able to save the new campaign. You should receive an error message from the validator.

Create a validator

In this section, you create a validation plug-in that is much like the `SimpleCampaignValidator`, but prevents the creation of campaigns that are called "badCampaign2."

To create a validator:

1. Make a copy of the sample validator `SimpleCampaignValidator.java`, located in
`<IBM_EMM_Home\Campaign_Home>\devkits\validation\src\com\unica\campaign\core\validation\samples`
2. Name the copy `MyCampaignValidator.java` and leave it in the same directory as the source.
3. Open `MyCampaignValidator.java` in an editor. Find the word "badCampaign" in the document and replace it with the word "badCampaign2."
4. Save the file, and close the editor.
5. Build the validators again. For details, see “Build the validators” on page 7.

Note: If your application server locks the `validate.jar` file while in use, you need to stop the server before building the validators.

6. Reconfigure `campaign_config.xml` to use your new class:

```
<property name="validationClass"
value="com.unica.campaign.core.validation.samples.MyCampaignValidator">
```

7. Test the validator. For details, see “Test the validator configuration” on page 8.

You should not be able to save campaigns named "badCampaign2."

Executable Sample

The sample validators also include a validator, `ExecutableCampaignValidator`, that can run an executable from the command line.

This section:

- Shows how to set up IBM Campaign to run the sample executable plug-in, and
- Describes how to create your own executable plug-ins that conform to using the executable usage interface.

Configure Campaign for the sample executable plug-in

To use the `ExecutableCampaignValidator`, set the properties described in “Configuring the Validation PDK” on page 4 as follows:

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

The sample script that ships with the PDK is a Bourne shell script for UNIX. It denies campaign creation to anyone that has the user name "badUser." You can view the code for that executable in this directory:

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

You need to develop your own script that performs relevant validation for your implementation. Scripting languages like PERL and Python are good candidates for text processing scripts like this; however, any language that can be run from the command line is acceptable.

Expected executable usage interface

`ExecutableCampaignValidator` calls an executable with a command line that contains the following arguments:

- `executable_name`: this is the string set in the `validatorConfigString` in IBM Marketing Platform.
- `data_filename`: this is the name of the file that the executable reads as input. The input data must be formatted in XML.
- `expected_result_filename`: this is the name of the file that the executable should send as output. The expected results are of the form `data XXX.xml` where `XXX` is a number.

– Here is an example of how successful data is sent:

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

– Here is an example of how failed data is sent:

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

- Text in the XML file needs to be encoded in regular ASCII characters or UTF-8.

Note: It is highly recommended that you provide easy-to-comprehend error messages to users so they can correct the problem before reattempting another save operation.

Example

This section contains an example of a validation scenario.

Prevent campaign edits

If you are trying to prevent someone editing a campaign from changing the campaign code, you can use a custom campaign validation routine. The routine would ensure that when saving the campaign, the following check is done:

```
new_campaign_code == old_campaign_code
```

To handle the case when the campaign is first being created, pass to the routine flag indicating whether the campaign being validated is new (creation) or existing (edit). If this flag indicates **edit**, then do the comparison of campaign codes.

The Campaign application sets this flag in the `InputValidationData` object that it then passes to the plug-in. The plug-in reads the flag when it is determining whether the validation is for a new or changed campaign.

Contacting IBM technical support

If you encounter a problem that you cannot resolve by consulting the documentation, your company's designated support contact can log a call with IBM technical support. Use the information in this section to ensure that your problem is resolved efficiently and successfully.

If you are not a designated support contact at your company, contact your IBM administrator for information.

Information to gather

Before you contact IBM technical support, gather the following information:

- A brief description of the nature of your issue.
- Detailed error messages that you see when the issue occurs.
- Detailed steps to reproduce the issue.
- Related log files, session files, configuration files, and data files.
- Information about your product and system environment, which you can obtain as described in "System information."

System information

When you call IBM technical support, you might be asked to provide information about your environment.

If your problem does not prevent you from logging in, much of this information is available on the About page, which provides information about your installed IBM applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, you can obtain the version number of any IBM application by viewing the `version.txt` file located under the installation directory for each application.

Contact information for IBM technical support

For ways to contact IBM technical support, see the IBM Product Technical Support website: (http://www-947.ibm.com/support/entry/portal/open_service_request).

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
170 Tracer Lane
Waltham, MA 02451
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Privacy Policy and Terms of Use Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. A cookie is a piece of data that a web site can send to your browser, which may then be stored on your computer as a tag that identifies your computer. In many cases, no personal information is collected by these cookies. If a Software Offering you are using enables you to collect personal information through cookies and similar technologies, we inform you about the specifics below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name, and other personal information for purposes of session management, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

Various jurisdictions regulate the collection of personal information through cookies and similar technologies. If the configurations deployed for this Software Offering provide you as customer the ability to collect personal information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for providing notice and consent where appropriate.

IBM requires that Clients (1) provide a clear and conspicuous link to Customer's website terms of use (e.g. privacy policy) which includes a link to IBM's and Client's data collection and use practices, (2) notify that cookies and clear gifs/web beacons are being placed on the visitor's computer by IBM on the Client's behalf along with an explanation of the purpose of such technology, and (3) to the extent required by law, obtain consent from website visitors prior to the placement of cookies and clear gifs/web beacons placed by Client or IBM on Client's behalf on website visitor's devices

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Online Privacy Statement at: <http://www.ibm.com/privacy/details/us/en> section entitled "Cookies, Web Beacons and Other Technologies."



Printed in USA