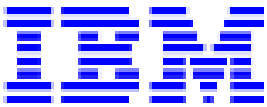
IBM FileNet Content Manager

FileNet Salesforce Connector Programming Guide



Version 5.8.2

Copyright

Before you use this information and the product it supports, read the information in "Notices" on page 18.

Overview

The IBM Salesforce Connector for FileNet v5.8.2 introduces the ability to programmatically control the behavior of the Salesforce Connector components. The following operations are supported:

- A new Lighting Web Component which allows adding a Document to a Salesforce record without showing the list of existing documents, or allowing the user to view or modify any of the document properties.
- New public API properties on the IBM Document List component which allow programmatically setting the Document class and its custom properties, when a new document is added.
- Ability to programmatically hide the Document Class and some or all of the custom properties on the Add Document dialog
- A new public API property on the IBM Document List component which allows programmatically hiding Document Actions on the Document Actions context menu.
- A new ability to specify custom pseudo record-Id, record URL, and record type values programmatically, when creating a new document, or when querying for existing documents.

To use these programmatic control capabilities, customers must create a custom Salesforce Lightning component which will programmatically call methods on the Lightning Components of the Salesforce Connector classes. This customer defined Salesforce Lightning component is referred to as a “wrapper component”.

Customers may further automate their Saleforce applications through the use of a scripting solution such as OmniScript, by having OmniScript call methods on the wrapper component.

This guide provides step-by-step instructions for developing custom Lightning Web Components using the new functionality in IBM Salesforce Connector v5.8.2.

After reading the guide and understanding how these Lightning Web Components are composed and deployed, you should be able to apply the same patterns when developing your own custom Lightning Web Components, which will programmatically control the behavior of the Salesforce Connector for FileNet.

Prerequisites

In order to use this guide, you will need the following pre-requisites:

- A Salesforce organization with the Salesforce Connector v5.8.2 is installed and configured.
- Administrator permissions, allowing you to deploy a new application to the Salesforce Org.
- Familiarity with developing Salesforce Lighting Web Components.

Prepare the Local Development Environment

To create and deploy a new Lightning Web Component, you need to set up a local Salesforce development environment. Follow the steps below:

1. Install the Salesforce Command Line Interface (CLI)
   https://developer.salesforce.com/tools/salesforcecli
2. Download and install the latest version of Visual Studio Code for your operating system.
   https://code.visualstudio.com/download
3. Launch Visual Studio Code and install the Salesforce Extension Pack.

Clone the Salesforce Connector Recipes Repository

To demonstrate the programmatic use of the IBM Salesforce Connector for FileNet components, a set of Salesforce Lightning Web components which act as “wrapper components” for the Connector components, has been developed. Each of these components constitutes a “recipe” which you can adapt to meet your business needs. The code for these components is available in a public GitHub repository as described below.

This GitHub repository contains the Salesforce Connector Recipes Application. The recipes in this application cover all the basics of composing IBM Salesforce Connector Components. Each recipe demonstrates how to compose a wrapper component for a IBM Salesforce Connector Lightning Web Component, specify a Document class, and set custom properties of a document.

To get started, open a terminal window, and clone the following repo.

git clone https://github.com/ibm-ecm/ibm-ecm-connector-salesforce-lwc-recipes

To open the project into Visual Studio Code:

cd ibm-ecm-connector-salesforce-lwc-recipes

code .

The project will be opened in Visual Studio Code. You can review the Lightning Web Component, JavaScript Utility Class, Apex Class, and other artifacts in this project.

Prepare the Object Store

The Salesforce Connector Recipes application uses specific Document classes and their associated properties. So, you need to create the following Document classes, Properties, and Choice Lists in your FileNet Object Store.

1. Document subclass: MyAccount

Property Name	Data Type
Description	String
AccountNumber	String
AnnualRevenue	Float
Rating	String
IsNew	Boolean
DueDate	DateTime

   Create a Choice List with the following choice items and associate with Rating property.
   - Hot (Hot)
   - Warm (Warm)
   - Cold (Cold)
   - None ()

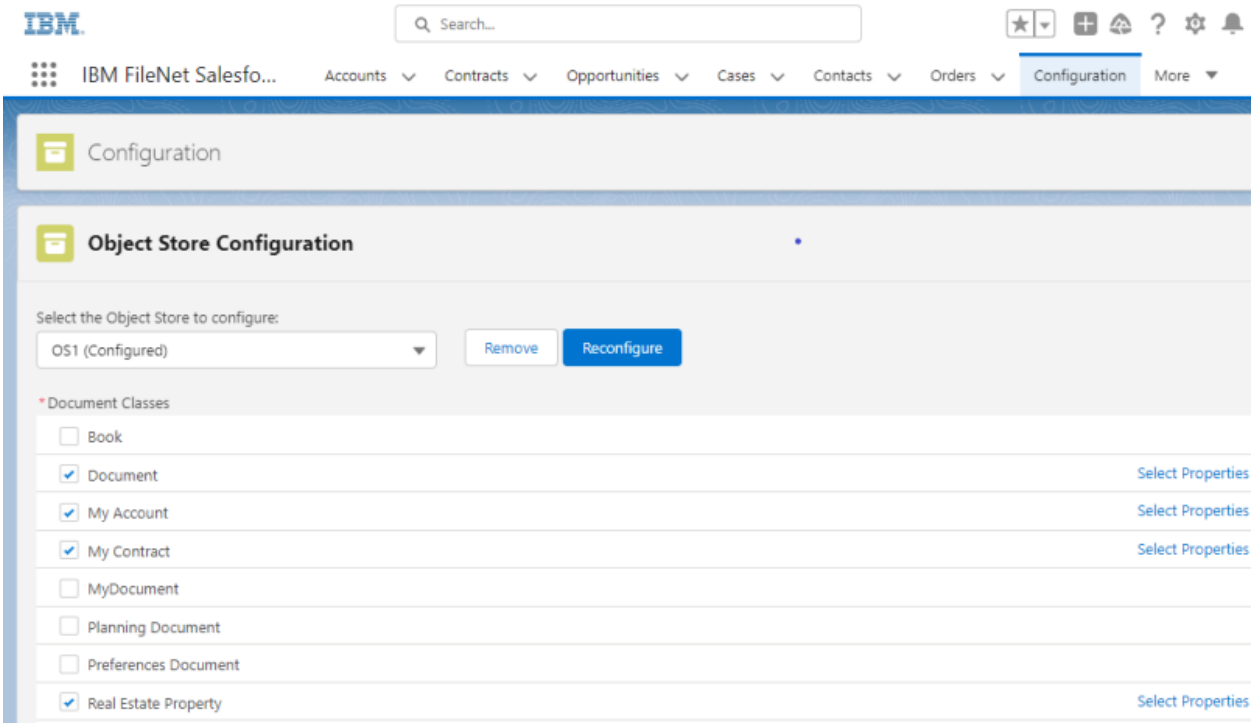
2. Document subclass: MyContract

Property Name	Data Type
Description	String
ContractNumber	String
ContractTerm	Integer

3. Document subclass: RealEstateProperty

Property Name	Data Type
Address	String
Type	String
Bathrooms	Float
Bedrooms	Integer
Garage	Integer
Price	Float
Sqft	Integer

After the Document classes are created, open the IBM Salesforce Connector Application from the App Launcher, click the Configuration tab, choose an Object Store, and select these Document classes and their associated properties as shown below.

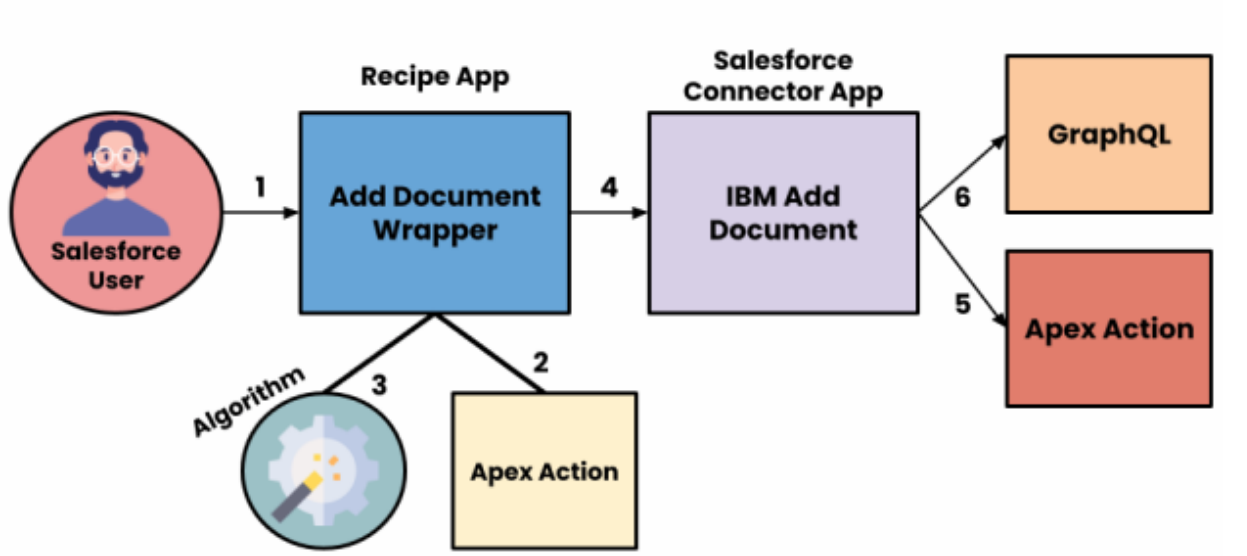


Understand the Lightning Web Components

In this section we will review each lighting web component from the Salesforce Connector Recipes application.

IBM Add Document Wrapper (addDocumentWrapper)

The following graphic describes the interaction between the Add Document Wrapper component from the Recipes Application and the Add Document component from the Salesforce Connector.



1. A Salesforce User accesses the Add Document Wrapper component on the Lightning Page.
2. The wrapper component accesses the Apex class to get the Salesforce Record context.
3. Based on the Salesforce Record type, the wrapper component creates an instance of the Document Class and sets its properties.
4. Once the user selects the files to upload, the Add Document Wrapper component passes the Document instance object to the Add Document component.
5. The Add Document component accesses the Apex class to get the Salesforce Record details.
6. The Add Document component then calls the IBM Content Services GraphQL API endpoint to upload the document to the FileNet Object Store.

The intent of the wrapper component is to display just a File Upload UI element, and hide the Document Class and its properties. The Document class and its properties are set programmatically as stated above.

The component provides an integrated way to upload multiple files. The File Upload UI element includes drag-and-drop files functionality.

The HTML template of the component wraps the “ibm_fn_cm-add-document” component from the Salesforce Connector Application’s namespace and sets its properties as shown in the above graphic.

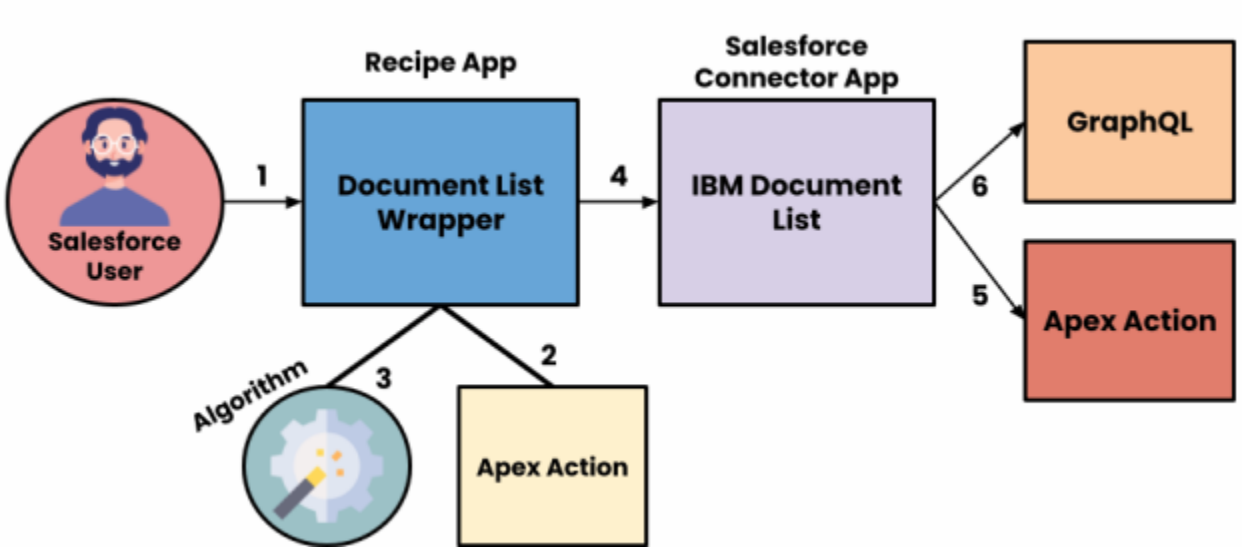
In the JavaScript file, the component initialization task is performed in the connectedCallback hook. The initialization task calls for an Apex method to retrieve the Salesforce Record name and set the internal state of the component.

Based on the objectApiName field, the docClass property's getter method constructs a Document Class object and its properties using the utility class called DocumentClass. Please refer to the documentation comments in the DocumentClass for further details.

The component markup declares an event listener called "onfileuploadcomplete," and the "handleFileUploadComplete" function gets called when the file upload is completed.

IBM Document List Wrapper (documentListWrapper)

The following graphic describes the interaction between the Document List Wrapper component from the Recipes Application and the Document List component from the Salesforce Connector.



1. A Salesforce User accesses the Document List Wrapper component on the Lightning Page.
2. The wrapper component accesses the Apex class to get the Salesforce Record context.
3. Based on the Salesforce Record type, the wrappercomponent creates an instance of the Document Class and sets its properties.
4. Once the user selects the files to upload, the Document List Wrapper component passes the Document instance object to the Document List component.
5. The Document List component accesses the Apex class to get the Salesforce Record details.
6. The Document List component then calls the IBM Content Services GraphQL API endpoint to upload the document to the Object Store.

The intent of this wrapper component is to hide the Document Class and its properties in the Add Document Dialog for the Account and Contract record types, and instead specify these properties programmatically. For all other record types (e.g., Opportunity), the wrapper component will show the base Document Class selected and show the Name property.

The HTML template of the component wraps the “ibm_fn_cm-documents-list” component from the Salesforce Connector Application’s namespace and sets its properties.

In the JavaScript file, the component initialization task is performed in the connectedCallback hook. The initialization task calls an Apex method to retrieve the Salesforce Record name and set the internal state of the component.

Based on the objectApiName field, the docClass property getter method constructs a Document Class object and its properties using the utility class called Document Class. Note the code in the documentListWrapper.js that sets the hideDocumentClass and hideNameProperty attributes to false when the record type is not Account or Contract.

The component markup also declares an event listener called “onfileuploadcomplete,” and the “handleFileUploadComplete” function gets called when the file upload is completed.

IBM Document List Wrapper with External Record (documentListWrapperExternalRecord)

This recipe component shows how to compose a Salesforce Document List component on a lightning page using data from external API.

The Salesforce Digital Experience site and CRM Experience site allows developers to deploy a custom Lightning Web component that uses data from an external API. For instance, a developer can utilize the component to show a real estate listing from an external data source on a Salesforce Community site. When a user opens the real estate property record from the listing view, a detailed page shows further information about the property.

The Salesforce Connector Document List could be placed on the external record page using a custom Lightning Web Component. This component wraps the Salesforce Connector Document List and allows for association between the FileNet Document and the external record.

The Salesforce Connector uses the Id class from the Apex system namespace to retrieve details of the Salesforce record id. For the external record scenario, the Id class won’t be able to retrieve details of the external record. Therefore, there is a need to differentiate the Salesforce record id from the external record id. For the external record, the custom Lightning Web component specifies a pseudo record id containing a pseudo-Id and pseudo record type.

The pseudo record id is composed of three parts separated by a colon. The first part is a single letter “C”, the second part is the name of the record, and the third part is the unique identifier of the record. For

example, the pseudo record id of the Real Estate Property record could be **C:RealEstateProperty:278388188**, where RealEstateProperty is a record type and 278388188 is the unique identifier. The custom Lightning Web component creates the pseudo record id from @objectApiName and @recordId API properties. Both @objectApiName and @recordId are required properties for this component.

In the JavaScript file, the component initialization task is performed in the connectedCallback hook. Based on the objectApiName and the recordId fields, the component generates a pseudo record id before passing it on to Salesforce Connector Document List. Even though RealEstateProperty Object in this component is a Salesforce Custom Object, the implementation details for creating pseudo record id and setting FileNet Document class properties will be same for the external data source.

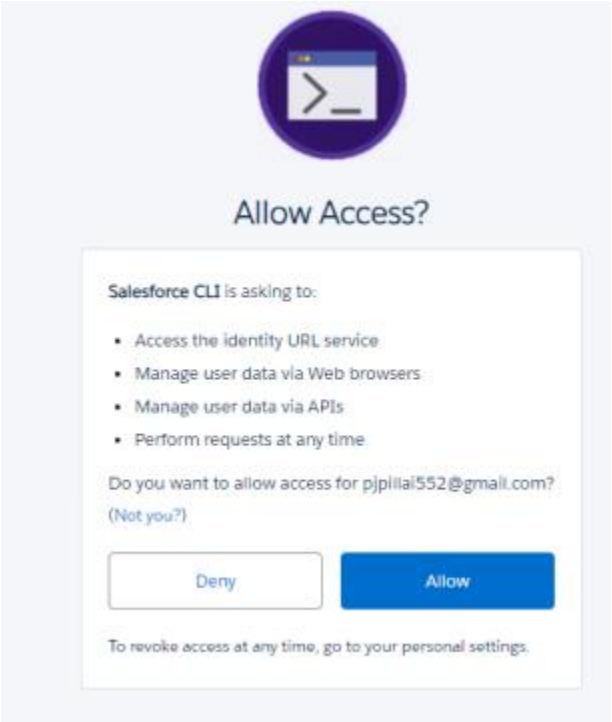
Deploy the Application

1. The first step is to authorize a Salesforce Org to allow use of the Salesforce CLI. Switch to the terminal window and make sure that you are in the Salesforce Connector Recipes Git repository home directory. Enter the following command and press enter:

   # sfdx org login web

   The default browser on your system should open the Salesforce login page. (If you are using the Salesforce Sandbox, refer to SFDX documentation for instructions on logging into the Sandbox).

   After you have logged in, for the first time you will get the following page asking you to allow access to the Salesforce CLI.



   Click the Allow button, and you will be logged into the Salesforce Org. On the terminal window, you will get the following message:

   Successfully authorized <your email address> with org ID <your org id>
   Next you must close this browser window to clear the browser cache.

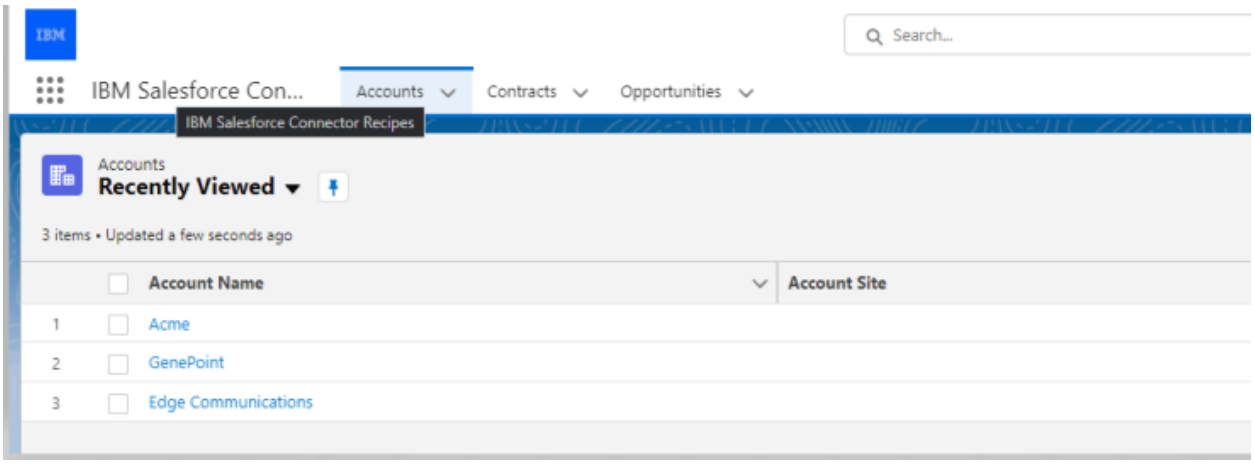
2. Now reopen the Salesforce Org using the following command listed below. You will be automatically logged into Salesforce Org.

   # sf org open

3. To deploy the App to the Salesforce Org, enter the following command in the terminal window:
   # sf project deploy start

4. Next, update the permission set to allow Salesforce Organization users to access the IBM Salesforce Connector Recipes Application.
   1. Click IBM Salesforce Connector Recipes User permission set.
   2. On the Permission set screen, click Manage Assignments.
   3. Select the user or users that you want to add to the permission set (or unselect users who you would like to remove from the permission set) and click Assign.
   4. Click Save to save your changes.

5. Click on the App Launcher. Click View All and you should see the "IBM Salesforce Connector Recipes" Application. Click this application and it will open in a new tab as shown below:



Add the IBM Add Document Wrapper to a Salesforce Record Page

This wrapper component can be added to any Salesforce Lightning page. This includes pages for custom Salesforce record types and Lighting pages within Salesforce Digital Experience sites.

Use the following steps to add the components to the Account page:

1. Go to the App Launcher, and start the IBM FileNet Salesforce Connector Recipes app.
2. Click the Accounts tab.
3. Select an individual account, and go to the Related tab. The IBM Documents widget is not visible.
4. From the Setup menu, select Edit page.
5. From the Lightning Components bar, under Custom, select the IBM Add Document Wrapper component and drag and drop it to your desired location on the Related Items tab.
6. Enter the Object Store Symbolic name in the Object Store field, and change the File Upload Label if needed.
7. Click Save to save the edited layout.
8. To activate these changes, click Activation. Click Assign as Org Default, then click Save.
9. Click Back to return to the main screen.

Repeat the steps for the Salesforce Contract and Opportunity pages.

Add the IBM Document List Wrapper to the Salesforce Record Page

This wrapper component can be added to any Salesforce Lightning page. This includes pages for custom Salesforce record types and Lighting pages within Salesforce Digital Experience sites.

Use the following steps to add the components to the Account page:

1. Go to the App Launcher, and start the IBM FileNet Salesforce Connector Recipes app.
2. Click the Accounts tab.
3. Select an individual account, and go to the Related tab. The IBM Documents widget is not visible.
4. From the Setup menu, select Edit page.
5. From the Lightning Components bar, under Custom, select the IBM Add Document Wrapper component and drag and drop it to your desired location on the Related Items tab.
6. Enter the Object Store Symbolic name in the Object Store field, and change the Title Label if needed.
7. Optionally change the “Visible Rows” setting, to control the number of document rows that are shown in the component.
8. In the Hide Document Action field, enter the list of actions (separated by a comma) if you wish to hide the action from the Document Action context menu (e.g., UploadNewVersion,AddTo,Delete).
   The valid action names are as follows:
   a. Properties
   b. View
   c. Download
   d. UploadNewVersion
   e. RemoveFromRecord
   f. AddToRecord
9. Click Save to save the edited layout.
10. To activate these changes, click Activation. Click Assign as Org Default, then click Save.
11. Click Back to return to the main screen.

Repeat the steps for the Salesforce Contract and Opportunity pages.

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 J74/G4 555 Bailey Avenue

San Jose, CA 95141 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.

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 J46A/G4 555 Bailey Avenue San Jose, CA 95141-1003
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.

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.

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.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. Other product and service names might be trademarks of IBM or other companies.