Version 11 Release 0.1
August 31, 2018

IBM Campaign Validation PDK Guide

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 17.

This edition applies to version 11, release 0, modification 1 of IBM Campaign and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1998, 2018.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

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

Contents of the Validation PDK 2
Two ways to use the validation API 2
Build a Java class plug-in that is loaded into the application 3
Call an application to handle validation 3
Offer versus campaign validation 3
Sample validators included in the Validation PDK . . 4
Test harness for the Validation PDK 4
Build scripts for the Validation PDK 5

Chapter 2. Developing validation plug-ins for Campaign 7

Setting up your environment to use the Validation PDK 7
Building the validator 7
Configuring Campaign to use a validation plug-in. . 8
validationClass 9
validationClasspath 9
validatorConfigString 9
Testing the validator configuration. 10
Creating a validator 10
Example validation scenario: prevent campaign edits 11

Chapter 3. Calling an application to handle validation 13

Configuring Campaign to use the sample executable plug-in 13
Expected executable usage interface 13

Before you contact IBM technical support 15

Notices 17

Trademarks 19
Privacy Policy and Terms of Use Considerations . . 19

© Copyright IBM Corp. 1998, 2018

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

Use the IBM® Validation Plug-in Developer's Kit (PDK) to develop custom validation logic for use in IBM Campaign.

You can create plug-ins to perform custom validation logic for campaigns, offers, or both.

Some possible uses of validation logic are:

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

The Validation PDK is a subclass of a more generic plug-in framework that is provided with IBM Campaign.

The Validation PDK contains Javadoc reference information for both the Plug-In API and the sample code. To view the documentation, open the following file in your web browser:

```
C:\IBM\IMS\Campaign_Home\devkits\validation\javadoc\index.html
```

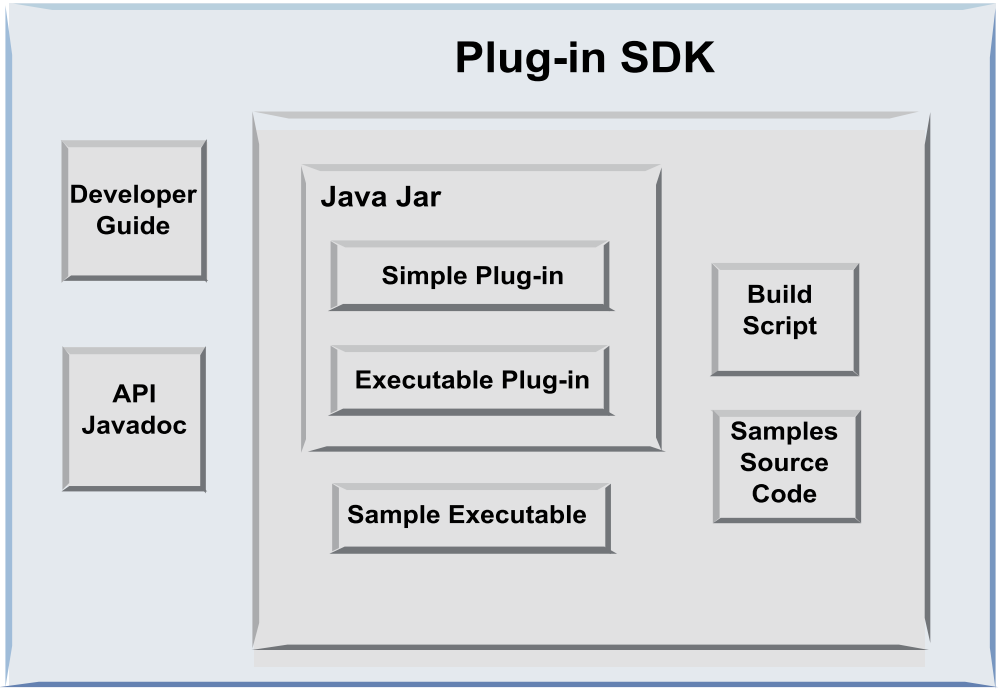
For example:

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

© Copyright IBM Corp. 1998, 2018

Contents of the Validation PDK

The Validation PDK contains components to develop Java™ plug-ins or command-line executables to add custom validation to IBM Campaign. The PDK contains documented, buildable examples of how to use the PDK.



The following table describes each component.

Table 1. Components of the Validation PDK

Component	Description
Developer guide	A PDF document titled *IBM Campaign Validation PDK Guide*.
API Javadoc	Reference information for the plug-in API.
Java .jar file	A sample JAR file that contains the sample plug-ins. The JAR file contains: • 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.
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.

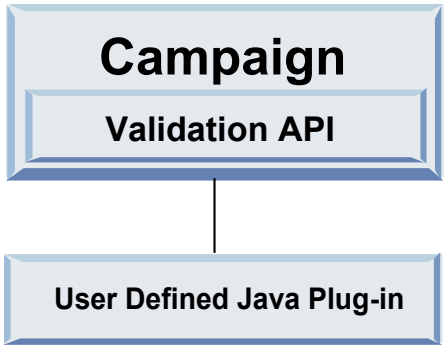
Two ways to use the validation API

There are two ways to use the Validation 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.

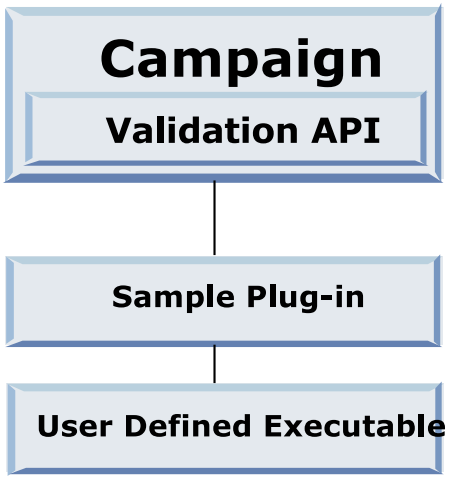
Build a Java class plug-in that is loaded into the application

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



Call an application to handle validation

You can use one of the included Validation PDK plug-ins to call out to an executable application to handle the validation.



The executable may be written in any language, but must reside on the IBM Campaign server and be 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 and 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.

Offer versus campaign validation

A plug-in that is made with the Campaign Validation PDK can perform custom validation logic for campaigns, offers, or both.

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.

Sample validators included in the Validation PDK

Two sample validators are included in the Campaign Validation PDK: `SimpleCampaignValidator` and `ExecutableCampaignValidator`.

- `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 before you edit it, so you can retain the original version if needed.
- `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:

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

  However, the real purpose of this example is for use as a command-line executable for validation. This file is 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 for the Validation PDK

Being able to test validation code without putting it into IBM Campaign speeds up the plug-in Developer's process.

Customers who use extreme programming and other agile methodologies use unit testing extensively. The Validation PDK supports these methodologies by offering a test harness for running a plug-in outside of Campaign.

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 doing any unit tests, run the build scripts using the "`ant jar`" command.
   - To create the plug-in and also do unit testing, run the build scripts using the "`ant run-test`" command.

Build scripts for the Validation PDK

The build scripts in the Validation PDK compile all of the classes in a directory and put them in a JAR file that is suitable for use in IBM Campaign.

The supplied build script uses the following directory:

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

Chapter 2. Developing validation plug-ins for Campaign

A plug-in is a Java class that is loaded at startup time and called whenever a campaign or offer is validated.

The validation occurs whenever a user saves a campaign. You can create your own Java plug-ins with the tools that are provided in the Validation PDK. The PDK contains source code for sample plug-ins and an Ant file (Apache Ant is a Java based build tool) that you use to compile plug-ins.

The following steps explain how to set up your environment to develop a plug-in and then walk you through the creation of your own plug-in.

1. "Setting up your environment to use the Validation PDK"
2. "Building the validator"
3. "Configuring Campaign to use a validation plug-in" on page 8
4. "Testing the validator configuration" on page 10
5. "Creating a validator" on page 10

Setting up your environment to use the Validation PDK

To use the Validation PDK with Campaign, you must modify your path and set the `JAVA_HOME` environment variable.

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

The PDK requires you to have Apache Ant and a Sun Java developer kit on your machine to create Java plug-ins. To ensure compatibility, use the Ant and JDK packages that come with your application server.

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 11gR1 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`

Building the validator

The IBM Campaign Validation PDK supplies an Ant script that can build all of the code in the sample files.

© Copyright IBM Corp. 1998, 2018

The default behavior for the script is to create a jar that contains the validation classes. Optionally, it can also create Javadoc 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 to the PDK directory `<IBM_IMS_Home\Campaign_Home>\devkits\validation\build`

   For example: `C:\IBM\IMS\Campaign\devkits\validation\build`

   This directory contains the Ant script, `build.xml`.
2. Run the Ant jar at the command line.
   - To create the plug-in without doing any unit tests, use the `ant jar` command.
   - To create the plug-in and also do unit testing, use the `ant run-test` command.

   Ant runs the script and produces a JAR file called `validator.jar` in the `lib` subdirectory. For example:

   `C:\IBM\IMS\Campaign\devkits\validation\build\lib`

You now have a custom validator that can be used in IBM Campaign. Your next step is to configure Campaign to use this validator.

Configuring Campaign to use a validation plug-in

To configure Campaign to use a validation plug-in, use the configuration settings at `Campaign > partitions > partition[n] > validation`.

The configuration properties tell Campaign how to find the plug-in class and they provide a way to pass configuration information to the plug-ins.

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

You can adjust the following validation configuration settings:

- "validationClass" on page 9
- "validationClasspath" on page 9
- "validatorConfigString" on page 9

To use the `SimpleCampaignValidator`, set the properties 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.

To use the `ExecutableCampaignValidator`, set the properties as follows:

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

validationClass

The `validationClass` tells Campaign the name of the class to use for validation with a Validation PDK plug-in.

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

validationClasspath

The `validationClasspath` tells Campaign the location of the class to use for validation with a Validation PDK plug-in.

Property	Description
Description	The path to the class that is used for custom validation.
Details	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 `c:\bea\user_projects\domains\mydomain`. If the path ends in a slash (/ for UNIX or \ for Windows), Campaign assumes that it points to the location of the Java plug-in class to be used. 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 following example. If the path is not set, Campaign does not attempt to load a plug-in.
Example	`/<CAMPAIGN_HOME>/devkits/validation/lib/validator.jar` This is the path on a UNIX platform that points to the JAR file that is packaged with the plug-in developer's kit.
Default	By default, no path is set: `<property name="validationClasspath" />`
See also	See “validationClass” for information about designating the class to use.

validatorConfigString

The `validatorConfigString` is passed into the validator plug-in when it is loaded by Campaign.

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

Testing the validator configuration

After building the `validator.jar` file that contains the `SimpleCampaignValidator` class and making the necessary configuration changes, you can test and use the plug-in.

The following plug-in example prevents Campaign users from saving a campaign that is named "badCampaign."

To test your configuration:

1. Redeploy your application server so the changes take effect. For instructions, see your server documentation.
2. Log in to IBM Campaign and go to the campaign creation page.
3. Create a campaign with the name **badCampaign** and try to save it.

If everything is properly configured, you are not able to save the new campaign. If you receive an error message from the validator, you know that it is working correctly.

Creating a validator

Follow these instructions to create a validation plug-in that is much like the `SimpleCampaignValidator`, but prevents the creation of campaigns that are called "badCampaign2."

1. Make a copy of the sample validator `SimpleCampaignValidator.java` in `<IBM_IMS_Home\Campaign_Home>\devkits\validation\src\com \unica\campaign\core\validation\samples`. Name the copy `MyCampaignValidator.java` and leave it in the same directory as the source. For example:

   `C:\IBM\IMS\Campaign\devkits\validation\src\com \unica\campaign\core\ validation\samples\MyCampaignValidator.java`
2. Open `MyCampaignValidator.java` in an editor. Find the word "badCampaign" in the document and replace it with the word "badCampaign2."
3. Save the file and close the editor.

4. Build the validators again. For details, see “Building the validator” on page 7. If your application server locks the `validate.jar` file while in use, stop the server before you build the validators.
5. Reconfigure `campaign_config.xml` to use your new class: `<property name="validationClass" value="com.unica.campaign.core.validation.samples.MyCampaignValidator">`
6. Test the validator. For details, see “Testing the validator configuration” on page 10.

Confirm that the validator works: You should not be able to save campaigns named "badCampaign2."

Example validation scenario: prevent campaign edits

This example explains how to use validation to prevent specific edits to a campaign.

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

```
new_campaign_code == old_campaign_code
```

To handle the case when the campaign is first being created, pass to the routine a flag that indicates whether the campaign being validated is new (creation) or existing (edit). If this flag indicates **edit**, then compare the 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 determines whether the validation is for a new or changed campaign.

Chapter 3. Calling an application to handle validation

The Validation PDK includes a sample validator, `ExecutableCampaignValidator`, which runs an executable, `validate.sh`, from the command line, to perform validation.

The following sections explain how to:

- Configure Campaign to run the sample executable plug-in, and
- Create your own executable plug-in that conforms to using the executable usage interface.

Configuring Campaign to use the sample executable plug-in

To use the `ExecutableCampaignValidator`, adjust the configuration settings at `Campaign > partitions > partition[n] > validation`.

Set the propertiesas 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 Validation PDK is a Bourne shell script for UNIX. It denies campaign creation to anyone who has the user name "badUser." You can view the code for that executable in the following 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 such as 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

The `ExecutableCampaignValidator` plug-in calls an executable file with a command line that contains the following arguments.

- `executable_name`: The string set in the `validatorConfigString` in IBM Marketing Platform.
- `data_filename`: The name of the file that the executable reads as input. The input data must be formatted in XML.
- `expected_result_filename`: 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:

© Copyright IBM Corp. 1998, 2018

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

– Text in the XML file must be encoded in regular ASCII characters or UTF-8.

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

Before you contact 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 these guidelines 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.

Note: Technical Support does not write or create API scripts. For assistance in implementing our API offerings, contact IBM Professional Services.

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, check for a `version.txt` file that is located under the installation directory for your application.

Contact information for IBM technical support

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

Note: To enter a support request, you must log in with an IBM account. This account must be linked to your IBM customer number. To learn more about associating your account with your IBM customer number, see **Support Resources > Entitled Software Support** on the Support Portal.

© Copyright IBM Corp. 1998, 2018

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 on 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 1998, 2018

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
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
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 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